BCS302	COMPUTER ORGANIZATION AND ARCHITECTURE	
Course Outcome (CO)		Bloom's Knowledge Level (KL)
At the end of course , the student will be able to understand		
CO 1	Study of the basic structure and operation of a digital computer system.	K_1, K_2
CO 2	Analysis of the design of arithmetic & logic unit and understanding of the fixed point and floating point arithmetic operations.	K_3, K_4
CO 3	Implementation of control unit techniques and the concept of Pipelining.	K_3
CO 4	Understanding the hierarchical memory system, cache memories and virtual memory.	K_2
CO 5	Understanding the different ways of communicating with I/O devices and standard I/O interfaces.	K_2, K_3
DETAILED SYLLABUS		3-1-0
Unit	Topic	Proposed Lecture
I	**Introduction:** Functional units of digital system and their interconnections, buses, bus architecture, types of buses and bus arbitration. Register, bus and memory transfer. Processor organization, general registers organization, stack organization and addressing modes.	08
II	**Arithmetic and logic unit:** Look ahead carries adders. Multiplication: Signed operand multiplication, Booths algorithm and array multiplier. Division and logic operations. Floating point arithmetic operation. Arithmetic & logic unit design. IEEE Standard for Floating Point Numbers	08
III	**Control Unit:** Instruction types, formats, instruction cycles and sub cycles (fetch and execute etc), micro operations, execution of a complete instruction. Program Control. Reduced Instruction Set Computer, Pipelining. Hardwire and micro programmed control: micro programme sequencing, concept of horizontal and vertical microprogramming.	08
IV	**Memory:** Basic concept and hierarchy, semiconductor RAM memories, 2D & 2 1/2D memory organization. ROM memories. Cache memories: concept and design issues & performance, address mapping and replacement. Auxiliary memories: magnetic disk, magnetic tape and optical disks. Virtual memory: concept implementation.	08
V	**Input / Output:** Peripheral devices, I/O interface, I/O ports, interrupts: interrupt hardware, types of interrupts and exceptions. Modes of Data Transfer: Programmed I/O, interrupt initiated I/O and Direct Memory Access., I/O channels and processors. Serial Communication: Synchronous & asynchronous communication, standard communication interfaces.	08

Text Books:

1. Computer System Architecture - M. Mano
2. Carl Hamacher, Zvonko Vranesic, Safwat Zaky Computer Organization, McGraw-Hill, Fifth Edition, Reprint 2011
3. John P. Hayes, Computer Architecture and Organization, Tata McGraw-Hill, Third Edition, 1998. Reference books
4. William Stallings, Computer Organization and Architecture-Designing for Performance, Pearson Education, Seventh edition, 2006.
5. Behrooz Parahami, "Computer Architecture", Oxford University Press, Eighth Impression, 2011.
6. David A. Patterson and John L. Hennessy, "Computer Architecture-A Quantitative Approach", Elsevier, a division of reed India Private Limited, Fifth edition, 2012.
7. Structured Computer Organization, Tanenbaum, PHI

CIA IMPORTANT QUESTIONS

UNIT – 1

1. Explain the functional units of digital system and their interconnections in detail.

   Or

   Explain the von neumann architecture of computer system in detail. Also give advantage and disadvantage of this architecture as compared to other architecture.
2. Draw the block diagram of computer CPU showing all the building basic block and hence explain the same in detail. Also give suitable reasons as how such arrangement can work as computer can work as a computer if connected to memory, I/O interface.
3. What is Bus? Draw the general structure of system bus and hence explain the different types of system bus with their relative functions.
4. What are the different bus structures available in computer system, explain in details with necessary block diagram required and also mention advantage and disadvantage of each architecture in detail.
5. What do you mean by term bus arbitration? Why we need bus arbitration? Explain different mechanism of bus arbitration in detail with required block diagram, also mention relative advantage and disadvantage of each mechanism.

   Or

   Compare and contrast different mechanism of bus arbitration in tabular format.
6. What do mean by RTL? How it is useful in computer system? Explain different features and components of RTL in detail.
7. What is memory transfer? What are the different registers associated for memory transfer? Discuss.
8. What do you mean by Bus transfer? Draw a diagram of bus system in which it uses:
   a. Three state buffers and a decoder.
   b. MUX which has 4 register of size 4 bit each.
9. Explain general purpose register based organization in detail.
10. What is stack? Give the organization of register stack with all necessary elements and explain the working of PUSH and POP operations in detail.
11. What is memory stack? Compare register stack and memory stack. Explain the role of memory stack in managing subroutines.
12. Explain an accumulator based central processing unit organization with block diagram.
13. What do you mean by term processor organization? Explain different types of processor organization in detail.
14. What do you mean by term addressing modes? Why do we need addressing modes? How addressing modes is significant for referring memory? List and explain different types of addressing modes in detail.
15. Define the term control word with an example, also explain the general format of control word and hence explain the same.

UNIT – 2

1. With the help of block diagram explain sequential arithmetic and logical unit in detail and hence explain how we can implement and design such ALU unit.
2. Design a 4-bit carry look ahead adder and explain its operation with an example.

   Or

   Explain in detail the principle of carry look head adder and design 4-bit CLA adder.
3. With the help of flowchart explain booth's multiplication algorithm for 2's complement number, and hence multiply the following numbers using the same:
   a. 15 * 13 b. -14 * 7 c. 11 * -8 d. -12 * -9
4. Explain booth's algorithm with its hardware implementation.
5. Explain array multiplier method with the help of example.
6. With the help of flowchart explain restoring and non-restoring division operation, and hence divide the following number using both method restoring and non-restoring:
   a. 19 / 12 b. 19 / 4 c. 15 / 4 d. 12 / 8
7. Draw the data path of sequential n-bit binary divider.
8. How floating point numbers are represented in computer, also give IEEE 754 standard for 32-bit and 64-bit floating point number format, hence represent $(-1448.215)_{10}$ and $(1460.125)_{10}$ in single precision and double precision format.
9. Draw a flowchart for adding and subtracting two fixed point binary numbers when negative numbers are signed 1's complement presentation.

UNIT – 3

1. What is an instruction in the context of computer organization ? Explain the purpose of the various elements of an instruction with the help of a instruction format.
2. Describe the types of instructions on the basis of address fields used in the instruction and hence evaluate the arithmetic statement X = A*[B+C*D+E(F*G+H)] using a general register computer with three address, two address, one address and zero address instruction format program to evaluate the expression.
3. Define instruction cycle and divide instruction cycle into sub cycles with the help of diagram, explain the sequence in which sub cycles are executed.
4. Write the steps in fetching a word from memory. Differentiate between a branch instruction and call subroutine instruction.
5. What are the different categories of micro-operations or instructions that may be carried out by CPU? Explain each category of micro-operations giving one example for each.
6. What is CISC ? Explain its characteristics.
7. What is RISC ? Explain its various characteristics.
8. Differentiate among the following:
   a. RISC and CISC based microprocessors.
   b. Hardwired and Micro-programmed control unit
   c. Horizontal organization and Vertical organization (Note: These are technique to group control signal)

9. What do you understand by hardwired control unit? With the help of block diagram explain the working of hardwired control unit in detail with relative advantages and disadvantages, also explain various methods to design hardwired control unit in detail.
10. What do you understand by micro-programmed control unit? With the help of block diagram explain the working of micro-programmed control unit in detail with relative advantages and disadvantages.

    Or

    With the help of necessary diagram explain the two technique of generating control signals.

11. What is a micro-program sequencer? With block diagram, explain the working of micro-program sequencer.

    Or

    With the help of block diagram explain address selection for control memory.
12. Explain the concept of vertical and horizontal multi-programming in detail.
13. Write Short notes on the following:
    a. Micro-operation b. Micro-code c. Micro-program d. Control memory
    e. Wide branch addressing f. Pre – fetching micro instruction g. Micro – instruction
14. What do you mean by term pipelining and how the idea of pipelining used in a computer? Explain different types of pipelining, and also define different modes of pipelining available.
15. What are the various pipeline performance measures? Explain in detail.

UNIT – 4

1. Explain concept of memory and hence describe memory hierarchy in detail.
2. Explain semiconductor RAM. Enlist the types of semiconductor memory and hence give a structure of commercial 8M × 8 bit DRAM chip.

   Or

   Explain dynamic RAM and static RAM and hence give the structure of commercial 8M×8 bit DRAM chip.
3. Explain 2D, 2.5D memory organization.
4. What is ROM ? Explain the types of semiconductor based ROM memories.
5. How main memory is useful in computer system? Explain the memory address map of RAM and ROM.
6. Write short note on cache memory. Discuss the design issues in cache design and hence explain how does it affect the performance of the computer system and how the performance of cache memory is measured?
7. What is meant by cache mapping? What are different types of mapping? Discuss different mapping techniques with examples.
8. Explain the different method of writing into cache in detail.
9. Explain the term locality of reference and hence differentiate among spatial locality and temporal locality.
10. Explain various page replacement algorithms in detail with example, hence explain the concept of belady's anomaly and also determine which page replacement algorithm suffer from this problem.
11. Explain auxiliary memory. What are the commonly used auxiliary memories?
12. Explain the concept of virtual memory in detail, also determine the methods to implement virtual memory.
13. What do you mean by CAM? Explain its major characteristics.

14. Explain the following memory schemes, also discuss their need and working :
    a. Interleaved memory
    b. Associative memory
15. Discuss the conceptual organization of a multilevel memory system used in computers.

UNIT – 5

16. What do you mean by the term I/O Interface? Enlist different function of I/O interface in detail. Also give suitable reason for the need of I/O interface in transferring information between internal storage and external I/O devices.
17. Write short notes on :- 1. Peripheral Devices. 2. I/O bus and 3. I/O Commands.
18. What do you mean by I/O processor? Have I/O processor interconnection with CPU? Justify your answer with necessary block diagram required.
19. Explain the term interrupt in detail with it's classification, also write the sequence that take place when an interrupt occur.
20. When a device interrupt comes how does the processor determine which device has issued the interrupt, hence determine how system resolve the priority of interrupt?
21. With the help of flow diagram explain programmed I/O method and interrupt initiated I/O method for controlling input-output operation in detail.
22. Explain DMA based data transfer method in detail, hence explain how DMA is able to transfer data without intervention of CPU.
23. Differentiate among the following:
    a. Isolated I/O & Memory Mapped I/O
    b. Vectored Interrupt & Non-Vectored Interrupt
    c. Serial transmission & Parallel transmission
    d. Synchronous data transfer & Asynchronous data transfer
24. Write short notes on the following:
    a. Input Output I/O Processor
    b. Input Output I/O Port
    c. Input Output I/O Channel
25. What do you mean by asynchronous data transfer? With the help of block diagram and timeline diagram explain different method of asynchronous data transfer in detail.
26. What do you mean by serial communication? Explain different mode of serial communication in detail.
27. Answer the following as asked:
    a. Why Read and Write control line in a DMA Controller are bi-directional?
    b. Describe cycle stealing in DMA
    c. What is the use of modem in synchronous communication?

COA ALL DIAGRAM

Draw the neat and labelled diagram of the following topics as mentioned :-

1. Block Diagram Of Von – Neumann Architecture.
2. Block Diagram Of Computer CPU.
3. Architecture Of Single Shared Bus.
4. Architecture Of Multiple Shared Bus.
5. Block Diagram To Show Implementation Of :-
    a) Daisy Chaining Method.
    b) Polling Method.
    c) Independent Method.
6. Block Diagram To Show Organization Of Stack And Also Show Implementation Of Stack Operation.
7. Block Diagram Of General Register Based Organization.
8. Block Diagram Of Accumulator Based CPU Organization.
9. Booth's Multiplication Algorithm Flow Chart.
10. Block Diagram To Show Hardware Implementation Of Signed Binary Multiplication For Booth's Algorithm.
11. Restoring Division Algorithm Flow Chart.
12. Non Restoring Division Algorithm Flow Chart.
13. Block Diagram To Show Hardware Implementation Of Binary Division.
14. Block Diagram To The Data Path For N-Bit Binary Adder
15. Digital Circuit To Represent 2*2 Bit Combinational Array Multiplier.
16. Block Diagram Of Control Unit.
17. Block Diagram Of 4-Bit Carry Look Ahead Adder.
18. Flow Chart For Adding And Subtracting Two Fixed Point Binary Numbers.
19. Logic Diagram Of Carry Look Ahead Adder.
20. Flow Chart To Show Instruction Cycle With Interrupt.
21. Flow Chart To Show Instruction Cycle Without Interrupt.
22. Block Diagram Of CISC.
23. Block Diagram Of RISC.
24. Block Diagram Of Hardwired Control Unit.
25. Block Diagram Of Micro-Program Control Unit.
26. Block Diagram Of Micro Program Sequencer.
27. Block Diagram To Show Pipelined System.
28. Space Time Diagram For Four Segment With Five Task.
29. Space Time Diagram For Six Segment With Eight Task.

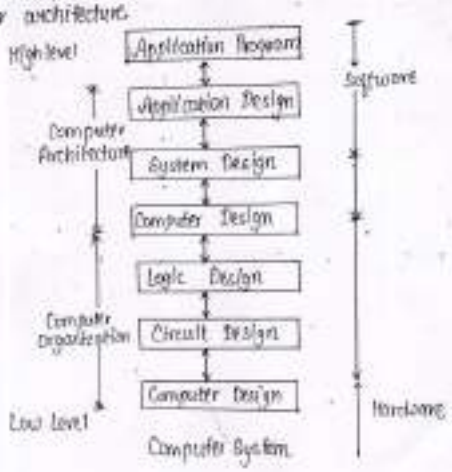
30. Diagram To Show Memory Hierarchy.
31. Block Diagram Of Cache Memory.
32. Block Diagram To Implement Concept Of Split Cache.
33. Diagram To Implement The Concept Of:-
    a) Associative Mapping.
    b) Direct Mapping.
    c) Set Associative Mapping.
34. Diagram To Show Concept Of Demand Paging.
35. Block Diagram To Show Organization Of Virtual Memory.
36. Diagram To Show Virtual Memory Concept Using Paging.
37. Flow Diagram Of Segment Translation.
38. Flow Diagram Of Address Translation.
39. Flow Diagram To Show Classification Of Memory.
40. Diagram To Show Mechanical Structure Of Magnetic Disk.
41. Diagram To Show Structure Of Magnetic Tape.
42. Block Diagram Of CASE.
43. Diagram To Show 2D Memory Organization.
44. Diagram To Show 2.5D Memory Organization.
45. Block Diagram To Show Structure Of I/O Interface.
46. Block Diagram To Show Communication Of CPU And IOP Processor.
47. Block Diagram Of Source Initiated Strobe Control And Destination Initiated Strobe Control.
48. Timing Diagram Of Source Initiated Strobe Control And Destination Initiated Strobe Control.
49. Block Diagram Of Source Initiated Handshaking And Destination Initiated Handshaking.
50. Timing Diagram Of Source Initiated Handshaking And Destination Initiated Handshaking.
51. Block Diagram To Implement Concept Of Daisy Chaining Priority.
52. Flow Diagram Of Programmed I/O Mode Of Transfer.
53. Flow Diagram Of Interrupt Initiated Mode Of Transfer.
54. Block Diagram Of DMA.
55. Block Diagram Of DMA Controller.

UNIT:-01

Introduction

Computer Architecture:- Computer Architecture refers to the design of internal workings of a computer system, including the CPU, memory and the other hardware. It involves decisions about the organisation of the hardware such as instructions set architecture, the data path design and the control unit design.

Computer Organisation:- Computer organization refers to the operational units and their interconnections used implement the architecture specification. It deals with how the components of a computer system are arranged and how they interact to perform the required operation. It concerns with the physical implementation of computer architecture.



Difference b/w Computer Organization and Architecture

Computer Architecture		Computer Organisation
Computer architecture describes what the computer does.	1.	The organisation describes how it does it.
Computer Architecture deals with the functional behaviour of a computer system.	2.	It deals with structural relationship.
Computer Architecture deals with higher level design issues.	3.	It deals with low level design issues.
As a programmer you can view architecture as a series of instructions, addressing modes and registers.	4.	The implementation of the architecture is called organization.
For designing a computer, its architecture fixed first.	5.	For designing a computer unit organization is decided after its architecture.
Computer architecture contains digital functions such as instruction set, registers, data types and addressing modes.	6.	The computer organization consists of physical units like circuit design, processors and adders.

Digital Computer Architecture

Functional Units of Digital System

The digital computer consists of five functionally units:- Input, Memory, Arithmetic and Logic, output and control unit.

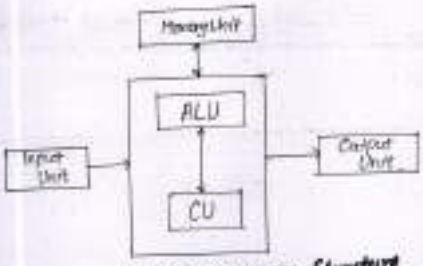


fig. Von-Neumann Structure

Input Unit:- The input unit accepts the digital information from user with the help of input devices such as keyboard, mouse, microphones etc.

Memory Unit:- The memory unit is used to store programs and data the data. Usually two types of memory devices are used in computer system.

1. Primary storage Memory device
2. Secondary storage Memory device

ALU:- Arithmetic and Logic Unit is responsible for performing arithmetical operation such as Add, subtract, multiplication and Division, logical operations like AND, OR, NOT.

Control Unit:- The control unit co-ordinates the control signals or timing signals to determine when a given action is to take place.

Output Unit:- Output Unit sends the processed result to the user using output devices such as monitor, printer etc.

Connections b/w the Processor and the main Memory

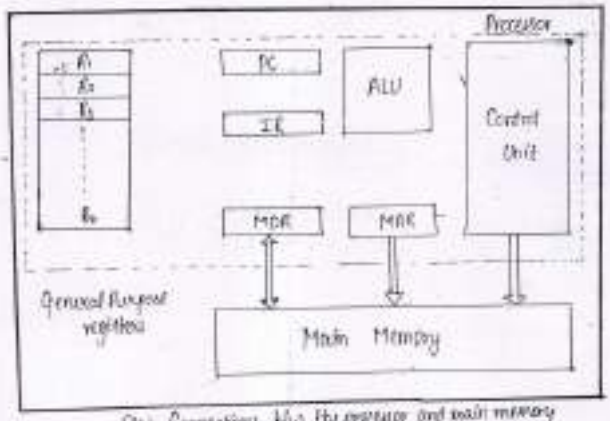


Fig:- Connections b/w the processor and main memory

To perform execution of instruction, the processor contains a no. of registers used for temporary storage of data and some special function register as shown in figure.

The special function register include program counter, instruction register, memory address register, and memory data register.

1. **Program Counter:-** A program is a series of instructions stored in the memory. These instructions tell the CPU exactly how to get the desired result.
   - The sequence of instructions execution is monitored by the program counter (PC).
   - It keeps track of which instruction is being executed and what the next instruction will be.
2. **Instruction Register (IR):-** It is used to hold the instructions that is currently being executed.
3. **Memory Address Register & Memory Data Register:-**
   - These registers are used to handle the data transfer between the main memory and the processor.
   - The MAR holds the address of the main memory to or from which data is to be transferred.
   - The MDR is also known as MBR (Memory Buffer Register) contains the data to be written into or read from the addressed word of the main memory.
4. **General Purpose Register:-** These are used to hold the operands for arithmetical and logical operation and/or used to store the result of the operation.

Buses:- A group of wires, called bus is used to provide necessary signals for communication between modules.

- Bus is a shared transmission medium.-
  1. Must only be used by one device at a time.
  2. When used to connect major components of computer system (CPU, memory, input output) is called a System bus.

- The system bus is separated into three functional groups:-
  1. Data Bus
  2. Address Bus
  3. Control Bus

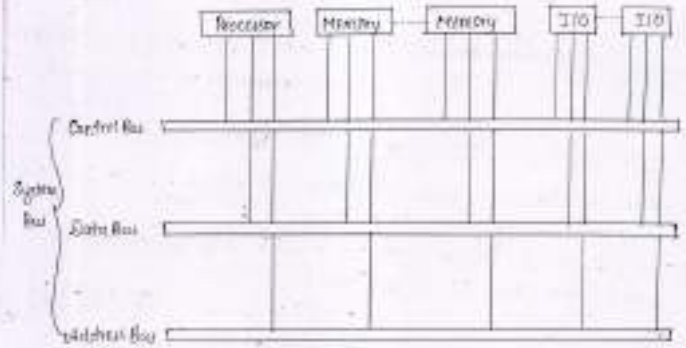


Fig:- Bus Interconnection System.

Data Buses:- Data bus lines are bidirectional. CPU can read data on these lines from memory or from port as well as send data out on these lines to a memory location or to a port.

Address Buses:- It is a unidirectional Bus width determines the maximum possible memory capacity of the system.

Control Buses:- Control access to and use of the data and address lines. Control lines include:-

* Memory read and memory write.
Memory se
* Input/Output read and Input/Output write
* Transfer acknowledgement
* Bus Request
* Bus Grant

Types of Bus Structure

1. Single Bus Structure :- In single bus structure, address bus, data bus and control bus are shown by a single bus called system bus.

In single bus structure, all units are connected to common bus system called system bus. However, with the single bus only two units can communicate with each other at a time.

The bus control lines are used to multiple request for use & usage of the bus.

The main advantage of single bus structure is its low cost and its flexibility for attaching peripheral devices.

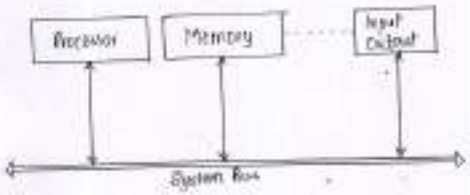


Fig. Single Bus Structure

Multiple Bus Structure:- Large number of devices on a single bus will cause performance suffer due to propagation delay and the bus may become a bottle neck.

Nowadays, the data transfer rates for video controllers and network interfaces are growing rapidly. The need of high speed shared bus is impractical to satisfy with a single bus. Thus, most computer system uses multiple buses. These buses have hierarchical structure as shown in figure:

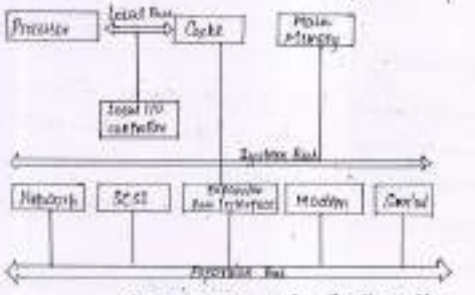


Fig. 1 Traditional Bus Configuration

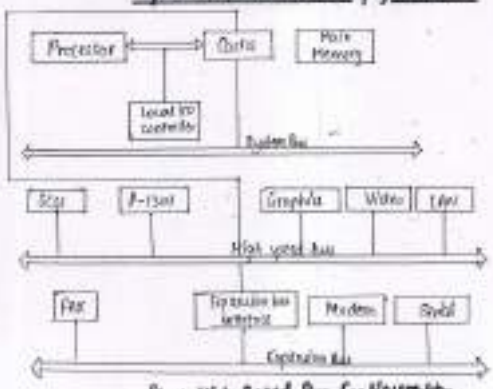
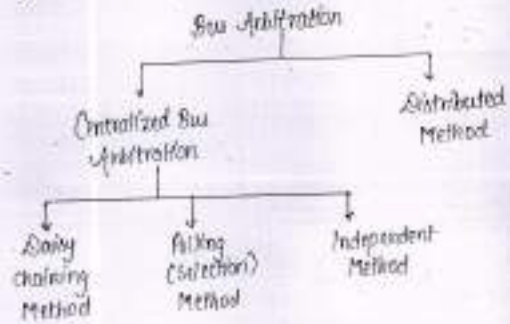


Fig. 2 High Speed Bus Configuration

Bus Arbitration:- The device that is allowed to initiate data transfer on the bus at any given time is called the bus master.

Bus arbitration is the process by which the next device to become the bus master is selected and bus mastership is transferred to it. The selection of bus master is usually done on the priority basis.

Approaches to Bus Arbitration



Centralized Bus Arbitration:- In centralized bus arbitration, a single bus arbiter performs the required arbitration. The bus arbiter may be the processor or a separate controller connected to the bus.

Centralized bus approach is implemented by three methods:

Daisy Chaining Method:- It is a simple and cheaper method. All masters make use of the same line for bus request.

In response to a bus request, the controller sends a bus grant if the bus is free. The bus grant signal serially propagates through each master until it encounters the first one that is requesting access to the bus.

* This master blocks the propagation of the bus grant signal, activate the busy line and gains the control of the bus. Therefore, any other requesting module will not receive the grant signal.

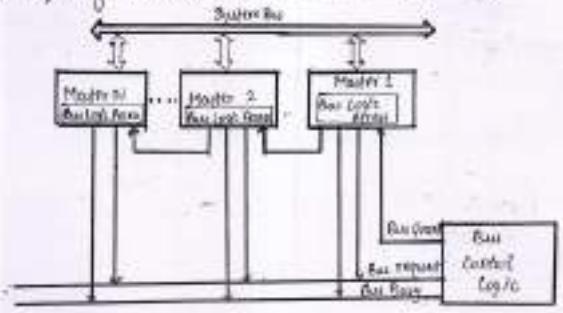


Fig. Daisy Chaining Method

Advantages:-
* It is a simple and cheaper method.
* It requires the least no. of lines and this no. is independent of the no. of masters in the system.

Disadvantages:-
* Failure of anyone master causes the whole system to fail.
* The priority of master is fixed by its physical location.
* The propagation delay of bus grant signal is proportional to the no. of masters in the system. This makes arbitration time slow and hence limits the no. of masters in the system.

Polling Method:- In this, the controller is used to generate the addresses for the masters.

- No. of address lines required depends on the no. of masters connected in the system. For eg: there are eight (8) masters connected in the system, at least 3 address lines are required.
- In response to a bus request, controller generates a sequence no. of master address. When a requesting master recognizes its address, it activates the busy line and begins to use the bus.

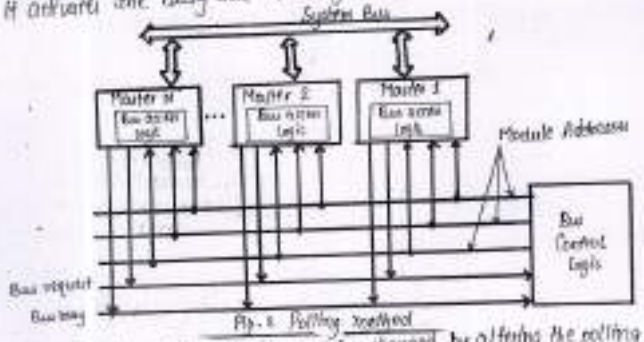


Fig:- Polling method

Advantages:-
- The priority can be changed by altering the polling sequence stored in the controller.
- If one module fails, entire system does not fail.

Disadvantages:- No. of address lines are variable. This may affect cost. This may require more time for configuration.

3. Independent Method:- In this scheme, each master has a separate pair of bus request and bus grant lines and each pair has a priority assigned to it.

- The controller selects higher priority request first and activates signal for it.

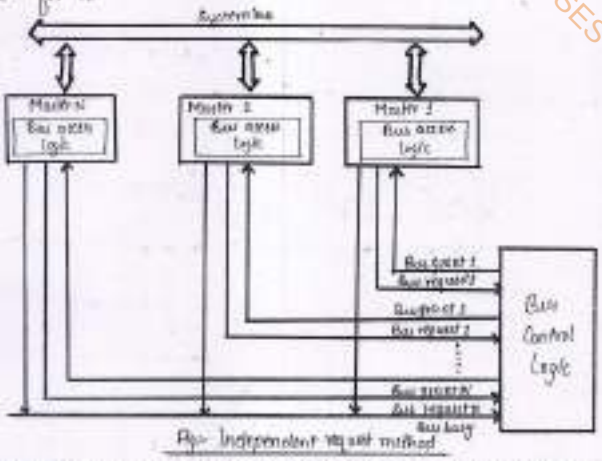


Fig:- Independent request method

Advantages:- Due to separate pairs of the bus request and bus grant signals, arbitration is fast and independent of the no. of masters in the system.

Disadvantages:- It requires more bus request and grant signals (i.e. 2×N lines for N modules).

Distributed Bus Arbitration

In distributed bus arbitration, all devices participate in the selection of the next bus master. In this scheme each device on the bus is assigned a 4-bit identification number. The no. of bits used for identification depends on the no. of devices connected on the bus. When one or more devices request for the control of bus, they initiate start-of-arbitration signal and place their 4-bit identification number on bus scheme, the device having highest no. has highest priority.

Register Transfer Language

The microoperation is an elementary operation performed in a computer system. When the data transfer of any microoperation occur in between registers it is known as register transfer.

The symbolic notation is used to describe or represent any microoperation belongs to register transfer forms a statement or system is known as register transfer language.

Example:-

a. $R_1 \leftarrow R_2$

b. $R_2 \leftarrow R_1 \oplus R_3$

R_1 — Destination Register

R_3 — Source Register

∴ Features of RTL :-

1. The presentation of digital function in register transfer logic is very user friendly.
2. Usage of register is easier instead of flip-flops and gates.
3. It describes the information flow and processing take among the data stored in the registers in a concise and precise manner.

∴ Basic Components of RTL :-

- The Register Transfer Logic Method uses four basic components to describe digital system. These are as follows:-

1. **Registers & their functions:-** Registers are the electronic circuit where information can be stored. The register includes all its counter parts such as shift register, counters and many more.
2. **Information:-** The information stored in the registers. The information may be binary nos., alphanumeric characters, control information or any other binary encoded information.
3. **Operations:-** The operations perform on the information stored in the registers. The operation performed on the data are called microoperations. The operation may be arithmetic, or logical operations.
4. **Control Functions:-** The control functions that activate operations and control the sequence of operation. The control function is basically, a binary variable, when it is logic one, then it initiates the operation otherwise it deactivates the operation.

Bus Transfer

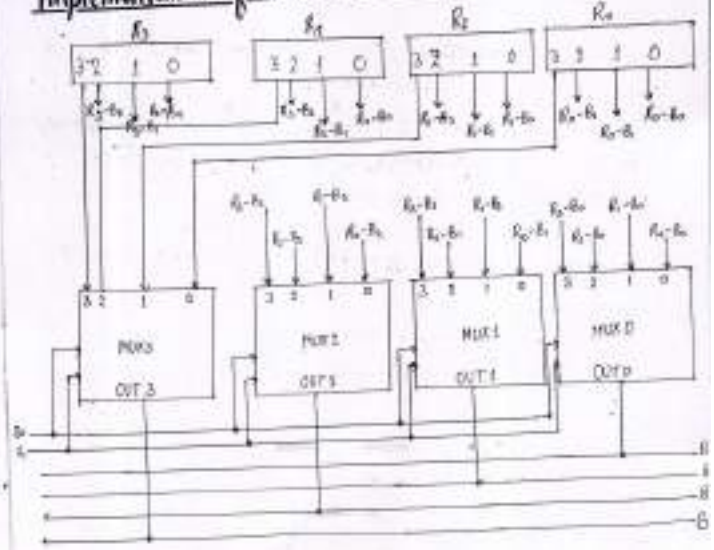
In a digital computer has many registers and it is necessary to provide data path between them. To transfer information from one register to another. The appropriate lines are used between each register, there will be excess no of wires and controlling of these wires make circuit complex.

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

The common bus scheme implemented into two ways -

1. Using multiplexer
2. Using Tri-State Buffer

Implementation of Common Bus Using Multiplexer



The figure shows the implementation of common bus system for 4 registers using multiplexers. Each register has four bits numbered 0 through 3 and they are routed through multiplexer to the common bus. Here, four multiplexers are used to select one of four bits of the source register. Each multiplexer has four input lines, data select lines, one output line. The four input lines of multiplexer zero are connected to the bit zero, outputs of four registers such that bit zero of register 0 is connected to input zero, bit zero of register one is connected to input one, bit zero of register two is connected to input two and bit zero of register three is connected to input three. Similarly inputs for MUX 1 are connected to bit 1 outputs, inputs for MUX 2 are connected to bit 2 and inputs for MUX 3 are connected to bit 3, outputs of register 0 through 3.

To avoid the complexity of the diagram only input connections for MUX 3 are physically shown.

The two selection lines S_1 and S_0 are connected to the selection inputs of all 4 multiplexers. These lines choose the 4-bits of one register and transfer them into the 4-line common bus through outlines. When $S_1 S_0 = 00$ the input zero of all four multiplexers are selected and applied to the output to transfer them on the common bus.

S_1	S_0	Selected Register
0	0	Register 0
0	1	Register 1
1	0	Register 2
1	1	Register 3

Table for Selection Table.

Memory Transfer

A memory is a collection of storage cell. Each cell stores one bit of information, the memory stores binary information in groups of bits called word. To access information from a particular word from the main memory to access, each word has different address.

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.

The data transfer between memory and processor takes place through the use of two processor registers usually called MAR (Memory Address Register) and Memory Data Register (MDR).

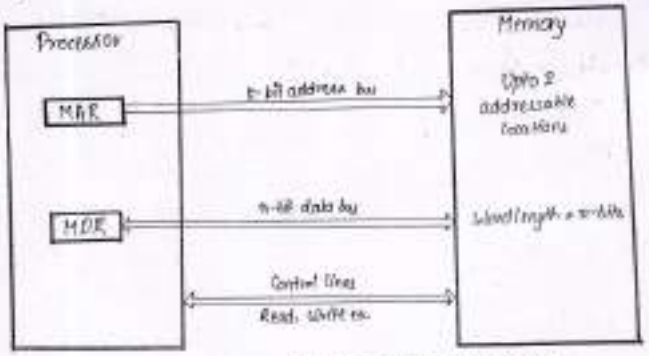


Fig. 1.7.16 Connection between memory and processor

In read operation, the address of the memory word is specified by address register, AR and the data word read from the memory is loaded into data register, DR, otherwise up this read operation is represented as :

Read : DR ← M[AR]

The right operation transfers the contents of a data register to a memory word selected by the address in the address register. The right operation is represented as :

Write : M[AR] ← DR

Processor Organization

There are three types of processor organization :

1. General Processor Organization (Accumulator based)
2. General Register Organization
3. Stack Organization

1. General Processor Organizations The general processor organization includes three major logic devices :

1. ALU
2. Registers
3. Control Unit

The internal data bus is used to transmit data between these logic devices.

1. ALU: It is major logic device contains the processors data processing logic.

The internal data bus of the processor is connected to the data input of ALU through the temporary register and the accumulator. The output is connected to the internal data buses.

The ALU works on either one or two data words depending on operation. It performs logical and arithmetical operation.

The output is stored in accumulator (A).

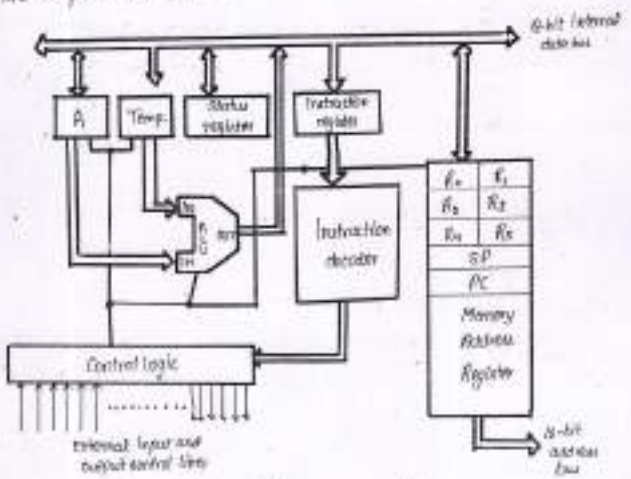


Fig. General Processor organization

2. Registers: The basic registers found in the most of the processors are the accumulator, program counter, stack pointer, the status register, the general purpose register, the memory address register, the instruction register and the temporary data register.

3. Control Logic: The control logic is responsible for working of all other parts of the processor together. It maintains the synchronization in operation of different parts in the processor.

The control logic receives the signal from instruction decoder which decodes the instructions stored in the instruction register.

The control logic generates the control signal necessary to carry out this instruction.

Internal Data Bus: The internal data bus connects the different parts of processor together and it enables the communication between these parts. The data transfer through the internal data bus is controlled by control logic.

2. General Register Organization

It shows that two registers are selected and two data flow between register and ALU take place.

A decoder is used to select a particular register. The output of each register is connected to two multiplexers from the two buses A and B.

The selection line in each multiplexer select the input data for a particular bus.

The A and B buses form the two inputs of an ALU.

The operation select line decides the micro-operation to be performed.

The result of the microoperation is available at the output bus. The output bus is connected to the inputs of all registers, thus the selecting a destination register, it is possible to store the result in it.

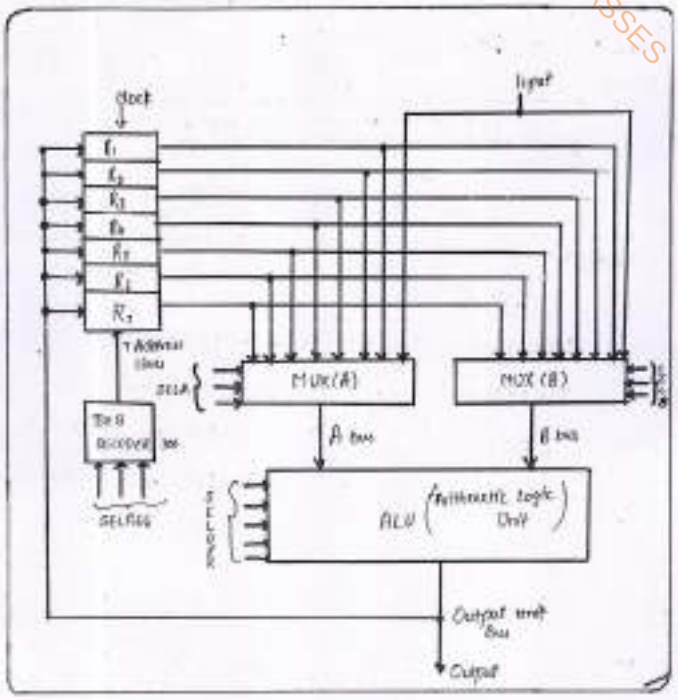


Fig. 2.11 General bus organization for five registers

Control Bus Word: The binary conversion of any micro-operation is known as control word. The combined value of a binary selection is specified control word.

Collection of 14-binary bits which specify any microoperation known as control word is represented as :

General format for Control word

Example: To perform the operation $R_5 \leftarrow R_1 + R_4$ we have to provide following binary selection variable to the select inputs

- SELA : 001 - To place the contents of R1 into bus A
- SELB : 010 - To place the contents of R_4 into bus B.
- SELOPR : 10010 - To perform the arithmetic addition A+B
- SELREG : 011 - To place the result available on output bus in R_5

Binary Code	SELA	SELB	SELREG
0 0 0	Input	Input	—
0 0 1	R_1	R_1	R_1
0 1 0	R_2	R_2	R_2
0 1 1	R_3	R_3	R_3
1 0 0	R_4	R_4	R_4
1 0 1	R_5	R_5	R_5
1 1 0	R_6	R_6	R_6
1 1 1	R_7	R_7	R_7

Operation selection code	Operation
00000	Transfer A
00001	Increment A
00010	A+B
00011	A-B
00100	Decrement A
00101	A AND B
00110	A OR B
00111	A XOR B
01000	Complement A
01001	shift right A
01010	Shift left A
01100	
01101	
01110	
01111	
10000	
11000	

3. Stack Organization

Stack is a list of data elements usually words or bytes with the accessing restriction that the elements can be added or removed at one end of the list only. This end is called the top of the stack and another end is known as bottom of the stack. Stack structure is also known as Last In First Out (LIFO) list. The term PUSH and POP are used to describe placing a new data element on the stack and removing the top of the data element from the stack respectively.

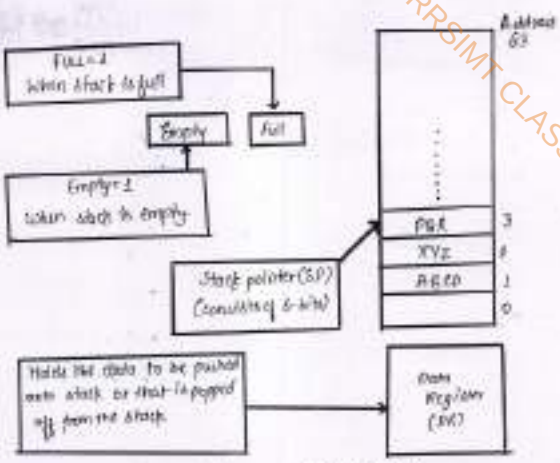


Fig- Block diagram of 64-word stack

Working of POP and PUSH:

POP (Performed if stack is not empty i.e. if EMPTY=0)

DR ← M[SP]
SP ← SP-1
If (SP=0) then (EMPTY ← 1)
FULL ← 0

Read item from the top of stack
Decrement stack pointer
Check if stack is empty
Ensure the stack not full

PUSH (Performed if stack is not full i.e. if FULL=0)

SP ← SP+1
M[SP] ← DR
If (SP=0) then (FULL ← 1)
EMPTY ← 0

Increment stack pointer
Write item on the top of stack
Check if stack is full
Mark the stack not empty

Register Stack

A stack can be placed in a portion of a memory unit or it can be organized as a collection of finite no. of CPU registers. In a figure, 32 bit register stack is represented. The stack pointer holds the address of the register that is currently the top of the stack.

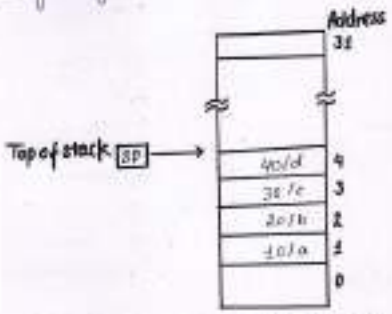
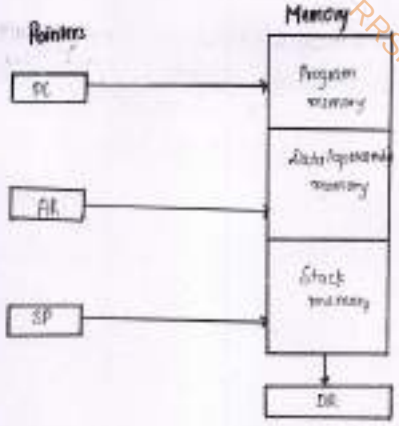


Fig 3.31: Block Diagram of a 32-word register stack

Memory Stack

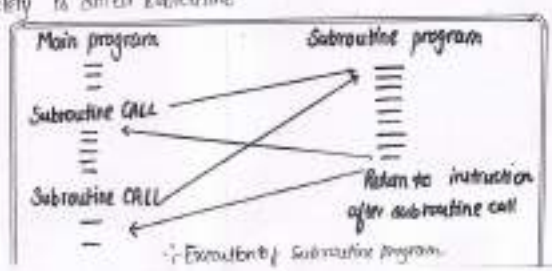
The operation of memory stack is exactly similar to the register stack. However, it is implemented using computer memory instead of CPU register array.

The memory stack has an advantage of large size but the operation on it is slower than that of register stack. This is because register stack is internal to the CPU and does not need any memory access.



Sharing of computer memory by program data and stack data.

Sub-routine: In some situations, it is better to execute part of a program that is not in sequence with the main program. For eg. there may be a part of a program that must be repeated many times during the execution for the entire program. Rather than writing repeated program again and again the programmer can write that part only once. This part is written separately. The part of the program which is written separately is called subroutine.



Execution of Subroutine program.

Addressing Modes

The address of the main memory can be specified directly within the instruction.

The rules for interpreting the address fields in the instruction is known as addressing mode.

There are following types of addressing modes:

S.No	Addressing mode	Example
1.	Register mode	MOV R1,R2
2.	Absolute mode	MOV R1,3000
3.	Immediate mode	MOV A,#20
4.	Register indirect	MOV A,(R2)
5.	Auto-increment	MOV(R1)+,R2
6.	Index mode	MOV 20(R1),R2
7.	Relative mode	JMP BACK
8.	Auto-Decrement	MOV(R1)-,R2
9.	Implied mode	CMA

Register mode: The operand is the content of processor register. The name of register is specified in the instruction.

Eg: MOV R1,R2 → This instruction copies the content of register R2 to R1.

Absolute or Direct Mode: The address of the location of the operand is given explicitly. (in a clear and calculated manner) in the instruction.

Eg: MOV A,2000 → This instruction copies the contents of memory location 2000 into the register A.

Immediate Mode: In this addressing mode, the operand is given explicitly in the instruction.

Eg: MOV A,#20 → This instruction copies the operand 20 in the register A.

Register Indirect Mode: The effective address of the operand is the content of a register in the main memory location whose address is given explicitly in the instruction.

Eg: EA=R

Eg: MOV A,(R2) → This instruction copies the contents of memory address by the contents of register R2 into the register A.

Index Mode: The effective address of the operand is generated by adding a constant value (specified in the instruction) called offset to the content of register.

$$EA = R + \underbrace{\text{offset}}_{\text{Low address part given in instruction}} \rightarrow (X_R)$$

Eg: MOV 20,(R1),R2 → This instruction loads the content of R2 into the memory location whose address is calculated by the addition of the content of Register R1 and constant value (offset or displacement) 20.

Relative Mode: The effective address is determined by the index mode using program counter in place of the general purpose register.

$$EA = PC + \text{Address part of Instruction}$$

Eg: JN2 BACK → This instruction causes program execution to go to the branch target location identified by the name BACK, if branch condition is satisfied.

Auto Increment Mode: The effective address of the operand is the contents of the register specified in the instruction. After accessing the operand the contents of register are incremented to address the next location.

Eg: MOV (R1)+,R2 → This instruction copies the content of register R2 into the memory location whose address is specified by content of register R1. After copy operation, the contents of register R2 are automatically incremented by 1.

Auto Decrement Mode: The contents of register specified in the instruction are decremented and then they are used as an effective address to access the memory location.

Eg: MOV (R1),-R2 → This instruction initially decrements the content of register R2 and then decremented contents of register R2 are used to address the memory location.

Implied Mode: In this addressing mode, opcode is specified in detailed and clear manner. This operation is performed in accumulator.

Eg: CMA → The above instruction complements the content of accumulator.

Example 24.24 At memory address 200, two word instruction, load to AC is stored with a mode bit as a most significant bit. A location 201 the address stored is 500. At location 202 next instruction is stored. The following are stored at different memory locations as shown above.

Memory location (Address)	Memory contents
399	450
400	700
201	500
500	800
702	325
800	300

If the content of PC is 200, while the contents of register R1 is 400, & register R2 400. If all the numbers and addresses are in decimal no. find out the contents of AC and effective address for the following addressing modes:

i) Direct address ii) Indirect address iii) Relative address
iv) Indirect address v) Register indirect addressing mode.

Soln:

	Load to AC	Mode bit
200	500	opcode + operands
201	next addr instruction	
202	⋮	
399	450	
400	700	
500	800	
600	900	
700	325	
800	300	

Given: PC = 200, R1 = 400, [Index] XR = 100 offset

	EA	AC
Direct Mode	500	800
Indirect Add.	(500) = 800	300
Relative Addr. mode	PC + Add Part of instruction to Add 202 + 50 = 702	325
Index	offset + Add Part in Index 100 + 500 = 600	900
Register Indirect	R1 = 400	700
Register Direct	–	400

Ques. 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:

(i) Immediate (ii) Direct (iii) Register indirect (iv) Relative

(v) Index with R1 as the index register. → $[X_R = R_1]$

Sol^n: $R_1 = 200$

300	Load to AC	Mode bit
301	400	
302	Next Add. Instruction	→ PC address

(i) EA = 301

(ii) EA = 400

(iii) EA = R_1 = 200

(iv) EA = PC Add. + Add. part given in instruction

EA = 302 + 400

EA = 702

(v) EA = X_R + Add. part given in ins.

= 200 + 400 $[X_R = R_1]$

EA = 600

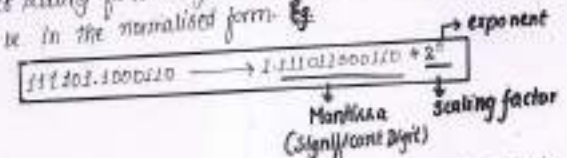
UNIT-02

Arithmetic and Logic Unit

Floating Point Representation:

To accomodate very large integers, and very small fractions, a computer must be able to represent numbers and operate on them in such a way that the position of the binary point is variable and is automatically adjusted as computation proceeds. In this case, the binary point is set to float, and the numbers are called floating point numbers. The floating point representation has three fields as:- sign, significand digit and exponent

To represent the number in floating point format, first binary point is shifted to right of the first bit and the number is multiplied by the correct scaling factor to get the same value. The number is said to be in the normalised form. Eg.



The standards for representing floating point numbers in 32 bits and 64 bits have been developed by the Institute of electrical and electronics engineers (IEEE).

It is also known as IEEE 754 standards.

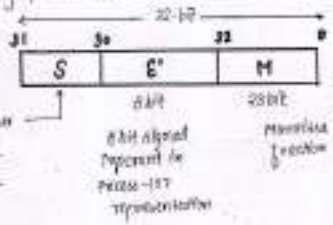
The 32-bit standard is called a single precision representation because it occupies a single 32-bit word. The 32-bits divided into three fields:- Field 1 sign, sign-bit (1 bit).

Field 2 exponent (8-bit)

Field 3 Mantissa (23-bit)

Instead of the sign exponent (E), the value actually stored in the Exponent field is E' = E + Base value.

In 32-bit floating point system, base is 127. Hence E' = E (actual) + 127 Excess



E' = E + 127

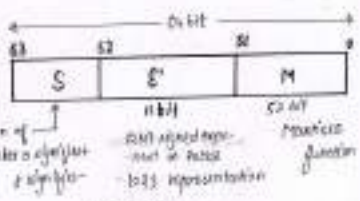
32 bit Representation.

* The 64-bit representation is called a double precision Representation. because it occupies two 32 bit words. The 64-bits are divided into three fields:- Field (1) sign - 1 bit

Field (2) exponent → 11 bit

Field (3) Mantissa → 52 bit.

* In double precision format, E' = E + 1023.



E' = E + 1023

★ Represent $(125.125)_{10}$ in single and double precision format.

Sol^n:

2 | 125	1
2 | 62	0
2 | 31	1
2 | 15	1
2 | 7	1
2 | 3	1
1	

$0.125 \times 2 = 0.250$ 0

$0.250 \times 2 = 0.500$ 0

$0.500 \times 2 = 1.000$ 1

$= (1111101.001)_2$

$= 1.11110100100 \times 2^{10}$

2 | 127	1
2 | 63	0
2 | 31	1
2 | 15	1
2 | 7	1
2 | 3	1
1	

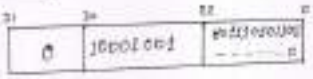
→ (1111101)

Single Precision Format

E' = E + 127

E' = 10 + 127

E' = 137



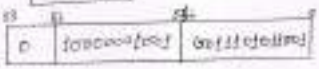
Double Precision Format

E' = E + 1023

E' = 10 + 1023

E' = 1033

E' = 10000001001



③ $(-309.1875)_{10}$

$309 = (100110101)$

$0.1875 \times 2 = 0.3750$ 0
$0.3750 \times 2 = 0.7500$ 0
$0.7500 \times 2 = 1.5000$ 1

$= -(100110101.0011)_2$

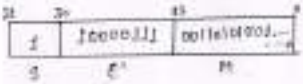
$= -1.00110101001 \times 2^8$

Single Precision

$E' = E + 127$

$E' = 8 + 127$

$E' = 135 = (10000111)_2$

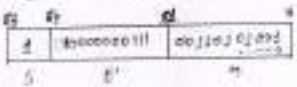


Double Precision

$E' = E + 1023$

$E' = 8 + 1023$

$E' = 1031 = (10000000111)_2$



④ $(1980.125)_{10}$

$1980 = (011110111100)$

$= (11110111100.001)_2$

$= (1.1110111100001 \times 2^{10}$

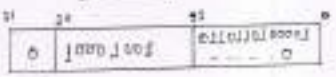
$.125 \times 2 = .250$ 0
$0.25 \times 2 = .500$ 0
$0.500 \times 2 = 1.00$ 1

Single Precision

$E' = E + 127$

$E' = 10 + 127$

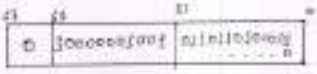
$E' = 137 = (10001001)_2$



Double Precision

$E' = E + 1023 = 10 + 1023$

$E' = 1033 = (10000001001)_2$



⑤ $(-19.1328)_{10}$

$19 = (10011)_2$

$0.1328 \times 2 = 0.2656$ 0
$0.2656 \times 2 = 0.5312$ 0
$0.5312 \times 2 = 1.0624$ 1

$(10011.0001)_2 = 1.0011000001 \times 2^4$

Single Precision

$E' = E + 127$

$E' = 4 + 127$

$E' = 131 = (10000011)_2$

Double Precision

$E' = E + 1023$

$E' = 4 + 1023$

$E' = 1027 = (10000000011)_2$

⑥ $(-1523.012)_{10}$

$(10111111011.0000)_2$

$1.0111111011 \times 2^{10}$

$E = 10$

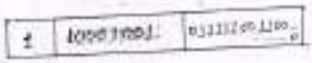
$0.012 \times 2 = 0.024$
$0.024 \times 2 = 0.048$
$0.048 \times 2 = 0.096$
$0.096 \times 2 = 0.192$
$0.192 \times 2 = 0.384$

Single Precision

$E' = E + 127$

$E' = 10 + 127$

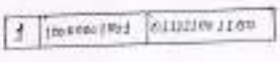
$E' = 137 = (10001001)_2$



Double Precision

$E' = E + 1023$

$E' = 1033 = (10000001001)_2$



⑥ $(1486.785)_{10}$

$= (10111001110.0011)_2$

$= 1.0111001110011 \times 2^{10}$

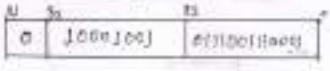
$E = 10$

Single Precision

$E' = E + 127$

$E' = 10 + 127$

$E' = 137 = (10001001)_2$

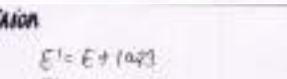


Double Precision

$E' = E + 1023$

$E' = 10 + 1023$

$E' = 1033 = (10000001001)_2$



⑦ $(1243.725)_{10}$

$(10011011011.1011)_2$

$= 1.0011011011011 \times 2^{10}$

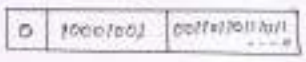
$E = 10$

Single Precision

$E' = E + 127$

$E' = 10 + 127$

$E' = 137 = (10001001)_2$

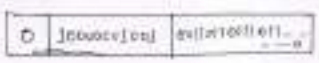


Double Precision

$E' = E + 1023$

$E' = 10 + 1023$

$E' = 1033 = (10000001001)_2$



⑧ $(-116.375)_{10}$

$= (1110100.011)_2$

$= 1.110100011 \times 2^6$

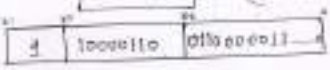
$E = 7$

Single Precision

$E' = E + 127$

$E' = 7 + 127$

$E' = 134 = (10000110)_2$

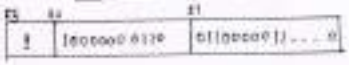


Double Precision

$E' = E + 1023$

$E' = 7 + 1023$

$E' = 1030 = (10000000110)_2$



(b) $(-1976.453)_{10}$

$= (11110111000.0111)_2$

$= 1.1110111000.0111 \times 2^{10}$

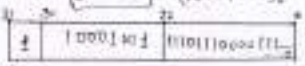
$E' = 10$

Single Precision

$E' = E + 127$

$E' = 10 + 127$

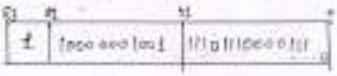
$E' = 137 = (10001001)_2$



Double Precision

$E' = E + 1023 = 10 + 1023$

$E' = 1033 = (10000001001)_2$



Parallel Adder

A single full-adder is capable of adding two one-bit numbers and an input carry. In order to add binary numbers with more than one bit, additional full-adders must be employed.

A n-bit parallel adder can be constructed using number of full adder circuits connected in parallel.

Figure shows the block diagram of n-bit parallel adder using n-number of full adder circuits connected in cascade i.e., the carry output of each adder is connected to the carry input of the next higher-order adder.

It should be noted that either a half adder can be used for least significant position or the carry input of a full adder is made 0 because there is no carry into the least significant bit position.

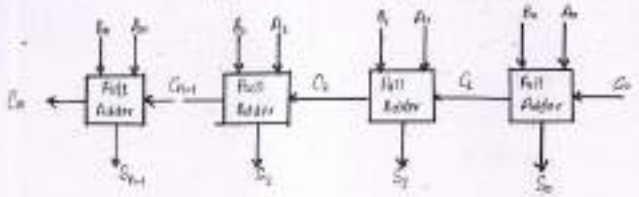


Fig. 4-4:- Block diagram of n-bit parallel adder

Parallel Subtractor:- The subtraction of binary numbers can be done most conveniently by means of complements.

The subtraction A-B can be done by taking 2's complement of B and adding it to A. The 2's complement can be obtained by taking the 1's complement and adding one to the least significant pair of bits.

The 1's complement can be implemented with inverters and a one can be added to the sum through the input carry to get 2's complement as shown in the fig.

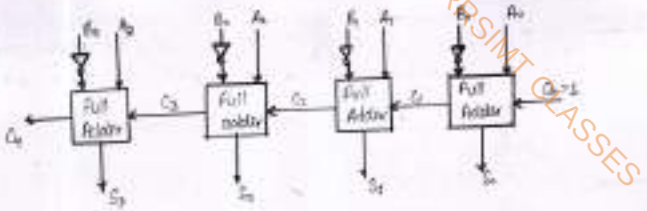


Fig. 4-bit parallel subtractor

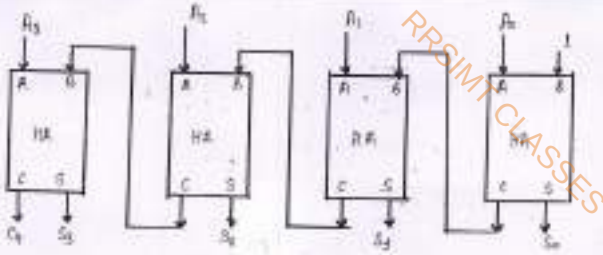


Fig 4.6:- 4-bit Incrementor using combinational circuit

Binary Incrementor :-

We have seen that the increment micro-operation adds one to a number in register. This micro-operation can be easily implemented with a binary counter. Every time the count enable is active, the clock pulse transition increments the count.

The counter is a sequential circuit. The fig shown a combinational circuit to implement increment micro-operation. Here, one of the inputs to the least significant half-adder (HA) is connected to logic 1 and the other input is connected to the least significant bit of the number to be incremented. The output carry from lower order half adder is connected to the one of the inputs of the next higher order half adder. Such circuits gets the four bit number from bits A_0 through A_3, adds one to it, and generates the incremented output in S_0 through S_3. If the number to be incremented is 1111, the output carry C_4 is generated and we get 0000 at S_0 through S_3.

Array Multiplier:-

The multiplication process of binary number is similar to decimal nos. Actually binary multiplication is simple than decimal multiplication because it involves 0 and 1 only.

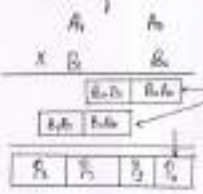
For multiplication in binary nos., it uses n-shifts and adds to multiply n-bit binary number.

The combinational logic circuit implemented to perform such multiplication is called combinational multiplier or array multiplier.

Let us generalize the multiplication process for 2X2 multiplier for two assigned 2-bit nos. - multiplicand $A = A_1A_0$ and multiplier $B = B_1B_0$.

The multiplication process involves multiplication of 2-bit nos and addition of 2-bit nos.

The multiplication of two bits can be implemented using 2 input AND gate whereas addition of 2 bits can be implemented using half adder.



$P_0 = A_0B_0$

$P_1 = A_1B_0 + A_0B_1$

$P_2 = A_1B_1$ + Carry of P_1

P_3 = carry of P_2

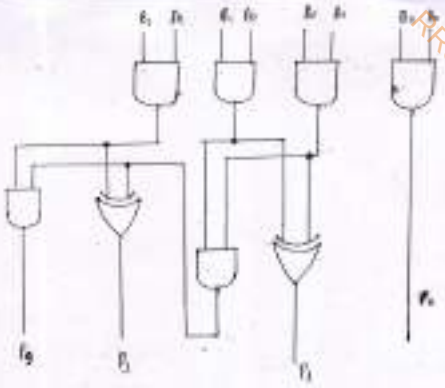
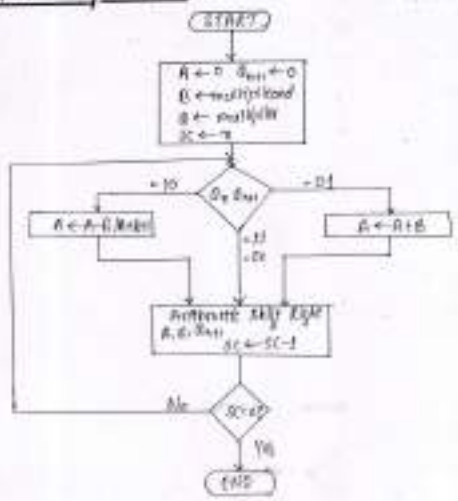


Fig 2.10.2 2x2 bit combinational array multiplier

Booth Multiplication



Algorithm:

Step 1:- Load $A = 0$; $Q_{n+1} = 0$
B = Multiplicand
Q = multiplier
SC = n

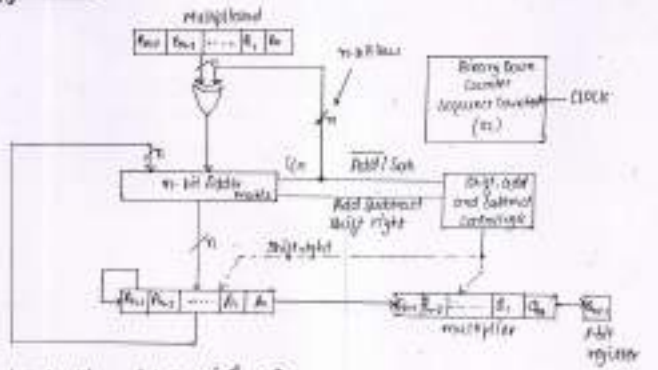
Step 2:- Check the status of $Q_n Q_{n+1}$ -
If $Q_n Q_{n+1}$ = 10 perform $A \leftarrow A - B$
If $Q_n Q_{n+1}$ = 01 perform $A \leftarrow A + B$

Step 3:- Arithmetic shift right: A, Q_n, Q_{n+1}

Step 4:- Decrement sequence counter if not zero, repeat steps through 4.

Step 5:- Stop

Algorithm



- Initial settings $A \leftarrow 0$ and $Q_{n+1} = 0$

∴ Hardware implementation of signed binary multiplication

---

① Evaluate

(i) $9 * 17$

$B = 9 = 001001$ $\quad \bar{B}+1 = 110111$

$Q = 17 = 010001$

Q_n	Q_{n+1}	B = 001001, $\bar{B}+1$ = 110111	A	Q	Q_{n+1}	SC
		Initial	000000	010001	0	110
1	0	$A \leftarrow A+\bar{B}+1$	110111 / 110111			
		Ashr (A,B,Q_{n+1})	111011	101000	1	101
0	1	$A \leftarrow A+B$	010011 / 001001			
		Ashr (A,B,Q_{n+1})	000100	110100	0	100
0	0	Ashr (A,B,Q_{n+1})	000010	011010	0	011
0	0	Ashr (A,B,Q_{n+1})	000001	001101	0	010
1	0	$A \leftarrow A+\bar{B}+1$	110111 / 111110			
		Ashr (A,B,Q_{n+1})	111110	110110	1	001
0	1	$A \leftarrow A+B$	000111 / 000111			
		Ashr (A,B,Q_{n+1})	000011	011011	0	000

$B * Q$ = 0000 1001 1011

---

② $-13 * 16$

$B = -13 = 11011 \Rightarrow \bar{B}+1 = 00101$

$Q = 16 = 010000$

$+13 = 01101$

↓ 2's complement

$+13 = 00101 + 1$

$-13 = 10011$

Q_n	Q_{n+1}	B = 11011, $\bar{B}+1$ = 00101	A	Q	Q_{n+1}	SC
		Initial	000000	010000	0	101
0	0	Ashr (A,B,Q_{n+1})	000000	001000	0	100
0	0	Ashr (A,B,Q_{n+1})	000000	000100	0	011
0	0	Ashr (A,B,Q_{n+1})	000000	000010	0	010
0	0	Ashr (A,B,Q_{n+1})	000000	000001	0	001
1	0	$A \leftarrow A+\bar{B}+1$	001101 / 001101			
		Ashr (A,B,Q_{n+1})	000110	100000	1	000
0	1	$A \leftarrow A+B$	110011 / 111000			
		Ashr (A,B,Q_{n+1})	111100	110000	0	000

$B * Q$ = 1111 0011 0000

⑤ 18 * -14

B = 18 = 01010 B+1 = 10110

Q = -14 = 11010

14 = 01110

↓ 2's complement

= 10101 + 1

= 10010

Q_n	Q_{n+1}	B = 01010 B+1 = 10110	A	Q	Q_{n+1}	SC
		Initial	00000	11010	0	110
0	0	Ashr(A,Q,Q_{n+1})	00000	01101	0	101
1	0	A ← A+B+1	10110 / 10110			
		Ashr(A,Q,Q_{n+1})	11011	00110	1	100
0	1	A ← A+B	01001 / 01001			
		Ashr(A,Q,Q_{n+1})	00100	10011	0	011
1	0	A ← A+B+1	10111 / 11000			
		Ashr(A,Q,Q_{n+1})	11100	01001	1	010
0	1	A ← A+B	01010 / 00100			
		Ashr(A,Q,Q_{n+1})	00010	00100	0	001
0	0	Ashr(A,Q,Q_{n+1})	00001	00010	0	000

B * Q = 11100010100

⑥ -12 * -11

B = -12 = 10100 B+1 = 01100

Q = -11 = 10101

12 = 01100 11 = 01011

↓ 2's comp ↓ 2's comp

10011+1 10100+1

-12 = 10100 -11 = 10101

B_n	Q_{n+1}	B = 10100 B+1 = 01100	A	Q	Q_{n+1}	SC
		Initial	00000	10101	0	101
1	0	A ← A+B+1	01100 / 01100			
		Ashr(A,Q,Q_{n+1})	00110	01010	1	100
0	1	A ← A+B	11010 / 11010			
		Ashr(A,Q,Q_{n+1})	11101	00101	0	011
1	0	A ← A+B+1	01100 / 01001			
		Ashr(A,Q,Q_{n+1})	00100	10010	1	010
0	1	A ← A+B	10100 / 11000			
		Ashr(A,Q,Q_{n+1})	11100	01001	0	001
1	0	A ← A+B+1	01100 / 01001			
		Ashr(A,Q,Q_{n+1})	00100	10100	1	000

B * Q = 00100101100

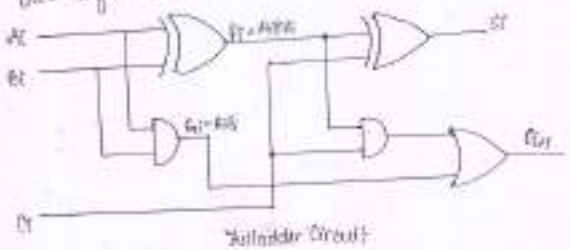
Look Ahead Carry Adder

- It is also known as fast adder / high speed adder.
- In ripple carry adder, the sum and carry outputs of any stage cannot be produced until the input carry occurs, this leads to a time delay in the addition process.
- This delay is known as carry propagation delay.

Considering the example:

```
  0101
+ 0011
------
  1000
```

- Addition of the LSB position produces a carry into the second position.
- This carry, when added to the bits of the second position produces a carry for the third position. The latter carry when added to the bits of third position, produces a carry to the next position.
- One method of speeding up of this process by eliminating inter stage carry delay is called look ahead carry addition.
- It uses two functions - carry generate (CG) and carry propagate (CP).
- Consider the full adder which is designed with the help of two half adders.



Full adder Circuit

- We define two functions carry generate and carry propagate as follows:

$$P_i = A_i \oplus B_i \quad \left.\right\} \text{---(1)}$$
$$G_i = A_i B_i$$

- The output sum and carry can be expressed as:

$$S_i = P_i \oplus C_i \quad \left.\right\} \text{---(2)}$$
$$C_{i+1} = G_i + P_i C_i$$

- G_i is called carry generate and it produces a carry when both A_i and B_i are 1, regardless of the input carry.
- P_i is called carry propagate because it is term associated with propagation of carry C_i to C_{i+1}.
- For example, 4 stage carry look ahead adder can be defined as follows.
- For initial condition, $C_i = C_0$, which is equal to zero.

Put i=0 in eq (2)

$$C_{i+1} = G_i + P_i C_i \quad \left.\right\} \text{---(3)}$$
$$C_1 = G_0 + P_0 C_0$$

i=0;

where $G_0 = A_0 B_0$ and $P_0 = A_0 \oplus B_0$ and $C_0 = 0$

Hence, $C_1 = G_0 + P_0 C_{in}$

Now put i=1 in eq (3)

i=1

$$C_2 = G_1 + P_1 C_1$$
$$C_2 = G_1 + P_1 (G_0 + P_0 C_{in})$$
$$C_2 = G_1 + P_1 G_0 + P_1 P_0 C_{in} \text{---(4)}$$

Now put i=2 in eq (3)

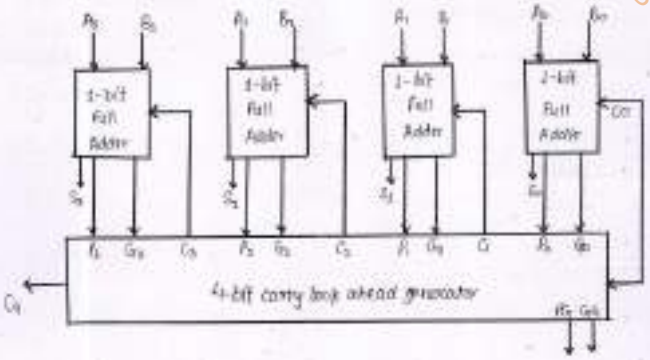
$$C_3 = G_2 + P_2 C_2$$
$$C_3 = G_2 + P_2 (G_1 + P_1 G_0 + P_1 P_0 C_{in})$$
$$C_3 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_{in} \text{---(5)}$$

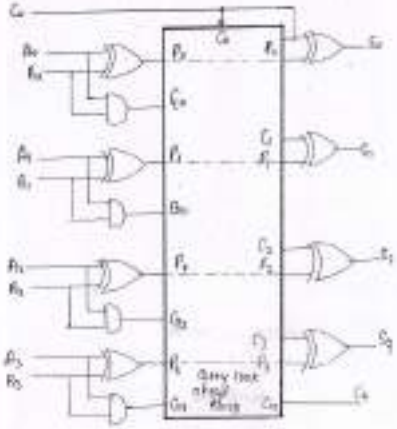
Now put i=3 eq^n ③

$C_0 = G_3 + P_3 C_3$

$C_1 = G_2 + P_2(G_1 + P_1 G_0 + P_1 P_0 C_0)$

$C_4 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 C_0$ ←④

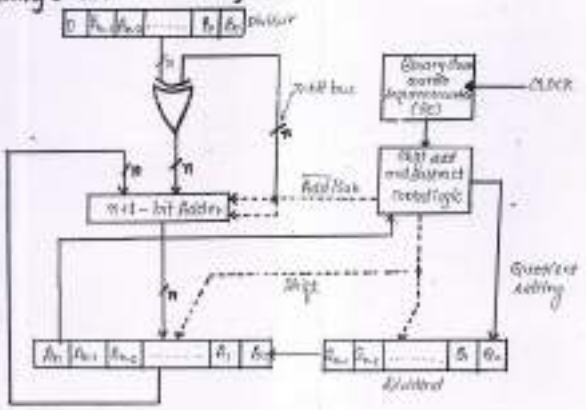




Booth's Division Algorithm

The division process for binary number is similar to decimal number. In division process, first the bits of the dividend are examined from left to right until the set of bits examined represents a no. of number greater than or equal to the divisor. until this condition occurs, 0's are placed in the qc quotient from left to right when the condition is satisfied one is placed in the quotient and the divisor is subtracted from the partial dividend. This result is referred as partial remainder.

Restoring Booth's Division Algorithm



Flowchart to implement binary division

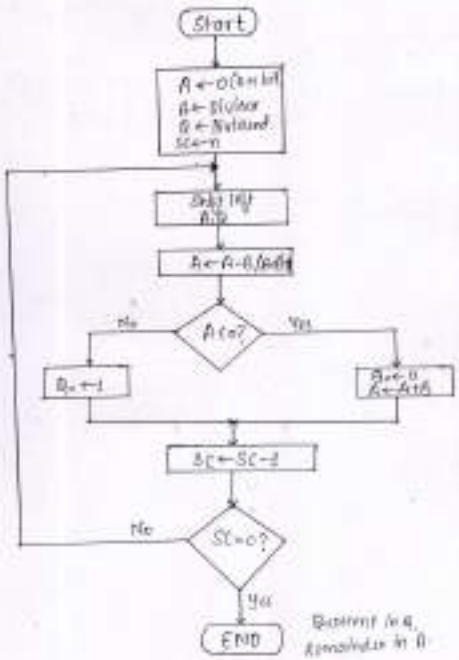
Division operation steps:

1. Shift A and Q left one binary operation position.
2. Subtract divisor (i.e. add 2's complement of divisor (B)) from A and place answer back in A ($A \leftarrow A - B$)
3. If the sign bit of A is 1, set Q_0 to 0 and add divisor back to A

(that is, restore A's otherwise, set Q_0 to 1.

4. Repeat steps 1, 2 and 3 n times.

A flowchart for division operation is as shown in Fig. 2.11.3



∴ Flowchart for restoring division algorithm

Perform the division of the following nos using restoring division algorithm.

1. Dividend = 1010 and Divisor = 0011

Divisor(B) = 0011

Dividend(Q) = 1010

B = 0011

$\bar{B}+1$ = 1101

Operation	A	Q	SC
Initial	00000	1010	100
Shift left(A,Q)	00001	0100	
$A \leftarrow A+\bar{B}+1$	11101		
	11110		
Set $Q_0 \leftarrow 0$	00011		
$A \leftarrow A+B$	00010		011
Shift left(A,Q)	00101	1000	
$A \leftarrow A+\bar{B}+1$	11101		
	00010		
Set $Q_0 \leftarrow 1$			001
Shift left(A,Q)	00101	0001	
$A \leftarrow A+\bar{B}+1$	11101		
	00010		000

∴ Quotient = 0011

Remainder = 00010

(ii) 17÷3

B = 3 = 00011

Q = 17 = 10001

B = 00011

B̄+1 = 11101

Operation	A	Q	SC
Initial	00000	10001	101
Shift left (A,Q)	00001	00010	
A ← A+B̄+1	11110		
	11110		
Set Q_0 ← 0	00011		
A ← A+B	00001		100
Shift left (A,Q)	00010	00100	
A ← A+B̄+1	11110		
	00000		010
Set Q_0 ← 1			
Shift left (A,Q)	00001	01001	
A ← A+B̄+1	11110		
	11111		001
Set Q_0 ← 0			
A ← A+B	00010	10010	
	00010		000

Remainder = 00010

Quotient = 00101

(iii) 1101/101

Q = 1101

B = 00101

B̄+1 = 11011

Operation	A	Q	SC
Initial	00000	1101	100
Shift left (A,Q)	00001	1010	
A ← A+B̄+1	11011		
	11100		
Set Q_0 ← 0	00101		
A ← A+B	00001		011
Shift left (A,Q)	00011	0100	
A ← A+B̄+1	11011		
	11110		
Set Q_0 ← 0	00101		
A ← A+B	00011		010
Shift left (A,Q)	00110	1001	
A ← A+B̄+1	11011		
	00001		
Set Q_0 ← 1			001

Shift left (A,Q) 00011 0010

A ← A+B̄+1 11011

Set Q_0 ← 0 11110

A ← A+B 00101

00011 000

Remainder = 00011

Quotient = 00010

(iv) 1000/111

Q = 1000

B = 00111

B̄+1 = 11001

Operation	A	Q	SC
Initial	00000	1000	100
Shift left (A,Q)	00001	0000	
A ← A+B̄+1	11001		
	11010		
Set Q_0 ← 0	00111		
A ← A+B	00001		011
Shift left (A,Q)	00010	0000	
A ← A+B̄+1	11001		
	11011		
Set Q_0 ← 0	00111		
A ← A+B	00010		010
Shift left (A,Q)	00100	0000	
A ← A+B̄+1	11001		
	11101		001
Set Q_0 ← 0			

Shift left (A,Q) 11101 0000

A ← A+B̄+1 11001

Set Q_0 ← 0 10110

A ← A+B 00111

10101 000

Remainder = 10101

Quotient = 0000

(v) 1101/101

Q = 1101

B = 00101

B̄+1 = 11011

Operation	A	Q	SC
Initial	00000	1101	100
Shift left (A,Q)	00001	1010	
A ← A+B̄+1	11011		
	11100		
Set Q_0 ← 0	00101		
A ← A+B	00001		011
Shift left (A,Q)	00011	0100	
A ← A+B̄+1	11011		
	11110		
Set Q_0 ← 0	00101		
A ← A+B	00011		010
Shift left (A,Q)	00110	1001	
A ← A+B̄+1	11011		
	00001		
Set Q_0 ← 1			001

Shift left (A,Q) 00011 1001

A ← A+B̄+1 11011

Set Q_0 ← 0 11110

A ← A+B 00101

00011 001

Remainder = 00011

Quotient = 0001

A ← A+B 11101 0001

00111

00100

Shift left (A,Q) 01000 0000

A ← A+B̄+1 11001

11101

Set Q_0 ← 0 00010 0000

A ← A+B 00111

00101 000

Shift left (A,Q) 01010 0000

A ← A+B̄+1 11001

00011

Set Q_0 ← 1 00110 0001

A ← A+B̄+1 11001

00001 001

Remainder = 00001

Quotient = 0011

Non-Restoring Division

Consider the sequence of operations that takes place after the subtraction operation in the restoring algorithm.

If A is positive

Shift left and subtract

divisor → 2A−B

If A is negative

Restore → A+B

shift left and subject subtract

divisor → 2(A+B)−B

= 2A+B

Looking at the above operations that take place after the subtraction operation A

Looking at the above operations, we can write following steps for non-restoring algorithm

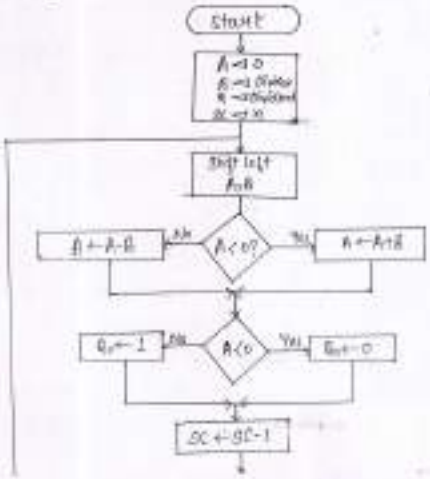
Step 1: If the sign of A is 0, shift A and Q left one bit position and subtract divisor from A, otherwise, shift A and Q left and add divisor to A. If the sign of A is 0, set Q_0 to 1, otherwise, set Q_0 to 0

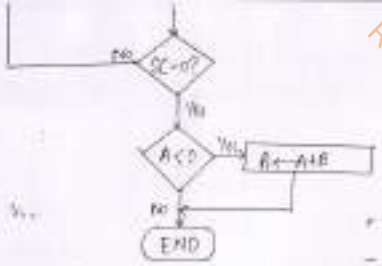
Step 2: Repeat steps 1 and 2 for n times

Step 3: If the sign of A is 1, add divisor to A

Note: Step 3 is required to leave the proper positive remainder in A at the end of n cycles

A flowchart for non-restoring division operation is as shown in Fig 2.11





(i) 1101 / 11

B = 1101

B = 0011

$\bar{B}$ = 1100

$\bar{B}+1$ = 1101

Operation	A	Q	SC
Initial	00000	1101	100
Shift left A,Q	00011	1010	
$A \leftarrow A+\bar{B}+1$	11101		
Set $Q_0 \leftarrow 0$	11110		011
Shift Left (A,Q)	11101	0101	
$A \leftarrow A+B$	00011		
Set $Q_0 \leftarrow 1$	00000		
Shift Left (A,Q)	00000	1010	
$A \leftarrow A+\bar{B}+1$	11101		
Set $Q_0 \leftarrow 0$	11101		001
Shift Left (A,Q)	11011	0100	
$A \leftarrow A+\bar{B}+1$	00011		
	11110		000
$A+B$	00011		Q = 0100
	00001		R = 00001

(ii) 17/3

Q = 17 = 10001

B = 3 = 00011

$\bar{B}$ = 11100

$\bar{B}+1$ = 11101

Operation	A	Q	SC
Initial	00000	10001	101
Shift left (A,Q)	00001	0010	
$A \leftarrow A+\bar{B}+1$	11110		
Set $Q_0 \leftarrow 0$	11110		100
Shift Left (A,Q)	11100	0100	
$A \leftarrow A+B$	00011		
Set $Q_0 \leftarrow 0$	11111		011
Shift Left (A,Q)	11110	1000	
$A \leftarrow A+\bar{B}+1$	00001		
Set $Q_0 \leftarrow 1$	00000		010
Shift Left (A,Q)	00001	0001	
$A \leftarrow A+\bar{B}+1$	11110		
Set $Q_0 \leftarrow 0$	11111		001
Shift Left (A,Q)	11111	0010	
$A \leftarrow A+B$	00011		
Set $Q_0 \leftarrow 1$	00010		000

(iii) 19/4

Q = 19 = 10011

B = 00100

$\bar{B}$ = 11011

$\bar{B}+1$ = 11100

Operation	A	Q	SC
Initial	000000	10011	101
Shift Left (A,Q)	00001	00110	
$A \leftarrow A+\bar{B}+1$	11101		
Set $Q_0 \leftarrow 0$	11101		100
Shift Left (A,Q)	11010	01100	
$A \leftarrow A+B$	00100		
Set $Q_0 \leftarrow 0$	11110		011
Shift Left (A,Q)	11100	11000	
$A \leftarrow A+B$	00100		
Set $Q_0 \leftarrow 0$	00000		010
Shift Left (A,Q)	00000	10000	
$A \leftarrow A+\bar{B}+1$	11100		
Set $Q_0 \leftarrow 0$	11100		001
Shift Left (A,Q)	11001	00000	
$A \leftarrow A+\bar{B}$	00100		
Set $Q_0 \leftarrow 0$	11101		000
$A+B$	00100		
	00001		

Comparison b/w restoring and non-restoring algorithm

Restoring	Non-restoring
Needs restoring of register A if the result of subtraction is negative.	Does not need restoring.
In each cycle content of register A is first shifted left and then divisor is subtracted from it.	In each cycle content of register A is first shifted left and then division is added or subtracted with the contents of register A depending on the sign of A.
Does not need restoring of remainder.	Needs restoring of remainder if remainder is negative.
Slower algorithm.	Faster algorithm.

Unit:-03

20/11/23

Control Unit

Instruction Cycle:- An instruction cycle involves three sub-cycles:-

1. Fetch:- The fetch phase reads the instruction from memory into the CPU.
2. Decode:- The decode phase interprets the opcode by decoding it.
3. Execute:- The execute phase performs the indicated operation.

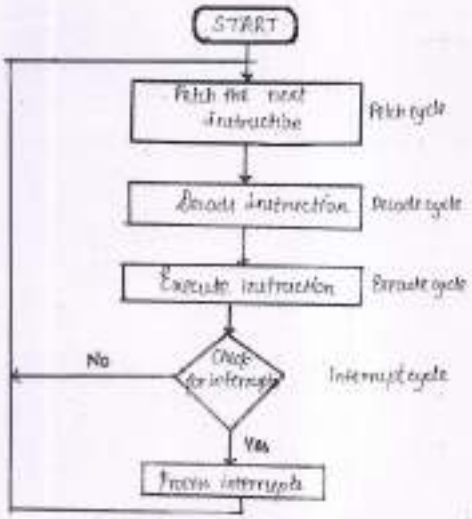


Fig:- Basic instruction cycle with interrupt cycle

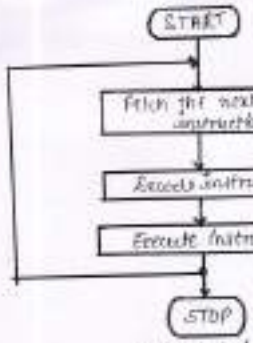
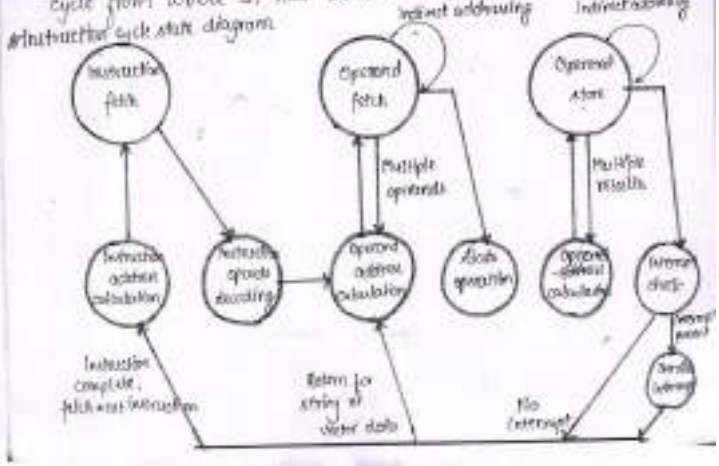


Fig:- Basic instruction cycle

Processor checks for valid interrupt request after each instruction cycle. If any valid interrupt request is present, processor saves the current process state and service the interrupt. Servicing the interrupt means executing interrupt cycle. After completing it, processor starts the new instruction cycle from where it has been interrupted.

Instruction cycle state diagram



Instruction format:- Various fields in instruction format are:-

(i) Opcode:- The operation code field in the instruction specify the operation to be performed. The operation code is specified by binary code, known as mnemonic opcode.

(ii) Source/Destination operands:- The source operand may be in CPU registers or in the memory. Many times the instruction specifies the address of the source operand so that operands can be accessed and operated by the CPU according to the instruction.
The operands on which we have to perform opcode are known as source operands.
The operation executed by the CPU may produce result. Most of the times the result is stored in one of the operands. Such operand is known as destination operand.

(iii) Next Instruction Address:- The next instruction address tells the CPU from where to fetch the next instruction after completion of current execution.

Various types of Instruction format

(i) 3- address instruction format.
(ii) 2- address instruction format.
(iii) 1- address instruction format.
(iv) 0- address instruction format.

Three-Address Instruction:- Three address instruction can be represented symbolically as:-

ADD A, B, C

where A, B, C are the variables. These variable names are assigned to distinct locations in the memory. In this instruction, operand B and C are called source operands and A is called

destination operand and ADD is the operation to be performed on the operands. Thus, the general instruction format for three address instruction is:

OPCODE, Destination operand, Source operand 1, Source op. 2

If n bits are required to specify one memory address, then 3n bits are required to specify three operands and 1 bit required to specify microoperation.

Example: To ADD to evaluate the arithmetic statement $Y=(A+B)*(C+D)$ by using 3-address instruction format

ADD R_1, A, B	$R_1 \leftarrow m[A] + m[B]$
ADD R_2, C, D	$R_2 \leftarrow m[C] + m[D]$
MUL Y, R_1, R_2	$m[Y] \leftarrow R_1 * R_2$

2. Two-Address Instruction

Two-address instruction can be represented symbolically as:

ADD A, B.

This instruction adds the content of variable A and B and stores the sum in variable A destroying its previous contents. Here operand B is source operand and operand A is served as both source and destination operand. The general instruction format as follows:

Opcode, Source operand, destination operand.

Example: $Y=(A+B)*(C+D)$ using 2- add. instruction format

MOV R_1, A	$R_1 \leftarrow m[A]$
ADD R_1, B	$R_1 \leftarrow R_1 + m[B]$
MOV R_2, C	$R_2 \leftarrow m[C]$
ADD R_2, D	$R_2 \leftarrow R_2 + m[D]$
MUL R_1, R_2	$R_1 \leftarrow R_1 * R_2$
MOV Y, R_1	$m[Y] \leftarrow R_1$

PUSH A	TOS $\leftarrow m[A]$
PUSH B	TOS $\leftarrow m[B]$
ADD	TOS $\leftarrow m[A] + m[B]$
PUSH C	TOS $\leftarrow m[C]$
PUSH D	TOS $\leftarrow m[D]$
ADD	TOS $\leftarrow m[C] + m[D]$
MUL	TOS $\leftarrow m[A] + m[B] * m[C] + m[D]$
POP Y	$m[Y] \leftarrow$ TOS

WAP to evaluate using 3, 2, 1, 0 address instruction format

$$Y = \frac{A[B+C(D+E)]}{F(G+H)}$$

3-address Instruction:

ADD R_1, D, E	$R_1 \leftarrow m[D] + m[E]$
MUL R_2, C, R_1	$R_2 \leftarrow m[C] * R_1$
ADD R_3, R_2, B	$R_3 \leftarrow m[B] + R_2$
MUL R_4, A, R_3	$R_4 \leftarrow m[A] * R_3$
ADD R_5, G, H	$R_5 \leftarrow m[G] + m[H]$
MUL R_6, F, R_5	$R_6 \leftarrow m[F] * R_5$
DIV X, R_4, R_6	$m[X] \leftarrow R_4 / R_6$

2-address Instruction:

MOV R_1, D	$R_1 \leftarrow m[D]$
ADD R_1, E	$R_1 \leftarrow R_1 + m[E]$
MUL R_1, C	$R_1 \leftarrow R_1 * m[C]$
ADD R_1, B	$R_1 \leftarrow R_1 + m[B]$
MUL R_1, A	$R_1 \leftarrow R_1 * m[A]$
MOV R_2, H	$R_2 \leftarrow m[H]$
ADD R_2, G	$R_2 \leftarrow R_2 + m[G]$
MUL R_2, F	$R_2 \leftarrow R_2 * m[F]$
DIV R_2, R_1	$R_2 \leftarrow R_2 / R_1$
MOV X, R_2	$m[X] \leftarrow R_2$

3. One-address instruction format

The one address instruction can be represented symbolically as:

ADD B.

This instruction adds the content of variable B into the processor register called accumulator and stores the sum back into the accumulator destroying the previous contents of the accumulator. In this instruction, the second operand is assumed to be implicitly tally in a unique location accumulator. The general instruction format for one address instruction is:

operation, source.

In one add instruction we use load and store instruction type.

Eg: $Y=(A+B)*(C+D)$

LOAD A	AC $\leftarrow m[A]$
ADD B	AC $\leftarrow$ AC $+ m[B]$
STORE P	$m[P] \leftarrow$ AC
LOAD C	AC $\leftarrow m[C]$
ADD D	AC $\leftarrow$ AC $+ m[D]$
MUL P	AC $\leftarrow$ AC $* P$
STORE Y	$m[Y] \leftarrow$ AC

4. Zero address instruction

In these instruction, the location of all operands are defined implicitly. Such instructions are found in a machine that store operands in a structure called Push down stack.

Eg: $Y=(A+B)*C+D$

Postfix: AB+CD+*

ADD R_1, G	$R_1 \leftarrow R_1 + m[G]$
MUL R_2, F	$R_2 \leftarrow R_2 * m[F]$
DIV R_1, R_2	$R_1 \leftarrow R_1 / R_2$
MOV X, R_1	$m[X] \leftarrow R_1$

1-address Instruction

LOAD H	AC $\leftarrow m[H]$
ADD G	AC $\leftarrow$ AC $+ m[G]$
MUL F	AC $\leftarrow$ AC $* m[F]$
STORE P	$m[P] \leftarrow$ AC
LOAD E	AC $\leftarrow m[E]$
ADD D	AC $\leftarrow$ AC $+ m[D]$
MUL C	AC $\leftarrow$ AC $* m[C]$
ADD B	AC $\leftarrow$ AC $+ m[B]$
MUL A	AC $\leftarrow$ AC $* m[A]$
DIV P	AC $\leftarrow$ AC $/ m[P]$
STORE X	X $\leftarrow$ AC

0-address Instruction

Postfix: ABCDE+++*FGH+*/

PUSH A	TOS $\leftarrow m[A]$
PUSH B	TOS $\leftarrow m[B]$
PUSH C	TOS $\leftarrow m[C]$
PUSH D	TOS $\leftarrow m[D]$
PUSH E	TOS $\leftarrow m[E]$
ADD	TOS $\leftarrow m[D] + m[E]$
MUL	TOS $\leftarrow m[D] + m[E] * m[C]$
ADD	TOS $\leftarrow m[B] + m[E] * m[C] + m[B]$
MUL	TOS $\leftarrow m[A] + m[E] * m[C] + m[B] * m[A]$
PUSH F	TOS $\leftarrow m[F]$
PUSH G	TOS $\leftarrow m[G]$
PUSH H	TOS $\leftarrow m[H]$

ADD	TOS ← m[0] + m[1]
MUL	TOS ← m[2] + m[3] * m[4]
DIV	TOS ← m[5] + m[6] * m[7] / m[8] * m[9]
POP	m[X] ← TOS

WAP to evaluate X = (A+B*C)/(D-E*F) using all addressing instructions

3-address instruction:

```
MUL  R1, B, C
ADD  R2, R1, A
DIV  R3, E, F
SUB  R4, D, R3
DIV  X, R4, R2
```

2-address instruction:

```
MOV  R1, C
MUL  R1, B
ADD  R1, A
MOV  R2, E
DIV  R2, F
MOV  R3, D
SUB  R3, R2
DIV  R1, R3
MOV  X, R1
```

1-address instruction:

```
LOAD E
DIV F
STORE R
LOAD D
SUB R
LOAD C
MUL B
ADD A
DIV R
STORE X
```

0-address instruction:

Postfix: ABC*+DEF/-/

```
PUSH A
PUSH B
PUSH C
MUL
ADD
PUSH D
PUSH E
PUSH F
DIV
SUB
DIV
POP X.
```

Instructions type: A computer has a set of instructions which can be classified as:

1. Data Transfer Instruction
2. Arithmetic Instructions
3. Logical Instructions
4. Shift and rotate Instructions
5. Program Control Instruction

1. Data Transfer Instruction:

Data transfer instructions perform the following data transfer operations:

- Data transfer between memory and CPU register.
- Data transfer b/w CPU and register.
- Data transfer b/w processor and input/output device.

Instruction	Mnemonic	Description
Load	LD	It transfers data from specified memory location to the processor register, usually accumulator.
Store	ST	It transfers data from processor register (usually accumulator) to the specified memory location.
Move	MOV	It transfers data between processor register and memory or between two memory words.
Exchange	XCH	It swaps data b/w two registers or a register and a memory word.
Pop	POP	It transfers data from stack memory (Top of stack) to the processor register.

2. Arithmetic Instructions:

Instruction	Mnemonic	Description
Add	ADD	It adds the contents of two operands.
Subtract	SUB	It subtracts the contents of two operand.
Increment	INC	It adds 1 to the value stored in a register or memory word.
Decrement	DEC	It subtracts 1 from the value stored in a register or memory word.
Multiply	MUL	It multiplies the contents of two operands.

3. Logical Instructions:

Instruction	Mnemonic	Description
AND	AND	It logically ANDs the individual bits of the operands.
OR	OR	It logically ORs the individual bits of the operands.
Clear	CLR	It causes the specified operand to be replaced by 0's.
Complement	COM	It gives 1's complement of the specified operand.
Set carry	STC	It sets the carry bit to 1.
Clear carry	CLC	It resets the carry bit to 0.

4. Shift and Rotate Instructions

Instruction	Mnemonic	Description
Logical shift left	SHL	It shifts the contents of the specified register 1-bit position towards left and fills vacant bit (rightmost bit) with 0.
Logical shift right	SHR	It shifts the contents of the specified register 1-bit position towards right and fills vacant bit (leftmost) with 0.
Rotate right	ROR	It rotates (circular-shifts) right the contents of specified register.
Rotate left	ROL	It rotates (circular-shifts) left the contents of specified register.
Arithmetic shift left	ASHL	It shifts the contents of specified register 1-bit position towards left and fills vacant bit (rightmost) with zero.

5. Program Control Instructions

Instruction	Mnemonic	Description
Branch	BR	It transfers program control to the specified address.
Jump	JMP	It transfers program control.
Skip	SKP	It skips the next instruction.
Call	CALL	It saves the address of the next instruction in the stack and transfers the program control to the specified address.
Return	RET	It reads the address from the top of stack and transfer the program counter to the read address.

Micro-operation:- To fetch, decode and execute cycles, the processor unit has to perform a set of operations called micro-operation. These operations include -

1. Transfer a word of data from one CPU register to the another.
2. Perform the arithmetic or logic operations on the data from the CPU registers and store the result in a CPU register.
3. Fetch a word of data from specified memory location and load them into a CPU register.
4. Store a word of data from a CPU register into specified memory location.

Micro-code:- The translation of symbolic micro-operation to binary produces a binary micro-program called micro-code.

Microprogram:- A sequence of one or more micro-operations designed to perform specific operation is called microprogram.

Control memory:- A memory that is part of control unit is referred to as a control memory. It stores the sequence of micro-operations to be performed to execute micro-instruction.

Microprogram Sequencer:- It is a sub-unit of microprogram control unit which presents an address to the control memory is called microprogram sequencer. The next address logic of the sequencer determines the specific address source to be loaded into the control address register. The block diagram of microprogram sequencer that selects one address from four sources and routes it into control address register (CAR).

The output from CAR provides the address for the control memory. The contents of CAR are incremented and applied to the multiplexer and to the stack register file. The register selected in the stack is determined by the stack pointer.

Inputs I_2, I_1 and I_0 & T derived from the CD and BR fields of microinstruction specify the operation for the sequencer. They specify the input source to the multiplexer also generate push and pop signals required for stack pointer operation.

Stack pointer is a 3-bit register and it gives the address of the stack file consist of $(2^3 = 8)$ eight registers. It initially, the stack pointer is cleared and point address zero in the stack. During POP operation, it is possible to write data into the stack at the address specified by the stack pointer. After data is written, stack pointer is incremented by 1 to get ready for the next push operation.

I_2	I_1	I_0	T	S_1	S_0	Operation	Description
X	0	0	X	0	0	CAR ← EAR	Transfer external address
1	0	1	1	0	1	CAR ← BAR, SP ← SP+1	Branch to subroutine and save the next instruction address in stack (Push operation)
0	0	1	1	0	1	CAR ← BAR	Transfer branch address
X	1	0	0	1	0	CAR ← SR	Transfer from stack register
0	1	1	1	1	1	CAR ← CAR+1	Increment address

Typical microoperation sequencer organization

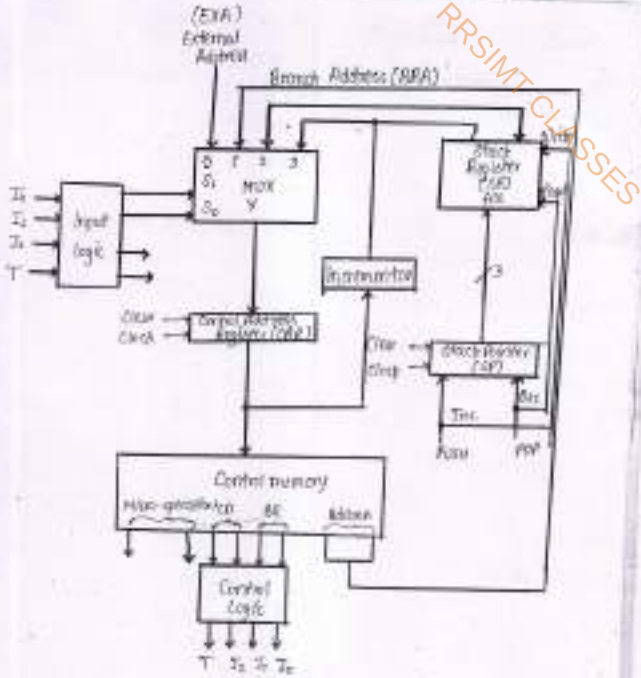


Fig. Typical microprogram sequencer organization
Address selection in control memory

Pre-fetching micro-instruction:- The disadvantage of micro-program control is that it results slower operating speed because of the time it takes to fetch micro-instruction from the control memory. This problem can be solved by pre-fetching the next micro-instruction while the current one is being executed. In this technique, the execution time can be overlapped with the fetch time.

Sometimes the address of the next micro-instruction is determined from the status flag and from the result of currently executed micro-instruction. In such cases, pre-fetching of micro-instruction sometimes fetches a wrong micro-instruction. So, in such case, the fetch must be repeated with correct address which require more complex hardware.

The pre-fetching technique is used to increase execution speed.

Comparison b/w hardwired control and micro-programmed control

S.No.	Characteristics	Hardwired control	Micro-programmed control
1.	Speed	Fast	Slow
2.	Implementation	Hardware	Software
3.	Flexibility	Not flexible	Flexible
4.	Ability to handle large/complex instruction set	Somewhat difficult	Easier
5.	Ability to support operating system and diagnostic features	Very difficult	Easy
6.	Design process	Difficult for new operation	Easy
7.	Memory	Not used	Control memory used (CRAM or ROM)
8.	Chip area efficiency	Requires less	Requires more area
9.	Used in	RISC processor	CISC processor
10.	Instruction generation	On the basis of input signal	On the basis of control line
11.	Control function	Implemented in hardware	Implemented in software
12.	Instruction set size	Usually under 100 instructions	Usually over 100 instructions
13.	ROM size	-	1K to 10K by 20-bit micro instructions

Hardwired Control Unit :-

In the hardwired control unit, uses fixed logic circuit to interpret instructions and generate control signals from them. In a hardwired control unit, all the control signals are generated by the hardware device.

The fixed logic circuits use contents of the control step counter, contents of instruction register, contents of the condition code flag and the external input signal. The fixed logic circuit block includes combinational circuit (decoder and encoder) that generates the required control outputs, depending on the state of all its inputs.

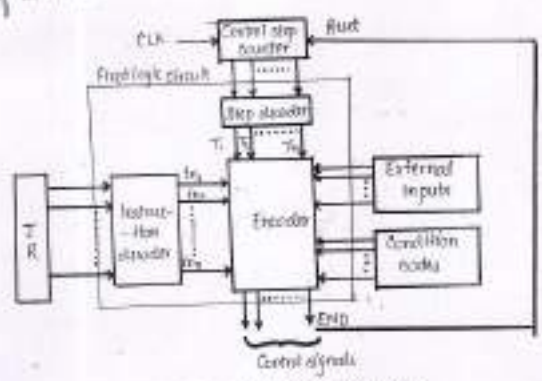
The instruction decoder decodes the instruction loaded in the IR. If IR is an 8-bit register then decoder generates $2^8 = 256$ lines, one for each instruction.

According to the code in the IR, only one line among all output lines of decoder goes high, that is, 1 and remaining lines are set to 0.

The step decoder provides a separate signal line for each step, time, in a control sequence.

The encoder gets the input from the instruction decoder, step decoder, external inputs and condition codes. It uses all these inputs to generate the individual control signals.

After execution of each instruction, end signal is generated which resets control step counter and make it ready for generation of next instruction.



Advantages of Hardwired Control Unit

1. Hardwired control unit is fast because control signals are generated by combinational circuits.
2. The delay in generation of control signals depends upon the no. of gates.
3. It has greater chip area efficiency since it uses less area on-chip.

Disadvantages of Hardwired Control Unit

1. More the control signals required by CPU, more complex will be the design of control unit.
2. Modifications to control signal are very difficult. That means it requires rearranging of wires in the hardware circuit.
3. It has greater chip area efficiency.

Microprogrammed Control Unit / Software Based Control Unit

Microprogramming is a method of control unit design in which the control signal selection and sequencing information is stored in control memory.

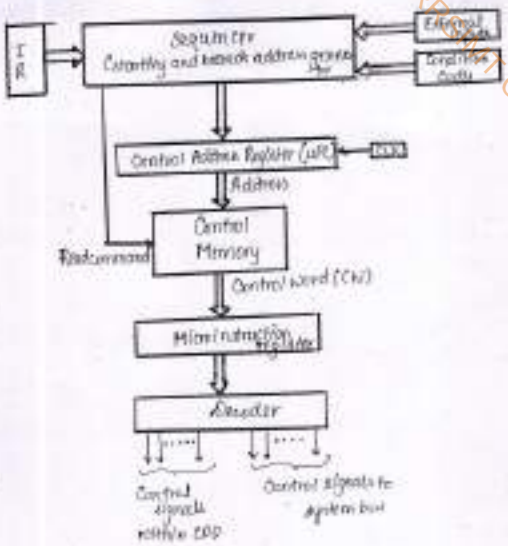
The control signals to be activated at any time which are specified by a microinstruction which is fetched from control memory.

The micro-programs are stored in control memory and their addresses are generated by microprogram sequencer.

The components of microprogram control unit works together as follows:-

1. The control address register (μPC) holds the address of next micro-instruction to be read.
2. Every time a new instruction is loaded into IR.
3. When address is available in control address register, the sequencer issues READ command to the control memory.
4. After issue of READ command, the word from the addressed location is read into the micro-instruction register.
5. The μPC is then automatically incremented by the clock.
6. The content of micro-instruction register generates control signals which are delivered to various parts of processor in the correct sequence.

a. It is difficult to correct mistake in original design or adding new feature in existing design of control unit.



Micro Instruction:- A each word in the control memory is a micro instruction which specify the control signals to be activated to perform one or more micro-operation. And instruction that control the data flow and sequencing in a processor and a more fundamental level of machine instruction is known as micro instruction.

x A micro instruction is usually consist of 4 parts x

A micro instruction is usually consists of four parts-

(1) Micro instruction field denoted as F_1, F_2, F_3

(2) Condition branching (CD)

(3) Branch field (BR)

(4) Address field (AD)

The general format of micro-instruction is as follows-

F_1	F_2	F_3	CD	BR	AD

Techniques of grouping control signals / List 2-techniques / Difference b/w Horizontal & Vertical Micro/ Types of Micro instruction

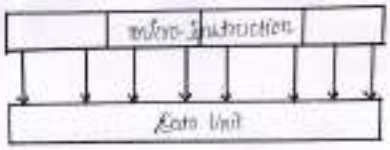
The grouping of control signals can be done using two techniques-

1. Vertical Micro Instruction Organization
2. Horizontal Micro Instruction organization

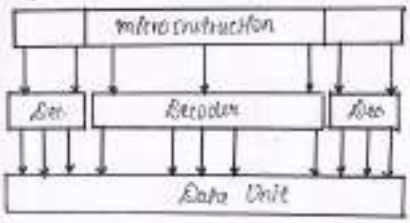
1. Horizontal Micro Instruction Organization:- With the help of directed binary format we can represent the control signals in the horizontal micro-program. Here n- bit encoding is needed n-control signals.

A higher degree of parallelism is provided by the horizontal organization.

With the help of single control point, each bit is identified in the horizontal microprogramming.



Vertical Micro Instruction Org^n:- In contrast to, the horizontal micro-programming a higher degree of encoding and variable format can be applied in the vertical microprogramming organization. With the help of vertical organization, we can minimize the length of micro instruction as well as prevent the length of micro instruction from being directly affected by the increasing memory capacity.



S.No.	Horizontal μ-programmed CU		Vertical μ-programmed CU
1.	It supports longer control word.	1.	It supports shorter control word.
2.	It allows a higher degree of parallelism. If degree is n then n control signals are enabled at a time.	2.	It allows a low degree of parallelism. i.e. the degree of parallelism is either 0 or 1
3.	No additional hardware is required.	3.	Additional hardware in the form of decoders is required to generate control signals.
4.	It is faster than a vertical micro-programmed control unit.	4.	It is slower than a horizontal micro-programmed control unit.
5.	It is less flexible than a vertical micro-programmed control unit.	5.	It is more flexible than a horizontal micro-programmed control unit.

Execution of a complete instruction

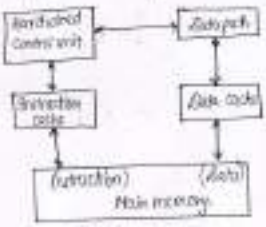
As we know that for performing any micro-operation, we have to follow instruction cycle. Execution is a sub-cycle of instruction cycle.

Let us find the complete control sequence for execution of the instruction ADD R_3, R_1. This instruction adds the contents of register R_1 and the contents of memory location specified by the register R_3 and store result in the memory location pointed by R_3. To execute this instruction, it is necessary to perform following actions:-

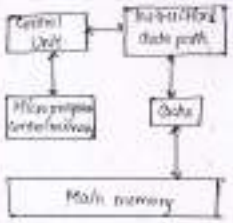
1. Fetch the instruction.
2. Fetch the operand from memory location pointed by R_3.
3. Perform the addition.
4. Store the result memory location pointed to the R_3.

S.No.	RISC (Reduced instruction Set Computer)	CISC (Complex instruction Set Computer)
1.	Multiple register sets, often consisting of more than 256 registers.	Single register set, often consisting of less than 16 registers total.
2.	Three registers operands allowed per instruction (for example- add R_1, R_2, R_3) Register mode	One or two register operands allowed per instruction (for example - add R_1, R_2) Memory mode

	RISC	CISC
3.	Parameter passing through efficient on-chip register windows	Parameter passing through inefficient off-chip memory
4.	Single-cycle instructions (except for load and store)	Multiple-cycle instructions.
5.	Hardwired control.	Micro-programmed control
6.	Highly pipelined / power supply requirement	Less pipelined / power supply requirement
7.	Simple instructions are few in number.	There are many complex instruction.
8.	Fixed length instructions.	Variable length instruction
9.	Complexity in compiler	Complexity in microcode.
10.	Only load and store instructions can access memory	Many instructions can access memory.
11.	Few addressing modes.	Many addressing modes.



RISC Architecture



CISC Architecture

Pipelining:- The term pipelining refers to the temporary overlapping of processing.

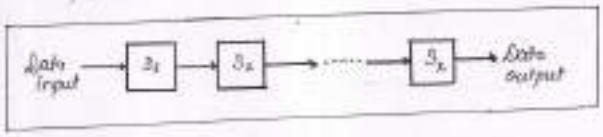
To achieve pipelining, one task must be divided into the sequence of sub-task, each of which can be executed by specialised hardware.

Pipelining is an implementation technique where multiple instructions are overlapped in execution.

The computer pipeline is divided in stages, each stage completes a part of an instruction in parallel.

Pipelining is a technique of decomposing a sequential process into sub-operations, each sub-process being executed in a special dedicated segment that operates concurrently with the all other segments.

Pipelining is a process of arrangement of hardware components of the CPU such that its overall performance is increased. Many instructions will be executed at a time in a pipeline processor i.e. more than one instruction can execute at a time in a pipeline processor.



Basic structure of pipeline processor

Eg: Perform the arithmetic operation $(A_i * B_i) + (C_i * D_i)$, specify a pipeline configuration to carry out the task. List the contents of all registers in a pipeline for $i = 1$ through 6.

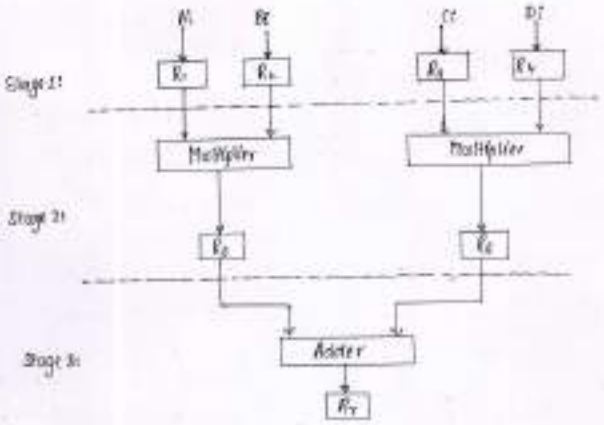


Fig: Pipelining processing

Stage 1: $R_1 \leftarrow A_i$ $R_2 \leftarrow B_i$ $R_3 \leftarrow C_i$ $R_4 \leftarrow D_i$

Stage 2: $R_5 \leftarrow R_1 * R_2$, $R_6 \leftarrow R_3 * R_4$

Stage 3: $R_7 \leftarrow R_5 + R_6$

Clock Pulse Number (i)	Stage 1				Stage 2		Stage 3
	R_1	R_2	R_3	R_4	R_5	R_6	R_7
1.	A_1	B_1	C_1	D_1	-	-	-
2.	A_2	B_2	C_2	D_2	$A_1 * B_1$	$C_1 * D_1$	-
3.	A_3	B_3	C_3	D_3	$A_2 * B_2$	$C_2 * D_2$	$A_1 * B_1 + C_1 * D_1$
4.	A_4	B_4	C_4	D_4	$A_3 * B_3$	$C_3 * D_3$	$A_2 * B_2 + C_2 * D_2$
5.	A_5	B_5	C_5	D_5	$A_4 * B_4$	$C_4 * D_4$	$A_3 * B_3 + C_3 * D_3$
6.	A_6	B_6	C_6	D_6	$A_5 * B_5$	$C_5 * D_5$	$A_4 * B_4 + C_4 * D_4$
7.	-	-	-	-	$A_6 * B_6$	$C_6 * D_6$	$A_5 * B_5 + C_5 * D_5$
8	-	-	-	-	-	-	$A_6 * B_6 + C_6 * D_6$

Table: Contents of all registers in the pipeline for $i = 1$ through 6

Types of pipelining

1. **Arithmetic pipelining:-** The ALU can be segmented for pipeline operation in a various data formats.
2. **Instruction pipelining:-** In this, a stream of instructions can be executed by overlapping fetch, decode and execute phase of an instruction cycle.
3. **Processor pipelining:-** This refers to pipeline processing of the same data stream by cascade of processors each of which processes a specific task.

Models of Pipelining

1. Asynchronous Pipeline model:- In asynchronous pipeline models, data flow among the pipeline stages is controlled by handshaking protocol. Every segment uses different - different clock pulse.

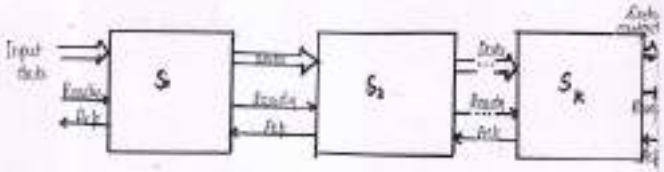
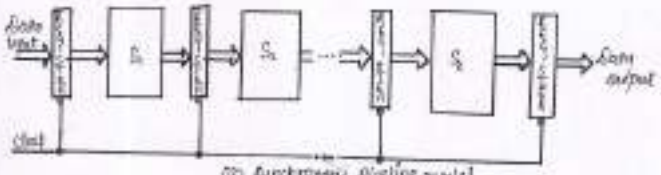
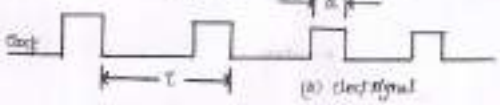


Fig 4.3.3 Asynchronous pipeline model

2. Synchronous Pipeline model:- In synchronous pipeline model clock high speed registers are used to interface two stages. At the falling edge of clock pulse the registers transfer data to the next stage simultaneously.



(a) Synchronous pipeline model



(b) Clock signal

Q:- Draw a space time diagram for a segment 6 stage pipeline representing the time. Each stage has eight tasks.

Sol:-

Time → / Stages	1	2	3	4	5	6	7	8	9	10	11	12	13
Stage 1	T_1^1	T_1^2	T_1^3	T_1^4	T_1^5	T_1^6	T_1^7	T_1^8					
Stage 2		T_2^1	T_2^2	T_2^3	T_2^4	T_2^5	T_2^6	T_2^7	T_2^8				
Stage 3			T_3^1	T_3^2	T_3^3	T_3^4	T_3^5	T_3^6	T_3^7	T_3^8			
Stage 4				T_4^1	T_4^2	T_4^3	T_4^4	T_4^5	T_4^6	T_4^7	T_4^8		
Stage 5					T_5^1	T_5^2	T_5^3	T_5^4	T_5^5	T_5^6	T_5^7	T_5^8	
Stage 6						T_6^1	T_6^2	T_6^3	T_6^4	T_6^5	T_6^6	T_6^7	T_6^8

T_j^i — j → segment / process; i → subprocess of segment.

Ques:- Draw a time space diagram for no. of processes = 4 and it can be decomposed in 5 sub-process.

Time → / Stages	1	2	3	4	5	6	7	8
Stage 1	T_1^1	T_1^2	T_1^3	T_1^4	T_1^5			
Stage 2		T_2^1	T_2^2	T_2^3	T_2^4	T_2^5		
Stage 3			T_3^1	T_3^2	T_3^3	T_3^4	T_3^5	
Stage 4				T_4^1	T_4^2	T_4^3	T_4^4	T_4^5
Stage 5					T_5^1	T_5^2	T_5^3	T_5^4

Unit :- 04

-:- Memory -:-

Page Replacement Algorithm

7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

FIFO

	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
1	7	7	7	2	2	2	2	4	4	4	0	0	0	0	0	0	0	7	7	7
2		0	0	0	3	3	3	2	2	2	2	3	3	3	2	2	2	2	0	0
3			1	1	1	0	0	0	3	3	3	3	3	2	2	2	1	1	1	1
				①							④	⑤					⑥	⑦		

Page hit = 5

Page fault = 20 − 5 = 15

LRU

	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
1	7	7	7	2	2	2	2	4	4	4	0	0	0	1	1	1	1	1	1	1
2		0	0	0	0	0	0	0	2	2	2	3	3	3	2	2	2	7	7	7
3			1	1	1	3	3	3	3	3	3	3	2	2	2	0	0	0	0	0
				①		②		③	④		⑤		⑥	⑦		⑧		⑨		

Page hit = 8

Page fault = 20 − 8 = 12

Optimal

	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
1	7	7	7	2	2	2	2	2	2	2	2	2	2	2	2	2	2	7	7	7
2		0	0	0	0	0	0	4	4	4	0	0	0	0	0	0	0	0	0	0
3			1	1	1	3	3	3	3	3	3	3	3	1	1	1	1	1	1	1
				①		②		③	④		⑤			⑥				⑦		

Page hit = 11 fault = 20 − 11 = 9

① 1 2 3 4 5 5 3 4 1 6 7 8 7 8 9 7 8 9 5 4 5 4 2

FIFO 1 1 1 1 5 5 6 5 5 5 5 8 8 8 8 8 8 8 8 8 8 2
2 2 2 2 2 2 2 1 1 1 1 1 9 9 9 9 9 9 9 9 9
3 3 3 3 3 3 2 2 2 6 6 6 6 6 6 6 5 5 5 5 5
4 4 4 4 4 4 4 7 7 7 7 7 7 7 7 4 4 4 4
④ ⑤ ⑥ ⑪ ⑫ ⑬ ⑭ ⑮ ⑯ ⑰

Page hit = 10

Page fault = 23 − 10 = 13

LRU 1 1 1 3 0 5 5 3 4 1 6 7 8 7 8 9 7 8 9 5 4 5 4 2
1 1 1 1 5 5 5 5 5 6 6 6 6 6 6 6 6 5 5 5 5 5
2 2 2 2 2 2 2 1 1 1 1 1 9 9 9 9 9 9 9 9 9
3 3 3 3 3 3 3 3 7 7 7 7 7 7 7 7 4 4 4 4
4 4 4 4 4 4 4 4 6 8 8 8 8 8 8 8 8 8 8 2
⑤ ⑥ ⑦ ⑨ ⑩ ⑪ ⑫ ⑬

Page hit = 8

Page fault = 23 − 8 = 15

Optimal 1 1 1 3 4 5 5 3 4 1 6 7 8 7 8 9 7 8 9 5 4 5 4 2
1 1 1 1 1 1 1 1 1 7 7 7 7 7 7 7 7 4 4 4 4
2 2 2 2 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
3 3 3 3 3 3 3 3 6 8 8 8 8 8 8 8 8 8 8 2
4 4 4 4 4 4 4 4 4 6 6 6 6 6 6 6 6 6 6 6
⑤ ⑥ ⑦ ⑧ ⑨ ⑩ ⑪ ⑫

Page hit = 12

Page fault = 23 − 12 = 11

	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
FIFO	1	1	1	4	4	4	5	6	6	6	1	2	3	3	3	2	1	2	3	6
		2	2	2	2	1	1	2	2	2	2	3	7	7	7	7	1	1	1	1
			3	3	3	3	5	5	5	1	1	1	1	6	6	6	6	6	3	3
				⑤					⑤					⑤			⑤			

no. of hits = 9

page faults = 20 − 4 = 16

	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
LRU	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
	1	1	1	4	4	4	5	5	5	1	1	1	7	7	7	2	2	2	2	2
		2	2	2	2	2	6	6	6	6	2	3	3	3	3	3	3	3	3	3
			3	3	3	1	1	1	2	2	2	2	6	6	6	1	1	1	6	6
				⑤					⑤					⑤				⑤	⑤	

no. of hits = 5

page faults = 20 − 5 = 15

	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
optimal	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
	1	1	1	1	1	1	1	1	1	1	1	3	3	3	3	3	3	3	3	3
		2	2	2	2	2	2	2	2	2	2	2	7	7	7	2	2	2	2	2
			3	4	4	4	5	6	6		6	6	6	6	6	7	1	1	1	6
				④	⑤				③	④	⑤				⑤	⑥			⑤	⑥

no. of hits = 9

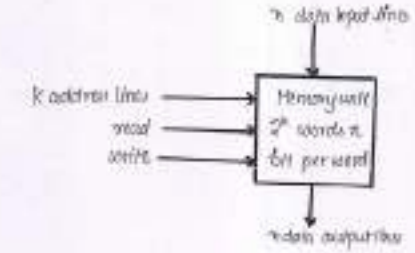
page faults = 20 − 9 = 11

Memory :- It is an electronic circuit that allow data to be stored and retrieve when required.

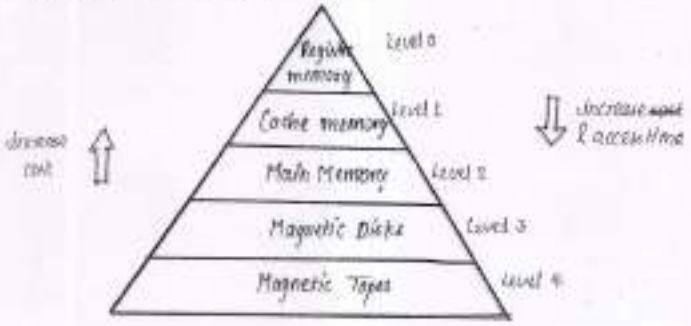
Memory unit that communicate directly with the CPU is known as main memory.

The storage device that provides backup storage is known as auxiliary memory.

Memories are made up with registers, each register holds/store k-bit data. Each location in memory is identified by an address. The total no. of bit that a memory can store is its capacity.



Memory Hierarchy



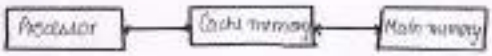
As moving down in the hierarchy, following occurs.

1. Increasing capacity
2. Increasing access time
3. Decreasing cost per bit

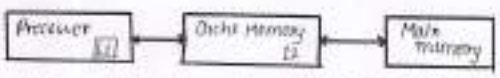
Thus, we can say that memory device at lower level is slower to access, smaller in size as compared to higher level.

Cache Memory:- It is defined as a very high speed memory used in computer system to compensate, the speed difference between the main memory access time.

A very high speed memory & cache memory is used to increase the speed of processing by making current program and data available to CPU at rapid rate.



It is also possible to place a smaller cache b/w cache memory and processor. This two cache resides in processor.



L1 is faster cache because it resides in a processor and smaller in size. Another cache system is the split cache that separated data cache memory. In this cache, a processor will use one cache to store code and another one store data.

Split Cache Organization

When the CPU wants to access memory, the cache is examined first if the word is found in cache is known as cache hit. If it is not available in cache then the desired memory block is copied in cache memory then it is used. This condition referred as cache miss.

The percentage of accesses where the processor finds the code or data word it needs, in the cache memory is called Hit Rate or Hit Ratio.

$$\text{Hit Rate} = \frac{\text{No. of hit}}{\text{Total No. of Cycle}} * 100$$

Ques. The application program in a computer system with cache use 400 instruction acquisition cycle from cache memory and 100 from main memory. What is the hit ratio? If the cache memory operate with zero wait state and the main memory has 3 cycle wait state what is the avg. no. of wait state during the program execution?

Sol:- no. of hits = 400

total no. of cycle = 400 + 100

Hit rate = $\frac{400}{400+100} * 100$

= $\frac{400}{500} * 100$

Hit Rate = 80%

Total no. of wait state = 400 * 0 + 100 * 3 = 300

Avg. wait state = $\frac{\text{Total no. of wait}}{\text{Total no. of cycle}}$

= $\frac{300}{500}$

= 0.6

There are various design issues in cache memory such as cache size, mapping techniques, block size, replacement algorithm and no of words per block.

Mapping Techniques in Cache

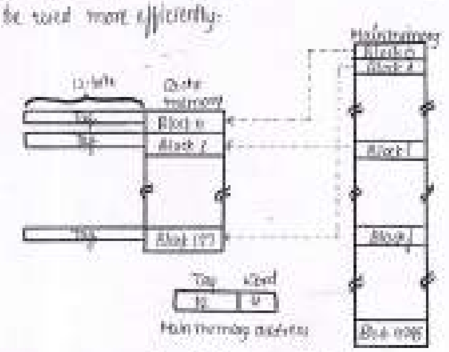
Mapping process is a method of transferring information from main memory to cache memory. There are three mapping process:-

1. Associative mapping process / Content - Based mapping
2. Direct mapping
3. Set Associative mapping

Associative mapping (Fully Associative mapping) / Content-Based

In this technique, a main memory block can be placed into any cache block position as there is no fix block, the memory address has only two fields - word and tag.

This technique gives complete freedom to choose the cache location in which to place the memory block. Thus the memory space in the cache can be used more efficiently.



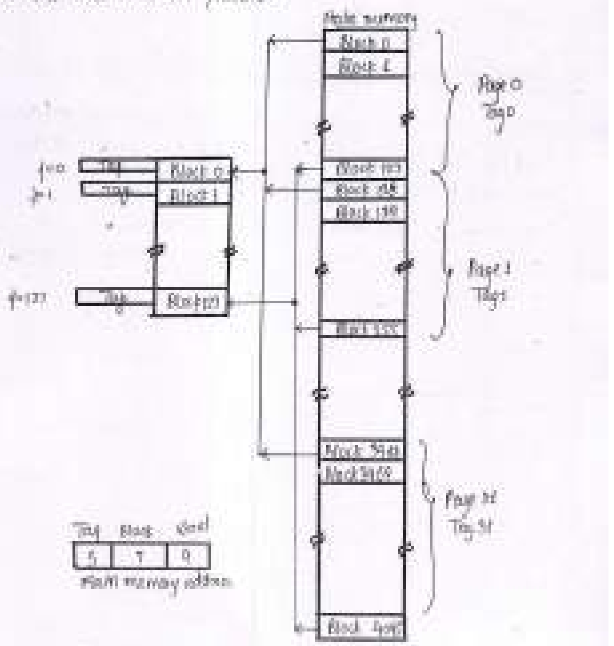
Associative mapped cache

S.No	Feature	Direct Mapping	Associative Mapping	Set-Associative
1.	Mapping Function	Simple modulo operation (Index)	Content-addressable memory (CAM).	Combination of direct and associative mapping, using a set index and associativity within the set.
2.	No of entries per set	1 (single)	Number of cache lines.	More than 1 (Typically a power of 2)
3.	Flexibility	Limited flexibility due to a fixed mapping	Flexible, any block can be placed anywhere in the cache.	Combine some flexibility with a structure for efficient searching.
4.	Search Mechanism	Direct access to a specific cache line using index.	Full search across all cache lines simultaneously	Search limited to a specific set, Parallel search within the set.
5.	Cache Hit Time	Fast due to direct mapping	Slower than direct mapping but faster than set associative.	Moderate, falls b/w direct and fully associative mapping.
6.	Cache Miss Handling	Simple, no competition for cache line.	Complex due to competition for cache lines among all blocks	Moderate complexity, limited competition within a set.
7.	Cost and complexity	Low cost, simple hardware	Higher cost, more complex hardware	Moderate cost and complexity.
8.	Example Use Case	Small embedded systems or limited-resource environments.	General-purpose systems with larger caches	Commonly used in modern processors for a balance of flexibility and performance.

Direct Mapping

It is the simplest mapping technique. In this technique, each block from the main memory has only one possible location in the cache organization. To implement such cache system, the address is divided into three fields - word field, block field and tag field.

The main drawback of direct map cache is that if processor needs to access same memory locations from two different pages of the main memory frequently, then only one of these locations can be in the cache at a time. Therefore, we can say that direct map cache is easy to implement but it is not flexible.



Difference b/w DRAM and SRAM

DRAM	SRAM
Constructed of tiny capacitors that leak electricity	Constructed of circuits similar to D flip-flops
Requires a recharge every few milliseconds to maintain its data.	Holds its contents as long as power is available.
Inexpensive	Expensive
Slower than SRAM.	Faster than DRAM.
Can store many bits per chip	Cannot store many bits per chip
Uses less power.	Uses more power.
Generates less heat	Generates more heat
Used for main memory	Used for cache.
Simple Structure - has a transistor and a capacitor	Complex structure - has flip flops.
Has a higher density	Has a lower density.

Set Associative Mapping

The set associative mapping is a combination of direct and associative mapping.

It contains several groups of direct map blocks that operate as several direct map cache in parallel.

The disadvantage of direct mapping is that two words of the same index but different tag cannot be stored at the same time into index but cache. As an improvement to this disadvantage, a third type of cache.

cache organization called set-associative mapping.
It is most flexible and give high hit ratio.
It is costlyto implement and hit ratio is increased.

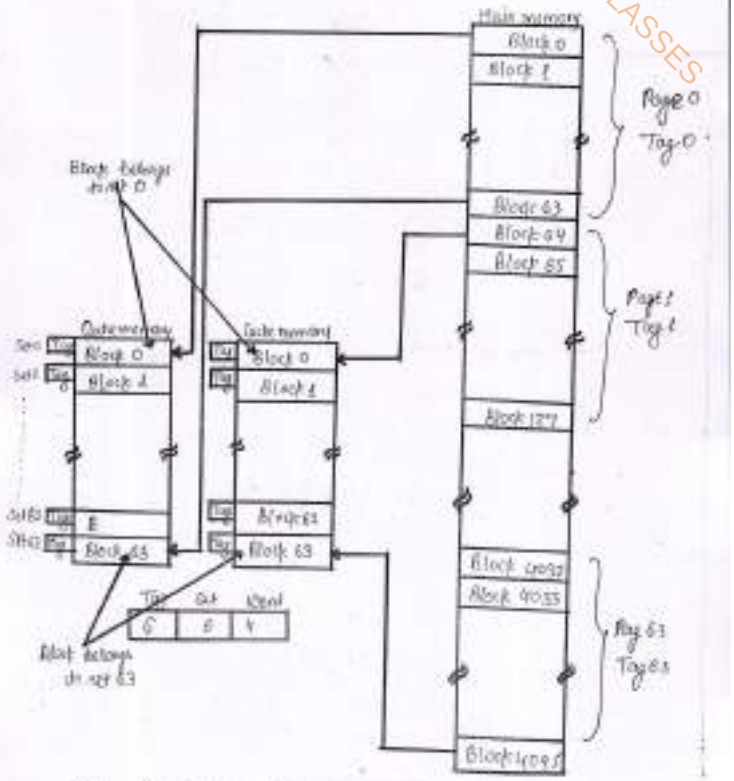


Fig:- Two-way set Associative Cache

Ques A block set associative cache consist of 64 blocks divided into 4 block set. The main memory contains 4096 blocks, each consist of 128 words of 16 bit length.

1. How many bit are there in main memory
2. How many bit are there in each of the tag, set and word field.

Sol: (1) no of bits in main memory = $4096 * 128 * 16$
$= 8388608$ bits

(2) main memory size = No of Block * no. of words per block
$= 4096 * 128$
$= 4k * 2^7$
$= 2^{12} * 2^7$
$= 2^{19}$
MM bit = 19 bit

No of words = 128
word bit = $2^7 = 7$ bit
cache size = 64 block
no of set = $\frac{\text{Total no. of block in cache}}{\text{no. of sets per block}}$
$= \frac{64}{4} = 16$
set bit = $2^4 = 4$ bit

Tag	Set	Word
8	4	7

MM bit
19 bit

Tag = MM bit − (WORD bit + set bit)
$= 19 - (4+7)$
Tag = 8

Ques 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 4 words. How many bits are there in tag, index, block and word field of the address format?

Sol:- Main memory size = $64K * 16$
$= 64 \times 2^{10} * 16$
$= 2^6 \times 2^{10} * 16$
MM bit = $2^{16} * 16$
MM bit = 16 bits

Cache memory size = 1K = 1024
no of word bits = no. of words
$= 4$
$= 2^2$
word bit = 2

no of block bits = $\frac{\text{cache size}}{\text{no of words per block}}$
$= \frac{1024}{4}$
$= 256$
$= 2^8$
no of block bits = 8 bits

No of Tag bit = 16 − (2+8)
$= 16 - 10$
Tag bit = 6

Tag	Block	word
6	8	2

16-bit

Ques A 4-way set associative cache uses blocks of 4 words. The cache can accommodate a total of 2048 words from the main memory. The main memory size is $128K * 32$. How many bits are there in tag, set and word in address format?

Sol: Main memory size = $128K * 32$
$= 2^7 * 2^{10} * 32$
$= 2^{17} * 32$
MM bit = 17 bits

no of word bits = no. of words
$= 4 = 2^2$
word bit = 2 bit

set bit =
no of set = $\frac{\text{Total no. of blocks in cache}}{\text{no. of sets per block}}$

no of block = $\frac{\text{cache size}}{\text{no of word per block}}$
$= \frac{2048}{4}$
no of blocks = 512
in cache

∴ no of set = $\frac{512}{4}$
no of set = 256
set bit = 2^8 = 8 bits

No of Tag bit = 17 − (2+8)
No of Tag bit = 7 bits

Tag	set	word
7	8	2

17 bits

Ques:- A direct map cache has the following parameters:

Cache size = 1K words

Block size = 128 words

main memory size = 64K words

Specify the no. of bits in tag, word field and block.

Sol:-

main memory size = 64K

$= 2^6 * 2^{10}$

$= 2^{16}$

MM bit = 16 bits

Cache size = 1K = 1024

Block size = 128

no. of words = 128

$= 2^7$

word bit = 7 bits

No. of blocks = $\frac{\text{cache size}}{\text{no. of words per block}}$

no. of blocks = $\frac{1024}{128}$

no. of blocks = 8

Block bit = $8 = 2^3$

Block bit = 3 bits

Tag bit = 16 - (7+3)

No. of Tag bits = 6 bit

Tag	Block	Word
6	3	7

16-bit

Ques:- A computer system has 4K word cache organized in block set associative manner with 4 block per set, 64 words per block. The main memory size contains 65536 blocks. Find many data are there in each of the tag, set and word field.

Sol:- main memory size = no. of blocks * no. of words per block

$= 65536 * 64$

$= 65\,K * 64$

$= 2^6 * 2^{10} * 2^6$

$= 2^{22}$

MM bit = 22 bit

no. of blocks = $\frac{\text{Cache size}}{\text{no. of words per block}}$

$= \frac{4K \times 1024\ B^{\vee}}{64_{B}}$

no. of blocks = 64

$= 2^6$

Block bits = 6 bits

No. of word bit = no. of words

$= 2^6$

$= 2^4$

word bit = 6 bits

no. of sets = $\frac{\text{no. of blocks}}{\text{no. of set per block}}$

$= \frac{64}{4}$

$= 16$

$= 2^4$

no. of set bits = 4 bits

No. of Tag bits = 22 - (4+6)

Tag = 12 bits

Tag	set	Word
12	4	6

22 bits

Classification of memory -

1. Based on principle of operation
   - a) Read Only Memory (ROM)
     - Mask ROM
     - PROM
     - EPROM
     - EEPROM
   - b) Read/Write Memory (RAM)
     - Static RAM
     - Dynamic RAM
2. Based on physical characteristics
   - a) Erasable
   - Non-erasable
   - volatile
   - non-volatile
3. Based on mode of access
   - a) Sequential
   - b) Random
4. Based on terminology used for fabrication
   - a) Bipolar
   - b) MOS

Secondary Storage Device

- Sequential access device
- Random Access
  - Magnetic disk
    - Floppy disk
    - Hard disk
    - Zip disk
  - Optical disk
    - CD ROM
    - WORM or read/write disk
  - Magnetic optical disk

Magnetic Disk:- It is an auxiliary memory that provides the bulk of secondary storage for modern computer system. A magnetic disk is a thin circular metal plate. It is coated with a thin magnetic film usually both sides.

- Digital information is stored on the magnetic disk by magnetizing the magnetic oxygen in a particular direction. Conceptually disk are relatively simple. Each disk plate has a flat circular shape shaped.
- Common plate diameter range is 5.8 to 5.25 inches. The disc surface of platter covered with magnetic material.
- Digital information can be stored on the magnetic field by applying current pulse of suitable to the magnetizing coil. The head are attached to disk arm that moves all the heads as unit.
- There may be thousands of cylinder in a disk drive and each track may contain hundreds of sectors.
- The storage capacity of commercial disk drive is in Gigabyte.

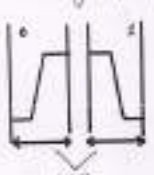
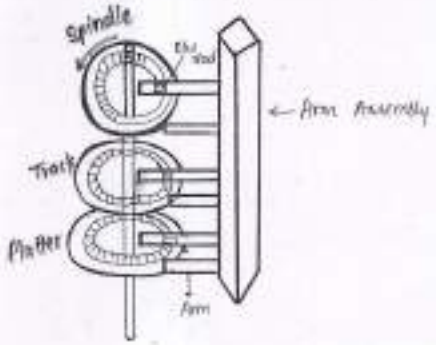
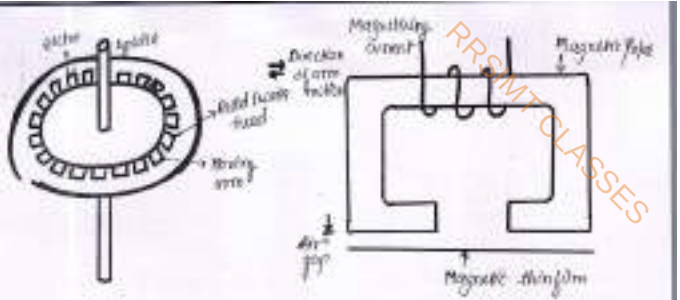


fig. Bit representation of magnetic disk



Mechanical structure of magnetic disk

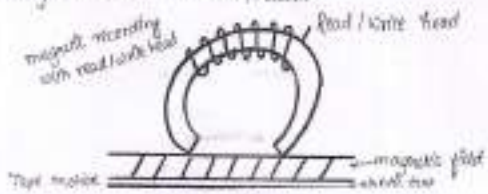


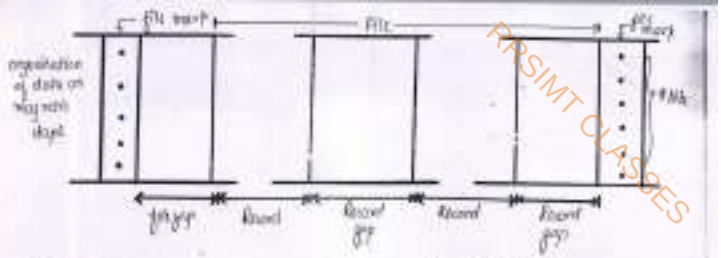
Magnetic Tape: Magnetic tape is one of the most popular storage medium for large data that are sequentially accessed and processed. The tape is formed by depositing magnetic film on a very thin (½ inches or ½ inches) wide plastic tape. The tape ribbon itself is stored in reels similar to tape. are automatically erased as new data is record in the same area.

The information is recorded on the tape with the help of read/write head. Although it is relatively permanent and can hold large quantity of data. its access time is slow compare with main memory and magnetic disk.

In addition, random access to magnetic tape is about a 1000 times slower than the random access to magnetic disk, so tapes are not very useful for secondary storage.

Tapes are mainly used for backup for data which is not frequently used. They can store 20GB - 40GB data.





Optical Disk: There are three basic types of optical disk:

1. CD ROM
2. DVD
3. WORM (Blue-Ray) [Write once Read Many]

CD's can store 700 MB of data. DVD's can store upto 8.4 GB of data and Blue ray which is newest technology of optical media can store upto 50GB of data.

This storage capacity is a clear advantage over floppy disk which only have the capacity of 1.44 MB.

Another advantage of this optical media over the profit is that it can read upto 4 times longer due to its durability.

Virtual Memory:

Virtual Memory technique that allow the execution of process that are not completely in main memory. The major advantage of this is, it can execute a program larger than memory as well as size of auxiliary memory.

When a program does not completely get into memory, it is divided into segments. Segments which are currently being executed are kept in main memory and the remaining segment are stored in the secondary storage.

If an executing program needs a segment which is not currently in the main memory, the required segment is copied from the main memory the secondary storage (swap in). When a new segment of a program is to be copied from main memory it must replace another segment (swap out).

In virtual memory, the address that processor issues to access either instructions or data are called virtual & logical address virtual memory can be implemented by two methods:

1. Paging
2. Segmentation

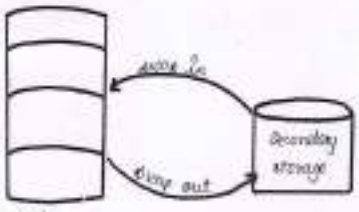


fig: Demand Paging

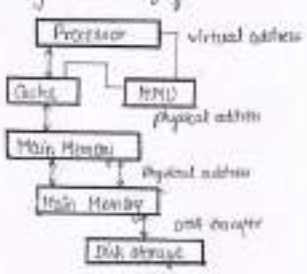


fig: Virtual Memory Organization

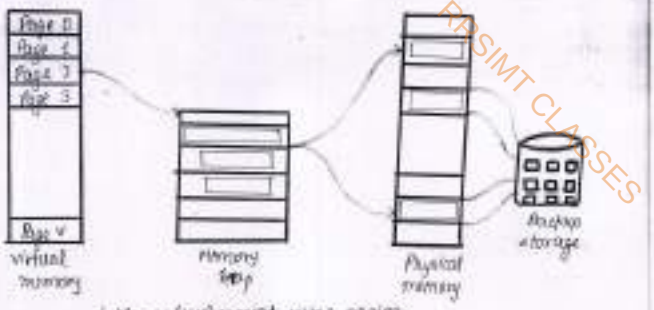


fig: Memory (VM) concept using paging

Address Translation in virtual Memory (Mapping process in virtual memory)

It involves two phases:

1. Segment Translation
2. Page Translation.

Segment Translation: A logical address consist of a selector and offset. A selector is the content of the segment register. Every segment selector has a linear base address associated with a segment descriptor. A selector is used to point a descriptor for the segment in description table. The linear base address from the descriptor is added to the offset to generate the linear address. This process is known as segmentation or segment translation.

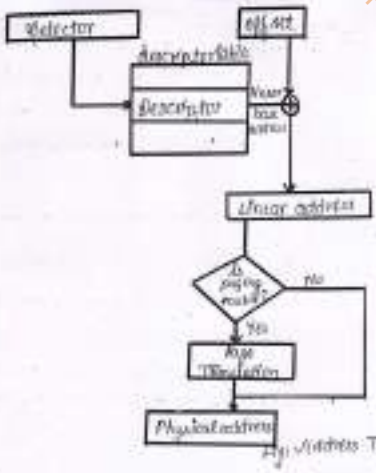
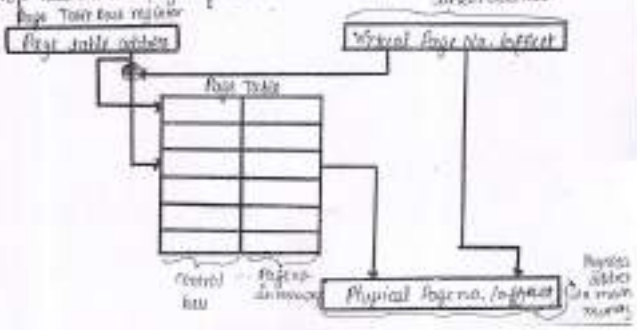


Fig: Address Translation.

Page Translation:- Page translation is the second phase of address translation, it transform a linear address generated by segment translation into physical address.

When paging is enabled, the linear address is broken into a virtual page no and page offset. The physical page no is not changed. The no. of bits in the page offset decides the page size.



Memory Management Hardware

A memory management system is a collection of hardware & software procedures for managing the various programs residing in the memory. Memory management system is a part of an operating system in computer. A collection of hardware components connected memory management unit.

The basic component of memory management unit are -

A mechanism for dynamic storage allocation and deallocation.

A mechanism for sharing common program stored in memory by different user.

A mechanism for protection of information against unauthorized access by user and preventing users from changing operating system functions.

Memory Protection

Memory protection is done by associating bits with the each page. These bits are kept in page table. A protection bit can define a page to be read / write, read only or hidden.

Protection bits can be checked to verify that whether any operation is done on read only. Every time a page is moved from one block to another. It would be necessary to update the block protection bit.

Page Replacement Policies

Page Replacement Policies are strategies used by operating system to decide which page to remove from memory when there is a page fault. Common policies include -

FIFO (First In First Out)

LRU (Least Recently Used)

Optimal Page Replacement

(1.) First In First Out (FIFO):- FIFO replaces the oldest page in the memory based on the order in which pages were brought into the memory.

It uses a simple queue data structure to keep track of the order in which pages were loaded.

While easy to implement, FIFO may not always perform well in terms of page hit rate.

Belady Anomaly:- As we know that when we increase the frame size, the page fault decreases and page hit increases.

As per Belady's conclusion, in a FIFO algorithm for a certain sequence of memory references for a certain time period when we increase the frame size, the page fault also increase. This will become known as Belady's Anomaly. **Eg:-** Consider the following memory reference sequence 1,2,3,4,1,2,5,1,2,3,4,5. Explore that these reference sequence suffer from belady's anomaly.

Ref:-	1	2	3	4	1	2	5	1	2	3	4	5
	1	1	1	4	4	4	5	5	5	5	5	5
		2	2	2	1	1	1	1	1	3	3	3
			3	3	3	2	2	2	2	2	4	4
							(F)	(H)	(H)		(F)	

Frame size = 3

Page hit = 3

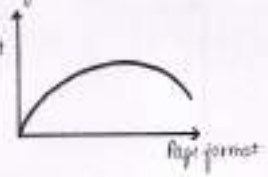
Page fault = 12-3 = 9

Frame size = 4

1	2	3	4	1	2	5	1	2	3	4	5
1	1	1	1	1	1	5	5	5	5	4	4
	2	2	2	2	2	2	1	1	1	1	5
		3	3	3	3	3	3	2	2	2	2
			4	4	4	4	4	4	3	3	3
						(F)	(F)				

Page hit = 2 Page fault = 10

Hence it suffers from Belady's anomaly because when we increase frame size, page fault also increases.



(2.) Least Recently Used (LRU):- LRU replace the page that has not been used for the longest time.

It depends on the principle that pages that have not been used recently are less likely to be used in the near future.

(3.) Optimal Page Replacement:- Optimal Replacement selects the page that will not be used for the longest period in the future.

It serves as a theoretical benchmark as it requires knowledge of the future page accesses.

It never suffers from Belady's Anomaly.

Thrashing:- Thrashing in computer system refers to a situation where a computer's performance drops significantly due to the constant swapping of data between the main memory and secondary storage.

This occurs when the system have high degree of multiprogramming, causing it to spend more time moving data between memory and storage than actually processing task. It leads to slow down in performance as the system struggles to memory resources.

Reasons of Thrashing

Thrashing occurs when a system is overloaded with too many tasks or processes, and the demand for physical memory exceeds the available RAM. This can happen due to:

Insufficient Physical Memory: If there is not enough RAM to handle the concurrent processes, the system has to depend on swapping data b/w RAM and the hard disk.

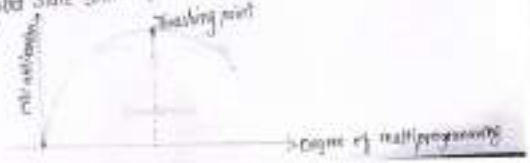
Inefficient Memory Management: Poorly optimized memory management algorithms can contribute to thrashing. For eg: If the system frequently accesses and evicts large chunks of data unnecessarily, it can lead to performance degradation.

Due Overcommitting: Allocating more virtual memory than system can practically handle result in thrashing. When the system tries to fulfill the excessive demand for virtual memory it spends more time on swapping.

Solutions of Thrashing

To remove or prevent thrashing consider these solutions:-

1. Increase Physical Memory
2. Optimize memory use
3. Use efficient paging algorithm
4. Adjust process priority.
5. Avoid overcommitting resources
6. Monitor and tune system performance
7. Consider solid state drives (SSD)



Locality of Reference

Locality of reference is a principle in memory management that suggests that when a program accesses a particular memory location, it is likely to access in the near future.

This principle is used for optimizing the performance of computer systems when few procedures, lines of codes that repeatedly call each other, these instructions are usually localized in cache memory. This is referred to as locality of reference.

There are two main types of locality of reference:-

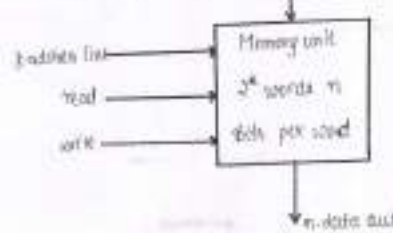
1. Temporal:- The temporal means that a recently executed instruction is likely to be executed again very soon. This temporal aspect of locality of reference suggests that whenever an instruction or data is first needed, it should be brought into the cache memory and it should remain there, until it is needed again.

2. Spatial:- The spatial means that instructions stored nearby the recently executed instructions are also likely to be executed soon.

The spatial aspect suggests that instead of bringing just one instruction from memory to be caught, it is wise to bring many instructions that reside at adjacent address as well.

2-D (2-Dimensional) Memory Organization

The general block diagram of memory organization is:-



Associative Memory (Content Addressable Memory / CAM)

A memory unit accessed by the content is called an associative memory or content addressable memory (CAM).

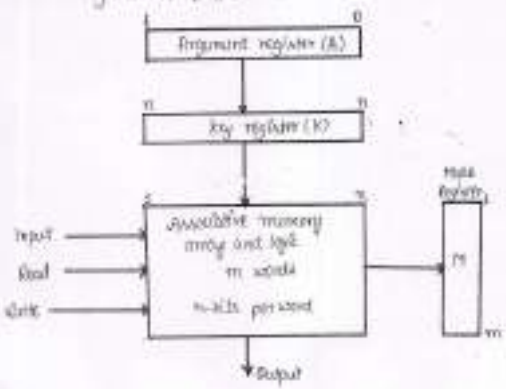
By using CAM, the time required to find an object in memory can be reduced.

In CAM, the memory is organised in such a way that the data itself serves as the lookup key.

This enables quick search and comparisons.

In a block diagram of an associative memory, consist of memory array with the match logic for n-bit words.

The argument register (A) and key register (K) each have n-bits per word. Each word in memory is compared in parallel with the content of argument register. The words that match with the word stored in argument register set a corresponding bit in the match register. Therefore, reading can be accomplished by sequential access to memory for those words whose corresponding bits in the match register have been set.



When there are k-address lines, then we can access 2^k words.

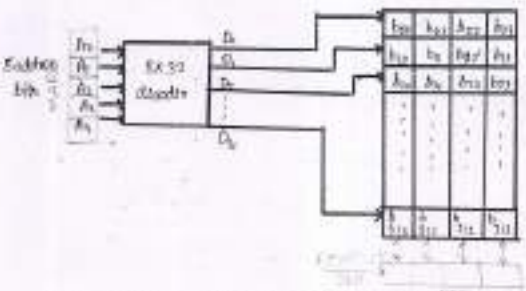
In 2D RAM organization a RAM of $2^n * m$ column 2^n memory words of m bits each.

It has a row select decoder of size $n * 2^n$ to select one out of 2^n words.

Each row (memory word) have m columns (one column for each individual bit).

In 2D RAM organization, hardware is fixed.

It requires more no. of logic gates. It is more complex because of large no. of AND and gates.



Here, we use 2 registers MAR and MDR. MAR holds the address of desired location from where we have to perform READ operation or we have to write memory word in memory.

MDR fetches all corresponding bits from memory in read operation and/or put the corresponding bits at particular address which is defined in MAR.

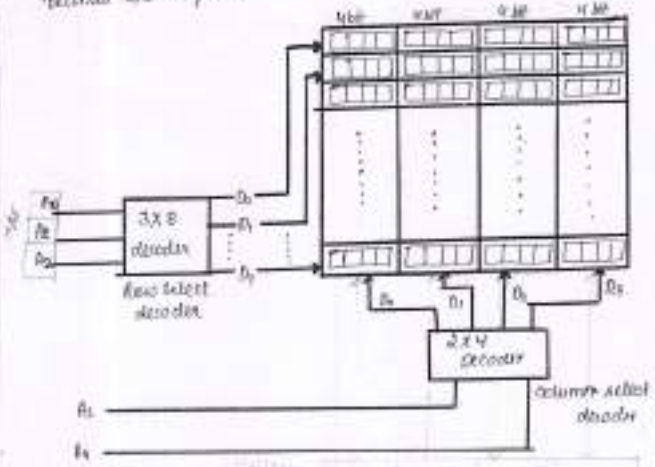
If we have to fetch one word from 1024 words then we have to use 10 x 1024 decoder which consists large no. of gates, circuit becomes complex. This drawback is reduced in 2-1/2 D RAM organization.

2½ / 2.5-D Organization

In 2½-D organization, the no. of address lines are divided into approximately equal size; one for row select and another for column select decoder.

In 3-D RAM, hardware is variable.

It requires less no. of logic gates because of less no. of gates circuit becomes less complex.



Instead of using one decoder (3x32), we use two decoders, one for row selection and another one for column selection. In above diagram we used 3x8 decoder for row selection and 2x4 for column selection.

Let us consider, if we pass address line in row decoder 000, it enables first row of memory segment and 00 in column decoder then it activates first column, finally first word of memory layout will be activated.

Speedup and Amdhal's Law

Amdhal's law is used to calculate the performance gain that can be obtained by improving some portion of a computer.

It states that the performance improvement to be gain from using some faster mode of execution is limited by the fraction of the time the faster mode can be used.

Speedup (performance improvement) tells us how much faster a task can be executed using the machine with the enhancement as compared to the original machine. It is defined as :-

$$\text{Speed up} = \frac{\text{performance for entire task using enhanced machine}}{\text{performance for entire task using original machine}}$$

$$\text{Speed up} = \frac{1}{(1-F_E) + \frac{F_E}{S_E}}$$

where F_E = fraction enhance

S_E = speed up enhance

UNIT 8-05

INPUT / OUTPUT

Input Output Interface :- Input Output Interface enables transfer of data btw internal storage and external input-output device.

Input output interface is a mechanism which provides a communication path btw processor and peripheral devices.

The purpose of use of input output interface is to resolve the differences btw the CPU and peripheral devices. These differences are :-

(1) As peripherals are electromagnetic and electromechanical devices and CPU is an electronic device, signal conversion becomes necessary wherever required.

(2) Due to difference in data transfer rate, synchronization of peripherals with the CPU may be required.

(3) Data codes and formats of each peripheral is different than CPU, a standard conversion is required.

(4) The operating modes of all peripherals are different. Thus, each of the peripherals connected to the CPU should be controlled so that the operation of the peripherals continue without any disturbance.

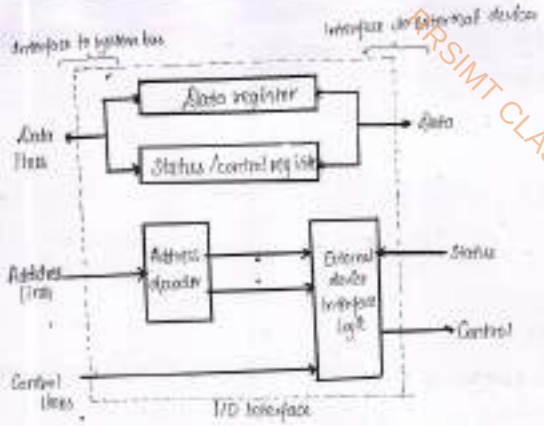
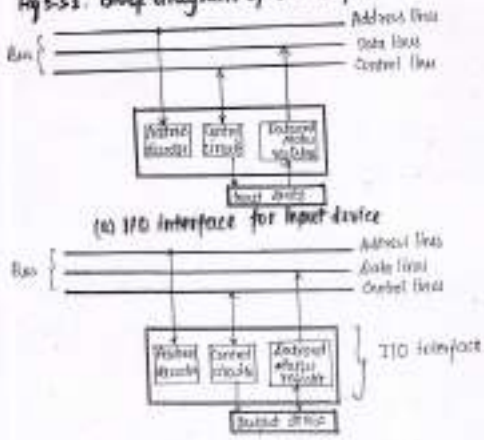


Fig 5.3.1. Block diagram of I/O interface



(a) I/O interface for input device

(b) I/O interface for output device

Functions of I/O Interface

1. **Control Timing:-** The input output interface coordinates the flow of data b/w internal and external storage or devices through the control timing signals.
2. **Processor Communication:-** Such communication involves different types of communication signals that are —
   1. Control signals from processor to I/O interface.
   2. Exchange of data b/w processor and I/O interface.
   3. Recognise the peripheral I/O among various I/O devices
3. **Device Communication:-** The I/O interface must be able to perform device communication which involves commands, status information and data
4. **Data Buffering:-** It is also an essential task due to the difference of speed (data transfer rate) of the peripherals
5. **Error Detection:-** The I/O interface is also responsible for error detection and reporting to the processor.

Input Output Processor

The Input Output Processor (IOP) has an ability to execute I/O instructions and execution of I/O operation. The I/O instructions are stored in main memory. When input output transfer is required, the CPU initiates an I/O transfer by instructing an I/O channel to execute program stored in main memory

/

the I/O processor can fetch and execute its own instruction.
The IOP supports multi processing environment. IOP and CPU can do processing simultaneously.
IOP does all work involved in I/O transfer including device setup, programmed I/O and DMA operation.

* Communication b/w CPU and IOP

The sequence of operations carried out during CPU and IOP communication are—

1. CPU checks the existence of I/O path by sending an instruction
2. In response to this, IOP puts the status word in the memory showing the condition of IOP (Busy, Ready etc.).
3. CPU checks the status word and if all conditions are OK, it sends the instruction to start I/O transfer along with the memory address where the IOP program is stored.
4. After this CPU continue another program.
5. IOP now conducts the I/O transfer using DMA and prepares status report.
6. On completion of I/O transfer, IOP now sends an interrupt request to the CPU.
7. The CPU responds to the interrupt by issuing an instruction to read the status from the IOP.
   The status indicates whether the transfer has been completed or any error occurred during the transfer.

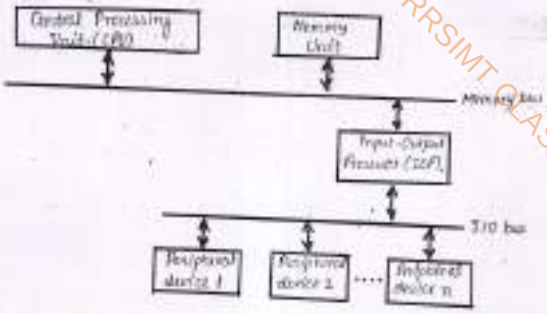


Fig 5.7.1. Block diagram of a computer with I/O processor

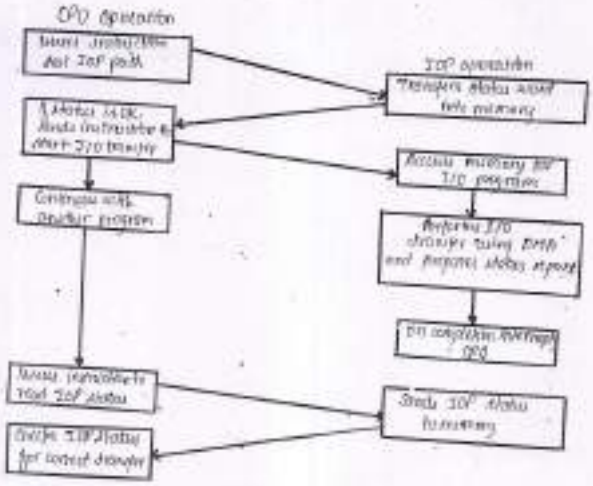


Fig 5.7.2. CPU and IOP communication

Asynchronous Data Transfer

The data transfer b/w two independent units by common clock or global clock is known as synchronous data transfer.

When both independent units are using their own private clock for data transfer then it is called asynchronous data transfer.

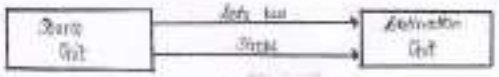
Two methods are used to transfer data in asynchronous method b/w two independent units :-

1. Strobe Control
2. Handshaking

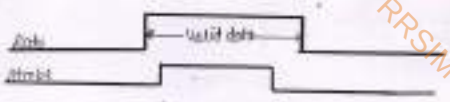
1. Strobe Control :- In strobe control asynchronous data transfer, a single control line (strobe pulse) is used to indicate to the other unit when the transfer has to occur. When the strobe signal may be activated by either the source or destination unit. There are two methods of strobe control.

1. Source Initiated strobe control data transfer

In this, the strobe signal informs the destination unit that the valid data is available on the data bus or from the source unit. The source unit first place the data on the data bus. After the data setup the source activates the strobe pulse. The strobe signal and data remain in active state for a sufficient time to allow the destination unit to receive the data. After that strobe is deactivated and source removes the data from the data bus.



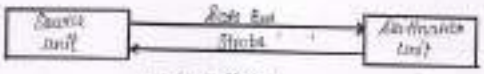
(a) Block diagram



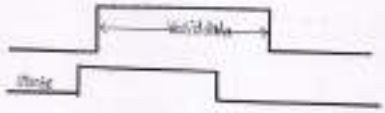
(b) Timing diagram

2. Destination Initiated strobe for data transfer

In this case first the destination unit activate the strobe pulse informing the source to provide the data. The source unit response by putting valid data on the data bus.



(a) Block diagram

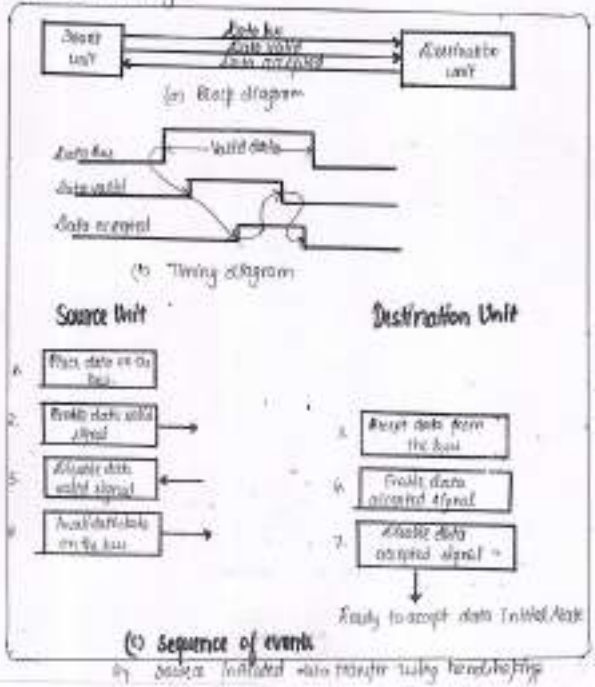


(b) Timing diagram

2. Handshaking :- The disadvantage of strobe control method is that the source unit that initiates the transfer has no idea whether the destination unit has actually accepted the data sent by it. Similarly a destination unit that initiates the transfer has no idea whether the source unit has actually placed the data on the bus. The handshake method solves this problem by introducing one or more signal called Acknowledge signal. The acknowledge signal provides a reply to the unit that initiates the data transfer.

-:- Source initiated data transfer using handshaking

In this method, source unit initiates the data transfer by placing the data on the bus and enabling its data valid signal. In response to this, the destination unit accepts the data and it sends data accepted (acknowledgement) signal to the source unit. The source unit then disables its data valid signal and destination unit disables data accepted signal and the system goes into its initial stage.

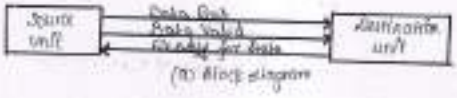


(c) Sequence of events

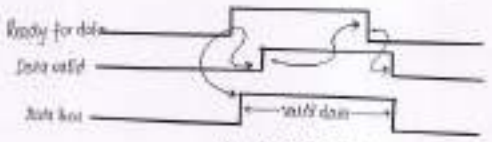
(a) Source initiated data transfer using handshaking

-:- Destination initiated data transfer using handshaking

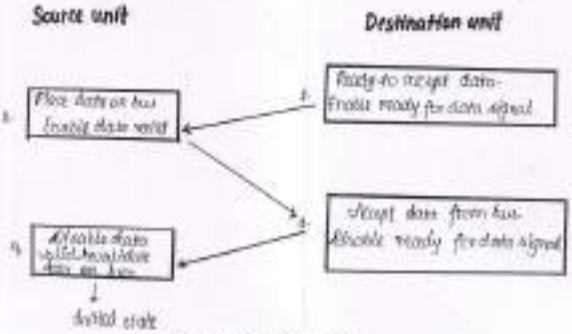
Here, the destination unit activates Ready for data signal when it is ready to accept data from source unit. In response to that, the source unit places the data on the bus and initiates the data valid signal. The destination unit then accepts the data from the data bus and disables the ready for data signal. Then the source unit disables data valid signal, after that system goes to the initial stage.



(a) Block diagram



(b) Timing diagram



(c) Sequence of events

Interrupt :- An interrupt is a process which allows the processor to suspend its current execution and respond to external or internal request.

Interrupt is the signal which allows the coordination of CPU from any program to continue for execution.

An interrupt can be provided to the processor in following two ways

① Internally by an instruction of a program

② Externally by any peripheral device.

Interrupt Handling:- The occurrence of an interrupt causes no. of events both in processor, hardware and software. Following sequence of events occurs:-

① Device issues an interrupt signal.

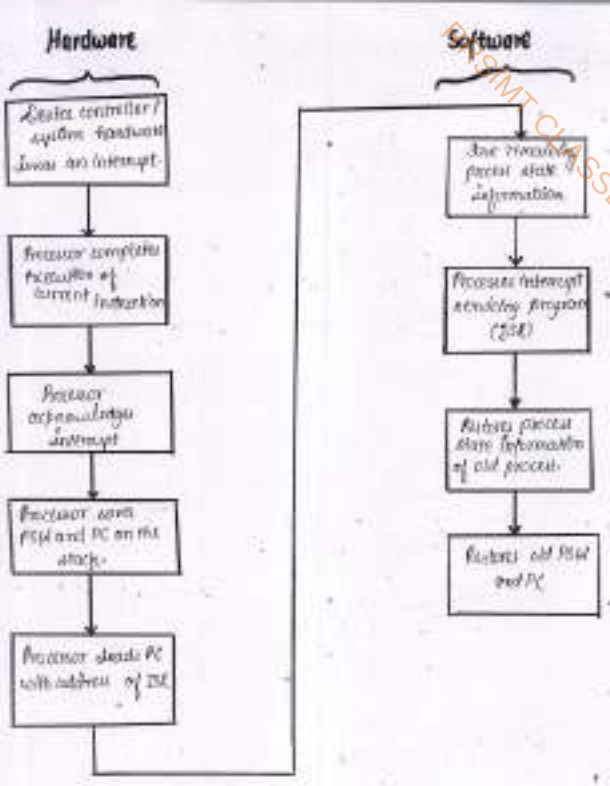
② Processor finishes/completes its current instruction before responding to the interrupt.

③ The processor checks the interrupt.

④ Processor transfers the control to the interrupt after saving the information of the current executing program in the registers called program status word. (PSW)

⑤ Processor executes the interrupt.

⑥ When interrupt processing is complete, the saved information is retrived from the PSW, then the normal execution resumes.



Types of Interrupts

1. Software & Hardware interrupt
2. Maskable and non maskable interrupt
3. Vector & non vectored interrupt

Software & Hardware Interrupt:- The interrupt signal generated from external devices and I/O devices are made interrupt to CPU when the instructions the ready is known as hardware interrupt.

Hardware interrupt can be internal (through microprocessor) and external (peripheral devices).

The interrupt signal generated from internal device and software program used to access any system call then, software interrupts are present.

Software interrupts can be divided into two types:-

① Normal Interrupt

② Exceptions

Maskable & Non-Maskable Interrupt:- Some interrupt request are required immediate response from the processor either serious damage may occur. They are programmed as non-maskable interrupt or enable interrupts.

Some interrupts are termed as maskable or disable interrupt because they may not be responded immediately or processor may reject it.

Vector & Non-Vectored Interrupt:- In a vectored interrupt, the branch address is assigned to the fixed location in the memory.

In a non-vectored interrupt the source supplies the branch information to the processor.

S.No	Vectored interrupt	Non-vectored interrupt
3.	Vectored interrupt are those interrupt that generates the interrupt request identifies itself directly to the processor	Non-vectored interrupt are those in which vector address is not predefined.
4.	Vectored interrupt has memory address.	A non-vectored interrupt do not have memory address.
5.	Vectored interrupt have fixed memory location for transfer of control for normal execution.	Non-vectored interrupt do not have fixed memory location for transfer of control for normal execution.
6.	The vectored interrupt allows the CPU to be able to decide what ISR to carry out in software.	When a non-vectored interrupt received, it jump into the program counter to fixed address in hardware.
7.	Response time is low.	Response time is high.
8.	TRAP is a vectored interrupt.	INTR is non-vectored interrupt.

Modes of Transfer

There are three modes of transfer:-

① Programmed I/O transfer

② Interrupt initiated I/O transfer

③ DMA (Direct memory Access)

Programmed I/O transfer:- In programmed I/O transfer, each data transfer is initiated by an instruction in a computer program.

In any computer system I/O operations are completely controlled by the processor when that system is said to be using programmed I/O.

When such a technique is used, processor executes program that initiate, direct and terminate the I/O operation, including checking device status, sending a read or write command and transferring the data.

It is the complete responsibility of the processor i.e., processor regularly checked monitoring of the input output interfaces and I/O.

No other task will be performed during the data transfer. It is a time consuming process. It keeps processor busy.

Difference b/w Serial and Parallel Communication

Serial Communication	Parallel Communication
• Data transmitted serially, one bit at a time.	• Data is transmitted parallely, all bits at a time.
• Low speed.	• High Speed.
• It has single transmission line.	• It has multiple transmission lines.
• Serial communication do not have any crosstalk problem.	• Parallel Communication may have crosstalk problem.
• Less expensive.	• More expensive.
• The bandwidth is higher.	• The bandwidth is lower.
• Serial communication does not have eye loops at high frequencies.	• Parallel communication may not work properly at high frequencies.
• It is not affected with noise problems.	• It may suffer with noise problems.
• It covers long distance when compared to parallel communication.	• It is used for short distances.
• Example: Serial communication like a computer and modem.	• Example: Parallel communication like a motherboard and hard disk.

Difference b/w Programmed I/O & Interrupt I/O Transfer

	Programmed I/O	Interrupt I/O
1.	In programmed I/O, processor has to check each I/O device in sequence and in effect 'ask' each one if it needs communication with the processor. This checking is achieved by continuous polling cycle and hence processor cannot execute other instructions in sequence.	External asynchronous input is used to tell the processor that I/O device needs its service and hence processor does not have to check whether I/O device needs it service or not
2.	It is implemented without interrupt hardware support.	It is implemented using interrupt hardware support.
3.	It does not need initialization of stack.	It needs initialization of stack.
4.	Reading/polling processor is busy and therefore have serious and detrimental effect on system throughput.	In interrupt driven I/O, the processor is allowed to execute its instructions in sequence and only stop to service I/O device when it is told to do so by the device itself. This increases system throughput.
5.	System throughput decreases as no. of I/O devices connected in the system increases.	System throughput does not depend on no. of I/O devices connected in the system.
6.	It does not depend on interrupt status.	Interrupt must be enabled to process interrupt driven I/O.

Interrupt Initiated I/O Transfer

Time consuming limitation of programmed I/O is avoided by interrupt facility. In the mean time, the processor can execute another program. When the interface determines, the device is ready for data transfer, it generates an interrupt request to the computer. On detecting the external interrupt, the processor stops executing program and starts the I/O transfer.

After this servicing is completed, the processor would resume exactly where it left off.

DMA (Direct Memory Access)

When large amount of data are to be transferred, the most efficient technique is known as DMA.

DMA is the technique for data transfer b/w internal & external storage or processor and peripheral without intervention of CPU

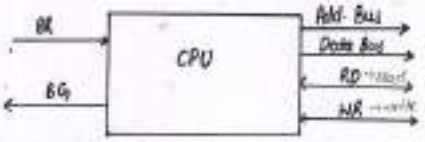


Fig: BR and BG signal in DMA

Bus Request (BR) input is used by DMA controller to request the processor to give up the bus control.

Bus Grant (BG) is an output from CPU to inform external DMA controller that status of buses.

So DMA burst transfer, a block sequence consisting of no. of memory words is transferred in a continuous while the DMA controller is the master of memory.

Eg: Cycle stealing method, which allow the DMA controller to transfer one data word at a time, after that it must return control of buses to the CPU. Here, DMA controller steals one memory cycle from processor.

DMA Controller: DMA controller requires a circuit of an interface to communicate with the processor and input/output device.

It consists of an address register and word count register. The address register and address lines are used for direct communication with memory.

A word count register specifies the no. of words that must be transferred. It is decremented by 1 after each word is transferred.

The address register contains an address of desired location in memory. The address bits go through bus buffer into the address buses.

Control register specifies the mode of transfer.

The registers in DMA are selected by the processor through address bus by enabling DS (DMA select) and RS (Register select) inputs.

The RD (Read) and WR (Write) lines are bidirectional.

When BG=0, the processor can communicate with the DMA register through the data bus to read from or write to the DMA registers.

When BG=1, the processor gives up the control over the buses and DMA can communicate directly with the memory.

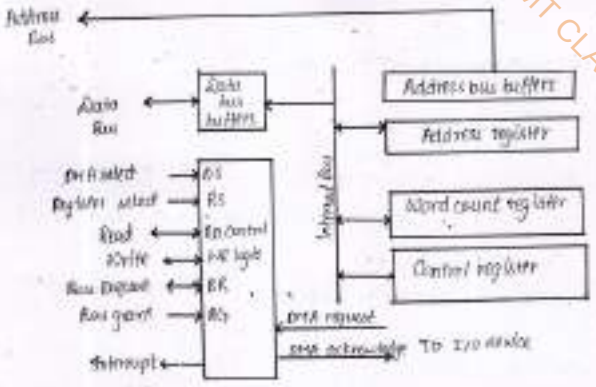
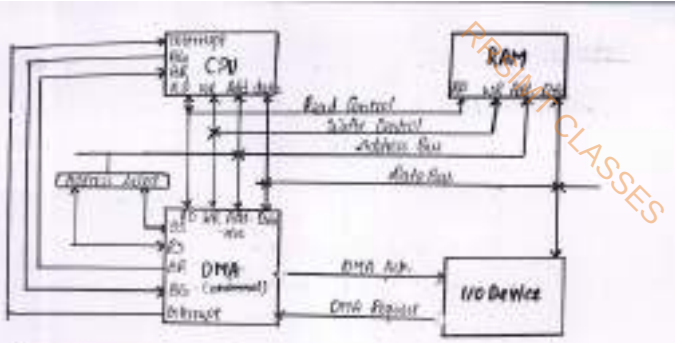


Fig. Block diagram of DMA controller

DMA Transfer :- When the peripheral device send a DMA request to DMA controller, DMA controller activates BR line and the processor responds with the BG line informing that the buses are disabled or activated then, DMA puts the current value of its address register into the address bus, initiates RD or WR signal and sends a DMA acknowledgement to the peripheral device. On receiving a DMA acknowledgement, the I/O device puts a word in the data bus (for write) or receives a word from data bus (for Read). Thus, the peripheral unit can communicate with the memory without processor's intervention.



Input/Output Port :- An input/output interface consists of circuit required to connect an I/O device to the bus. On the one side of interface, the bus signals for address, data and control are required. On the other side, there is a data path with its associated controls to transfer data b/w interface and the I/O device. This is called a port.

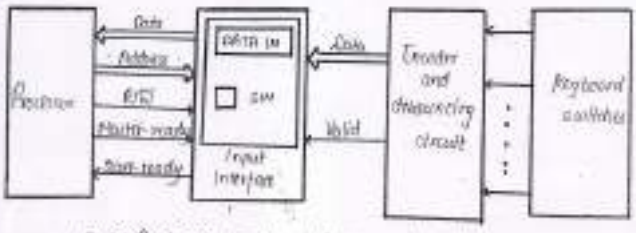


Fig. An interface components for connecting a keyboard to a processor

The port can be classified as :- serial port and parallel port.

Serial Port:- A serial port is used to transmit/receive data serially i.e. one bit at a time. A key feature of an interface circuit for a serial port is that it is capable of communicating in a bit serial manner on the device side and in a bit parallel on the processor side.

Parallel Port:- Parallel port is used to send or receive data having group of bits (eg. 8 bit or 16 bit) at a time.

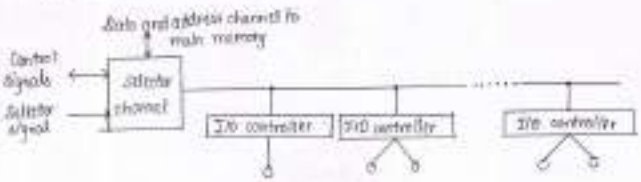
∴ Input/Output Channels

An I/O channel has a special purpose processor. This processor has an ability to execute I/O instruction and it can have complete control over the I/O operation.

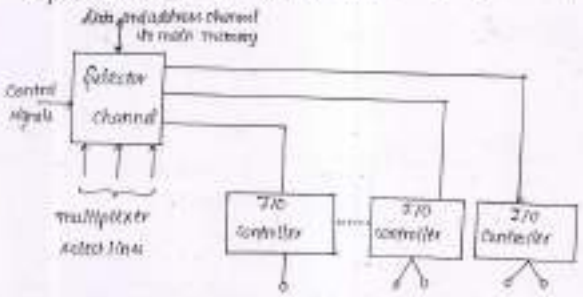
The I/O instructions are stored in the main memory. When I/O transfer is required, the CPU initiates an I/O transfer by instructing I/O channel to execute I/O program stored in the main memory.

There are two types of I/O channel :-

1. **Selector Channel**:- The selector channel controls multiple high speed devices. At a time only one device is accessed for data transfer. Each device is handled by a controller or I/O module.



2. **Multiplexer channel**:- A multiplexer channel can handle/work I/O multiple devices at the same time. A multiplexer channel does the time multiplexing and communicates with I/O controller in an allotted time slot. In multiplexer channel, different devices can have different speeds. However, this is the best technique for low speed devices.



Difference b/w Asynchronous & Synchronous Serial Communication

S.No.	Asynchronous serial comm.ⁿ	Synchronous serial comm.ⁿ
1.	Transmitter and receiver are not synchronized by clock.	Transmitter and receiver are synchronized by clock.
2.	Bits of data are transmitted at constant rate.	Data bits are transmitted with synchronization of clock.
3.	Character may receive at any rate at receiver.	Character is received at constant rate.
4.	Data transfer is character oriented.	Data transfer happens in blocks.
5.	Used in low speed transmission at about speed less than 20kbit/sec.	Used in high speed transmissions.

Start and stop bits are required to establish communication of each character.	start and stop bits are not required to establish communication of each character; however synchronization bits are required to manage the data block.

Exceptions:- An interrupt is an event that causes the execution of one program to be suspended and the execution of another program is started.

The term 'exception' often used to refer any event that causes an interruption. Hence, I/O interrupts are one example of an exception.

Following are the different kinds of exceptions-

1. **Recovery from errors:-** Various techniques are available which ensures the proper working of computer hardware. For eg:- An error checking code, is included by many computers in the main memory which allows detecting errors in stored data. In case of any error control hardware detects it and informs the processor.

   The processor may also interrupt a program if it detects an error or an unusual condition while executing any instruction.

   Eg:- An attempt to divide by 0.

2. **Debugging:-** Another important exception is used as a support in debbing programs. System software usually includes a program called "debugger" which helps the programmer to find errors in a program.
   - The debugger uses exceptions to provide two important facilities called trace and break points.

3. **Privilege exceptions:-** To protect the operating system from being corrupted by the user programs, certain instructions can be executed only while the processor is in the supervisor mode. These instructions are called privilege instructions.

Priority Interrupt

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 at the same time.

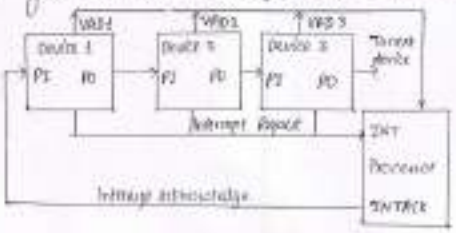
A priority system is a combination of hardware and software techniques.

Priority interrupt can be reached by Daisy Chaining Approach.

Daisy Chaining approach:- This is a serial connection method. The device having highest priority is placed first and followed by lowest priority. When more than one interrupt request are initiated at a time, the processor responds by interrupt acknowledgement (INTA).

This signal is received by device 1 at its PI (Priority In). The acknowledgement signal is passed to the next device through PO (priority out) only if device 1 is not requiring for interrupt.

If device 1 has a pending interrupt, it blocks the acknowledgement signal from the next device by placing 0 in PO. (PO=0)



Standard Communication Interface:-

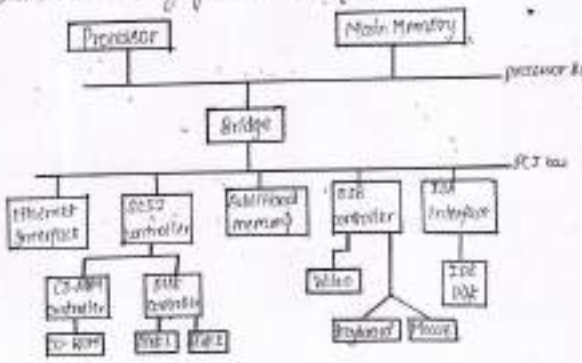
There are many alternatives for bus design. A single bus is not suitable for every communication in a computer system.

In personal computer processor is mounted on the motherboard. Processor require high speed connection. The motherboard provides another bus that support various devices.

The two buses are connected by a circuit called bridge.

There are various number of standards developed for bus expansion. Eg:- PCI (Peripheral Component Interconnect), SCSI (Small computer system interface) and USB (Universal serial Bus).

PCI standard is used for expansion bus on the motherboard. SCSI has its high speed parallel bus intended for devices which need the large amount of data transfer. USB is used for serial transmission to fulfil the needs of device like keyboard, mouse etc.



(Standard Communication Interface)

Difference btw I/O mapped and Memory mapped I/O of 8086

Sr.No.	I/O Mapped I/O	Memory Mapped I/O
1.	I/O device is treated as an I/O device and hence given an I/O address.	I/O device is treated like a memory device and hence given a memory address.
2.	I/O device has an 8 or 16-bit I/O address.	I/O device has a 20-bit memory address.
3.	I/O device is given IOR# and IOW# control signals.	I/O device is given MEMR# and MEMW# control signals.
4.	Decoding is easier due to fewer address lines.	Decoding is more complex due to more address lines.
5.	Decoding is cheaper.	Decoding is more expensive.
6.	Uses fewer gates so less delays.	More gates add more delay hence slower.
7.	Allows max 2^16 = 65536 I/O devices.	Allows many more I/O devices as I/O addresses are now 20 bits.
8.	I/O devices can only be accessed by IN and OUT instruction.	I/O devices can now be accessed using any memory instruction.
9.	ONLY AL/AH/AX registers can be used to transfer data with the I/O device.	Any register can be used to transfer data with the I/O device.