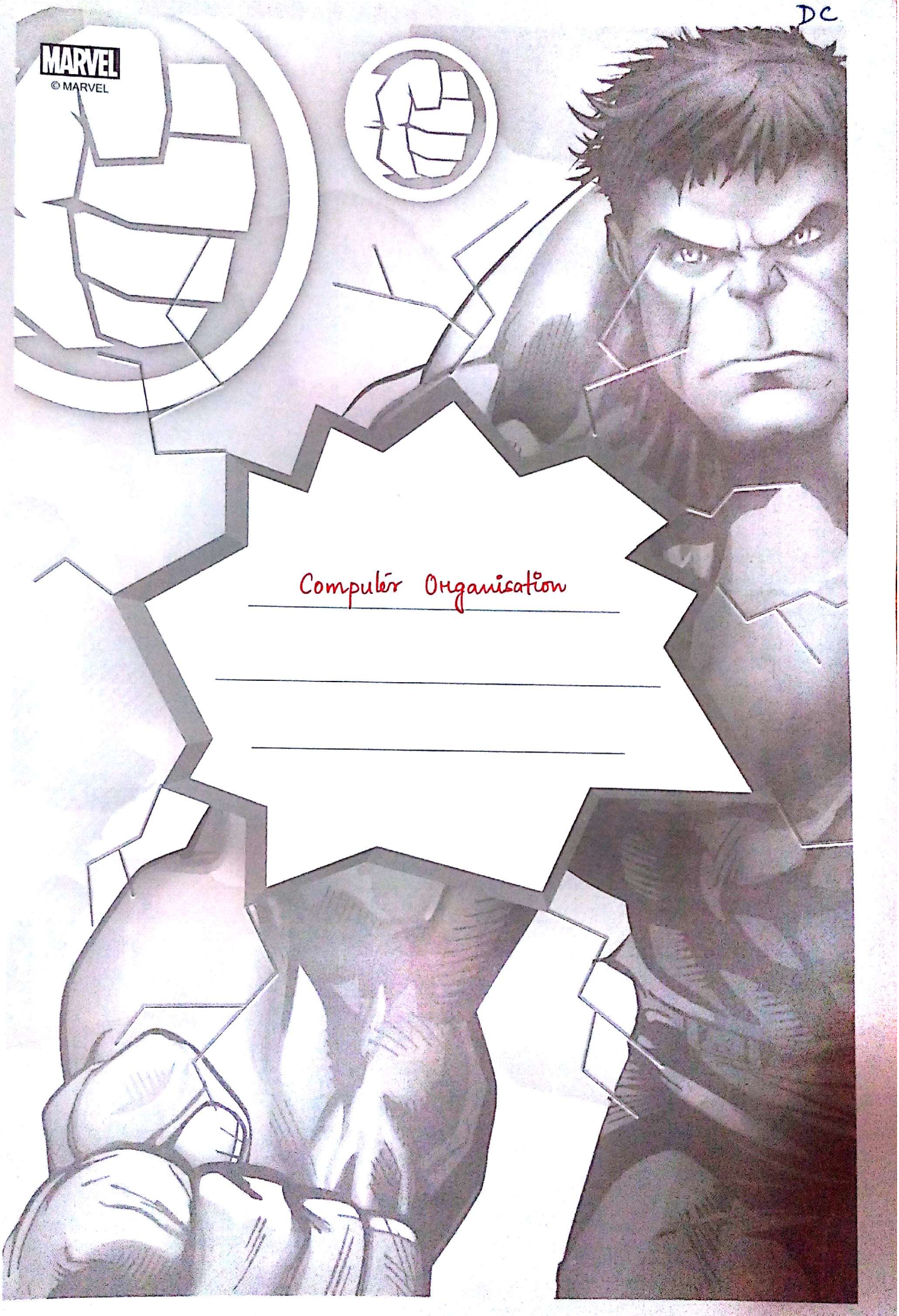
DC



Computer Organisation

Scanned by CamScanner

11th August, 2017

Analog signal

- Consists of not discrete values, but of continuous values.

Sampling of a Analog signal is the digitisation of an analog signal into multiple discrete intervals, say at time t (where $t = t_2 - t_1 = t_3 - t_2 \ldots$)

Digital signal

- Consists of discrete mathematical functional output

* The Components Of Modern Digital Computer

a. Human Resources

- Naive
- Application programmers
- S/W designers
- System Administrator

b. Software

Def^n Instructions : a collection of valid operators and operands

Def^n Program : a set of instructions which is passed into or from the device

Def^n software : a set of programs which perform interrelated task.

Types of software

- System → Interacts with hardware
  eg : OS
- Application → Interacts with user
  eg : MS office

Scanned by CamScanner

1st September, 2017

Classification of instructions

Based on number of fields

a. 3-Address instruction (3 Fields)
b. 2- " " (2 Field)
c. 1- " " (1 field)
d. 0- " " (zero of one field)

- op-code field
- Addressing mode field
- operand field (operand location field)

assuming that all the instructions are in ASSEMBLY LEVEL language.

a. 3-Address instruction

Let $X = (A+B) - (C+D)$ be an arithmatic equation. Represent this in 3-Address instruction format where X, A, B, C & D are the memory addresses.

```
ADD   R1, A, B
ADD   R2, C, D
SUB   X, R1, R2
```

b. 2-Address instruction

$X = (A+B) - (C+D)$

```
LOAD   R1, A
ADD    R1, B
LOAD   R2, C
ADD    R2, D
SUB    R1, R2
STORE  X, R1
```

Scanned by CamScanner

c. 1- Address Instruction

Accumulator: It is a special type of register. This uses 1- Address instruction. Before binary operations, one of the data is stored within the Accumulator. After the operation, the result is by default stored in the Accumulator.

$X = (A+B) - (C+D)$

```
LOAD  C
ADD   D
STORE T1
LOAD  A
ADD   B
SUB   T1
STORE X
```

d. 0- Address Instruction

While performing a binary operation, the two topmost elements are popped, operated; and the result is pushed back into the stack.

$X = (A+B) - (C+D)$

```
PUSH D
PUSH C
ADD
PUSH A
PUSH B
ADD
SUB
POP  X
```

Scanned by CamScanner

ADDRESSING MODES & ITS CLASSIFICATION :

- Different ways to specify the operand (or operand location) within the instruction definition.

- Example : ADI 05 → It is an 1-Address instruction in ASSEMBLY LANGUAGE

↓ implies

[ACC] ← [ACC] + 05

CLASSIFICATION

1. Memory Addressing Mode (Direct & Indirect)
2. Register Addressing Mode (Direct & Indirect)
3. Implied / Implicit
4. Immediate
5. Stack Addressing Mode
6. Auto-increment (Pre & Post)
7. Auto-decrement (Pre & Post)
8. Relative Addressing Mode
9. Base Register Addressing Mode
10. Index " " "

Scanned by CamScanner

① Memory Direct Addressing Mode (load/store instruction)

Use :

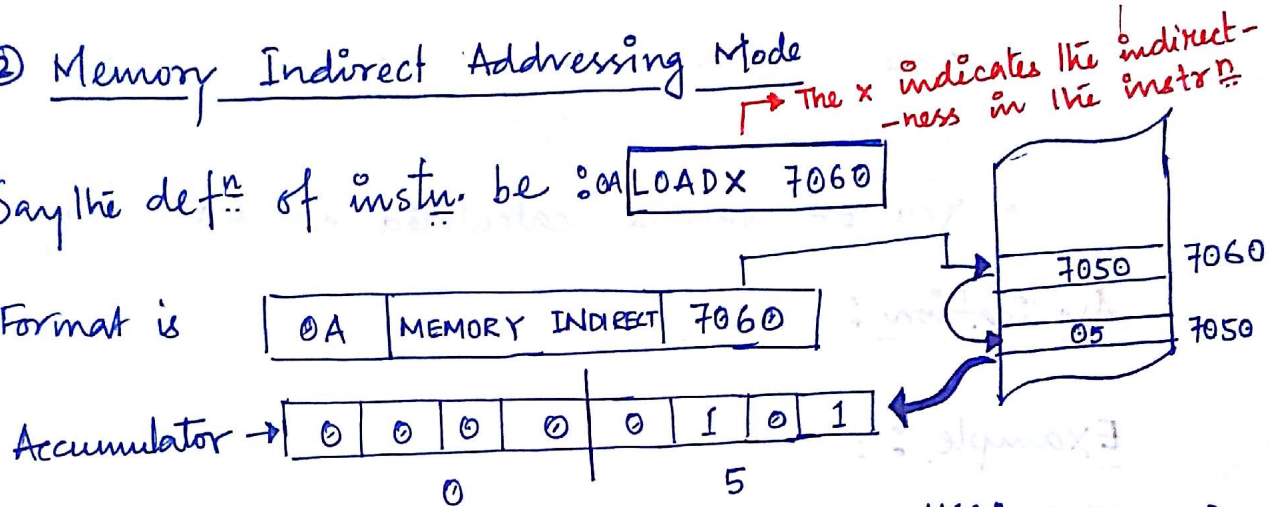
Say the defⁿ of instⁿ be : 0A | LOAD 7050 | [Acc] ← M[7050]

↳ called effective address of the data / original address of data

② Memory Indirect Addressing Mode

→ The x indicates the indirect-ness in the instrⁿ

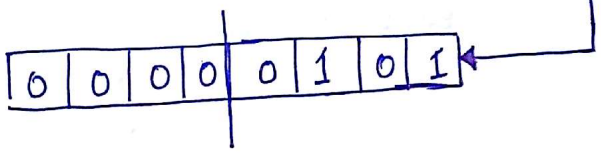
Say the defⁿ of instⁿ be : 0A | LOADX 7060 |



③ Register Direct Addressing Mode (load/store; Arith/Logical)

Use :

: 0B | ADD R1 |



④ Register Indirect Addressing Mode

: 1F | LOADX R1 |

EA = [R1] of operand

Scanned by CamScanner

~~Aug~~ 4th September, 2017

☐ BASE REGISTER ADDRESSING MODE :

Definition :

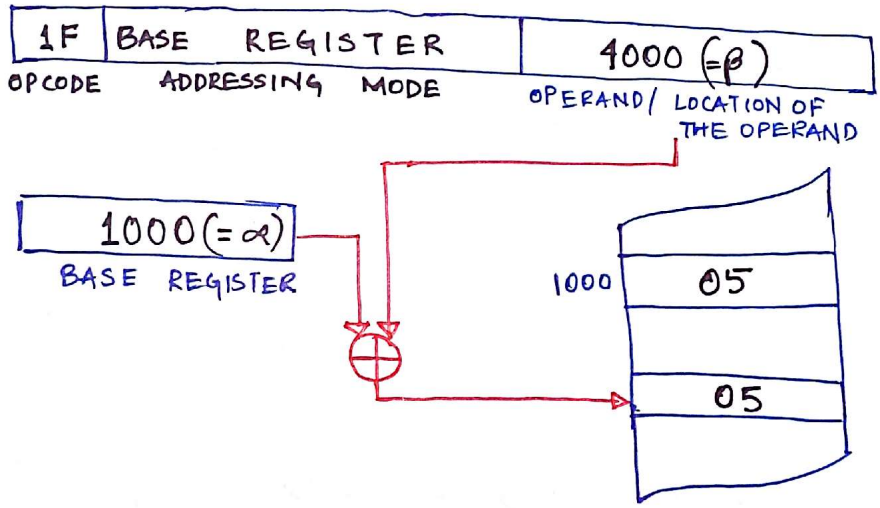
- Instruction definition holds the offset/displacement value (β)
- Base register holds the EA (effective address) of the operand/data (α)
- New EA can be calculated as $EA = \alpha + \beta$

Application : Relocation of Memory

Example : 1

LOAD 4000

Here, in LOAD 4000, 4000 is not the data but the Memory Location of the operand.



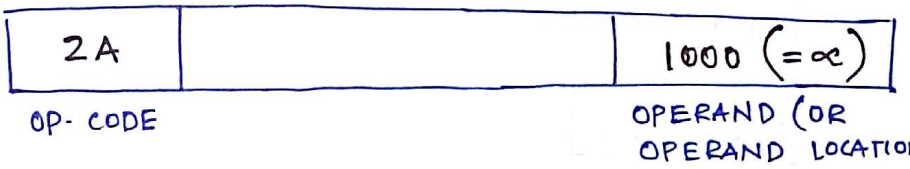
Scanned by CamScanner

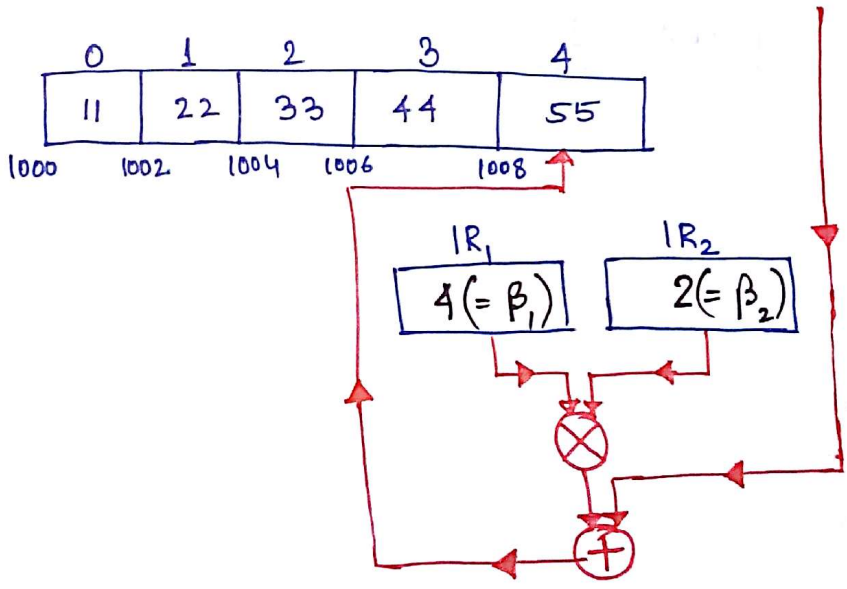
□ INDEX REGISTER ADDRESSING MODE :

- **Definition :**
  - One index register holds the index number of the array. (β_1)
  - Another index register holds the displacement/step size of the data (β_2). numerically = sizeof(data type)
  - Instruction definition holds the Base Address of the array. (α)
  - New EA = $\alpha + \beta_1 \beta_2$

- **Example :**

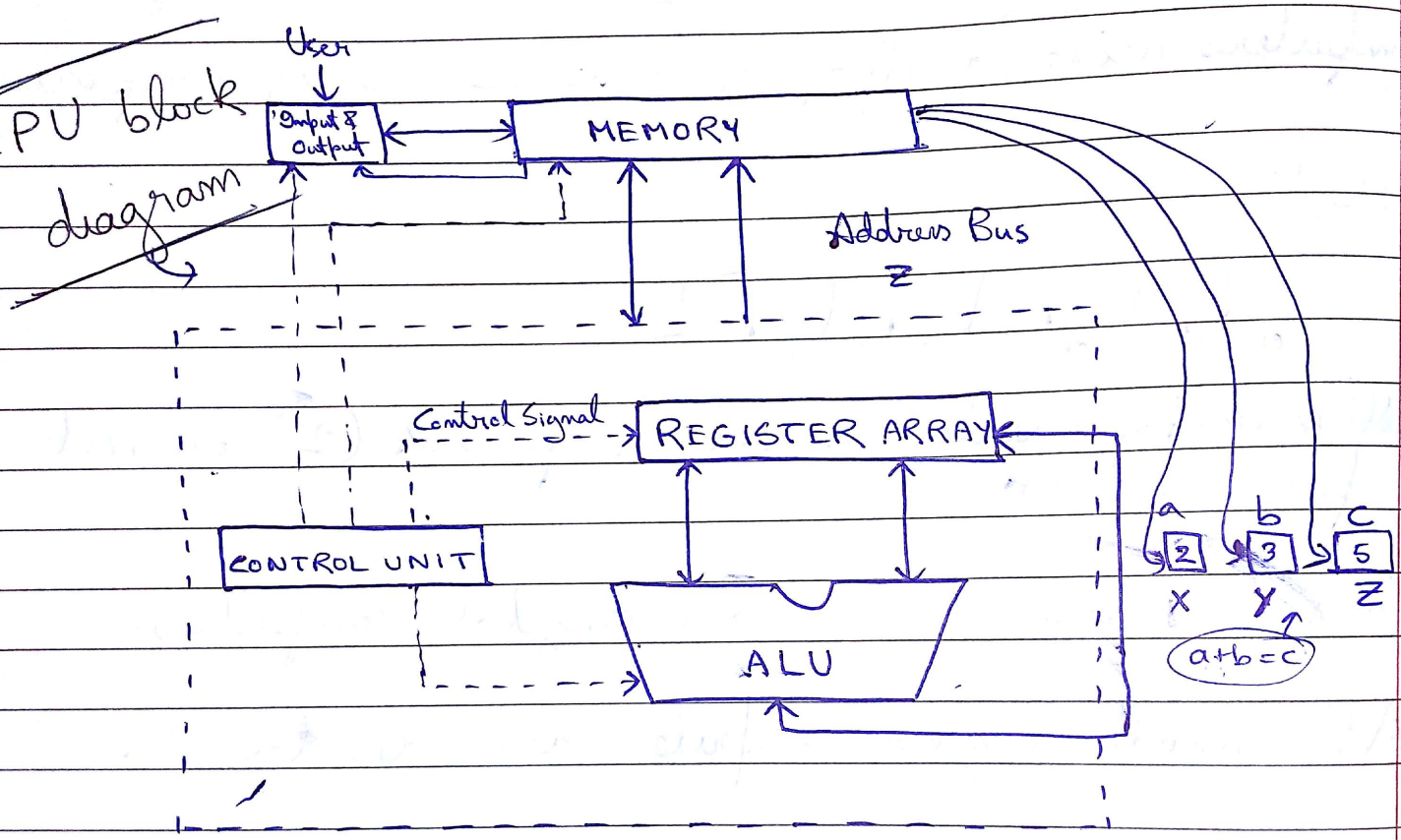
STORE 1000 2A →





Scanned by CamScanner

CPU block diagram



Every instruction is identified by a code that is known as op-code (operational code). Op-code is signalled from CU to ALU. By this op-code they know which operation is should be performed.

Scanned by CamScanner

* Compiler lists all the errors of a program in a stack (LIFO). So it needs extra memory. But interpreter checks a program line by line. So until the error in a line is not corrected, the interpreter will not execute the next line. So, interpreter needs less memory coz stack is not needed.

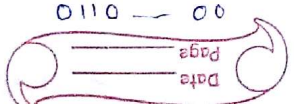
Compiler	Interpreter
(iii) Compiler is slower than interpreter	(iii) Interpreter is faster than the compiler.
(iv) 3rd gen. and 4th gen languages	(iv) 1st gen & 2nd gen. languages

→ TURBO C++ behaves as a compiler when (Alt + F9) is pressed
" " " " interpreter " F8 is pressed.
So, it works both as a compiler and interpreter.

• Machine Level Vs. Assembly Level Vs High Level

Machine	Assembly	High
(1) Understandable by hardware.	(1) Understandable by assembler (is a collection of system software programs which are machine dependent)	(1) Understandable by user only.
(2) Dependent (on hardware)	(2) Dependent (on machine) (+ve)	(2) Machine independent
(3) Faster than assembly and high level (combination of binary codes and digital logic)	(3) Consts of pnemonics or alphanumeric codes, which require time to be converted to binary code. Hence it is slower than machine level codes	(3) High level language is slower than both, as it converts to .ASM file and then machine level files.
(4) Binary Codes	(4) Alphanumeric codes or pnemonics	(4) c = a+b;

0110 — 00



.ADD, C, A, B
M[C] = M[A] + M[B]
[ ] - content

Scanned by CamScanner

IMMEDIATE ADDRESSING MODE

- **Definition :** If the operand (or one of the operands) is specified explicitly within the instruction definition then it is known as immediate addressing mode.
- **Applications:** Helps us perform Arithmatic, logical and shift operations.
- **Example :**

```
ADI 05
ORI 05
MVI 07
CIRA 01
```

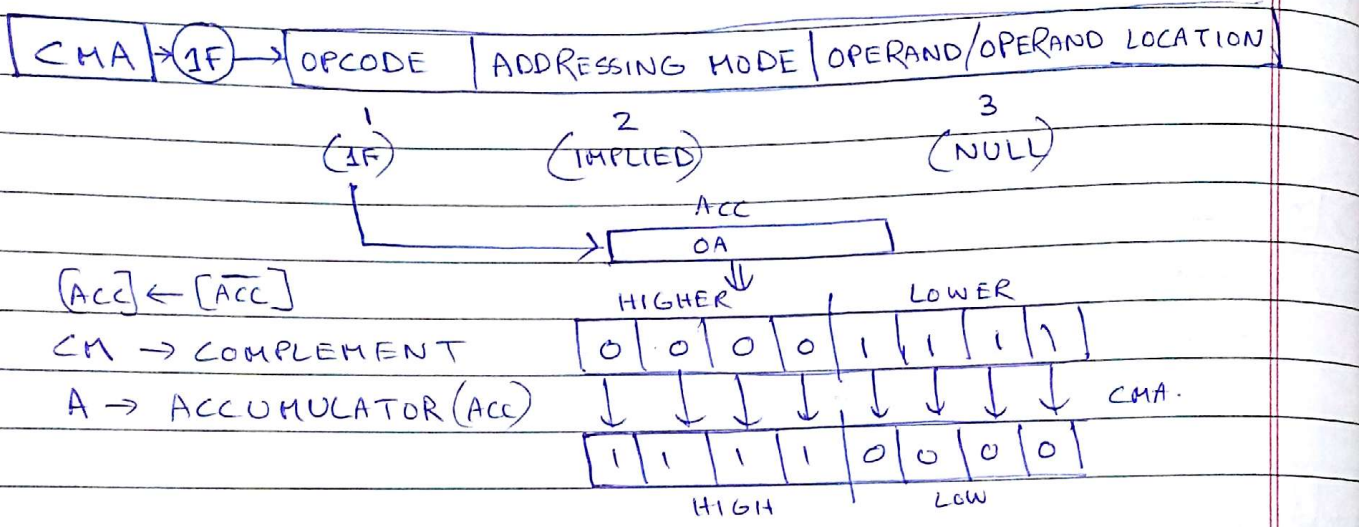
Scanned by CamScanner

IMPLIED ADDRESSING MODE

Defⁿ:- When operand is not explicitly specified in the instruc" defⁿ. (CMA)

Applicatⁿ:- 9ⁿ Accumulator-Based-CPU-Organisation; 9ⁿ shift, complement operations. (RAL) (RAR)

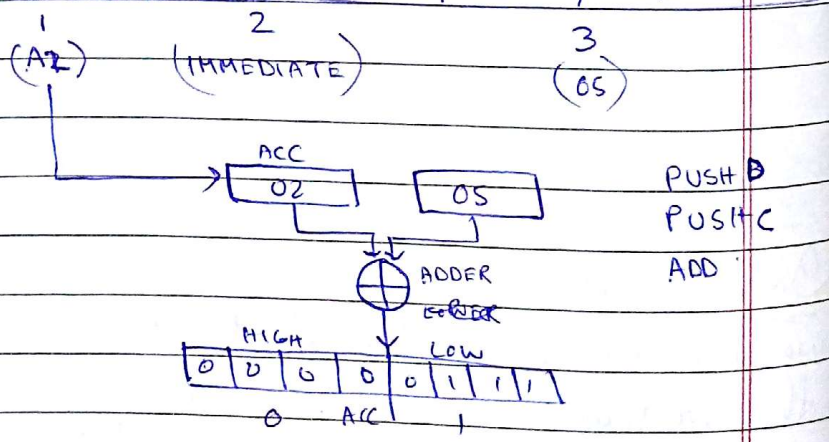
Example:-



IMMEDIATE ADDRESSING MODE

- **Defⁿ:-** When operand is specified explicitly ~~in~~ with the instruc" defⁿ.
- **Application:-** 9ⁿ arithmatic (ADI 07), Logical (ANI OR); load/store (MVI 05) inst

Example:- ADI 05 → A2



Scanned by CamScanner

AUTO INCREMENT ADDRESSING MODE

Defⁿ:- When EA (eff. address) of one of the operand is used in the instructⁿ is stored in a register and each time register content is updated by step size (d), when we try to access the register.

Example:-

ADD R1, (R2)+ → (A2) → | OPCODE | ADDRESSING MODE | OPERAND/OPERAND LOCATION |

1 2 3

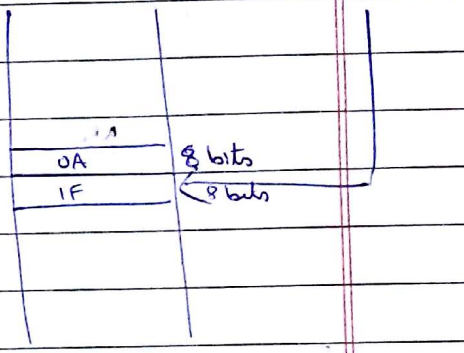
(A2) (POST AUTO INR) (R2)

$M[EA] \leftarrow [R_2]$

$[R_1] \leftarrow [R_1] + M[EA]$ $[R_2] \leftarrow [R_2] + d$

$d = 2$ Byte $\leftarrow 1000 + 2$

$EA = 1000$ $\leftarrow 1002$



11.09.17

- **Instruction Cycle:-** A method to execute or complet an instructⁿ by means of several stages / subcycles is calle instructⁿ cycle.

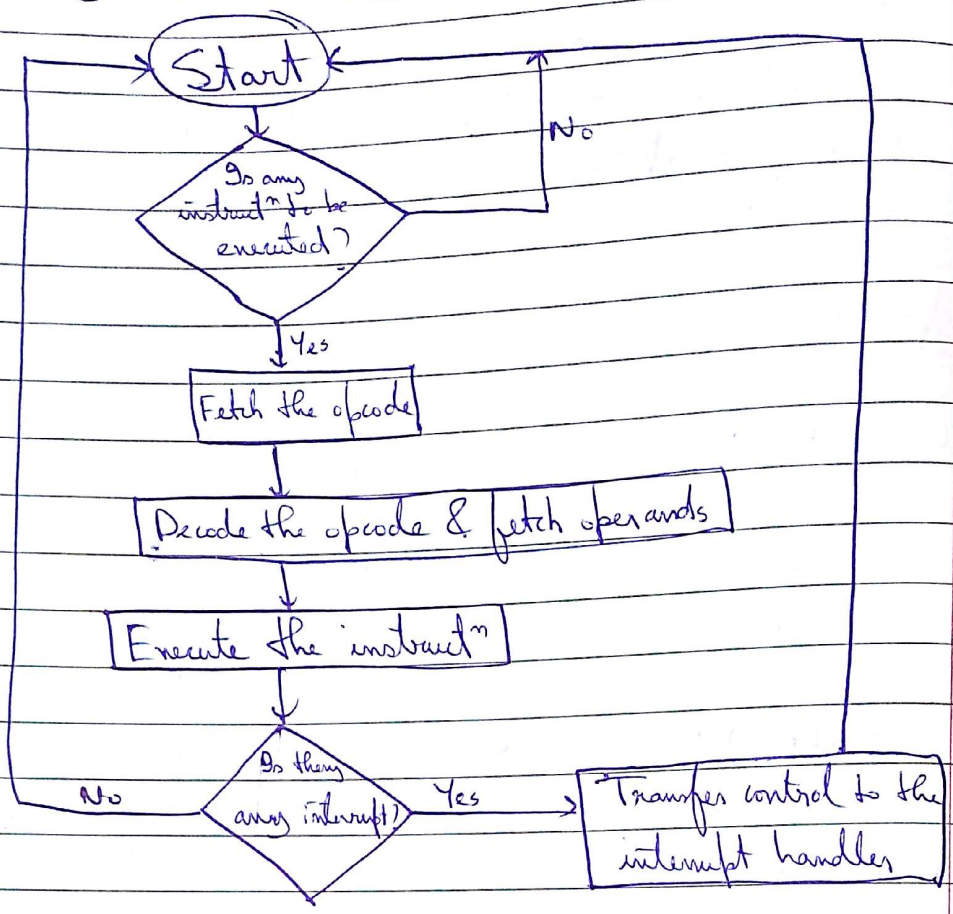
- **Instructⁿ Cycle's Subcycles:-**

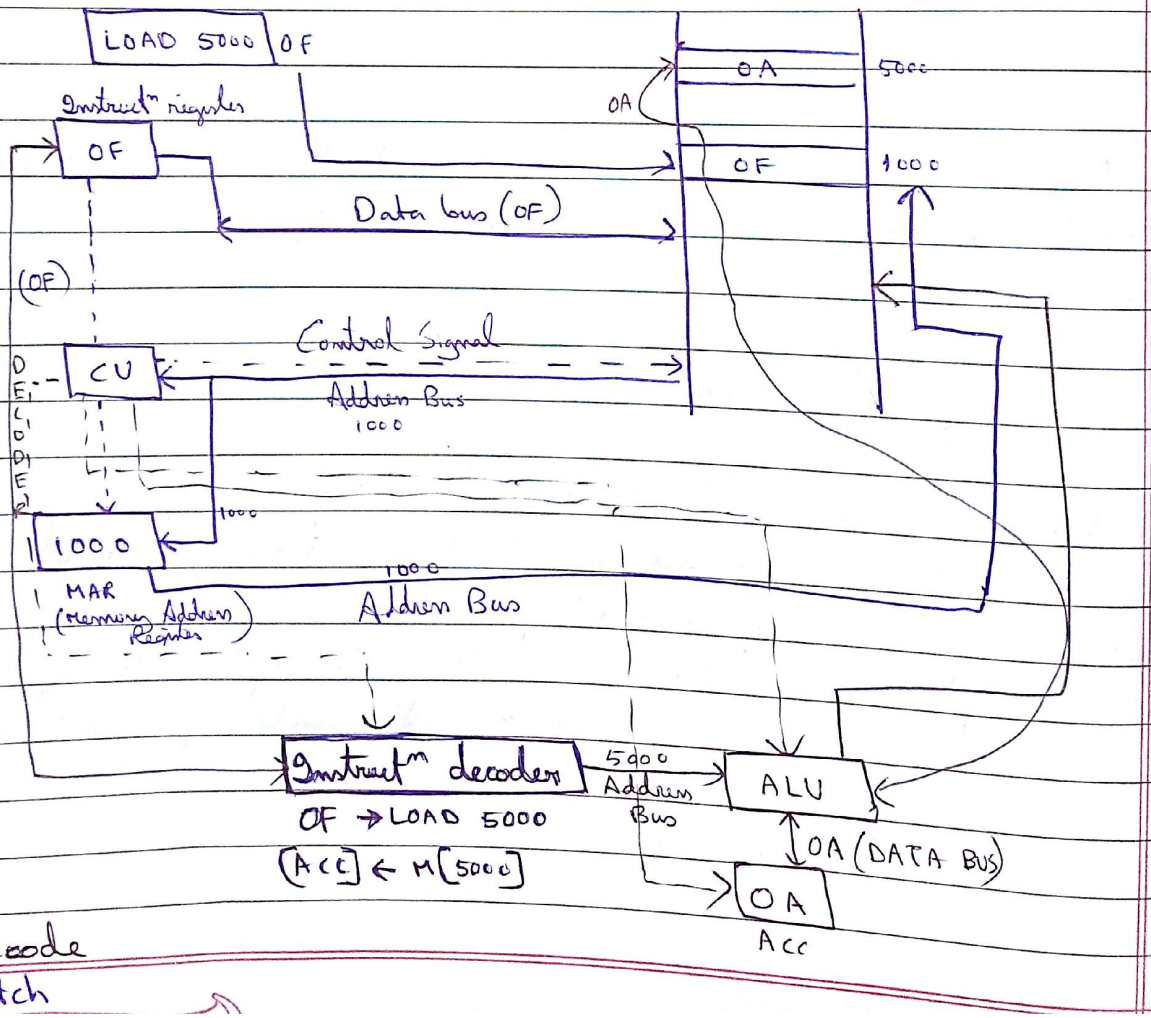
(a) Fetch subcycle (Opcode fetch)
(b) Decode " (Analysing opcode & fetching operand)
(c) Execute " (operatⁿ performed)
(d) Interrupt " (Entertain the interrupt/interrupt is handled by the interrupt handler)

(P.T.O)

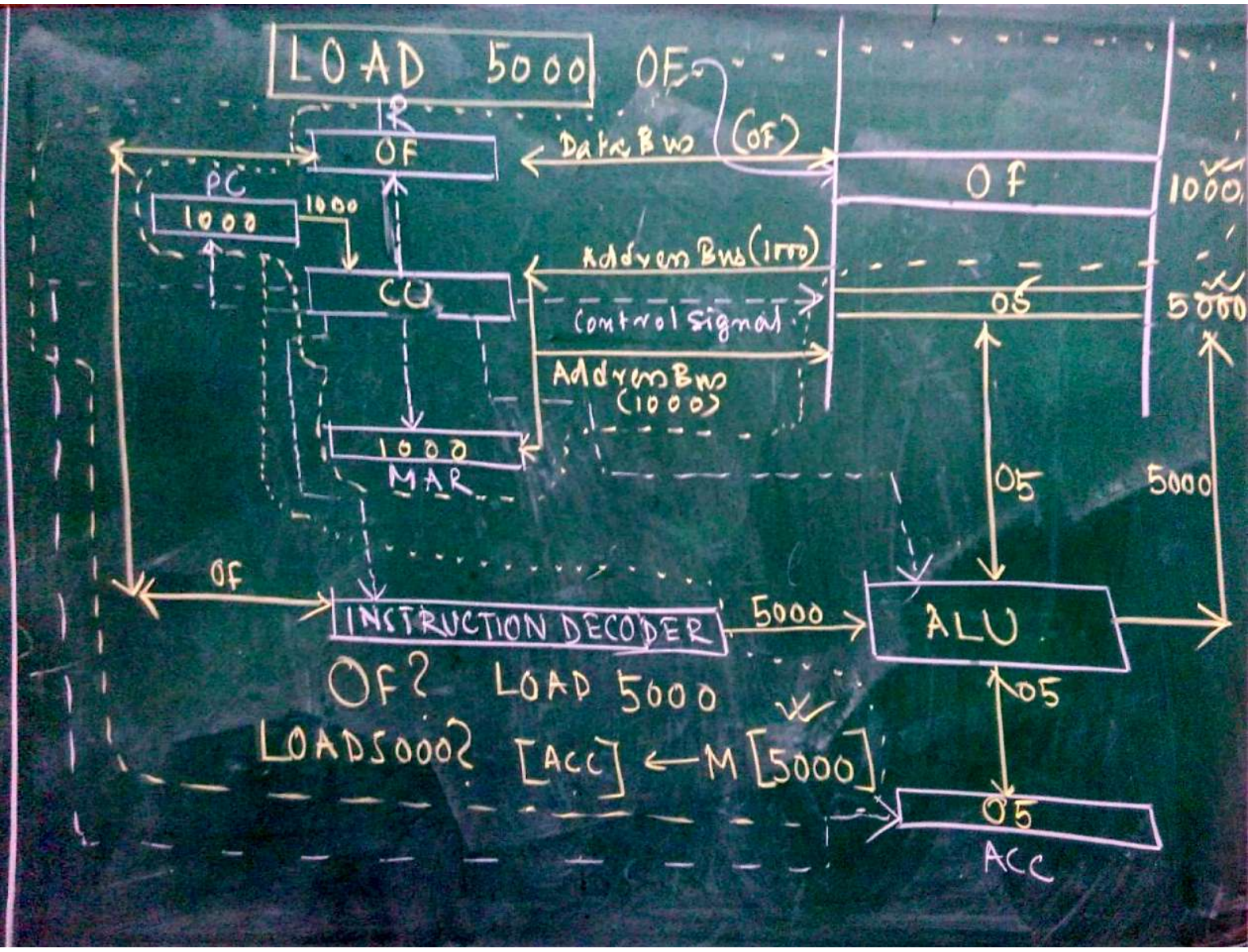
Scanned by CamScanner

Flow chart for instruct^{n} cycle:-





Scanned by CamScanner



Scanned by CamScanner

15/9/17

• VON NEUMANN ARC OR PRINCESTONE ARC

• Points: (i) Developed by Von Neumann and his associates in 1946
(ii) " in Princestone University (US)
(iii) Proposed stored program concept.

• CPU :-
Divided into 2 parts - (a) Data Processing Unit or Data Unit path ((AC + MQ + DR) + ALC) + Address & Data Bus

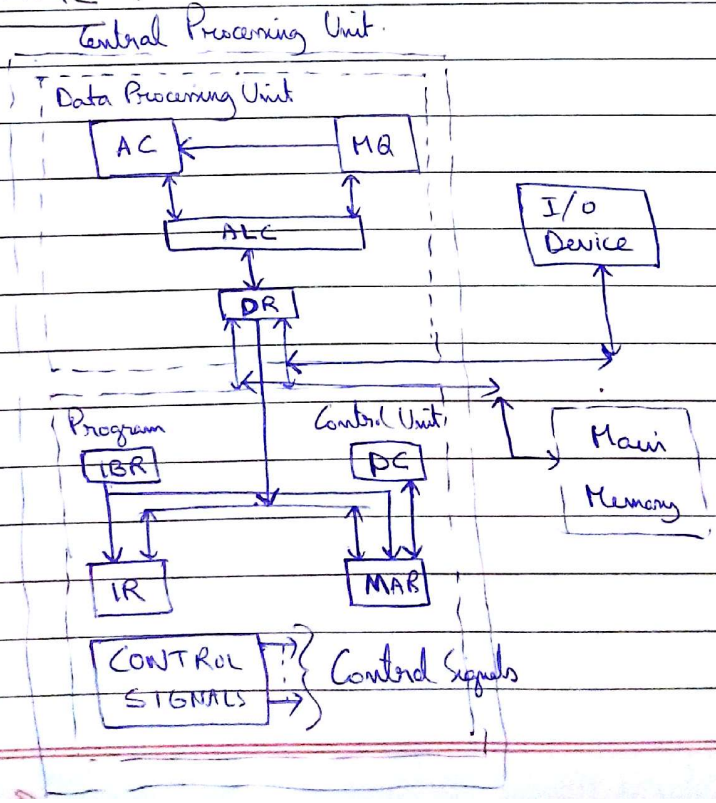
(b) Program Control Unit & or Control Unit ((IBR + MAR + PC + IR + Control ckt)

• Instruct^n format :- [OPCODE (19–12) ADDRESS (11–0)] 20 bits.

• Data :- [SIGN (39) MAGNITUDE (38–0)] 40 bits

→ CPU is made up of high speed registers and VT(S) (Vacuum Tube)

• ISA ARC :-



Data path → ALU + CPU registers

ALC → older version of ALU

IBR → Instruct^n Buffer Register.

IR → Instruct^n Register

MAR → Memory Address Register

PC → Program Counter

Scanned by CamScanner

64
KB
256
3840
960

• Memory :-

→ Program or opcode & Data both stored in the same memory

→ Slow speed memory and high speed CPU i.e speed mismatch b/w memory & CPU. (Von Neumann Bottleneck)

→ Sol^{n} :- (a) Introduc^{n} of Cache memory.
(b) " " RISC Arc.
→ Reduced Instruct^{n} Set Architect

→ Memory is random acces type.

→ Made of with CRT(S) (Cathode Ray Tube)

→ Total no. of addresses in memory = $2^{12} \approx 4096$.

• Harvard Architecture

Read from book.

Scanned by CamScanner

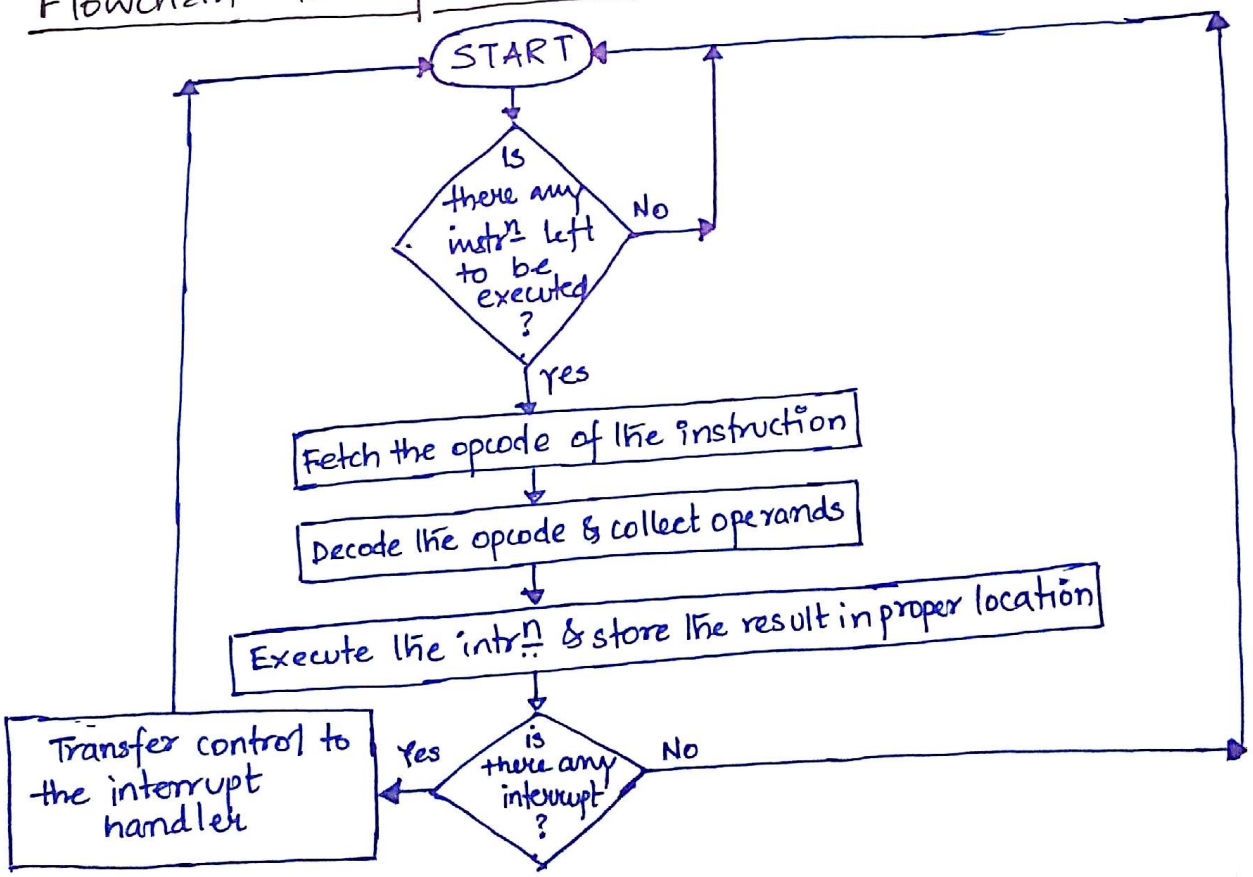
11th September, 2017

INSTRUCTION CYCLE

- Defn : A set of steps through which an instruction can be completed/executed.

- Subcycles of instruction cycle :
  a) Fetch subcycle (Fetch opcode)
  b) Decode subcycle (To analyse which instruction is to be executed and which operands are needed.)
  c) Execute subcycle (To perform the operation in ALU & to store the proper output in memory)
  d) Interrupt subcycle (Whenever there is an interrupt, the control is handed over to the interrupt handler → solving an interrupt)

- Flowchart to represent instruction cycle



Scanned by CamScanner