COMPUTER ORGANIZATION & ARCHITECTURE

NOTES

SERIES -1

LinkedIn ATUL KUMAR
7777888822

Ch-1 Basic Structures of Computers

A Computer Architecture vs Organization:

Computer Architecture: It refers to the relationship between different hardware components of a computer system. It refers to those attributes of a system visible to a programmer i.e. the attributes that have a direct impact on the logical execution of a program. Examples of architecture attributes include the instruction set, I/O mechanisms

Computer organisation: It refers to the operational units and their interconnection that realize the architecture specification. It deals with how operational attributes are linked together to realize the architecture specifications.
Examples of organization attributes include control signals, interface between computer and peripherals and the memory technology used

Computer Architecture	Computer Organisation
1. It is the abstract model of a computer and is designed based on programmer's view.	1. A computer's organisation is the effective realization of the designed architecture.
2. It refers to the interrelationship between different hardware units of computer system.	2. It refers to the operational units and their interconnection that realize architecture specification
3. It deals with the attributes that have a direct impact on the logical execution of a program.	3. It deals with how operational attributes are linked together to realize the architecture specification
4. The architecture deals with what the computer does.	4. The organisation deals with how the computer does
5. It decides whether there should be a multiply instruction	5. It decides how to calculate the multiply instruction.
6. The architecture is predecided	6. The organization of a computer can be changed without modifying its architecture.

in Atul Kumar
7777888822

B Characteristics of Computer:

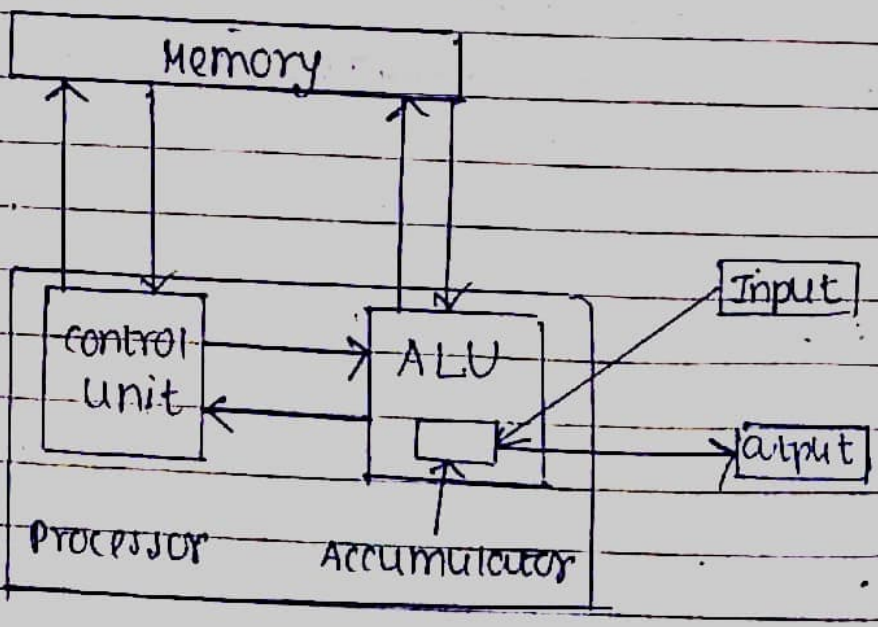
The most fundamental characteristics of computer are

(a) high accuracy
(b) high speed
(c) versatility
(d) diligence
(e) storage capability
(f) efficient storage and retrieval
(g) multitasking
(h) reliability

C Von-Neumann Architecture:

The Von-Neumann architecture is a sequential processor which act as a stored-program digital computer. This uses a processing unit and a single separate storage structure to hold both instructions and data. It is named after mathematician and early computer scientist John Von Neumann.

Von Neumann Architecture consists of a CPU, memory and I/O devices as shown in this figure. The program is stored ~~in time and~~ in memory. The CPU fetches an instruction from the memory at a time and executes it. One shared memory for instructions and data with one data bus and one address bus between processor and memory constitute Von Neumann Architecture. Its design is simple. It is mostly used to interface to external memory.



Atul Kumar
7777888822

Page No.
Date

Neumann machines are called control flow computer because instruction are executed sequentially as controlled by a program counter register. Even in parallel computers, the basic building blocks are Neumann processors.

0 Performance:

The performance of a computer deals with the speed of execution of a program. Systems that execute programs in less time are said to have higher performance.
Hence performance is directly related to speed of the system.

Various parameters to determine performance of a computer system are as follows:

- **Response time:** The time taken between the start and completion of task is called response time or execution time.

- **Bandwidth:** The amount of work done in a given time is the throughput or bandwidth.

- **Relative performance:** If x and y are two computers then relative performance is expressed as

$$\frac{\text{Performance}_x}{\text{Performance}_y} = \frac{\text{Execution time}_y}{\text{Execution time}_x} = n$$

For eg: if a computer A runs a program in 20 seconds and computer B runs the same program in 30 seconds, How much faster is A than B.

Sol:
$$\frac{\text{Performance}_A}{\text{Performance}_B} = \frac{\text{Execution}_B}{\text{Execution}_A} = \frac{30}{20} = 1.5$$

Atul Kumar
7777888822

Date

- **Processor clock:** Processor circuits are controlled by a timing signal called a clock. The clock defines the regular time intervals, called clock cycles. A clock cycle is a single electronics pulse of a CPU.

→ The length of one cycle is denoted by P.

→ The clock rate is inverse of the length of one clock cycle. i.e. $R = 1/P$, where R is clock rate which is measured in cycles per second or Hz.

→ The length P of one clock cycle is an important parameter that affects the processor performance.

→ There are two possibilities of increasing clock rate, R.

(a) First improving IC technology makes logic circuit faster which reduces time needed to complete a basic step and hence clock rate R increases.

(b) Secondly reducing amount of processing done in one basic step also makes it possible to reduce clock period P.

- **Basic Performance Equation:**

The basic performance equation is given by

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

T → processor time needed to execute a program i.e. performance parameter or program execution time

N → number of machine language instructions needed to complete execution of a program.

S → average number of basic steps required to execute one machine instruction.

R → clock rate of processor in cycles per second

$R = \frac{1}{P}$ where P → length of one clock cycle.

Atul Kumar
7777888822

- **Speed up:** It is the ratio of the execution time before and after a change is made.

$$\text{Speed up} = \frac{(\text{Execution time}) \text{ before}}{(\text{Execution time}) \text{ after}}$$

E **Factors that affect Performance:**

The performance with which a computer executes a program is affected by following criteria:

- **Pipeline and superscalar operation:**

→ A substantial improvement in the performance can be achieved by overlapping the execution of successive instructions, using a technique called as pipelining. Pipelining increases the rate of executing instructions significantly

Let S is total number of basic steps or clock cycles required to execute an instruction. The pipelining causes the effective value of S close to 1.

→ A high degree of concurrency can be achieved if multiple instruction pipelines are implemented in the processor. This means that multiple functional units are used creating parallel paths through which different instructions can be executed in parallel.

With such an arrangement it becomes possible to start the execution of several instructions in every clock cycle. This mode of operation is called superscalar execution.

With help of pipelining and superscalar execution we are reducing the value of S. In order to reduce value of S (< 1) the following points are worth noting:

(a) required to increase clock rate that are entirely caused by improvements in IC technology.

(b) improves the IC technology to make the circuits faster.

(c) reduces amount of processing done in one basic step.

Atul Kumar
7777888822

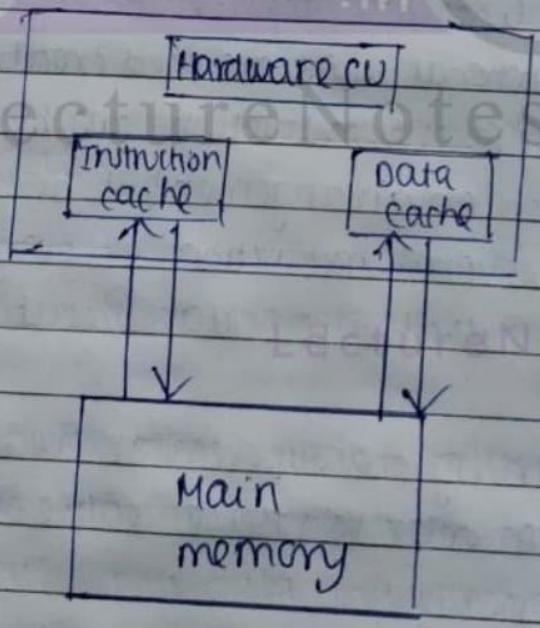
Scanned by CamScanner

- **Instruction set Architecture:**

→ The architectural design of CPU is based on instruction set. An instruction consists of opcode and one or more operands. Opcode specifies the operation to be performed over operands.
For eg: ADD A,B where ADD is the opcode and A,B are the operands.

→ Simple instruction needs small number of steps to execute whereas complex instruction requires a larger number of steps. For a processor that has only simple instructions a large number of instructions may be required to perform a given problem. This could lead to a large value for N and small value of S whereas just the opposite situation for a processor with only complex instructions.

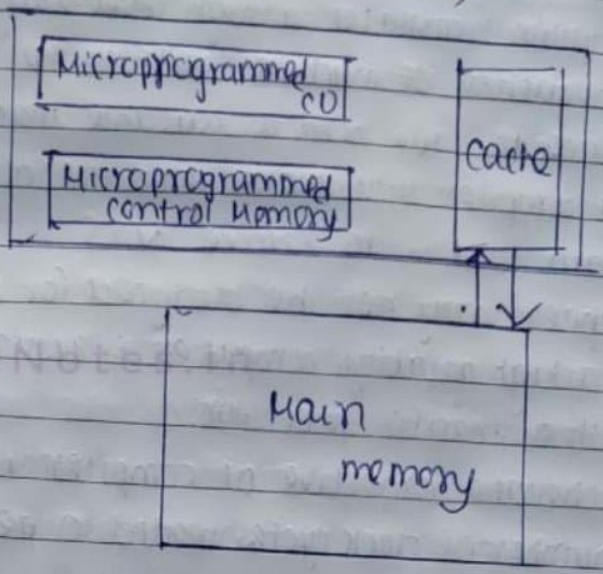
→ There are two instruction set design approaches such as
(a) RISC (Reduced Instruction Set Computer)



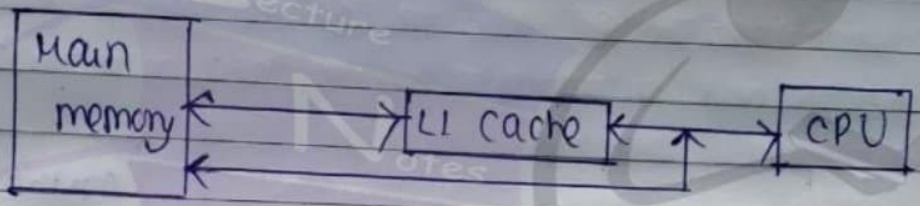
Atul Kumar
7777888822

Scanned by CamScanner

(b) CISC (Complex Instruction Set Computers)



- **Cache memory:**



It is an intermediate speed memory that lies between processor and main memory. At the start of execution all program instructions and required data are stored in the memory. As the ~~instruction~~ execution proceeds instructions are fetched one over the bus into the processor and copy is stored in the cache.

Later if the instruction ~~call~~ is needed a second time, it is read directly from the cache memory. The processor and a relatively small cache memory can be fabricated on a single integrated circuit chip as shown in the above diagram.

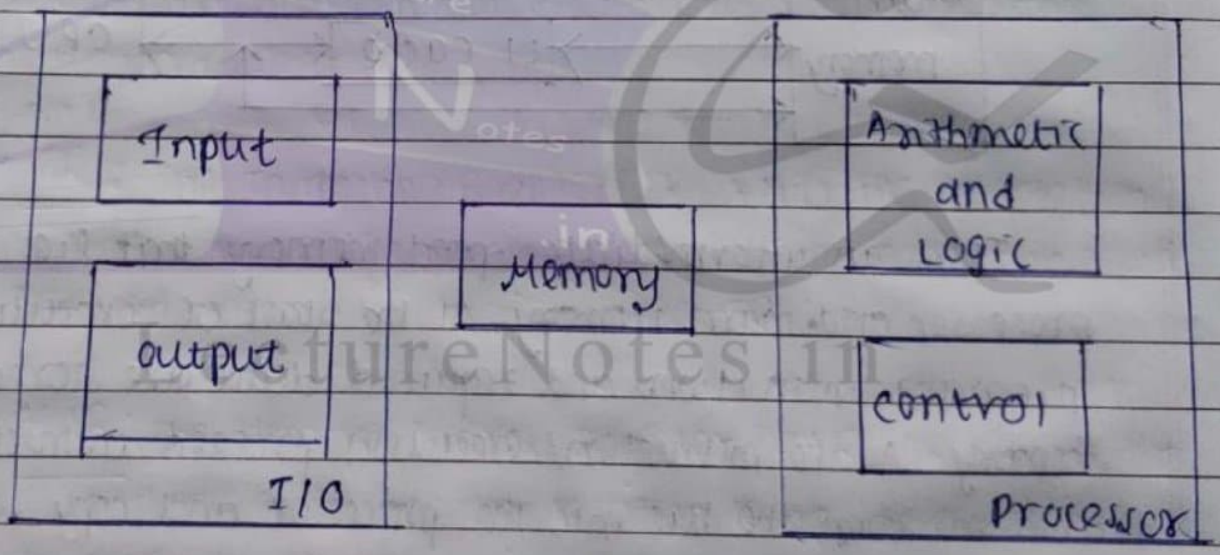
Thus overall speed is improved.

Atul Kumar
7777888822

Scanned by CamScanner

- **Compiler:**
  → A compiler translates a high level language program into a sequence of machine instructions
  → To reduce V, we need a suitable machine instruction set and a compiler that makes good use of it.
  → Its main goal is to reduce N X S
  → A compiler may not be designed for a specific processor however a high quality computer is usually designed for, and with a specific processor.
  → The ultimate objective of compiler is ~~used~~ to reduce the total number of clock cycles needed to perform a required programming task.

F Basic Functional units of a computer:



- **Input device:** The device which allows entering data and programs into the computer is called input unit. It takes inputs raw data and performs some processing giving out processed data.
  Eg: keyboard, mouse, scanner, OMR, OCR, MICR

Atul Kumar 7777888822

Scanned by CamScanner

- **Output device:** This is the process of producing results from the data for getting useful information. Again the output is also stored inside the computer for further processing. The device which produces output from computer is called output device.
  Eg: monitor, printer, plotter, speaker, etc.

- **CPU:** CPU acts as the heart of the computer system. It takes all major decisions, makes all sorts of calculations and directs different parts of the computer functions by activating and controlling operations.
  The CPU consists of ALU, CU and Registers

(i) **ALU:** The task of performing operations and logical operations is called processing. The actual processing of the data and instruction are performed by Arithmetic Logical Unit (ALU).
→ The major operations performed by ALU are arithmetic operations such as addition, subtraction, multiplication, division, increment, decrement and logical operations such as AND, OR, NOT, comparision, etc.
→ Data is transferred to ALU from storage unit and after processing the output is returned back to storage unit for further processing.

(ii) **CU:** → Control unit performs controlling of all system operations like input, processing and output.
→ It takes generates various control signals and timing signals.
→ Control Unit is responsible for co-ordinating various operations using time signal.
→ Control Unit determines the sequence in which computer programs and instructions are executed.
→ Control Unit thereby co ordinates the activities of computer's peripheral equipments as they perform the input and the output.

Atul Kumar 7777888822

Scanned by CamScanner

(iii) Registers: Register is a group of flipflops which can store data and instructions. The processor consists of a group of registers for its internal operations. These registers are used to hold the instructions, data, and memory address and helps for manipulating data in ALU.

→ AC: The Accumulator (AC) register is a general purpose processing register. The AC holds one of the operands and also result of the operation. One of the operands is kept in the accumulator and the other operand is kept in the temporary register.

→ PC: The Program Counter (PC) contains the memory address of the next instruction to be fetched and executed. The PC also has 12 bits and it holds address of the next instruction to be read from memory after current instruction is executed.

→ IR: The Instruction Register (IR) holds the instruction that is currently being executed. The instruction read from the memory is stored in the IR.

→ MAR: The Memory Address Register (MAR/AR) holds the address of the memory location to be accessed. The MAR has 12 bits since this is the width of the a memory address. This memory address holds the memory addresses of data and instructions.

→ MDR: The ~~Mem~~ Memory Data Register (MDR/DR) contains the data to be written into or read out of the memory address. The MDR holds the operand read from memory whose address is specified by MAR. It acts like a buffer.

Atul Kumar 7777888822

Scanned by CamScanner

→ TR: The Temporary Register (TR) is used for holding temporary data during processing.

→ I/OAR and I/OBR: An I/O Address Register (I/OAR) specifies the address of a particular I/O device.
An I/O Buffer Register (I/OBR) is used for the exchange of data between I/O module and the CPU.

- INPR: The input register (INPR) receives an 8 bit character from an input device.
- OUTR: The output register (OUTR) holds an 8 bit character from an output device.

Basic Operational Concepts

Activity in a computer is governed by a set of instructions called program. To perform a task, an appropriate program consisting of a list of instructions is stored in the memory. The processing required to execute a single instruction is called an instruction cycle. A basic instruction cycle consists of simplified two steps such as fetch cycle (FC) and execute cycle (EC)

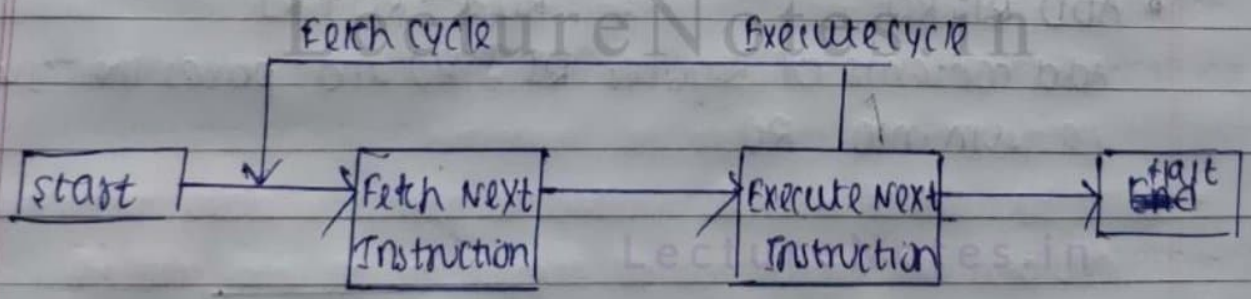


Fig: Basic instruction cycle

The processor fetches instruction from memory one instruction at a time. The time needed to fetch an instruction is called Fetch cycle (FC).
The processor executes the instruction.
The time needed to execute an instruction is called execution cycle (EC).

Atul Kumar 7777888822
Scanned by CamScanner

Consider a typical instruction

- ADD LOCA, R0 : R0 ← R0 + M[LOCA]

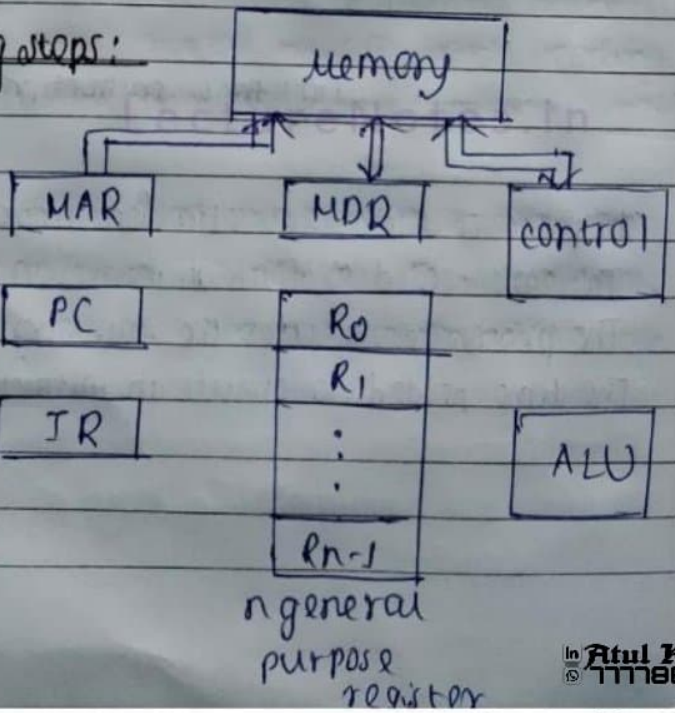
- Add the operand at memory location LOCA to the operand in a register R0 in the processor.
- Place the sum into register R0.
- The original contents of LOCA are preserved.
- The original contents of R0 are overwritten.
- Instruction is fetched from the memory into the processor - the operand at LOCA is fetched and added to the contents of R0
- The resulting sum is stored in the register R0.

This instruction can be rewritten with separate memory access and ALU operation as follows

- LOAD LOCA, R1 : R1 ← M[LOCA]
  Transfer contents of memory location A to register R1

- ADD R1, R0 : R0 ← R0 + R1
  Add contents of Register R1 & R0 and places the result i.e. sum into R0.

- Typical operating steps:



Atul Kumar
7777888822
Scanned by CamScanner

Prime Marik
Page No.
Date

- Programs reside in the memory through input devices.
- PC is set to point to the 1st instruction
- The contents of PC are transferred to the MAR.
- A read signal is sent to the memory.
- The 1st signal is read out and loaded into MDR
- The contents of MDR are transferred to IR
- Decoded and executed the instruction in ALU
- ~~Store the~~ Perform operation in ALU
- Store the result back to → general purpose register
  → memory (address to MAR, result to MDR)
- During the executed execution PC is incremented to the next instruction.

- **Interrupt**: → Normal execution of a program may be temporarily interrupted if some devices require urgent servicing, to do the one device raises an Interrupt signal.

→ An interrupt is a request signal from an I/O device for service by the processor.

→ The processor provides the requested service by executing an appropriate interrupt service routine.

→ When the interrupt-routine service is completed the state of the processor is restored so that the interrupted program may continue.

The sequences of micro operations for instruction fetch are expressed as follows:

Step 1: PC → MAR

Step 2: Memory → MDR

Step 3: PC → PC + 1

Step 4: MDR → IR

in Atul Kumar
© 7777888822

Scanned by CamScanner

H Bus Structures

A bus is a set of wires that connect several devices within a computer system. A group of lines serves as the connection path to several devices. Each line or wire of the bus ot carry 1 bit of information. The lines carry data or address or control signal.

There are different types of buses used in CPU based on types of signals carried.

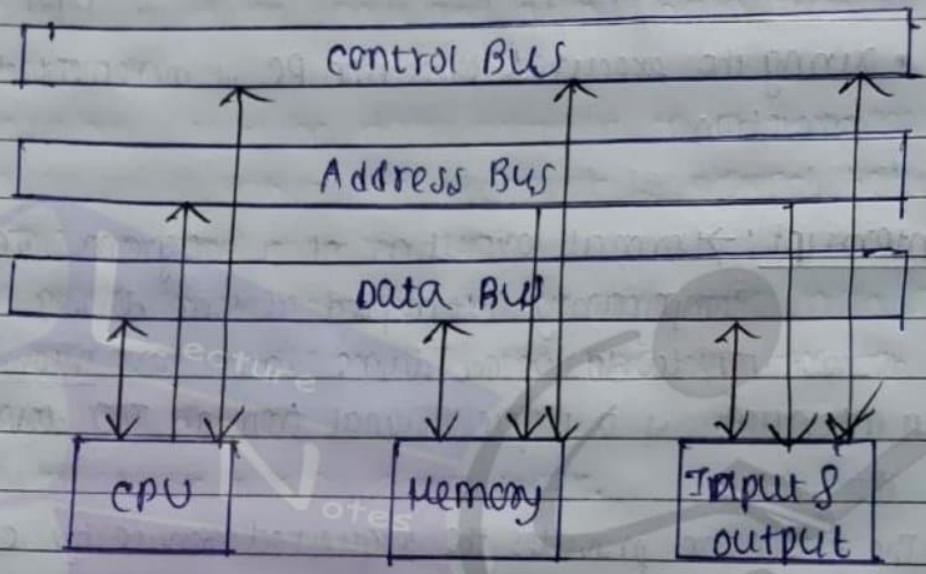


Fig: System Bus

- **Data Bus**: Set of wires dedicated to transfer data within CPU and Memory, input or output devices are called data bus. It is bidirectional as CPU needs sending or receiving data. Maximum addressing capability is nBit

- **Address Bus**: Set of wires that are used to transfer the addresses of memory or I/O devices. It is unidirectional. For n-bit address bus size the maximum addressing capability is 2^n.

Atul Kumar
7777888822

Scanned by CamScanner

- **Control Bus:** CPU uses a control bus to send control signals to memory and I/O devices. These are bidirectional. Some control signals are Read, Write and Opcode, Reset, Ready, etc. This is a dedicated bus as all timing signals are generated according to control signal.

(A) **Single Bus Structure:**

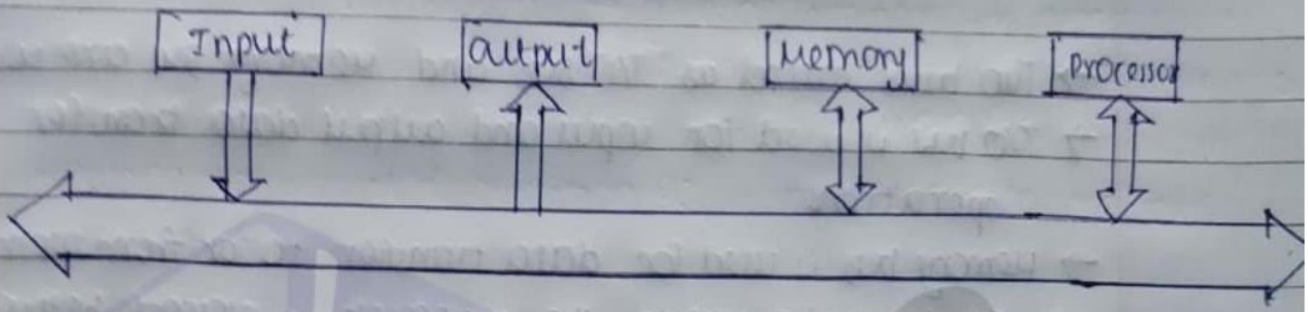
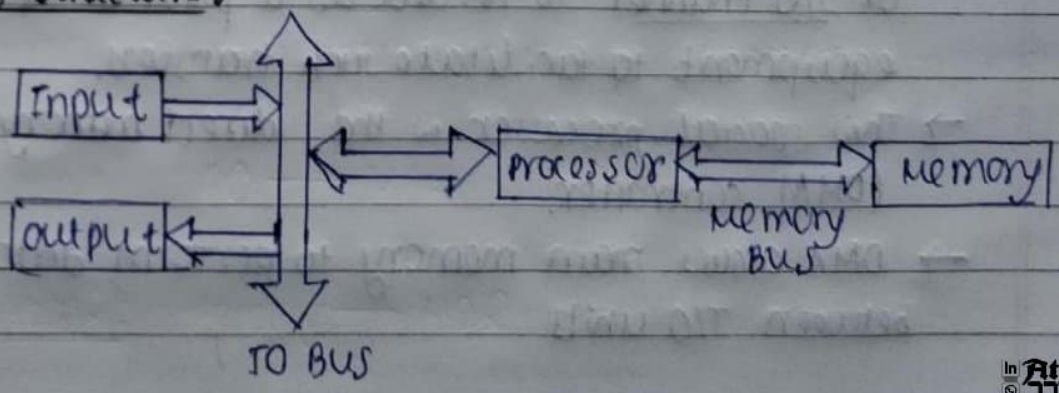


Fig: Configuration of Single Bus

→ All units are connected to a single bus.
→ It is simple and is low of cost.
→ Has a disadvantage of limited speed
→ Only 2 units can participate in data transfer.
→ Bus control lines are used to arbitrate multiple requests for use of the bus.
→ Buffer registers are used to hold information during transfers.

(B) **Two Bus Structure:**



In Atul Kumar
7777888822

Fig: Two Bus Configuration-1

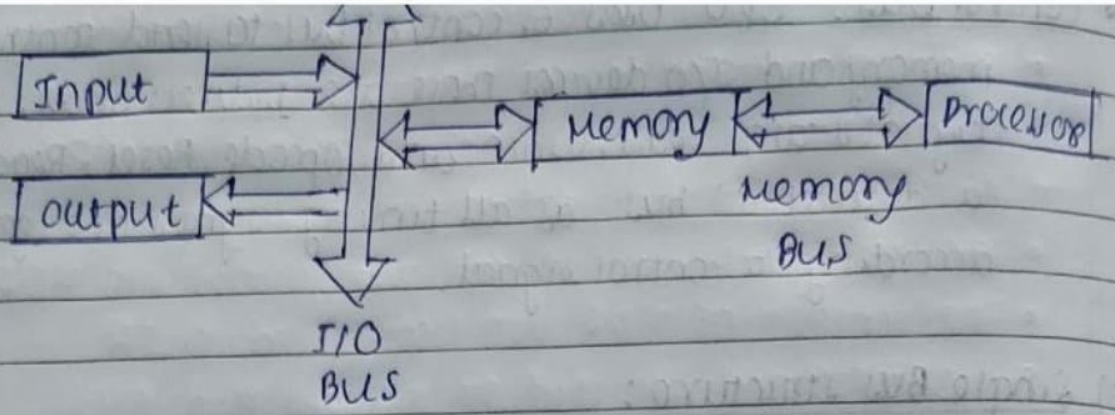


Fig: Two Bus configuration-2

→ Two buses called as I/O bus and Memory Bus are used
→ I/O bus is used for input and output data transfer operations
→ Memory bus is used for data transfer to or from memory.
→ In 1st configuration, the processor is placed between the I/O unit and memory unit.
→ The processor is responsible for any data transfer.
→ The processor is idle most of the time waiting for these slow devices.
→ The processor acts as a messenger.

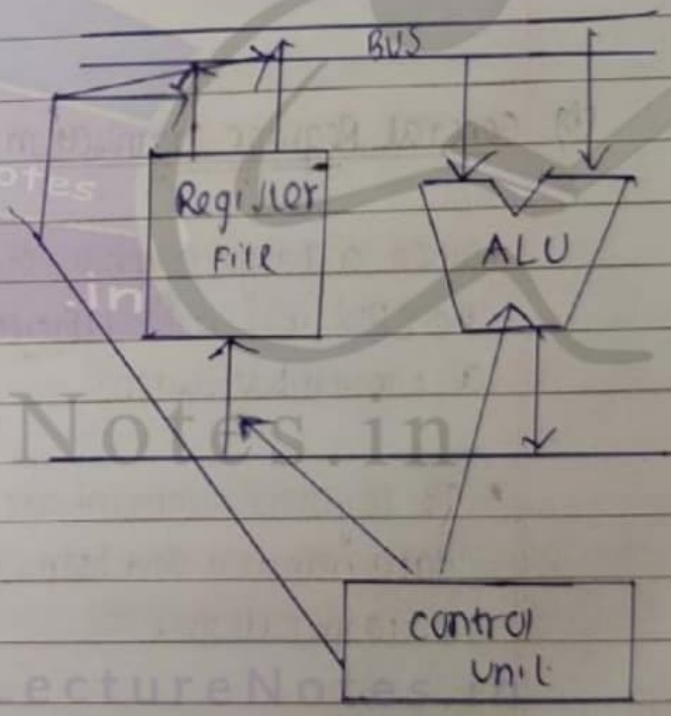
→ In second configuration memory is placed between I/O devices and processor. I/O transfers are made directly to or from the memory
→ A special purpose processor called peripheral processor or I/O channel is needed as a part of the I/O equipment to facilitate such transfers.
→ This special processor is the direct memory access (DMA) controller.
→ DMA allows main memory to perform data transfer between I/O units

Atul Kumar
7777888822

Ch 2: || Central Processing Unit ||

(1) Different CPU organisations:

- Storage components : Registers, Flags
- Execution components : Arithmetic Logic Unit (ALU)
  - arithmetic calculation
  - logical computation
  - shifts / rotates
- Transfer components — BUS
- control components — control unit



(a) Processor organisation:

In general most processors are organized in one of 3 ways:

- Single register (Accumulator) organization
- general register organization
- stack organization

Atul Kumar 7777888822

(a) Accumulator organisation:

- all operations are performed with an implied mode of operation.
- The instruction format in this type of computer uses one address field.

Eg: ADD (X) → address of the operand

- AC ← AC + M[X]
  - AC is the accumulator register
  - M[X] is the memory word located at address X

(b) General Register Organisation:

- When a large number of registers are included in the CPU it is most efficient to connect them through a common bus system.
- The registers communicate with each other for direct data transfers and helps for performing various microoperations.
- Hence it is necessary to provide a common unit that can perform all the arithmetic, logic and shift micro-operations in the ~~co~~processor.

Atul Kumar
7777888822

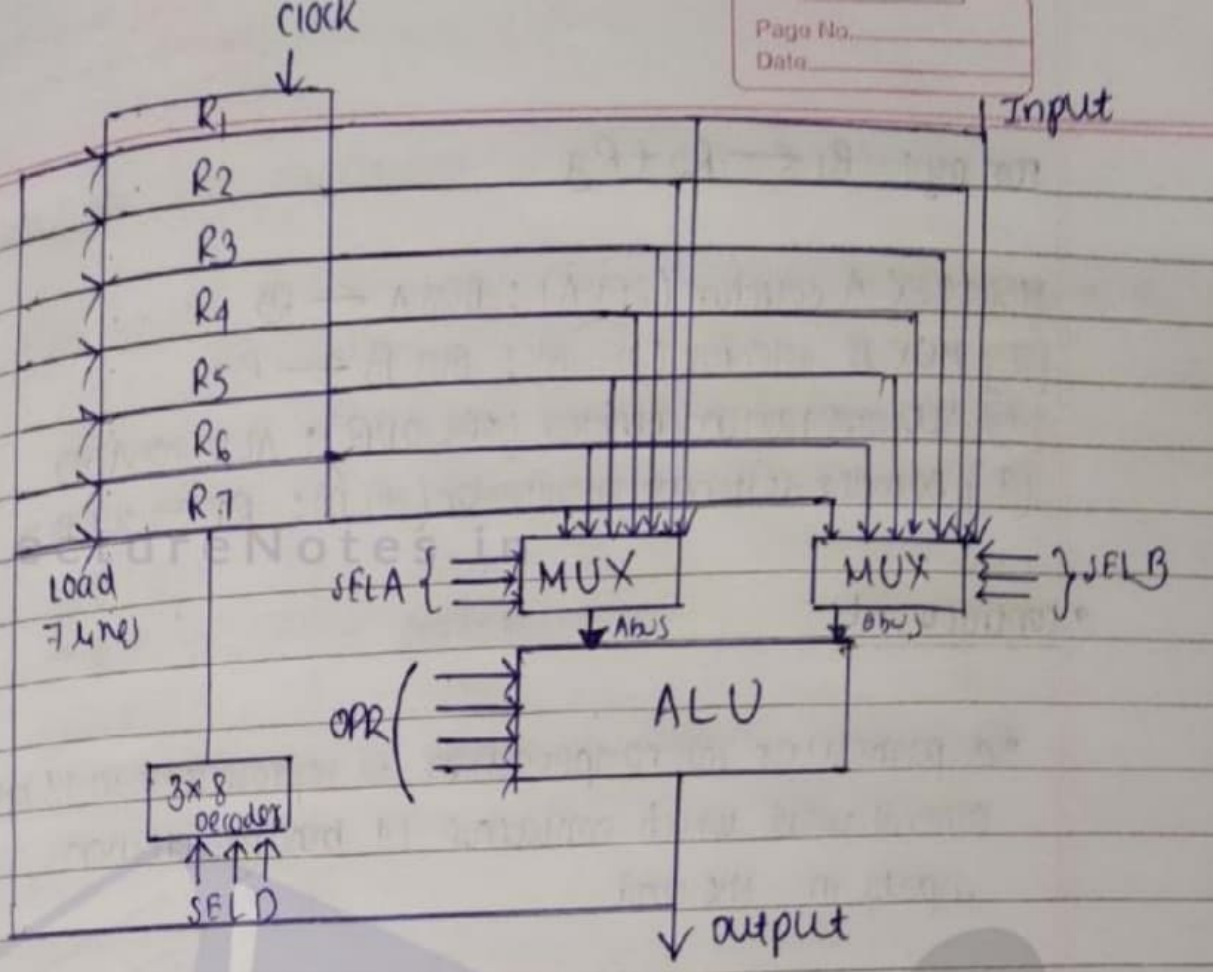


Fig: Register set with common ALU

- consists of registers named R1 to R7
- 2 multiplexers are used to form two buses A and B.
- The A and B buses form the inputs to the ALU.
- The operation selected in the ALU performs the microoperation on the data obtained from the buses.
- The result of microoperation ~~on data obtai~~ goes to the output and also connected into the inputs of all registers
- The register that receives information from the output bus is selected by 3x8 decoder.
- The decoder selects one of the register inputs and provides a path for transfer of data between output bus and inputs of selected register.

- The control unit operates CPU bus system and directs the information flow.

Atul Kumar
7777888822

for eg: $R1 \leftarrow R2 + R3$

[1] MUX A selector (SELA) : BUSA $\leftarrow$ R2
[2] MUX B selector (SEL B) : BUS B $\leftarrow$ R3
[3] ALU operation selector (SEL OPR): ALU to ADD
[4] Decoder destination selector (SELD): R1 $\leftarrow$ out BUS

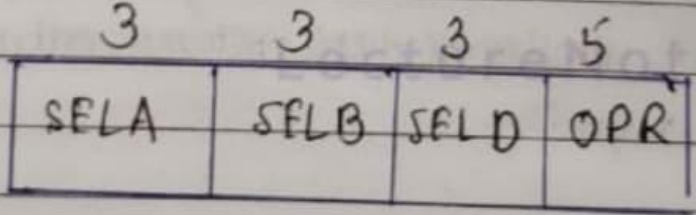
- control word:

- a particular micro operation is represented in 14 bit control word which contains 14 binary selection inputs in the unit.

- The control word consists of 4 fields in which 3 fields contains 3 bits for SELA, SELB, SELD and 1 field OPR has 5 bits.

- 3 bits of SELA select a source register for input A of the ALU
- 3 bits of SELB select a register for the B input of
- 3 bits of SELD select a destination register using the decoder.

- The 5 bits of OPR select one of the operations in the ALU.



LectureNotes.in
Atul Kumar
7717888822

Table: Codes for ALU operations

OPR select	Operation	Symbol
00000	Transfer A	TSFA
00001	Increment A	INCA
00010	ADD A+B	ADD
00101	Subtract A-B	SUB
00110	Decrement A	DECA
01000	AND A and B	AND
01010	OR A and B	OR

Eg: $R_1 \leftarrow R_2 + R_3$ Find out the control word

The control word will be 010 011 001 00101

Field	SFLA	SFLB	SFLD	OPR
symbol:	R2	R3	R1	SUB
control word:	010	011	001	00101

LectureNotes.in

(c) Stack organisation:

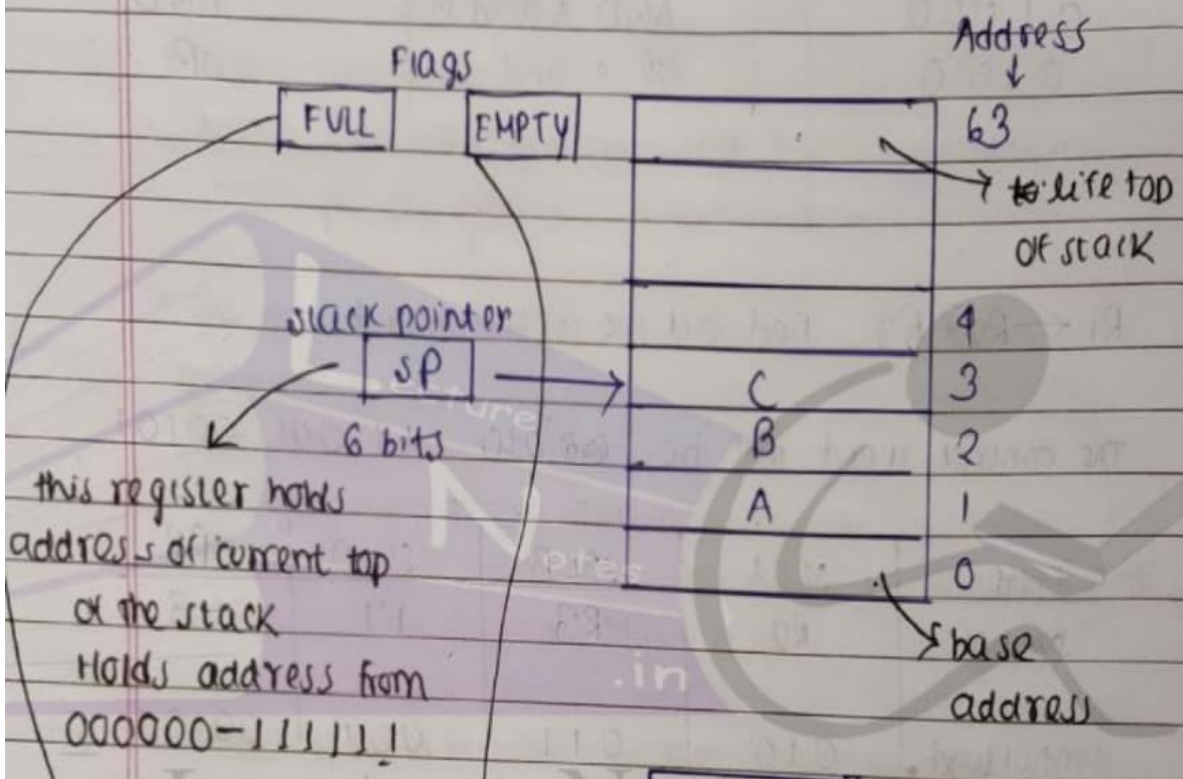
- Stack is LIFO list
- A segment of the memory can be used as a stack.
- The stack contains a continuous memory blocks arranged in the form of a list which allows removing the data from top of stack and inserting data into top of the stack i.e. data entered last will be removed first.

Atul Kumar 7777888822

There are two types of stack

Register stack:

- It can be a portion of the memory or it can be organized as a collection of memory words or registers
- It contains continuous memory words.



LectureNotes.in
Atul Kumar
7717888822

Two operations on the stack are:

(a) PUSH operation

(b) POP operation

(a) Push operation:

Push operation performs the following steps

- SP ← SP+1 (Increment SP)
- M[SP] ← DR (remove contents of DR to top of stack)
- If (SP = 0), then FULL = 1 (check if stack is full)
- EMPTY = 0 then set EMPTY = 0

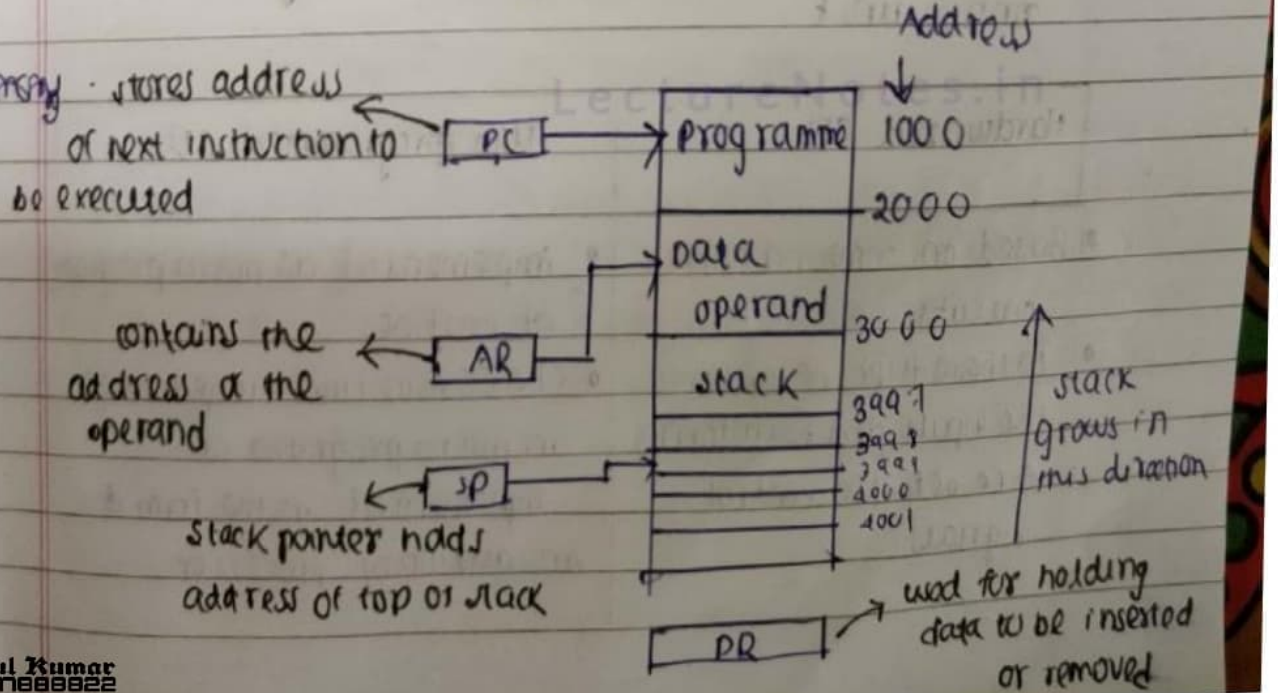
(b) Pop operation:

Pop operation performs following steps.

- DR ← M[SP] (removed data from TOS to DR)
- SP ← SP-1 (decrement SP)
- If (SP = 0) then EMPTY = 1 (check if stack is EMPTY)
- FULL = 0 then set FULL = 0

Memory stack: in 3 sections — Program → contains set of instr; Data (operand) → contains operan; Stack → continuous memory loca

- Memory stack is organized in a portion of the memory to have continuous memory locations



Atul Kumar 7778888922

Two operations are carried out in the stack

(a) Push

(b) POP

(a) Push operation:

Step 1: • SP ← SP - 1 (SP is determinate by 1 to locate the next address of stack)

Step 2: • M[SP] ← DR (contents of DR is transferred to top of stack)

(b) POP operation:

Step 1: • DR ← M[SP] (data on current top of stack is removed and stored in DR)

Step 2: • ~~M[SP]~~ SP ← SP + 1 (SP is incremented by 1)

LectureNotes.in

② Difference between Hardwired and Microprogramed control unit

Hardwared CU	Microprogrammed CU
• Based on combinational circuits • in these types of systems the inputs and transforms are set into control signals	• implemented as micro programs of routines • The control unit implemented in micro program is implemented in the form of an auxiliary processor.

Atul Kumar
7777888822

(c) These units are faster and are known to have a more complex structure

(d) The cost is high.

(e) The hardware is required to be reconfigured every time a change required

(c) These type of circuits are simple but comparatively slower.

(d) The cost is low.

(e) The main advantage is once the hardware configuration is established there is no need for further hardware changes.

(3) Instruction Formats :

Instruction format consists of 3 fields

- **Op-code field:** specifies the operation to be performed
- **Address field:** designates memory address(es) or processor register(s).
- **Mode field:** determines how the address field is to be interpreted or determines direct or indirect addressing of operand.

The number of address fields in the instruction format depends on the internal organization of CPU.

- single accumulator organisation
- general register organisation
- stack organisation

LectureNotes.in
Atul Kumar
7777888822

(i) Single accumulator organisation

- in this format AC is taken as one of the address of the operand.
- And the result of the operation also gets stored in the AC.

ADD X : AC ← AC + M[X]

(ii) General register organisation :

- addresses of the operands used in the instructions are the address of registers or address of memory
- This type of organisation have 3 addresses or 2 addresses of operands.

ADD R1, A, B : R1 ← M[A] + M[B]
ADD R1, R2, R3 : R1 ← R2 + R3
MOV R1, R2 : R1 ← R2
ADD R1, R2 : R1 ← R1 + R2

(iii) stack organisation

- In this type of organisation there is no address of the operands.
- It performs operation on the data stored in the top of the stack.

Hence such type of instruction format is called zero address instruction

Eg: ADD : TOS ← TOS + (TOS-1)

LectureNotes.in
Atul Kumar 7777888822

this operation requires PUSH and POP instruction for insert and dete delete operation respectively.

PUSH A : TOS ← M[A]
PUSH B : TOS ← M[B]
ADD [redacted] : TOS ← M A+B
POP X : M[X] ← TOS

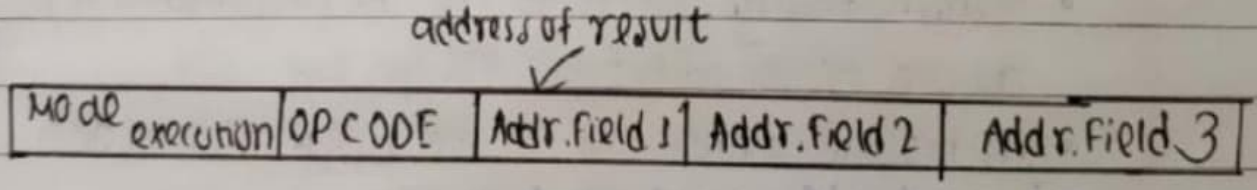
Types of Instruction Format:

Q Consider expression F = (A × B) + (C × D)

- A, B, C, D are memory addresses where operands are stored
- F is another memory addres where result is stored

(a) 3-address instruction:

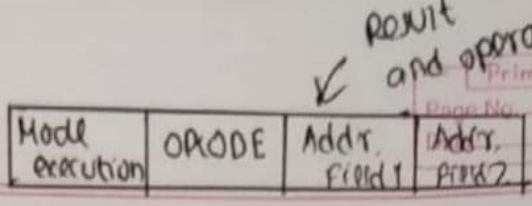
- this format uses 3 addresses of the operand along with the opcode
- the 1st address is address of result and subsequent addresses are meant for the address of the operands.
- this type of instruction format has few instructions.



so for F = (A × B) + (C × D)

MUL R1, A, B : R1 ← M[A] × M[B]
MUL R2, C, D : R2 ← M[C] × M[D]
ADD [redacted] F, R1, R2 : M[F] ← R1 + R2

Atul Kumar
7777888822



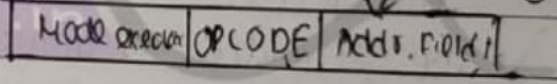
(b) 2-address instruction:

- consists of operation code and 2 addresses of the operands
- The addresses are meant for the processor register or the memory address.
- The 1st address contains one of the operands and the result of the operation.

Eg: F = (A×B) + (C×D)

```
MOV R1, A   : R1 ← M[A]
MUL R1, B   : R1 ← R1 × M[B]
MOV R2, C   : R2 ← M[C]
MUL R2, D   : R2 ← R2 × M[D]
ADD R1, R2  : R1 ← R1 + R2
MOV F, R1   : M[F] ← R1
```

(c) 1-address instruction:



- uses the implied addressing mode
- automatically refers to the AC (accumulator) as one of the address of the operand
- so the AC stores one of the operand and also stores the result of the operation
- Hence instruction contains only one address of the operand.

Eg: F = (A×B) + (C×D)

```
LOAD A  : AC ← M[A]
MUL B   : AC ← AC × M[B]
STORE T : T ← AC
LOAD C  : AC ← M[C]
MUL D   : AC ← AC × M[D]
```

Atul Kumar 7777888822

ADD I: AC ← AC + M[I]
STORE F: M[F] ← AC

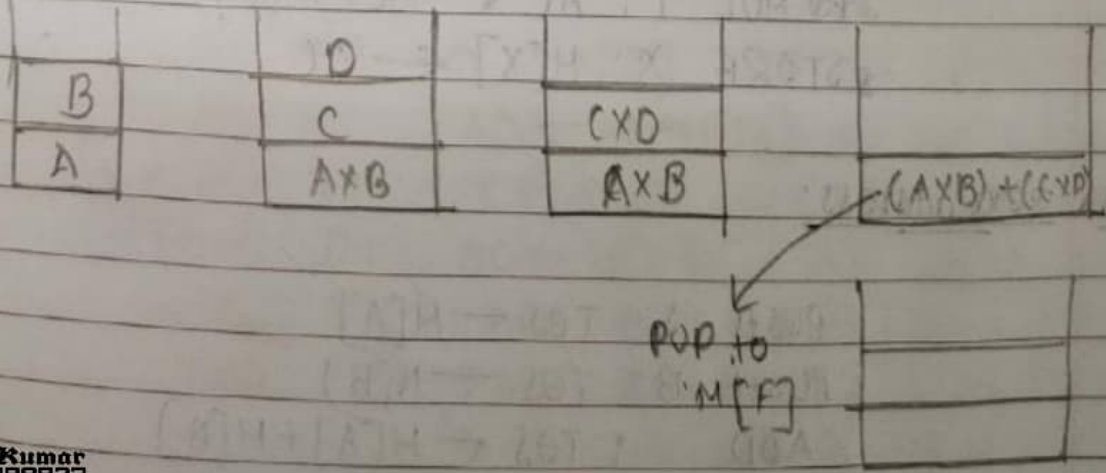
(d) Zero address instruction:

- This type of instruction is based on the stack organisation
- The stack organisation of the CPU does not use address of the operand in the instruction.
- It automatically uses the element of the TOS.
- Hence such type of instruction format that doesn't use the address of the operand is called zero address instruction

Mode execution	OPCODE

For eg: F = (A×B) + (C×D)

PUSH A: TOS ← M[A]
PUSH B: TOS ← M[B]
MUL : TOS ← M[A] × M[B]
PUSH C: TOS ← M[C]
PUSH D: TOS ← M[D]
MUL : TOS ← M[C] × M[D]
ADD : TOS ← (A×B) + (C×D)
POP F: M[F] ← TOS



Atul Kumar
7777888822

Q Program to evaluate expression X = (A+B) * (C+D)

A 3 address:-

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

B 2 address:

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

C 1 address:

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

D zero address:

PUSH A : TOS ← M[A]
PUSH B : TOS ← M[B]
ADD : TOS ← M[A] + M[B]
PUSH C : TOS ← M[C]
PUSH D : TOS ← M[D]

Atul Kumar
7777888822

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

Q Program to evaluate $Y = \frac{A-B}{C+(D \times E)}$

A 3 address:

```
SUB  R1, A, B  : M[A] - M[B]
MUL  R2, D, E  : M[D] * M[E]
ADD  R2, R2, C : M[R2] + M[C]
DIV  Y, R1, R2 : R1 ÷ R2
```

B 2 address:

```
MOV  R1, A  : R1 ← M[A]
SUB  R1, B  : R1 ← R1 - M[B]
MOV  R2, D  : R2 ← M[D]
MUL  R2, E  : R2 ← R2 * M[E]
ADD  R2, C  : R2 ← R2 + M[C]
DIV  R1, R2 : R1 ← R1 ÷ R2
MOV  Y, R1  : Y ← R1
```

C 1 address:

```
LOAD A :  AC ← M[A]
SUB  B :  AC ← AC - M[B]
STORE T:  T ← AC
LOAD D :  AC ← M[D]
MUL  E :  AC ← AC * M[E]
ADD  C :  AC ← AC + M[C]
```

Atul Kumar
7777888822

c 1-address

LOAD D : AC ← M[D]
MUL E : AC ← AC × M[E]
ADD C : AC ← AC + M[C]
STORE T : T ← AC
LOAD A : AC ← M[A]
SUB B : AC ← AC - M[B]
DIV T : AC ← AC ÷ T
STORE Y : Y ← T

d 0-address:

PUSH D : TOS ← M[D]
PUSH E : TOS ← M[E]
MUL : TOS ← M[D] * M[E]
PUSH C : TOS ← M[C]
ADD : TOS ← (D * E) + C
PUSH A : TOS ← M[A]
PUSH B : TOS ← M[B]
SUB : TOS ← M[A] - M[B]
DIV : TOS ← (A-B) ÷ (C + (D * E))
POP X : M[X] ← TOS

Q Program to execute $X = \frac{A-B+C*(D*E-F)}{H*K}$

3 addrs:
MUL R1, D, E : R1 ← M[D] * M[E]
SUB R2, R1, F : R2 ← R1 - M[F]
MUL R3, R2, C : R3 ← R2 * M[C]
SUB R4, A, B : R4 ← M[A] - M[B]
ADD R5, R4, R3 : R5 ← R4 + R3
MUL R6, H, K : R6 ← M[H] * M[K]
DIV X, R5, R6 : X ← R5 ÷ R6

Atul Kumar
7717888822

2 addr:

```
MOV R1,D  : R1 ← M[D]
MUL R1,E  : R1 ← R1 * M[E]
SUB R1,F  : R1 ← R1 - M[F]
MUL R1,C  : R1 ← R1 * M[C]
MOV R2,A  : R2 ← M[A]
SUB R2,B  : R2 ← R2 - M[B]
ADD R1,R2 : R1 ← R1 + R2
MOV R3,H  : R3 ← M[H]
MUL R3,K  : R3 ← R3 * M[K]
DIV R1,R3 : R1 ← R1 ÷ R3
MOV X,R1  : X ← R1
```

3 addr:

```
LOAD D    : AC ← M[D]
MUL E     : AC ← AC * M[E]
SUB F     : AC ← AC - M[F]
MUL C     : AC ← AC * M[C]
~~LOAD A~~ STORE T : M[T] ← AC
LOAD A    : AC ← M[A]
SUB B     : AC ← AC - M[B]
ADD T     : AC ← AC + M[T]
STORE Y   : M[Y] ← AC
LOAD H    : AC ← M[H]
MUL K     : AC ← AC * M[K]
DIV Y     : AC ← A
STORE X   : STORE X
```

cancel

0 addr:

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

Atul Kumar
7777888822

RISC (Reduce Instruction set computer) Format:

- Instruction set of a typical RISC processor is restricted to use load and store instruction when communicating between memory and CPU
- LOAD & STORE → have 1 memory and 1 register address.

- ALL other instructions → 3 address instruction format

Q Program to evaluate $X = (A+B) * (C+D)$

```
LOAD  R1, A  : R1 ← M[A]
LOAD  R2, B  : R2 ← M[B]
LOAD  R3, C  : R3 ← M[C]
LOAD  R4, D  : R4 ← M[D]
ADD   R1, R1, R2 : R1 ← R1 + R2
ADD   R3, R3, R4 : R3 ← R3 + R4
MUL   R1, R1, R3 : R1 ← R1 * R3
STORE X, R1  : M[X] ← R1
```

(4) Addressing Modes:

- An instruction consists of an opcode and operand or address of the operand.
- The method by which the address of the operand is specified in the instruction is known as instruction addressing mode.

Atul Kumar
7777888822

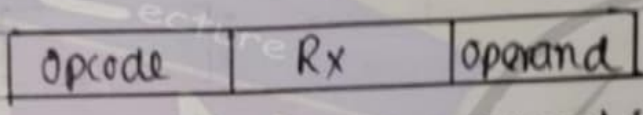
- The address of the operand may be a processor register or a memory address.

- Types:

a) Immediate Addressing Mode:

- The addressing mode in which the instruction that contains the operand itself along with operation itself code is called immediate addressing mode.

- In this case the operand is specified instead of address; hence the operand is immediately used for the operation to be performed



operand is a constant

- Immediate mode is only used to specify the value of a source operand

Eg: 8085 CPU use this type

MVI A, 8bit data : A ← A+8 bit data

ADI 8bit data : AC ← AC + 8 bit data

(b) Implied Addressing Mode:- single register (AC) (1 address)

stack organization (SP) (0 address)

- The process by which AC is automatically implied as address of the operand is called as implied or implicit addressing.

- This happens in a single AC CPU organization where the AC contains the operand.

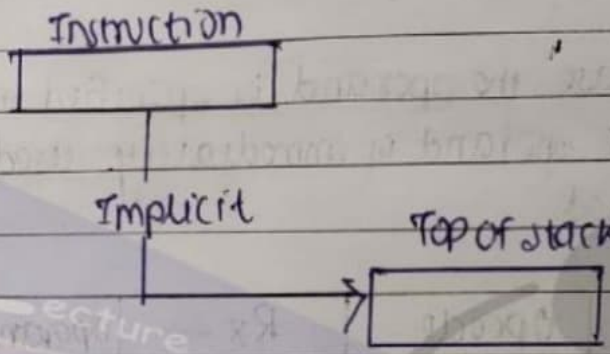
LectureNotes.in

Atul Kumar

- In stack organized computer the address is automatically ~~implied~~ imply the top of the stack.

- Effective Address (EA) = AC
  OR
  Effective Address (EA) = SP

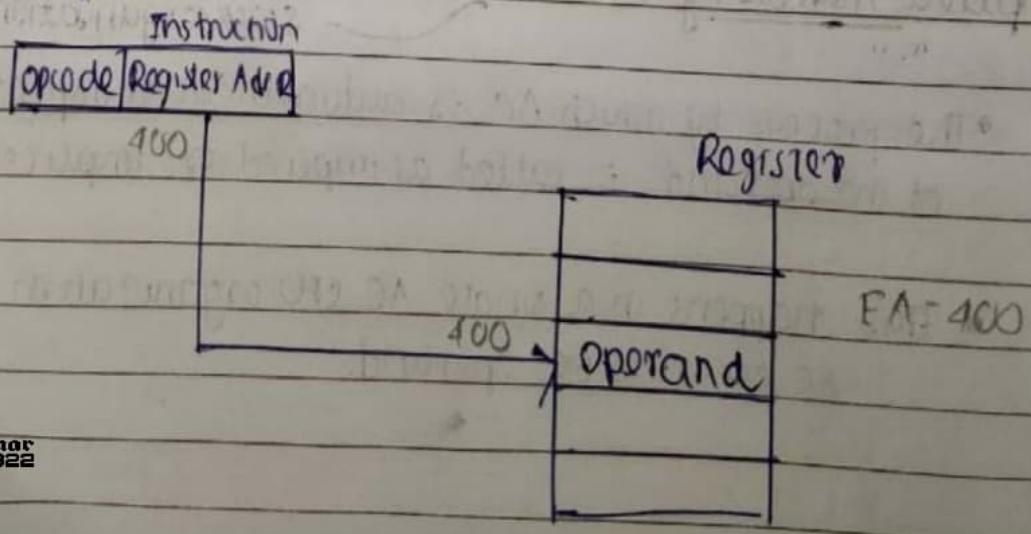
- Examples: From Basic computer
  CLA, CME, RAR, RLA, RRC



(C) Register Addressing Mode:

- The addressing mode in which the name of the processor register is specified in the instruction that contains the operand is called register addressing mode.

- It retrieves data from the specified register and performs a specific operation based on the opcode.



Aful Kumar
7777888822

- shorter address than memory address
- saving address field instruction
- Faster to acquire an operand than the memory addressing
- Ref
  - Effective Address (EA) = IR(REG) → Register field of IR

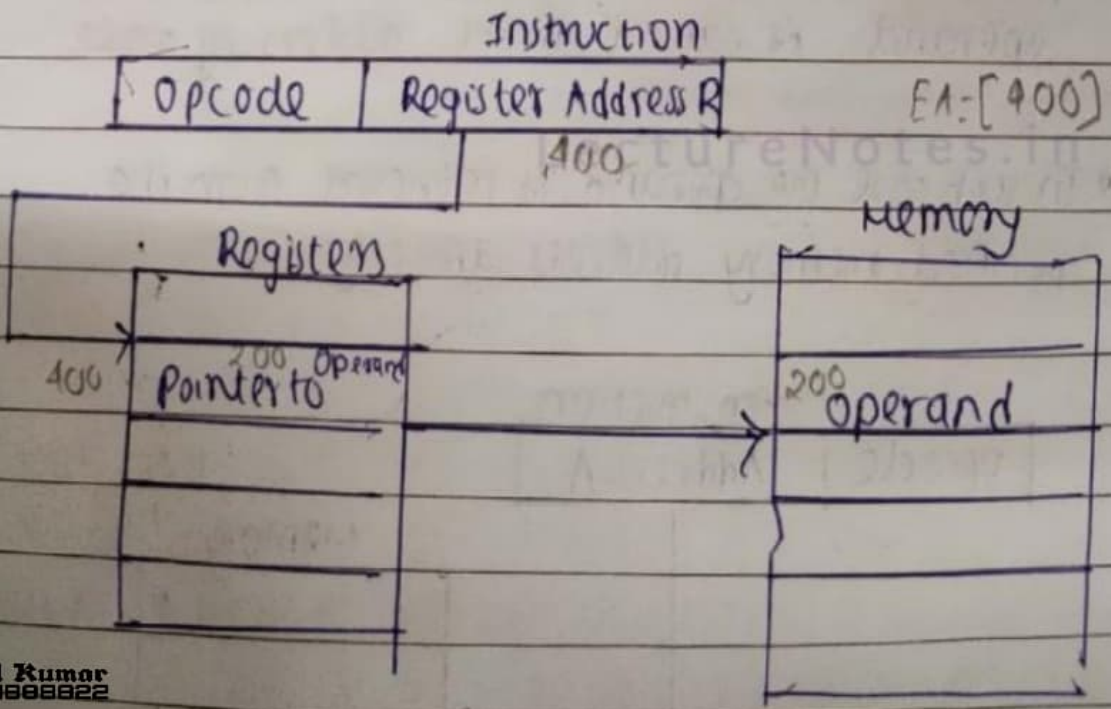
- Example – 8085 CPU use this type.

```
MOVE  A,B     // A ← [B]
ADD   C       // AC ← [AC] + C
```

(d) Register Indirect Addressing mode:

- The process by which the address of the operand is specified in a register that helps to access the data from the memory address is called register indirect addressing mode.

- A register pair holds the address of the memory where the data is stored.



by Atul Kumar
© 7717888822

- Save instruction bit since register address is shorter than memory address.
- slower to acquire an operand than both the register addressing or memory addressing.
- Effective Address (EA) = [IR(REG)]

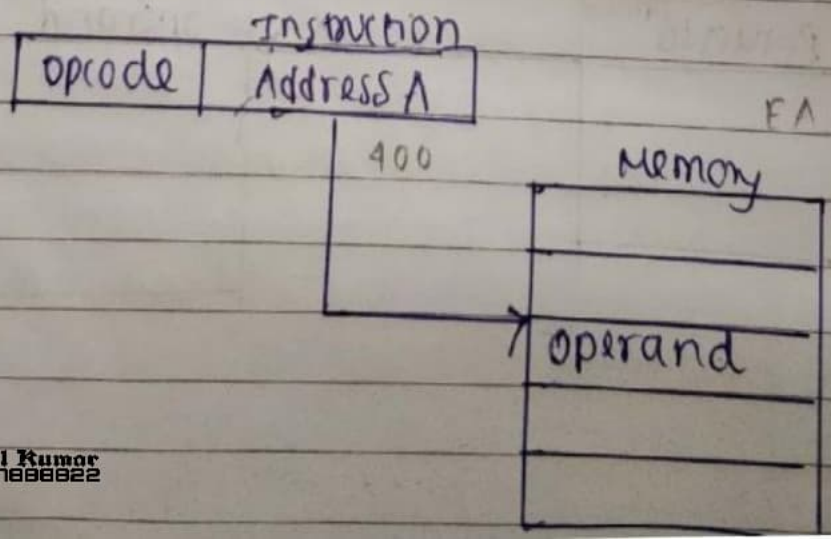
  ([x]: content of x means address of operand)

- Example - 8085 CPU use this type

```
MOVE  A, M[X]          : A ← M[X]
ADD   M[X]             : AC ← [AC] + M[X]
LXI   HL, 16 bit address : HL ← 16 bit address
```

e) Direct Addressing mode:

- The process by which the memory address is specified in the instruction that contains the operand is called direct addressing mode.
- In such case the operand is retrieved from the specified memory address directly.



Atul Kumar
7717888822

- Faster than other memory addressing modes
- Too many bits are needed to specify

- Effective Address (EA) = IR (address) → address field of IR

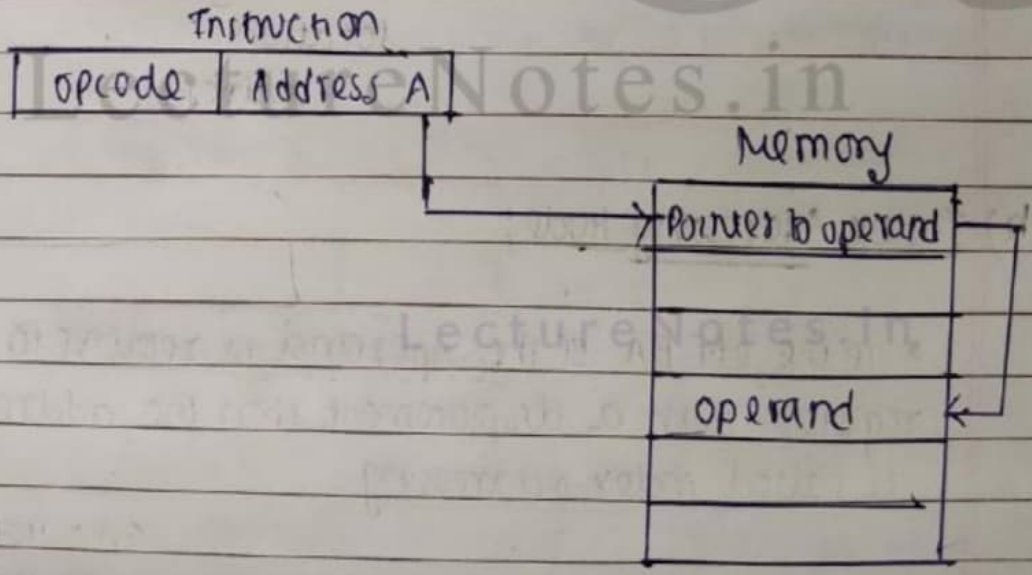
- Example:

LDA 16 bit address // : [AC] ← [content of addre
STA 16 bit address : [Memory of address] ← [A

(E) Indirect Addressing mode:

- The process by which the memory address of the operand is specified in another memory address that contains the operand is called indirect addressing mode.

- The operand is retrieved from the memory address through another memory address.



- Slower to acquire an operand because of additional memory access.

- Effective Address (EA) = M [IREG (address)]
  // IR (address):

- Example: ADD M[300] M[IREG(address)]
  // AC ← [AC] + M[300] [1350]

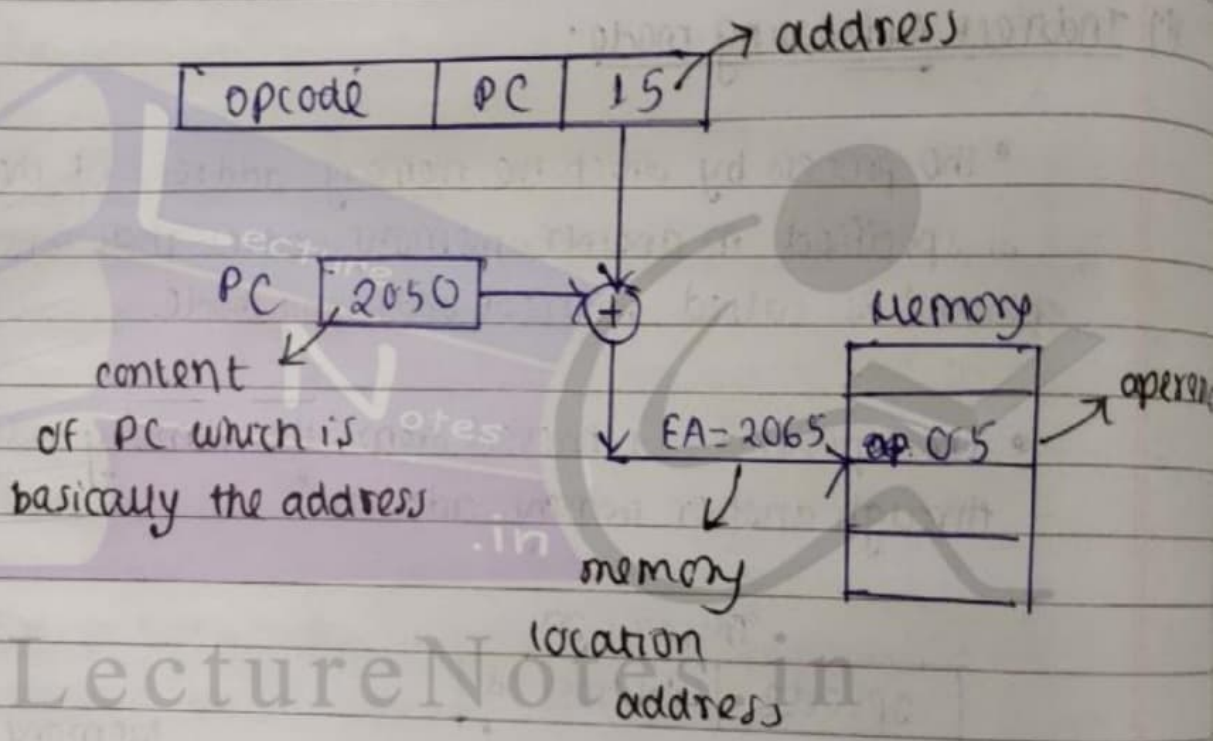
Atul Kumar 7778888822

(g) Relative Addressing Mode:

- there are instruction format where the address of the operand is relative to the contents of a register such as program counter (PC)

- In this case EA is calculated to retrieve the operand.

- Effective address (EA) = [PC] + IR (address)



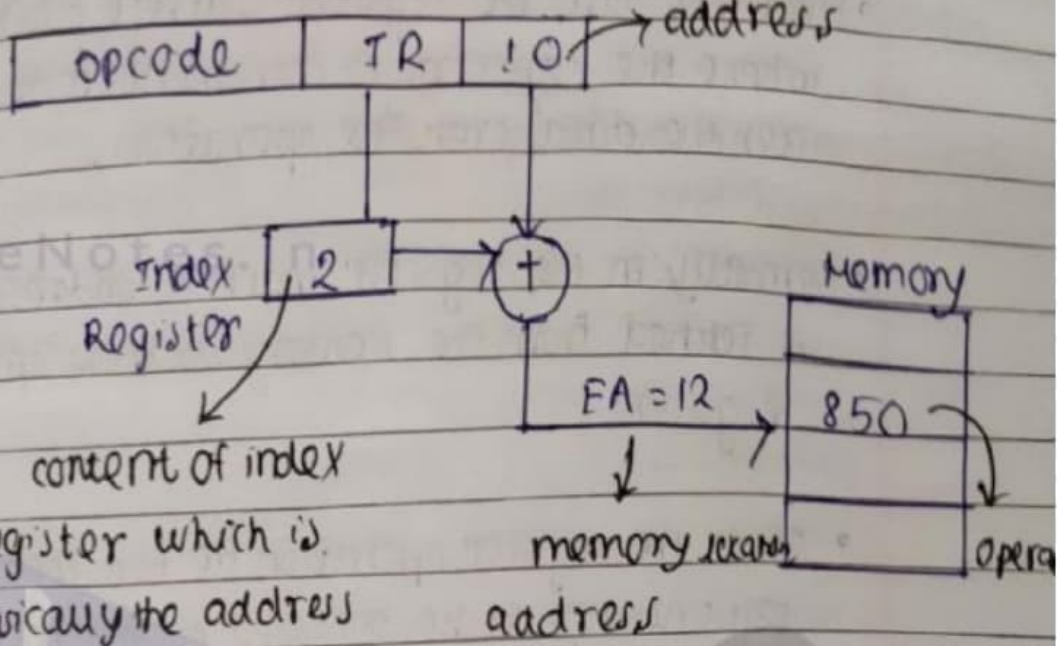
(h) Index Addressing Mode:

- If the EA of the operand is relative to an index register with a displacement then the addressing mode is called index addressing.

- It will be calculated as the addition of the content of index register and 8bit or 16 bit constant number.

LectureNotes.in
Atul Kumar
7777888822

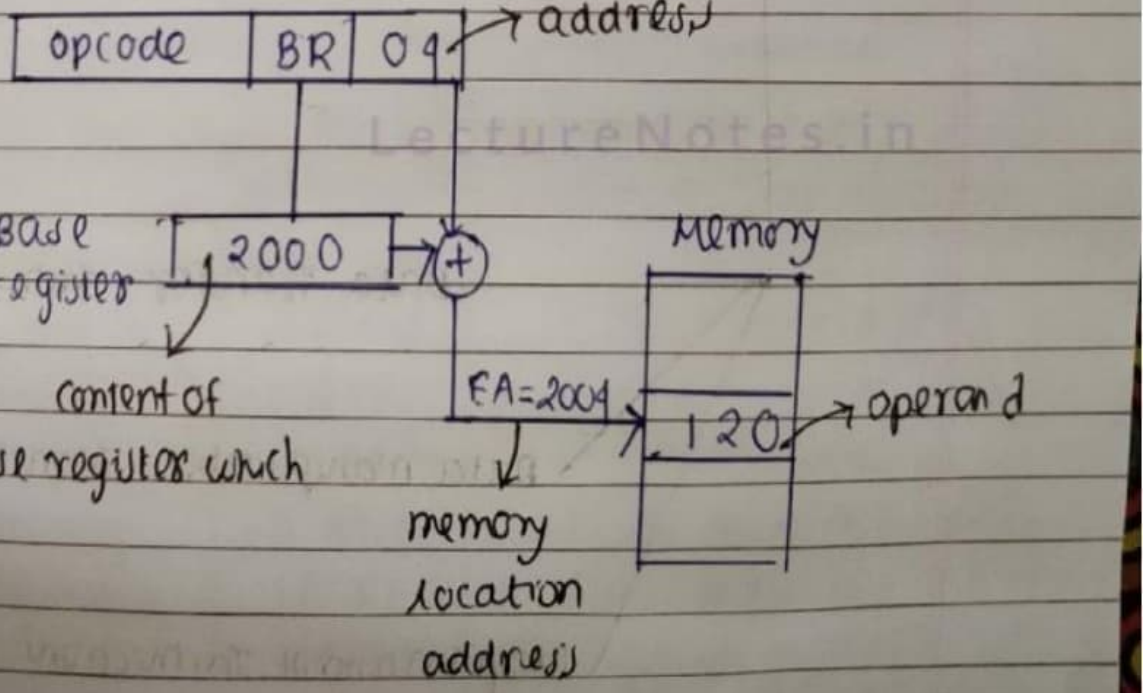
• Effective address (EA) = (Index Register) + IR (address)



(i) Base Register Addressing Mode

• If the EA of the operand is relative to a base register with a displacement then it is called base addressing.

• Effective address (EA) = [Base register] ~~+displacement~~ + IR (address)



Atul Kumar
7777888822

(j) Auto Increment and Auto Decrement :

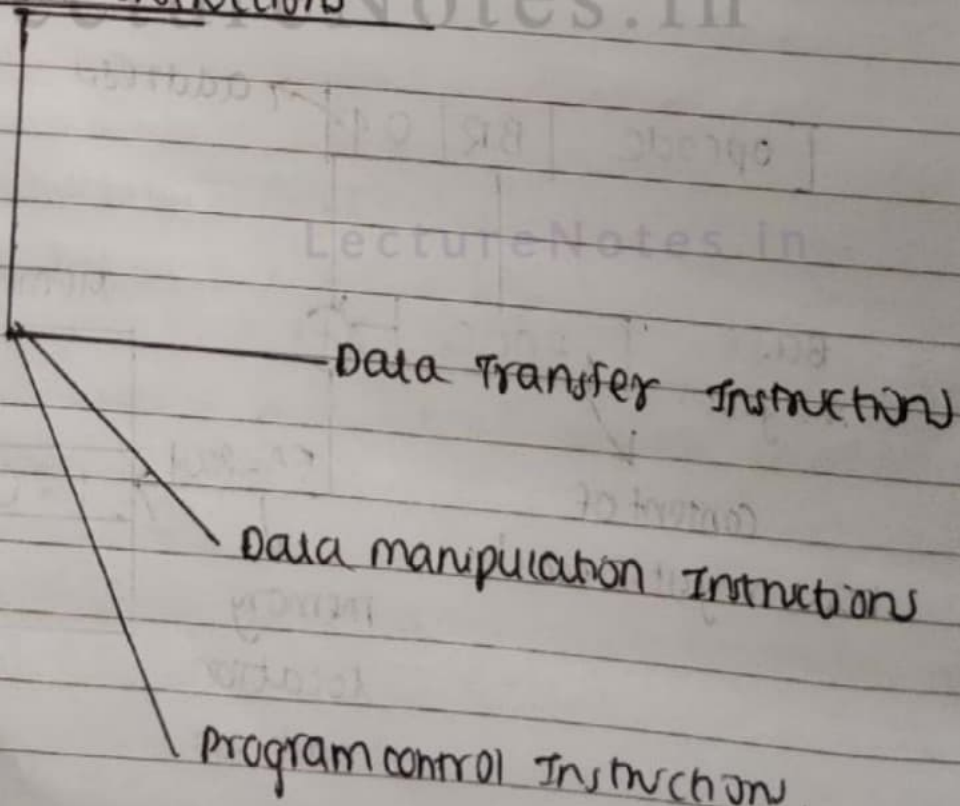
- These are also the register indirect addressing mode where the register is incremented or decremented after the data transfer operation.

- Normally in the register indirect addressing, the operand is fetched from the memory address specified in a register.

- After the register operation the next memory address is sequentially to be accessed by incrementing the register. This is autoincrement.

- If the register is documented to access the previous memory location in reverse order, then it is called auto decrement.

⑤ Computer Instructions



Atul Kumar
7777888822

(a) Data Transfer Instruction

- Cause transfer of data from one location to another without changing the information content
- Transfer may occur between two registers, between memory and processor register, between processor registers and input/output

Typical Data Transfer Instruction

Name	Mnemonic	Function
Load	LD	memory to processor register
Store	ST	processor register to memory
Move	MOV	processor register to memory memory to memory processor register to GPR(General Purpose
Exchange	XCH	swap between two registers data.
Input	IN	data between processor register and input terminal
Output	OUT	data between processor register and output terminal
Push Pop	PUSH POP	data between processor register and memory stack

Data Transfer Instructions with Different Addressing modes

Mode	Assembly convention	Register Transfer
Direct address	LD ADR	AC ← M[ADR]
Indirect address	LD @ADR	AC ← M[M[ADR]]
Relative address	LD $ADR	AC ← M[PC + ADR]
Immediate operand	LD #NBR	AC ← NBR
Index addressing	LD ADR(X)	AC ← M[ADR + XR]
Register	LD R1	AC ← R1
Register Indirect	LD(R1)	AC ← M[R1]
Auto increment	LD(R1)+	AC ← M[R1], R1 ← R1 + 1
Auto decrement	LD -(R1)	R1 ← R1 - 1, AC ← M[R1]

LectureNotes.in
Atul Kumar 7777888822

(D) Data Manipulation Instructions:

Arithmetic Instruction

Name	Mnemonic	Function
Increment	INC	Increment register value by 1
Decrement	DEC	Decrement value by 1
Add	ADD	operation will be performed over operand
Subtract	MUL	
Divide	DIV	
Add with carry	ADDC	
Subtract with Borrow	SUBB	
Negate (2's compl.)	NEG	

Logical and Bit manipulation Instr.

Name	Mnemonic	function
clear	CLR	operand replaced by 0
complement	COM	Produce 1's compl.
AND	AND	
OR	OR	
Exclusive OR	XOR	
clear carry	CLRC	clear carry bit
set carry	SETC	set carry bit
complement carry	COMC	complement carry bit
Enable interrupt	EI	
Disable interrupt	DI	

in Atul Kumar
© 7777888822

Shift Instructions:

(a) Logical shift operation

- shift is the operation that allows bits of a word are moved to the left or right.

Two types
- Logical shift Right (SHR)
- Logical shift left (SHL)

Eg:

Original: 1 1 1 0 1 0 1 0

Left shifted: 1 1 0 1 0 1 0 0 ← 0 inserted to right

Original: 1 1 1 0 1 0 1 0

Right shifted: 0 1 1 1 0 1 0 1

0 inserted to left

- In a register transfer language the following notation is used
  - shl for a logical shift left
  - shr for a logical shift right

Eg: R2 ← shr R2

R3 ← shl R3

(b) Arithmetic shift operation:

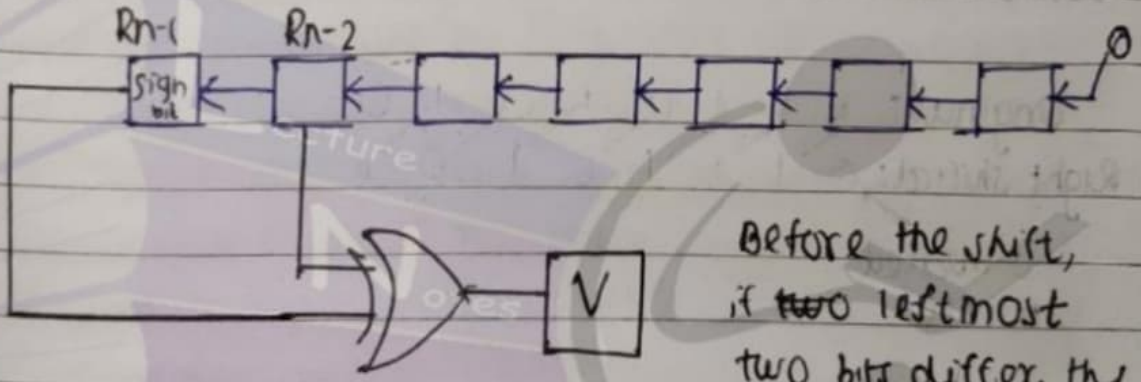
- An arithmetic shift is meant for signed binary numbers (integer)

LectureNotes.in
Atul Kumar
7777888822

• Arithmetic left shift

- this instruction is just a synonym for SHL
- multiplies a signed number by 2
- Insert a '0' to the right most bit and shift each bit to left.
- If the Rn-1 and Rn-2 bits are differed then overflow and stored in flag bit Vs

• A left arithmetic shift operation must be checked for the overflow



Before the shift, if ~~two~~ leftmost two bits differ, the shift will result in an overflow.

Eg: R3 ← SHLA R3

Eg:

sign XOR = 0

+5 : 0 0 1 0 1

left shift : 0 ← 1 ← 0 ← 1 ← 0 ← in 0

sign XOR = 1

+10 : 0 1 0 1 0

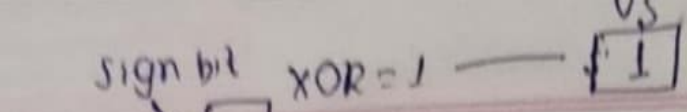
left shift : 1 0 1 0 0 ← in 0

Rn-1 and Rn-2 differs

0 ⊕ 1 = 1 ← Vs

LectureNotes.in
Atul Kumar
7777888822

Page No.
Date



-5 : 1 0 1 0 1

1 0 1 0 ← 1's comp

+ 1

1 1 0 1 1 ← 2's complement

left shift: 1 0 1 1 0 in 0

1 0 0 1 ← 1's complement

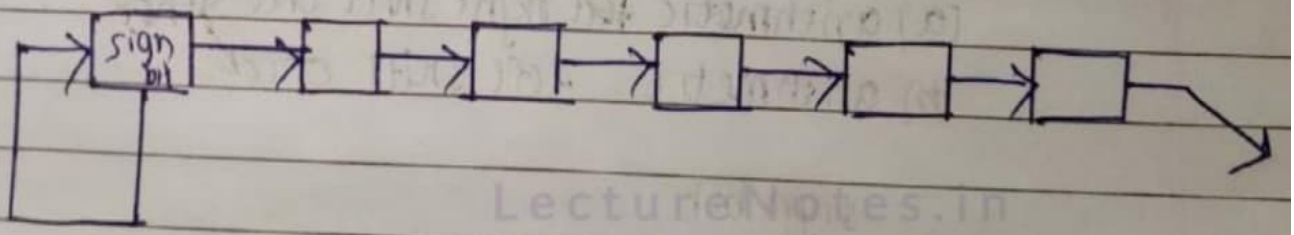
+ 1

1 1 0 1 0 ← 2's complement

$= (-10)$ $\therefore (-5) * 2 = (-10)$

- **Arithmetic Right Shift:**
  - Preserved the sign bit in left most position.
  - Sign bit is shifted to the right together with the rest of bits but the sign bit remains unchanged.
  - divides a signed number by 2.

LectureNotes.in



Eg: R2 ← SHRA R2

Eg:

+10 : 0 1 0 1 0 ← last bit discarded

0 0 1 0 1

in Atul Kumar
© 7777888822

sign bit

-10 : (1) 1 0 1 0

0 1 0 1 ← 1's compl.

+ 1

(1) 0 1 1 0 ← 2's compl. & discard last bit

right shift (1) 1 1 0 1 1

0 1 0 0 ← 1's compl.

+ 1

(1) 0 1 0 1

(-10) = (-5) i.e. (-10)%.2 = (-5)

Q There are 2 different registers of 8 bit length R1 and R2 which contains 11000011 (R1) and 11011101 (R2). and each number already represented in 2's complement form.
What is the decimal equivalent of register content, if we execute

(a) arithmetic right shift once then
(b) arithmetic left shift once

For R1:

sign bit

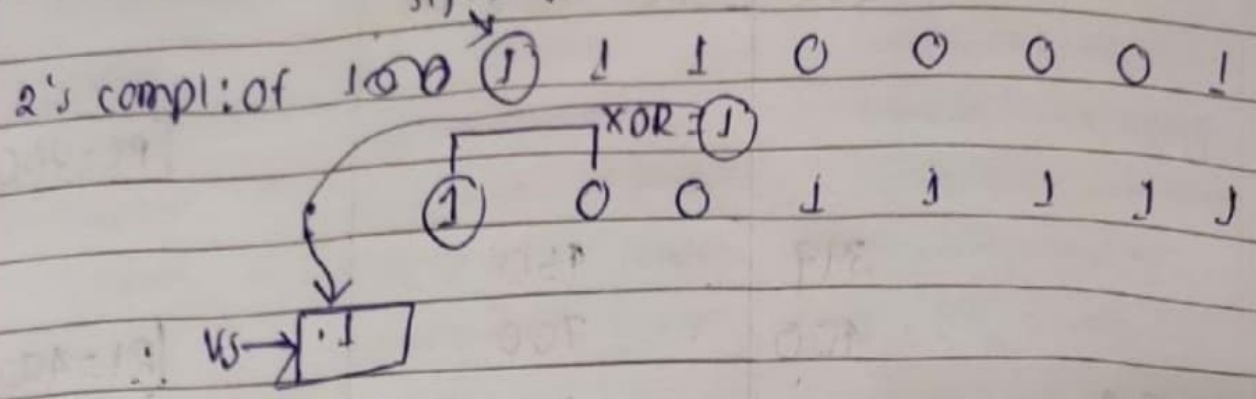
R1 ← (1) 1 0 0 0 0 1 1

right shift: 1 1 1 0 0 0 0 1

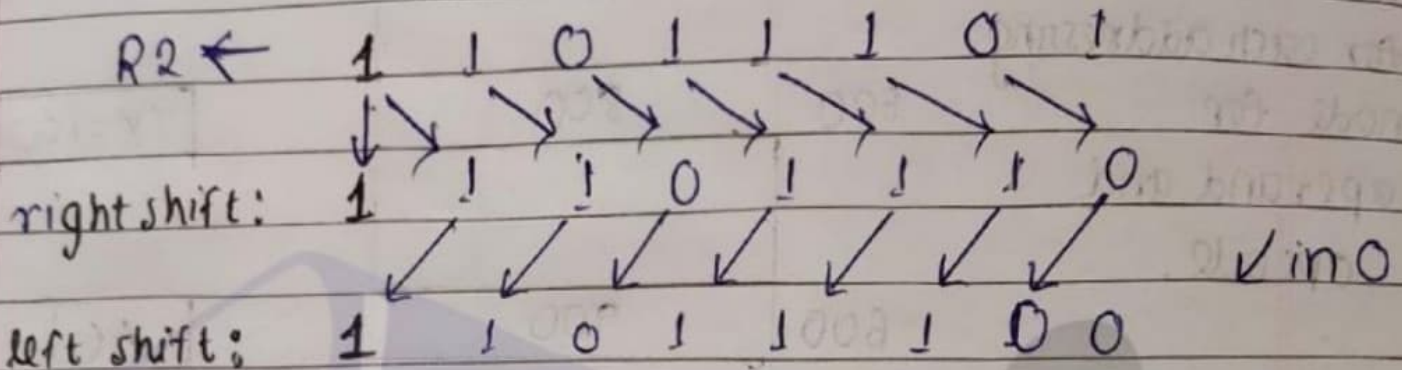
left shift: 1 1 0 0 0 0 1 0 ← zero

Atul Kumar
7717888822

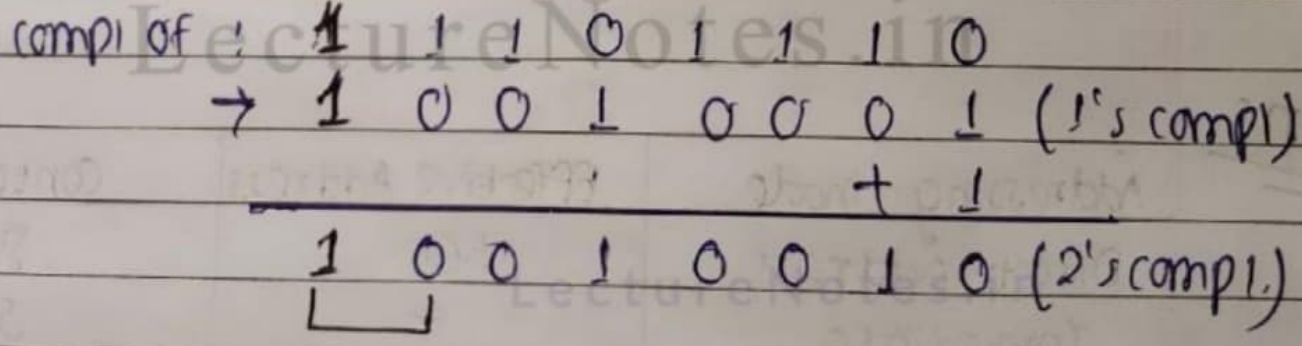
To find out the bit stored in the V_S or flag bit we have to find XOR of the ~~out previous~~ 2's compl of previous data



For R2:



To find out the bit stored in V_S we need to find XOR of R_{n-2} and R_{n-1} bit i.e. 2's compliment of data received after doing right shift.



XOR = 1 ⊕ 0 = (1)

∴ V_S → [1]

LectureNotes.in
Atul Kumar
7777888822

Q2

Address	Memory	
200	Load to AC	Mode
201	Address = 500	
202	Next instruction	
399	450	
400	700	
500	800	
600	900	
702	325	
800	300	

PC = 200

R1 = 400

IX = 100

AC

Calculate EA for each addressing mode for operand and opcode.

Ans

Addressing mode	Effective Address	Content of AC
Direct address	500	800
Immediate	—	500
Indirect address	800	300
Relative address	702 (PC + 500)	325
Indexed address	600 (IX + 500)	900
Register	—	400
Register Indirect	400 (AC ← (R1))	700
Autoincrement	400 (AC ← (R1)+)	700
Autodecrement	399 (AC ← −(R))	450

LectureNotes.in
Atul Kumar
7777888822

Q3

Address	Memory
200	Opcode
201	Address = 500 = operand
202	Next Instruction
B	
399	450
400	700
500	800
600	900
702	325
800	300
900	255
1000	205

PC
200

R1
400

XR = 100

Calculate E.A for each addressing mode for ~~operating~~ operand and opcode

Atul Kumar
7777888822

Ans	Addressing mode	E.A.	content/operand
(1)	Immediate	—	500
(2)	Register	R1	[R1]/400
(3)	Absolute/Direct	500	800/[500]
(4)	Register Indirect	400/[R1]	700/[400]
(5)	Memory Indirect	[500]/800	300/[800]
(6)	Index (IR+IX)	600/(500+100)	900/[600]
(7)	Base (IR+BR)	900/(500+400)	255/[900]
(8)	Base with Index	500/(400+100)	800/[500]
(9)	~~Index~~ Base with index and offset	1000/(400+100+sc)	205/[1000]
(10)	Relative	702/(500+202)	325/[702]
(11)	Auto increment	400	700/[400]
(12)	Auto decrement	399	450/[399]

Q3 Register R_1 and R_2 of a computer contains the decimal value 1200 and 4600. what is the EA of memory operand is each of the following instruction

(a) LOAD 20(R1), R5
(b) ADD -(R2), R5
(c) SUB (R1)+, R5
(d) MOV #3000, R5
(e) STORE R5, 30(R1+R2)
(f) ADD R5, 30(R1+R2)

R1	R2
1200	4600

Ans

(a) LOAD 20(R1), R5

↓

this signifies index addressing

$\therefore EA = X + [R1]$

$= 20 + 1200$

$= 1220$

Atul Kumar
7777888822

(b) ADD -(R2), R5

↓

this signifies autodecrement

EA = 4600 - 1 = 4599

(c) SUB (R1)+, R5

↓

this signifies autoincrement

EA = 1200

(d) MOV #3000, R5

↓

this signifies immediate addressing mode

EA = not applicable to find as memory operands are not given

OR

EA = address of the instruction

(e) STORE R5, 30(R1+R2)

↓ source which is a register operand

↘ destination i.e. content of R5 is stored here

address of destination = 30 + [R1] + [R2]

EA = not required to calculate

LectureNotes.in

in Atul Kumar
7777888822

(f) ADD R5, 30 (R1, R2)

↓

signifies base with index

So EA = 30 + [R1] + [R2]

= 3000 + 1200 + 4600

= 5830

Q4 How many operations are required for each of the instructions.

(a) LOAD 20 (R1), R2

- instruction fetch
- operand fetch from 1220

(2) operations

(b) ADD -(R2), R5

- instruction fetch
- operand fetch from 3599

(2) operations

(c) SUB (R1)+, R5

- instruction fetch
- data fetch from 1200

(2) operations

(d) MOV #3000, R5

- instruction fetch

(1) operation

(e) ADD R5, 30(R1, R2)

- instruction fetch
- fetching data in this location, R5
- storing data in

(3) operati

Atul Kumar
7717888822

(f) STORE R5, 30(R1, R2)

- instruction fetch
- storing data in location R5

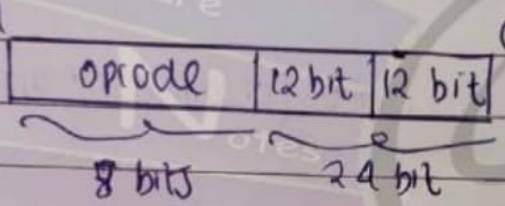
③ operations

Q5 A computer has 32 bit instruction and 12 bit address. That system supports 250 number of 2 address instructions. How many 1 address instructions are possible.

Ans

Instruction length = 32 bit
memory capacity of system = 2^{12}.

250, 2 address instructions are there
it requires 2 address instructions of 12 bit each



2^8 = 256 opcodes are possible.
2^8 actual no of instructions are possible

For 1 address instruction:
operand needs 1-address i.e. 12 bit
for opcode it requires (32-12) = 20 bits

so 2^{20} no of actual instructions are possible
But 2^{20} ~~del~~ address instructions are not there as some are 2 address instructions (∵ it is combination of 2 and 1 addr. instructions)

Total no. of 2 address instructions = $(2^8 - 250)$ = 256 - 250 = 6

so 6 bits remaining are used for calculating 1 address instr.

∴ No. of 1 address instructions = $(2^8 - 250) \times 2^{12}$

Atul Kumar
7777888822

Q6 The memory unit of computer has 256K words of 32 bit each. The computer has an instruction format with 4 fields, an opcode field, mode field to specify 1 of 7 addressing modes, a register address field to specify processor 1 of 60 registers and a memory address field.

Specify the instruction format and no. of bits in each field if the instruction is in 1 memory word.

Ans

memory size = 256K x 32 bit where $K = 2^{10}$

word size = 32 bit

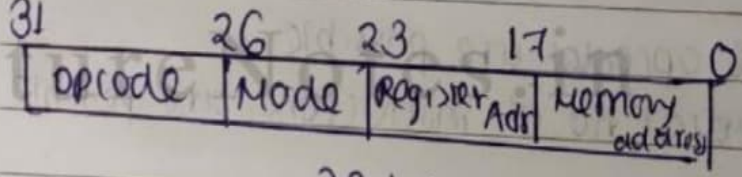
Address bus width = 256K

$= 2^8 \times 2^{10}$

$= 2^{18}$

so we require 18 bits = address length

Instruction size = 32 bit (occupies 1 word)



32 bit instruction

- memory address = 18 bit (0-17)

60 registers are supported by the system

∴ 6 bit is required to address one of the register

[∵ $2^6 = 64$ since our processor is of 60 registers if we use $2^5 = 32$ then 32 no. of processors will be supported which is wrong]

Atul Kumar
7777888822

- so register address = 6 bit (17-23)

7 bits addressing modes are supported so $2^3 = 8$

- so for mode 3 bits are needed (23-26)

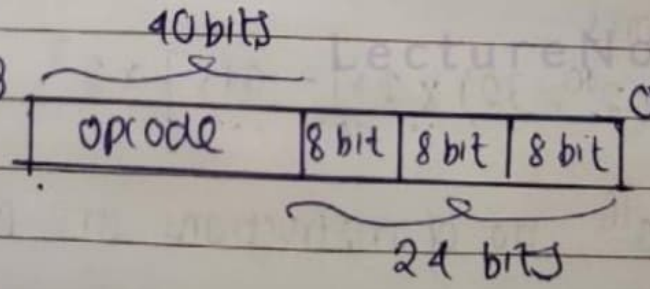
(∵ 7 is nearest to 8 so ~~no~~ we do 2^3 not 2^2)

- ∴ OPCODE field ~~ee~~ requires $32-(3+6+18)$
$= 5$ bit (26-31)

Q7 A computer has 64 bit instruction and 8 bit address. 32 number of 3-address solutions are there, 248 2-address solutions are there.
Calculate how many 1-instructions are there.

Ans Instruction length = 64 bits
memory capacity of system = 2^8

32 no. of 3 address instructions are there.
It requires 3 address of 8 bit each that is 24 bits out of 64 bit solution.



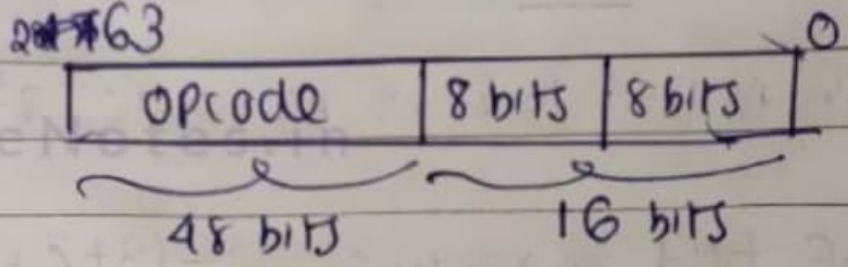
∴ 2^{40} actual 3 address instructions

∴ For 32 bit 3 address instructions = $(2^{40}-32)$ bits

Atul Kumar
7777888822

Again,

248 number of address instructions are there. It requires 2 addresses of 8 bit each. that is 16 bits out of 64 bit instructions



So 2^{48} actual 2 address instructions are presents

For 2 address instruction

$$= 2^{48} - 2^{40} \times 8 2^8$$

out of these 3 address instructions are there.

So for

2 address instruction we have

$$= ((2^{40} - 32) \times 2^8)$$

For 1 address we have

no. of instructions

$$= \left[((2^{40} - 32) \times 2^8) - 248 \right] \times 2^8$$

$= 7 \times 10^{16}$ no. of instructions are possible.

Atul Kumar
7777888822

(6) Distinguish between CISC and RISC.

CISC	RISC
• Large number of instructions	• Relatively fewer number of instructions
• Large number of addressing modes	• Relatively fewer number of addressing modes.
• Instruction format is of variable length	• Instruction format is of fixed length.
• It is prominent on hardware.	• It is prominent on software.
• Transistors used for storing instructions are complex	• Transistors are used for storing memory which is less complex
• Operated on multi-clock	• operated on single clock
• ~~High~~ Has high cycles per second.	• Has low cycles per second.
• cannot have large number of registers	• can have large number of registers.
• Mainly used for personal computers	• Mainly used for real time applications.

LectureNotes.in

Atul Kumar
7777888822

1) Memory Location, Addresses and Operation:

- Memory consists of many millions of storage cells each of which can store 1 bit data as 0/1.

- Data is usually accessed in n bit groups as word.
(where n is called word length)

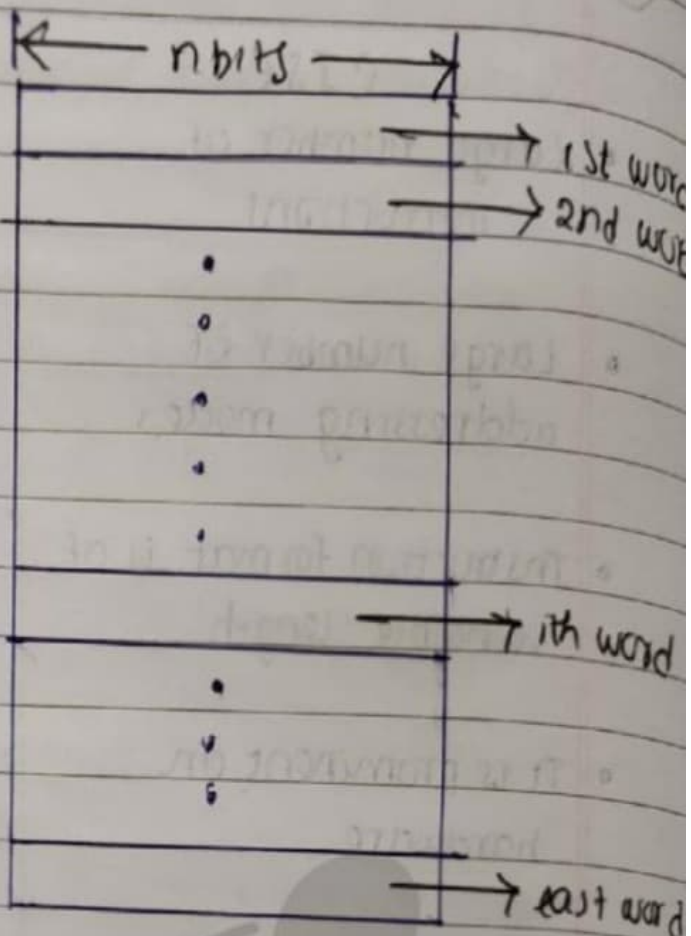


Fig: Memory words

- 32 bit word length example:

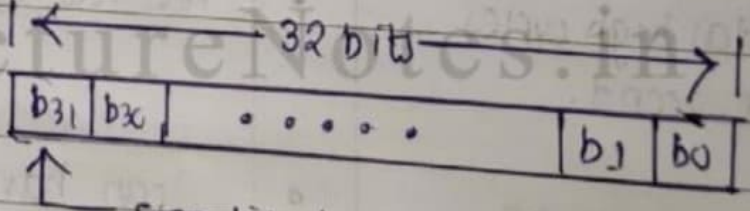


Fig: a signed integer

Atul Kumar
7777888822

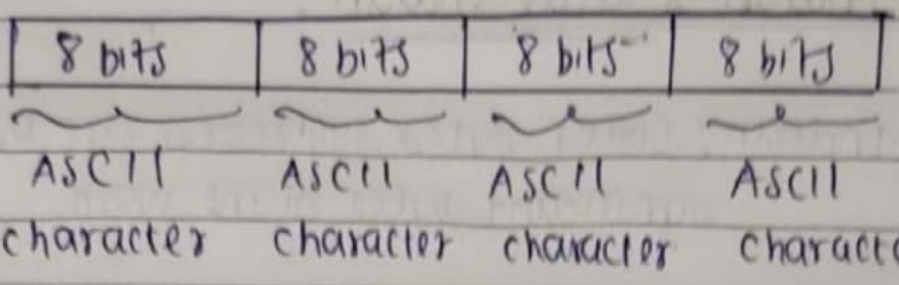


Fig: Four characters

- To retrieve information from memory, either for one word or one byte (8-bit) addresses for each location are needed.
- A K bit address memory has 2^K memory locations, namely 0 to 2^K-1 → called as memory space.

Eg:
- 24 bit memory: $2^{24} = 16,777,216 = 16M$ $(1M = 2^{20})$
- 32 bit memory: $2^{32} = 4G$ $(1G = 2^{30})$
- $1K$ (kilo) $= 2^{10}$
- $1T$ (tera) $= 2^{40}$

- It is impractical to assign distinct addresses to individual bit locations in the memory.
- The most practical assignment is to have successive addresses refer to successive byte locations in the memory called as byte addressable memory
- Byte locations have addresses 0, 1, 2, ...
- If a word length is 32 bit the successive words are located at 0, 4, 8, ... with each word consists of 4 bytes.

In © Atul Kumar 7777888822

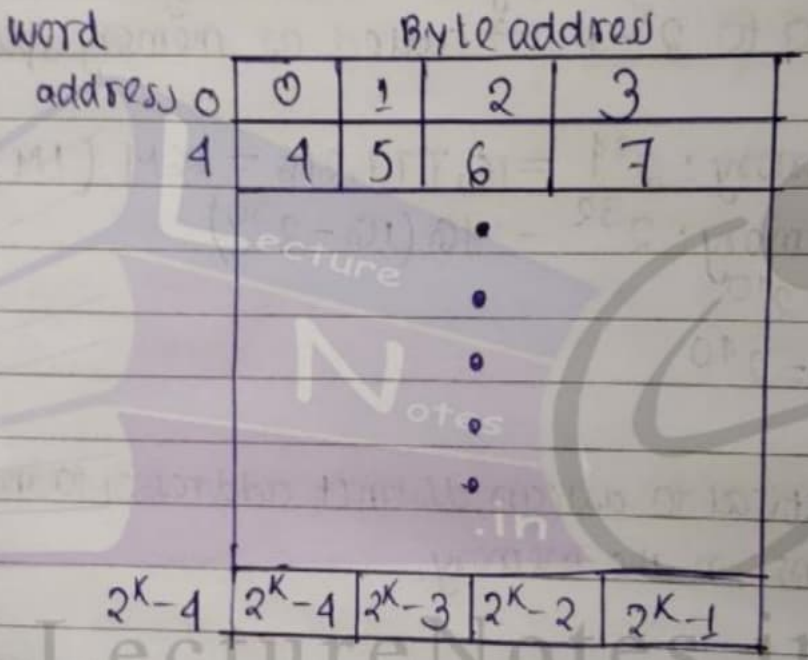
Big-Endian & Little-Endian:

- Big-Endian: Lower bytes addresses are used for the most significant bytes of the word.

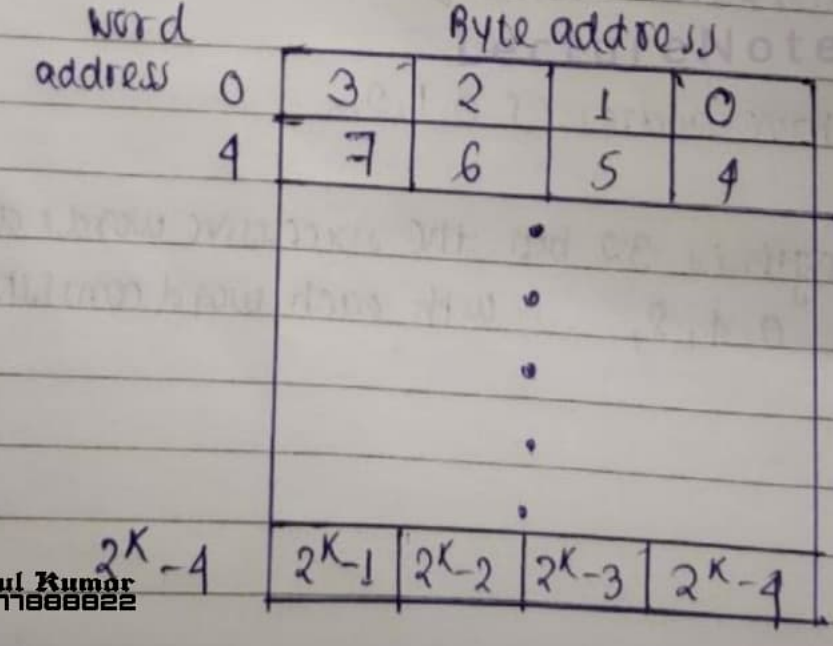
- Little-Endian: (opposite ordering): lower byte addresses are used for less significant bytes of word.

LectureNotes.in

(a) Big-endian assignment:



(b) Little-endian assignment;



Atul Kumar
7777888822

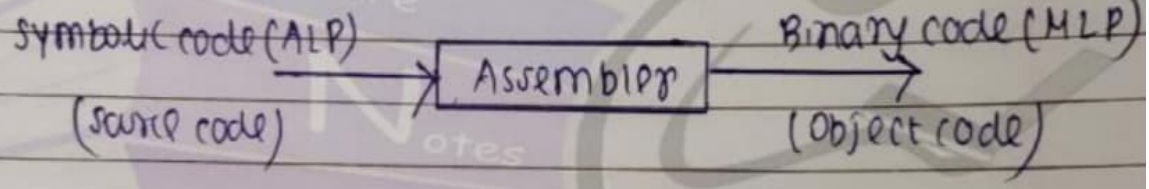
(8) Assembly Language

- The program that substitutes symbols in place of binary number is called symbolic code.

- It is also called as the assembly language program.
- It consists of various symbols such as ADD, DIV, MUL, SUB, MOV, etc. to construct the assembly code.

- This assembly code is converted to binary code by a language translator called as assembler.

- The assembler converts the symbolic code called as source code to binary code called as object code.



- It is easier to write the program in compare to a machine language program.

- The assembly language program consists of 3 fields. such as:
  - (i) Label field
  - (ii) Instruction field
  - (iii) comment field

Label field	Instruction Field		Comments
	Mnemonic	operands	
	MOV	A, B	Move contents of register B to accumulator.
	.....	.....	

LectureNotes.in
Atul Kumar
7777888822

IA-32

IA-32 Processor

Uses a 32 bit address to access memory

Address space → 2^{32} = 4GB locations/ addresses

Uses 8 bit or 32 bit operands.

Implements Little Endian Scheme.

Register Organization

GPRs are classified into 3 categories → used to keep data or address information.

1.Data Registers(Named as A,B,C,D)

JUST REMEMBER THE TERMINOLOGY

Register A is used in an instruction.

So Next is, what is the size of data you are accessing from register A.

If you are going to use 8 bit data, then which 8 bit you want to use?

Lower 8 bit or higher 8 bit

Continued

For Lower 8 bits→ AL is used in an instruction.

For Higher 8 bits→ AH is used in an instruction.

For 16 bits→ you are accessing from AH and AL, AX is used in an instruction.

For 32 bits→EAX is used in an instruction.

Diagrammatically representation of A register is

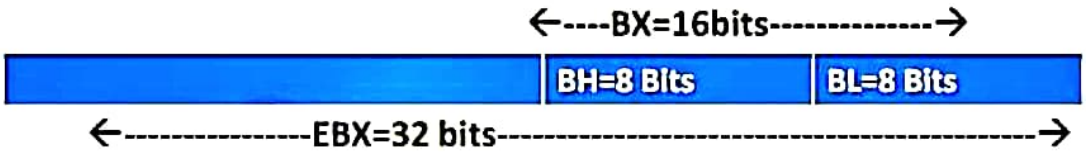
←----AX=16bits→

←----------EAX=32 bits----------→

Used as Accumulator.J

Continued

Diagrammatically representation of B register is

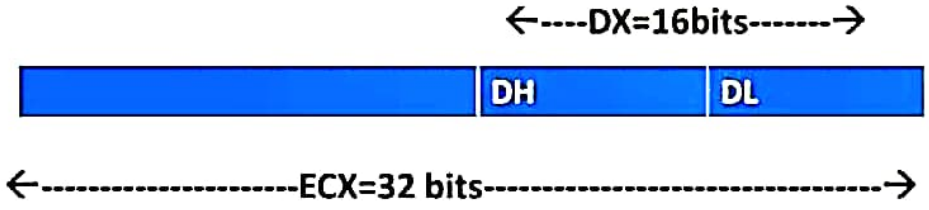


Used to keep offset/ base which is used to form address.

Diagrammatically representation of C register is

Used as Counter.

Similarly D register for Destination address. EDX=32,



Pointer Registers

2.Pointer Registers

SP(Stack pointer)→ used to access stack.

BP(Base Pointer)→ used to access consecutive memory locations

←--------ESP=32 Bits----------→

←----------EBP=32 Bits----------→

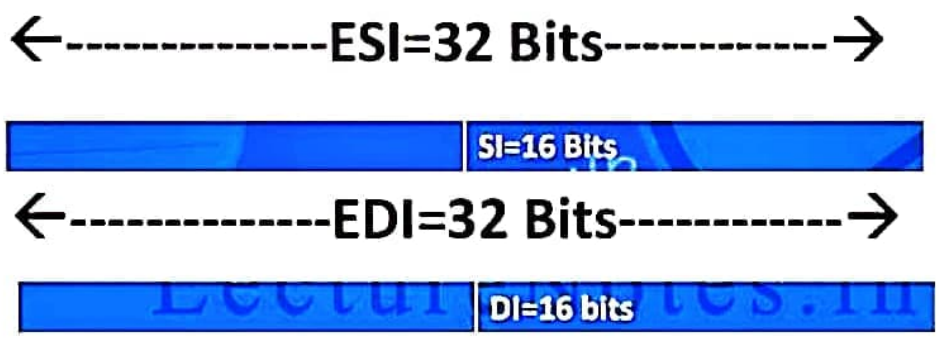
Atul Kumar
7777888822

Index Registes

3.Index Registers

SI(Source Index)-> used to access source data.

DI(Destination Index)→ used to access Destination data.



Atul Kumar
7777888822

Special Registers

Besides GPRs, Two Special registers are available.

IP(Instruction Pointer)-> used to access instructions of a program.(same function as PC).

←----------EIP=32 Bits----------→

Status Registers→ to keep the status.

←----------EFLAGS=32 Bits--------→

Atul Kumar
7777888822

Segment Registers

IA-32 uses a segmented memory(whole memory is divided into segments).

Here memory address is calculated as content of segment register and content of index/base/offset.

6 types of segment registers are available and all are of 16 bits.

1.CS(Code segment)→ where program is kept. So address of instruction=[CS]+[IP] (Same roll as PC)

Segment Registers Continued

2. SS(Stack Segment)→ keeps the base address of stack. Address of stack element is calculated as [SS]+[SP].
3. DS(Data Segment)→ For accessing data
4. ES(Extra Segment)→ For accessing data also.

In higher version processor of IA-32 series,

5 &6.→ FS and GS are used to access program.

IA-32 Addressing Modes

Instruction format=Opcode dst,src;

dst=destination, src=source

1. **Immediate addressing mode→ Data is a part of instruction.**

   MOV AX,32; 32⇒[AX]

   E.A.= Value in the instruction

2. **Direct Addressing Mode→ Address is a part of instruction. [ ] is used to indicate the direct addressing mode.**

   MOV AX, [32]

   E.A= Address / location

Addressing mode Continued

3. Register Direct Addressing mode→ Register is specified in the instruction.

MOV AX,BX;

E.A= Register

4.Register Indirect Addressing Mode→ Register is specified in the instruction and that register holds the address of operand

MOV AX,[BX];

E.A= [Register]

Addressing mode Continued

The E.A. is calculated as follows

E.A= Base Reg +Index Reg + Displacement

5. Base with displacement→

MOV AX, [EBP+60]

E.A= [Reg]+ displacement

6.Index with displacement→

MOV AX, [SI*4+10]

E.A= [Reg* S]+ displacement

S= Scale factor=1/2/4/8

Addressing mode Continued

7. Base with index →

MOV AX, [BP+SI*4];

E.A= [Reg1]+ [Reg2]*S

8.Base with Index with displacement→

MOV AX, [BP+SI*4 +10];

E.A= [Reg1]+ [Reg2]*S +Displacement

9. Implied Addressing mode→

The address of operand is not specified in the instruction.

Atul Kumar
7777888822

FLOATING-POINT DECIMAL NUMBER

- $-123456.0 \times 10^{-1} = 12345.6 \times 10^{0}$

$= 1234.56 \times 10^{1}$

$= 123.456 \times 10^{2}$

$= 12.3456 \times 10^{3}$

$= 1.23456 \times 10^{4}$ (normalised)

- **Three pieces of information represents a number:**
  - Sign of the number
  - Significant value
  - Signed exponent of 10.
- **Example**
  - The range of a fixed-point decimal system with six digits, of which two are after the decimal point, is 0.00 to 9999.99.
  - The range of a floating-point representation form m.mmm $\times 10^{ee}$ is 0.0×10^{0} to 9.999×10^{99}.

Atul Kumar
7777888822

IN C PROGRAM

- Data of type float and double are represented as binary floating-point numbers.
- These are approximations of real numbers as like an int, an approximation of integers.

IEEE NOTATION

IEEE Floating Point notation is the standard representation in use. There are two representations:

- Single precision.
- Double precision.

Both have an implied base of 2.

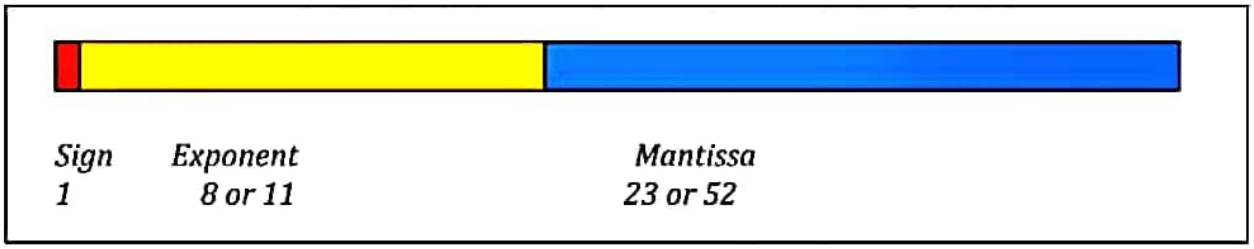
Single precision:

- 32 bits (23-bit mantissa, 8-bit exponent in excess-127 representation)

Double precision:

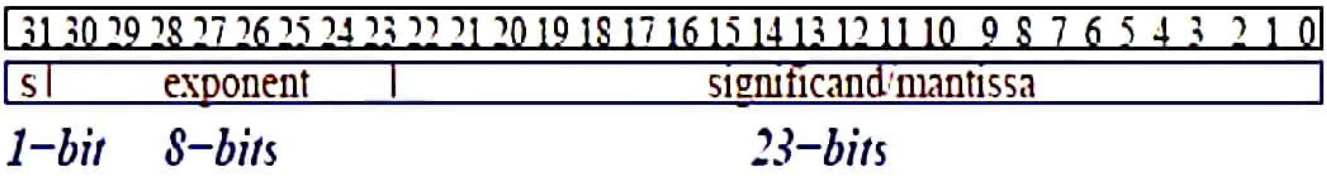
- 64 bits (52-bit mantissa, 11-bit exponent in excess-1023 representation)

Fractional mantissa, with an implied binary point at immediate left.

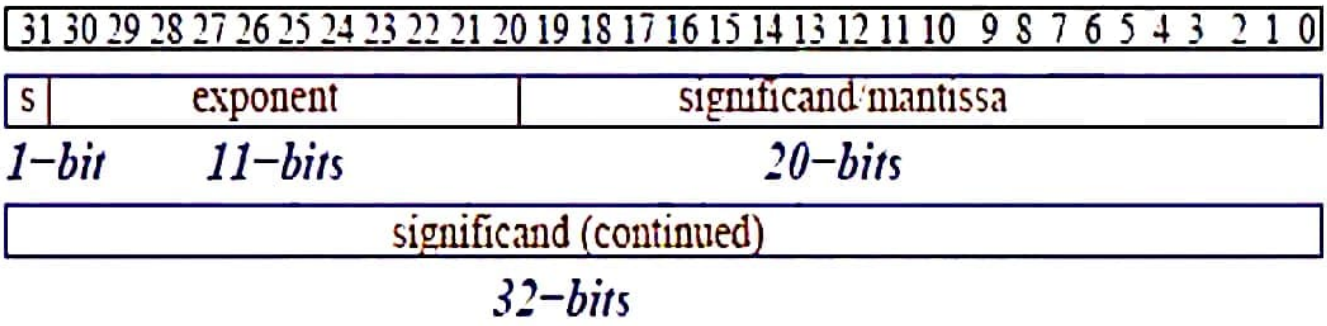


Atul Kumar
7777888822

IEEE NOTATION



Single Precession (32-bit)

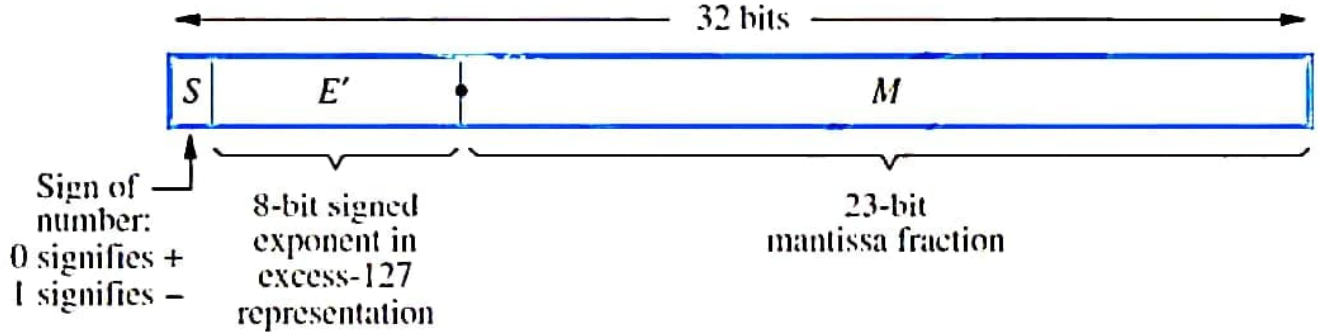


Double Precession (64-bit)

Atul Kumar
7777888822

IEEE REPRESENTATION

1. Sign of the number is given in the first bit, followed by a representation of the exponent (to the base 2) of the scale factor.
2. In case of signed exponent, E , the value stored in the exponent field is an unsigned integer E ' = E + 127



$$\text{Value represented} = \pm 1.M \times 2^{E'-127}$$

(a) Single precision

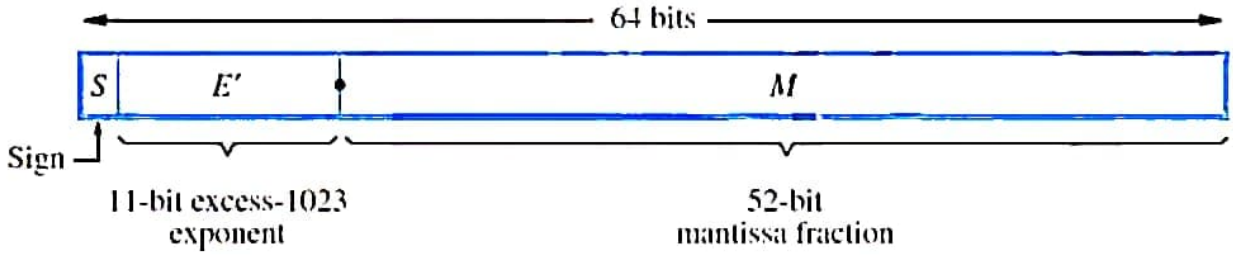
$$\text{Value represented} = 1.001010\ldots0 \times 2^{-87}$$

(b) Example of a single-precision number

Atul Kumar
7778888822

IEEE REPRESENTATION

1. Sign of the number is given in the first bit, followed by a representation of the exponent (to the base 2) of the scale factor.
2. In case of signed exponent, E , the value stored in the exponent field is an unsigned integer E ' = E + 1023



Value represented = $\pm 1.M \times 2^{E'-1023}$

(c) Double precision

Atul Kumar
7777888822

IEEE REPRESENTATION

- Floating point numbers have to be represented in a normalized form to maximize the use of available mantissa digits.
- In a base-2 representation, this implies that the MSB of the mantissa is always equal to 1.
- If every number is normalized, then the MSB of the mantissa is always 1.
- IEEE notation assumes that all numbers are normalized so that the MSB of the mantissa is a 1 and does not store this bit.

Atul Kumar
7777888822

An Example

- Consider the following 32-bit pattern
- 1 1011 0110 011 0000 0000 0000 0000 0000
- The value is
- $(-1)^1 \times 2^{10110110-01111111} \times 1.011$
- $= -1.375 \times 2^{55}$

Atul Kumar
7777888822

An Example

- Consider the decimal number: +105.625.
- The equivalent binary representation is :- +1101001.101

  + 1101001.101

  $= + 1.101001101 \times 2^{6}$

  $= + 1.101001101 \times 2^{133-127}$

  $= + 1.101001101 \times 2^{10000101-01111111}$

- In IEEE format:
- 0 1000 0101 101 0011 0100 0000 0000 0000

Atul Kumar
7777888822

IEEE EXPONENT FIELD RANGE

1. In the IEEE representation, the exponent is in excess-127 (excess-1023) notation.
2. The actual exponents represented are:

$$-126 <= E <= 127 \quad and \quad -1022 <= E <= 1023$$
$$not$$
$$-127 <= E <= 128 \quad and \quad -1023 <= E <= 1024$$

This is because the IEEE uses the exponents -127 and 128 (and -1023 and 1024), that is the actual values 0 and 255 to represent special case.

Atul Kumar
7777888822

IEEE REPRESENTATION

Special Case

1. When E ’ = 0 (-127+127) and
   Mantissa fraction (M) = 0 ,
   then the value exact ZERO is represented.
2. When E ’ = 255 (-128+127) and
   Mantissa fraction (M) = 0 ,
   then the value is ∞ (infinity) represented.

Atul Kumar
7778888822

IEEE REPRESENTATION

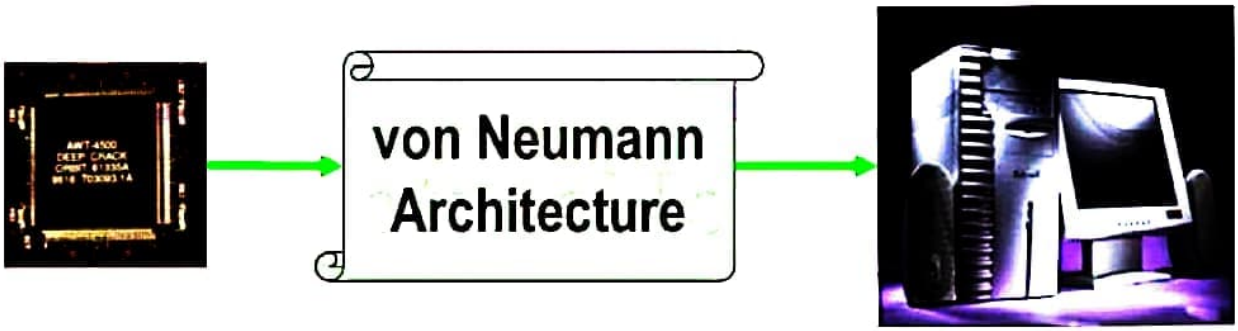
Special Case

3. When E ’ = 0 (-127+127) and
   Mantissa fraction (M) ≠ 0 ,
   then value is represented as Denormal Number.
4. When E ’ = 255 (-128+127) and
   Mantissa fraction (M) ≠ 0 ,
   then value is represented as Not a Number (NaN).

Atul Kumar
7777888822

Computer Organization I

Lecture 3: von Neumann Architecture (Part I)



Atul Kumar
7777888822

- ✓ General Architecture of von Neumann Machine
  - Memory Subsystem;
  - Arithmetic Logic Unit;
  - Control Unit;

Atul Kumar
7777888822

- ✓ Understand how the von Neumann machine works
- ✓ Understand the main functions of components included in von Neumann architecture

von Neumann Architecture - why it is important?

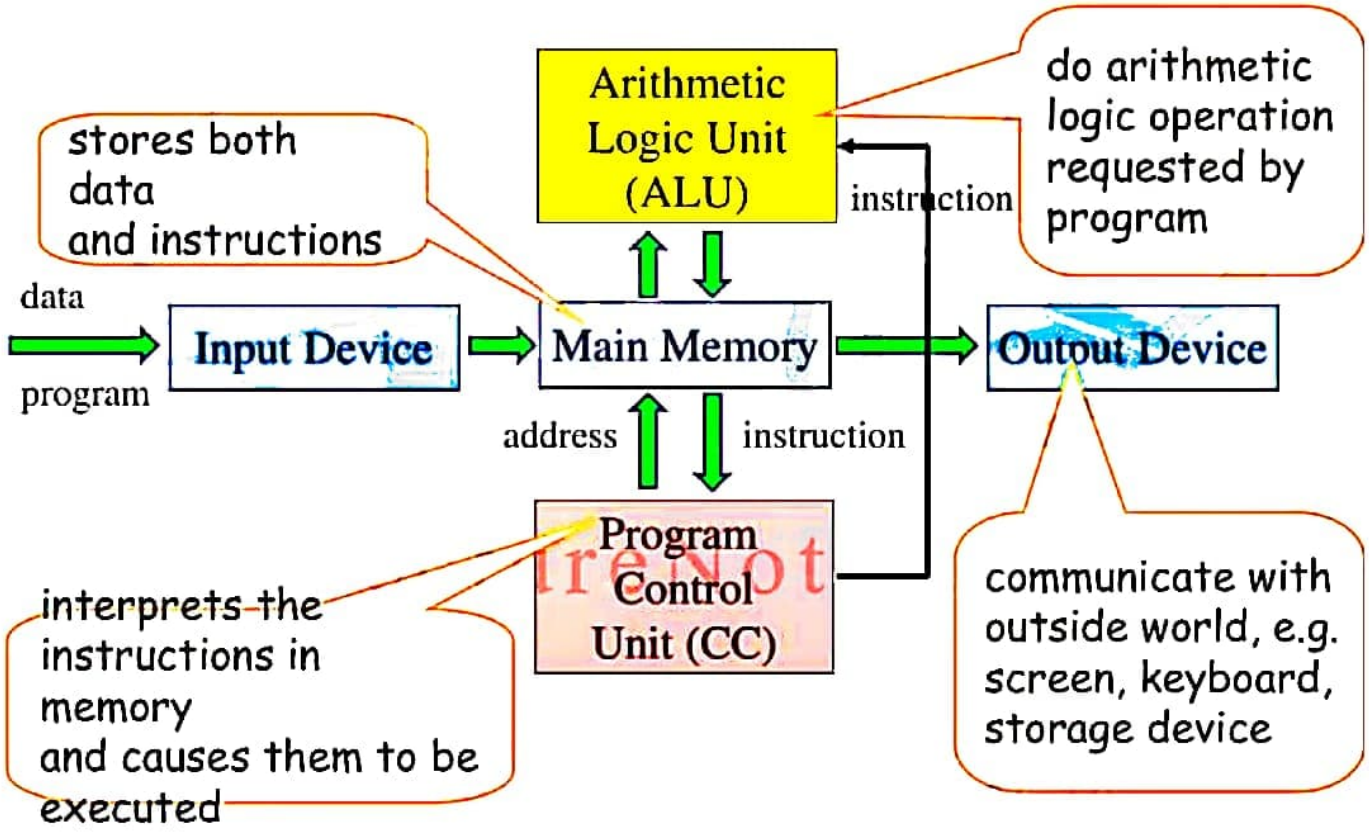
- All computers more or less based on the same basic design, the von Neumann Architecture! what ever it be a multi-million dollar mainframe or a Palm Pilot.




- The von Neumann architecture describes a general framework, or structure, that a computer's **hardware**, **programming**, and **data** should follow.

Atul Kumar
7777888822

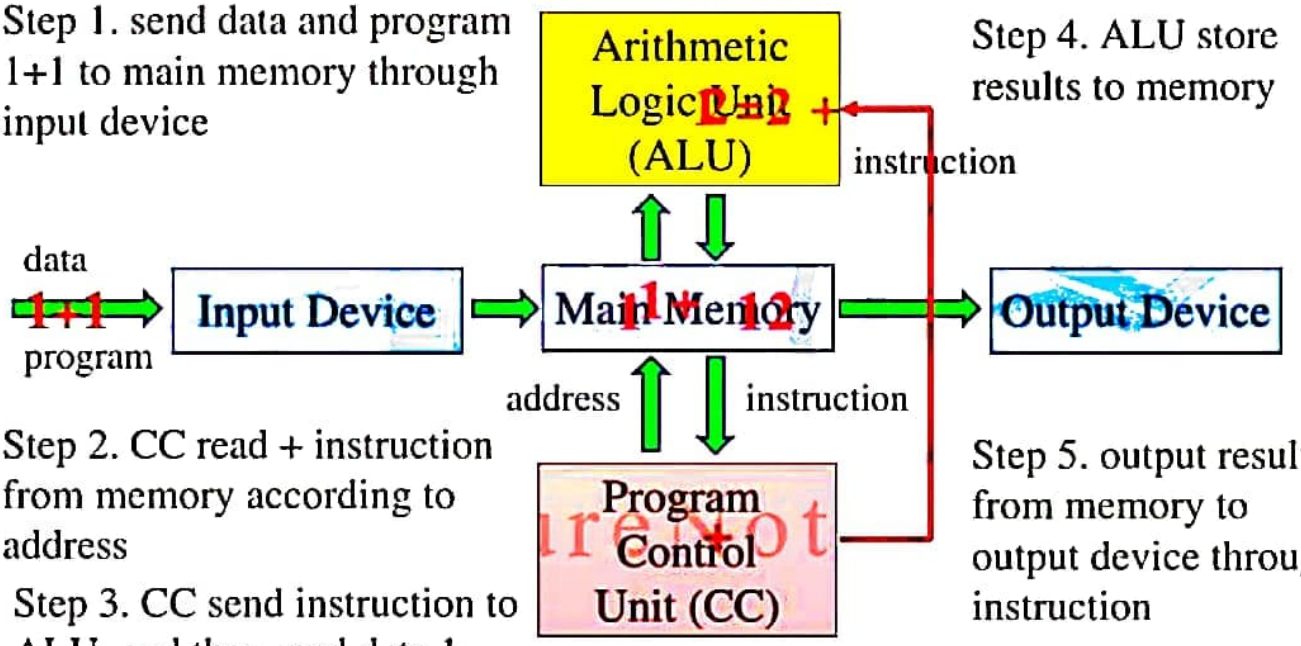
von Neumann Architecture
- what is von Neumann architecture?



Atul Kumar
7777888822

von Neumann Architecture
- how von Neumann computer works?

Step 1. send data and program 1+1 to main memory through input device



Step 4. ALU store results to memory

Step 2. CC read + instruction from memory according to address

Step 3. CC send instruction to ALU, and then read data 1 and 1 from memory to ALU according to address

Step 5. output results from memory to output device through instruction

Atul Kumar
7777888822

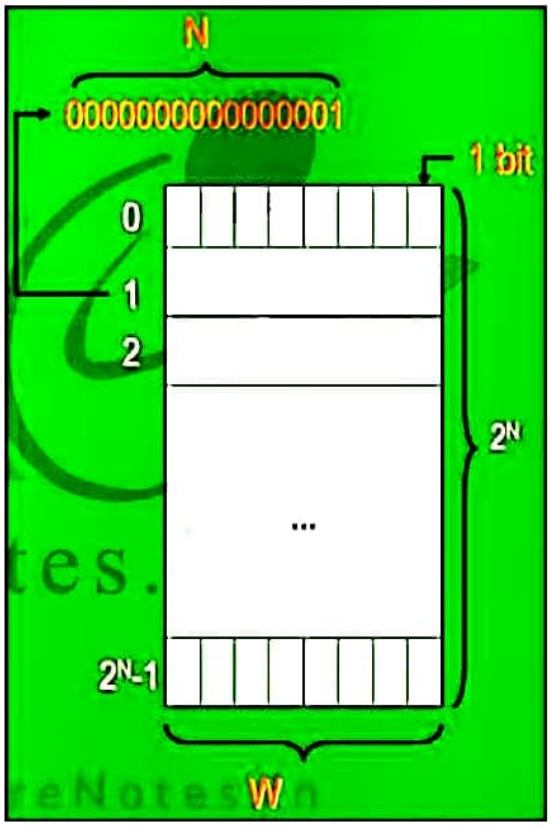
von Neumann Architecture - what is memory subsystem?

- **Memory, also RAM (Random Access Memory)**
  - Consists of many memory cells (storage units) of a fixed size.
    Each cell has an address associated with it: 0, 1, …
  - Each cell has two important characteristics:
    (1) its address (where it is),
    (2) its contents (what's stored at the given location).

Atul Kumar
7777888822

von Neumann Architecture - what is memory subsystem?

- Memory width (W)
  - How many bits is each memory cell, typically one byte (=8 bits)
- Address width (N)
  - How many bits used to represent each address, determines the maximum memory size = address space
  - If address width is N-bits, then address space is 2^N (0,1,…,2^N-1)
- Address space
  - the number of uniquely identifiable memory locations.



Atul Kumar
7777888822

von Neumann Architecture - what is memory subsystem?

For example, for a 256 Mbyte memory, suppose its memory width is 8 bits (or 1 bytes)

What is its address space?

2^{28}

What is its address width?

28

Atul Kumar 7777888822

von Neumann Architecture - what is memory subsystem?

For another example

a memory space is as follows:

000	
001	
010	
011	
100	0000 0110
101	
110	0000 0010
111	

- what is its memory width?
  8 bits
- what is its address space?
  8 bytes
- what is its address width?
  3
- what is the content of memory location address (4 = 100)?
  (6 = 0000 0110)

Atul Kumar
7777888822

von Neumann Architecture - what is memory subsystem?

Measurements about RAM Size & Speed

- **Memory sizes:**
  - Kilobyte (KB) = 2^{10} = 1,024 bytes ~ 1 thousand
  - Megabyte(MB) = 2^{20} = 1,048,576 bytes ~ 1 million
  - Gigabyte (GB)= 2^{30} = 1,073,741,824 bytes ~ 1 billion
- **Memory access time (read from/ write to memory)**
  - 50-75 nanoseconds (1 nsec. = 0.000000001 sec.)
- **RAM is:**
  - volatile (can only store when power is on)

von Neumann Architecture - what is memory subsystem?

Operations on Memory

- Fetch (address):
  - Fetch a copy of the content of memory cell with the specified address.
  - Non-destructive, copies value in memory cell.
- Store (address, value):
  - Store the specific value into the memory cell specified by address.
  - Destructive, overwrites the previous value of the memory cell.

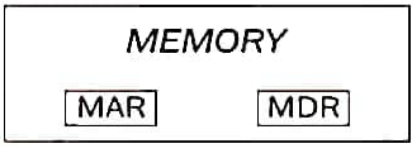
von Neumann Architecture - what is memory subsystem?

Interface to Memory

- How does processing unit get data to/from memory?

  MAR: Memory Address Register

  MDR: Memory Data Register



- Fetch (Address):
  1. Load the address (A) into the MAR.
  2. Copy the content of memory cell with specified address into MDR.
- Store (Address, Value):
  1. Load the address into MAR; load the value into MDR
  2. copy content of MDR into memory cell with specified address.

Atul Kumar
7777888822

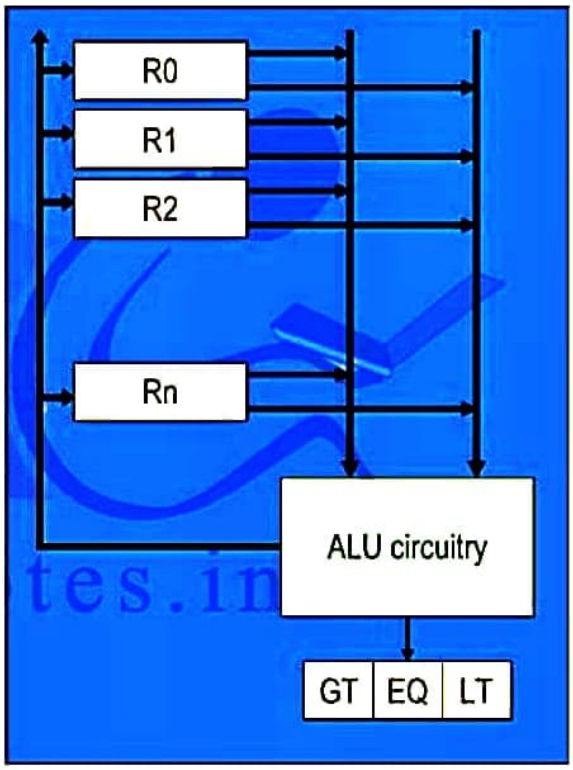
von Neumann Architecture - what is ALU subsystem?

- The ALU (Arithmetic/Logic Unit) performs
  - mathematical operations (+, -, x, /, ...)
  - logic operations (=, <, >, and, or, not, ...)
- In today's computers integrated into the CPU
- Consists of:
  - Circuits to do the arithmetic/logic operations.
  - Registers (fast storage units) to store intermediate computational results.
  - Bus that connects the two.

Atul Kumar
7777888822

von Neumann Architecture - what is the structure of ALU subsystem?

- Registers:
  - very fast local memory cells, that store operands of operations and intermediate results.
  - CCR (condition code register), a special purpose register that stores the result of <, = , > operations
- ALU circuitry:
  - Contains an array of circuits to do mathematical/logic operations.
- Bus: Data path interconnecting the registers to the ALU circuitry.



Atul Kumar
7779888822

von Neumann Architecture - what is control unit subsystem?

- Program is stored in memory
  - as machine language instructions, in binary

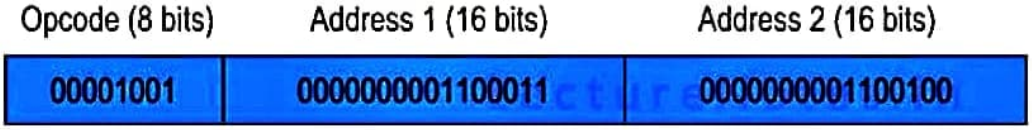
- The task of the control unit is to execute programs by repeatedly:
  - Fetch from memory the next instruction to be executed.
  - Decode it, that is, determine what is to be done.
  - Execute it by issuing the appropriate signals to the ALU, memory, and I/O subsystems.
  - Continues until the HALT instruction

Atul Kumar
7777888822

von Neumann Architecture - what is control unit subsystem?

Machine Language Instructions

- A machine language instruction consists of:
  - Operation code/opcode, telling which operation to perform
  - Address field(s)/operands, telling the memory addresses of the values on which the operation works.
- For Example: ADD X, Y (Add content of memory locations X and Y, and store back in memory location Y).
- Assume: opcode for ADD is 9, and addresses X=99, Y=100



Atul Kumar
7777888822