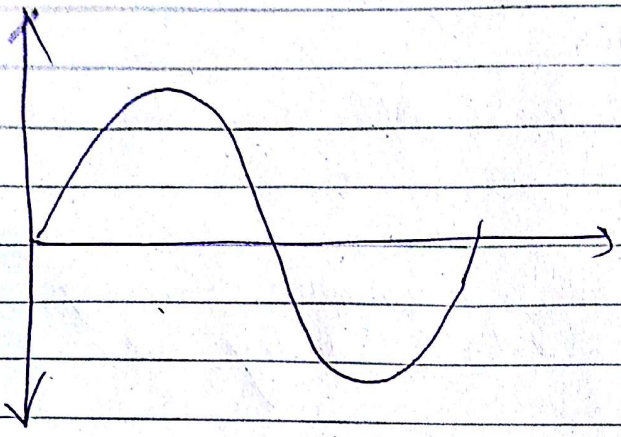
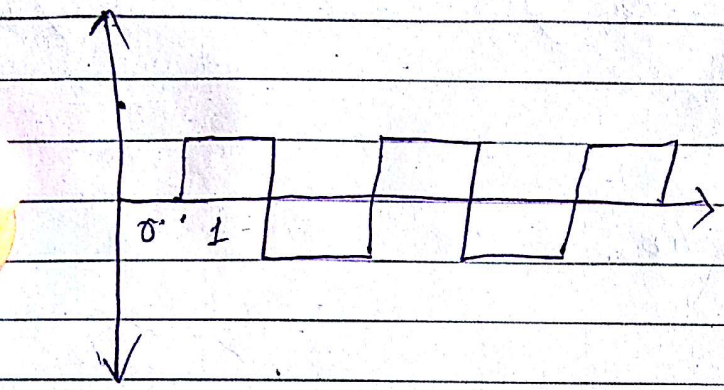
PREVIEW



Analog signals have variation of voltages



Digital signals have either voltage high or no voltage

Components of Modern digital computer :-

a) Human Resources → NAIVE.
b) Softwares → Application programmer
c) Hardwares. → Software designer
→ System administration

- Naive who have no knowledge about how system is operating.
- Application programmers → those who work in MNCs and developing JAVA, etc.
- Software designer → those who generate a blue print of any software

- System administrator → person responsible for organising the whole system in any organisation.
  - Eg :- Data Base administrator (DBA)

(b) Softwares :-

- Instructions → Set of operations & operands.
- Program → A set of instructions
- Software → A set of programs.

Types of softwares :-

System software → which help us to interact with hardwares - Eg OS.

Application " → which helps us to interact with user but not directly with hardware Eg MS-office

(c) Hardware :-

(i) CPU (CU + ALU & Registers Array)
(ii) Memory (CPU register, cache, RAM, ROM)
(iii) Input & Output

- CU acts as a brain of computer system. It supervises rest of the hardwares using a control signal.

- ALU performs three types of functions —

(1) Arithmetic.
(2) logical (AND, OR, NOT)
(3) Shift (left shift, right shift)

Register arrays / CPU registers :—

Special type of memory which can hold a small amt. of data temporarily. Among various types of memories, these are the smallest in size, and highest in speed.

- Memory :— It is basically designed with the help of digital logic circuits called FLIP-FLOP (F/F). It is able to hold certain data until a pulse is generated and it is sequential.

- Cache :— All modern generation microprocessors support various types of cache memory. It is greater in size than register and are faster as compared to primary and secondary memory. Its size is greater than registers but less than RAM, ROM.
It holds recently used data or instructions

- RAM / ROM . (diff) (Explanation)
- Sec. memory — Largest size, least speed
  Operated by I/O processors
  (Permanent devices).

(c) Input / Output → (Peripherals)
Communicate with main computer system (CPU & memory) via I/O processor

---

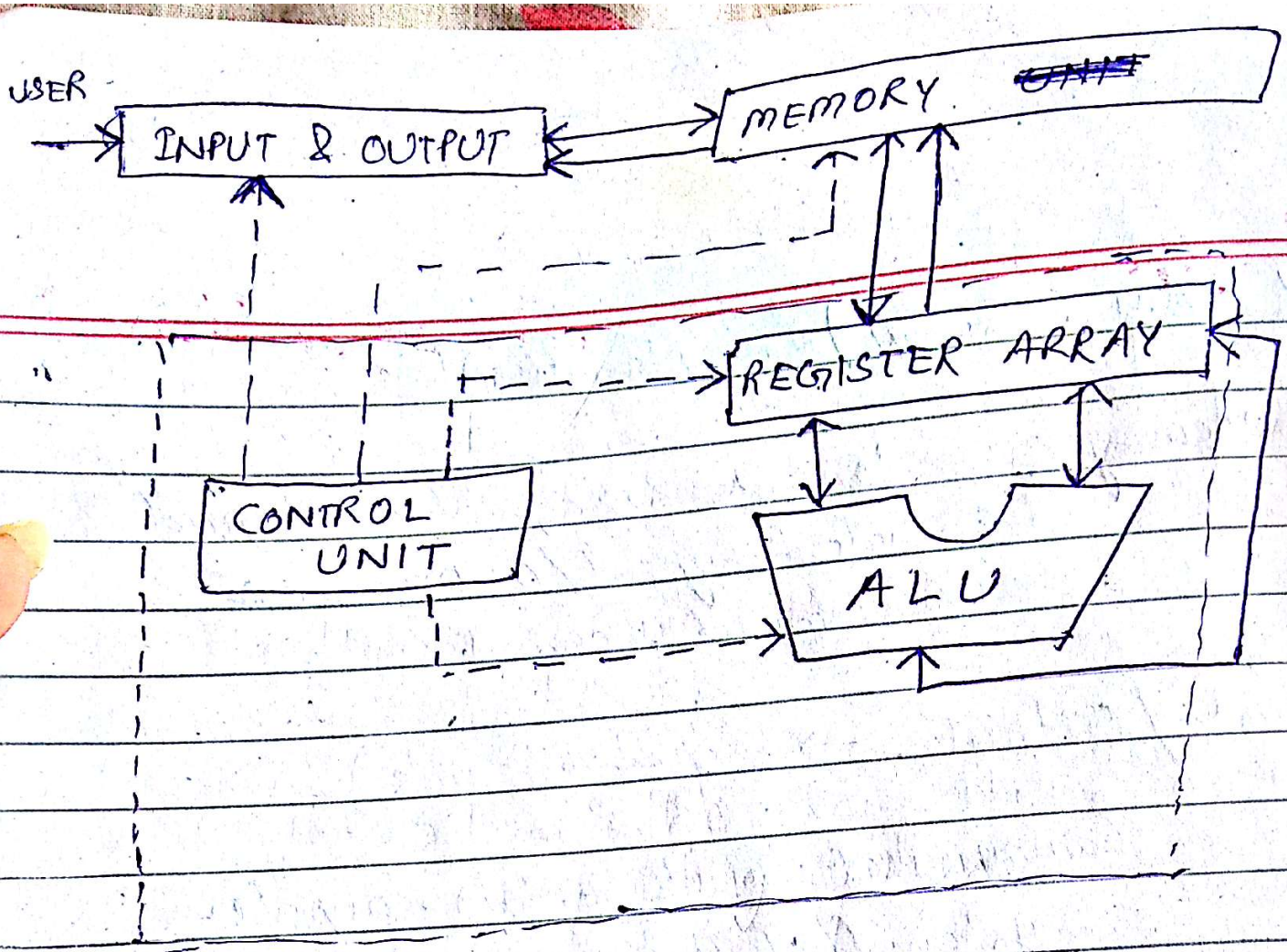
* BLOCK DIAGRAM OF A MODERN DIGITAL COMPUTER

For a block diagram to be drawn, the component must-

(1) Be internal component
(2) have a physical existence

Block diagram of hardware components is only possible bcz they satisfy booth the conditions.

Bus :- A communication channel inside a computer
They can travel →.
- Address :- Address Bus (↑)
- Data : Data bus (travels information only) (↕)
- Control signal : Control bus (transfers control signals only generated from CU) (↑)
- System bus : Travels either address / data / control signals or mixtures of them (↕)



* Sampling of an analog signal.

↑ firmware is a chip that holds BIOS. (Basic Input Output System).

- Topic 3: Compiler & interpreter.
- Topic 4: Assembley lang, higlevel lang.

D.C.

- Topic 1: Task for operating system:

(1) User & H/W (Hardware) interaction
(2) Process Management
(3) Memory "
(4) Disk "
(5) File "
(6) Protection "
(7) Input & output Management

- Topic 2: Conversion of a program into a process

C source code (P.C)
↓
preprocessor
↓
Expandable source code (P.I)
↓
Compiler
↓
Assembly code (P.ASM)
↓
Assembler
↓
Readable object code (P.OBJ)
+
Object codes of library functions
↓
Linker
↓
~~Readable~~ Executable code (P.EXE)

* A program in execution is said to be a process.
* Light weight process is called Thread (sub process)

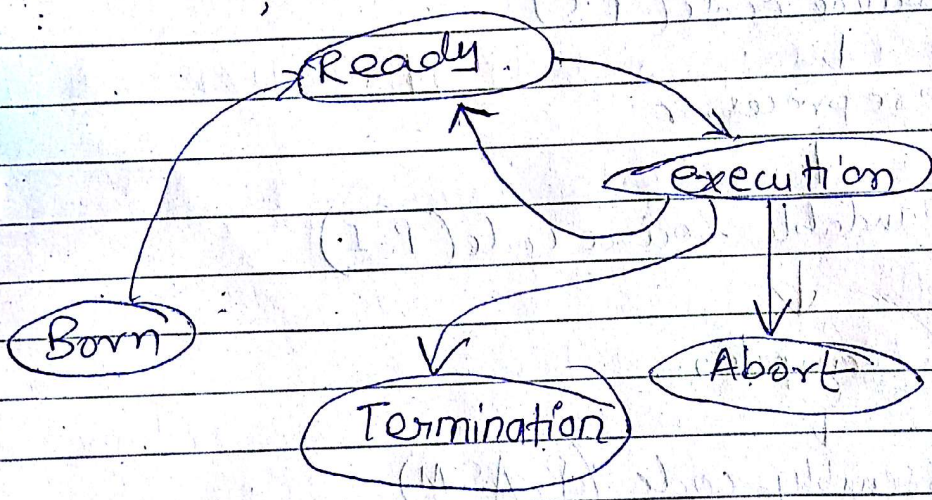
Preview:

* TOPIC 1 : PROGRAM TO PROCESS CONVERSION:

* Assembler → System software program, machine dependent.

* Loader → load the executable code into our operating system.

* 2) PROCESS STEPS:-



execution → ready (when interruption occurs but it is not strong enough)

execution to abort (when interruption occurs ~~but~~ which cannot be ignored)

execution → termination (when the process is normally completed)

* Types of OS :-

(1) Batch processing operating System : MS DOS

(2) Multiprocessor :- WINDOWS, UNIX.

(3) Multitasking :- WINDOWS, UNIX

(4) Multithreading :- UNIX

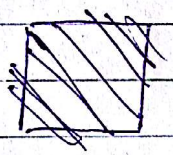
(5) Time sharing :- Unix.

(6) Real Time :- Rocket launching system

(7) Distributed OS :- Amoeba.

RAM / ROM

(1) Heterogeneous } m×n.
(2) Homogeneous }



Source Code
↓
Lexical Analyser
↓
Syntax Analyzer
↓
Semantic analyzer
↓
Intermediate Code generator.
↓
Machine independent Code optimizer
↓
code generator.
↓
Machine dependent code optimizer.
↓
Target Code

* Compiler v/s Interpreter

Compiler	Interpreter	Properties
(1) Extra memory space.	(1) No need for extra memory space.	(1) Space
(2) Very difficult to design.	(2) Relatively easy to design.	(2) Complexity
(3) Compiler is slow.	(3) It is fast.	(3) Speed
(4) 3RD generation lang and 4TH mostly written in " "	(4) Mostly written in 1st generation & 2nd generation lang.	(4) Language

* Machine level v/s Assembly level v/s High level lang.

* Introduction to Instruction :-

(1) Operators :-

(2) Operands :-

(3) Instruction is a combination of operators & operands.

(4) Instruction set :-

(5) Instruction formats :- Logical representation of the instruction.

① OP code
② Addressing code
③ Address field / Data code / Operand.

* Classification of instructions :-

a) 3 - Address instruction (3 fields)
b) 2 - Address instruction (2 fields)
c) 1 - Address instruction (1 field)
d) 0 - Address instruction (0/1 field)

M[ ] → content of memory location
[ ] → content

a) 3-Address Instruction :-

Let :

$X = (A+B) - (C+D)$: Represent this arithmetic exp with respect to 3 address instruction where X, A, B, C, D are memory locations.

ADD (R1), A, B. (result is stored in R1 register.)	$[R_1] \leftarrow M[A] + M[B]$
ADD (R2), C, D. (Result in R2 register)	$[R_2] \leftarrow M[C] + M[D]$
SUB X, R1, R2	$M[X] \leftarrow [R_1] - [R_2]$

b) 2-Address Instruction :-

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

LOAD R1, A	$R1 \leftarrow M[A]$
ADD R1, B	$[R1] \leftarrow [R_1] + M[B]$
LOAD R2, C	$[R_2] \leftarrow M[C]$
ADD R2, D	$[R_2] \leftarrow [R_2] + M[D]$
SUB R1, R2	$[R_1] \leftarrow [R_1] - [R_2]$
STORE X, R1	$M[X] \leftarrow [R_1]$

1 Address Instruction :-

STORE → to transfer the data from accumulator to any other variable

* Accumalator :- Before addition / Sub. one of the data is stored in accumulator and other the result will be stored in accumulator.

[ACC] → Content of accumulator

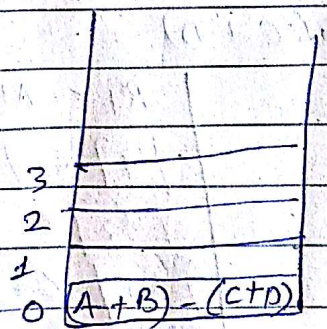
$X = (A+B) - (C+D)$

LOAD C	[ACC] ← M[C]
ADD D	[ACC] ← [ACC] + M[D]
STORE T1	M[T1] ← [ACC] (C+D)
LOAD A	[ACC] ← M[A]
ADD B	[ACC] ← [ACC] + M[B]
SUB T1	[ACC] ← [ACC] − M[T1]
STORE X	M[X] ← [ACC] {Storing the data from the accumulator to the memory location X}

d) 0 - Address Instruction : (Stack)

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

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



* Addressing modes & its classification :-

def :-

Different ways to specify the operand or operand location within the instruction definition is known as addressing mode.

~~To specify operand/location of the ope~~

Example :- OF | ADI 05 | [ACC] ← [ACC] + 05

| ADD | [TOS] ← [TOS] + [TOS-1] |

1	2	3
OPCODE	ADDRESSING MODE	OPERAND/LOC OF THE OPERAND
OF	1	05

contains 2 digits (8 bits) & usually written in hexadecimal code.

* Classification :-

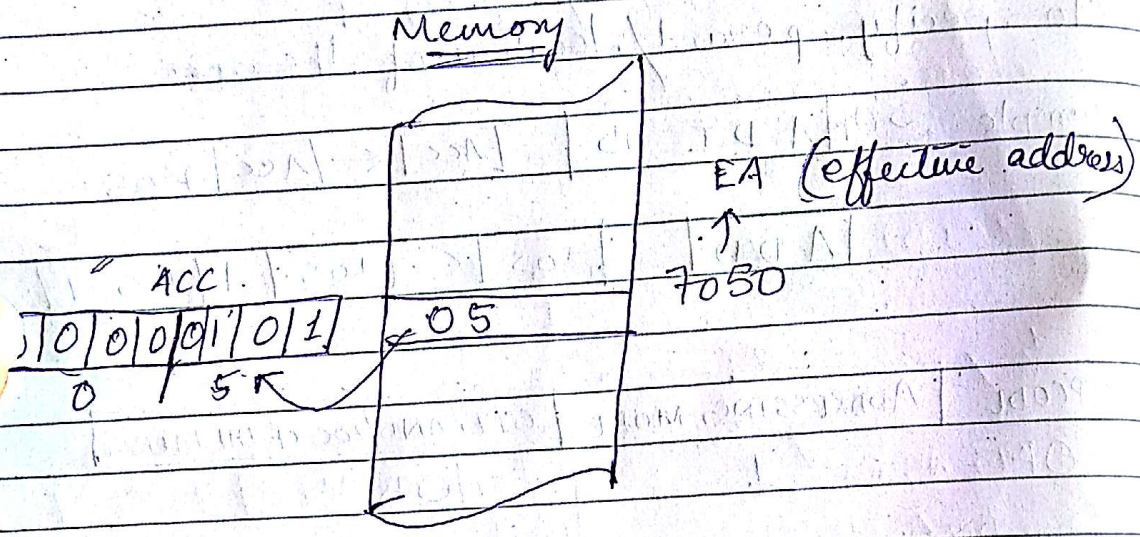
1. Memory Addressing Mode (Direct & Indirect)
2. Register " " (" ")
3. Implied/Implicit (~~Direct & Indirect~~)
4. Immediate " "
5. Stack " "
6. Auto increment " "
   (Pre & Post)
7. Auto decrement " "
8. Relative " "
9. Base register " "
10. Index " "

* The instructions are written in Assembly language & Assembly lang understands hexadecimal code. So, we use OPCODE to represent it in hexadecimal no.

1a) Memory Direct :- (load/store instruction)

Def of instruction :- OA | LOAD 7050 |

[ACC] ← M[7050]



1b) Memory Indirect :-

Def of instruction : OA | LOADX 7060 |

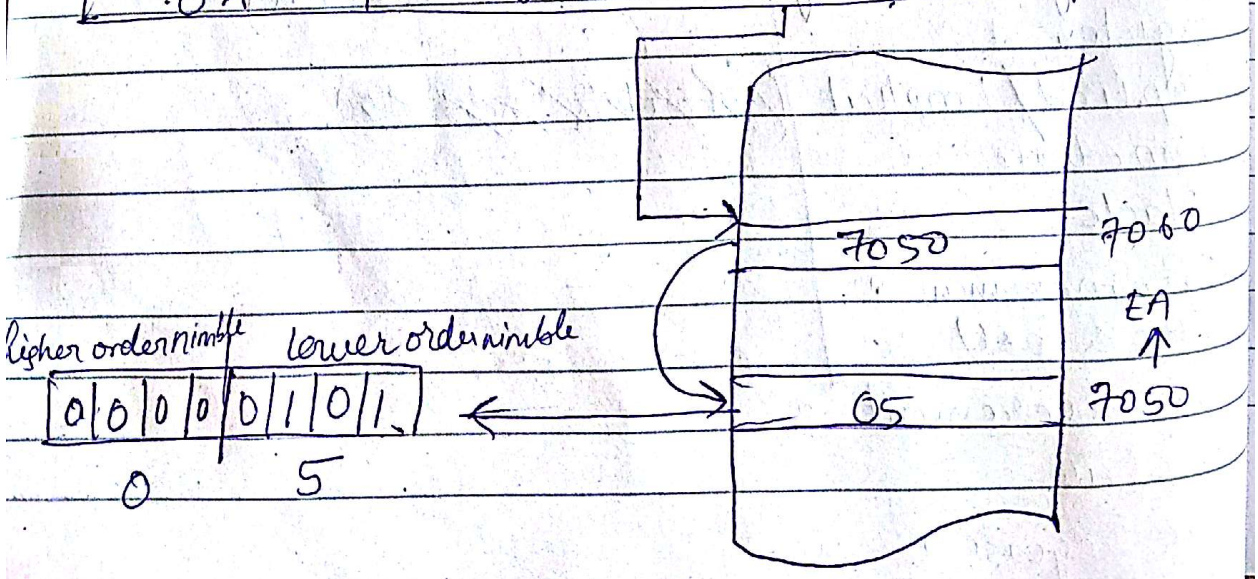
(X stands for indirect)

Mathematical eq :- EA = [7060]

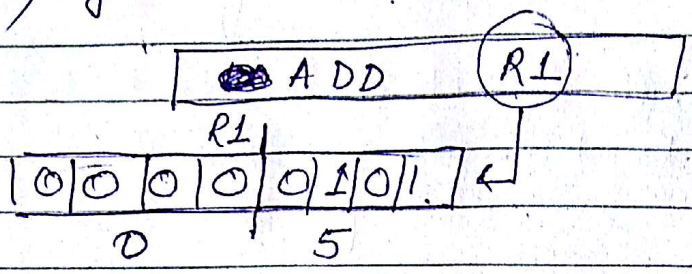
(content of 7060)

1 2 3

| OA | Memory indirect | 7060 |



2) a) Register Direct :- [ACC] ← [R1] + [ACC]



Since the location (register) is specified directly, so, Register direct.

2) b) Register indirect :-

LF [LOADX R1] :