MOD - 4

* Von Neumann architecture: (Stored Program memory architecture).

→ The Von Neumann architecture is based on three concepts:

a) Data instructions are stored in single read write memory.

b) contents of memory are addressable by location, regardless of data.

c) Execution occurs in sequential pattern from one instruction to next

→ • Program & data is stored in main memory.
  └ instruction set.

→ CPU → Registers, ALU, control unit.

• ALU ⇒ we have circuits to perform arithmetic & logical operations.

• Registers ⇒ stores data temporarily, small size, fastest

• Program counter → address of the next instruction is stored here.

• Accumulator counter → to store intermediate results.

• MBR (memory buffer register), IR, IBR, etc Registers.

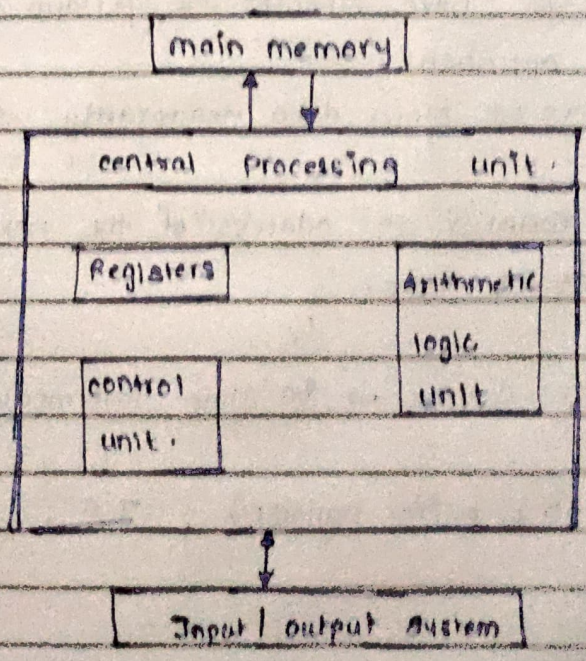
- Control unit : Here we have timing signal & control signals. / coordinate instructions & other units.

-> Timing signals : which instruction

- It gets, which instruction should be executed at a certain time : so that instructions are executed in a sequence.

-> Control signal :

- how to pass an instruction, how to read, access the registers.
- how to control the processes seamlessly is done by control signals.
- how an instruction is to be passed to hardware, is controlled by control signal.



word ⇒ memory representation unit, can be multiple byte.
word size ⇒ no of bit in word.

- components are attached using Bus, like address bus, data bus, etc.

- stored memory program
- data / instruction is stored in memory.
- CPU ⇒ Registers, ALU, control unit.

* General Purpose Registers

- size → generally 16 bits.

address is stored.

* Address register ⇒ (AR) ⇒ takes the address of the instruction stored and gives it to memory.
  - Memory decodes the address & fetches data from that location.

word is stored.

* Data Register (DR) ⇒
  - How many bits will be there in a word is stored in ~~regis~~ Data Register.

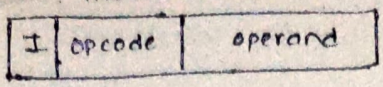
intermediate data i.e. word is stored

* Accumulator ⇒
  - To store ~~inde~~ intermediate data, eg: data from ALU, can be stored in accumulator.
  - To store data for a small period, we store it in accumulator.

address is stored

* Program counter (PC) ⇒
  - To store address of next instruction.
  - points the address of next instruction.

* format of instruction



Page No.
Date

* Instruction Register (IR) ⇒ 16 bit instruction.
  - operand → location from where data is to be fetched.
  - opcode → operation to be performed.
  - I → direct (0) addressing, indirect (1) addressing.

* Temporary Register (TR) ⇒ 16 bits (data)
  - To store temporary data.

* Input Register (INPR) ⇒
  - To fetch data from Input devices & send to ALU.

* Output Register (OUTR) ⇒
  - output from ALU, to the output devices.

Functional units of computer:

→ Basic units:
1. Input } I/O unit
2. Output }
3. Memory
4. ALU } CPU.
5. Control unit }

Information handeled by computer.

* Computer takes instructions or data.
  - Program is a set of instructions to perform task.
    - program is stored in memory.
    - processor fetches the required instructions.

* Data → numbers & encoded char. used as operands by instructions.

* Input units: → accepts & reads input data.
  → convert input into binary.

* output units → display output.
  → binary to layman language.

* CPU → ALU, Registers, Control unit.

* Registers → high speed storeage unit, faster than memory.
  → stores data and results
  → for temperory storeage of operands.
  → made of FF.

* ALU → performs logical & arithmetic operations.
  - To execute an instruction,
    1) operands are brough in ALU from memory & stored in registers
    2) operation is performed
    3) Result is stored in memory again.

* Control unit : controls all operations

- coordinates activities of all units, by issuing control signals & timing signals.
- control signal issued by control unit governs the data transfer, which instruction should be passed to hardware, along with some timing signal & then appropriate operation takeplace.
- timing signal sets timing for execution of instruction, when a particular instruction is to be executed.

* Memory unit : stores data & program.

1) Primary memory → main memory (RAM).

- fast, as attached with processor.
- programs & data are stored while execution.
- contains large no. of semiconductor storeage cells.
  - cell = 1 bit info
  - Group of cells = word.
  - distinct address is assigned to each word.
- no. of bits in word = word length. (16 to 64 bits)
- time required to access word from memory is memory access time.

* RAM → Random access Memory.

- any location can be reached in short time.
- Read & write memory,
- Volatile memory (temporary)
- used to hold programs while they are running.

* ROM → Read only memory

- only read memory, user can't edit ROM.
- non volatile - permeanant memory
- contents are pre recorded by computer manufacturer. & permenantly stored at time of manufacture
- contains - system programs, routine programs that runs computer, bios etc.

* Cache → small, fast RAM units, coupled with processor, often on the same IC chip.

* Memory hierarchy → cpu → cache & then main memory

2) Secondary memory :

- Also called auxiliary memory.
- to store large amount of data & programs
- Non volatile memory, retains data with/wo electrical power.
- Cheaper but slower than primary memory.
- Secondary memory is not directly accessed by CPU, instead data is accessed from secondary memory, loaded to RAM & sent to processor.

Instruction Cycle:

- Program containing a sequence of instructions, executed in computer by going through a cycle.

- 4 parts of instruction cycle:
  a) Fetch instruction from memory
  b) Decode instruction
  c) check for direct / indirect addressing
  d) Execute.

- This process continues.

Fetch & Decode:

- Initially program counter is loaded with address of first instruction.

- sequence counter (SC) is set to (0 zero).
  providing timing signal (T_0).

- After each clock pulse, SC is incremented by 1, thus timing signal goes through $T_0, T_1, \ldots$

- During T_0, address is transfered from PC to address register (AR).
  $(AR \leftarrow PC)$

- At T_1, memory at AR is passed to IR (instruction register) & PC is incremented by 1, for address of next instruction.
  $IR \leftarrow M[AR]$, $(PC = PC + 1)$

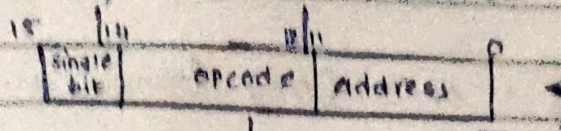
T_0 → AR ← PC

T_1 → M[AR] → IR , PC ← PC + 1

T_2 → (IR) → AR ← IR(0-11) , decode opcode IR(12-14)

T_3 → I ← IR(15) .

- at T_3, op code in Instruction Register (IR) is decoded, the indirect bit is transferred to FF I, & address part is transferred to Address register (AR)



decode opcode (12 - 14) IR

- AR ← IR(0-11)
- I ← IR(15)
  - 16th bit .

Put the three parts of IR in respective registers.

- Note that PC is to be incremented after each clock pulse i.e. $T_0, T_1, \ldots$

* Type of Instruction

- During T_3, Control unit determines the type of instruction that was read from memory

* memory reference instruction:

- $D_7 = 0$, opcode will be 000 to 110 .
- $D_7 = 0$ & $I = 1$, memory reference inst. with indirect addr.

- micro operations for indirect addr. condition can be represented as AR ← M[AR]

* Register Reference / I/O Reference instruction :

- $D_7 = 1$ & $I = 0$ Register reference .
- $D_7 = 1$ & $I = 1$ I/O instruction .

- The three instructions are divided into 4 paths.

- at T_3,

$D_7' I T_3$: AR ← M[AR] . (memory reference).

$D_7' IT_8$: Nothing

$D_7' IT_8$: execute register reference inst.

$D_7' IT_3$: execute I/O instruction.

$T_0 \rightarrow$ SC ← 0 , AR ← PC

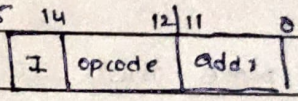
$T_1 \rightarrow$ IR ← M[AR] , PC = PC + 1

$T_2 \rightarrow$ (IR) ⇒

I ← IR (15)

AR ← IR (0-11)

decode opcode ← IR (12-14) .



$T_3 \rightarrow$ $D_7 = 0$ → opcode is from 000 to 110.

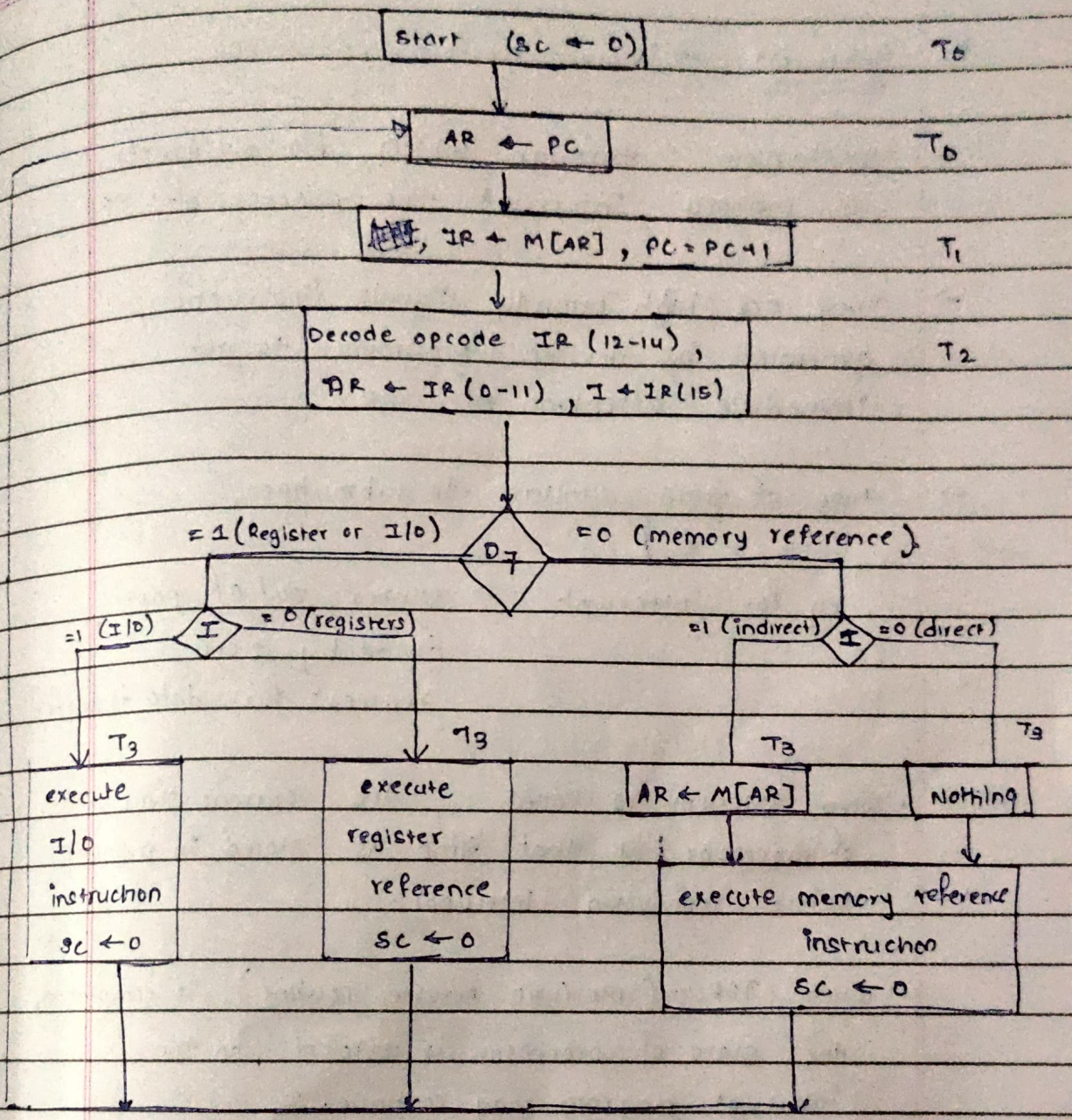
$D_7 = 0$, $I = 1$ → memory reference inst.

AR ← M[AR] .

→ $D_7 = 1$ $I = 0$ → Register reference execution.

→ $D_7 = 1$ $I = 1$ → I/O instruction execution.

Flowchart


Start (SC ← 0)
T0
AR ← PC
T0
IR ← M[AR], PC = PC+1
T1
Decode opcode IR(12-14),
AR ← IR(0-11), I ← IR(15)
T2
D7
=1 (Register or I/O)
=0 (memory reference)
I
=1 (I/O)
=0 (registers)
I
=1 (indirect)
=0 (direct)
T3
T3
T3
T3
execute I/O instruction SC ← 0
execute register reference SC ← 0
AR ← M[AR]
Nothing
execute memory reference instruction
SC ← 0

* Interrupt → improve performance of CPU.

→ mechanism, through which i/o devices or memory interrupt the process of CPU.

→ They are high priority signals/instruction, generated by devices / programs to get immediate attention of CPU.

→ type of ~~soft~~ similar to subroutine,

eg for interrupt → printer out of paper,
divide by zero,
request for data transfer

→ • When interupt is called, the current state of processor at that time is stored in memory before receiving interuppt.

• When ISR – (interrupt service routine) is completed, the state of processor is restored, so the interrupt program may continue.

→ makes CPU efficient by saving time of CPU.

→ overhead required to service the interrupt.
↳ interrupt latency.

⇒ How it increases CPU efficiency ?

→ • The processor keeps on checking the flag bit from other units eg I/O, if they are ready to send or not.

• The processor waits untill the other units is ready, the wait time leads to slowing down CPU.

• In such case, the I/O unit sends the interrupt to CPU when it is ready & in mean while time, CPU completes its other tasks.

• Thus efficiency is increased.

⇒ Interrupt facility can be enabled or disable by F.F. IEN

Main Memory

• Data is transfered in form of words
• word = group of bits
• word length = no. of bits in word.
• address is associated to each word.

SRAM = (Static RAM)

• retains stored information as the power supply is ON.
• High cost
• consume more power
• High speed than DRAM
• No need of memory refresh → made of F.F.
• Cache memory — eg

* DRAM (Dynamic RAM):

- retains information for short time (milisec.) even when power is ON.
  because, it used capacitors which gets discharged.
- Thus, needs memory refresh.
- cheaper & moderate speed.
- consumes less power.

eg: main memory.

* ROM

① Prom ⇒ (Programable ROM)

② EPROM ⇒ (erasable programable ROM)
deletes entire chip.

③ EEPROM ⇒ (electrically erasable programable ROM)
→ deletes specific bytes / blocks.

SRAM	DRAM
• store info, when power supply is ON.	• stores info for few millisec even when power is ON
• Made of FF. & transistors (MOSFET)	• Made of capacitors & transistor (MOS)
• Bit stored in the form of voltage.	• Bit stored in the form of electric charge.
• No leakage of charge.	• charge leakage from capacitor.
• no need to refresh	• need to be refreshed.
• no refreshing circuit.	• requires refreshing circuit.
• For single memory cell, 6 transistors are used	• for single memory cell, only 1 transistor is used
• consumes more power	• consumes less power.
• Faster than DRAM	• slower than SRAM
• more expensive	• less expensive.
• size of memory cell is large	• size of memory cell is smaller.
• low density (mem. cell per area)	• high density.
• used in cache memory	• used in main memory. eg: DDR, DDR2, DDR3...

Memory hierarchy