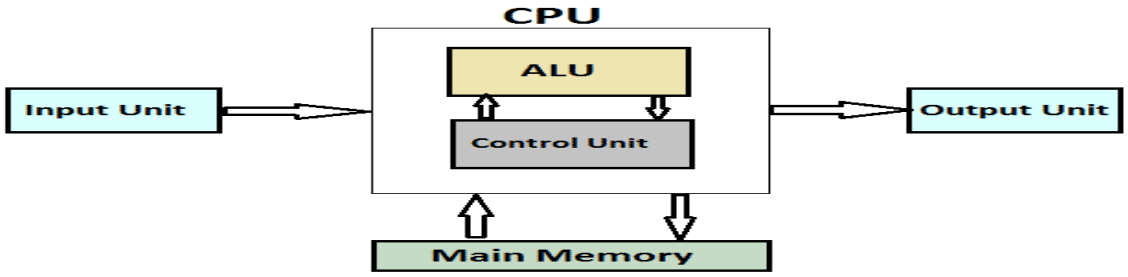
UNIT-1

Difference between computer Architecture and computer organization

Computer Architecture	Computer Organization
Computer Architecture is concerned with the way hardware components are connected together to form a computer system.	Computer Organization is concerned with the structure and behaviour of a computer system as seen by the user.
It acts as the interface between hardware and software.	It deals with the components of a connection in a system.
Computer Architecture helps us to understand the functionalities of a system.	Computer Organization tells us how exactly all the units in the system are arranged and interconnected.
A programmer can view architecture in terms of instructions, addressing modes and registers.	Whereas Organization expresses the realization of architecture.
While designing a computer system architecture is considered first.	An organization is done on the basis of architecture.
Computer Architecture deals with high-level design issues.	Computer Organization deals with low-level design issues.
Architecture involves Logic (Instruction sets, Addressing modes, Data types, Cache optimization)	Organization involves Physical Components (Circuit design, Adders, Signals, Peripherals)

Functional units of digital system and their interconnections



CPU:

A CPU is an electronics circuit used in a computer that fetches the input instructions or commands from the memory unit, performs arithmetic and logic operations and stores this processed data back to memory.

Control Unit

The Control Unit is an internal part of a CPU that co-ordinates the instructions and data flow

between CPU and other components of the computer. It is the CU that directs the operations of a central processing unit by sending timing and control signals.

Arithmetic Logic Unit(ALU)

The ALU is an internal electronic circuitry of a CPU that performs all the arithmetic and logical operations in a computer. The ALU receives three types of inputs.

- Control signal from CU (Control Unit)
- Data(operands) to be operated from(memory)
- Status information from operations done previously.

When all the instructions have been operated, the output that consists of data is stored in memory and a status information is stored in internal registers of a CPU.

Working of a CPU

All the CPUs regardless of their origin or type performs a basic instruction cycle that consists of three steps named *Fetch, decode and Execute*

Fetch

A program consists of a number of instructions. Various programs are stored in memory. During this step, the CPU reads instruction that is to be operated from a particular address in the memory. The *program counter* of CPU keeps the record of address of the instructions.

Decode

A circuitry called i*nstruction decoder* decodes all the instructions fetched from the memory. The instructions are decoded to various signals that control other areas of CPU.

Execute

In the last step, the CPU executes the instruction.

Memory:-

A memory is just like a human brain. It is used to store data and instructions. Computer memory is the storage space in the computer, where data is to be processed and instructions required for processing are stored.
Memory is primarily of four types –

- CPU Register
- Cache Memory
- Primary Memory/Main Memory
- Secondary Memory

CPU Register:--

Register is the fastest memory in computer which resides on CPU. This memory is accessed by CPU very fast. But register memory are very expensive.

Cache Memory

Cache memory is a very high speed memory which can speed up the CPU. It acts as a buffer between the CPU and the main memory. It is used to hold those parts of data and program which are most frequently used by the CPU. The parts of data and programs are transferred from the disk to cache memory by the operating system, from where the CPU can access them.

Primary Memory(RAM,ROM)

RAM (Main Memory)

Primary memory holds only those data and instructions on which the computer is currently working. It has a limited capacity and data is lost when power is switched off. It is generally made up of semiconductor device. These memories are not as fast as cache memory. The data and instruction required to be processed resides in the main memory. It is divided into two subcategories RAM and ROM.

RAM is of two types −

- Static RAM (SRAM)
- Dynamic RAM (DRAM)

Static RAM (SRAM)

The word **static** indicates that the memory retains its contents as long as power is being supplied. However, data is lost when the power gets down due to volatile nature. SRAM chips use a matrix of 6-transistors and no capacitors. Transistors do not require power to prevent leakage, so SRAM need not be refreshed on a regular basis.
There is extra space in the matrix, hence SRAM uses more chips than DRAM for the same amount of storage space, making the manufacturing costs higher. SRAM is thus used as cache memory and has very fast access.

Dynamic RAM (DRAM)

DRAM, unlike SRAM, must be continually **refreshed** in order to maintain the data. This is done by placing the memory on a refresh circuit that rewrites the data several hundred times per second. DRAM is used for most system memory as it is cheap and small. All DRAMs are made up of memory cells, which are composed of one capacitor and one transistor.

ROM:-Read only memory

Once data has been written onto a **ROM** chip, it cannot be removed and can only be read. **ROM** retains its contents even when the computer is turned off. **ROM** is referred to as being nonvolatile, whereas RAM is volatile.

Secondary Memory or Auxiliary memory

This type of memory is also known as external memory or non-volatile. It is slower than the main memory. These are used for storing data/information permanently. CPU directly does not access these memories, instead they are accessed via input-output . The contents of secondary memories are first transferred to the main memory, and then the CPU can access it. For example, Hardisk, CD-ROM, DVD, pen-drives etc.

What is Bus

=>Bus is a subsystem that is used to transfer data and other information between devices. Means various devices in computer like(Memory, CPU, I/O and Other) are communicate with each other through buses.

=>In general, a bus is said to be as the communication pathway connecting two or more devices.

=>Typically, a bus consists of multiple communication Pathways or lines which are either in the form of wires or metal lines etched in a card or board(Printed Circuit Board).Each line is capable of transmitting binary 1 and binary 0.

Types Of Bus:

There are two types of buses

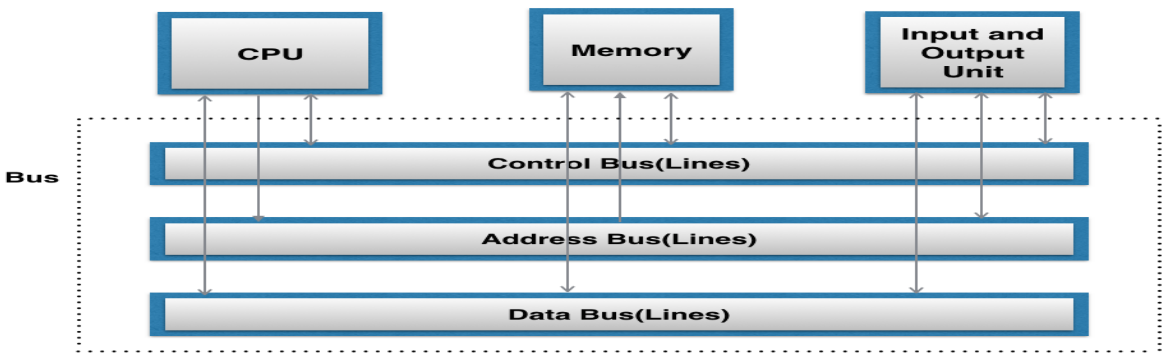
1. System Bus or Internal Bus
2. Peripheral Bus(I/O Bus /External Bus):

1.System Bus:

A Bus that connects major computer components (Processor, Memory, I/O) is called a System Bus. It is also Known as "front side " Bus.

Types of System Bus:

i.Data Bus ii. Address Bus iii. Control Bus



i.Data Bus:

Data Bus provide a path for moving data between system modules.It is bidirectional which means data lines are used to transfer data in both directions. As an example, CPU can read data on these bus from memory as well as send data out of these lines to a memory location or to a port. And in any bus the no. of lines in data bus are either 8,16,32 or more depending on size of bus.

ii. Address Bus:

Address Lines are collectively called as address bus. In any bus, the no. of lines in address are usually 16,20,24, or more depending on type and architecture of bus.On these lines, CPU sends out the address of memory location on I/O Port that is to be written on or read from.

iii. Control Bus:

Control Lines are collectively called as Control Bus. Control Lines are used by CPUs for Communicating with other devices within the computer.
As an example-CPU sends signals on the Control bus to enable the outputs of address memory devices and I/O devices. Typical Control Lines signals are:

-Memory Read
-Memory Write

2.Peripheral Bus(I/O Bus /External Bus /Expansion Bus):

A peripheral bus is a computer bus designed to support computer peripheral like printers,hard drives etc.

Example:

ISA(Industry Standard Architecture Bus):

An **Industry Standard Architecture** bus (ISA bus) developed by IBM in 1981 is a 16 bits computer bus that allows additional expansion cards to be connected to a computer's motherboard .Up to end of 1990s almost all PCs computers were equipped with ISA Bus, but it was progressively replaced by the PCI Bus,which offer a better performance.

MCA(Micro Channel Architecture): Stands for "Micro Channel Architecture." It is an expansion **bus** created by IBM in 1987 that was used in the company's PS/2 desktop computers.

EISA(Extended Industry Standard Architecture): *EISA* is extended to 32 bits and allows more than one *CPU* to share the *bus.*

SCSI: Small *Computer* System Interface (*SCSI*) is a set of standards for physically connecting and transferring data between *computers* and peripheral devices.

AGP: The Accelerated Graphics Port (*AGP*) was designed as a high-speed point-to-point channel for attaching a Graphics card to a *computer* system. It was originally designed as a successor to PCI-type connections for video cards.

PCI(Peripheral Component Interconnect): PCI Bus connects the CPU and expansion boards such as modem cards ,network cards and sound cards.These expansion boards are normally plugged into expansion slots on the motherboard.That's why PCI bus is also known as expansion bus or external Bus.

PCI-X: short for Peripheral Component Interconnect eXtended, is a computer bus Which are faster than PCI.

USB(Universal Serial Bus): Universal Serial Bus is used to attach USB devices like Pen Drive, etc to CPU.

Bus Architecture:

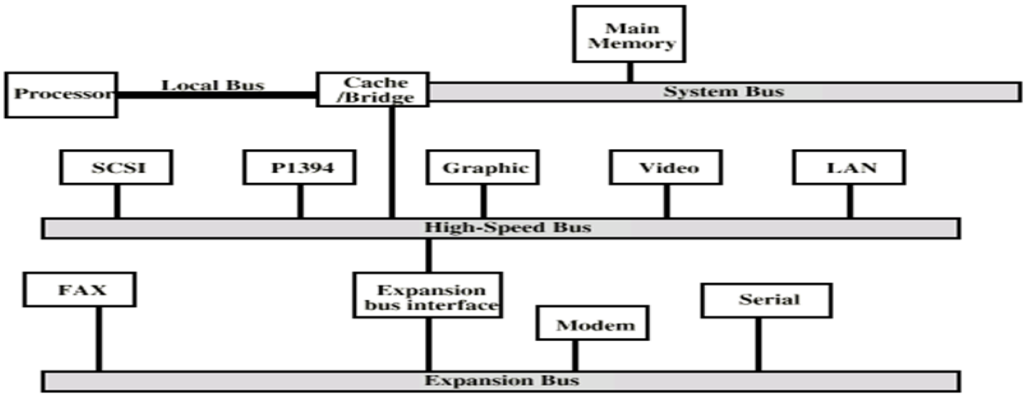


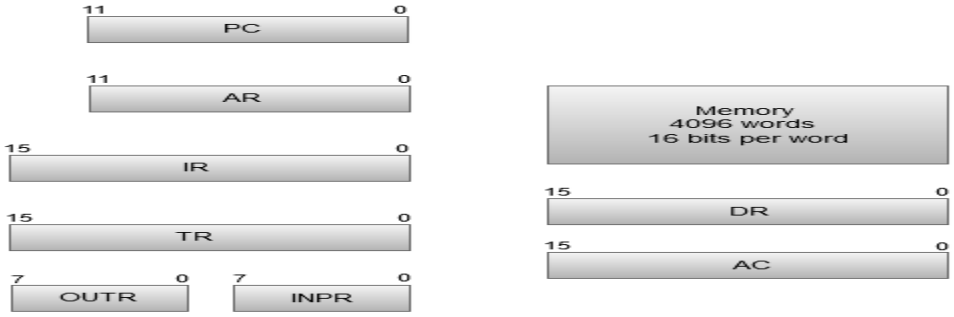
Fig above shows a High performance Bus Architecture. There is a local bus that connects the processor to a cache controller, which is in turn connected to a system bus that supports main memory. The cache controller is integrated into a bridge, that connects to the high-speed bus. This bus supports connections to high-speed LANs, and graphics workstation controllers, as well as interface controllers to local peripheral buses, including SCSI . The latter is a high-speed bus arrangement specifically designed to support high-capacity I/O devices. Lower-speed devices are still supported off an expansion bus, with an interface buffering traffic between the expansion bus and the high-speed bus.The advantage of this arrangement is that the high-speed bus brings high demand devices into closer integration with the processor.

Registers:

Register is a very fast computer memory, used to store data/instruction in-execution.

Register is a group of flip-flops with each flip-flop capable of storing **one bit** of information. An *n-bit* register has a group of *n flip-flops* and is capable of storing binary information of *n-bits*.

Register and Memory Configuration of a basic computer:



- In the Above fig The Memory unit has a capacity of 4096 words, and each word contains 16 bits.
- DR (Data Register) : contains 16 bits which hold the operand read from the memory location.
- MAR (The Memory Address Register): contains 12 bits which hold the address for the memory location.
- PC (The Program Counter) :also contains 12 bits which hold the address of the next instruction to be read from memory after the current instruction is

executed.

- AC (The Accumulator):register is a general purpose processing register.
- IR (Instruction register):The instruction read from memory is placed in the Instruction register (IR).
- TR(The Temporary Register) :is used for holding the temporary data during the processing.
- IR (The Input Registers) :holds the input characters given by the user.
- OR (Output Registers) :holds the output after processing the input data.

Micro-Operations

The operations executed on data stored in registers are called micro-operations. A micro-operation is an elementary operation performed on the information stored in one or more registers.

Types of Micro-Operations

The micro-operations in digital computers are of 4 types:

1. Register transfer or Register transfer micro-operations or Register transfer language: transfer binary information from one register to another.
2. Arithmetic micro-operations : perform arithmetic operations on numeric data stored in registers.
3. Logic micro-operations : perform bit manipulation operation on non-numeric data stored in registers.
4. Shift micro-operations : perform shift micro-operations performed on data.

Arithmetic Micro-Operations

Some of the basic Arithmetic micro-operations are addition, subtraction, increment and decrement.

Ex: Add Micro-Operation

It is defined by the following statement:

$R3 \leftarrow R1 + R2$

The above statement instructs the data or contents of register R1 to be added to data or content of register R2 and the sum should be transferred to register R3.

Ex: Subtract Micro-Operation

It is defined by the following statement:

$R3 \leftarrow R1 + R2' + 1$

In subtract micro-operation, instead of using minus operator we take **1's compliment** and add 1 to the register which gets subtracted, i.e **R1 - R2** is equivalent to **R3 → R1**

+ R2' + 1

Ex: Increment/Decrement Micro-Operation

Increment and decrement micro-operations are generally performed by adding and subtracting 1 to and from the register respectively.

R1 ← R1 + 1(Increment)
R1 ← R1 − 1(Decrement)

Logic Micro-Operations:

These are binary micro-operations performed on the bits stored in the registers.

For example the X-OR micro-operation with the contents of two registers R1 and R2.

R1 ← R1 X-OR R2

Shift Micro-Operations

These are used for serial transfer of data. That means we can shift the contents of the register to the left or right. In the **shift left** operation the serial input transfers a bit to the right most position and in **shift right** operation the serial input transfers a bit to the left most position.

Register Transfer or Register Transfer Operation or Register Transfer Language(RTL):

The symbolic notation used to describe the micro-operation transfers amongst registers is called **Register transfer language**.

Following are some commonly used registers:

1. **Accumulator(AC)**: This is the most common register, used to store data taken out from the memory.
2. **General Purpose Registers**: This is used to store data intermediate results during program execution. It can be accessed via assembly programming.

   Ex: R1, R2, R3

3. **Special Purpose Registers**: Users do not access these registers. These registers are for Computer system,
   - **MAR or AR:** Memory Address Register are those registers that holds the address for memory unit.
   - **MBR or DR:** Memory Buffer Register stores instruction and data received from the memory and sent from the memory.
   - **PC:** Program Counter points to the next instruction to be executed.
   - **IR:** Instruction Register holds the instruction to be executed.

Register Transfer:

Information transferred from one register to another is designated in symbolic form by means of replacement operator.

R2 ← R1

It denotes the transfer of the data from register R1 into R2.

Normally we want the transfer to occur only in predetermined control condition. This can be shown by following **if-then** statement: if (P=1) then (R2 ← R1)

Here P is a control signal generated in the control section.

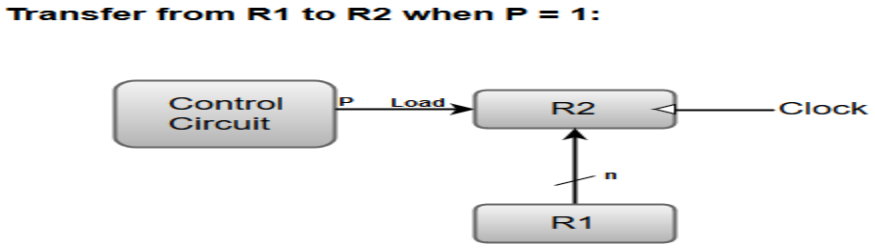
Control Function

A control function is a Boolean variable that is equal to 1 or 0. The control function for above register transfer can bes shown as:

P: R2 ← R1

The control condition is terminated with a colon. It shows that transfer operation can be executed only if P=1.

The following image shows the block diagram that depicts the transfer of data from R1 to R2.



Here, the letter 'n' indicates the number of bits for the register. The 'n' outputs of the register R1 are connected to the 'n' inputs of register R2.

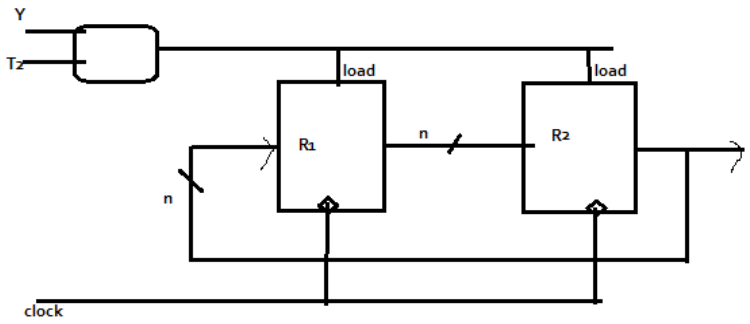
A load input is activated by the control variable 'P' which is transferred to the register R2.

Numericals based on Register transfer:

1.Show the block diagram of the hardware that implements the following register Transfer statement:

yT2: R2 ←R1, R1 ← R2

Sol:



2. Represent the following conditional control statement by two register transfer statements With control function.

If (P=1) then (R1←R2) else if (Q=1) then (R1←R3)

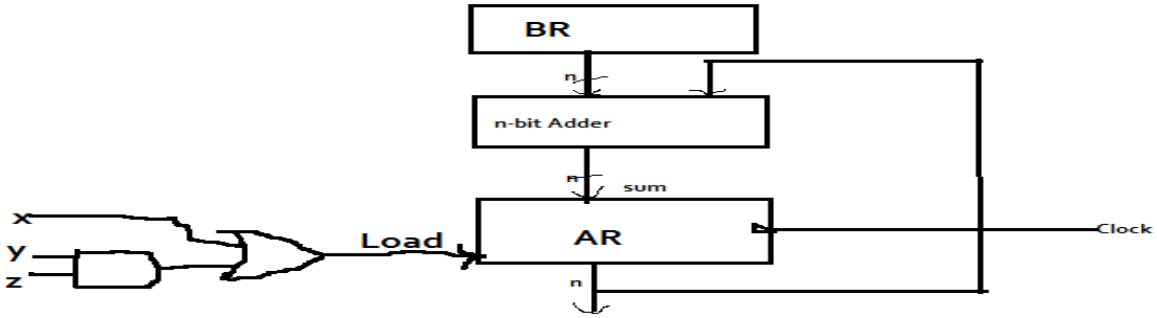
Sol: P: R1←R2
P'Q:R1←R3

Ques. Draw the block diagram for the hardware that implements the following statements

X+YZ:AR←AR+BR

Where AR and BR are two n-bit registers and x,y and z are control variables include the logic gates for the control function.

Sol:



Bus Transfers:

A digital system composed of many registers, and paths must be provided to

transfer information from one register to another. The number of wires connecting all of the registers will be excessive if separate lines are used between each register and all other registers in the system.

A bus consists of a set of common lines, one for each bit of register, through which binary information is transferred one at a time. Control signals determine which register is selected by the bus during a particular register transfer.

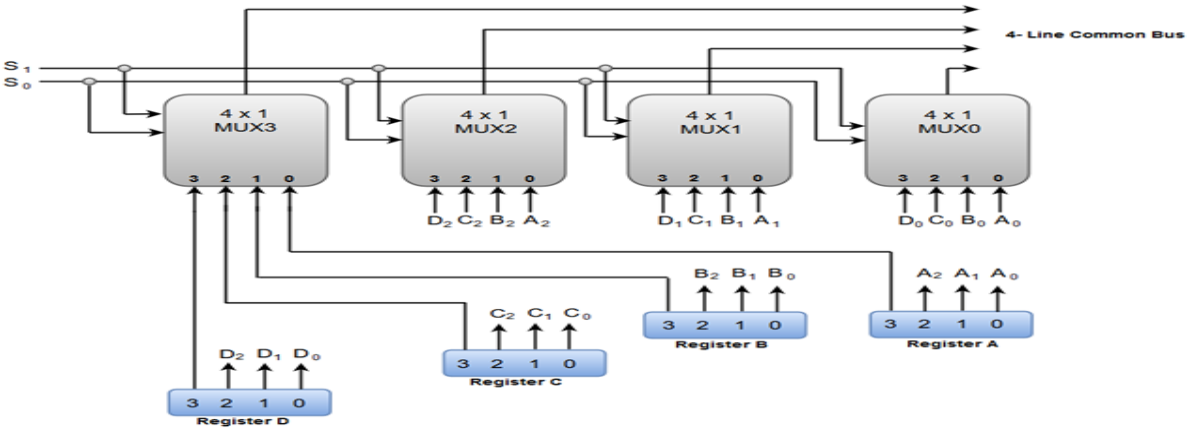
Note: The number of multiplexers needed to construct the bus is equal to the number of bits in each register. The size of each multiplexer must be No of register.

For example: a common bus for four registers of 4-bits so each requires 4 multiplexers, one for each line in the bus. Size of multiplexer=4*1

The following block diagram shows a Bus system for four registers. It is constructed with the help of four 4 * 1 Multiplexers each having four data inputs (0 through 3) and two selection inputs (S1 and S2).

For instance, output 1 of register A is connected to input 0 of MUX1 and so on.

Bus System for 4 Registers:



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.

The following function table shows the register that is selected by the bus for each of the four possible binary values of the Selection lines.

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

Numericals based on bus transfer:

Q:A digital computer has a common bus system for 16 registers of 32 bits each.
(i) How many selection input are there in each multiplexer?
(ii) What size of multiplexers is needed?
(iii) How many multiplexers are there in a bus?

Sol:

(i)How many selection input are there in each multiplexer?
2^n=No. of Registers; n=selection input of multiplexer
2^n=16; here n=4
Therefore 4 selection input lines should be there in each multiplexer.

(ii)What size of multiplexers is needed?
size of multiplexers= Total number of register X 1
= 16 X 1
Multiplexer of 16 x 1 size is needed to design the above defined common bus.

(iii)How many multiplexers are there in a bus? No. of multiplexers = bits of register
= 32
32 multiplexers are needed in a bus.

Memory Transfer

Most of the standard notations used for specifying operations on memory transfer are stated below.

- 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.
- A memory word is designated by the letter **M**.
- We must specify the address of memory word while writing the memory transfer operations.
- The **address register** is designated by **AR** and the **data register** by **DR**.

  Read Operation: read operation can be stated as:

  Read: DR ← M [AR]

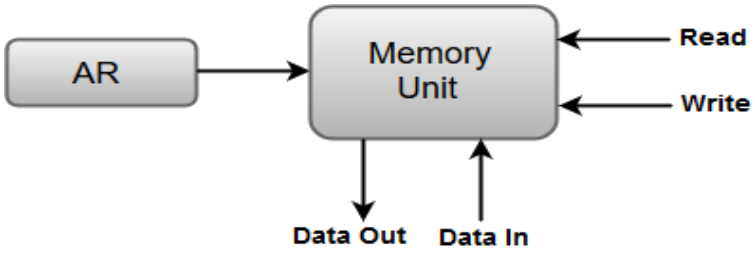
- The **Read** statement causes a transfer of information into the data register

(DR) from the memory word (M) selected by the address register (AR).

Write Operation: write operation can be stated as:

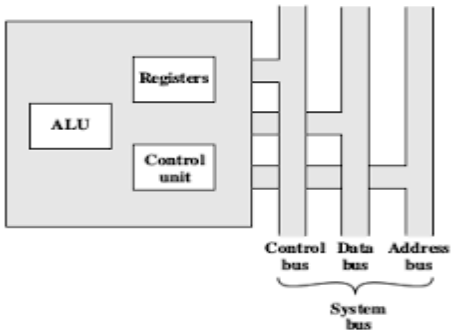
Write: M [AR] ← R1

- The Write statement causes a transfer of information from register R1 into the memory word (M) selected by address register (AR)

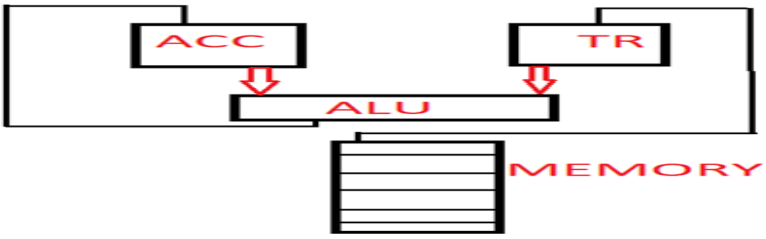


Processor Organization:

Processor is the brain of digital computer system. It contains ALU , control Unit and Registers .
There are three types of Processor Organization.
1.Accumulator Organization
2.General Register Organization
3.Stack Organization



1.Accumulator Organzation: The computers, present in the early days of computer history, had accumulator based CPUs. In this type of CPU organization, the accumulator register is used implicitly for processing all instructions of a program and store the results into the accumulator. The instruction format that is used by this CPU Organisation is **One address field**. Due to this the CPU is known as **One Address Machine**.



The main points about Single Accumulator based CPU Organisation are:

1. In this CPU Organization, the first ALU operand is always stored into the Accumulator and the second operand is present either in Registers or in the Memory.
2. Accumulator is the default address thus after data manipulation the results are stored into the accumulator.
3. One address instruction is used in this type of organization.

The format of instruction is: Opcode + Address

Opcode indicates the type of operation to be performed.
Mainly two types of operation are performed in single accumulator based CPU organization:

1. **Data transfer operation –**
   In this type of operation, the data is transferred from a source to a destination.

   For ex: LOAD X, STORE Y

   Here LOAD is memory read operation that is data is transfer from memory to accumulator and STORE is memory write operation that is data is transfer from accumulator to memory.

ALU operation –
In this type of operation, arithmetic operations are performed on the data.
For ex: MULT X

where X is the address of the operand. The MULT instruction in this example performs the operation,

AC <-- AC * M[X]

AC is the Accumulator and M[X] is the memory word located at location X.

Advantages –

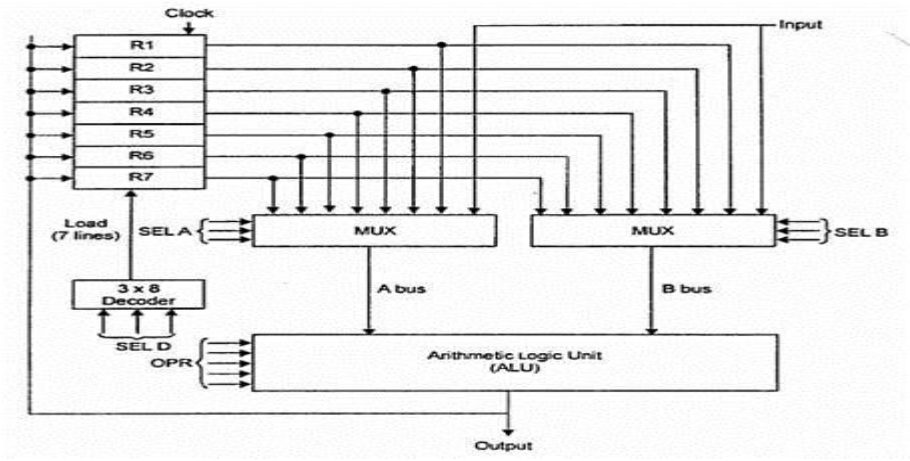
- One of the operands is always held by the accumulator register. This results in short instructions and less memory space.
- Instruction cycle takes less time because it saves time in instruction fetching from memory.

Disadvantages –

- When complex expressions are computed, program size increases due to the usage of many short instructions to execute it. Thus memory size increases.
- As the number of instructions increases for a program, the execution time increases.

2.General Register Organization:

When we are using multiple general purpose registers, instead of single accumulator register, in the CPU Organization then this type of organization is known as General register based CPU Organization.



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** : **10010** – to perform the arithmetic addition A+B
4. **SEL D : 011** – to place the result available on output bus in R3.

Operation selection code	Operation	symbol
0000	Transfer A	TSFA
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 A	SHL

What is CONTROL WORD?

- The combined value of a binary selection inputs specifies the control word.
- It consist of four fields SELA,SELB,and SELD or SELREG contains three bit each and SELOPR field contains four bits thus the total bits in the control word are 13-bits.

SEL A	SELB	SELREG OR SELD	SELOPR

FORMATE OF 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.

CONTROL WORD FOR OPERATION R2 = R1+R3

SEL A	SEL B	SEL D OR SELREG	SELOPR
001	011	010	0010

The advantages of General register based CPU organization –

- Efficiency of CPU increases as there are large number of registers are used in this organization.
- Less memory space is used to store the program since the instructions are written in compact way.

The disadvantages of General register based CPU organization –

- Care should be taken to avoid unnecessary usage of registers. Thus, compilers need to be more intelligent in this aspect.
- Since large number of registers are used, thus extra cost is required in this organization.

Numerical based on Register Organization:

Q1). A bus-organized CPU has 16 registers with 32 bits in each, and an ALU and a destination decoder.
(a). How many multiplexers are there in the A bus, and What is the size of each multiplexer?
(b). How many selection inputs are needed for MUX A and MUX B?
(c). How many inputs and outputs are there in the decoder?
(d).How many inputs and outputs are there in the ALU for data, including input and output carries ?
(e).Formulate a control word for the system assuming that the ALU has 35 operations.

Sol: given, Number of register =16 registers.
Number of bits in each register=32 bits

a. since, No. of Mux=No. of bits in each register
= 32 Mux
Size of Mux= No of register*1
=16*1
b. Since, size of Mux=16*1=2^4 *1= 4 selection inputs
c. Since, No of register=16
So to select 16 register we need 4-to-16 line decoder
So inputs of Decoder=4 and output of Decoder=16.
d. No of bits in each register=32

So, data input lines for ALU=32+32+1(here 32 from bus A and 32 from Bus B and one external input)

Data output lines ALU=32+1(32 is the result of the output of ALU and one output for external input)

e. Control word will be of

SELA=4bit, SELB=4bit , SELD=4 bit, OPR(Total ALU operation)=35=2^6 =6 bits

4bits 4bits 4bits 6bits =total=18bits

SELA	SELB	SELD	OPR

Control word

3.STACK ORGANIZATION

Stack is a storage structure that stores information in such a way that the last item stored is the first item retrieved. It is based on the principle of LIFO (Last-in-first-out). The stack in digital computers is a group of memory locations with a register that holds the address of top of element. This register that holds the address of top of element of the stack is called ***Stack Pointer***.

Stack Operations

The two operations of a stack are:

1. **Push:** Inserts an item on top of stack.
2. **Pop:** Deletes an item from top of stack.

Implementation of Stack

In digital computers, stack can be implemented in two ways:

1. Register Stack
2. Memory Stack

Register Stack

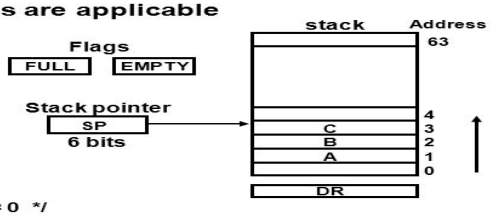
A stack can be organized as a collection of finite number of registers that are used to store temporary information during the execution of a program. The stack pointer (SP) is a register that holds the address of top of element of the stack.

The one bit register FULL is set to 1 when stack is FULL.
The one bit register EMPTY is set to 1 when stack is EMPTY.

❖REGISTER STACK ORGANIZATION

Stack
- Very useful feature for nested subroutines, nested interrupt services
- Also efficient for arithmetic expression evaluation
- Storage which can be accessed in LIFO
- Pointer: SP
- Only PUSH and POP operations are applicable

Register Stack



Push, Pop operations

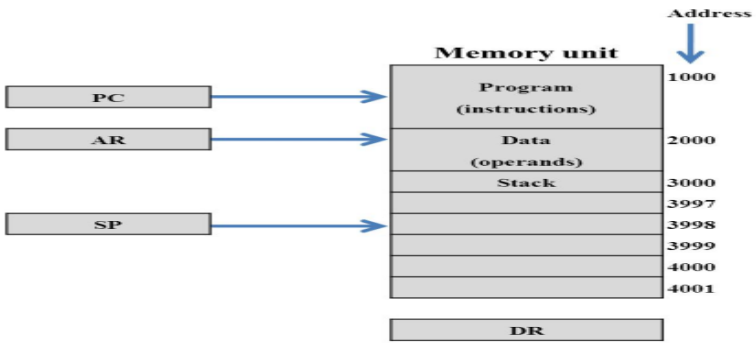
```
/* Initially, SP = 0, EMPTY = 1, FULL = 0 */
```

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

8

Memory Stack

A stack can be implemented in a random access memory (RAM) attached to a CPU. The implementation of a stack in the CPU is done by assigning a portion of memory to a stack operation and using a processor register as a stack pointer. The starting memory location of the stack is specified by the processor register as *stack pointer*.



MEMORY STACK

- Push Operation

$$SP \leftarrow SP - 1$$
$$M[SP] \leftarrow DR$$

The first item is stored at *address 4000*

- Pop Operation

$$DR \leftarrow M[SP]$$
$$SP \leftarrow SP + 1$$

RPN(Reverse Policy Notation)

Using a Stack

Reverse Polish Notation (RPN) is a mathematical notation in which every operator follows all of its operands.

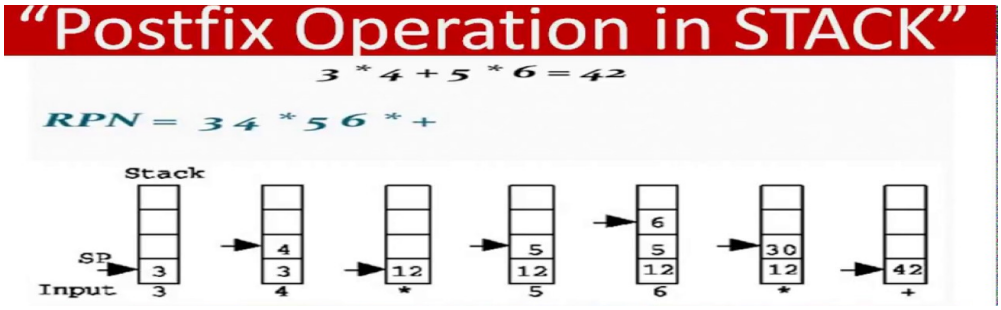
For example, the addition "3+4" is written in RPN as "3 4 +", and the expression "6+ ((3+4) ×5) −7" is written as "6 3 4 + 5 × + 7 −".

Write a program that evaluates very simple RPN expressions. Assume that the expressions consist of integers operands and the four basic arithmetic operations: {+, −, *, /}.

Infix	Reverse Polish notation
A + B × C	A B C × +
A × B + C	A B × C +
A × B + C × D	A B × C D × +
(A + B) / (C − D)	A B + C D − /
A × B / C	A B × C /
((A + B) × C + D)/(E + F + G)	A B + C × D + E F + G + /

Evaluation of RPN using Stack:

Stacks can be used to **evaluate** postfix notation equations (also known as Reverse Polish notation). So the algorithm moves along the expression, pushing each operand on the **stack** while operators cause two items to be popped off the **stack** , **evaluated** and the result pushed back on the **stacks** .



Important Numerical based on Stack organization:

Q1. Convert the following numerical arithmetic expression into reverse polish notation and show the stack operation for evaluating the numerical result.
(3+4)[10(2+6)+8]

Q2.let SP= 000000 in the stack. How many items are there in the stack if:
a. FULL = 1 and EMTY = 0?
b. FULL= 0 and EMTY = 1?

Sol:

a. FULL=1 and EMPTY=0
In this stack will be FULL with 64 items

b. FULL= 0 and EMTY = 1?
In this stack is Empty.

Q3. A stack is organized such that SP always points at the next empty location on the stack. This means that SP can be initialized to 4000 and the first item in the stack is stored in location 4000. List the microoperations for the PUSH and POP.

Sol:

MEMORY STACK

- Push Operation

$$SP \leftarrow SP - 1$$
$$M[SP] \leftarrow DR$$

The first item is stored at *address 4000*

- Pop Operation

$$DR \leftarrow M[SP]$$
$$SP \leftarrow SP + 1$$

Instruction Formats :

(Zero, One, Two and Three Address Instruction)

Computer perform task on the basis of instruction provided. An instruction in computer comprises of groups called fields. These field contains different information as for computers every thing is in 0 and 1 so each field has different significance on the basis of which a CPU decide what to perform. The most common fields are:

Instruction Format

- Operation field which specifies the operation to be performed like addition.
- Address field which contain the location of operand, i.e., register or memory location.
- Mode field which specifies how operand is to be founded.

An instruction is of various length depending upon the number of addresses it contain. Generally CPU organization are of three types on the basis of number of address fields:

1. Single Accumulator organization (one address instruction)
2. General register organization (Three or Two address instruction)
3. Stack organization (Zero address instruction)

1.Single Accumulator organization (one address instruction)

In first organization operation is done involving a special register called accumulator.

The main points about Single Accumulator based CPU Organisation are:

4. In this CPU Organization, the first ALU operand is always stored into the Accumulator and the second operand is present either in Registers or in the Memory.
5. Accumulator is the default address thus after data manipulation the results are stored into the accumulator.
6. One address instruction is used in this type of organization.

The format of instruction is: Opcode + Address

Opcode indicates the type of operation to be performed.
Mainly two types of operation are performed in single accumulator based CPU organization:

2. **Data transfer operation –**
   In this type of operation, the data is transferred from a source to a destination.

   For ex: LOAD X, STORE Y

   Here LOAD is memory read operation that is data is transfer from memory to accumulator and STORE is memory write operation that is data is transfer from accumulator to memory.

ALU operation –
In this type of operation, arithmetic operations are performed on the data.
For ex: MULT X

where X is the address of the operand. The MULT instruction in this example performs the operation,

AC <-- AC * M[X]

AC is the Accumulator and M[X] is the memory word located at location X.

2.General register organization (Three or Two address instruction)

In it multiple registers are used for the computation purpose.

Ex: ADD R1 R2 R3 R1←R2+R3// Here it is using Three addresses one address of R1 register , one address of R2 register , one address of R3 register .

It can be write with two register if we store result in one of the register.

EX: ADD R1 R2 ; R1←R1+R2;// here it is using only two addresses because result is stored in one of the two register.

Note: for moving data we use MOVE instruction in two address.

3. Stack organization :

It work on stack basis operation due to which it does not contain any address field.

We use PUSH , POP in stack.

Note: In the all above organization we use ADD, SUB, MUL, DIV.

It is assumed that the computer has processor General registers are R1, R2, R3, R4...
The symbol M [A] denotes the operand at memory address symbolized by A

Example: Write a program to evaluate the arithmetic statement:

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

(a).Using a general register computer with three address instructions.
(b).Using a general register computer with two address instructions.
©. Using an accumulator type computer with one address instructions.
(d).Using a stack organized computer with zero-address operation instructions

THREE-ADDRESS INSTRUCTIONS: X = (A+B)*(C+D)

ADD	R1, A, B	R1 ← M [A] + M [B]
ADD	R2, C, D	R2 ← M [C] + M [D]
MUL	X, R1, R2	M [X] ← R1 * R2

TWO-ADDRESS INSTRUCTIONS: X = (A+B)*(C+D)

MOV	R1, A	R1 ← M [A]
ADD	R1, B	R1 ← R1 + M [B]
MOV	R2, C	R2 ← M [C]
ADD	R2, D	R2 ← R2 + M [D]
MUL	R1, R2	R1 ← R1*R2
MOV	X, R1	M [X] ← R1

The MOV instruction moves or transfers the operands to and from memory and processor registers. The first
symbol listed in an instruction is assumed to be both a source and the destination where the result of the operation is
transferred.

ONE-ADDRESS INSTRUCTIONS
One-address instructions use an implied accumulator (AC) register for all data manipulation. AC contains
the result of all operations. The program to evaluate X = (A + B) * (C + D) is

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

All operations are done between the AC register and a memory operand. T is the address of a temporary memory
location required for storing the intermediate result.

ZERO-ADDRESS INSTRUCTIONS:

A stack-organized computer does not use an address field for the instructions ADD and MUL. The PUSH and POP
instructions, however, need an address field to specify the operand that communicates with the stack. The following
program shows how X = (A + B) * (C + D) will be written for a stack organized computer. (TOS stands for top of stack)

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

Ques. Write a program to evaluate the arithmetic statement:

X= A*[B+C*(D+E)]/F*(G+H)

(a).Using a general register computer with three address instructions.
(b).Using a general register computer with two address instructions.
©. Using an accumulator type computer with one address instructions.
(d).Using a stack organized computer with zero-address operation instructions

Sol: solve it according to above example.

Addressing Modes-

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

Types of Addressing Modes-

In computer architecture, there are following types of addressing modes-

1. Implied / Implicit Addressing Mode
2. Immediate Addressing Mode
3. Register Direct Addressing Mode
4. Register Indirect Addressing Mode
5. Direct Addressing Mode
6. Indirect Addressing Mode
7. Auto-Increment/ Auto-Decrement Addressing Mode
8. Relative Addressing Mode
9. Indexed Addressing Mode
10. Base Register Addressing Mode

1.Implied Addressing Mode-

- The definition of the instruction itself specify the operands implicitly.
- It is also called as **implicit addressing mode**.

Examples-

The instruction “Complement Accumulator” is an implied mode instruction.

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. Immediate Addressing Mode-

In this addressing mode,

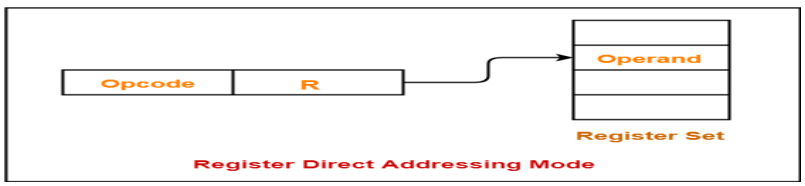
- The operand is specified in the instruction explicitly.
- Instead of address field, an operand field is present that contains the operand.

Examples-

MOV R #20 initializes register R to a constant value 20.

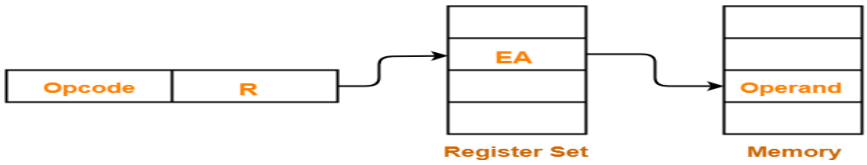
3. Register Direct Addressing Mode- In this addressing mode,

- 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.



4.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.



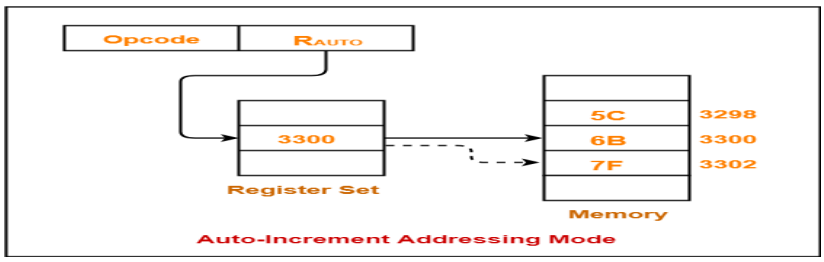
Register Indirect Addressing Mode

5.Auto-Increment/ Auto-Decrement Addressing Mode

- This addressing mode is a special case of Register Indirect Addressing Mode where-
- **Effective Address of the Operand**
- **= Content of Register**

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.
- Only one reference to memory is required to fetch the operand.



Auto-Increment Addressing Mode

Assume operand size = 2 bytes.

Here,

- 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.
- At memory address 3302, the next operand will be found.

NOTE-In auto-increment addressing mode,

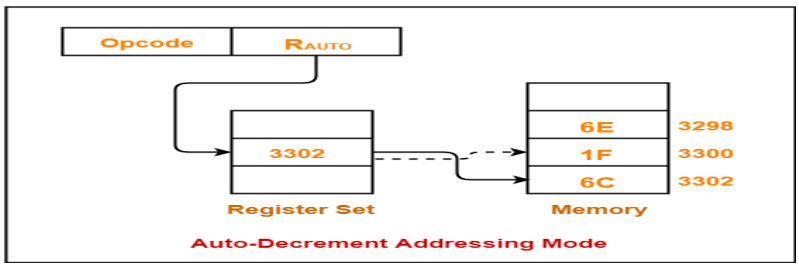
- First, the operand value is fetched.
- Then, the instruction register R_{AUTO} value is incremented by step size 'd'.

Auto-Decrement Addressing Mode-

Effective Address of the Operand= Content of Register – Step Size

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.



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.

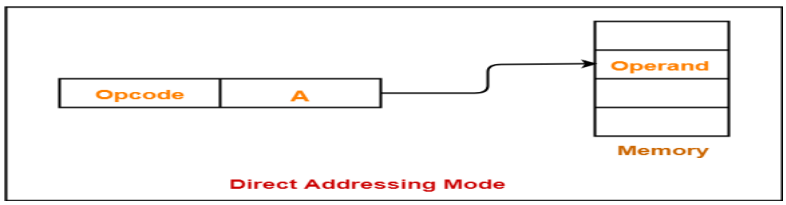
NOTE- In auto-decrement addressing mode,

- First, the instruction register R_{AUTO} value is decremented by step size ‘d’.
- Then, the operand value is fetched.

6. Direct Addressing Mode-

In this addressing mode,

- The address field of the instruction contains the effective address of the operand.
- Only one reference to memory is required to fetch the operand.
- It is also called as **absolute addressing mode**.



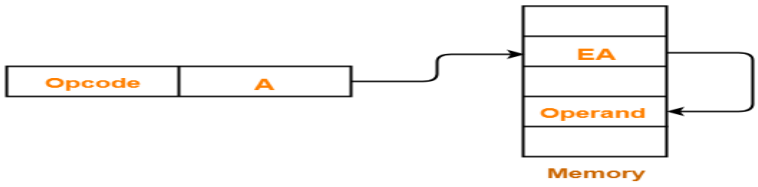
note-

- This addressing mode is similar to Register direct addressing mode.
- The only difference is address field of the instruction refers to a main memory instead of CPU Register.

7.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.



Indirect Addressing Mode

It is interesting to note-

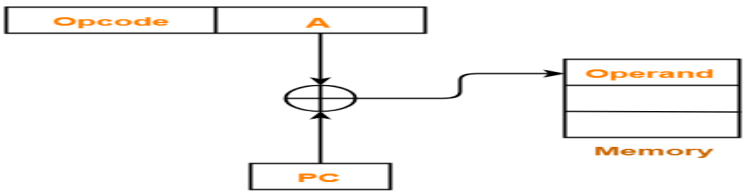
- This addressing mode is similar to Register indirect addressing mode.
- The only difference is address field of the instruction refers to Memory.

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



Relative Addressing Mode

NOTE-

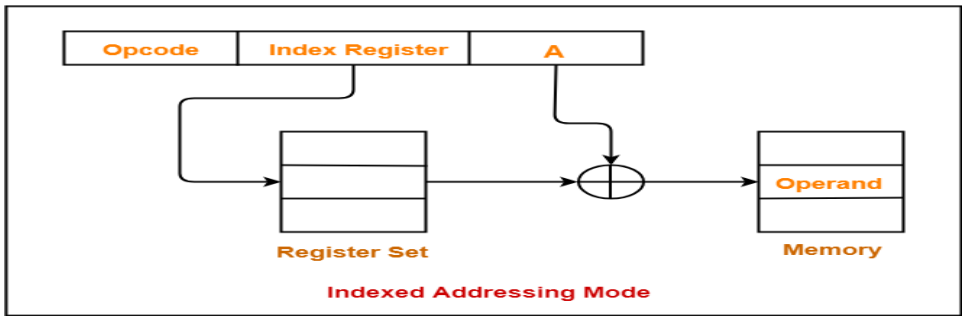
- **Program counter** (PC) always contains the address of the next instruction to be executed.
- After fetching the address of the instruction, the value of program counter immediately increases.
- The value increases irrespective of whether the fetched instruction has completely executed or not.

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.

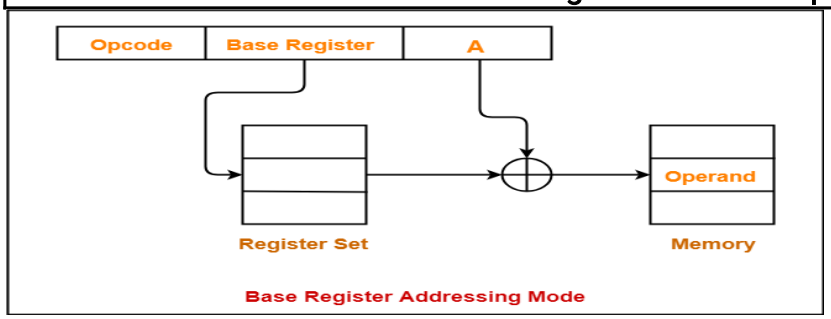
Effective Address= Content of Index Register + Address part of the instruction



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



Numerical based on Addressing Modes:

Q1.A Computer has 32-bits instructions and 12-bits addresses .If there are 250 two-address instruction, How many one – address instructions can be formulated?

Sol: instruction size=32-bits

Instruction address=12 bits

Total two-address instruction=250= 2^8 =8-bit will used to make opcode

So , two address instruction format will be

8-bit 12-bit 12-bit =32 bits

Opcode	Address1	Address2

But there are 250 two address instruction, remaining 2^8 =256 i.e (256-250=6 combination can be used for one address instruction)

So, Maximum number of one address instruction

Opcode	Address

6-bit 12-bit

Will be

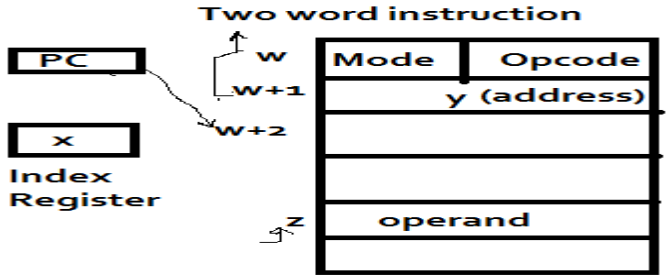
Opcode

$6*2^{12}$
So, maximum number of one address instruction=$6*2^{12}$=24576 Ans

(4). A two- word instruction is stored in memory at an address designated by the symbol W. The address Field of the instruction (stored at W+1) is designated by the symbol Y. The operand used during the Execution of the instruction is stored at an address symbolized by Z. An index register contains the Value X. State how Z is calculated from the other addresses if the addressing mode of the instruction Is

a. Direct
b. Indirect
c. Relative
d. Indexed

Sol: Here, z=Effective Address , below fig according to question



a. Direct addressing mode:

   Effective address(z)=y

b. Indirect Addressing mode:

   Effective address(z)= M[y]

c. Relative addressing mode:

   Effective address(z)=address part of instruction+content of program counter

   z =y+w+2

d. Index addressing mode:

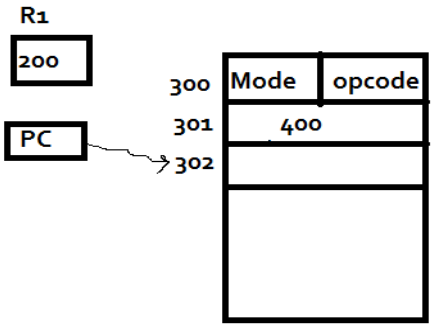
   Effective address(z)= address part of instruction + content of index register

   z = y+x

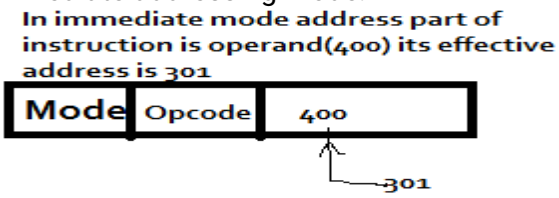
(6). An instruction is stored at location 300 with its address field at location 301. The address field has the value 400. A processor register R1 contains the number 200. Evaluate the effective address if the addressing mode of the instruction is
(a).direct (b) immediate (C) relative (d) register indirect (d) Index with R1 as the

index register.

Sol: single Instruction is stored in 300 and 301 addresses of memory. Address field of the instruction has value 400.



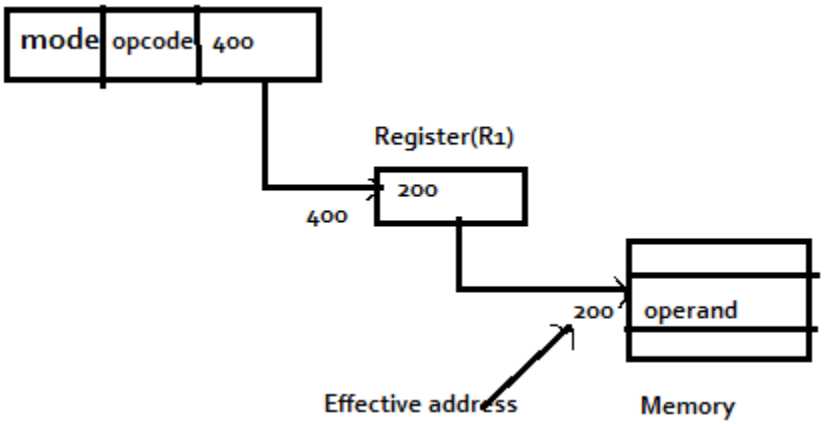
a. Direct addressing mode:
   Effective address: 400 Ans
b. Immediate addressing mode:



So effective address :301 Ans

c . Relative addressing mode:
Effective address= Address of instruction + content of PC
= 400+302
=702 Ans

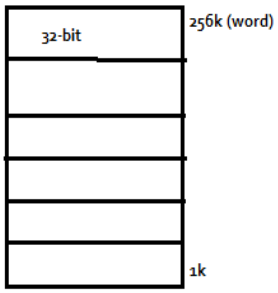
d. Register Indirect addressing mode: According to question



So Effective address=200 Ans

e . Index addressing mode:
Effective address= Address of instruction + content of Index Register
= 400+200
=600 Ans

Q.The memory unit of a computer has 256K words of 32 bits each. The computer has an Instruction Format with four fields: and operation code field, a mode field to specify one of seven addressing Modes, a register address field to specify one of 60 processor registers, and a memory address. Specify the instruction format and the number of bits in each field if the instruction is in one memory Word.
Sol: Memory size= 256k word of 32 bits each
$=2^8*2^{10}$(1k=1024byte $=2^{10}$)
$=2^{18}$
=>18 bits memory address



The computer has instruction format with four fields:
1.An operation code field(opcode field)
2.A mode field to specify one of seven addressing modes=> 7 addressing mode require 2^3=8 =>3-bit to represent the mode field.
3. Register address field to specify one of 60 processor registers=> 2^6=64 =>6-bit to represent the Register address field.
4. Memory address=18 bits (as above find from memory size)

For finding opcode field bits=>total instruction size=32 bits-(mode+register address field+memory address fild)
=32-(3+6+18)
=32-27=5-bits
Instruction format for above field will be as:

Opcode	Mode field	Register Address field	Memory Address field
5-bits	3-bits	6-bits	18-bits

=32- bits