# UNIT- BASIC STRUCTURE OF COMPUTERS

## 1. Computer types:

- Computer is a fast electronic calculating machine that accepts digitized input information, processes it and produces the resulting information.
- The list of instructions is called a Computer program and the internal storage is called Computer memory.
- Computer are differ in size, cost and computational power.
- Most common computer is PC. (Homes, offices, schools). in the form of desktop computer. (disk, CD-ROM)
- DC have processing units, storage units, visual display, audio o/p units, and keyboard.
- portable notebook computers (like thin briefcase).
- Workstations - with high resolution graphics input / output capability, more computational power than personal computer. (used in engineering applications
- Enterprise systems and servers (at the low end of range)
- Super Computers (at the high end)

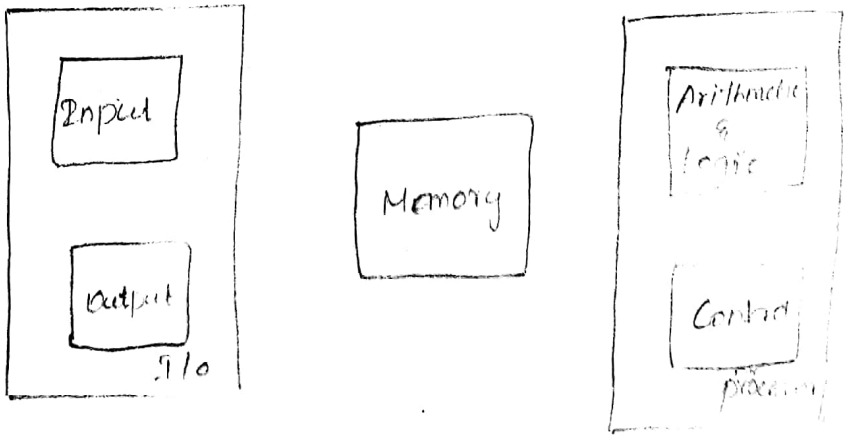
Enterprise Computer Systems :. or mainframes used for business data processing also provide more computational power and storage capacity than work station.

Servers :. Contains storage units also capable of handling large volume of requests. (used in business, education, personal user communities)

Super Computers :. used for large-scale numerical calculation (such as weather forecasting and aircraft design and simulation).

1.2 Functional units :.

- Functionally Independent main parts.
  - * Input
  - * memory
  - * Arithmetic and logic
  - * Output
  - * Control units



Input units:

→ accepts coded information from human operators from electromechanical devices such as keyboards, or from other computers over digital communication lines.

→ The infor/- either stored in memory for later use or immediately used by arithmetic and logic circuitary

→ The processing steps are determined by a prog stored in the memory.

→ finally the results are sent back to the output unit.

→ All of these actions are co-ordinated by Control unit

→ Input and output unit collectively referred to as I/o unit. (Input / output)

→ Instructions or machine instructions are explicit commands that,

• Govern the transfer of information within a computer as well as btw the computers and its I/o devic

• Specify the arithmetic and logic operations to performed.

Input unit:

- Computer accepts the coded infor + through input units; which read the data. (Keyboard)

- Corresponding letter automatically translated into binary code.

→ Input devices Joysticks, trackballs, and mouses. these are used as graphic input devices.

## Memory unit:

→ primary function is to store the data.

→ 2 classes

1. primary storage
2. secondary storage.

### Primary memory:

→ fast memory that operates at electronic speed.

→ prgms must be stored in the memory while they are being executed.

→ m/y contains large num of storage cells each capable of storing one bit of information.

→ The memory is organized.

→ Address is associated with each memory location.

→ The no. of bits in each word is reffered as word length of the computer. (word length range from 16 to 64 bits

→ The time required to access one word is called memory access time.

→ The small and fast RAM units are called as Cach RAM They are tightly coupled with the processor.

→ The largest and slowest unit is reffered to as main memory.

→ Secondary storage is used to store large amount of data and many programs.

**Arithmetic and Logic unit :.**

→ most of the operations are executed in ALU of the processor.

→ eg: Suppose two nos are located in the mly to be added. They are brought into the processor, then the addition is carried out by the ALU. Then the sum stored in mly or retained in the processor for immediate use.

→ when operands are brought into the processor they are stored in high speed storage element called registers.

→ Each register can store one word of data.

→ Access time to registers are faster compare with Cache units

**Output unit :.**

→ function is to send the o/p to the outside world.

→ eg: printer. (Capable of printing 10,000 lines per minute)

**Control unit :.**

- It sends control signals to other units and senses their states.

→ Data transfer blw the processor and memory are also controlled by control unit by through timing signals.

(Timing - determine when a given action is to take place)

## Operation of a Computer:

→ Computer accepts the data through an input unit and store in the memory.

→ Those info/- fetched into arithmetic and logic unit.

→ processed info/- leaves the computer through an output unit.

→ All the activities are ~~controlled~~ directed by the control unit.

## Basic operational Concepts:

→ To perform a given task, list of progra instructions are stored in the memory.

→ Individual instructions are brought from the mly into processor, which executes the operations.

→ Data to be used as operands and stored in the mly.

Add LOCA, R0

↓

This instr/- adds the operand at mly location Loc A and place the sum into Register R0.

Steps:

→ Instruction fetched from mly into the processor.

→ operand at LOCA is fetched and added to the content of R0.

→ finally sum is stored in register R0.

Load LocA, R1
Add R1, R0
} → This add inst/- combines memory access operation co/It ALU operation.

→ there inst/- transfer contents of m/y location LOC A into processor register R1, then second inst/- adds the content of registers R1 and R0 and places sum into R0.

→ After that the former content will be destroyed.

→ Transfer blw the memory and processor are started by sending the address of m/y location.

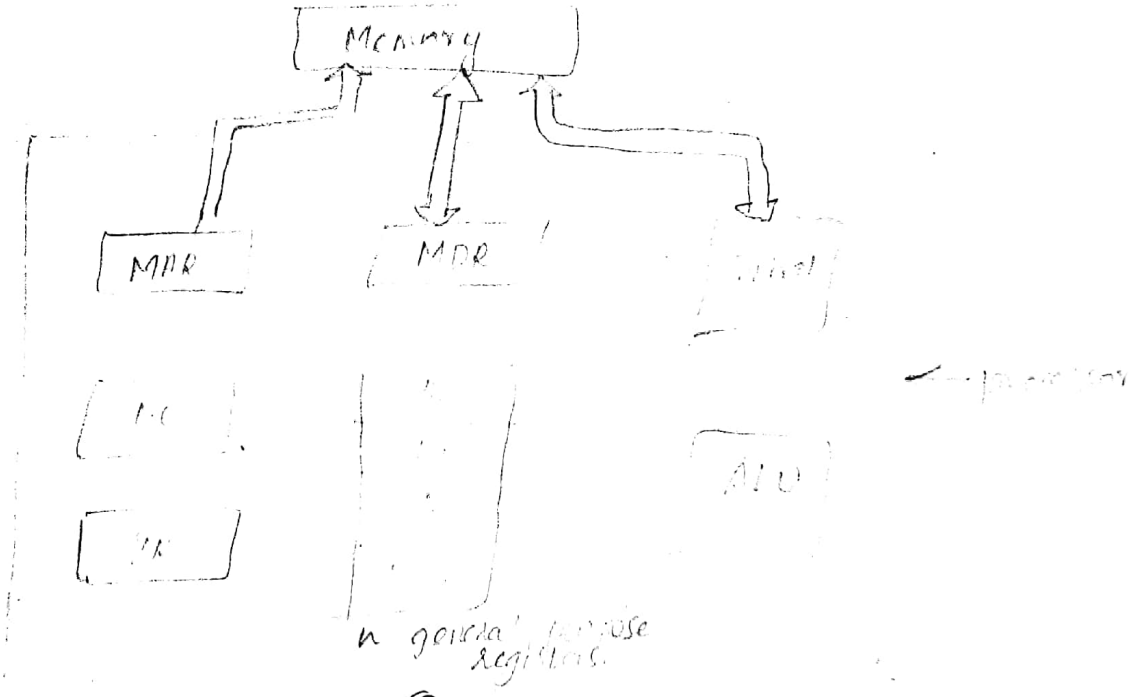



Fig ②.
Connections blw processor and memory.

↑ Instruction Register
→ IR holds the instructions. It's o/p available to the ctrl ckts which generates the timing signals.

→ PC - program Control - Keeps the track of the execution of a prg.
- Contains the memory address of the next instruction to be fetched and executed.

→ The MAR holds the address of the location to be accessed.

- The MDR contains the data to be written or read. Out of the address location.

- Execution of the pgm starts when PC set into first inst.

- The content of the PC transfered to MAR and Read control signal is sent to the memory.

- Next the content of MDR is transfered to IR. Then the instr is ready to be decoded and executed.

- when the operand has been read from MDR to memory, it is fetched transfered from the MDR to the ALU.

- By this way ALU perform the desired operation

- If the result of this operation stored in memory, then it is sent to the MDR.

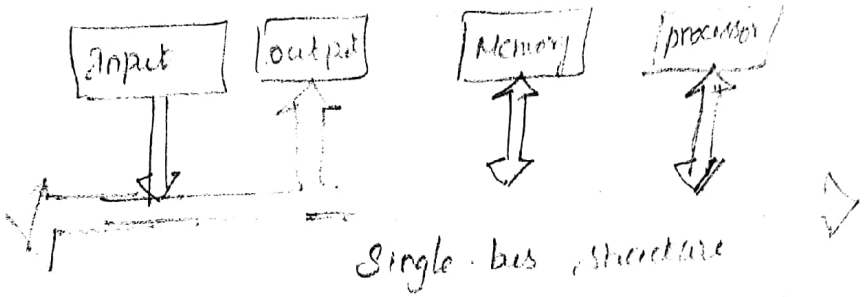
- Some time, during the execution of Current inst, PC contents are incremented. So that PC points to the next instruction to be executed.

Bus structures :

- TO achieve speed of operation, a Computer must be organized. So that all units can handle the data at a given time.

- when a word of data transferred btw units all bits are transferred in parallel, ie the bits are transferred simultaneously over many wires, or lines.

- A group of lines that serves as a connecting path for several devices is called Bus.




Single bus structure

- Single bus - only one transfer at a time. only two units can actively use the bus at given time.

adv: low cost, flexibility (for attaching peripheral devices).

- Multiple Bus - .

* Achieve more concurrency
* two or more transfer can possible at a time. But Cost ↑.

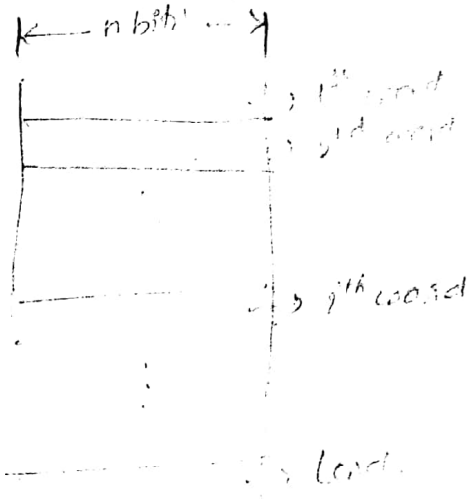
- A Buffer register should be included with the device to hold the information during transfers.

eg: processor to printer data transfer.

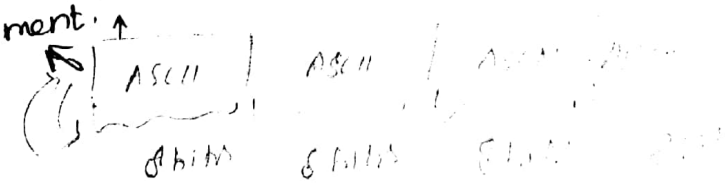
processor → data to → Buffer. Once the Buffer manager is loaded fully the printer automatically start the printing.

## Memory locations and addresses:

- Number, char, instructions are stored in mly.
- The mly consists of many millions of storage cells, each can store a bit of infor.




- The mly of a computer can be represented as a collection of words.
  → The term by addressable mly is used for this assignment.




→ If the word length of the computer is 32 bits a single word can store 32 2's complement or four ASCII characters. each occupying 8 bit.

- A unit of 8 bit is called byte.

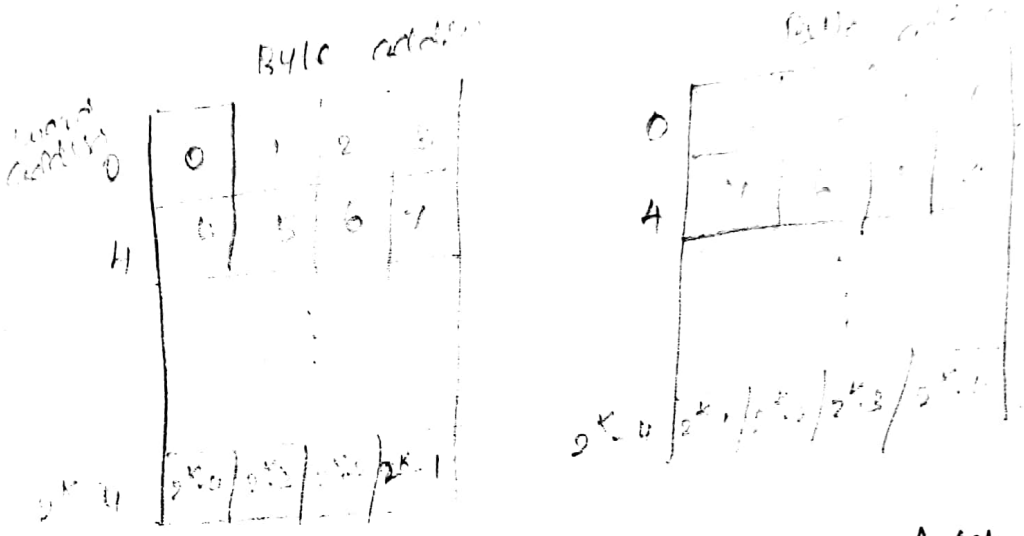
## * BYTE ADDRESSABILITY:

- Three basic Quantities : bit, byte and word.
- A byte is always 8 bits. But a word length is from 16 to 64 bits.
- So difficult to assign distinct addresses to individual bits locations in the mly.

- Byte locations have addresses 0, 1, 2... Thus if the word length of the machine is 32 bits, successive words are located at addresses 0, 4, 8... with each word consisting of 4 bytes.

*BIG - ENDIAN and LITTLE - ENDIAN assignments:

- 2 ways that byte addresses can be assigned across the words.



- Big-endian used when lower bytes addresses are used for the more significant bytes of the word.

- Little-endian used when lower bytes addresses are used for less significant bytes of the word.

- words more significant and less significant are used in relation to assign weights to bits

- Both big-endian and little-endian are used in commercial machines.

Memory operations:

Two basic operations are

1) Load - Load operation transfer a copy of the content of the specific memory location to the processor.

(ii) Store - store operation transfers an item of information the processor to a specific memory location.

Instruction and Instruction Sequencing :.

- A Computer must have instr/- capable of performing 4 types of operations.

1) Data transfer btw the memory and the processor Register.

2) Arithmetic and logic operations on data.

3) program sequencing and control.

4) I/o transfers.

Register transfer Notation :.

- The transfer of infor/- from one location in the computer to another.

- Names for the addresses of memory locations may be LOC, PLACE, A, VAR 2.

- processor register may be $R_0$, $R_5$

- I/o Registers names - DATAIN OUTSTATUS

The Content of a location are denoted by placing square brackets around the name of the location.

Thus, the operation $R_1 \leftarrow [LOC]$

- Consider the operation that adds the contents of register $R_1$ and $R_2$ and then places their sum into Register $R_3$. This action is indicated as

* $R_3 \leftarrow [R_1] + [R_2]$

This type of notation is known as Register transfer notation.

Assemble language Notation :.

The transfer from memory location LOC to processor register $R_1$ is specified by the statement

* MOV LOC, $R_1$

- The contents of the LOC are unchanged by the execution, But the old content of Register $R_1$ are overwritten.

- Adding two numbers contained in processor register $R_1$ and $R_2$ and placing their sum in $R_3$ can be specified by the assembly language statement

| Add $R_1, R_2, R_3$ |
|---|

BASIC Instruction Types :.

1. Three address Instruction
2. Two address Instruction
3. One address Instructions.

- which performs the operation $B \leftarrow [A] + [B]$ when the sum is calculated, the result is sent to the memory and stored in location B, replacing the original contents of this location. This means that operand B is both a source and a destination.

- A processor register, usually called the accumulator, may be used for the purpose of one-address instructions

**Add A**

- Adds the content of mly location A to the content of the accumulator register and place the sum back into the accumulator.

Load A → Copies the contents of mly location A into the accumulator

Store A → Copies the contents of the accumulator into the memory location.

eg: Arithmetic operations in processor.

C = A+B, this task can be performed by the instruction sequence

```
Move  A, Ri
Move  B, Rj
Add   Ri, Rj
Move  Rj, C
```

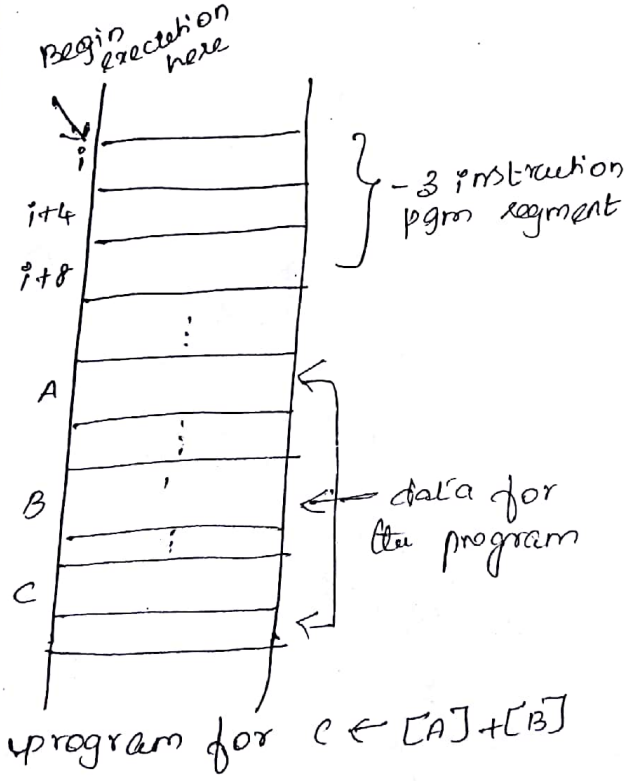
- In processor where one operand may be in the memory but the other must be in the register, the instruction sequence for the required task

```
Move  A, Ri
Add   B, Ri
Move  Ri, C
```

- Zero address instruction - Location of all operands are defined implicitly - structure called pushdown stack.

## Instruction Execution and Straight-Line Sequencing:

- Consider the instruction C ← [A] + [B]. We assume that the word length is 32 bits and the memory is addressable.




program for C ← [A] + [B]

- The three address instructions are in the pgm are in successive word locations, starting at location i. Since each instruction is 4 byte along the second & third instructions start at addresses i+4 and i+8.

- Let us consider how this program is executed. The processor contains a register called a program counter (PC), which holds the address of instruction to be executed next.

- To begin executing a program, the address of the first instruction must be placed into the PC.

- Then the processor control circuit use the information in the PC to fetch and execute instructions, one at a time, in order of increasing address. This is called straight line processing sequencing.

- During the execution of each each instruction, the PC is incremented by 4 to point the next instruction.

- Executing a given instruction in a two-phase procedure.

1. Instruction Set
2. Instruction Execute.

Instruction fetch:

- The instruction is fetched from the memory location whose address is in the PC.

- This instruction is placed in the instruction register (IR) in the processor.

Instruction execute :.

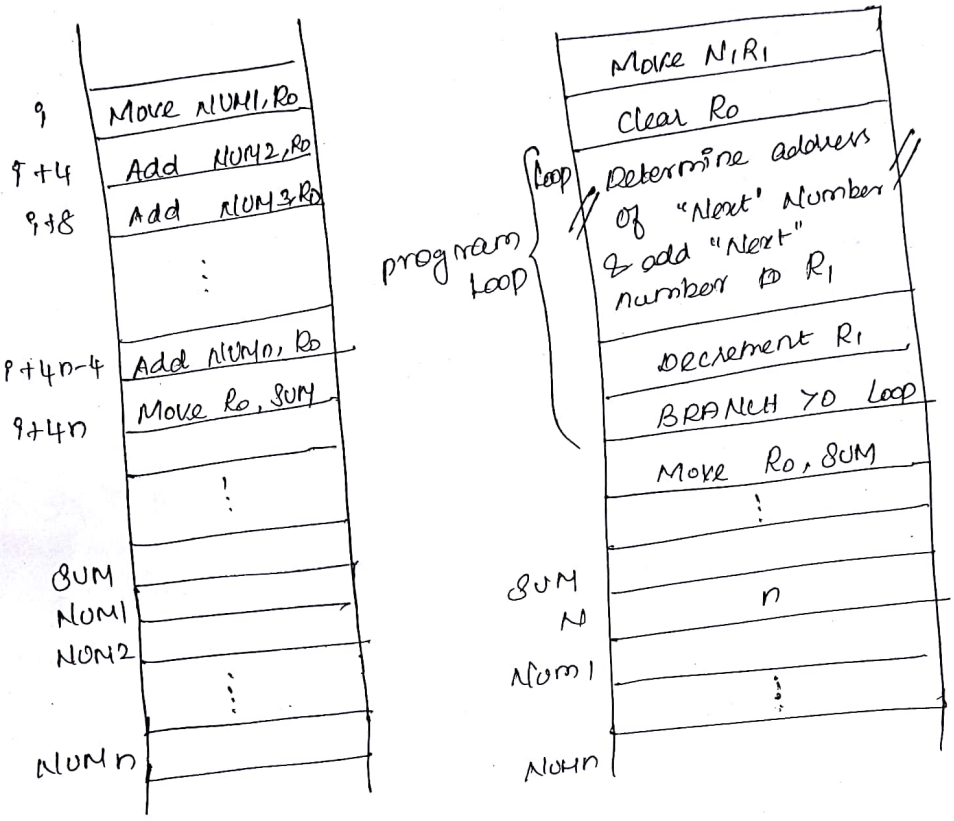
- The instruction in IR is examined to determine which operation is to be performed. The specified operation is then performed by the processor.

- This often involves fetching operands from the memory or from processor register, performing a arithmetic or logic operation and storing the result in the destination location.

BRANCHING :.

- Consider the task of adding a list of n numbers. The addresses of the memory location containing the n numbers are symbolically given as NUM 1, NUM 2, ... NUM n and a separate Add instruction is used to add each number to the contents of register R0. After all the numbers have been added the result is placed in memory location SUM.

- Instead of using a long list of Add instructions it is possible to place a single add instruction in a program loop. The loop is a straight line sequence of instructions executes as many times as needed.




- It starts at location loop and ends at the instruction Branch >0. Assume that the number of entries in the list n is stored in memory location N

- Register $R_1$ is used as a Counter to determine the number of times the loop is executed. Hence, the contents of location N are loaded into register $R_1$ at the beginning of the program.

Then, within the body of the loop, the instruction

| Decrement $R_1$ |
|---|

is reduces the contents of $R_1$ by 1 each time through the loop. Execution of loop is repeated as long as result of the decrement

operation is greater than zero.

- Branch instruction loads a new value into the program Counter. As a result, the processor fetches and executes the instructions at this new address, called branch target.

- A Conditional branch instruction causes a branch only if a specified Condition satisfied. If the Condition is not satisfied, the PC is incremented in the normal way and the next instruction in sequential address order is fetched and executed.

## Assembly language:-

- Machine instructions are represented by patterns of 0's and 1's. In assembly language normal words are replaced by acronyms called mnemonics. Such as Mov, ADD, INC, and BR.

- A Complete set of such symbolic names and rules for

- Programs written in a assembly language can be automatically translated into a sequence of machine instructions by a program called an assembler.

- The user program in its original alphanumeric text format is called a Source program and the assembled machine language program is called an object program.

- Assembler directives are used by the assembler while it translates a source program into an object program. If the assembler is to produce an object program according to know,

- ✓ How to interpret the names
- ✓ where to place the instructions in memory.
- ✓ where to place the data operands in the rely

EQU - Informs the assembler about the value of the label.

Ex: SUM EQU 200

ORIGIN - Specifies that the instructions of the object program to be loaded in the memory.

END - End of the source program text.

DATAWORD Commands and RESERVE Commands are used for reserves memory space.

CONDITION CODES:

- The processor keeps track of information about the result of various operations. for
- This is accomplished by encoding the required information in individual bits often called Condition Code flags
- These flags are usually grouped together in a special processor register called the Condition Code (or) Status register.

- Four commonly used flags are,

N (negative) → Set to 1 if the result is negative, otherwise cleared to 0.

Z (zero) → Set to 1 if the result is 0; otherwise cleared to 0.

V (overflow) → Set to 1 if arithmetic overflow occurs.

C (carry) → Set to 1 if a carry-out results from the operation.

## ADDRESSING MODES:

- The different ways in which the location of an operand is specified in an instruction are referred to as addressing modes.

- Various addressing modes are

1. Register mode
2. Absolute mode
3. Immediate mode
4. Indirect mode
5. Indexed mode
6. Relative mode.

- Additional addressing modes are,

(i) auto increment mode

(ii) Auto decrement mode.

Register mode → The operand is the content of a processor register; the name (address) of the register is given in the instruction.

Eg: MOV LOC, R2.

Absolute mode → The operand is in a memory location; The address of this given in the instruction.

eg: MOV LOC, R2. → uses two modes. processor registers are used as temporary storage location where the data are in a register are accessed using the register mode.

Immediate mode: - The operand is given implicitly in the instruction.

eg: Move 200, R0

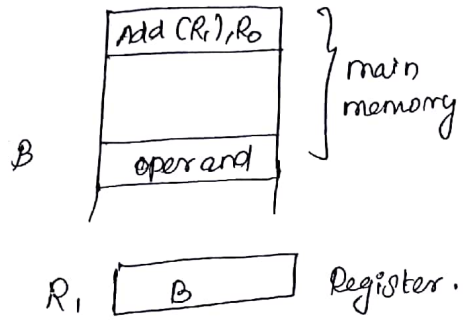
we can write the above inst. in the form

Move #200, R0.

Indirection and Pointers: -

- Here the instruction doesn't give the operand or its address explicitly.
- The effective address of the operand is the contents of a register or memory location whose address appears in the instruction.
- The register or memory location that contains the address of an operand is called a pointer.

eg:




- To execute the instruction, the processor uses the value B which is in register $R_1$ as the effective address of the operand.

Indexing and arrays:

- The effective address of the operand is generated by adding a constant value to the content of the register. Special register called Index register used for this purpose.

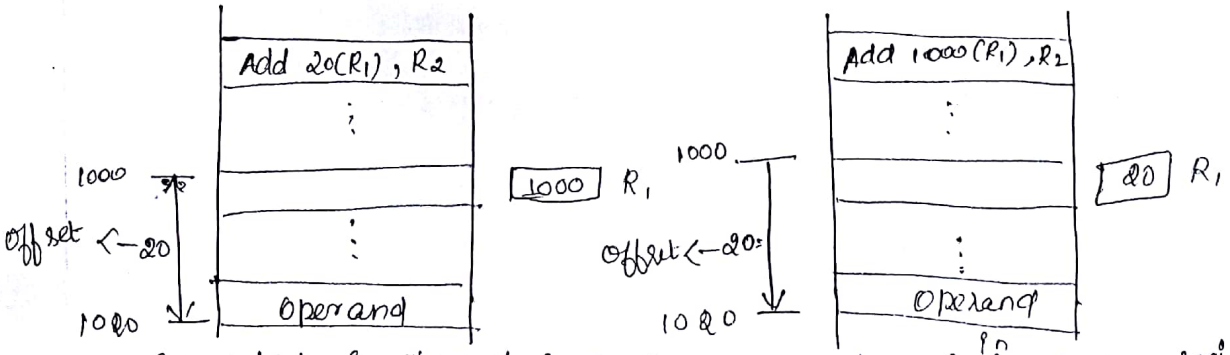
- we indicate the Index mode symbolically as,

$X(R_i)$, where $X \Rightarrow$ Constant value contained in the instruction and $R_i \Rightarrow$ Name of the register.

- Effective address of the operand is given by

$$EA = X + [R_i]$$

- Two ways of using Index mode are




(a) Offset is given as Constant (b) Offset is in the Index register.

Fig (a) The index register $R_1$ Contains the address of a memory location and the value of 'X' defines an offset from this address to the location where the operand is found.

Fig (b) The Constant X Corresponds to a memory address and the contents of the index register define the offset to the operand.

- In either case, the EA is the SUM of the two values, One is explicitly in the instruction and other is stored in a register.

Relative addressing:

- The effective address is determined by the index mode using the program Counter in place of the general-purpose register $R_i$.
- This mode Can be used to access data operands. But its most common use is to specify the target address in branch instructions.
- An instruction such as, Branch > 0 loop Causes program execution to go to the branch target location identified by the name loop if the branch condition is satisfied.

Autoincrement mode:

- The effective address of the operand is the contents of a register specified in the instruction.
- After accessing the operand, the content of this register are automatically incremented to point to the next item in a list.

Auto increment mode is written as,

$(R_i)^+$

Autodecrement mode:

- The contents of the a register specified in the instruction are first automatically decremented and then used as the effective address of the operand.

Auto decrement mode is written as;

$-(R_i)$

BASIC INPUT/OUTPUT Operations:-

- Consider a task that reads in character input from a keyboard and produces character output on a display screen. A simple way of performing such I/O tasks is to use a method know as program-Controlled I/O.
- The rate of the data transfer from the keyboard to a processor is limited by the typing speed of the user. [Few char/sec]

- The rate of output transfers from the processor to the display is much higher. It is determined by the rate at which characters can be transmitted over the link between the computer and the display device. [Thousand char/sec].

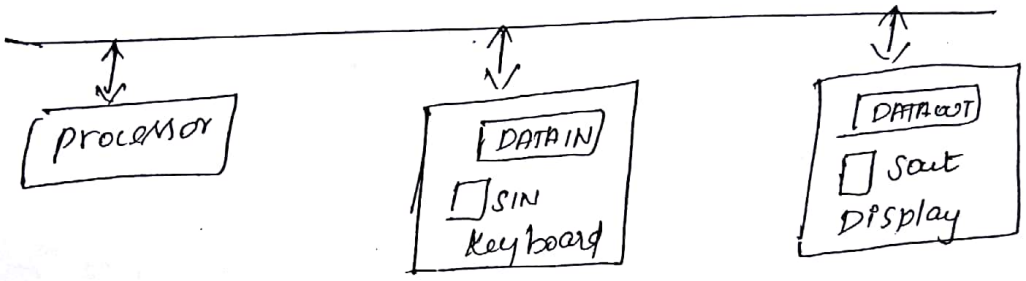



Fig: Bus connection for processor, keyboard and display.

- Consider the problem of moving a character code from the keyboard to the processor. Striking a key stores the corresponding character code in an 8-bit buffer register associated with the keyboard. This register is called DATAIN

- To inform the processor that a valid character is in DATAIN a status control flag SIN is set to 1. A program monitors SIN and when SIN is set to 1, the processor reads the contents of DATAIN.

- When characters are transferred from the processor to the display, a buffer register DATAOUT and a status control flag SOUT are used for this transfer. When SOUT equals to 1, the display is ready to receive a character.

– The buffer registers DATAIN and DATAOUT and the status flags SIN and SOUT are part of Circuitry Commonly known as device interface.

– In Order to perform I/O transfers, we need machine instructions that can check the state of the status flag and transfer data between the processor and the I/O devices.

Status flag is used to reflect the result of the operation.

For eg: The processor can monitor the keyboard status flag SIN and transfer a character from DATAIN to register $R_1$ by the following sequence of operations

```
READWAIT   Branch to READWAIT if SIN = 0
           INPUT from DATAIN to R1
```

– The following operations is used for transferring output to the display.

```
WRITEWAIT  Branch to WRITEWAIT if SOUT=0
           Output from R1 to DATAOUT.
```

– The branch operation is normally implemented by two machine instructions. The first instruction tests the status flag and the second performs the branch operation.

## Memory - Mapped I/O :-

- The addresses issued by the processor to access instructions and Operands always refer to memory locations.
- Many Computers use an arrangement called memory - mapped I/O in which some memory address values are used to refer to peripheral device buffer registers such as DATAIN and DATAOUT.

For eg: The Content of the keyboard character buffer DATAIN can be transferred to register $R_1$ in the processor by the instruction.

```
Move byte  DATAIN, R1
```

Similarly, the contents of register $R_1$ can be transferred to DATAOUT by the instruction,

```
Move byte  R1, DATAOUT
```

- The read operation is implemented by the machine instruction sequence

```
READWAIT   Test bit  #3, INSTATUS
           Branch=0  READWAIT
           Movebyte  DATAIN, R1
```

- The write operation may be implemented as,

```
WRITEWAIT  Test bit  #3, OUTSTATUS
           Branch=0  WRITEWAIT
           Move BYTE R1, DATAOUT.
```

- The TestBit instruction tests the state of one bit in the destination location, where the bit position to be tested is indicated by the first Operand.
- If the bit tested is equal to 0, then the condition of the branch instruction is true, and a branch is made to the beginning of the wait loop.

- when the device is ready (i.e) when the bit tested becomes equal to 1, the data are read from the input buffer or written into the output buffer.

## Multiprocessor & Multicomputers:

- Large Computer systems may contain a number of processor unit, in which case they are called Multiprocessor systems.

- These system either execute a number of different application task in parallel or they execute subtasks of a single large task in parallel.

- All processors usually have access to all of the memory in such systems and the term ~~shared~~ shared-memory multiprocessor system.

- The high performance of these systems comes with much increased complexity and cost.

- In Contrast to multiprocessor systems, it is also possible to use an interConnected group of Complete Computers (multiComputer) to achieve high total Computational Power.

- The Computer normally have access only to their own memory units.

- when the tasks they are executing need to communicate data, they do by exchanging messages over a communication link. this property seperate them from shared-memory multiprocessor. also called message passing multicomputers.

Performance :

- Most important measure of a computer is how quickly it can execute programs.

- The speed with which a computer executes program is executed by the design of its hardware & its machine language instructions.

- The total time required to execute the program is called elapsed time - is a measure of the performace of the entire computer system.

Performance Consideration - Active processor in certain period. (ii) the sum of these periods as the processor time needed to be execute the program.

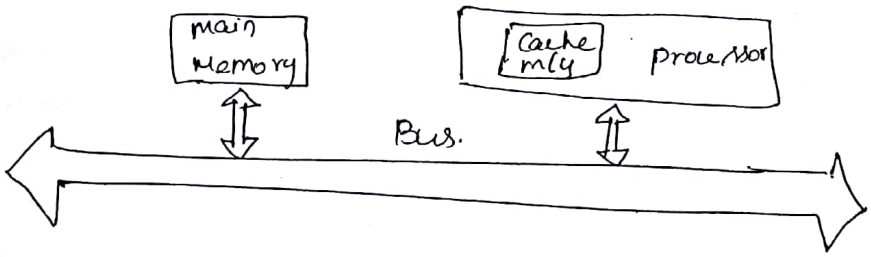
Elapsed time - depends on all units in a computer system.

Processor time - depends on hardware involved in the execution of individual machine instructions.

## Processor Cache :-

- As execution proceeds, instructions are fetched one by one over the bus into the processor and copy is placed in the Cache.

- when the execution of an instruction calls for any data located in the main memory, the data are fetched and a copy is placed in the Cache. Later if same instruction needed a second time, it is read directly from the Cache.




## Processor Clock :-

- Processor circuits are controlled by a timing signal called a clock. The clock defines regular time intervals called clock cycles.

- To execute a machine instruction, the Processor divides the action to be performed into a sequence of basic steps, such that each step can be completed in one clock cycle.

- The term cycle / second is called hertz (Hz)

## Basic performance equation :-

$$T = \frac{N \times S}{R}$$

T - The processor time required to execute a prg

N - Actual no. of machine instruction execution.

S - Average no. of basic steps needed to execute one machine instructions.

R - Clock rate (cycles per second)

## Performance Measurement :-

- The performance measure is the time it take a compute to execute a given benchmark.

- System performance Evaluation Corporation (SPEC) rating is computed as follows,

$$\text{SPEC rating} = \frac{\text{Running time on the reference computer}}{\text{Running time on the computer under test}}$$

- Over all SPEC rating for the Computer is given by

$$\text{SPEC rating} = \left(\prod_{i=1}^{n} \text{SPEC}_i\right)^{1/n}$$

- where n is the number of pgms in the suite.