COMPUTER ORGANIZATION AND ARCHITECTURE (BCS302)

UNIT-1

ONE SHOT

Today's Target

- RESULT ORIENTED
- ALL 7 AND 2 MARKS QUESTIONS COVER
- ALL AKTUS' pervious year questions covered

By PRAGYA RAJVANSHI

B.Tech, M.Tech(C.S.E)

- ONE SHOT Revision PDF Notes are available in the Application for FREE.
- Application link in Description below.

What is computer?

Computer: A computer is an electronic device that takes input from the user, processes it under the control of set of instructions and generates output. It has the ability to store, retrieve, and process data.

➢A computer is made up of multiple parts and components that facilitate user functionality. A computer has two primary categories:

Hardware: Physical structure that houses a computer's processor, memory, storage, communication ports and peripheral devices

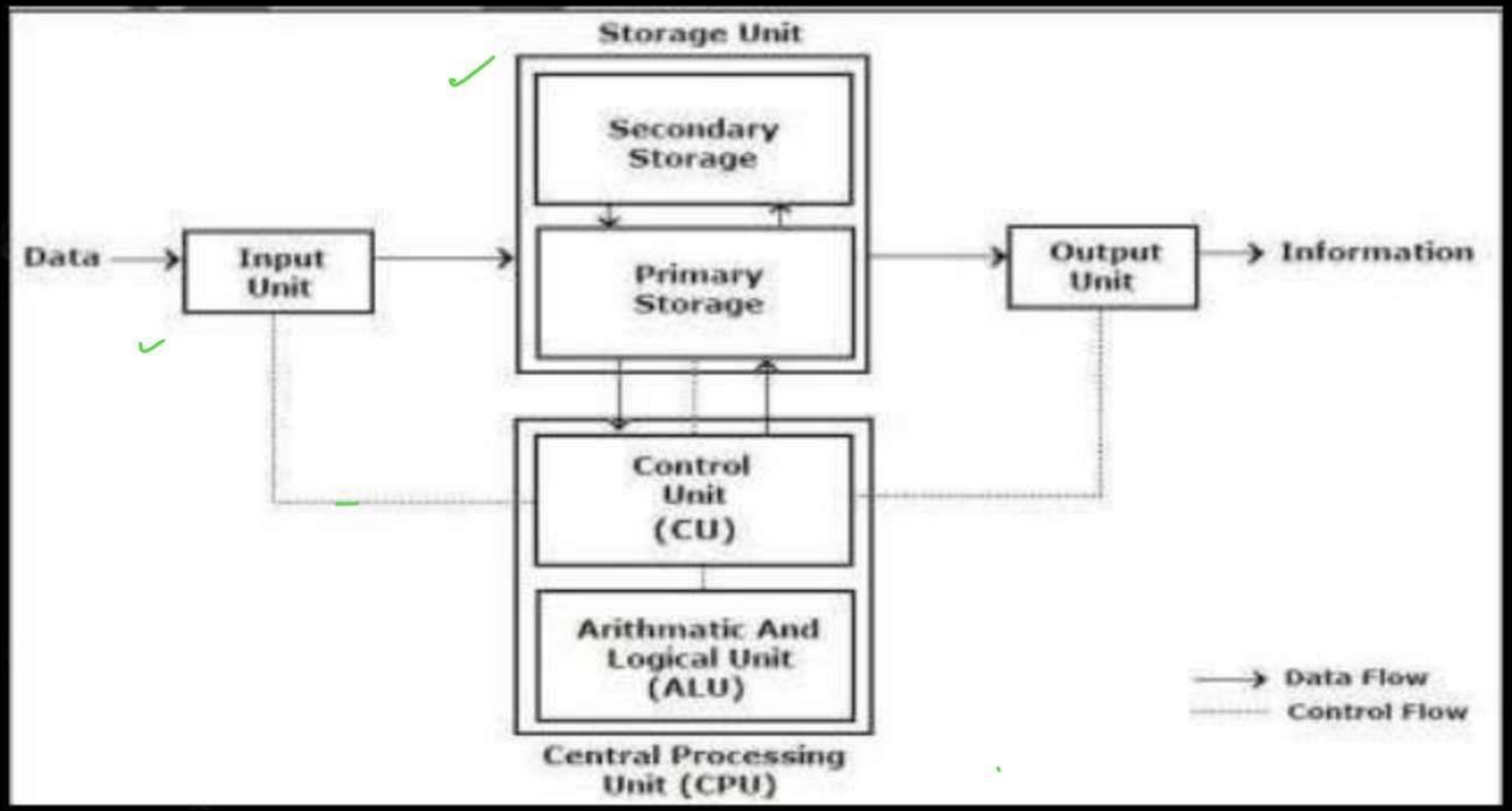
Software: Includes operating system (OS) and software applications.

Difference between-

2 Marks

COMPUTER ARCHITECTURE	COMPUTER ORGANIZATION
The Architecture describes what the computer does.	The Organization describes how it does it.
Computer Architecture deals with the functional behavior of computer systems.	Computer Organization deals with a structural relationship.
It deals with high-level design issues.	It deals with ~~high~~ low-level design issues.
Architecture indicates its hardware	Whereas Organization indicates its performance.
For designing a computer, its architecture is fixed first.	For designing a computer, an organization is decided after its architecture.
It makes the computer's hardware visible.	It offers details on how well the computer performs.

Functional Unit of Digital System



Functional Unit of Digital System

1. Input Unit:

- We need to first enter the data & instruction in the computer system, before any computation begins.

- This task is done by the input devices. (E.g. : keyboard, mouse, scanner, digital camera etc).

Functional Unit of Digital System

2. Storage Unit:

- The data & instruction that are entered have to be stored in the computer. Similarly, the end results & the intermediate results also have to be stored somewhere before being passed to the output unit.
- The storage unit provides solution to all these issues. This storage unit is designed to save the initial data, the intermediate result & the final result.

5

$\boxed{2+3}+5$

10

Functional Unit of Digital System

i) **Primary Storage:**

RAM – Volatile

ROM – Non-volatile

- It is also called main memory of computer.
- It is computer memory that a processor or computer accesses first or directly.
- It allows a processor to access running execution applications and services that are temporarily stored in a specific memory location. It is of two types RAM and ROM.

RAM
- Static RAM
- Dynamic RAM (ms)

ROM
- PROM
- EPROM → U.V. light
- EEPROM → Electrical waves

Functional Unit of Digital System

ii) Secondary Storage:

External memory

- The secondary storage, also called as the auxiliary storage, can retain information even when the system is off.

- It is basically used for holding the program instructions & data on which the computer is not working on currently, but needs to process them later.

Functional Unit of Digital System

3) Central Processing Unit:

- Central Processing Unit (CPU) has two major components: ALU (Arithmetic Logic Unit) and CU (Control Unit).

- The CPU is the brain of the computer. In a computer system, all the major calculations & comparisons are made inside the CPU.

Functional Unit of Digital System

(i)Arithmetic Logic Unit:

- The actual execution of the instructions (arithmetic or logical operations) takes place over here.
- The ALU performs simple addition, subtraction, multiplication, division, and logic operations, such as OR and AND.
- Intermediate results that are generated in ALU are temporarily transferred back to the primary storage, until needed later. Hence, data may move from the primary storage to ALU & back again to storage, many times, before the processing is done.

2+3*5

2+15 — Intermediate Result

17 — final

Functional Unit of Digital System

(ii) Control Unit:

- This unit controls the operations of all parts of the computer but does not carry out any actual data processing.
- It manages and coordinates all the units of the system. It also communicates with Input/output devices for transfer of data or results from the storage units.

Functional Unit of Digital System

4. output Unit:

- Output unit accepts the results produced by the computer in coded form.
- It converts these coded results to human readable form.
- Finally, it displays the converted results to the outside world with the help of output devices (E.g. monitors, printers, projectors etc).

WHAT IS BUS?

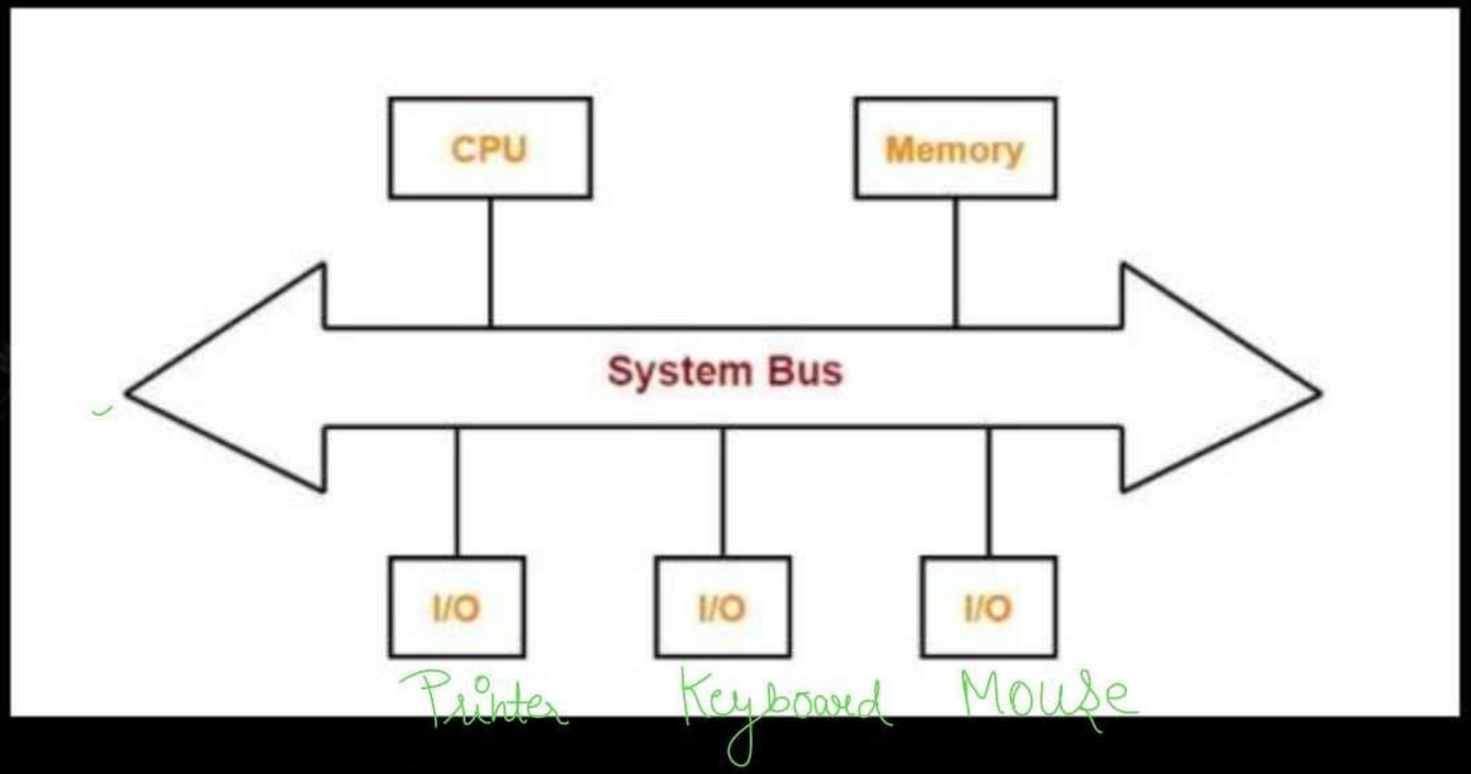
- A bus is a set of electrical wires (lines) that connects the various hardware components of a computer system.

- It works as a communication pathway through which information flows from one hardware component to the other hardware component.

WHAT IS SYSTEM BUS?

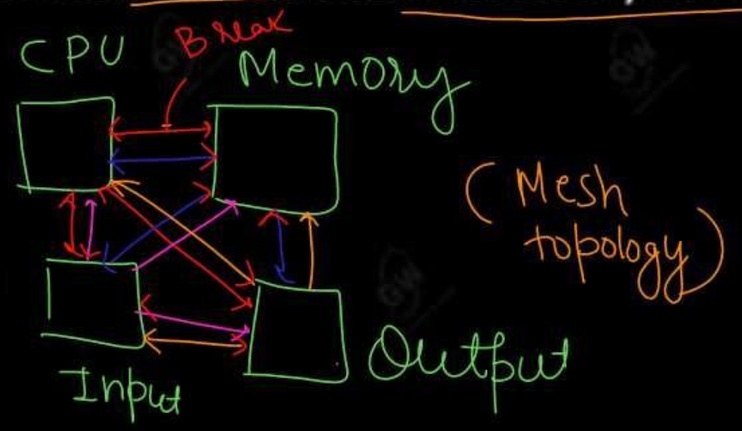
- A bus that connects major components (CPU, memory and I/O devices) of a computer system is called as a **System Bus.**

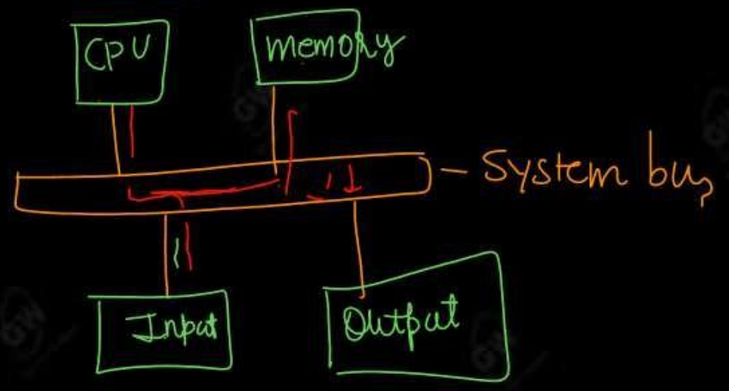
External
Internal



WHY DO WE NEED BUS?

- A computer system is made of different components such as memory, ALU, registers etc.
- Each component should be able to communicate with other for proper execution of instructions and information flow.
- If we try to implement a mesh topology among different components, it would be really expensive.
- So, we use a common component to connect each necessary component i.e. BUS.

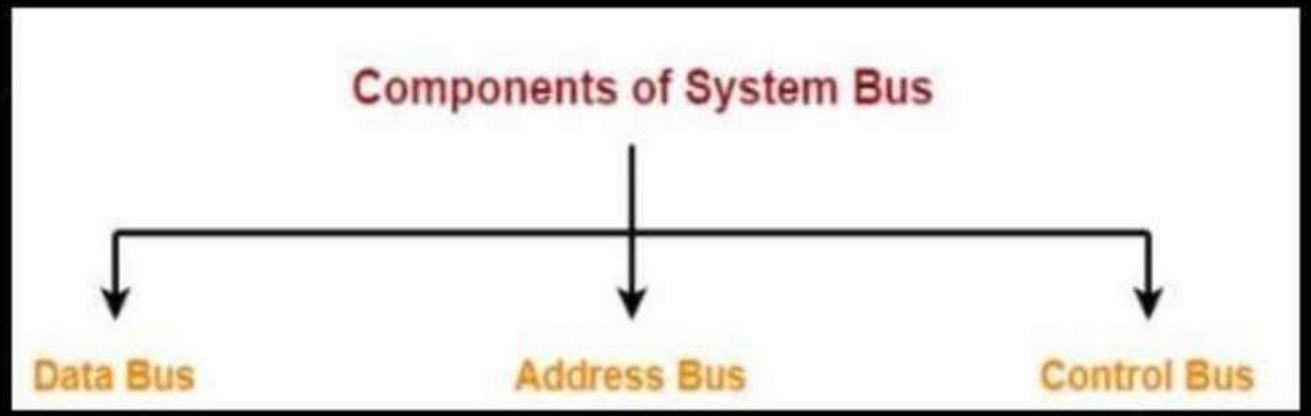




TYPES OF BUS

The system bus consists of three major components-

- Data Bus
- Address Bus (Memory location)
- Control Bus



DATA BUS

As the name suggests, data bus is used for transmitting the data / instruction from CPU to memory/IO and vice-versa.

- Data bus carry the data.
- Data bus is a bidirectional bus.
- Data bus fetch the instructions from memory.
- Data bus used to store the result of an instruction into memory.

CPU → memory
CPU → I/O — Printer
memory — CPU
I/O → CPU — Keyboard / Mouse

DATA BUS

- Data bus carry commands to an I/O device controller or port.
- Data bus carry data from a device controller or port.
- Data bus issue data to a device controller or port.

Control bus

- As the name suggests, control bus is used to transfer the control and timing signals from one component to the other component.
- The CPU uses control bus to communicate with the devices that are connected to the computer system.
- The CPU transmits different types of control signals to the system components.
- It is bi-directional.

Control and timing signal

- control signals are generated in the control unit of CPU.
- Timing signals are used to synchronize the memory and I/O operations with a CPU clock.

Typical control signals hold by control bus-

- **Memory read** – Data from memory address location to be placed on data bus.
- **Memory write** – Data from data bus to be placed on memory address location.
- **I/O Read** – Data from I/O address location to be placed on data bus.
- **I/O Write** – Data from data bus to be placed on I/O address location.

Control and timing signal

Other control signals hold by control bus are interrupt, interrupt acknowledge, bus request, bus grant and several others.

The type of action taking place on the system bus is indicated by these control signals.

EXAMPLE

control bus to perform the memory read or write operation from the main memory. Similarly, when the processor wants to read from an I/O device, it generates the I/O read signal.

Address bus

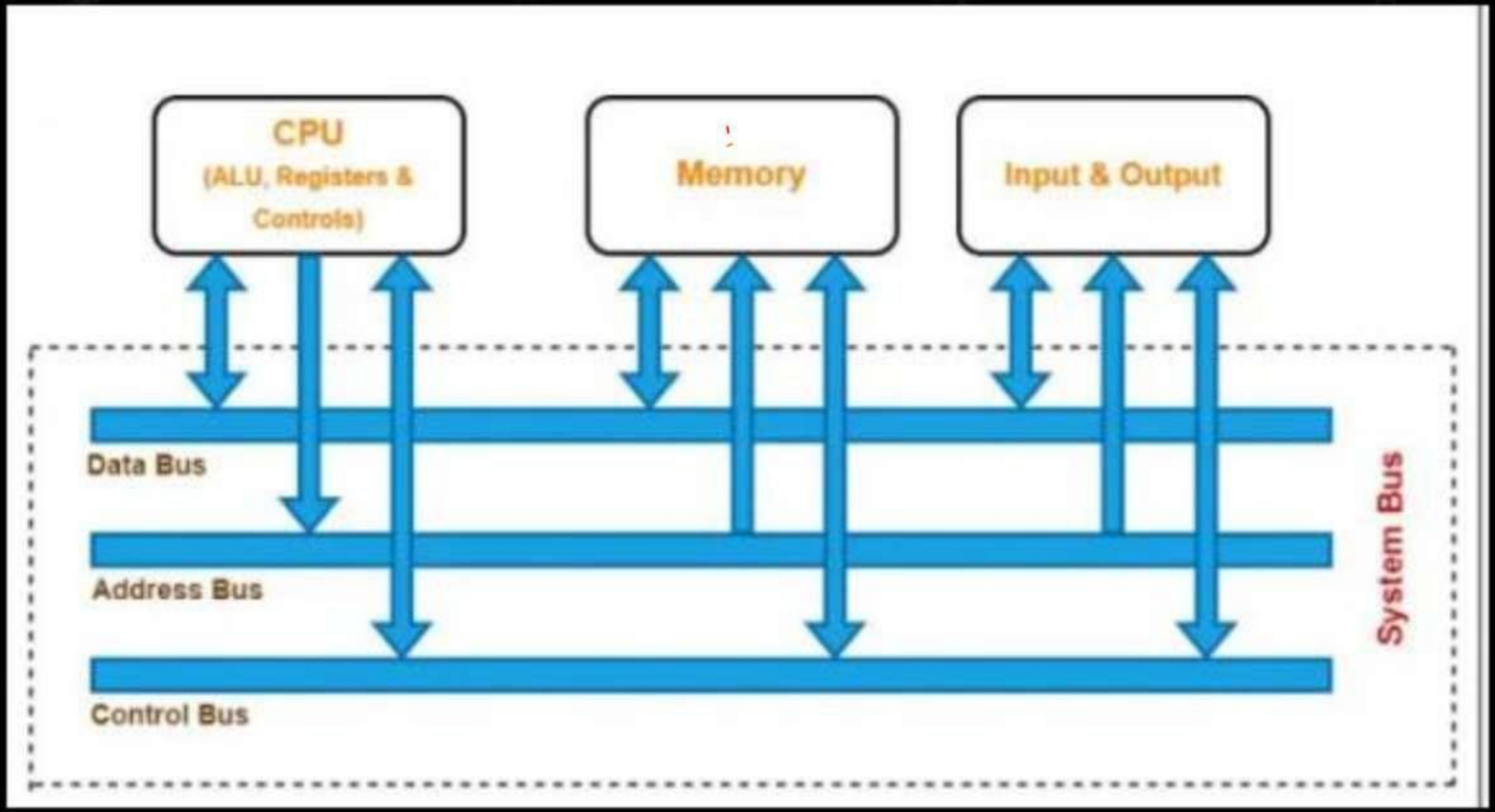
- As the name suggests, address bus is used to carry address from CPU to memory/IO devices.
- It is used to identify the particular location in memory.
- It carries the source or destination address of data i.e. where to store or from where to retrieve the data.
- It is uni-directional.

Address bus

EXAMPLE

➢When CPU wants to read or write data, it sends the memory read or memory write control signal on the control bus to perform the memory read or write operation from the main memory and the address of the memory location is sent on the address bus.

Bus Architecture?



Bus Architecture

The system bus consists of three major components-

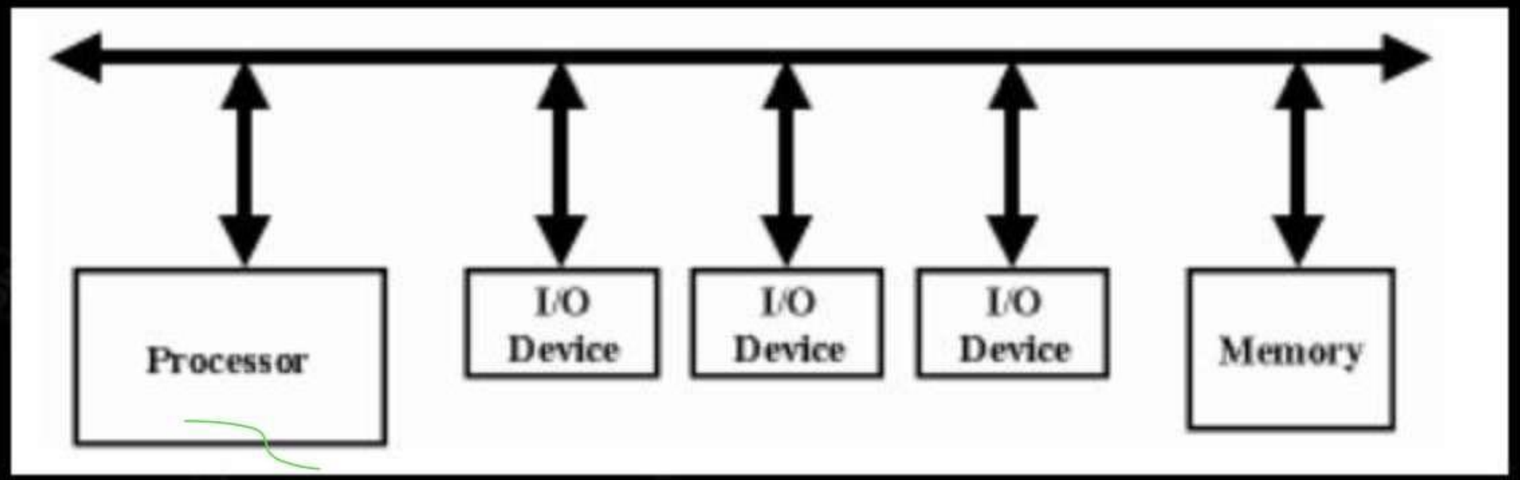
- Data Bus
- Address Bus
- Control Bus

YOU HAVE TO WRITE ALL OF THESE BUSES (Detailed Explaination already given in Previous Slides)

The principle use of system bus is high speed data transfer between CPU AND MEMEORY

Advantages of system bus

- New devices can be added easily
- Low Cost because a single set of wires is shared in multiple ways



Disadvantages of system bus

- The bandwidth of that bus can limit the maximum I/O throughput.

- The maximum bus speed is largely limited by:

  - The length of the bus.
  - The number of devices on the bus.

AKTU QUESTIONS (LECTURE 1)

↑ 2 Marks

Q1	List and briefly define the main structural component of a computer.	AKTU2021-22
Q2	Define computer architecture and computer organization. 2 Marks	AKTU 2020-21 AKTU 2019-20
Q3	What is the basic functional unit of computer? 7 Marks	AKTU 2019-20

1. CPU
2. memory
3. Input
4. Output

AKTU QUESTIONS(LECTURE 2)

Q1	What is bus? Explain different types of bus available. 2 Marks 7 Marks	AKTU2017-18 AKTU 2018-19
Q2	Define bus architecture.	AKTU 2015-16

Possiblity (1) Diffrence b/w Address bus and data bus
data & control
Control and data
2 Marks

BUS ARBITRATION

- In a computer system, multiple devices, such as the CPU, memory, and I/O controllers, are connected to a common communication pathway, known as a bus.

- In order to transfer data between these devices, they need to have access to the bus.

- Bus arbitration is the process of resolving conflicts that arise when multiple devices attempt to access the bus at the same time.

- When multiple devices try to use the bus simultaneously, it can lead to data corruption and system instability.

- To prevent this, a bus arbitration mechanism is used to ensure that only one device has access to the bus at any given time

BUS ARBITRATION

➢BUS MASTER (2Marks)

The controller that has access to a bus at an instance is known as a **Bus master**

➢ BUS ARBITER (2Marks)

The **Bus Arbiter** decides who would become the current bus master.

➢BUS ARBITRATION

Bus Arbitration refers to the process by which the current bus master accesses and then leaves the control of the bus and passes it to another bus requesting processor unit.

BUS ARBITRATION

Two approaches are followed for the bus Arbitration:

Centralized Bus Arbitration - In which the necessary arbitration is carried out by alone bus arbitrator.

Distributive Bus Arbitration - In which every device takes part in choosing the new bus master. A 4bit identification number is allocated to each device on the bus. The created ID will decide the device's priority.

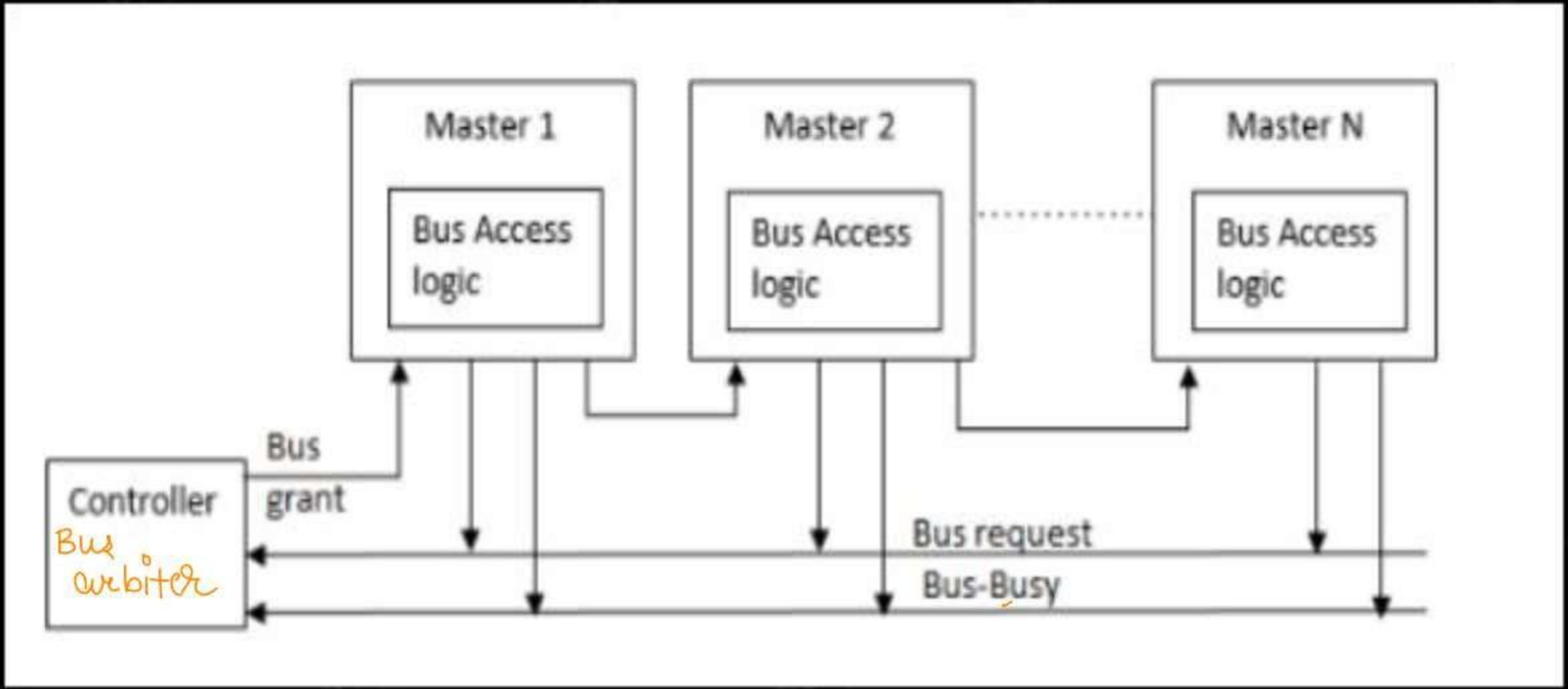
BUS ARBITRATION

Centralized Bus Arbitration is of three types-.

- Daisy Chaining Method
- Rotating or Polling Priority Method
- Independent Request or Fixed Priority Method

2 Marks

Daisy Chaining Method



Daisy Chaining Method

- All bus master use the same line for bus request.
- If the bus busy line is inactive, the bus controller gives the bus grant signal.
- The bus grant signal serially propagate through each master until it encounter the first one that is requesting access to the bus.
- This master block the propagation of the bus grant signal, activates the busy line and gain the control of the bus.
- Therefore any other requesting module will not receive the grant signal and can not get the bus access

Daisy Chaining Method

Advantages:

a) Simple design

b) Less no. of control lines.

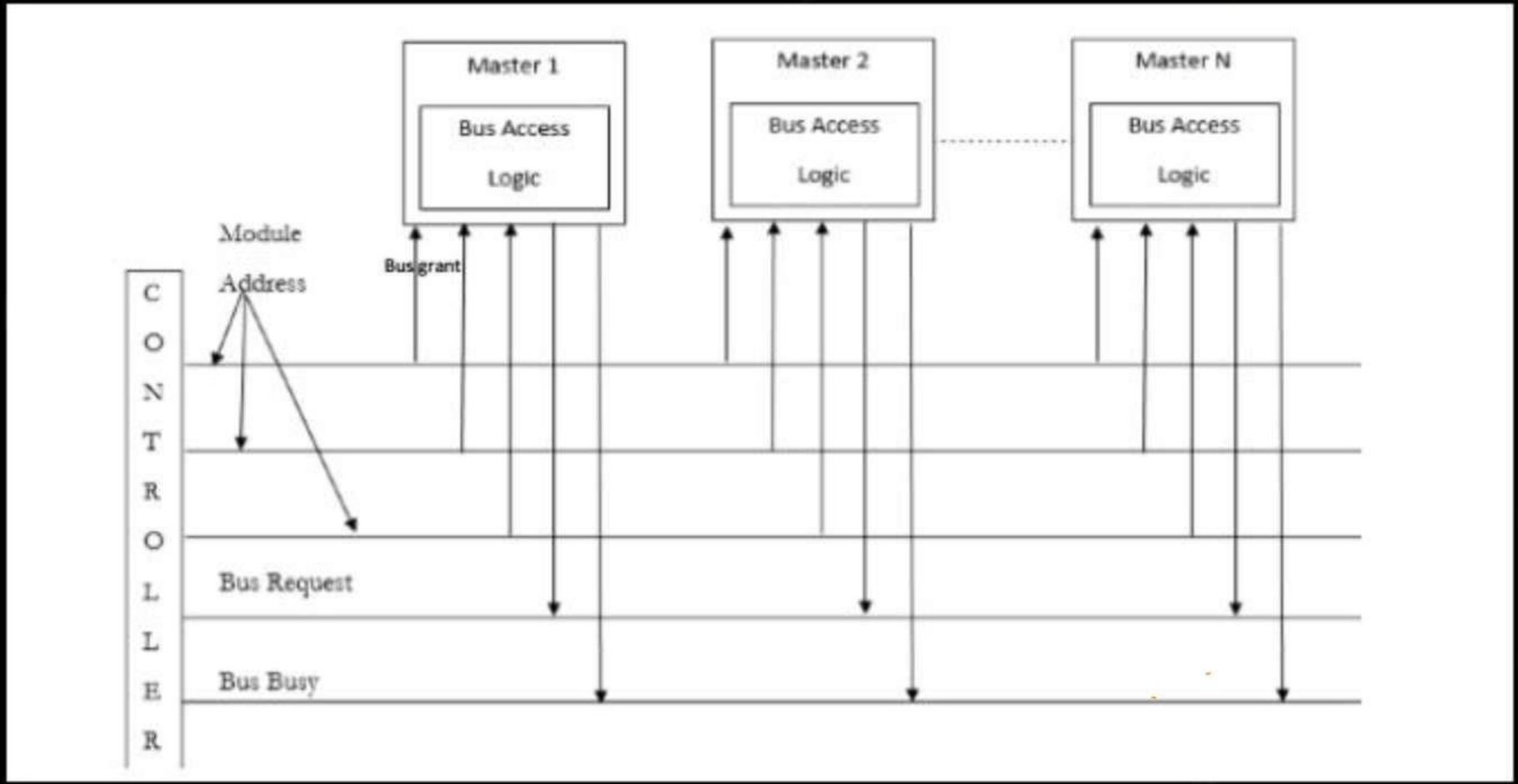
Disadvantages:

a) Priority depends on the physical location of master.

b) Propagation delay due to serially granting of bus.

c) Failure of one of the devices may fail entire system.

Rotating or Polling Priority Method



0	0	— A
0	1	— B
1	0	— C
1	1	— E

2 — bit

4 2^2

8 2^3 — bit

2^2	2^1	2^0
4	2	1

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

$2^4 = 16$

Rotating or Polling Priority Method

- Here all bus masters use the same line for bus request.
- Here controller generate binary address for the master. (To connect 8 bus master we need 3 address lines 2^3= 8).
- In response to a bus request, the controller "polls" the bus masters by sending a sequence of bus master address on address lines.
- When requesting master recognizes its address, it activates the bus busylines and takes control of the bus.

Rotating or Polling Priority Method

Advantages –

This method does not favor any particular device and processor.

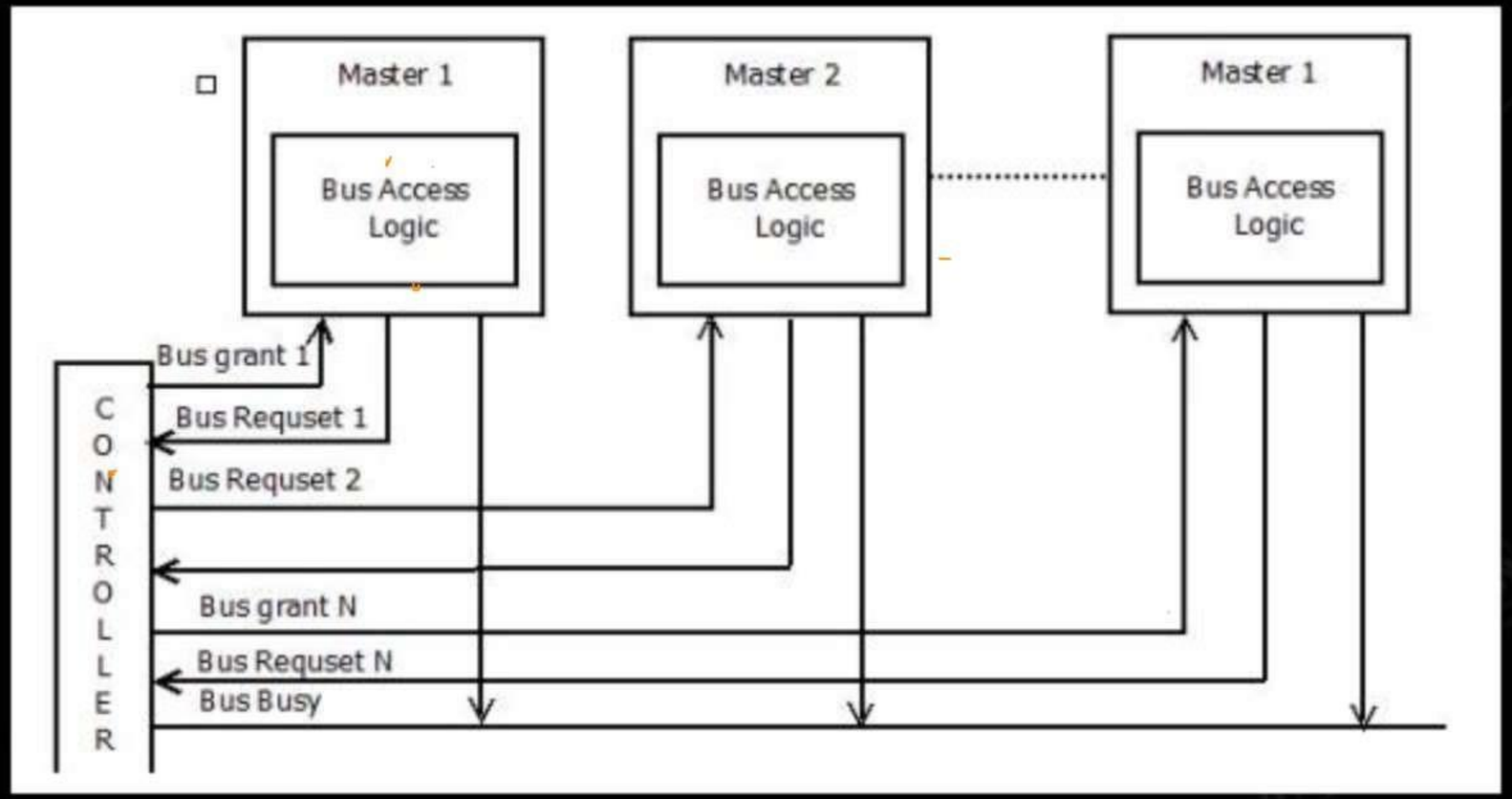
The method is also quite simple.

If one device fails then the entire system will not stop working.

Disadvantages –

Adding bus masters is difficult as increases the number of address lines of the circuit.

Independent Request or Fixed Priority Method



Independent Request or Fixed Priority Method

- All bus masters have their individual bus request and bus grant lines.
- The controller thus knows which master has requested, so bus is granted t that master.
- Priorities of the masters are predefined so on simultaneous bus requests, the bus is granted based on the priority, provided the bus busy line is not active.
- The controller consists of encoder and decoder logic for priorities.

Independent Request or Fixed Priority Method

Advantages:

a) Bus arbitration is fast.

b) Speed independent of no. of devices connected.

Disadvantages:

a) No. of control lines required is more. Hence connecting a large number of bus masters is difficult.

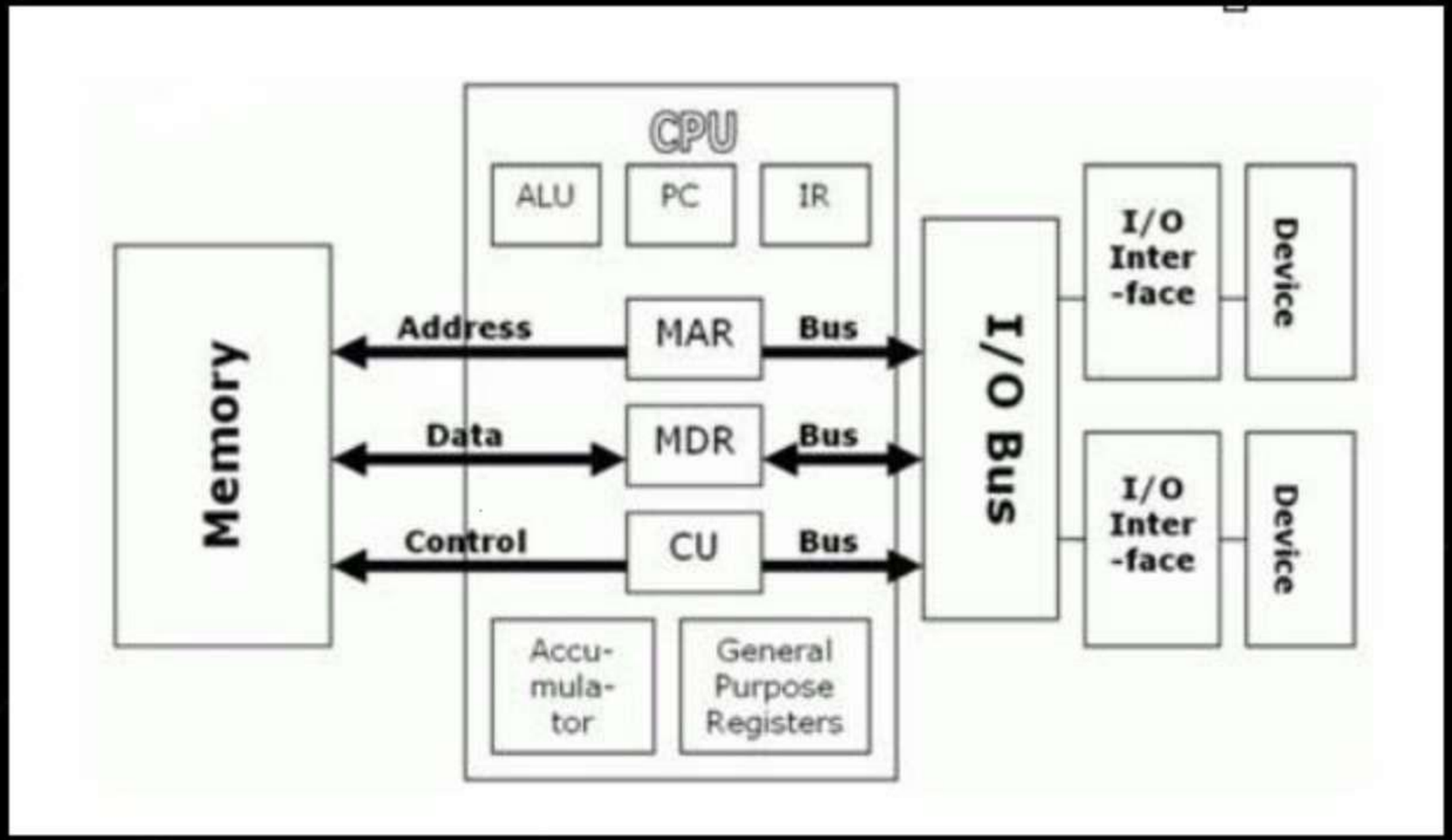
SUMMARY

PARAMETERS	DAISY CHAINING	POLLING	INDEPENDENT REQUEST
DESIGN	Simplest	Little complex	complex
REALIABILITY	Not reliable if any system fails	Fully Reliable	Fully Reliable
PERFORMANCE	weak	Better	Excellent
UPGRADATION	Easy	Reconfiguration requires	complexity increase
COST	Minimum	less	High

AKTU QUESTIONS(LECTURE 3)

Q1	What is bus arbitration? Explain its types in detail. 2/7 Marks	AKTU2019-20 AKTU 2021-22
Q2	Difference between daisy and polling arbitration 2 Marks	AKTU 2014-15

Block diagram of CPU



Block diagram of CPU

The computer system consists of these parts-

- Central Processing Unit (CPU)
- Arithmetic logic unit (ALU)
- control unit (CU).
- Input Devices
- Output Devices.

→ Storage (Memory)
- Primary Storage
  - RAM
    - Read/write memory
    - volatile
    - CPU directly access
    - two types
      - SRAM
      - DRAM
  - ROM
    - Non-volatile
    - Not CPU directly Access
- Secondary Storage

Block diagram of CPU

➢ Program counter

Program counter (PC) is a CPU register in the computer processor which has the address of the next instruction to be executed from memory.

➢ Memory Address Register (MAR):

This register holds the address of memory where CPU wants to read or write data. When the CPU wants to store some data in the memory or reads the data from the memory, it places the address of the required memory location in the MAR.

➢ Accumulator:

The accumulator is a register in which intermediate arithmetic and logical results are stored

Block diagram of CPU

➢ Instruction Register:

Once an instruction is fetched from main memory, it is stored in the Instruction Register. The control unit takes instruction from this register, decodes and executes it by sending signals to the appropriate component of the computer to carry out the task.

➢ Control Unit:

It decodes the instructions, and controls all the other internal components of the CPU to make it work.

➢ System bus

A bus that connects major components (CPU, memory and I/O devices) of a computer system is called as a **System Bus**

Three main component of system bus – data bus, control bus , address bus

Block diagram of CPU

- **Data bus**

As the name suggests, data bus is used for transmitting the data / instruction from CPU to memory/IO and vice-versa.

- **Address bus**

As the name suggests, address bus is used to carry address from CPU to memory/IO devices.

- **I/O bus**

The I/O bus is the route used for peripheral devices to interact with the computer processor

Block diagram of CPU

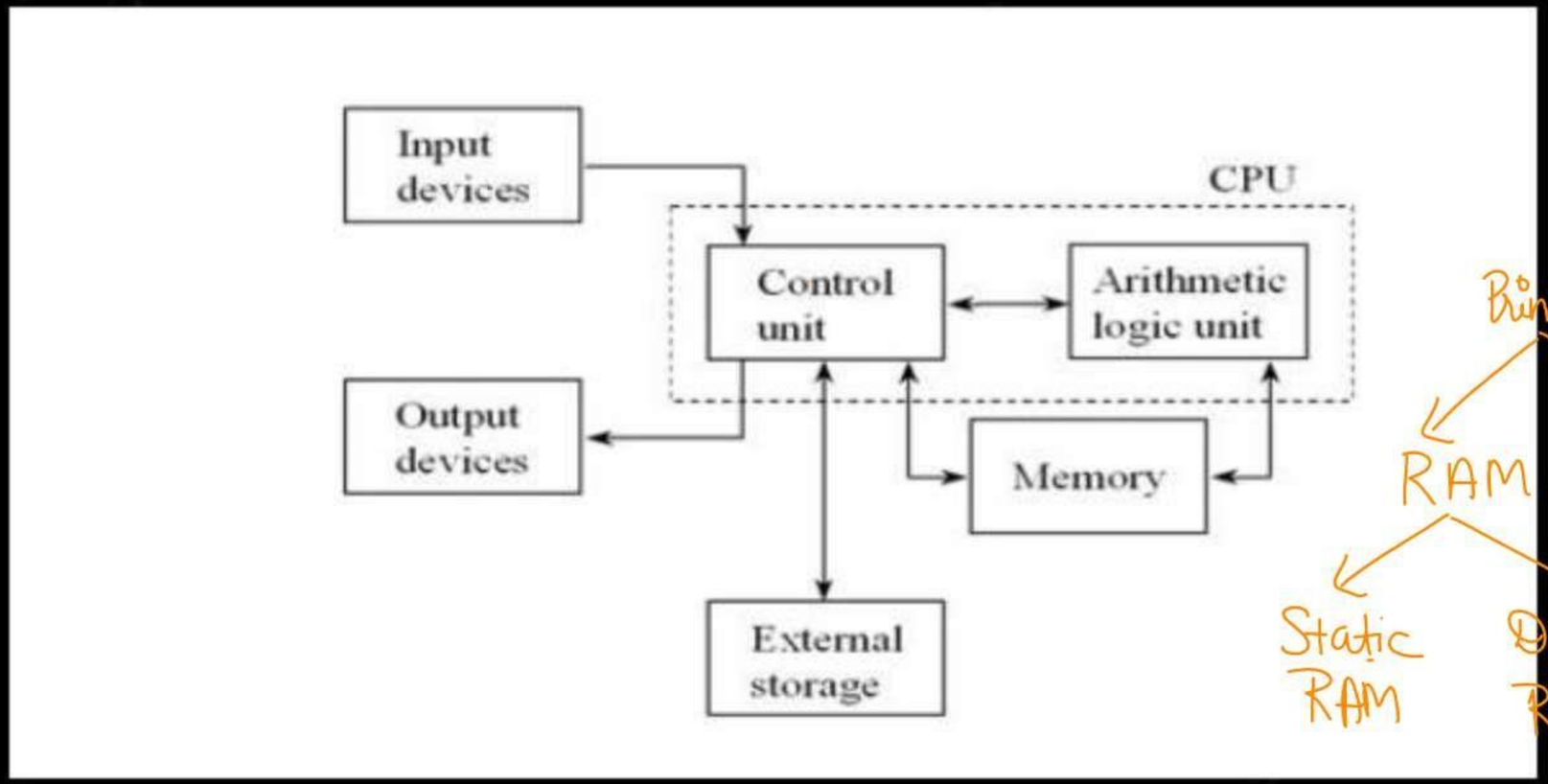
- **memory data register (MDR)**

A register used for holding information (either program words or data words) that is in the process of being transferred from the memory.

- **General purpose register**

General purpose registers are extra registers that are present in the CPU and are utilized anytime data or a memory location is required. These registers are used for storing operands and pointers to the central processor, or vice versa

Arrangement of Computer Processing Unit with Memory, I/O



Memory
- Primary
  - RAM
    - Static RAM
    - Dynamic RAM
  - ROM
    - PROM
    - EPROM — UV light
    - EEPROM — Electrical waves
- Secondary

Arrangement of Computer Processing Unit with Memory, I/O

You have to write about

- CPU
- MEMORY AND ITS TYPES
- CONTROL UNIT
- INPUT DEVICES
- OUTPUT DEVICES

ALREADY GIVEN DETAIL NOTES IN LECTURE 1 FUNCTION UNIT OF COMPUTER SYSTEM

Memory transfer

- Memory transfer means to fetch(read) or store(write) data
- The transfer of information from a memory unit to the user end is called a **Read** operation.
- The transfer of new information to be stored in the memory is called a **Write** operation.

2 Marks

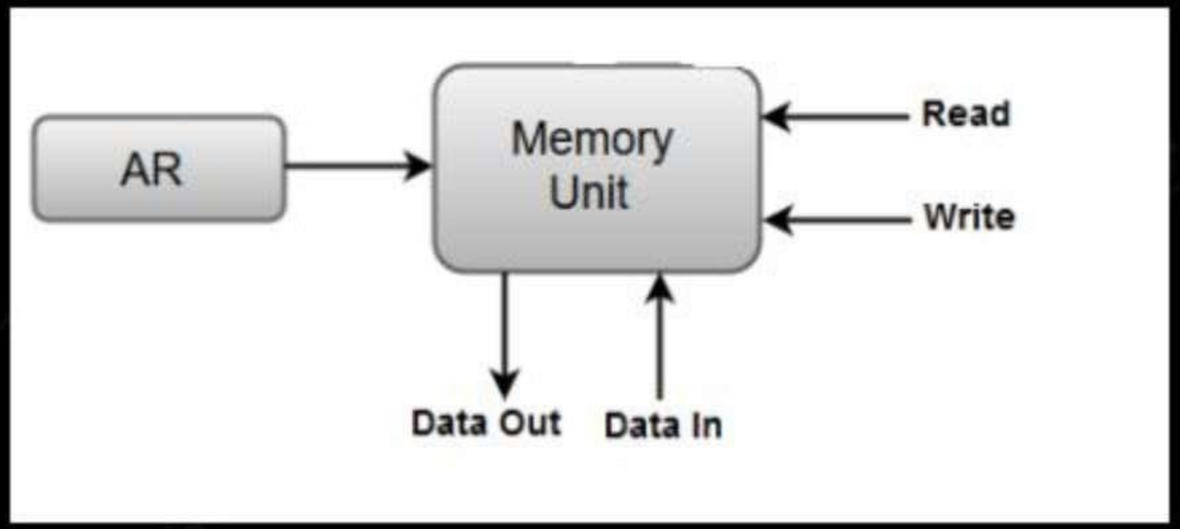
- A memory word is designated by the letter **M**.
- Two register associated for memory transfer-
  - Address register
  - Data register

Memory transfer

- Address register is used to store the memory address and data register is used to store the data.
- The required information is selected from the memory location by address and that address stored in address register.
- The **address register** is designated by **AR** and the **data register** by **DR**.
- Thus, a read operation can be stated as:
- Read: DR ← M [AR]
- The **Read** statement causes a transfer of information into the data register (DR) from the memory word (M) selected by the address register (AR).
-

Memory transfer

- And the corresponding write operation can be stated as:
- Write: M [AR] ← R1
- The Write statement causes a transfer of information from register R1 into the memory word (M) selected by address register (AR).



Bus transfer

- Bus Transfer is the most efficient way to transfer data. The data transferred in the bus transfer is collected using bus lines.
- In a digital system of registers, a path must be provided to move information.
- Suppose separate lines are used between each register and all other registers in the system. In that case, the number of wires connecting all of the registers will be excessive because we are connecting each register with another register.
- A bus structure will not require an excessive connection. Thus it is very useful in transferring information.

Bus transfer

- A bus is made up of a collection of common lines, one for each bit of a register, that are used to transfer binary data one by one.
- There are two methods in bus transfer:
- Bus transfer using Multiplexer
- Bus transfer using Three states bus buffer

Single shared bus benefits

Single shared bus is so widely used as an interconnection medium in both sequential and parallel computer because of the following reason

- The single shared bus is the simplest and least expensive way connecting several components or processor to a set of memory modules.

- It allow compatibility and provides ease of operation

Single shared bus drawback

- One of the disadvantages of single bus system, if there is any fault occur, all system affected and the other feeders pass over.
- This system supply low flexibility and immunity.
- Single bus structure has disadvantages of limited speed since usually only two units can participate in a data transfer at any one time. This means that an arbitration system is required and that units will be forced to wait.

Benefits of multiple bus architecture

Multiple Bus Organization Improves Efficiency

- In a single-bus architecture, all components including the central processing unit, memory and peripherals share a common bus.

- When many devices need the bus at the same time, this creates a state of conflict called bus contention; some wait for the bus while another has control of it.

- The waiting wastes time, slowing the computer down, Multiple buses permit several devices to work simultaneously, reducing time spent waiting and improving the computer's speed.

- Performance improvements are the main reason for having multiple buses in a computer de System

Benefits of multiple bus architecture

Additional Buses Allow Expansion

Having multiple buses available gives you more choices for connecting devices to your computer, as hardware makers may offer the same component for more than one bus type.

Multi-Core Requires Multiple Buses

- A single central processing unit places heavy demands on the bus that carries memory data and peripheral traffic for hard drives, networks and printers.

- since the mid-2000s, however, most computers have adopted a multi-core model that require additional buses. To keep each core busy and productive, the new bus designs ferry increased amounts of information in and out of the microprocessor, keeping wait times to a minimum.

.

AKTU QUESTIONS(LECTURE 4)

Q1	Discuss the advantages and disadvantages of polling and daisy chaining bus arbitration schemes	AKTU2015-16
Q2	List the three control signal	AKTU 2018-19
Q3	Define bus transfer and memory transfer	AKTU 2022-23
Q4	Draw a block diagram of CPU showing all the basic building block such as program counter ,accumulator, address and data register, instruction register, control unit etc. and describe how such an arrangement can work as computer, if connected properly to memory, input/output etc	AKTU 2016-17
Q5	What is memory transfer? What are the different register needed for memory transfer	AKTU 2013-14
Q6	How memory read and write operation performed in computer system	AKTU 2022-23

Multiplexer

- Multiplexer is a combinational circuit that has 2^n input lines and a single output line.
- Simply, the multiplexer is a multi-input and single-output combinational circuit.
- The binary information is received from the input lines and directed to the output line.
- On the basis of the values of the selection lines, one of these data inputs will be connected to the output
- A multiplexer is also treated as **Mux.**

WHAT DO YOU MEAN BY COMBINATIONAL CIRCUIT?

A circuit in which the output depends on the present combination of inputs.

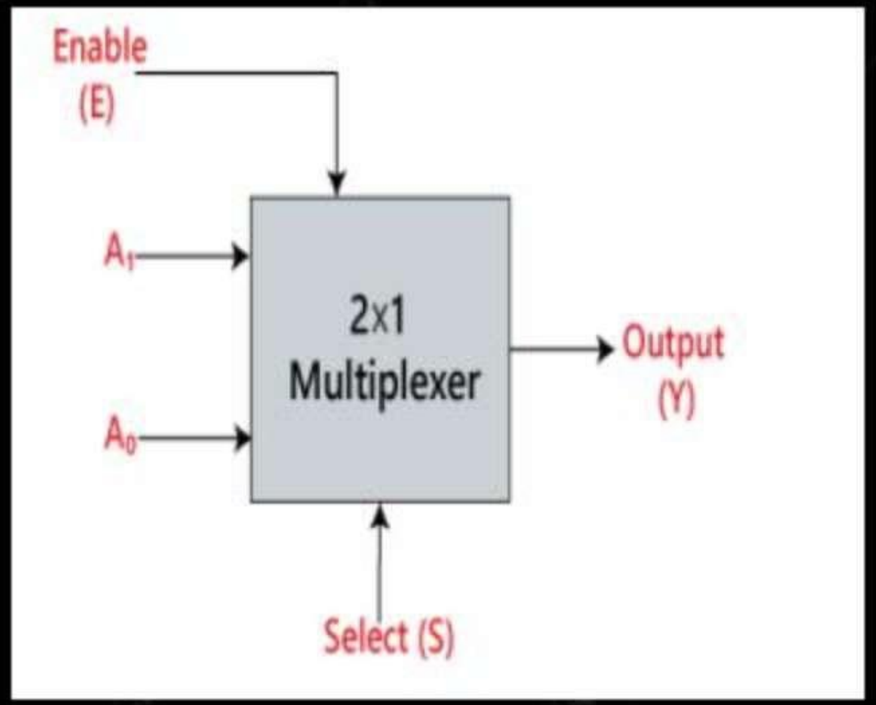
$0 \rightarrow 0 \quad A$

$1 \rightarrow 1 \quad B$

0	0	A
0	1	B
1	0	C
1	1	D

$n \rightarrow 2^n \rightarrow 1$

Multiplexer(2X1 MUX)



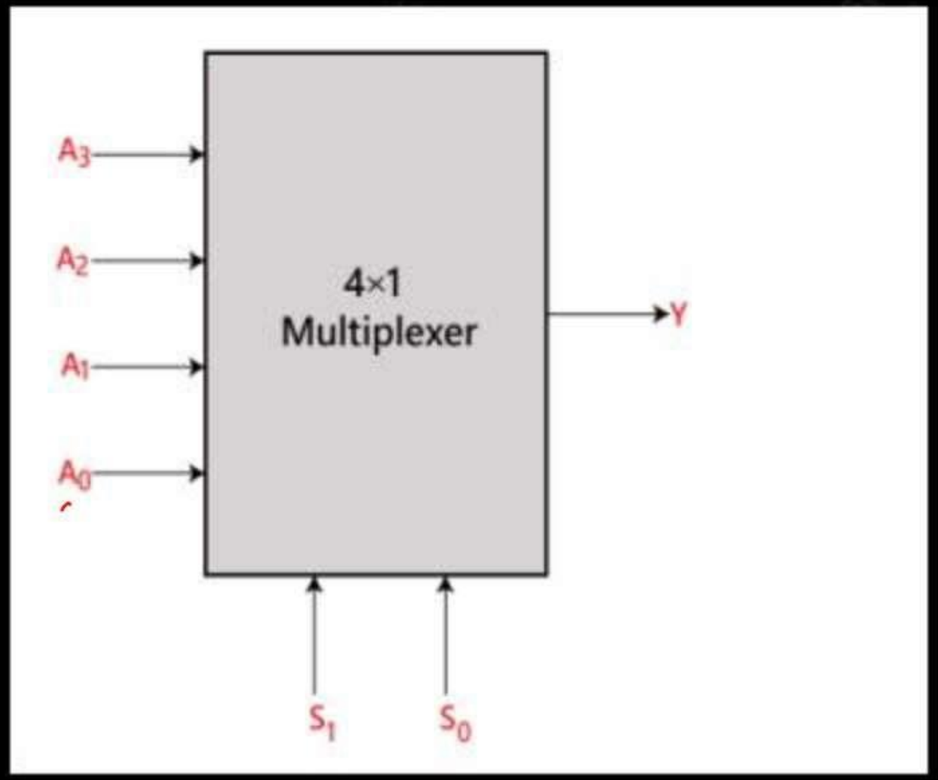
E	S0	Y
X	X	X
1	0	A0
1	1	A1

IIP OIP

Truth Table:

INPUTS	Output
S_0	Y
0	A_0
1	A_1

Multiplexer(4X1 MUX)

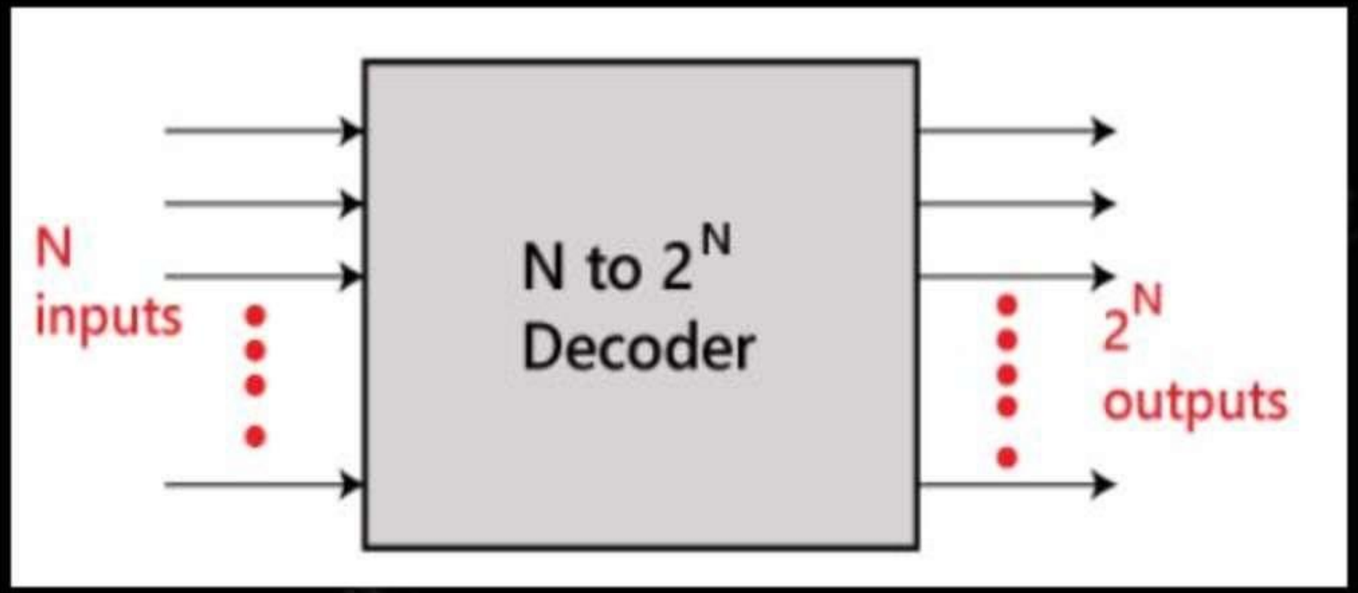


Truth Table:

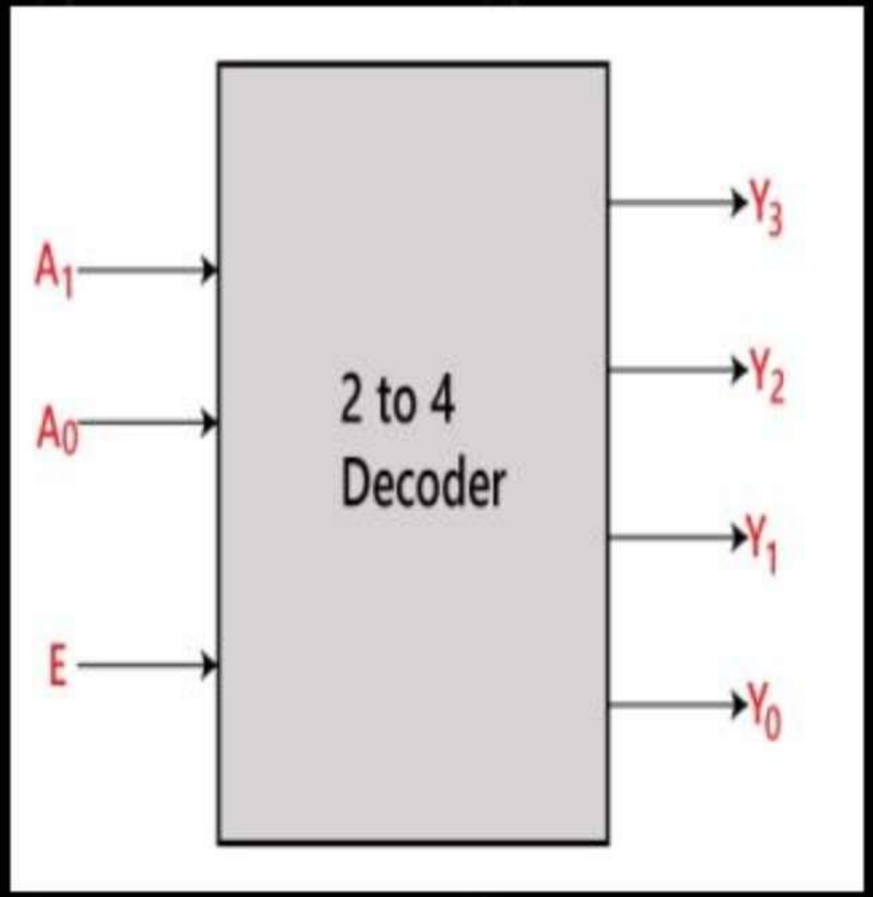
INPUTS		Output
S_1	S_0	Y
0	0	A_0
0	1	A_1
1	0	A_2
1	1	A_3

DECODER

- The combinational circuit that change the binary information into 2^N output lines is known as **Decoders.**
- The binary information is passed in the form of N input lines. The output lines define the 2^N-bit code for the binary information..



DECODER

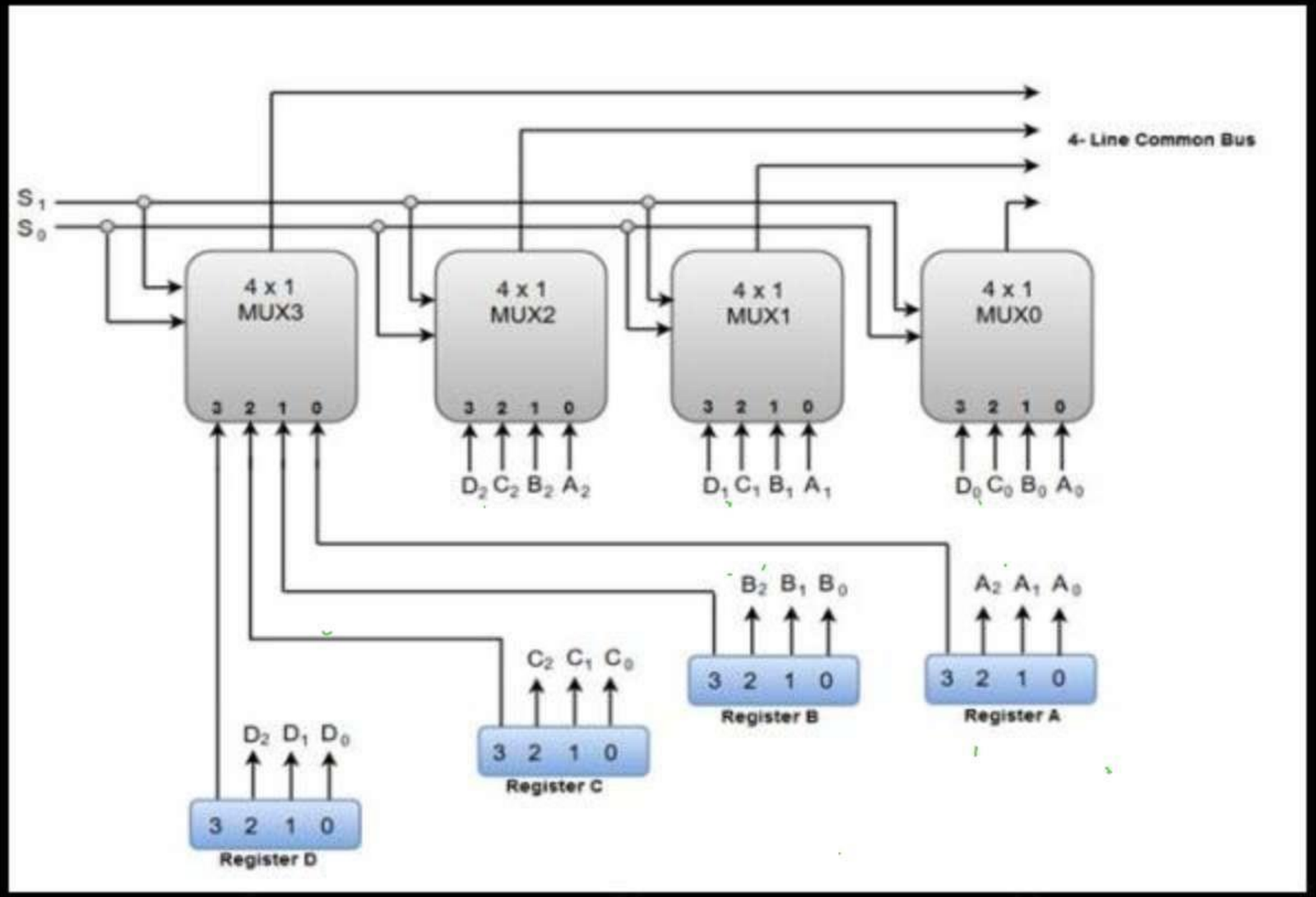


Enable	INPUTS		OUTPUTS			
E	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	X	X	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

Bus transfer through multiplexer

- A typical digital computer has many registers, and paths must be provided to transfer information from one register to another.
- The number of wires will be excessive if separate lines are used between each register and all other registers in the system.
- A more efficient scheme for transferring information between registers in a multiple-register configuration is a common bus system. A bus structure consists of a set of common lines, one for each bit of a register, through which binary information is transferred one at a time.
- Control signals determine which register is selected by the bus during each particular register transfer.
- One way of constructing a common bus system is with multiplexers. The multiplexers select the source register whose binary information is then placed on the bus.

Bus transfer through multiplexer



Bus transfer through multiplexer

➢A The two selection lines S1 and S2 are connected to the selection inputs of all four multiplexers. The selection lines choose the four bits of one register and transfer them into the four-line common bus.

➢When both of the select lines are at low logic, i.e. S1S0 = 00, the 0 data inputs of all four multiplexers are selected and applied to the outputs that forms the bus. This, in turn, causes the bus lines to receive the content of register A since the outputs of this register are connected to the 0 data inputs of the multiplexers.

Similarly, when S1S0 = 01, register B is selected, and the bus lines will receive the content provided by register B.

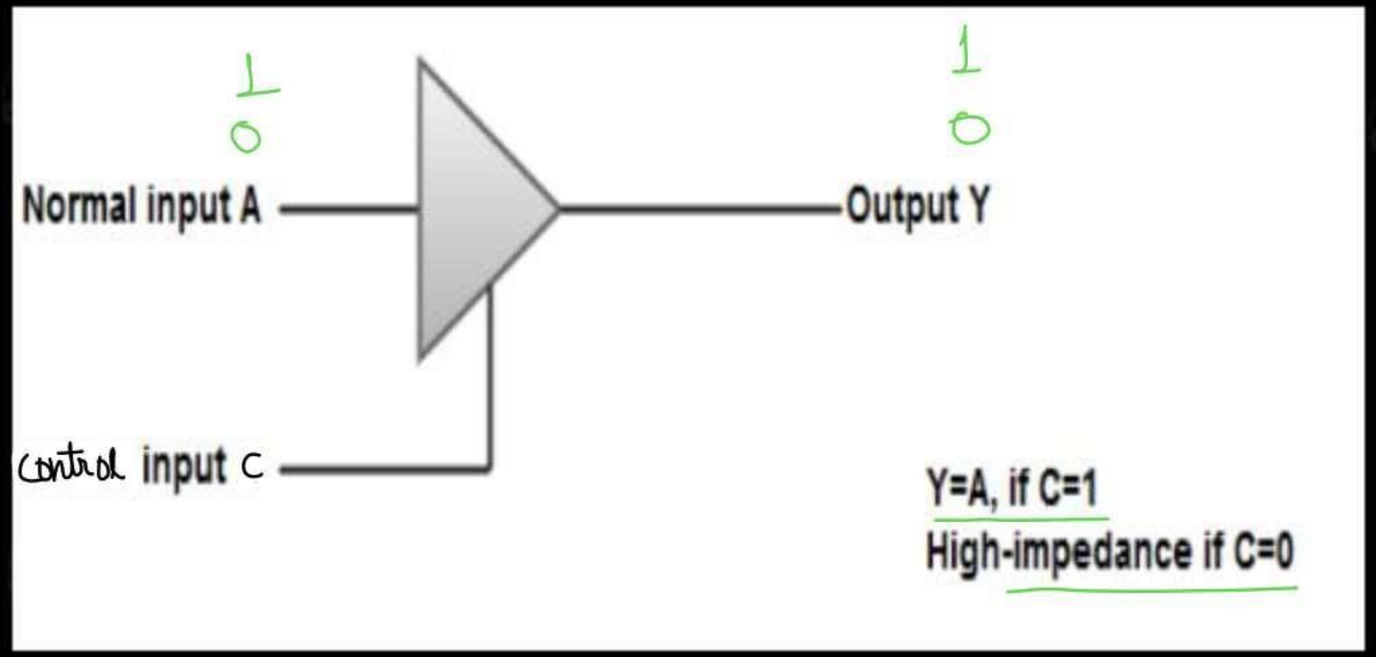
S1	S0	Register Selected
0	0	A
0	1	B
1	0	C
1	1	D

Bus transfer through three state gates

three-state bus buffer

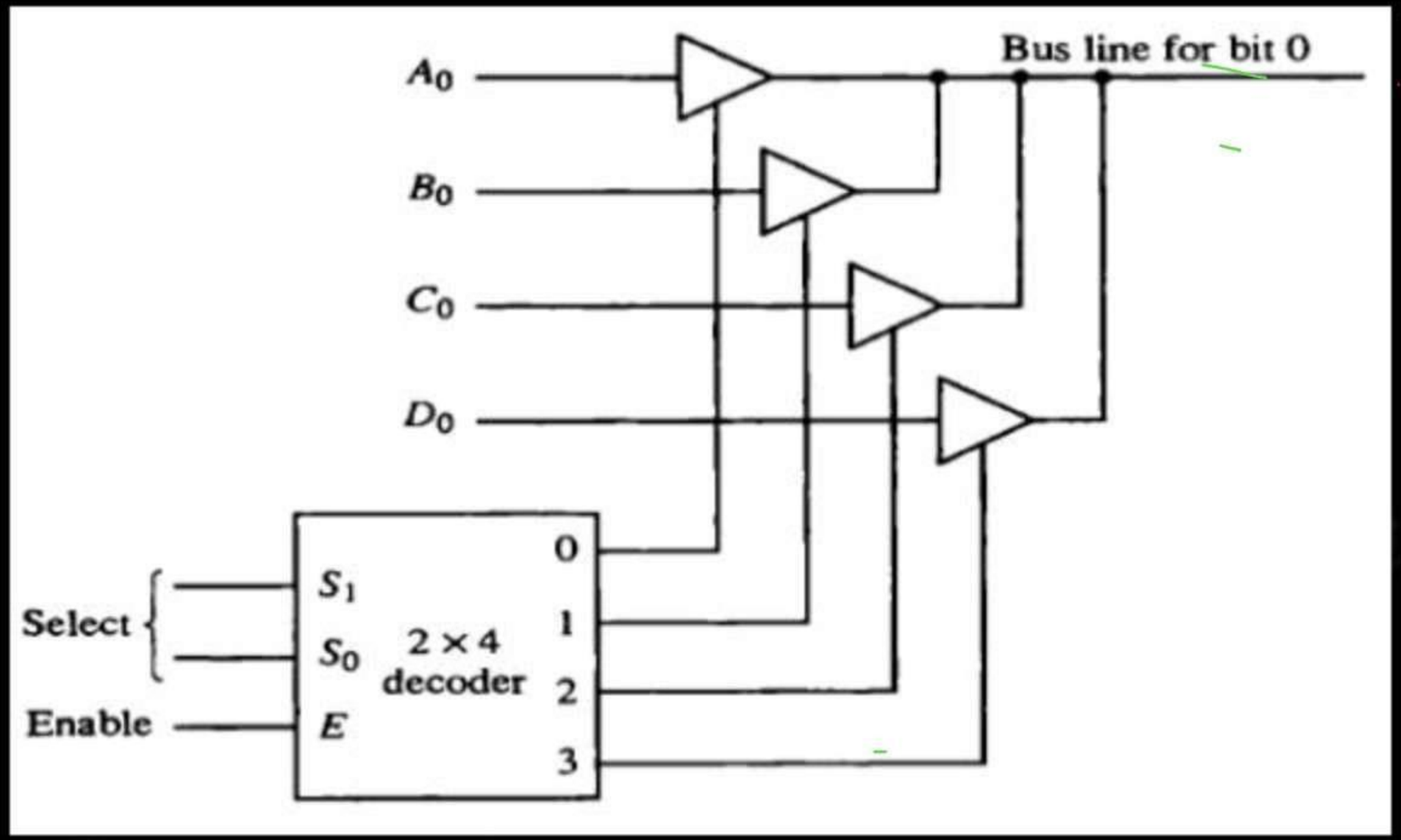
- A bus system can be constructed with three-state gates instead of multiplexers
.
- A three-state gate is a digital circuit that exhibits three states.
- Two of the states are signals equivalent to logic 1 and 0 as in a conventional gate.
- The third state is a high-impedance state.
- The high-impedance state behaves like an open circuit, which means that the output is disconnected and does not have a logic significance.

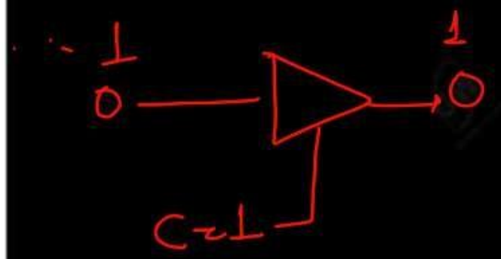
Bus transfer through three state bus buffer

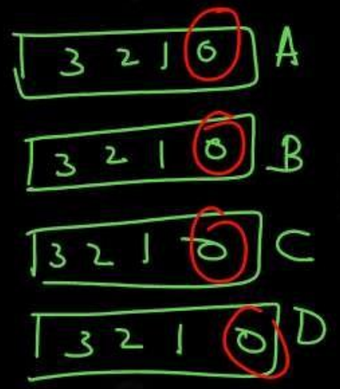


The graphical symbol of a three-state buffer gate can be represented as:

Bus transfer through three bus buffer







Bus transfer through three state bus buffer

- The construction of a bus system with three-state buffers is demonstrated in Fig. The outputs of four buffers are connected together to form a single bus line.

- The control inputs to the buffers determine which of the four normal inputs will communicate with the bus line . No more than one buffer may be in the active state at any given time.

- The connected buffers must be controlled so that only one three-state buffer has access to the bus line while all other buffers are maintained in a high impedance state.

Bus transfer through three state bus buffer

- One way to ensure that no more than one control input is active at any given time is to use a decoder, as shown in the diagram.

- When the enable input of the decoder is 0, all of its four outputs are 0, and the bus line is in a high-impedance state because all four buffers are disabled.

- When the enable input is active, one of the three-state buffers will be active, depending on the binary value in the select inputs of the decoder.

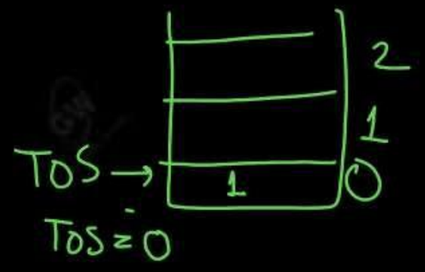
AKTU QUESTIONS

Q1	Draw a diagram of a bus system in which its uses 3 states buffer& a decoder instead of multiplexer	AKTU 2021-22 7 Marks
Q2	Draw a diagram of a bus system using MUX which has four register of size 4 bits each	AKTU 2020-21 7 Marks

STACK

- A Stack is a linear data structure that follows the LIFO (Last-In-First-Out) principle.
- Stack has one end,
- It contains only one pointer top pointer pointing to the topmost element of the stack.
- Whenever an element is added in the stack, it is added on the top of the stack, and the element can be deleted only from the stack.

top

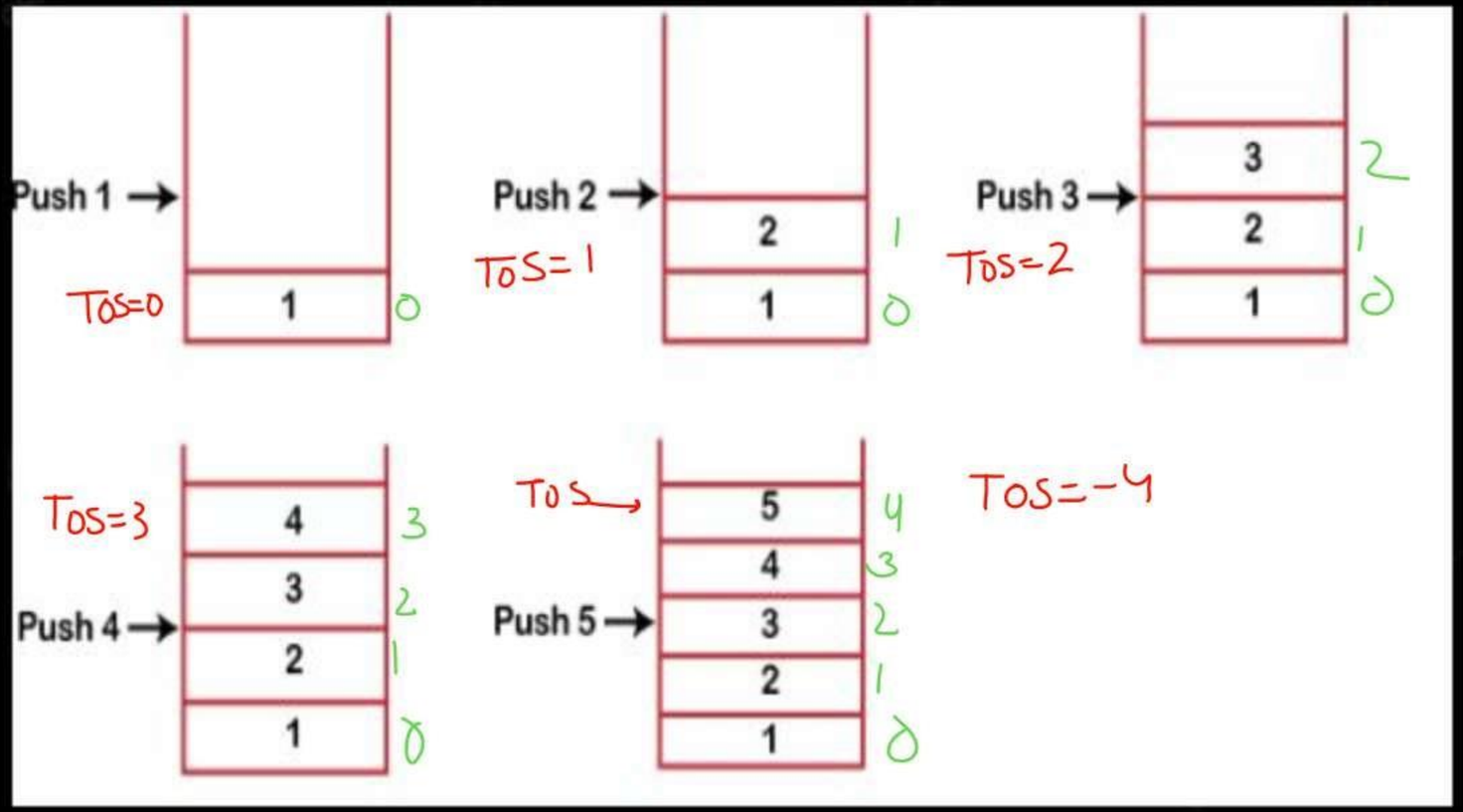


PUSH → for adding new element

POP → for Removing an element

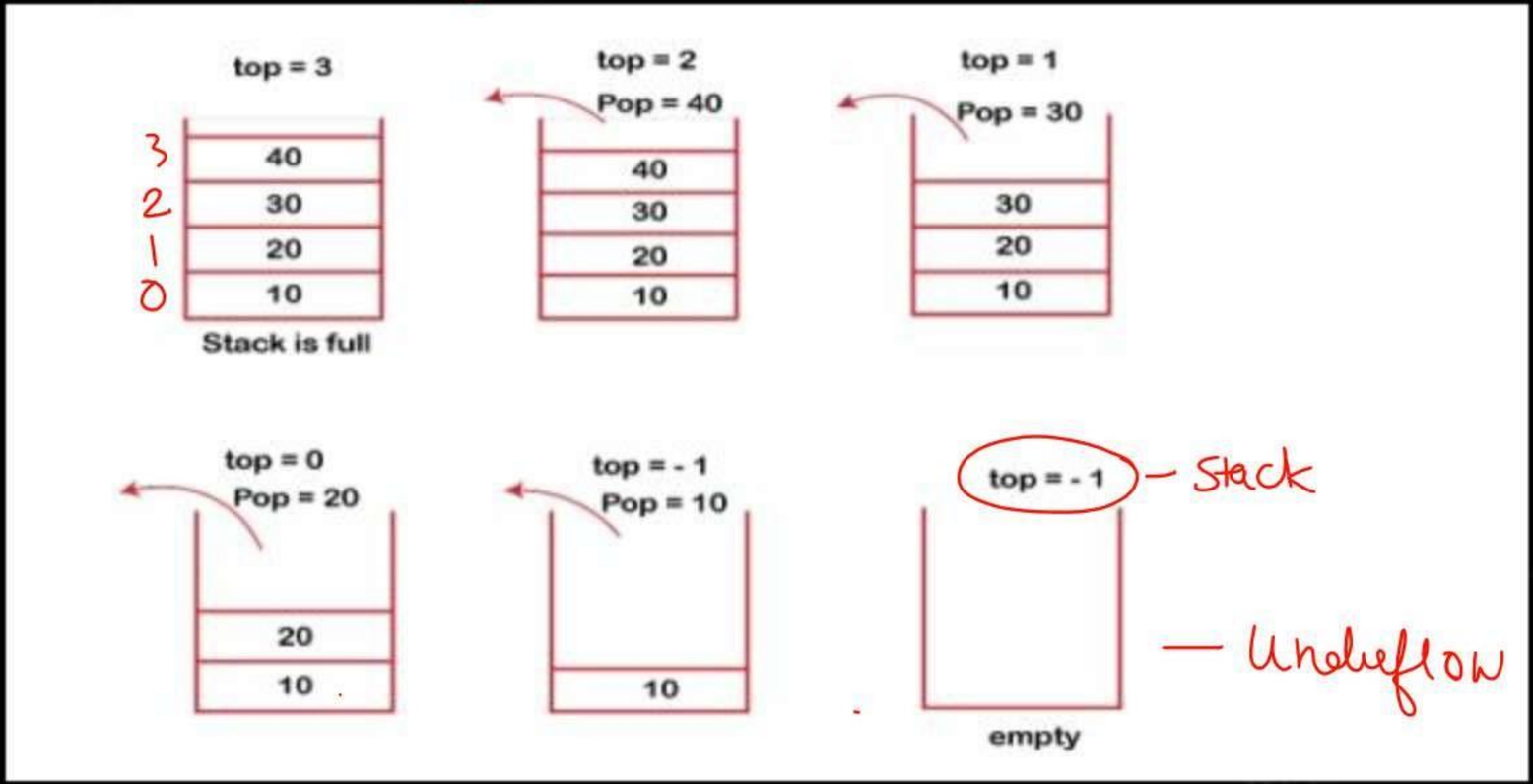
TOS → It is a pointer pointing to the top most element

STACK


Push 1
TOS=-1
TOS=0
1
0
Push 2
TOS=1
2
1
1
0
Push 3
TOS=2
3
2
2
1
1
0
TOS=3
4
3
3
2
Push 4
2
1
1
0
TOS
5
4
4
3
Push 5
3
2
2
1
1
0
TOS=-4

STACK

Overflow



Infix and Postfix

(a+b+c)*(c+d)) infix expression

(ab+c+)*(cd+)

ab+c+cd+* postfix expression

① ②

(a+b+c)*(c+d)

(ab+ +c)*((+d)

(ab+c+)*(cd+)

x y

x*y

xy*

(ab+c+)(cd+)*

a+b — Prefix + ab

Infix

a+b — Infix

operator

a and b operand

ab+ postfix

Operand operator

a*b+c — Infix

ab* +c

x — let

x+c

xc+

ab*c+

Instruction Format

- Instruction formats refer to the way instructions are encoded and represented in machine language. There are several types of instruction formats, including zero, one, two, and three-address instructions
- An opcode is a collection of bits that represents the basic operations including add, subtract, multiply, complement, and shift.
- Operands are definite elements of computer instruction that show what information is to be operated on

Zero address instruction

This instruction does not have an operand field, and the location of operands is implicitly represented. The stack-organized computer system supports these instructions. To evaluate the arithmetic expression, it is required to convert it into reverse polish notation.

Postfix

Zero address instruction

TOS: Top of the Stack Expression: X = (A+B)*(C+D) Post fix : X = AB+CD+*

PUSH A	TOS ← A
PUSH B	TOS ← B
ADD	TOS ← (A + B)
PUSH C	TOS ← C
PUSH D	TOS ← D
ADD	TOS ← (C + D)
MUL	TOS ← (C + D) ∗ (A + B)
POP X	M [X] ← TOS

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

(AB+)*(CD+)

X (let) Y (let)

X*Y

XY*

AB+CD+*

$2*3+5$

$6+5=11$

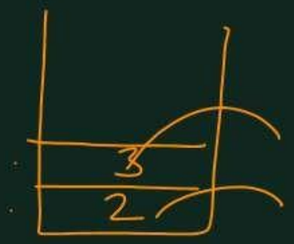
$2*3+5$

$23*+5$

$23*5+$

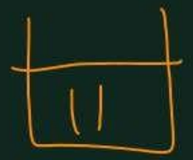
Postfix

PUSH 2
PUSH 3
MUL
PUSH 5
ADD



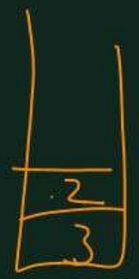
$2*3$

$6+5=11$



$3-2$

$32-$



$2-3=-1$

$3-2=1$

Zero address instruction

Zero-address instructions:

Advantages:
They are simple and can be executed quickly since they do not require any operand fetching or addressing. They also take up less memory space.

Disadvantages:
They can be limited in their functionality and do not allow for much flexibility in terms of addressing modes or operand types

One address instruction

- This instruction uses an implied accumulator for data manipulation operations.
- An accumulator is a register used by the CPU to perform logical operations.
- In one address instruction, the accumulator is implied, and hence, it does not require an explicit reference.
- For multiplication and division, there is a need for a second register. However, here we will neglect the second register and assume that the accumulator contains the result of all the operations.

MODE	OPCODE	OPERAND

One address instruction

- **Example:** The program to evaluate X = (A + B) ∗ (C + D) is as follows:

LOAD A	AC ← M [A]
ADD B	AC ← A [C] + M [B]
STORE T	M [T] ← AC
LOAD C	AC ← M [C]
ADD D	AC ← AC + M [D]
MUL T	AC ← AC ∗ M [T]
STORE X	M [X] ← AC

LOAD
STORE
MUL
DIV
ADD
SUB

One address instruction

- All operations are done between the accumulator(AC) register and a memory operand.
- M[ ] is any memory location.
- M[T] addresses a temporary memory location for storing the intermediate result.
- This instruction format has only one operand field. This address field uses two special instructions to perform data transfer,
- namely:
- LOAD: This is used to transfer the data to the accumulator.
- STORE: This is used to move the data from the accumulator to the memory

One address instruction

- These instructions specify one operand or address, which typically refers to a memory location or register. The instruction operates on the contents of that operand, and the result may be stored in the same or a different location.

One address instruction

One-address instructions:

Advantages:

They allow for a wide range of addressing modes, making them more flexible than zero-address instructions. They also require less memory space than two or three-address instructions.

Disadvantages:

They can be slower to execute since they require operand fetching and addressing

Two address instruction

➢These instructions specify two operand or address, which typically refers to a memory location or register. The instruction operates on the contents of that operand, and the result may be stored in the same or a different location.

MODE	OPCODE	OPERAND 1	OPERAND 2

Two address instruction

- This is common in commercial computers.

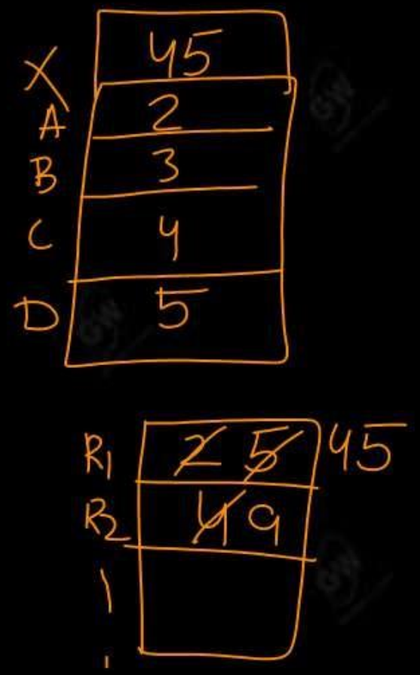
- Here two addresses can be specified in the instruction. Unlike earlier in one address instruction, the result was stored in the accumulator, here the result can be stored at different locations rather than just accumulators, but require more number of bit to represent address.

Mode	opcode	Destination address	Source address

Two address instruction

Example: The program to evaluate X = (A + B) ∗ (C + D) is as follows:

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 X, R_1	$M[X] \leftarrow R_1$.



The MOV instruction transfers the operands to the memory from the processor registers. R1, R2 registers.

Two address instruction

Two-address instructions:

Advantages:

They allow for more complex operations and can be more efficient than one-address instructions since they allow for two operands to be processed in a single instruction. They also allow for a wide range of addressing modes.

Disadvantages:

They require more memory space than one-address

Three address instruction

These instructions specify three operands or addresses, which may be memory locations or registers. The instruction operates on the contents of all three operands, and the result may be stored in the same or a different location

MODE	OPCODE	OPERAND 1	OPERAND 2	OPERAND 3

Three address instruction

- This has three address field to specify a register or a memory location.
- Program created are much short in size but number of bits per instruction increase.
- These instructions make creation of program much easier but it does not mean that program will run much faster because now instruction only contain more information but each micro operation (changing content of register, loading address in address bus etc.) will be performed in one cycle only

Mode	opcode	Destination address	Source address	Source address

Three address instruction

Example: The program in assembly language X = (A + B) ∗ (C + D) Consider the instructions given below that explain each instruction's register transfer operation.

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 X, R_1, R_2	$M[X] \leftarrow R_1 * R_2$

Three address instruction

Advantages:

- They allow for even more complex operations and can be more efficient than two-address instructions since they allow for three operands to be processed in a single instruction. They also allow for a wide range of addressing modes.

7 Marks

Disadvantages:

- They require even more memory space than two-address instructions and can be slower to execute since they require operand fetching and addressing.

- Overall, the choice of instruction format depends on the specific requirements of the computer architecture and the trade-offs between code size, execution time, and flexibility

Precedence & Associativity

L to R ←

* / %
+ −

20*3/6

60/6

= 6

20*3+6/6

60+6/6

60+1 = 61

AKTU QUESTIONS

Q1 Write a program to evaluate the arithmetic statements **AKTU 2018-19**

X=A-B+C*(D*E-F)/G+H*K

By using (i) two address instruction (ii) one address instructions (iii) zero address instruction (iv) three address

Stack address instruction

three address

$X = A - B + \overbrace{C * (D * E - F)}^{R_1} / G + \overbrace{H * K}^{R_2}$

MUL R_1, D, E	$R_1 \leftarrow M[D] * M[E]$
SUB R_1, R_1, F	$R_1 \leftarrow R_1 - M[F]$
MUL R_1, C, R_1	$R_1 \leftarrow M[C] * R_1$
DIV R_1, R_1 G	$R_1 \leftarrow R_1 / M[G]$
MUL R_2, H, K	$R_2 \leftarrow M[H] * M[K]$
SUB R_3, A, B	$R_3 \leftarrow M[A] - M[B]$
ADD R_3, R_3, R_1	$R_3 \leftarrow R_3 + R_1$
ADD X, R_3, R_2	$M[X] \leftarrow R_3 + R_2$

Two address

$$X = A - B + \overbrace{C * (D * E - F)}^{R_1} / G + H * K$$

MOV R_1, D	$R_1 \leftarrow M[D]$
MUL R_1, E	$R_1 \leftarrow R_1 * M[E]$
SUB R_1, F	$R_1 \leftarrow R_1 - M[F]$
MUL R_1, C	$R_1 \leftarrow R_1 * M[C]$
DIV R_1, G	$R_1 \leftarrow R_1 / M[G]$
MOV R_2, H	$R_2 \leftarrow M[H]$
MUL R_2, K	$R_2 \leftarrow R_2 * M[K]$
MOV R_3, A	$R_3 \leftarrow M[A]$
SUB R_3, B	$R_3 \leftarrow R_3 - M[B]$
ADD R_3, R_1	$R_3 \leftarrow R_3 + R_1$
ADD R_3, R_2	$R_3 \leftarrow R_3 + R_2$
MOV X, R_3	$M[X] \leftarrow R_3$

$$\underline{A - B} + \underline{R_1} + \underbrace{H * K}$$

$$\underline{A - B} + R_1 + R_2$$

$$\underbrace{R_3 + R_1}_{R_3} + R_2$$

$$\underbrace{R_3 + R_2}_{R_3}$$

One address

$X = A - B + C * (D * E - F) / G + H * K$

LOAD D	$AC \leftarrow M[D]$
MUL E	$AC \leftarrow AC * M[E]$
SUB F	$AC \leftarrow AC - M[F]$
MUL C	$AC \leftarrow AC * M[C]$
DIV C	$AC \leftarrow AC / M[G]$
STORE T	$M[T] \leftarrow AC$
LOAD H	$AC \leftarrow M[H]$
MUL K	$AC \leftarrow AC * M[K]$
STORE W	$M[W] \leftarrow AC$
LOAD A	$AC \leftarrow M[A]$
SUB B	$AC \leftarrow AC - M[B]$
ADD T	$AC \leftarrow AC + M[T]$
ADD W	$AC \leftarrow AC + M[W]$
STORE X	$M[X] \leftarrow AC$

Zero Address

$X = A-B+C*(D*E-F)/G+H*K$

$X = A-B+C*((DE*)-F)/G+H*K$

$X = A-B+C*(DE*F-)/G+H*K$

$X = A-B+(CDE*F-*)/G+H*K$

$X = A-B+(CDE*F-*G/)+H*K$

$X = A-B+(CDE*F-*G/)+(HK*)$

$(AB-)+(CDE*F-*G/)+(HK*)$

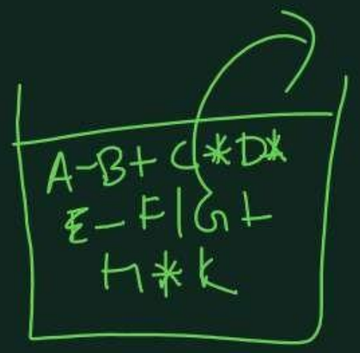
$(AB-CDE*F-*G/+)+(HK*)$

$\boxed{AB-CDE*F-*G/+HK*+}$ → Postfix Expression

Zero Address

X = AB-CDE*F-*G/+HK*+

PUSH A	TOS ← A
PUSH B	TOS ← B
SUB	TOS ← A-B
PUSH C	TOS ← C
PUSH D	TOS ← D
PUSH E	TOS ← E
MUL	TOS ← D*E
PUSH F	TOS ← F
SUB	TOS D*E-F
MUL	TOS ← C*D*E-F
PUSH G	TOS ← G
DIV	TOS ← C*D*E-F/G
ADD	TOS ← A-B+C*D*E-F/G
PUSH H	TOS ← H
PUSH K	TOS ← K



MUL	TOS ← H*K
ADD	TOS ← A-B+C*D*E-F/G+H*K
POP X	M[X] ← TOS

AKTU QUESTIONS

Q2 Write a program to evaluate the arithmetic statements AKTU 2021-22

X=(A-B)*(((C-D*E)/F)/G)

By using (i) two address instruction (ii) one address instructions (iii) zero address instruction using stack (iv) three address

three address

$X = \overset{R_1}{(A-B)} * (\overbrace{(((C-D*E)/F)/G}^{R_2})$

SUB R_1, A, B	$R_1 \leftarrow M[A] - M[B]$
MUL R_2, D, E	$R_2 \leftarrow M[D] * M[E]$
SUB R_2, C, R_2	$R_2 \leftarrow M[C] - R_2$
DIV R_2, R_2, F	$R_2 \leftarrow R_2 / M[F]$
DIV R_2, R_2, G	$R_2 \leftarrow R_2 / M[G]$
MUL X, R_1, R_2	$M[X] \leftarrow R_1 * R_2$

Twoaddress

R_1

$$X = (A - B) * (((C - D * E) / F) / G)$$

R_3 R_3 R_2

MOV R_1, A	$R_1 \leftarrow M[A]$
SUB R_1, B	$R_1 \leftarrow R_1 - M[B]$
MOV R_2, D	$R_2 \leftarrow M[D]$
MUL R_2, E	$R_2 \leftarrow R_2 * M[E]$
MOV R_3, C	$R_3 \leftarrow M[C]$
SUB R_3, R_2	$R_3 \leftarrow R_3 - R_2$
DIV R_3, F	$R_3 \leftarrow R_3 / M[F]$
DIV R_3, G	$R_3 \leftarrow R_3 / M[G]$
MUL R_1, R_3	$R_1 \leftarrow R_1 * R_3$
MOV X, R_1	$M[X] \leftarrow R_1$

One address

$$X = \overset{T}{(A-B)} * \overbrace{(((C-D*E)/F)/G)}^{AC}$$

LOAD A	$AC \leftarrow M[A]$
SUB B	$AC \leftarrow AC - M[B]$
STORE T	$M[T] \leftarrow AC$
LOAD D	$AC \leftarrow M[D]$
MULE	$AC \leftarrow AC * M[E]$
STORE W	$M[W] \leftarrow AC$
LOAD C	$AC \leftarrow M[C]$
SUB W	$AC \leftarrow AC - M[W]$
DIVF	$AC \leftarrow AC / M[F]$
DIV G	$AC \leftarrow AC / M[G]$
MULT	$AC \leftarrow AC * M[T]$
STORE X	$M[X] \leftarrow AC$

Zero address

$X = (A-B)*(((C-D*E)|F)/G)$

$(AB-)*(((C-D*E)|F)/G)$

$(AB-)*(((C-(DE*))|F)/G)$

$(AB-)*(((CDE*-)|F)/G)$

$(AB-)*(((CDE*-F|)|G)$

$(AB-)*(CDE*-F|G|)$

$\boxed{AB-CDE*-F|G|*}$ → Postfix Expression

Zero address (stack)

AB- CDE*-F|G|*

PUSH A	TOS ← A
PUSH B	TOS ← B
SUB	TOS ← A-B
PUSH C	TOS ← C
PUSH D	TOS ← D
PUSH E	TOS ← E
MUL	TOS ← D*E
SUB	TOS ← C-D*E
PUSH F	TOS ← F
DIV	TOS ← C-D*E|F
PUSH G	TOS ← G
DIV	TOS ← C-D*E|F|G
MUL	TOS ← A-B*C-D*E|F|G
POP X	M[x] ← TOS

A-B*C-D*
E|F|G

TOS ← C-D*E A-B
C-D*E|F

AKTU QUESTIONS

Q3 Evaluate the arithmetic statement- **AKTU 2020-21**

X=A+B*(C*D+F*(F+G))

Using stack organised computer with zero address operation instruction

zero address

X= A+B*(C*D+F*(F+G))

A+B*(C*D+F*(FG+))

A+B*((CD*)+F*(FG+))

A+B*((CD*)+(FFG+*))

A+B*(CD*FFG+*+)

A+(BCD*FFG+*+*)

ABCD*FFG+*+*+ – Postfix

Zero address

ABCD*FFG+*+*+

PUSH A	TOS ← A
PUSH B	TOS ← B
PUSH C	TOS ← C
PUSH D	TOS ← D
MUL	TOS ← C*D
PUSH F	TOS ← F
PUSH F	TOS ← F
PUSH G	TOS ← G
ADD	TOS ← F+G
MUL	TOS ← F*F+G
ADD	TOS ← C*D+F*F+G
MUL	TOS ← B*C*D+F*F+G
ADD	TOS ← A+B*C*D+F*F+G
POP x	M[x] ← TOS

Three address

$$X = A + B * C * D + F * (F + G)$$

R_2 R_2 R_1 R_2

ADD R_1, F, G	$R_1 \leftarrow M[F] + M[G]$
MUL R_2, C, D	$R_2 \leftarrow M[C] * M[D]$
MUL R_1, F, R_1	$R_1 \leftarrow M[F] * R_1$
ADD R_2, R_2, R_1	$R_2 \leftarrow R_2 + R_1$
MUL R_2, B, R_2	$R_2 \leftarrow M[B] * R_2$
ADD X, A, R_2	$M[X] \leftarrow M[A] + R_2$

two address

$$X = A + \overset{R_3}{B} * \overbrace{C\ \overset{R_2}{C} * D + \overbrace{F * (F + G)}^{R_1}}^{R_2}$$

MOV R_1, F	$R_1 \leftarrow M[F]$
ADD R_1 G	$R_1 \leftarrow R_1 + M[G]$
MOV R_2, C	$R_2 \leftarrow M[C]$
MUL R_2, D	$R_2 \leftarrow R_2 * M[D]$
MUL R_1 F	$R_1 \leftarrow R_1 * M[F]$
ADD R_2, R_1	$R_2 \leftarrow R_2 + R_1$
MOV R_3, B	$R_3 \leftarrow M[B]$
MUL R_3, R_2	$R_3 \leftarrow R_3 * R_2$
ADD R_3, A	$R_3 \leftarrow R_3 + M[A]$
MOV X, R_3	$M[X] \leftarrow R_3$

Oneaddress

$$X = A + B * (\overbrace{C * D}^{W} + \overbrace{F * (F + G)}^{T})$$

LOAD F	AC ← M[F]
ADD G	AC ← AC + M[G]
STORE T	M[T] ← AC
LOAD C	AC ← M[C]
MUL D	AC ← AC * M[D]
STORE W	M[W] ← AC
LOAD T	AC ← M[T]
MUL F	AC ← AC * M[F]
ADD W	AC ← AC + M[W]
MUL B	AC ← AC * M[B]
ADD A	AC ← AC + M[A]
STORX	M[X] ← AC

AKTU QUESTIONS(LECTURE 7)

Q1	Write a program to evaluate the arithmetic statements P=((X-Y+Z)*(A^B)/(C^D*F) By using (i) two address instruction (ii) one address instructions (iii) zero address instruction	AKTU 2022-23 7 Marks
Q2	Write a program to evaluate the arithmetic statements X=(A-B)*(((C-D*E)/F)/G) By using (i) two address instruction (ii) one address instructions (iii) zero address instruction using stack	AKTU 2021-22
Q3	Evaluate the arithmetic statement- X=A+B*(C*D+F*(F+G)) Using stack organised computer with zero address operation instruction	AKTU 2020-21
Q4	Evaluate the arithmetic statements X=(A-B)*(C+D) By using (i) two address instruction (ii) one address instructions (iii) zero address instruction using stack	AKTU 2018-19

Q5	Write a program to evaluate the arithmetic statements P=((A+B+C+D)*(Z*Y/(K*F) By using (i) two address instruction (ii) one address instructions (iii) zero address instruction
Q6	Write a program to evaluate the arithmetic statements X=(A+C+T)/W/S/Z+E By using (i) two address instruction (ii) one address instructions (iii) zero address instruction using stack

UNIT 1 : Introduction
Lecture -8

Today's Target

- Addressing modes
- AKTU PYQs

By PRAGYA RAJVANSHI

B.Tech, M.Tech(C.S.E)

What is addressing modes

The different ways of specifying the location of an operand in an instruction are called as **addressing modes**.

Types of addressing modes

1. Implied / Implicit Addressing Mode
2. Stack Addressing Mode
3. Immediate Addressing Mode
4. Direct Addressing Mode
5. Indirect Addressing Mode
6. Register Direct Addressing Mode
7. Register Indirect Addressing Mode

operand

Opcode	Address

2 3,

a+b

2+3

8.Relative Addressing Mode

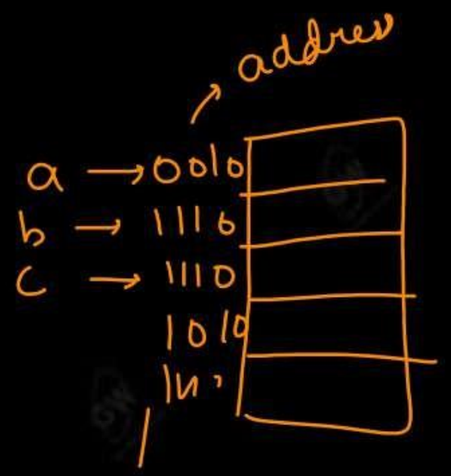
9.Indexed Addressing Mode

10.Base Register Addressing Mode

11.Auto-Increment Addressing Mode

12. Auto-Decrement Addressing Mode

effective address



1. Implied Addressing Mode

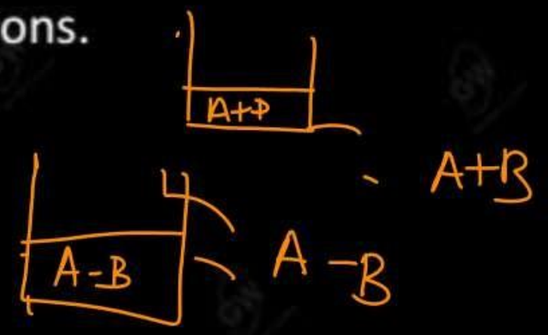
- In this addressing mode, The definition of the instruction itself specify the operands implicitly.
- It is also called as **implicit addressing mode**.
- No need to specify operand explicitly.

Examples-

- The instruction "Complement Accumulator" CMA "clear accumulator" CLA is an implied mode instruction.

ADD SUB

- In a stack organized computer, Zero Address Instructions are implied mode instructions.
- (since operands are always implied to be present on the top of the stack)



2. Stack Addressing Mode

- In this addressing mode, The operand is contained at the top of the stack.

Example-

- ADD
- This instruction simply pops out two symbols contained at the top of the stack.
- The addition of those two operands is performed.
- The result so obtained after addition is pushed again at the top of the stack.

3. Immediate Addressing Mode

- In this addressing mode, The operand is specified in the instruction explicitly.
- Instead of address field, an operand field is present that contains the operand

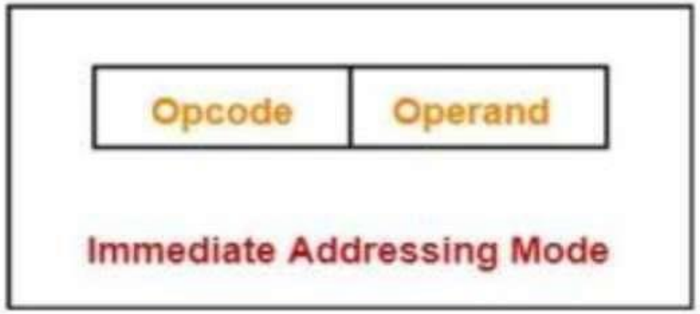
Examples- (No Memory Reference

- ADD 10 will increment the value stored in the accumulator by 10. AC ← AC+10
- MOV R #20 initializes register R to a constant value 20. R ← 20

constant

#20

hash



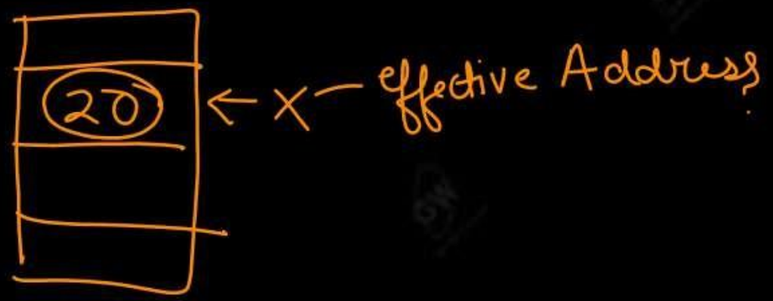
Immediate Addressing Mode

4. Direct Addressing Mode

- In this addressing mode, The address field of the instruction contains the effective address of the operand.
- The effective address is a term that describes the address of an operand that is stored in memory
- Only one reference to memory is required to fetch the operand.
- It is also called as **absolute addressing mode**.

SUB X

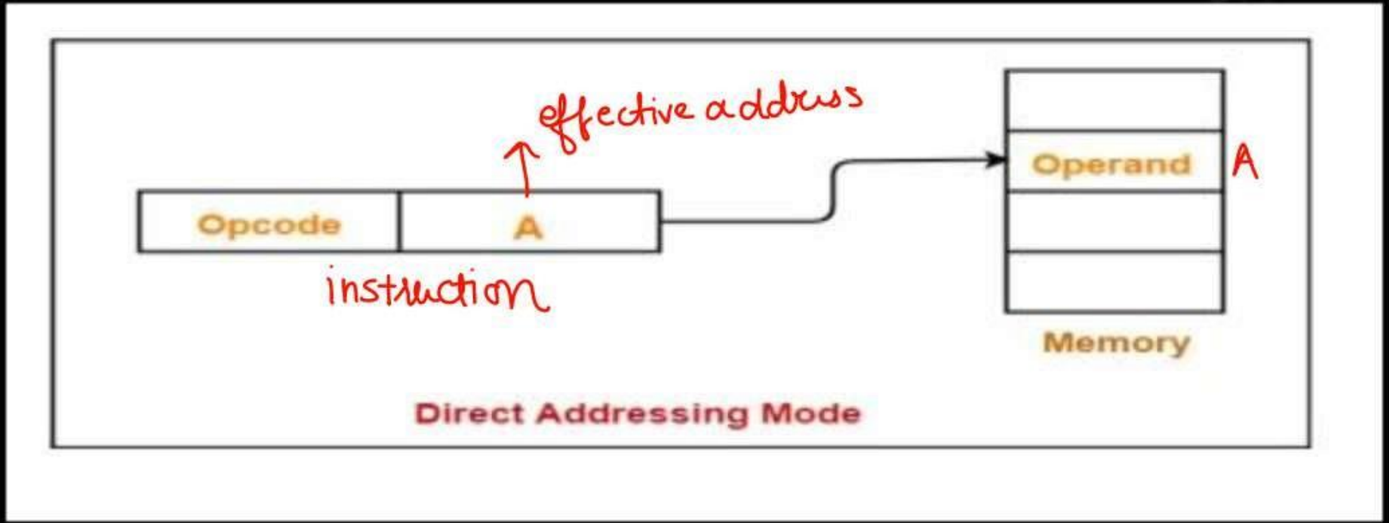
Opcode	Address



4. Direct Addressing Mode

Example-

ADD X will increment the value stored in the accumulator by the value stored at memory location X.
AC ← AC + [X]



5. Indirect Addressing Mode

In this addressing mode , The address field of the instruction specifies the address of memory location that contains the effective address of the operand.

➢Two references to memory are required to fetch the operand.

Example-

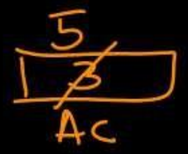
➢ADD X will increment the value stored in the accumulator by the value stored at memory location specified by X.

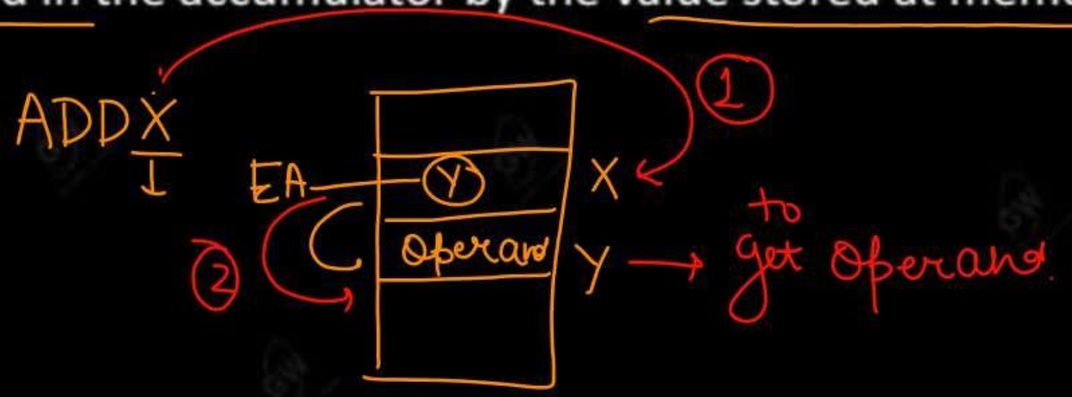
➢AC ← AC + [[X]]

[Y]

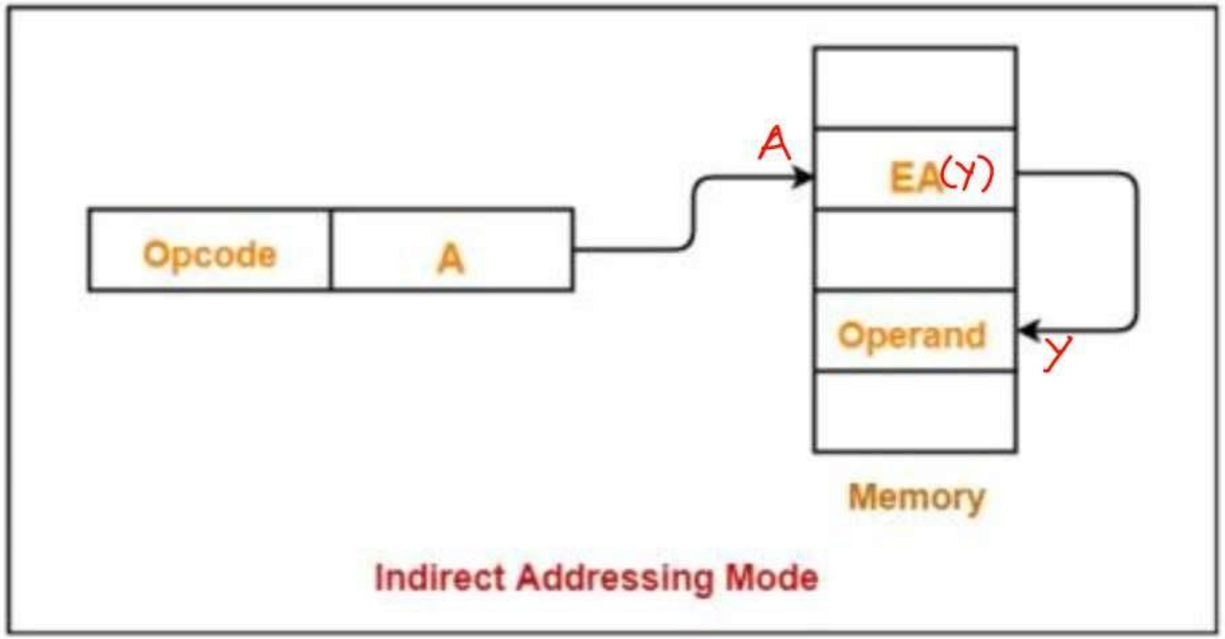
AC+ Operand(2)

3+2=5





5. Indirect Addressing Mode



Indirect Addressing Mode

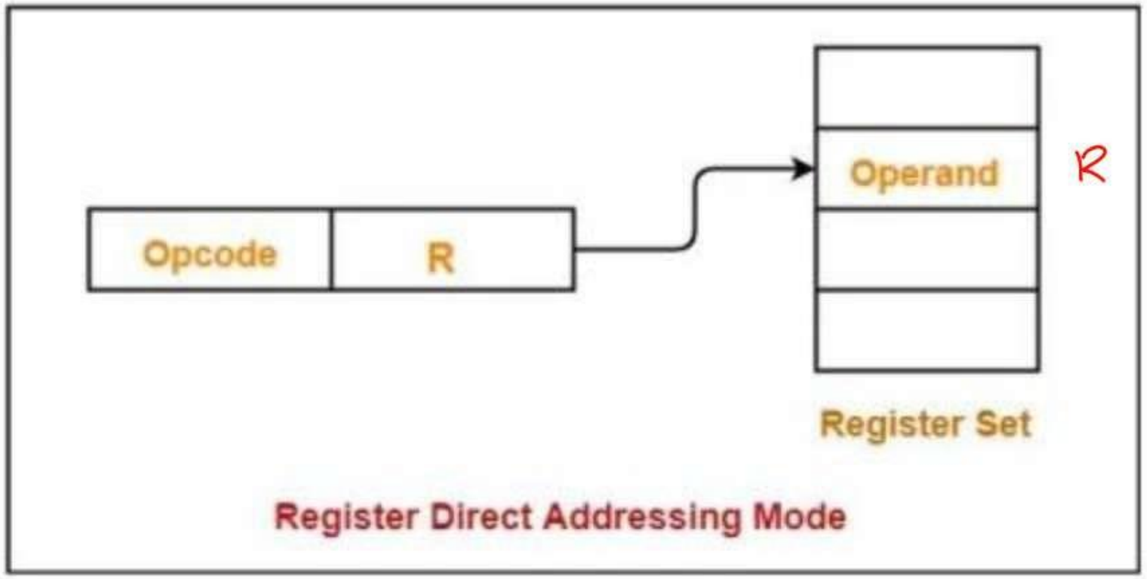
6. Register Direct Addressing Mode

- In this addressing mode, The operand is contained in a register set.
- The address field of the instruction refers to a CPU register that contains the operand.
- No reference to memory is required to fetch the operand.

Example-

- ADD R will increment the value stored in the accumulator by the content of register R.
- AC ← AC + [R]
- It is interesting to note-
- This addressing mode is similar to direct addressing mode.
- The only difference is address field of the instruction refers to a CPU register instead of main memory.

6. Register Direct Addressing Mode



Register Direct Addressing Mode

7. Register Indirect Addressing Mode-

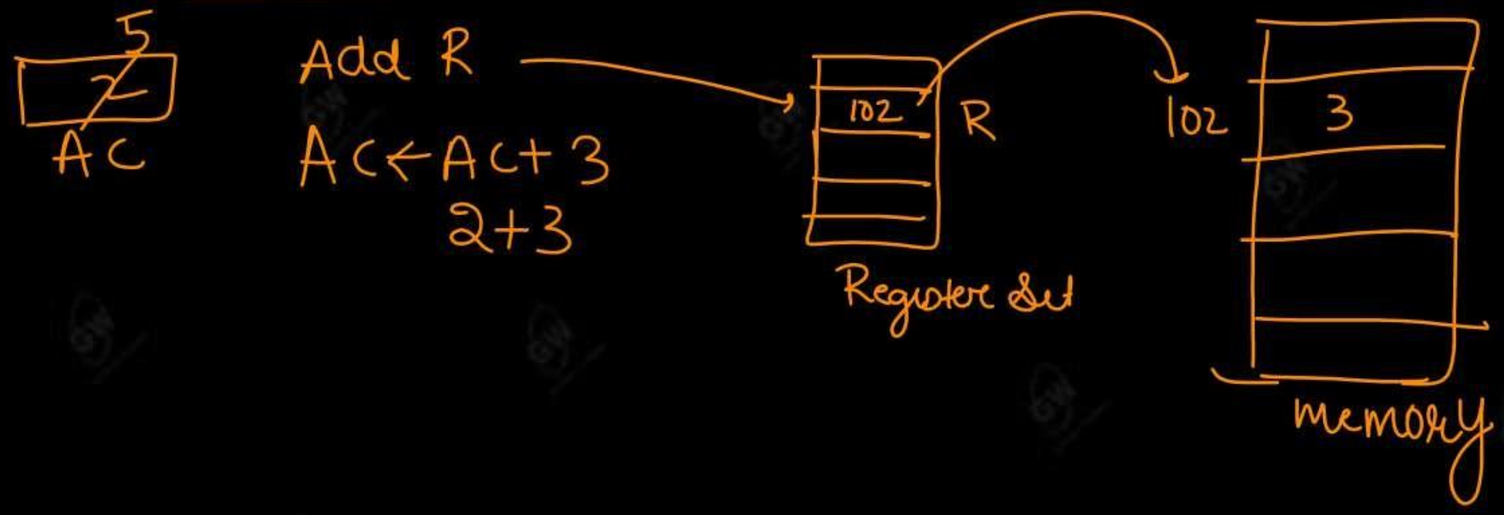
- In this addressing mode, The address field of the instruction refers to a CPU register that contains the effective address of the operand.
- Only one reference to memory is required to fetch the operand.

Example-

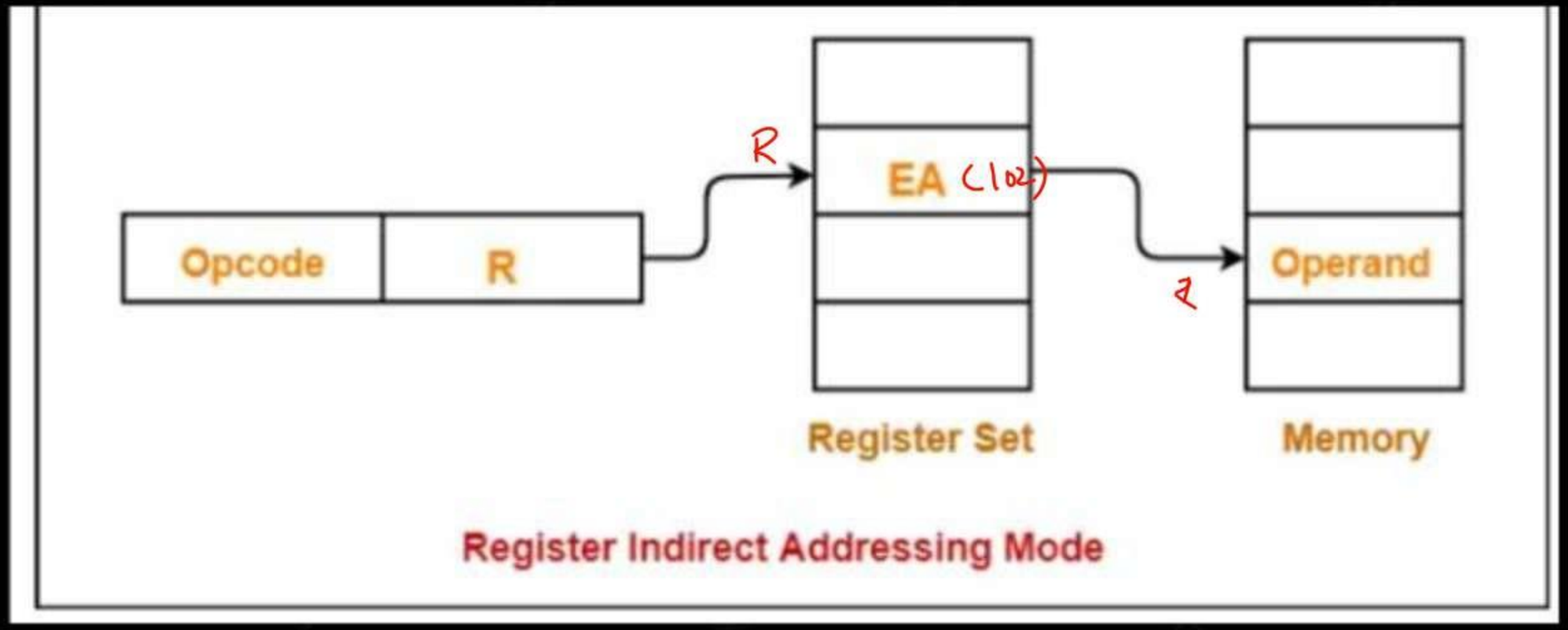
- ADD R will increment the value stored in the accumulator by the content of memory location specified in register R.
- AC ← AC + [[R]]

2 + [102]

2 + 3 = 5



7. Register Indirect Addressing Mode



Register Indirect Addressing Mode

7. Register Indirect Addressing Mode

NOTE-

- It is interesting to note-
- This addressing mode is similar to indirect addressing mode.
- The only difference is address field of the instruction refers to a CPU register

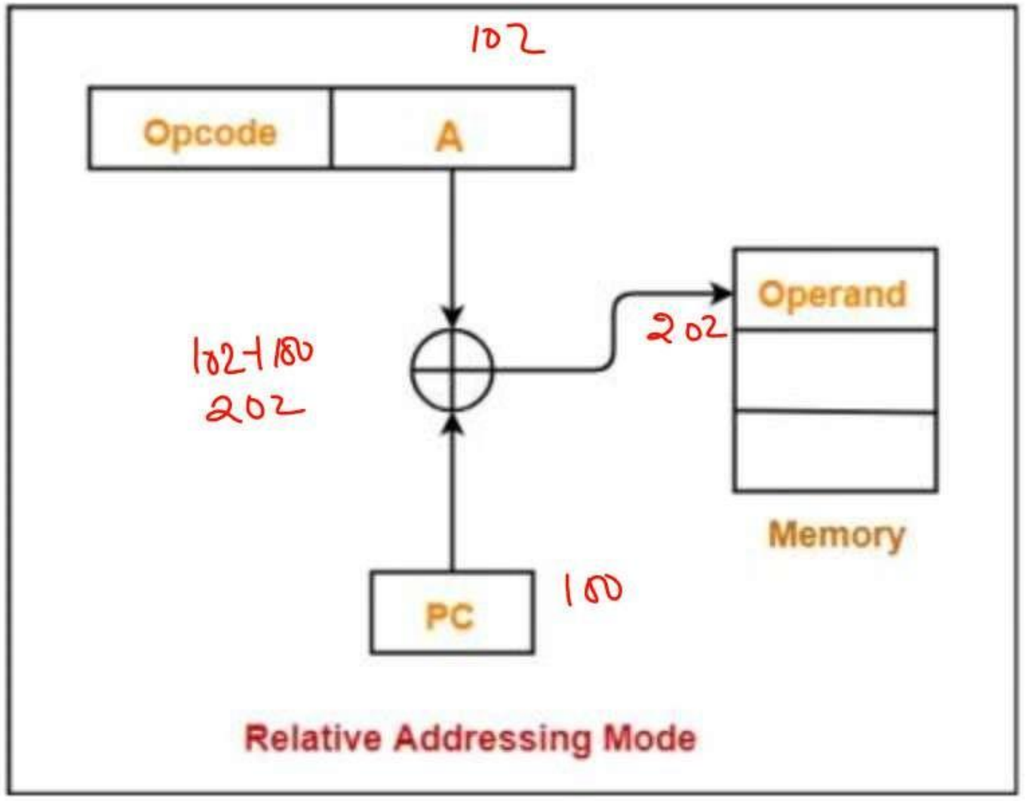
8. Relative Addressing Mode

- In this addressing mode, Effective address of the operand is obtained by adding the content of program counter with the address part of the instruction.
- **Effective Address = Content of Program Counter + Address part of the instruction**

NOTE-

- **Program counter** (PC) always contains the address of the next instruction to be executed.
- After fetching the address of the instruction, the value of program counter immediately increases.
- The value increases irrespective of whether the fetched instruction has completely executed or not.

8. Relative Addressing Mode-



Relative Addressing Mode

9. Indexed Addressing Mode

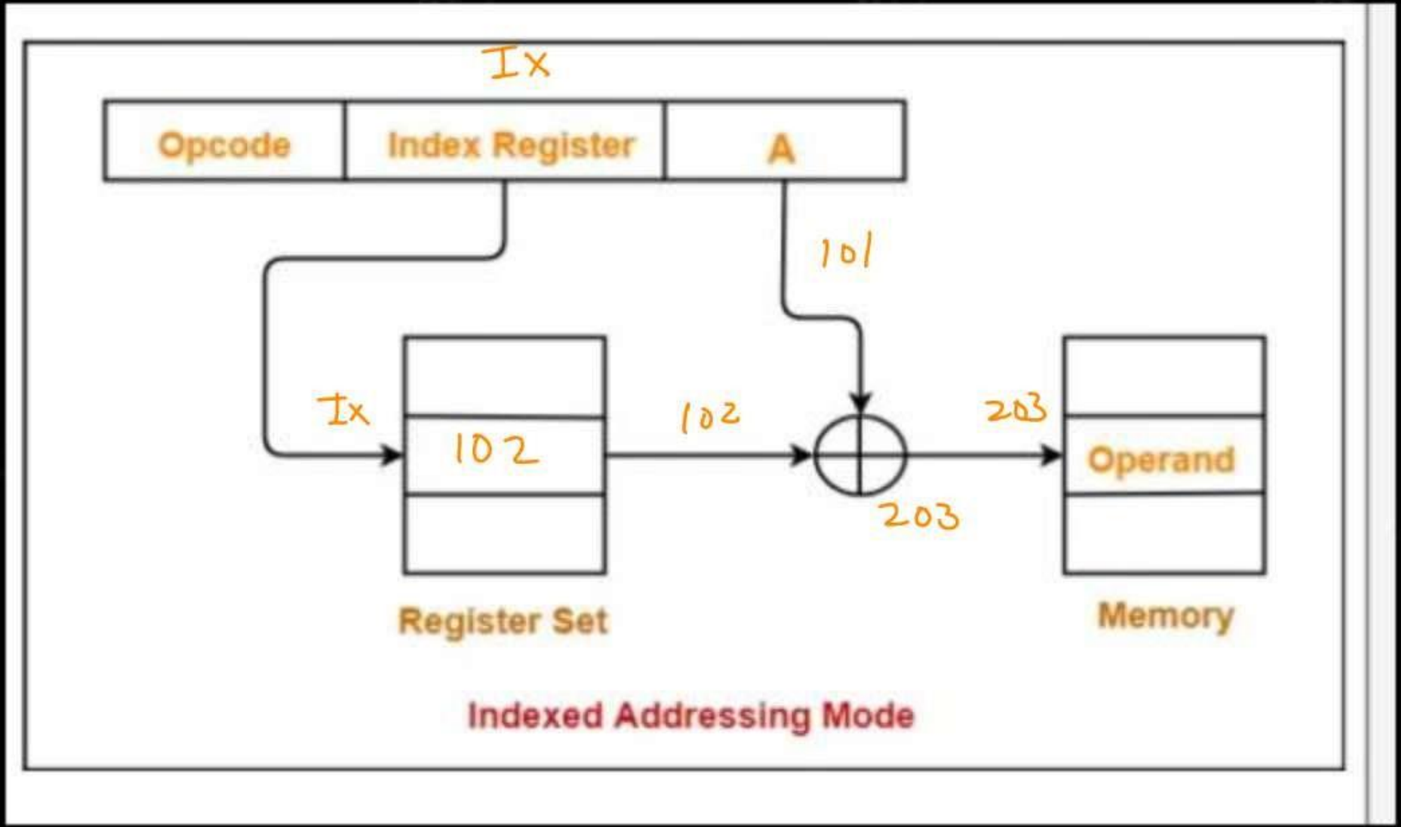
- In this addressing mode,
- Effective address of the operand is obtained by adding the content of index register with the address part of the instruction.

- **Effective Address = Content of Index Register + Address part of the instruction**

Index register

- An index register in a computer's CPU is **a processor register used for pointing to operand addresses during the run of a program**

9. Indexed Addressing Mode-

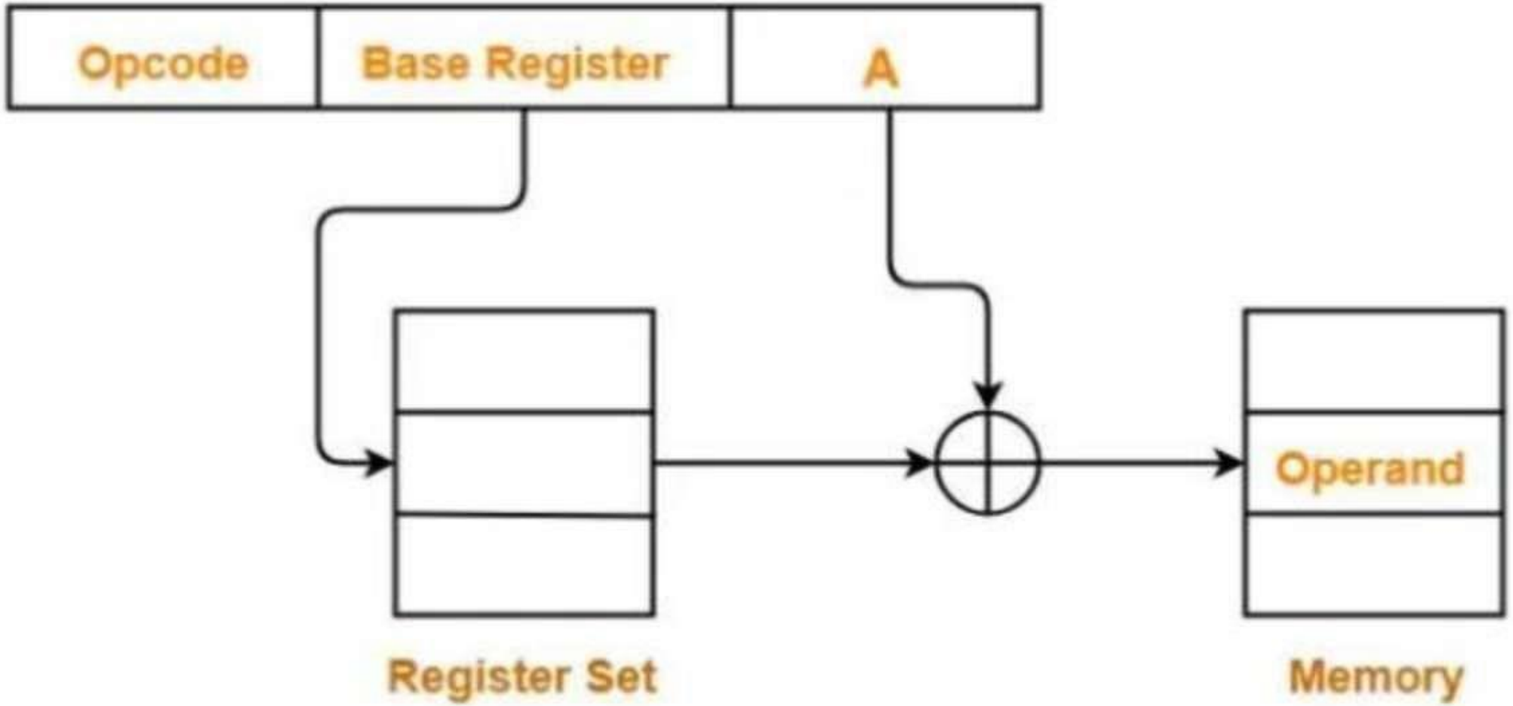


10. Base Register Addressing Mode

- In this addressing mode, Effective address of the operand is obtained by adding the content of base register with the address part of the instruction.

- **Effective Address = Content of Base Register + Address part of the instruction**

10. Base Register Addressing Mode



Base Register Addressing Mode

11. Auto-Increment Addressing Mode

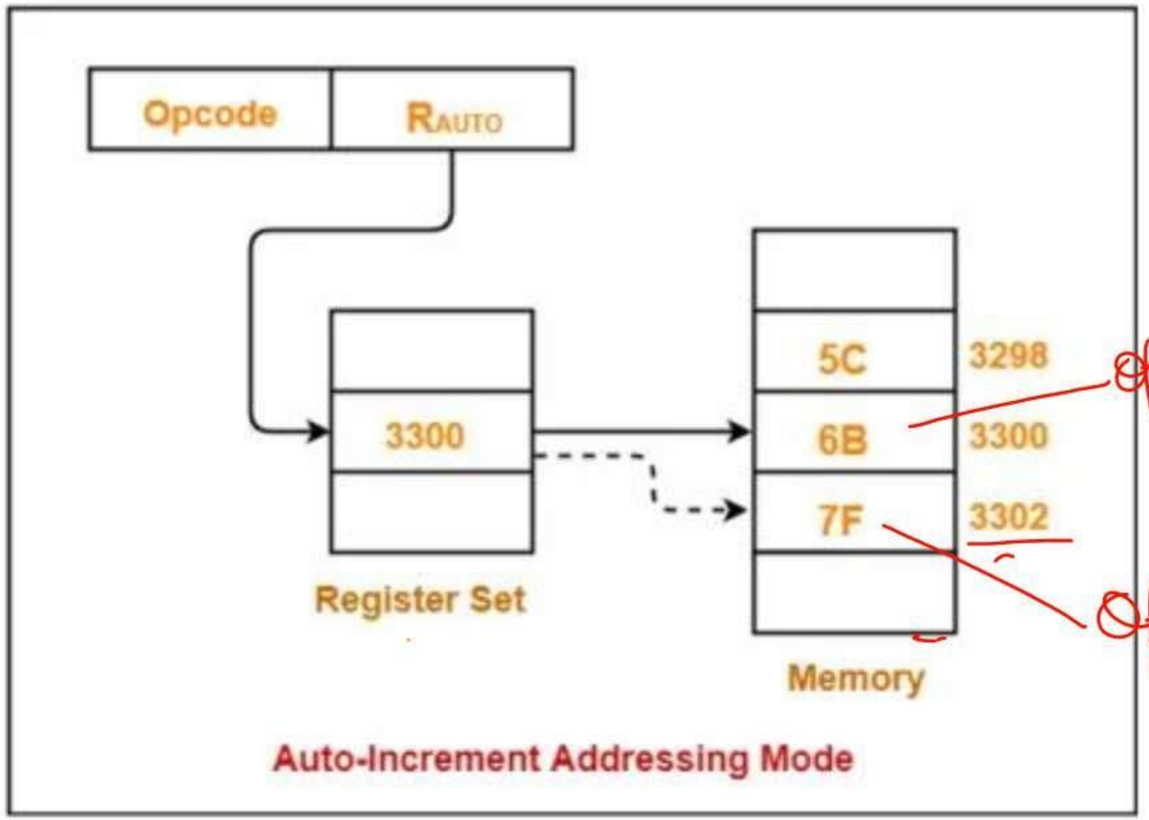
- This addressing mode is a special case of Register Indirect Addressing Mode where-
- **Effective Address of the Operand = Content of Register**
- **Effective Address of the Operand = Content of Register +d**
- In this addressing mode, After accessing the operand, the content of the register is automatically incremented by step size 'd'. Step size 'd' depends on the size of operand accessed

11. Auto-Increment Addressing Mode

- Only one reference to memory is required to fetch the operand.
- Assume operand size is 2 byte,
- After fetching the operand 6B, the instruction register R_{AUTO} will be automatically incremented by 2.
- Then, updated value of R_{AUTO} will be 3300 + 2 = 3302. — EA
- At memory address 3302, the next operand will be found.

3300

11. Auto-Increment Addressing Mode



Auto-Increment Addressing Mode

each operand size 2

Operand 1

Operand 2

12. Auto-Decrement Addressing Mode-

- This addressing mode is again a special case of Register Indirect Addressing Mode where-
- **Effective Address of the Operand = Content of Register**
- **Effective Address of the Operand = Content of Register – Step Size(d)**
- In this addressing mode, First, the content of the register is decremented by step size 'd'.
- Step size 'd' depends on the size of operand accessed. After decrementing, the operand is read.
- Only one reference to memory is required to fetch the operand

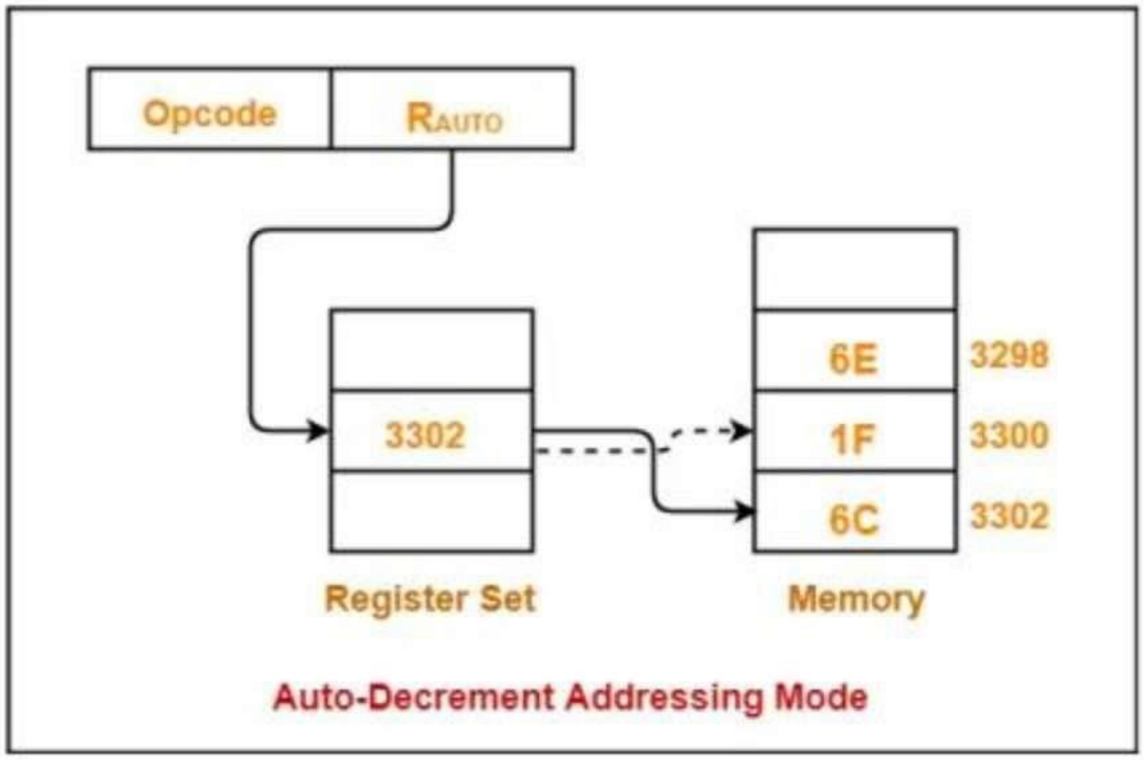
12. Auto-Decrement Addressing Mode-

- This Assume operand size = 2 bytes.
- Here, First, the instruction register R_{AUTO} will be decremented by 2.
- Then, updated value of R_{AUTO} will be 3302 – 2 = 3300.
- At memory address 3300, the operand will be found.

NOTE-

- In auto-decrement addressing mode,
- First, the instruction register R_{AUTO} value is decremented by step size 'd'.
- Then, the operand value is fetched.

12. Auto-Decrement Addressing Mode-



Auto-Decrement Addressing Mode

EXAMPLE

50	LOAD TO AC	MODE
51	ADDRESS=100	
52	NEXT INSTRUCTION	
	.....	
97	450	
98	700	
99	810	
100	105	
101	35	
102	45	
103	65	
104	75	
105	500	
152	70	

memory do calc.

Relative address= pc+ address (52+100) =152

Index address= xr +address(3+100) =103

Pc= program counter= contain next instruction address (52)

r1=98, xr= 3

Opcode | Address

Auto-In = EA (Address Part)
EA+d

implied

Opcode | Operand

EXAMPLE

operand

Addressing mode	Effective address	Content of effective address
Implied/implicit	-	-
stack	-	-
immediate	-	100
direct	100	105
Indirect	105	500
Register direct	-	98
Register indirect	98	700
Relative	152	70
indexed	103	65
Auto-increment d 2byte	100 | 102 | 104	105 | 45 | 75
Auto-decerement d 2byte	100 | 98	105 | 700

AKTU QUESTIONS

Q1	Explain various addressing mode with the help of example 1. Direct 2. implied 3. indexed 4. register indirect 5. Immediate	AKTU 2014-15
Q2	What is effective address how it is calculated in different addressing mode	AKTU 2022-23
Q3	Explain different kind of addressing modes with example	AKTU 2019-20

7 Marks

AKTU QUESTIONS

Q4 Evaluate the arithmetic statement-

X=A+B*C/D-E*F+G*H

USING 0,1,2,3 ADDRESS

$X = A + \overbrace{B * C / D}^{R_1} - \overbrace{E * F}^{R_2} + \overbrace{G * H}^{R_3}$

three address

MUL R_1, B, C	$R_1 \leftarrow M[C] * M[D]$
DIV R_1, R_1, D	$R_1 \leftarrow R_1 / M[D]$
MUL R_2, E, F	$R_2 \leftarrow M[E] * M[F]$
MUL R_3, G, H	$R_3 \leftarrow M[G] * M[H]$
ADD R_1, A, R_1	$R_1 \leftarrow M[A] + R_1$
SUB R_1, R_1, R_2	$R_1 \leftarrow R_1 - R_2$
ADD X, R_1, R_3	$M[X] \leftarrow R_1 + R_3$

$\underbrace{A + R_1}_{} - R_2 + R_3$

$\underbrace{R_1 - R_2}_{} + R_3$

$R_1 + R_3$

two address

$X = A + \overbrace{B * C / D}^{R_1} - \overbrace{E * F}^{R_2} + \overbrace{G * H}^{R_3}$

MOV R_1, B	$R_1 \leftarrow M[B]$
MUL R_1, C	$R_1 \leftarrow R_1 * M[C]$
DIV R_1, D	$R_1 \leftarrow R_1 / M[D]$
MOV R_2, E	$R_2 \leftarrow M[E]$
MUL R_2, F	$R_2 \leftarrow R_2 * M[F]$
MOV R_3, G	$R_3 \leftarrow M[G]$
MUL R_3, H	$R_3 \leftarrow R_3 * M[H]$
ADD R_1, A	$R_1 \leftarrow R_1 + M[A]$
SUB R_1, R_2	$R_1 \leftarrow R_1 - R_2$
ADD R_1, R_3	$R_1 \leftarrow R_1 + R_3$
MOV X, R_1	$M[X] \leftarrow R_1$

$$X = A + \overbrace{B * C}^{T} / D - \overbrace{E * F}^{W} + G * H$$

One address

LOAD B	AC ← M[B]
MUL C	AC ← AC * M[C]
DIV D	AC ← AC / M[D]
STORE T	M[T] ← AC
LOAD E	AC ← M[E]
MUL F	AC ← AC * M[F]
STORE W	M[W] ← AC
LOAD G	AC ← M[G]
MUL H	AC ← AC * M[H]
STORE J	M[J] ← AC
LOAD A	AC ← M[A]
ADD T	AC ← AC + M[T]
SUB W	AC ← AC − M[W]
ADD J	AC ← AC + M[J]
STORE X	M[X] ← AC

A+T−W+J

Zero address

$X = A + B * C / D - E * F + G * H$

$A + (BC*) / D - E * F + G * H$

$A + (BC*D/) - E * F + G * H$

$A + (BC*D/) - (EF*) + (GH*)$

$(ABC*D/+) - (EF*) + (GH*)$

$(ABC*D/+EF*-) + (GH*)$

$\boxed{ABC*D/+EF*-GH*+}$

Postfix Expression

$ABC*D/+EF*-GH*+$

PUSH A	TOS ← A
PUSH B	TOS ← B
PUSH C	TOS ← C
MUL	TOS ← B * C
PUSH D	TOS ← D
DIV	TOS ← B * C / D
ADD	TOS ← A + B * C / D
PUSH E	TOS ← E
PUSH F	TOS ← F
MUL	TOS ← E * F
SUB	TOS ← A + B * C / D - E * F
PUSH G	TOS ← G
PUSH H	TOS ← H
MUL	TOS ← G * H
ADD	TOS ← A + B * C / D - E * F + G * H
POP X	M[X] ← TOS

$A + B * C / D - E * F + G * H$

AKTU QUESTIONS

Q5 Write a program to evaluate the arithmetic statements **AKTU 2022-23**

P=((X-Y+Z)*(A^B)/(C^D*F)

By using (i) two address instruction (ii) one address instructions (iii) zero address instruction

^ - Exp

three address

Exp↑*

^ - XOR

*↑XOR

$P = ((\overset{R_1}{X-Y+Z}) * (\overset{R_2}{A \wedge B}) / (\overset{R_3}{C \wedge D * F})$

SUB R_1, X, Y	$R_1 \leftarrow M[X] - M[Y]$
ADD R_1, R_1, Z	$R_1 \leftarrow R_1 + M[Z]$
Exp R_2, A, B	$R_2 \leftarrow M[A] \wedge M[B]$
Exp R_3, C, D	$R_3 \leftarrow M[C] \wedge M[D]$
MUL R_3, R_3, F	$R_3 \leftarrow R_3 * M[F]$
MUL $R_1, R_1 * R_2$	$R_1 \leftarrow R_1 * R_2$
DIV P, R_1, R_3	$M[P] \leftarrow R_1 R_3$

$P = ((\overset{R_1}{X-Y+Z}) * (\overset{R_2}{A \wedge B}) / (\overset{R_3}{C \wedge D * F})$

SUB R_1, X, Y	$R_1 \leftarrow M[X] - M[Y]$
ADD R_1, R_1, Z	$R_1 \leftarrow R_1 + M[Z]$
XOR $R_2, A\ B$	$R_2 \leftarrow M[A] \wedge M[B]$
MUL R_3, D, F	$R_3 \leftarrow M[D] * M[F]$
XOR R_3, C, R_3	$R_3 \leftarrow M[C] \wedge R_3$
MUL R_1, R_1, R_2	$R_1 \leftarrow R_1 * R_2$
DIV P, R_1, R_3	$R_1 \leftarrow R_1 / R_3$

Two address

$\wedge$ EXP

$$\overset{R_1}{((X-Y+Z)} * \overset{R_2}{(A \wedge B)} / \overset{R_3}{((C \wedge D * F)}$$

MOV R_1, X	$R_1 \leftarrow M[X]$
SUB R_1, Y	$R_1 \leftarrow R_1 - M[Y]$
ADD R_1, Z	$R_1 \leftarrow R_1 + M[Z]$
MOV R_2, A	$R_2 \leftarrow M[A]$
EXP R_2, B	$R_2 \leftarrow R_2 \wedge M[B]$
MOV R_3, C	$R_3 \leftarrow M[C]$
EXP R_3, D	$R_3 \leftarrow R_3 \wedge M[D]$
MUL R_3, F	$R_3 \leftarrow R_3 * M[F]$
MUL R_1, R_2	$R_1 \leftarrow R_1 * R_2$
DIV R_1, R_3	$R_1 \leftarrow R_1 / R_3$
MOV X, R_1	$M[X] \leftarrow R_1$

$$\overset{R_1}{(X-Y+Z)} * \overset{R_2}{(A \wedge B)} / (C \wedge D * F)$$

MOV R_1, X	$R_1 \leftarrow M[X]$
SUB R_1, Y	$R_1 \leftarrow R_1 - M[Y]$
ADD R_1, Z	$R_1 \leftarrow R_1 + M[Z]$
MOV R_2, A	$R_2 \leftarrow M[A]$
XOR R_2, B	$R_2 \leftarrow R_2 \wedge M[B]$
MOV R_3, D	$R_3 \leftarrow M[D]$
MUL R_3, F	$R_3 \leftarrow R_3 * M[F]$
XOR R_3, C	$R_3 \leftarrow R_3 \wedge M[C]$
MUL R_1, R_2	$R_1 \leftarrow R_1 * R_2$
DIV R_1, R_3	$R_1 \leftarrow R_1 / R_3$
MOV X, R_1	$M[X] \leftarrow R_1$

$\wedge$ - XOR

One address

$$\overset{T}{((X-Y+Z)} * \overbrace{(A \wedge B)}^{W} / (\overbrace{C \wedge D * F}^{Z}))$$

LOAD X	$AC \leftarrow M[X]$
SUB Y	$AC \leftarrow AC - M[Y]$
ADD Z	$AC \leftarrow AC + M[Z]$
STORE T	$M[T] \leftarrow AC$
LOAD A	$AC \leftarrow M[A]$
XOR B	$AC \leftarrow AC \wedge M[B]$
STORE W	$M[W] \leftarrow AC$
LOAD D	$AC \leftarrow M[D]$
MUL F	$AC \leftarrow AC * M[F]$
XOR C	$AC \leftarrow AC \wedge M[C]$
STORE Z	$M[Z] \leftarrow AC$
LOAD T	$AC \leftarrow M[T]$
MUL W	$AC \leftarrow AC * M[W]$
DIV Z	$AC \leftarrow AC / M[Z]$
STORE P	$M[P] \leftarrow AC$

EXP B $\quad AC \leftarrow AC \wedge M[B]$

LOAD C	$AC \leftarrow M[C]$
EXP D	$AC \leftarrow AC \wedge M[D]$
MUL F	$AC \leftarrow AC * M[F]$

^-XOR

(X-Y+Z) * (A^B) | (C^D*F)

((XY-)+Z) * (A^B) | (C^D*F)

(XY-Z+) * (AB^) | (C^DF*)

(XY-Z+) * (AB^) | (CDF*^)

(XY-Z+ AB^*) | (CDF*^)

| XY-Z+AB^* CDF*^ |

firstly perform multiply then XOR

(X-Y+Z) * (A^B) | (C^D*F)

(XY-Z+) * (AB^) | ((CD^)* F)

(XY-Z+) * (AB^) | (CD^F*)

(XY-Z+ AB^*) | (CD^F*)

| XY-Z+AB^* CD^F* |

^-EXP

firstly perform Exp then Multiply

PROCESSOR ORGANISATION

Three types of processor organization

- General purpose register organization
- Stack based organization
- Accumulator based organization

0 1

0
1

$2^1 = 2$

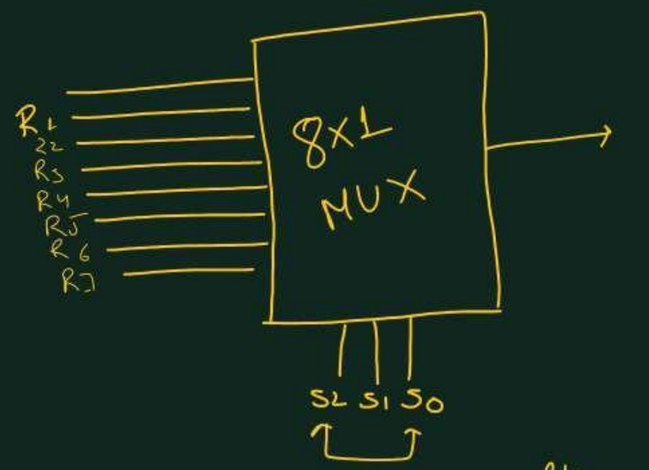
0 0
0 1
1 0
1 1

$2^2 = 4$

0 0 0
0 0 1
0 1 0
0 1 1
1 0 0
1 0 1
1 1 0
1 1 1

$2^3 = 8$

Multiplexer



Selection line

000

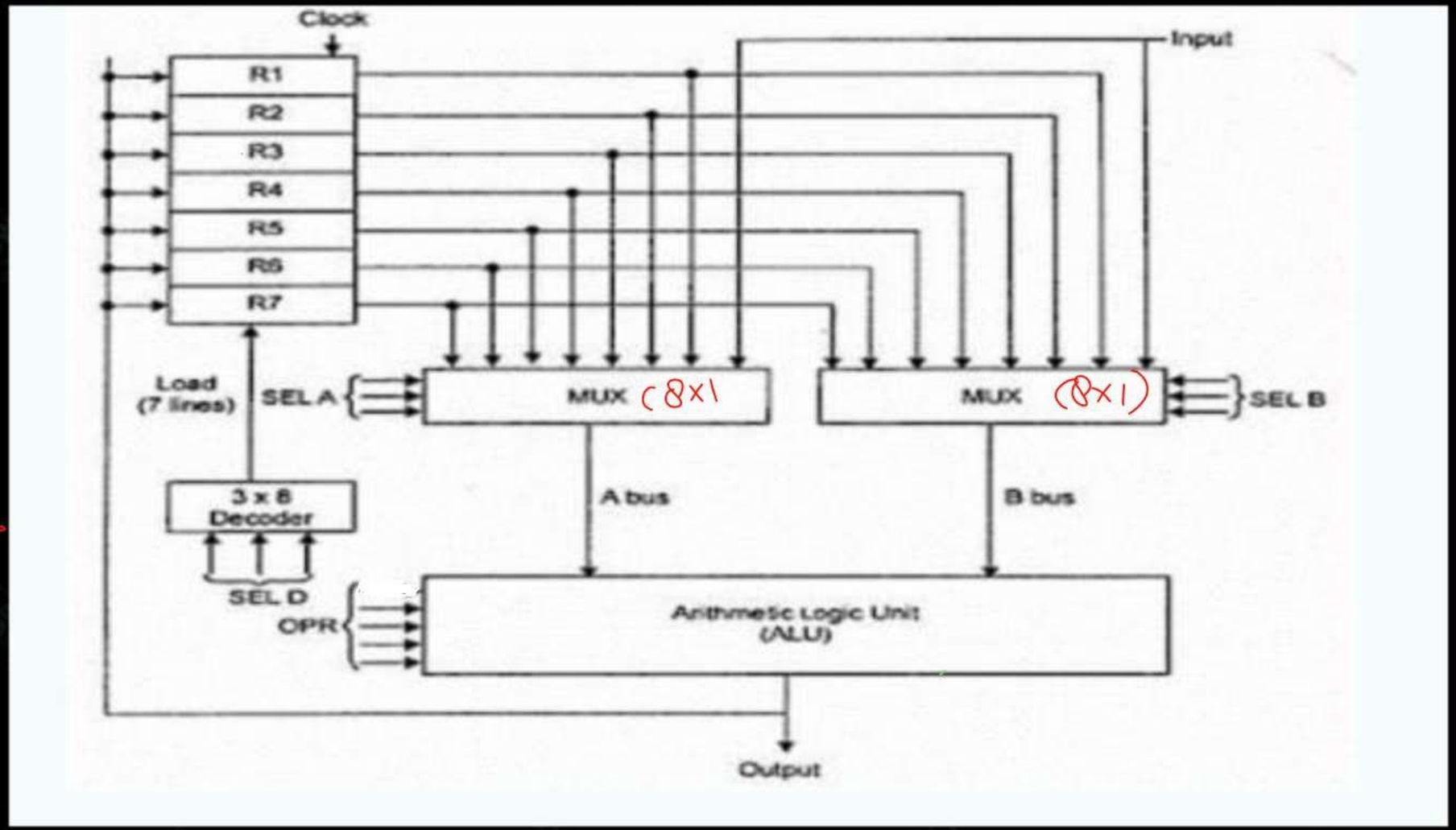
2^2	2^1	2^0	
4	2	1	
S_2	S_1	S_0	
0	0	0	R_1
0	0	1	R_2
0	1	0	R_3
0	1	1	R_4
1	0	0	R_5
1	0	1	R_6
1	1	0	R_7
1	1	1	

GENERAL REGISTER ORGANISATION

General Register organization

- Generally CPU has seven general registers. Register organization show how registers are selected and how data flow between register and ALU.
- A decoder is used to select a particular register. The output of each register is connected to two multiplexers to form the two buses A and B. The selection lines in each multiplexer select the input data for the particular bus.
- The A and B buses form the two inputs of an ALU.
- The operation select lines decide the micro operation to be performed by ALU. The result of the micro operation is available at the output bus. The output bus connected to the inputs of all registers, thus by selecting a destination register it is possible to store the result in it.

A bus organization for 7 CPU register



~~000~~
001 – R_1
010 – R_2
011 – R_3
100 – R_4
101 – R_5
110 – R_6
111 R_7

GENERAL REGISTER ORGANISATION

- In this organization, the registers communicate with each others not only for data transfer, but also while performing various micro operation
- Seven registers are used for general purpose, the output of each register is connected to two multiplexer inputs.
- Three select lines are used to select any one of the seven register and the content of the selected registers are supplied to ALU.
- The buses A and B are used to form the inputs to common ALU.
- The operation to be performed is selected in the ALU and is determined by the arithmetic or logic micro-operation by using the function select line(OPR).

GENERAL REGISTER ORGANISATION

- The result of the micro-operations is available as output data and also goes into the input of all the register
- Any one of the destination register receives the information from the output bus which is selected by the decoder

Register and multiplexer input selection code

BINARY CODE	SELA (MUX)	SELB (MUX)	SELD (Decoder)
000	INPUT	INPUT	--
001	R1	R1	R1
010	R2	R2	R2
011	R3	R3	R3
100	R4	R4	R4
101	R5	R5	R5
110	R6	R6	R6
111	R7	R7	R7

OPERATION SELECTION CODE	OPERATION	SYMBOL
0000	Transfer A	TSF A
0001	Increment A	INC A
0010	A+B	ADD
0011	A-B	SUB
0100	DECREMENT A	DEC
0101	A AND B	AND
0110	A OR B	OR
0111	A XOR B	XOR
1000	COMPLEMENT A	COMA
1001	SHIFT RIGHT A	SHR
1010	SHIFT LEFT B	SHL

Control word

The combined value of a binary selection inputs specifies the control word.

- It consist of four fields SEL A, SEL B, and SEL D or SELREG contains three bit each and SELOPR field contains four bits thus the total bits in the control word are 13-bits

SLE A	SEL B	SEL D 0R SEL REG	SEL OPR

EXAMPLE

To perform the operation ***R3 = R1+R2*** We have to provide following binary selection variable to the select inputs.

1. **SEL A : 001** -To place the contents of R1 into bus A.
2. **SEL B : 010** - to place the contents of R2 into bus B
3. **SEL OPR : 0010** – to perform the arithmetic addition A+B
4. **SEL REG or SEL D : 011** – to place the result available on output bus in R3.

SLE A 001	SEL B 010	SEL D OR SEL REG 011	SEL OPR 0010

Control word

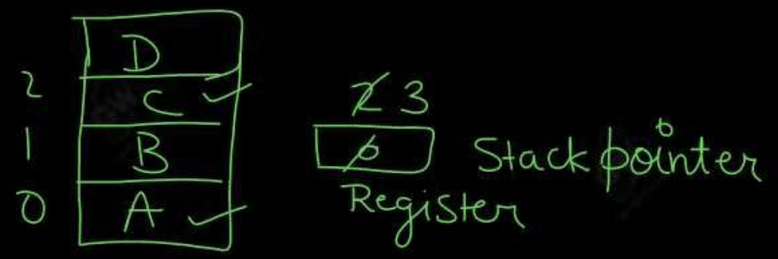
1. The three bit of SELA select a source registers of the a input of the ALU.
2. The three bits of SELB select a source registers of the b input of the ALU.
3. The three bits of SELED or SELREG select a destination register using the decoder.
4. The four bits of SELOPR select the operation to be performed by ALU.

STACK BASED ORGANISATION

- The Last In First Out (LIFO) list is another name for stack. It is the CPU's most crucial feature. It saves information so that the last element saved is retrieved first. A memory space with an address register is called a stack. This register, known as the Stack Pointer, affects the stack's address (SP). The address of the element at the top of the stack is continuously influenced by the stack pointer.
- **Implementation of Stack**

The stack can be implemented using two ways:

- ✓ Register Stack
- ✓ Memory Stack



STACK BASED ORGANISATION

$2^6 \rightarrow$ 6bit

Implementation of Registers as a Stack.

- The Implementation of registers as a stack. Mostly 64 general registers are use for stack implementation, so 6 bits are require to address 64 registers. SP is a stack pointer (top of the stack) which is use to store the address of that location where we have to push or pop any data. As SP stores the address of that location where we PUSH or POP data, so its size will be 6 bits. We use two flags registers named as FULL and EMPTY.
- If the value of FULL is 1, then Empty will be zero and vice versa.
- and If FULL value is 1 then it means there is no more space to enter any new data.
- If EMPTY value is 1 than it means we can add new data is stack organized registers.

STACK BASED ORGANISATION

2^6 2^5 2^4 2^3 2^2 2^1 2^0
64 32 16 8 4 2 1

Full | Empty

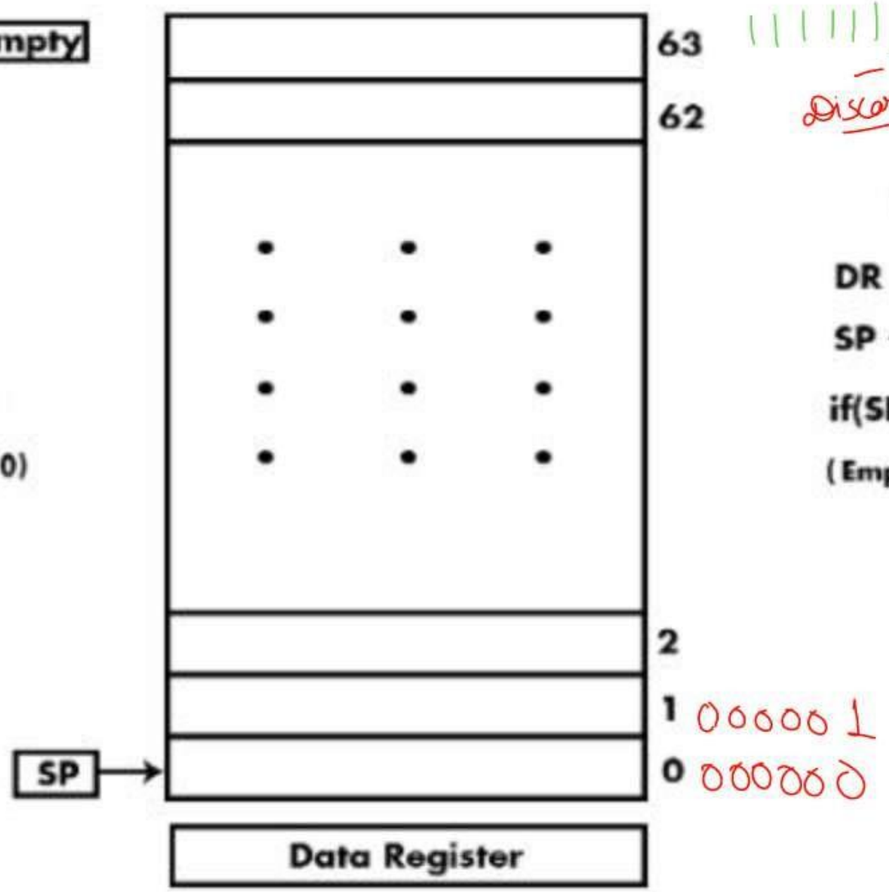
PUSH

SP ← SP + 1

M [SP] ← DR

if(SP=0) then

(FULL ← 1) AND (EMPTY ← 0)



POP

DR ← M [SP]

SP ← SP - 1

if(SP=0) then

(Empty ← 1) AND (FULL ← 0)

Discard

element – 1 address

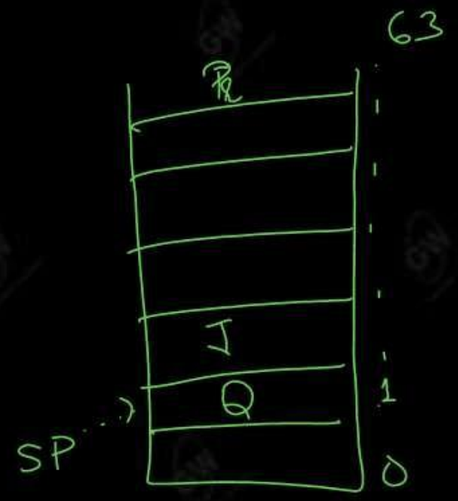
1st element – 0 address

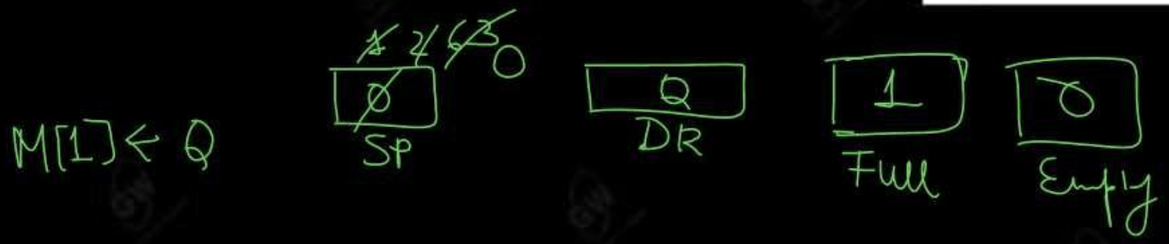
STACK BASED ORGANISATION

How to PUSH DATA?

- As in diagram SP points Zero. When we want to add any data from the data register to the stack register. SP incremented by 1 and Points the first general register having address 1. If we want to add new data then SP incremented to again and data will be pushed to the second general register and so on. In this way, 63 (1 to 63) data values can be added to different 64 general-purpose registers.
- When 64th data value is add to the register stack organization then SP points to ZERO because (64= 1000000). As SP size is 6-bits and a higher bit is discarded and least 6-bits are considered. So, SP value becomes (000000=0). When SP=0 then FULL=1 and EMPTY=0 which means there is no more space to PUSH 65th data value in stack

STACK BASED ORGANISATION





PUSH

SP ← SP+1

M[SP] ← DR

if(SP=0) then

(FULL ← 1) AND (EMPTY ← 0)

STACK BASED ORGANISATION

How to POP DATA?

If we want to retrieve data from stack to Data Register then we use POP command. Memory location of SP Address is POP and sends data to Data Register. SP, decremented by 1. In same way next memory location of SP address is POP. We can remove more data from stack until SP points to zero.

When SP = 0 then EMPTY =1 and FULL = 0. It means all stack is empty.

POP
DR ← M[SP]
SP ← SP-1
if(SP=0) then
(EMPTY ← 1) AND (FULL ← 0)

STACK BASED ORGANISATION

DATA REGISTER

It holds the data to be written into or to be read from the stack

EMPTY REGISTER

It can store 1 bit of information. It set 1 when stack is empty.

FULL REGISTER

It can store 1 bit of information. It set 1 when stack is full

STACK POINTER

It contain a value in binary each of 6 bits , which is the address of the top of stack

STACK BASED ORGANISATION

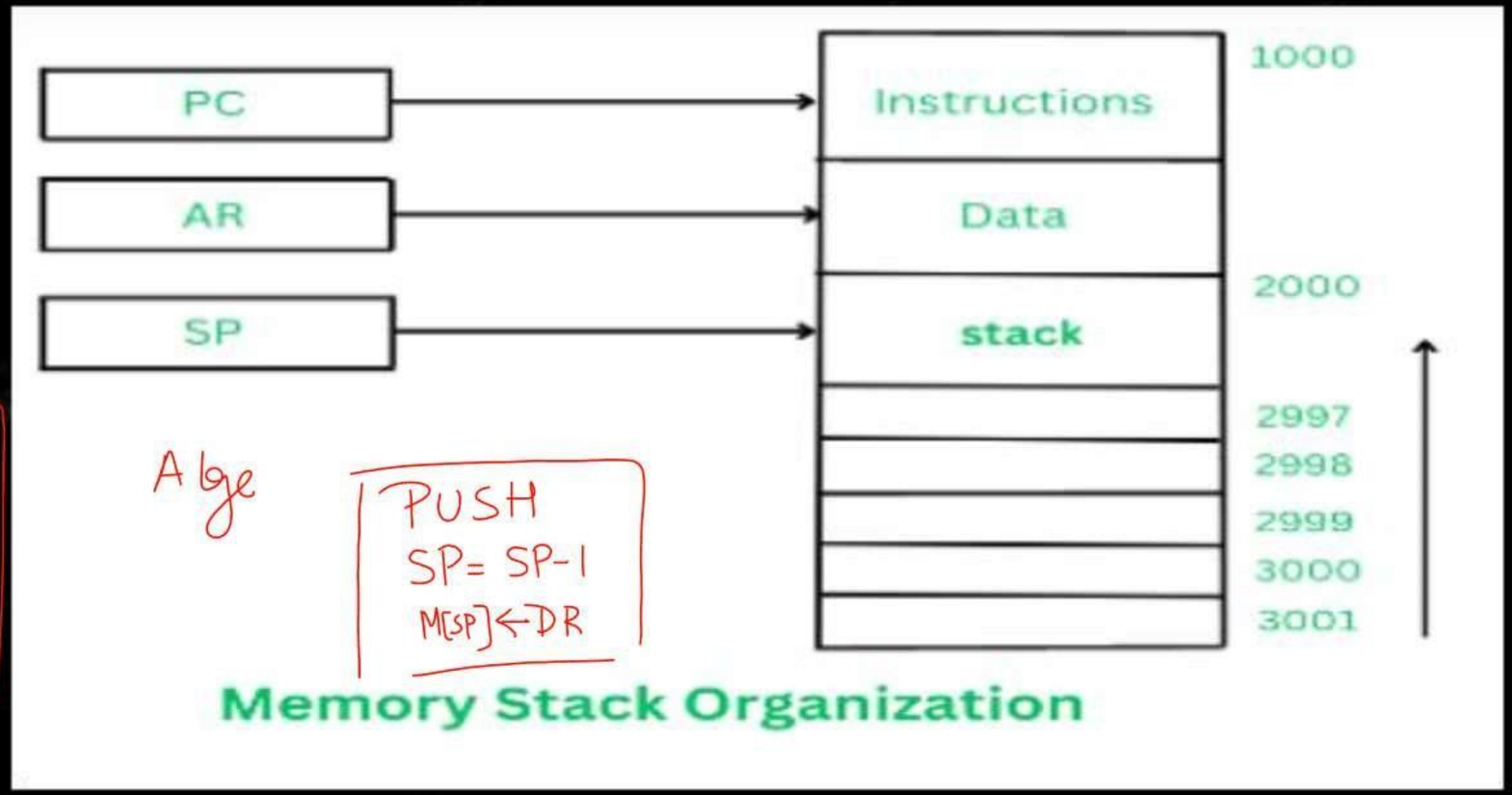
Memory Stack

- A stack may be implemented in a computer's random access memory (RAM). A stack is implemented in the CPU by allocating a chunk of memory to a stack operation and utilizing a processor register as a stack pointer. The stack pointer is a CPU register that specifies the stack's initial memory address.

STACK BASED ORGANISATION

- A portion of memory is assigned to a stack operation to implement the stack in the CPU. Here the processor register is used as a Stack Pointer (SP). The below figure shows the portion of computer memory divided into three segments: Program Instructions, Data, and Stack.
- **Program Counter (PC):** It is a register that points to the address of the **next instruction** that is going to be executed in the program.
- **Address Register (AR):** This register points at the collection of data and is used during the execute phase to **read an operand**.
- **Stack Pointer (SP):** It points at the top of the stack and is used to **push or pop** the data items in or from the stack.

STACK BASED ORGANISATION



Algo

POP

DR ← M[SP]

SP = SP+1

Algo

PUSH

SP = SP-1

M[SP] ← DR

What is subroutine?

- A set of instructions that are used repeatedly in a program can be referred to as a Subroutine. Only one copy of this Instruction is stored in the memory. When a Subroutine is required it can be called many times during the Execution of a particular program. A call Subroutine Instruction calls the Subroutine. Care Should be taken while returning a Subroutine as a Subroutine can be called from a different place from the memory

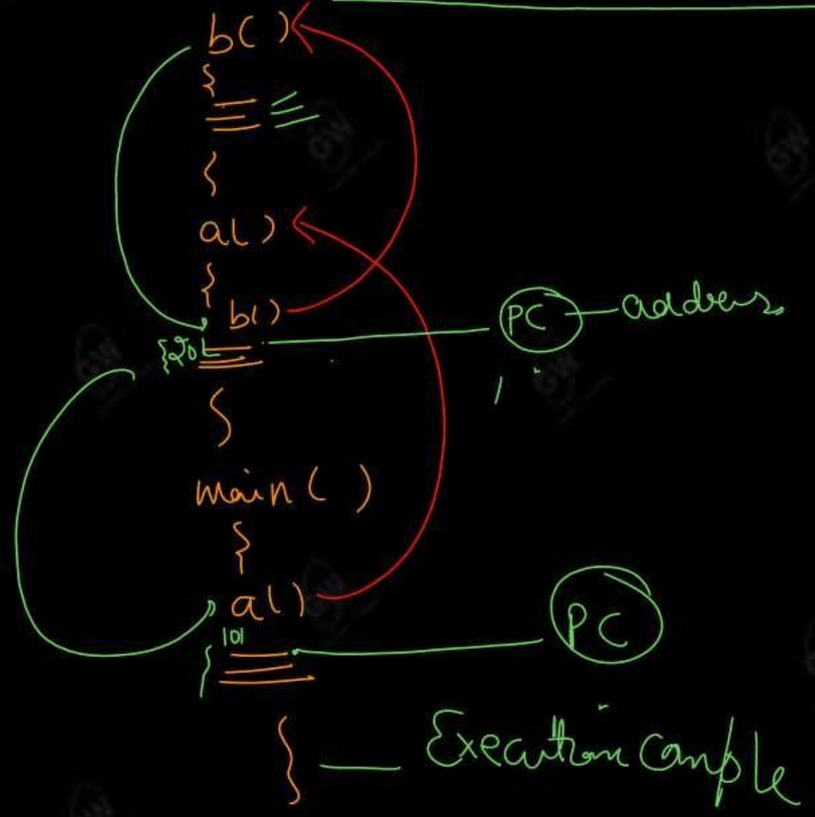
- The content of the PC must be Saved by the call Subroutine Instruction to make a correct return to the calling program

ROLE OF MANAGING SUBROUTINE

The stack support program execution by maintaining automatic process state data.

If the main subroutine of a program for example, invokes function a() which in turn invokes b() , function b() will eventually return control to function a() which in turn will return control to the main function.

```
b()
{
.....
}
a()
{
b()
}
main()
{ a() }
```



ROLE OF MANAGING SUBROUTINE

- To return control to the proper location, the sequence of return address must be stored
- A stack is well suited for maintaining this information because it is a dynamic data structure that can support any level of nesting within memory constraints
- When a subroutine is called, the address of the next instruction to execute is pushed onto the stack
- When the subroutine return, this returned address is popped from the stack, and program execution jumps to the specified location.
- The information maintained in the stack reflects the execution state of the process at any given instant .

ROLE OF MANAGING SUBROUTINE

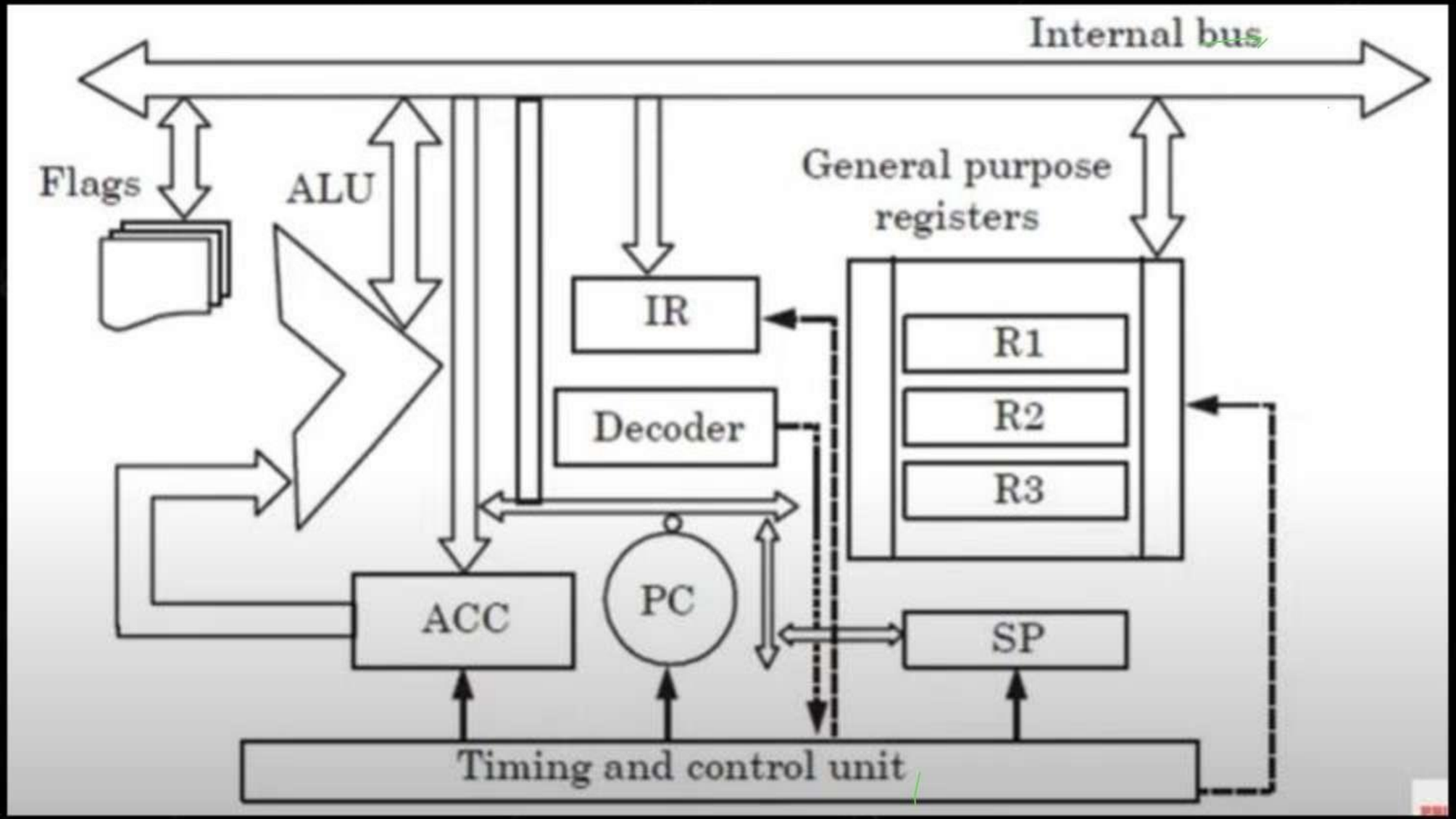
- In addition to the return address, the stack is used to store the argument to the subroutine as well as local variable.
- Information pushed onto the stack as a result of a function is called a frame. The address of the current frame is stored in the frame .
- When a subroutine is called, the frame pointer for the calling routine is also pushed onto the stack so that it can be restored when subroutine exit.

STACK FRAME FOR b()
STACK FRAME FOR a()
STACK FRAME FOR MAIN()

ROLE OF MANAGING SUBROUTINE

REGISTER STACK	MEMORY STACK
Register stack are momentary spaces for internal processes that are being done by the processor	Memory stack are a series of memory spaces that is used in the processes that is done by processor and are temporarily stored in register
Register stack is generally on CPU	Memory stack is on RAM
Access to register stack is faster	Access to memory stack is slower
It is a limited in size	It is large in size

ACCUMULATOR BASED ORGANISATION



ACCUMULATOR BASED ORGANISATION

Program counter:

- program counter (PC) is a CPU register in the computer processor which has the address of the next instruction to be executed from memory.

Accumulator:

- The accumulator is a register in which intermediate arithmetic and logical results are stored.

Instruction register (IR) :

- it is used to hold the instruction that is currently being executed.

ALU:

An arithmetic-logic unit is the part of a central processing unit that carries out arithmetic and logic operations on the operand

ACCUMULATOR BASED ORGANISATION

- BUS

A bus that connects major components (CPU, memory and I/O devices) of a computer system is called as a System Bus.

The system bus consists of three major components-Data Bus, Address Bus, Control Bus

- General purpose register

General purpose registers are extra registers that are present in the CPU and are utilized anytime data or a memory location is required. These registers are used for storing operands and pointers. These are mainly used for holding the following:

- Operands for logical and arithmetic operations
- Operands for address calculation
- Memory pointer

ACCUMULATOR BASED ORGANISATION

FLAG

- the **Flag register** is a Special Purpose Register. Depending upon the value of the result after any arithmetic and logical operation.
- Carry , sign , parity flag etc

TIMING AND CONTROL SIGNAL

This block manages the sequencing of event on a timeline between various component of a CPU

DECODER

It decode the fetch instruction

AKTU QUESTIONS

Instruction format, addressing mode

Q1	What is processor organization? Explain different types of processor organization Processor organization includes the architecture, instruction set, and the design of the processor's internal components, such as its registers, arithmetic logic unit (ALU), and control unit	AKTU 2020-21
Q2	What is stack? Give the organization of register stack with all necessary element and explain the working of push and pop	AKTU2016-17
Q3	What is memory stack?Explain it role in managing subroutine with the help of neat diagram	AKTU 2016-17, 2013-14
Q4	Explain about the stack organization use in processor. What do you understand by register stack	AKTU 2022-23
Q4	Explain stack organization in detail? Register stack Memory stack	AKTU 2019-20

GW
GATEWAY CLASSES

AKTU Full Courses (Paid)
Download Gateway Classes Application
From Google Play store
All Subjects

Link in Description

Thank You