SCIENT INSTITUTE OF TECHNOLOGY

Ibrahimpatnam. R.R Dist - 501506

(Approved by AICTE & Affiliated to JNTUH, Hyderabad)

COURSE FILE

NAME OF THE FACULTY	: Ms.BANALA RAMYA SREE
DEPARTMENT	: COMPUTER SCIENCE & ENGINEERING
YEAR & SEM	: II YEAR II SEM
COURSE	: COMPUTER ORGANIZATION & ARCHITECTURE
COURSE CODE	: CS304PC
REGULATION	: R18
BRANCH & SECTION	: COMPUTER SCIENCE & ENGINEERING(A&B)
ACADEMIC YEAR	: 2021-2022
NO. OF CREDITS	: 3

SCIENT INSTITUTE OF TECHNOLOGY

Ibrahimpatnam. R.R Dist - 501506

(Approved by AICTE & Affiliated to JNTUH, Hyderabad)

INDEX

SNO	TOPIC	PAGE NO's
1	Academic Calendar	1-5
2	Vision & Mission of College	6
3	Vision & Mission of Department	7
4	Program Educational Objectives(PEOs)	8
5	Program Outcomes(POs)	9
6	Program Specific Outcomes(PSOs)	10
7	Course Objectives	11
8	Syllabus	12
9	Course Outcomes with Blooms taxonomy levels	13
10	CO Mapping with POs & PSOs	14
11	Class Time table	15-16
12	Individual Time Table	17-20
13	Lesson Plan	21-24
14	Short Answer Questions(Unit wise)	25-29
15	Long Answer Questions(Unit wise)	30-40
16	Objective Questions(Unit wise)	41-45
17	Lecture Notes(Unit wise)	46-150
18	JNTUH Previous question papers	151-156
19	MID QP's with BT Levels	151-156
20	MID Marks	157-158
21	MID Answer Scripts	159-170
22	Result Analysis	171-172
23	Others(if any)	

SCIENT INSTITUTE OF TECHNOLOGY

Ibrahimpatnam, R.R Dist 501506

(NAAC Accredited, Approved by AICTE & Affiliated to JNTUH)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT ACADEMIC CALENDAR : 2021-2022

B.TECH II YEAR I SEMESTER

Date: 18th Oct, 2021

PART	EVENT	DATE OF COMMENCEMENT	DATE OF COVERAGE
1st	Dussehra Recess	11th Oct,2021	16th Oct,2021
	COMMENCEMENT OF CLASS WORK	18th Oct, 2021	
	Instructions for covering Unit 1st to 2.5th	18th Oct, 2021	11th Dec, 2021
	Instructions for covering unit 1st	18th Oct, 2021	03rd Nov, 2021
	Department meeting & Review meeting of syllabus Coverage of Unit 1st	05th Nov, 2021 at 3:30 PM at HOD Office	
	Unit 1st Revision	06th Nov, 2021	
	Class Internal test-Unit 1st	08th Nov, 2021	
	Instructions for covering unit 2nd	09th Nov, 2021	27th Nov, 2021
	Department meeting & Review meeting of syllabus Coverage of Unit 2nd	29th Nov, 2021 at 3:30 PM at HOD Office	
	2nd Unit Revision	29th Nov, 2021	
	Class Internal test-Unit 2nd	30th Nov, 2021	
	Instructions for covering unit 2.5	1st Dec, 2021	09th Dec, 2021
	Unit 2.5th Revision	10th Dec, 2021	
	Class Internal test-Unit 2.5th	11th Dec, 2021	
	Department meeting & Review meeting of syllabus Coverage of Unit 2.5th	11th Dec, 2021 at 3:30 PM at HOD Office	
	1st Mid Examinations (Theory)	13th Dec, 2021	18th Dec, 2021
	Mid Examinations (Practical)	16th Dec, 2021	18th Dec, 2021
2nd	Instructions for covering Unit 2.5th to 5th	20th Dec, 2021	12th Jan, 2022
	Instruction for covering Unit 2.5th – 3rd	20th Dec, 2021	29th Dec, 2021
	Department meeting & Review meeting of syllabus Coverage of Unit 3rd	29th Dec, 2021 at 2:30 PM at HOD Office	
	Unit 3rd Revision	30th Dec, 2021	
	Class Internal test: Unit 3rd	31st Dec, 2021	
	Instructions for covering unit 4th	3rd Jan, 2022	22nd Jan, 2022
	Department meeting & Review meeting of syllabus Coverage of Unit 4th	24th Jan, 2022 at 3:30 PM at HOD Office	
	4th Unit Revision	24th Jan, 2022	
	Class Internal test: Unit 4th	25th Jan, 2022	
	Instructions for covering unit 5th	27th Jan, 2021	10th Feb, 2022
	Department meeting & Review meeting of syllabus Coverage of Unit 5th	11th Feb, 2022 at 3:30 PM at HOD Office	
	Unit 5th Revision	11th Feb, 2022	
	Class Internal test: Unit 5th	12th Feb, 2022	
	2nd Mid Examinations (Theory)	14th Feb, 2022	19th Feb, 2022
	Preparation Holidays & Practical Examination	19th Jan, 2022	25th Jan, 2022
	End Semester Regular & Supplementary Examination	28th Feb, 2022	12th March, 2022

HOD

(Mr.SMD SHAFIULLA)

VISION AND MISSION OF COLLEGE

VISION

To impart quality education in building Engineering and Management professionals striving for a symbiosis of innovative technological excellence, research and human values with global standards to meet skills, knowledge and behaviour of industry and societal needs.

MISSION

- To achieve excellence by imparting innovative Teaching & Learning and Research.
- To generate, empower, disseminate, and preserve knowledge and information.
- To render social relevant technical services and inculcating entrepreneurial talents in technological advancements.
- To nurture, inculcate and develop skills, knowledge and attitudes to render technical services for industry and societal needs.

VISION AND MISSION OF DEPARTMENT

VISION

The Vision of the department is to produce competent graduates suitable for industries and organizations at global level including research and development with Social responsibility.

MISSION

- To provide an open environment to foster professional and personal growth with a strong theoretical and practical background.
- Emphasis on hardware and software development making the graduates industry ready with social ethics.
- Further the Department is to provide training and to partner with Global entities in education and research

Program Educational Objectives (PEOs):

PEO1	Students will establish themselves as effective professionals by solving real problems through the use of computer science knowledge and with attention to team work, effective communication, critical thinking and problem solving skills.
PEO2	Students will develop professional skills that prepare them for immediate employment and for life-long learning in advanced areas of computer science and related fields.
PEO3	Students will demonstrate their ability to adapt to a rapidly changing environment by having learned and applied new skills and new technologies
PEO4	Students will be provided with an educational foundation that prepares them for excellence, leadership roles along diverse career paths with encouragement to professional ethics and active participation needed for a successful career.

Program Outcomes (POs):

PO1-Engineering Knowledge: Apply the knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the solution of complex engineering problems.

PO2-Problem Analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences and Engineering sciences.

PO3-Design/Development of Solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4- Conduct Investigations of Complex Problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5- Modern Tool Usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6-The Engineer and Society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7-Environment and Sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8- Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9- Individual and Team Work: Function effectively as an individual and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10- Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11- Project Management and Finance: Demonstrate knowledge and understanding of the engineering management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12-Life-long Learning: Recognize the need for and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological change.

Program Specific Outcomes (PSOs):

PSO1	Understand, design and analyze computer programs in the areas related to Algorithms, System Software, Web design, Big data, Artificial Intelligence, Machine Learning and Networking.
PSO2	Focus on improving software reliability, network security or information retrieval systems.
PSO3	Make use of modern computer tools for creating innovative career paths, to be an entrepreneur and desire for higher studies

SYLLABUS
CS304PC: COMPUTER ORGANIZATION AND ARCHITECTURE

UNIT - I

Digital Computers: Introduction, Block diagram of Digital Computer, Definition of Computer Organization, Computer Design and Computer Architecture.

Register Transfer Language and Micro operations: Register Transfer language, Register Transfer, Bus and memory transfers, Arithmetic Micro operations, logic micro operations, shift micro operations, Arithmetic logic shift unit.

Basic Computer Organization and Design: Instruction codes, Computer Registers Computer instructions, Timing and Control, Instruction cycle, Memory Reference Instructions, Input – Output and Interrupt.

UNIT - II

Microprogrammed Control: Control memory, Address sequencing, micro program example, design of control unit.

Central Processing Unit: General Register Organization, Instruction Formats, Addressing modes,Data Transfer and Manipulation, Program Control.

UNIT - III

Data Representation: Data types, Complements, Fixed Point Representation, Floating Point Representation.

Computer Arithmetic: Addition and subtraction, multiplication Algorithms, Division Algorithms,Floating – point Arithmetic operations. Decimal Arithmetic unit, Decimal Arithmetic operations.

UNIT - IV

Input-Output Organization: Input-Output Interface, Asynchronous data transfer, Modes of Transfer,Priority Interrupt Direct memory Access.

Memory Organization: Memory Hierarchy, Main Memory, Auxiliary memory, Associate Memory,Cache Memory.

UNIT - V

Reduced Instruction Set Computer: CISC Characteristics, RISC Characteristics.

Pipeline and Vector Processing: Parallel Processing, Pipelining, Arithmetic Pipeline, Instruction Pipeline, RISC Pipeline, Vector Processing, Array Processor.

Multi Processors: Characteristics of Multiprocessors, Interconnection Structures, Interprocessor arbitration, Interprocessor communication and synchronization, Cache Coherence.

TEXT BOOK:

1. Computer System Architecture – M. Moris Mano, Third Edition, Pearson/PHI.

REFERENCE BOOKS:

1. Computer Organization – Car Hamacher, Zvonks Vranesic, Safea Zaky, Vth Edition, McGraw Hill.
2. Computer Organization and Architecture – William Stallings Sixth Edition, Pearson/PHI.
3. Structured Computer Organization – Andrew S. Tanenbaum, 4th Edition, PHI/Pearson.

Course Outcomes

SNO	COURSE OUTCOMES	BT Level
1	Understanding the Basics of instructions sets and their impact on processor design	L1,L2,L3
2	Demonstrating the role of functional parts of a computer system, the purpose of control unit, different instruction sets formats and addressing modes.	L1,L2
3	Understanding the basics of computer arithmetic and its operations are performed on different data types like floating point and decimal point operations	L1,L2,L3
4	Analyzing the hardware parts of a computer system and its operations and understanding the hierarchy of memory organization.	L1,L2
5	Understanding the different instruction sets and pipeline techniques, the purpose of multi processors and its intercommunication process.	L1,L2,L3

Course Mapping with POs & PSOs

CO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
1	2	3	2	2					1			1	2	1	2
2	3	3	3	2					1			1	1	3	3
3	3	2	2	2									2	1	2
[illegible]	3	3	2	2					1			1	2	2	1
5	2	3	2	1								1	3	2	2
Avg :	2.6	2.8	2.2	1.8	0	0	0	0	0.6	0	0	0.8	2	1.8	2

NOTE : 1 -> low, 2 -> medium and 3 -> high

SCIENT INSTITUTE OF TECHNOLOGY

Ibrahimpatnam. R.R Dist - 501506

(Approved by AICTE & Affiliated to JNTUH, Hyderabad)

Class Time Table

CSE DEPARTMENT TIMETABLE
ACADEMIC YEAR 2021 – 2022

B-Tech. II Year /I Semester
Branch: CSE - A

W.E.F: 18-10-2021
L.H No: S-201

TIME	9:20 AM - 10:20 AM	10:20 AM - 11:20 AM	11:20 AM - 12:20 AM	12:20 PM-1:00PM	1:00 PM - 2:00 PM	2:00 PM – 3:00 PM	3:00 PM - 4:00 PM
DAY / PERIOD NO.	1st	2nd	3rd	LUNCH	4th	5th	6th
MON	C++ LAB				COA	ADE	DS
TUE	COA	COSM	DS		DS LAB		
WED	ADE LAB-A/ITWS LAB-B				COA	C++	ADE
THU	COA	COSM	DS		LOB	ADE	C++
FRI	COSM	C++	COSM		ADE	SPORTS	COA
SAT	C++	DS	COSM		ITWS LAB-A/ADE LAB-B		

SUBJECT	NAME OF THE FACULTY
DS	K.DEEPTHI
C++	SMD.SHAFIULLA
COA	B. RAMYA SREE
COSM	B. SABITHA
ADE	P.LAXMAN
DS LAB	S. NAGAJYOTHI/ M. SWETHA
C++ LAB	PRAVEEN SHINDE/ E.RAJU
ADE LAB	R.SRIRAMULU
ITWS LAB	M. SWETHA/ D.VENU GOPAL

CLASS-INCHARGE
(Mr. PRAVEEN SHINDE)

HOD

PRINCIPAL

SCIENT INSTITUTE OF TECHNOLOGY

Ibrahimpatnam. R.R Dist - 501506

(Approved by AICTE & Affiliated to JNTUH, Hyderabad)

CSE DEPARTMENT TIMETABLE
ACADEMIC YEAR 2021 – 2022

B. Tech. II Year /I Semester
Branch: CSE - B

W.E.F: 18-10-2021
L.H No: S-201

TIME	9:20 AM - 10:20 AM	10:20 AM - 11:20 AM	11:20 AM - 12:20 AM	12:20 PM-1:00PM	1:00 PM - 2:00 PM	2:00 PM - 3:00 PM	3:00 PM - 4:00 PM
DAY / PERIOD NO.	1st	2nd	3rd	L	4th	5th	6th
MON	ADE	DS	COA	U	DS LAB		
TUE	COSM	C++	COA		ADE	DS	SPORTS
WED	DS	COA	C++	N	COSM	COSM	DS
THU	COSM	LIB	C++	C	ITWS LAB/ADE LAB		
FRI	ADE LAB/ITWS LAB				C++	ADE	DS
SAT	COA	COSM	ADE	H	C++ LAB		

SUBJECT	NAME OF THE FACULTY
DS	K.DEEPTHI
C++	SMD.SHAFIULLA
COA	B. RAMYA SREE
COSM	B. SABITHA
ADE	P.LAXMAN
DS LAB	S. NAGAJYOTHI/ M. SWETHA
C++ LAB	PRAVEEN SHINDE/ E.RAJU
ADE LAB	V.DEVENDER
ITWS LAB	M. SWETHA/ D.VENU GOPAL

K. Anoosha
CLASS-INCHARGE
(Mrs. K.ANOOSHA)

HOD

PRINCIPAL

SCIENT INSTITUTE OF TECHNOLOGY

Ibrahimpatnam. R.R Dist - 501506

(Approved by AICTE & Affiliated to JNTUH, Hyderabad)

Individual Time Table

Ms. B. RAMYA SREE							
TIME	9:20 AM - 10:20 AM	10:20 AM - 11:20 AM	11:20 AM - 12:20 AM	12:20 PM-1:00 PM	1:00 PM - 2:00 PM	2:00 PM - 3:00 PM	3:00 PM -4:00 PM
DAY/PERIOD NO.	1st	2nd	3rd	LUNCH	4th	5th	6th
MON			COA-B		COA-A		
TUE	COA-A		COA-B				
WED		COA-B			COA-A		
THU	COA-A						
FRI							COA-A
SAT	COA-B						

LESSON PLAN

LECTURE SCHEDULE (COMPUTER ORGANIZATION AND ARCHITECTURE 2021-22 SEM-I) CSE-A

SNO	DATE	Topic(s)	No of Periods	Cumulative no of periods	Teaching Methodology (Chalk and Talk, PPTs, Video Lectures etc.)	Resources (T1, R1, OR1,OR2..)
		UNIT I				
1	20/10/2021	Introduction, Block diagram of Digital Computer Definition of Computer Organization, Computer Design and Computer Architecture.	1	1	Chalk and Talk	T1
2	21/10/2021	Register Transfer language, Register Transfer	1	2	Chalk and Talk	
3	22-10-2021,23-10-2021	Bus and Memory transfers	2	3	Chalk and Talk	
4	25-10-2021,26-10-2021	Arithmetic Micro operations	2	5	Chalk and Talk	
5	28/10/2021	Logic micro operations	1	7	Chalk and Talk	
6	29/10/2021	Shift micro operations, Arithmetic logic shift unit	1	8	Chalk and Talk	
7	03/11/2021	Instruction codes	1	9	Chalk and Talk	
8	05/11/2021	Computer Registers, Computer Instructions	1	10	Chalk and Talk	
9	08/11/2021	Timing and Control	1	11	Chalk and Talk	
10	9-11-2021,10-11-2021	Instruction Cycle	2	12	Chalk and Talk	
11	12/11/2021	Memory Reference Instructions	1	14	Chalk and Talk	
12	12/11/2021	Input-Output and Interrupt	1	15	Chalk and Talk	
		UNIT II				
13	15-11-2021,16-11-2021	Control memory	2	17	Chalk and Talk	T1
14	18/11/2021	Address sequencing	1	18	Chalk and Talk	
15	22/11/2021	Micro program example	1	19	Chalk and Talk	
16	23/11/2021	Design of control unit	1	20	Chalk and Talk	
17	25/11/2021	General register Organization	1	21	Chalk and Talk	
18	29/11/2021	Instruction formats, Addressing Modes	1	22	Chalk and Talk	
19	30-11-2021,1-12-2021	Data Transfer and Manipulation	2	24	Chalk and Talk	
20	02-12-2021,02-12-2021	Program Control	2	26	Chalk and Talk	
		UNIT III				
21	03/12/2021	Data types, Complements	1	27	Chalk and Talk	T1
22	07/12/2021	Fixed Point Representation, Floating Point Representation	1	28	Chalk and Talk	
23	10/12/2021	Addition and subtraction	1	29	Chalk and Talk	
24	16/12/2021	Multiplication Algorithms	1	30	Chalk and Talk	
25	17/12/2021	Division Algorithms	1	31	Chalk and Talk	
26	20/12/2021	Floating-point Arithmetic Operations	1	32	Chalk and Talk	

SNO	DATE	Topic(s)	No of Periods	Cumulative no of periods	Teaching Methodology (Chalk and Talk, PPTs, Video Lectures etc.)	Resources (T1, R1, OR1,OR2..)
27	21-12-2021,23-12-2021	Decimal Arithmetic unit	2	34	Chalk and Talk	
28	27/12/2021	Decimal Arithmetic operations	1	35	Chalk and Talk	
		UNIT IV				
29	28/12/2021	Input-Output Interface	1	36	Chalk and Talk	T1
30	03-12-2021,06-12-2021	Asynchronous data transfer	2	38	Chalk and Talk	
31	17/01/2021	Modes of Transfer	1	39	Chalk and Talk	
32	20/01/2021	Priority Interrupt	1	40	Chalk and Talk	
33	21/01/2021	Direct Memory Access	1	41	Chalk and Talk	
34	25-01-2021,27-01-2021	Memory Hierarchy, Main Memory	2	43	Chalk and Talk	
[illegible]	28/01/2021	Auxiliary Memory	1	44	Chalk and Talk	
[illegible]	31/01/2021	Associate Memory	1	45	Chalk and Talk	
37	01-02-2022,02-02-2022	Cache Memory	2	47	Chalk and Talk	
38	04/02/2022	CISC Characteristics, RISC Characteristics	1	48	Chalk and Talk	
		UNIT V				
39	05/02/2022	Parallel Processing	1	49	Chalk and Talk	T1
40	08/02/2022	Pipelining, Arithmetic Pipeline	1	50	Chalk and Talk	
41	09-02-2022,09-02-2022	Instruction Pipeline	2	52	Chalk and Talk	
42	10/02/2022	RISC Pipeline	1	53	Chalk and Talk	
43	16-02-2022,17-02-2022	Vector Processing, Array Processor	2	55	Chalk and Talk	
44	18/02/2022	Characteristics of Multiprocessors	1	56	Chalk and Talk	
45	21/02/2022	Interconnection Structures, Inter processor arbitration	1	57	Chalk and Talk	
46	23-02-2022,24-02-2022	Inter processor communication and Synchronization	2	59	Chalk and Talk	
47	25/02/2022	Cache Coherence	1	60	Chalk and Talk	

FACULTY SIGNATURE

HOD

PRINCIPAL

LESSON PLAN

LECTURE SCHEDULE (COMPUTER ORGANIZATION AND ARCHITECTURE 2021-22 SEM-I) CSE-B

SNO	DATE	Topic(s)	No of Periods	Cumulative no of periods	Teaching Methodology (Chalk and Talk, PPTs, Video Lectures etc.)	Resources (T1, R1, OR1,OR2..)
		UNIT I				
1	20/10/2021	Introduction, Block diagram of Digital Computer Definition of Computer Organization, Computer Design and Computer Architecture.	1	1	Chalk and Talk	T1
2	23/10/2021	Register Transfer language, Register Transfer	1	2	Chalk and Talk	
3	23-10-2021,25-10-2021	Bus and Memory transfers	2	3	Chalk and Talk	
4	26-10-2021,28-10-2021	Arithmetic Micro operations	2	5	Chalk and Talk	
5	30/10/2021	Logic micro operations	1	7	Chalk and Talk	
6	01/11/2021	Shift micro operations, Arithmetic logic shift unit	1	8	Chalk and Talk	
7	02/11/2021	Instruction codes	1	9	Chalk and Talk	
8	05/11/2021	Computer Registers, Computer Instructions	1	10	Chalk and Talk	
9	08/11/2021	Timing and Control	1	11	Chalk and Talk	
10	9-11-2021,12-11-2021	Instruction Cycle	2	12	Chalk and Talk	
11	15/11/2021	Memory Reference Instructions	1	14	Chalk and Talk	
12	16/11/2021	Input-Output and Interrupt	1	15	Chalk and Talk	
		UNIT II				
13	18-11-2021,20-11-2021	Control memory	2	17	Chalk and Talk	T1
14	22/11/2021	Address sequencing	1	18	Chalk and Talk	
15	23/11/2021	Micro program example	1	19	Chalk and Talk	
16	26/11/2021	Design of control unit	1	20	Chalk and Talk	
17	27/11/2021	General register Organization	1	21	Chalk and Talk	
18	29/11/2021	Instruction formats, Addressing Modes	1	22	Chalk and Talk	
19	30-11-2021,02-12-2021	Data Transfer and Manipulation	2	24	Chalk and Talk	
20	04/12/2021	Program Control	2	26	Chalk and Talk	
		UNIT III				
21	06/12/2021	Data types, Complements	1	27	Chalk and Talk	T1
22	07/12/2021	Fixed Point Representation, Floating Point Representation	1	28	Chalk and Talk	
23	10/12/2021	Addition and subtraction	1	29	Chalk and Talk	
24	16/12/2021	Multiplication Algorithms	1	30	Chalk and Talk	
25	17/12/2021	Division Algorithms	1	31	Chalk and Talk	
26	20/12/2021	Floating-point Arithmetic Operations	1	32	Chalk and Talk	

SNO	DATE	Topic(s)	No of Periods	Cumulative no of periods	Teaching Methodology (Chalk and Talk, PPTs, Video Lectures etc.)	Resources (T1, R1, OR1,OR2..)
27	21-12-2021,23-12-2021	Decimal Arithmetic unit	2	34	Chalk and Talk	
28	27/12/2021	Decimal Arithmetic operations	1	35	Chalk and Talk	
		UNIT IV				
29	28/12/2021	Input-Output Interface	1	36	Chalk and Talk	T1
30	03-12-2021,06-12-2021	Asynchronous data transfer	2	38	Chalk and Talk	
31	17/01/2021	Modes of Transfer	1	39	Chalk and Talk	
32	20/01/2021	Priority Interrupt	1	40	Chalk and Talk	
33	21/01/2021	Direct Memory Access	1	41	Chalk and Talk	
34	25-01-2021,27-01-2021	Memory Hierarchy, Main Memory	2	43	Chalk and Talk	
35	28/01/2021	Auxiliary Memory	1	44	Chalk and Talk	
36	31/01/2021	Associate Memory	1	45	Chalk and Talk	
37	01-02-2022,02-02-2022	Cache Memory	2	47	Chalk and Talk	
38	04/02/2022	CISC Characteristics, RISC Characteristics	1	48	Chalk and Talk	
		UNIT V				
39	05/02/2022	Parallel Processing	1	49	Chalk and Talk	T1
40	08/02/2022	Pipelining, Arithmetic Pipeline	1	50	Chalk and Talk	
41	09-02-2022,10-02-2022	Instruction Pipeline	2	52	Chalk and Talk	
42	11/02/2022	RISC Pipeline	1	53	Chalk and Talk	
43	16/02/2022	Vector Processing, Array Processor	2	55	Chalk and Talk	
44	17/02/2022	Characteristics of Multiprocessors	1	56	Chalk and Talk	
45	18/02/2022	Interconnection Structures, Inter processor arbitration	1	57	Chalk and Talk	
46	23-02-2022,24-02-2022	Inter processor communication and Synchronization	2	59	Chalk and Talk	
47	25/02/2022	Cache Coherence	1	60	Chalk and Talk	

FACULTY SIGNATURE

HOD

PRINCIPAL

UNIT-I

① (a) Explain functional units of digital Computer
(b) ~~wt~~ Explain different Computer instructions

– Dec 2017.

② a) Design a 4-bit Combinational circuit decrementer using four full adders.
(b) Derive the Control gates for the write input of the memory in the basic Computer – Dec-2019.

③ a) A digital Computer has a Common bus system for 16 Registers of 32 bits each. The bus is Constructed with multiplexers.
(i) How many selection inputs are there in an multiplexer?
(ii) What sizes of multiplexers are needed?
(iii) How many multiplexers are there in the bus – Dec-2019
(b) Construct Common Bus structure for 4 Registers.

– April 2018

④ List the Registers for the basic Computer and give their functionality in program execution.

– May 2019

⑤ a) Discuss about shift microoperations
b) An 8-bit Register Contains the binary value 10011100. What is the register value after an arithmetic shift right? Starting from initial ~~va~~ number 10011100, determine the register value after

an arithmetic shift left, and state whether there an overflow.

(c) Starting from initial value R = 11011101, determine the sequence of binary values in R after a logical shift-left, followed by a circular shift-Right, followed by a logical shift-right and a circular shift-left. — Nov 2020

6) Give a brief note on instruction Cycle — Dec 2018

7) Explain Memory Reference Instructions — Nov 2016.

8) Explain Timing and Control of basic Computer. — May 2015

9) What are the difference between a direct and an indirect address instruction. How many references to memory are needed for each type of instruction to bring an operand into a processor Register. — Dec 2017

COA Important Questions.

UNIT-II

1) Compare hardwired control and microprogrammed control unit Dec 2019, May 2019

2) With neat sketch Explain the block diagram of microprogrammed control unit Nov 2020

3) Elucidate the functioning of a Microprogram Sequencer April 2018

4) Formulate a mapping procedure that provides eight consecutive micro instruction for each routine. The operation code has 7 bits and control memory has 4096 words April 2018

5) What is addressing mode? List different types of addressing modes. Dec 2017

6) Explain various instruction formats May 2016

7) Evaluate the following arithmetic statement using zero, one, two and three address instructions. Use the conventional symbols and instructions May 2019

a) $X = (A+B) * (C+D)$

b) $X = \dfrac{A - B + C * (D * E - F)}{G + H * K}$

8) Explain about Program Control Instructions May 2017

Explain in detail about Data transfer and Manipulation instructions. - Nov 2020

What is an interrupt ? what are the uses of interrupts ? Explain about the different types of interrupts. Dec 2015.

UNIT-3

Important Questions

1) Convert the following decimal numbers to binary : 1231 ; 673 & 1998 — Dec 2019

2) Perform the arithmetic operations (+42) + (-13) and (-42) - (-13) in binary using 2's Complement representation for negative numbers. — Nov 2020.

3) Explain in detail with neat sketch of Booth Multiplication Algorithm — April 2018

4) Show that there can be no mantissa overflow after a multiplication operation — Dec 2019

5) Show the step-by-step multiplication process using booth algorithm when the following binary numbers are multiplied — April 2018

$$(+33) \times (-12)$$

6) What is floating point representation? Explain the IEEE standard for floating point representation with examples — Dec 2019.

7) Explain the ~~flowpoin~~ floating point Addition-Subtraction unit with a neat diagram — Dec 2019, Apr-18

Differentiate Isolated I/O and Memory mapped I/O what are advantages and disadvantages of each - Nov 2016 - May 2017

Explain about Source initiated transfer and destination initiated transfer using handshaking and Strobe Control with neat diagram - May 2017

Do Explain Daisy chain Priority Interrupt - May 2016

Design a circuit for a 4x4 First In First Out Buffer and explain its functionality - May 2019

Illustrate Asynchronous Communication Interface in detail. - April 2018

Describe the data transfer method in DMA - Dec - 2017

Describe DMA Controller - Dec 2016

Compare interrupt driven data transfer scheme with DMA. Using block diagram explain interrupt driven Dec 2018

How many characters per second can be transmitted over 1200 - baud line in each of the following modes? (Assume character code of 8 bits).

(a) Synchronous Serial transmission

(b) Asynchronous Serial transmission with two stop bits

(c) Asynchronous serial transmission with one stop bit - Dec 2015

10) Give a neat sketch that illustrates the Compon in a typical memory hierarchy. Dec 2019

11) Explain Associative memory hardware organization in detail Dec 2019

12) Explain in detail about Associative Mapping technique, and set Associative Mapping Dec 2017

13) A Computer uses RAM chips of 1024x1 capacity

(a) How many chips are needed, and how should their address lines be Connected to provide a memory capacity of 1024 bytes

(b) How many chips are needed to provide a memory capacity of 16K bytes. Explain in words how the chips are to be Connected to the address bus — Dec 2019, May 2017

UNIT -5

Important Questions.

1. Explain the arithmetic pipeline with Example. April 2018

2. What is pipelining ? what are its principles. Dec 2017

3. a) Discuss various Conflicts that might arise in a pipeline. How they are resolved

   b) Draw and explain the structure of General purpose Multicomputer. - Dec 2019

4. Elucidate Array processor - Apr 2018

5. Draw a space-time diagram for a six-segment pipeline showing the time it takes to process eight tasks. - Nov 2020

6. Consider the multiplication of two matrices 40 x 40 matrices using a vector processor

   (a) How many product terms are there in each inner product and how many inner products must be evaluated.

   (b) How many multiply add operations are needed to calculate the product matrix - Dec 2019

7. Explain various Inter Connection structures Apr 2018

UNIT-1

1) A sequence of instructions for the computer is called program
2) Microoperation is an elementary operation performed with the data stored in registers
3) A control function is a boolean variable that is equal to 1 or 0.
4) The CPU contains registers to store the data.
5) The symbolic notation used to describe the microoperation transfer among registers is called as Register Transfer Language
6) In three-state bus buffers the third state is high-Impedance
7) The digital circuit that generates the arithmetic sum of two binary numbers of any length is called Binary Adder.
8) Exclusive-OR operation can be used to selectively set bits of a register.
9) In Circular shift there is no loss of information.
10) An Arithmetic shift is a microoperation that shifts a signed binary number to the left or right.
11) An Instruction Code is a group of bits that instruct the computer to perform a specific operation

Scanned with CamScanner

12) The Opcode of an instruction is a group of bits that define operation

13) Program Counter (PC) is a processor register that holds the address of the next instruction to be executed

14) The timing for all registers in the basic computer is controlled by Master clock.

15) The clock pulses are applied to all flip-flops and Registers in the system and control unit.

16) control unit is organized in two ways. hardwired Control and Microprogrammed Control.

17) In hardwired organization, the control logic is implemented with decoders, ~~Multiplexers~~ flip-flops.

18) In Microprogrammed organization, the control information is stored in control memory.

19) fetch, decode and Execute are the three major phases of instruction cycle.

20) A computer can understand Binary language

21) A Multiplexer is a combinational circuit that receives binary information from one of 2^n input data lines and direct it to a single output line.

22) A Register is a group of flip-flops capable of storing binary information.

23) The transfer of information from a memory word to the outside environment is called a read operation

Scanned with CamScanner

24) Computer Architecture is concerned with the structure and behavior of the computer.

25) A combinational circuit that performs the arithmetic addition of two bits is called a Half adder.

26) A Combinational circuit that performs addition of three bits is called a full adder.

27) A flip-flop is a binary cell capable of storing one bit of information.

28) In Indirect address, the instruction specifies an address of a memory word in which address of the operand is found.

29) Memory Address register is used to hold the address of the memory.

30) The mask operation is an AND microoperation.

Choose the Correct Answers.

1) A _____ is a Co

Scanned with CamScanner

1) Control Unit in digital computers used to initiate the sequence of microoperations.
2) Hardwired and Microprogrammed are the two methods to implement control unit.
3) The control variables at any given time can be represented by string of 0's and 1's is called Control word
4) A memory that is a part of control unit is referred to as Control Memory.
5) A sequence of microinstructions constitutes a microprogram.
6) The Control address register specifies the address of the microinstruction.
7) The control data register is also called as pipeline register.
8) Microinstructions are stored in control memory in groups, with each group specifying a routine.
9) The transformation from the instruction code bits to an address in control memory where the routine is located is referred to as a mapping.
10) Control data register holds the present microinstruction
11) In microinstruction format, F1, F2, F3 are three fields used to specify microoperations for the computer.
12) In microprogrammed control unit, the control memory can be a ROM (Read Only Memory).

Scanned with CamScanner

13) The NOP symbol is used when the microinstruction has no microoperation.

14) Computers with stack organization would have PUSH and POP instructions.

15) In accumulator type organization, all operations are performed with an implied accumulator register.

16) One-Address instructions use an implied accumulator (AC) register for all data manipulations.

17) In Direct address mode the effective address is equal to the address part of the instruction.

18) An Example of general register type of organization is ADD R1, R2.

19) Program Counter holds the address of the instruction to be executed next and is incremented each time an instruction is fetched from memory.

20) In Indexed Addressing mode, the content of an index register is added to the address part of the instruction to obtain effective address.

21) Data transfer instruction cause transfer of data from one location to another.

22) program Control instructions provide decision-making capabilities and can change the path by program when executed.

23) Exchange instruction swap information between two registers or register and memory word.

Scanned with CamScanner

24) The four basic arithmetic operations are Addition, Subtraction, Multiplication & Division.

25) Logical operations perform binary operations on strings of bits stored in registers.

26) OR logical operation can be used for selectively complement bits of an operand.

27) Status bits are also called as flag (or) condition-code bits.

28) Recursive Subroutine is a subroutine that calls itself.

29) Subroutine is a self contained sequence of instructions that performs a given computational task.

30) Program Interrupt refers to the transfer of program control from a currently running program to another service program.

31) Interrupts causes break in the normal execution of a program.

32) Internal interrupts are also called as traps.

33) The collection of all status bit conditions in the CPU is sometimes called a program status word (PSW)

Scanned with CamScanner

UNIT - III

I. Choose The Correct Answers.

① A single unit which is composed of small group of bits is known as [D]

A) bit B) bug C) flag D) byte

② BCD stands for [A]

A) Binary coded Decimal
B) Binary coded digitals
C) Binary characters Decimals.
D) Binary conducting Decimals

③ System in which fractions are written by extending binary notation is called [A]

A) fixed-point representation.
B) floating-point representation.
C) binary. digits representation.
D) Single rotation representation.

④ Method of representing numbers such as 0's and 1's is called. [A]

A) binary notation
B) Primary notation
C) Secondary notation
D) Variable notation.

⑤ Numbers that are written with base 16 are classified as [B]

A) Whole numbers B) hexadecimal C) Exponential integers
D) Mantissa

Scanned by CamScanner

(6) One value which is written with mantissa and exponent is called [D]

A) Exponent B) Mantissa c) Base power

D) Floating point number

(7) Binary strings which are formed by replacing 0's by 1's and 1's by 0's is referred as [A]

A) One's Complement

B) Two's Complement

c) One's string

D) two's string

(8) The result of binary addition 1+1+1 is — [D]

A) Carry 0, sum 0. B) Carry 0, Sum 1

c) Carry 1, sum 0 D) Carry 1, Sum 1.

(9) Which of the following is a multiplication algorithm? [D]

A) Comparison method B) Restoring Method

c) Quick Sort Algorithm D) Booth's Algorithm.

10) Booth algorithm gives procedure for multiplying binary integers in [C]

A) Signed magnitude representation

B) Unsigned representation.

c) 2's complement representation.

D) None of the Above.

Scanned by CamScanner

11) first operand is called multiplicand and second [d]
a) Divisor b) Dividend c) Multiplication d) Multiplier

12) The decimal number 80 can be represented in [D]
BCD code as
A) 1000 0001 B) 0101 0000 C) 0010 0001
D) 1000 0000

13) The 1's complements of the binary number [B]
1101101 is
A) 0000010 B) 0010010 C) 0010011 D) 1100011.

14) Two's complement can be obtained from one's [A]
Complement by
A) adding 1 B) subtracting 1
C) Putting 1 as the leading bit D) None of these

15) what is the binary equivalent of the hexadecimal [C].
number B3 ?
A) 1011 0001 B) 1101 0011 C) 1011 0011
D) 1001 1111

16) _____ is a procedure for multiplying
binary numbers in signed-2's complement. (a)
(a) Booth's Algorithm (b) Restoring Algorithm
(c) Non-Restoring Algorithm (d) None.

Scanned by CamScanner

II Fill in the blanks

1) Full form of ASCII American standard code for Information Interchange.

2) An alphanumeric character set is a set of elements that includes 10 decimal digits, 26 letters of alphabet and special characters.

3) ASCII uses Seven bits to code 128 characters.

4) 9's Complement of 546700 is (999999 - 546700 =) 453299.

5) 2's complement of 101100 is (010011 + 1) 010100

6) 1, 0 is the binary values used to represent the negative number and positive number

7) A floating-point is said to be normalized if the most significant digit of mantissa is non-zero.

8) Floating-point is interpreted to represent a number in $m \times r^e$.

9) The solution to a problem is stated by a finite number of well-defined procedural steps is called an algorithm.

Scanned by CamScanner

10) Multiplication is a process of successive shift and add operations.

11) Booth algorithm is used for multiplying binary integers in signed-2's complement

12) A floating point number that has 0 in MSB position of the mantissa is said to have an underflow.

13) 16 is the radix value of hexadecimal

14) BCD stands for Binary-coded decimal.

15) Division of two fixed-point binary numbers is done with paper & pencil by a process of successive compare, shift and subtract operations.

Scanned by CamScanner

UNIT - IV

I choose the Correct Answers :-

1) The external devices connected to the I/O module are also known as _____. (a)

a) Peripheral devices
b) hardware devices.
c) Internet devices.
d) None of these

2) When I/O devices and the memory share the same address space, the arrangement is called _____. (b)

a) I/O mapped I/O
b) Memory-mapped I/O
c) Processor-mapped I/O
d) device-mapped I/O

3) The _____ enables the device to recognize its address when it appears on address lines. (d)

a) Control circuits.
b) data and status registers.
c) Input device.
d) address decoder

4) The advantage of I/O mapped devices to memory mapped is (c)

a) The former offers faster transfer of data
b) The devices connected using I/O mapping have a bigger buffer space.
c) The devices have to deal with fewer address lines
d) No advantage as such.

Scanned by CamScanner

5) The method of accessing the I/O devices by repeatedly checking the status flag is (a)

a) Program - controlled I/O
b) Memory - mapped I/O
c) I/O - mapped
d) None of the above

6) The method of synchronising the processor with the I/O device in which the device sends a signal when it is ready is (c)

a) Exceptions
b) Signal Handling
c) interrupts
d) DMA

7) The Method which offers higher speeds of I/O transfers is (d)

a) Interrupts
b) Memory Mapping
c) Program - Controlled I/O
d) DMA.

8) The processor where in the processor constantly checks the status flags is called as (a)

a) Polling
b) Inspection
c) Reviewing
d) Echoing

Scanned by CamScanner

9) The interrupts request line is a part of (b)

a) Data line
b) Control line
c) Address line
d) None of the above

10) _____ register is used for the purpose of controlling the status of each interrupt request in parallel priority interrupt (d)

a) Mass
b) Mark
c) Make
d) Mask.

11) The effectiveness of the cache memory is based on the property of _____ (a)

a) Locality of Reference
b) Memory localisation
c) Memory size.
d) None of the above

12) A page fault (c)

a) occurs when there is an error in a specific page.
b) occurs when a program accesses a page of main memory
c) occurs when a program accesses a page not currently in main memory.
d) occurs when a program accesses a page belonging to another page.

Scanned by CamScanner

13. The correspondence between the main memory blocks and those in the cache is given by (b)

a) Hash function.
b) Mapping function
c) Locale function
d) Assign function.

14) The technique for searching for a block by going through all the tags is (c)

a) Linear Search
b) Binary Search
c) Associate Search
d) None of the above.

15) The Algorithm to remove and place new contents into the cache is called ____ (a)

a) Replacement Algorithms
b) Renewal Algorithms.
c) Updation.
d) None of the above.

Scanned by CamScanner

UNIT- IV

II fill in the blanks :-

1) Baud rate is defined as the rate at which serial information is transmitted and is equivalent to the data transfer in bits per second.

2) A Control Command is issued to activate the peripheral and to inform it what to do.

3) Input or output devices attached to the Computer are called as peripherals.

4) Magnetic tape are used mostly for storing files of data.

5) A Controller is housed separately or may be physically integrated with the peripheral.

6) A status Command is used to test the status conditions in the interface and the peripheral

7) In Isolated I/O method the I/O addresses are isolated from memory addresses.

8) In Asynchronous data transfer instead of Common clock, Control signals are used to assist the data transfer.

9) DMA allows transfer of block of data directly between an external device and the main memory, without intervention by the processor.

10) The memory unit that communicates directly with the CPU is called the main memory.

11) Devices that provide backup storage are called auxiliary memory.

Scanned by CamScanner

(12) Disks that are permanently attached to the unit assembly and cannot be removed by the users are called hard disks.

(13) The performance of cache memory is frequently measured in terms of a quantity called hit ratio.

(14) The ROM portion of main memory is needed for storing an initial program called bootstrap loader.

(15) CAM stands for Content addressable memory.

(16) Example of hardware Priority Interrupts

(1) Daisy chaining ~~Method~~ priority.

(2) Parallel priority Interrupt

(17) Polling procedure is used to identify the highest-priority source by software-means.

Scanned by CamScanner

UNIT-V

Choose the Correct Answers

1) An instruction pipeline can be implemented by means of (b)
   a) LIFO buffer
   b) FIFO buffer
   c) Stack
   d) None of the above.

2) The pipelining process is also called as (b)
   a) Superscalar operation
   b) Assembly line operation
   c) Von Neumann cycle
   d) None of the above.

3) The fetch and execution cycles are interleaved with the help of _____ (b)
   a) Modification in processor architecture
   b) clock
   c) Special unit
   d) Control unit

4) The computer architecture aimed at reducing the time of execution of instructions is _____ (b)
   a) CISC
   b) RISC
   c) ISA
   d) ANNA

Scanned by CamScanner

5) Pipelining is a unique feature of —— (a)

a) RISC
b) CISC
c) ISA
d) IANA

6) The expression 'delayed load' is used in context of (c)

a) Processor - printer Communication
b) memory - monitor Communication.
c) pipelining
d) None of the above

7) Pipelining processing occurs on (c)

a) data stream
b) Instruction stream
c) both a and b
d) None of the above

8) _____ arise when an instruction depends on the result of previous instruction (a)

a) Data dependency
b) Resource conflict
c) Branch difficulties.
d) None of the above

Scanned by CamScanner

9) An instruction to provide small delay in program is (b)

a) LDA

b) NOP

c) BEA

d) None of the above

10) _____ performs computations on large arrays. (b)

(a) Multi processors

(b) Array processor

(c) Vector processor

(d) None of the above.

11) An _____ operates on a stream of instructions by overlapping fetch, decode and execute phases. (a)

a) Instruction pipeline

b) Arithmetic pipeline

c) RISC pipeline.

d) None of the above

12) _____ occurs when two segments wants to access to memory at same time. (c)

a) Data dependency

b) Branch difficulties

c) Resource conflicts

d) None of the above.

Scanned by CamScanner

13) _____ is maintained by the instruction fetch segment of the pipeline for high speed registed file. (b).

a) Branch target Buffer
b) loop buffer
c) Both a and b
d) None of the above

14) The compiler is designed to detect a data conflict and reorders the instructions to delay conflicting data by adding NOP instruction is referred to as (b)

a) Delayed buffer
b) delayed load
c) hardware interlock
d) None of the above.

15) _____ technique, the hardware detect a conflict and avoids it by routing the data through special paths to pipeline registers (c)

a) Branch target Buffer.
b) delayed branch
c) Operand forwarding.
d) None of the above

Scanned by CamScanner

UNIT V

Fill in the blanks.

1. A Computer with a large number of instructions is classified as Complex instruction set Computer.
2. Processing data concurrently is known as parallel processing.
3. MIMD stands for Multiple Instruction stream, multiple data stream.
4. SIMD represents an organization that includes many processing units under the supervision of a common control unit.
5. Performing fetch, decode and execute cycles for several instructions simultaneously to reduce overall processing time is referred to as instruction pipeline.
6. A Commercial Computer with vector instructions and pipelined floating-point arithmetic operations is referred to as supercomputer.
7. Array processors performs computations on large arrays of data
8. A Multiprocessor system in which each processor has its own private local memory is known as loosely coupled system.

Scanned by CamScanner

(9) A multiprocessor system with common shared memory is known as tightly coupled multiprocessor.

(10) Branch difficulties rise from branch and other instructions that change the value of PC.

(11) In Arithmetic pipeline divides an arithmetic operation into suboperations for execution in pipeline.

(12) RISC stands for Reduced Instruction set Computer.

(13) In Synchronous type of bus transfer, time slice is known in advance to both source and destination units.

(14) The Least Recently Used (LRU) algorithm gives the highest priority to the requesting device that has not used the bus for longer interval.

(15) Parallel processing may occur in instruction stream, data stream.

Scanned by CamScanner

Unit-I

Digital Computers

Introduction:-

The digital computer is an electronic computer in which the input consisting of combinations of numbers, letters or other characters written in an appropriate programming language but represented internally in binary notation.

→ The word digital implies that the information in the computer is represented by variables, which takes limited number of discrete values.

→ The first electronic digital computers was developed in 1940's mainly for the purpose of numerical computations. In this case, the discrete elements are the digits. Thus from this application it is termed as digital computer.

→ In practice, the digital computers function more reliably if only two states are used, because of the physical restriction of components and because of human logic tends to be binary.

→ Digital computers use the binary number system which has only two digits 0 and 1

→ A binary digit is called a 'bit'.

Scanned by CamScanner

→ Group of bits represents the information in the digital computer. The information may include letters, or decimal digits.

→ By using various coding techniques, the group of bits are used to develop set of instructions for performing computations.

Example

Consider,

$(1001011)_2$ – binary number

| Converted to

$(\quad)_{10}$ – decimal number system.

1001011 is converted by multiplying each bit by base 2 raised to an integer power which is

$$1\times2^6+0\times2^5+0\times2^4+1\times2^3+0\times2^2+1\times2^1+1\times2^0=75$$

→ The seven bits 1001011 represent a binary number whose decimal equivalent is 75.

→ The same group of bits also represents the letter K.

→ In other words, group of bits in digital computer are used to represent many different things which is similar to letters of an alphabet are used to construct different languages such as french & English.

Scanned by CamScanner

A computer system can be subdivided into two functional Entities.

(1) hardware

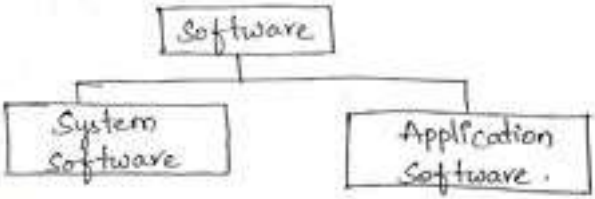
(2) software.

Hardware :- The hardware of the computer consists of all electronic components and electromechanical devices that comprise the physical entity of a device

Software :- Software consists of instructions and data that the computer manipulates to perform various data processing tasks.

program :- Sequence of instructions for the computer is called program.

Software is further divided into two types.



System software :- It is a collection of programs whose purpose is to make more effective use of the computer Eg:- Operating System

Application software :- Application programs written by the users for solving particular problem.

Scanned by CamScanner

Example:- A high level language program written by a user to solve particular data processing needs is a application program, but the compiler that translates the high-level program to machine language is a system software.

Block Diagram of a Digital Computer

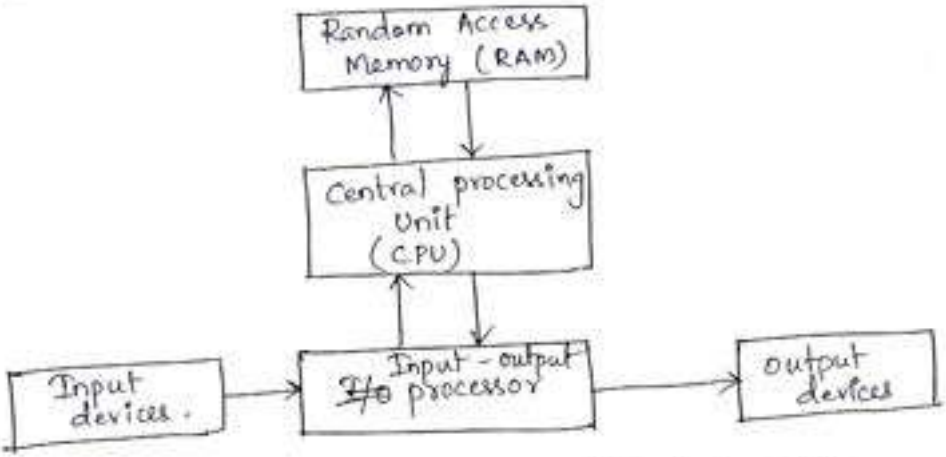


fig:- Block diagram of digital Computer.

Hardware of the Computer can be divided into three main parts.

(1) CPU
(2) RAM
(3) I/O. (Input-Output)

(1) CPU (Central processing unit):- It contains an arithmetic logic unit for performing arithmetic operations and for manipulating the data

Scanned by CamScanner

→ It also contains registers for storing data and control circuits for fetching and executing instructions

Control Unit :-

It is one of the component of cpu, which acts like a supervisor seeing that things are done in proper fashion.

→ It coordinates various operations using time signal.

→ It extracts instructions from memory and decodes and executes them by calling on the ALU when necessary.

→ All types of registers are managed by control unit

ALU (Arithmetic Logic Unit) :-

→ The actual processing of data and instruction are performed by ALU.

→ Data is transferred to ALU from storage unit. After processing the output is returned back to storage unit.

→ The major Arithmetic operations done by ALU are addition, subtraction, multiplication, division, logic and comparisions.

Scanned by CamScanner

Registers:-

Registers are the temporary storage areas of the computer processor.

→ It is managed by control Unit.

→ Registers holds data, instruction and address that are needed by program while running.

The above Control unit, ALU and registers are the part of the CPU.

② RAM :- Random Access Memory

RAM is the internal memory of the CPU which is used to store data, instructions, program results.

→ It is Volatile, i.e, data stored in it is lost when we switch off the computer or if there is a power failure.

→ The CPU can access any location in memory at random and retrieve the binary information within fixed interval of time.

So that's why it is called as 'Random Access Memory'

Scanned by CamScanner

③ Input-Output Processor (I/O processor)

The I/O processor contains electronic circuits for communicating and controlling the transfer of information between the computer and the outside world.

→ The input and output devices connected to the computer may include keyboard, printers, magnetic disk drives and other communicating devices.

Scanned by CamScanner

Computer Organization :-

-> It is concerned with the way the hardware components operate and the way they are connected together to form the computer system.

-> Once the hardware components are placed we verify the organizational structure operated as intended.

Computer Design :-

-> It is concerned with the hardware design of the computer.

-> Once the specifications to design the computer is ready, it is the task of the designer to develop hardware for the system.

-> It also deals with the determination of what hardware should be used and how the parts are connected.

-> It is also sometimes referred as 'Computer Implementation'.

Computer architecture :-

Computer architecture is concerned with the structure and behavior of the computer.

-> It also includes the information format, information set and techniques for addressing memory.

Scanned by CamScanner

→ It is also concerned with the specifications of the various functional modules, such as processors and memories and structuring them together into a computer system.

Scanned by CamScanner

Register Transfer Languages :-

Introduction

A digital system is a interconnection of digital hardware modules which does the information processing task.

→ The modules may be registers, decoders, arithmetic elements and control logic.

→ These modules are connected with common data and control paths to form digital computer.

→ Registers are the best Example for digital modules which is used to hold data, instruction, address.

Microoperations :- The operations executed on data stored in registers are called microoperations.

→ Microoperation is a elementary operation performed on the information stored in one or more registers

→ The result of microoperation may replace the previous binary information of a register or it may be transferred to another register.

Examples of microoperation:

Store, shift, Count, clear and hold.

Scanned by CamScanner

The internal hardware organization of a digital computer can be best defined by

1. The set of registers it contains and their function.
2. The sequence of microoperations performed on binary information in the registers.
3. The control that initiates the sequence of microoperation.

→ We can explain the sequence of microoperation in a computer by using words, but this procedure involves lengthy descriptive explanation.

→ It is more convenient to use symbols to describe the sequence of transfers between registers and various Arithmetic and logic microoperation.

Register Transfer language :-

Definition :- The symbolic notation used to describe the microoperation transfers among the registers is called a Register Transfer Language.

→ The term "Register Transfer" implies the availability of hardware logic circuits that can perform a stated microoperation and transfer the result of the operation to the same or another registers.

Scanned by CamScanner

→ where as the term "language" which is similar to programming language. Programming language is a procedure for writing symbols to specify the computational process.

→ A Register Transfer Language is a system for expressing in symbolic form the microoperation sequences among the registers of a digital module.

<u>for Example</u> :- To add two no's following microoperation sequences has to be performed.

1. Load first number in register 1
2. Load second number in register 2
3. Perform add microoperation.
4. Store the result in the destination register

→ These are the 4 sequence operation which are performed to add 2 no's.

Scanned by CamScanner

Register Transfer

→ Generally registers are represented by using capital letters. Sometimes it may be followed by number.

Eg :-

Registers Representation

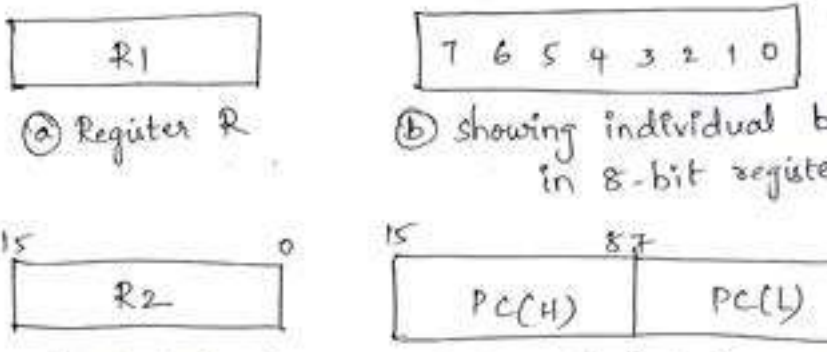
MAR - Memory Address Register

PC - Program Counter.

IR - Instruction Register.

R1 - Processor Register.

→ The individual flip flops in an n-bit register are numbered in sequence from 0 to n-1



(a) Register R

(b) showing individual bits in 8-bit register.

c) Numbering of bits.

d) divided into two parts.

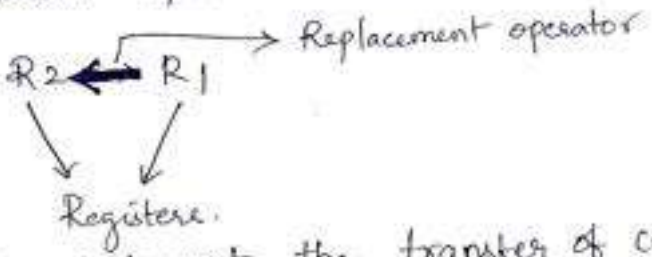
fig:- block diagram of register.

The numbering of 16-bit register can be marked on top of the box.

Scanned by CamScanner

→ 16-bit register in the above fig(d) represent PC. The numbers PC(0-7) represent low order byte and PC(8-15) represent high-order byte.

→ Information transfer from one register to another is designated in symbolic form by means of a replacement operator



In this statement the transfer of content occurs from Register R1 to R2

→ If we want the transfer to occur only under a pre-determined control condition. It can be shown by means of an if then statement.

$$\text{if } (P=1) \text{ then } (R_2 \leftarrow R1)$$

where P is a control signal generated in control section.

It is sometimes convenient to separate the control variables from the register transfer operation by specifying a control function.

Scanned by CamScanner

Control function :-

A control function is a boolean variable that is equal to 1 or 0. The control function is included in the statement as follows.

$$\boxed{P : R_2 \leftarrow R_1}$$

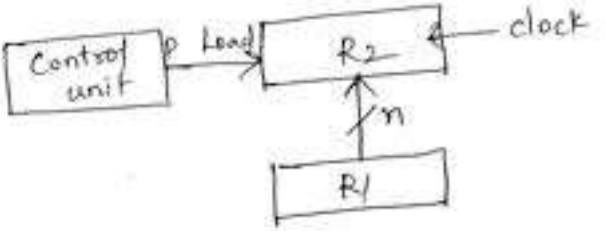
Statement with control function.

→ Control function is terminated with a colon (:)

→ It symbolizes the requirement that the Transfer operation be executed by the hardware only if P=1.

→ Every statement written in a register transfer notation implies a hardware construction for implementing the transfer.

→ The below figure shows the block diagram that explains the transfer from R1 to R2



(a) Transfer from R1 to R2 when P = 1

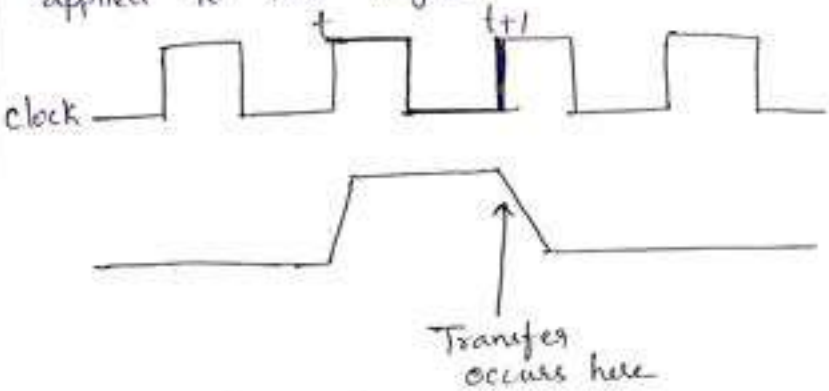
In the above block diagram. R1 & R2 are registers in which n outputs of R1 are connected to the n i/p's of register R2

Scanned by CamScanner

The value of n is replaced by the actual number when the length of the register is known.

→ Register R2 has a load input that is activated by the control variable P.

→ It is assumed that the control variable is synchronized with the same clock as the one applied to the register.



b) Timing diagram

In the timing diagram, P is activated in the control section by the rising edge of a clock pulse at time 't'.

→ The next positive transition of the clock at time 't+1' finds that load input is active and here the data inputs of R2 are then loaded into the register in parallel.

→ P goes to 0 at time 't+1'.

Scanned by CamScanner

→ In the register transfer statement, the clock is not included as a variable.

→ All transactions occur during a clock-edge transition.

→ Even though the control condition such as P becomes active just after time 't', the actual transfer does not occur until the register is triggered by the next positive transition of the clock at time 't+1'.

$T: R_2 \leftarrow R1, \quad R1 \leftarrow R_2$

In the above statement, there are two register transfers which are separated by comma but the exchange of contents of two registers occurs during only one common clock pulse provided at T=1.

Basic Symbols for Register Transfer:

Symbol	Description	Examples
Letters (and numerals)	Denotes a Register	MAR, R2
Parentheses ( )	Denotes a part a register	R2 (0-7), R2 (L)
Arrow ←	Denotes transfer of information	R2 ← R1
Comma ,	Separates two microoperations	R2 ← R1, R1 ← R2

Scanned by CamScanner

Bus and Memory Transfer

Introduction

→ A typical digital computer has many registers, we need to connect each of these registers with individual registers such that information transfer occurs between these registers.

→ So, for that case we require some common bus system

→ The number of wires will be excessive if separate lines are used between each register and all other registers in the system.

for Example :- we have two registers A & B with each 4-bit.

If we want to transfer the data b/w this two register A & B we need to make the line wirings. It looks like in the fig a.

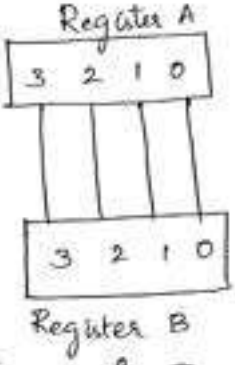


fig (a).

→ Now consider we add one more register c to the above register, and we need to transfer the data from register c to register A and register c to register B. The line wirings may look like in the fig (b)

Scanned by CamScanner

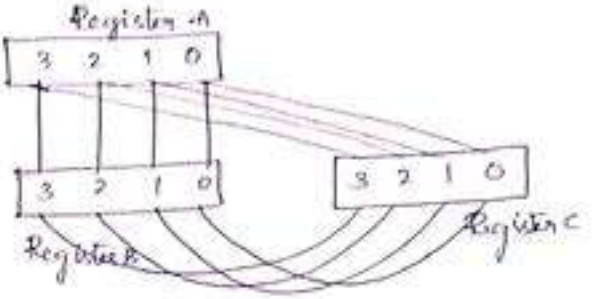


fig (b).

These ab line wiringe between register c and A and register B and C becoming Complex when we added register C.

→ If we add one more register D and connect all the registers A, B, C then the complexity increases.

→ In order to avoid the above complexity we have a solution i.e., common bus system.

Common bus System:-

→ A more Efficient scheme for transferring information in a multiple-register configuration is a common bus system.

→ A bus structure consists of a set of common lines, one for each bit of a register through which binary information is transferred one at a time.

→ Control signals determine which register is selected by the bus during each register transfer.

Scanned by CamScanner

Constructing a Common Bus System:- (for 4 Registers)

→ One way of constructing a common bus system is with multiplexers. Consider 4 Multiplexers to construct this bus system

Multiplexer:- A multiplexer is a combinational circuit that receives binary information from one of 2^n input data lines and directs it to a single output line.

→ Consider 4 Registers each of 4 bits, numbered 0 to 3, and two selection inputs, S_1 and S_0.

→ Multiplexers select the source register whose binary information is then placed on the bus.

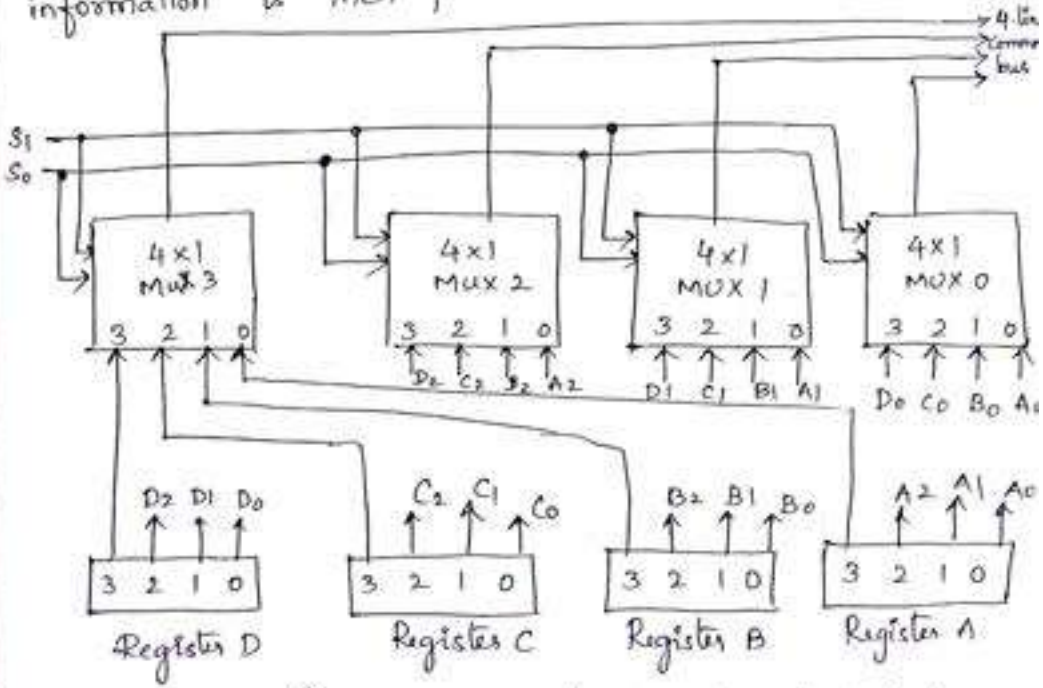


fig:- Common Bus System for four registers.

Scanned by CamScanner

In the above figure, In order not to complicate the diagram with 16 lines crossing each other, we use labels to show connections from the output's of the register to i/p of the multiplexer.

→ We took 4 Multiplexers of 4X1 size means it takes 4 inputs and produces only 1 output

→ The register D3 output line is connected to the multiplexer 3rd line. The line 2 of the same multiplexer is connected to the 3rd bit or C3 bit of Register C.

→ That means the same significant position bits in each register are connected to the data inputs of one multiplexer to form one line of the bus.

→ Thus MUX 0 multiplexer connected with the four 0 bits i.e, (A0, B0, C0, D0) of the registers, MUX 1 multiplexer connected with the four 1 bits of i.e, (A1, B1, C1, D1) of the registers, and similarly for the other bits.

→ ~~Now~~ The inputs are ready for the multiplexers now we consider the select lines. There are two select lines S0 and S1 which are connected to the each of the selection inputs of all

Scanned by CamScanner

four multiplexers.

→ We have 4 outputs each one from 1 Multiplexer which goes to a four line common bus.

→ The selection lines S_1 and S_0 choose the four bits of one register and transfer them into the four-line common bus.

→ When $S_1 S_0 = 00$, the 0 data inputs of all four multiplexers are selected and applied to the outputs that form the bus.

→ This causes the bus lines to receive the content of register A (A0, A1, A2, A3), since the outputs of this register are connected to the 0 data inputs i.e., (A0, A1, A2, A3) of the multiplexers.

Table: Function Table for Bus.

S_1	S_0	Register Selected
0	0	A
0	1	B
1	0	C
1	1	D

→ The output of the multiplexers is connected to the common Bus system. Now, on the bus there are bits A0, A1, A2, A3 bits register A.

→ The o/p's of Multiplexers MUX3, MUX2, MUX1, MUX0

Scanned by CamScanner

i.e., A3, A2, A1, A0 are now placed on the Common bus line.

Scanned by CamScanner

→ A bus system :

→ The number of multiplexers needed to construct the bus is equal to n, where n is the no of bits in each register.

→ The size of the multiplexer must be k×1 since it multiplexes k data lines.

for Example:- ① A common bus for 8 registers of 16 bits each requires 16 multiplexers, one for each line in the bus.

② And Each multiplexer must have eight input lines and three selection lines to multiplex one significant bit in the eight registers.

→ The symbolic statement for a bus transfer between the bus, register may be implied in the statement as follows :

$$BUS \leftarrow C \quad , \quad R1 \leftarrow BUS$$

. The content of register C is placed on the bus, and the content of the bus is loaded into the register R1 by activating load control input.

Scanned by CamScanner

Three - State Bus Buffers

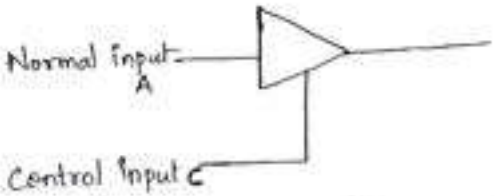
Buffer :- A data buffer is a region of a physical memory storage used to temporarily store data while it is being moved from one place to another. ~~or~~ ~~to~~

→ A Bus system can be constructed with three-state gates instead of multiplexers.

→ A three-state gate is a digital circuit that exhibits three states.

→ Two signals equivalent to logic 1 and logic 0. and the third state is a high-impedance state.

→ The high-impedance state behaves like third state open circuit, which means that the output is disconnected.



output Y = A if C = 1

High impedance if C = 0

Input A	C	output Y
0	1	0
1	1	1
0	0	High impedance
1	0	High impedance

fig :- Graphic Symbols for three-state buffer.

→ It has both a normal input and Control input.

→ The Control input determines the output state.

Scanned by CamScanner

→ When the control input is equal to 1, the output is enabled and gate behaves like normal buffer, with the output equal to the normal input.

→ When the control input = 0, the output is disabled and the gate goes to a high-impedance state i.e, output is disconnected.

→ Because of this high-impedance state, a large number of three-state gate outputs can be connected with wires to form a common-bus line without Endangering loading effects.

Construction of bus system with three-state buffers

→ Consider four buffers, whose outputs are connected to the single bus line.

→ The four buffers represents the 4 bits of data i.e, A_0, B_0, C_0, D_0 which are the normal inputs to buffers.

→ Not more than one buffer is active at any given time.

→ The control inputs to the buffers determine which of the four normal inputs will communicate with the bus line.

→ The buffers are connected, must be controlled so that only one three-state buffer has access to

Scanned by CamScanner

the bus line and the remaining buffers are in high-impedance state.

→ Inorder to ensure that no more than one control input is active at any given time is to use a "decoder".

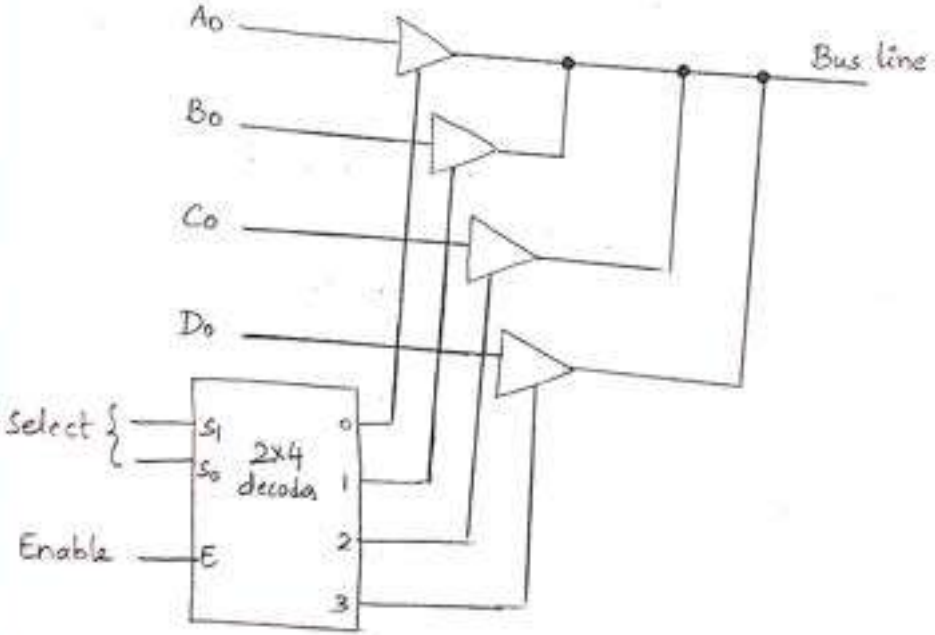


figure :- Bus line for three-state buffers

→ when the Enable input = 0, all four outputs = 0 and the bus line is in high-impedance state because all buffers are disabled.

→ when the Enable input = 1 (active), one of the three-state buffers will be active, depending

Scanned by CamScanner

upon the binary value in select lines S_1 and S_0.

Select inputs →	S_0	S_1	
binary values	0	0	0 → A_0 is active can access bus line
	0	1	1 → B_0 is active, A_0, C_0, D_0 - high-impedance state
	1	0	2 → C_0 is active, all other in high-impedance state
	1	1	3 → D_0 is active, all other in high-impedance state.

Note :- To construct a common bus for 4 Registers using three-state buffers, we need n circuits with 4 buffers as in the above fig.

→ Each group of four buffers receives one significant bit from the four registers.

→ Each common output produces one of the lines for the common bus for a total of n lines.

→ But only one decoder is necessary to select between the four registers.

Scanned by CamScanner

Memory Transfer

Memory:-

A memory unit is a collection of storage cells together with associated circuits, needed to transfer information in and out of storage.

→ The memory stores binary information in group of bits called "words".

→ A memory word is a group of 0's and 1's and may represent a number, an instruction code, one or more alphanumeric characters.

→ A Group of 8 bits is called a 'byte'.

→ Each word in memory is assigned an identification number, called as 'address' starting from 0 to 2^k-1 where k is the no. of address lines.

Memory Transfer

→ The Transfer of information from a memory word to the outside environment is called a 'read operation'

→ The transfer of new information to be stored into the memory is called a 'Write operation'

Scanned by CamScanner

→ A memory word is symbolized by the letter 'M'.

→ The particular memory word among the many available words can be selected by the memory address during the transfer.

Eg:- 1 M[AR]

M represents Memory word

Address is Enclosed within the square brackets which specifies the transfer to be done to this particular address.

Eg 2 :- Consider a memory unit that receives the address from a register, called the 'address register', symbolized by AR.

→ The data are transferred to another register, called the data register, symbolized by DR

Read operation can be stated as:

$$\boxed{\text{Read : } DR \leftarrow M[AR]}$$

This causes transfer of information into DR from the memory word M selected by the address in AR.

Scanned by CamScanner

The write operation transfers the content from a data register to a memory word M selected by the address.

→ we consider the data is in Register R1 and the address is in AR.

The write operation can be stated as follows

$$\boxed{\text{Write}: M[AR] \leftarrow R1}$$

This causes the transfer of information from R1 into the memory word M selected by the address in AR.

Arithmetic Microoperations :-

Microoperation :-

A microoperation is an elementary operation performed with the data stored in registers.

The microoperations most often encountered in digital computers are classified into four categories :

1) Register transfer microoperations transfer binary information from one register to another.
2) Arithmetic micro operations perform arithmetic operation on numeric data stored in registers.

Scanned by CamScanner

(3) Logic microoperations perform bit manipulation operations on nonnumeric data stored in registers.

(4) Shift microoperations perform shift operations on data stored in register.

Arithmetic Microoperations

Arithmetic Microoperations perform arithmetic operations on numeric data stored in registers.

→ The arithmetic operations may be addition, subtraction, increment and decrement.

Eg:- Add Microoperation defined by

$$\boxed{R3 \leftarrow R1 + R2}$$

The content of R1 is added to Register R2 and the result is stored in another Register R3

Subtract Microoperation

$$\boxed{R3 \leftarrow R1 - R2}$$

The content of R2 is subtracted from R1 and the result is stored in R3

But in binary number subtraction is implemented through complementation and addition. Instead of using '−' (minus) operator, we can specify the

Scanned by CamScanner

Subtraction as follows

$$\boxed{R3 \leftarrow R1 + \overline{R2} + 1}$$

$\overline{R2}$ is 1's Complement of R2

$\overline{R2} + 1 \rightarrow$ Adding 1 to 1's complement produces the 2's Complement.

Now R1 is added to 2's Complement of R2 (which is equivalent to R1 − R2) and the result is stored in R3 register.

Table :- Arithmetic Microoperation.

Symbolic Notation	Description.
$R3 \leftarrow R1 + R2$	Contents of R1 plus R2 transferred to R3.
$R3 \leftarrow R1 - R2$	Contents of R1 minus R2 transferred to R3
$R2 \leftarrow \overline{R2}$	1's Complement on the Contents of R2
$R2 \leftarrow \overline{R2} + 1$	2's Complement the contents of R2
$R3 \leftarrow R1 + \overline{R2} + 1$	R1 plus 2's complement of R2 (Subtraction)
$R1 \leftarrow R1 + 1$	Increment the contents of R1 by 1
$R1 \leftarrow R1 - 1$	decrement the Contents of R1 by 1

Scanned by CamScanner

Binary Adder

The digital circuit that generates the arithmetic sum of two binary numbers of any length is called a "binary Adder".

for Example :- Consider Registers R1 & R2 which Contains 4 bits each. The addition microoperation Can be performed on Registers as follows.

Carry(c)	1	1	1	0
R1	1	1	0	1
R2 +	0	1	1	1
Sum(S)	1	0	1	0	0

Addition	Sum	carry
0+0	0	0
0+1	1	0
1+0	1	0
1+1	0	1

So, to implement add microoperation with hardware we need

1. Registers to hold the data
2. digital Component that perform arithmetic addition.

Hardware Implementation of 4bit Binary Adder :-

4-bit Binary Adder :-

To implement binary addition we need a 'FULL ADDER'

Scanned by CamScanner

Full adder :- A digital circuit that forms the arithmetic sum of two bits and a pervious carry is called a full-adder.

→ Consider two registers R1 & R2 whose data bits inputs as A_n & B_n to the full adder

→ To construct 4-bit binary Adder we need 4-full adders.

→ These full-adders are connected in cascade, with each full adder having 2 inputs A & B and 2 outputs carry and sum (we take initially carry = 0).

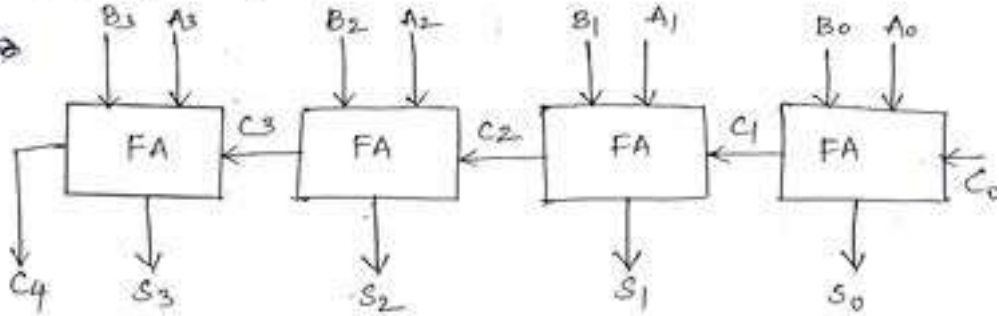


fig:- 4-bit binary adder

→ Initially, A_0, B_0 are the inputs to the full adder which perform addition operation on it. Since, we perform add operation from right to left so firstly we add the least significant bits or low-order bits with carry (C_0) is '0'.

Scanned by CamScanner

→ $A_0 + B_0$ generates two outputs they are S_0 and C_1. ~~Now~~ C_1 All the carries are connected in chain through the full adders. C_1 is input to the next full adder.

→ All the bits from right to left are added in the FA connected to it by leaving the carry to next FA and S as the output.

→ The S outputs of all the full-adders generate the required sum bits. and C_4 is the output carry.

Note:-

for n-bit binary adder requires n-full adders.

→ The output carry from each full adder is connected as input carry to the next higher order full adders.

→ The n-data bits for the A inputs come from one register (such as R1) and the n-data bits for the B inputs come from another register (R2)

→ The sum can be transferred to a third register (R3) or one of the source register (R1 or R2) replacing its previous content.

Scanned by CamScanner

Binary Adder - Subtractor :-

→ Binary - Adder Subtractor will perform both binary addition and binary subtraction using a single digital circuit.

→ The subtraction operation of A-B can be done by taking 2's complement of B and adding it to A i.e, $A+\bar{B}+1$. The 2's complement can be obtained by taking the 1's complement and adding one to the least significant pair of bits.

Eg :- Add Microoperation

$R3 \leftarrow R1 + R2$

Subtract Microoperation

$R3 \leftarrow R1 - R2$

$R3 \leftarrow R1 + \overline{R2} + 1$

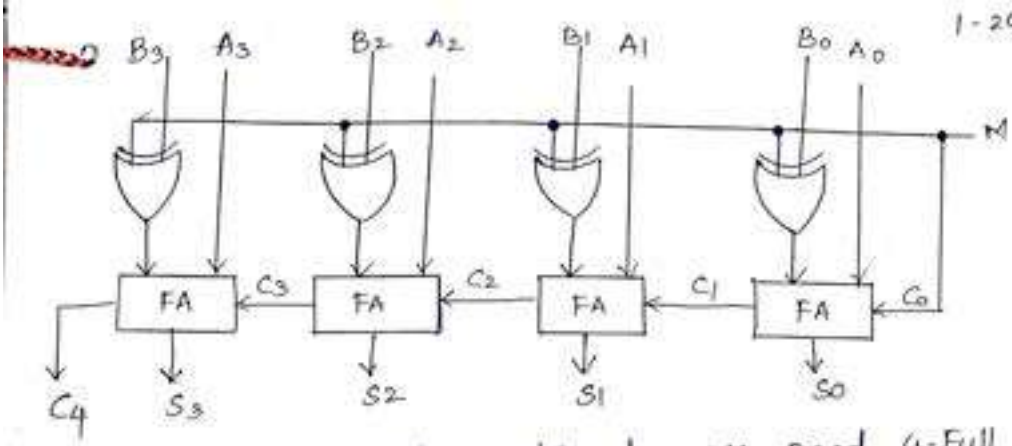
→ The 1's complement can be implemented with inverters and a one can be added to the sum through the input carry.

→ 4-bit binary Adder - subtractor construction :-

→ The addition and subtraction operations can be combined into one common circuit by including an exclusive - OR gate with each full-adder.

→ A full-adder is a combinational circuit which performs add operation by taking three input bits.

Scanned by CamScanner



→ So, for 4-bit adder-subtractor we need 4-Full Adders with three inputs each.

→ one input from A register. (A_0, A_1, A_2, A_3) and Second input is output of XOR of B_0 and M. third input is the carry bit which is propagated from ~~for~~ right to left.

→ The mode input M controls the operation.

when M = 0 the circuit is adder

M = 1 the circuit is subtractor.

→ Each XOR gate receives input M and one of the inputs of B.

→ When M = 0, we have $B \oplus 0 = B$. The full adder receive the value of B, the input carry is 0, and the circuit performs A plus B.

→ when M = 1 we have $B \oplus 1 = B'$ and $C_0 = 1$. The B inputs are all complemented and a 1 is

Scanned by CamScanner

added through the input carry. The circuit perf the operation A plus the 2's Complement of B.

$M = 0$: $A + B$ (Addition) $(\because B \oplus 0 = B)$

$M = 1$: $A + \bar{B} + 1$ (Subtraction)

Binary Incrementer

→ The increment microoperation adds one to a number in a register

Eg :-

C	0	0	1	
R1	1	1	0	1 → least significant bit
				1
Sum	1	1	1	0

In the above Example, we have 4-bit size register so we have 4 bits in R1. we carried addition to 2 bits only for the next Consequetive bits are also 2 bits and we are getting 2 outputs sum and carry

So we can half adder to implement the 4-bit binary incrementer.

Note :- Full adder is used to perform arithmetic sum of three-inputs.

Scanned by CamScanner

Construction of 4-bit binary Incrementor :-

	0	0	0	1
R1	1	1	0	1
+				1
Sum	1	1	1	0

→ for 4-bit binary incrementor we use 4-half adders

→ one of the inputs to the least significant half-adder (HA) is connected to logic-1 and the other input is the least significant bit of the number.

→ The output carry from one half-adder is connected to one of the inputs of the next-higher-order half-adder.

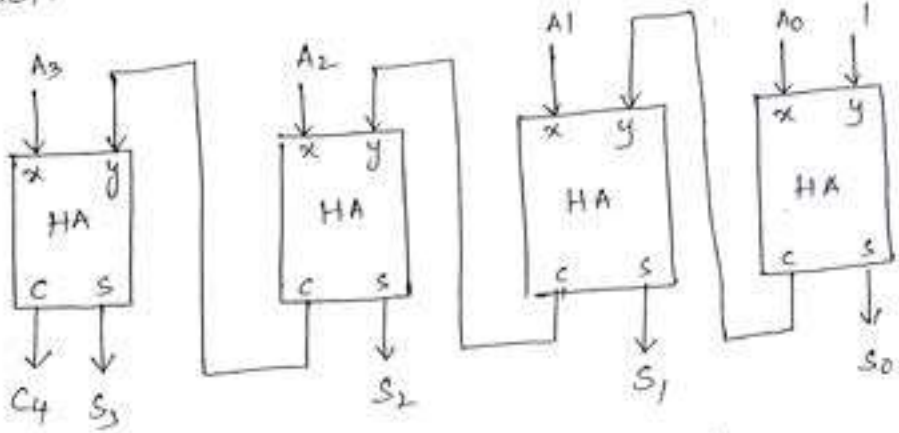


fig:- 4-bit binary Incrementor.

→ The circuit receives one input for each HA from register i.e., A_0, A_1, A_2, A_3 adds one to the A_0 and then carry is propagated to the next HA and for each HA the sum/output S_0, S_1, S_2, S_3 are generated.

Scanned by CamScanner

Note :-

→ for n-bit binary incrementer by extending the diagram to include n half-adders.

→ Always the least significant bit should be connected to logic 1.

Arithmetic Circuit

→ The arithmetic microoperations can be implemented in one composite arithmetic circuit.

→ The basic component of an arithmetic circuit is the parallel adder.

→ By controlling the data inputs to the adder, it is possible to obtain different types of arithmetic operations.

→ The 4-bit arithmetic circuit has four full-adder circuits that constitute the 4-bit adder and 4-Multiplexers for choosing different operations

→ There are two 4-bit inputs (A_0, A_1, A_2, A_3) and (B_0, B_1, B_2, B_3) and 4-bit output (D_0, D_1, D_2, D_3)

→ The 4-A inputs are connected to the 4 Full adders and each of the four inputs from B are connected to the data inputs of the multiplexers.

Scanned by CamScanner

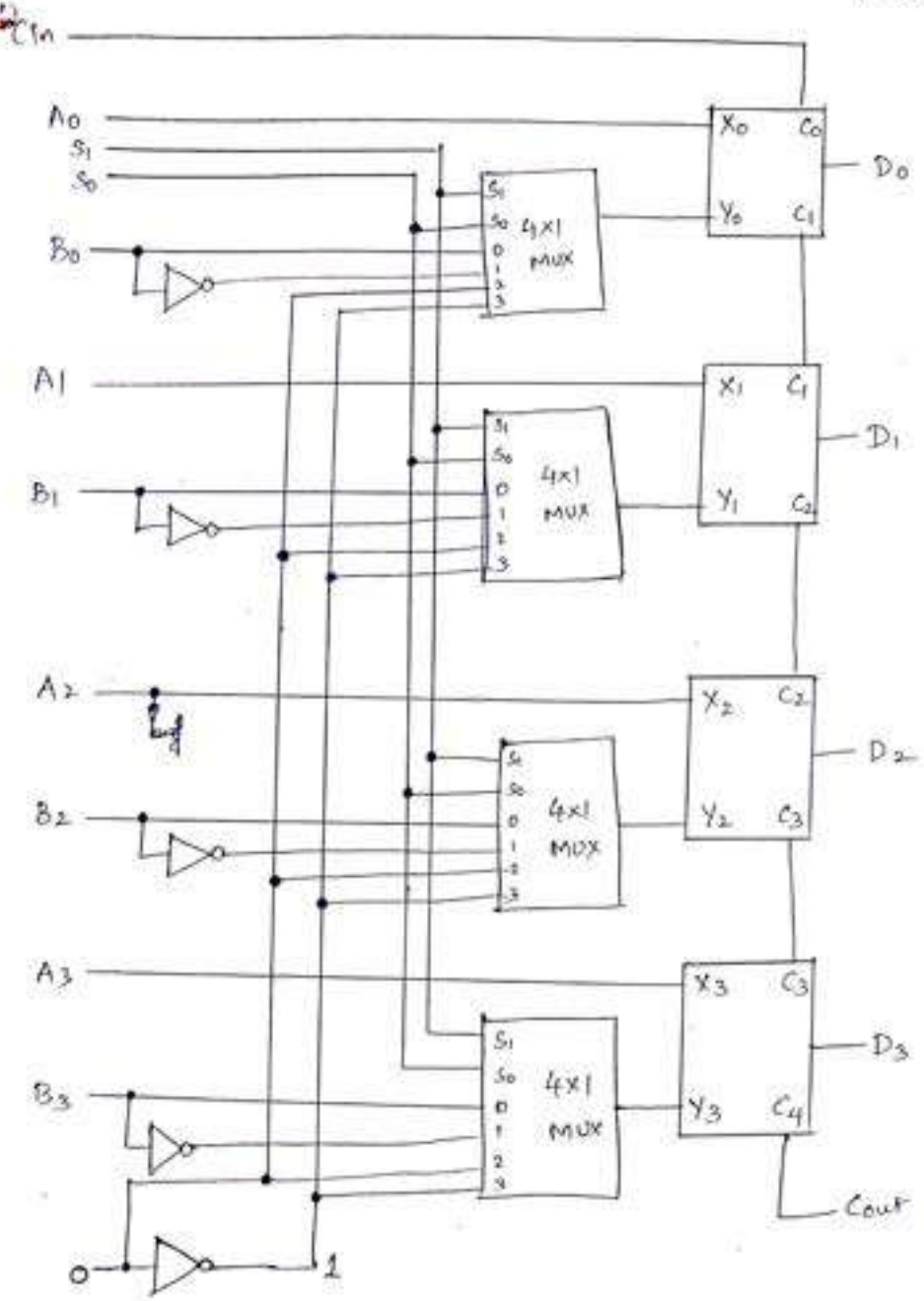


fig:- 4-bit Arithmetic circuit

Scanned by CamScanner

→ The Complement of B ie, $\overline{B}$ or B' is also connected to M as input to Com multiplexer.

→ The other two data inputs are connected to logic-0 and logic-1

→ logic-0 is fixed voltage value and logic-1 signal can be generated through an inverter whose input is 0.

→ The four multiplexers are controlled by two selection inputs, S_1 and S_0

→ The input carry Cin goes to the carry input of the FA in the least significant position

→ The other carries a connected from one stage to the next.

The output of the binary adder is calculated from the following arithmetic sum:

$$\boxed{D = A + Y + C_{in}.}$$

A - 4-bit binary number which inputs as X to FA

Y - 4 bit binary number which inputs to binary adder

Cin - input carry which can be equal to 0 or 1

Scanned by CamScanner

+ = arithmetic plus.

→ By Controlling the Y with two selection inputs S_0 and S_1 and making $C_{in} = 0$ or 1, it is possible to generate eight arithmetic microoperations.

Select			Input	Output	Microoperation
S_1	S_0	C_{in}	Y	$D = A + Y + C_{in}$	
0	0	0	B	$D = A + B$	Add
0	0	1	B	$D = A + B + 1$	Add with carry
0	1	0	$\overline{B}$	$D = A + \overline{B}$	Subtract with borrow
0	1	1	$\overline{B}$	$D = A + \overline{B} + 1$	Subtract
1	0	0	0	$D = A$	Transfer A
1	0	1	0	$D = A + 1$	Increment A
1	1	0	1	$D = A - 1$	Decrement A
1	1	1	1	$D = A$	Transfer A

Table :- Arithmetic Circuit function Table

Scanned by CamScanner

Addition (with carry and without carry)

when $S_1S_0 = 00$, the value of B is applied to the Y-inputs of the adder if $C_{in} = 0$ the output $D = A + B$

$$D = A + Y + C_{in} \qquad (S_1S_0 = 00,\ Y = B,\ C_{in} = 0)$$

$$D = A + B + 0$$

$$\boxed{D = A + B}$$ addition operation is performed (without carry)

if $S_1S_0 = 00$, $C_{in} = 1$, $Y = B$

$$D = A + Y + C_{in} \qquad (Y = B \text{ Given})$$

$$\boxed{D = A + B + 1}$$ (addition with carry)

S_1S_0 when	decimal value
00	0
01	1
10	2
11	3

Subtraction :-

when $S_1S_0 = 01$ (decimal value is 1) which takes as input as 1 for multiplexer) the complement of B is applied to the Y-inputs of the adder i.e., $Y = \overline{B}$ and $C_{in} = 1$ then

$$S_1S_0 = 01,\ Y = \overline{B},\ C_{in} = 1$$

$$D = A + Y + C_{in}$$

$$= A + \overline{B} + 1 \qquad (\text{subtraction } A - B)$$

when $Y = \overline{B}$, $C_{in} = 0$ then

$$D = A + Y + C_{in}$$

Scanned by CamScanner

$D = A + \overline{B} + 0$ (Subtract with borrow i.e., A-B-1)

Increment

when $S_1 S_0 = 10$ (which is decimal no-2 w.r.t multiplexer)

the inputs from B are neglected, all 0's are inserted into the Y-inputs. The output becomes

$D = A + 0 + C_{in}$ $(Y = 0)$.

if $C_{in} = 0$

$D = A + 0 + 0$

$\therefore$ $\boxed{D = A}$ (transfer from input A to output D)

if $C_{in} = 1$

$D = A + 0 + 1$

$D = A + 1$ (The value of A is incremented by 1)

Decrement :-

If $S_1 S_0 = 11$ (decimal value is 3) all 1's are inserted into Y's input of the adder to produce decrement operation

$\boxed{D = A - 1 \quad \text{if } C_{in} = 0}$

Eg :- A-1

$$\begin{array}{r} 1101 \\ - \quad 1 \\ \hline 1100 \end{array} \xrightarrow{\text{2' complement (1' complement + 1)}} \begin{array}{r} 1101 \\ + \; 1111 \\ \hline 1\,1100 \end{array}$$

discard the carry

Scanned by CamScanner

The decrement operation of A-1 can be done as $F = A +$ 2'Complement of 1

(2' Complement of 1 = 1's of complement of 1 + 1)

i.e, A + 2's of Complement of 1

0001 (Normal 1 can be written as 0001)

1110 1's complement

+1 2's complement.

$F = A + 1111$ → decrement operation

Now A is added to 2's Complement of 1 (ie, 0001 after 2's Complement is 1111) to decrement the value of A by 1.

$\boxed{F = A + 1111}$ gives the decrement value of A

when $C_{in} = 1$, then $D = A - 1 + 1 = A$ direct transfer of input A to output D.

$\boxed{D = A}$

Scanned by CamScanner

Logic Microoperations

→ Logic microoperations specify binary operations for strings of bits stored in the registers.

→ These operations consider each bit of the register separately and treat them as a binary variables.

Eg:- Consider XOR microoperation with the contents in the registers R1 and R2

$$P: R1 \leftarrow R1 \oplus R2$$

It specifies a logical microoperation to be executed on the individual bits of the registers provided that the control variable P = 1

→ Consider R1 contain 1010 and R2 contain 1100.

The XOR operation stated as

Contents of R1	1010
Contents of R2	1100
	0110

→ Content of R1 after P=1

The content of R1, after the execution of the microoperation is equal to bit-by-bit exclusive OR operation on the bits in R2 and previous values of R1.

→ These logical microoperations are used in scientific computations for making logical decisions

Scanned by CamScanner

Special Symbols :-

→ Special symbols will be adopted for the logic microoperations OR, AND and complement to distinguish corresponding symbols used to express Boolean functions or Control functions

Eg :-

Symbols	Used to Represent
$\vee$	OR Microoperation
$\wedge$	AND Microoperation
$\overline{B}$ (Bar on top of Register)	1's Complement

→ When the symbol + occurs in a microoperation, it will denote an arithmetic plus, when + occurs in a control function it will denote an OR operation.

Eg :- $P+Q : R1 \leftarrow R2 + R3$, $R4 \leftarrow R5 \vee R6$

'+' between P & Q is an OR operation ~~whereas~~ between two binary variables of a control function. The '+' between R2 and R3 specifies an add microoperation

List of Logic Microoperations :-

→ There are 16 different logic operations that can be performed with two binary variables.

Scanned by CamScanner

Table :- Sixteen logic Microoperations

Boolean function	Microoperation	Name.
$F_0 = 0$	$F \leftarrow 0$	clear
$F_1 = xy$	$F \leftarrow A \wedge B$	AND
$F_2 = xy'$	$F \leftarrow A \wedge \bar{B}$	
$F_3 = x$	$F \leftarrow A$	Transfer A
$F_4 = x'y$	$F \leftarrow \bar{A} \wedge B$	
$F_5 = y$	$F \leftarrow B$	Transfer B
$F_6 = x \oplus y$	$F \leftarrow A \oplus B$	Exclusive - OR
$F_7 = x + y$	$F \leftarrow A \vee B$	OR
$F_8 = (x+y)'$	$F \leftarrow \overline{A \vee B}$	NOR
$F_9 = (x \oplus y)'$	$F \leftarrow \overline{A \oplus B}$	Exclusive NOR
$F_{10} = y'$	$F \leftarrow \bar{B}$	Complement B
$F_{11} = x + y'$	$F \leftarrow A \vee \bar{B}$	
$F_{12} = x'$	$F \leftarrow \bar{A}$	Complement A
$F_{13} = x' + y$	$F \leftarrow \bar{A} \vee B$	
$F_{14} = (xy)'$	$F \leftarrow \overline{A \wedge B}$	NAND
$F_{15} = 1$	$F \leftarrow$ all 1's	Set to all 1's

Scanned by CamScanner

Truth tables for 16 functions of two variables

x	y	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

The above table explains the table of 16 logic microoperation from truth values.

Hardware Implementation of logic microoperations

→ Inorder to implement logic microoperation we need logic gates for each bit and or pair of bits in the registers to perform the required logic function.

→ Although these are 16 logic microoperations, most computers use only four - AND, OR, XOR (Exclusive OR) and complement - from which all others can be derived.

→ The circuit that generates the four basic logic microoperations consists of four gates and a multiplexer

→ Each of the four logic operations is generated through a gate that performs the required logic.

Scanned by CamScanner

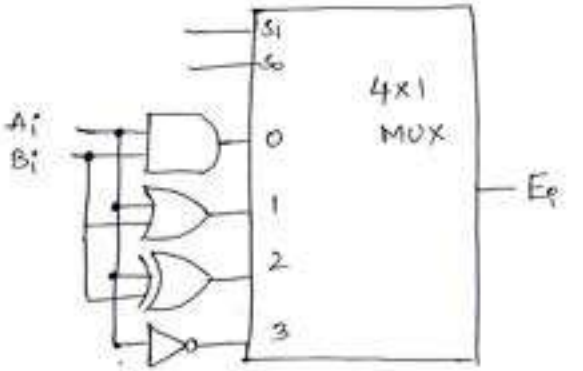


fig (a) Logic Diagram

S_1	S_0	output	operation
0	0	$E = A \wedge B$	AND
0	1	$E = A \vee B$	OR
1	0	$E = A \oplus B$	XOR
1	1	$E = \bar{A}$	Complement

fig (b) Functional table.

→ The outputs of the gates are supplied as input to the multiplexer

→ The Two selection inputs S_1 and S_0 choose one of the data inputs of the multiplexer and directs the value to the output.

→ fig b Explains about the output generated and operation performed when based on the selection inputs S_1 and S_0.

Applications of logic Microoperations

1. Selective set Operation :-

→ The selective-set operation sets to 1 the bits in register A where there are corresponding 1's in register B.

Scanned by CamScanner

→ It does not affect bit positions that have 0's in B.

→

Eg:-

1 0 1 0	A before
1 1 0 0	B (logic operand)
1 1 1 0	A after

The OR microoperation can be used to selectively set bits of a register.

2. Selective Complement Operation:-

→ The selective - complement operation complements bits in A where there are corresponding 1's in B.

→ It does not affect bit positions that have 0's in B.

1 0 1 0	A before
1 1 0 0	B (logic operand)
0 1 1 0	

→ The Exclusive - OR microoperation can be used to selectively set bits of a register.

3. Selective clear Operation :-

→ The selective - clear operation clears to 0 the bits in A only where there are corresponding 1's in B.

→ It does not affect bit positions that have 0's in B

Scanned by CamScanner

1 0 1 0	A before
1 1 0 0	B (logic operand)
0 0 1 0	A after.

→ The corresponding logic operation is $A \leftarrow A \wedge B'$.

4) Mask Operation

→ The mask operation is similar to the selective-clear operation except that the bits of A are cleared only where there are corresponding 0's in B.

1 0 1 0	A before
1 1 0 0	B (logic operand)
1 0 0 0	A after.

→ The mask operation can be achieved by an 'AND' microoperation.

5) Insert Operation

→ The insert operation inserts a new value into a group of bits.

→ This is divided into two parts.

1) Masking them
2) ORing them with required value.

→ The mask operation is an AND microoperation and the insert operation is an OR microoperation

Scanned by CamScanner

Consider an Example of inserting 1001 in place

0110

Mask

A	0110	1010
B	0000	1111
A	0000	1010

AND operation

Insert

A	0000	1010
B	1001	0000
A	1001	1010

OR operation

(6) Clear operation

→ The clear operation compares the words in A and B and produces an all 0's result if the two numbers are equal

1	0	1	0	A
1	0	1	0	B
0	0	0	0	A ← A ⊕ B

→ This operation is achieved by an exclusive-OR microoperation.

Scanned by CamScanner

Shift Microoperations:-

→ Shift Microoperations are used for serial transfer of data.

→ These microoperations are used in combination with arithmetic, logic, and other data-processing operations

→ The contents of the register can be shifted to the left or the right.

→ During a shift-left operation the serial input transfers a bit into the rightmost position.

→ During a shift-right operation the serial input transfers a bit into leftmost position.

→ There are three types of shifts:

(i) Logical
(ii) Circular
(iii) Arithmetic.

Logical shift:-

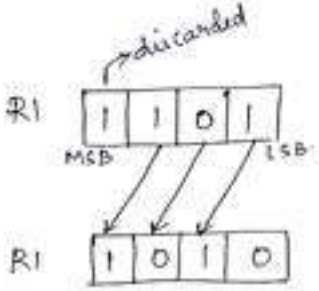
A logical shift is one that transfers 0 through the serial input.

→ we adopt, the symbols shl and shr for logical shift-left and shift-right microoperations.

Example:-

$R1 \leftarrow shl\ R1$
$R2 \leftarrow shr\ R2$

Scanned by CamScanner



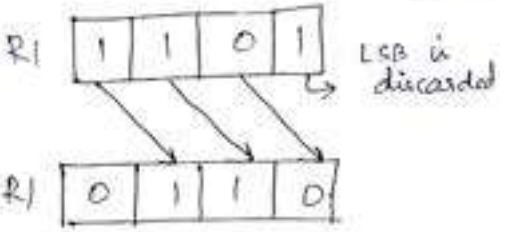


fig:- logical shift left : shl fig:- logical shift right : shr

Note :- 1 1 0 1

MSB (Most significant Bit)

LSB (least significant Bit)

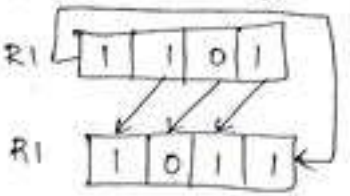
→ After shifting all the bits left - right and right to left, the End position is assumed to be 0 during a logical shift.

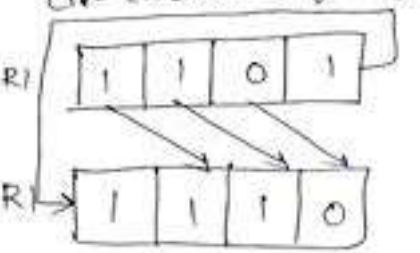
Circular shift :-

→ The circular shift also known as a rotate operation circulates the bits of the register around the two Ends without loss of information.

→ This is can be achieved by connecting the serial output of the shift register to its serial input.

Eg :- cil - circular shift left cir - circular shift right





Scanned by CamScanner

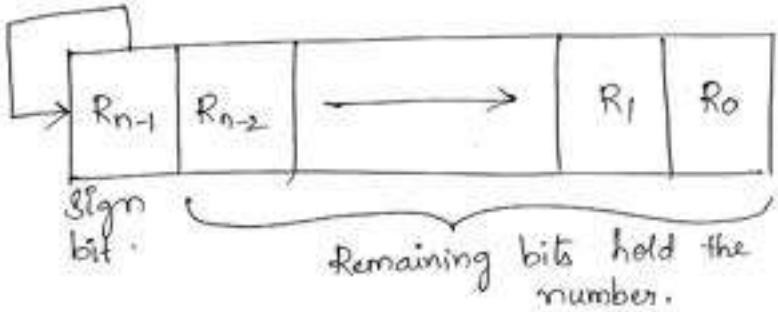
3. Arithmetic shift :-

An arithmetic shift is a microoperation that shifts a signed binary number to the left or right.

→ An arithmetic shift-left multiplies a signed binary number by 2.

→ An arithmetic shift-right divides the number by 2.

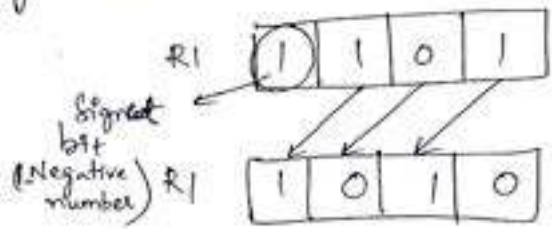
→ Arithmetic shifts must leave the sign bit unchanged because the sign of the number remains the same when it is multiply or divide by 2.



Sign bit
0 — positive.
1 — negative.

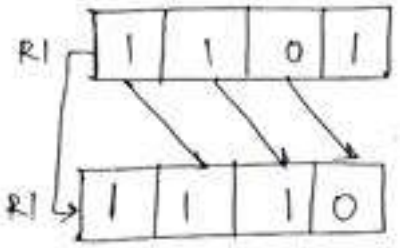
Negative numbers are in 2's complement form.

Eg:- ashl - arithmetic shift left



Scanned by CamScanner

ashr - arithmetic shift right



(MSB = 1
Negative number)

(In shift right operation MSB is Copied/~~to~~ shifted to the same R1)

→ So, In arithmetic shift-left inserts a 0 into R_0, and shifts all other bits to the left.

→ The initial bit R_{n-1} is lost and replaced by the bit from R_{n-2}

→ whereas in shift-right operation, the last MSB bit should be copied as it is after the shift.

Hardware Implementation :-

4-bit Combinational circuit shifter :-

→ A Combinational circuit shifter can be constructed with multiplexers.

→ The 4-bit shifter has four data inputs, A_0 through A_3, and four data outputs, H_0 through H_3.

→ There are two serial inputs, one for the shift left (I_L) and the other for shift right (I_R).

Scanned by CamScanner

→ When the selection input S=0, the input data are shifted right

→ When S=1, the input data are shifted left

→ A shifter with n data inputs and outputs requires n multiplexers.

→ The two serial inputs can be controlled by another multiplexer to provide the three possible types of shifts.

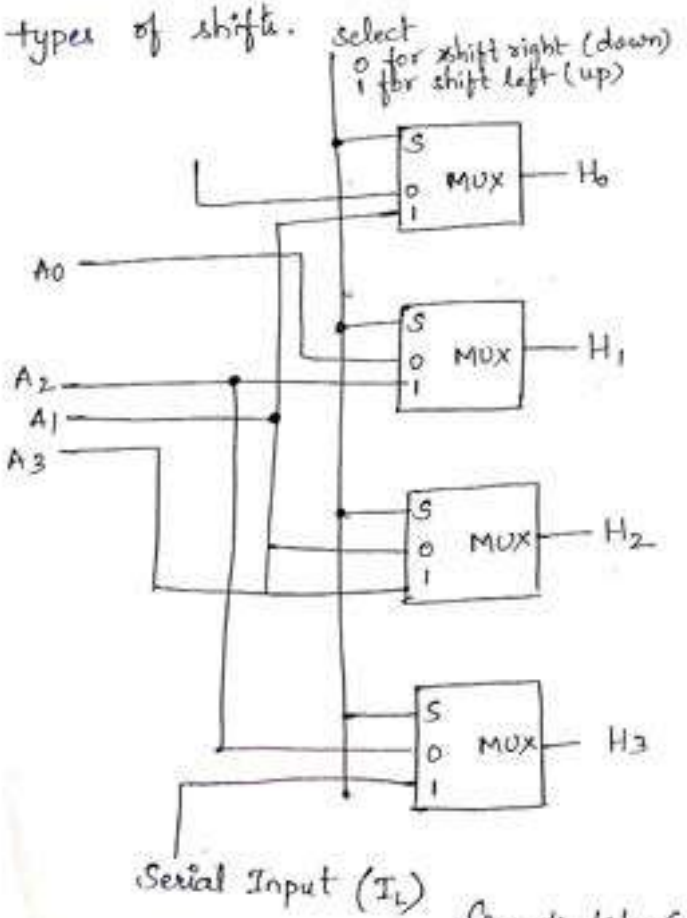


fig:- 4-bit Combinational circuit shifter.

Scanned by CamScanner

Functional Table

Select	output			
S	H_0	H_1	H_2	H_3
0	I_R	A_0	A_1	A_2
1	A_1	A_2	A_3	I_L

Table :- shift Microoperations

Symbolic designation	Description
R ← shl R	shift-left Register R
R ← shrR	Shift-Right Register R
R ← cil R	circular shift-left register R
R ← cir R	circular shift-right register R
R ← ashl R	Arithmetic shift-left R
R ← ashr R	Arithmetic shift-right R.

Scanned by CamScanner

Arithmetic Logic Shift Unit :-

→ Instead of individual registers performing the microoperations directly Computer System Employ a number of storage registers connected to a common operational unit called an ALU i.e, arithmetic logic unit.

→ To perform a microoperation, the contents of specified registers are placed in the inputs of the Common ALU.

→ The ALU performs an operation and the result of the operation is then transferred to a destination register.

→ The ALU is a Combinational circuit so that the entire register transfer operation through the ALU and into the destination register can be performed during one clock pulse.

→ The shift operations are performed in a separate unit.

→ The Arithmetic, Logic and shift circuits which we discussed in previous topic can be combined into one ALU with Common selection Variables.

Scanned by CamScanner

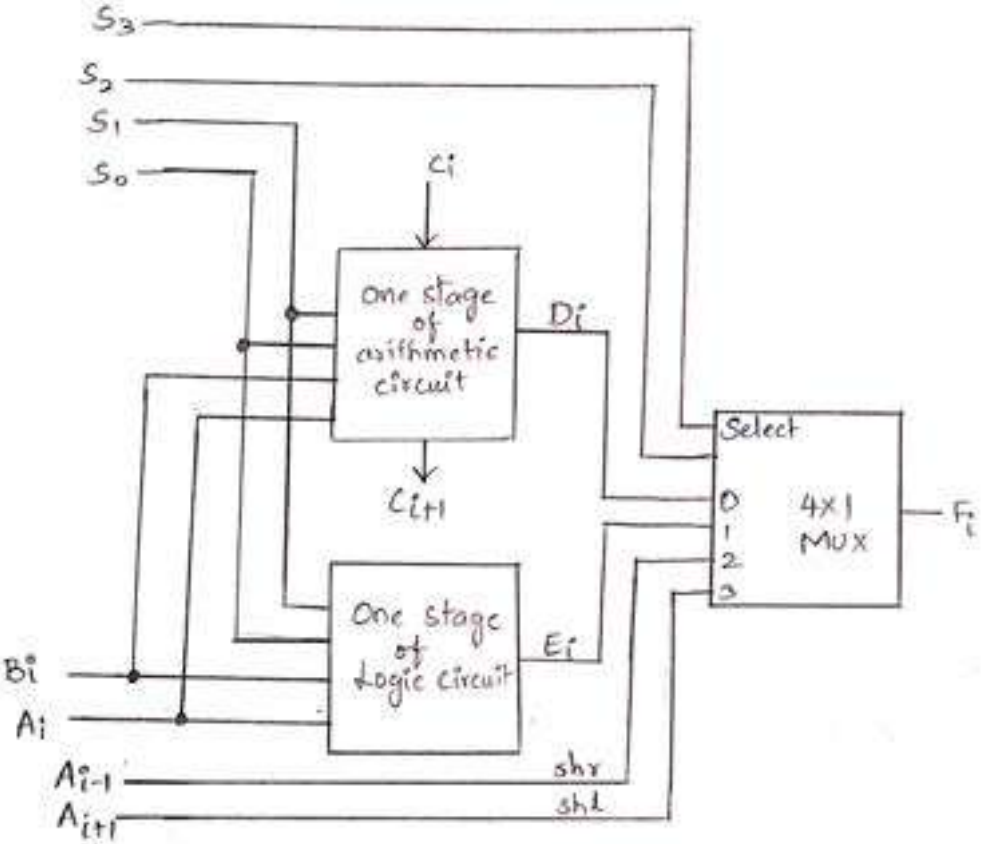


fig :- One stage of arithmetic logic shift un

→ The inputs A_i and B_i are applied to both the arithmetic and logic units.

→ A particular microoperation is selected with inputs S_1 and S_0

→ A 4×1 Multiplexer at the output chooses between an arithmetic output in E_i and a logic

Scanned by CamScanner

Output in H_i

→ The data in the multiplexer are selected with inputs S_3 and S_2.

→ The other two data inputs are received by the multiplexer they are A_{i-1} for the shift-right operation and A_{i+1} for the shift-left operation.

→ The circuit must be repeated n times for an n-bit ALU.

→ The output carry C_{i+1} of given arithmetic stage must be connected to the input carry C_i of the next stage in sequence.

→ The input carry to the first stage is the input carry C_{in}, which provides a selection variable for the arithmetic operations.

→ The circuit provides eight arithmetic, four logic and two shift operations.

→ Each operation is selected with the five variables S_3, S_2, S_1, S_0 and C_{in}.

→ The input carry C_{in} is used for selecting an arithmetic operation only.

Scanned by CamScanner

Operation select					Operation	Function
S_3	S_2	S_1	S_0	C_{in}		
0	0	0	0	0	$F = A$	Transfer A
0	0	0	0	1	$F = A+1$	Increment A
0	0	0	1	0	$F = A+B$	Addition
0	0	0	1	1	$F = A+B+1$	Add with carry
0	0	1	0	0	$F = A+\overline{B}$	Subtract with borrow
0	0	1	0	1	$F = A+\overline{B}+1$	Subtraction
0	0	1	1	0	$F = A-1$	Decrement A
0	0	1	1	1	$F = A$	Transfer A
0	1	0	0	X	$F = A \wedge B$	AND
0	1	0	1	X	$F = A \vee B$	OR
0	1	1	0	X	$F = A \oplus B$	XOR
0	1	1	1	X	$F = \overline{A}$	Complement A
1	0	X	X	X	$F = shr\ A$	Shift right A into F
1	1	X	X	X	$F = shl\ A$	Shift left A into F.

Table :- Function table for Arithmetic logic shift Unit

→ The 8 arithmetic operations are selected with $S_3S_2=00$.

→ The next four are logic operations and are selected with $S_3S_2=01$.

→ The input carry has no effect during logic operations and is marked as 'don't care' X's.

→ The last two operations are shift operations are selected with $S_3S_2=10$ and 11. The other i/p's have no effect on shift.

Scanned by CamScanner

Instruction Codes:-

→ The internal organization of a digital system is defined by the sequence of microoperation it performs on data stored in its register.

→ The general purpose digital computer is capable of executing various microoperations and can be instructed as to what sequence of operations it must perform.

program:-

A program is a set of instructions that specify the operations, operands and the sequence by which processing has to occur.

Computer Instruction:-

→ A Computer Instruction is a binary code that specifies a sequence of microoperations for the computer.

→ Instruction codes together with data are stored in memory.

→ The Computer reads each instruction from memory and places it in a control register.

→ The control then interprets the binary code of the instruction and proceeds to execute by issuing a sequence of microoperation.

Instruction code

→ An instruction code is a group of bits that instructs the computer to perform a specific operation

Scanned by CamScanner

Eg:- ADD $\boxed{457}$ → address of the operand in memory

operation code (or opcode) — instruction

Operation Code (OPCODE):-

→ The operation code of an instruction is a group of bits that define such operations as add, subtract, multiply, shift and complement.

→ Unique binary code is assigned to every opcode is fixed during the designing of the computer.

→ The number of bits required for the operation code of an instruction depends on the total number of operations available in the computer.

Eg:- Consider there are 16 distinct operations then the size of opcode will be 4 bits ~~since~~ because 16 is written as 2^4 (4 bits for opcode).

Relation between computer operation and microoperation:-

→ An Computer operation is a part of an instruction stored in computer memory.

→ It is a binary code that tells the computer to perform a specific operation.

→ The control unit receives the instruction from memory and interprets the operation code bits.

→ It then issues a sequence of control signals to initiate microoperations.

Scanned by CamScanner

For every operation code, the control issues a sequence of microoperations needed for the hardware implementation of the specified operation.

→ So, it is sometimes called a macrooperation because it specifies a set of microoperations.

→ The operation part of the instruction code specifies the operation to be performed.

→ An instruction code not only specify the operation but also the registers or memory word where are the operands are to be found, where the result is to be stored.

→ Memory words can be specified in instruction codes by their address.

→ Processor registers can be specified by assigning to the instruction another binary code of k-bits that specifies one of 2^k registers.

Stored Program Organization

→ In this concept, we will look at how the instructions, operations, opcode, operands, data are stored in the memory

→ The simplest way to organize a computer is to have one processor register and an instruction code formats with two parts.

Scanned by CamScanner

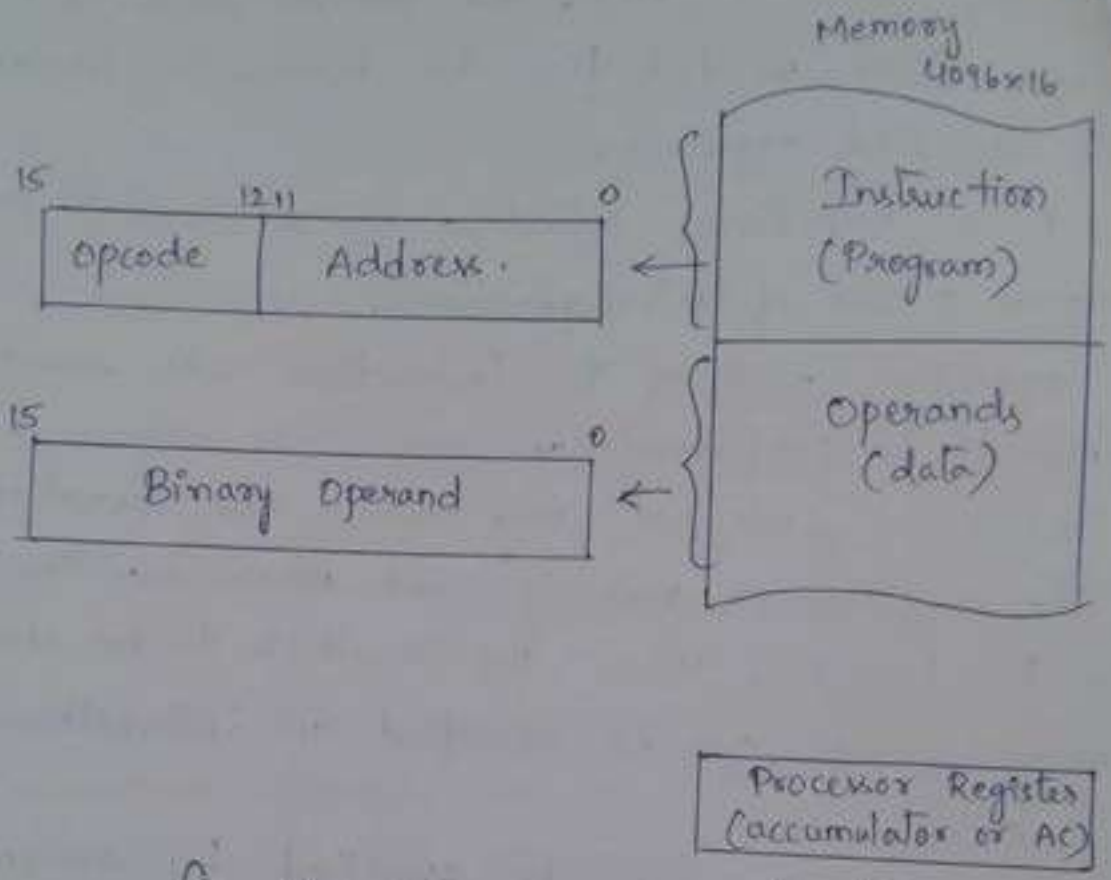


fig:- Stored Program organization.

→ In the above memory organization, instructions are stored in one section of memory and data in another

→ for a memory unit 4096 words we need only 12 bits to specify an address (since $2^{12} = 4096$), but each instruction code is stored in 16-bit memory word, 4 bits for operation code (opcode)

→ In The instruction code of the memory is divided into 2 parts.

→ The first part specifies the operation to be performed (4 bits) and second specifies an address.

Scanned by CamScanner

→ The memory address tells the control where to find an operand in memory.

→ This operand is read from memory and used as the data to be operated on together with the data stored in the processor register.

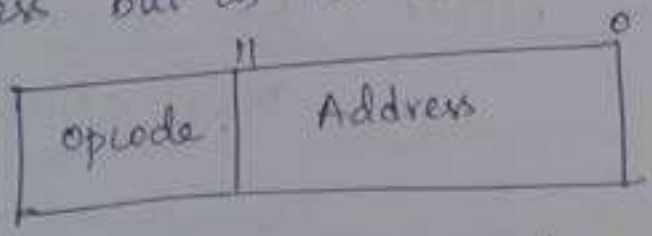
accumulator (AC) :-

It is also known as processor register, the operation is performed with the memory operand and the content of AC.

→ It accumulates the data or result or operands fetched from the memory.

Direct Address and InDirect Address :- (or Instruction Formats)

→ If we use address bits of an instruction code not as an address but as the actual operand



→ ~~If~~ * when second part of an instruction specifies an operand, the instruction is said to have an immediate operand.

→ Direct Address :-

when the second part specifies the address of an operand, the instruction is said to have a 'direct address'.

Scanned by CamScanner

Indirect Address

when the bits in the second part of the instruction designate an address of a memory word in which the address of the operand is found.

→ One bit of the instruction code can be used to distinguish between Direct and Indirect Address.

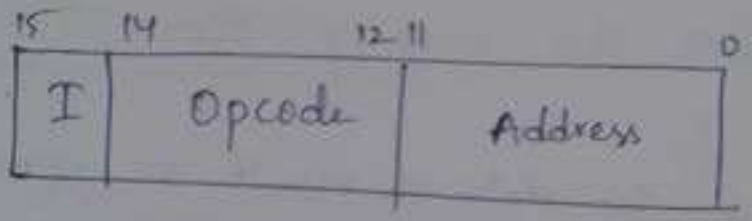
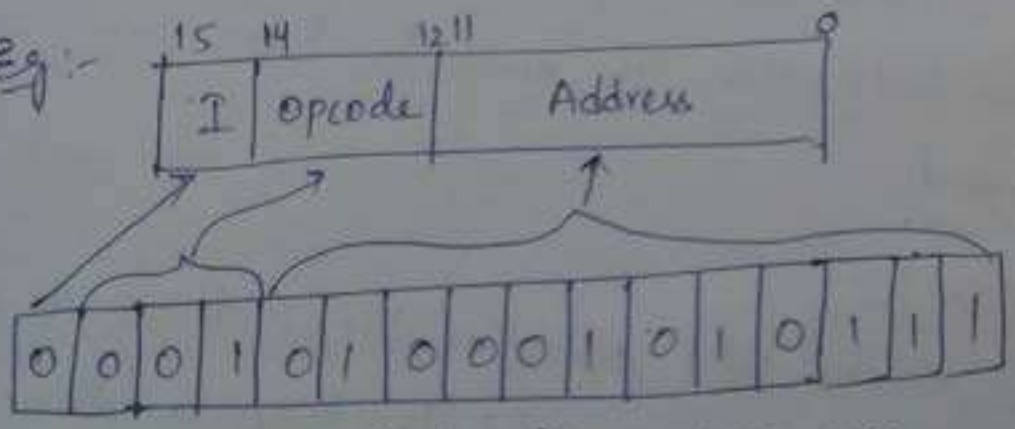


fig (a) Instruction format.

The above format consists of 3-bit operation code, a 12-bit address, I is an address mode bit.

address Mode bit (I)	type of address
0	Direct address
1	Indirect address

Eg:-



Add instruction - ADD 457.

Scanned by CamScanner

Direct Addressing & Indirect Addressing

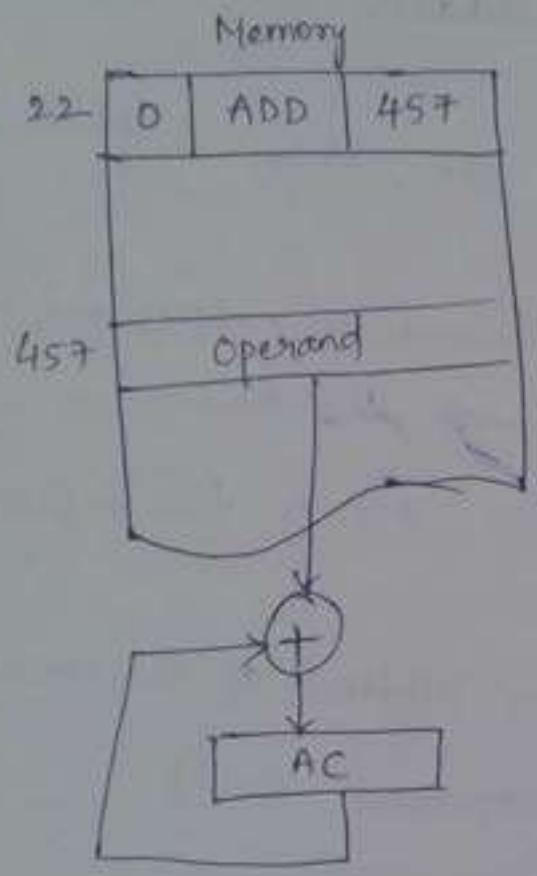


fig b:- Direct Address

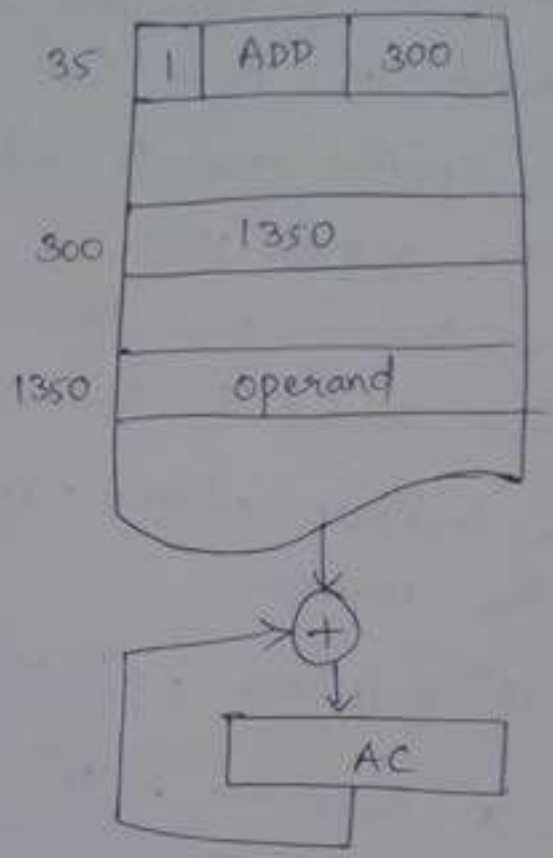


fig (c):- Indirect address.

A direct Address instruction is shown in the fig b. The instruction is placed in the memory location 22

→ The I bit is 0, so, it is a direct address instruction.

→ The opcode specifies ADD operation and the operands are available in operands section in the memory at address 457

→ This is added to the contents of AC.

A Indirect Address instruction is shown in the fig c. The instruction is placed in the memory location 35 with I bit as 1, so it is Indirect address Instruction.

Scanned by CamScanner

→ The opcode specifies the ADD operation and the address 300 specifies the address at which operands are there.

→ The control goes to address 300, where it finds another address 1350.

→ The control goes 1350 and founds the operand which is now added to the content of the AC.

→ The indirect address instruction needs two references to the memory

1st Reference - To read the address of the operand

2nd Reference - To read operand itself.

Computer Registers :-

→ Computer instructions are stored in consecutive memory locations and are executed sequentially one at a time.

→ The control reads an instruction from a specific address in memory and executes it. It then continues by reading the next instruction in sequence and executes it, and so on.

→ This type of instruction sequencing needs a counter to calculate the address of the next instruction after execution of the current instruction is completed.

Scanned by CamScanner

PC (Program Counter):-

→ The size of PC is 12 bits.
→ It holds the address of the next instruction to be executed.

AR (Address Register)

→ It is also of size 12 bits.
→ It holds the address of the memory i.e, it points to the memory.

IR (Instruction Register):-

→ The size of IR is 16 bits
→ It holds the instruction code.

TR (Temporary Register):-

→ It is of 16 bits
→ It holds the temporary data generated during the microoperations are carried out.

DR (Data Register)

→ It is of 16 bits
→ It holds the memory operand, the data which is fetched from memory is stored in DR.

AC (Accumulator)

→ It is also known as PROCESSOR REGISTER.
→ It is of size 16 bits
→ The result generated after microoperation is stored in AC.
→ The main processing of data is carried out on to this register.

OUTR (output Register)

→ It is of size 8 bits.
→ It holds output character, the character that we want to sent to output device.

Scanned by CamScanner

INPR :- (Input Register)

It holds the input character i.e, the data given from input device is stored in INPR.

Memory :-

→ The memory unit has a capacity of 4096 words and each word contains 16 bits.

→ 12 bits specify the address word in the instruction format where as 3 bits for the operation, and 1 bit to specify direct or indirect address.

Table :- List of Registers for the basic Computer

Register Symbol	Number of bits	Register Name	Function
DR	16	Data Register	Holds memory operand
AR	12	Address register	Holds address for memory
AC	16	Accumulator	Processor Register
IR	16	Instruction Register	Holds Instruction code
PC	12	Program Counter	Holds address of instruction
TR	16	Temporary Register	Holds temporary data.
INPR	8	Input Register	Holds input character
OUTR	8	output Register	Holds output character

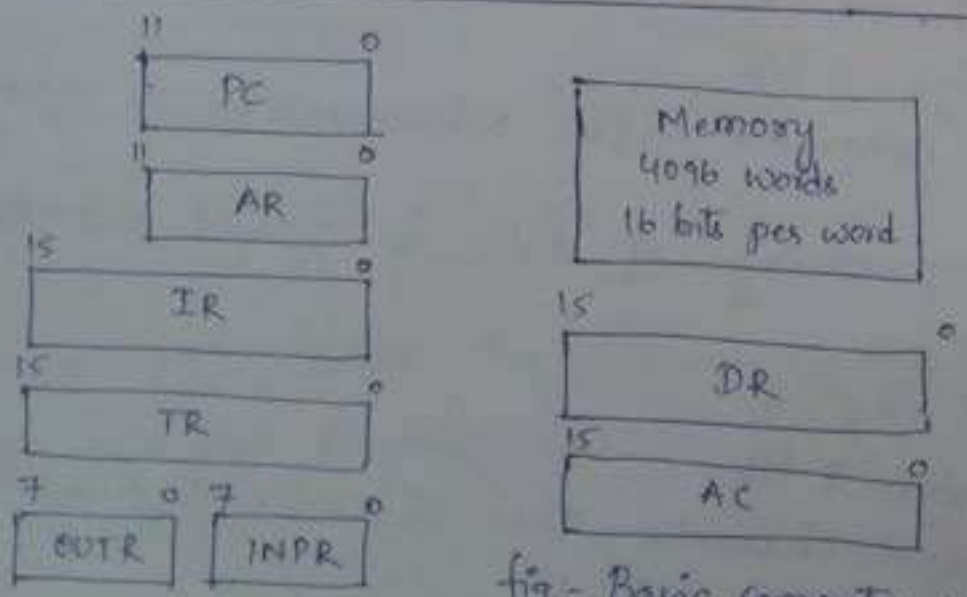


fig.- Basic computer registers & Memory

Scanned by CamScanner

Common Bus System

→ The outputs of seven registers and memory are connected to the common bus.

→ The specific output that is selected for the bus lines at any given time is determined by the selection variables S_0, S_1, S_2

Eg:- The number along the output of DR is 3. The 16-bit outputs of DR are placed on the bus lines when $S_2S_1S_0 = 011$ since this is the binary value of decimal 3.

→ The lines from the common bus are connected to the inputs of each register and the data inputs of the memory.

→ The particular register whose LD (load) input is enabled receives the data from the bus.

→ INR Increments the content of the register by 1

→ CLR, clear the content present in that register.

→ AR, PC, DR, AC, TR contains LD, INR & CLR and has input and output lines from the bus.

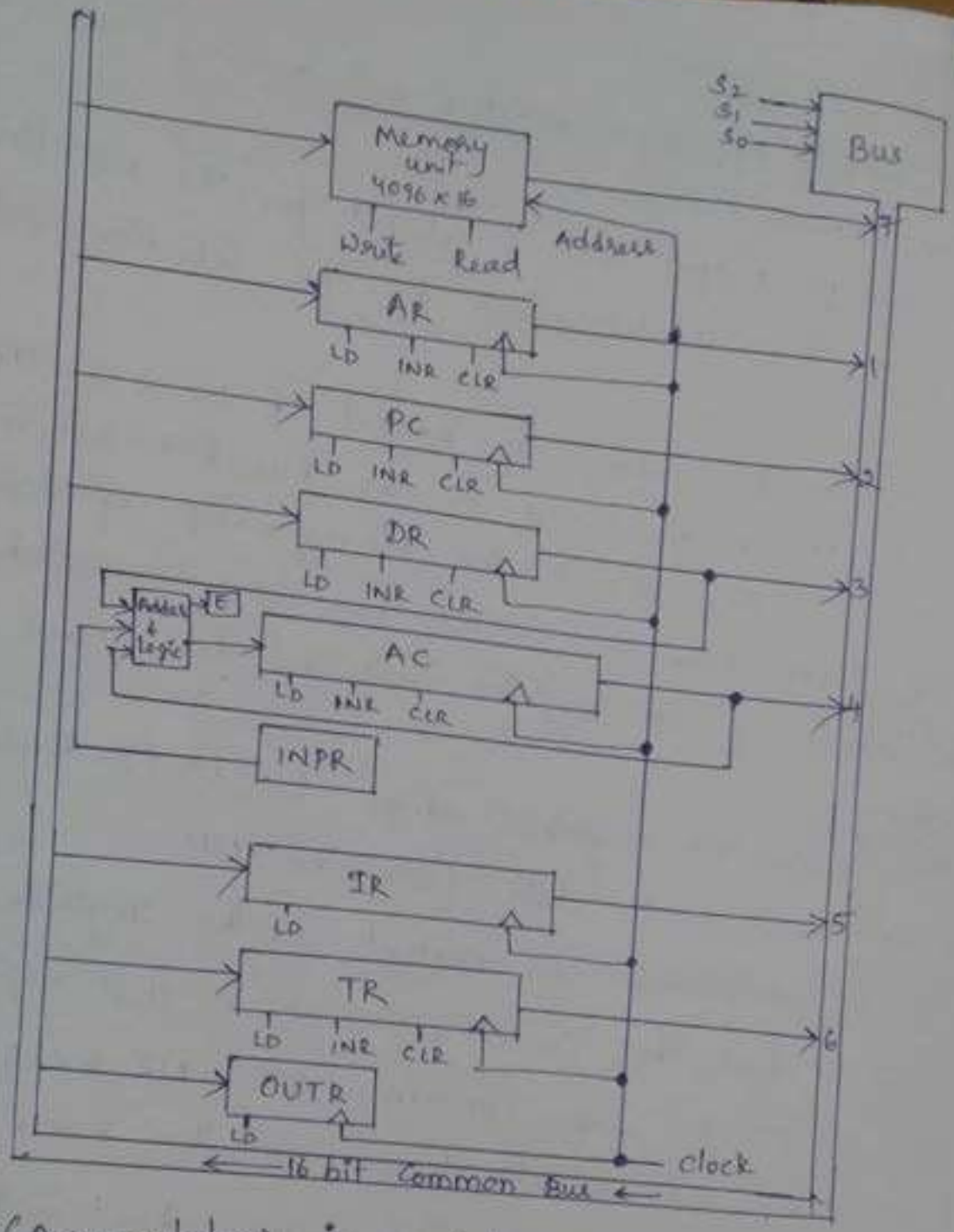
→ Memory unit has two operations Read and Write which are initiated when the selection inputs $S_2S_1S_0 = 111$.

→ Each & Every Register except INPR is connected to the clock.

→ INPR does not require any clock.

→ OUTR has an input line from 16-bit common bus.

Scanned by CamScanner



→ AC (Accumulator) is mainly used for processing does not have any input from bus but it is connected to output of Adder & logic unit.

→ Adder & Logic unit is nothing but a ALU which perform arithmetic operations, logic, shift operation gets three inputs, one from INPR Register another from DR (Data Register) and from output of AC itself.

Scanned by CamScanner

Four registers DR, AC, IR and TR have 16 bits each. Two registers, AR and PC have 12 bits each since they hold a memory address.

→ When the contents of AR or PC are applied to the 16-bit common bus, the four MSB are set to 0's.

→ INPR receives a character from an input device which is then transferred to AC.

→ OUTR receives a character from AC and delivers to output device.

→ Memory address is connected to AR. Therefore, AR must always be used to specify a memory address.

→ As AC gets three inputs, the one input which is received from the output of AC used to implement register microoperations such as complement AC and shift AC.

→ Another set of 16-bits input come from the data register DR. The inputs from DR and AC are used for Arithmetic and Logic microoperations such as add DR to AC (or) AND DR to AC the result is ~~translated to~~ transferred to AC and the end carry out of addition is used to stored in flip-flop E.

Note:- The content of any register can be applied on to the bus and an operation can be performed in adder and logic circuit during the same clock cycle.

Scanned by CamScanner

Eg :- The two microoperations, can be executed at the same time

$DR \leftarrow AC$ and $AC \leftarrow DR$.

This can be done by placing the content of AC on the bus (with $S_2 S_1 S_0 = 100$), enabling the LD input of DR transferring the content of DR through the adder and logic circuit into AC. and enabling the LD (load) input of AC, all during the same clock cycle.

Computer Instructions :-

→ The basic computer has three instruction formats they are.

(1) Memory Reference Instruction.

(2) Register - Reference Instruction.

(3) Input - Output Instruction.

→ Each format has 16 bits, the operation code (opcode) part of the instruction contains three bits and the remaining 13 bits depends on the operation code encountered.

Memory Reference Instruction :-

→ A memory Reference instruction uses 12 bits to specify an address and one to specify the addressing mode I. I is equal to 0 for direct address and to 1 for Indirect address.

→ The remaining 3 bits for opcode.

Scanned by CamScanner

Register Reference Instructions:-

A register reference Instruction are recognized by the operation code 111 with a 0 in the leftmost bit (bit 15) of the instruction.

→ A register-reference Instruction specify an operation on or a test of the AC register.

→ An operand from memory is not needed, therefore 12-bits are used to specify the operation

Input-Output Instruction:-

→ An input-output Instruction does not need a reference to memory and is recognized by the operation code 111 with a 1 in the left-most bit of the instruction.

→ The remaining 12 bits are used to specify the type of input-output operation or test performed.

fig:- Basic Computer Instruction formats.

15	14	12 11	0
I	Opcode	Address	

(opcode = 000 through 110)

(a) Memory-reference instruction

15	12 11	0
0 1 1 1	Register operation	

(opcode = 111, I = 0)

(b) Register-Reference instruction.

1 1 1 1	I/O operation

(opcode = 111, I = 1)

c) Input-output instruction

Scanned by CamScanner

The type of instruction is recognized by the computer control from the four bits in positions 12 through 15 of the instruction.

If the three opcode bits in positions 12 through 14 are not equal to 111, the instruction is a memory reference ~~memory~~ instruction type and the bit in position 15 is taken as the addressing mode I.

→ If the 3-bit opcode is equal to 111, control then inspects the bit in position 15. If this bit is 0 then instruction is a register-reference type. If the bit is 1, the instruction is an input-output type.

The symbol designation is a three-letter word
→ and represents an abbreviation intended for programmers
p and users. The hexadecimal code is equal to the
re equivalent hexadecimal number of binary code used for
er the instruction. by reducing the 16 bits of an instruction
N code to four digits.

→ A memory reference instruction has an address
Sy part of 12 bits denoted by 3 x's which stands for
m three hexadecimal digits. when I=0, the last four
t bits of an instruction have an hexadecimal digit
equivalent ~~to~~ from 0 to 6 when I=0, it is
equivalent from 8 to E.

Scanned by CamScanner

Instruction Set Completeness

A Computer should have a set of instructions so that the user can construct machine language programs to evaluate any function that is known to be Computable.

→ The set of instructions are said to be complete if the Computer includes a sufficient number of instructions in each of the following categories.

1) Arithmetic, logic and shift instructions.
2) Instructions for moving information to and from memory and processor register.
3) Program Control instructions together with instructions that check status conditions.
4) Input and output instructions.

Arithmetic, logic and shift instructions provide Computational capabilities for processing the type of data. Program control instructions such as branch instructions are used to change the sequence in which the program is executed.

Scanned by CamScanner

Timing and Control :-

The timing for all registers in the basic computer is controlled by a master clock generator. The clock pulses are applied to all flip-flops, registers in the system and in the control unit.

→ These clock pulses do not change the state of the register unless it is enabled by control signal.

→ The control signals are generated by the control unit and provide control inputs for the multiplexers in the common bus, control inputs in processor registers and microoperations for the accumulator.

Control organization can be done in two major types.

(i) hardwired control

(ii) Microprogrammed control.

Hardwired control :-

In this, control logic is implemented with gates, flip-flops, decoders and other digital circuits.

→ It can be optimized to produce a fast mode of operation.

→ If the design is modified or changed, it requires changes in the wiring among the various components.

Microprogrammed Control organization :- The control information is stored in control memory.

→ The control memory is programmed to initiate the sequence of microoperations.

Scanned by CamScanner

In the microprogrammed control, any required changes or modifications can be done by updating the microprogram in control memory.

Block Diagram of Control Unit :-

→ Control unit consists of Two decoders, a sequence counter, and a number of control logic gates.

→ An instruction from memory is read and placed in the IR (Instruction Register)

→ IR is of 16 bits, it contains I bit, the operation code, and bits 0 through 11.

→ The operation code of 12 to 14 bits are decoded with 3×8 decoder.

→ The 8 outputs are designated by the symbols D_0 to D_7.

→ Bit 15 is transferred to a flip-flop and bits 0 to 11 are applied to control logic gates.

→ The outputs of the counter are decoded into 16 timing signals T_0 through T_{15}.

→ The sequence counter SC can be incremented or cleared synchronously

→ Sometimes the counter is cleared to 0, causing the next active timing signal to be T_0.

Eg :- Consider the case where SC is incremented to provide timing signals T_0, T_1, T_2, T_3 and T_4 in sequence. At time T_4, SC is cleared to 0, if decoder output D_3 is active. It can be expressed as

$$D_3T_4 : SC \leftarrow 0.$$

Scanned by CamScanner

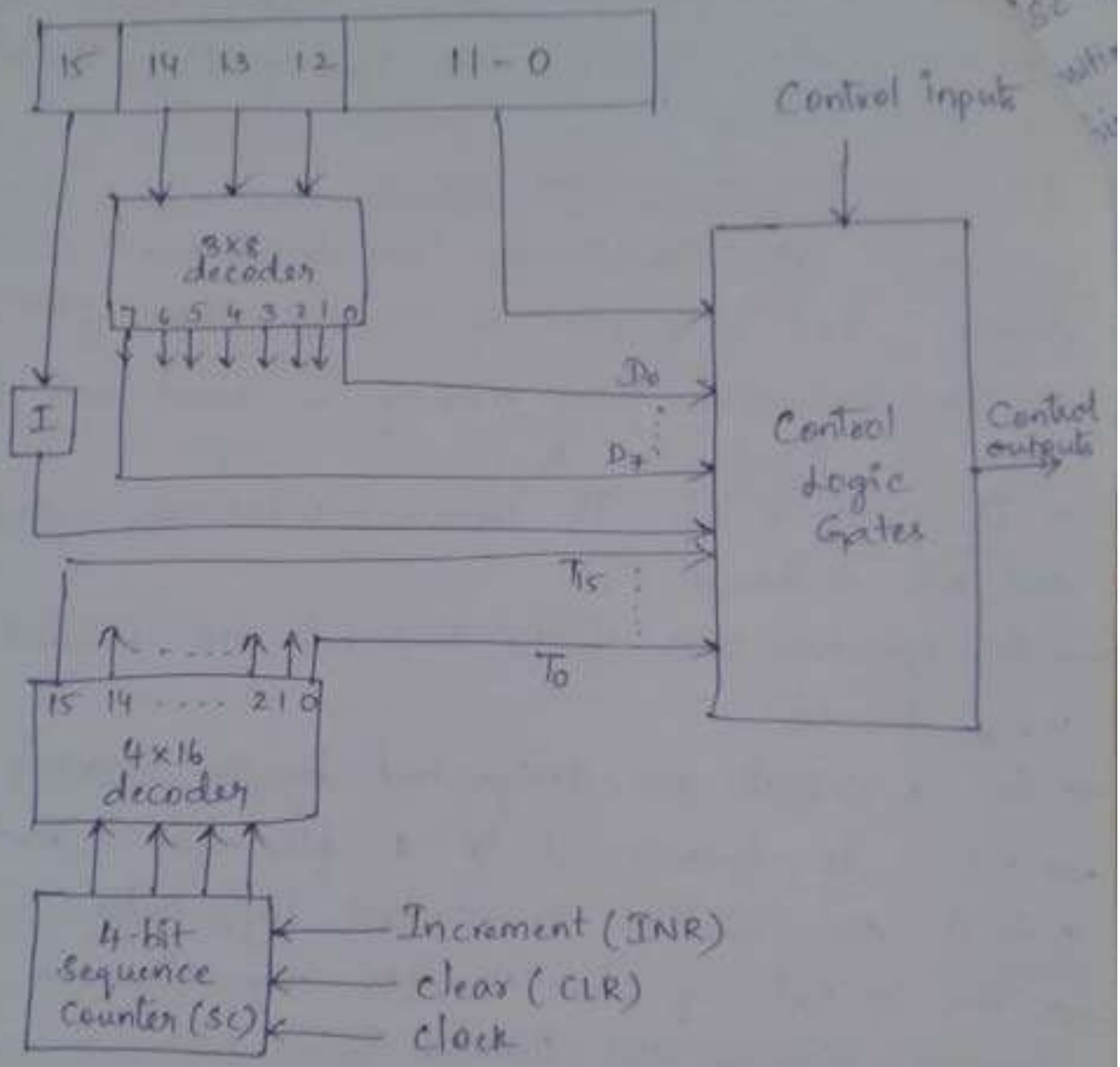


fig:- Control Unit of basic Computer.

The timing diagram shows the time relationship of the control signals.

→ The sequence counter SC responds to the positive transition of the clock.

→ If the first positive transition of the clock clears SC to 0 then it in turn activates the timing signal T_0 out of the decoder.

Scanned by CamScanner

SC is incremented with every positive clock transition, unless its CLR input is active.

→ This produces the sequence of timing signals T_0, T_1, T_2, T_3, T_4 and so on. If SC is not cleared, the timing signals will continue with T_5, T_6 upto T_{15} and back to T_0

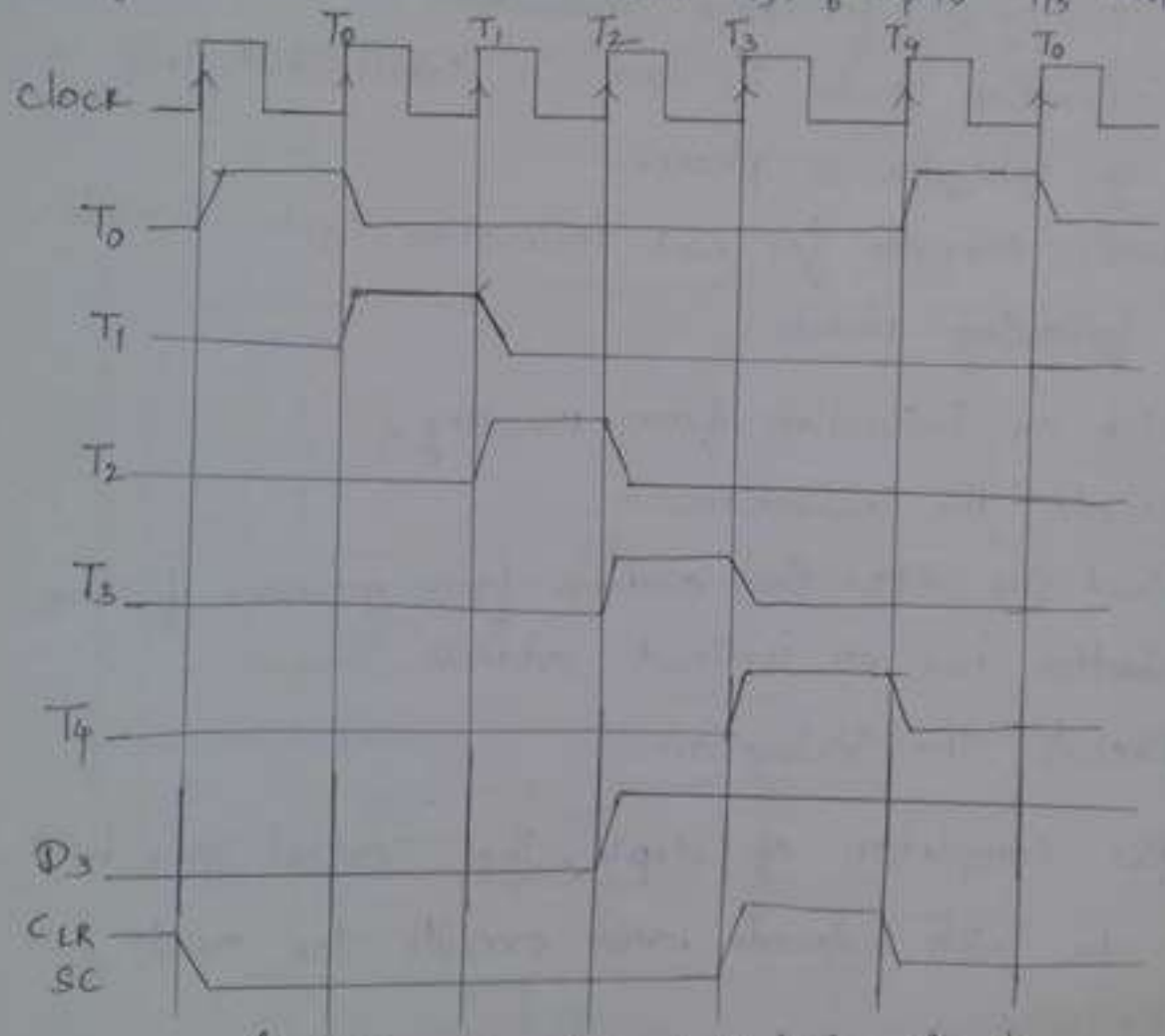


fig:- Example of control timing signals.

The last three waveforms show how SC is cleared when $D_3T_4 = 1$. output D_3 from the operation decoder becomes active at the end of timing signal T_2.

→ When timing signal T_4 becomes active, the output of control function D_3T_4 becomes active this signal is applied to the CLR input of SC. The counter becomes 0 and timing signal T_0 become active. instead of T_5.

Scanned by CamScanner

Instruction Cycle :-

→ A program consists of sequence of instructions which reside in the memory unit.

→ The program is executed in the computer by going through a cycle for each instruction.

→ Each instruction cycle in turn is subdivided into a sequence of subcycles or phases.

→ The basic computer for each instruction cycle consists of the following phases:

1. Fetch an instruction from memory.
2. Decode the instruction.
3. Read the effective address from memory if the instruction has an indirect address.
4. Execute the instruction.

After completion of step 4, the control goes back to step 1 to fetch, decode and execute the next instruction.

Fetch and decode :-

→ Initially the program counter is loaded with the address of the first instruction in the program

→ The sequence counter SC is cleared to 0, providing a decoded timing signal T_0 and then increment the SC by one, so that the timing signals go through a sequence T_0, T_1, T_2, and so on.

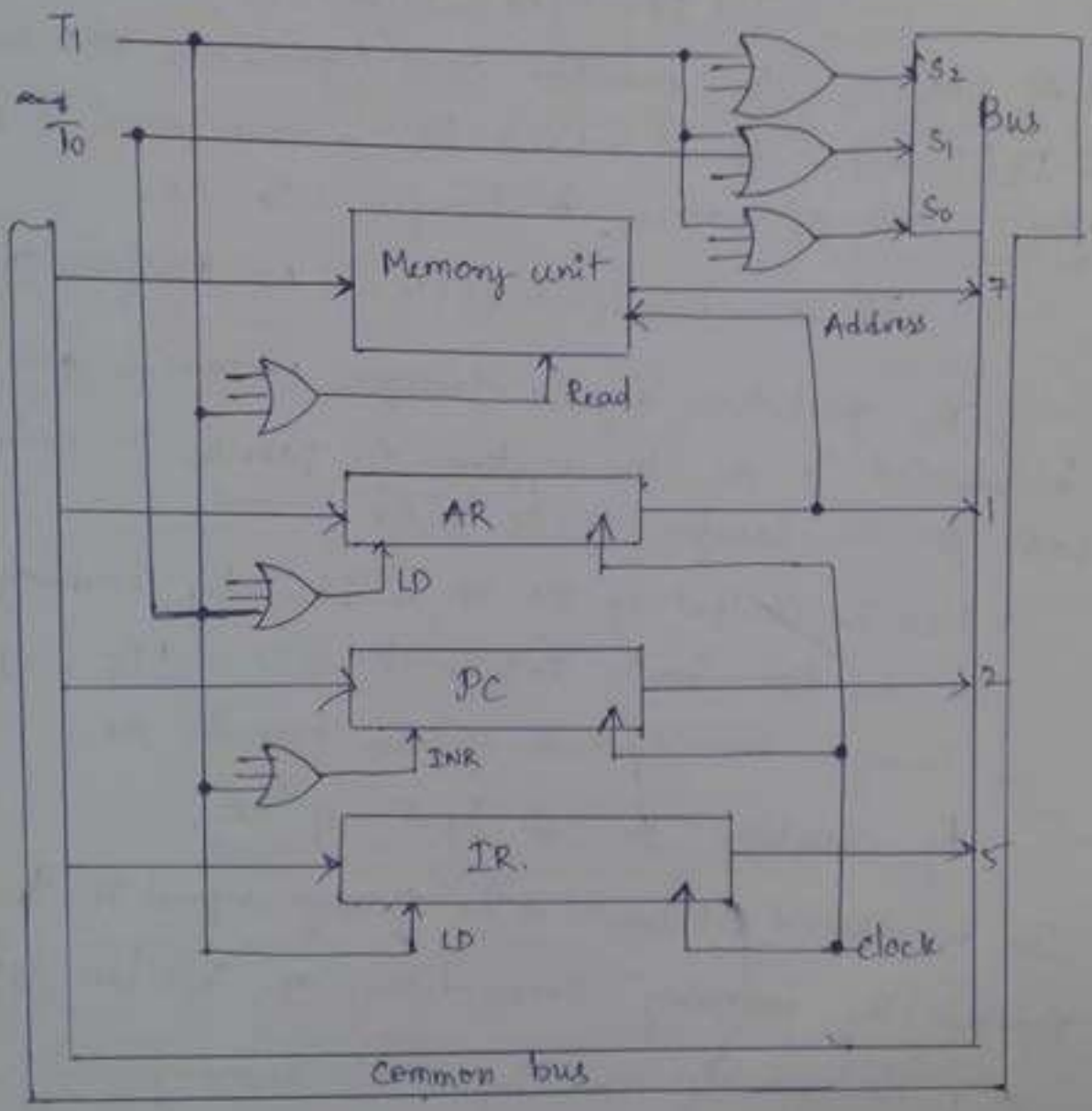
Scanned by CamScanner

Consider the following microoperations for the fetch and decode phases of register transfer statement.

T_0 : $AR \leftarrow PC$

T_1 : $IR \leftarrow M[AR]$, $PC \leftarrow PC+1$

T_2 : $D_0, \ldots, D_7 \leftarrow$ Decode $IR(12-14)$, $AR \leftarrow IR(0-11)$, $I \leftarrow IR(15)$



Since AR is connected to the address inputs of memory, it is necessary to transfer the address present in PC to AR during clock transition associated with timing signal T_0

T_0 : $AR \leftarrow PC$.

Scanned by CamScanner

→ The instruction read from memory is then placed in the IR register with clock transition associated with time signal T_1. At the same time PC is incremented by one to prepare it for the address of the next instruction in the program.

$$T_1 : IR \leftarrow M[AR],\ PC \leftarrow PC + 1$$

→ At time T_2, the operation code (opcode) is decoded IR (12-14), the indirect bit is transferred to flip flop I, and the address is transferred to AR.

$$T_2 : D_3 \leftarrow \text{Decode } IR(12\text{-}14),\ AR \leftarrow IR(0\text{-}11),\ I \leftarrow IR(15)$$

The first two register transfer statements are implemented in the bus system to provide the data path for the transfer of PC to AR.

1. Place the content of PC on to bus by enabling the selection inputs $S_2 S_1 S_0$ to 010. (PC is 2)
2. Transfer the content of the bus to AR by enabling the LD input of AR.

In the second statement at the timing signal T_1 to provide the following connections in the bus system.

1. Enable the read input of memory.
2. place the content of memory onto the bus by making $S_2 S_1 S_0 = 111$.
3. Transfer the content of the bus to IR by enabling the LD input of IR
4. Increment PC by enabling INR input of PC.

Scanned by CamScanner

Determine the type of Instruction :-

The timing signal that is active after the decoding is T_3. During time T_3, the control unit determines the type of instruction that was just read from memory. we know that they are 3 type of instructions

(1) Memory-reference type Instruction
(2) Register-reference type Instruction
(3) I/O - type Instruction.

The below flowchart presents an initial configuration for the instruction cycle and shows how the control determines the instruction type after the decoding

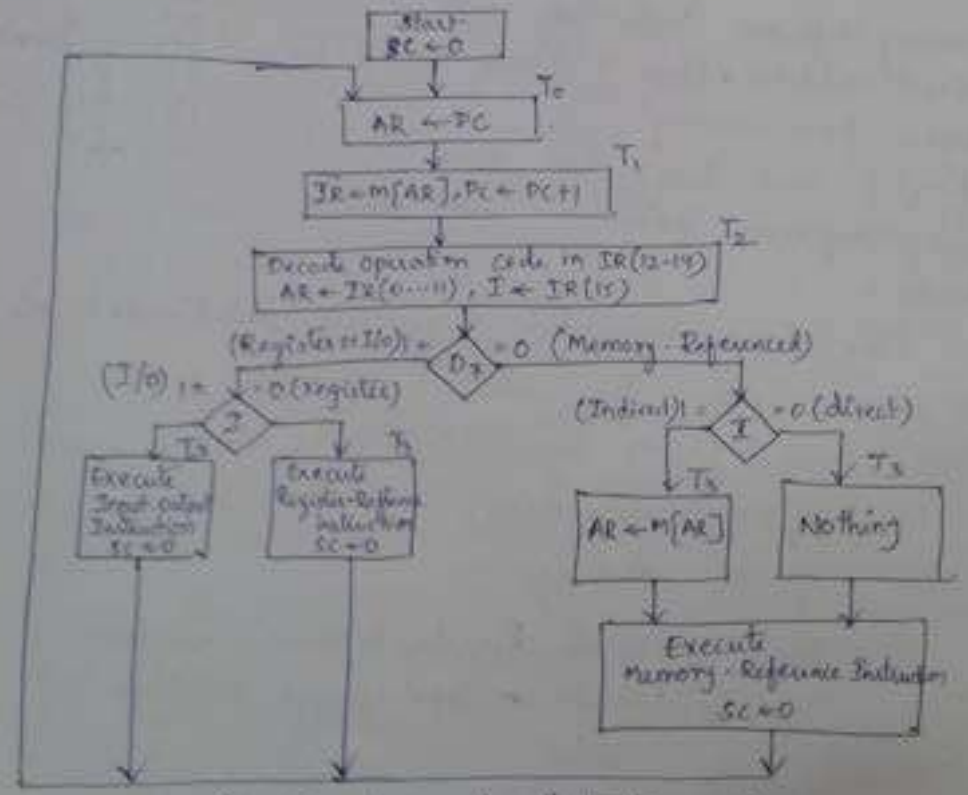


fig: flow chart for instruction cycle.

Scanned by CamScanner

→ Decode output D_7 is equal to 1 if the operand is equal to binary 111

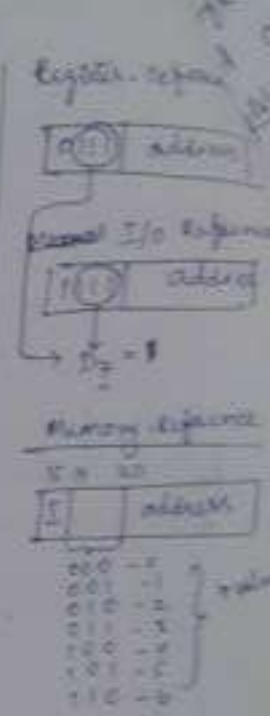
→ If $D_7 = 1$, the instruction may be register reference or I/O type.

→ If $D_7 = 0$, the operation code may exists between seven values 000 to 110 specifying a memory-reference instruction

→ Now control inspects the value of the first bit which is available in flip-flop I

→ If $D_7 = 0$ and $I = 1$ then it is a memory reference Instruction with indirect address, then it reads the effective address from memory.

→ If $D_7 = 0$ and $I = 0$ then it is a memory reference instruction with direct address.



The three instruction types are subdivided into four separate paths. The selected operation is activated with timing signal T_3

$D_7' I T_3$: $AR \leftarrow M[AR]$

$D_7' I' T_3$: Nothing

$D_7 I' T_3$: Execute Register Reference instruction

$D_7 I' T_3$: Execute an Input-Output instruction

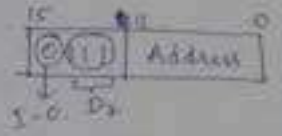
Scanned by CamScanner

After the instruction is executed, SC is cleared to 0 and control returns to the fetch phase with $T_0 = 1$.

1-47

Register Reference Instructions :-

Register Reference Instructions are recognized by the control when $D_7 = 1$ and $I = 0$.



→ These instructions use bits 0 to 11 of instruction code to specify the one of 12 instructions.

→ Total 12 bits are available in IR(0-11). they are transferred to AR during time T_2

→ After transferring the address to AR, the respective instruction is executed at T_3 timing variable.

→ Each Control function needs a boolean relation, so $D_7 I' T_3$ is designated as r.

→ The control function is distinguished by one of the bits in IR(0-11) so it can be denoted by $r B_i$ (i refers to bit i of IR)

Eg:- The instruction CLA has an hexadecimal code 7800 whose binary equivalent is 0111 1000 0000 0000 (7 8 0 0). The first bit is 0 which is equivalent to I' and next three bits (111) refers to opcode and Bit 11 is 1 and is recognized by B_{11}. so the control function that initiates the microoperation for this instruction is $D_7 I' T_3 B_{11} = r B_{11}$.

Scanned by CamScanner

Table: Execution of Register Reference Instructions

$D_7 I' T_3 = r$ (common to all register reference instructions)

$IR(i) = B_i$ [bit in IR(0-11) that specifies the operation].

	r:	$SC \leftarrow 0$	clear SC
CLA	rB_{11}:	$AC \leftarrow 0$	clear AC
CLE	rB_{10}:	$E \leftarrow 0$	clear E
CMA	rB_9:	$AC \leftarrow \overline{AC}$	Complement AC
CME	rB_8:	$E \leftarrow \overline{E}$	Complement E
CIR	rB_7:	$AC \leftarrow shr\ AC, AC(15) \leftarrow E, E \leftarrow AC(0)$	circulate right
CIL	rB_6:	$AC \leftarrow shl\ AC, AC(0) \leftarrow E, E \leftarrow AC(15)$	circulate left
INC	rB_5:	$AC^* \leftarrow AC + 1$	Increment AC
SPA	rB_4:	if $(AC(15) = 0)$ then $(PC \leftarrow PC + 1)$	skip if positive
SNA	rB_3:	if $(AC(15) = 1)$ then $(PC \leftarrow PC + 1)$	skip if negative
SZA	rB_2:	if $(AC = 0)$ then $(PC \leftarrow PC + 1)$	skip if AC zero
SZE	rB_1:	if $(E = 0)$ then $(PC \leftarrow PC + 1)$	skip if E zero
HLT	rB_0:	$S \leftarrow 0$ (S is start-stop flip flop)	Halt computer

The first seven instructions perform clear, complement circular shift and increment microoperations on AC or E. The next four instructions skip to the next instruction if the condition is satisfied. The HLT instruction clears a start-stop flip flop S and stops the sequence counter from counting, to restore we have to set the flip flop manually.

Scanned by CamScanner

Memory Reference Instructions :-

The decode output D_i for $i = 0,1,2,3,4,5$ & 6 from the operation decoder that belongs to each instruction is included in the table. The effective address of the instruction is in the address register AR and was placed there during timing signal T_2 when $I=0$ or during timing signal T_3 when $I=1$.

→ The execution of the memory-reference instructions start with timing signal T_4.

→ The symbolic description of each instruction is specified in the table in terms of register transfer notation.

Symbol	Operation decoder	Symbolic Description
AND	D_0	$AC \leftarrow AC \wedge M[AR]$
ADD	D_1	$AC \leftarrow AC + M[AR], E \leftarrow C_{out}$
LDA	D_2	$AC \leftarrow M[AR]$
STA	D_3	$M[AR] \leftarrow AC$
BUN	D_4	$PC \leftarrow AR$
BSA	D_5	$M[AR] \leftarrow PC, PC \leftarrow AR+1$
ISZ	D_6	$M[AR] \leftarrow M[AR]+1$ If $M[AR]+1 = 0$ then $PC \leftarrow PC+1$

Table :- Memory-Reference Instructions.

Note : ① The actual execution of the instruction in the bus system will require a sequence of microoperations.

② This is because data stored in memory cannot be processed directly.

③ The data must be read from memory to a register where they can be operated on with logic circuits.

Scanned by CamScanner

T.I.:

The operation of each instruction and listing the control functions and microoperations needed for their execution is explained below :-

AND to AC

This instruction performs the AND logic operation on pair of bits in AC and the memory word specified by the effective address. The result is transferred to AC.

→ The microoperations that execute the instruction are

D_0T_4 : $DR \leftarrow M[AR]$

D_0T_5 : $AC \leftarrow AC \wedge DR$, $SC \leftarrow 0$.

The control function for this instruction uses the decoder D_0 since the output of the decoder is active AND operation is executed because the binary code value is '000'. Two timing signals are needed to execute the instruction. T_4 transfers the operand from memory to DR and T_5 transfers the result of AC AND DR to AC.

ADD to AC :-

This instruction adds the content of memory word specified by the effective address to the value of AC. The sum is transferred to AC and the carry output C_{out} is transferred to E (flip flop) of (Extended accumulator). The microoperations include

D_1T_4 : $DR \leftarrow M[AR]$

D_1T_5 : $AC \leftarrow AC + DR$, $E \leftarrow C_{out}$, $SC \leftarrow 0$.

Scanned by CamScanner

The two timing signals T_4 and T_5 are used again for the operation of decoder D_1.

LDA : load to AC :-

This instruction transfers the memory word specified by the effective address to AC.

The microoperations needed to execute this instruction are

$$D_2 T_4 : DR \leftarrow M[AR]$$
$$D_2 T_5 : AC \leftarrow DR , SC \leftarrow 0$$

In the common bus system, there is no direct path from bus into AC. The adder and logic circuit receive information from DR which can be transferred into AC. So, it is necessary to read from memory word into DR first and then transfer the content of DR into AC.

STA : store AC :-

This instruction stores the content of AC into the memory word specified by the effective address. output of AC is placed on to the bus and data input of memory is connected to bus, we can execute this instruction with one microoperation

$$D_3 T_4 : M[AR] \leftarrow AC , SC \leftarrow 0.$$

BUN : Branch Unconditionally :-

This instruction transfers the program to the instruction specified by the effective address.

→ Note :- PC (Program Counter Register) holds the address of the instruction to be read from memory in the next instruction cycle.

Scanned by CamScanner

→ PC is incremented at time T_1 to prepare it for address of the next instruction in the program sequence.

→ But the BUN instruction allows the programmer to specify the instruction out of sequence order of instruction in the memory so that the program branches (or JUMPS) unconditionally.

The instruction is executed with one microoperation.

$$D_4T_4 : PC \leftarrow AR, SC \leftarrow 0$$

Effective address from AR is transferred to Commonbus to PC.

BSA : Branch and Save Return Address :-

This instruction is useful for branching to a portion of the program called a Subroutine or procedure.

→ When executed, the BSA instruction stores the address of the next instruction in sequence into a memory location specified by the effective address.

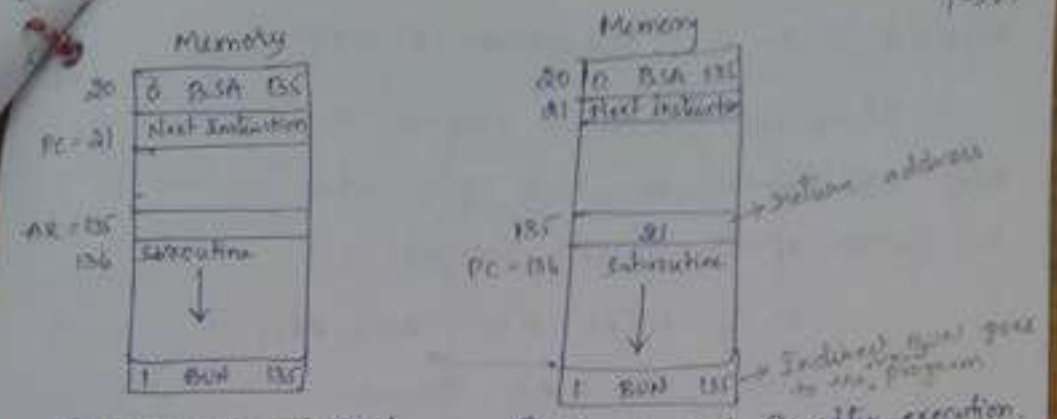
→ The effective address + 1 is then transferred to PC to serve as the address of the next instruction in the subroutine.

$$M[AR] \leftarrow PC, PC \leftarrow AR + 1$$

Note *.**

The BSA instruction is assumed to be memory at address 20. The I bit is 0 and address part of instruction is referred to address 135. After the fetch and decode phases, PC contains 21, which is the address of the next instruction. (which is the return address)

Scanned by CamScanner



(a) Memory, PC and AR at Time T_4

(b) Memory and PC after execution.

fig:- Example of BSA instruction execution.

AR holds the address 135

$M[135] \leftarrow 21$, $PC \leftarrow 135+1 = 136$

The result of above operation is shown in part (b) of the figure.

Return address (21) is stored in the memory location (135) and the control continues with the subroutine program starting from 136.

→ The return to the original program is accomplished by means of an indirect BUN indicated at the end of the subroutine.

→ when this instruction is executed, control goes to the indirect phase where it reads the effective address at location 135 where it finds previously saved address 21.

The BSA instruction performs the function usually referred to as a 'SUB ROUTINE CALL'. The indirect BUN instruction at the end of the subroutine performs the function

referred to as a 'SUBROUTINE RETURN'.

It is not possible to perform the operation of BSA instruction in one clock cycle when we use the bus system of the basic computer.

D_5T_4 : $M[AR] \leftarrow PC$, $AR \leftarrow AR+1$

D_5T_5 : $PC \leftarrow AR$, $SC \leftarrow 0$

ISZ : Increment and skip if zero

$M[AR] \leftarrow M[AR]+1$

if $M[AR]+1 = 0$ then $PC \leftarrow PC+1$

This instruction increments the word specified by the effective address and if incremented value is equal to 0 then PC is incremented by 1.

→ The programmer usually stores negative number in the memory word. If this negative number is repeatedly incremented by 1, it eventually reaches to '0'.

→ So, PC is incremented by 1 in order to skip that next instruction in the program.

Note : It is not possible increment a word inside the memory

→ So, the address is transfer to DR and then DR is incremented by 1 and store back into memory.

Sequence of microoperations

D_6T_4 : $DR \leftarrow M[AR]$

D_6T_5 : $DR \leftarrow DR+1$

D_6T_6 : $M[AR] \leftarrow DR$, if $(DR=0)$ then $(PC \leftarrow PC+1)$, $SC \leftarrow 0$

Scanned by CamScanner

Control flowchart

The flowchart showing all microoperations for the execution of the seven memory-reference instructions. The control functions are indicated on the top of the box. Microoperations are performed during time T_4, T_5 or T_6 depend upon the opcode value.

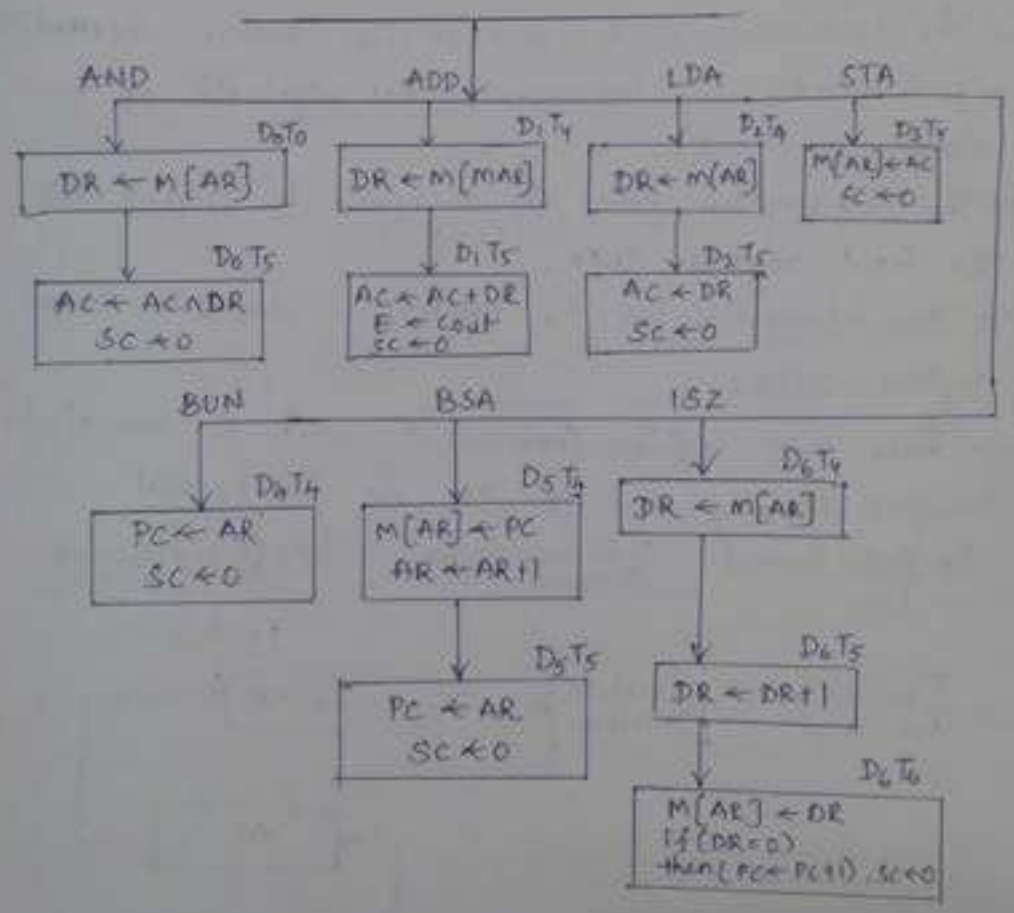


fig:- FLOW CHART for MEMORY-REFERENCE INSTRUCTION.

Scanned by CamScanner

Input - Output and Interrupt

→ A computer communicates with the external environment inorder to get instructions and data stor to memory from input device and display the results to the output devices.

Input-output Configuration

→ The terminals sends and receives serial information

→ Each quantity of information has eight bits of an alphanumeric code

→ The information from the keyboard is shifted into the input register INPR.

→ The information for the printer is stored in output register OUTR.

→ These two register communicate with a communication interface serially and A with AC in parallel

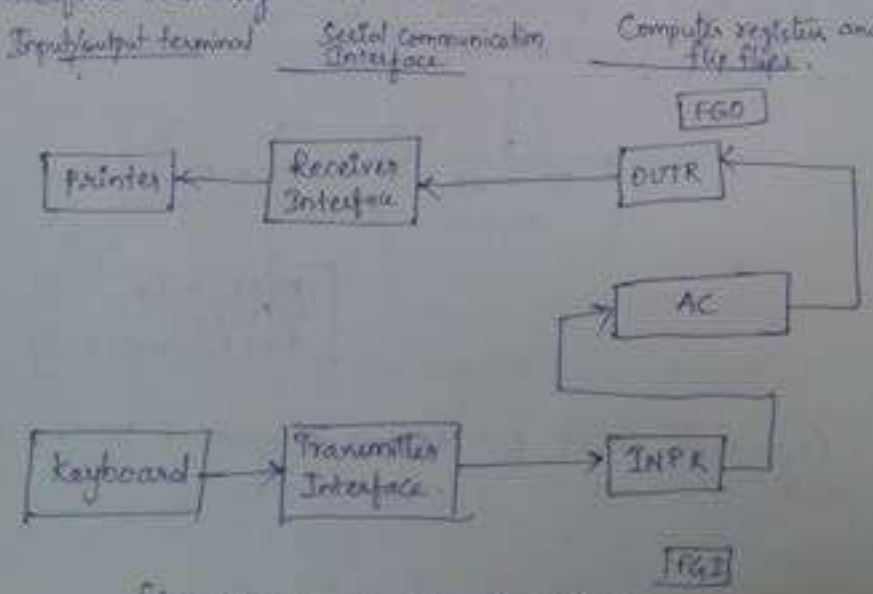


fig: Input-output Configuration.

Scanned by CamScanner

Transmitter Interface :-

The transmitter Interface receives serial information from keyboard and transmits to INPR.

Receiver Interface :-

The receiver interface receives information from OUTR and sends it to the printer serially.

INPR and FGI

The input register INPR consists of eight bits and holds an alphanumeric input information

→ FGI is a 1-bit input flag used for control flip-flop.

→ The flag bit is set to 1, the new information is available in the input device and cleared to 0 when the information is accepted by the computer.

process of information is as follows :-

1) Initially FGI is cleared to 0.
2) when we struck a keyboard, an 8 bit alphanumeric code is shifted to INPR and input flag FGI is set to 1. During FGI = 1 state the information in INPR is not changed.
3) computer check flag bit if it is 1, the information is transferred to AC and FGI is cleared to '0'.

Scanned by CamScanner

OUTR and FGO :-

→ The output register (OUTR) works similarly but the flow of information is reversed.

→ The output flag (FGO) is set to 1

→ The Computer checks the flag bit if it is 1, the information from AC is transferred in parallel to OUTR and FGO is cleared to 0.

→ After printing the corresponding character on the Screen again it sets FGO to 1.

→ The Computer does not load into OUTR when FGO is 0 because the Condition is indicating that it is in the process of printing the character.

Input-Output Instructions

Input-output instructions are needed for transferring information to and from AC register, for checking flip flops and for controlling interrupt facility.

→ I/O instructions have an opcode 1111 and are recognized by the Control when $D_7 = 1$ and $I = 1$

→ The Remaining bits of the instruction specify the particular operation.

→ These instructions The Control functions and microoperations are listed in the below table.

I/O Instruction.

Scanned by CamScanner

$D_7 I T_3 = p$ (Common to all I/O instructions)
$IR(i) = B_i$ [bit in IR (6-11) that specifies the instruction]

	p:	$SC \leftarrow 0$	clear SC
INP	pB_6:	$AC(0-7) \leftarrow INPR, FGI \leftarrow 0$	Input character
OUT	pB_7:	$OUTR \leftarrow AC(0-7), FGO \leftarrow 0$	output character
SKI	pB_8:	If $(FGI = 1)$ then $(PC \leftarrow PC+1)$	skip on input flag
SKO	pB_9:	If $(FGO = 1)$ then $(PC \leftarrow PC+1)$	skip on o/p flag
ION	pB_{10}:	$IEN \leftarrow 1$	Interrupt enable
IOF	pB_{11}:	$IEN \leftarrow 0$	Interrupt disable.

Table :- Input-output Instructions.

→ These Instructions are executed with timing signal T_3.
→ with a boolean relation for control function would be $D_7 I T_3$. which is indicated with 'p'.
→ The control function is distinguish by one of the bit of IR (6-11) by the symbol B_i (i refers to bit in IR)

Control functions can be denoted by

pB_i for $i = 6$ through 11

<u>INP</u> :- INP Instruction transfers the input information from INPR to AC and clears the input flag to 0

<u>OUT</u> :- OUT instruction transfers the 8 LSB of AC into OUTR and clear the output flag to 0.

→ The next two instructions in the above table checks the flags and skip to the next instruction if FGI or FGO is equal to 1

<u>Note</u> :- The instruction that is skipped will be a branch instruction.

→ the last two instructions set and clear the interrupt enable flip flop IEN

Program Interrupt

What is an Interrupt?

A signal that gets the attention of the CPU and is usually generated when I/O is required.

Eg:- Hardware interrupts are generated when a key is pressed or when the mouse is moved.

The process of communication in Input-output organization is referred to as programmed control transfer.

→ The computer keeps checking the flag bit for an information transfer from/to the I/O devices.

→ The computer is wasting time while checking the flag bit every time instead of doing some useful processing task.

→ The above disadvantage in the programmed controlled procedure can be overcomed by the "external device informing the computer when it is ready for the transfer".

→ This type of transfer uses the Interrupt facility.

→ While the computer is running a program, it does not check the flags. However, when a flag is set, the computer is interrupted from proceeding the current program and is informed with that the flag has been set.

→ The computer deviates from the current program to take care of the Input-output transfer.

Scanned by CamScanner

After the transfer, the control returns back to the program to continue.

→ The interrupt enable flip-flop IEN can be set and cleared with two instructions.

→ when IEN = 0, the flags cannot interrupt computer.

→ when IEN = 1, the computer can be interrupted.

→ with the help of these two instructions, it provides the programmer to make decision whether or not to use the interrupt facility.

How the interrupt is handled by the computer?

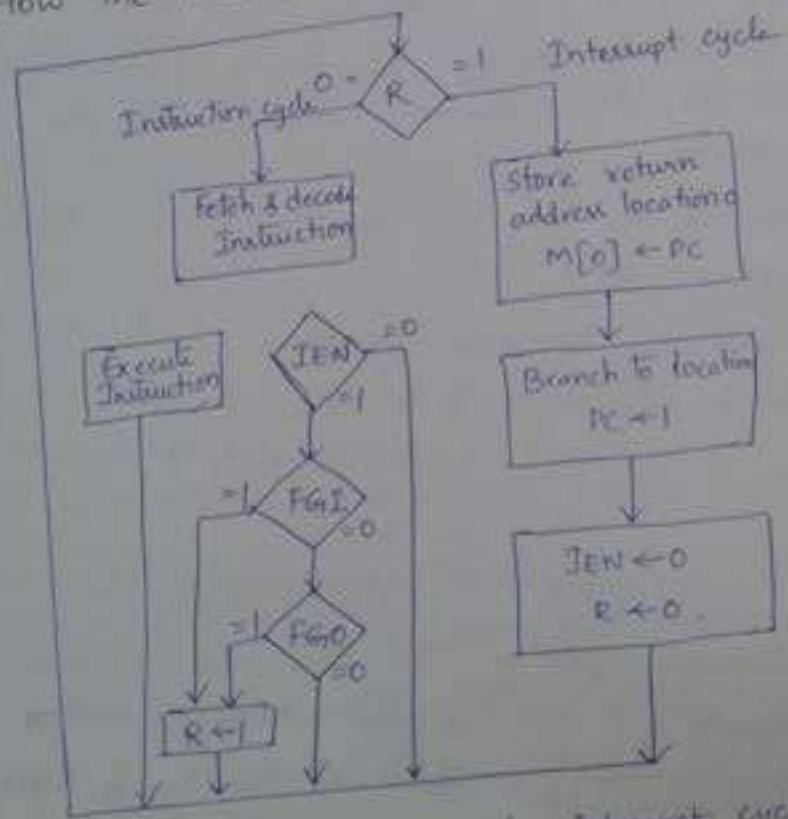


fig:- flowchart for interrupt cycle

→ The interrupt is handled by the computer can be explained by means of the flowchart.

→ An interrupt flip-flop R is included in the computer.

→ When R=0, the computer goes through an instruction cycle

→ Again during the execution phase of instruction cycle, IEN value is checked by the control.

→ If IEN=0, it indicates that the programmer does not want to use the interrupt, then the control continues with the next instruction cycle.

→ If 'IEN=1', control will now check the flag bits. If flag bits are 0, it indicates that neither the input or output registers are ready for transfer of information. Again control continues with the next instruction cycle.

→ If either of the flag bits are set to 1 if IEN=1, flip flop R is set to 1.

→ At the end of the execution phase, again the control checks the value of R, if it is equal to 1, it goes to an interrupt cycle instead of instruction cycle.

<u>Interrupt cycle phase</u>:- (in flowchart)

→ The interrupt cycle is the hardware implementation of a branch and save return address operation.

→ The return address available in the PC is stored

Scanned by CamScanner

specific location where it is found later by the program returns to the instruction at which it was interrupted.

→ The location at which we store PC value may be registers of processor or stack or specific memory location.

→ Here in the flowchart, we choose the memory location at address 0 as the place for storing the return address

→ Control then inserts address 1 into PC and clears IEN and R so that no more interrupts can occur.

<u>Interrupt cycle</u>

→ Now, we list the register transfer statements for the interrupt cycle

→ The interrupt cycle is initiated after the last execution phase if the interrupt flip flop R = 1.

→ flip flop is set to 1 if IEN = 1 and either FGI or FGO = 1

→ The condition for setting flip flop R to 1 can be expressed in the following register transfer statement:

$$T_0' T_1' T_2' (IEN)(FGI + FGO) : R \leftarrow 1$$

+ → indicates b/w the FGI & FGO in the control function is logical OR operation.

which is ANDed with IEN and $T_0' T_1' T_2'$.

Scanned by CamScanner

→ we modify the fetch and decode phases of the instruction by AND the three timing signals (T_0, T_1, T_2) with R', so that fetch and decode is recognized by the control functions $R'T_0$, $R'T_1$, and $R'T_2$

→ ~~After~~ The reason is after the instruction is executed sc is cleared to 0, the control will go with fetch phase if R=0 otherwise, if R=1 it go through interrupt cycle.

→ The interrupt cycle stores the return address (available in PC) into memory location 0, branches to memory location 1 and clears IER, R and SC to 0.

→ The following sequence of microoperations describes about the process above.

RT_0 : $AR \leftarrow 0$, $TR \leftarrow PC$

RT_1 : $M[AR] \leftarrow TR$, $PC \leftarrow 0$

RT_2 : $PC \leftarrow PC+1$, $IEN \leftarrow 0$, $R \leftarrow 0$, $SC \leftarrow 0$

Scanned by CamScanner

Unit - II

MICROPROGRAMMED CONTROL

CONTROL MEMORY :-

→ The major functional parts in a digital computer are CPU, Memory and Input-output.

→ The main digital Hardware functional units of CPU are Control unit, arithmetic and logic unit (ALU) and registers.

→ The function of Control unit is to initiate sequence of microoperations and directs the operation of the processor.

→ Two methods of implementing control unit are;

(i) Hardwired Control

(ii) Microprogrammed Control.

Hardwired Control:-

→ If the control signals are generated by hardware using conventional logic design techniques, then it is said to be hardwired.

→ It involves the use of fixed instructions, fixed logic blocks of and/or arrays, decoders, encoders multiplexers etc.

Key characteristics of hardwired control

→ High speed operations

→ Expensive

→ Relatively Complex

→ No flexibility of adding new instructions.

Scanned by CamScanner

Examples :-

Intel 8085, Motorola 6802, zilog 80 and RISC CPU's.

Disadvantage:-

→ If the design has to be modified or changed it requires changes in the wiring among the various components.

Microprogramming Control :-

In this, the control information is stored in the control memory. The control memory is programmed to initiate the required sequence of microoperations.

→ In the microprogramming control, any required changes or modifications can be done by updating microprogram in control unit.

Eg:- Intel 8080, Motorola 68000, and any CISC CPU's.

Control function

→ It specifies a microoperation is a binary value, which is equal to either 1 or 0.

→ The control unit initiates a series of sequential steps of microoperations.

Control Word :-

The control variable which can be represented by string of 1's and 0's called as 'control word'.

→ Control words can be programmed to perform various operations on the components of the system.

Scanned by CamScanner

Microprogrammed Control Unit:-

A control unit whose binary control variables are stored in memory is a microprogrammed control unit.

Microinstruction:-

→ The microinstruction specifies one or more microoperations for the system

→ Each word in control memory contains microinstruction

Microprogram:-

→ A sequence of microinstructions constitutes a microprogram

Alterations of the microprogram are not needed, once the control unit is in operation, the control memory can be a read-only memory (ROM).

→ The contents of the words in ROM are fixed and cannot be altered by simple programming and there is no writing facility for ROM.

→ The use of microprogram involves placing all the control variables in words of ROM for use by the control unit for read operations.

Control Memory:-

→ A memory that is a part of a control unit is referred to as control memory

Scanned by CamScanner

A Computer that employs a microprogrammed control unit will have two memories:

(i) Main Memory

(ii) Control Memory

Main Memory :- Main memory is available to the user for storing the programs.

→ It contains user programs machine instructions and data

→ The contents of main memory may alter when the data are manipulated.

Control memory :-

→ The control memory holds a fixed microprogram

→ Microprogram consists of microinstructions that specify various internal control signals for execution of register microoperations.

→ Each machine instruction initiates a series of microinstructions in control memory.

→ These microinstructions generate the microoperations to fetch the instruction from the main memory, evaluate the effective address and execute the operation specified by the op instruction and returns the control again to the fetch phase.

Scanned by CamScanner

Block Diagram and General Configuration of Microprogrammed Control unit :

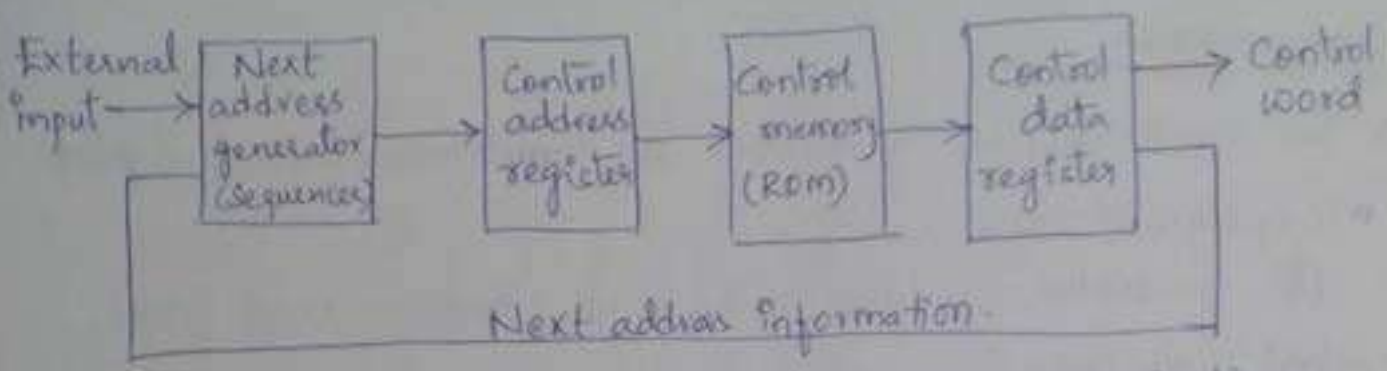


fig :- Microprogrammed Control Organization.

→ The control memory is assumed to be ROM within which all control information is permanently stored.

→ The control address register specifies the address of the microinstruction.

→ The microinstruction contains a control word that specifies one or more microoperations for the data processor.

→ Once these operations are executed, the control must determine the next address.

→ The location of the next microinstruction may be next in sequence or may be located somewhere else in the control memory.

→ for this reason, some bits of the present micro instruction is used generate the address of the next microinstruction.

→ Next address can also be the function of external input conditions.

Scanned by CamScanner

→ while the microoperations are being executed, the next address is computed in the next address register to read the next microinstruction.

Sequencer :-

→ The next address generator is sometimes called a prog 'microprogram sequencer'.

→ It generates address sequence that is read from control memory.

→ The next microinstruction address can be specified in several ways, depending on sequencer inputs.

Pipeline Register :-

→ The control data register is sometimes called as 'pipeline Register'.

→ The control data register holds the present microoperation

→ It allows the execution of the microoperations specified by the control word simultaneously with the generation of the next microinstruction.

Main Advantages of Microprogrammed Control :-

→ The main advantage is once the hardware configuration is established, there should be no need further hardware or wiring changes.

→ If we want different control sequence for the system, we need to specify a different set of microinstructions for control memory.

→ The hardware configuration should not be changed, only we need to change the microprogram in control memory

Scanned by CamScanner

Address Sequencing:-

→ Microinstructions are stored in control memory in groups. The microinstructions with each group specifies a 'routine'.

→ Each Computer instruction has its own microprogram routine in Control memory to generate the microoperations that executes the instruction.

→ The hardware that controls the address sequencing of the control memory must be capable of sequencing the microinstructions within a routine and be able to branch from one routine to another.

→ Let us enumerate the steps that the control unit undergo during the execution of a single computer instruction.

Step 1:- Let us consider an initial address is loaded into CAR (Control Address register) when power is turned on in the computer. This address is usually the address of the first microinstruction that activates the instruction fetch routine.

→ The fetch routine may be sequenced by incrementing the CAR through the rest of the microinstructions. At the end of the fetch routine, the instruction should be in the instruction register of the computer.

→ Fetch routine contains microinstructions. After each execution of microinstruction, we can go to next by incrementing

Scanned by CamScanner

the executed instruction which is present in instruction register.

Step 2:-

→ The control memory next must go through the routine that determines the effective address of the operand.

→ The instruction may have the status bit that specify various addressing modes like indirect address and index registers.

→ The address (effective) computation can be done by reaching branch microoperation which is conditioned on the status of the mode bit of the instruction.

→ If the effective address computation is completed, then address of the operand is available in the memory address register.

Step 3:- Now, we need to generate micro-operations, that execute the instruction fetched from the memory.

→ The steps of the micro-operation is generated in processor register, this register depends on the opcode part of the instruction.

→ The transformation from the instruction code bits to an address in control memory where the routine is located is referred as 'mapping' process.

→ A Mapping procedure is a rule that transforms the instruction code into control Memory Address (CMR).

Step 4:- Once the required routine is reached, the microinstructions that execute the instruction may be

Scanned by CamScanner

sequenced by incrementing the CAR.

But sometimes the sequence of microoperations depend upon the status bits in processor register. So at that time the microprograms use subroutine.

→ Microprograms contain subroutines which require an external register to store the return address.

→ Return address cannot be stored in ROM because writing capability is not available for it.

Hence, the address sequencing capabilities required in a control memory are:

1. Incrementing of the control address register.
2. Unconditional branch or conditional branch depending on status bit conditions.
3. A Mapping process from the bits of the instruction to an address for control memory.
4. A facility for subroutine call and return.

The below mentioned figure is the block diagram of a control memory and the associated hardware needed for selecting next microinstructions address. The diagram shows four different paths from which the CAR receives the address.

Scanned by CamScanner

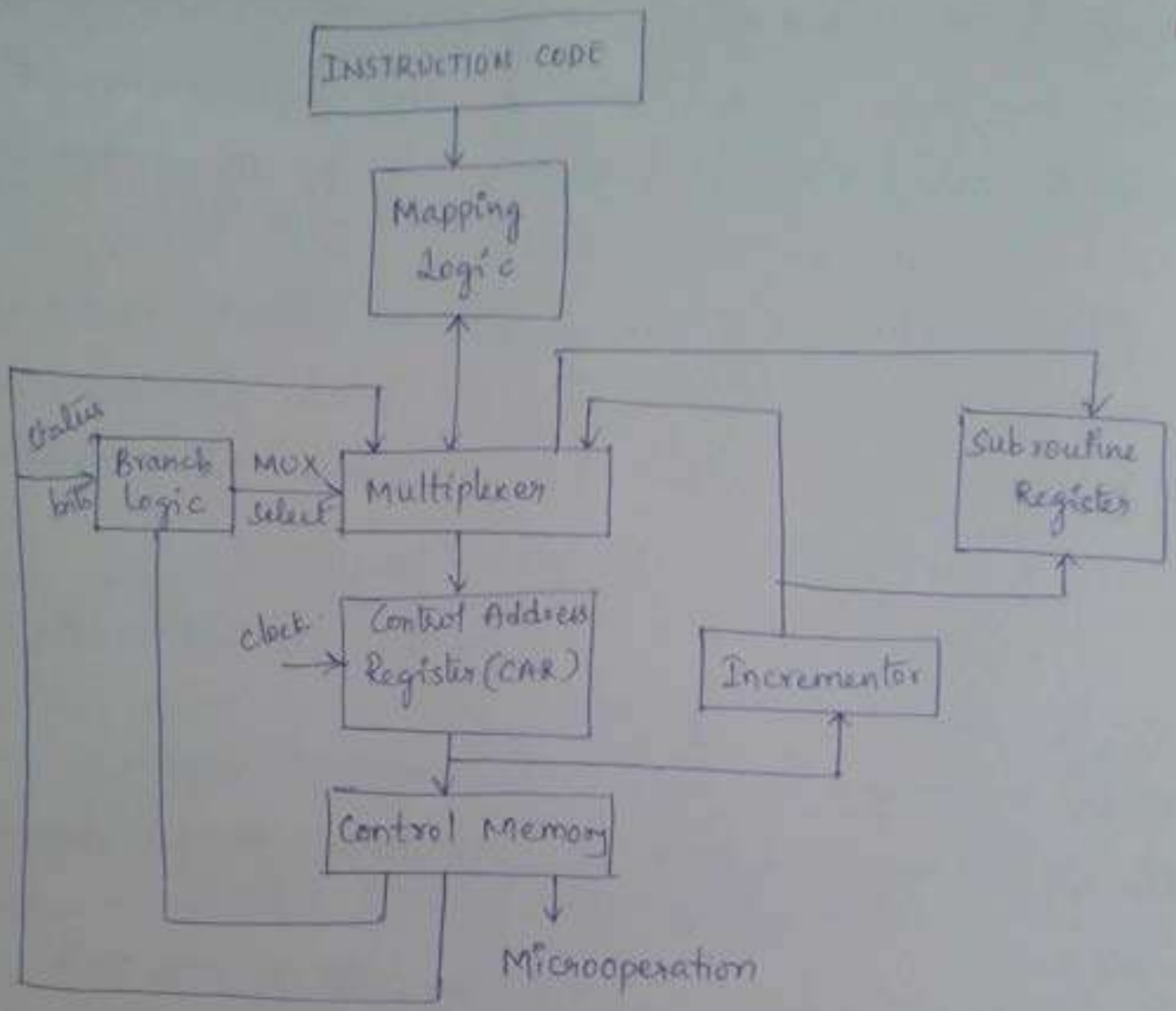


fig:- Block diagram of Control Memory / selection of address for Control memory

→ Incrementor increments the Content of the control address register by one, to select the next microinstruction sequence.

→ An external address is transferred into control memory via mapping logic circuit

→ The return address is stored in subroutine register.

→ Branching is acheived by specifying the branch address in one field. Conditional branching is acheived by using part of a microinstruction to select a specific status bit.

Scanned by CamScanner

Conditional Branching:-

→ The branch logic provides decision-making capabilities in the control unit.

→ The status conditions are special bits in the system that provide parameter information such as carry-out of an adder, the sign-bit of a number, the mode bits of an instruction and i/o status conditions.

→ The status bits, together with the field in the microinstructions that specify a branch address, control the conditional branch decisions generated in the branch logic.

→ The branch logic hardware may be implemented in a variety of ways.

→ The simplest way is to test the specified condition and branch to the indicated address if condition is true; otherwise the address register is incremented.

→ It is implemented by multiplexers

→ Suppose there are 8 status bit conditions in the system. Three bits in the microinstructions are used to specify one of eight status bit condition ($2^3 = 8$).

→ A 1 output in the multiplexer generates a control signal to transfer to the branch address

→ A 0 output in the multiplexer causes the address register to be incremented.

An Unconditional branch microinstruction can be implemented by loading the branch address from control memory into CAR by fixing the status bit

Scanned by CamScanner

at the input of the multiplexer, so it is always equal to 1.

Mapping of Instruction:-

→ A special type of branch exists when a microinstruction specifies a branch to the first word in the control memory where a microprogram routine for an instruction is located.

→ The status bits for this type are the bits in the operation code part of the instruction.

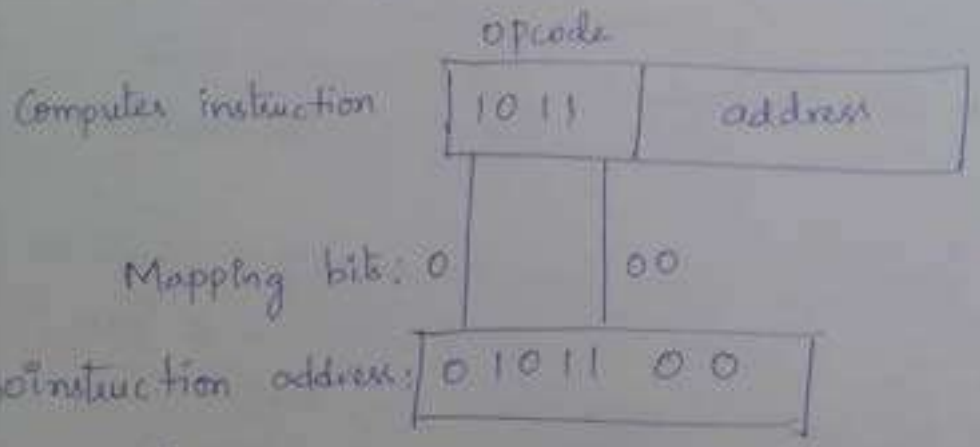


fig: Mapping from computer instruction code to microinstruction address.

Eg :- The above format has a opcode of four bits which can specify 16 distinct instructions ($2^4 = 16$)

→ Assume control memory has 128 words, requiring an address of seven bits.

→ for each operation code there exists a microprogram routine in control memory that executes the instruction.

→ One simple mapping which consists of placing a 0 in the most significant bit of the address, transferring the four operation code bits, and clearing the two least significant bits of CAR.

Scanned by CamScanner

→ This provides for each computer instruction a microprogram routine with a capacity of four microinstructions.

→ If the routines needs more than four microinstructions it can use addresses 1000000 through 1111111

Subroutines :-

→ Subroutines are programs that are used by other routines to accomplish a particular task.

→ A subroutine can be called from any point within the main body of the microprogram.

→ Microinstructions can be saved by employing subroutines that use common section of microcode.

for Example :- The sequence of microoperations needed to generate the effective address of the operand of an instruction is common to all memory reference instructions

→ Microprograms that use subroutines must have a provision for storing the return address during a subroutine call.

→ This can be done by placing the incremented output from the control address register into a subroutine register.

Scanned by CamScanner

Microprogram Example:-

→ Once the Configuration of a computer and its microprogram control unit is established, the designer's task is to generate the microcode for the control memory. This code generation is called microprogramming.

Computer Configuration:-

→ It consists of two memory units:

* Main memory for storing instructions and data
* Control memory for storing the microprogram.

→ four register are associated with processor register they are program Counter PC, Address Register AR, data register DR and accumulator register (AC).

→ The function of these registers are similar to the previous topic.

→ The Control unit has a control address register (CAR) and subroutine register (SBR).

The transfer of information among the registers in the processor is done through multiplexers rather than a Common bus.

→ DR receive information from AC, PC or memory

→ AR receive information from PC or DR.

→ PC receive information only from AR.

→ The ALU performs microoperations with data from AC and DR and places the result in AC.

Scanned by CamScanner

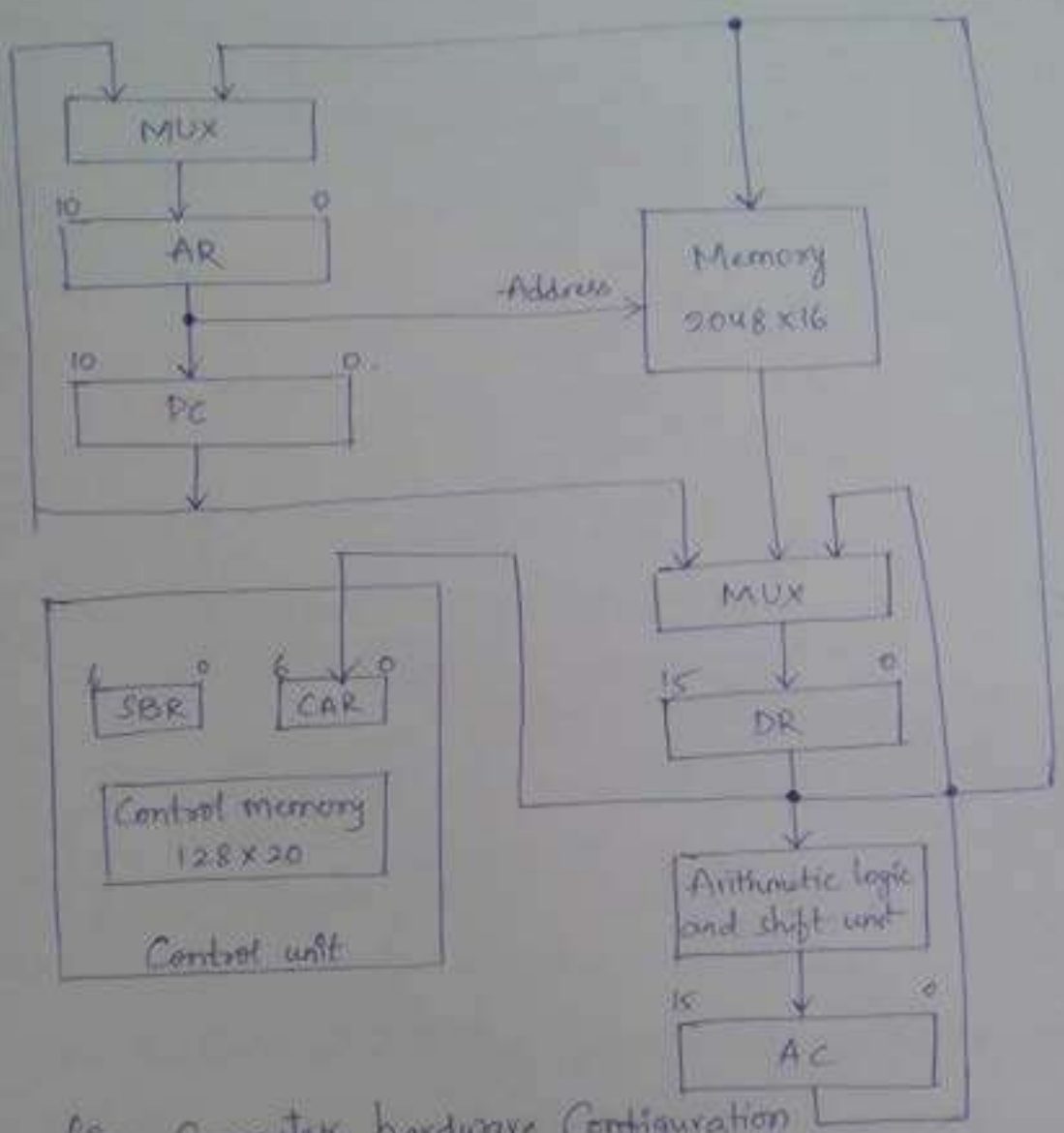


fig:- Computer hardware Configuration

Computer Instruction format :-

→ Computer Instruction format consists of three fields:

a 1bit field for indirect addressing symbolized by I

4 bit for opcode

11 bit for address field

Scanned by CamScanner

15	14 ... 11	10 ... 0
I	opcode	Address

(a) Instruction format.

Symbol	Opcode	Description
ADD	0000	AC → AC + M[EA]
BRANCH	0001	If (AC < 0) then (PC ← EA)
STORE	0010	M[EA] ← AC
EXCHANGE	0011	AC ← M[EA], M[EA] ← AC

EA is effective address

(b) four computer instructions

fig :- Computer Instructions.

→ The ADD instruction adds the content of the operand found in the effective address to the content of AC.

→ The BRANCH instruction causes a branch to the effective address if the operand in AC is negative.

→ The STORE instruction transfers the content of AC into the memory word.

→ The EXCHANGE instruction swaps the data between AC and memory word specified by the effective address.

Scanned by CamScanner

Microinstruction format :-

→ The microinstruction format for the control memory consists of 20 bits which are divided into four functional parts.

→ The three fields F1, F2 and F3 specify microoperations for the computer.

3	3	3	2	2	7
F1	F2	F3	CD	BR	AD

F1, F2, F3 : Microoperation field.

CD : Condition for branching.

BR : Branch field.

AD : Address field.

fig:- Microinstruction Code format (20 bits)

→ The CD field selects status bit Condition.

→ The BR field specifies the type or branch to be used.

→ The AD field contains a branch address.

→ The microoperations are subdivided into three fields of three bits each.

→ The three bits in each field are encoded to specify seven distinct microoperations.

→ This gives a total of 21 microoperations ($2^3+2^3+2^3=21$ since it starts from 0)

→ No more than three microoperations can be chosen for a microinstruction, one from each field.

Scanned by CamScanner

→ Binary code 000 for no operation.

Eg :- A microinstruction can specify two simultaneous microoperations from F2 and F3 and not from F1

$DR \leftarrow M[AR]$ with F2 = 100

and $PC \leftarrow PC+1$ with F3 = 101

Table: Symbols and Binary code for Microinstruction fields

F1	Microoperation	Symbol
000	None	NOP
001	$AC \leftarrow AC + DR$	ADD
010	$AC \leftarrow 0$	CLRAC
011	$AC \leftarrow AC+1$	INCAC
100	$AC \leftarrow DR$	DRTAC
101	$AR \leftarrow DR(0-10)$	DRTAR
110	$AR \leftarrow PC$	PCTAR
111	$M[AR] \leftarrow DR$	WRITE

→ We should not specify the conflicting microoperations simultaneously

Eg :- A microoperation field 010 001 000 has no meaning because it specifies the operations to clear AC to 0 and subtract DR from AC at the same time.

Scanned by CamScanner

Note:-

→ Each microoperation in the table is defined with a register transfer statement and is assigned a symbol for use in a symbolic microprogram.

→ All transfer-type microoperations symbols use five letters.

→ first two letters designate the source register, the third letter is always T and last two letters indicate destination register.

Eg:- $AC \leftarrow DR$ (F1=100) has the symbol DRTAC which stands for a transfer from DR to AC.

F2	Microoperation	Symbol
000	None	NOP
001	$AC \leftarrow AC - DR$	SUB
010	$AC \leftarrow AC \vee DR$	OR
011	$AC \leftarrow AC \wedge DR$	AND
100	$DR \leftarrow M[AR]$	READ
101	$DR \leftarrow AC$	ACTDR
110	$DR \leftarrow DR+1$	INCDR
111	$DR(0-10) \leftarrow PC$	PCTDR

F3	Microoperation	Symbol
000	None	NOP
001	$AC \leftarrow AC \oplus DR$	XOR
010	$AC \leftarrow \overline{AC}$	COM
011	$AC \leftarrow shl\ AC$	SHL
100	$AC \leftarrow shr\ AC$	SHR
101	$PC \leftarrow PC+1$	INCPC
110	$PC \leftarrow AR$	ARTPC
111	Reserved	

Table:- Symbols & Binary code for Microoperation fields.

Scanned by CamScanner

Condition field (CD):-

The CD (Condition) field consists of two bits which are encoded to specify four status bits conditions.

→ The first condition is always 1, so that a reference to CD=00 will always find the condition to be true. If it is used with the BR field, it provides unconditional branch operation.

→ The indirect bit I is available from bit 15 of DR after an instruction is read from memory.

→ The sign bit of AC provides the next status bit.

→ The zero value, symbolized by Z, is a binary variable whose value is equal to 1, if all the bits in AC are equal to '0'.

CD	Condition	Symbol	Comments
00	Always = 1	U	Unconditional branch
01	DR(15)	I	Indirect Address bit
10	AC(15)	S	Sign bit of AC
11	AC = 0	Z	Zero value in AC.

→ The BR (branch) field consists of two bits.

→ It is used along with the address field AD, to choose the address of the next microinstruction

→ when BR = 00, the control performs a jump (JMP) operation. when BR = 01 it performs a CALL operation.

→ The two operations are identical except that a Call microinstruction stores the return address in the subroutine register.

Scanned by CamScanner

→ The jump and Call operations depend on the value of the CD field, if the CD is equal to 1, the next address in the AD field is transferred to the Control Address Register (CAR).

BR	Symbol	Function
00	JMP	CAR ← AD if condition = 1 CAR ← CAR+1, if condition = 0
01	CALL	CAR ← AD, SBR ← CAR+1 if Condition = 1 CAR ← CAR+1 if condition = 0
10	RET	CAR ← SBR (Return from subroutine)
11	MAP	CAR(2-5) ← DR(11-14), CAR(0,1,6) ← 0

Symbolic Microinstructions:-

→ A symbolic microprogram can be translated into its binary equivalent by means of an assembler.

→ Each line of the assembly language microprogram defines a symbolic microinstruction.

→ Each symbolic microinstruction is divided into five fields:

1. Labels
2. Microoperations
3. CD
4. BR &
5. AD.

Scanned by CamScanner

① The label field may be empty or it may spec a symbolic address. A label is terminated with :

② The microoperations field consists of one or more symbols, separated by commas. There may be no more than one symbol from each F field.

③ The CD field has one of the letters U, I, S or Z.

④ The BR field contains one of the four symbols specified in the table.

⑤ The AD field specifies a value for the address field of the microinstruction in one of three ways:

ⓐ with a symbolic address, which must also appear as a label.

ⓑ with the symbol NEXT to designate the next address in sequence.

ⓒ when the BR feild contains RET or MAP symbol, the AD field is left empty.

We will also use pseudoinstruction ORG to define the origin or first address of a microprogram routine. ORG 64 informs the assembler to place the next microinstruction in control memory at decimal address 64

Eg :-

ORG 64

FETCH : PCTAR, READ, INCPC, DRTAR } microinstructions

Label

U, U, U } CD

JMP, JMP, MAP } BR field

NEXT, NEXT } AD field

The Fetch Routine :-

→ The control memory has 128 words, each word contains 20 bits.

→ The first 64 words (addresses 0 to 63) are to be occupied by the routines for 16 instructions (~~16 × 4 bits each = 64~~).

→ The last 64 words may be used for another purpose.

→ The Convenient starting location for the fetch routine is address 64.

→ The microinstructions needed for fetch routine are

$AR \leftarrow PC$

$DR \leftarrow M[AR]$, $PC \leftarrow PC+1$

$AR \leftarrow DR(0-10)$, $CAR(2-5) \leftarrow DR(11-14)$, $CAR(0,1,6) \leftarrow 0$

The address is transferred from PC to AR, then read from memory into DR. from these the address part is transferred to AR and then control is transferred to one of 16 routines by mapping the operation code part of the instruction from DR into CAR.

→ The fetch routine needs three microinstructions, which are placed in the control memory at addresses 64, 65 and 66.

Symbolic Microprogram for fetch routine :-

```
        ORG 64
FETCH : PCTAR          U   JMP   NEXT
        READ, INCPC    U   JMP   NEXT
        DRTAR          U   MAP
```

Scanned by CamScanner

Symbolic Microprogram

What is a Microprogram?:-

Several Microinstructions together constitutes a microprogram.

The two ways to write a microprogram are

(1) Symbolic Microprogram.

(2) Binary Microprogram.

Symbolic Microprogram:-

→ The execution of the third (MAP) micro instruction in the fetch routine results in a branch to address 0xxxx00, where xxxx refers to the four bits of the operation code.

→ In each routine we must provide microinstructions for evaluating the effective address and for executing the instruction. The indirect address mode is associated with all memory-reference instructions.

→ Indirect address considers the address part of the instruction as the address where the effective address is stored rather than address of the operand.

→ Memory has to be accessed to get effective address, which is then transferred to AR.

→ The return from subroutine (RET) transfers the address from SBR to CAR, thus returning to the second microinstruction of the ADD routine.

Scanned by CamScanner

- The execution of the ADD instruction is carried out by the microinstructions at addresses 1 and 2.

→ The first microinstruction reads the operand from memory into DR.

→ The second microinstruction performs an add microoperation with the content of DR and AC and then jumps back to the beginning of fetch routine.

Table :- Symbolic Microprogram

Label	Microoperations	CD	BR	AD
	ORG 0			
ADD:	NOP	I	CALL	INDRCT
	READ	U	JMP	NEXT
	ADD	U	JMP	FETCH
	ORG 4			
BRANCH:	NOP	S	JMP	OVER
	NOP	U	JMP	FETCH
OVER:	NOP	I	CALL	INDRCT
	ARTPC	U	JMP	FETCH
	ORG 8			
STORE:	NOP	I	CALL	INDRCT
	ACTDR	U	JMP	NEXT
	WRITE	U	JMP	FETCH
	ORG 12			
EXCHANGE:	NOP	I	CALL	INDRCT
	READ	U	JMP	NEXT
	ACTDR, DRTAC	U	JMP	NEXT
	WRITE	U	JMP	FETCH

Scanned by CamScanner

	ORG 64			
FETCH :	PCTAR	U	JMP	NEXT
	READ, INCPC	U	JMP	NEXT
	DRTAR	U	SMAP	~~NEXT~~
INDRCT :	READ	U	JMP	NEXT
	DRTAR	U	RET	

Binary Microprogram:-

→ The symbolic microprogram is a convenient form for writing microprograms in a way that people can read and ~~write~~ understand.

→ But that's not the way the microprogram is stored in memory.

→ The symbolic microprogram must be translated to binary either by means of an assembler program or by the user if the microprogram is simple enough.

→ The addresses for control memory are given in both decimal and binary. The binary content of each microinstruction is derived from the symbols.

→ Note :- Address 3 has no equivalent in the symbolic microprogram since the ADD routine has only three microinstructions at addresses 0, 1 and 2. The next routine starts at address 4. All 0's are specified in this location and never be used. If some errors occur or if noise signal sets CAR to the value of 3.

Scanned by CamScanner

Table:- Binary Microprogram for Control Memory.

Micro Routine	Address Decimal	Address Binary	Binary Microinstruction F1	F2	F3	CD	BR	AD
ADD	0	0000000	000	000	000	01	01	1000011
	1	0000001	000	100	000	00	00	0000010
	2	0000010	001	000	000	00	00	1000000
	3	0000011	000	000	000	00	00	1000000
BRANCH	4	0000100	000	000	000	10	00	0000110
	5	0000101	000	000	000	00	00	1000000
	6	0000110	000	000	000	01	01	1000011
	7	0000111	000	000	110	00	00	1000000
	⋮							
FETCH:	64	1000000	110	000	000	00	00	1000001
	65	1000001	000	100	101	00	00	1000010
	66	1000010	101	000	000	00	11	0000000
INDRCT	67	1000011	000	100	000	00	00	1000100
	68	1000100	101	000	000	00	10	0000000

If a writable control memory is employed, the ROM is replaced by a RAM.

→ The advantage of using RAM for control memory is that the microprogram can be altered by writing new patterns of 1's and 0's instead of hardware procedures.

Scanned by CamScanner

Design of Control Unit :-

→ The bits of the microinstruction are usually divided into fields, with each field defining a distinct, separate function.

→ The various fields in instruction formats provide control bits to initiate the microoperations and special bits to specify the way that the next address is to be evaluated.

→ The number of control bits can be reduced by grouping mutually exclusive variables into fields and encoding the k bits in each field to provide 2^k microoperations.

→ Each field requires a decoder to produce the corresponding control signals.

→ The outputs of the decoders are connected to the appropriate inputs in the processor unit.

decoding of F-fields

→ The three decoders ~~and~~ of three fields and their connections must be made from their outputs.

→ Each field is decoded with 3x8 decoder to provide eight outputs.

→ Each of this output must be properly connected to the circuit to initiate the microoperation.

Scanned by CamScanner

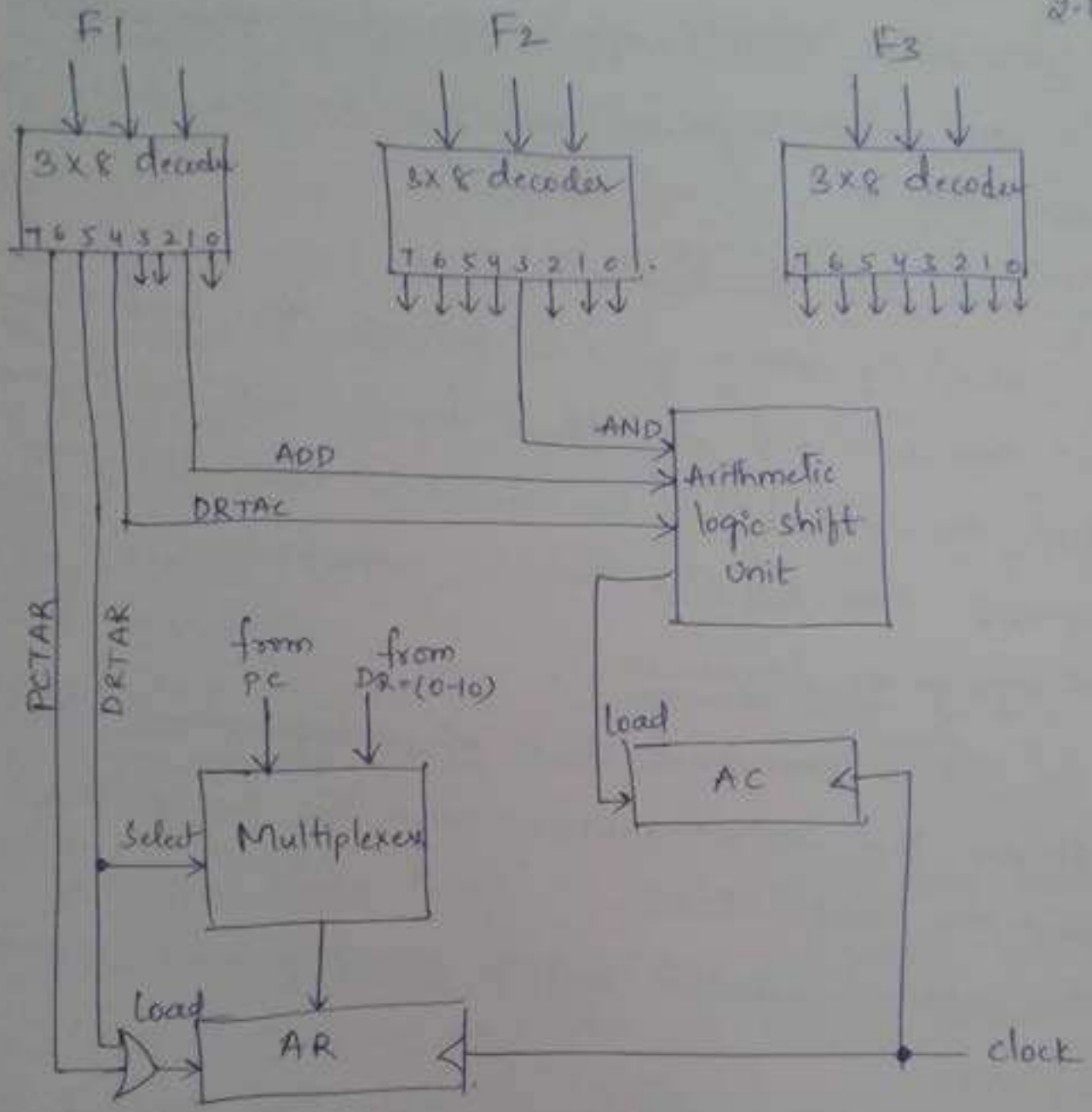


fig:- decoding of microoperation fields.

Eg:- when F1 = 101, the next clock pulse transition transfers the content of DR(0-10) to AR.

→ output 5 and 6 of decoder F1 are connected to the load input of AR so that either one of these outputs is active, information from multiplexers is transferred to AR.

Scanned by CamScanner

→ The other outputs of the decoder that initiate transfers b/w registers and must be connected in the same way.

- Arithmetic and logic shift Unit :-

Instead of using gates to generate the control signals for AND, ADD and DR, these inputs will now come from the outputs of the decoder associated with the symbols AND, ADD and DRTAC, respectively.

Microprogram Sequencer

→ The basic components of a microprogrammed control unit are the control memory and the circuits that select the next address.

→ The address selection part is called a microprogram sequencer.

→ A microprogram sequencer can be constructed with digital functions to suit a particular application.

→ To guarantee a wide range of acceptability, an IC must be provided in the organization that can be adapted to a wide range of applications.

→ The purpose of a microprogram sequencer is to present an address to the control memory so that a microinstruction may be read and executed.

→ Some sequencers provide an output register which can function as a address register for control memory.

Scanned by CamScanner

Block Diagram of microprogram Sequencer :-

- There are Two multiplexers in the circuit.
- The first multiplexer selects an address from one of four sources and routes into a CAR.
- The second multiplexer tests the value of a selected status bit and the result of the test is applied to an input logic unit.
- The output from CAR provides the address for the Control memory.
- The Content of CAR is incremented and applied to one of the multiplexer inputs and to the subroutine register SBR.
- The other three inputs to multiplexer 1 come from the address field of the present microinstruction, from the output of SBR, and from an external source that maps the instruction.
- The CD field of the microinstruction selects one of the status bits in the second multiplexer.
- If the bit selected is equal to 1, the T (test) = 1; otherwise it is equal to 0.
- The T value together with the two bits from the BR (branch) field goes to an input logic circuit.
- The input logic in a particular sequencer will determine the type of operation.

Scanned by CamScanner

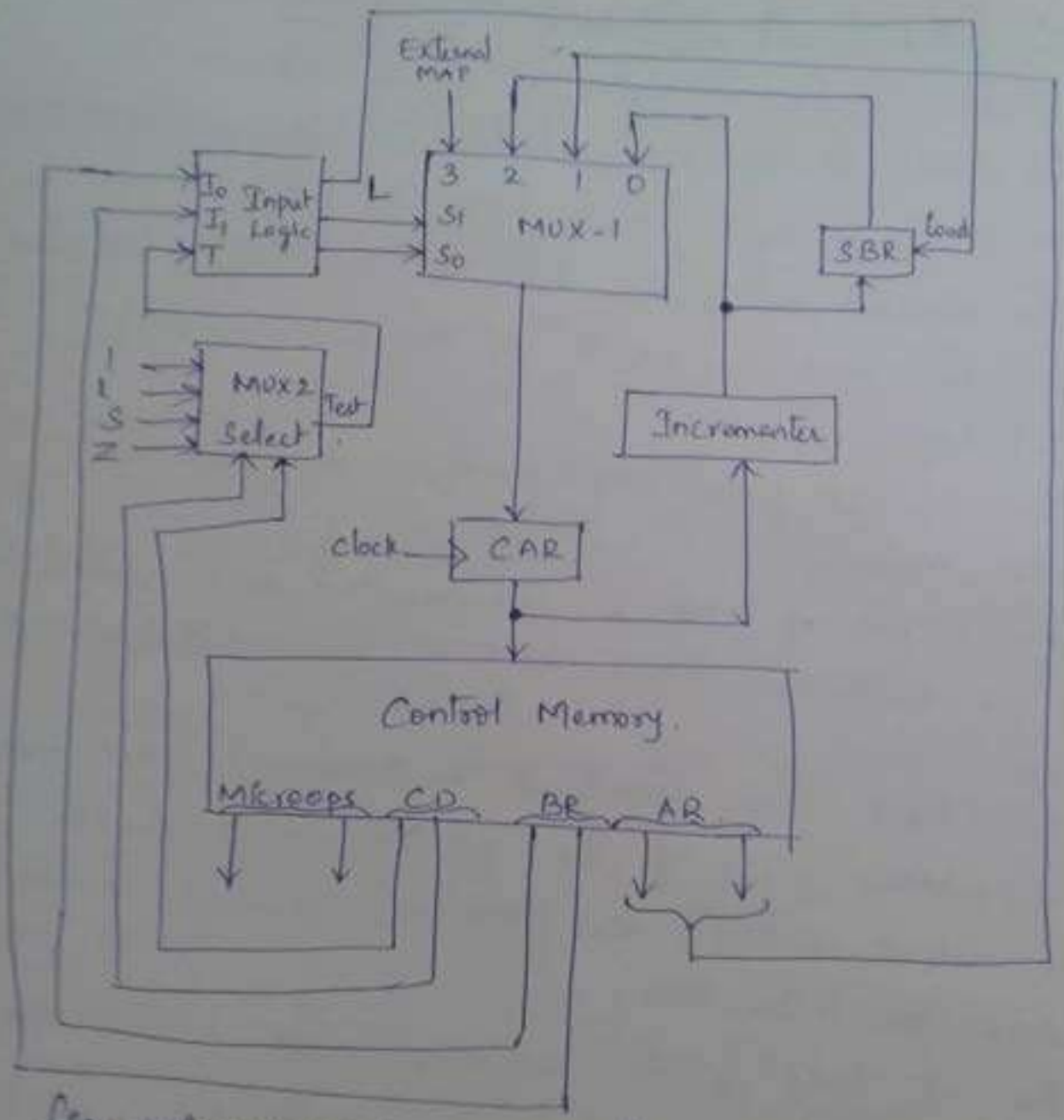


fig:- Microprogram sequencer for a Control memory.

Table: Input logic Truth Table for Microprogram Sequencer.

BR Field		Input I_1	I_0	T	MUX 1 S_1	S_0	Load SBR L
0	0	0	0	0	0	0	0
0	0	0	0	1	0	1	0
0	1	0	1	0	0	0	0
0	1	0	1	1	1	1	1
1	0	1	0	X	1	0	0
1	1	1	1	X	1	1	0

function as a ..

Scanned by CamScanner

Boolean function:-

$$S_1 = I_1$$
$$S_0 = I_1 I_0 + I_1' T$$
$$L = I_1' I_0 T$$

Typical sequencer operations are: increment, branch or jump, call and return from subroutine, load an external address, push or pop the stack, and other address sequencing operations.

→ with three inputs, the sequencer can provide up to eight address sequencing operations.

→ some commercial sequencers have three or four inputs in addition to the T inputs and thus provide a wider range of operations.

Scanned by CamScanner

Central Processing Unit

Introduction :-

The part of the computer that performs the bulk of data-processing operations is called central Processing Unit.

→ The CPU is made up of three major parts.

(1) Register Set

(2) Arithmetic Logic Unit (ALU)

(3) Control Unit.

→ Registers set stores the intermediate data used during the execution of the instructions.

→ The ALU performs the required microoperations required for executing the instructions.

→ The Control Unit supervises the transfer of information among the registers and instructs the ALU to perform the appropriate operation at that moment.

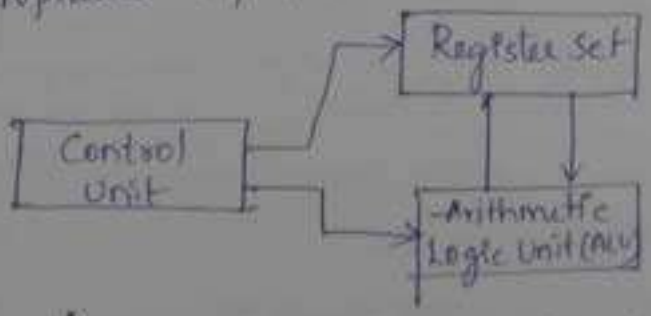


fig:- Major Components of CPU.

→ Computer architecture is sometimes defined as the computer structure and behavior as seen by the programmer that uses machine language instructions.

→ This includes instruction formats, addressing modes, the instruction set, and general organization of the CPU

Scanned by CamScanner

→ Based on general organization of CPU register leading to two computer architectures

(i) Reduced Instruction set Computer. (RISC)
(ii) Complex Instruction Set Computer (CISC).

→ Based on memory usage for programs and data, two architectures are evolved they are

(i) Nonembedded Computers
(ii) Embedded Computers

→ Nonembedded Computer architecture follows Stored program Computer (SPC) architectures in which programs and data reside in the same memory system

Eg:- All desktop system such as personal Computers.

→ Embedded architecture are basically Harvard Computer architectures in which programs and data reside in different memory system

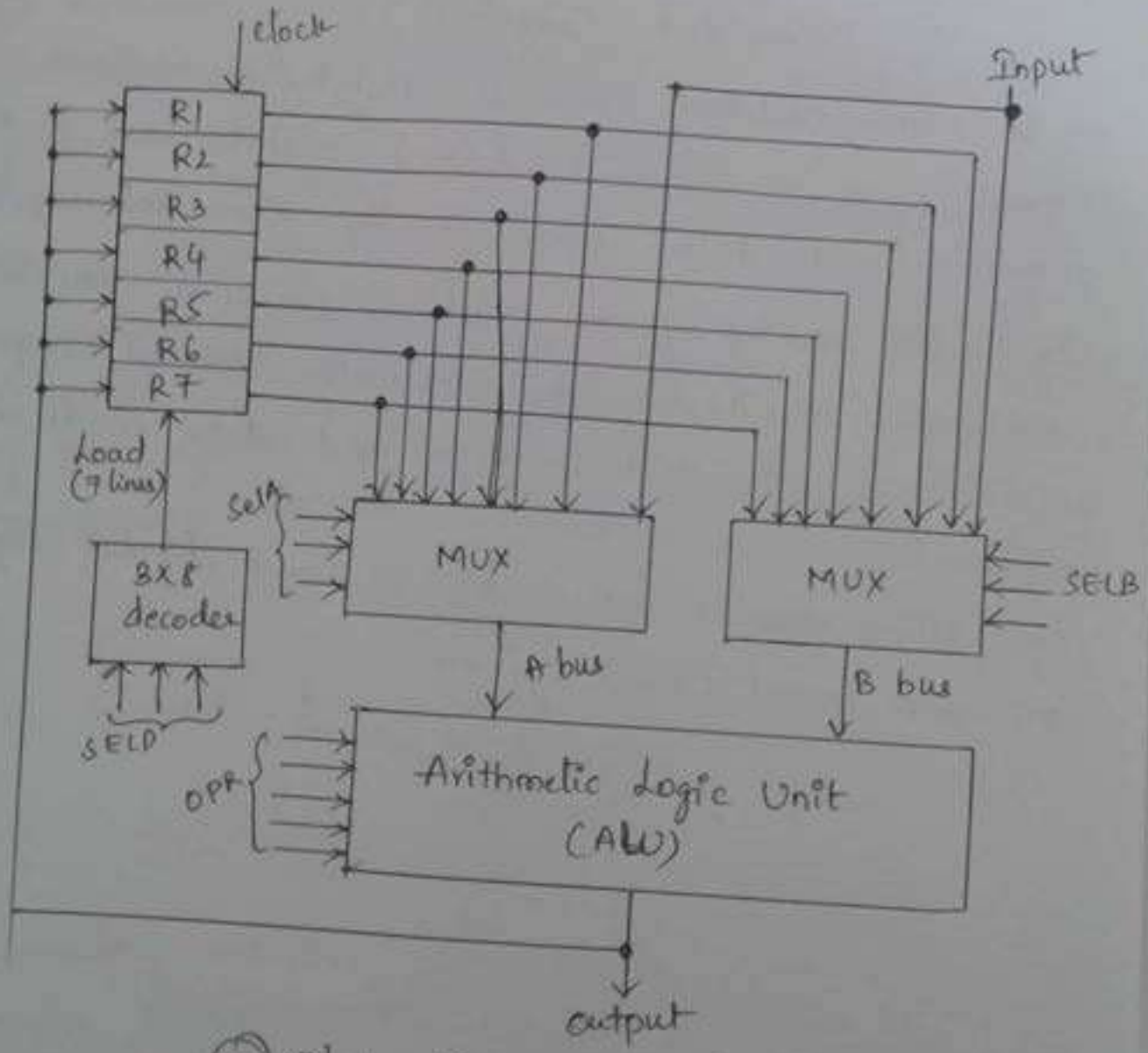
Eg:- Microcontroller-based systems and digital signal processor based (DSP) systems.

Scanned by CamScanner

General Register Organization :-

→ The registers communicate with each other not only for direct data transfers, but also for performing various microoperations.

→ It is necessary to provide a common unit that can perform all the arithmetic, logic and shift microoperations in the processor.



(a) Block diagram for Register set with Common ALU.

→ A bus organization for seven CPU Registers is shown above

-> The output of each register is connected to two multiplexers (MUX) to form two buses bus A and bus B.
-> The select lines in each multiplexer select one register or the input data for the particular bus.
-> The A and B bus inputs are connected to common ALU.
-> The operation selected in the ALU determines the arithmetic or logic microoperation that is to be performed.
-> The result of the microoperation is available for output data and also goes into the inputs of all the register.
-> The register receives the information from the output bus is selected by a decoder.
-> The decoder activates one of the register load inputs thus providing a transfer path b/w the data in the output bus and inputs of selected destination register.
-> Control Unit that operates the CPU bus system directs the information flow through the registers and ALU.

<u>for Example</u> :- R1 ← R2 + R3.

The Control must provide binary selection variables for the following selector inputs:

1. MUX A selector (SELA): To place the content of R2 in bus A
2. MUX B selector (SELB): To place the content of R3 on bus B
3. ALU operation selector (OPR): To provide addition A+B.
4. Decoder destination selector (SELD): To transfer the content of the output bus into R1.

Scanned by CamScanner

Control Word

→ There are 14 binary selection inputs in the Unit, and their combined value specifies a 'Control word'

3	3	3	5
SELA	SELB	SELD	OPR

b) Control Word for register set with common ALU.

→ It consists of 4 fields among them 3 fields contains three bits each, one field has five bits.

→ The Three bits of SELA select a source register for the A input of ALU

→ The three bits of SELB select a register for input B of the ALU.

→ The three bits of SELD select a destination register using the decoder and seven load outputs.

→ The five bits of OPR select one of the operations in the ALU.

Binary code	SELA	SELB	SELD
000	Input	Input	None
001	R1	R1	R1
010	R2	R2	R2
011	R3	R3	R3
100	R4	R4	R4
101	R5	R5	R5
110	R6	R6	R6
111	R7	R7	R7

Table: Encoding of Register selection fields

Scanned by CamScanner

→ SELA, SELB, SELD register selected is the one whose decimal number is equivalent to binary number.

→ when SELA = SELB = 000, multiplex selects the external input data

→ when SELD = 000, no destination register is selected.

ALU :-

→ It provides arithmetic and logic operations

→ It also must provide shift operations. The shifter may be placed in the input of ALU to provide preshift capability or at the output to provide postshifting capability.

→ The OPR field has five bits and each operation is designated with symbolic name.

OPR SELECT	Operation	Symbol
00000	Transfer A	TSFA
00001	Increment A	INCA
00010	Add A+B	ADD
00101	Subtract A-B	SUB
00110	Decrement A	DECA
01000	AND A and B	AND
01010	OR A and B	OR
01100	XOR A and B	XOR
01110	Complement A	COMA
10000	Shift Right A	SHRA
11000	Shift left A	SHLA

Table: ALU Operations Encoding.

Scanned by CamScanner

Example of Microoperations

→ The control word of 14 bits is needed to specify a microoperation in the CPU.

for Example Consider the subtraction microoperation given by the statement:

$$R1 \leftarrow R2 - R3$$

R2 – input A of the ALU
R3 – Input B of the ALU
R1 – destination Register.
R2 – R3 – ALU operation to subtract A – B.

The binary control word for the above subtract microoperation is as follows:

Field:	SELA	SELB	SELD	OPR
Symbol:	R2	R3	R1	SUB
Control Word:	010	011	001	00101

Table: Examples of Microoperations for the CPU.

Microoperation	Symbolic Designation				Control Word
	SELA	SELB	SELD	OPR	
R1 ← R2 - R3	R2	R3	R1	SUB	010 011 001 00101
R4 ← R4 ∨ R5	R4	R5	R4	OR	100 101 100 01010
R6 ← R6 + 1	R6	–	R6	INCA	110 000 110 00001
R7 ← R1	R1	–	R7	TSFA	001 000 111 00000
Output ← R2	R2	–	None	TSFA	010 000 000 00000
output ← input	Input	–	None	TSFA	000 000 000 00000
R4 ← shl R4	R4	–	R4	SHLA	100 000 100 11000
R5 ← 0	R5	R5	R5	XOR	101 101 101 01100

Scanned by CamScanner

Instruction formats :-

A computer will usually have wide variety of instruction formats. It is the function of the control unit within the CPU to interpret each instruction code and provide necessary control functions.

→ The bits of the instruction are divided into groups called fields. The most common fields found in instruction formats are.

(1) An operation code field to specify the type of operation

(2) An address field that designates a memory address or processor registers.

(3) A mode field the specifies the way the operand or Effective address is determined.

→ Operations specified by computer instructions are executed on some data stored in memory or processor registers.

→ Operands reside in memory are specified by memory address.

→ Operands which are residing in processor registers are specified with a register address.

→ A register address is a binary number of k bits defines one of 2^k registers in CPU.

→ Thus there are 16 Registers R0 through R15 in CPU with register address field of four bits.

Scanned by CamScanner

Eg:- The binary number 0101 designate register R5.

→ Computers may have instructions of several different lengths containing varying number of addresses. Most computers fall into one of three types of CPU organization:

1) Single accumulator Organization.
2) General Register Organization.
3) Stack Organization.

Single Accumulator Organization:-

→ All the operations are performed with an implied accumulator register (AC)

→ The instruction format in this type of Computer uses one address field

Eg:- The instruction that specified an arithmetic addition is defined by assembly language. Instruction as

ADD X

where X is the address of the operand.

ADD implies that AC ← AC + M[X].

where AC is the accumulator register

M[X] – Memory word located at address 'X'.

General Register Organization:-

→ The Instruction format in this type of Computer needs three register address fields.

Scanned by CamScanner

9:- Instruction for arithmetic addition may be written in assembly language is.

ADD R1, R2, R3

denotes $R1 \leftarrow R2 + R3$.

The number of address fields can be reduced from three to two as

ADD R1, R2

denoted $R1 \leftarrow R1 + R2$.

only two register address R1 and R2 are specified in this instruction.

→ Computers with multiple processor registers use the move instruction with mnemonic MOV to symbolize transfer

MOV R1, R2

denotes transfer $R1 \leftarrow R2$

→ General register type Computers employ two or three address fields in the instruction format.

ADD R1, X

denotes $R1 \leftarrow R1 + M[X]$. It has two address fields one for register R1 and other for memory address X

Stack Organization:-

→ Computers with stack organization would have PUSH and POP instructions. which require an address field.

Scanned by CamScanner

The Instruction is

PUSH X

It will push the word at address X to top of the stack

→ Operation type instructions do not need an address field in stack-organized computers.

The instruction is

ADD

only operation code with no address field.
It pops the two top numbers from stack, adds it and pushes the sum into the stack.

Instructions are categorized into different formats with respect to the operand fields in the instructions.

1. Three Address Instructions.
2. Two Address Instruction.
3. One Address Instruction.
4. Zero Address Instruction.
5. RISC Instruction.

<u>Three Address Instruction</u>:-

→ Computers with three-address instruction formats can use each address field to specify either a processor register or a memory operand.

→ The program in assembly language that evaluates $X = (A+B) * (C+D)$ is shown below

Scanned by CamScanner

ADD	R1, A, B	R1 ← M[A] + M[B]
ADD	R2, C, D	R2 ← M[C] + M[D]
MUL	X, R1, R2	M[X] ← R1 * R2

Computer has two processor registers R1 and R2. The symbol M[A] denotes the operand at memory address symbolized by A.

→ The advantage of the three-address format is that it results in short programs when evaluating arithmetic expressions.

→ The disadvantage is that the binary-coded instructions require too many bits to specify address.

Two-Address Instructions:-

→ It is most common in commercial computers.

→ The address field can specify either a processor register or a memory word.

Eg:- X = (A+B) * (C+D)

MOV	R1, A	R1 ← M[A]
ADD	R1, B	R1 ← R1 + M[B]
MOV	R2, C	R2 ← M[C]
ADD	R2, D	R2 ← R2 + M[D]
MUL	R1, R2	R1 ← R1 * R2
MOV	X, R1	M[X] ← R1

The MOV instructions moves or transfers the operands to and from memory and processor registers

Scanned by CamScanner

One-Address Instructions:-

→ One-address instructions use an accumulator register for all data manipulation.

→ Here, we neglect the second register and assume that AC contains the result of all operations.

Eg:- X = (A+B) * (C+D)

LOAD	A	AC ← M[A]
ADD	B	AC ← AC + M[B]
STORE	T	M[T] ← AC
LOAD	C	AC ← M[C]
ADD	D	AC ← AC + M[D]
MUL	T	AC ← AC * M[T]
STORE	X	M[X] ← AC.

T is temporary register for storing the intermediate result.

Zero-Address Instructions:-

→ A stack-organized computer does not use an address field for the instructions ADD and MUL

→ The PUSH and POP instructions, need an address field to specify the operand that communicates with the stack.

PUSH	A	TOS ← A
PUSH	B	TOS ← B
ADD		TOS ← (A+B)
PUSH	C	TOS ← C
PUSH	D	TOS ← D
ADD		TOS ← (C+D)
MUL		TOS ← (C+D) * (A+B)
POP	X	M[X] ← TOS

TOS - Top of the stack

Scanned by CamScanner

RISC Instructions:-

→ The Instruction set of a typical RISC processor is restricted to the use of load and store instructions, when communicating with CPU and memory.

→ A program for a RISC-type CPU consists of LOAD and STORE instructions that have one memory and one register address, Computational-type instructions that have three addresses with all three specifying processor registers.

Eg:- $X = (A+B) * (C+D)$

LOAD	R1, A	R1 ← M[A]
LOAD	R2, B	R2 ← M[B]
LOAD	R3, C	R3 ← M[C]
LOAD	R4, D	R4 ← M[D]
ADD	R1, R1, R2	R1 ← R1 + R2
ADD	R3, R3, R2	R3 ← R3 + R4
MUL	R1, R1, R3	R1 ← R1 * R3
STORE	X, R1	M[X] ← R1

Addressing Modes :-

→ The addressing mode specifies a rule for interpreting or modifying the address field of the instruction before the operand is actually referenced.

→ Computers use addressing mode techniques for the purpose of the following provisions:

Scanned by CamScanner

1. To give programming versatility to the user by providing such facilities as pointers to memory, counters for loop control, indexing of data and program relocation.

2. To reduce the number of bits in the addressing field of the instruction.

→ The control unit of a computer is designed to go through an instruction cycle that is divided into three major ~~parts~~ phases;

1. Fetch the instruction from memory.
2. Decode the instruction.
3. Execute the instruction.

Program Counter :-

→ PC holds the address of the instruction to be executed next and is incremented each time an instruction is fetched from memory.

Mode field :- The mode field is used to locate the operands needed for the operation

opcode	Mode	Address

→ The mode field is used to locate the operands needed for the operation.

Scanned by CamScanner

→ The following are the different types of addressing modes:

Implied mode :-

→ In this mode the operands are specified implicitly in the definition of the instruction.

→ All register reference instructions that use an accumulator are implied-mode instructions.

→ Zero-address instructions are implied-mode instruction.

Immediate mode :-

→ In this mode the operand is specified in the instruction itself

→ That means, it has operand field rather than an address field.

→ It is also used to initializing register to Constant value.

→ The operand field contains the actual operand to be used in Conjunction with the operation.

Register Mode :-

→ In this mode the operands are in registers that within the CPU. The particular register selected from a register field in the instruction.

→ A k-bit field can specify any one of 2^k registers.

Register Indirect Mode :-

→ In this mode the instruction specifies a register in the CPU whose Contents give the address of the operand in memory.

→ Before using this mode, the programmer ensure the memory address of the operand is placed in processor register.

Scanned by CamScanner

→ ~~the advantage~~

5) Autoincrement or Autodecrement mode :-

→ This is similar to the register indirect mode expect that the register is incremented or decremented after (or before) its value is used to access memory.

→ When the address stored in the register refers to a table of data in memory, it is necessary to increment or decremented the register after every access to the table.

6) Direct Address Mode :-

→ In this mode the effective address is equal to the address part of the instruction. The operand resides in memory and its address is given directly by the address field of the instruction.

7) Indirect Address Mode :-

→ In This mode the address field of the instruction will give the address where the effective address is stored in memory

Effective address = address part of instruction + content of CPU register.

8) Relative Address Mode :-

→ In this mode the Content of the program Counter is added to the address of the instruction inorder to obtain the effective address.

→ The address part of the instruction usually a signed number which can be either positive or negative.

Scanned by CamScanner

9) Indexed Addressing Mode:-

→ In this mode, the content of an index register is added to the address part of the instruction to obtain the effective address

→ The indexed register is a special CPU register that contain an index value.

→ The address field of the instruction defines the beginning address of a data array in memory.

→ The distance between the beginning address and the address of the operand is the index value stored in the index register.

10) Base Register Addressing Mode:-

→ In this mode, the content of a base register is added to the address part of the instruction to obtain the effective address.

→ A base register is assumed to hold a base address and the address field of the instruction gives a displacement relative to this base address.

→ This mode is used for relocation of programs in memory

Scanned by CamScanner

Data Transfer and Manipulation.

Most computer instructions can be classified into three categories:

1. Data transfer Instructions.
2. Data Manipulation instructions.
3. Program Control instructions.

Data transfer Instructions:-

→ Data transfer instructions move data from one place in the computer to another without changing the data content.

→ Most common transfers are b/w memory and processor registers, b/w processor registers and I/O and b/w the processor register themselves.

→ The load instruction is mainly used to transfer from memory to a processor register, usually an accumulator

→ Store instruction is used to transfer from a processor register into memory.

→ Move instruction has been used in computers with multiple CPU registers to designate a transfer from one register to another. It also been used for data transfers between CPU registers and memory or between two memory words.

→ Exchange instruction swaps information b/w two registers or a register and a memory word.

Scanned by CamScanner

→ The input and output instructions transfer data among processor register and input or output terminals.

→ The push and pop instructions transfer data between processor registers and a memory stack.

Table: Typical Data Transfer Instructions

Name	Mnemonic
Load	LD
Store	ST
Move	MOV
Exchange	XCH
Input	IN
Output	OUT
Push	PUSH
Pop	POP

Data Manipulation Instructions :-

→ Data Manipulation instructions perform operations on data.

→ It is divided into three basic types :

1. Arithmetic instructions.
2. Logical and bit manipulation instructions.
3. Shift instructions.

Arithmetic Instructions :-

→ The four basic arithmetic operations are addition, subtraction, multiplication and division.

→ Most Computers provide instructions for all four operations

→ Multiplication and division must then be generated by means of software routines.

Scanned by CamScanner

→ The Increment instruction adds 1 to the value stored in a register or memory word.

→ The decrement instruction subtracts 1 from a value stored in a register or memory word.

→ The add, subtract, multiply and divide instructions may be available for different types of data.

→ The data type is assumed to be in the processor registers during execution.

→ The number of bits in any register is of finite length and results are also finite.

Name	Mnemonic
Increment	INC
Decrement	DEC
Add	ADD
Subtract	SUB
Multiply	MUL
Divide	DIV
Add with Carry	ADDC
Subtract with borrow	SUBB
Negate (2's complement)	NEG

Table: Typical Arithmetic Instructions

Logical and Bit Manipulation Instruction :-

→ Logical instructions perform binary operations on strings of bits stored in the registers.

→ They are useful for manipulating individual bits or a group of bits that represent binary-coded information.

Scanned by CamScanner

→ The clear Instruction causes the specified operand to be replaced by 0's.

→ The Complement instruction produces the 1's Complement by inverting all the bits of the operand.

→ The AND, OR and XOR instructions produce the Corresponding logical operations on individual bits of the operands.

→ There are three bit manipulation operations:

a selected bit can be cleared to 0 or Can be set 1 or Can be Complemented.

Name	Mnemonic
clear	CLR
Complement	COM
AND	AND
OR	OR
Exclusive-OR	XOR
Clear Carry	CLRC
Set Carry	SETC
Complement Carry	COMC
Enable Interrupt	EI
Disable interrupt	DI

Table: Logical & Bit Manipulation Instructions.

Shift Instructions:-

→ Instructions to shift the Content of an operand are quite useful in several variations.

→ shifts are operations in which the bits of a word are moved to the left or Right.

→ The bit shifted in at the end of the word determines the type of shift used.

→ Shift Instructions may specify either logical

Scanned by CamScanner

shifts, arithmetic shifts or rotate-type operations.

Name	Mnemonic
Logical shift right	SHR
Logical shift left	SHL
Arithmetic shift right	SHRA
Arithmetic shift left	SHLA
Rotate right	ROR
Rotate left	ROL
Rotate right thro' carry	RORC
Rotate left thro' carry	ROLC

Table: Shift Instructions.

Program Control :-

→ Instructions are always stored in successive memory locations. when processed in the CPU, the instructions are fetched from consecutive memory locations and executed.

→ Each time an instruction is fetched from memory, the program counter is incremented for the next instruction in sequence

→ After the execution of a data transfer or data manipulation instruction, control returns to the fetch cycle with the program counter containing the address of the instruction next in sequence.

→ Branch and Jump instructions may be conditional or unconditional. An unconditional branch instruction causes a branch to the specified address without any condition.

Scanned by CamScanner

The Conditional branch Instruction specifies a Condition such as branch if positive or branch if zero.

Name	Mnemonic
Branch	BR
Jump	JMP
Skip	SKP
Call	CALL
Return	RET
Compare (by subtracting)	CMP
Test (by ANDing)	TST

→ The skip instruction does not need an address field and is therefore a zero address instruction.

→ A conditional skip instruction will skip the next instruction if the condition is met.

→ The call and return instructions are used in conjunction with subroutines.

→ The Compare Instruction performs a subtraction between two operands, but the result of the operation is not retained.

→ The test instruction performs the logical AND of two operands.

Status bits of Interest are the carry bit, the sign bit, a zero indication and an overflow condition.

Scanned by CamScanner

Status Bit Conditions:-

→ It is sometimes convenient to supplement the ALU circuit in the CPU with a status register where status bit conditions be stored for further analysis.

→ Status bits are also called Condition - Code bits or flag bi

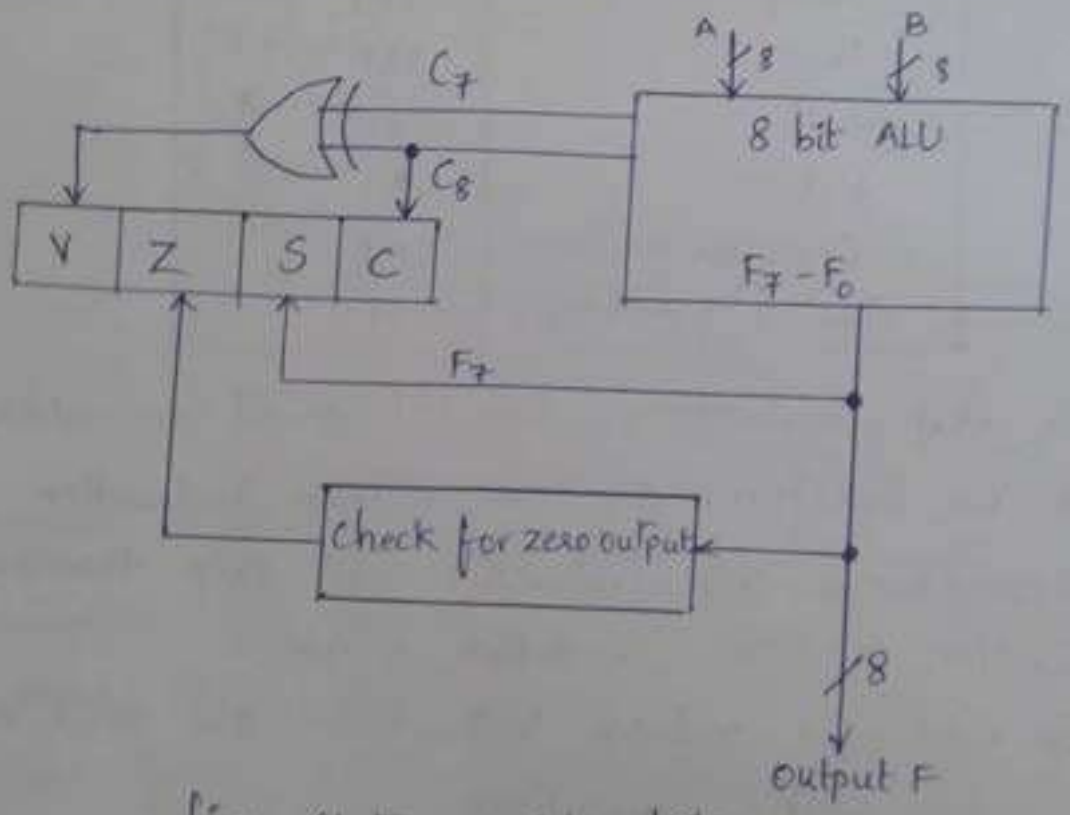


fig:- Status register bits

→ In the above block diagram there are four Status bits are symbolized by C, S, Z and V. The bits are set or cleared as a result of an operation performed in the ALU.

1) Bit C (carry) is set to 1, if the end carry C_8 is 1. It is cleared to 0 if the carry is 0.

2) Bit S (sign) is set to 1, if the highest-order bit F_7 is 1. It is set to 0 if ~~set to~~ the bit is 0

Scanned by CamScanner

3) Bit z (zero) is set to 1 if the output of the ALU Contains all 0's. It is cleared to 0 otherwise.
In other words z=1, if the output is zero and z=0, if the output is not zero.

4) Bit V (overflow) is set to 1 if the exclusives-OR of the last two carries is equal to 1 and cleared to 0. This is the condition for an overflow when negative numbers are in 2's complement.

→ The status bits can be checked after an ALU operation to determine certain relationships that exist between the values of A and B.

→ If bit V is set after the addition of two signed numbers, it indicates an overflow condition.

→ If Z is set after an exclusive OR operation, it indicates that A = B.

→ A single bit A can be checked to determine if it is 0 or 1 by masking all bits except the bit in question and then checking the z status bit.

<u>Conditional Branch Instructions</u> :-

→ The below table lists the most common branch instructions that occur in the processor.

→ Each mnemonic is constructed with the letter B (for branch), when the opposite condition state is used, the letter N (for no) is inserted to define the opposite state.

→ If the stated condition is true, program control is transferred to the address specified by the instruction.

Scanned by CamScanner

Mnemonic	Branch Condition	Tested Condition
BZ	Branch if zero	Z=1
BNZ	Branch if not zero	Z=0
BC	Branch if carry	C=1
BNC	Branch if no carry	C=0
BP	Branch if plus	S=0
BM	Branch if minus	S=1
BV	Branch if overflow	V=1
BNC	Branch if no overflow	V=0.
~~Unsigned Compare Condition (A-B)~~		
BHI		

Subroutine Call and Return :-

→ A subroutine is a self-contained sequence of instruction that performs a given computational task.

→ During the execution of a program, a subroutine may be called to perform its function.

→ Each time a subroutine is called, a branch is executed to the beginning of the subroutine to start executing its set of instructions.

→ After execution of subroutine, ~~it goes back~~ a branch is made back to the main program.

→ The instruction that transfers program control to a subroutine can be known as 'call subroutine' or 'jump to subroutine', 'branch to subroutine', or branch and save address'.

Scanned by CamScanner

→ A call subroutine instruction consists of an operation code together with an address that specifies the beginning of the subroutine.

→ The instruction is executed by performing two operations

(1) The address of the next instruction available in the program counter is stored in a temporary location so that subroutines knows where to return.

(2) Control is transferred to the beginning of the subroutine.

The last instruction of every subroutine, commonly called 'return from subroutine'

Program Interrupt :-

→ The concept of program interrupt is used to handle a variety of problems that arise out of normal program sequence.

→ Program Interrupt refers to the transfer of program control from a currently running program to another service program as a result of an external or internal generated request.

→ After a program has been interrupted and the service routine been executed, the CPU must return to same state that it was when the interrupt occured.

Scanned by CamScanner

→ The state of the CPU at the end of the execute cycle is determined from:

1) The Content of the program Counter.
2) The Content of all processor registers.
3) The Content of certain status Conditions.

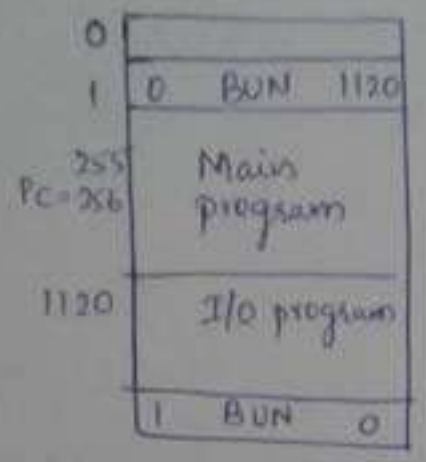
→ The interrupt facility allows the running program to proceed until the input or output device sets its ready flag whenever a flag is set to 1, the Computer Completes the execution of the instruction in progress and then acknowledges the interrupt.

→ The result of this action is that the return address is started in location 0. The instruction in location is then performed, this initiates a service routine for the input or output transfer. The service routine Can be stored in location 1.

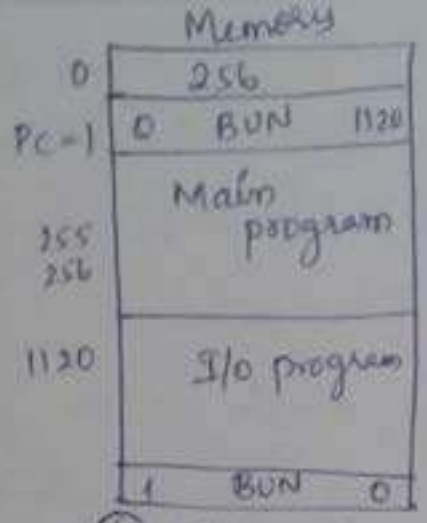
→ The service routine must have instructions to perform the following tasks.

1) Save Contents of processor registers.
2) Check which flag is set.
3) Service the device whose flag is set.
4) Restore contents of processor registers.
5) Turn the interrupt facility on.
6) Return to the running program.

Scanned by CamScanner



(a) Before Interrupt



(b) After interrupt cycle.

Types of Interrupts :-

There are three major types of interrupts that cause a break in the normal execution of a program. They can be classified as :

1. External Interrupts
2. Internal Interrupts
3. Software interrupts.

External Interrupts :-

→ External interrupts come from input-output (I/O) devices, from a timing device, from a circuit monitoring the power supply, or from any other external source.

→ Examples that cause external interrupts are I/O device requesting transfer of data, I/O device finished transfer of data, elapsed time of an event, or power failure.

→ Power failure interrupt may have as its service routine a program that transfers the complete state of the CPU into a memory in the few milliseconds before power ceases.

Scanned by CamScanner

→ External interrupts are asynchronous. External interrupts depend on external conditions that are independent of the program being executed at the time.

2) Internal Interrupts :-

→ Internal interrupts arise from illegal or erroneous use of an instruction or data. Internal interrupts are also called 'traps'.

→ Examples of interrupts caused by internal error conditions are register overflow, attempt to divide by zero, an invalid operation code, stack overflow.
These error conditions usually occur as a result of a premature termination of the instruction execution.

→ The service program that processes the internal interrupt determines the corrective measure to be taken.

Software Interrupts :-

→ A software interrupt is a special call instruction that behaves like an interrupt rather than a subroutine call.

→ It can be used by the programmer to initiate an interrupt procedure at any desired point in the program.

→ The most common use of software interrupt is associated with a supervisor call instruction. This instruction provides means for switching from a CPU user mode to the supervisor mode.

→ When an input or output transfer is required, the supervisor mode is requested by means of a supervisor call instruction.

Scanned by CamScanner

→ This instruction causes a software interrupt that stores the old CPU state and brings in a new PSW that belongs to the supervisor mode.

→ The calling program must pass information to the operating system inorder to specify the particular task requested.

Scanned by CamScanner

Data Representation

Data Types :-

→ The datatypes found in the registers of digital computers may be classified as being one of the following categories:

(1) Numbers used in arithmetic computations.
(2) Letters of the alphabet used in data processing.
(3) Other discrete symbols used for specific purposes.

→ All types of data, except binary numbers, are represented in Computer registers in binary coded form.

→ This is because registers are made up of flip-flops and flip-flop's are two state devices that can store only 1's and 0's.

Number System :-

Radix

- Uses R distinct symbols for each digit.

  Example $A_R = a_{n-1} a_{n-2} \cdots a_1 a_0 . a_{-1} \cdots a_{-m}$

- Radix point (•) separates the integer portion and the fractional portion.

- Example:

  R = 10 Decimal Number system , R = 2 Binary

  R = 8 Octal , R = 16 Hexadecimal

1. Decimal :-

- The decimal number system in everyday use employs the radix 10 system.

Scanned by CamScanner

- The 10 symbols are 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9

Example :-

The string of digits 724.5 is interpreted to represent the quantity

$7 \times 10^2 + 2 \times 10^1 + 4 \times 10^0 + 5 \times 10^{-1}$

that is, 7 hundreds, plus 2 tens, plus 4 units, plus 5 tenths.

→ Every decimal number can be similarly interpreted to find the quantity it represents.

2. Binary :-

→ The binary number system uses the radix 2.

→ The two digit symbols used are 0 and 1. The string of digits 101101 is interpreted to represent the quantity

$1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 45$

→ To distinguish between different radix numbers, the digits will be enclosed in parantheses and the radix of the number inserted as a subscript

for Example :-

To show equality between decimal and binary forty five we will write

$(101101)_2 = (45)_{10}$.

3. Octal :-

→ octal number uses the radix 8.

→ The eight symbols of the octal system are 0, 1, 2, 3, 4, 5, 6 and 7

Scanned by CamScanner

for example, octal 736.4 is converted to decimal as follows:

$$(736.4)_8 = 7\times8^2 + 3\times8^1 + 6\times8^0 + 4\times8^{-1}$$

$$= 7\times64 + 3\times8 + 6\times1 + 4/8$$

$$= (478.5)_{10}$$

Hexadecimal:-

- Hexadecimal number system uses the (radix 16)

→ The 16 symbols of the hexadecimal system are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F

→ Hexadecimal digits, the symbols A, B, C, D, E and F correspond to the decimal numbers 10, 11, 12, 13, 14 and 15 respectively.

→ The equivalent decimal number of hexadecimal F3 is obtained from the following calculation:

$$(F3)_{16} = F\times16+3 = 15\times16+3 = (243)_{10}.$$

Complements:-

→ Complements are used in digital computers for simplifying the subtraction operation and for logical manipulation.

→ There are two types of complements for each base r system:

1) The r's complement
2) The (r-1)'s complement.

Scanned by CamScanner

→ Th. when the value of the base r is substituted in the name, the two types are referred to as the 2's and 1's complement for binary numbers and the 10's and 9's Complement for decimal numbers.

(r-1)'s Complement :-

* 9's Complement :-

→ Given a number N in base r having n digits, the $(r-1)$'s Complement of N is defined as $(r^n-1)-N$ for decimal numbers $r=10$ and $r-1=9$, so that 9's Complement of N is $(10^n-1)-N$

→ Now 10^n represents a number that consists of a single 1 followed by n 0's.

→ 10^n-1 is a number represented by n 9's.

for Example :- with $n=4$ we have $10^4=10000$ and $10^4-1=9999$.

→ It follows that the 9's Complement of a decimal number is obtained by subtracting each digit from 9.

→ for Example The 9's Complement of 546700 is $999999-546700=453299$.

→ The 9's Complement of 12389 is

$$99999-12389=87610.$$

Scanned by CamScanner

* 1's Complement :-

→ for binary numbers, $r = 2$, and $r-1 = 1$, so the 1's Complement of N is $(2^n - 1) - N$.

→ Again 2^n is represented by a binary number that consists of a 1 followed by n 0's.

→ $2^n - 1$ is a binary number represented by n 1's for Example, with $n = 4$, we have $24 = (10000)_2$ and

$(24-1) = (01111)_2$

→ Thus the 1's complement of a binary number is obtained by subtracting each digit from 1.

→ However, the subtraction of a binary digit from 1 causes the bit to change from 0 to 1 or from 1 to 0.

→ Therefore, the 1's complement of a binary number is formed by changing 1's into 0's and 0's into 1's.

for Example :- 1's Complement of 1011001 is

0100110

1's complement of 0001111 is

1110000.

(r's) Complement :-

→ The r's Complement of an n-digit number N in base r is defined as $r^n - N$ for $N \neq 0$ and 0 for $N = 0$.

→ Comparing with the $(r-1)$'s Complement, we note that the r's Complement is obtained by adding 1 to the

Scanned by CamScanner

$(r-1)$'s Complement since $r^n - N = [(r^n - 1) - N] + 1$

* 10's Complement :-

→ Thus the 10's Complement of a decimal is obtained by adding 1 to 9's Complement value.

$$\boxed{\text{9's Complement} + 1 = \text{10's Complement}}$$

Eg:- 10's Complement of 2389 is

9's complement of 2389

$$\begin{array}{r} 9999 \\ -2389 \\ \hline 7610 \\ +\quad 1 \\ \hline 7611 \end{array}$$

- 9's Complement
- 10's Complement

* 2's Complement :-

→ The 2's Complement of binary value can be obtained by adding 1 to the 1's complement value.

Eg:- 2's Complement of 101100 is

1's Complement of 101100 is 010011 - 1's Compl

$$\begin{array}{r} 010011 \\ +1 \\ \hline 010100 \end{array}$$

- 2's Compl

→ 2's Complement notation solves the problem of the relationship between positive and negative numbers, and achieves accurate results in subtractions.

Scanned by CamScanner

Subtraction of Unsigned Numbers in Complement Number System.

→ The subtraction of two n-digit unsigned numbers M-N in base 'r' can be done as follows:

Add the minuend M to the r's Complement of the subtrahend N.

This performs $M + r^n - N = M - N + r^n$.

1) If $M \geq N$, the sum will produce an end carry r^n which is discarded, and what is left is the result M-N.

2) If $M < N$, the sum does not produce an end carry and is equal to $r^n - (N-M)$, which is the r's Complement of (N-M). To obtain the answer in a familiar form, take the r's Complement of the sum and place a negative sign in front.

<u>for Example</u>:- The subtraction 72532 - 13250 = 59282.

The 10's Complement of 13250 is (99999 - 13250 = 86749 (9's Complement) to get 10's Complement add 1 to 86750+1 so) 86750

$$
\begin{array}{lr}
M & = 72532 \\
\text{10's Complement of } N & = +86750. \\
\hline
\text{Sum} & = 159282
\end{array}
$$

Discard the End Carry

So Answer is $\boxed{59282}$.

Scanned by CamScanner

Now Consider an Example with M<N. The subtraction 13250 - 72532 produces negative number 59282. using the procedure with Complements, we have

$$M = 13250.$$
$$\text{10's Complement of } N = +27468$$
$$\text{Sum } 159282$$

Discard the End carry 1.

Answer is negative 59282 = 10's Complement of 40718.

10's Complement of 72532 is

$$\begin{array}{r} 99999 \\ -72532 \\ \hline 27467 \\ +1 \\ \hline 27468 \end{array}$$

→ Subtraction for binary numbers are done with the Complements in the same above manner.

→ Using the two binary numbers X = 1010100 and Y = 1000011, we perform the subtraction X-Y and Y-X using 2's complements;

X-Y:-

$$X = 1010100$$
$$\text{2's Complement } Y = 0111101$$
$$\text{Sum} = 10010001$$

Discard the End carry 2^7

So X-Y = 0010001

2's Complement of Y

$$Y = 1000011$$
$$\text{1's Complement} = 0111100$$
$$+1$$
$$\text{2's compl} = 0111101$$

Y-X:-

$$Y = 1000011$$
$$\text{2's Comple of } X = 0101100$$
$$\text{Sum} = 1101111$$

There is no end carry

Answer is negative 0010001 = 2's complement of 1101111

2' complement of X

$$X = 1010100$$
$$\text{1's complement of } X$$
$$X = 0101011$$
$$\text{2's Comp} \quad +1$$
$$0101100$$

Scanned by CamScanner

Fixed Point Representation :-

→ Positive integers, including zero can be represented ~~by~~ as unsigned numbers

→ However to represent negative integers, we need a notation for negative values.

→ In Ordinary arithmetic, a negative number is indicated by a minus sign and a positive number with plus sign.

→ Because of hardware limitations, Computers must represent everything with 1's and 0's, including the sign of a number.

→ As a Consequence, it is customary to represent the sign with a bit placed in the left most position of the number.

→ Sign bit is equal to 0 for positive and 1 for negative

→ In addition to sign, a number may have a binary (or decimal) point.

→ The position of the binary point is needed to represent fractions, integers or mixed integer-~~fixed~~fraction numbers.

Binary fixed point Representation can be shown below:

$$X = x_n x_{n-1} x_{n-2} \cdots x_1 x_0 \cdot x_{-1} x_{-2} \cdots x_{-m}$$

Sign bit (x_n) : 0 for positive
1 for negative.

Remaining bits $(x_{n-1} x_{n-2} \cdots x_1 x_0 \cdot x_{-1} x_{-2} \cdots x_{-m})$

Scanned by CamScanner

→ The Representation of the binary point in a register is complicated by the fact that it is characterized by a position in the register.

→ There are two ways of specifying the position of the binary point in a register :

1.) By giving it a fixed position.

2) By employing a floating-point representation.

→ The fixed point method assumes that the binary point is always fixed in one position.

→ The two positions most widely used are

(1) A binary point in the extreme left of the register to make the stored number a fraction, and

(2) A binary point in the extreme right of the register to make the stored number an integer.

Integer Representation of Signed Numbers :-

→ when an integer binary number is positive, the sign is represented by 0 and the magnitude by a positive binary number

→ when the number is negative, the sign is represented by 1 but the rest of the number may be represented in one of three possible ways:

1) Signed - magnitude representation.

2) Signed 1's complement representation.

3) Signed 2's Complement representation.

→ The signed-magnitude representation of a negative number consists of the magnitude and a negative sign.

Scanned by CamScanner

→ In the other two representations, the negative number is represented in either the 1's or 2's Complement of its positive value.

→ As an Example, Consider the signed number 14 stored in an 8-bit register.

1. In signed-magnitude Representation | 0001110
2. In signed-1's Complement Representation | 1110001
3. In signed-2's Complement Representation | 1110010

→ The signed-magnitude representation of -14 is obtained from +14 by Complementing only the sign bit

→ The signed-1's Complement representation of -14 is obtained by Complementing all bits of +14, including the sign bit.

→ The signed-2's Complement representation is obtained by taking the 2's Complement of the positive number including the sign bit.

Arithmetic Addition:-

1) Compare their signs.
2) if two signs are the same, ADD the two magnitudes
Lookout for an overflow.
3) If not the same, Compare the relative magnitudes of the numbers and then SUBTRACT the smaller from the larger → need a subtractor to add.
4) Determine the sign of the result.

Scanned by CamScanner

eg:

+6	00000110		-6	11111010 (2's comp of -6)
+13	00001101		+13	00001101
+19	00010011		+7	00000111

+6	00000110		-6	11111010
-13	11110011		-13	11110011
-7	11111001		-19	11101101

In above all four cases, the operation performed is always addition, including the sign bits.

→ Any carry out of the sign bit position is discarded and negative results are automatically in 2's complement form.

Arithmetic Subtraction:-

→ Subtraction of two signed binary numbers can be done by using 2's complement.

→ Take the 2's complement of the subtrahend (including the sign bit) and add it to the minuend (including the sign bit).

→ A carry out of the sign bit position is discarded.

→ The subtraction operation can be changed to an addition operation if the sign of the subtrahend is changed.

Scanned by CamScanner

→ This is demonstrated by the following relationship:

$$(\pm A) - (+B) = (\pm A) + (-B)$$

$$(\pm A) - (-B) = (\pm A) + (+B)$$

→ But changing a positive number to a negative number is easily done by taking its 2's complement.

→ The reverse is also true because the complement of a negative number in complement form produces the equivalent positive number.

→ Consider the subtraction of $(-6) - (-13) = +7$

→ In binary with eight bits this is written as

11111010 - 11110011.

→ The subtraction is changed to addition by taking the 2's complement of the subtrahend (-13) to give (+13).

→ In binary this is 11111010 + 00001101 = 100000111

→ By removing the end carry, we obtain the correct answer 00000111 (+7).

<u>Overflow</u>:-

→ When two numbers of n digits each are added and the sum occupies n+1 digits, we say that an overflow occured.

→ An overflow is a problem in digital computers because the width of registers is finite.

→ A Result that contains n+1 bits cannot be accommodated in a register with a standard length of n bits.

Scanned by CamScanner

→ for this reason, many computers detect the occurence of an overflow, and when it occurs, a corresponding flip-flop is set which can then be checked by the user.

→ An overflow cannot occur after an addition if one number is positive and the other is negative, since adding a positive number to a negative number produces a result that is smaller than the larger of the two original numbers.

→ An overflow may occur if the two numbers added are either positive or both negative.

Consider the following Example:

→ Two signed binary numbers, +70 and +80 are stored in two 8-bit registers.

→ The range of numbers that each register can accomodate is from binary +127 to binary -128.

→ Since the sum of the two numbers is +150, it exceeds the capacity of the 8-bit register.

→ This is true if the numbers are either positive or both negative.

→ The two additions in binary are shown below together with the last two carries

Carries	0 1	Carries:	1 1
+70	0 1000110	-70	1 0111010
+80	0 1010000	-80	1 0110000
+150.	1 0010110	-150.	1 0110000

Scanned by CamScanner

→ Note that the 8-bit result that should have been positive has a negative sign bit and the 8-bit result that should have been negative has a positive sign bit.

→ If, however, the carry out of the sign bit position is taken as the sign bit of the result, the 9-bit answer so obtained will be correct.

→ Since the answer cannot be accommodated within 8-bits we say that an overflow occured.

→ An overflow condition can be detected by observing the carry into the sign bit position and the carry out of the sign bit position.

→ If these two carries are not equal, an overflow condition is produced.

→ This is indicated in the Examples where the two carries are explicitly shown.

→ If the two carries are applied to an exclusive-OR gate, an overflow will be detected when the output of the gate is equal to 1.

Floating-Point Representation

→ The flo

Scanned by CamScanner

Floating-Point Representation

→ The floating-point representation of a number has two parts.

→ The first part represents a signed, fixed-point number called the mantissa.

→ The second part designates the position of the decimal (or binary) point and is called the exponent.

→ The fixed-point mantissa may be a fraction or integer. for Example:, the decimal number +6132.789 is represented in floating-point with fraction and an exponent as follows:

Fraction	Exponent
+0.6132789	+04

→ floating point is always interpreted to represent a number in the following form:

$$m \times r^e$$

m - mantissa } are physically represented
e - exponent } in the register (including their signs).

→ The radix r and the radix-point position of the mantissa are always assumed.

→ A floating-point binary number is represented in a similar manner except that it uses base 2 for the exponent.

for Example, the binary number +1001.11 is represented with a 8-bit fraction and 6-bit exponent as follows:

Scanned by CamScanner

Fraction	Exponent
01001110	000100

→ The fraction has a 0 in the leftmost position to denote positive.
→ The binary point of the fraction follows the sign bit but is not shown in the register.
→ The exponent has the equivalent binary number +4 (i.e., $(000100)_2 = (4)_{10}$)
→ The floating point number is equivalent to :

$$m \times 2^e = +(.1001110) \times 2^{+4}$$

<u>Normalization</u> :-

→ A floating-point number is said to be normalized if the most significant digit of the mantissa is nonzero. for Example, the decimal number 350 is normalized but 00035 are not.
→ Regardless of where the position of the radix point is assumed to be in the mantissa, the number is normalized only if its leftmost digit is nonzero.
<u>for Example</u> :- The 8-bit binary number 00011010 is not normalized because of the three leading 0's.
→ The number can be normalized by shifting it three positions to the left and discarding the leading 0s to obtain 11010000.
→ The three shifts multiply the number by $2^3 = 8$
→ To keep the same value for the floating point number, the exponent must be subtracted by

Scanned by CamScanner

normalized because it does not have a nonzero digit.

→ It is usually represented in floating-point by all 0's in the mantissa and exponent.

→ Arithmetic operations with floating-point numbers are more complicated than arithmetic operations with fixed-point numbers and their execution takes longer and requires more complex hardware.

→ However, floating point representation is a must for scientific computations because of the scaling problems involved with fixed-point computations.

→ Many computers and all electronic calculators have the built-in capability of performing floating-point arithmetic operations.

Byte Format:

Byte 1

$$\underbrace{SEEEE}_{\text{Exponent}} \underset{\text{Binary point}}{\bullet} \underbrace{IMMMMMMM\ MMMMMMMMM\ MMMMMMMM}_{\text{Mantissa}}$$

S = sign of Mantissa, E = Exponent bits in 2's complement

M = Mantissa Bits

Examples

$13 = 1101 = 0.1101 \times 2^4$

$= 00000100\ 11010000\ 00000000\ 00000000$

$-17 = -10001 = -0.10001 \times 2^5$

$= 10000101\ 10001000\ 00000000\ 00000000$

$-0.125 = -0.001 = -.1 \times 2^{-2}$

$= 11111110\ 10000000\ 00000000\ 00000000$

Scanned by CamScanner

Computer Arithmetic

Introduction:-

→ Arithmetic instructions in digital Computers manipulate data to produce results necessary for the solution of Computational problems.

→ The four basic arithmetic operations are addition, subtraction, multiplication and division.

→ An arithmetic processor is the part of a processor unit that executes arithmetic operations.

→ The data type reside in processor registers during the execution of an arithmetic instruction is specified in the definition of the instruction.

→ An arithmetic instruction may specify binary or decimal data, and in each case the data may be in fixed-point or floating-point form.

→ fixed-point numbers may represent integers or fractions.

→ The designer must be thoroughly familiar with the sequence of steps to be followed in order to carry out the operation and achieve a correct result.

→ The solution to any problem that is stated by a finite number of well-defined procedural steps is called an 'algorithm'.

→ An 'algorithm' will contain a number of procedural steps which are dependent on results of previous steps

→ ~~The~~ A convenient method for presenting algorithms is a 'flowchart'.

Scanned by CamScanner

We consider addition, subtraction, multiplication and division for the following types of data:

1. Fixed-point binary data in signed-magnitude representation
2. Fixed-point binary data in signed-2's complement representation.
3. Floating-point binary data.
4. Binary-coded decimal (BCD) data.

Addition and Subtraction

Three ways of representing negative fixed-point binary numbers:

- Signed Magnitude
- Signed 1's Complement
- Signed 2's Complement.

→ Most computers use the signed-2's complement for arithmetic operations with integers.

→ for floating-point operations, most computers use the signed-magnitude representation.

Addition and Subtraction with Signed-Magnitude Data:-

→ We designate the magnitude of the two numbers is familiar because it is used in by A and B. When signed numbers are added or subtracted, we find that there are eight different conditions to consider, depending on the sign of the numbers and the operation performed.

Scanned by CamScanner

Addition (Subtraction) algorithm:-

→ when the signs of A and B are identical (different) add the two magnitudes and attach the sign of A to the result.

→ when the signs of A and B are different (identical) Compare the magnitudes and subtract the smaller number from the larger.

→ choose the sign of the result to be the same as A if A>B or the Complement of the sign of A if A<B.

Hardware Implementation:-

→ To implement the two arithmetic operations with hardware, it is first necessary that the Two numbers be stored in registers.

→ Let A and B be two registers that hold the magnitudes of the numbers and A_s and B_s be Two flip-flops that hold the Corresponding signs.

→ first a parallel-adder is needed to perform the microoperation A+B, Comparator circuit for A>B, A=B or A<B. Two parallel-subtractor circuits are needed to perform the microoperations A-B and B-A.

→ The sign relationship can be determined from an exclusive-OR gate with A_s and B_s as inputs.

Scanned by CamScanner

Block Diagram for signed-magnitude addition and Subtraction :-

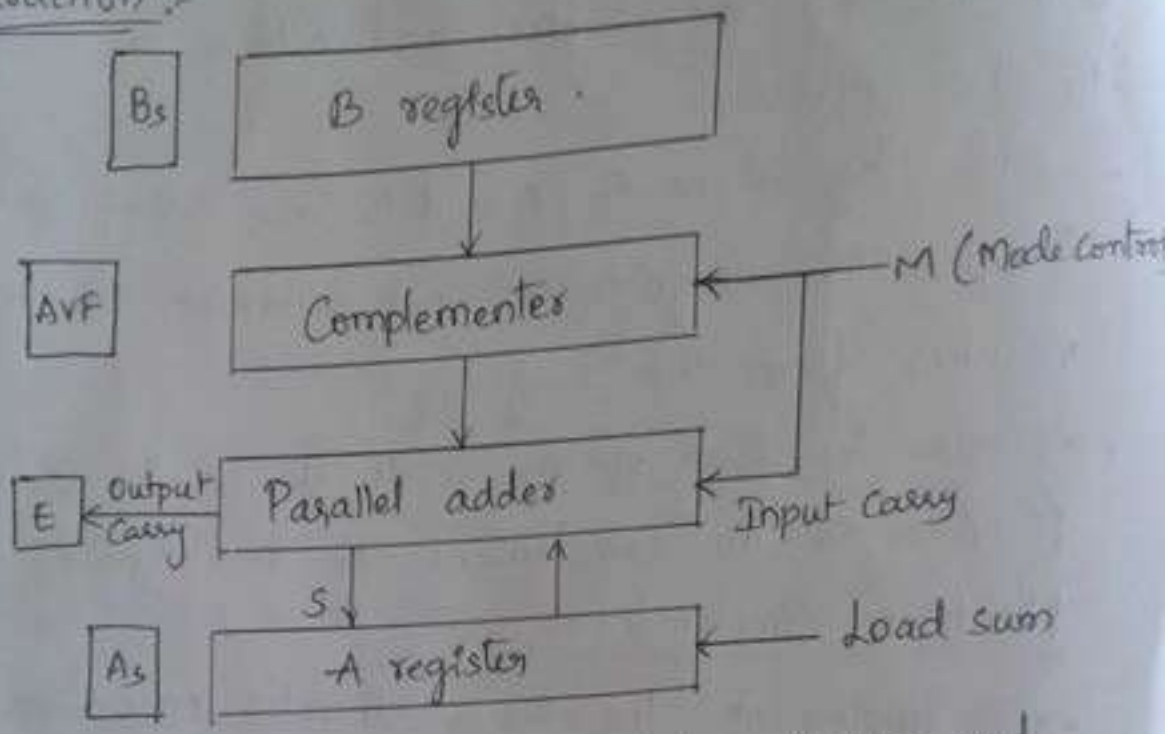


fig:- Hardware for signed-magnitude addition and subtraction.

-> It consists of registers A and B and sign flip-flops A_s and B_s

-> Subtraction is done by adding A to the 2's complement of B.

-> The output carry is transferred to flip-flop E.

-> The Add Overflow flipflop (AVF) holds the overflow bit when A and B are added.

-> The addition of A+B is done through the parallel adder.

-> The S(sum) output of the adder is applied to the input of A register.

Scanned by CamScanner

→ The Complementer provides an output of B or Complement B.

→ The Complementer Consists of XOR gates and parallel adder Consists of full adder circuits.

→ The M is applied as Input carry to the adder, when M=0, the o/p of B is transferred to the adder and the input carry is 0, the output of the adder is equal to the sum A+B.

When M=1, the 1's Complement of B is applied to the adder, the input carry=1 and output S=A+B+1

Hardware Algorithm

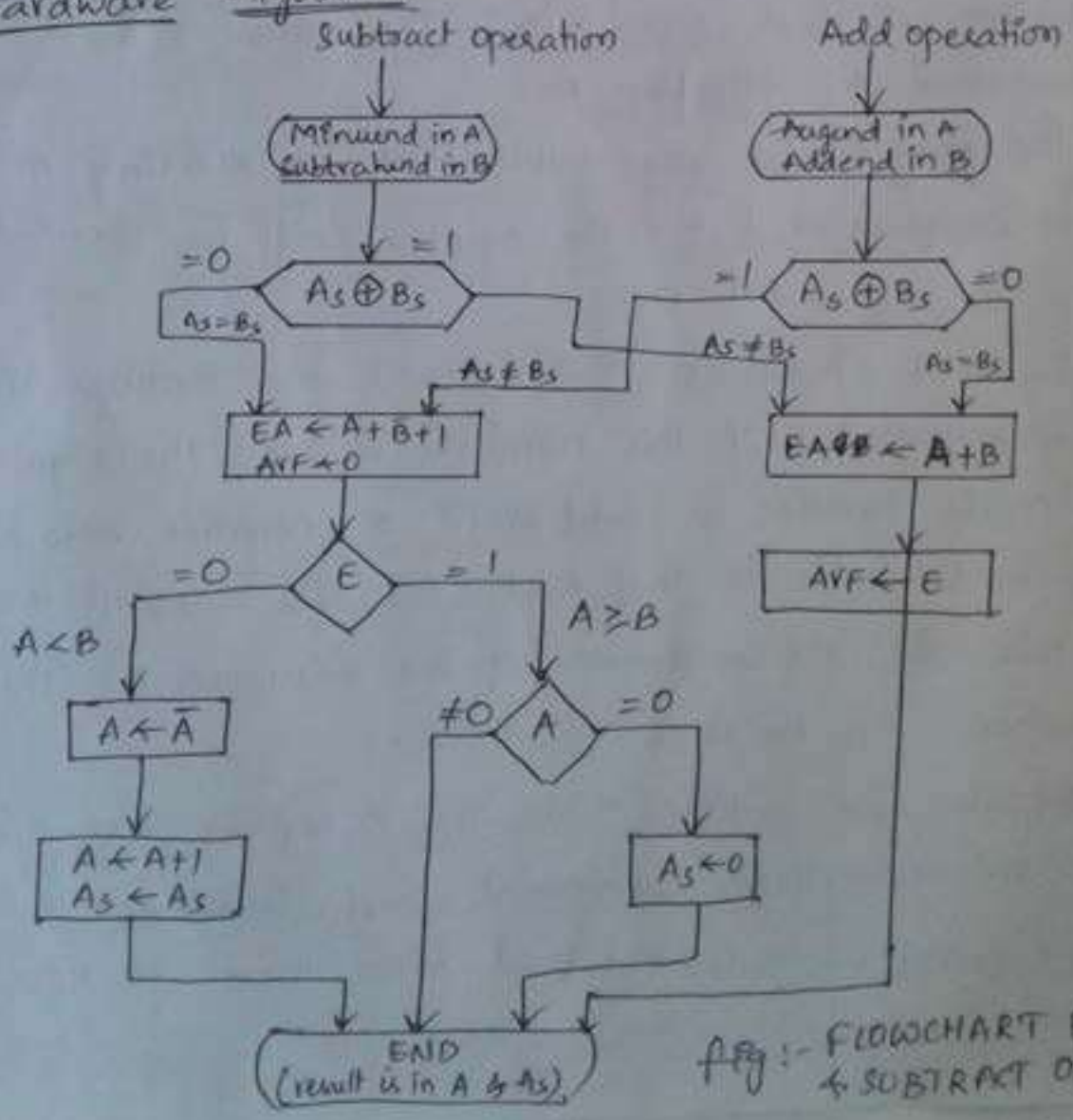


fig:- FLOWCHART FOR ADD & SUBTRACT OPERATION

Scanned by CamScanner

→ In the flowchart, the two signs A_s and B_s are compared by an exclusive OR gate

if the output of the gate is 0 the signs are identical
if it is 1, the signs are different.

→ For an add operation, identical signs dictate that the magnitudes be added

→ for a subtract operation, different signs dictate that the magnitudes be added.

→ The magnitudes are added with a microoperation $EA \leftarrow A+B$, where EA is a register that combines E & A.

→ The carry in E after the addition constitutes an overflow if it is equal to 1. The value of E is transferred to flip flop AVF.

→ The magnitudes are subtracted by adding A to the 2's complemented B. No overflow will be occured so AVF is cleared to 0.

→ 1 in E indicates that $A >= B$ and number in A is correct result. If this number is zero, the sign A must be made positive to ~~add~~ avoid a negative zero.

→ 0 in E indicates that $A < B$. For this case it is necessary to take the 2's complement of the value in A. The operation can be done is $A \leftarrow \bar{A}+1$

→ However, we assume that the A register has circuits for microoperations complement and increment, so the 2's complement is obtained from these two microoperations

Scanned by CamScanner

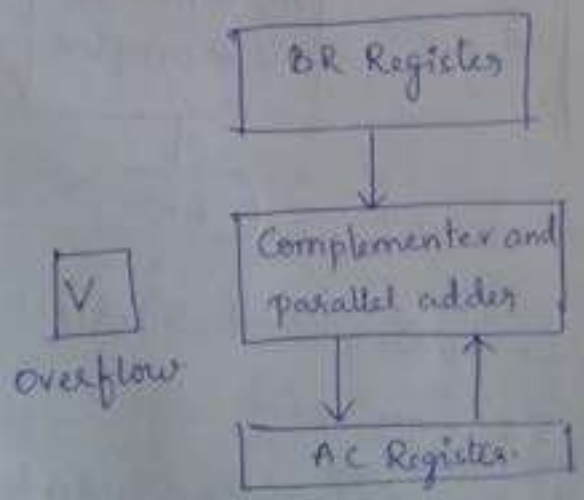
→ In the other paths of the flowchart, the sign of the result is the same as the sign of A. So no change in A is required. However when A<B, the sign of the result is the complement of the original sign of A. It is then necessary to complement A to obtain correct sign.

→ The final result is found in register A and its sign in As. The value in AVF is overflow indication and the final value of E is immaterial.

Addition and subtraction with signed 2's Complement Data

→ The addition of two numbers in signed 2's complement form consists of adding the numbers with the sign bits treated the same as the other bits of the number.

→ The subtraction consists of first taking the 2's complement of the subtrahend and then adding it to the minuend.



→ We name A Register from previous block diagram for add and subtract operations as AC and the B register BR.

→ The leftmost bit in AC and BR represents the sign

bits of the numbers.

→ The two sign bits are added and subtracted together with the other bits in the complementer and parallel adder.

→ The overflow flip flop V is set to 1, if there is an overflow.

→ The output carry in this case is discarded.

Algorithm

→ The algorithm for adding and subtracting two binary numbers in signed 2's complement representation.

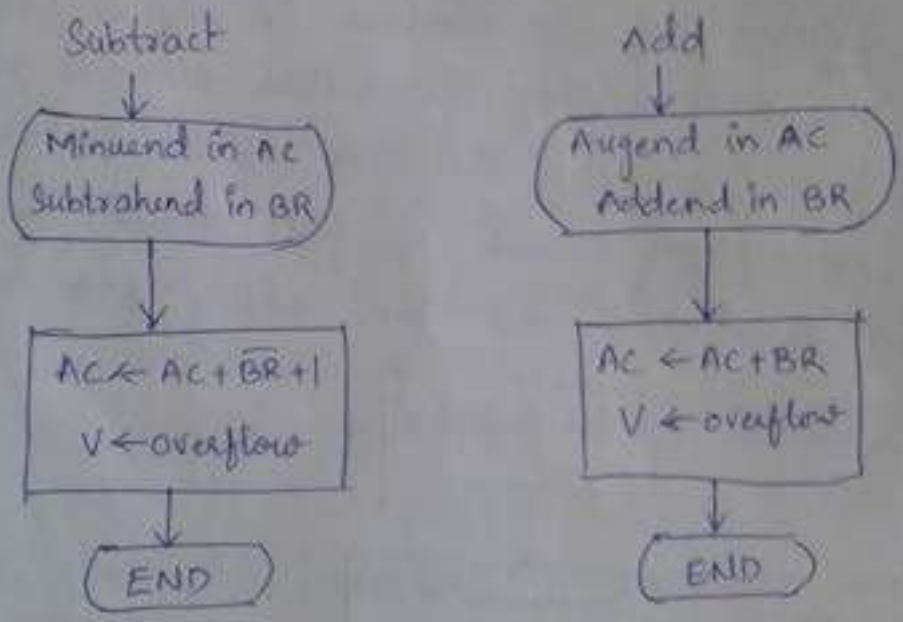


fig:- Algorithm for adding and subtracting numbers in signed 2's complement.

→ The sum is obtained by adding the contents of AC and BR.

→ The overflow bit V is set to 1 if the exclusive OR of the last two carries is 1, and it is cleared to 0 otherwise.

Scanned by CamScanner

→ The subtraction operation is accomplished by adding the content of AC to the 2's Complement of BR.

→ ~~the~~ Taking the 2's Complement of BR has the effect of changing a positive number to negative and vice versa

→ An overflow must be checked during this operation because the two numbers are added could have the same sign.

Multiplication Algorithms :-

→ Multiplication of two-fixed point binary numbers in signed-magnitude is represented by a process of successive shift and add operations.

Eg :-

```
  23          10111     Multiplicand
  19        x 10011     Multiplier
 ----       -------
             10111
            10111   +
           00000
          00000
         10111
         ---------
         110110101     product
```

→ If the multiplier bit is a 1, the multiplicand is copied down, otherwise zero's are copied down.

→ The numbers copied down in successive lines are shifted one position to the left from the previous ~~no~~ numbers. Finally, the numbers are added and their sum forms the product.

→ The sign of the product is determined from the signs of multiplicand and multiplier.

Scanned by CamScanner

Hardware Implementation for signed-Magnitude Data

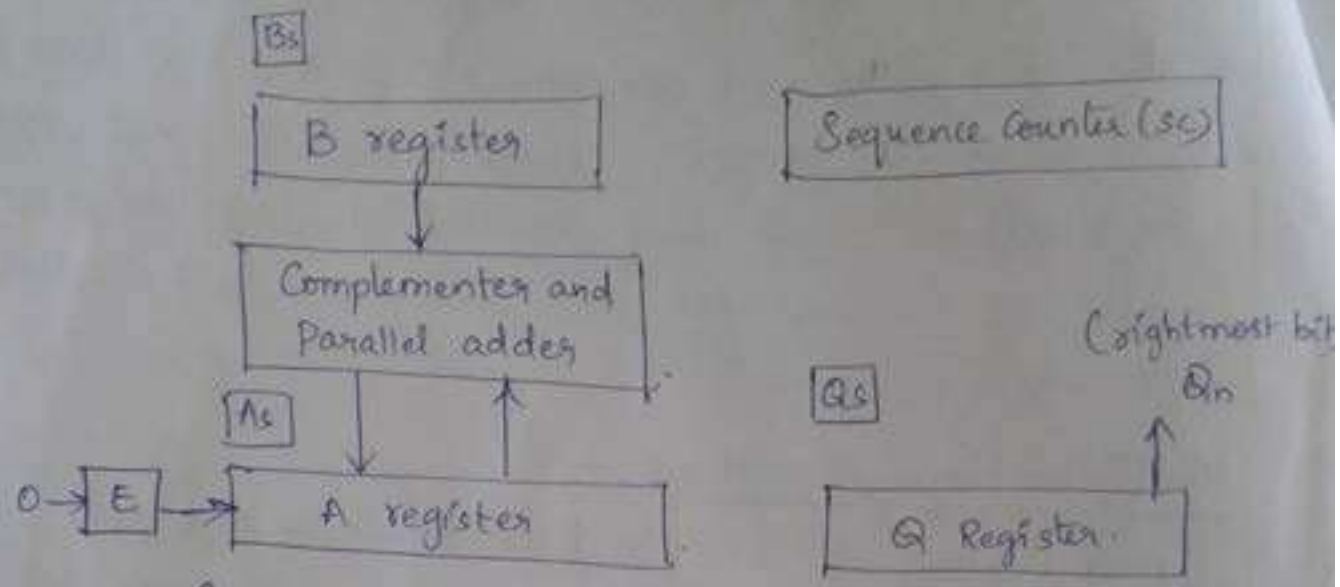


fig: Hardware for multiply operation.

→ The multiplier is stored in the Q register and its sign in Qs.

→ The sequence Counter (SC) is initially set to a number equal to the number of bits in multiplier.

→ The counter is decremented by 1 after forming each partial product.

→ when the content of the counter reaches zero, the product is formed and the process stops.

→ Initially, the multiplicand is in register B and the multiplier in Q.

→ The sum of A and B forms a partial product which is transferred to the EA register.

→ Both partial product and multiplier are shifted to the right.

→ Right is denoted with the statement shr EAQ denotes right shift.

Scanned by CamScanner

→ Least significant bit of A is shifted to the most significant position of Q, the bit from E is shifted into the most significant position of A, and 0 is shifted into E.

Hardware Algorithm

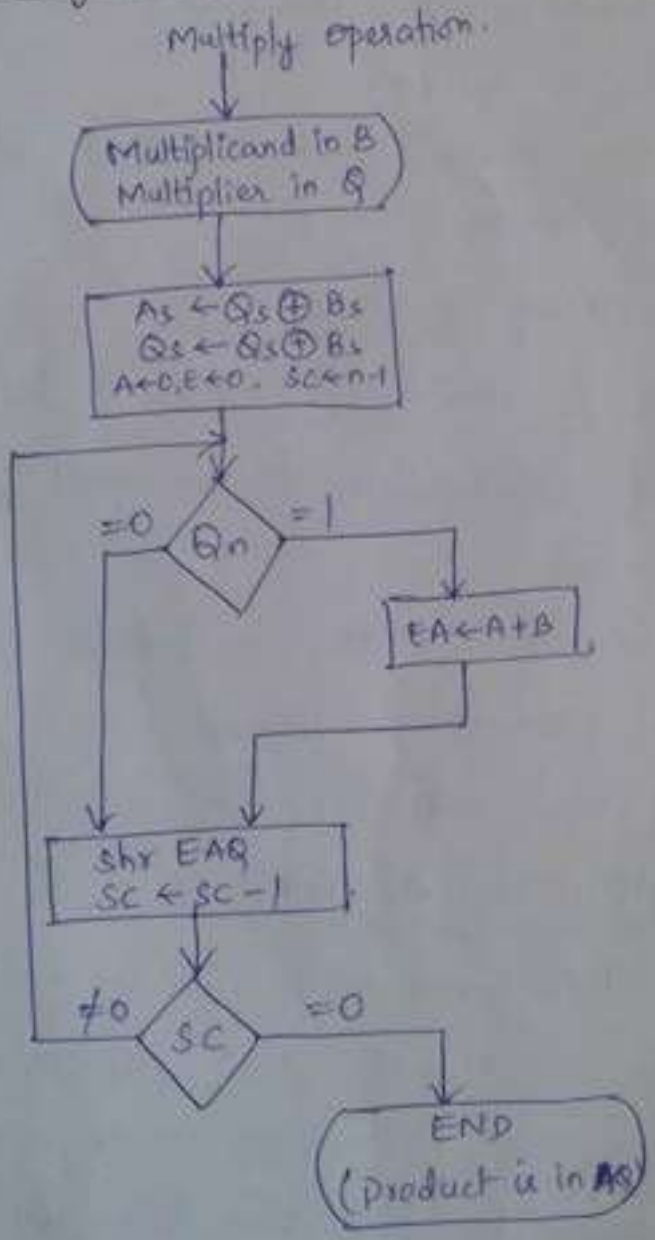


fig:- flowchart for multiply operation.

→ Initially the multiplicand is in B and the multiplier in Q. and corresponding signs is in B_s and Q_s.

Scanned by CamScanner

→ The signs are compared, and both A and Q are set to correspond to the sign of the product.

→ A & E registers are cleared and sequence counter is set to a number equal to number of bits of the multiplier.

→ After the initialization, the low order bit of the multiplier in Q_n is tested.

→ If it is 1, the multiplicand in B is added to the present partial product in A.

→ If it is 0, nothing is done.

→ Register EAQ is then shifted once to the right.

→ The sequence counter is decremented by 1 and its new value checked.

→ If it is not equal to zero, the process is repeated and a new partial product is formed.

→ The process stops when SC = 0.

Booth Multiplication Algorithm :-

→ Booth algorithm gives a procedure for multiplying binary integers in signed - 2's complement representation.

→ It operates on the fact that strings of 0's in the multiplier require no addition but just shifting and a string of 1's in the multiplier require ~~no additions~~ ~~but just shifting~~. ~~and a str~~ from ~~one~~ bit weight 2^k to weight 2^m can be treated as $2^{k+1} - 2^m$.

Scanned by CamScanner

Suppose the binary number 001110 (+14) has a string 1's from 2^3 to 2^1 ($k=3, m=1$). The number can be represented as $2^{k+1} - 2^m = 2^4 - 2^1 = 16 - 2 = 14$. Therefore, the multiplication M×14 M is Multiplicand and 14 the multiplier

$$M \times 2^4 - M \times 2^1.$$

→ Thus the product can be obtained by shifting the binary multiplicand M four times to the left and subtracted left once.

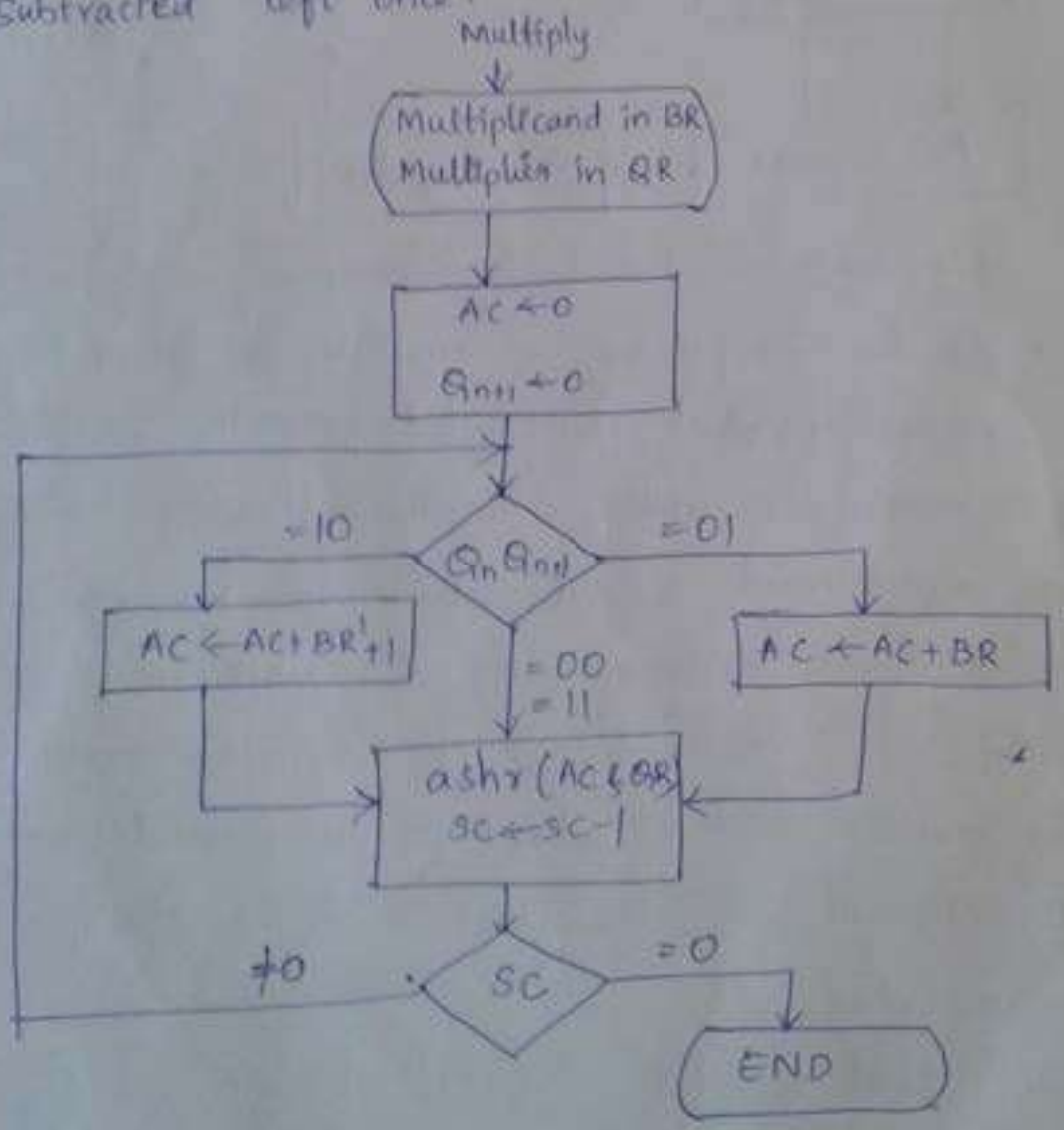


fig:- flowchart for Booth algorithm for multiplication of signed 2's complement numbers.

Scanned by CamScanner

→ As in all multiplication schemes, booth algorithm requires examination of the multiplier bits and shift of partial product.

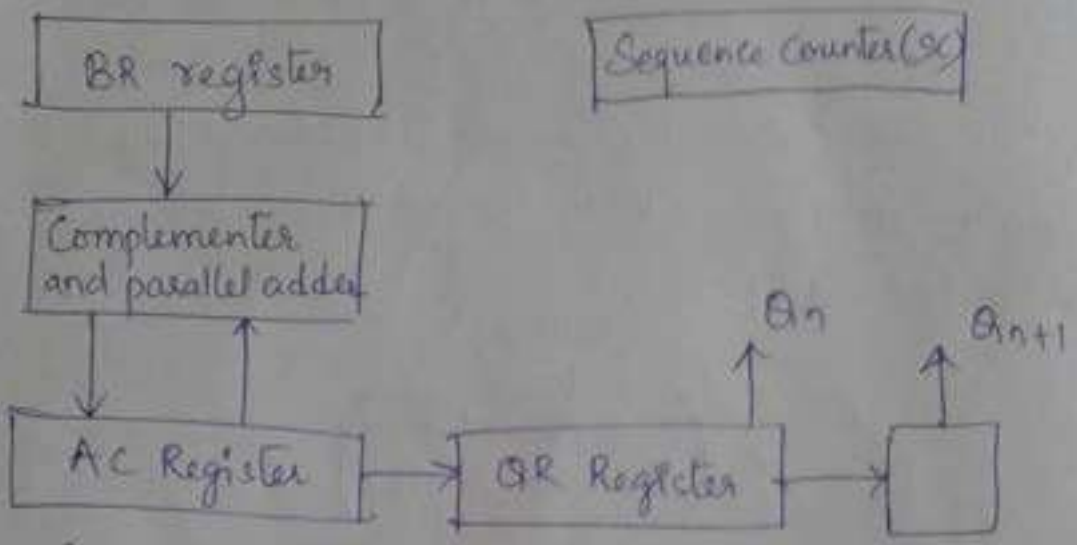


fig: Hardware for Booth algorithm.

→ Prior to the shifting, the multiplicand may be added to the partial product, subtracted from the partial or left unchanged according to the following rules:

1) The multiplicand is subtracted from the partial product upon encountering the first least significant 1 in a string of 1's in the multiplier.
2) The multiplicand is added to the partial product upon encountering the first 0 in the string 0's in the multiplier.
3) The partial product does not change when multiplier bit is identical to the previous multiplier bit.

→ The algorithm work for positive and negative multipliers in 2's complement representation.

Scanned by CamScanner

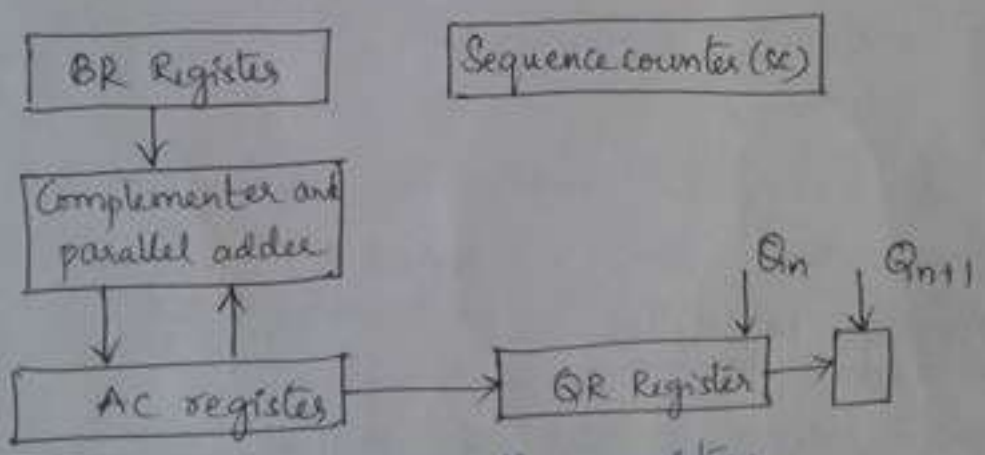
→ The two bits of the multiplier in Q_n and Q_{n+1} are inspected.

→ If the two bits are equal to 10, it means that the first 1 in the string of 1's has been encountered.

→ If the two bits are equal to 01, it means that the first 0 in a string of 0's has been encountered. This requires the addition of the multiplicand to the partial product in AC.

→ When the two bits are equal, the partial product does not change.

Block Diagram for Booth Algorithm.



→ AC, BR and QR are the registers.

→ Q_n designates the least significant bit of the multiplier in register QR.

→ AC and the appended bit Q_{n+1} are initially cleared to 0 and SC is set to number n equal to the number of bits in the multiplier.

→ The Q_n and Q_{n+1} of multiplier bits are inspected.

→ If the two bits are equal to 10, it means the

Scanned by CamScanner

first 1 in a string of 1's has been encountered.
→ This requires a subtraction of the multiplicand from the partial product in AC.
→ If the two bits are 01, it means that first 0 in the strings of 0's has been encountered.
→ This requires addition of the multiplicand to the partial product in AC.
→ When two bits are equal, the partial product does not change.
→ Next step is to shift right the partial product and the multiplier (including bit Q_{n+1}). This is an arithmetic shift right (ashr) operation which shifts AC and QR to the right and leaves the sign bit in AC unchanged.
→ The sequence counter is decremented and the computational loop is repeated n times.

Array Multiplier

→ The above method of multiplication is a sequential operation that requires a sequence of add and shift microoperations.
→ The multiplication can be done with one microoperation by means of a combinational circuit that forms the product bits all at once.
→ This is the fast way of multiplying since it takes the time for the signals to propagate through the gates that form the multiplication array

Scanned by CamScanner

$$
\begin{array}{cccc}
 & & b_1 & b_0 \\
 & & a_1 & a_0 \\
\hline
 & & a_0b_1 & a_0b_0 \\
 & a_1b_1 & a_1b_0 & \\
\hline
c_3 & c_2 & c_1 & c_0 \\
\hline
\end{array}
$$

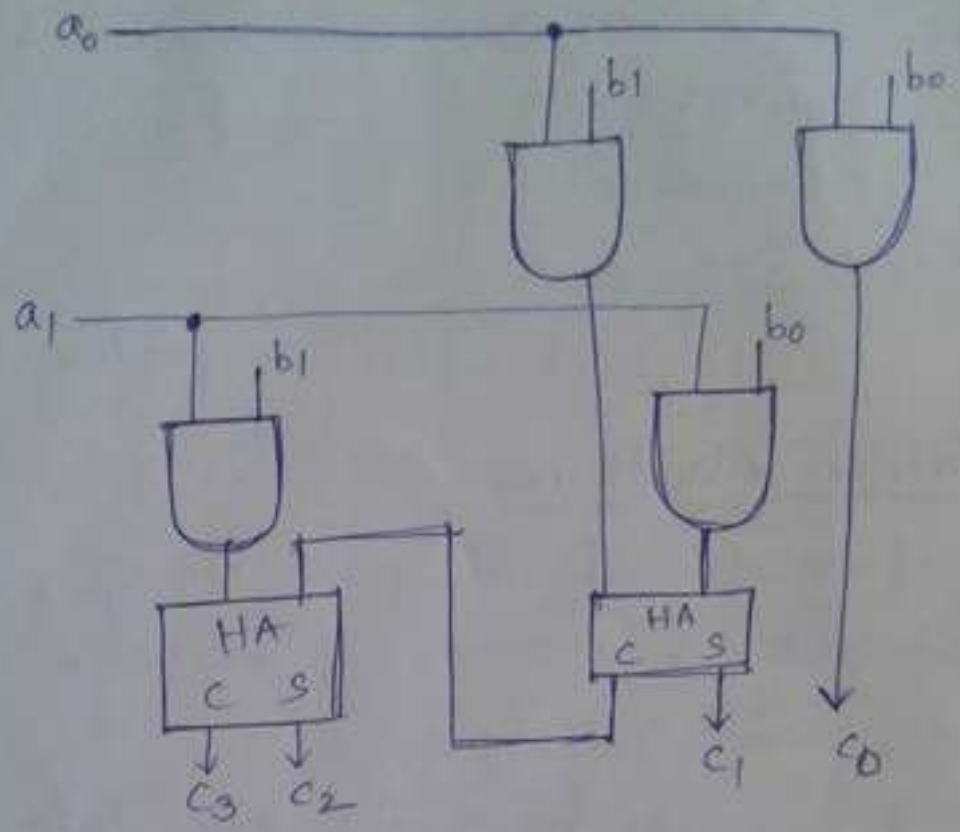


fig 2 bit by 2-bit array multiplier.

→ The multiplicand bits are b_1 and b_0 and multiplier bits are a_1 and a_0 and the product is $c_3c_2c_1c_0$.

→ The first partial product is formed by multiplying a_0 by b_1, b_0. The multiplication of two bits such as a_0 and b_0 produces a 1 if both bits are 1; otherwise 0, which is identical to AND operation and can be implemented using AND gates.

→ The second partial product is formed by a_1 with b_1 b_0 and is shifted one position left.

→ Now both the partial products are added using two Half adders circuits.

Note :- The least significant bit of the product does not go through an adder since it is formed by output of AND gate.

Scanned by CamScanner

→ A Combinational circuit binary multiplier with more bits can be constructed in a similar fashion.

→ for j multiplier bit
k multiplicand bits
we need j×k AND gates
(j-1) k-bit adders to produce product j+k bits

Division Algorithms :-

→ Division of two fixed-point binary numbers in signed-magnitude is a process of successive compare, shift, and subtract operations.

→ Binary division is simpler than decimal division because the Quotient digits are either 0 or 1.

division process

→ The divisor B consists of five bits and the dividend A of 10 bits.

→ The 5 MSB's of the dividend are compared with the divisor

→ Since the 5 bit number is smaller than B, now compare 6 MSB bits of A with B.

→ So, now 6 bit number is greater than B, so we place 1 for the quotient bit in the sixth position above the dividend.

→ The divisor is then shifted once to the right and subtracted from the dividend.

→ The difference is called a partial remainder

Scanned by CamScanner

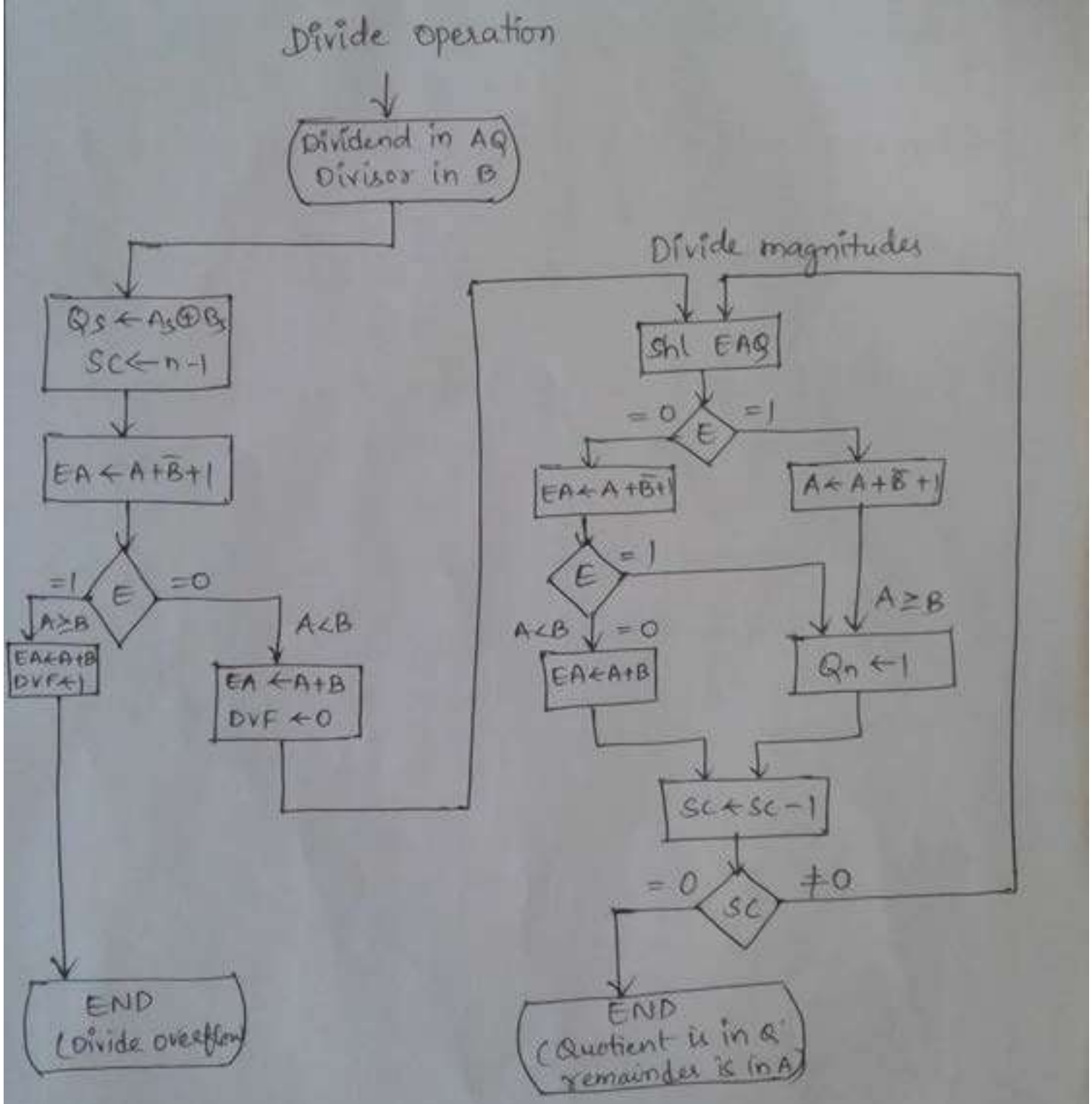


fig:- flowchart for divide operation.

Scanned by CamScanner

→ The process is continued by comparing a partial remainder with the divisor.

→ If the partial remainder is greater than or equal to the divisor, the quotient bit is equal to 1.

→ The divisor is then shifted right and subtracted from the partial remainder.

→ If the partial remainder is smaller than the divisor, the quotient bit is 0 and no subtraction is needed.

→ Result will be quotient and a remainder.

```
                  11010
Divisor:
B = 10001 ) 0111000000
            01110
            011100
          -  10001
            010110
             10001
            001010
             010100
              10001
             000110
              00110
```

Quotient = Q

Dividend = A

5 bits of A < B, quotient has 5 bits

6 bits of A ≥ B

Shift right B and subtract; Enter 1 in Q

7 bits of remainder ≥ B

Shift right B and subtract, Enter 1 in Q

Remainder < B; Enter 0 in Q, shift right B

Remainder ≥ B

Shift right B and subtract; Enter 1

Remainder < B; enter 0 in Q

Final remainder.

Hardware Implementation for Signed-Magnitude Data

→ The hardware for implementing the division operation is identical to that required for multiplication.

→ The divisor is stored in B register and double-length dividend is stored in registers A and Q.

→ The dividend is shifted to the left and the divisor is subtracted by adding the its 2's complement value.

Scanned by CamScanner

If E=1, it signifies that $A \geq B$. A quotient bit 1 is inserted into Q_n and the partial remainder is shifted to the left to repeat the process.

→ If E=0, it signifies that $A < B$ so the quotient in Q_n remains a 0.

→ The value of B is then added to restore the partial remainder in A to its previous value.

→ The partial remainder is shifted to the left and the process is repeated again until all five quotient bits are formed.

Divisor B = 10001 $\qquad$ $\bar{B}+1 = 01111$

	E	A	Q	SC
Dividend:		01110	00000	5
Shl EAQ	0	11100	00000	
add B+1		01111		
E=1	1	1 01011		
set Q_n=1		01011	00001	4
Shl EAQ	0	10110	00010	
Add $\bar{B}+1$		01111		
E=1	1	00101		
set Q_n=1	1	00101	00011	3
Shl EAQ		01010	00110	
Add $\bar{B}+1$		01111		
E=0; leave Q_n=0	0	11001	00110	
Add B		10001		
Restore remainder	1	01010		2

Scanned by CamScanner

	E	A	Q	SC
Shl EAQ	0	10100	01100	
Add $\bar{B}+1$		01111		
E = 1; ~~leave~~	1	00011	~~10110~~	
Set $Q_n = 1$	1	00011	01101	1
Shl EAQ	0	00110	11010	
Add $\bar{B}+1$		01111		
E = 0; leave $Q_n = 0$	0	10101	11010	
Add B		10001		
Restore remainder	1	00110	11010	0
Neglect E				
Remainder in A :		00110		
Quotient in Q :			11010	

Eg:- Binary division with digital hardware

Hardware Algorithm :-

→ The hardware divide algorithm consists

→ The dividend is in A and Q and the divisor in B.

→ The sign of the result is transferred into Q_s to be part of the quotient.

→ A constant is set into the sequence counter SC to specify the number of bits in the quotient.

→ The division of the magnitudes starts by shifting the dividend in AQ to the left with the high-order bit shifted into E.

→ If the bit is shifted into E is 1, we know that

Scanned by CamScanner

EA > B because EA consists of a 1 followed by n-1 bits while B consists of only n-1 bits.

→ In this case, B must be subtracted from EA and inserted into Q_n for the quotient bit.

→ Since register A is missing the high-order bit of the dividend (which is in E), its value is $EA - 2^{n-1}$

→ Adding to this value the 2's complement of B results in

$$\left(EA - 2^{n-1}\right) + \left(2^{n-1} - B\right) = EA - B$$

→ If the shift-left operation inserts a 0 into E, the divisor is subtracted by adding its 2's complement value and the carry is transferred into E.

→ If E = 1, it signifies that $A \geq B$ therefore Q_n is set to 1

→ If E = 0, it signifies that $A < B$ and the original number is restored by adding B to A.

Other Algorithms :-

Restoring Method :-

→ The hardware we just described is called the 'restoring method' because the partial remainder is restored by adding the divisor to the negative difference.

→ Two other methods are available for dividing numbers, the 'Comparision method' and the

Scanned by CamScanner

non-restoring method.

→ In the Comparison method A and B are compared prior to the subtraction operation.

→ Then if $A \geq B$, B is subtracted from A. If $A < B$ nothing is done.

→ The partial remainder is shifted left and the numbers are compared again.

Floating-Point Arithmetic Operations:-

Addition and subtraction:-

→ During addition or subtraction, the two floating-point operands are in AC and BR.

→ The sum or difference is formed in the AC.

→ The algorithm is divided into four consecutive parts:

1) Check for zeros.
2) Align the mantissas.
3) Add or subtract the mantissas.
4) Normalize the result.

→ The flowchart for adding or subtracting two floating-point binary numbers is shown below.

→ If BR is equal to zero, the operation is terminated with the value in the AC being the result.

→ If AC = 0, we transfer the content of BR into AC and also complement its sign if the numbers are to be subtracted.

Scanned by CamScanner

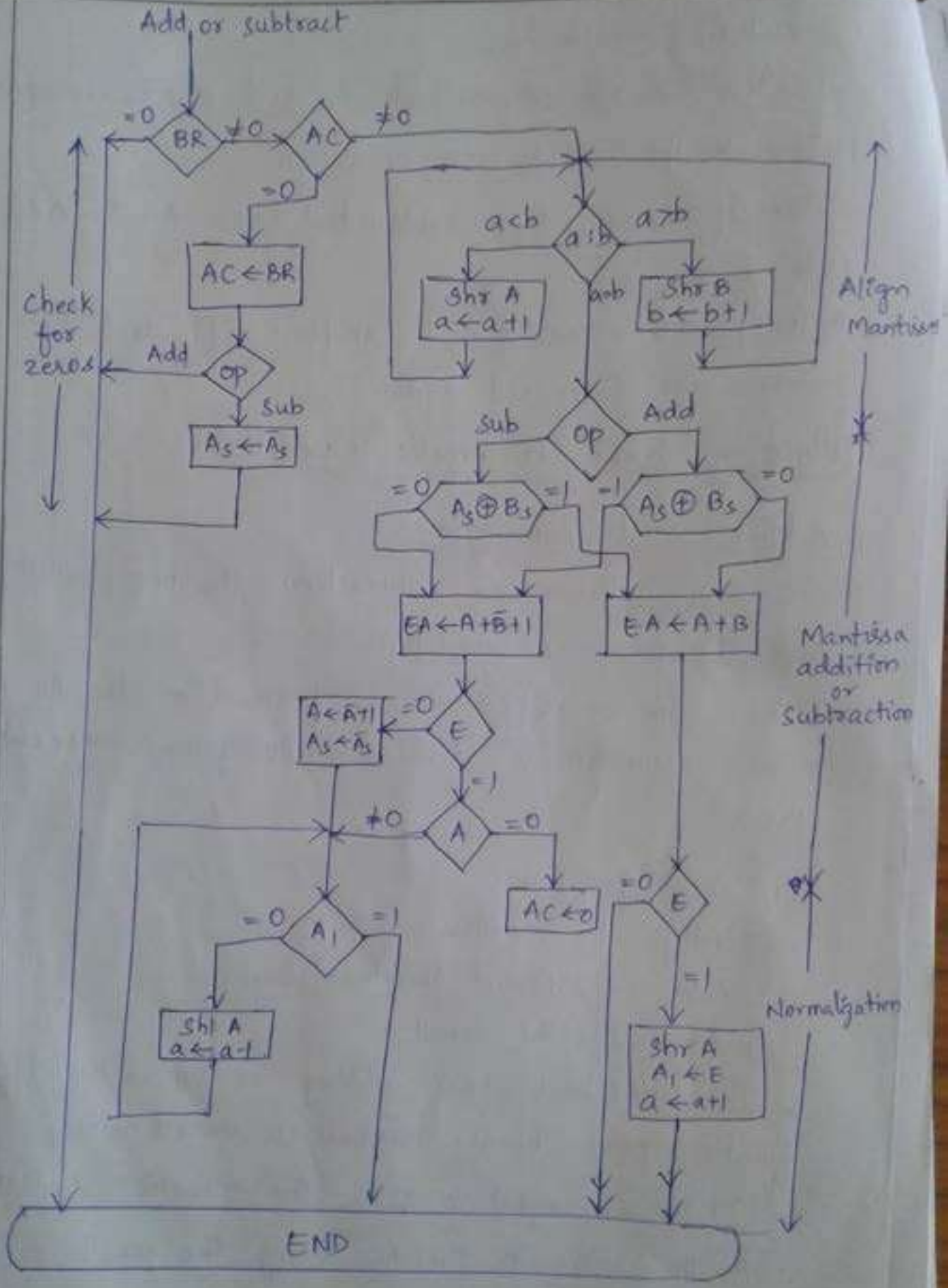


fig:- Addition and subtraction of floating-point Numbers

Scanned by CamScanner

if neither number is equal to zero, we proceed to align the mantissas.

→ The magnitude Comparator attached to exponents a and b provides three outputs that indicate their relative magnitude

→ If the two exponents are equal we go to perform the arithmetic operation.

→ If the exponents are not equal, the mantissa having the smaller exponent is shifted to the right and its exponent incremented.

→ This process is repeated until the two exponents are equal.

→ The magnitude part is added or subtracted depending on the magnitude operation and the signs of the two mantissas.

→ If an overflow occurs when the magnitudes are added, it is transferred into flip flop E.

→ If E is equal to 1, the bit is transferred into A_1 and all other bits of A are shifted right.

→ The exponent must be incremented to maintain the correct number.

→ If the magnitudes were subtracted, the result may be zero or may have an underflow

→ If the mantissa is zero, AC is made zero.

→ The mantissa has an underflow if the most significant bit in position A_1 is 0.

Scanned by CamScanner

→ In that case, the mantissa is shifted left and the exponent decremented.

→ The bit in A_1 is checked again and the process is repeated until it is equal to 1.

→ If $A_1 = 1$, the mantissa is normalized and the operation is completed.

Multiplication:-

→ The multiplication algorithm can be subdivided into four parts:

1) check for zeros.
2) Add the exponents.
3) Multiply the mantissas.
4) Normalize the product.

Flowchart for floating point Multiplication

→ The two operands are checked to determine if they contain a zero.

→ If either operand is equal to zero, the product in Ac is set to zero and the operation is terminated.

→ The exponent of the multiplier is in 'q' and the adder is between exponents 'a' and 'b'.

→ Transfer the exponents from 'q' to 'a', add the two exponents, and transfer the sum into 'a'.

→ Since both exponents are biased by the addition of a constant, the exponent sum will have double

Scanned by CamScanner

this bias.

→ The correct bias exponent for the product is obtained by subtracting the bias number from the sum

→ The multiplication of the mantissa is done as in the fixed-point case with the product residing in A and Q.

→ The product may have an ~~overflow~~ underflow, so the most significant bit in A is checked.

→ If it is a 1, the product is already normalized.

→ If it is a 0, the mantissa in AQ is shifted left and the exponent decremented.

<u>Note</u>:- Only one normalization shift is necessary. The multiplier and multiplicand were originally normalized and contained fractions.

→ Only the value in the AC is taken as the product.

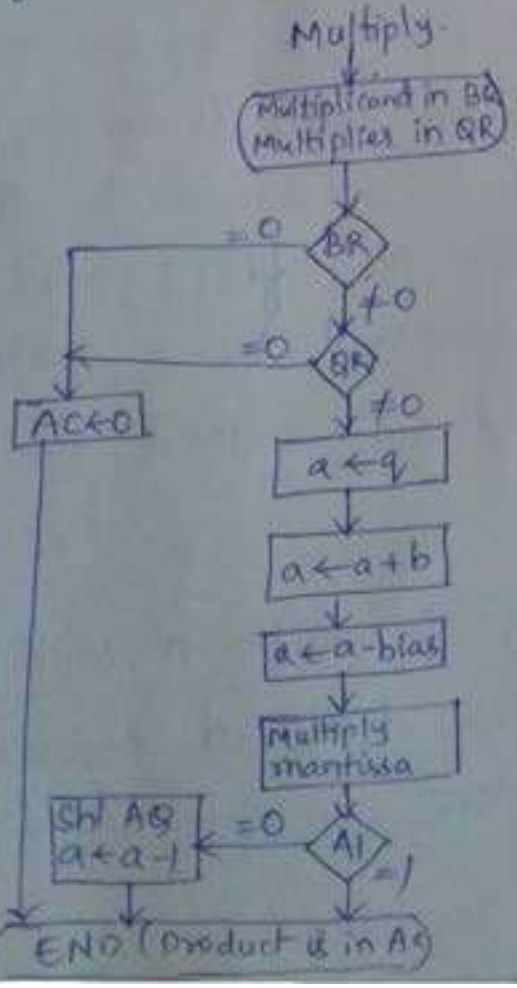


Fig:- Multiplication of floating-point numbers.

Scanned by CamScanner

Division :-

→ The division algorithm can be subdivided into five parts :

1) check for zeros.
2) Initialize registers and evaluate the sign.
3) Align the dividend
4) Subtract the exponents
5) Divide the mantissas

→ The two operands are checked for zero.

→ If the divisor is zero, it attemp to divide by zero, which is an illegal operation. and the operation terminates.

→ An alternative procedure would be to set the quotient in QR to most positive number or to the most negative number possible.

→ If the operands are not zero, we proceed to determine the sign of the quotient and store it in Q_s.

→ The dividend alignment is similar to the divide-overflow check in the fixed-point operation.

→ The two fractions are compared by a subtraction test. The carry in E determines their relative magnitude.

→ If $A \geq B$, it is necessary to shift A once to the right and increment the dividend exponents.

→ Next, the divisor exponent is subtracted from the dividend exponent.

Scanned by CamScanner

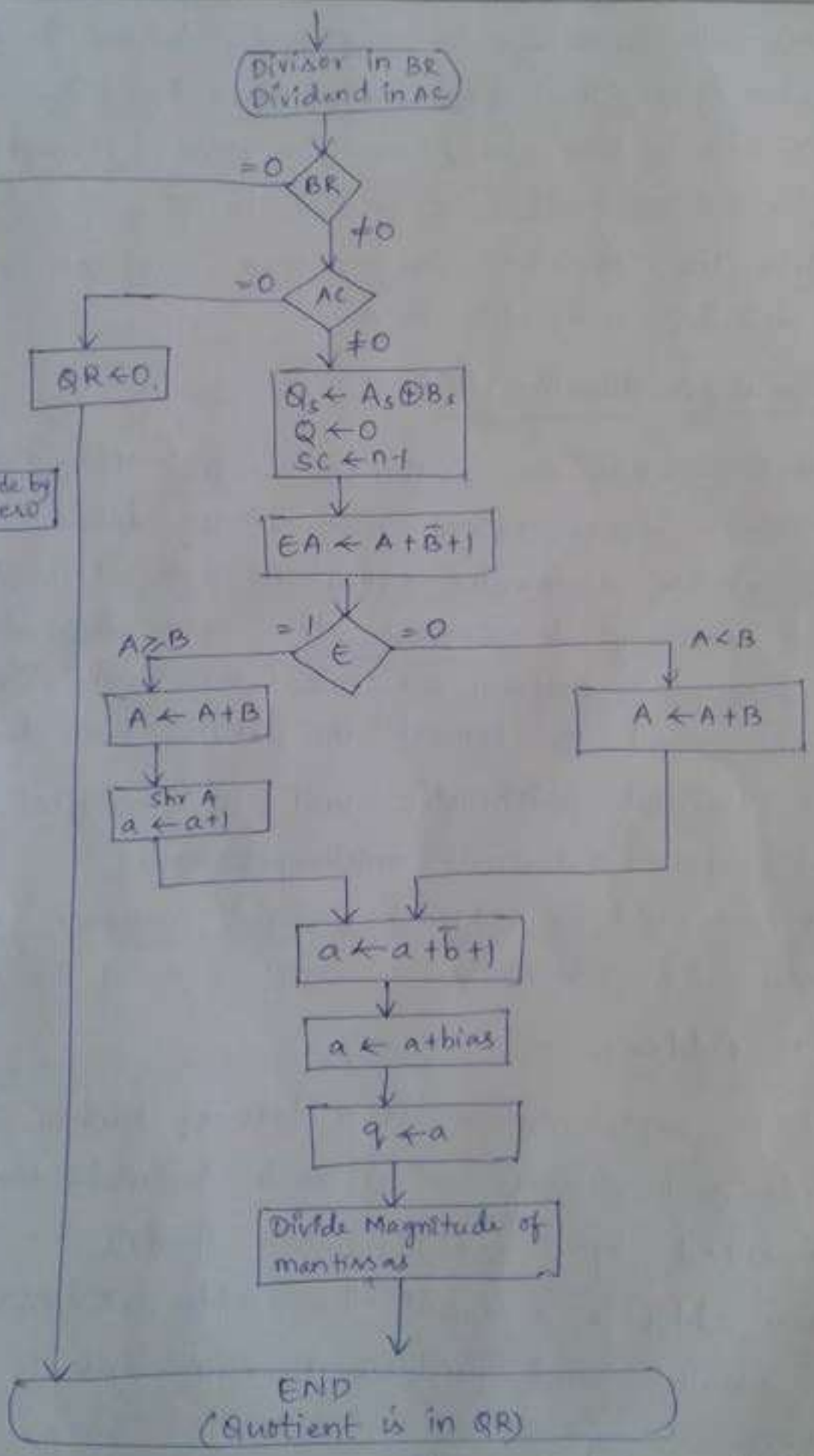


fig:- Division of floating-point numbers.

Scanned by CamScanner

→ Since both exponents were originally biased, the subtraction operation gives the difference without the bias.
→ The bias is then added and the result transferred into q because the quotient is formed in QR.
→ After the operation, the mantissa quotient resides in Q and the remainder in A.

Decimal Arithmetic Unit

→ A CPU with an arithmetic logic unit can perform arithmetic microoperations with binary data.
→ To perform arithmetic operations with decimal data, it is necessary to convert the input decimal numbers to binary, to perform all calculations with binary numbers, and to convert the results into decimal.
→ A decimal arithmetic unit is a digital function that performs decimal microoperations.
→ It can add or subtract decimal numbers, usually by forming the 9's and 10's complement of the subtrahend.

BCD Adder:-

→ BCD representation is a class of binary encodings of decimal numbers where each decimal digit is represented by a fixed number of bits.
→ BCD adder is a circuit that adds two BCD digits in parallel and produces a sum digit in BCD form.

Scanned by CamScanner

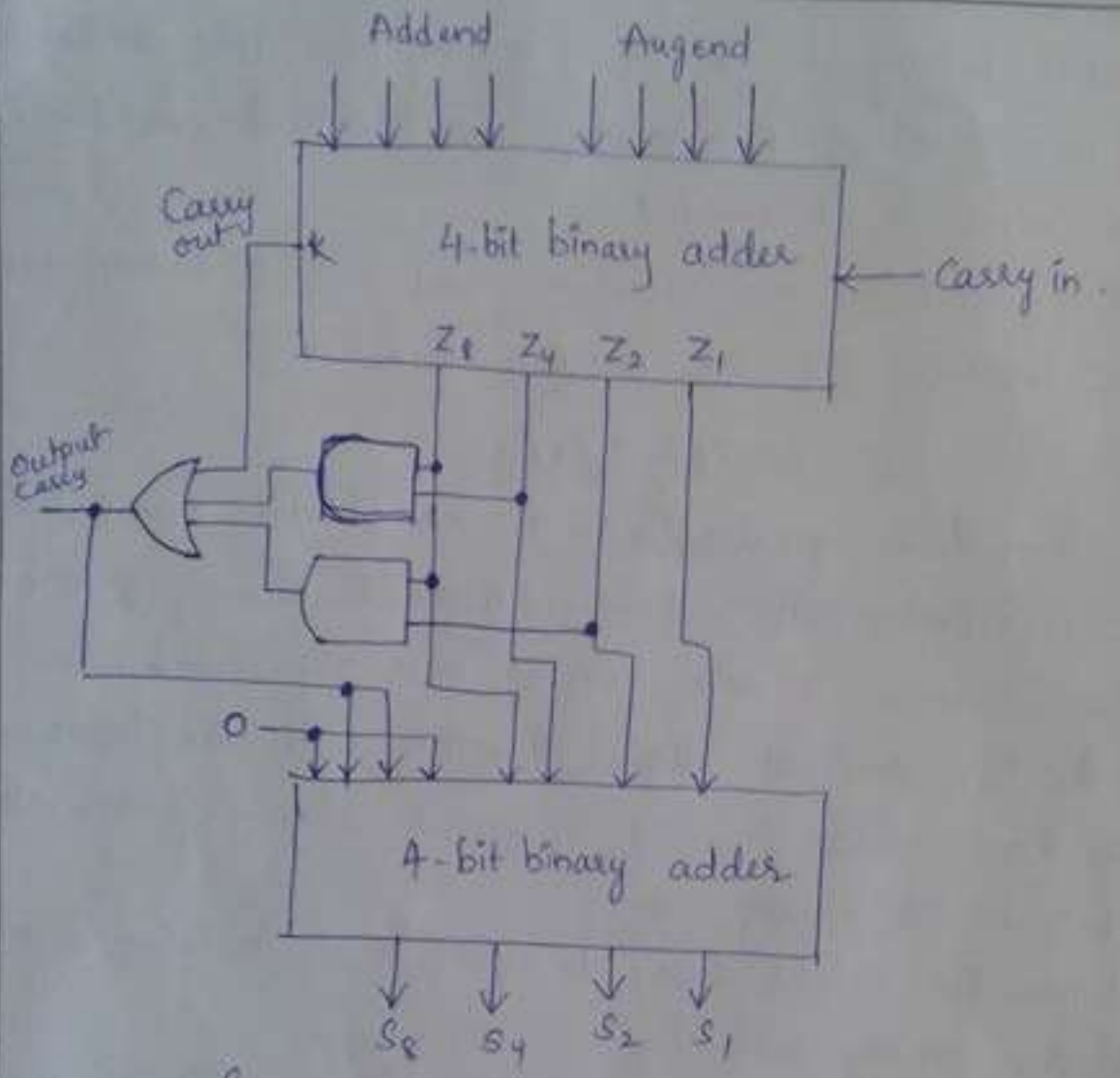


fig:- BCD Adder.

→ Since each input digit does not exceed 9, the output sum cannot be greater than 19 (9+9+1).

For Example:- Suppose we apply two BCD digits to 4-bit binary adder.

→ The adder will form the sum in binary and produce a result that may range from 0 to 19.

→ The binary numbers are represented by K, Z_8, Z_4, Z_2, Z_1

→ K is the carry and subscripts under the Z represent the weights 8, 4, 2 and 1 that can be assigned to the four bits in the BCD Code.

Scanned by CamScanner

→ when binary sum is equal to or less than or equal to 9, corresponding BCD number is identical and therefore no conversion is needed.

→ The condition for correction and output carry can be expressed by the boolean function:

$$C = K + Z_8 Z_4 + Z_8 Z_2$$

→ when it is greater than 9, we obtain non valid BCD representation, then additional binary 6 to binary sum converts it into correct BCD representation.

→ The two decimal digits, together with the input-carry are first added in the top-4-bit binary adder to produce the binary sum.

→ when the output carry is equal to 0, nothing is added to the binary sum.

→ when $C = 1$, binary 0110 is added to the binary sum using bottom 4-bit binary adder.

→ The output carry generated from the bottom binary-adder may be ignored.

BCD Subtraction :-

→ Subtraction needs the 9's and 10's complement of the subtrahend and adding it to the minuend.

→ The 9's complement of a BCD digit can also be obtained through a combinational circuit.

→ when this circuit is attached to a BCD adder, the result is a BCD adder/subtractor.

Scanned by CamScanner

→ Let the subtrahend (or addend) digit be denoted by the four binary variables B_8, B_4, B_2 and B_1.

→ Let M be a mode bit that controls the add/subtract operation.

→ when $M=0$, the two digits are added.

→ when $M=1$, the digits are subtracted.

→ Let the binary variables x_8, x_4, x_2 and x_1 be the outputs of the 9's complementer circuit.

→ The Boolean functions for the 9's complementer circuit are

$$x_1 = B_1M' + B_1'M$$

$$x_2 = B_2$$

$$x_4 = B_4M' + (B_4'B_2 + B_4B_2')M$$

$$x_8 = B_8M' + B_8'B_4'B_2'M.$$

Decimal Arithmetic Operations:-

→ Decimal numbers in BCD are stored in computer registers in groups of four bits.

→ Each 4-bit group represents a decimal digit and must be taken as a unit when performing decimal microoperations.

→ Adding 1 to the 9's complement produces the 10's complement.

→ Thus, for decimal numbers, the symbol $A \leftarrow A + \bar{B} + 1$ denotes a transfer of the decimal sum formed by adding the original content A to the 10's complement of

Scanned by CamScanner

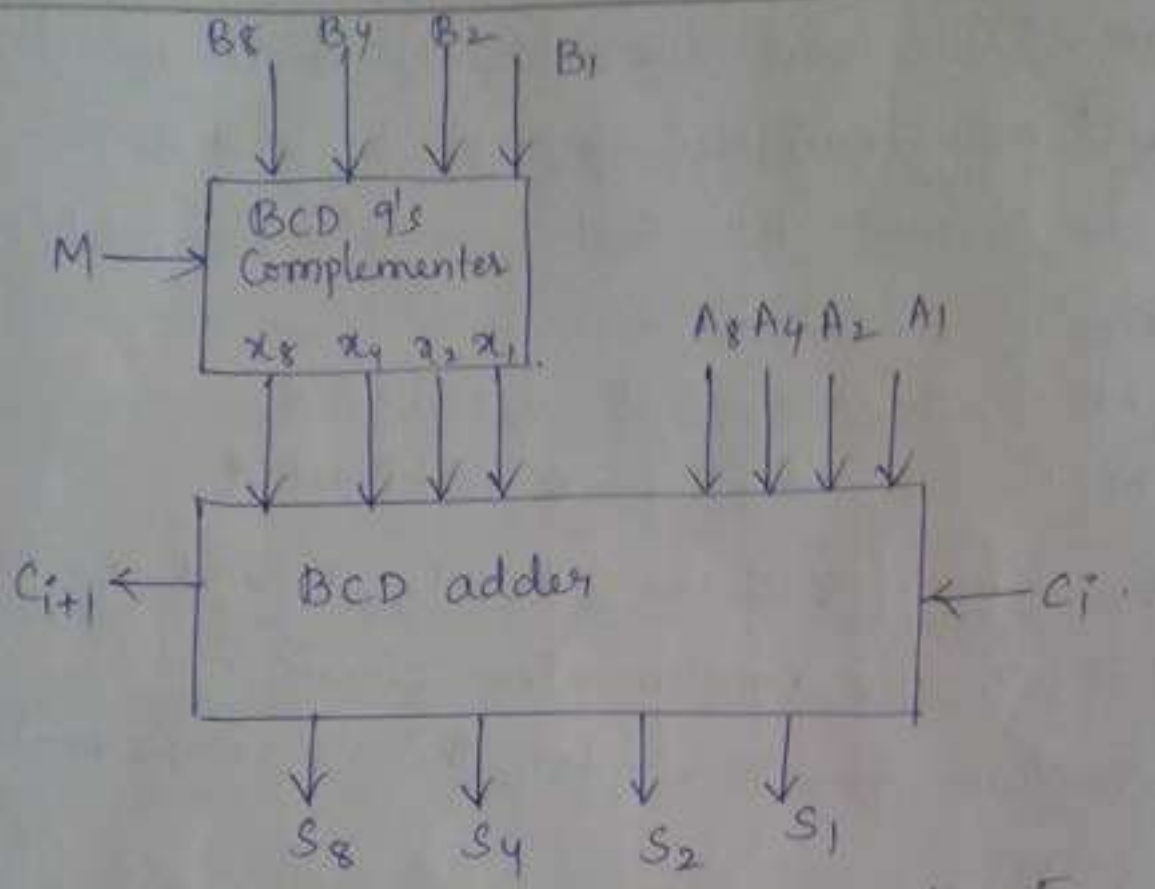


fig: One stage of a decimal arithmetic unit.

Symbolic Designation	Description
$A \leftarrow A+B$	Add decimal numbers and transfer sum to A
$\bar{B}$	9's complement of B
$A \leftarrow A+\bar{B}+1$	Content of A plus 10's complement of B into A
$Q_L \leftarrow Q_L+1$	Increment BCD number in Q_L
dshr A	Decimal shift-right register A
dshl A	Decimal shift-left register A

Table- Decimal Arithmetic Microoperation Symbols.

Scanned by CamScanner

Addition and Subtraction

→ The addition and subtraction algorithm of binary signed-magnitude number applies also to decimal signed-magnitude numbers. Similarly the signed 2's complement binary numbers applies to decimal signed-10's complement numbers.

→ Decimal data can be added in three different ways.

Parallel Method :-

→ Parallel method uses a decimal arithmetic unit composed of as many BCD adders as there are digits in the number.

→ The sum is formed in parallel and requires only one microoperation.

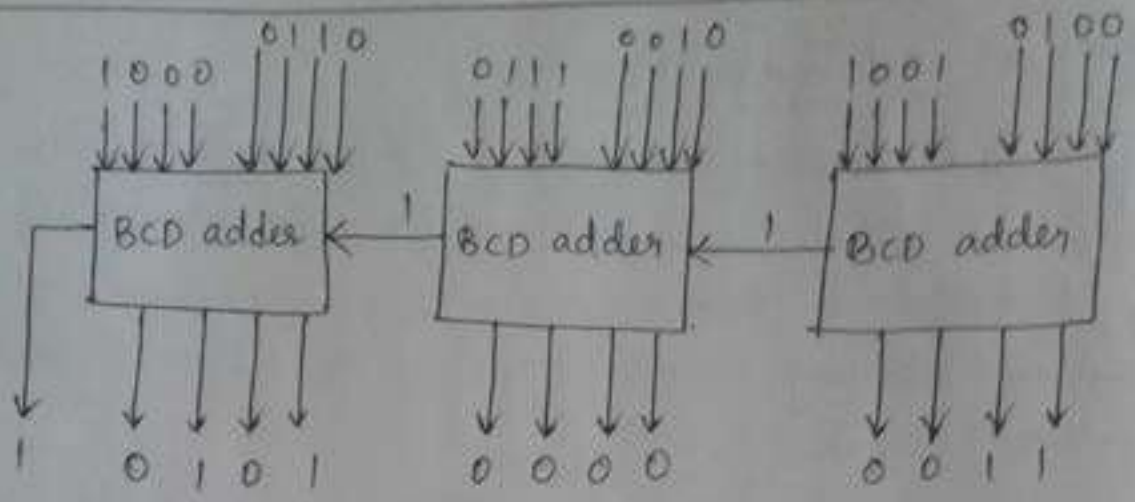
Digit-Serial, bit parallel Method :-

→ In the digit-serial bit-parallel method, the digits are applied to a single BCD adder serially, while the bits of each coded digit are transferred in parallel.

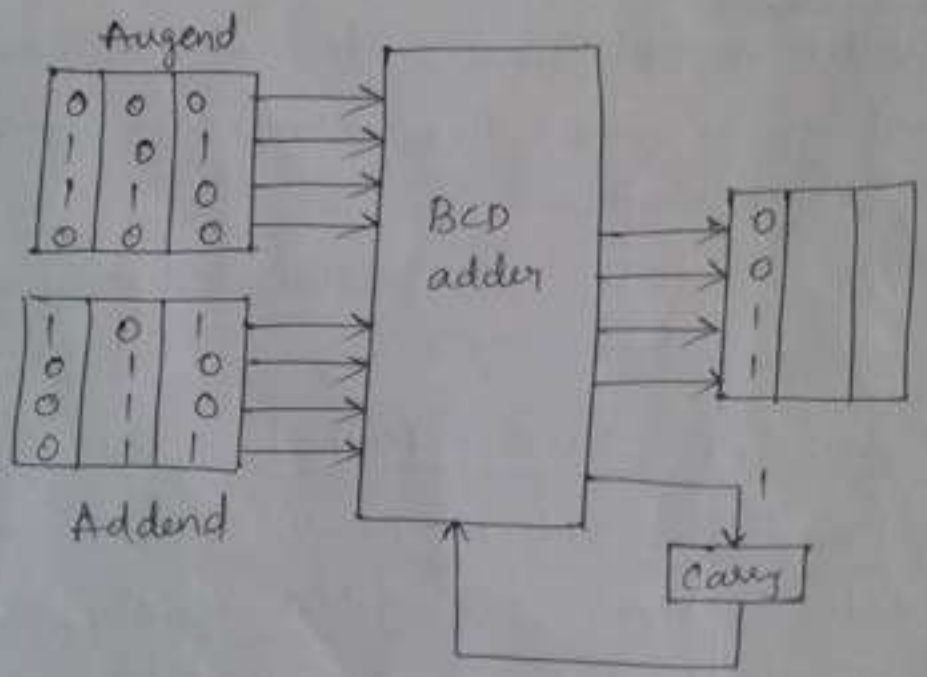
→ The sum is formed by shifting the decimal numbers through the BCD adder one at a time.

The parallel method is fast but requires a large number of adders. The digit-serial bit parallel method requires only one BCD adder, which is shared by all the digits.

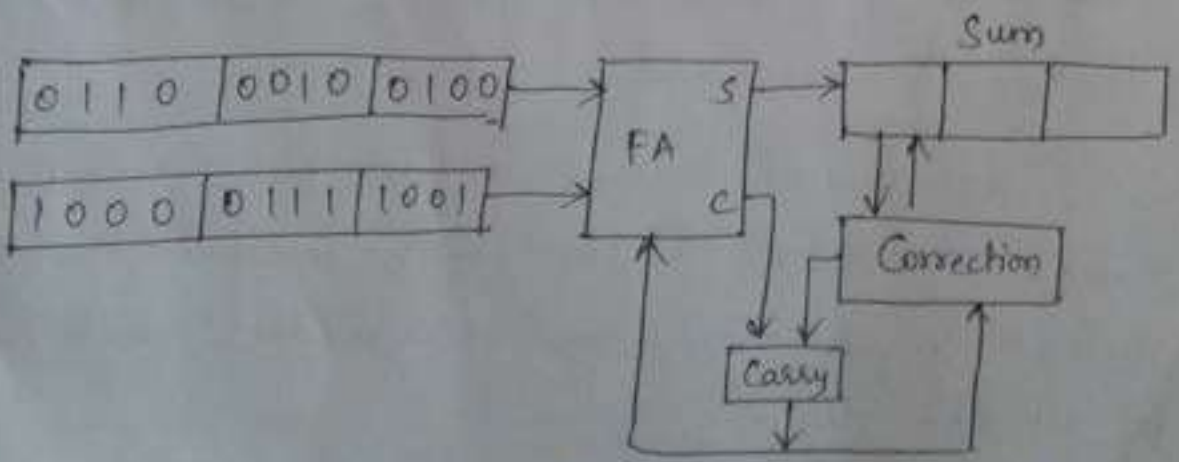
Scanned by CamScanner



(a) Parallel decimal addition : 624+879=1503



(b) Digit-serial, bit parallel decimal addition.



(c) All serial decimal addition.

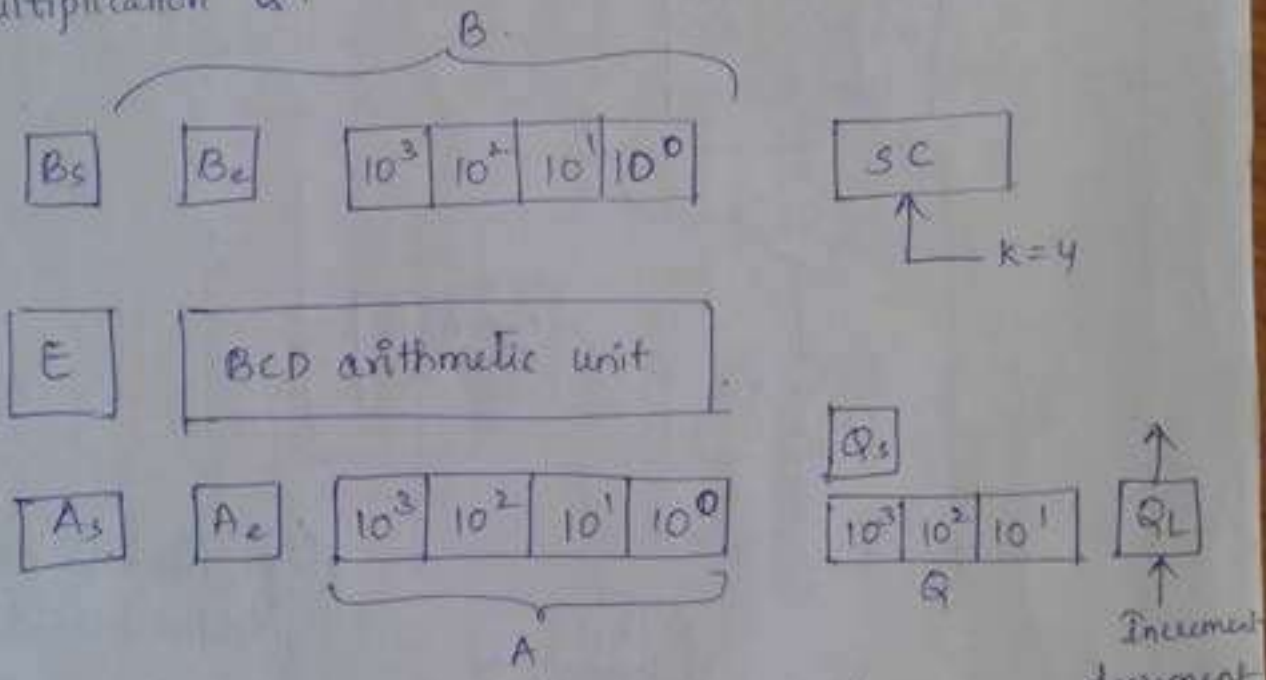
fig:- Three ways of adding decimal numbers.

Scanned by CamScanner

Multiplication :-

→ The decimal multiplication is similar to binary except for the way the partial products are formed.

→ The register organization for the decimal multiplication is.



→ There are three registers, A, B and Q, each having a corresponding sign flip-flop A_s, B_s and Q_s

→ A_e and B_e each of 2 bits provide an extension of one more digit to the registers.

→ The BCD arithmetic unit adds the five digits in parallel and places the sum in the five digit A register.

→ The end carry goes to flip-flop E.

→ A_e is used an overflow while adding the multiplicand to the partial product.

→ The purpose of B_e is to form the 9's complement of the divisor when subtracted from the partial remainder during the division operation.

Scanned by CamScanner

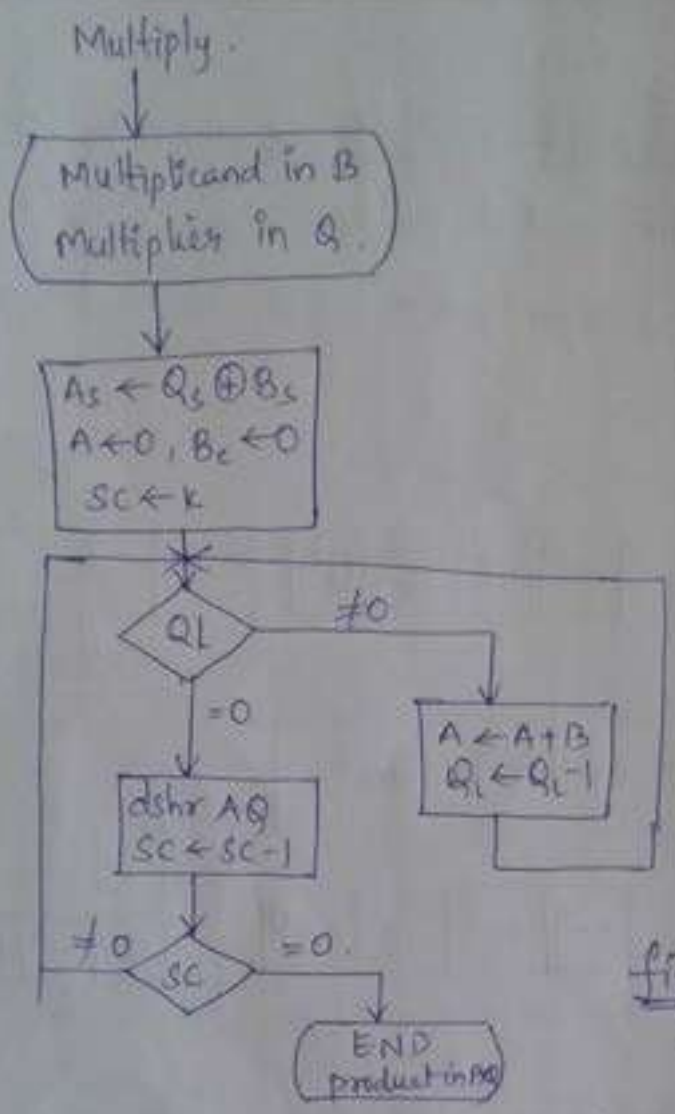


fig:- Flowchart for decimal multiplication

→ A register and B_e are cleared and Sequence Counter SC is set to number k equal to the number of digits in the multiplier.

→ The low-order digit of the multiplier in Q_L is checked

→ If it is not equal to 0, the multiplicand in B is added to partial product in A and Q_L is decremented.

→ Q_L is checked again and process is repeated until it is equal to 0.

→ In this way, the multiplicand B is added to the partial product a number of times equal to the multiplier digit.

→ Next, the partial product and the multiplier are shifted once to the right. This places zero in A_e and transfers next multiplier into Q_L. Process is repeated k times to form product in AQ.

Scanned by CamScanner

Division :-

→ Division of decimals is similar to binary division except quotient digits may have any of the 10 values from 0 to 9.

→ The divisor is subtracted from the dividend or partial remainder as many times as necessary until a negative remainder result.

→ Correct remainder is then restored by adding the divisor.

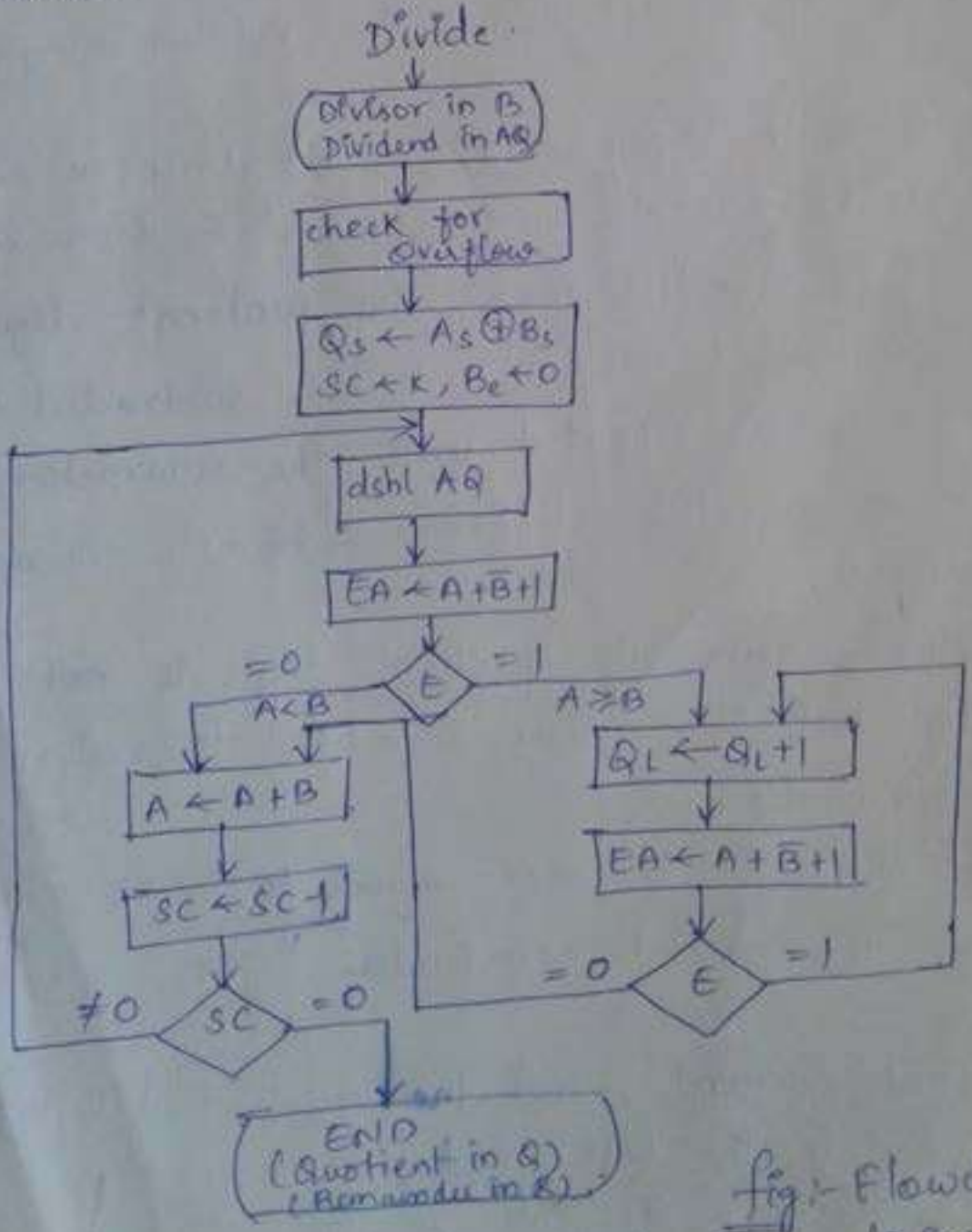


fig:- Flowchart for decimal division

Scanned by CamScanner

→ It is similar to algorithm with binary data except the way the quotient bits are formed.

→ The dividend is shifted to the left, with its MSB placed in A_E.

→ The divisor is then subtracted by adding its 10's complement value.

→ Since B_E is initially cleared, its complement value is 9 as required.

→ The carry in E determines the relative magnitude of A and B.

→ If E=0 then it signifies A<B: The divisor is added to restore the partial remainder and Q_L stays at 0.

→ If E=1, it signifies A≥B: The quotient digit in Q_L is incremented once and divisor subtracted again.

→ This process is repeated until the subtraction results in a negative difference which is recognized by E being 0.

→ When this occurs, the quotient digit is not incremented but the divisor is added to restore the positive remainder.

→ The quotient digit is made equal to the number of times that the partial remainder "goes" into the divisor.

→ The partial remainder and the quotient bits are shifted once to the left and process is repeated k times to form k quotient digits.

→ The remainder is found in register A and quotient in register Q where E is neglected.

Scanned by CamScanner

Floating point Representation

→ A floating point number in computer registers consists of two parts :

a mantissa m and an exponent e.

$$\boxed{m \times r^{e}}$$

where r is the radix.

→ The mantissa may be a fraction or an integer.

→ The location of radix and value of radix r is assumed.

→ Eg :- The decimal number = 537.25

$m = 53725 \quad e = 3.$

$$\boxed{.53725 \times 10^{3}} \qquad (\because m \times r^{e})$$

→ A floating point number is normalized if the MSB of mantissa is nonzero

→ If 48 bits are used for representing floating point number with 36 bits for mantissa and 12 for exponent

36 = 35 - magnitude , 1 for sign bit

12 = 11 - Exponent magnitude , 1 for sign bit.

$$\pm(1-2^{-35}) \times 2^{2047} \qquad (2047 = 2^{11}-1)$$

Scanned by CamScanner

Overflow:-

when two normalized mantissas are added the sum may contain an overflow digit.

→ It is corrected by shifting the sum one to the right and incrementing the exponent.

Underflow:-

A floating point number that has 0 in MSB of mantissa is said to have underflow. To normalize the result we need to shift the mantissa left and decrement the exponent until a nonzero digit appear in the first position

Eg:-

$$\begin{array}{r} .56780 \times 10^5 \\ -\ .56430 \times 10^5 \\ \hline .00350 \times 10^5 \end{array}$$

after underflow.

$.35000 \times 10^{5-2=3}$

35000×10^3.

Scanned by CamScanner

4. Input - Output Organization

Introduction :-

-> Input-output subsystem of a computer, referred to as I/O, provides an efficient mode of communication between the central system and the outside environment.

-> Programs and data must be entered into computer memory for processing and results obtained from computations are displayed for the user.

-> When input information is transferred to the processor via a slow keyboard, the processor will be idle most of the time while waiting for the information to arrive.

-> To use the computer efficiently, a large amount of programs and data must be prepared in advance and transmitted into a storage medium such as magnetic tapes or disks.

-> The information in the disk is then transferred into computer memory at a rapid rate.

-> Results of programs are also transferred into a high-speed storage, such as disks, from which it is transferred later into a printer

-> Peripherals :-

-> Input or output devices attached to the computer are also called as peripherals

-> There are three types of peripherals. They are

Scanned by CamScanner

1) input peripherals
2) Output peripherals.
3) Input-output peripherals.

-> These peripherals may be digital or analog, and serial or parallel.

-> Peripherals that provide auxiliary storage for the system are magnetic disks and tapes.

-> Peripherals are of electromechanical and electromagnetic devices.

<u>Monitor</u> <u>and</u> <u>Keyboard</u>:-

-> Video monitors are the most commonly used peripherals.

-> They consist of a keyboard as the input device and a display unit as the output device

-> The most popular video monitors are, CRT (cathode Ray Tube).

-> The CRT contains an electronic gun that sends an electronic beam to a phosphorescent screen in front of the tube.

-> The beam can be deflected horizontally and vertically

-> To produce a pattern on the screen, a grid inside the CRT receives a variable voltage that cause the beam to hit the screen and make it glow at selected spots.

Scanned by CamScanner

Printer :-

→ Printer is an output devices provide a permanent record on paper of computer output data or text.

→ There are three types of printers

1) Daisy wheel
2) dot matrix
3) Laser printers.

Magnetic Tape :-

→ Magnetic tapes are used mostly for storing files of data

Magnetic disk :-

→ Magnetic disks have high-speed rotational surfaces coated with magnetic material.

→ Access is achieved by moving a read-write mechanism to a track in the magnetized surface.

4.1 Input - Output Interface :-

→ Input-output interface provides a method for transferring information between internal storage and external I/O devices.

→ Peripherals connected to a computer need special communication links for interfacing them with the CPU.

→ This communication link is used to resolve the differences that exist between central computer and peripheral.

Scanned by CamScanner

Major differences b/w CPU and phesipheral :-

1) Peripherals are electromechanical and electromagnetic devices whose operation is different from the operation of CPU and memory which are electronic devices. The Conversion of signal values may be required.

2) The data transfer rate of peripherals is usually slower than the transfer rate of the CPU so, we need synchronization mechanism.

3) Data codes and formats in peripherals differ from the word format in CPU and memory.

→ The operation modes of peripherals are different from each other and each must be controlled so as not to disturb the operation of other peripherals connected to the CPU.

To resolve these differences, we need a special hardware components between the CPU and peripherals to supervise and synchronize all input and output transfers.

→ These components are called 'interface' units because they interface between CPU and peripherals.

Scanned by CamScanner

Main functions of Input-Output interface:-

- data Conversion
- Synchronization
- Device selection.

Data Conversion:-

→ It refers to Conversion between digital and analog signals and conversion between serial and parallel data formats.

Synchronization:-

→ It refers to matching of operating speeds of CPU and other peripherals.

Device selection:-

→ It refers to selection of I/O device by CPU in a queue manner.

Note:- Each and every I/O device consists of a controller which supervises the operations of the particular mechanism in the peripheral.

I/O Bus and Interface Modules:-

→ The communication link between the processor and several peripherals is the "Interface".

→ These interfaces of each peripheral and processor is connected to the I/O bus.

→ The I/O bus consists of data lines, address lines and control lines.

Scanned by CamScanner

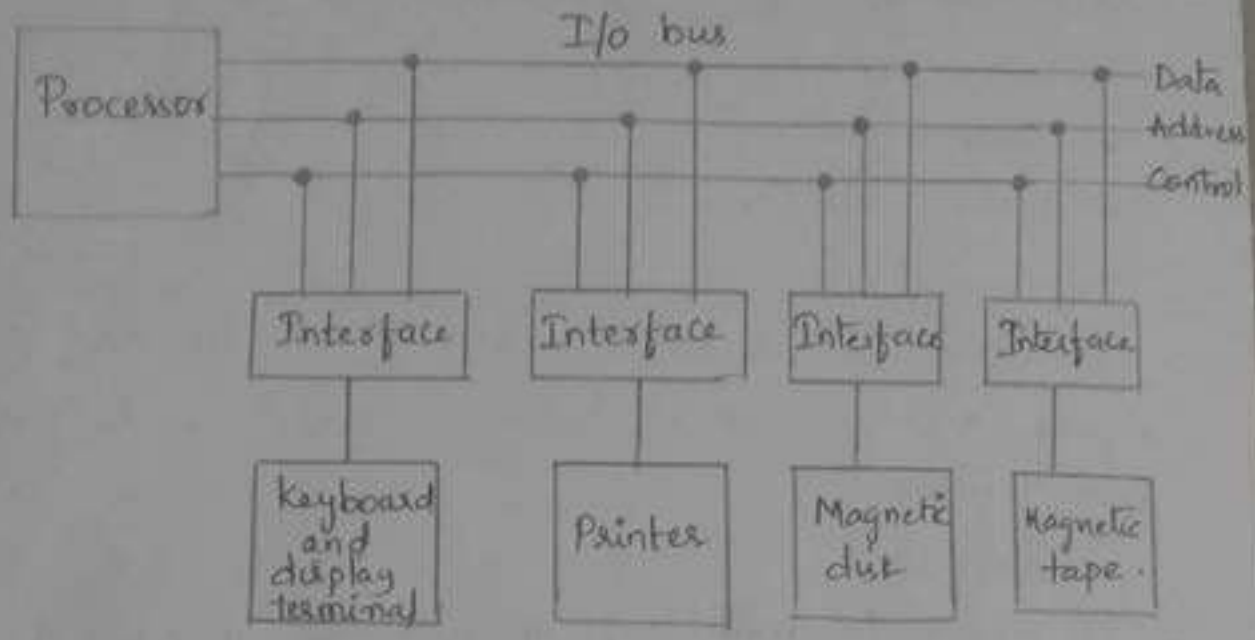


fig:- Connection of I/o bus to input-output devices.

→ Each peripheral is attached to the interface unit.

→ Each peripheral also contains the controller unit to supervises the operations of particular mechanism in the peripheral.

→ Each interface decodes the address and control received from I/o bus, interprets them for the peripheral and provides signals for the peripheral controller.

→ Interface synchronizes the data flow and supervise the transfer between peripheral and processor

→ Each peripheral has its own controller that operates the particular electromechanical device.

Eg:- The printer controller controls the paper motion, the print timing and selection of printing characters.

Scanned by CamScanner

→ A controller may be housed separately or may be physically integrated with the peripheral.

→ To communicate with the particular device, the processor places a device address on the address lines.

→ Each interface attached to the I/O bus contains an address decoder that monitors the address lines.

→ When interface detects its own address, it activates the path between the bus lines and the device that it controls.

→ At the same time, the processor provides a function code in the control lines.

→ The interface selected responds to the function code and proceeds to execute it.

→ function code is nothing but the I/O command, an instruction that is executed in the interface and its attached peripheral unit.

→ There are four types of commands that an interface may receive. They are

1) Control
2) Status
3) data Output
4) data input.

<u>Control Command</u> :-

→ It is issued to activate the peripheral and to inform it what to do.

Scanned by CamScanner

for Example :- A magnetic tape unit may be instructed to backspace the tape by one record, to rewind the tape or to start the tape moving in the forward direction.

→ The particular control Command depends on the peripheral, and each peripheral can has different and set of control Commands depending on its mode of operation.

Status Command :-

→ A status Command is used to test various status conditions in the interface and peripheral.

Example :- Computer may wish to check the status of the peripheral before a transfer is initiated.

→ During the transfer, the errors may occur. so it is indicated by setting the status register and can be read by processor.

Data Output Command :-

→ A data output Command causes the interface to respond by transferring data from the bus into one of its registers.

Data Input Command :-

→ The interface receives the data item from the peripheral and places it in the its buffer register

→ The processor checks the status Command and then issues a data input Command.

Scanned by CamScanner

I/O Versus Memory Bus

→ In addition to communicating with I/O, the processor must communicate with the memory unit.

→ Like I/O bus, the memory bus also contains data, address and control lines.

→ There are 3 ways the computer buses can be used to communicate with memory and I/O:

1) Use two separate buses, one for memory and other for I/O.
2) Use one common bus for both memory and I/O but have separate control lines.
3) Use one common bus for memory and I/O with common control lines.

→ In the first method, the computer has independent set of data, address, and control buses one for accessing memory and the other for I/O.

→ This is done in computer by using IOP (Input-output processor).

→ The memory communicates with both the CPU and the IOP through a memory bus.

Scanned by CamScanner

Isolated versus Memory-mapped I/O

Isolated I/O :-

→ In isolated I/O Configuration, the CPU has distinct input and output instructions and each instruction is associated with the address of an interface register.

→ The CPU fetches and decodes the operation code of an input or output instruction, it places the address associated with the instruction into the common address lines.

→ At the same time, it enables the I/O read or I/O write control line.

→ This informs the external components that are attached to the common bus that the address in the address lines is for an interface register and not for a memory word.

→ In isolated I/O method isolates the memory addresses and I/O addresses. so that memory address values are not affected by interface address

→ Both memory and I/O have its own address space.

→ The I/O read and I/O write control lines are enabled during a memory transfer. and memory read or memory write operations are enabled during a memory transfer.

Scanned by CamScanner

→ The other alternative is to use the same memory space for both memory and I/O.

Memory-Mapped I/O :-

→ In this method, only one set of read and write signals will be there

→ The computer treats the interface registers as a part of a memory.

→ Such configuration is referred to as memory-mapped I/O.

→ In this memory-mapped I/O organization there are no specific input or output instructions.

→ The same instruction can be used to manipulate the data present in memory word or in interface register.

→ A segment of the memory is reserved for the I/O registers

→ The assigned addresses for interface registers cannot be used for memory words.

→ Computers with memory-mapped I/O can use memory-type instructions to access I/O data.

→ The load and store instructions used for reading and writing from memory can be used to input and output data from I/O registers.

Scanned by CamScanner

Example of I/O Interface :-

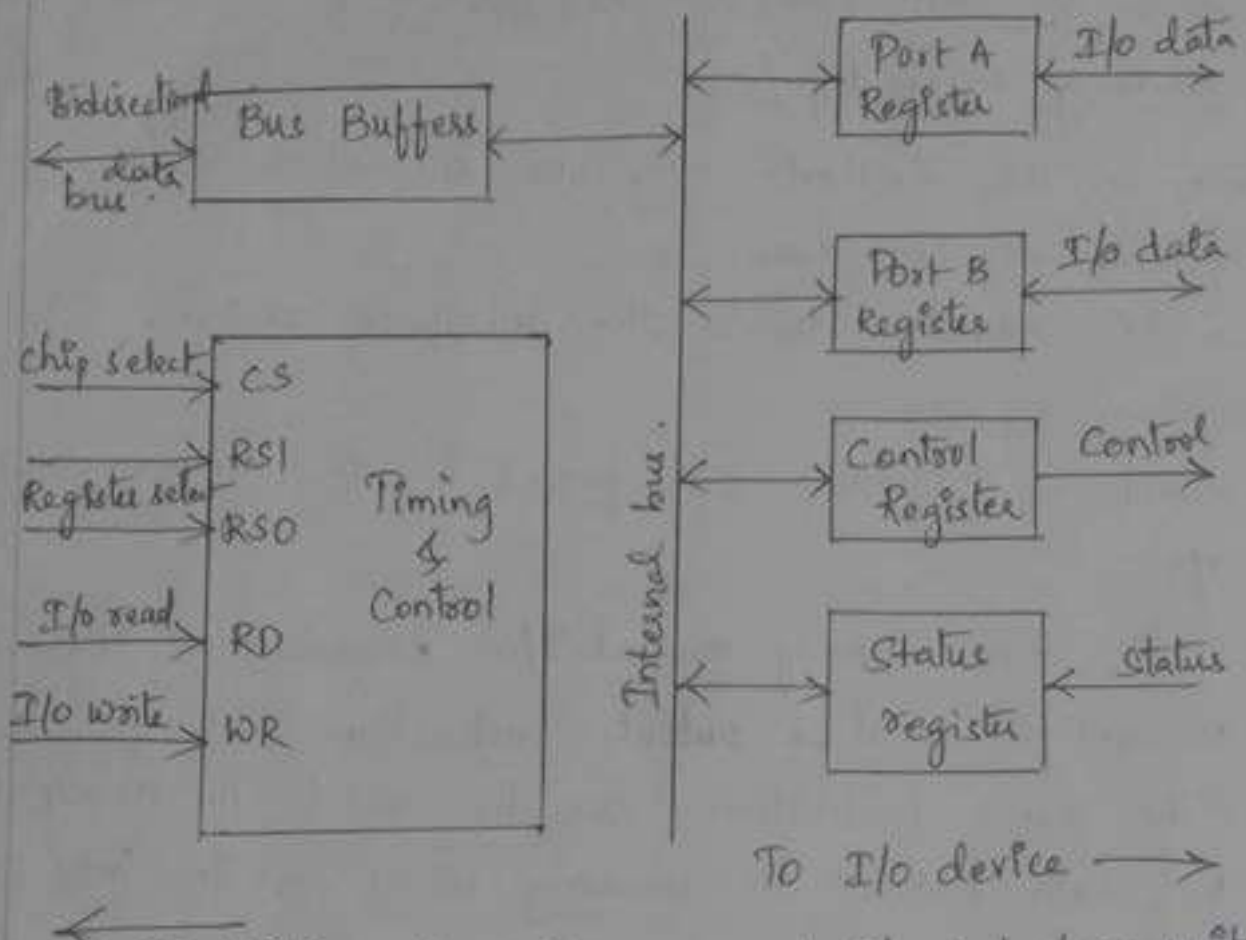


fig :- Example of I/O interface unit.

→ It consists of two data registers called ports, a control register, a status register, bus buffers and timing and control circuit.

→ The interface communicates with the CPU through the data bus.

→ The chip select and register select inputs determine the address assigned to the interface.

→ The I/O read and writes are two control lines that specify an input or output respectively.

Scanned by CamScanner

→ The four registers communicate directly with the I/O device attached to the interface.

→ The I/O data to and from the device can be transferred into either port A or port B.

→ Example :- If the interface is connected to a printer, it will only output data, and if it services a character reader, it will only input data.

→ A magnetic disk unit transfers data in both directions but not at the same time, so the interface use bidirectional unit.

→ A command is passed to the I/O device by sending a word to the appropriate interface register.

→ The control register receives control information from the CPU. By loading appropriate bits into the control register, the interface and the I/O device attached to it can be placed in a variety of operating modes.

→ for Example :- Port A may be defined as an input port and port B as an output port.

→ A magnetic tape unit may be instructed to rewind the tape or to start the tape moving in the forward direction.

→ The bits in the status register are used for status conditions and for recording errors that may occur during the transfer.

Scanned by CamScanner

For Example, a status bit may indicate that port A has received a new data item from the I/O device.

→ Another bit in the status register may indicate that a parity error has occured during the transfer.

→ A circuit must be provided externally to detect the address assigned to the interface register.

→ This circuit enables the chip select (CS) input when the interface is selected by the address bus.

→ The two register select inputs RS1 and RS0 are usually connected to the two least significant lines of the address bus.

→ These two inputs select one of the four registers in the interface as specified in the table

CS	RS1	RS0	Register Selected
0	x	x	None, bus in high-impedance
1	0	0	Port A register
1	0	1	Port B register
1	1	0	Control register
1	1	1	Status register

→ The content of the selected register is transfer into the CPU via the data bus when the I/O read signal is enabled.

Scanned by CamScanner

→ The CPU transfers binary information into the selected register via the data bus when the I/O write input is enabled.

Asynchronous Data Transfer:-

→ Generally clock pulse is applied to the registers so that all the data transfers occur between the registers during the occurence of a clock pulse.

→ CPU and I/O are two independently designed units.

→ If the registers in CPU and register in I/O share a same or common clock, the transfer b/w these two units is said to be 'synchronous'.

→ In most cases, the internal timing in each unit is independent from the other then if each uses a own private clock for internal registers.

→ In this case, the two units are said to be asynchronous to each other.

→ Asynchronous data transfer between two independent units requires that control signals be transmitted between the communicating units to indicate the time at which data is being transmitted.

→ To achieve this we follow

1) strobe ~~pulse~~ control method
2) Handshaking method.

Scanned by CamScanner

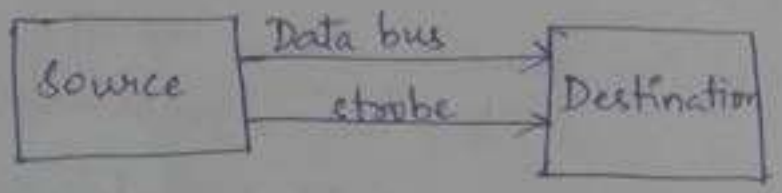
Strobe Control

→ A strobe pulse supplied by one of the units to indicate to the other unit when the transfer has to occur.

→ This method of asynchronous data transfer employs a single control line to time each transfer.

→ The strobe pulse may be activated by either the source or the destination unit.

Source initiated strobe transfer :-



→ The 'data bus' has multiple lines to transfer the binary information (byte or word) from source unit to destination unit.

→ The 'strobe' is a signal line that informs the destination unit when a valid data is available on the bus.

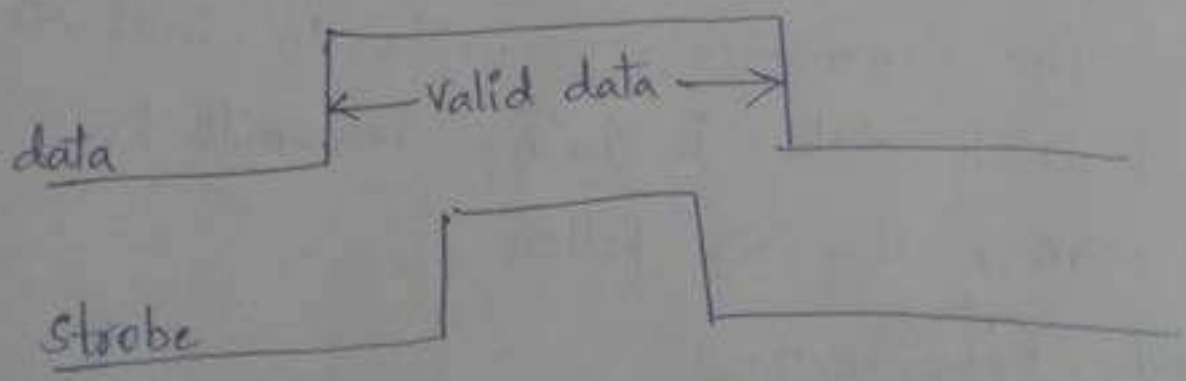


fig. Timing diagram

Scanned by CamScanner

→ In the timing diagram, the source first places the data on the data bus.

→ After some delay to ensure the data settle to a steady value, then the strobe activates the strobe pulse.

→ strobe pulse will be active for certain amount of time in which the information on the data bus is received by the destination unit.

→ The transfer of data occurs at the falling edge of strobe pulse.

→ The source removes the data from the bus after certain period of time after it disables the strobe pulse.

→ The strobe pulse disable indicates the data bus does not contain valid data.

Destination initiate strobe for data transfer :-

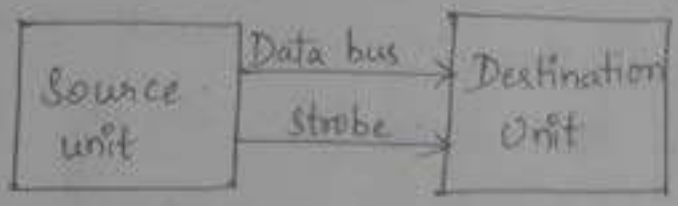


fig :- Block diagram

→ In this case, the destination unit activates the strobe pulse informing the source to provide the data.

→ The source unit responds by placing a requesting binary information on the data bus.

→ The data must be valid and remain in the bus

until destination unit accepts it.

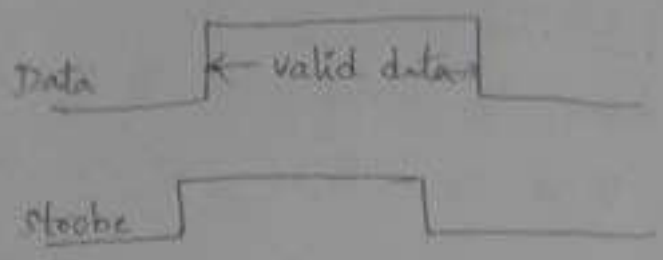


fig: Timing Diagram.

→ The falling edge of the strobe pulse can be used again to trigger a destination register.
→ Then the destination unit disables the strobe.
→ The source removes the data after certain interval of time.

In many computers, the strobe pulse is controlled by the clock pulses in the CPU. In the source initiated strobe the source unit may be a CPU and destination unit may be a memory. The CPU places the data on the bus and informs the memory unit then it is a write operation.

In the destination initiated strobe, the destination may be CPU sends the control for memory-read to the source (memory). Then the CPU initiates the memory read ~~memory~~ & places the selected word on the bus.

The transfer of data between the CPU and the interface unit uses the strobe pulse transfer.

Scanned by CamScanner

Disadvantages of strobe control transfer

→ The source unit that initiates the transfer has no way of knowing whether the destination unit has actually received the data item that is placed on the bus.

→ Similarly, the destination unit that initiates the transfer has no way of knowing whether the source unit has actually placed the data on the bus.

Handshaking Method

→ The disadvantages of strobe control transfer is solved by handshake method by introducing a second control signal that provides a reply to the unit that initiates the transfer.

Definition :-

→ The unit receiving the data item responds with another control signal to acknowledge receipt of the data. This type of agreement between two independent units is referred to as 'handshaking'.

→ The basic principle of handshaking contains two control signals.

(1) One control signal is in the same direction as the data flow in the bus from source to destination to inform the destination unit whether there are valid data in the bus.

Scanned by CamScanner

→ The other control line is in the other direction from destination to the source. It is used by the destination unit to inform the source whether it can accept data.

→ Again handshaking can be defined in two ways

(i) Source initiated transfer

(ii) Destination - Initiated transfer

Source initiated transfer:-

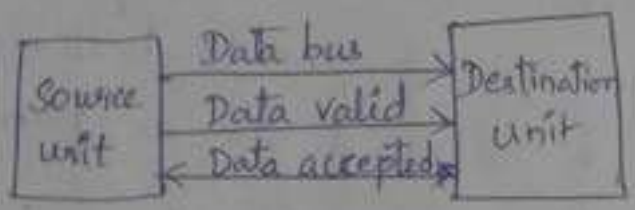
→ The data transfer procedure when initiated by the source.

→ The two handshaking lines are

(i) data valid

(ii) data accepted.

→ data valid is generated by source unit and data accepted generated by destination unit.



block diagram.

→ The timing diagram shows the exchange of signals between the two units.

→ The source unit initiates the transfer by placing the data on the bus and enabling its 'data valid' signal.

Scanned by CamScanner

→ The 'data accepted' signal is activated by the destination unit after it accepts the data from the bus.
→ The source unit then disables its 'data valid' signal, which invalidates the data on the bus.
→ The destination unit then disables its 'data accepted' signal and system goes into its initial state.

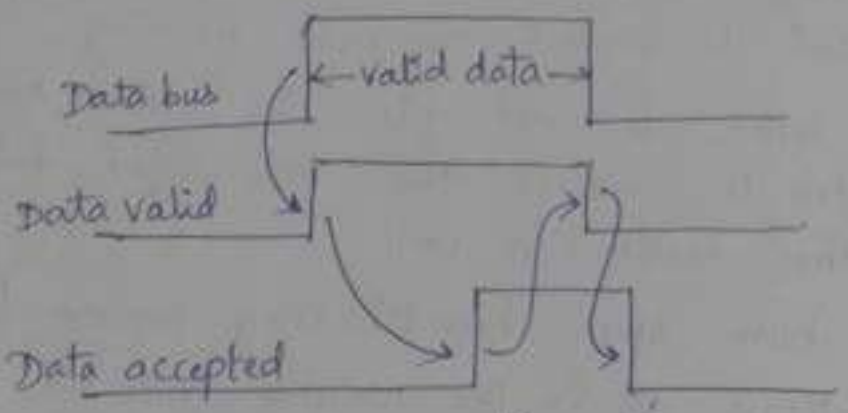


fig :- Timing diagram

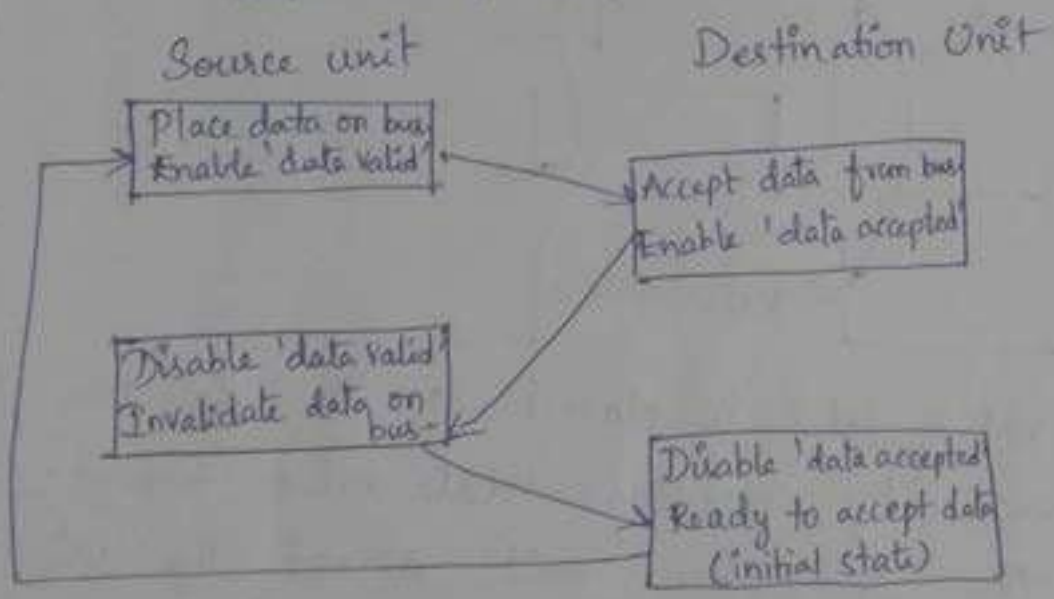


fig :- Sequence of events.

→ The source does not send the next data item until after the destination unit show its readiness to accept new data by disabling its 'data accepted' signal.

Scanned by CamScanner

Destination initiated transfer

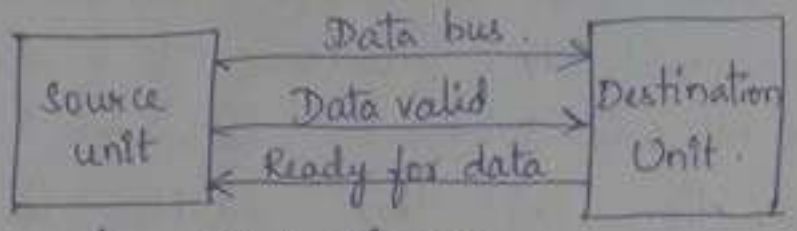
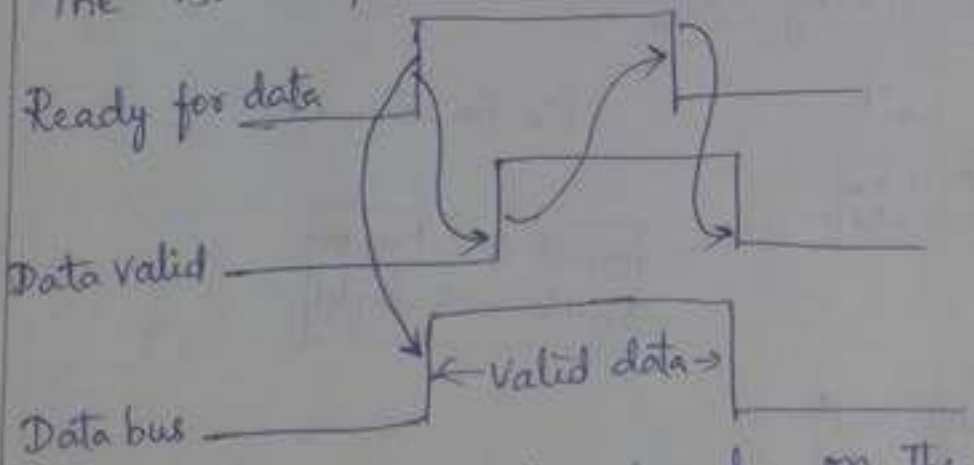


fig:- Block Diagram

→ Here the destination unit has been changed to 'ready for data' to reflect the new meaning.

→ The source unit does not place the data on the bus until after it receives the 'ready for data' signal from the destination unit.

→ After the above step, handshaking process follows the same pattern as in the source-initiated case.



→ The valid data is placed on the data bus by the source by enabling 'data valid' signal.

→ Then the destination unit accepts the data from bus and disables 'ready for data'.

→ Then the source disables 'data valid' by invalidating the data on bus.

Scanned by CamScanner

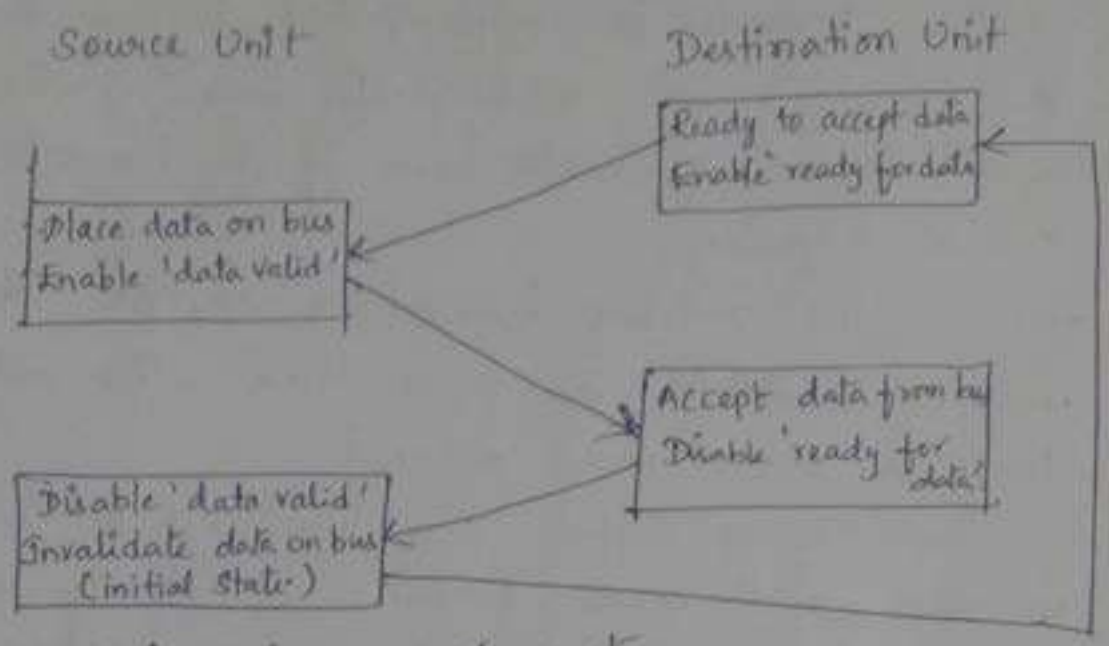


fig :- Sequence of events

Advantages of handshaking scheme

→ It provides a high degree of flexibility and reliability because the successful completion of a data transfer depends on active participation by both units.

→ If one unit is faulty, the data transfer will not be completed

How a faulty data transfer can be rectified?

→ B/w the source & destination unit if one of the unit is faulty the data transfer is not completed.

→ Such an error can be detected by means of a 'timeout' mechanism, which produces an alarm if the data transfer is not completed within a predetermined time.

Scanned by CamScanner

→ The timeout is implemented by means of a internal clock that starts counting time when the unit enables one of its handshaking control signals.

→ If the return handshake signal does not respond within a given time, the unit assumes that an error has occured.

→ The timing signal can be used to interrupt the processor and hence executes a service routine that takes the appropriate error recovery action.

Asynchronous Serial Transfer:-

→ The transfer of data between two units may be done in

(i) Parallel or.

(ii) Serial

parallel data transmission:-

→ In parallel data transmission, each bit of the message has its own path and the total message is transmitted at the same time.

→ n-bit message must be transmitted through n-separate conductor paths.

→ Parallel transmission is faster but requires many wires.

→ It is used for short distances and where speed is important.

Scanned by CamScanner

Serial data transmission:-

→ In this transmission, each bit in the message is sent in sequence one at a time.

→ It requires one pair of conductors or one conductor and a common ground.

→ It is slower but it is less expensive since it requires only one pair of conductors.

→ Serial transmission can be again divided into two type

① Synchronous Serial transmission

② Asynchronous Serial transmission.

Synchronous Transmission:-

→ In synchronous transmission, both units (sender and receiver) share a common clock frequency

→ Bits are transmitted continuously according to the clock pulses.

→ Synchronization signals are transmitted periodically between the two units to keep their clocks in step with each other.

Asynchronous Transmission.

→ In Asynchronous transmission, the binary information is sent only when it is available and the line remains idle where is there is no data.

→ Here the data or bits are sent in blocks

→ Individual clocks are Existed be for each sender

Scanned by CamScanner

and receiver unlike in Synchronous transmission, common clock is used to synchronous the transmission of information continuously.

→ Since, the Asynchronous transmission does not contain the common clock to send information between the sender and receiver, here we use special bits which are inserted at both ends of the character code.

→ Each character consists of

(1) Start bit
(2) Character bits
(3) stop bit.

→ The first bit in the character, is always 0. to indicate the beginning of a character.

→ The last bit in the character, is always 1. called as 'stop bit'

→ The transmitter rests at the 1-state when no characters are transmitted.

A transmitted character can be detected by the sender from knowledge of the transmission rules:

1) When a character is not being sent, the line is kept in the 1-state

2) The initiation of a character transmission is detected from the start bit, which is always 0.

Scanned by CamScanner

→ The character bit always follow the start bit.

→ After the last bit of the character is transmitted a stop bit is detected when the line returns to the 1-state for atleast one bit time.

Using the above rules, the receiver can detect the start bit when the line goes from 1 to 0.

→ At the end of the character the line is held at the 1-state for the period of atleast one or two bit times so that both transmitter and receiver can resynchronize.

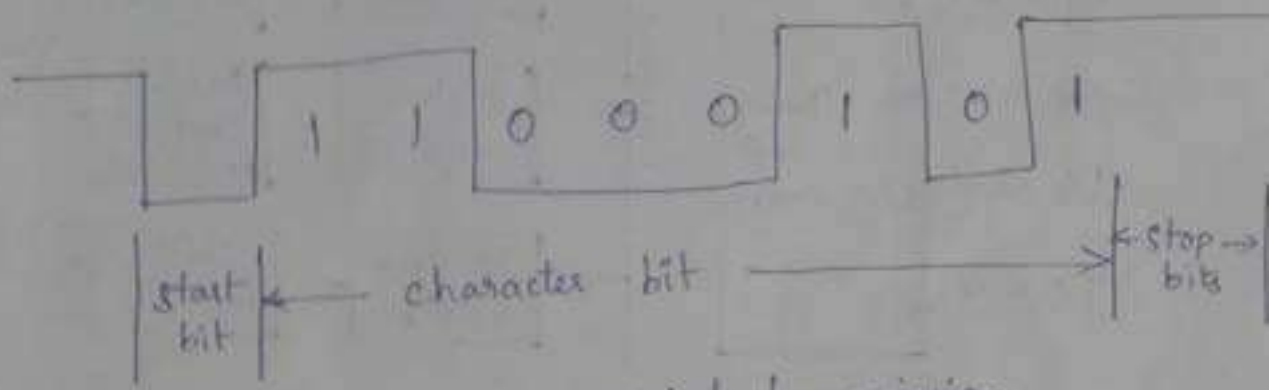
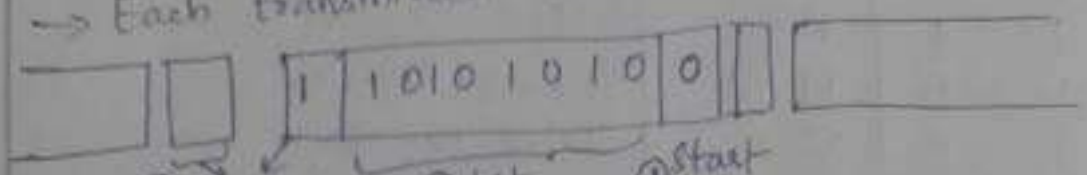


fig - Asynchronous serial transmission.

Eg:- Consider a terminal whose transfer rate = 10 characters per second

→ Each transmitted character contains 11 bits



② stop bits ⑧ bits ① start bit

2 | 8 | 1 = 11 bits (2+8+1)

→ 10 characters per second means that each character takes 0.1s for transfer.

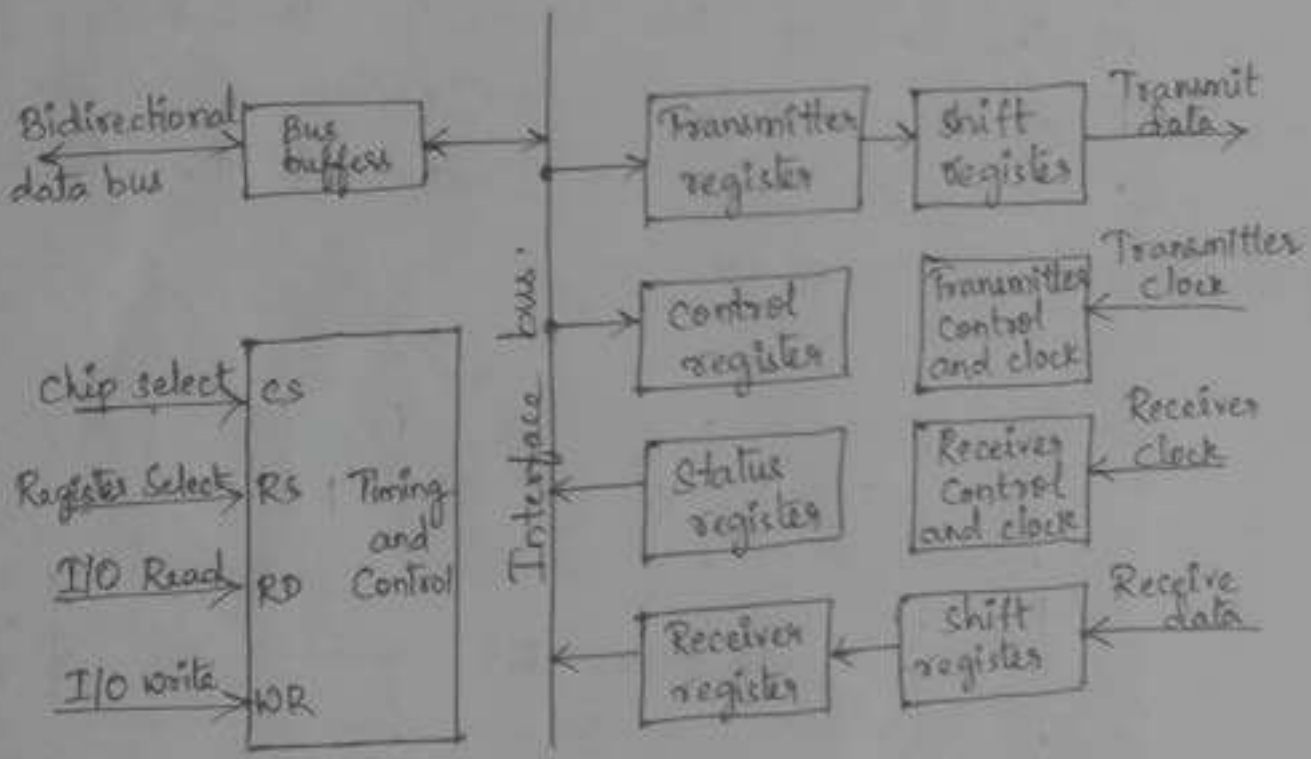
Scanned by CamScanner

→ Since, there are 11 bits, so, it follows bit time of 9.09 ms.

baud rate :-

→ The rate at which serial information is transmitted and is equivalent to the data transfer in bits per sec.

Asynchronous Communication Interface :-



CS	RS	Operation	Register selected
0	X	X	None : data bus in high-impedance
0	0	WR	Transmitter register
1	1	WR	Control Register.
1	0	RD	Receiver Register.
1	1	RD	Status Register.

fig:- Block diagram of a typical asynchronous communication interface.

Scanned by CamScanner

The above block diagram functions as both a transmitter and a receiver.

→ The interface is initialized for a particular mode of transfer by means of a control byte that is loaded into its control register.

→ The transmitter register accepts a data byte from the CPU through the data bus.

→ This byte is transferred to a shift register for serial transmission.

→ The receiver portion receives serial information into another shift register, after accumulating complete data byte it is transferred to receiver register.

→ The CPU can select the receiver register to read the byte through the data bus.

→ The bits in the status register are used for input and output flags and for recording certain errors that may occur during the transmission which can also be read by the CPU

→ The chip select and the read and write control lines communicate with the CPU.

→ The chip select (CS) input is used to select the interface through the address bus.

→ The Register select (RS) is associated with the read (RD) and write (WR) controls.

→ The register selected is a function of the RS value and the RD and WR status.

Scanned by CamScanner

Operation of asynchronous Communication Interface :-

→ The operation of asynchronous Communication interface is initialized by the CPU by sending a byte to the Control register.

→ The initialization procedure places the interface in a specific mode of operation as it defines certain parameters such as

(1) baud rate to use
(2) how many bits are in each character.
(3) whether to generate and check parity.
(4) How many stop bits are appended to each character.

→ Two bits in the status register are used as flags
- one bit to indicate whether transmitter register is empty.
- another bit to indicate whether the receiver register is full.

Transmitter :-

Operation of transmitter

→ The CPU reads the status register and checks the flags to see whether transmitter register is empty.

→ If it is empty, the CPU transfers a character to the transmitter register and interface clears the flag to mark the register as full.

Scanned by CamScanner

→ The first bit in the transmitter shift register is set '0' to generate start bit.
→ The character is transferred in parallel from the transmitter register to the shift register and appropriate number of stop bits are appended
→ Now, the transmitter register is marked empty.
→ The character can now be transmitted one bit at a time by shifting the data in the shift register at a specified baud rate.
→ The CPU can transfer another character to the transmitter register after checking the flag in the status register.
→ The interface is said to be double buffered because a new character can be loaded as soon as the previous one starts transmission.

Receiver :-

Operation of the receiver portion of the interface :-
→ The receive data input is in the 1-state when the line is idle.
→ The receiver control monitors the receive data line for a 0 signal to detect the occurrence of a start bit.
→ Once the start bit is detected, the incoming bits of the character are shifted into shift register at certain/prescribed baud rate.
→ After receiving the data bits, the interface checks for the parity and stop bits.

Scanned by CamScanner

→ The character without start and stop bits is then transferred in parallel from the shift register to the receiver register.
→ The flag in the status register is set to indicate that the receiver register is full.
→ The CPU reads the status register and checks the flag, and if set, it reads the data from the receiver register.

Errors during transmission :-

→ The interface checks the possible errors during transmission and sets appropriate bits in the status register.
→ The CPU can read the status register at any time check if any errors occurred.
→ Three possible errors which is checked by interface during transmission are

i) Parity error
ii) Framing error
iii) Overrun error.

→ Parity error occurs if the number of 1's in the received data is not the correct parity.
→ A framing error occurs if the right number of stop bits is not detected at the end of the received character.
→ A overrun error occurs if the CPU does not read the character from the receiver register before the next one arrives in the shift register. It results in loss of character.

Scanned by CamScanner

First-In, First-Out Buffer

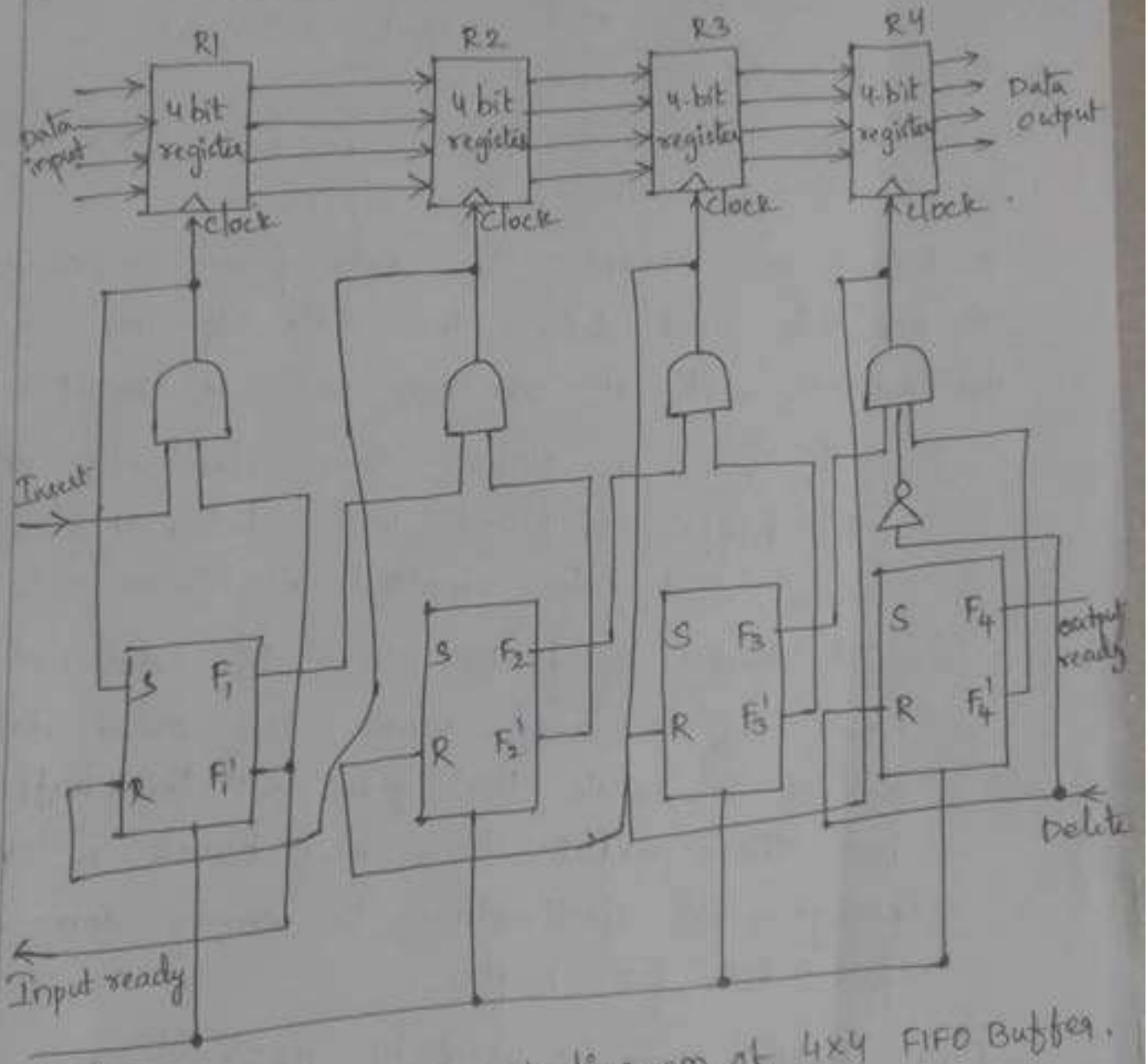


fig :- circuit diagram of 4x4 FIFO Buffer.

→ A first in, first out (FIFO) buffer is a memory unit that stores information in such a way that the item first in is the item first out.

→ A FIFO buffer comes with separate input and output terminals.

Scanned by CamScanner

→ The important feature of this buffer is that it can input data and output data at two different rates of transfer.

→ The output data is always in the same order in which data entered the buffer.

→ FIFO can accept the data from source unit at one rate and deliver the data to the destination unit at another rate of transfer.

(i) If source is slower than destination unit, the buffer is filled with data at slow rate and later emptied at higher rate.

(ii) If source is faster than the destination, Now FIFO is used where the source data arrives in bursts that fills out the buffer but there exists time b/w bursts is long enough for destination to empty some or full data from buffer.

→ FIFO buffer can be used in applications where the data is transferred asynchronously.

→ The logic diagram of a typical 4x4 FIFO buffer consists of 4-bit registers R_I, $I = 1,2,3,4$ and control register with flip-flops F_i $i = 1,2,3,4$ one for each register.

Scanned by CamScanner

→ The FIFO stores four words of four bits each.

→ A flip flop F_i in the control register that is set to 1 indicates that a 4-bit data word is stored in corresponding register RI.

$F_i = 0$ - register does not contain valid data

→ Control register directs the movement of data through the registers.

→ F_i bit of control register is set ($F_i = 1$) and F_{i+1} bit is reset ($F'_{i+1} = 1$), a clock is generated causing register R(I+1) to accept the data from register RI.

→ The same clock transition sets F_{i+1} to 1 and resets F_i to 0. This causes the control flag to move one position to the right together the data unit there are empty locations ahead of it.

→ This ripple-through operation stops when the data reach a register RI with the next flip-flop F_{i+1} being set to 1 or at the last register R4.

→ Data are inserted into buffer provided that the 'input ready' signal is enabled which occurs by setting Flip-flop F_1 is reset. (ie., R1 is empty).

→ 'insert' control line is used for loading data from input lines by enabling clock in R1.

→ The same clock sets F_1, which disables the 'input ready' control.

Scanned by CamScanner

→ Now ripple-through process begins provided that R2 is empty

→ The data is transferred from R1 to R2 and F1 is cleared, which indicates that input ready line is now available for R1 which now accepts the another data word.

→ If FIFO is full, F1 remains set and 'input ready' lines stays in '0' state.

→ 'The output ready' control line is enabled when the last control flip flop F4 is set indicates valid data in register R4.

→ After accepting the data in R4 by destination unit, it enables 'delete control signal', F4 is reset causing 'output ready' is disabled.

→ Only after 'delete signal' goes to 0 then only the data from R3 move to R4.

'input ready' & 'insert' } Constitute → destination initiated pair of handshake.

'output ready' & 'delete' } Constitute → Source initiatrated pair of handshake.

Scanned by CamScanner

Modes of transfer

→ Binary information received from an external device is usually stored in memory for later processing.

→ The CPU merely executes the I/O instructions and may accept the data temporarily, but the ultimate source or destination is memory.

→ The data transfer b/w the central computer and I/O devices may be handled in a variety of modes.

→ The three possible modes from of data transfer from or to peripherals are.

(1) Programmed I/O

(2) Interrupt-initiated I/O.

(3) Direct Memory Access (DMA).

Programmed I/O :-

→ Programmed I/O operations are the part of instructions of a computer program.

→ Each data transfer is initiated by an instruction in the program.

→ The transfer is to and from CPU register and peripherals

→ Transfering data under program control requires constant monitoring of the peripheral by the CPU.

→ The CPU stays in a program loop until the

Scanned by CamScanner

I/O unit indicates that it is ready for transfer.

→ This is a time-consuming process since it keeps the processor busy needlessly.

→ It is avoided by using the interrupt facility and special commands to inform the interface to issue an interrupt request signal when the data are available from the device.

Interrupt-Initiated I/O :-

→ In programmed I/O, the processor keeps monitoring the interface of peripherals which is waste of time. Instead it is avoided by using interrupt facility and special commands to inform the interface to issue interrupt request signal when the data is available.

→ In the mean time the CPU can proceed the execute another program.

→ The interface keeps monitoring the device, when the device is ready for data transfer then the interface generates a interrupt request to the computer.

→ After detecting the external interrupt request signal, the CPU momentarily stops the task it is processing, branches to a service program to process the I/O transfer, and returns back to task after the transfer.

Scanned by CamScanner

Direct Memory Access :- (DMA)

→ In DMA, the interface transfers data into and out of the memory unit through the memory bus.

→ The CPU initiates the transfer by supplying the interface with the starting address and the no. of words needed to be transferred and then CPU proceeds to execute the other tasks.

→ when the transfer is made, the DMA requests memory cycles through the memory bus.

→ when the requested is granted by memory controller, the DMA transfers the data directly into memory.

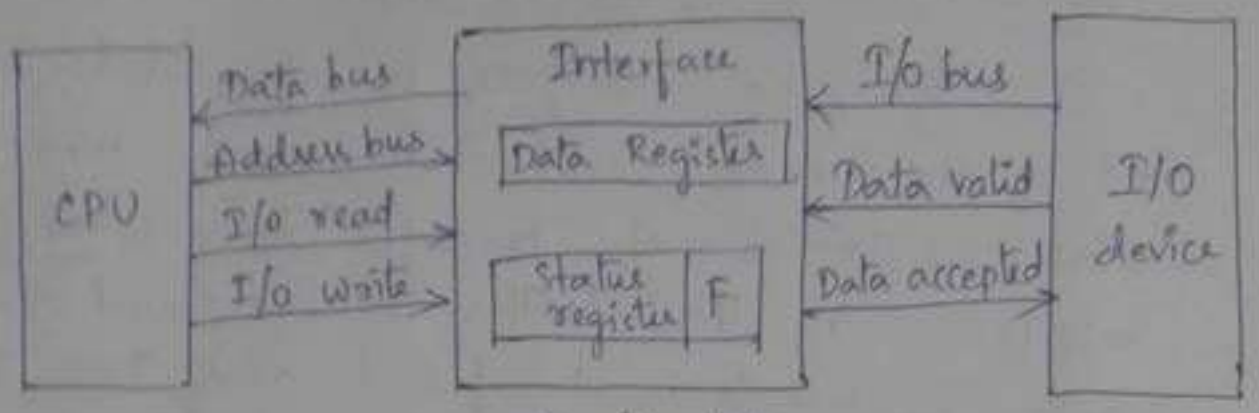
→ If any memory access operations are there with CPU, it just delays to allow the direct memory I/O transfers.

Example of Programmed I/O :-

→ In the programmed I/O method, the I/O device does not have direct access to memory.

→ A transfer from I/O to memory requires the execution of several instructions by the CPU including an input instruction to transfer data from the device to the CPU and a store instruction to transfer the data from CPU to memory.

Scanned by CamScanner



F = flag bit

fig :- Data transfer from I/O device to CPU.

→ The I/O device transfers bytes of data one at a time as they are available.

→ when the bytes of data are available then the device places it in the I/O bus and enables its 'data valid line'.

→ The interface accepts the byte into data register which is referred as flag bit or 'F', now interface disable data valid line

→ The device does not transfer any further byte or words until the data accepted line is disabled by the interface.

→ The program is written for the computer to check the flag in status register to determine whether the device placed the data byte in the data register.

→ If the flag is equal to 1, the CPU reads the data from the data register.

Scanned by CamScanner

If the flag bit is cleared to 0, the interface disables the data accepted line and the device can then transfer the next data byte.

→ The transfer of each byte requires three instruction

1) Read the status register
2) Check the status of the flag bit and branch to step 1 if not set or to step 3 if set
3) Read the data register.

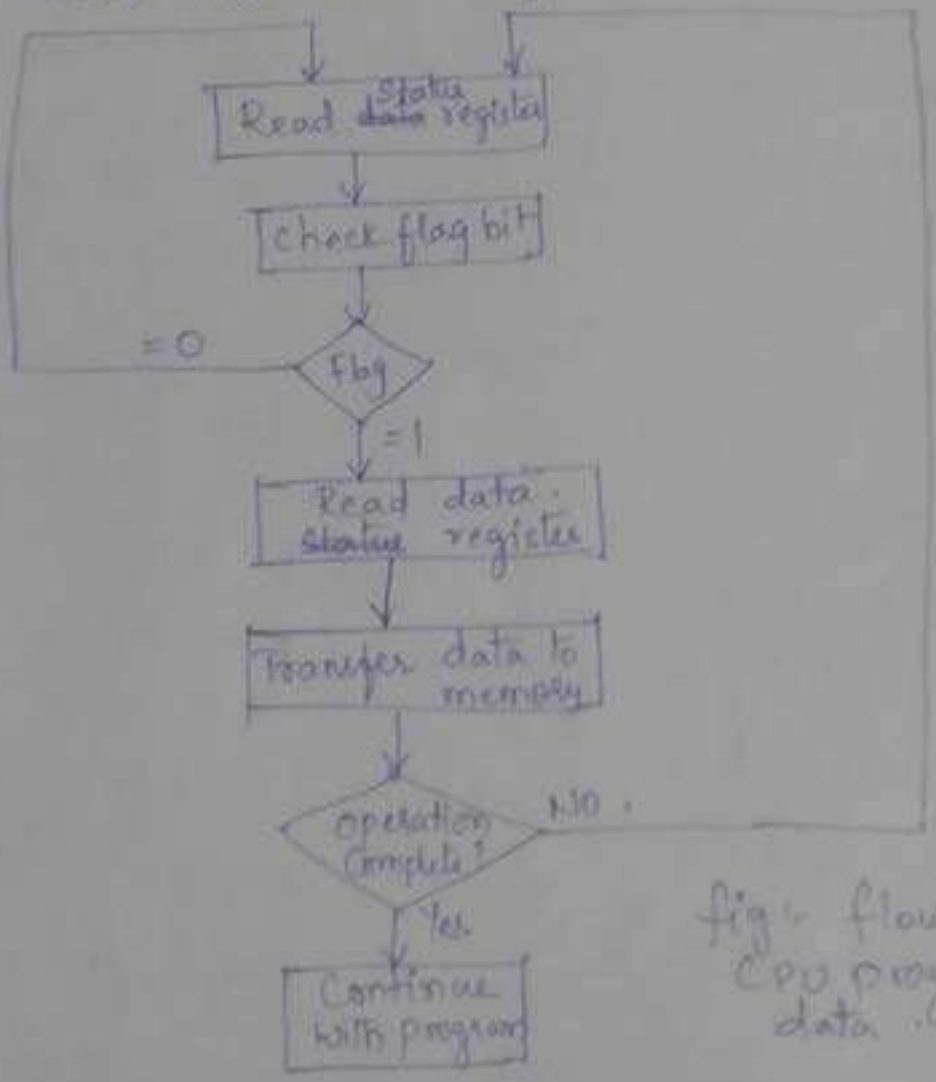


fig:- flowchart for CPU program to input data.

→ The programmed I/O method is particularly useful in small low-speed computers.

→ The difference in data transfer rate between the CPU and I/O device makes this type of transfer inefficient.

Scanned by CamScanner

Each byte is read into a CPU register and then transferred to memory with store instruction.

Interrupt - Initiated I/O

→ CPU constantly monitoring the flag bit to let the Interface inform the computer when it is ready to transfer data.

→ Programmed I/O is an time-consuming process, so in order to keep a CPU in busy state introduce an interrupt and special commands to inform the interface to issue an Interrupt signal when data are available from the device.

→ In the mean time, the CPU can proceed to execute another program

→ When the device is ready for data transfer, it generates an interrupt request to the Computer

→ when there is an external interrupt signal is generated, the CPU will stops the execution of the original program and branches to service program to process the I/O transfer and then returns back to the original program

→ The CPU responds to interrupt signal by storing the return address from the program counter into memory stack and then control branches to a service routine that processes the required I/O transfer.

Scanned by CamScanner

The way that the processor chooses the branch address of the service routine varies.

→ In principle, there are two methods for accomplishing

(1) Vectored Interrupt
(2) Non-Vectored interrupt.

→ In Non-vectored interrupt, the branch address is assigned to a fixed location in memory.

→ In a Vectored interrupt, the source that interrupts supplies the branch information to the computer. This information is called the 'interrupt vector'.

Software considerations :-

→ The previous topics deal with basic hardware needed to interface I/O devices to a computer system

→ A computer must also have software routines for controlling peripherals and for transfer of data between the processor and peripherals.

→ I/O routines must issue control commands to activate the peripheral and to check the device status to determine when it is ready for data transfer.

→ In some cases, command is given to execute a device function such as stop tape or print character.

Scanned by CamScanner

→ In Interrupt Controlled transfers, the I/O software must issue commands to the peripheral to interrupt when ready and to service the interrupt when it occurs.

→ I/O routines are usually available as operating system procedures and the user refers to the established routines to specify the type of transfer without going deep into machine language programs.

Priority Interrupt :-

→ Data transfer between the CPU and an I/O device is initiated by the CPU.

→ However, the CPU cannot start the transfer unless the device is ready to communicate with CPU.

→ The readiness of the device can be determined from an interrupt signal.

→ The CPU responds to the interrupt request by storing the return address from PC to memory stack and then the program branches to the service routine to processes the transfer.

→ In a typical application a number of I/O devices are attached to the computer, with each device being able to originate an interrupt request.

→ The first task of the interrupt system is to identify the source of the interrupt.

→ There is possibility that several sources will request service simultaneously.

Scanned by CamScanner

→ In such case the system must also decide which device to service first.

Priority Interrupt :-

Def :- A priority interrupt is a system that establishes a priority over the various sources to determine which condition is to be serviced first when two or more requests arrive simultaneously.

→ Devices with high speed transfers such as magnetic disks are given high priority, and slow devices such as keyboard receive low priority.

→ When two devices interrupt at the same time, the higher priority device is serviced first.

→ If higher priority interrupt devices are ignored or delayed could have serious consequences.

→ Establishing the priority of interrupts can be done by software or hardware.

Polling :-

→ A polling procedure is used to identify the highest priority source by software.

→ In this method there is one common branch address for all interrupts.

→ The program that takes care of interrupts begins at the branch address and polls the interrupt sources in sequence.

Scanned by CamScanner

→ The highest priority source is tested first, and if its interrupt signal is on, control branches to a service routine for this source.
→ otherwise, the next-lower priority source is tested, and so on.
→ Thus the initial service routine for all interrupts consists of a program that tests the interrupt sources in sequence and branches to one of many possible service routines.

Disadvantage :-

The disadvantage of the software method is that if there are many interrupts, the time required to poll them can exceed the time available to service the I/O device.

Hardware priority - Interrupt Unit :-

→ It accepts interrupt requests ~~has~~ the ~~bio~~ from many sources, determines which has highest priority and issues an interrupt request.
→ To speed the operation, each interrupt source has its own interrupt vector to access its own service routine directly.
→ Thus no polling is required because all decision are established by the hardware priority - interrupt unit.
→ It can be established by serial or parallel connection of interrupt lines.

Scanned by CamScanner

→ The serial connection is known as 'daisy chaining method'.

Daisy-chaining Priority:-

→ The daisy-chaining method of establishing priority consists of a serial connection of all devices that request an interrupt.

→ The device with the highest priority is placed in the first position followed by lowest priority placed in the last chain.

→

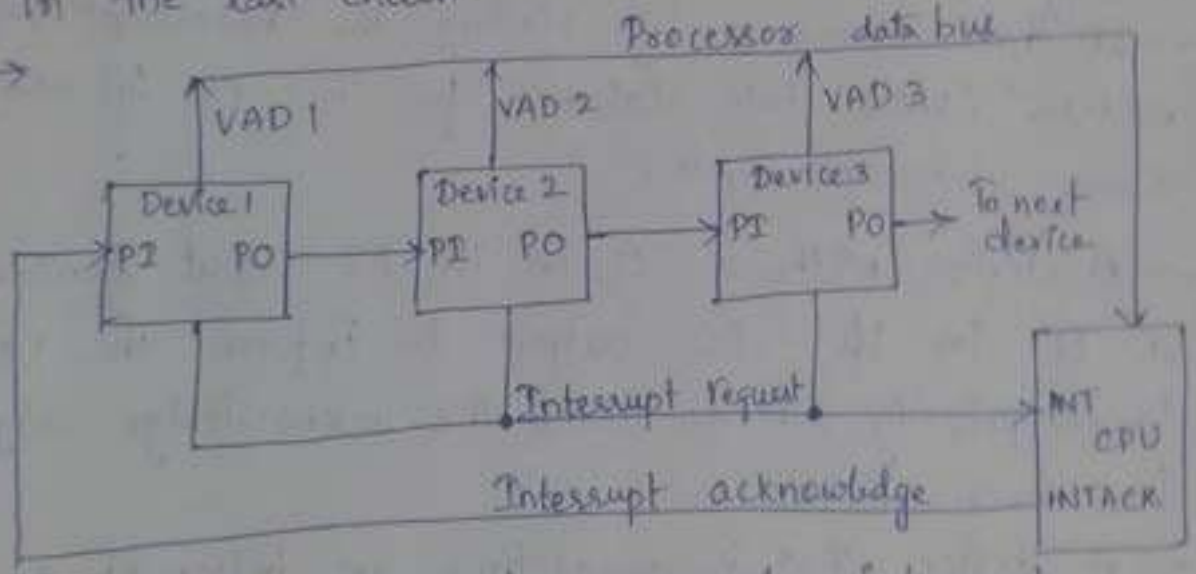


fig:- Daisy-chain priority interrupt.

→ The interrupt request line is common to all devices.

→ If any device has its interrupt signal in the low-level state, the interrupt line goes to low level state and enables the interrupt input in the CPU.

→ When no interrupts are pending, the interrupt line stays in the high-level state and CPU does not recognize any interrupts.

Scanned by CamScanner

→ The CPU responds to an interrupt request by enabling the interrupt acknowledge line.

→ This signal is received by device 1 at its PI

→ The acknowledge signal passes on to the next device through the PO output only if device 1 is not requesting ~~t~~ an interrupt.

→ If device 1 has an pending interrupt, it blocks the acknowledge signal from the next device by placing a '0' in PO output

→ It then proceeds by placing the 'interrupt vector address' (VAD) into databus for the CPU to use during interrupt cycle.

→ A device with a 0 in its PI input generates a 0 in its PO output to inform the next lower priority device that the acknowledge signal has been blocked.

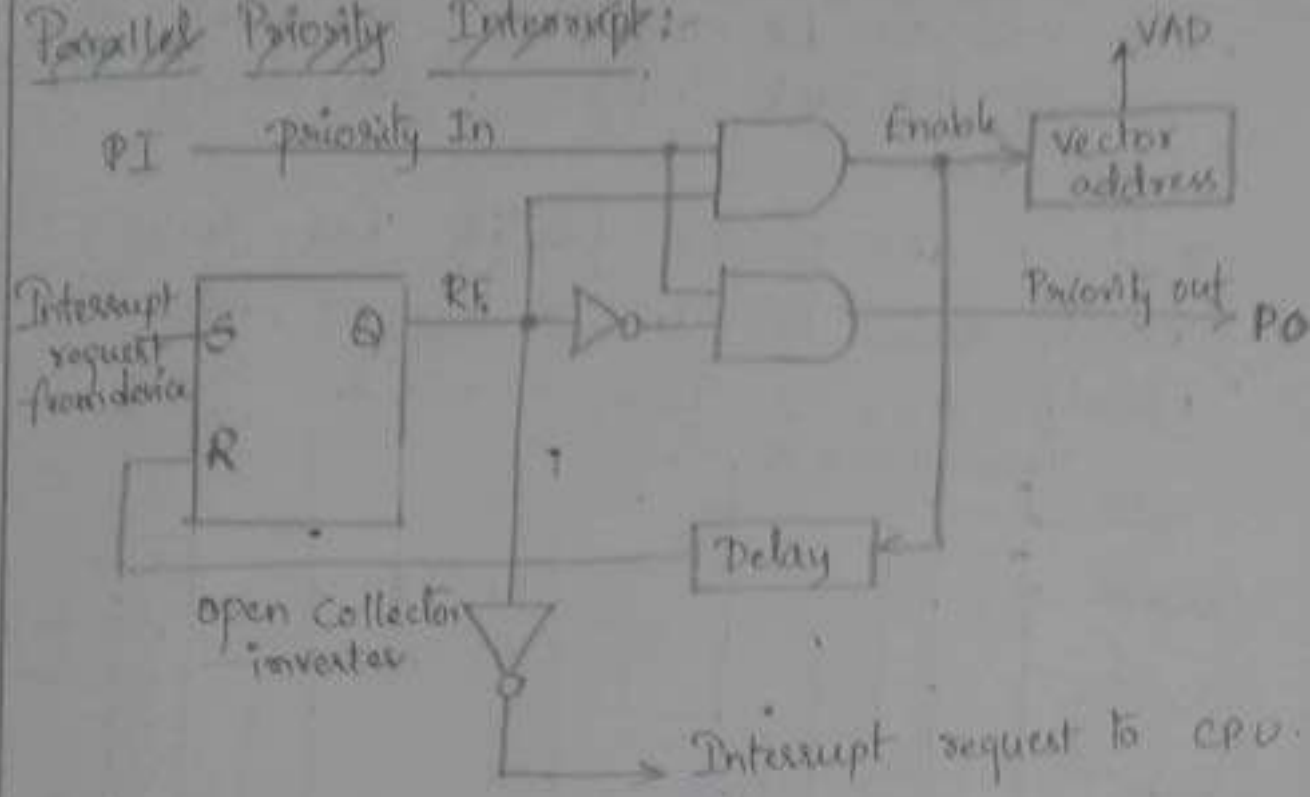
→ A device that is requesting an interrupt and has a 1 in its PI input will intercept the acknowledge signal by placing 0 in its PO output

→ If the device does not have pending interrupt it transmits the acknowledge signal to the next device by placing a 1 in its PO output.

→ Thus the device with PI = 1 and PO = 0 is the one with the highest priority that is requesting an interrupt, and this device places its VAD on the data bus.

Scanned by CamScanner

Parallel Priority Interrupt:



PI	RF	PO	Enable
0	0	0	0
0	1	0	0
1	0	1	0
1	1	0	1

fig:- One stage of daisy chain priority arrangement

Parallel Priority Interrupt:-

→ The parallel priority interrupt method uses a register whose bits are set separately by the interrup signal from each device.

→ Priority is established according to the position of the bits in the register.

→ In addition to the interrupt register, the circuit may include a mask register whose purpose is to control the status of each interrupt request.

Scanned by CamScanner

→ The mask register can be programmed to disable lower-priority interrupts while a higher-priority device is being serviced.

→ It can also provide facility that allows a higher priority device to interrupt the CPU while a lower-priority device is being serviced.

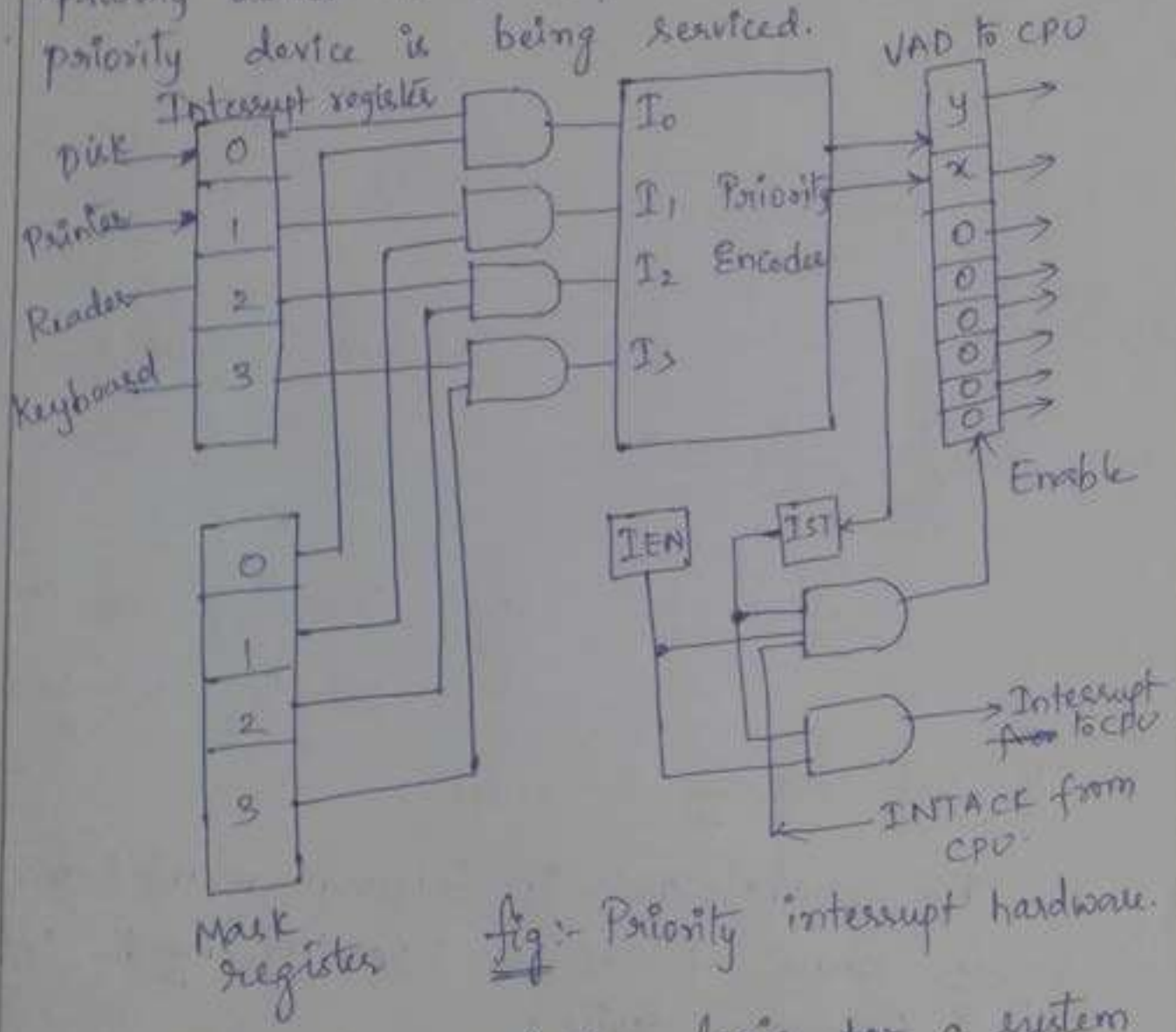


fig:- Priority interrupt hardware.

The above priority logic for a system of four interrupt sources consists of an interrupt register whose bits are set by external conditions and cleared by program instructions.

Scanned by CamScanner

→ In the fig, the ~~pas~~ magnetic disk (high speed device) given highest priority and followed by the next least priorities, printer, character reader and keyboard.

→ The mask register has same bits of interrupt register.

→ By using program instructions we can set or reset any bit in mask register.

→ Each interrupt bit and its corresponding mask register bit are applied to AND gate to produce four ~~interrupts~~ inputs to priority encoder.

→ The priority encoder generates two bits of vector address, which is transferred to CPU.

→ Another o/p from the encoder sets an interrupt status flip-flop IST, when an interrupt that is not masked occurs.

→ IEN (Interrupt Enable Flip-flop) set or cleared by the pgm to provide an overall control over the interrupt system.

→ The outputs of IST and IEN 'ANDed' to provide a common interrupt signal for the CPU.

→ The interrupt acknowledge (INTACK) signal from the CPU enables the bus buffers in the output register and a vector address VAD is placed on data bus.

Scanned by CamScanner

What is priority Encoder :-

→ The priority Encoder is a circuit that implements the priority function.

→ The logic of the priority Encoder is "two or more inputs arrive at the same time, the input having the highest priority is given precedence".

Inputs				outputs			Boolean function
I_0	I_1	I_2	I_3	x	y	IST	
1	x	x	x	0	0	1	
0	1	x	x	0	1	1	$x = I_0' I_1'$
0	0	1	x	1	0	1	$y = I_0' I_1 + I_0' I_2'$
0	0	0	1	1	1	1	$(IST) = I_0 + I_1 + I_2 + I_3$
0	0	0	0	x	x	0.	

→ I_0 has highest priority, If $I_0 = 1$, the output generated is $xy = 00$

→ I_1 is next priority level. output is 01 if $I_1 = 1$, $I_0 = 0$ (regardless of values of two other lower-priority inputs).

→ IST = 1 when one or more inputs are equal to 1., if all inputs are 0, IST = 0

→ Boolean function is the internal logic of the encoder.

Scanned by CamScanner

Interrupt cycle:-

→ The interrupt enable flip-flop IEN can be set or cleared by program instructions.

→ when IEN is cleared, the interrupt request coming from IST is neglected by the CPU.

→ The program controlled IEN bit allows the programmer to choose whether to use the interrupt facility.

→ If an instruction to clear IEN has been inserted in the program, it means that the user does not want his program to be interrupted.

→ If IEN is set, indicates that the interrupt facility will be used while the current pgm is running.

→ Most Computers clears IEN = 0 by using internal hardware every time an interrupt is acknowledged by the processor.

→ At the end of each instruction cycle the CPU checks IEN and the interrupt signal from IST. If either of (IEN) or (IST) is 0 control continues with next instruction if not i.e; IEN = IST = 1 then the CPU goes to interrupt cycle.

→ The micro operations performed during the interrupt cycle by CPU are.

Scanned by CamScanner

$SP \leftarrow SP - 1$	Decrement stack pointer
$M[SP] \leftarrow PC$	push PC into stack.
$INTACK \leftarrow 1$	Enable interrupt acknowledge.
$PC \leftarrow VAD$	Transfer vector address to PC
$IEN \leftarrow 0$	Disable further interrupts
Go to fetch next instruction.	

Software Routines :-

→ The priority interrupt system is a combination of hardware and software techniques.

→ The computer must have software routines for servicing the interrupt requests and for controlling the interrupt hardware registers.

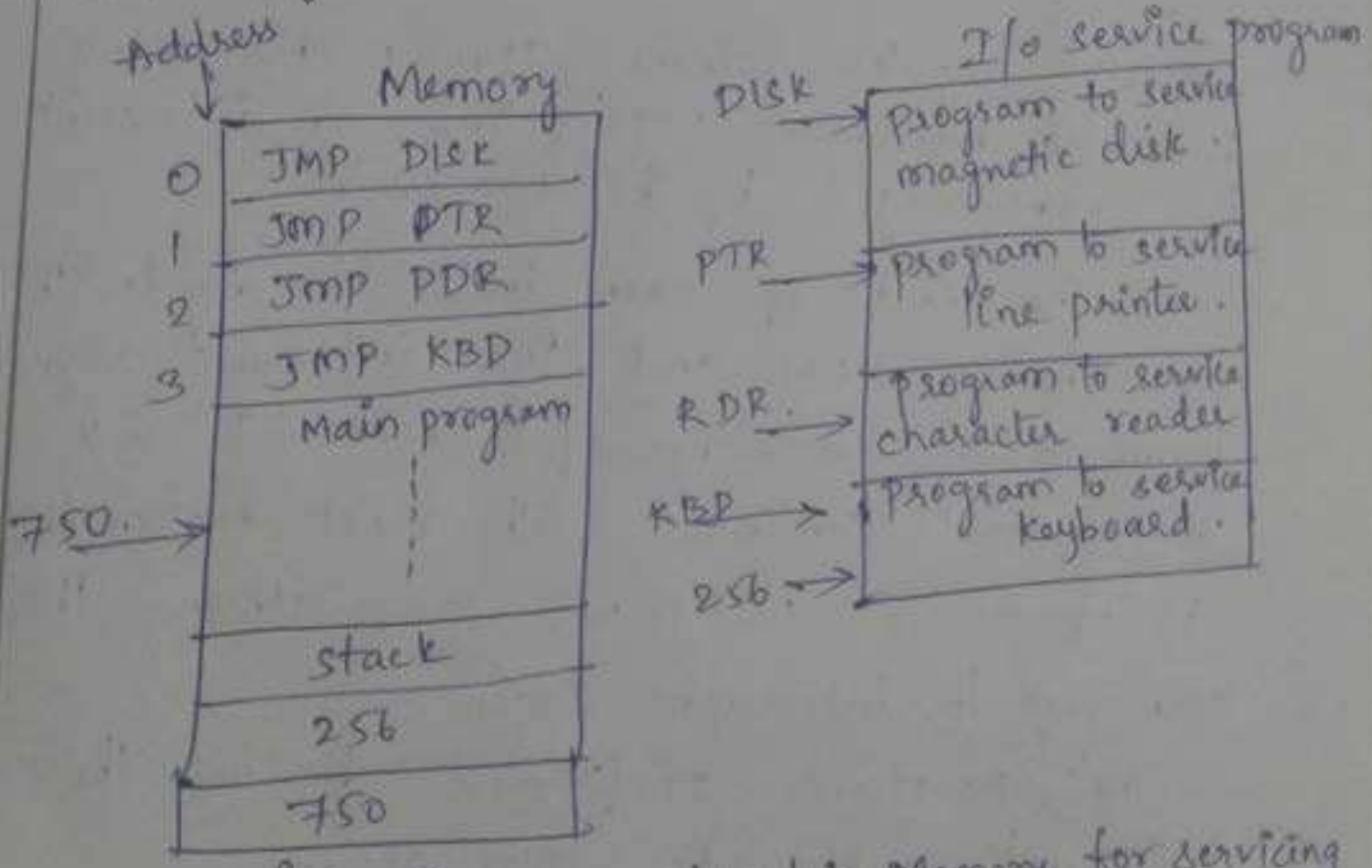


fig :- Programs stored in Memory for servicing interrupts.

Scanned by CamScanner

→ Each device has its own service program that can be reached through a Jump (JMP) instruction stored at the assigned vector address.

→ The SYMBOLIC NAME of each routine represents the starting address of the service program.

→ The stack is used for storing the return address after each interrupt.

Initial and final operations

→ Each interrupt service routine must have an Initial and final set of operations for controlling the registers in the hardware interrupt system.

→ IEN is cleared to 0 at the end of interrupt cycle.

→ IEN is set again to enable highest-priority interrupt requests, but not before lower-priority interrupts are disabled.

→ The initial sequence of each interrupt service routine must have instructions to control the interrupt hardware in following manner :-

1) clear lower-level mask register bits.
2) clear Interrupt status bit IST.
3) Save contents of processor registers.

Scanned by CamScanner

4) Set IEN = 1 (Interrupt Enable bit)

5) Proceed with service routine.

→ The lower-level mask register bits are cleared to prevent these conditions from enabling the interrupt.

→ ~~Although~~ IST is cleared to 0, so that it can be set when a higher-priority interrupt occurs.

→ processor registers are saved because they may be needed by the program when control returns back.

→ IEN is then set to allow other (higher-priority) interrupts and the computer proceeds to service the interrupt request.

The final sequence in each interrupt service routine must have instructions to control the interrupt hardware is :-

1) clear interrupt enable bit IEN

2) Restore contents of processor registers.

3) clear the bit in the interrupt register belonging to the source that has been serviced.

4) Set lower-level priority bits in the mask register.

5) Restore Return address into PC and set IEN.

Scanned by CamScanner

Direct Memory Access (DMA) :-

The transfer of data between a fast storage device such as magnetic disk is often limited by the speed of the CPU.

→ Removing the CPU from the path and letting the peripheral device manage the memory buses directly would improve the speed of transfer.

→ This transfer technique is called DMA.

→ During DMA transfer, the CPU is idle and has no control of the memory buses.

→ A DMA controller takes over the buses to manage the transfer directly between the I/O device and memory.

→ The CPU may be in idle state in many ways. one method is used in microprocessors is to disable the buses through special control signals.

→ The control signals of the CPU to facilitate the DMA transfer is.

Bus request (BR) :-

Bus request input is used by the DMA controller to request the CPU to get control of buses. when this is input is active, the CPU terminates the execution of the current instruction and places the address bus, data bus

Scanned by CamScanner

and read and write lines into the high-impedance state.

Bus Grant :-

The CPU activates the bus grant (BG) o/p to inform the external DMA that the buses are in the high-impedance state. Then the DMA will take over the control of the buses without any processor intervention.

→ When the DMA terminates the transfer, it disables the bus request line, the CPU disables the bus grant, takes control of the buses and return to its normal operation.

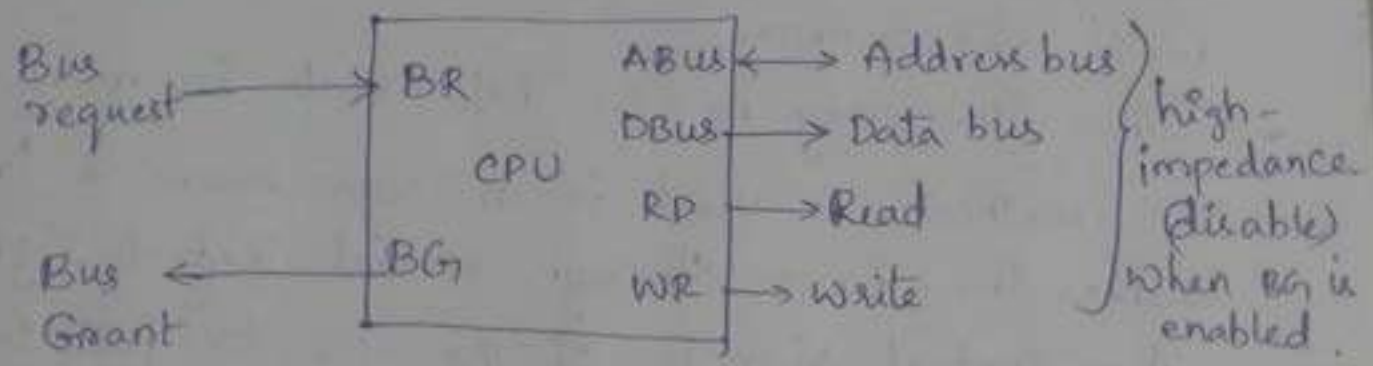


fig :- CPU bus signals for DMA transfer

→ When the DMA takes control of the bus system, it communicates directly with the memory.

→ The transfer can be done in different ways:

1) Burst Transfer :- In this a block sequence consisting of a number of memory words is transferred in a continuous burst while the

Scanned by CamScanner

DMA Controller is master of all the buses. This mode of transfer can be used by fast device such as magnetic disks etc.

2) Cycle Stealing:-

In this, the DMA controller to transfer one data word at a time, after which it must return control of the buses to the CPU. The CPU delays its operation for one memory cycle to allow the direct memory I/O transfer to steal one memory cycle.

DMA Controller:-

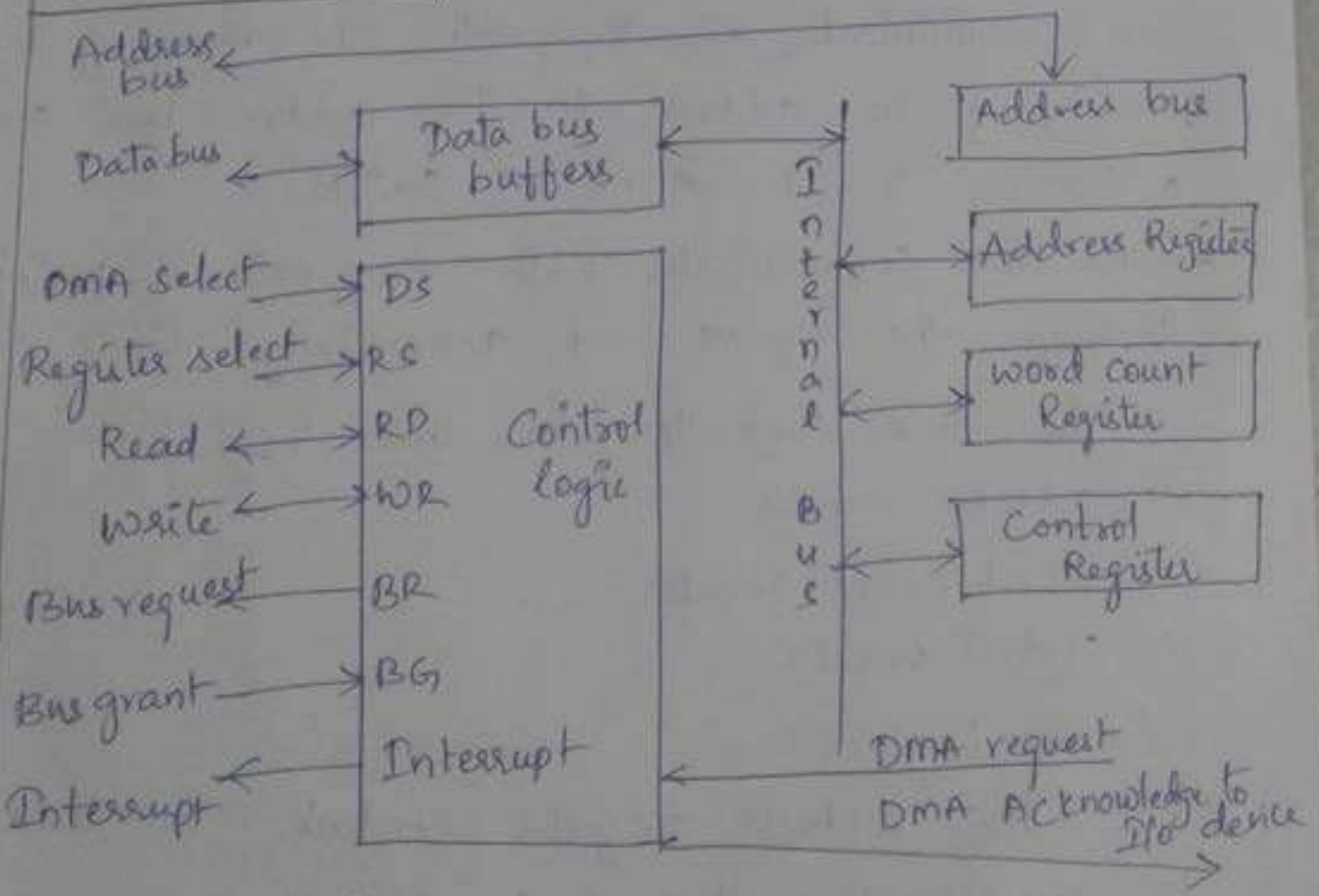


fig:- DMA Controller.

Scanned by CamScanner

→ The Communication is done with CPU via the data bus and control lines.

The registers in the DMA are selected by the CPU, through the address bus by enabling the DS (DMA select) and RS (register select) inputs.

→ The read and write inputs are bidirectional.

→ when the BG (bus grant) input is 0, the CPU communicate with the DMA registers through the data bus to read from or write to the DMA transfer registers.

→ when BG = 1, the CPU has the buses and the DMA can communicate directly with the memory by specifying an address in the address bus and activating the RD or WR control.

→ The DMA communicates with the external peripheral through the request and acknowledge lines.

The DMA Controller has three registers:

(1) address register.

(2) word count register.

(3) control register

Address Register :-

The Address Register contains an address to specify the desired location in memory.

→ The address bits go through bus buffers into

Scanned by CamScanner

the address bus.

→ The address register is incremented after each word that is transferred to memory.

Word Count :-

→ The word count register holds the number of words to be transferred. This register is decremented by one after each word is transferred.

Control Register :-

→ The control register specifies the mode of transfer

→ All the registers in the DMA appear to the CPU as I/O interface.

→ Thus the CPU can read from or write into the DMA registers under program control via databus.

→ The CPU initiates the DMA by sending the following instruction information through the databus:

(1) The starting address of the memory block where data are available (for read) or where data are to be stored (for write).

2) The word count, which is the number of words in the memory block.

(3) Control to specify the mode of transfer such as read or write.

(4) A control to start the DMA transfer.

Scanned by CamScanner

DMA transfer:

→ The position of the DMA controller among the other components in a system is illustrated is shown in below fig:

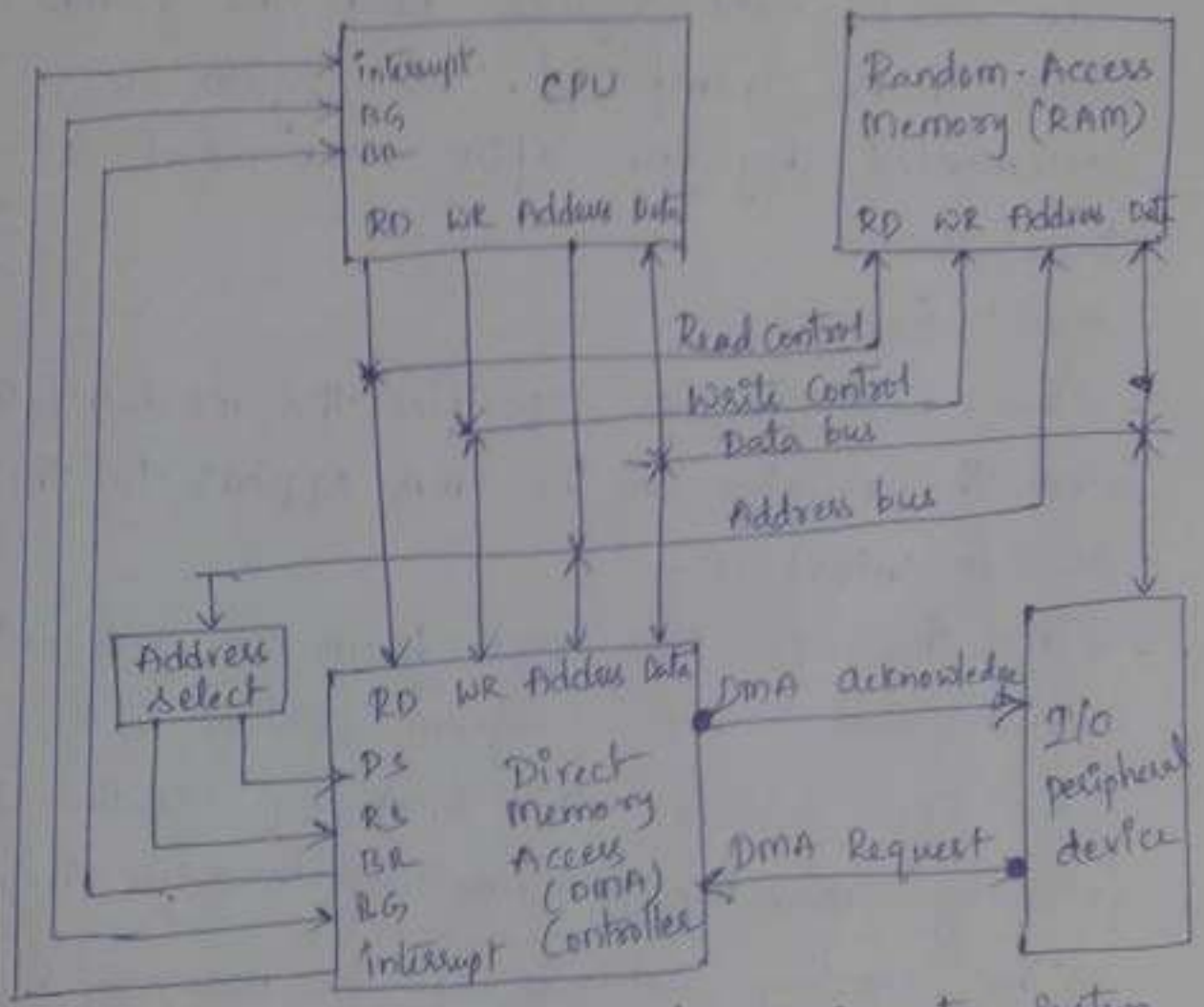


fig:- DMA transfer in a Computer System.

The CPU communicates with the DMA through the data and address buses with any interface unit.

→ The DMA has its own address, which activates the DS and RS lines. The CPU initializes the DMA through the data bus.

Scanned by CamScanner

→ Once the DMA receives the start control command, it can start the transfer between the peripheral device and the memory

→ when the peripheral devices sends a DMA request, the DMA controller activates the BR line informing the CPU to relinquish the buses.

→ The CPU responds with its BG line, informing the DMA that its buses are disabled. The DMA then puts the current value of its address register into the address bus, initiates the RD or WR signal and sends a DMA acknowledge to the peripheral device.

→ The RD & WR are bidirectional lines.

→ The direction of transfer depends on status of BG line when BG=0, the RD and WR are i/p lines allowing CPU to communicate with internal DMA registers.

→ BG=1 the RD and WR are o/p lines from DMA to RAM to specify read or write operations.

→ When peripheral devices receives a DMA receives acknowledge, it puts a word in the data bus (write) or receives a word from data bus (Read)

→ DMA controls read and write operations and supplies address for the memory.

Scanned by CamScanner

→ The peripheral unit then can communicate with RAM thro' data bus for direct transfer b/w two units while CPU is momentarily disabled.

→ for each word transferred by DMA, Increments its address register and decrements its word count register.

→ word Count does not reach 0, the DMA checks the request line coming from peripheral.

→ If the word count reach 0, the DMA stops further transfer and removes its bus request. (DMA Confirms from CPU thro' interrupt

→ The CPU responds to the interrupt given by the DMA and reads the word count register.

→ the '0' value in word count register indicates that all words are transferred successfully.

→ The CPU can read the word count registers at any time.

→ DMA Controller may have more than one channel. In this case, each channel has request and acknowledge pair of control signals which are connected to separate peripheral devices.

Scanned by CamScanner

Advantages of DMA

→ It is used for fast transfer of information between magnetic ~~to~~ disks and memory.

→ It is also useful for updating the display in an interactive terminal.

→ An image of screen display of the terminal is kept in memory which can be updated under program control.

→ The contents of memory can be transferred to the screen periodically by means of DMA transfer.

Scanned by CamScanner

Memory Organization

Memory Hierarchy :-

-> The Memory unit is an essential Component in any digital Computer since it is needed for storing programs and data

-> Most general-purpose Computers would run more efficiently if they were equipped with additional storage beyond the capacity of the main memory

-> Most Computers accumulate and continue to accumulate large data-processing software which is not all required by the ~~accumul~~ processor. So we use low-cost storage devices to backup for storing data which is not currently used by CPU

Main Memory :-

-> The memory unit that communicates directly with the CPU is called the main memory

-> Only programs and data currently needed by the processor reside in main memory

Auxiliary Memory :-

-> Devices that provide backup storage are called auxiliary memory.

Eg :- Magnetic disks, Magnetic tapes.

Scanned by CamScanner

→ All other information is stored in auxiliary memory and transferred to main memory when needed.

→ The total memory capacity of a computer can be visualized as being a hierarchy of components

→ The memory hierarchy system contains all storage devices from the slow but high capacity auxiliary memory to a fast main memory to an even smaller and faster cache memory accessible to the high-speed processing logic.

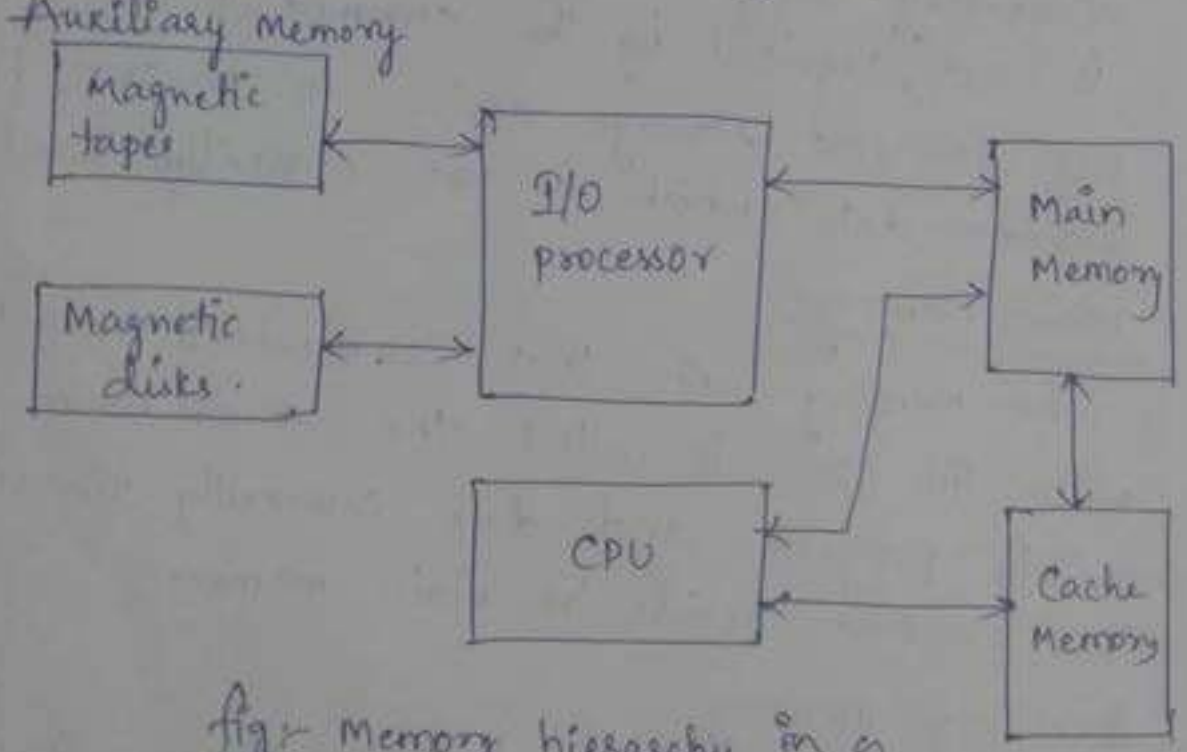


fig:- Memory hierarchy in a Computer system

→ At the bottom of hierarchy are the relatively slow magnetic tapes to store removable files. next

Scanned by CamScanner

magnetic disks to store backup data. The main memory occupies a central position by being able to communicate directly with the CPU and auxiliary memory through I/O processor.

→ A special very-high speed memory called a 'cache' is sometimes used to increase the speed of processing by making current programe and data available to the CPU at a rapid rate.

→ In order to compensate the speed differences between main memory access time and processor logic.

→ CPU logic usually faster than main memory access time, with the result that processing speed is limited primarily by the speed of main memory.

→ A technique used to compensate for the mismatch in operating speeds is to employ extremely fast, small cache between the CPU and main memory, whose access time is close to processor logic clock cycle time.

→ The cache stores segments of programe currently being executed in the CPU and temporary data frequently needed in the present calculations.

Scanned by CamScanner

→ While the I/O processor manages data transfer b/w auxiliary memory and main memory whereas the cache is concerned with the transfer of info. b/w main memory and CPU.

→ The Reason for two/three levels of memory hierarchy is storage capacity of memory increases, the cost per bit for storing binary information decreases and the access time of the memory becomes longer.

→ Auxiliary Memory has large storage capacity is relatively inexpensive, but has low access speed compared to main memory.

→ Cache memory is very small, relatively expensive and has very high access speed.

→ The overall goal of using a memory hierarchy is to obtain highest possible average access speed while minimizing the total cost of entire memory system.

→ Auxiliary and cache memories are used for different purposes.

→ The cache holds those parts of the program and data that are most heavily used, while auxiliary memory holds those parts that are not presently used by CPU.

Scanned by CamScanner

→ CPU has direct access to main memory and cache but not to auxiliary memory.

→ Access time ratio between cache and main memory is about 1 to 7.

Eg :- Cache memory access time = 100 ns

Main memory access time = 700 ns

Auxiliary memory access time = 1000 times of main memory

→ Many operating systems are designed to enable the CPU to process a number of independent programs concurrently. This concept is called 'multiprogramming'.

→ A program with its data normally resides in auxiliary memory. When the program or a segment of the program is to be executed, it is transferred to main memory to be executed by the CPU.

→ The part of the computer system that supervises the flow of information between auxiliary memory and main memory is called the 'memory management system'.

Scanned by CamScanner

Main Memory :-

→ The main memory is the central storage unit in a computer system.

→ It is relatively large and fast memory used to store programs and data during the computer operation.

→ Integrated chips circuit RAM chips are available in two possible operating modes

(1) static RAM

(2) Dynamic RAM

Static RAM :-

→ The static RAM consists essentially of internal flip flops that store the binary information.

→ The stored information remains valid as long as power is applied to the unit.

→ It is easier to use and has shorter read and write cycles.

Dynamic RAM :-

→ The dynamic RAM stores the binary information in the form of electric charges that are applied to capacitors.

→ The capacitors are provided inside the chip by MOS transistors.

→ The dynamic RAM offers reduced power consumption and larger storage capacity in a single memory chip.

Scanned by CamScanner

→ The major applications of static RAM and Dynamic RAM are.

① SRAM is used in implementing the cache memories.

② DRAM is used for implementing the main memory.

ROM (Read Only Memory):-

→ RAM refers to random access memory but it is used to designate a read/write memory to distinguish it from a 'read-only memory', although ROM is also random access.

→ ROM is used for storing programs that are permanently resident in the Computer and for tables of constants that do not change in value once the production of computer is completed.

→ ROM portion of main memory is needed for storing an initial program called a 'bootstrap loader'.

→ 'Bootstrap loader function is to start the Computer software operating when power is turned on.

→ Since RAM is volatile, its contents are destroyed when power is turned off.

→ The contents of ROM remain unchanged after

Scanned by CamScanner

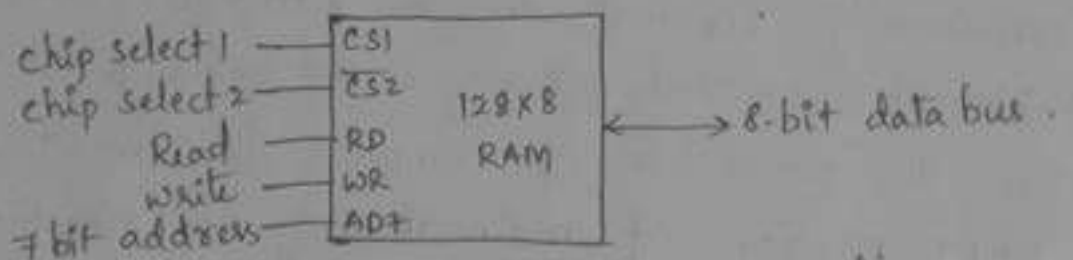
power is turned off and on again.

→ The startup of a computer consists of turning the power on and starting the execution of an initial program.

→ when power is turned on, the hardware of the computer sets the program counter (PC) to the first address of the bootstrap loader.

→ The bootstrap program loads a portion of the operating system from disk to main memory and control is then transferred to the operating system.

RAM and ROM chips:-



(a) Block diagram of RAM chip.

CS1	$\overline{CS2}$	RD	WR	Memory function	State of data bus
0	0	x	x	Inhibit	High-Impedance
0	1	x	x	Inhibit	High impedance
1	0	0	0	Inhibit	High impedance
1	0	0	1	Write	Input data to RAM
1	0	1	x	Read	output data from RAM
1	1	x	x	Inhibit	High impedance

(b) function table of RAM.

Scanned by CamScanner

→ RAM chip is better suited for communication with the CPU if it has one or more control inputs to select the chip when needed.

→ Another common feature is bidirectional data bus that allows transfer of data either from memory to CPU for read operation and or from CPU to memory during write operation.

→ A bidirectional bus is constructed using three-state buffer where its o/p is placed in one of three possible states:

1. a signal equivalent to logic 1
2. a signal equivalent to logic 0
3. high-impedance state. (It is like open circuit)

The block diagram of a RAM memory capacity is 128 words of eight bits (one byte) per word. It requires 7 bit address and an 8 bit bidirectional data bus. The read and write i/p's specify the memory operation and two chip select (CS) control inputs are for enabling the chip only when it is selected by the microprocessor.

The function table specifies the operation of the RAM chip. The unit is in operation only when $CS1 = 1$ and $\overline{CS2} = 0$. If the chip select inputs are not enabled or enabled and if read or write inputs are not enabled, the memory is

Scanned by CamScanner

inhibited and data bus is in high-impedance state. When $CS1 = 1$ and $\overline{CS2} = 0$, the memory can be placed in a write or read mode. When the WR input is enabled, the memory stores a byte from the data bus into a location specified by the address input lines. When the RD input is enabled the content of the selected byte is placed on the data bus.

ROM chip :-

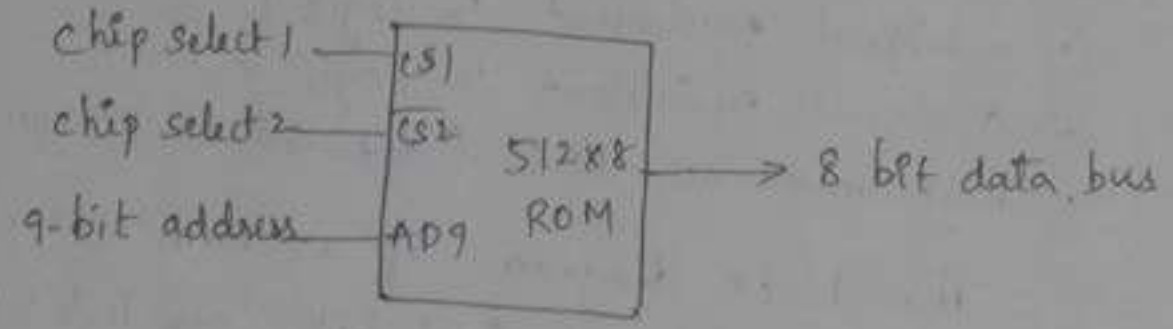


fig:- Typical ROM chip.

For the same-size chip, it is possible to have more bits of ROM than of RAM, because the internal binary cells in ROM occupy less space than in RAM i.e, 512-byte ROM, while RAM has only 128 bytes.

The nine address lines in the ROM chip specify any one of the 512 bytes stored in it. The two chip select inputs must be $CS1 = 1$ and $\overline{CS2} = 0$ for the unit to ~~separate~~ operate.

Scanned by CamScanner

otherwise, the data bus is in a high-impedance state. There is no need of read or control control because the unit can only read. Thus when the chip is enabled by the two select inputs, the byte selected, by the address lines appear on the data bus.

Memory Address Map :-

→ The designer of a computer system must calculate the amount of memory required for the particular application and assign it to either RAM or ROM.

→ The interconnection b/w memory and processor is then established from knowledge of the size of memory needed and the type of RAM and ROM

→ The addressing of memory can be established by means of a table that specifies the memory address assigned to each chip.

→ The table called 'memory address map' is a pictorial representation of assigned address space for each chip in the system.

Eg :- RAM = 512 bytes
ROM = 512 bytes

The memory address map for this configuration is shown in the table below.

Scanned by CamScanner

Component	Hexadecimal address	Address bus									
		10	9	8	7	6	5	4	3	2	1
RAM1	0000 - 007F	0	0	0	x	x	x	x	x	x	x
RAM2	0080 - 00FF	0	0	1	x	x	x	x	x	x	x
RAM3	0100 - 017F	0	1	0	x	x	x	x	x	x	x
RAM4	0180 - 01FF	0	1	1	x	x	x	x	x	x	x
ROM	0200 - 03FF	1	x	x	x	x	x	x	x	x	x

fig :- Memory Address Map for Microprocessor

The hexadecimal address column assigns a range of hexadecimal equivalent address for each chip. There are 16 lines in address bus, the table shows only 10 lines because the other 6 are not used in this Example ~~an~~ because they are assumed to be zero.

→ The 'x's represents that address bus lines are connected to the address inputs in each chip.

→ The RAM chips have 128 bytes and need seven address lines ($\because 128 = 2^7$)

ROM - 512 bytes ($\because 512 = 2^9$) = 9 address lines.

→ The x's are assigned to lower order bus lines (1 through 7 for RAM and 1-9 for ROM).

→ To distinguish 4 RAM chips by assigning to each a different address.

Scanned by CamScanner

→ We choose bus lines 8 and 9 to represent four distinct binary combinations.

→ The distinction b/w RAM and ROM address is done with another bus line.

→ when line 10 is 0, the CPU selects a RAM and when line is equal to 1 it selects ROM.

Memory connection to CPU :-

→ RAM and ROM are connected to CPU by data and address buses.

→ The low-order lines in the address bus select the byte within the chips and other lines in the address bus select a particular chip through CS

The Configuration gives a memory capacity of 512 bytes of RAM and 512 bytes of ROM. It implements the memory map of table shown in previous topic. Each RAM receives the seven low-order bits of the address bus to select 128 possible bytes. The particular RAM chip is selected by using address lines 8 and 9 in the address bus.

9 bit	8 bit	
0	0	– RAM1
0	1	– RAM2
1	0	– RAM3
1	1	– RAM4.

Scanned by CamScanner

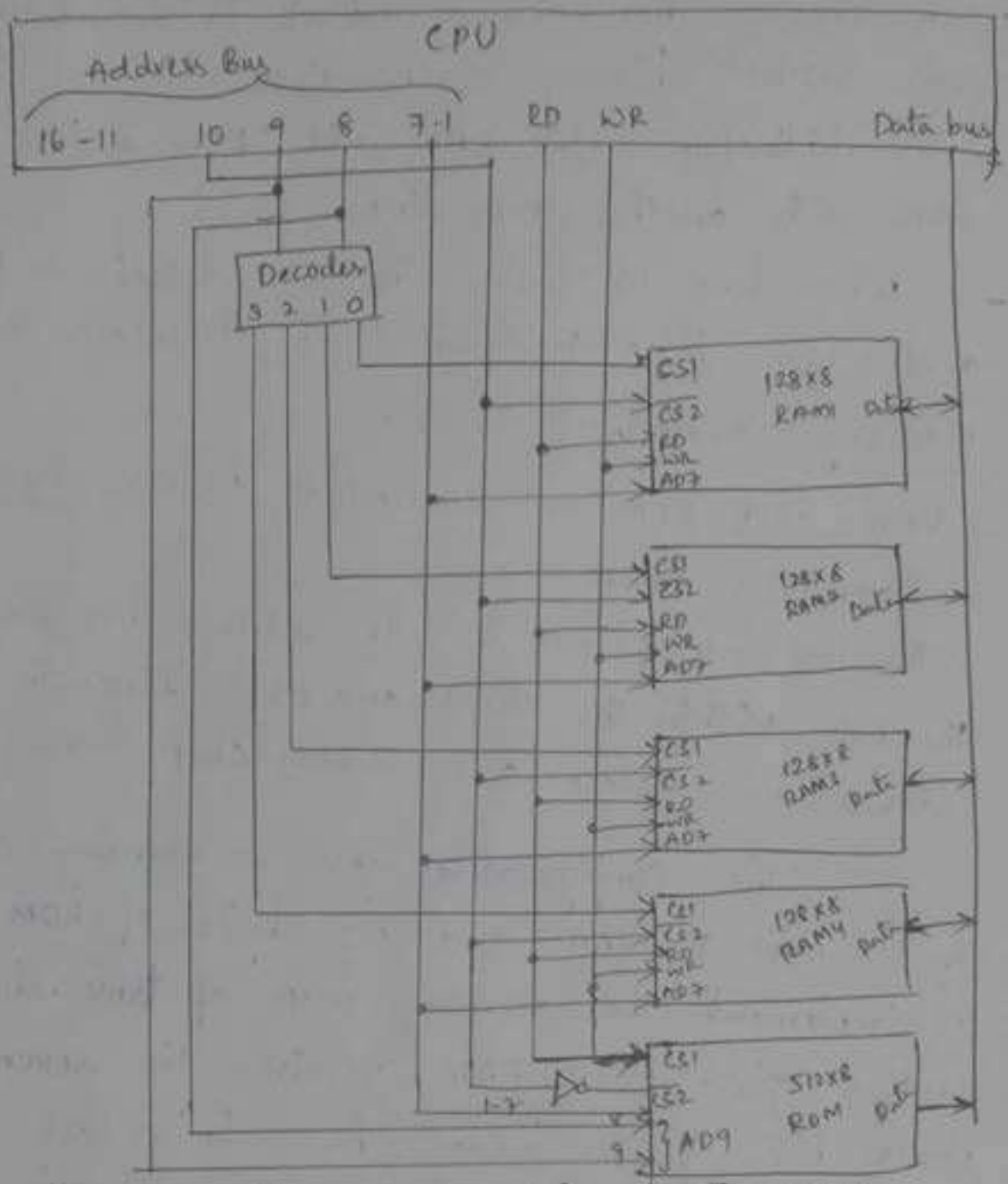


fig 12.4: Memory Connection to the CPU

→ RAM and ROM chips are selected by using the bus line 10. (RAM - 10th bit is 1, ROM - 10th bit is 0).

→ The other chip select i/p in the ROM is connected to RD control line of the ROM chip for read operation.

→ Address lines 1 to 9 are applied to i/p of ROM directly

→ This assigns addresses 0 to 511 to RAM and 512 to 1022 to ROM.

Scanned by CamScanner

Auxiliary Memory

→ The most common auxiliary memory devices used in computer systems are magnetic disks and tapes

→ The important characteristics of any device are its access mode, access time, transfer rate, capacity and cost.

Access time :-

The average time required to reach a storage location in memory and obtain its content is called the 'access time'.

→ In electromechanical devices with moving parts such as disks and tapes, the access time consists of a 'seek' time required to position the read-write head to a location and a 'transfer' time required to transfer data to or from the device.

→ Auxiliary storage is organized in records or blocks.

→ A record is a specified number of characters or words.

Transfer Rate :- The transfer rate is the number of characters or words that the device can transfer per second, after it has been positioned at the beginning of the record.

→ Magnetic drums and disks are quite similar in operation, both consist of high-speed rotating surfaces coated with a magnetic recording mechanism.

Scanned by CamScanner

→ Bits are recorded as magnetic spots on the surface as it passes a stationary mechanism called a 'write head'.

→ Stored bits are detected by a change in magnetic field produced by a recorded spot on the surface as it passes through a 'read head'.

Magnetic Disks:-

→ A magnetic disk is a circular plate constructed of metal or plastic coated with magnetized material.

→ Several disks may be stacked on one spindle with read/write heads available on each surface.

→ All disks rotate together at high speed and are not stopped or slated for access purpose.

→ Bits are stored in the magnetized surface in spots along concentric circles called 'tracks'.

→ The tracks are divided into sections called 'sectors'.

→ The track address bits are used by a mechanical assembly to move the head into the specified track position before reading or writing.

→ Permanent timing tracks are used in disks to synchronize the bits and recognize the sectors.

→ The disk system is addressed by address bits that specify the disk number, the disk surface, the sector number and the track within the sector.

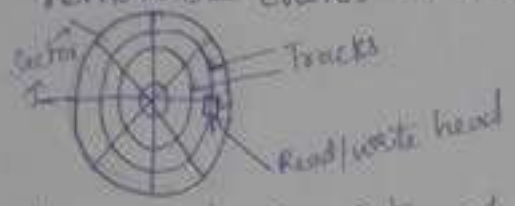
Scanned by CamScanner

→ After the read/write heads are positioned in the specified ~~sector~~ track, ~~under the read/write head~~ the system has to wait until the rotating disk reaches the specified sector under the read/write head.

→ Information transfer is very fast once the beginning of a sector has been reached.

→ Disks that are permanently attached to the unit assembly and cannot be removed by occasional user are called 'hard disks'.

→ A disk drive with removable disks is called a 'floppy disk'.



Magnetic Tape :-

→ A magnetic tape transport consists of the electrical, mechanical and electronic components to provide the parts and control mechanism for a magnetic tape unit.

→ The tape itself is a strip of plastic coated with a magnetic recording medium

→ Bits are recorded as magnetic spots on the tape along several tracks.

→ Seven or nine bits are recorded simultaneously to form a character.

→ Magnetic tape units can be stopped, started to move forward or in reverse or can be rewound.

→ However, they cannot be started or stopped fast enough b/w individual characters.

→ Information is recorded in blocks referred to as 'records'.

Scanned by CamScanner

→ Gaps of unrecorded tape are inserted between records where the tape can be stopped.
→ The tape starts moving while in a gap and attains its constant speed by the time it reaches the next record.
→ Each record on tape has an identification bit pattern at the beginning and end.
→ By reading the bit pattern at beginning, the tape control identifies the record number.
→ By reading the bit pattern at the end of the record, the control recognizes the beginning of a gap.
→ A tape unit is addressed by specifying the record number and the number of characters in the record. Records may be fixed or variable length.

Associative Memory :-

→ A memory unit accessed by content is called 'associate memory' or 'content addressable memory' (CAM).
→ Many data processing applications require the search of items in a table stored in memory.
→ An assembler program searches the symbol address table inorder to extract the binary equivalent of symbol.
→ The search procedure is a strategy for choosing a sequence of addresses, reading the content of memory at each address, and comparing

Scanned by CamScanner

The information read with the item being searched until a match occure.

→ The number of accesses to memory depends on the location of the item and the efficiency of the search algorithm.

→ Many search algorithms have been developed to minimize the number of accesses

→ The time taken to find an item can be reduced, if stored data can be identified for access by the content of the data itself rather than by an address.

→ Associate Memory is one such type of memory, stored data can be identified or accessed based on content rather than address.

→ when word is written in memory, no address is given.

→ The memory finds the empty unused location to store the word.

→ Searches can be done on an entire word or on a specific field within a word.

→ An associative memory is more expensive than RAM because each cell must have storage capability as well as logic circuits for matching its content with an external Argument.

Scanned by CamScanner

Hardware Organization

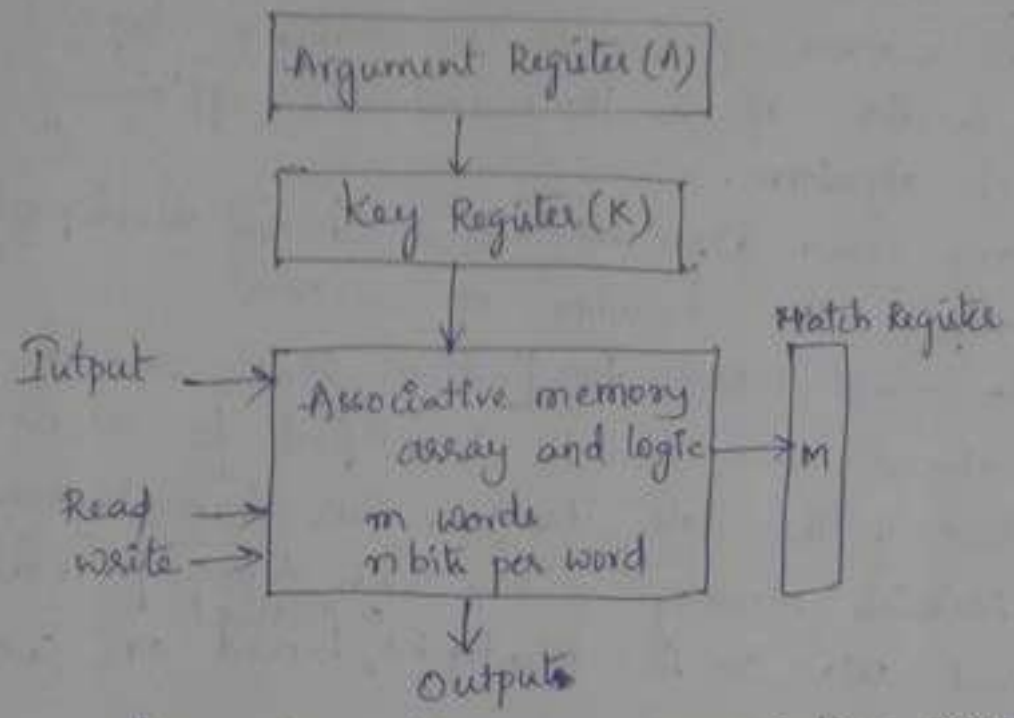


fig:- Block diagram of associative memory.

→ It consists of a memory array and logic for m words with n bits per word.

→ The argument Register A and key register K each have n bits, one for each bit of a word.

→ The Match Register M has 'm' bits, one for each memory word.

→ Each word in memory ~~match register m bits~~ is compared in parallel with the content of the argument register.

→ The words that match the bits of the argument register set a corresponding bit in the match register.

→ After matching process, bits in match register that

Scanned by CamScanner

have been set indicate the fact that their corresponding words have been matched.

→ Reading is accomplished by sequential access to memory for those words whose corresponding bits in the match register is set.

→ The key register provides a mask for choosing a particular field or key in the argument word.

→ The entire argument is compared with each memory word if the key register contains all 1's.

→ otherwise, only those bits in the argument that have 1's in their corresponding position of the key register are compared.

→ Thus the key register provides a mask or identifying piece of information which specifies how the reference to memory is made.

Eg:-

A	101	111100
K	111	000000
word 1	100	111100, no match
word 2	101	000001, match.

only the three leftmost bits of A are compared with memory words because K has 1's in these positions.

word 2 matches the unmasked argument field because the three leftmost bits of the argument and the word are equal.

Scanned by CamScanner

Relation between memory array and external registers in an associative memory :-

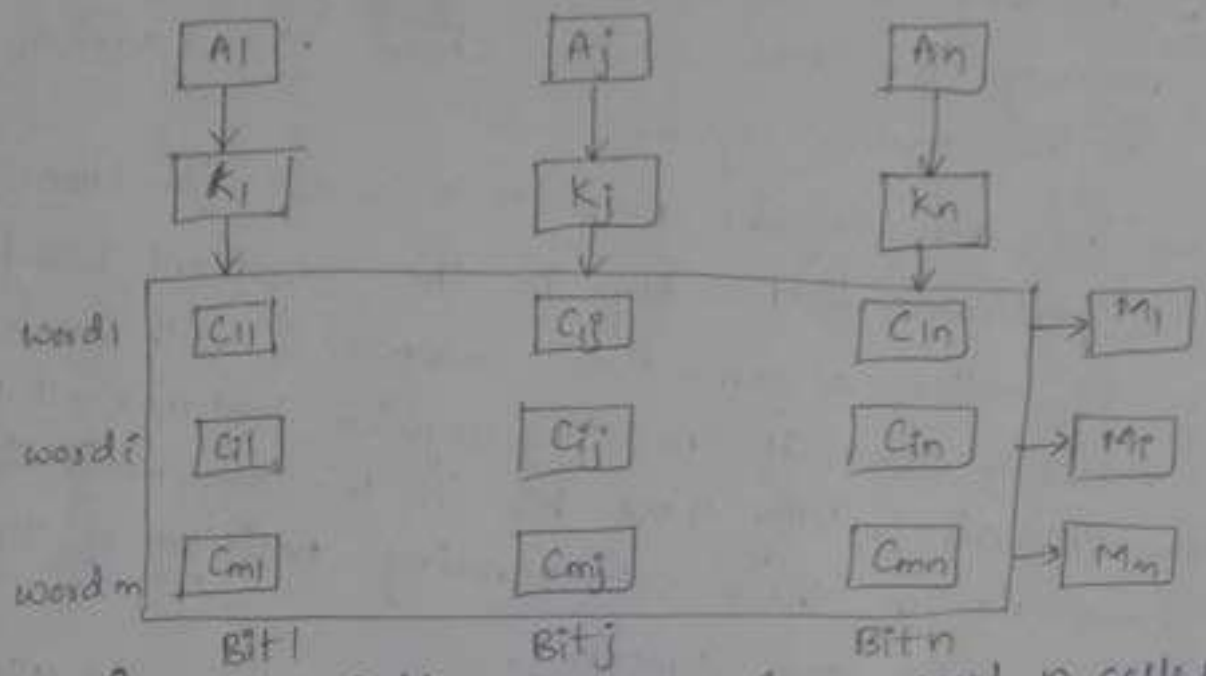


fig :- Associative memory of m word, n cells per word.

→ The cells in the array are marked by the letter c with two subscripts.

→ The first subscript defines word number and second specifies bit position in the word.

→ Cell C_{ij} for bit j in word i.

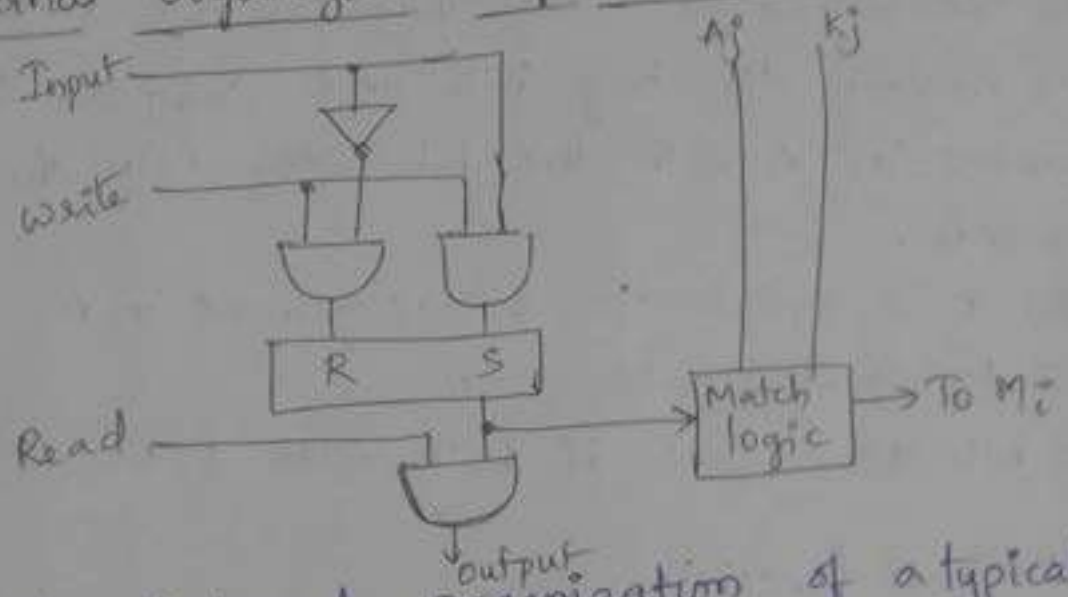
→ A bit A_j in the argument register is compared with all the bits in column j of the array provided that $K_j = 1$

→ If match occurs between all the unmasked bits of the argument and the bits in the word i, the corresponding bit M_i in the match register is set to 1.

Scanned by CamScanner

→ If one or more unmasked bits of the argument and the word do not match M_i is cleared to 0.

Internal organization of a cell in associative memory



The internal organization of a typical cell C_{ij} consists of flip-flop storage elements F_{ij} and the circuits for reading, writing, and matching the cell.

→ The input bit is transferred into the storage cell during a write operation.

→ The bit stored is read out during a read operation.

→ The match logic compares the content of the storage cell with the corresponding unmasked bit of the argument and provides an output for the decision logic that sets the bit in M_i.

Scanned by CamScanner

Match logic :-

→ The match logic for each word can be derived from the comparison algorithm for two binary numbers

→ we neglect the key bits and compare the argument in A with the bits stored in the cells of the words.

→ word i is equal to the argument in A if $A_j = f_{ij}$ for $j = 1, 2, \ldots, n$

→ Two bits are equal if both are '1' or '0'

Boolean function

$$\boxed{x_j = A_j F_{ij} + A_j' f_{ij}'}$$

$x_j = 1$, if the pair of bits in position 'j' are equal

otherwise $x_j = 0$

for a word i to be equal to argument in A, we must have all x_j variables equal to '1'. This is the condition to set match bit, M_i to 1.

Boolean function:

$$\boxed{M_i = x_1 x_2 x_3 \ldots x_n}$$

and constitutes the AND operation of all pairs of matched bits in a word.

Scanned by CamScanner

→ We use key bit k_j in the comparison logic.

If $k_j = 0$, the corresponding bits of A_j and f_{ij} need no comparison.

If $k_j = 1$, they must be compared.

This can be achieved by ORing each term with k_j' thus:

$$x_j + k_j' = \begin{cases} x_j & \text{if } k_j = 1 \\ 1 & \text{if } k_j = 0. \end{cases}$$

when $k_j = 1$,

$$k_j' = 0 \text{ and } x_j + 0 = x_j$$

when $k_j = 0$, then $k_j' = 1$

$$x_j + 1 = 1$$

→ A term $(x_j + k_j')$ will be in the 1-state if its pair of bits is not compared.

→ This is necessary because each term is ANDed will all other terms so that an output of 1 will have no effect, only when $k_j = 1$

⇒ Match logic for word i in associative memory can be expressed in boolean function is;

$$M_i = (x_1 + k_1')(x_2 + k_2')(x_3 + k_3') \ldots (x_n + k_n')$$

Each term is equal to 1 if its corresponding $k_j = 0$

if $k_j = 1$, the term will be either 0 or 1 depending on the value of x_j

Scanned by CamScanner

Boolean function can also be expressed as:

$$M_i = \prod_{j=1}^{n} \left(A_j F_{ij} + A_j' F_{ij}' + K_j \right)$$

Match logic for one word circuit:-

→ Each cell requires two AND gates and one OR gate.

→ The inverters for A_j and K_j are needed once for each column are used for all bits in the column.

→ The o/p of all OR gates in the cells of the same word go to the input of a common AND gate to generate the match signal for M_i

→ M_i = logic 1 if match occurs or else 0 if no match occurs.

→ Note that if the key register contains all 0's o/p of M_i will be a 1 irrespective of the value of A or word.

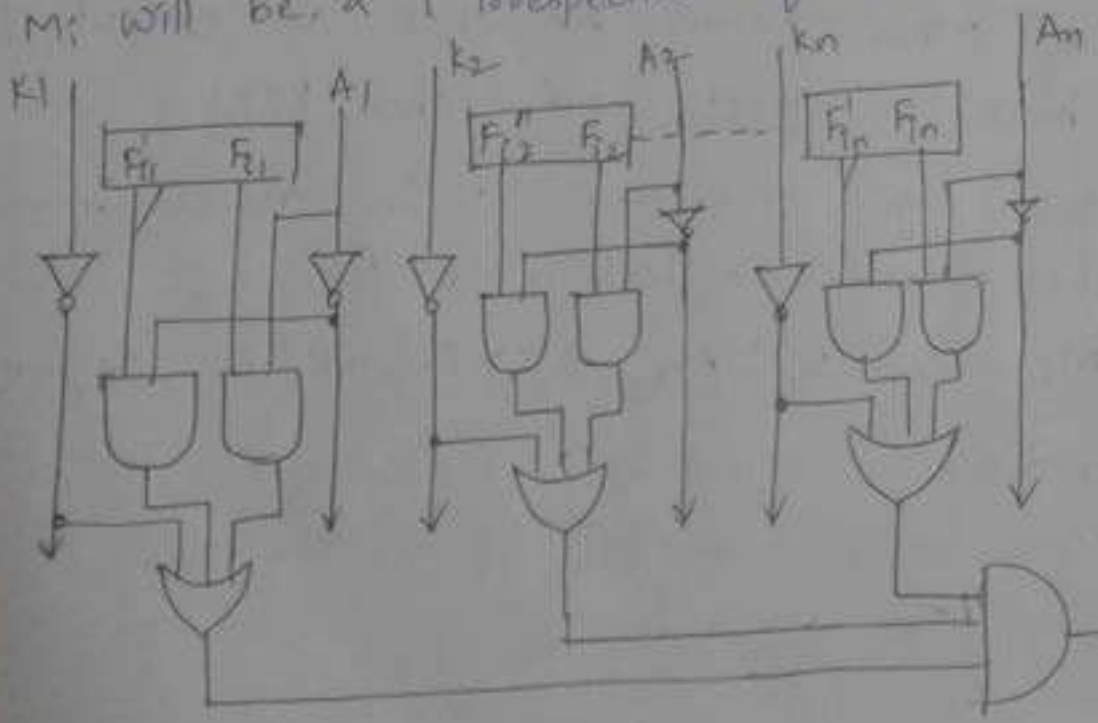


fig:- Match logic for one word of associative memory

Read Operation:-

→ If more than one word in memory matches the unmasked argument field, all the matched words will have 1's in the corresponding bit position of the match register.

→ It is then necessary to scan the bits of the match register one at a time.

→ The matched words are read in sequence by applying a read signal to each word line whose corresponding M_i bit is a 1.

→ The associative memory stores a table with no two identical items under a given key.

→ In this case, only one word may match the unmasked argument field.

→ By connecting o/p M_i directly to the read line in the same word position, the content of the matched word will be presented automatically at the output lines and no special read command signal is needed.

Write Operation:-

→ An associative memory must have a write capability for storing the information to be searched.

→ If the entire memory is loaded with new information at once prior to a search operation then the writing can be done by addressing each location in sequence.

Scanned by CamScanner

→ This will make the device a RAM for writing and a content addressable memory for reading.

→ The advantage here is that the address for i/p can be decoded as in RAM.

→ Thus, instead of having m address lines, one for each word in the memory, the number of address lines can be reduced by the decoder to 'd' lines, where $\boxed{m = 2^d}$.

→ If unwanted words have to be deleted and new words inserted one at a time, there is need for a special register to distinguish b/w active and inactive words.

→ This register, sometimes called a 'tag register' would have as many bits as there are words in the memory.

→ for every active word stored in memory, the corresponding bit in the tag register is set to 1.

→ A word is deleted from memory by clearing its tag bit to '0'.

→ words are stored in memory by scanning the tag register until the first 0 bit is encountered.

→ This gives the first available inactive word and a position for writing a new word.

Scanned by CamScanner

Cache Memory:-

Locality of Reference:-

Analyses of a large number of typical programs has shown that the references to memory at any given interval of time tend to be confined within a few localized areas in memory. This is called as 'locality of reference'.

→ A typical computer program flows in a straight line fashion with program loops and subroutine calls encountered frequently.

→ When a program loop is executed, CPU refers to set of instructions repeatedly forms loops.

→ Every time a given subroutine is called, its set of instructions are fetched from memory.

→ Thus loops, subroutines tend to localize the references to memory for fetching instructions.

→ Table-lookup procedures repeatedly refer to that portion in memory where the table is stored.

→ Iterative procedures refer to common memory locations confined within a local portion of memory.

The result of all these observations is the locality of reference property, which states that over a short interval of time, the addresses generated by a typical program refer to few localized areas of memory repeatedly.

Scanned by CamScanner

Cache memory :-

→ If the active portions of program and data are placed in a fast small memory, the average memory access time can be reduced, thus reducing the total execution time of the program.

→ Such a fast ~~ex~~ small memory is referred to as 'cache memory'.

→ It is placed b/w CPU and main memory.

→ The cache access time is less than the access time of memory by a factor of 5 or 10.

→ cache is the fastest component in the memory hierarchy and approaches the speed of CPU.

→ The fundamental idea of cache organization is that by keeping the most frequently accessed instructions and data in the fast cache memory, the average memory access time will approach the access time of the cache.

→ large fraction of memory requests will be found in the fast cache memory because of the locality of reference property of programs.

Basic operations of cache :-

→ when the CPU needs to access memory, the cache is examined, if the word is found in the cache, it is read from fast memory.

Scanned by CamScanner

→ If the word addressed by the CPU is not found in the cache, the main memory is accessed to read the word

→ A block of words containing the one just accessed is then placed in cache memory from main memory.

→ A block size may vary from 1 word to about 16 words

→ Some data are transferred to cache so that future references to memory find the required words in the fast cache memory.

Performance of cache memory:-

→ Performance of cache memory is frequently measured in terms of a quantity called 'hit ratio'

→ When the CPU refers to memory and finds the word in cache it is said to be 'hit'

→ If the word is not found in cache, it is in main memory it counts as a 'miss'

→ The ratio of number of hits divided by the total CPU references to memory is the 'hit ratio'.

→ The hit ratio is best measured experimentally by running programs in the computer and measuring the no. of hits and misses during a given interval of time.

→ Hit ratio's of 0.9 and higher have been reported.

Scanned by CamScanner

→ The average memory access time of a computer system can be improved by use of cache.

→ If the hit ratio is high enough so that most of the time the CPU accesses the cache instead of main memory, the average access time is closer to the access time of fast cache memory.

→ The Basic characteristic of cache memory is its fast access time. Therefore very little or no time must be wasted when searching for words in the cache.

→ The transformation of data from main memory to cache memory is referred to as a 'mapping' process. There are 3 types of mapping.

1. Associative mapping
2. Direct Mapping
3. Set-Associative mapping

fig :- Example of cache memory

The main memory can store 32K words of 12 bits each. The cache is capable of storing 512 of these words at any given time. For every word stored in

Scanned by CamScanner

in cache, there is a duplicate copy in 'main memory'.

→ The CPU communicates with both memories.

→ It first sends a 15-bit address to cache.

→ If there is a hit, the CPU accepts 12-bit data from cache.

→ If there is a miss, the CPU reads the word from main memory and the word is then transferred to cache.

Mapping Techniques :-

Associative Mapping :-

→ The fastest and most flexible cache organization uses an associative memory

→ The associative memory stores both the address and content (data) of the memory word.

→ This permits any location in cache to store any word from main memory.

CPU address (15 bits)

Argument Register

Address	Data
01000	3450
02777	6710
22345	1234

fig:- Associative Mapping cache (all numbers in octal)

Scanned by CamScanner

→ The address value of 15 bits is represented in 5 digit octal number and 12 bit word is represented in 4 digit octal number.

→ A CPU address of 15-bits is placed in the argument register and the associative memory is searched for a matching address.

→ If the address is found, the 12-bit data is read and sent to the CPU.

→ If no match occurs, the main memory is accessed for the word.

→ The address - data pair is then transferred to the associative cache memory.

→ If the cache is full, an address - data pair is displaced to make room for a pair.

→ The decision to what pair has to be replaced is determined by 'replacement Algorithm'.

→ A simple procedure is to replace cells of the cache in round-robin order whenever a new word is requested from main memory.

Direct Mapping:-

→ Associative memories are expensive compared to RAM because of added logic associated with each cell.

→ In this method of mapping, the CPU address of 15 bits is divided into two fields.

→ The nine LSB constitute 'index' field' and

Scanned by CamScanner

remaining six bits form the tag field.

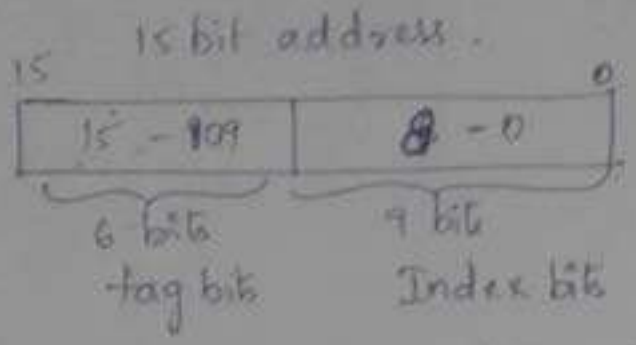
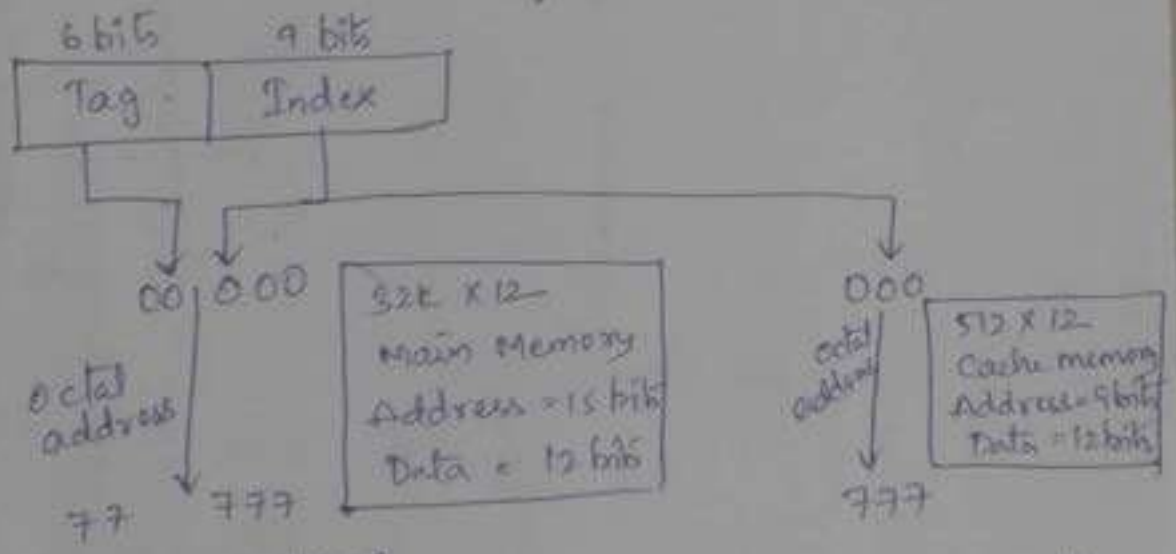


fig :- Addressing relationships b/w main and cache memories



→ The number of bits of index field is equal to the number of address bits required to access the cache memory.

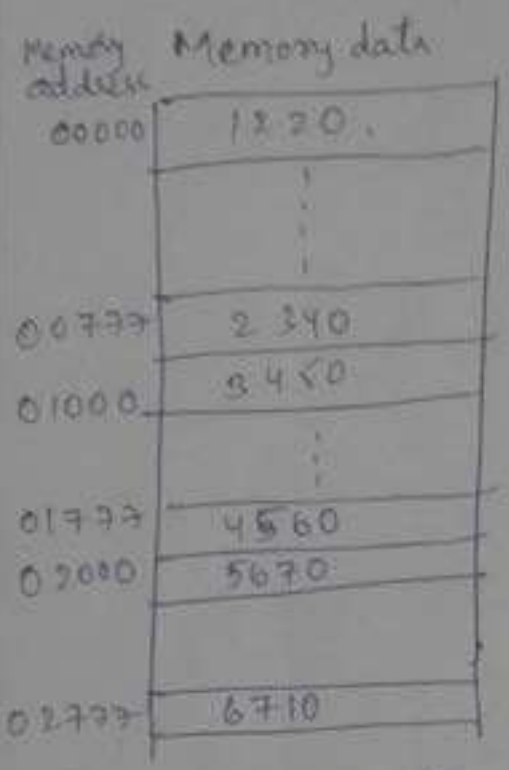
→ In general there are 2^k words in cache and 2^n words in memory.

→ The n-bit memory address is divided into 2 fields: K bits for index field and n-k bits for tag field.

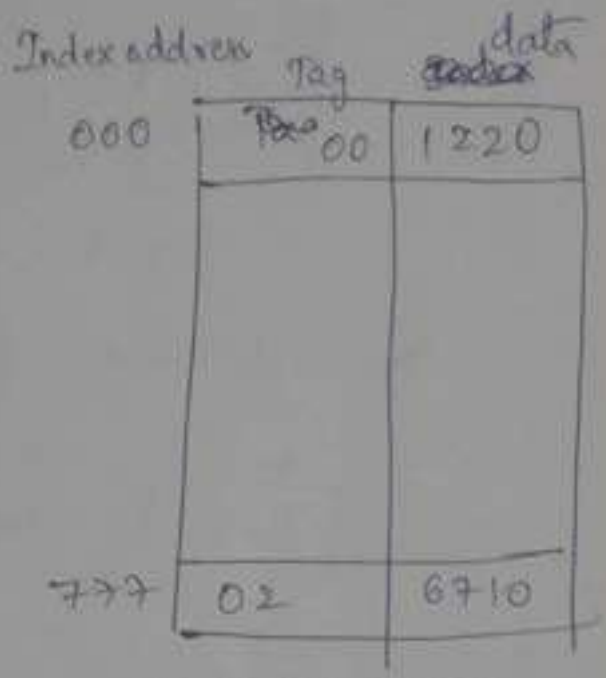
→ The direct mapping cache organization uses the n-bit address to access the main memory and the

k-bit index to access the cache.

Internal organization of the words in cache :-



(a) Main memory



b) Cache memory

fig :- Direct mapping cache organization.

→ Each word in cache consists of the data word and its associated tag.

→ When a new word is first brought into cache the tag bits are stored alongside the data bits.

→ When the CPU generates a memory request, the index field is used for the address to access the cache.

→ The tag field of the CPU address is compared with the tag in the word read from the cache

→ If the two tags match, there is a hit and desired data word is in cache.

Scanned by CamScanner

→ If there is no match b/w the tags, then there is a miss the required word is read from main memory.

Disadvantage of Direct Mapping :-

→ The hit ratio can drop considerably if two or more words whose addresses have the same index but different tags are accessed repeatedly.

Example of Direct Mapping :-

	Index	Tag	Data
	000	01	3450
Block 0	007	01	6578
	010		
Block 1	017		
	020		
	027		
	⋮	⋮	⋮
	770	02	
Block 63	777	02	6710

6	6	3
Tag	Block	word

Index.

fig:- Direct mapping cache with block size of 8 words.

→ The word at address 0 is presently stored in the cache (index = 000, tag = 00, data = 1220).

Suppose CPU wants to access the word at address 02000.

→ The index address is 000, so it is used to access the cache.

→ The two tags are then compared. The cache tag is 00 but the address tag is 02, which does not produce

Scanned by CamScanner

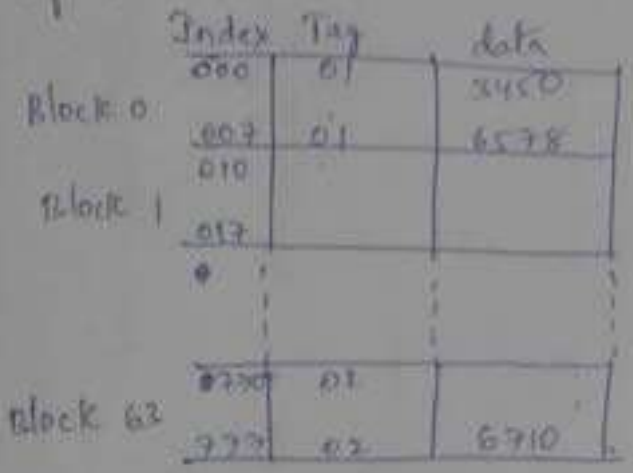
a match.

→ Therefore the main memory is accessed and the data word 5670 is transferred to CPU.

→ The cache word at index address 000 is then replaced with tag of 02 and data of 5670.

direct mapping using blocks :-

→ The same organization but using a block size of 8 words.



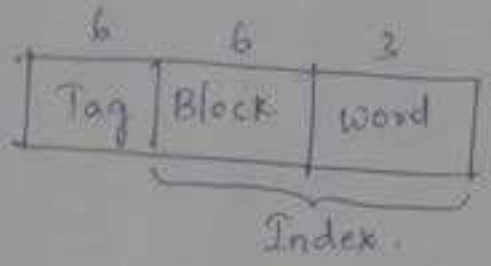


fig :- Direct mapping cache with block size of 8 words.

→ The index field is divided into two parts; The block field and the word field.

→ In a 512-word cache there are 64 blocks of 8 words each ($\because 64 \times 8 = 512$).

→ The block number is specified with a 6-bit field and the word within the block is specified with a 3-bit field.

→ The tag field stored within the cache is common to all eight words of the same block.

Scanned by CamScanner

→ Every time a miss occurs, so entire block of 8 words must be transferred from main memory to cache memory.

Set-Associative Mapping :-

→ It is an improvement over the direct mapping organization in that each word of cache can store two or more words of memory under the same index address.

→ Each data word is stored together with its tag and the number of tag-data items in one word of cache form a 'set'.

Index	Tag	Data	Tag	Data
000	01	3450	02	5670
777	02	6710	00	2340

fig: Two-way set associative mapping cache.

→ Each index address refers to two data words and their associated Tags. Each tag requires six bits and each data word has 12 bits so word length is $2(6+12) = 36$ bits.

→ An index address of 9 bits can accommodate 512 words.

→ Thus the size of cache memory is 512×36.

Scanned by CamScanner

→ It can accomodate 1024 words of main memory since each word of cache contains 2 data words.

→ In general, a set-associative cache of set size K will accomodate K words of main memory in each word of cache.

→ The words stored at addresses 01000 and 02000 of main memory are stored in cache memory at index address 000.

→ Simillarly, word at addresses 02777 and 00777 are stored in cache at index address 777.

→ when the CPU generates a memory request, the index value of the address is used to access the cache.

→ The tag field of the CPU address is then compared with both tags in cache to determine if a match occurs.

→ The comparison logic is done by an associative search of the tags in the set similar to an associative memory search; thus the name "set-associative"

→ The hit ratio will improve as the set size increases because more words with the same index but different tags can reside in cache.

→ It requires more complex comparison logic because there is an increase in the set size increases the no. of bits in words of cache.

Scanned by CamScanner

Replacement Algorithms:-

→ when a miss occurs in a set-associative cache and the set is full, it is necessary to replace one of the tag-data items with the new value.

→ The Most Common Replacement Algorithms used are;

1. Random Replacement
2. First in, first out (FIFO)
3. Least Recently Used (LRU)

→ with the Random Replacement policy the Control chooses one tag-data item for replacement at random.

→ The FIFO procedure selects for replacement the item that has been in the set the longest.

→ The ~~FIF~~ LRU Algorithm selects for replacement the item that has been least recently used by the CPU.

Both FIFO and LRU can be implemented by adding a few extra bits in each word of cach

Scanned by CamScanner

UNIT-V

Reduced Instruction Set Computer :-

Introduction :-

→ The instruction set chosen for a particular computer determines the way that machine language programs are constructed.

→ Early computers had small and simple instruction sets, forced by the need to minimize the hardware used to implement them.

→ Digital computer became cheaper with the advent of integrated circuits, computer instructions tended to increase both in number and complexity.

→ Many computers has more than 100 or sometimes even more than 200 instructions.

→ These computers employ a variety of data types and a large number of addressing modes.

→ A computer with a large number of instructions is classified as 'complex instruction set computer' (CISC)

→ In early 1980's, a number of computer designers recommended that computers use fewer instructions with simple constructs so they can be executed much faster within the CPU without having to use memory.

Scanned by CamScanner

UNIT-V

Reduced Instruction Set Computer :-

Introduction :-

→ The instruction set chosen for a particular computer determines the way that machine language programs are constructed.

→ Early computers had small and simple instruction sets, forced by the need to minimize the hardware used to implement them.

→ Digital computer became cheaper with the advent of integrated circuits, computer instructions tended to increase both in number and complexity.

→ Many computers has more than 100 or sometimes even more than 200 instructions.

→ These computers employ a variety of data types and a large number of addressing modes.

→ A computer with a large number of instructions is classified as 'complex instruction set computer' (CISC)

→ In early 1980's, a number of computer designers recommended that computers use fewer instructions with simple constructs so they can be executed much faster within the CPU without having to use memory.

Scanned by CamScanner

→ This type of computer having fewer instructions is called as 'reduced instruction set computer' or RISC.

CISC characteristics :-

→ The design of an instruction set for a computer must take into consideration not only the machine language constructs but also the requirements imposed on the use of 'high-level programming languages'.

→ The translation from high-level to machine language programs is done by means of 'compiler programs'.

→ One reason to choose CISC is to desire to simplify the compilation and improve the overall computer performance.

→ The task of the compile is to generate machine instruction for each high-level language statement.

Eg:- CISC architectures

Digital Equipment Corporation VAX computer

IBM 370 computer.

→ Another characteristic of CISC architecture is the incorporation of variable-length instruction formats.

→ Instructions that require register operands may be only 2 bytes in length but instruction that need two memory addresses may need 5 bytes to include the entire instruction code.

Scanned by CamScanner

The instructions of CISC processor provide direct manipulation of operands in memory

Eg :- ADD instruction may specify one operand in memory through index ~~regates~~ addressing and a second operand in memory through a direct addressing. Another memory location to store the SUM

→ More instructions and addressing modes are incorporated into a computer, the more hardware logic is needed to implement and support them and this may cause the computations to slow down.

<u>Major characteristics of CISC architecture are</u> :-

1. A large number of instructions - typically from 100 - 250 instructions.
2. Some instructions that perform specialized tasks and are used infrequently.
3. A large variety of addressing modes (5 - 20 different modes).
4. Variable - length instructions formats.
5. Instructions that manipulate operands in memory

Scanned by CamScanner

RISC characteristics:

→ The concept of RISC architecture involves an attempt to reduce execution time by simplifying the instruction set of the computer.

Major characteristics of RISC processor are:-

1. Relatively few instructions.
2. Relatively few addressing modes.
3. Memory access limited to load and store instructions.
4. All operations done with the registers of the CPU.
5. Fixed-length, easily decoded instruction format
6. Single-cycle initiated instruction execution.
7. Hardwired rather than microprogrammed control.

An important aspect of RISC instruction formats is that it is easy to decode. Thus the operation code and registers fields of the instruction code can be accessed simultaneously by the control.

For faster operations a hardwired control is preferable over a microprogrammed control.

A characteristic of RISC processor is their ability to execute one instruction per clock cycle. This is done by overlapping fetch, decode and execute

Scanned by CamScanner

phases of two or three instructions by using a procedure referred to as pipelining.

A load and store instruction may require two clock cycles because access to memory takes more time than register operations.

Efficient pipelining, as well as few other characteristics are sometimes attributed to RISC

1. A relatively large number of registers in processor unit
2. Use of overlapped register windows to speed-up procedure call and return.
3. Efficient instruction pipeline.
4. Compiler support for efficient translation of high-level language programs into machine-level language programs.

Scanned by CamScanner

Pipeline and Vector Processing :-

parallel processing :-

Parallel processing is a term used to denote a large class of technique that are used to provide simultaneous data processing tasks.

A parallel processing system is able to perform concurrent data processing to achieve faster execution time.

Eg :- while an instruction is being executed in the ALU, the next instruction can be read from memory. The system may have two or more ALU's and be able to execute two or more instructions at the same time.

The purpose of parallel processing is to speed up the computer processing capability and increase its throughput. The throughput is defined as the amount of processing that can be accomplished during a given interval of time.

The amount of hardware increases with parallel processing and the cost of the system increases.

Parallel processing can be viewed from various levels of complexity. At the lowest level, we distinguish b/w parallel and serial operations by the type of registers. Shift registers operate in serial fashion one bit at a time and parallel load

Scanned by CamScanner

registers operate with the all bits of the word simultaneously.

Parallel processing at higher level can be achieved by having a multiplicity of functional units that perform identical or different operations simultaneously. Parallel processing is established by distributing the data among multiple functional units.

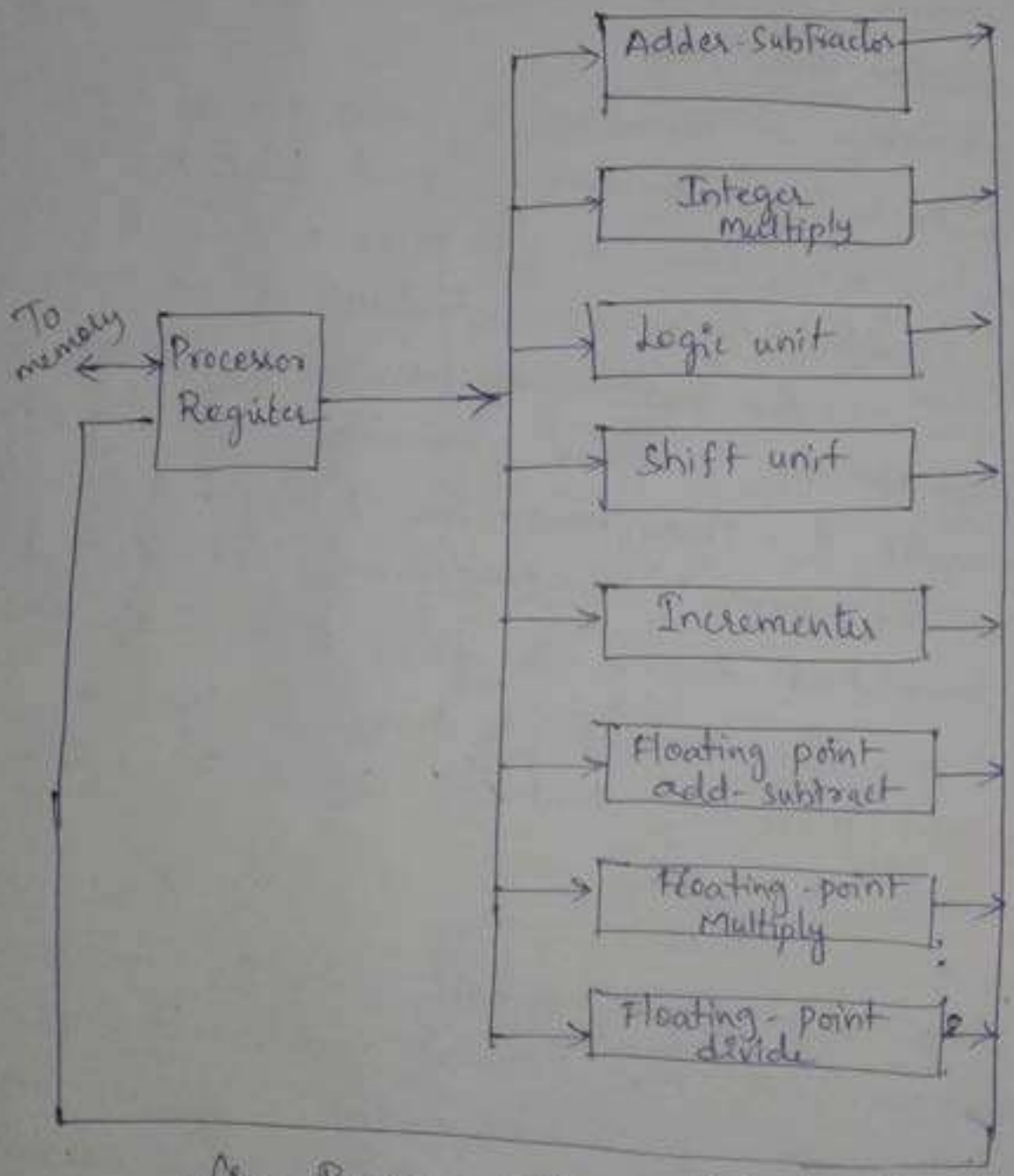


fig:- Processor with multiple functional unit.

Scanned by CamScanner

The above figure shows one possible way of separating execution unit into eight functional units operating in parallel.

The operands in the registers are applied to one of the units depending on the operation specified by the instruction association with the operands.

The adder and integer multiplier perform the arithmetic operations with integer number. The floating-point operations are separated into three circuits operating in parallel. The logic, shift & increment operations can be performed concurrently on different data. All units are independent of each other so one number can be shifted with another number being incremented. A multifunctional organization is usually associated with a complex CU to coordinate all the activities among various components.

There are a variety of ways that parallel processing can be classified. It can be considered as internal organization of the processors, from the interconnection structure b/w processors or from the flow of information through the system.

Scanned by CamScanner

One classification which introduced by M.J. flynn Consider the organization of a computer system by the no. of instructions and data items that are manipulated.

The sequence of instruction read from memory constitutes an instruction stream. The operations performs the data in the processor constitutes a data stream.

Flynn's classification divides computers into four major groups as follows:

SISD (single instruction stream, single data stream).
SIMD (single instruction stream, multiple data stream).
MISD (Multiple instruction stream, single data stream).
MIMD (Multiple instruction stream, multiple data stream).

SISD:- SISD represents the organization of a single computer containing a control unit, a processor unit & a memory unit. Instructions are executed sequentially and the system may/maynot have internal parallel processing.

SIMD: SIMD Represents an organization that includes many processing units under the supervision of a common control unit. All the processors receive the same instruction from the control unit, but operate on different items of data. The share memory units must contain multiple modules so that it can will the processors simultaneously.

Scanned by CamScanner

MISD: MISD structure is theoritical interest but there is no practical system.

MIMD :- MIMD organization refers to a computer system capable of processing several programs at the same time.

Pipelining :-

→ Pipelining is a technique of decomposing a sequential process into suboperations, with each sub-process being executed in a special dedicated segment that operates concurrently with all other segment.

→ Each segment performs partial processing in which the tasks is partitioned. The result obtained from the computation in each segment is transferred to the next segment in the pipeline.

→ The final result is obtained after the data have passed through all segments.

→ It is characteristic of pipelines that several computations can be in progress in distinct segments at the same time.

→ The overlapping is also possible, by associating a register with each segment in the pipeline.

→ The simplest way is to implement pipeline is to imagine that each segment consists of an i/p register followed by combinational circuit.

→ The register holds the data & in a given segment is applied to the i/p register of the next segment.

→ A clock is applied to all registers. In this way,

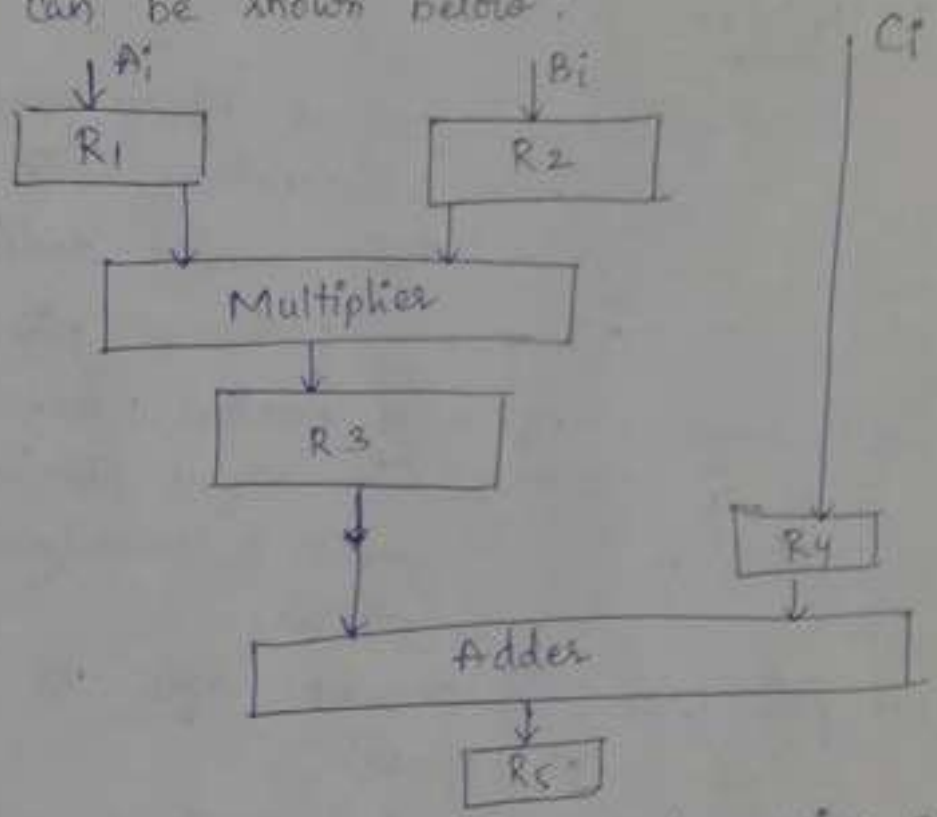
Scanned by CamScanner

information flows through the pipeline one step at a time.

Eg:- Suppose that want to perform the combined multiply & add operations with a stream of numbers

$$A_i * B_i * C_i \text{ for } i = 1 \text{ to } 7.$$

This can be shown below:



The registers R1 through R5 that receive new data with every clock pulse. The multiplier & adder are combinational circuits. The suboperations performed in each segment of the pipeline are follows:

$R1 \leftarrow A_i$, $R2 \leftarrow B_i$	Input A_i & B_i
$R3 \leftarrow R1 * R2$, $R4 \leftarrow C_i$	Multiply & i/p C_i
$R5 \leftarrow R3 + R4$	Add C_i to product

Scanned by CamScanner

The operands pass through all four segments in a fixed sequence.

→ Each segment consists of a combinational circuit S_i that performs a suboperation over the data stream flowing through the pipe.

→ The segments are separated by registers R_i that holds the intermediate results between the stages.

→ Information flows between adjacent stages under the control of a common clock applied to all registers simultaneously.

→ The 'task' is defined as the total operation performed going through all the segments in the pipeline.

Space time diagram :-

The behaviour of a pipeline can be illustrated with a space-time diagram. The diagram shows the segment utilization as a function of time. This can be shown in below fig.

	1	2	3	4	5	6	7	8	9
segment 1	T_1	T_2	T_3	T_4	T_5	T_6			
segment 2		T_1	T_2	T_3	T_4	T_5	T_6		
3			T_1	T_2	T_3	T_4	T_5	T_6	
4				T_1	T_2	T_3	T_4	T_5	T_6

fig: Space time diagram for pipeline.

Scanned by CamScanner

General Considerations :-

The general ~~structure of a four-segment~~ pipeline ~~is shown below~~

Table : Content of Registers in pipeline Example

clock pulse Number	Segment 1		Segment 2		Segment 3
	R1	R2	R3	R4	R5
1	A_1	B_1	–	–	–
2	A_2	B_2	$A_1 * B_1$	C_1	–
3	A_3	B_3	$A_2 * B_2$	C_2	$A_1 * B_1 + C_1$
4	A_4	B_4	$A_3 * B_3$	C_3	$A_2 * B_2 + C_2$
5	A_5	B_5	$A_4 * B_4$	C_4	$A_3 * B_3 + C_3$
6	A_6	B_6	$A_5 * B_5$	C_5	$A_4 * B_4 + C_4$
7	A_7	B_7	$A_6 * B_6$	C_6	$A_5 * B_5 + C_5$
8	–	–	$A_7 * B_7$	C_7	$A_6 * B_6 + C_6$
9	–	–	–	–	$A_7 * B_7 + C_7$

The first clock pulse transfers A_1 and B_1 into R1 and R2. The second clock pulse transfers the product of R1 and R2 into R3 and C_1 into R4. → The same clock pulse transfers A_2 and B_2 into R1 and R2. The third clock pulse operates on all three segments simultaneously.

General Considerations :-

The general structure of a four-segment pipeline is shown below

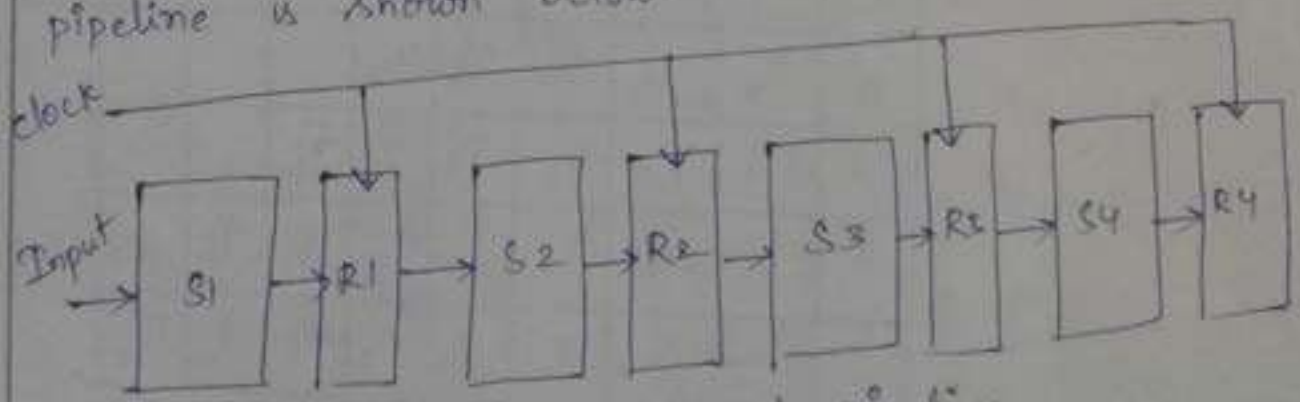


fig:- four segment pipeline.

Scanned by CamScanner

The diagram shows 6 tasks T_1 through T_6 executed in four segments. Interally task T_1 is handled by segment 1.

→ After the first clock, segment 2 is busy with T_1, while segment 1 is busy with task T_2.

Continuing this manner, the first task T_1 is completed after the fourth clock cycle. Once the pipeline is full, it takes only one clock period to obtain an o/p.

Now consider the case where a k-segment pipeline with a clock cycle time t_p is used to execute 'n' tasks.

The first task T_1 requires a time equal to kt_p to complete its operation. The remaining $(n-1)$ tasks emerge from the pipe at the rate of one task per clock cycle & they will be completed after a time equal to $(n-1)t_p$.

Therefore, to complete 'n' tasks using a k-segment pipeline requires $k+(n-1)$ clock cycles.

Consider a Non-pipeline unit that performs the same operation & takes a time equal to t_n to complete each task.

The total time required for 'n' tasks is nt_n.

The speedup of a pipeline processing over an equivalent non pipeline processing is defined by

Scanned by CamScanner

the ratio.

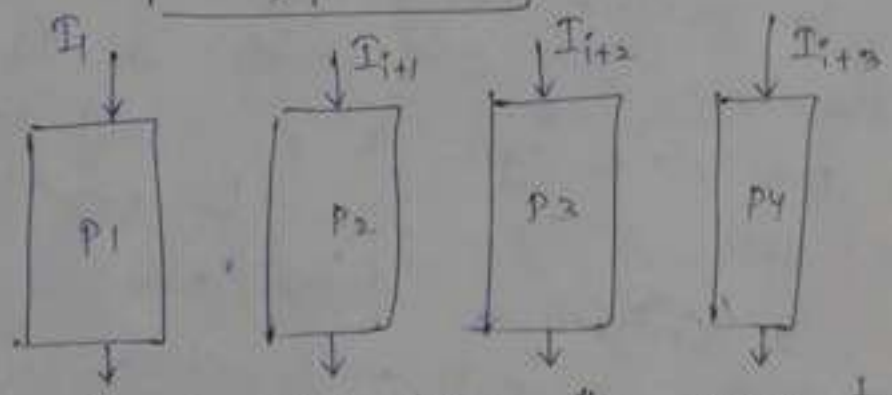
$$S = \frac{n t_n}{(k+(n-1))t_p}$$

As the no. of tasks increases, n becomes much larger than k-1, & k+n-1 approaches the value of n under this condition, the speed up becomes.

$$S = \frac{t_n}{t_p}.$$

We assume that the time it takes to process a task is the same in the pipeline & non-pipeline circuits; we have $t_n = k t_p$. The speed up reduces to

$$S = \frac{k\cancel{t_p}}{\cancel{t_p}} = k$$



Each p circuit performs the same task of an equivalent pipeline circuit. The parallel i/p's accepts the data simultaneously & perform four tasks at the same time.

The above fig. shows SIMD, in this also same instruction is used to operate on multiple data in parallel.

Scanned by CamScanner

Pipeline can't operate for many reasons. It has maximum theoritical rate. we are n't always going to be considered that the non-pipe circuit has the same time delay as that of pipeline circuit. pipeline provides a faster operation over a purely serial sequence.

Pipeline organization can be applicable in two areas.

1) Arithmetic pipeline
2) Instruction pipeline

An arithmetic pipeline divides an arithmetic operation into suboperations for execution in the pipeline segments.

An instruction pipeline operates on a stream of instructions by overlapping the fetch, decode & execute phases of the instruction cycle.

Parallel processing can be applied to following topics :-

(1) Pipeline Processing
(2) Vector Processing
(3) Array Processing.

Parallel processing points

Scanned by CamScanner

Arithmetic pipeline:-

The pipeline arithmetic units are usually found in high speed Computers.

→ They are used to implement floating-point operations, multiplication of fixed-point numbers.

→ The floating-point operations are easily decomposed into sub-operations.

→ The i/p's to the floating-point adder pipeline are two normalized floating-point binary numbers:

$$X = A \times 2^a \quad , \quad Y = B \times 2^b$$

A, B are two fractions that represent the mantissa and a & b are the exponents.

The floating-point addition & subtraction can be performed in four segments which is shown in fig.

The register labeled 'R' are placed b/w the segments to store intermediate results. The suboperations that are performed to four segments are:

1) Compare the exponents
2) Align the mantissas
3) Add or subtract the mantissas.
4) Normalize the result.

Scanned by CamScanner

The exponents are compared by subtracting them to their difference. The larger exponent is chosen as the exponent of the result. The exponent difference determines how many times the mantissa associated with the smaller exponent must be shifted to right.

The result is normalized in segment D. When an overflow occurs, the mantissa of the sum or difference is shifted right & the exponent incremented by one.

If an underflow occurs, the no. of leading zero's in the mantissa determines the number of left shifts in the mantissa and the number that must be subtracted then from the exponent. be

Eg :- Two normalized floating point Numbers

$$X = 0.9504 \times 10^3, \quad Y = 0.8200 \times 10^2$$

The two exponents are subtracted $3-2=1$. The larger exponent 3 is chosen as the exponent of the result. The next segment shifts the mantissa of Y to the right to obtain,

$$X = 0.9504 \times 10^3, \quad Y = 0.0820 \times 10^3.$$

The exponents are same, then addition of two mantissas in segment S produces the sum.

$$Z = 1.0324 \times 10^3$$

The sum is adjusting by normalizing the result.

$$Z = 0.10324 \times 10^4$$

Scanned by CamScanner

This is done by the shifting the mantissa once to the right & incrementing the exponent by one to obtain the normalized sum.

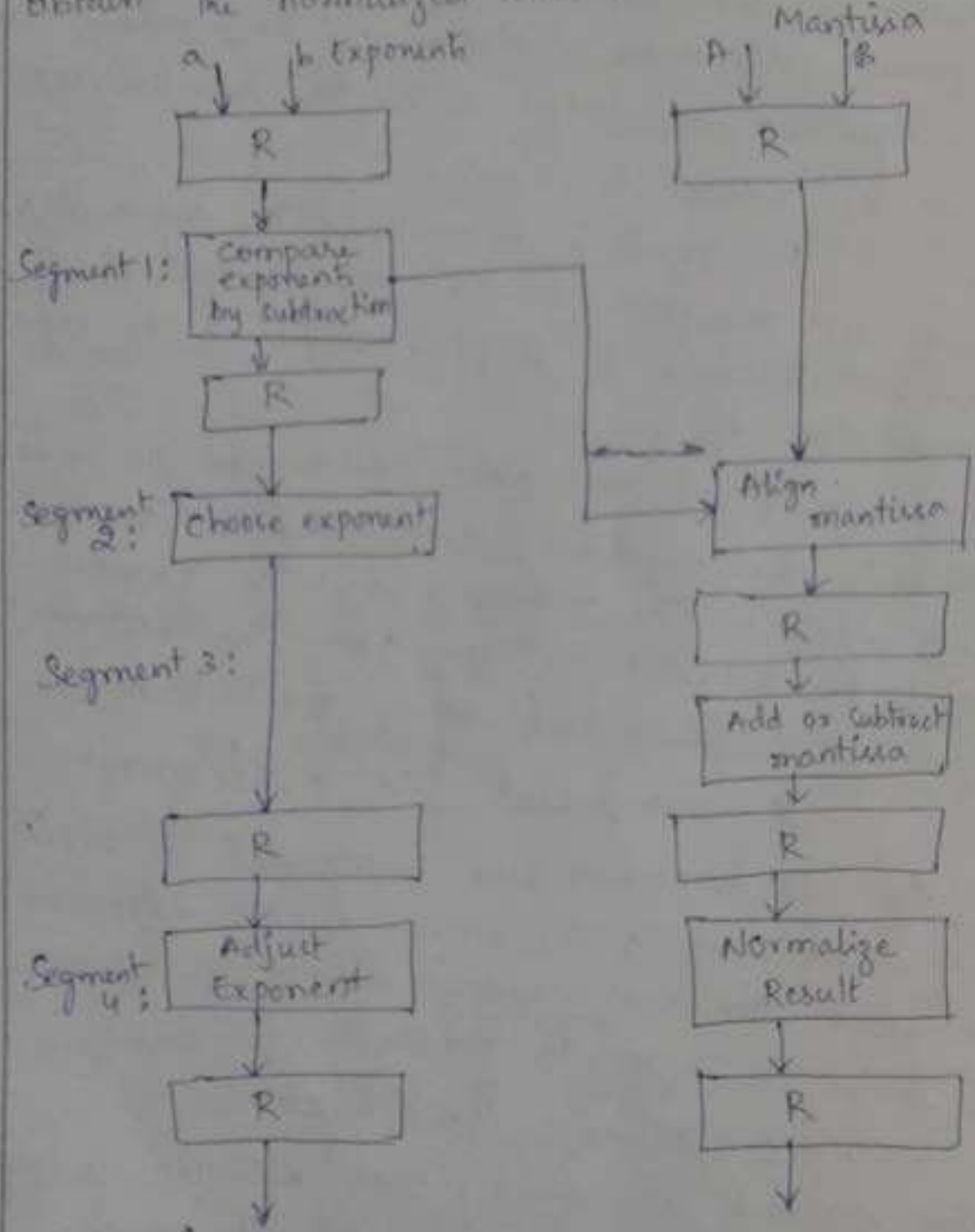


fig: Pipeline for floating-point addition and subtraction.

The sum is adjusted by normalizing the result. This is done by shifting mantissa to right at once.

$Z = 0.10324 \times 10^4$.

Scanned by CamScanner

Instruction Pipeline :-

An instruction pipeline reads consecutive locations from memory while previous instructions are being executed in other segments. This causes the instruction fetch & execute phases to overlap and performs simultaneous operation.

Consider a computer with an instruction fetch unit and an instruction execution unit designed to provide a two segment pipeline. The instruction fetch segment can be implemented by means of a first-in, FIFO buffer. This unit uses queue whenever the execution unit is not using memory, the control increments the PC & uses its address value to read consecutive instructions from memory.

→ Thus an instruction stream can be placed on the queue, waiting for decoding and processing by the execution segment.

→ This means reduce the average access time for memory for reading instructions and it works in a efficient way.

→ whenever there is space in the FIFO buffer, the control unit initiates the next instruction fetch phase. The buffer acts as a queue from which

Scanned by CamScanner

Control then extracts the instruction for the execution unit.

Computers with complex instructions require other phases in addition to fetch & execute phases. The following steps are.

1) Fetch the instruction from memory.
2) Decode the instruction.
3) Calculate the effective address.
4) Fetch the operands from memory.
5) Execute the instruction.
6) Store the result.

The instruction pipeline can prevent from difficulty at its maximum rate. Different segments may take different times to operate on the incoming information. Some segments are skipped for certain operations

for eg. A register mode instruction doesnot need in effective address but remaining other segments may require it. The instruction must wait until another one is finished. So, Memory access conflicts are resolved by using two memory buses for accessing instructions & data in separate modules.

Scanned by CamScanner

Example :- four Segment Instruction Pipeline

→ Assume, decode and effective address is Combined into one segment.

→ Execution and storing result can be combined into one segment

It reduced instruction pipeline into four segments

S1 : Fetch Instruction FI

S2 : Decode + cal. Eff. address DA

S3 : Fetch operand from memory FO

S4 : Execute and store result (EX)

Instruction cycle in CPU for four segment :-

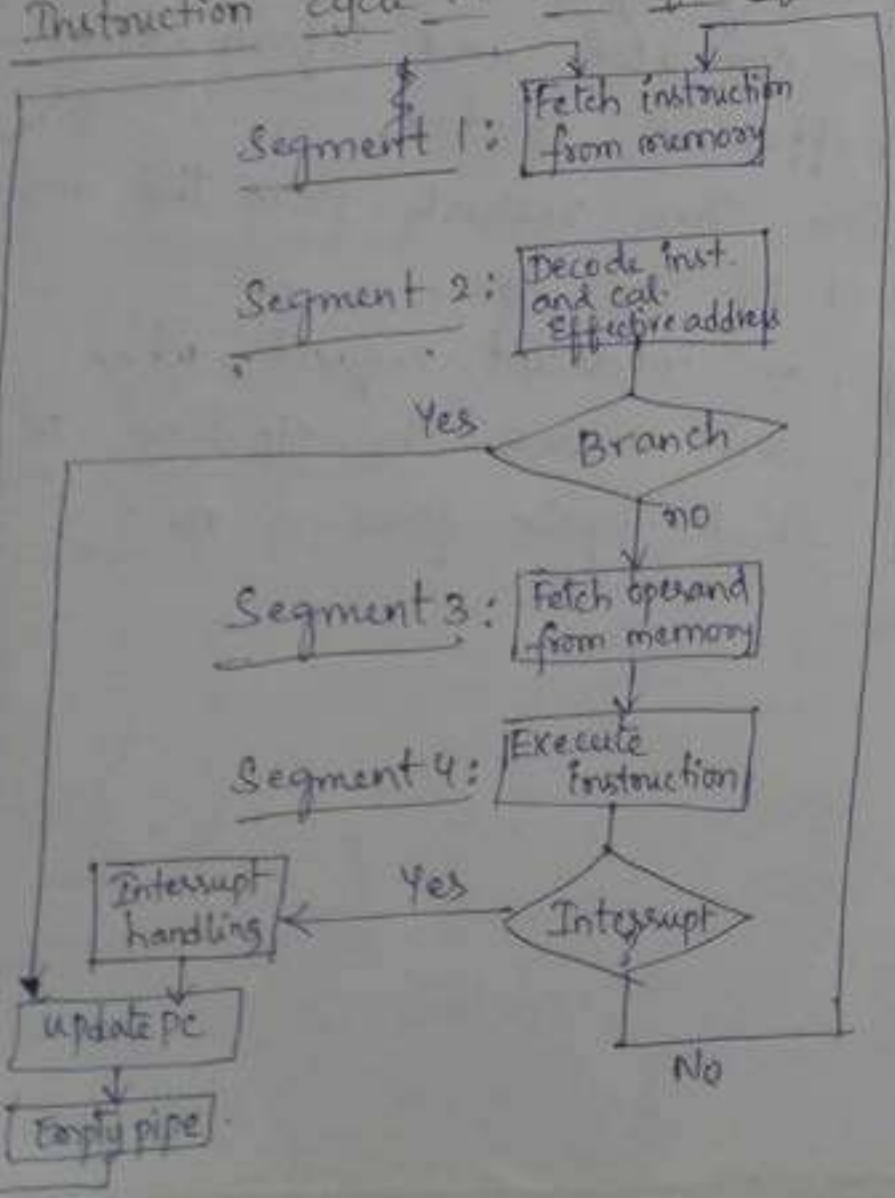


fig:- four-segment CPU pipeline.

Scanned by CamScanner

→ while an instruction is being executed in segment 4, the next instruction in sequence is busy fetching an operand from memory in segment 3.

→ The effective address may be calculated in a ~~separate memory in segmen~~ arithmetic unit for the third instruction. and whenever memory is available, the fourth and subsequent instructions can be fetched into instruction FIFO.

→ Once in while, the instruction may cause branch out of normal sequence. In that case, the pending operations in last two segments are completed and information stored in buffer is deleted.

→ The pipeline then restarts from the new address stored in the PC.

→ Similarly, an interrupt request, when acknowledged, will cause the pipeline to empty and start again from a new address value.

Scanned by CamScanner

Step:	1	2	3	4	5	6	7	8	9	10	11	12	13
Instruction 1	FI	DA	FO	EX									
2		FI	DA	FO	EX								
Branch 3			FI	DA	FO	EX							
4				FI	–	–	FI	DA	FO	EX			
5					–	–	–	FI	DA	FO	EX		
6									FI	DA	FO	EX	
7										FI	DA	FO	EX

fig :- Timing of instruction pipeline.

The time in the horizontal axis is divided into steps of equal duration.

FI - fetch instruction
DA - Decode and calculate effective address
FO - Fetches the operand
EX - Executes the instruction

→ In step 4, instruction 1 is being executed in segment EX, the operand for instruction 2 is being fetched in segment FO, instruction 3 is being decoded in segment DA, instruction 4 is being fetched from memory in segment FI.

→ As soon as the instruction is decoded in segment DA in step 4, the transfer from FI to DA of other instructions is halted, until branch instructions is executed in step 6.

Scanned by CamScanner

Three major difficulties which make the instruction pipeline to deviate from its normal operation.

(1) Resource Conflicts :- It caused by access to memory by two segments at the same time.

→ It can be resolved by using separate instruction and data memories.

(2) Data dependency :- It arise when an instruction depends on the result of previous instruction, but this result is not yet available.

3) Branch difficulties :- It arise from branch and other instructions that change the value of PC.

Data Dependency :-

→ A data dependency occurs when an instruction needs data that are not yet available.

Eg :- An instruction in FO segment may need to fetch an operand that is being generated at same time by the previous instruction in segment Ex'.

→ Similarly, an address dependency may occur when an operand address cannot be calculated because the information needed by the addressing mode is not available.

→ Pipelined computers deal with such conflicts between data dependencies in a variety of ways

Scanned by CamScanner

① hardware interlocks :-

An interlock is a circuit that detects instructions whose source operands are destinations of instructions farther up in the pipeline.

→ This approach maintains the program sequence by using hardware to insert the required delays.

② Operand forwarding :-

→ A special hardware to detect a conflict and then avoid it by routing the data through special paths between pipeline segments.

Eg :- Instead of transferring an ALU result to destination register, the hardware checks the destination operand and if it is needed as a source in the next instruction, then it passes the result directly to ALU input.

③ Delayed Load :- Delaying the loading of conflicting data by inserting no-operation instructions.

Handling of Branch Instructions :-

→ Branch instruction breaks the normal sequence of the instruction stream, causing difficulties in the operation of the instruction pipeline.

→ Various hardware techniques are implemented to minimize the performance degradation, caused by instruction branching.

Scanned by CamScanner

Prefetch target Instruction :-

One way of handling the conditional branch is to prefetch target Instruction in addition to the instruction following the branch.

→ If the branch condition is successful, the pipeline continues from the branch target instruction.

Branch target Buffer :- or BTB

→ The BTB is an associative memory included in the fetch segment of the pipeline.

→ Every Entry in the BTB consists of the address of a previously executed branch instruction and the target instruction for that branch.

→ When the pipeline decodes a branch instruction, it searches the associative memory BTB for the address of the instruction.

The advantage of this scheme is that branch instructions that have occured previously are readily available in the pipeline without interruption.

Loop Buffer :-

A variation of the BTB is the 'loop Buffer'. This is a small very high speed register file maintained by the instruction

Scanned by CamScanner

fetch segment of the pipeline. When a program loop is detected in the program, it is stored in the loop buffer in its entirety, including all branches.

branch prediction :-

A pipeline with branch prediction uses some additional logic to guess the outcome of a conditional branch instruction before it is executed. → A correct prediction eliminates the wasted time caused by branch penalties.

Delayed branch :- It is employed in RISC processors. In this method, the compiler detects the branch instructions and rearranges the machine language code sequence by inserting useful instructions that keep the pipeline operating without interruptions.

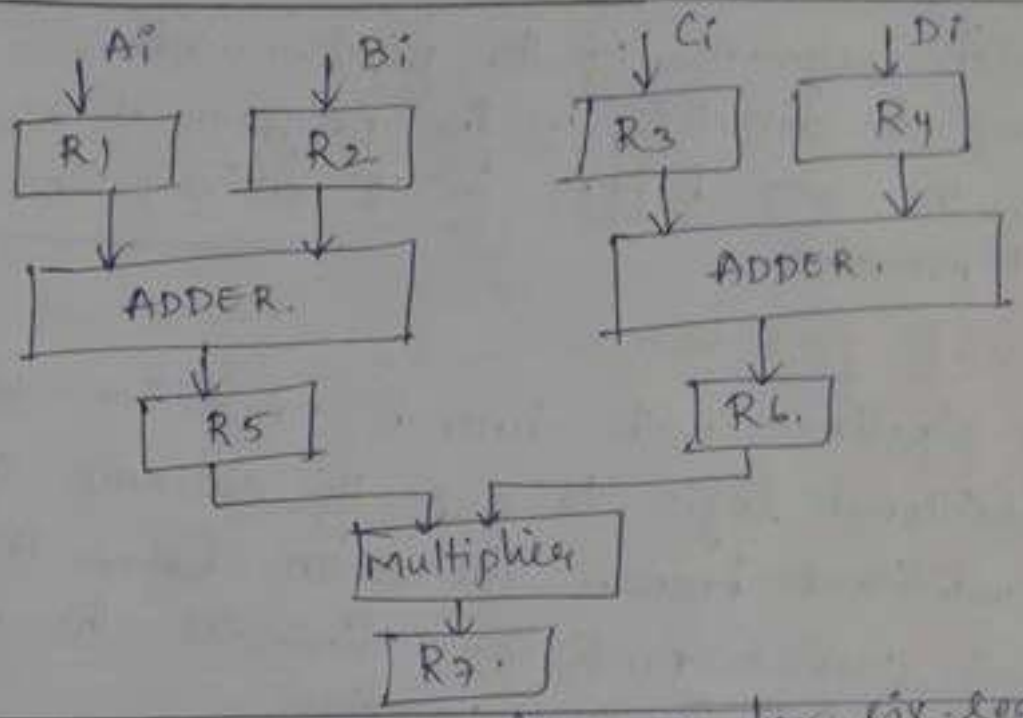
Eg :- Insertion of no-operation instruction after a branch instruction.

PROBLEMS

Q) Construct pipeline configuration to carry the task. List the contents of all registers in the pipeline for $i = 1$ through 6.

$$(A_i + B_i)(C_i + D_i)$$

Scanned by CamScanner



Q) Draw a space-time diagram for six-segment pipeline showing the time it takes to process eight tasks.

Sol:

Segment	1	2	3	4	5	6	7	8	9	10	11	12	13
1	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8					
2		T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8				
3			T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8			
4				T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8		
5					T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	
6						T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8

$(k+n-1)t_p = 6+8-1 = 13$ cycles.

k = no. of segments n = No. of Tasks.

Q) Determine no. of clock cycles it takes to process 200 tasks in a six segment pipeline.

Sol: Given n = no. of tasks = 200, k = no. of segments = 6

No. of clock cycles $= k+n-1$
$= 6+200-1 \Rightarrow 206-1 \Rightarrow 205$ clock cycle.

Scanned by CamScanner

Q) A Nonpipeline system takes 50 ns to process a task. The same task can be processed in a six-segment pipeline with a clock cycle of 10 ns. Determine the speedup ratio of pipeline for 100 tasks. Maximum speed can be achieved?

Sol:- $n = 100$ tasks.

$t_{np} = 50\,ns$ $\quad t_p = 10\,ns$, $k = 6$ segments

$$\boxed{\text{Speedup ratio } S = \frac{n t_{np}}{(k+n-1)t_p}} = \frac{100 \times 50}{(6+100-1) \times 10}$$

$$= \frac{100 \times 50}{106 \times 10} = \frac{500}{106} \simeq 4.76.$$

$$\boxed{S_{max} = \frac{t_n}{t_p}} = \frac{50}{10} = 5$$

Q) The pipeline has following propagation time: 40 ns for operands to be read from memory into registers R1 and R2, 45 ns for multiplier, 5 ns to transfer to R3 and 15 ns to add the two numbers into R5 (Refer arithmetic pipeline diagram)

a) what is the minimum clock cycle time that can be used?

Ans:- $t_p = 45 + 5 = 50\,ns$ $\quad (\because k = 3)$.

b) A nonpipeline performs the same operation by removing R3 and R4. How long it multiply and add operands without pipeline

Ans:- $t_n = 40 + 45 + 15 = 100\,ns$.

Scanned by CamScanner

c) Calculate the speedup ratio of pipeline for 10 tasks and again for 100 tasks.

$n = 100$, $t_p = 50ns$, $t_n = 100ns$ [from above (a) & (b) info]

Ans :-

$$S = \frac{n t_n}{(k+n-1) t_p} = \frac{10 \times 100}{(3+9) \times 50}$$

$$= 1.67 \text{ for } n=10$$

$$= \frac{100 \times 100}{(3+99) \times 50} = 1.96 \text{ for } n=100$$

Note :- t_p & t_n values are taken from (a) & (b) bits of this problem.

d) What is the maximum speedup that can be achieved ?

$$S_{max} = \frac{t_n}{t_p} = \frac{100}{50} = 2$$

Q) The time delay of four segments in a pipeline of fig :- asithmetic pipeline (floating point numbers add) are as follows;

$t_1 = 50ns$, $t_2 = 30ns$, $t_3 = 95ns$, and $t_4 = 45ns$.

The interface registers delay time $t_4 = 5ns$.

a) How long would it take to add 100 pairs of numbers in the pipeline ?

Sol :- clock cycle $= 95 + 5 = 100ns$, Time to add 100 pairs $n = 100$, $k = 4$, $t_p = 100ns$, Time $= (k+n-1)t_p = 10,200ns$

b) How can we reduce the total time to about one-half of the time calculated in part (a)

Sol :- Divide segment 3 into two segments of $50+5=55$ and $45+5=50ns$. $t_p = 55ns$, $k=5$; $t_p = 55ns$, $k=5$

$$(k+n-1)t_p = (5+99)\,55 = 5,720ns = 5.72\,\mu s.$$

Scanned by CamScanner

RISC Pipeline :-

RISC has ability to use an instruction pipeline in an effective manner.

→ The simplicity of the instruction set can be utilized to implement an instruction pipeline using a small no. of suboperations with each being executed in one clock cycle.

→ The instruction format is of fixed length, so, the decoding of the operation can occur at the same time as the register selection

→ All data manipulation instructions have register-to-register operations. All operands are in registers, there is no need for calculating any effective address.

→ The instruction pipeline can be implemented with two or three segments.

1) One segment fetches the instruction from program memory.
2) Other segment executes the instruction in the ALU
3) Third segment may be used to store the result of the ALU operation in a destination register.

Scanned by CamScanner

The data transfer instructions in RISC are limited to load & store instructions. These instructions use register indirect addressing. They usually need three or four stages in the pipeline.

→ To prevent conflicts b/w a memory access to fetch an instruction & to load or store an operand, the RISC provide two separate buses with two memories:

1) for storing the instructions.
2) for storing the data

The two memories can some time operate at the same speed as the CPU clock & are referred to as cache memories.

The major advantages of RISC is, it has ability to execute instructions at the rate of one per clock cycle.

→ The main goal is to achieve a single-cycle instruction execution

The advantage of RISC over CISC is that RISC can achieve pipeline segments in ~~the pipeline~~, requiring just one clock cycle while uses many segments in the pipeline, with the longest segment requiring two or more clock cycles.

Scanned by CamScanner

...truction ... from ... k cycle.

The RISC support another characteristic ie, Compiles which translates HLL into Machine language program.

The following examples show how a compiler can optimize the machine language program to compensate for pipeline Conflicts.

Three segment Instruction Pipeline :-

The data Manipulation instructions operate on data in processor registers.

→ The data transfer instructions are load and store instructions that use effective address obtained from the addition of two registers or a register and a displacement Constant provided in the instruction.

→ The program Control instructions use register values & a constant to evaluate the branch address, which is transferred to a register or the program Counter PC.

Consider the h/w operation for such a Computer. The Control fetches the instruction from program memory into an IR. The instruction is decoded at the same time that the registers needed for the execution of the instruction are selected.

Scanned by CamScanner

→ The processor unit consists of a no. of registers and an ALU that performs the necessary arithmetic logic and shift operations.

→ A data memory is used to load or store the data from a selected register in the register file.

→ The instruction cycle can be divided into three suboperations & implemented in three segments.

I : Instruction fetch

A : ALU operation

E : Execute operation.

The I segment fetches the instruction from pgm memory. The instruction is decoded & an ALU operation is performed in A segment. The ALU is used for three different function depending on the decoded ~~infor~~ instruction. It performs an operation for a data manipulation instruction, it evaluates the effective address for a load or store instruction or it calculates the branch address for an program control instruction. The E segment directs the o/p of the ALU to one of three destinations depending on the decoded instruction. It translates the result of the ALU operation into a destination register.

Clock cycles.

Scanned by CamScanner

(no-operation) instruction. This is a type of instruction that is fetched from memory but has no operation, thus wasting a clock cycle. This concept of delaying the use of data loaded from memory is referred to as 'delayed load'.

→ The same program with no-op instruction inserted after the load to R2 instruction.

clock cycle:	1	2	3	4	5	6	7
1. Load R1	I	A	E				
2. Load R2		I	A	E			
3. No-operation			I	A	E		
4. Add R1+R2				I	A	E	
5. Store R3					I	A	E

fig:- ~~Three-segment~~ pipeline timing with delayed load.

The data is loaded into R2 @ clock cycle 4. The add instruction uses the value of R2 in step 5. Thus the no-op instruction is used to advance one clock cycle in order to compensate for the data conflict in the pipeline.

Scanned by CamScanner

Delayed Load :-

Consider the operation of the following four instructions :

1. LOAD : R1 ← M[address 1]
2. LOAD : R2 ← M[address 2]
3. ADD : R3 ← R1 + R2
4. STORE : M[address 3] ← R3.

In three segment pipeline proceeds without interruptions, there will be a 'data conflict' in instruction 3 because the operand in R2 is not yet available in the A segment (ALU).

Clock Cycles	1	2	3	4	5	6
1. Load R1	I	A	E			
2. Load R2		I	A	E		
3. Add R1+R2			I	A	E	
4. Store R3				I	A	E

(a) pipeline timing with data conflict

The E segment in clock cycle 4 is in process of placing the memory data into R2. The A segment in clock cycle 4 is using the data from R2, but the value in R2 will not be the correct value since it has not yet been transferred from memory.

→ If the compiler does not find a useful instruction to put after load, it inserts no-op

Delayed Branch :-

→ The Branch instruction delays the pipeline operation until the instruction at the branch address is fetched.

→ The method used by RISC processors is to rely on the Compiler to redefine the branches so that they effect at the proper time in the pipeline. This method is called as 'delayed branch'

→ The Compiler for a processor that uses delayed branches is designed to analyze the instructions before and after the branch and rearrange the program sequence by inserting useful instructions in the delay steps.

→ The program for branch instructions consists of five instructions :

Load from memory to R1
Increment R2
Add R3 to R4
Subtract R5 from R6
Branch to address X

→ The Compiler inserts (in fig) two no-op instructions after the branch. The branch address X is transferred to PC in clock cycle 7. The fetching of instruction at X is

Scanned by CamScanner

delayed by two clock cycles by the no-op instructions. The instruction at X starts the fetch phase at clock cycle 8 after the PC has been updated.

clock cycles:	1	2	3	4	5	6	7	8	9	10
1. Load	I	A	E							
2. Increment		I	A	E						
3. Add			I	A	E					
4. Subtract				I	A	E				
5. Branch to X					I	A	E			
6. No-operation						I	A	E		
7. No-operation							I	A	E	
8. Instruction in X								I	A	E

(a) Using no-operation instructions.

clock cycles:	1	2	3	4	5	6	7	8
1. Load	I	A	E					
2. Increment		I	A	E				
3. Branch to X			I	A	E			
4. Add				I	A	E		
5. Subtract					I	A	E	
6. Instruction in X						I	A	E

(b) Rearranging the Instructions

fig:- Example of delayed branch

→ The pgm is rearranged, by placing add & sub inst. after branch inst. Inspection of the pipeline timing shows that PC is updated to the value of X in clock cycle 5, but the add & sub inst. are fetched from memory and executed in proper sequence

Vector Processing :-

In many scientific & Engineering applications the problems can be formulated in terms of vectors & matrices that lend themselves to vector processing.

Computers with vector processing capabilities are in demand in specialized applications.

The following are representative applications areas where vector processing are used.

- Long range weather forecasting
- petroleum explorations
- seismic data analysis
- Medical diagnosis
- Aerodynamics & space flight simulations.
- Artificial intelligence & expert systems.
- Mapping the human genome.
- Image processing.

To achieve the required level of high performance it is necessary to utilize the fastest & most reliable h/w & innovative procedures from vector & parallel processing

Scanned by CamScanner

Vector operations :-

Many scientific problems require arithmetic operations on large arrays of numbers. These numbers are usually formulated as vectors and matrices of floating-point numbers.

A vector is an ordered set of one-dimensional array of data items. A vector 'V' of length n is represented as row vector by

$V = [V_1\ V_2\ V_3 \ldots V_n]$

→ It may be represented as Column vector. Consequently, operations on vectors must be broken down into single computations with subscripted variables.

→ The element V_i of vector V is written as V(I) and the index I refers to a memory address or register where the number is stored.

Consider the following 'Do' loop :

```
DO  20 I = 1, 100
    C(I) = B(I) + A(I)
```

This a program for adding two vectors A & B of length 100 to produce a vector C.

A possible instruction format for a vector instruction is shown below :

Operation Code	Base address source 1	Base address source 2	Base address destination	Vector length

Scanned by CamScanner

Matrix Multiplication :-

Matrix Multiplication is one of the most computational intensive operations performed in computers with vector processors.

→ The multiplication of two nxn matrices consists of n^2 inner products or n^3 multiply-add operations.

→ In nxm matrix of numbers has 'n' rows and 'm' columns and may be considered as constituting a set of n row vectors or a set of 'm' column vectors

Eg:- Multiplication of 3x3 matrices A and B

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \times \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix}$$

The product matrix C is a 3x3 matrix whose elements are related to elements of A and B by the inner product :

$$c_{ij} = \sum_{k=1}^{3} a_{ik} \times b_{kj}$$

Eg :- $i = 1, j = 1$

$$c_{11} = a_{11} b_{11} + a_{12} b_{21} + a_{12} b_{31}$$

This requires three multiplications and three additions. The total number of multiplications or additions required to compute the matrix product

Scanned by CamScanner

is $9 \times 3 = 27$.

In general, the inner product consists of the sum of K product terms of the form.

$$C = A_1B_1 + A_2B_2 + A_3B_3 + A_4B_4 + \dots + A_KB_K$$

In typical applications K may be equal to 100 or 1000.

The inner product calculation on a pipeline vector processor is shown in below figure.

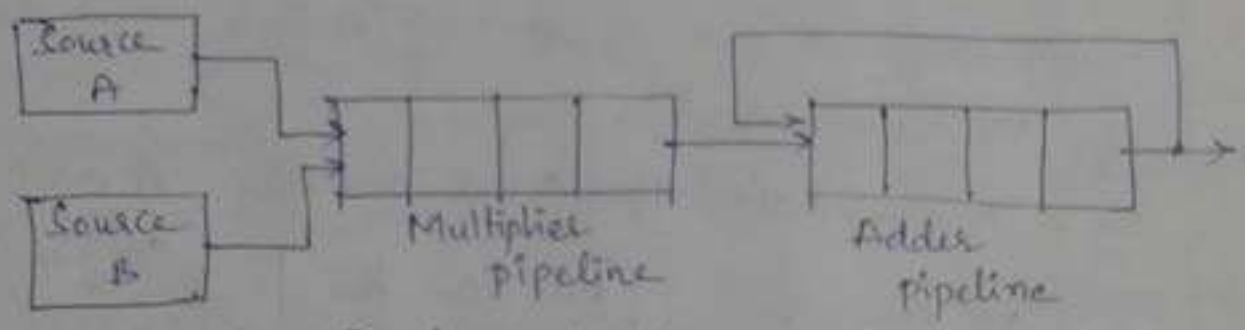


fig :- pipeline for calculating an inner product.

The values of A and B are either in memory or processor registers.

→ The floating-point multiplier pipeline and the floating-point adder pipeline are assumed to have four segments each.

→ All segment registers in the multiplier and adder are initialized to 0.

→ Therefore, the o/p of the adder is 0 for the first eight cycles until both pipes are full.

→ A_i and B_i pairs are brought in and multiplied at a rate of one pair per cycle.

Scanned by CamScanner

→ After first four cycles, the products begin to be added to the o/p of the adder.

→ During the next four cycles 0 is added to the products entering the adder pipeline.

→ At the end of the eighth cycle, the first four products A_1B_1 through A_4B_4 are the four adder segments and next four products A_5B_5 through A_8B_8 are in the multiplier segments.

→ Thus the ninth cycle starts the addition $A_1B_1 + A_5B_5$ in the adder pipeline.

→ The tenth cycle starts the addition $A_2B_2 + A_6B_6$ and so on.

$$
\begin{aligned}
C = {} & A_1B_1 + A_5B_5 + A_9B_9 + A_{13}B_{13} + \cdots \\
& + A_2B_2 + A_6B_6 + A_{10}B_{10} + A_{14}B_{14} + \cdots \\
& + A_3B_3 + A_7B_7 + A_{11}B_{11} + A_{15}B_{15} \cdots \\
& + A_4B_4 + A_8B_8 + A_{12}B_{12} + A_{16}B_{16} \cdots
\end{aligned}
$$

Memory Interleaving:-

pipeline and vector processors often require simultaneous access to memory from two or more sources. An instruction pipeline may require the fetching of an instruction and an operand at the same time from two different segments.

→ Similarly, an arithmetic pipeline usually requires two or more operands to enter the pipeline at the same time.

Scanned by CamScanner

→ Instead of using two memory buses for simultaneous access, the memory can be partitioned into a number of modules connected to a common memory address and data buses.

→ A memory module is a memory array together with its own address and data registers.

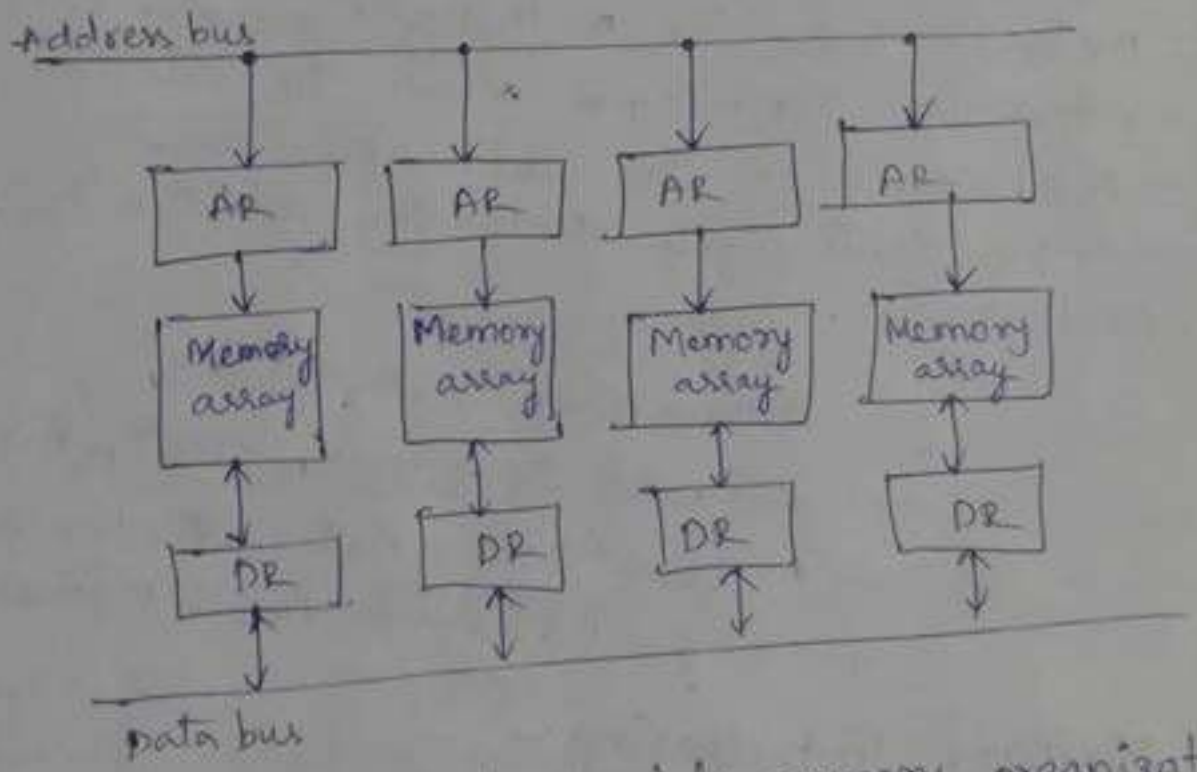


fig:- Multiple module memory organization.

→ Each memory array has its own address register AR and data register DR.

→ The address Register receive information from a common address bus and the data registers communicate with a bidirectional bus.

→ The two LSB of the address can be used to distinguish b/w the four modules.

Scanned by CamScanner

→ The modular system permits one module to initiate a memory access while other modules are in the process of reading or writing a word and each modules can honor a memory request

Advantage of Modular Memory :-

→ It allows the use of a technique called 'interleaving'.

→ In an interleaved memory, different sets of addresses are assigned to different memory modules.

Eg :- In a two-module memory system, the even addresses may be in one module and the odd remaining ~~bits~~ addresses in the other.

→ when the no. of modules is a power of 2, the LSB of the address select a memory module and remaining bits designate the specific location to be accessed within the selected module.

Scanned by CamScanner

Superscalar Processors :-

→ A Superscalar processor architecture has a form of parallelism on a single chip allowing the system as a whole to run much faster than it would otherwise be able to at a given clock speed.

→ A superscalar processor architecture fetches, executes and return result from more than one instruction during a single pipeline stage.

→ A Scalar processor processes one data ~~on~~ item at a time. In a Vector processor, a single instruction operates on multiple data items simultaneously.

→ So, we can say that superscalar processor is mixture of scalar processor and vector processor

→ Each instruction processes one data item, but there are multiple processing units so that multiple instructions can be processing separate data items at the same time.

→ But just processing multiple instructions at the same time does not make an architecture superscalar.

→ where processor may be loading an instruction while doing arithmetic for previous one and storing the results from the one before that is not superscalar processing.

Scanned by CamScanner

→ In a superscalar processor, there are several functional units of the same type, along with additional circuitry to dispatch instructions to the units.

→ Eg:- most superscalar designs include more than one arithmetic and logic unit. The dispatcher reads instructions from memory and decides which ones can be run in parallel, dispatching them to two units.

→ Seymour Cray's CDC 6600 in 1965 is the first superscalar design.

RS6000 from IBM in 1990 - world's first superscalar RISC microprocessor.

Super Computers :-

→ A commercial computer with vector instructions and pipelined floating-point arithmetic operations is referred to as 'super computer'.

→ Supercomputers are very powerful, high-performance machines used mostly for scientific computations.

→ To speed up the operations, the components are packed tightly together to minimize the distance that the electronic signals have to travel.

→ It also uses special techniques for removing the heat from circuits to prevent them from burning up because of close proximity.

Scanned by CamScanner

→ The instruction set of supercomputers contains the standard data transfer, data manipulation, and program control instructions of general-purpose computers.

→ It is best known for high speed, fast and large memory system and extensive use of parallel processing. It has multiple functional units and each unit has its own pipeline configuration.

→ supercomputers is capable of general-purpose applications found in all other computers and some scientific applications, such as numerical weather forecasting, seismic wave analysis, and space research.

→ flops :- A measure used to evaluate computers in their ability to perform a given number of floating-point operations per second is called as flops.

→ The 'megaflops' is used to denote million flops and 'gigaflops' to denote billion flops.

→ If the processor can calculate floating-point operations through a pipeline each cycle time, it has ability 50 to 250 megaflops.

→ first Super Computer developed in 1976 is ~~the~~ 'Cray-1'. Early model supercomputer is 'Fujitsu VP200'.

Scanned by CamScanner

Array Processors :-

An array processor is a processor that performs computations on large arrays of data.

This array processor is divided into two types.

1) Attached Array processor.
2) SIMD Array processor.

An attached array processor is an "auxiliary processor" attached to a general-purpose computer. It is intended to improve the performance of the host computer in specific numerical computation tasks.

An SIMD array processor is a processor that has a single-instruction multiple-data organization. It manipulates vector instructions by means of multiple functional units.

Attached Array Processor :-

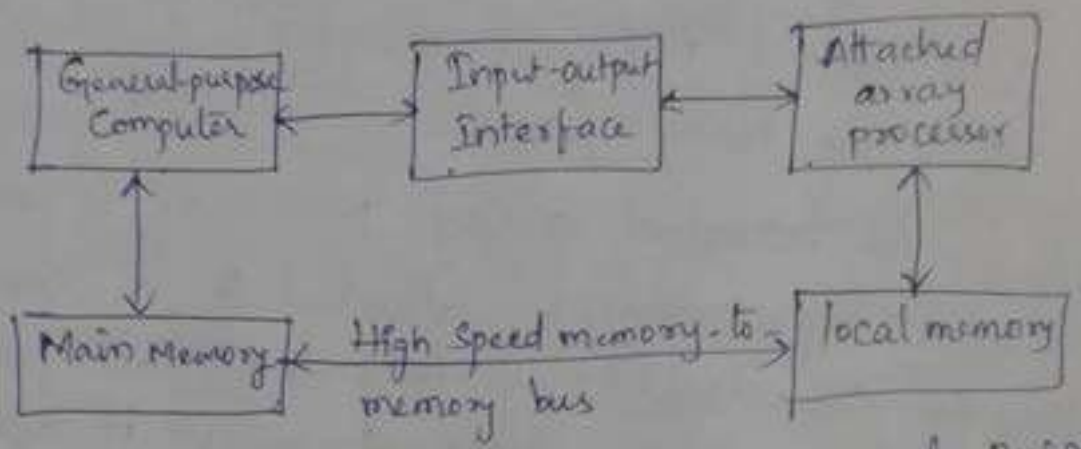
It is designed for a peripheral for a conventional host computer & its purpose to enhance the performance of the computer by providing vector processing for complex scientific applications.

→ It includes an arithmetic unit containing

Scanned by CamScanner

one or more pipelined floating point adders and multipliers. The array processor can be programmed by the user to accomodate a variety of complex arithmetic problems.

The below fig shows the interconnection of an attached array processor to host computer.



The host computer is a general-purpose ~~comp~~ commercial computer and the attached processor is a back-end machine driven by the host computer.

The array processor is connected to the computer through an i/p & o/p controller of the computer. The data for the attached processor are transferred from main memory to local memory thro' high-speed bus. The general purpose computer without the attached processor serves the user that need conventional data processing. The system with the attached processor satisfies the needs for complex arithmetic applications.

Scanned by CamScanner

5-28

SIMD Array Processor :-

→ An SIMD Array processor is a computer with multiple processing units operating in parallel.

→ The processing units are synchronized to perform the same operation under the control of a common control unit, thus providing single Instruction stream, multiple data stream (SIMD) organization.

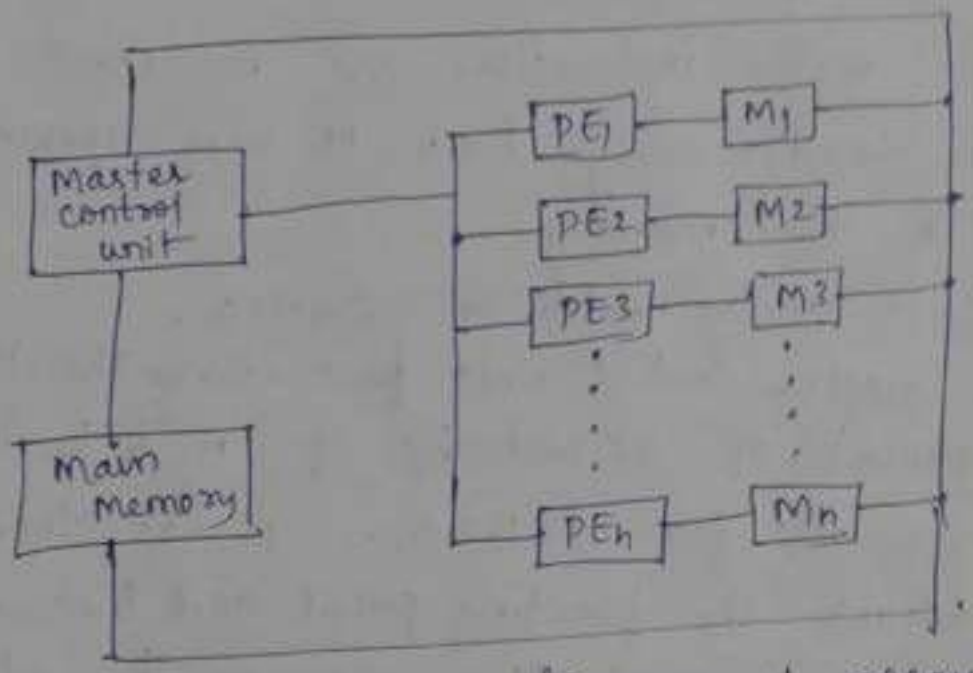


fig :- General block diagram of array processor.

→ It contains a set of identical processing elements (PEs), each having a local memory M.

→ Each processor element includes ALU, a floating point arithmetic unit and working registers.

→ The master control unit controls the operation

Scanned by CamScanner

in the processor elements.

→ The main memory is used for storage of the program.

→ The function of the master control unit is to decode the instructions and determine how the instruction is to be executed.

→ scalar and program control instructions are directly executed within the master control unit.

→ The vector instructions are broadcast to all PEs simultaneously. Each PE uses operands stored in its local memory.

Eg:- Vector Addition $C = A + B$.

The master control unit first stores the ith component of a_i and b_i of A & B in local memory M_i for $i = 1, 2, 3, \ldots n$. It, then broadcasts the floating point add instruction $C_i = a_i + b_i$ to all PEs, causing the addition to take place simultaneously. where result C_i is stored in fixed locations in each local memory.

→ The best known 'SIMD' array processor is the 'ILLIAC IV Computer' developed at university Illinois manufactured by Burroughs Corporation.

Scanned by CamScanner

Multiprocessors :-

Characteristics of Multiprocessors :-

A Multiprocessor system is an interconnection of two or more CPU's with memory & i/o equipment. The term "processor" in multiprocessor can have either CPU or i/o processor.

Multiprocessor are classified as Multiple instruction stream, multiple data stream (MIMD) systems.

There are some similarities b/w multiprocessor and multicomputer systems since both support concurrent operations.

Computers are interconnected with each other by means of communication lines to form a computer network. The network consists of several autonomous computers that may or may not communicate with each other.

A multiprocessor system is controlled by one O.S that provides interaction b/w processors & all the components of the system cooperate in the solution of a problem.

The benefits derived from a multiprocessor organization is an improved system performance. The system derives its high performance from the fact that

Scanned by CamScanner

Computations can proceed in parallel in one of two ways.

① Multiple independent jobs can be made to operate in parallel.

② A single job can be partitioned into multiple parallel tasks

Multiprocessors are classified by the way the memory is organized

1) Tightly Coupled Multiprocessor
2) Loosely Coupled Multiprocessor

A Multiprocessor system with Common shared memory is classified as a shared memory or tightly Coupled multi processor.

In this, each provide a cache memory with each CPU.

There is a global Common memory that all CPU's can access information can be shared among the CPU's by placing it in the Common global memory.

Another Alternative model of microprocessor is distributed memory or loosely coupled system. Each PE has its own private local memory. The processors are tied together by a switching scheme designed to route information from one processor to another through a message-passing scheme.

Scanned by CamScanner

The processors relay program & data to other processors in packets. A packet consists of an address, the data content & some error detection ~~error~~ code. The packets are addressed to a specific processor or taken by the first available processor.

Loosely coupled systems are most efficient when the interaction b/w tasks is minimal, where as tightly coupled systems can tolerate a higher degree of interaction b/w tasks.

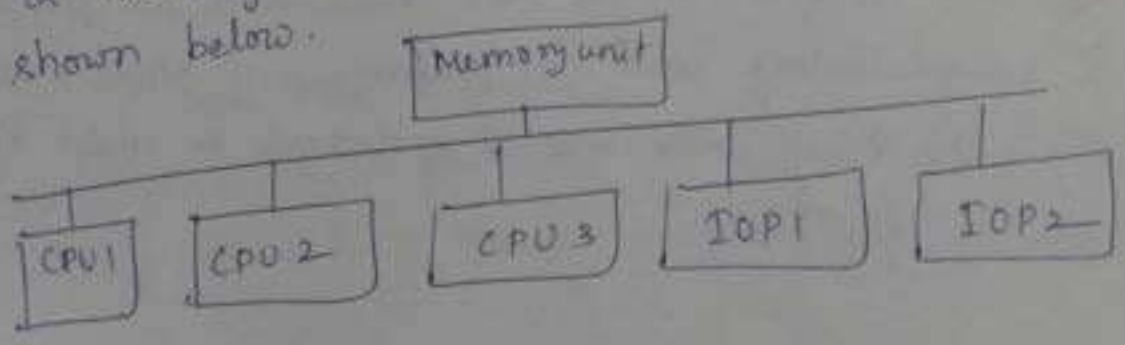
Interconnection structures :-

There are several physical forms available for establishing an interconnection n/w.

1. Time-shared common bus
2. Multiport Memory
3. cross-bar switch
4. Multistage switching
5. Hypercube system

Time shared Common bus :-

A common bus multiprocessor system consists of a no. of processors connected through a common path to a memory unit. A time shared common bus is shown below.



Scanned by CamScanner

In this, only one processor can communicate with the memory or another processor at any given time.

→ Transfer of operations are conducted by the processor that is in control of the bus at the time.

→ Any other processor wishing to initiate a transfer must first determine the availability status of the bus, only after the bus becomes available can the processor address the destination unit to initiate the transfer.

→ A command is issued to inform the destination unit what operation is to be performed.

→ The receiving unit recognizes its address in the bus and responds to the control signals from the sender, after which the transfer is initiated.

→ The system may exhibit transfer conflicts since one common bus is shared by all processors.

→ These conflicts since one common bus is shared by incorporating a bus controller that establishes priorities among the requesting units.

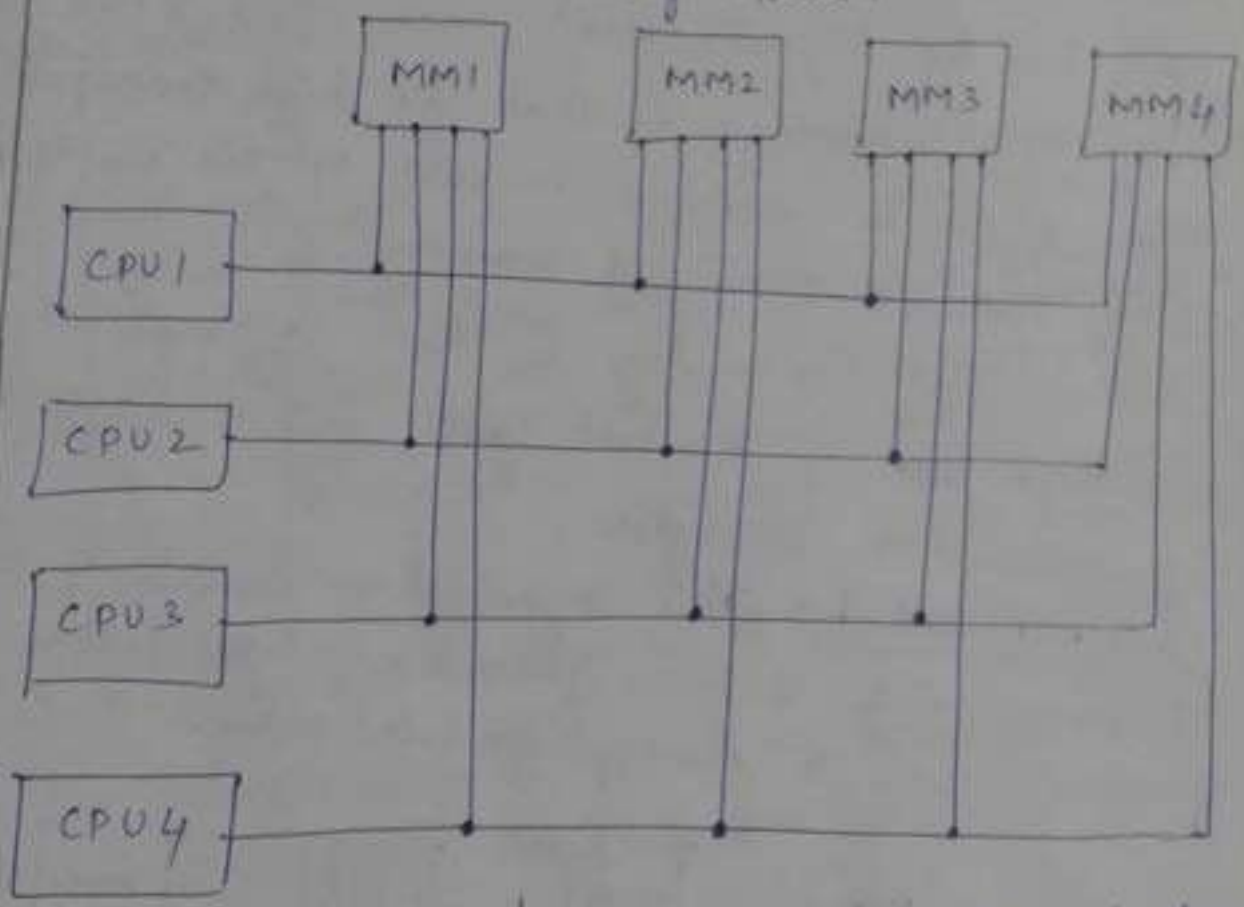
A single common bus system is restricted to one transfer at a time i.e, when one processor is communicating with the memory, all other processors are either busy with internal operations or must be idle waiting for bus.

Scanned by CamScanner

Multiport Memory :-

A multiport memory system consists of separate buses b/w each memory module & each CPU. The below fig consists for four CPU's & four memory modules(MM

Memory Modules



Each processor bus is connected to each memory module. A processor bus consists of the address, data and control lines required to communicate with memory. The MM's is said to have four ports & each port accommodates one of the buses.

The module must have control logic to determine which port will have access to memory at any given time. Memory access conflicts are resolved

Scanned by CamScanner

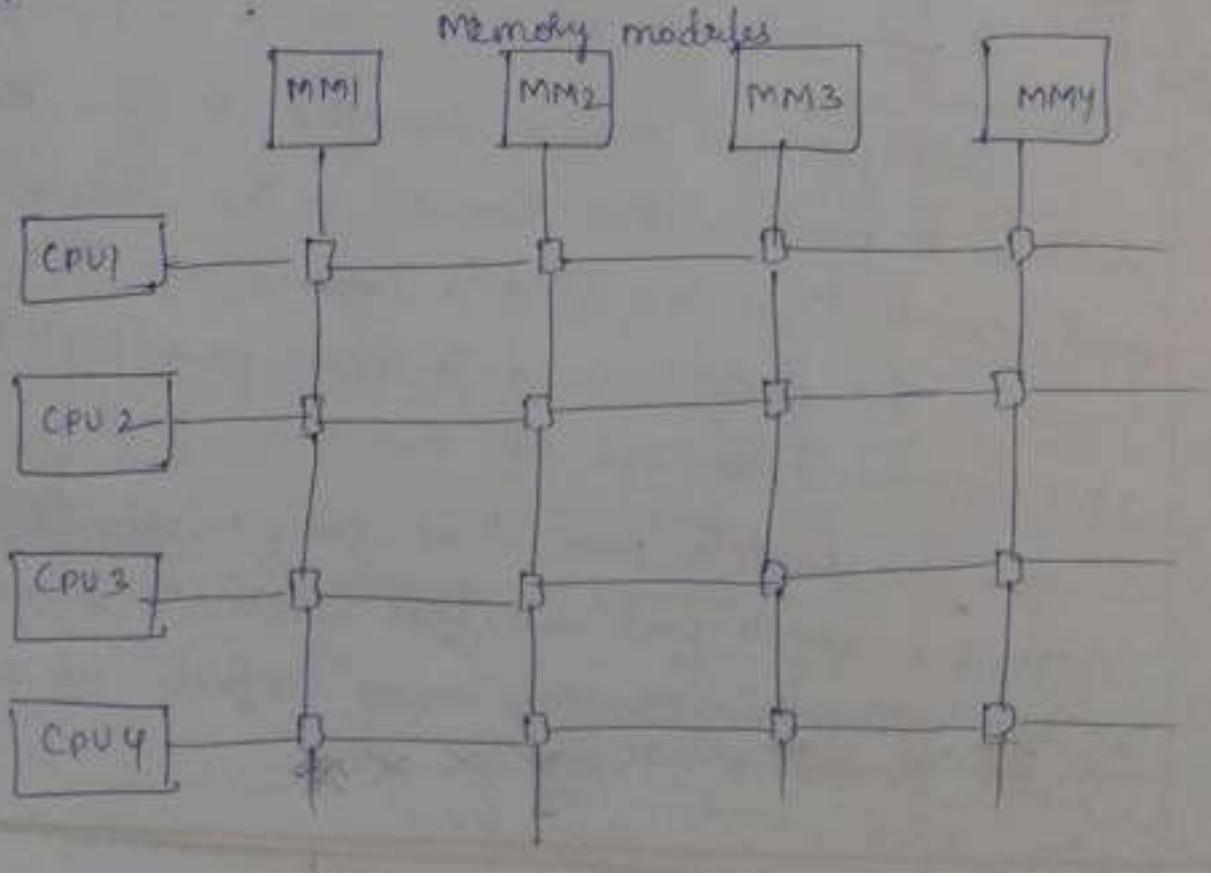
by assigning fixed priorities to each memory port. The priority for memory access associated with each processor may be established by the physical port position that its bus occupies in each Module.

CPU1 has the highest-priority than CPU2, CPU3, CPU4.

The advantage is that the high transfer rate it can be achieved because of the multiple paths b/w processors and memory.

The disadvantage is that it requires expensive memory control logic and a large no. of cables and connectors.

<u>Crossbar Switch</u>:- This organization consists of a no. of cross points that are placed at intersections b/w processor buses & memory module ports.



Scanned by CamScanner

→ The small square in each crosspoint is a switch that determines the path from the processor to memory module.

→ Each switch point has a control logic to setup the transfer path between a processor and memory.

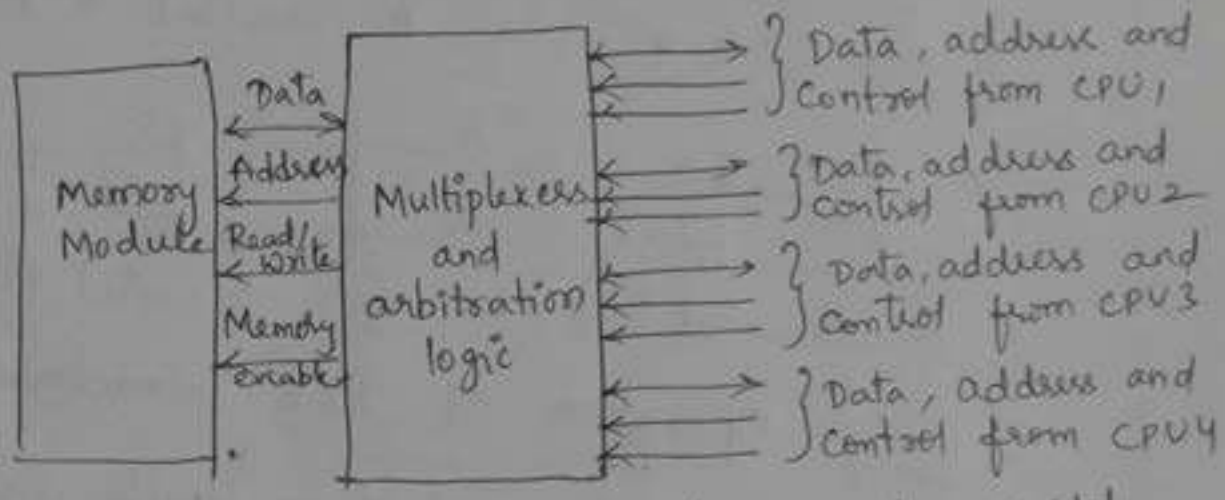


fig:- Block diagram of crossbar switch.

→ The circuit consists of multiplexers that select the data, address and control from one CPU for communication with memory module.

→ Priority levels are established by the arbitration logic to select one CPU. when two or more CPU attempt to access same memory.

→ The multiplexers are controlled with binary code that is generated by priority encoder within the arbitration logic.

Scanned by CamScanner

Multistage Switching Network :-

→ The basic component of a multistage network is a two-input, two-output interchange switch.

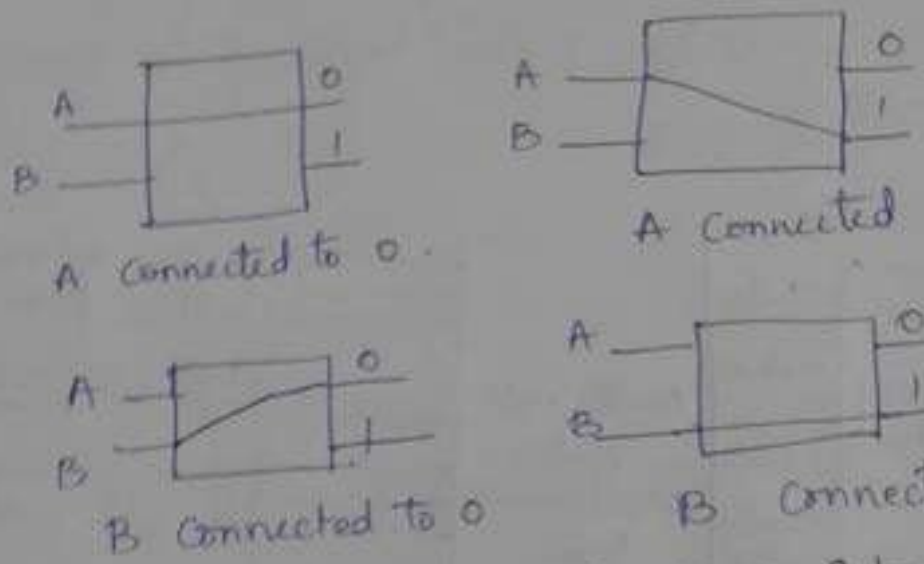


fig :- Operation of 2×2 interchange switch.

→ The 2×2 switch has two inputs labeled A and B, and two outputs, labeled 0 and 1.

→ There are control signals associated with the switch that establish the interconnection b/w the input and output terminals.

→ The switch has capability of connecting input A to either of o/p's.

→ Terminal B of the switch behaves is the similar manner.

→ The switch also has the capability to arbitrate b/w conflicting requests if i/p's A and B both request the same o/p terminal only one of them will be connected and other is blocked.

Scanned by CamScanner

→ 2x2 switch as a building block, it is possible to build a multistage network to control the communication b/w a number of sources and destinations.

→ To know how is done, we consider binary tree of 2x2 switches.

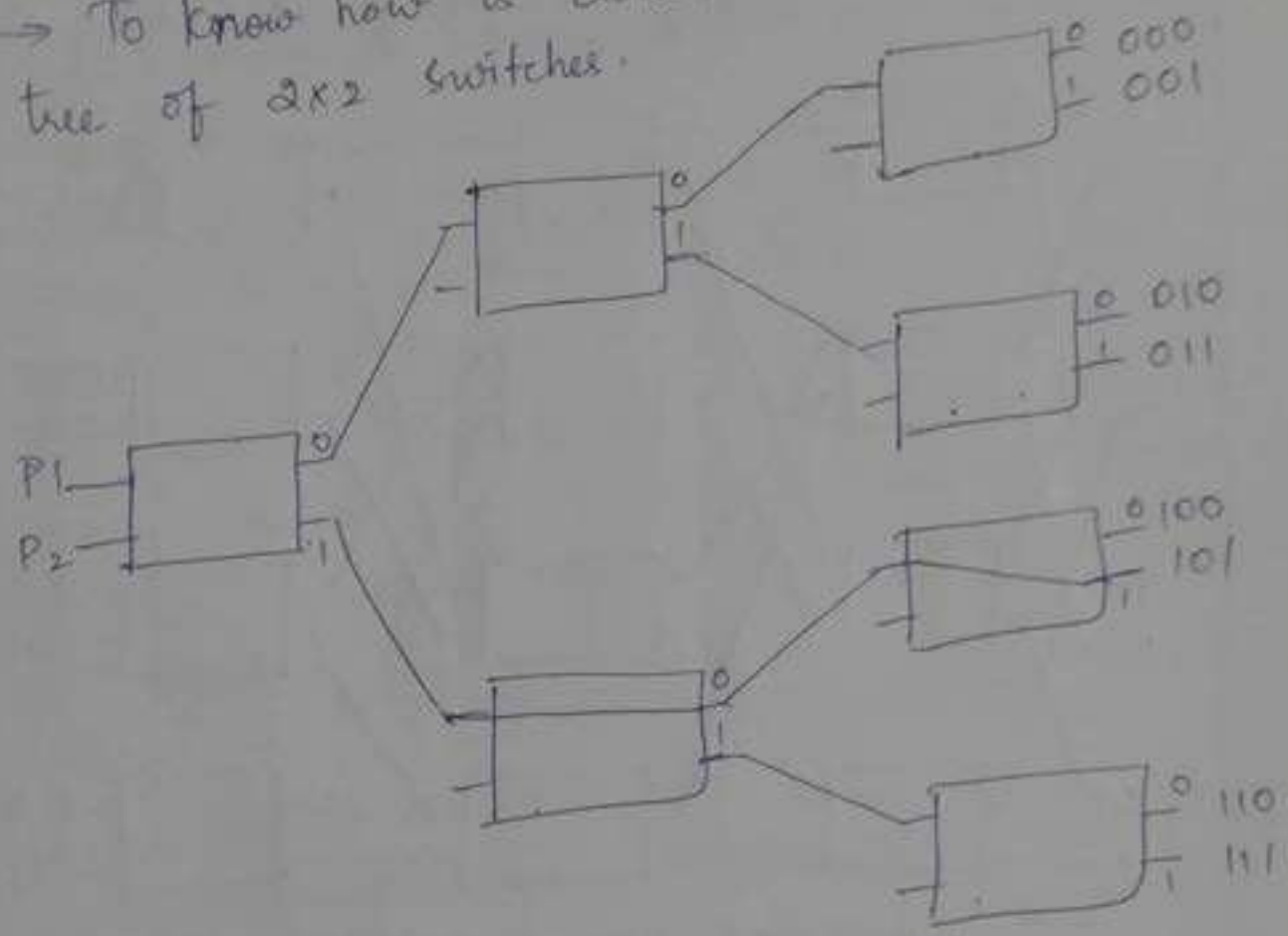


fig:- Binary tree with 2x2 switches.

→ The two processors P_1 and P_2 are connected through switches to eight memory modules marked in binary from 000 to 111. The path from a source to destination is determined from the binary bits of the destination number.

Many different topologies have been proposed for multistage switching networks to control processor-memory communication in a

tightly coupled multiprocessor system or control the communication b/w processing elements in loosely coupled system.

→ One such topology is 'omega switching Network'.

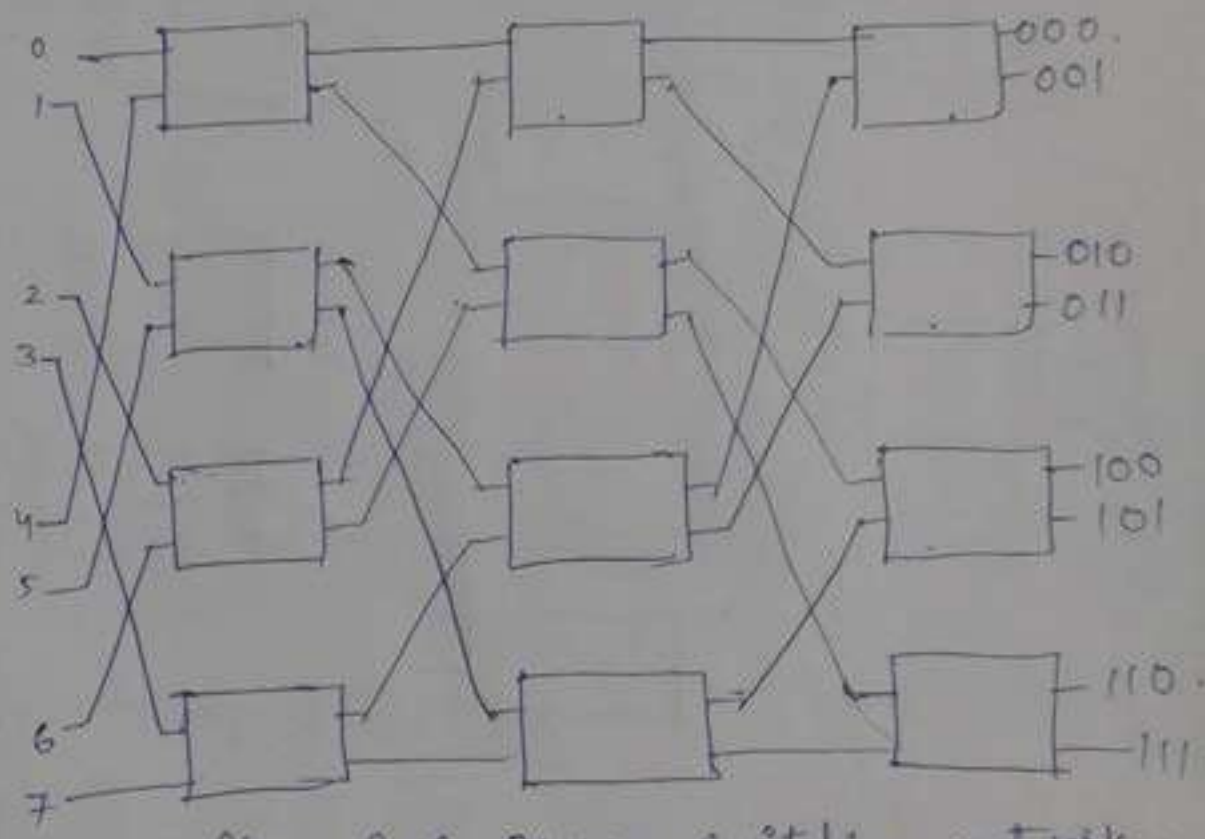


fig :- 8x8 Omega switching network.

A particular request is initiated in the switching network by the source, which sends a 3-bit pattern representing the destination number. As the binary pattern moves through the network, each level examines a different bit to determine the 2x2 switch setting.

In tightly coupled multiprocessor system, the source is a processor and destination is memory modules where as in loosely coupled, both source and destination are processing elements.

[illegible] the same o/p terminal only one [illegible]

Scanned by CamScanner

Hypercube Interconnection:-

→ The hypercube or binary n-cube multiprocessor structure is a loosely coupled system composed by $N = 2^n$ processors interconnected in a n-dimensional binary cube.

→ Each processor forms a 'node' of a cube. Although it is customary to refer to each node as having a processor, it contains not only a CPU but also local memory and I/O interface.

→ Each processor has direct communication paths to n other neighbour processor. These paths correspond to 'edges' of the cube.

→ There are 2^n distinct n-bit addresses that can be assigned to the processors.

→ Each processor address differs from that of each of 'n' neighbors by exactly one bit position.

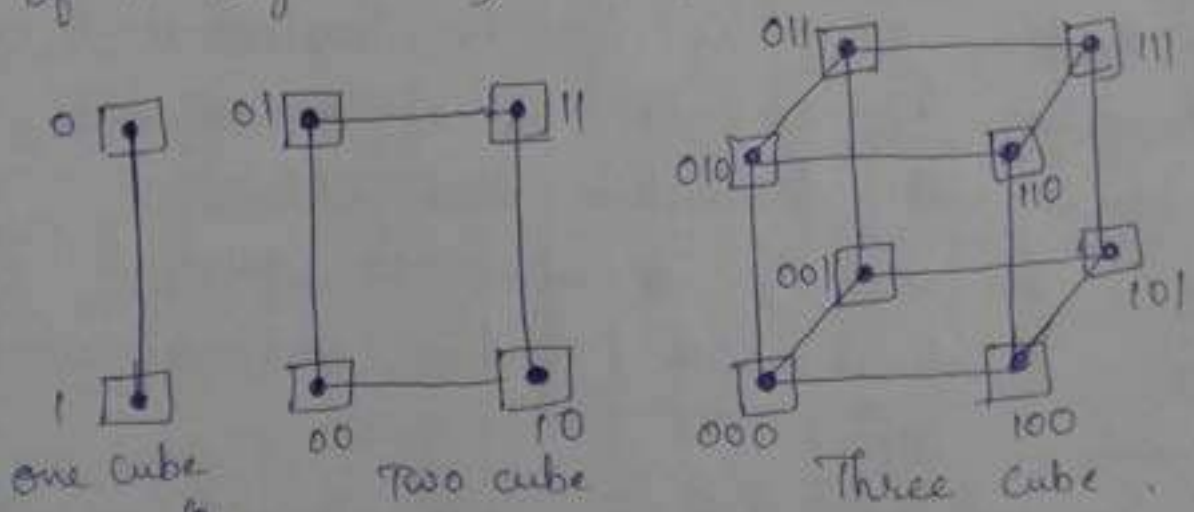


fig:- Hypercube structures for $n = 1, 2, 3$.

→ A one-cube structure has $n=1$ and $2^n = 2^1 = 2$.

→ It contains two processors interconnected by a single path.

→ A two cube structure has $n=2$ and $2^n = 2^2 = 4$. It has four nodes later connected as a square

→ A three cube structure has eight nodes ($n=3$, $N=2^3$, $N=8$) interconnected as a cube.

→ A n cube structure has 2^n nodes with a processor residing in each node.

→ Each node is assigned a binary address in such a way that addresses of two neighbors differ in exactly one bit position.

→ Routing messages through an n-cube structure may take from one to n links from source node to destination node.

→ Routing procedure can be implemented by the computing the exclusive-OR of source node address with destination node address.

Eg:- The message at 010 going to 001 produces XOR of two addresses and produces 011.

→ The message can be sent along the second axis to 000 and then through third axis to 001.

Scanned by CamScanner

Interprocessor Arbitration :-

A memory bus consists of lines for transferring the data, address & read/write information.

→ An I/O bus is used to transfer information to & from input & output devices.

→ A bus that connects major components in a multiprocessor system such as CPU's, IOP's & memory is called 'system bus'.

→ The processors may request the system bus at the same time. Arbitration must then be performed to resolve this multiple contention for the shared resources. The arbitration logic would be part of the system bus controller placed b/w local bus & the system bus.

System Bus :-

A typical system bus consists of approximately 100 sign lines. These lines are divided into three functional units i.e, data, address & control.

→ In addition to these there are power distribution lines that supply power to the components.

→ The data lines provide a path for the transfer of data b/w processors & common memory. The number of data lines is usually a multiple of 8 i.e, 16 & 32.

→ The address lines are used to identify a memory address or any other source or destination such as i/p or o/p ports.

Scanned by CamScanner

Data transfers over the system bus are divided into two types.

1) Synchronous bus.

2) Asynchronous bus.

In synchronous bus, each data item is transferred during a time slice known to both source and destination units. synchronization is achieved by driving both units from a common clock source. An alternative procedure is to have separate clocks of approximately the same frequency in each unit. The synchronization signals are transmitted periodically.

In Asynchronous bus, each data item being transferred is accompanied by handshaking control signals to indicate when the data are transferred from the source & received by the destination.

The control lines provide signals for controlling the information b/w units. The timing signals indicate validity of data & address information. The Command signals specify operations to be performed.

The six bus arbitration signals are used for interprocessor arbitration.

Scanned by CamScanner

Serial Arbitration Procedure:-

Arbitration procedures service all processors requests on the basis of established priorities.

→ A hardware bus priority resolving technique can be established by means of serial or parallel connection of units requesting control of system bus.

→ The serial priority resolving technique is obtained from a daisy chain connection of bus arbitration circuits.

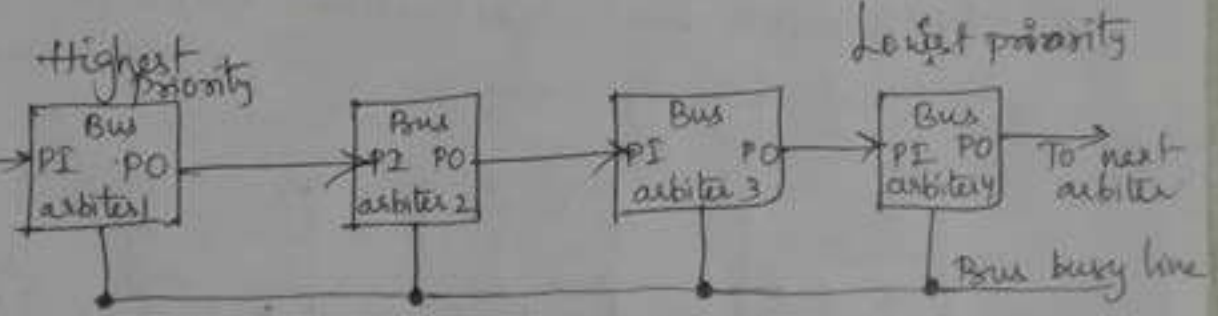


fig:- Serial (daisy chain) arbitration.

→ It is assumed that each processor has its own bus arbiter logic with priority-in and priority-out lines.

→ PO of each arbiter is connected to the priority-in (PI) of the next lower priority arbiter. The PI of highest priority unit is maintained at logic 1 values. The highest priority unit in the system will always receive access to the system bus when it requests it.

Scanned by CamScanner

→ The PO output for a particular arbiter is equal to 1 if its PI = 1 and processor associated with the arbiter logic is not requesting control of the bus

→ Processor requests control of the bus and the corresponding arbiter finds ~~the~~ its PI = 1 and it sets PO = 0.

→ Processor whose arbiter has PI = 1 and PO = 0 is the one that is given control of the system bus.

Parallel Arbitration Logic :-

→ The parallel bus arbitration technique uses an external priority encoder and a decoder

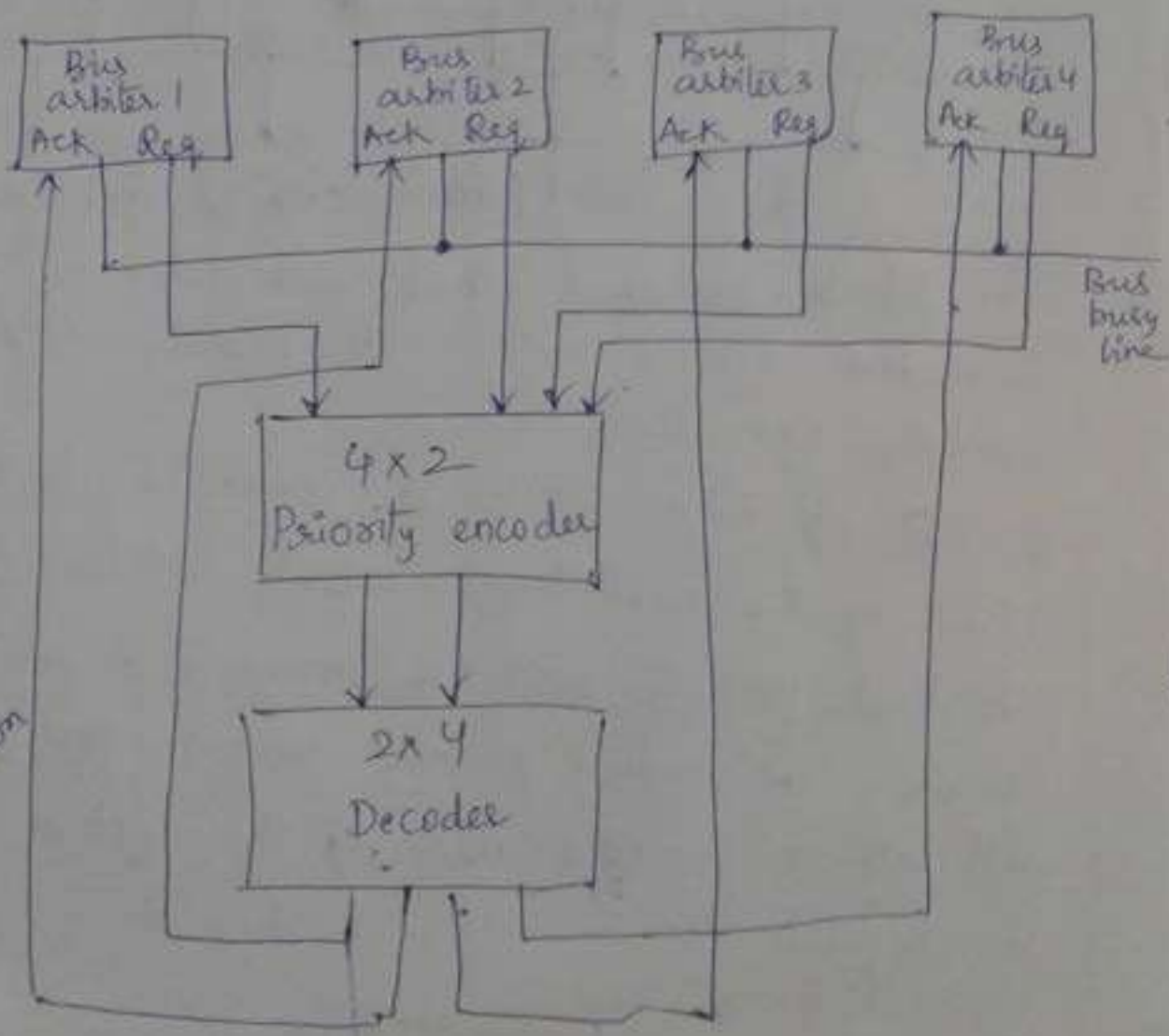


fig :- parallel arbitration

Scanned by CamScanner

→ Each bus arbiter in the parallel scheme has a bus request output line and bus acknowledge input line.
→ Each arbiter enables the request line when its processor is requesting access to the system bus. The processor takes control of the bus if its acknowledge input line is enabled. The bus busy line provides an orderly transfer of control in daisy chaining case.
→ The request lines from four arbiters going into a 4x2 priority encoder. The output of the encoder generates a 2-bit code which represents the highest-priority unit among those requesting the bus. The 2 bit code from encoder output drives a 2x4 decoder which enables the proper acknowledge line to grant bus access to the highest-priority unit

Dynamic Arbitration Algorithms :-

→ The two arbitration procedures just described uses a static priority algorithm since the priority of each device is fixed by the way it is connected to the bus.
→ In dynamic priority algorithm gives the system the capability for changing the priority of devices while the system is in operation.
→ Few arbitration procedures that use dynamic priority algorithms.

Scanned by CamScanner

Time Slice Algorithm:- The time slice algorithm allocates a fixed length time slice of bus time that is offered sequentially to each processor in a round-robin fashion.

→ The service given to each system is independent of its location along the bus.

Polling:-

In a bus system that uses polling, the bus grant signal is replaced by a set of lines called poll lines which are connected to all units. These lines are used by bus controller sequences through the addresses in a prescribed manner.

LRU (Least Recently Used):- This Algorithm gives the highest priority to the requesting device that has not used the bus for the longest interval. The priorities are adjusted after a number of bus cycles according to LRU algorithm.

FIFO (First Come, First Serve):- This scheme requests are served in the order received, the bus controller establishes a queue arranged according to the time that the bus requests arrive.

Rotating daisy-chain:- This procedure is dynamic extension of the daisy-chain algorithm. In this scheme there is no central bus controller, and the priority line is connected from priority-out of the last device back to the priority-in of the first device in closed loop.

Scanned by CamScanner

Interprocessor Communication & Synchronization:-

The various processors in a multiprocessor system must be provided with a facility for communicating with each other. A Communication path can be established through Common I/O channels. In shared memory, is to set aside a portion of memory that is accessible to all processors.

The primary use of Common memory is to act as a message center similar to a mailbox, where each processor can leave messages for other processors & pickup messages intended for it.

The sending processor structures a request, a message or a procedure & places it in the memory mailbox. Status bits residing in Common memory are generally used to indicate the condition of the mailbox, whether there is meaningful messages or not.

The receiving processor can check the ~~mailbo~~ mailbox, to determine if there are valid messages for it. The response lies if this procedure can be time Consuming.

An Efficient procedure is for the sending processor to alert the receiving processor directly

Scanned by CamScanner

by means of an Interrupt signal. This can be used through a s/w - initiated interprocessor interrupt by means of an instruction in the program of one processor which when executed produces an external Interrupt condition in a second processor

In addition to shared Memory, a multiprocessor system may have other shared resources. for eg, a magnetic disk storage unit connected to an IOP may be available to all CPU's. This provides a facility for sharing of system. A communication pass is also established.

To prevent conflicting use of shared ~~resources~~ resources by several processors there must be providing resources to each processors.

This task is given to the O.S

There are three different organizations, for design of O.S for multiprocessors.

1) Master-slave configuration.
2) Separate operating system.
3) Distributed operating system.

→ In a Master slave mode, one processor is treated as master which performs all

Scanned by CamScanner

O.S functions. The remaining processors denoted as slaves which not perform O.S functions. If slave processor needs an O.S service it must request it by interrupting the master & waiting until the current program can be interrupted.

In a separate O.S, each processor can execute the O.S routines it needs. This organization is suitable for loosely coupled systems where every processor may have its own copy of the entire O.S.

In a distributed O.S, the O.S routines are distributed among the available processors. Each particular O.S function is assigned to only one processor at a time and the execution may be assigned to different processors at different times. This most suitable for floating operating systems.

In a loosely coupled multiprocessor system the memory is distributed among the processors & there is no shared memory for passing information. The communication b/w processor is by means of message passing through I/O channels.

Scanned by CamScanner

A message is then sent with a header and various data objects used to communicate b/w nodes. There may be a no. of possible paths available to send the message b/w any two nodes. The O.S in each node contains routing information indicating the alternative paths that can be used to send a message to other nodes.

The communication efficiency of the interprocessor network depends on the communication routing protocol, processor speed data link speed and the topology of the network.

Interprocessor Synchronization:-

→ The instruction set of a multiprocessor contains basic instructions that are used to implement communication and synchronization b/w cooperating processes. Communication refers to the exchange of data b/w different processes.

→ Synchronization refers to the special case where the data used to communicate b/w processors is control information.

→ It is needed to enforce the correct sequence of processes and to ensure mutually exclusive access to shared writable data.

Scanned by CamScanner

Mutual Exclusion with a semaphore:-

Multiprocessor system must provide a mechanism that will gurantee orderly access to shared memory and other shared resources.

→ It is necessary to protect data from being changed simultaneously by two or more processors.

→ This mechanism is termed as 'mutual Exclusion'.

→ Mutual Exclusion must be provided in a multiprocessor system to enable one processor to exclude or lock out access to a shared resource by other processors when it is in 'critical section'.

→ A 'critical section' is a program sequence that, once begun, must complete execution before another processor accesses the same shared Resource.

Semaphore :-

A binary variable called a 'semaphore' is often used to indicate whether or not a processor is executing a critical section.

→ A semaphore is a Software Controlled flag that is stored in a memory location that all processors can access.

→ when semaphore = 1 then it means the program

Scanned by CamScanner

is executing a critical program, so shared memory is not available.

→ When semaphore = 0, the shared memory is available to any requesting processor.

→ Testing and setting the semaphore is itself a critical operation and must be performed as a single indivisible operation.

→ A semaphore is initialized by means of a test and set instruction in conjunction with a hardware 'lock' mechanism.

Scanned by CamScanner

Cache Coherence :-

We are using separate caches for each processor is to reduce the average access time in each processor. The same information may reside in the no. of copies in some caches and main memory. The ability of the system to execute memory operation correctly, the multiple copies must be kept identical. Here it arises a 'Cache coherence' problem.

A memory scheme is 'Coherent' if the value returned on a load instruction is always the value given by the latest store instruction with the same address. Without a proper solution to the cache coherence problem, caching can't be used in bus-oriented multiprocessors with two or more processors.

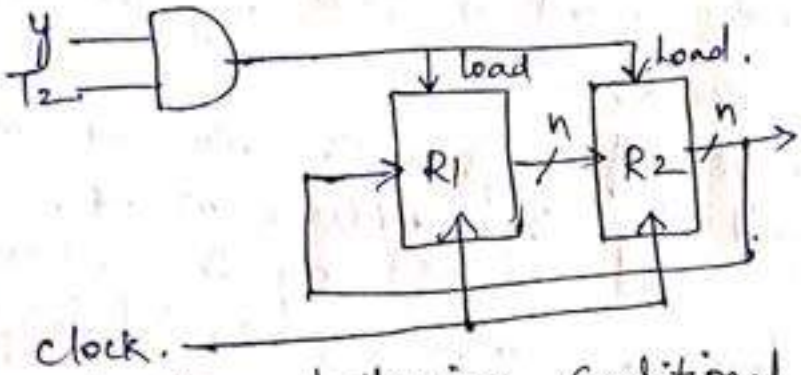
Scanned by CamScanner

UNIT-1

Q) Show the block diagram of the hardware that implements the following register transfer statement:

$$yT_2 : R2 \leftarrow R1 , R1 \leftarrow R2.$$

Sol:-



Q) Represent the following conditional control statement by two register transfer statements with control functions

If (P = 1) then (R1 ← R2) else if (Q = 1) then (R1 ← R3)

Sol:-

$P : R1 \leftarrow R2$

$P'Q : R1 \leftarrow R3.$

Q) A digital computer has a common bus system for 16 registers of 32 bits each. The bus is constructed with multiplexers.

(a) How many selection inputs are there in each multiplexer. $(16 = 2^4)$

Sol:- 4 selection lines to select one of 16 registers

(b) What size of multiplexers are needed

Sol. 16 x 1 multiplexer.

(c) How many multiplexers are there in the bus.

Sol:- 32 multiplexers, one for each bit of the register.

Scanned by CamScanner

Q) The following transfer statements specify a memory. Explain the memory operation in each case.

(a) $R2 \leftarrow M[AR]$ (b) $M[AR] \leftarrow R3$ (c) $R5 \leftarrow M[R5]$.

Ans:- (a) Read memory word specified by the address in AR into register R2.

(b) Write content of register R3 into the memory word specified by the address in AR.

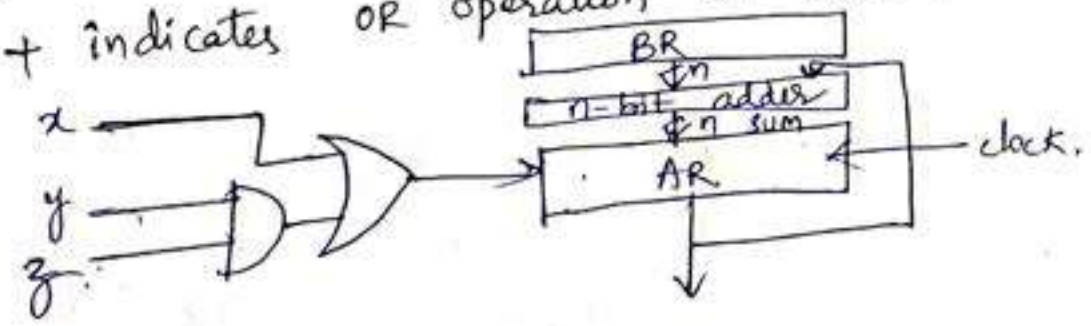
(c) Read memory word specified by the address in R5 and transfer content to R5 (destroys previous value).

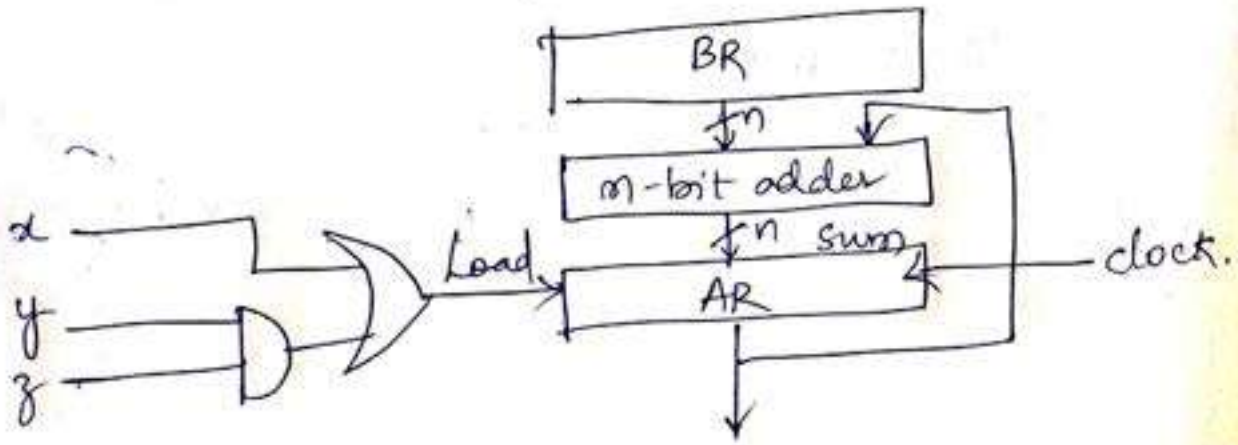
Q) Draw a block diagram for the hardware that implements the following statements:

$x + yz : AR \leftarrow AR + BR$

AR, BR are n-bit registers and x, y, z are control variables. Include logic gates for control function. + indicates OR operation in control or boolean function

Ans





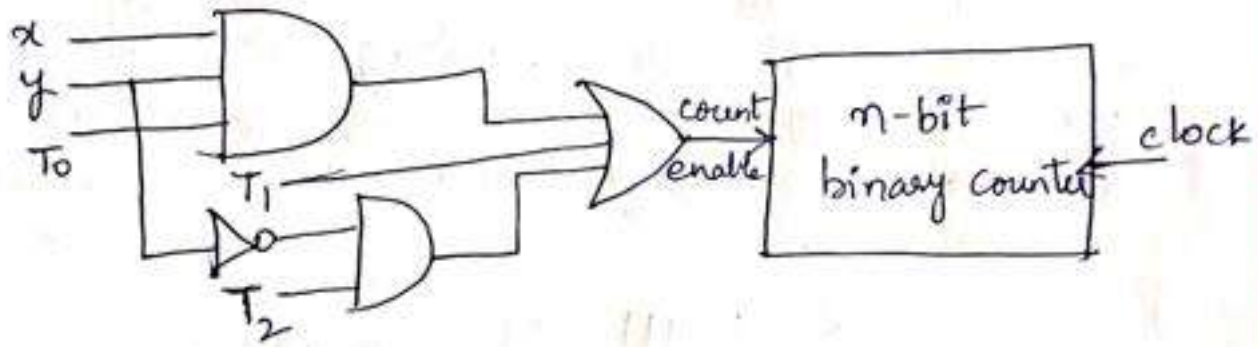
Scanned by CamScanner

Q) Consider the following register transfer statements for two 4-bit registers R1 and R2

$xT: \quad R1 \leftarrow R1 + R2$

$x'T: \quad R1 \leftarrow R2$

Sol:



Q) The 8-bit Registers AR, BR, CR and DR initially have the following values:

AR : 11110010
BR = 11111111
CR = 10111001
DR = 11101010

Determine the 8-bit values in each register after the execution of the following sequence of microoperation

$AR \leftarrow AR + BR$	Add BR to AR.
$CR \leftarrow CR \wedge DR,\ BR \leftarrow BR+1$	AND DR to CR, inc BR
$AR \leftarrow AR - CR$.	Subtract CR from AR.

Sol:

$AR = 11110010$
$BR = 11111111$
$AR = 11110001$

$CR \leftarrow CR \wedge DR$
$CR = 10111001$
AND $DR = 11101010$
$CR = 10101000$

$BR \leftarrow BR+1$
$BR = 11111111$
$+1$
$BR = 00000000$

Scanned by CamScanner

$AR \leftarrow AR - CR$.

$AR = 11110010$

$CR = 10111001$

$AR = 01001001$

Q) ~~The~~ An 8 bit register contains the binary value 10011100. What is the register value after the an arithmetic shift right? Starting from the initial number 10011100, determine the register value after an arithmetic shift left, and state whether there is an overflow.

$R = 1011100$.

Arithmetic shift Right = 10111 00 → 11011 10

In ashr the sign bit in MSB. is copied as it is after shifting.

Arithmetic shift left : 00111000.

overflow because a negative number changed to positive.

Q) Register A holds the 8-bit binary 11011001. Determine the B operand and the logic microoperation to be performed inorder to change the value in A to :

(a) 01101101 (b) 11111101

Ans

$A = 11011001$

XOR $B = 00110100$ (B? what operation with what bits)

01101101

$A \oplus B = 01101101$.

XOR operation with 00110100 gives 01101101

Scanned by CamScanner

Basic Organization And Design.

Q) A computer uses a memory unit with 256k words of 32 bits each. A binary instruction code is stored in one word of memory. The instruction has four parts: an indirect bit, an operation code, a register code part to specify one of 64 registers and an address part.

(a) How many bits are there in operation code, the register code part, and the address part.

(b) Draw the instruction word format and indicate the number of bits in each part.

(c) How many bits are there in data and address inputs of the memory?

Sol: Given :-

$256k \times 32$

$2^8 \times 2^{10} = 2^{18}$ → address bits

64 Register $= 2^6$

~~Total instruction bit = 32~~

(a) Address = 18 bits

Register Code = 6 bits

Indirect bit = 1 bit

25 bits

$32 - 25 = 7$ bits for opcode.

(b)

1	7	6	18
I	opcode	registers	Address

(c) Data : 32 bits , address : 18 bits

Scanned by CamScanner

(b) 11111101

A = 11011001
B = 00100101 (OR)
A∨B. 11111101

A = 11011001
B = 11111101 OR.
A∨B 11111101

Q) starting from an initial value of R = 11011101, determine the sequence of binary values in R after a logical shift-left, followed by a circular shift-right followed by logical shift-right and circular shift left

Sol:-

R = 1 1 0 1 1 1 0 1

logical shift left: 1 0 1 1 1 0 1 0 — Insert 0 at LSB

Circular shift right: 0 1 0 1 1 1 0 1

logical shift right: 0 0 1 0 1 1 1 0 — Insert 0 at MSB, because it is empty

circular shift left: 0 1 0 1 1 1 0 0

Scanned by CamScanner

Q) The Content of AC in the basic Computer is hexadecimal A937 and the initial value of E is 1. Determine the Contents of AC, E, PC, AR and IR in hexadecimal after the execution of the CLA instruction. Repeat 11 more times, starting from each one of the register-reference instructions. The initial value of PC is hexadecimal 021.

Sol:

	E	AC	PC	AR	IR
Initial	1	A937	021	—	—
CLA	1	0000	022	800	7800
CLE	0	A937	022	400	7400
CMA	1	56C8	022	200	7200
CME	0	A937	022	100	7100
CIR	1	D49B	022	080	7080
CIL	1	526F	022	040	7040
INC	1	A938	022	020	7020
SPA	1	A937	022	010	7010
SNA	1	A937	023	008	7008
SZA	1	A937	022	004	7004
SZE	1	A937	022	002	7002
HLT	1	A937	022	001	7001

Q) The Content of PC in the basic Computer is 3AF The Content of AC is 7EC3. The Content of memory at address 3AF is 932E. The Content of memory at address 32E is 09AC. The Content of memory at address 9AC is 8B9F

(a) what is the instruction that will be fetched and executed next?

Scanned by CamScanner

b) show the binary operation that will be performed in the AC when the instructions is executed.

c) Give the Contents of registers PC, AR, DR, AC and IR in hexadecimal and the values of E, I and the sequence Counter SC in binary at the end of the instruction cycle.

Sol: (a) $9 = (1001)_2$

$1\ \underbrace{001}_{ADD}$

$I = 1$

ADD I 32E

Memory

3AF	932E	
	09AC	32E
	8B9F	9AC

AC = 7EC3.

(b) AC = 7EC3

DR = 8B9F

F A 62

(26-16)(22-16)(18-16)

(c) PC = 3AF+1 = 3B0

AR = 9AC

DR = 8B9F

AC = 0A62

IR = 932E

E = 1

I = 1

SC = 0000

Rough work

A = 10
B = 11
C = 12
D = 13
E = 14
F = 15

26 − 16 = 10

12 + 14 = 21

21 − 15 = 6

22 − 16 = 6

12 + 14 = 26

26 − 16

Scanned by CamScanner

UNIT-II

(1) What is the difference between hardwired Control and microprogrammed Control.

Hardwired Control Unit	Microprogrammed Control Unit.
(1) The Control signals are generated by hardware using logic is called hardwired Control Unit	(1) The Control signals are generated by microprograms in the Control memory.
(2) RISC style of instruction.	(2) CISC style of instructions.
(3) modification is difficult as The Control Unit is hardwire.	(3) modifications are easy in case of microprogrammed CU as it will require change in the code only.
(4) It works well for simple instruction.	(4) It works well for Complex Instructions also.
(5) No Control Memory is required.	(5) Control memory is required to store microprogram
(6) High speed operation, expensive, relatively Complex	(6) Comparatively slow, Not that costly, systematic method for controlling microoperations
Eg :- Intel 8085, RISC CPU's.	Eg :- Intel 8080, CISC CPU's

Scanned by CamScanner

Q) What is the difference between a microprocessor and a microprogram? Is it possible to design a microprocessor without a microprogram? Are all microprogrammed computers also microprocessors.

Sol: A microprocessor is electronic component that is used by the computer to do its work.

→ It is CPU on a single integrated circuit chip.

Microprogram :- It is a program for a sequence of microoperations.

→ The CU of a microprocessor can be hardwired or microprogrammed, depending on the specific design.

→ A microprogrammed computer does not have to be a microprocessor.

Q) Define :-

Microoperation :- A microoperation is an elementary operation performed with the data stored in registers.

Microinstruction :- An instruction of a microprogram in control memory.

Microprogram :- A set of or sequence of micro-instructions

Microcode :- A set of or sequence of microinstructions.

Scanned by CamScanner

2-3

... of instruction is stored in

2-2

Q) The system uses a control memory of 1024 words of 32 bits each. The microinstruction has three fields as shown. The microoperation field has 16 bits.

(a) How many bits are there in branch address field and the select field.

(b) If there are 16 status bits in the system, how many bits of the branch logic are used to select a status bit?

(c) How many bits are left to select an input for the multiplexers?

Sol: Control memory = 1024×32 Microoperation field = 16 bits

$= 2^{10} \times 32$

(a)

6	10 bits	16 bits	= 32 bits
Select	Address.	Microoperation	

(b) 4 bits

(c) 2 bits.

Q) The Control Memory has 4096 words of 24 bits each

(a) How many bits are there in CAR

(b) How many bits are there in each of four inputs, shown going into the multiplexers.

(c) What are the number of inputs in each multiplexer and how many multiplexers are needed?

Sol: Control Memory = $2^{12} \times 24$

(a) 12 bits

(b) 12 bits

(c) 12 multiplexers, each of size 4 to-1 line.

Scanned by CamScanner

Q) Use the mapping procedure, give the first microinstruction address for the following operation code:
(a) 0010; (b) 1011 ; (c) 1111.

Sol. a) 0001000 = 8
b) 0101100 = 44
c) 0111100 = 60

opcode

0010.	address

Mapping 0 | xxxx | 00

0 0010 00

Same for

0 1011 00

0 1111 00

Q) Formulate a mapping procedure that provides eight consecutive microinstructions for each routine. The operation code has 6 bits and control memory has 2048 words.

Sol:- Given :-
Opcode = 6 bits , Control Memory = 2048 words
~~address~~ = 2048
$= 2^{11}$
address bits = 11 bits.

00 | xxxxxx | 000

Q) Convert the following symbolic microoperations to register transfer statements and to binary.

Sol: (a) READ, INCPC

READ DR ← M[AR] F2 = 100.
INCPC ~~DRTAC~~ PC ← PC+1 F3 = 101

(b) ACTDR, DRTAC.

ACTDR DR ← AC F2 = 101
DRTAC AC ← DR. F1 = 100

(c) ARTPC, DRTAC, WRITE

ARTPC PC ← AR F3 = 110
DRTAC AC ← DR F1 = 100
WRITE M[AR] ← DR F1 = 111

Scanned by CamScanner

Q) A relative mode branch type of instruction is stored in memory at an address equivalent to decimal 750. The branch is made to an address equivalent to decimal 500.

(a) What should be the value of the relative address field of the instruction (in decimal)?

(b) Determine the relative address value in binary using 12 bits. (Why must the numbers be in 2's Complement).

(c) Determine the binary value in PC after the fetch phase and calculate the binary value of 500. Then show that the binary value in PC plus the relative address calculated in part (b) is equal to binary value of 500.

Sol: a) Relative address = 500 − 751 = −251

b) 251 = 000011111011 ; −251 = 111100000101

c) PC = 751 = 001011101111 ; 500 = 000111110100

PC = 751 = 001011101111
RA = −251 = + 111100000101
EA = 500 = 000111110100.

Q) An instruction is stored at location 300 with its address field at location 301. The address field has the value 400. A processor register R1 contains the number 200. Evaluate the effective address if the addressing mode of the instruction is

(a) direct (b) immediate ; (c) relative; (d) register indirect; (e) index with R1 as the index register.

Sol: Effective address

(a) Direct : 400
(b) Immediate : 301
(c) Relative : 302 + 400 = 702
(d) Register indirect = 200.
(e) Indexed : 200 + 400 = 600.

Memory.

PC→300		Opcode Mode
R1→200	301	400
	302	Next inst.

Scanned by CamScanner

Q) (a) Write a program to evaluate the arithmetic statement.

Q) The memory unit of a computer has 256K words of 32 bit each. The computer has an instruction format with four fields: an operation code field, a mode field to specify one of seven addressing modes, a register address field to specify one of 60 processor registers, and a memory address. Specify the instruction format and the number of bits in each field if the in instruction is in one memory word.

Sol: Given 256K x 32 bit each. $256K = 2^8 \times 2^{10} = 2^{18}$ → address bits

Sol: Q)

address bits = 18 bits

Mode = 3 bits (Given 7 address modes.) 0-7 = 000 - 111

Register = 6 bits (60 processor registers) $2^6 = 64 \cong 60$)

27 bits

opcode +5

32 bits

Opcode	Mode	Register	Address
5	3	6	18 = 32

Sol: Q) The two instruction is stored in memory at an address designated by the symbol W. The address field of the instruction (stored at W+1) is designated by the symbol Y. The operand used during the execution of the instruction is stored at address symbolized by Z. An index register contains the value X. State how Z is calculated from the other addresses if the addressing mode of the instruction is

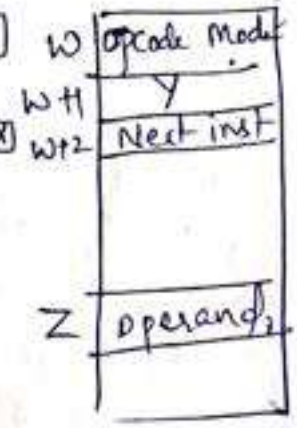
Z = Effective address

(a) direct addressing mode :- Z = Y.

(b) Indirect addressing mode : Z = M[Y]

(c) Relative : Z = Y + W + Z

(d) Indexed :- Z = Y + X



Scanned by CamScanner

Q) Perform the logic AND, OR and XOR with the two binary strings 10011100 and 10101010.

Sol:

```
10011100
10101010   AND
--------
10001000
```

```
10011100
10101010   OR.
--------
10111110
```

```
10011100
10101010   XOR
--------
00110110
```

NOTE :- DO NOT PERFORM ADDITION.

Q) Given the 16 bit value 1001101011001101. What operation must be performed inorder to :

a) clear to 0 the first eight bits ?

b) Set to 1 the last eight bits ?

c) Complement the middle eight bits ?.

Sol) (a) Inorder to clear the first 8 bits to 0, we need to perform AND operation with 0000000011111111

```
10011010 | 11001101.
00000000 | 11111111
--------------------
00000000 | 11001101
```

(b) set to 1 the last eight bits.

we need to perform OR operation with 0000000011111111

```
10011010 11001101
00000000 11111111
--------------------
10011010 11111111
```

(c) Inorder to complement the middle eight bits.

we need to perform XOR with 0000111111110000

Scanned by CamScanner

$$\begin{array}{rl} & 10011010 \quad 11001101 \\ \text{XOR} \oplus & 00001111 \quad 11110000 \\ \hline & 10010101 \quad 00111101 \end{array}$$

Complemented

Q) An 8 bit register contains the value 01111011 and the carry bit is equal to 1. Perform the eight shift operations. Each time, start from the initial value given above.

Sol:-

Initial : 01111011 C=1

SHR : 00111101

SHL : 11110110.

SHRA : 00111101

SHLA : 11110110 (overflow)

ROR : 10111101

ROL : 11110110.

RORC : 10111101

ROLC : 11110111

Q) The program in a computer compares two signed numbers A and B by performing the subtraction A-B and updating status bits. Let A = 01000001 and B = 10000100.

(a) Evaluate the difference and interpret the binary result

$$\begin{array}{rl} A = & 01000001 = +65 \\ B = & 10000100 = -124 \\ \hline A-B = & 10111101 \quad +189. \end{array} = \begin{array}{c} + \\ 0 \\ + \end{array} \underbrace{10111101}_{189} = 9\text{bits}$$

(b) Determine the value of S, Z, and V

S = 1 (sign reversal), Z = 0, V = 1 (overflow)

+189 > 127. (because the result is not 0's)

(c) List the conditional branch instructions that will have a true condition : BGT, BGE, BNE.

Scanned by CamScanner

UNIT - 3

Part-1 : Data Representation.

1) Convert the following binary numbers to decimal:
101110 ; 1110101 ; and 110110100.

Sol: $(101110)_2 = 1\times2^5 + 0\times2^4 + 1\times2^3 + 1\times2^2 + 1\times2^1 + 0\times2^0$

$= 32 + 0 + 8 + 4 + 2 + 0 = 46.$

$(1110101)_2 = 1\times2^6 + 1\times2^5 + 1\times2^4 + 0\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0$

$= 64 + 32 + 16 + 0 + 4 + 0 + 1 = 117.$

$(110110100)_2 = 1\times2^8 + 1\times2^7 + 0\times2^6 + 1\times2^5 + 1\times2^4 + 0\times2^3 + 1\times2^2 + 0\times2^1 + 0\times2^0$

$= 256 + 128 + 0 + 32 + 16 + 0 + 4 + 0 + 0$

$= 436.$

2) Convert the following numbers with the indicated bases to decimal:

Sol: $(12121)_3 = 1\times3^4 + 2\times3^3 + 1\times3^2 + 2\times3^1 + 1\times3^0$

$= 81 + 54 + 9 + 6 + 1 = 151.$

$(4310)_5 = 4\times5^3 + 3\times5^2 + 1\times5^1 + 0\times5^0.$

$= 500 + 75 + 5 = 580.$

$(50)_7 = 5\times7^1 + 0\times7^0 = 35$

$(198)_{12} = 1\times12^2 + 9\times12^1 + 8\times12^0 = 144 + 108 + 8 = 260$

HINT :- $(xyzxyz)_{\text{any base}}$ to decimal.

each and every digit of given number is multiplied with that same number base raised with the powers from 0 to n.

Scanned by CamScanner

3(a) Convert the following decimal numbers to binary.

Sol

2 | 1231	2 | 673	2 | 1998
2 | 615 - 1	2 | 336 - 1	2 | 999 - 0
2 | 307 - 1	2 | 168 - 0	2 | 499 - 1
2 | 153 - 1	2 | 84 - 0	2 | 249 - 1
2 | 76 - 1	2 | 42 - 0	2 | 124 - 1
2 | 38 - 0	2 | 21 - 0	2 | 62 - 0
2 | 19 - 0	2 | 10 - 1	2 | 31 - 0
2 | 9 - 1	2 | 5 - 0	2 | 15 - 1
2 | 4 - 1	2 | 2 - 1	2 | 7 - 1
2 | 2 - 0	1 - 0	2 | 3 - 1
1 - 0.		1 - 1
$(10011001111)_2$	$(1010100001)_2$.	$(11111001110)_2$

(or)

$1231 = 1024 + 128 + 64 + 15 = 2^{10} + 2^7 + 2^6 + 2^3 + 2^2 + 2 + 1$

$(10011001111)_2$

$673_{10} = 512 + 128 + 32 + 1 = 2^9 + 2^7 + 2^5 + 1$

$= (1010100001)_2$

$(1998)_{10} = 1024 + 512 + 256 + 128 + 64 + 8 + 4 + 2$

$= 2^{10} + 2^9 + 2^8 + 2^7 + 2^6 + 2^3 + 2^2 + 2^1 . (11111001110)_2$

---

4) Convert the following decimal numbers to the bases indicated.

a) $(7562)_{10}$ to Octal

8 | 7562
8 | 945 - 2
8 | 118 - 1
8 | 14 - 6
1 - 6

$(16612)_8$.

b) $(1938)_{10} = (\quad)_{16}$

16 | 1938
16 | 121 - 2
16 | 7 - 9

$(792)_{16}$.

Scanned by CamScanner

(c) $(175)_{10} = (\ ?\ )_2$

$$\begin{array}{r|l} 2 & 175 \\ 2 & 87-1 \\ 2 & 43-1 \\ 2 & 21-1 \\ 2 & 10-1 \\ 2 & 5-0 \\ 2 & 2-1 \\ & 1-0 \end{array} \qquad (10101111)_2$$

Q) Convert the hexadecimal number F3A7C2 to binary and Octal.

Sol:- Hexadecimal number = F3A7C2

Write binary code for each digit of hexadecimal number.

F 3 A 7 C 2

$(1111\ 0011\ 1010\ 0111\ 1100\ 0010)_2$

$(11110011101001111000010)_2 - (\ ?\ )_8$.

Now divide the converted binary number to 3 digits as a parts and then convert to octal.

$\overline{11}\,\overline{110}\,\overline{011}\,\overline{101}\,\overline{001}\,\overline{111}\,\overline{000}\,\overline{010}$

$(7\ 4\ 7\ 2\ 3\ 7\ 0\ 2)_8 = (74723702)_8$

Scanned by CamScanner

Q) What is the radix of the numbers if the solution to the quadratic equation '$x^2 - 10x + 31 = 0$' is $x = 5, x = 8$?

Sol:

$$(x^2 - 10x + 31)_r = ((x-5)(x-8))_{10}.$$

$$(x^2 - 10x + 31)_r = x^2 - (5+8)_{10}x + (40)_{10}.$$

$10_r = (13)_{10}$.

$1 \times r^1 + 0 \times r^0 = 13$

$r + 0 = 13$

$\boxed{r = 13}$

Illy $(31)_r = (40)_{10}$

$3 \times r^1 + 1 \times r^0 = 40.$

$3r + 1 = 40$

$3r = 39$

$r = 39/3 = 13$

$\boxed{r = 13}$

Q) Represent decimal 215 in (12 bit register)
(a) binary (b) binary-coded octal (c) in BCD
d) Binary coded hexadecimal.

Sol:- a) $(215)_{10} - (\ )_2$

$$\begin{array}{r|l} 2 & 215 \\ 2 & 107 - 1 \\ 2 & 53 - 1 \\ 2 & 26 - 1 \\ 2 & 13 - 0 \\ 2 & 6 - 1 \\ 2 & 3 - 0 \\ & 1 - 1 \end{array}$$

$(11010111)_2$

000011010111 - in 12 bits.

b) Binary-coded Octal

$\underbrace{000}_{0}\ \underbrace{011}_{3}\ \underbrace{010}_{2}\ \underbrace{111}_{7} = 327.$

Scanned by CamScanner

ⓒ In BCD.

215 — 0010 0001 0101 - 0010 0001 0101

d) In binary-coded hexadecimal.

$(215)_{10}$ — ($)_2$ — ($)_{16}$.

000011 010111

0000 1101 0111

0 13 7 $(D7)_{16}$.

Q) Obtain the 9's Complement of following eight-digit decimal number

ⓐ 12349876 ⓑ 00980100.

9's complement :- Subtracting each given number digit with 9

```
 99999999
-12349876
 87650123
```

```
 99999999
-00980100
 99019899
```

ⓒ 90009951

```
 99999999
-90009951
 09990048
```

ⓓ 00000000

```
 99999999
-00000000
 99999999
```

Q) Obtain the 10's Complement of the following six-digit decimal numbers

ⓐ 10's Complement can be obtained by performing the 9's Complement to the given number and adding 1 to Least significant Bit.

10's Complement = 9's Complement of given number + 1

Scanned by CamScanner

(a) 123900

9's

999999
− 123900
876099

10's :- 876099
+1
876100

(b) 090657

9's complement

999999
− 090657
909342

10's complement

909342
+1
909343

(c) 100000

9's complement

999999
− 100000
899999

10's

899999
+1
900000

(d) 000000

9's complement

999999
− 000000
999999

10's complement is

999999
+1
1 000000

2) Obtain the 1's and 2's Complement of the following eight-digit binary numbers.

(a) 10101110.

1's Complement

10101110
01010001

2's Complement.

01010001
+1
01010010

	Sum	c
0+0	0	0
0+1	1	0
1+0	1	0
1+1	0	1

(b) 10000001

1's Complement = 01111110

2's Complement = 01111110
+1
01111111

Scanned by CamScanner

(c) 10000000

1's Complement =

01111111

2's Complement ;

01111111
+1
10000000

(d) 00000001

1's complement of 00000001 is

11111110

2's Complement

11111110
+1
11111111

(e) 00000000

1's Complement

11111111

2's Complement =

11111111
+1
100000000

Q) Perform the subtraction with the following unsigned decimal numbers taking the 10's Complement of subtrahend.

(a) 5250 − 1321 (b) 1753 − 8640 (c) 20 − 100 (d) 1200 − 250.

Sol:- Subtraction for Unsigned Numbers :-

A − Minuend
− B − Subtrahend
difference

→ The subtraction of two n-digit unsigned number M-N in base 'r' can be done as follows:

1) Add minuend (M) to the r's Complement of 'N' (i.e., $M + (r^n - N) = M - N + r^n$.)

2) If $M \geq N$, the sum will produce an end carry r^n which is discarded; the left over result is M-N.

3) If $M < N$, the sum does not produce an end carry and equal to $r^n - (N-M)$ which is r's Complement of N-M. To obtain correct result do the r's complement of sum and place a −ve sign in front.

Scanned by CamScanner

(a) 5250 − 1321

Subtrahend = 5250.

Minuend = 1321

10's complement of Minuend

$$\begin{array}{r} 9999 \\ -1321 \\ \hline 8678 \\ +1 \\ \hline 8679 \end{array}$$ — 10's Complement

Now Add 10's complement of Minuend to Subtrahend (5250)

$$\begin{array}{r} 5250 \\ 8679 \\ \hline 13929 \end{array}$$

End carry is produced.

So we can say that 5250 is greater than 1321 (M ≥ N)

∴ Discard the carry and the sum is $\boxed{3929}$

(b) 1753 − 8640

Subtrahend = 1753

Minuend = 8640.

10's complement of minuend is (9's compl +1)

$$\begin{array}{r} 9999 \\ -8640 \\ \hline 1359 \\ +1 \\ \hline 1360 \end{array}$$ — 10's Comp.

Now Add 10's comple- of Minuend to Subtra-hend

$$\begin{array}{r} 1753 \\ +1360 \\ \hline 3113 \end{array}$$

End carry is not produced

So M < N.

→ To get the Correct result do the 10's Complement of 3113 and put −ve in front of result

i.e.,

$$\begin{array}{r} 9999 \\ 3113 \\ \hline 6886 \\ +1 \\ \hline -6887 \end{array}$$

Scanned by CamScanner

(c) 20 - 100 (first Equalize the no. of digits)

10's complement of 100.

999 - 9's complem
-100
899
+1 - 10's comp.
900

Now Add 900 to 20.

020
900
920

No End carry produced so, M<N. To get Correct result perform 10's complement to the result

999 - 9's Comp.
-920
079
+1 - 10's Comp.
-78

10's Comp 920 = -78

(d) 1200 - 250

Equalize the no. of digits of ~~both~~ subtrahend or minuend.

So, here 1200 has 4 digits 250 has 3 digits.

→ So, it will be

1200 - 0250

Now take 10's complement of 0250 i.e,

9999 - 9's Comp
-0250
9749
+1
9750 - 10's Comp.

Now Add 1200 to 9750.

1200
+9750
10950

End carry has generated so M>N; discarde the carry so

Ans ~~if~~ $\boxed{1200-250}$ using 10's Comp of & Minuend is $\boxed{950}$

Scanned by CamScanner

Perform the subtraction with the following unsigned binary numbers by taking the 2's Complement of the subtrahend.

(a) 11010 – 10000

(b) 11010 – 1101

(c) 100 – 110000

(d) 1010100 – 1010100.

For all above mentioned problems.

first step:- check the no. of digits in both subtrahend & Minuend. do the 2's Complement of subtrahend. and then add to minuend.

second step:- After adding the Minuend + 2's Comp of subtrahend, if a carry is generated then $M \geq N$, else $M < N$ the solution is –ve and it and is in 2's complement. To get Original result do the 2's Complement solution that will be (negative no. – number)

(a) $\boxed{11010 - 10000.}$

2's Complement of 10000.

1's Complement of 10000 is 01111

2's Complement is

$$\begin{array}{r} 01111 \\ +1 \\ \hline 10000 \\ \hline \end{array}$$

Now add 10000 to 11010

$$\begin{array}{r} 10000 \\ 11010 \\ \hline 1\ 01010 \end{array}$$

→ Carry generated

Carry generated that means subtrahend is greater than Minuend ($M \geq N$).

Now discard the Carry and the result will be.

$\boxed{01010}$

Scanned by CamScanner

Q) Show the contents of registers E, A, Q and SC

3-6

(b) 11010 – 1101
(5 digits) (4 digits)

11010 – 01101
(add 0 to make it 5 digits in MSB)

11010 – 01101

2's comp of 01101

01101
1's comp 10010
+1
2's comp 10011

Add 10011 to 11010

11010
+10011
1 01101

End carry generated

so M (11010) > N (01101)

so Discard carry

$\boxed{\text{Result} = 01101}$

(c) 100 – 110000
(3 digits (6 digits)

[Add 3 0's to 100 to make it 6 digits in MSB position]

000100 – 110000

2's Comp of 110000 :-

1's comp 001111
2's comp 001111
+1
010000

Add 010000 to 000100 (Minuend)

000100
+010000
010100

End carry not generated so

M < N

That means the result is in 2's comp because it is -ve.

To get original result do perform 2's comp of result we got.

010100
101011
+1
101100

010100 = –101100.

Scanned by CamScanner

(d) 1010100 - 1010100.

2's comp of 1010100 (subtrahend)

$\boxed{\text{1's Comp} + 1 = \text{2's Comp.}}$

```
  1010100.
  0101011 ← 1's comp.
        1+1 ← 2's comp.
  0101100
```

Now add 0101100 to 1010100.

```
  1010100
  0101100
1 0000000
```

End carry generated so, it is $M \geq N$.

Result = 0000000.

NOTE :- Negative numbers are always Represented in 2's Complement.

Eg :- 4 - 5, Here 5 is -ve, so it should be represented in 2's Complement.

Scanned by CamScanner

Q) Perform the arithmetic operations $(+42)+(-13)$ and $(-42)-(-13)$ in binary using signed-2's complement representation for negative numbers.

Sol:- $(+42)+(-13)$ $\qquad$ $(-42)-(-13)$

$(32+8+2)$ $\quad$ 32 16 8 4 2 1

$+42 = \boxed{101010}$

-42 = 2's comp of 101010.

$$\begin{array}{r} 010101 \\ +1 \\ \hline -42 \quad 010110 \end{array}$$

$(8+4+1)$ $\quad$ 32 16 8 4 2 1

$+13 = 001101$

-13 = 2's Comp. of 001101

$$\begin{array}{r} 110010 \\ +1 \\ \hline 110011 \end{array}$$

NOTE :- In this question i took 6 digits to represent the no. You may also take 8 digits. If they mention then consider those no. of digits to construct the given number in binary.

REMEMBER :- Always Remember (-ve) numbers are represented in 2'S COMPLEMENT.

(a) $+42 = 101010$

$+(-13) = 110011$

$29 \quad 1\,011101$ (16 8 4 1)

Carry generated i.e., $M>N$ $(42>-13)$ discard the carry.

$\boxed{\text{Result} = 011101}$

(b) $-42-(-13)$ $\quad$ $(-)\times(-)=(+)$

$(-42) = 010110$

$(+13) = 001101$

$-29 \quad 100011$

End carry not generated because $-42<13$ $(M<N)$ the Result will be -ve i.e., it will be in 2's complement.

→ To get back the -ve no do the 2's comp to 100011

$\Rightarrow 100011 = -(011101)$

Scanned by CamScanner

Q) Perform the arithmetic operations (+70) + (+80) and (−70) + (−80) with binary numbers in signed 2's Complement representation. Use 8 bits to accomodate each number together with its sign. Show that overflow occurs in both cases, that the last two carries are unequal, and that there is a sign reversal.

Sol: Overflow :- when two numbers of 'n' digits each are added and the sum occupies '(n+1)' digits so we can say that overflow occured.

→ During unsigned number addition, overflow is detected from end carry out of MSB bit

Signed numbers :- MSB or left most bit always represents the sign, the overflow occurs b/w the numbers whose sign's are Equal.

Given :-

8 bits to represent The given number.

128 64 32 16 8 4 2 1

+70 = 0 1 0 0 0 1 1 0
(sign bit)

128 64 32 16 8 4 2 1

+80 = 0 1 0 1 0 0 0 0
(sign bit)

−70 =
2's comp = 1 0 1 1 1 0 0 1
+1
1 0 1 1 1 0 1 0
(sign bit)

−80 =
2's comp (because −80)
1 0 1 0 1 1 1 1
+1
1 0 1 1 0 0 0 0
(sign bit)

Scanned by CamScanner

SIGN BIT.

$$
\begin{array}{rcl}
+70 & = & \overset{\textcircled{0}\,\textcircled{1}\,0\,0\,0\,0\,0\,0}{0\,1\,0\,0\,0\,1\,1\,0} \leftarrow \text{carry} \\
+80 & = & 0\,1\,0\,1\,0\,0\,0\,0 \\
\hline
+150 & & 1\,1\,0\,0\,1\,0\,1\,1\,0
\end{array}
$$

Observe carry into the sign bit (1) & carry out of the sign bit (0) } If both are unequal or not equal then overflow occured

If both carries are applied to EX-OR Gate the output will be (1)

A	B	A ⊕ B
0	0	0
0	1	1
1	0	1
1	1	0

SIGN BIT

$$
\begin{array}{rcl}
-70 & - & \overset{\textcircled{1}\,\textcircled{0}\,1\,1\,0\,0\,0\,0\,0}{1\,0\,1\,1\,1\,0\,1\,0} \text{ carry} \\
-80 & = & 1\,0\,1\,1\,0\,0\,0\,0 \\
\hline
-150 & & 0\,1\,0\,1\,0\,1\,0
\end{array}
$$

Carry in to SIGN bit 0 } not equal so overflow
Carry out of SIGN bit 1 } occurred.

EXOR of 0 and 1 is '1'

~~Ans of sign +70, +80 is = 1 0010110~~.

Q) Perform the following arithmetic operations with the decimal numbers using signed - 10's Complement Representation for negative numbers.

(a) $(-638) + (+785)$

(b) $(-638) - (+185)$

Scanned by CamScanner

Sol: a) $-638 + (+785)$

-638 is -ve number so it should be converted to 10's complement of given no.

So.

$$\begin{array}{r} 999 \leftarrow \text{9's comp} \\ 638 \\ \hline 361 \\ +1 \leftarrow \text{10's comp} \\ \hline 362 \end{array}$$

Add 362 to 785 $\Rightarrow$

$$\begin{array}{r} 9362 \\ 0785 \\ \hline 0147 \end{array}$$

$$\begin{array}{r} 9362 \\ +0785 \\ \hline 0147 \end{array}$$

Discard carry, so Result is 147

b) $-638 - (+185)$

$$\begin{array}{rrrrr} & (9's) & (10's) & & \\ -638 = & 999 & 361 & = & 9362 \\ & -638 & +1 & & \\ & 361 & 362 & & \\ -185 = & 999 & 814 & - & (+)\,815 \\ & 185 & +1 & & \\ & 814 & 815 & & 9177 \\ -823 & & & & \end{array}$$

$-823 = 9177$

NOTE :- In decimal numbers -ve ~~sign~~ numbers is represented using 9 and +ve number with 0

Q) Q) A 36-bit floating point binary number has 8 bits plus sign for exponent and 26 bits plus sign bit for mantissa. The mantissa is a normalized fraction. Numbers in mantissa and exponent are in signed magnitude representation. What are largest and smallest positive numbers that can be represented excluding 0?

Sol:- $m \times r^{e}$ . . $(-/+)m$ = mantissa

m = 26 bits + sign bit . $(-/+)e$ = exponent

e = 8 bits + sign bit

Mantissa

	26 bits

Exponent

S	8 bits

$26 + 8 = 36$ bits

Largest:- $+0.1111\ldots1$ Largest $+1.1111111$

$1 - 2^{-26}$ $+255$

The Largest Number $\underbrace{(1-2^{-26})}_{m} \times 2^{+55 \to e}$.

Smallest (normalized) : $+0.1000\ldots0$ -11111111

2^{-1} -255.

(Smallest Number) $(2^{-1}) \times 2^{-255} = 2^{-256}$.

Scanned by CamScanner

Q) Represent the Number $(+46.5)_{10}$ as a floating-point binary number with 24 bits. The normalized fraction mantissa has 16 bits and the exponent has 8 bits.

Sol:

$$46.5 = 32+8+4+2+0.5$$

$$(46.5)_{10} = (101110.1)_2$$

2 | 46	$0.5 \times 2 = 1.0$
2 | 23 - 0	
2 | 11 - 1	
2 | 5 - 1	
2 | 2 - 1	
1 - 0	
101110.1	

Sign

$\underbrace{0 \quad 1011101000000000}_{\text{16 bit mantissa}}$ $\underbrace{00000110}_{\text{8 bit exponent}}$

101110.1

1011101×2^6

Scanned by CamScanner

Computer Arithmetic.

Q) Show the contents of registers E, A, Q and SC during the process of multiplication of two binary number 11111 (multiplicand) and 10101 (multiplier). The signs are not included.

Sol: The above problem is normal multiplication

Given :-

multiplicand = 11111 multiplier = 10101

B = 11111 [(31)]$_{10}$. Q = 10101 (21)$_{10}$

A = 00000. , SC = No. of bits in multiplicand = 5

Multiplicand B = 11111	E	A	(Multiplier) Q	SC
INITIAL	0	00000	10101	101 (5)
Multiplier in Q $Q_n = 1$; add B first partial product shift right EAQ	Q	11111 11111 0 1 1 1 1	10101 11010	100 (4)
$Q_n = 0$ add B shift right EAQ	0	01111 00111	11010 11101	011 (3)
$Q_n = 1$ add B shift right EAQ	1	00111 +11111 00010 10011	11101 01110	010 (2)
$Q_n = 0$ shift right EAQ		01001	10111	001 (1)
$Q_n = 1$, add B shift right EAQ	1	01001 +11111 01000 10100	10111 01011	000 (0)

Scanned by CamScanner

Sc becomes 0, the result i.e., product is in

'AQ'

10 $\underbrace{1010001011}_{651}$.

B = 31, Q = 21

31 X 21

= 651.

NOTE :- B contains multiplicand.

A Contains no. of zeros equal to size of multiplicand & multiplier.

Q Contains multiplier.

SC Contains the no. of bits in multiplier.

AQ Contains product at the end of the multiplication (when SC=0)

Write Initial values in E A Q & SC registers.

Then check the Q_n bit i.e., the LSB of Q.

if $Q_n = 1$

then add B to A and result should be shifted to right.

if $Q_n = 0$.

then just shift the bits to right.

Scanned by CamScanner

(3-2) BOOTH ALGORITHM PROBLEMS

Q) Perform the multiplication using Booth's Algorithm. to the following

(a) 38 × 12 (b) −38 × 12 (c) 38 × −12 (d) −38 × −12

Sol:- Booth Algorithm:- Booth Algorithm gives a procedure for multiplying binary integers in signed-2's complement representation.

(a) 38 × 12 = 456.

38 = 0100110 12 = 0001100.

BR = 0100110 QR = 0001100

$\overline{BR}+1$ = 1011001 SC = 111 (7)

+1 AC = 0000000.

1011010

38 can fit into.
36 16 8 4 2 1
1 0 0 0 1 0.
but we need to represent the sign also so take 7 bits)

$Q_n Q_{n+1}$	BR = 0100110 $\overline{BR}+1$ = 1011010.	AC = 0000000	QR = 0001100	Q_{n+1}	SC.
	Initial	0000000	0001100	0	111
00	ashr	0000000	0000110	0	110
00	ashr	0000000	0000011	0	101
10	Subtract BR (i.e., Add $\overline{BR}+1$)	0000000 + 1011010			
	ashr	1011010 1101101	0000011 0000001	1	100
11	ashr	1110110	1000000	1	011
01	add BR	1110110 + 0100110 0001100			
	ashr	0001110	1000000 0100000	0	010
00	ashr	0000111	0010000	0	001
00	ashr	0000011	1001000	0	000

Product is in AC & QR
128 64 32 16 8 4 2 1
0000011 1001000.

$\boxed{38 \times 12}$ = 456.

256+128
+64+8
= 456.

Scanned by CamScanner

(b) $-38 \times 12 = -456$

$BR = -38 = 1011010$ ⟹

$\overline{BR}+1 = 0100101$
$+1$
0100110

$QR = 0001100$ (12), $AC = 0000000$
$SC = 111$ (7)

Remember :- Every negative no. is represented in its 2's Complement

$38 = 0100110$
$-38 = 1011001$
$+1$
1011010

plicand

$Q_n Q_{n+1}$	$BR = 1011010$ $\overline{BR}+1 = 0100110$	$AC = 0000000$	$QR = 0001100$ (Q_n)	Q_{n+1}	SC
	Initial	0000000	0001100	0	111
0 0	ashr	0000000	0000110	0	110
0 0	ashr	0000000	0000011	0	101
1 0	subtract BR. (i.e., add $\overline{BR}+1$)	0000000 +0100110 0100110	0000011		
	ashr	0010011	0000001	1	100
1 1	ashr	0001001	1000000	1	011
0 1	add BR	0001001 +1011010 1100011	1000000		
	ashr	1110001	1100000	0	010
0 0	ashr	1111000	1110000	0	001
0 0	ashr	1111100	0111000	0	000

product will be in 2's complement.
because $-38 \times 12 = -456$. (Negative no.'s will be in 2's complement)

1111100 0111000 ← 2's complement of -456
0000011 1000111
+1
(1000011 1001000) ← original no (-456)

Again do 2's Complement to get Original No.

(c) $= 38 \times -12 = -456$.

BR = 0100110

$\overline{BR}+1$ = 2's Comp of BR

$= 1011001 + 1 = 1011010$

-12 = should be Represented 2's Comp because -ve

= 2's Comp of 0001100

$= 1110011 + 1$

-12 = 1110100

QR =

BR = 0100110 QR = 1110100. SC = 111 (7).

$\overline{BR}+1$ = 1011010 AC = 0000000.

$Q_n Q_{n+1}$	BR = 0100110 $\overline{BR}+1$ = 1011010	AC = 0000000	QR = 1110100 (Q_n)	Q_{n+1}	SC
	Initial	0000000	1110100	0.	111
00	ashr	0000000	0111010	0	110
00	ashr	0000000	0011101	0	101
10	subtract BR (i.e, add $\overline{BR}+1$)	0000000 +1011010 1011010.	0011101		
	ashr	1011101	0001110	1	100
01	Add BR.	1101101 +0100110 10010011	0001110		
	ashr	0001001	1000111	0	011
10	subtract BR	0001001 +1011010 1100011	1000111		
	ashr	1100001	1100011	1	010
11	ashr	1111000	1110001	1	001
11	ashr	11111000	0111000	1	000.

product = 1111100 0111000. ← 2's Complement of -456.

Scanned by CamScanner

a) $-38 \times -12 = +456.$

$BR = -38 = 1011010$

$\overline{BR}+1 = (\cancel{0100}) = 0100110.$

$QR = -12 = 0001100$

2's comp $= 1110011$
$+1$

$QR = 1110100$

BR = -ve number so it should be represented in 2's complement.

$\overline{BR}+1$ = 2's (2'comp of BR) = BR (38).

$AC = 0000000$, $SC = 111$ (7) count

Q_nQ_{n+1}	BR = 1011010 $\overline{BR}+1$ = 0100110	AC = 0000000	QR = 1110100 (Q_n)	Q_{n+1}	SC
	Initial.	0000000	1110100.	0	111
00.	ashr	0000000	0111010	0	110
00	ashr	0000000	0011101	0	101
10	Subtract BR (i.e., add $\overline{BR}+1$)	0000000 +0100110 0100110	0011101		
	ashr	0010011	0001110	1	100
01	Add BR	0010011 +1011010 1101101	0001110		
	ashr	1110110	1000111	0	011
10	Subtract $\overline{BR}+1$	1110110 +0100110 1001100	1000111		
	ashr	0001110	0100011	1	010
11	ashr	0000111	0010001	1	001
11	ashr.	0000011	1001000	1	000

result 456

b)

Scanned by CamScanner

product = (AC) 0000011 (QR) 1001000
256 128 64 32 16 8 4 2 1

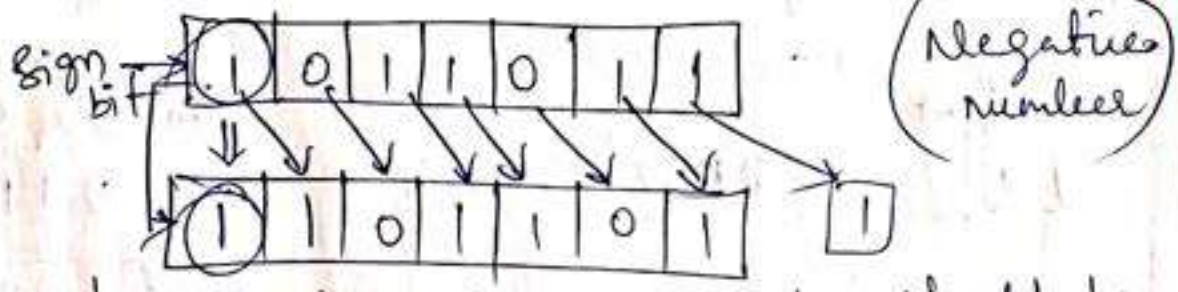
= 456.

256
+128
+ 64
8

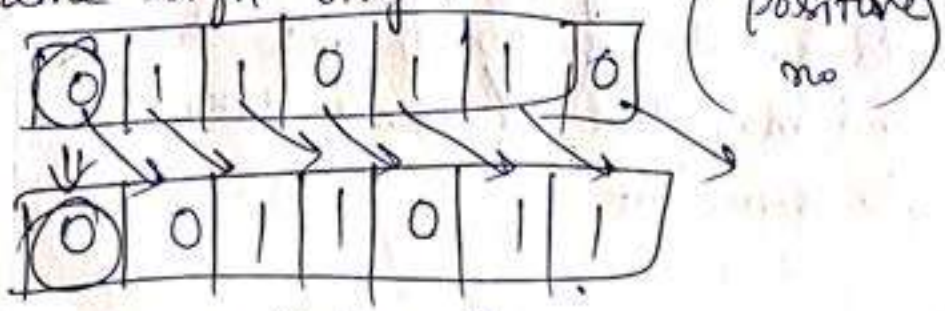
NOTE :-

* 00 — arithmetic shift Right.
  01 — subtract BR (i.e., add $\overline{BR}+1$)
  10 — Add BR to AC.
  11 — arithmetic shift Right.

* In Arithmetic shift Right, the sign bit will remain as it is after shifting also.



After shifting also the number should be in the same sign only.



* BR Contains multiplicand,
  If it is -ve, perform 2's complement and store it as BR.
  Again perform 2's complement of now what is there in BR reg. as $\overline{BR}+1$

Scanned by CamScanner

* If you are given a decimal numbers. then represent them. also with sign bit in MSB position.

Eg:- 38 =

32	16	8	4	2	1
1	0	0	1	1	0

Here 38 requires '6' bits to represent in binary number so we have to take another bit 0 in front i.e, in MSB to represent SIGN BIT. So it requires total '7' bits.

+38 =

sign bit	magnitude
0	1 0 0 1 1 0

= 7 bits.

Eg:- +32

sign bit	
0	1 0 0 0 0 0

= 7 bits

-32 in 2's complement.

1 0 1 1 1 1 1
+1
0 1 0 0 0 0 0

* If they mention in the question then take those no. of bits to represent both sign and magnitude (do not take Extra bit (6 bits) to represent)

Scanned by CamScanner

DIVISION.

Q) Show the contents of registers E, A, Q and SC during the process of division of (Use dividend of 8 bits)

(a) 10100011 by 1011 b) 00001111 by 0011

Sol (a) 10100011 by 1011

Dividend = 10100011

Divisor = 1011

A Q.

~~10100011~~

A Q

1010 0011

B = 1011

$\bar{B}+1$ = 0100
+1
0101

SC = No. of bits in divisor = 4.

(128 64 32 16 8 4 2 1)
10100011
128+32+2+1 = 163

1011
= 11 → Dividend

(11) (163) (14)
−11
Divisor 53 Quotient
−44
(9) → Remainder

B = 1011 $\bar{B}+1$ = 0101	E	A	Q	SC
Initial Dividend:		1010	0011	4
Shl EAQ	1	0100	0110	
add $\bar{B}+1$		0101 1001		
E=1, set Q_n = 1	1	1001 1001	0110 0111	3
Shl EAQ	1	0010	1110	
add $\bar{B}+1$		0101 0111	1110	
E=1, set Q_n=1	1	0111	1111	2
Shl EAQ.	0	1111	1110.	
add $\bar{B}+1$	1	0101 0100	1110	
E=1, set Q_n=1		0100	1111	1

Scanned by CamScanner

	E	A	Q	SC
Shl EAQ	0	1001	1110	
Add $\bar{B}+1$		0101		
	0	1110	1110.	
E = 0, $Q_n = 0$				
Restore Remainder		1110		
Add B		+1011		
	1	1001	1110.	0
		Discard E		
		Remainder	= A = 1001 (9)	
		Quotient	= 1110 (14)	

(b) 00001111 by 0011

AQ = Dividend = 0000 1111 (15)

B = Divisor = 0011 (3)

$\bar{B}$ = 1100

+1

$\bar{B}+1$ = 1101

A = 0000

Q = 1111

B = 0011

$\bar{B}+1$ = 1101.

SC = 100 (4) No. of bits in divisor

3)15(5

15

0

Remainder = 0

Quotient = 5

Divisor = 3

Dividend = 15

Scanned by CamScanner

$B = 0011$ $\overline{B}+1 = 1101$	E	A	Q	SC
Dividend.		0000	1111	4.
Shl EAQ	0	0001	1110.	
Add $\overline{B}+1$	0	1101 1110	1110.	
$E=0, Q_n=0$		1110		
Restore Remainder Add B.	1	+ 0011 0001	1110	3
Shl EAQ	0	0011.	1100.	
Add $\overline{B}+1$	1	1101 0000	1100.	2
$E=1, Q_n=1$		0000	1101	
Shl EAQ	0	0001	1010.	
Add $\overline{B}+1$	0	1101 1110	1010	1
$E=0, Q_n=0$		1110	1010.	
Add B.		+ 0011	1010	
Restore Rem	1	0001		
Shl EAQ	0	0011	0100.	
Add $\overline{B}+1$.	1	1101 0000	0100	0
$E=1, Q_n=1$		0000	0101	

A = Remainder = 0000 = 0

Q = Quotient = 0101 = 5

3)15(5
15
0.

Scanned by CamScanner

→ Dividend is shared b/w two register A and Q with equal bits.

→ Divisor is stored B.

→ 2's complement of B is $\bar{B}+1$

→ SC (Sequence Counter tells how the iteration has to be performed)

SC = No. of bits in divisor (B)

Step 1 :- Shift left EAQ. Contents.

Step 2 :- Perform subtraction i.e, Add $\bar{B}+1$ to A.

Step 3 :- If during the addition $\bar{B}+1$ to A generates carry. that is replaced in E flip flop. If not in step 1 during the shift left operation the same value will be checked.

Step 4 :- If E=1 then set $Q_n=1$

If E=0, $Q_n=0$.

Add B to restore Remainder.

Decrement SC by 1 value

Continue from step 1 to step 4.

Step 5 :- until SC=0.

Scanned by CamScanner

Q) A binary floating point number has seven bits for a biased exponent. The constant used for the bias is 64.

(a) List the biased representation of all exponents from −64 to +63.

(b) Show that a 7-bit magnitude comparator can be used to compare the relative magnitude of the two exponents.

(c) Show that after the addition of two biased exponents it is necessary to subtract 64 in order to have a biased exponents sum. How would you subtract 64 by adding the 2's complement value?

(d) Show that after the subtraction of two biased exponents it is necessary to add 64 in order to have a biased exponent difference.

Sol (a) e = exponent e+64 = biased Exponent.

e	e+64
−64	−64+64=0
−63	−63+64=1
−62	−62+64=2
⋮	⋮
−1	−1+64=63
0	0+64=64
+1	1+64=65
⋮	⋮
+62	62+64=126
+63	63+64=127

Scanned by CamScanner

(b) The biased exponent follows the same algorithm as magnitude Comparator

(c) $(e_1+64)+(e_2+64) = (e_1+e_2+64)+64$.

Subtract 64 to obtain biased exponent sum.

(d) $(e_1+64)-(e_2-64) = e_1+e_2$

add 64 to obtain biased exponent difference.

Scanned by CamScanner

UNIT- IV

1. INPUT-OUTPUT ORGANIZATION

PROBLEMS

1) Indicate whether the following constitute a control, status or data transfer commands. (TOPIC - I/O INTERFACE).

a) skip next instruction if flag is set - Status Command
b) Seek a given record on a magnetic disk - Control Command
c) check if I/O device is ready - Status Command
d) Move printer paper to beginning of next page - Control Command
e) Read interface status register - Data input Command

2) How many characters per second can be transmitted over a 1200-baud line in each of the following modes? (Assume a character code of 8 bits). (TOPIC:- Asynchronous Serial Trans.)

a) Synchronous Serial transmission.

$$\frac{1200}{8} = 150 \text{ characters per second. (cps)}$$

b) Asynchronous serial transmission with two stop bits.

$$\frac{1200}{11} = 109 \text{ cps}$$

c) Asynchronous serial transmission with one stop bit.

$$\frac{1200}{10} = 120 \text{ cps.}$$

3) Information is inserted into a FIFO buffer at a rate of 'm' bytes per second. The information is deleted at a rate of 'n' byte per second. The maximum capacity of the buffer is k bytes.

a) How long does it take for an empty buffer to fill up when $m > n$? (TOPIC:- FIFO BUFFER)

$$\frac{k \text{ bytes}}{(m-n) \text{ bytes/sec}} = \frac{k}{m-n} \text{ sec.}$$

Scanned by CamScanner

(b) How long does it take for a full buffer to empty when $m<n$?

Sol:- $\frac{K}{n-m}$ sec

(c) Is the FIFO buffer needed if $m=n$?

Sol:- NO need for FIFO.

(4) What are the values of input ready and output ready and control bits F_1 through F_4 when (TOPIC :- FIFO BUFFER)

(a) The buffer is empty ?

	Input ready	output ready	$F_1 - F_4$.
Empty buffer	1	0	0000

(b) The buffer is full

Full buffer.	0	1	1111

(c) The buffer contains two data items ?

Two data items	1	1	0011.

(5) Why are the read and write control lines in DMA Controller bidirectional ? Under what condition and for what purpose are they used as inputs ? Under what condition and for what purpose are they used as outputs ?

Ans :- When the CPU communicates with DMA Controller, the read and write lines are used as inputs from the CPU to the DMA controller.

When the DMA Controller communicates with memory, the read and writer are used as outputs from the DMA to memory.

Scanned by CamScanner

6) It is necessary to transfer 256 words from a magnetic disk to a memory section starting from address 1230. The transfer is by means of DMA as shown in the figure TOPIC :- DMA.

(a) Give the initial values that the CPU must transfer to the DMA controller.

Sol :- CPU initializes the DMA by sending the word count, address and controls to be used during the DMA transfer, to its appropriate registers

(i) 256 to word count register.

(ii) DMA address register = 1230.

(iii) Bits to the Control register to specify a 'WRITE' Operation

(b) Given the step-by-step account of the actions taken during the input of the first two words.

1. I/O device sends a "DMA request"
2. DMA sends BR (Bus Request) to CPU
3. CPU responds with a BG (Bus Grant).
4. Contents of DMA address register are placed in address bus.
5. DMA sends "DMA acknowledge" to I/O device and enables the write control line to memory.
6. Data word is placed on data bus by I/O device.
7. Increment DMA address register by 1 and Decrement DMA word count register by 1.
8. Repeat steps 4-7 for each data word to be transferred.

Scanned by CamScanner

Q) Why does DMA have priority over the CPU when both request a memory transfer?

Ans The CPU can wait to fetch instructions and data from memory without any damage occuring except loss of time, DMA usually transfers data from a device that cannot be stopped since information continues to flow so loss of data may occur.

Q) What are the differences between isolated I/O and memory mapped I/O? What are the advantages and disadvantages of each.

Ans.

	Isolated I/O		Memory Mapped I/O
1	Def:- A method to perform I/O operations b/w the CPU and peripheral devices in a computer which uses two separate address spaces for memory and I/O devices	1	Def:- A method to perform input/output operations b/w the CPU and peripheral devices in a computer which uses one address space for memory and I/O devices.
2	Uses two separate address spaces for memory and I/O.	2.	Uses the same address space for both memory & I/O devices.
3.	All addresses can be used by memory.	3	As the memory mapped I/O uses one address space for both I/O & memory, the available addresses for memory are minimum
4.	Uses the ~~same~~ separate Instructions for ~~both~~ I/O and memory, read & write operations.	4.	Uses the same instructions for both I/O and memory operations.

Scanned by CamScanner

5.	More Efficient	5	Less Efficient.
6.	I/O read, I/O write separate control lines for I/O transfer and Memory read & Memory write ctrl lines for Memory transfer.	6.	Only One set of read and write control lines to refer either to memory or to I/O.
7.	Computers with isolated I/O use separate memory instruction for memory & separate I/O instructions for I/O.	7.	Computers with memory mapped I/O can use memory - type Instructions ~~for~~ to access I/O data
8	Advantages:- → It is more Efficient due to separate buses,	8	Advantages. (Disadvantages) ① The size of addressable memory is reduced due to addition of I/O. address so there is less. space Advantages:- Simpler logic or same instructions are used for both memory & I/O.

Scanned by CamScanner

MEMORY ORGANIZATION.

1) (a) How many 128 x 8 RAM chips are needed to provide a memory capacity of 2048 bytes?

Sol:- $\frac{2048}{128} = 16$ chips

b) How many lines of the address bus must be used to access 2048 bytes of memory? How many of these lines will be common to all chips.

$2048 = 2^{11}$ 11 lines to address 2048 bytes.

$128 = 2^{7}$ 7 lines to address each chip

4 lines to decoder for selecting 16 chips.

(c) How many lines must be decoded for chip select? Specify the size of the decoders.

Sol:- 4 x 16 decoder (Takes 4 as inputs)

2) A computer uses RAM chips of 1024x1 capacity.

(a) How many chips are needed, and how should their address lines be connected to provide a memory capacity of 1024 bytes? 1024 = 1024x8 bits

Sol:- 8 chips are needed with address lines connected in parallel. $\left(\frac{1024 \times 8}{1024} = 8\right)$

(b) How many chips are needed to provide a memory capacity of 16K bytes? Explain in words how the chips are to be connected to address bus.

Sol:- 16x8 = 128 chips. Use 14 address lines ($16K = 2^{14}$)

10 lines to specify chip address

4 lines ~~for~~ are decoded into 16-chip select inputs.

Scanned by CamScanner

③ ~~A ROM chip of 1024x8 bits has four~~

A Computer employs RAM chips of 256x8 and ROM chips of 1024x8. The Computer system needs 2K bytes of RAM, 4K bytes of ROM, and four Interface units each with four registers. A Memory mapped I/O Config. is used. The two highest-order bits of the address bus are assigned 00 for RAM, 01 for ROM and 10 for interface registers.

ⓐ How many RAM and ROM chips are needed?

Sol:- Given :-

RAM — 256x8 , ROM — 1024x8

Size of RAM = 256 x 8

Memory size required = 2K bytes

= 2 x 1024 bytes

= 2 x 1024 x 8 bit.

Total Number of RAM chips required

$$= \frac{2 \times \cancel{1024}^{4} \times \cancel{8}}{\cancel{256}_{1} \times \cancel{8}} = \boxed{8 \text{ chips}}$$

Size of ROM = 1024 x 8

Memory size required = 4K bytes = 4 x 1024 x 8 bits

Total Number of ROM chips required $= \frac{4 \times 1024 \times 8}{1024 \times 8} = \boxed{4 \text{ chips}}$

Hence 8 RAM chips and 4 ROM chips are required.

Scanned by CamScanner

(b) ~~Draw a memory address map for the system.~~

Sol: (c) Give the address range in hexadecimal for RAM, ROM, and interface

(b) Draw a memory address map for the system

Sol:-

Component	Address	16 15 14 13	12 11 10 9	8 7 6 5	4 3 2 1
RAM	0000-0FFF	0 0 0 0	5x32 decoder	x x x	x x x x
ROM	1000-1FFF	0 0 0		x	x x x x x x x x

to $\overline{CS2}$ — 3x8 decoder

Interface $4 \times 4 = 16$ registers ; $16 = 2^4$

Q) A digital computer has a memory unit of 64K×16 and a cache memory of 1K words. The cache uses direct mapping with a block size of four words.

(a) How many bits are there in the tag, index, block and word fields of the address format.

$1K = 1024$

Sol:- Given:-

Memory = 64K×16

$= 64 \times 2^{10} \times 16 = 2^6 \times 10^{10} \times 6$

$= 2^{16} \times 16.$

$2^{\text{address bits}}$

16 = address bits.

cache = 1K words = 2^{10} words (1024)

block size = 4 words.

TAG	BLOCK	WORD

NOTE 2^k words in cache, 2^n words in main memory

2^{10}, 2^{16}

n in main memory.

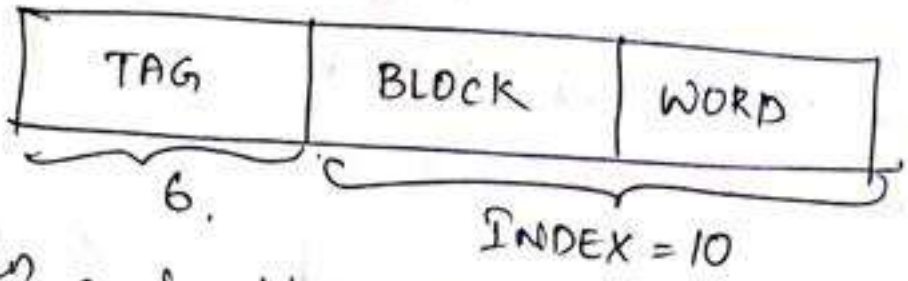
k bits (Index field) | n-k bits (Tag field)

Scanned by CamScanner

Index field = k = 10 bits

Tag field = n-k = 16-10 = 6 bits.

(n = 16 in memory, k = 10 in Cache)



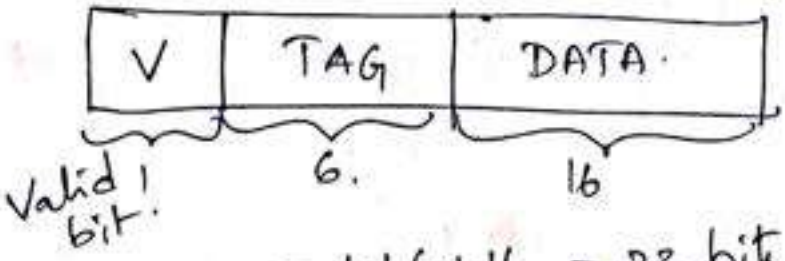
Given each block size = 4

Cache = 1024

No. of block bits = $\frac{1024}{\text{each block size}} = \frac{1024}{4} = 256 = 2^{8}$

Block = 8
word = 2 } Total = 10 (Index)

(b) How many bits are there in ~~the~~ each word of cache, and how are they divided into functions? Include a valid bit.



64K × 16 → data bits.

= 1+6+16 = 23 bits.

(c) How many blocks can the cache accommodate?

Sol 2^8 = 256 blocks of 4 words each.

Scanned by CamScanner

Q) An address space is specified by 24 bits and the corresponding memory space by 16 bits.

a) How many words are there in the address space?

Sol:- Address space = 24 bits. $2^{24} = 16M$ words

b) How many words are there in the memory space?

Sol:- Memory space = 16 bits $2^{16} = 64K$ words.

Q) The access time of a cache memory is 100ns and that of main memory 1000 ns. It is estimated that 80 percent of the memory requests are for read and the remaining 20 percent for write. The hit ratio for read accesses only is 0.9. A write through procedure is used.

(a) What is the average access time of the system considering only memory read cycles?

Sol:- $0.9 \times \underbrace{100}_{\text{cache access}} + 0.1 \times \underbrace{1000}_{\text{cache + memory access}} = 90 + 110 = 200$ n sec

b) What is the average access time of the system for both read and write requests?

Sol:- $0.2 \times \underbrace{1000}_{\text{write access}} + 0.8 \times \underbrace{200}_{\text{read access from (a)}} = 200 + 160 = 360$ n sec

c) What is the hit ratio taking into considerations the write cycle?

Sol:- Hit ratio $= 0.8 \times 0.9 = 0.72$.

Scanned by CamScanner

V UNIT

MULTIPROCESSORS

1) Difference between tightly coupled multiprocessors and loosely coupled multiprocessors from the viewpoint of hardware organization.

Tightly Coupled Multiprocessors	Loosely Coupled Multiprocessors
① Tighty Coupled multiprocessors require that all processors in the system have access to a common, global memory. (Shared memory)	① In loosely coupled multiprocessors, the memory is distributed and a mechanism is required to provide message-passing between the processors. (distributed memory)
② Tightly coupled systems are easier to program. since no special steps are required to make shared data available to two or more processors.	② A Loosely Coupled system required that sharing of data be implemented by the messages.
③ This type of multiprocessors has low data rate.	③ This type of multiprocessor system has high data rate.
④ The cost of loosely coupled multiprocessor system is less.	④ Tightly coupled multiprocessor system is more costly.
⑤ Memory conflicts does not take place.	⑤ Chances for memory conflicts.

Scanned by CamScanner

R18

Code No: 153AG

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD

B.Tech II Year I Semester Examinations, March - 2021

COMPUTER ORGANIZATION AND ARCHITECTURE

(Computer Science and Engineering)

Time: 3 hours Max. Marks: 75

Answer any five questions
All questions carry equal marks

- - -

1.a) Discuss the functional units of a digital computer.
b) Demonstrate construction of a common bus system with multiplexers. [7+8]

2.a) Design a 4-bit combinational circuit decrementer using four full-adder circuits.
b) What is the difference between a direct and an indirect address instruction? How many references to memory are needed for each type of instruction to bring an operand into a processor register? [7+8]

3.a) Discuss the need of memory stack and stack limits.
b) Explain the general register organization of the processor. [7+8]

4. Explain addition and subtraction of floating point numbers with an example and necessary flowchart. [15]

5.a) A two way set associative cache has lines of 16 bytes and a total size of 8 K bytes. The 64 Mbytes main memory is byte addressable. Show the format of main memory address.
b) How does SDRAM differ from ordinary DRAM? [8+7]

6.a) Explain the major differences between the central computer and peripheral. How to resolve these differences?
b) Discuss the Strobe control method of Asynchronous data transfer. [8+7]

7.a) What is parallel processing? Explain Flynn's classification of computer.
b) Illustrate vector operations and vector processing. [8+7]

8.a) Discuss about RISC Pipeline.
b) What is cache coherence problem? Discuss solutions for it. [7+8]

---ooOoo---

R18

Code No: 153AG

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD

B.Tech II Year I Semester Examinations, October - 2020

COMPUTER ORGANIZATION AND ARCHITECTURE

(Computer Science and Engineering)

Time: 2 hours | Max. Marks: 75

Answer any five questions
All questions carry equal marks

1.a) Draw the bus system for four registers and explain.
b) An 8-bit register contains the binary value 10011100. What is the register value after an Arithmetic Shift Right? Starting from the initial number 10011100, determine the register value after an arithmetic Shift Left, and state whether there is an overflow. [7+8]

2. Draw block diagram of a control memory and the associated hardware needed for selecting the next micro instruction address. [15]

3. Perform the arithmetic operation (+42)+(-13) and (-42)-(-13) in binary using signed 2's complement representation for negative numbers. [15]

4.a) Differentiate between Isolated I/O and memory-mapped I/O.
b) Explain programmed-I/O in detail. [8+7]

5.a) Write the major characteristics of RISC processors.
b) Draw a space-time diagram for a four-segment pipeline showing the time it takes to process six tasks and explain. [7+8]

6.a) Draw the flowchart for instruction cycle and explain.
b) Explain the following instructions: BUN, ISZ, BSA, LDA, STA. [7+8]

7. Explain various Data Manipulation instructions with examples. [15]

8. With an example, explain Booth Multiplication algorithm. [15]

---ooOoo---

Code No: 153AG **R18**

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD

B.Tech II Year I Semester Examinations, September - 2021

COMPUTER ORGANIZATION AND ARCHITECTURE

(Computer Science and Engineering)

Time: 3 hours **Max. Marks: 75**

Answer any five questions

All questions carry equal marks

1.a) Draw the block diagram of a digital computer and explain the purpose of each part.

b) Design a 4-bit combinational circuit decrementer using four full-adder circuits. [6+9]

2. What are the common fields found in instruction format? Explain various instruction formats based on types of CPU Organization? [15]

3. Perform the arithmetic operation (+41)+(-13) and (-41)-(-13) in binary using signed 2's complement representation for negative numbers. [15]

4.a) Draw the block diagram of a typical DMA controller and explain.

b) Explain Daisy-Chain priority interrupt in detail. [8+7]

5.a) Construct a diagram for a 4×4 omega Switching network. Show the switch setting required to connect input 3 to output 1.

b) Give a brief note on mutual exclusion with a semaphore. [9+6]

6.a) Differentiate between computer organization and computer architecture.

b) Explain the Stored Program organization in detail. [7+8]

7. Explain the microprogram sequencer for a control memory with a neat diagram. [15]

8. Derive an algorithm in flowchart form for adding and subtracting two fixed point binary numbers when negative numbers are in the signed-2's complement representation. [15]

---ooOoo---

SCIENT INSTITUTE OF TECHNOLOGY

B.Tech II Year / I Sem I-MID Examination , December-2021

SUBJECT :Computer Organization and Architecture **BRANCH : CSE**

MARKS : 10 **DURATION : 1 Hour** **DATE: 15/12/2021**

Each Question Carries Five Marks **Answer any TWO from the following :**

Sl.No	Questions	Bloom Taxanomy
1	A) Explain with a neat sketch of bus system using multiplexers. B) A digital computer has a common bus system for 16 registers of 32 bits each. The bus is constructed with multiplexers. a. How many selection inputs are there in each multiplexer? b. What size of multiplexers are needed? c. How many multiplexers are there in the bus?	L1,L3
2	Explain Memory Reference Instructions.	L1
3	Explain Data Manipulation Instructions.	L1
4	A) Explain Addressing modes. B) Perform the logical AND,OR, and XOR with the two binary strings 10011100 and 10101010	L1,L2

A

SCIENT INSTITUTE OF TECHNOLOGY
B.Tech II Year / I Sem II-MID Examination, FEB -2022

SUBJECT: COMPUTER ORGANIZATION AND ARCHITECTURE | BRANCH: CSE

MARKS: 10 | DURATION: 1 Hour | DATE: 16/02/2022

Each Question Carries Five Marks | Answer any TWO from the following

Sl .No	Questions	Bloom Taxonomy
1	Explain about Source Initiated Transfer and Destination initiated transfer using Strobe control with neat sketch	L1
2	What is Floating point representation? Explain the IEEE standard for Floating point representation with examples.	L2
3	Convert the following numbers with the indicated bases to decimal $(12121)_3$; $(4310)_4$; $(50)_7$ and $(198)_{12}$	L2
4	a) Draw a space-time diagram for a four-segment pipeline showing the time it takes to process six tasks and explain b) Write the major characteristics of RISC processors	L2 & L1

B

SCIENT INSTITUTE OF TECHNOLOGY
B.Tech II Year / I Sem II-MID Examination , FEB-2022

SUBJECT : COMPUTER ORGANIZATION AND ARCHITECTURE **BRANCH : CSE**

MARKS : 10 **DURATION : 1 Hour** **DATE: 16/02/2022**

Each Question Carries Five Marks **Answer any TWO from the following :**

Sl .No	Questions	Bloom Taxanomy
1	Differences between Isolated IO and Memory mapped IO	L3
2	Perform the arithmetic operations (+70)+(+80) and (-70)+(-80) with the binary numbers in signed-2's complement representation . Use eight bits to accommodate each number together with its sign. Show that overflow occurs in both cases that the last two carries are unequal and that there is a sign reversal	L2&L3
3	Explain Booth Multiplication algorithm with neat sketch, flowchart and example	L1
4	Explain about DMA controller with a neat sketch	L1

C

SCIENT INSTITUTE OF TECHNOLOGY
B.Tech II Year /I Sem II-MID Examination, FEB-2022

SUBJECT : COMPUTER ORGANIZATION AND ARCHITECTURE
BRANCH : CSE
MARKS : 10
DURATION : 1 Hour
DATE: 16/02/2022

Each Question Carries Five Marks
Answer any TWO from the following :

Sl .No	Questions	Bloom Taxanomy
1	Give a neat Sketch that illustrates components in a typical memory hierarchy	L1
2	Perform the arithmetic operations (+42)+(-13) and (-42)-(-13) in binary using 2's complement representation for negative numbers	L1
3	A) Perform the subtraction with the following unsigned binary numbers by taking the 10's complement of the subtrahend i) 5250-1321 ii) 20-100 B) Write algorithm and also draw flowchart of floating point multiplication and division	L2,L1
4	Explain about following interconnected Structures a) Time-shared common bus b) Multiport Memory	L1

SCIENT INSTITUTE OF TECHNOLOGY

B.Tech II Year / I Sem I-MID Examination , December-2021

SUBJECT :Computer Organization and Architecture **BRANCH : CSE**

MARKS : 10 **DURATION : 1 Hour** **DATE: 15/12/2021**

Each Question Carries Five Marks **Answer any TWO from the following :**

Sl.No	Questions	Bloom Taxanomy
1	**A) Explain and Design a 4-bit Combinational circuit decrementer using four full adder circuits.** **B) The 8-bit registers AR, BR, CR, and DR initially have the following values:** **AR =11110010** **BR =11111111** **CR =101 11001** **DR =11101010** Determine the 8-bit values in each register after the execution of the following sequence of microoperations. i. AR <- AR + BR Add BR to AR ii. CR <- CR ∧ DR, AND DR to CR iii. BR <- BR + 1 increment BR iv. AR <-AR - CR, Subtract CR from AR	L2,L3
2	**A) A computer uses a memory unit with 256K words of 32 bits each. A binary instruction code is stored in one word of memory. The instruction has four parts: an indirect bit, an operation code, a register code part to specify one of 64 registers, and an address part.** a. How many bits are there in the operation code, the register code part, and the address part? b. Draw the instruction word format and indicate the number of bits in each part. c. How many bits are there in the data and address inputs of the memory? **B) What is the difference between a direct and an indirect address instruction? How many references to memory are needed for such type of instruction to bring an operand into a processor register?**	L3,L2
3	Explain about Address Sequencing in Microprogrammed Control Unit	L1
4	Write a program to evaluate the arithmetic statement: $x = \frac{A - B + C * (D * E - F)}{G + H * K}$ a. Using a general register computer with three address instructions. b. Using a general register computer with two address instructions.	L3

SCIENT INSTITUTE OF TECHNOLOGY

B.Tech II Year / I Sem I-MID Examination , December-2021

SUBJECT :Computer Organization and Architecture **BRANCH : CSE**

MARKS : 10 **DURATION : 1 Hour** **DATE: 15/12/2021**

Each Question Carries Five Marks **Answer any TWO from the following :**

Sl.No	Questions	Bloom Taxanomy
1	A)Explain Shift Microoperations B) Starting from an initial value of R =11011101, determine the sequence of binary values in R after a logical shift-left, followed by a circular shift-right, followed by a logical shift-right and a circular shift-left.	L1,L3
2	Explain instruction cycle with the help of flow chart	L1
3	A) Explain difference between hardwired control unit and microprogrammed control unit B) Explain Micro programmed control organization with neat sketch	L3,L1
4	4) A)An instruction is stored at location 300 with its address field at location 301 .The address field has the value 400. A processor register R 1 contains the number 200. Evaluate the effective address if the addressing mode of the instruction is (a) direct; (b) immediate; (c) relative; (d) register indirect; (e) index with R1 as the index register. B) The memory unit of a computer has 256K words of 32 bits each. The computer has an instruction format with four fields: an operation code field, a mode field to specify one of seven addressing modes, a register address field to specify one of 60 processor registers, and a memory address. Specify the instruction format and the number of bits in each field if the in instruction is in one memory word.	L3,L3

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD
HYDERABAD-500085

Scient Institute of Technology(C0)

B.Tech - R18 - II Year - I Semester

COMPUTER SCIENCE AND ENGINEERING

Final University Consolidated Internal Marks Report-Date- 2022-03-05 15.41.04

HNTO	15301	15306	15307	15317	15332	153AB	153AG	153AJ	153AK	153BK
20C01A0501	19	24	22	22	85	22	22	24	18	23
20C01A0502	22	24	25	22	85	21	19	25	19	20
20C01A0503	18	18	22	18	95	19	20	24	17	18
20C01A0504	17	19	19	20	75	18	16	20	17	16
20C01A0505	10	-1	20	15	75	18	13	14	14	15
20C01A0506	17	16	18	19	89	20	17	21	15	17
20C01A0507	20	19	21	19	75	22	17	21	17	18
20C01A0509	14	17	17	16	76	13	14	8	10	12
20C01A0510	15	11	10	21	77	22	14	19	12	12
20C01A0511	19	14	16	21	76	22	12	19	16	18
20C01A0512	14	-1	10	18	70	13	12	7	9	9
20C01A0513	14	13	14	18	-1	17	7	7	3	4
20C01A0514	15	20	25	19	80	20	15	19	18	17
20C01A0515	10	-1	10	15	75	14	5	4	8	5
20C01A0517	15	17	14	19	80	18	14	18	14	14
20C01A0518	22	21	23	22	85	23	18	21	20	21
20C01A0519	19	15	18	21	74	19	15	20	16	17
20C01A0521	15	9	17	19	76	16	17	11	12	16
20C01A0522	22	24	24	22	84	22	21	25	24	20
20C01A0525	24	25	25	19	91	23	23	25	25	25
20C01A0526	21	23	22	20	79	22	18	24	21	22
20C01A0527	20	22	22	20	90	19	20	23	24	21
20C01A0528	14	18	18	19	80	18	14	12	11	12
20C01A0529	14	18	15	21	70	18	14	15	11	11
20C01A0530	22	18	24	22	76	22	17	22	19	17
20C01A0531	10	8	11	16	-1	5	5	4	11	2
20C01A0532	16	9	11	17	71	14	16	14	16	15
20C01A0533	14	7	10	18	76	16	12	10	10	9
20C01A0534	23	24	22	22	78	23	23	24	25	23
20C01A0535	20	15	20	21	78	22	19	22	17	20
20C01A0537	20	22	18	21	78	21	14	19	17	14
20C01A0538	15	-1	12	15	70	18	8	16	10	10

HNTO	15301	15306	15307	15317	15332	153AB	153AG	153AJ	153AK	153BK
20C01A0587	21	25	23	18	89	23	20	23	20	21
20C01A0588	18	16	20	20	64	13	14	19	4	11
20C01A0590	18	17	18	19	64	23	19	21	17	17
20C01A0591	20	22	23	21	90	22	19	21	20	19
20C01A0592	21	20	21	20	86	22	16	22	19	18
20C01A0593	19	15	17	15	85	14	14	14	10	12
20C01A0594	24	18	23	21	89	20	16	21	14	14
20C01A0595	18	17	18	19	83	21	14	15	16	17
20C01A0596	22	21	22	22	90	19	19	20	24	23
20C01A0597	22	23	19	22	90	21	23	25	19	23
20C01A0598	14	14	18	5	72	8	14	10	17	15
20C01A0599	18	14	16	14	68	19	11	16	9	12
20C01A05A0	20	17	18	18	89	14	12	15	12	15
20C01A05A1	23	19	20	20	86	20	17	18	16	19
20C01A05A2	21	7	18	18	80	19	13	16	14	14
20C01A05A3	20	8	18	15	-1	14	11	6	8	14
20C01A05A4	19	14	19	18	87	21	16	19	15	18
20C01A05A5	21	17	17	21	89	18	16	18	22	18
20C01A05A6	20	21	21	22	89	16	21	25	22	22
20C01A05A7	22	22	25	19	90	21	14	21	15	19
20C01A05A8	20	19	18	17	80	17	14	22	14	16
20C01A05A9	18	5	17	19	83	14	14	17	10	12
20C01A05B0	23	21	19	20	87	21	20	23	24	21
20C01A05B1	22	17	20	20	84	16	17	23	19	17
20C01A05B2	20	20	20	20	90	20	15	23	19	15
21C05A0501	21	21	24	18	79	21	17	16	16	19
21C05A0502	19	19	20	21	70	21	18	20	17	18
21C05A0503	16	22	20	15	-1	19	23	23	15	18
21C05A0504	-	-	-	-1	-1	7	8	10	4	6
21C05A0505	20	16	21	17	84	18	11	11	13	12
21C05A0506	23	23	24	23	83	22	18	18	17	17
21C05A0507	21	20	25	19	86	21	19	20	14	17
21C05A0508	23	17	19	21	87	22	15	18	12	15
21C05A0509	15	17	22	19	88	18	17	19	15	14
21C05A0510	23	14	19	16	81	20	17	19	18	15
21C05A0511	23	24	23	20	83	18	18	12	17	21
21C05A0512	21	17	22	18	-1	15	11	9	12	14
21C05A0513	-1	10	14	16	-1	9	12	6	6	9
21C05A0514	20	14	20	17	79	20	15	16	16	14
21C05A0515	18	19	23	19	80	20	16	17	15	19
Total:117	2052	1850	2180	2067	8232	2088	1726	1981	1734	1789

SCIENT INSTITUTE OF TECHNOLOGY

II B.Tech I Sem (R18) MID Examination Marks : Branch CSE-A : 2021 - 22

Subject: Computer Organisation & Architecture Name of the Faculty: B. Ramya Sree.

S.NO	H T NUMBER	NAME OF THE STUDENT	MID I				MID II				AVERAGE MARKS
			SUB	OBJ	ASGN	TOT	SUB	OBJ	ASGN	TOT	
1	20C01A0501	AAKANKSHA PANDEY	8	8	5	21	9	8	5	22	22
2	20C01A0502	AKULA SHAILAJA	5	7	5	17	8	8	5	21	19
3	20C01A0503	ALUGUBELLY JAYANTH REDDY	7	6	5	18	9	8	5	22	20
4	20C01A0504	ANANTHARAPU SAI MURALI	4	3	5	12	9	6	5	20	16
5	20C01A0505	ARKUTI SANTHOSH	4	2	5	11	3	6	5	14	13 ✓
6	20C01A0506	B SITARAMARAJU	5	5	5	15	6	8	5	19	17
7	20C01A0507	BANDE SHREYA	6	3	5	14	7	7	5	19	17
8	20C01A0509	BASHAPAKA SAI SHANKAR	5	3	4	12	3	7	5	15	14
9	20C01A0510	BATTU SRIKANTH	2	2	5	9	6	10	02	18	14
10	20C01A0511	BODA YAMUNA	23	3	5	11	3	5	5	13	12 ✓
11	20C01A0512	BOMMA SAI PAVAN	63	23	5	13	3	3	5	11	12 ✓
12	20C01A0513	BUGGA VENNELA	1	2	5	8	0	0	5	05	7
13	20C01A0514	CHALAMALA SHETTY SRINIVAS MOUTTU	6	4	5	15	5	5	5	15	15
14	20C01A0515	CHALLA SHIVAKUMAR	04	6	0	10	0	0	0	0	05
15	20C01A0516	CHAPALA PRAMOD REDDY	—								
16	20C01A0517	CHINNAPAGA VENNELA	3	3	5	11	6	5	5	16	14
17	20C01A0518	CHINTHALA SNEHALATHA	7	5	5	17	7	6	5	18	18
18	20C01A0519	D BHARATH KUMAR GOUD	4	3	5	12	7	6	5	18	15
19	20C01A0521	DODDA SAI MANIKANTA TEJA	8	3	5	16	4	9	5	18	17
20	20C01A0522	EDLA DEVENDAR	9	6	5	20	8	9	5	22	21
21	20C01A0525	ERI BATRICES JOY	7	8	5	20	10	10	5	25	23
22	20C01A0526	ERUKONDA VAMSHI	6	6	5	17	7	7	5	19	18
23	20C01A0527	G HARISH	83	4	5	17	9	9	5	23	20
24	20C01A0528	G UDAY KUMAR	0	2	5	7	8	9	3	20	14
25	20C01A0529	GANDHAM UDAY KIRAN	2	4	5	11	7	9	0	16	14
26	20C01A0530	GANNAPUREDDY NIKITHA	6	4	5	15	6	7	5	18	17
27	20C01A0531	GUGULOTH PAVAN	1	3	5	9	0	0	0	00	04
28	20C01A0532	GUMMADI VENKATA MANOJ	8	4	5	17	4	6	5	15	16
29	20C01A0533	JAJIPURAM SRIHARI GOUD	5	3	5	13	2	4	5	11	12 ✓
30	20C01A0534	JAKKIDI SIRIPRIYA	9	8	5	22	10	9	5	24	23
31	20C01A0535	JALLA POOJITHA	6	6	5	17	7	8	5	20	19
32	20C01A0537	K NAVEEN KUMAR	2	4	5	11	5	6	5	16	14
33	20C01A0538	KALIGOTA GOUTHAM	0	2	5	7	0	3	5	8	8
34	20C01A0539	KALVAKOLU BHANU PRAKASH REDDY	8	8	5	21	4	6	5	15	18
35	20C01A0540	KAMEPALLI KAVITHA	2	4	5	11	6	8	5	19	15
36	20C01A0541	KAMISHETTI LAKSHMI PRAVALLIKA	9	9	5	23	10	9	5	24	24
37	20C01A0542	KANDALA SNEHA LATHA	4	3	5	12	8	6	5	19	16

39	20C01A0545	KAREDDY SANJAY NATH REDDY	7	5	5	17	6	5	5	16	17
40	20C01A0546	KARNATI NEHA REDDY	4	2	5	11	7	4	5	16	14
41	20C01A0547	KOLA GOPI	1	3	5	9	7	10	1	18	14
42	20C01A0548	KOMMULA ABHIRAM REDDY	0	3	5	8	1	3	5	9	9
43	20C01A0549	KOMMULA ABINAYA	2	2	5	9	7	6	5	18	14
44	20C01A0550	KOTHA SAI DIXIT REDDY	7	4	5	16	10	6	5	21	19
45	20C01A0551	LATTUPALLY INDU	7	4	4	15	9	7	5	21	19
46	20C01A0552	LEDALLA VAISHNAVI	5	3	5	13	5	7	5	17	15
47	20C01A0553	CHINTHALAPALLY ARUNREDDY									
48	20C01A0554	LOKASANI SUNIL					AB				
49	20C01A0555	MADHALVEDU SAI AKASH	2	1	3	6	1	6	5	12	09
50	20C01A0556	MAHAMMAD JAVID	9	7	5	21	10	8	5	23	22
51	20C01A0557	MALGAM SRAVANI	7	5	5	17	9	6	5	20	19
52	20C01A0558	MALLEPALLI MANEESHA	3	3	5	11	4	7	5	16	14
53	20C01A0559	MAMIDI GANGAMANI	2	4	5	11	5	6	5	16	15
54	20C01A0560	MANDA NAGAMANI	1	3	5	9	4	AB			
55	20C01A0561	MANDADI DIVYA	9	6	5	20	7	9	5	21	21
56	20C01A0562	MANDAPATI RAVI	0	3	5	8	6	8	5	19	14
57	20C01A0563	MARAGOUNI SATHVIKA	3	3	5	11	8	7	5	20	16
58	20C01A0564	MD HABEEB									
59	20C01A0565	MOHAMMAD AMEERALI	6	2	5	13	1	8	5	14	14
60	20C01A0566	MOTAPOTHU SREEKANTH	8	8	5	21	7	6	5	18	20

Signature of the Faculty

Signature of the HOD

Signature of the Principal

SCIENT INSTITUTE OF TECHNOLOGY

II B.Tech I Sem (R18) MID Examination Marks : Branch CSE-B : 2021 - 22

Subject: COA Name of the Faculty: B. Ramya Sree

S.NO	H T NUMBER	NAME OF THE STUDENT	MID I				MID II				AVERAGE MARKS
			SUB	OBJ	ASGN	TOT	SUB	OBJ	ASGN	TOT	
1	20C01A0567	MUDIGONDA MANJULA	0	3	5	8	2	6	5	13	12
2	20C01A0568	MUDIREDDY NIKHITHA	3	5	5	13	4	8	5	17	15
3	20C01A0569	MUSKU LAXMI PAVANI	1	3	5	9	4	6	5	15	12
4	20C01A0570	N VENNELA	0	0	5	5	1	4	5	10	8
5	20C01A0571	NALLA ARAVIND REDDY	0	1	5	6	1	7	5	13	10
6	20C01A0572	NALLA NAGESH	0	3	5	8	2	6	5	13	11
7	20C01A0573	ODELA SUJATHA	12	5	5	17	7	7	5	19	18
8	20C01A0574	P ARAVIND	0	3	5	8	6	4	5	15	12
9	20C01A0575	P NAVEEN KUMAR									
10	20C01A0576	P VENKAT SANJAY	6	5	5	16	6	6	5	17	17
11	20C01A0577	PAGADALA SHAILAJA	0	4	5	9	2	4	5	11	10
12	20C01A0578	PAGILLA PAVANKALYAN	0	3	5	8	1	6	5	12	10
13	20C01A0579	PAPIGARI SHIVA KUMAR	2	8	5	15	5	6	5	16	16
14	20C01A0580	PATHANABOINA GAYATHRI	3	2	04	9	8	5	5	18	14
15	20C01A0581	PATHLAVATH RAGHU	0	0	05	05	5	4	5	14	10
16	20C01A0582	PEDDIREDDY POOJITHA	6	9	5	20	8	7	5	20	20
17	~~20C01A0583~~	~~PIDUGU ANISH~~									
18	20C01A0584	PIDUGU SUSHMEETHA SREE	8	8	5	21	5	6	5	16	19
19	20C01A0585	PILLI KURUVAMMA					AB				
20	20C01A0586	POLLAYIL CHRISTOPHER JENNIE	5	5	02	12	7	8	0	15	14
21	20C01A0587	PONUGOTI SAI KUMAR	7	8	5	20	7	7	5	19	20
22	20C01A0588	PULIKANTI THARUN	0	3	5	8	8	10	1	19	14
23	20C01A0590	RACHAKONDA PAVAN KUMAR	7	8	5	20	7	6	5	18	19
24	20C01A0591	RACHURI POOJITHA	6	8	5	20	6	7	5	18	19
25	20C01A0592	REDDEDDI CHAITHANYA	7	4	5	16	4	6	5	15	16
26	20C01A0593	REGALLA SAIKIRAN	2	5	4	11	4	7	5	16	14
27	20C01A0594	REKULAPALLY ASHWINI	7	6	5	18	5	4	5	14	16
28	20C01A0595	SAMANAPALLY SANTHOSH KUMAR	1	4	5	10	5	7	5	17	14
29	20C01A0596	SONTIREDDY RAKESH REDDY	4	7	5	16	9	7	5	21	19
30	20C01A0597	SUDINI KAVITHA	9	9	5	23	10	7	5	22	23
31	20C01A0598	SYED MUMTAZ ALI SABERI	5	5	0	10	8	10	0	18	14
32	20C01A0599	T YATHIN	6	3	3	6	4	6	5	15	11
33	20C01A05A0	TANKASHALA SAI BHAVANI	1	3	5	9	4	5	5	14	12
34	20C01A05A1	TEJAVATH LAVANYA	3	8	5	16	7	6	5	18	17
35	20C01A05A2	TIRUPATHI SRIDHAR	4	3	5	12	2	6	5	13	13
36	20C01A05A3	UPPERLA UDAY KUMAR	2	1	5	8	6	2	5	13	11
37	20C01A05A4	VADDULA ROHITH REDDY	4	6	4	14	6	6	5	17	16
38	20C01A05A5	VALDAS AJAY KUMAR GOUD	4	5	5	14	6	7	5	18	16
39	20C01A05A6	VALLALA NIKITHA	8	5	5	18	10	8	5	23	21
40	20C01A05A7	VATTAPU HAVILA	1	4	5	10	6	6	5	17	14
41	20C01A05A8	VEMU PRANITHA	2	2	5	9	5	8	5	18	14

42	20C01A05A9	VENNA BHARATH REDDY	1	2	5	8	8	4	3	20	14
43	20C01A05B0	VISAVARAM NAVYA	6	8	5	19	7	8	5	20	20
44	20C01A05B1	VOORE PAVAN	6	5	5	16	6	6	5	17	17
45	20C01A05B2	VUPPULA LIKITHA REDDY	3	5	5	13	4	7	5	16	15
46	21C05A0501	ALE MANASA	4	6	5	15	6	7	5	18	17
47	21C05A0502	CHINTHAKAYALA DILEEP	5	4	5	14	7	10	5	22	18
48	21C05A0503	D AASHRITHA	9	9	5	23	8	10	5	23	23
49	21C05A0504	DHANVAY LAXMI KANTH	8	3	5	16	—	—	—	—	08
50	21C05A0505	ENUGULA ASHOK KUMAR	1	3	5	9	3	4	5	12	11
51	21C05A0506	GOLLAPALLI NANDINI	7	6	5	18	7	6	5	18	18
52	21C05A0507	GUNDE PRIYANKA	7	7	5	19	6	7	5	18	19
53	21C05A0508	KARNATI MANOJ	1	5	5	11	6	7	5	18	15
54	21C05A0509	KASARLA GANESH	5	5	5	16	8	10	0	18	17
55	21C05A0510	M SANDEEP YADAV	4	7	5	16	7	6	5	18	17
56	21C05A0511	NARUM BHAGYA SRI	6	5	5	16	8	6	5	19	18
57	21C05A0512	PARRA SACHIN	0	4	5	9	1	7	5	13	11
58	21C05A0513	POLOJU DEEPIKA	2	3	5	10	6	2	5	13	12
59	21C05A0514	POTHARAJU GANESH	4	4	5	13	5	7	5	17	15
60	21C05A0515	T TRIMOKSHA	5	5	3	13	5	9	5	19	16

Signature of the Faculty

Signature of the HOD

Signature of the Principal

SCIENT INSTITUTE OF TECHNOLOGY 8625

Ibrahimpatnam, R.R. Dist. - 501 506.

INTERNAL EXAMS ANSWER BOOK

6
10

H.T. No. 20C01A0582 Year / Branch : II - CSE

Mid Exam No. : I - Mid Subject : COA

Signature of Invigilator : 15/12/2021 Date : 15/12/21

1A. Shift Microoperations:- Shift Microoperations are used for serial transfer of data/information.

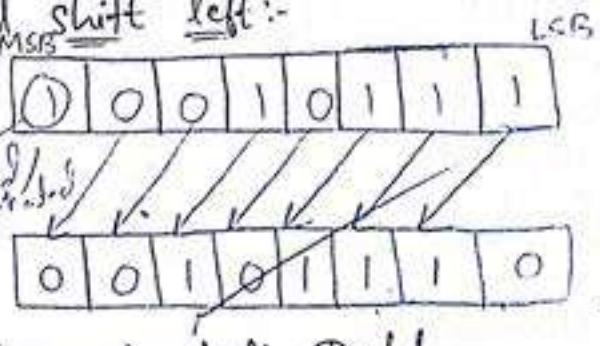
→ The contents of bits in a register are shifted from left or right.

→ There are 3 types of shift Microoperations
(i) logical Microoperation
(ii) Circular Microoperation
(iii) Arithmetic Microoperation

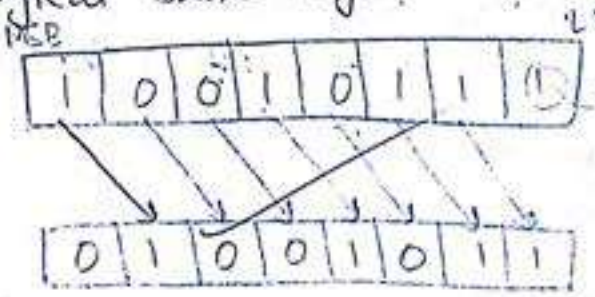
(i) Logical Microoperation:- Logical Microoperation is the one that transfers 0 through the serial input.

→ Logical Microoperation is divided into 2 types
(i) logical shift left (shl)
(ii) logical shift Right (shr)

Ex:- logical shift left:-



Ex: for logical shift Right



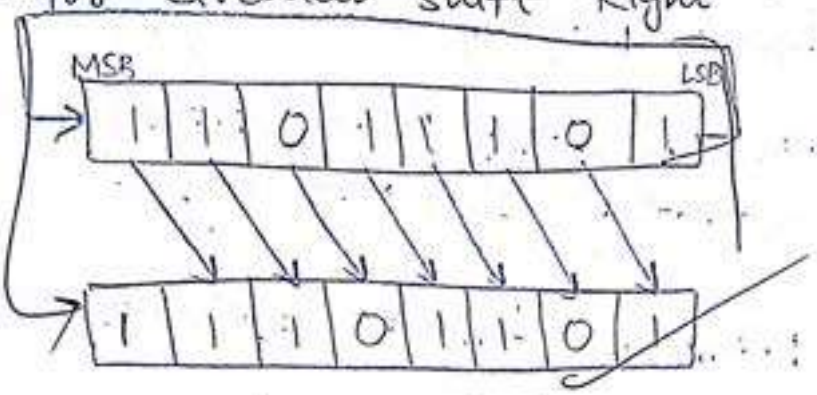
Circular Microoperation:- It circulates the bits of Registers from two ends without any loss of information.

→ In this shifting can be done from left or right.

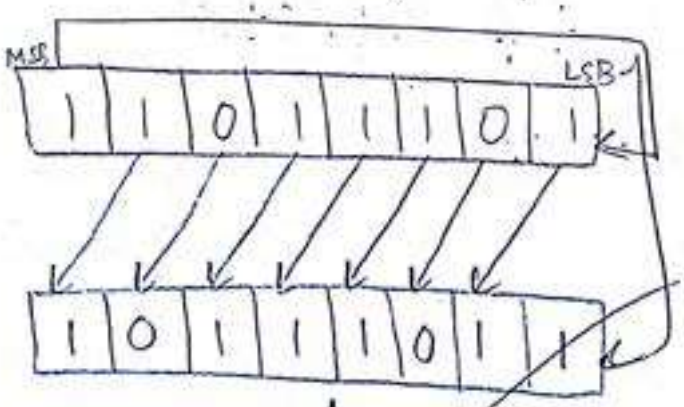
→ Circular Micro Operation is divided into two types. They are,

(i) Circular shift Right (cir)
(ii) Circular shift left (cil)

Ex: for circular shift Right



Ex:- for circular shift left

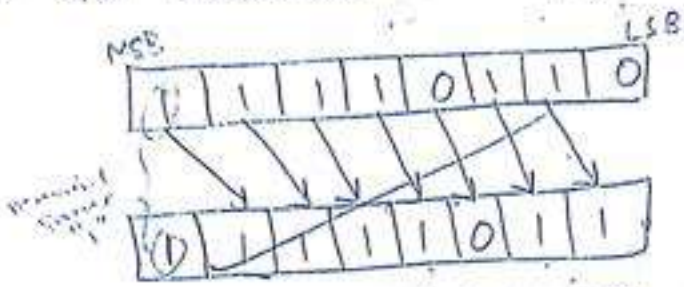


Arithmetic Microoperation:-

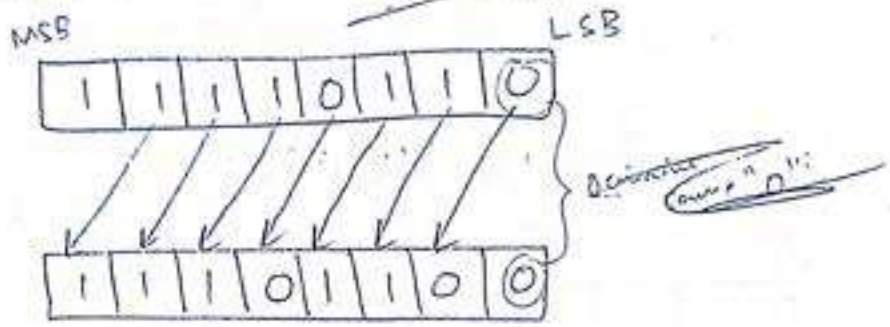
→ Arithmetic Microoperation is divided into 2 types

(i) Arithmetic shift left (shAL) – divide by 2
(ii) Arithmetic shift Right (shAR) – Multiply by 2

Ex:- for Arithmetic Shift Right :-



Ex:- for Arithmetic shift left



4B) Ans Given Memory unit = 256K × 32

$= 256 \times 1024 \times 32$

$= 2^8 \times 2^{10} \times 32$

$= 2^{18} \times 32$ → address bits

$\therefore$ Address bits = 18

There are Seven addressing modes to address the modes we need three bits (0 to 76, 111 & 000)

$\Rightarrow$ No. of Registers = 60 (Approx to 64, which can be written as 2^6)

$\therefore$ No. of Registers = 6

opcode	Mode	Register	Address
5	3	6	18

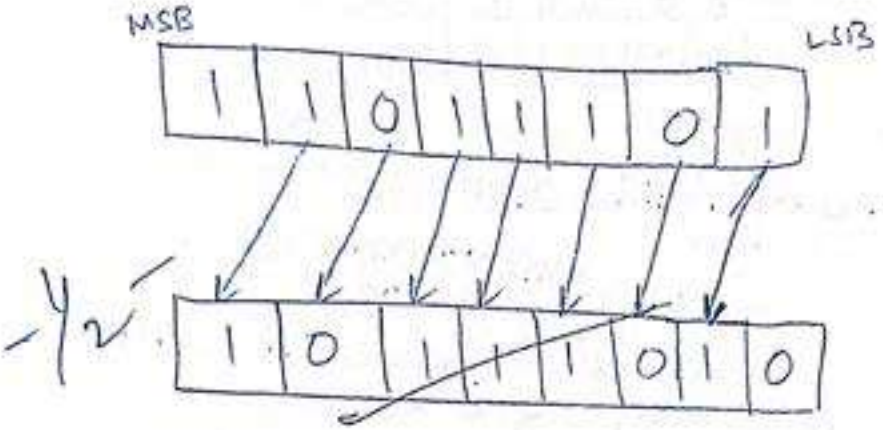
Address = 18
Registers = 06
Modes = 03

Total = 27

opcode(x) = 32 − 27
= 5

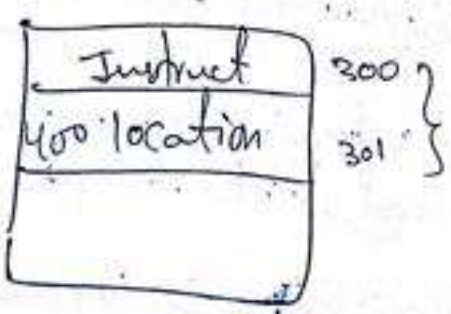
2

1B) Ans Given value R = 11011101



4

4A



a) 300
b) 301
c) 702
d) 400 + 200 = 600

1

SCIENT INSTITUTE OF TECHNOLOGY

Ibrahimpatnam, R.R. Dist. - 501 506.

INTERNAL EXAMS ANSWER BOOK

9
10

H.T. No. 2 0 C 0 1 A 0 5 3 4 Year / Branch : 2nd yr / CSE

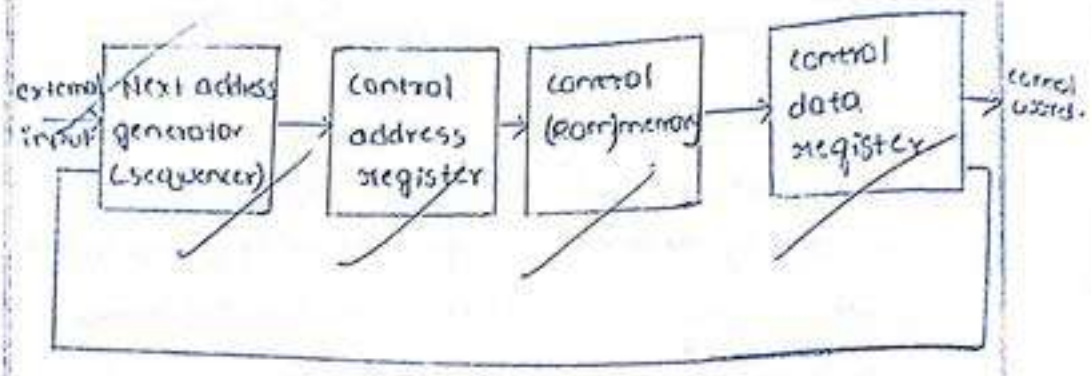
Mid Exam No. : 1st mid Subject : COA

Signature of Invigilator : Date : 15/12/2021

3) a) Ans:-

Hardwired	microprogrammed.
1) If the control signals are generated by the hard wired by using conviventional logical design techniques.	1) In this, the control information is stored in the control memory. The control memory programmed in the iteration of sequence of microoperation.
2) It involves the fixed micro instructions and fixed logical blocks and for of arrays, decoders, encoders, multiplexers etc.	2) In this the design can be change or modified by updating the micro instructions to control unit.
3) Key characteristics of Hardwired are:- 1) High speed operation 2) expensive 3) Relatively complex 4) No flexbility of adding new instructions.	3) The microinstructions of a control word (or) unit can perform one are more than one of the ~~data processor~~. 4) The control unit has to pass the address in binary value which equal to either 0 and 1.
4) If any change's required or modified then the design wiring has to change with various components.	
5) Ex:- Intel 8085, mortola 6800, RISC's CPU's.	5) Ex:- Intel 8080, mortola 6800, CISC's CPU's.

Ans:- b) micro-programmed control diagram:



1) The control word information is to be assume to be in ROM where information is stored permantely.

2) The next address register specifies the address of the next instruction.

3) The control unit contains of various address to be stored and access in the data processor.

4) Once the instructions are executed, it determine's for the next address register instruction.

5) For the next address it may be in location or may not be in sequence location and else can locate somewhere in the microinstruction.

6) For this reason, the present microinstructions are used to generate the next address instructions.

7) The next address can be in serial sequential of input-conditions.

8) So, that the address of the next address is computed by the present microinstructions.

9) The execute is done and the next address is being computed by the instructions.

2

1)
a) Shift microoperations :- In shift microoperations the data can be transfered in serial input transfer by using binary information in to left or right.

Shift microoperations are classified into 3 types:-

During shift left operation the input must transfer the serial input data to right most position.

During shift right operation the serial input must transfer to left most position.

They are '3' types:-

1) logical shift 2) circular shift 3) Arithematic shift.

logical shift :- In logical shift operation the serial input '0' through serial input.

Symbols:-

logical shift left = shl

logical shift right = shr

Ex:- $R_1 \leftarrow shl\ R1$

$R_2 \leftarrow shr.\ R1$

1 1 0 1
1 0 1 0
shl.

1 1 0 1
0 1 1 0
shr.

Circular shift operation :- In circular shift operation is also called as rotate shift as it rotate to both the end without lost of information.

Symbols:-

circular shift left left : cil

circular shift right : cir

(1) 1 1 1 0

1 1 0 1

cil

1 1 1 0

0 1 1 1

cir.

Arithmatic shift :- In arithmatic shift the signed is of binary value go through left or right.

In arithmatic shift left the signed is multipled by 2.

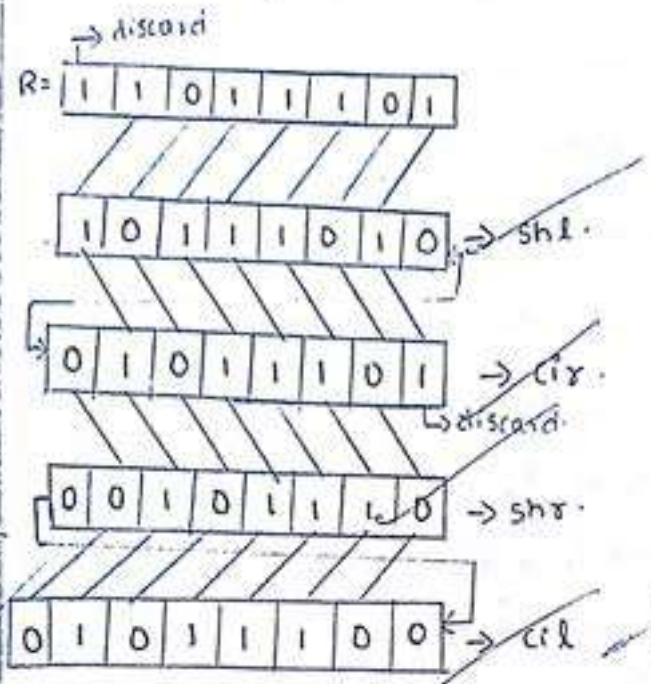
In arithmatic shift right the signed value is divided by 2.

In arithmatic shift the signed value should not change because it remains same by multiply or divided by 2.

R_{n-2}	R_{n-1}	- - - - -	R_1

(3) Ans:-

B) R = 11011101.



Extra

23

4) a) Ans:-

Given location = 300

rgtrs = 200.

Addres = 400..

(i) Direct :- In this mode, the operand of the address is equal to instruction address.

Direct = 400

(ii) Immediate :- In this mode, the operand is specified in instructions itself.

That means, it has operand mode rather than address field.

Immediate = 301 ~~400~~.

(iii) Relative Address :-

PC = 301 + 400

= 702.

Scanned with CamScanner

(d) Register indirect :- In this mode, the operand address will the CPU's whose contents will be register address.

Register indirect = 200.

(e) Index Register :- In this mode, the Index register content is added to address field to obtain address of the next field address.

Index Register = 200 + 400
= 600.

2

4
(B) Addressing mode = 256K × 32
Ans :-
= 256 × 1024
= $2^8 \times 2^{10}$
= 2^{18}.

∴ Addressing ~~mode~~ = 18 bits.

(ii) There are seven addressing mode. Therefore there are 3 bits.

(iii) Register are 60 approximately equal to 64

$64 = 2^6$

∴ No. of Register = 6.

(iv) No. of operations code = 5

2

18
16
3
22
15
32

SCIENT INSTITUTE OF TECHNOLOGY 11300

Ibrahimpatnam, R.R. Dist - 501 506.

INTERNAL EXAMS ANSWER BOOK

4	
10	

H.T. No. 2 0 C 0 1 A 0 5 5 8 Year / Branch: 2nd CSE

Mid Exam No.: 2 Subject: COA

Signature of Invigilator: Date: 16/02/2022

9) overflow :- when two number of 'n' digits each are added and the sum of overflow occurs 'n+1' digits so we can say that overflow occurs.

→ During numbers addition. overflow detected from the overflow of the occures.

signed :- MSB or left most bit over-flow with respected from the overflow numbers are signe, whose and are equal

(+70) + (+80) and (-70) + (-80)

+70 = 0100011 0 +80 = 0101000

-70 = 1011101
-80 = 1010111
1010100
+1
1010100

+ 70 = 01000110

+ 80 = 0101000

+ 70 = 1011101
1010111
+ 80 = 1001000

01000110
10101111
+1

1011101
+1
1011100

10101111
01000110

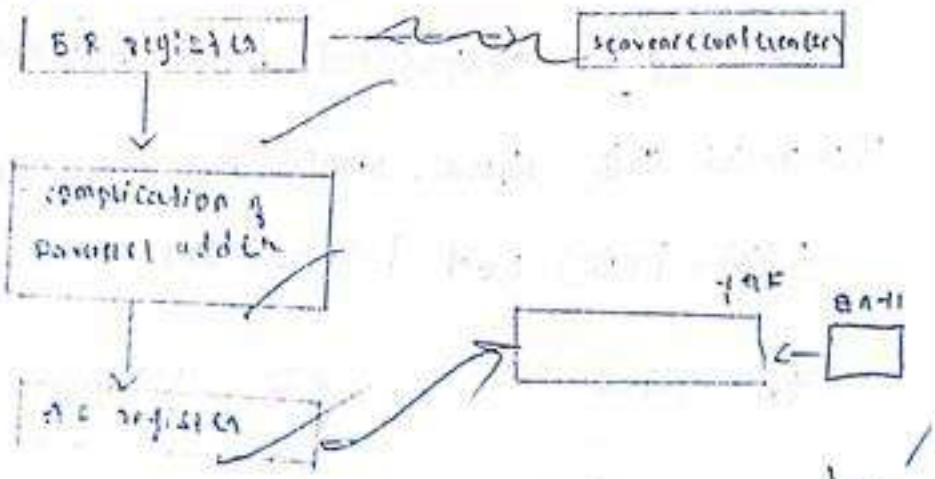
= 23

10101111
+1
10100000

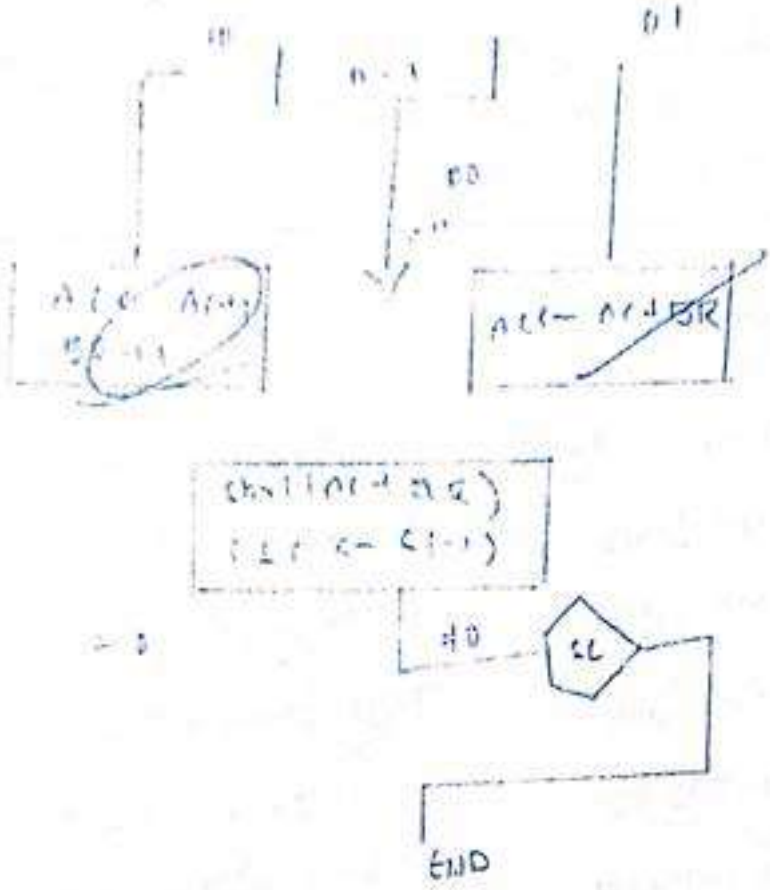
Overflow occurs = 1011100

overflow occures = 10100000

3) Multiplication algorithm with flowchart.



multiply
multiplication in BR
multiplier in QR


END

SCIENT INSTITUTE OF TECHNOLOGY 1527.

Ibrahimpatnam, R.R. Dist - 501 506.

INTERNAL EXAMS ANSWER BOOK

H.T. No. 9 0 C 0 1 A 0 5 9 9 Year / Branch : CSE (A)

Mid Exam No. : 2 Subject : Chem COA

Signature of Invigilator : Date : 16/2/2022

7

10

1)

Isolated IO

1) Def: A method to perform I/O operations b/w the CPU and peripheral devices in a computer which uses two seperate address space for memory and I/O devices.

2) Use two seperate address space for memory and I/O.

3) All address can be used by memory.

4) Uses the same seperate instructions for I/O and memory read and write operations.

5) More efficient.

6) I/O read, I/O write seperate control lines for I/O registers memory & read and memory writes control lines too memory.

Memory mapped IO

1) Def: A method to perform input and outputs devices, operations b/w the CPU and peripheral devices in a computer which use one address space for memory and I/O devices.

2) Uses the same address space for both memory I/O devices.

3) As the memory mapped I/O uses one address space too both I/O and memory the available address.

4) Uses the same instruction for both I/O and memory operations.

5) Less efficient.

6) only one set of read & write control lines to refer either to memory to I/O.

2) overflow when two number of 'n' digits each are added and the sum occupies n+1 digits so we can say that overflow occured.

→ during unsigned number addition, overflow is defected form end carry out of MSB bit signed numbers: MSB, or left most Bit always separates the sign the overflow b/w the numbers where sign are equal.

Given:

8 bits to separate the given numbers.

+70 = 01000110

+80 = 01010000

(signed)

+70 = 01000110

-70 =

2's comp = 10111001

+1

10111011

↑

(sign) Bit.

2's comp (because -80)

80 - 10101111

+1

10110000

↑

(sign) Bit.

3

+70 = 01000110

+80 = 0101000

+150 1100100110

Observe carry after the sign Bit (1)

carry out of the sign bit (0)

} If both are unequal or not equal than overflow occurred.

If both carries are applied to Ex-OR Gate.

the output will be 1.

-70 = 1 0 1 1 1 0 1 0

+80 = 1 0 1 1 0 0 0 0

0 1 1 0 1 0 1 0

A	B	A + B
0	0	0
0	1	1
1	0	1
1	1	0

SCIENT INSTITUTE OF TECHNOLOGY

B. Tech II YEAR - I SEM (2020 Batch) R18 Regular JUNE-2022

Results Analysis

EEE : FACULTY NAMES AND SUBJECTS

S. No	SUBJECT NAME	FACULTY NAME	Registered Students	Appeared Students	Passed Students	Failed Students	% of Pass
1	EM-I	Mr.B.SRINIVAS	51	46	25	21	54.35
2	AE	Mrs.G.SWARNALTHA	51	46	22	24	47.83
3	EMF	Mr.K.SAIDULU	51	44	18	26	40.91
4	EM	Mr.MD.MOINUDDIN	51	46	16	30	34.78
5	ECA	Ms.R.THANUJA	51	46	15	31	32.61
		OVER ALL %	51	46	9	37	19.57

ECE : FACULTY NAMES AND SUBJECTS

S. No	SUBJECT NAME	FACULTY NAME	Registered Students	Appeared Students	Passed Students	Failed Students	% of Pass
1	EDC	Mr.G.NARESH	53	52	35	17	67.31
2	PTSP	Mr.P.LAXMAN	53	52	33	19	63.46
3	NATL	Mr.K.SAIDULU	53	52	24	28	46.15
4	DSD	Mr.B.BHAGAVATHI RAO	53	52	21	31	40.38
5	S&S	Ms.G.PRIYANKA	53	52	14	38	26.92
		OVER ALL %	53	52	8	44	15.38

CSE : FACULTY NAMES AND SUBJECTS

S NO	SUBJECT NAME	FACULTY NAME	Registered Students	Appeared Students	Passed Students	Failed Students	% of Pass
1	OOPS C++	Mr.SMD.SHAFFIULLA	116	111	76	35	68.47
2	DS	Mrs.K.DEEPTHI	116	110	74	36	67.27
3	COA	Ms.B.RAMYASREE	116	109	46	63	42.20
4	ADE	Mr.P.LAXMAN	116	111	45	66	40.54
5	COSM	Mrs.B.SABITHA	116	111	36	75	32.43
		OVER ALL %	116	111	20	91	18.02

I/C EXAMINATION BRANCH
SCIENT INSTITUTE OF TECHNOLOGY
I/C EXAMS

PRINCIPAL