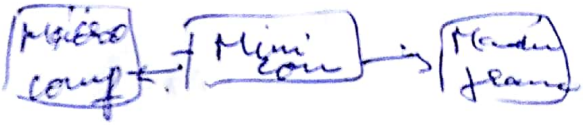
* what is minicomputer? (Mid range computer).

Ans. A minicomp, is a class of smaller computers that developed in Mid 1960s and sold for much less than mainframe and mid-size computers from IBM and its direct competitors.

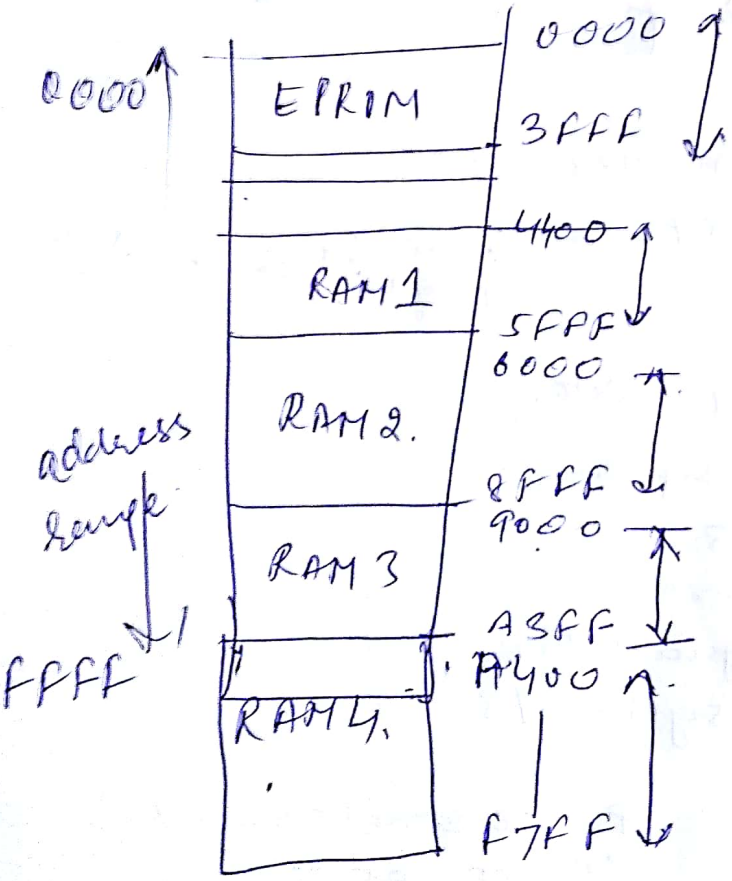
→ Minicomputer is mini version of mainframe computer; is smaller and cheaper and size is small than mainframe and supercomputer. (used for scientific purpose)

→ slower speed
→ speed faster than microcomputer
→ more than one CPU
→ expensive
→ small memory.
→ used in banking sector



→ Examples: IBM-system 360
PDP-8
HP-3000
Honeywell 200
IBM system/3

* Micro computer :- (PC personal computer) desktop, laptop
→ Single user computer



CEC10 (CO) Sem-4 Sec-3	Computer Engg CEC10 - Microprocessors	WED	(1-2) RNo ↓ 5221	(4-5) RNo 5222
		Friday	1-2 (5221)	

Course Contents:-

1. Introduction: Overview of μprocessors, microcontrollers & DSP, 8085 pin dia, arch, Instr. set, programming
2. Arch 8086 μp:
3. Assembly language of 8086.
4. Interfacing with 8086: - with ROM's, RAM's, 8255 PPI, DMA cont 8237, - - 80386, 80486 - 32 bit μp
5. High end processors:-

books ① Ramesh S Gaonkar → Mp. arch, prog, and Application with 8085 → Prentice Hall

② D V. Hall :- μp & Interfacing Prog & Hardware TMH - 2nd edition

③ Barry B. Brey "The Intel μp, arch", 6th edition)

④ Uffenbeck "The 8086 family design" PHI, 2nd edition

Lec - 1st

→ General definition of mini computers,
→ µprocessors
→ µcontrollers
→ DSP's.
→ 8085 µp - pin description
→ Internal architecture

Contents of 1st chapter.

* Mini computers.

Lec - 1st

→ Historical background.
→ Moore's law
→ evolution of IC tech
evolution tree of µp's
key features of µp & µC's

→ Hitorical background:

1947: invention of transistor (William Shockley, in bell laboratories)
before this vacuum tubes were used (bulky, large in size, consume loss of power) unreliable)
→ Solid state devices (semicond Germanium & Si)
→ current controlled devices → current can be controlled by adding impurities.

1959: invention of Integrated ckts developed by fairchild Semiconductor's
planar technology.

1965:- Moore's law -:
→ component density would double every year.
1975 → component density would double every 18 months
1995 →

1971:- Development of 1st µp. by Intel based on IC.
→ 4-bit µp. → 4004 → CPU on a chip
→ Texas instruments
→ Motorola
→ Fairchild semiconductor
} Manufacturers in race of producing µp's.

CPU
ALU
Registers
Timing & control circuit
→ Any intelligent electronic sys. can be developed using µp.

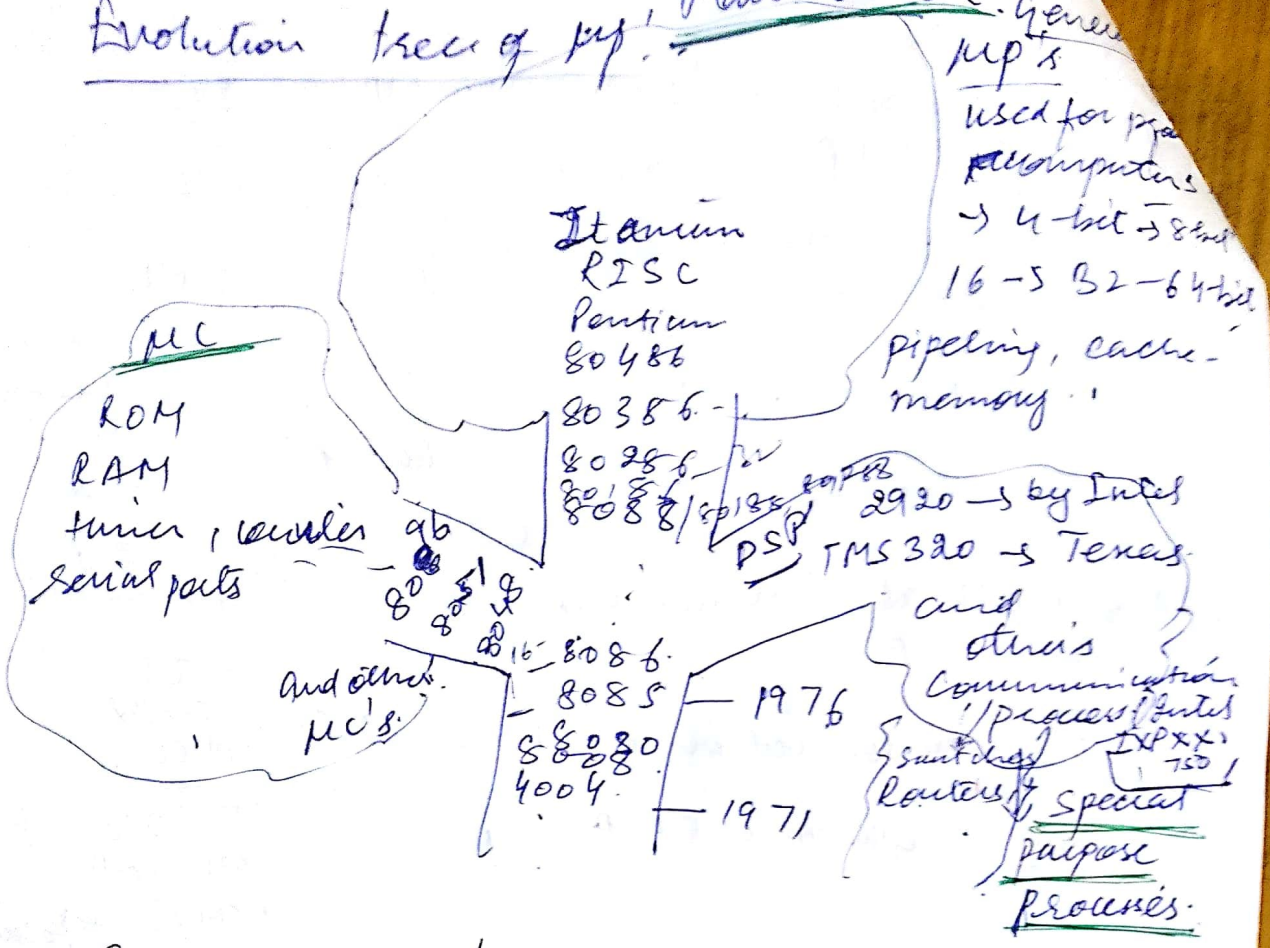
1976: Introduction of First µController by Intel 8048.
Computer on a chip µC.

CPU
RAM
ROM
I/O ports
Timing & control circuit
Serial port

'Evolution of IC technology'

Year	Technology	No. of devices	typical products
1947	Invention of trans.	1	
1950-1960	discrete components	1	Junction diode and transistor
1961-1965	SSI era	10 to 100	Planar devices, logic gates, FFs.
1966-1970	MSI era	100 to 1000	Counters, MUX, Decoder's, adder etc.
1971-1979	LSI	1000 to 20,000	8-bit µp, RAM, ROM.
1980-1984	VLSI	20,000 to 50,000	DSPs, RISC, 16-bit, 32-bit
1985 beyond	ULSI	> 50,000 trans/IC	64-bit µp,

Evolution tree of µp:



✱ Features of µp's:

1. Smaller size (miniaturized devices since large no. of devices are embedded into IC)
2. cost (cost of production is reduced)
3. higher reliable (inspite of more no of devices over the chip, inspite of ↑ing complexity)
4. Lower power consumption. (CMOS)
5. higher versatility / flexibility → user can use the same software by changing few hardware
6. More powerful (as no of trans. goes on ↑ing from 1971 → — —)

pplications:

1. General purpose µp's: Based on
→ Desktop, PC's, laptops, workstations, servers, Supercomputers. ← 32 bit, 64 bit

µcontrollers :- embedded system
[comb. of hardware + software]
designed for specific application
→ used in consumer electronics for making toys, Cameras, camcoder, — — Robots.

→ consumer products { washing machines, refrigerators
→ Instrumentation { intelligent inst. like CRO, oscilloscope
medical equip { ICU used in — and so — —
→ process control { for data acquisition and controlling various industries —
→ Communication { Telephone (cordless set - answering machine)
cordless phones, mobile ph.
→ office equip { fax machines, printer, PABX
→ Emerging multimedia application { Mobile phone, Tele conferencing equipments

→ SP µp's applications:

→ DSP's. processors. → switches
→ Routers
→ Intrution / detection
→ control processing

→ specially designed to process signals.

↓

Received digitised signal

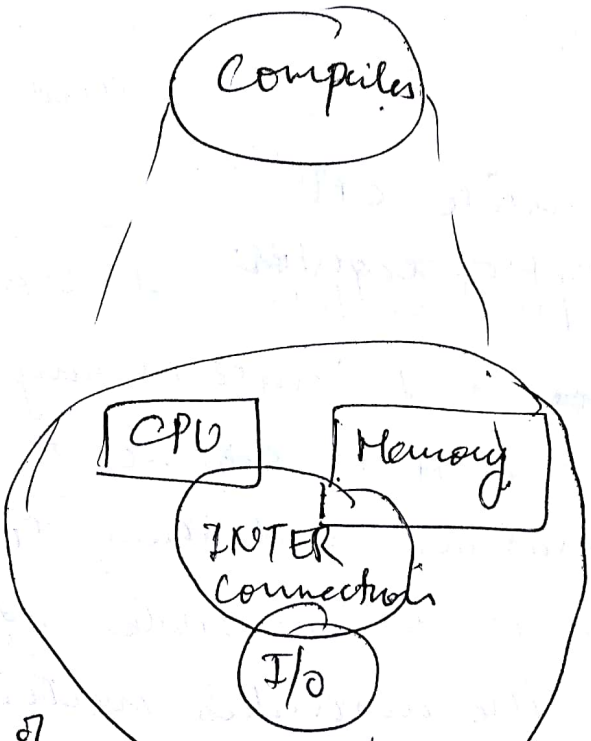
i/p → perform some mathematical operation → give result to an o/p device.

→ DSP's also accept analog signal using ADC,
→ process them using digital signal process and send result to DAC.

Microprocessor is CPU on-a-chip, means µp is used to realize computers.

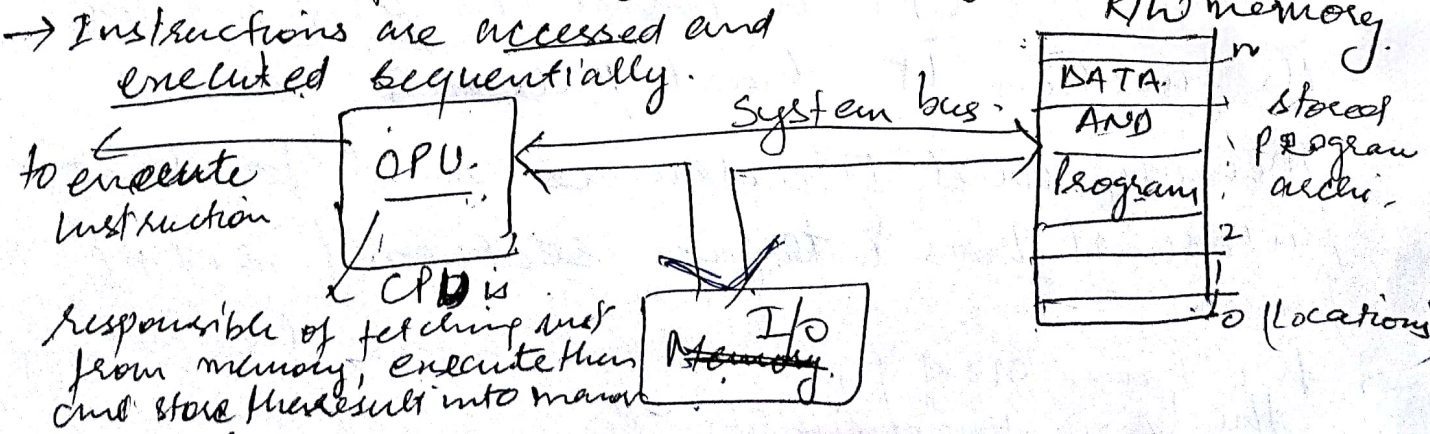
→ Before µp arch. we discuss structure of general purpose computers :-

Here it is, Top level Structure of comp



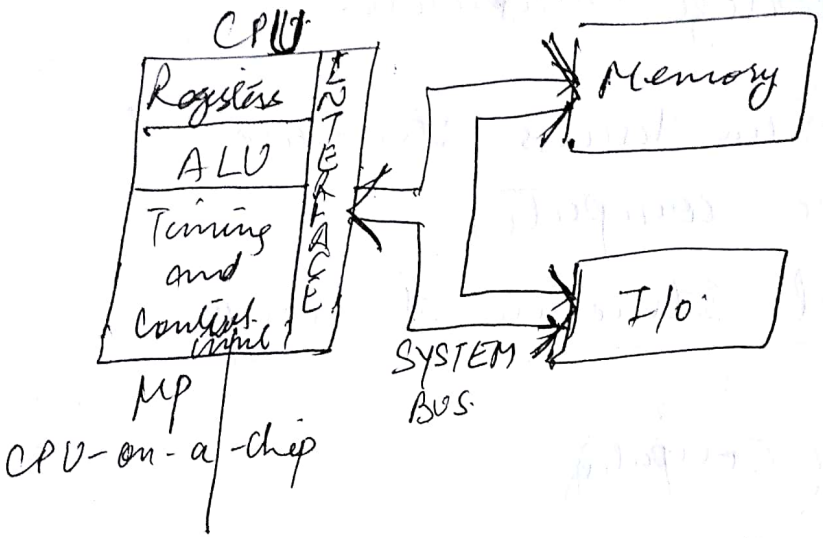
Characteristics of Von Neumann Architecture / Classical comp-arch.

→ Both data, instructions are stored in R/W Memory.
→ The contents of R/W memory are accessed by location.
→ Instructions are accessed and executed sequentially.



* Intel 8085 (8-bit) NMOS µp.

* Microprocessor Architecture (we replace CPU as µp here)



→ 40 pin IC fabricated on a single LSI chip
→ uses a single +5V DC supply.
→ clk speed is 3MHz
→ clk cycle is 320ns
→ 80 basic instructions and 246 opcodes

Now what is inside CPU?

→ ~~we require a set of registers~~
→ The func of CPU is to fetch inst. from Memory and ~~I/O devices~~ and temporarily store them in registers, and ALU require to perform certain operations and then Timing and control unit is responsible for coordinating and control the activities within the CPU and also control various memory devices and I/O devices.

→ Now the CPU is inside the IC, but for its function it has to interact with memory and I/O devices. and for that purpose it has Interface section (set of pins)

→ We have no of µp's for the last few decades but the basic functionality remain the same.

So once you have basic understanding of I/p then its easy to learn for others - -

→ So, first we discuss 8085 (8-bit) µp.

→ Let us focus on diff. section of CPU!

→ Register section: A register is set of F/F's which is used to store inst, ~~and~~ data and the no. of F/f is depends on what is the size of data, inst and size of address.

Registers

→ ~~how much~~ and also the execution time will be decided by general purpose registers.

→ ease of prog: depends on no' of register and its type.

→ these are used as 8-bit or 16-bit register's in µp's, we shall discuss it later on - -

General purpose

(these reg are not used to store particular type of info; these can be used to store data, address and as a consequence name Gen. purpose kan be

→ no of Gen. purpose registers and the no. of bits in diff. reg., will decide its functionality.

→ ~~Size~~ the no. of general purpose reg in particular µp will decide the size of program, that means to realise a particular task, how many inst are required.

② Special purpose register:

→ Accumulator → used to store one of operands of an arith and logical operation.
→ Program counter → keep tracks of memory addresses of the instructions in a prog location while they are being executed.
→ address is generated by PC.
→ Status register → will reflect outcome of inst. execution, means, whether there is an overflow, positive, -ve results, parity etc.

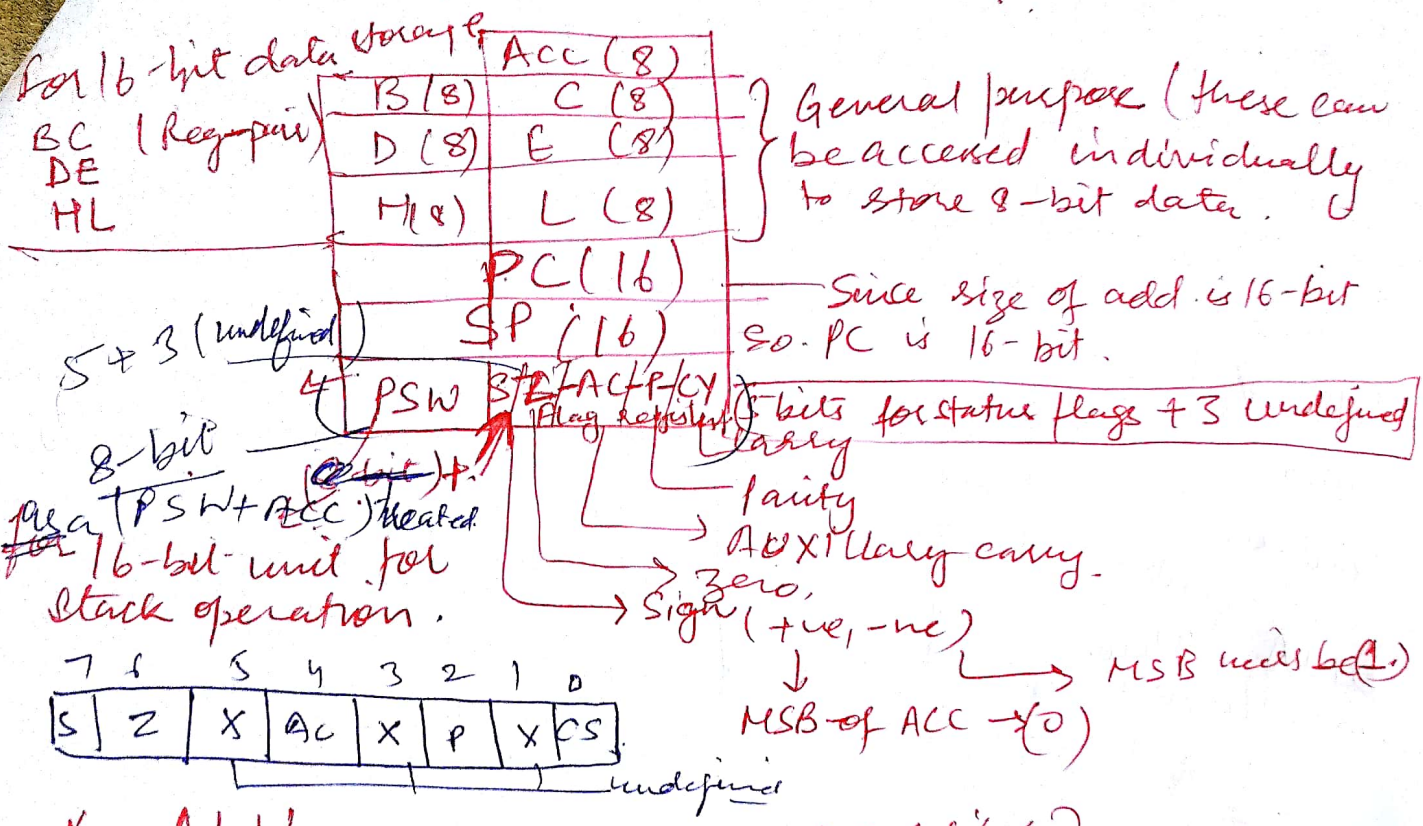
→ Stack pointer: used to implement one very important data structure i.e. stack, i.e. may be familiar with → data structure various applications.
It is used for particularly, for interrupt handling, whenever we require subroutine call, stack is used to store information which we require after returning from interrupt or subroutine call.
→ So in µp, stack is handled with SP-register.

③ User accessible or not: All these registers are user accessible, means with helps of inst./prog we can access them (write/read).

→ Memory address register
→ " data " " as buffer register
→ Instruction register → holds the opcode of the inst which is being decoded and executed.
→ temp register hold results temp.

} Not accessible but perform useful functions

Registers organisation of 8085:



* ALU

→ Arth. operations → addition, Sub-, Inc, Dec } in 8085 (8-bit)

→ Log. oper → SHIFT, And, OR, XOR, NOT, Clear, Compare, Rotate } 8085 can perform

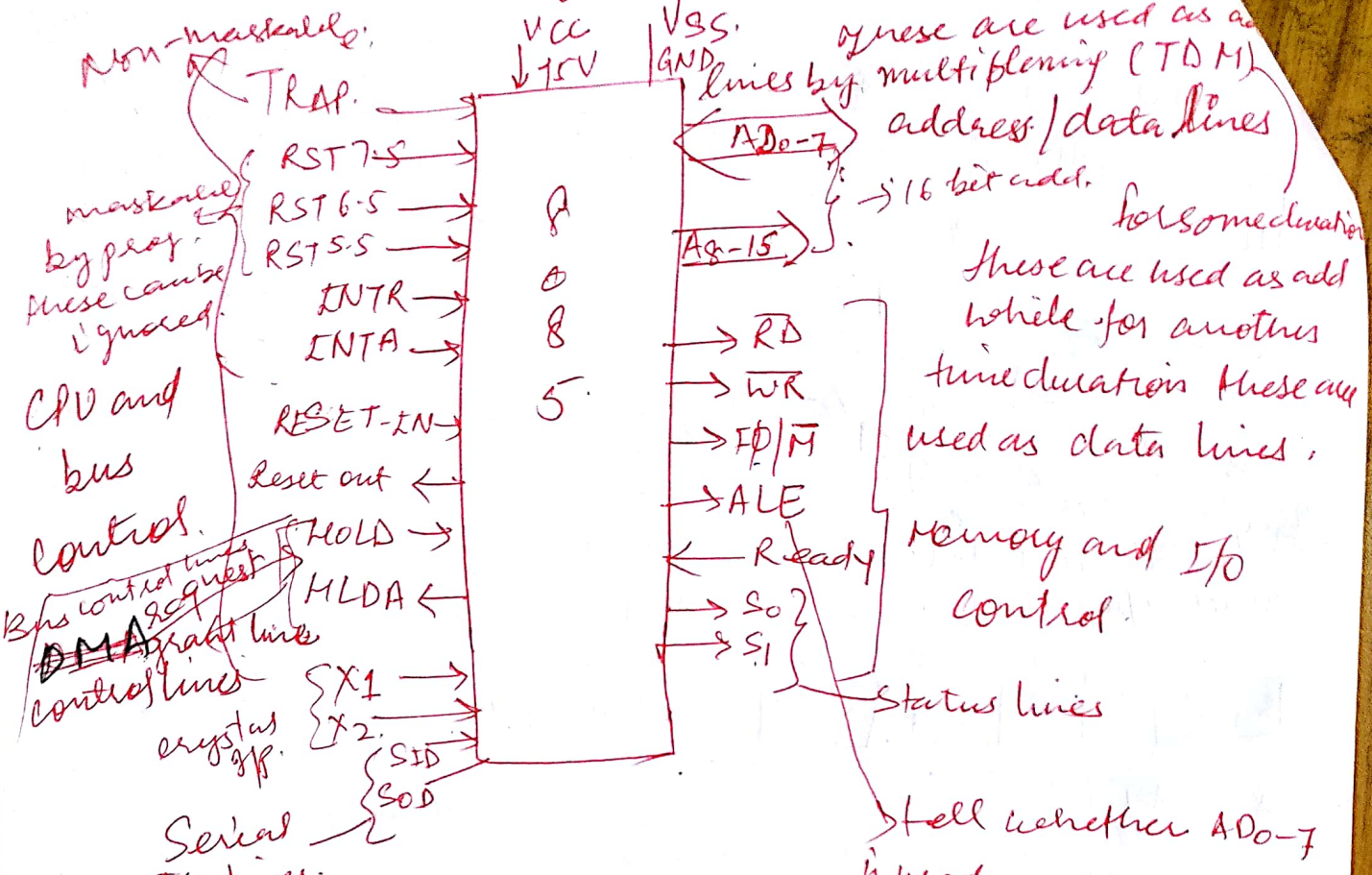
* Interface section: is decided the technology as well as commertical aspects, for eg: the no. of pins is dependent on technology.

→ Memory and I/O control lines.
R/W, IO/M, Ready/wait, ALE, Status lines, address lines, data lines

→ CPU and Bus control pins.
Reset, interrupt, bus request/bus grant lines)

→ Utility lines → Power supply lines (Vcc, GND.
Clk lines, I/O lines

8085 Interface lines



These are used as address/data lines by multiplexing (TDM)

→ 16 bit addr.

for some duration these are used as address while for another time duration these are used as data lines.

→ tell whether AD0-7 is used as address or data

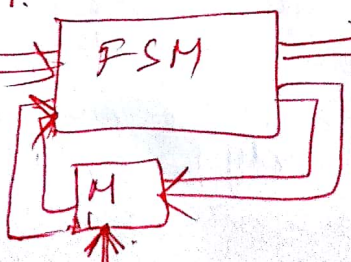
means when it is active

If means it is used as address when not active it is used as data.

* Timing and control unit: Coordinates and controls all internal devices/units (ALU, registers and external units).

It is nothing but FSM.

→ It controls data flow i/p's b/w CPU and peripherals



→ It provides status, control and timing signals which are required for operation of memory and I/O devices

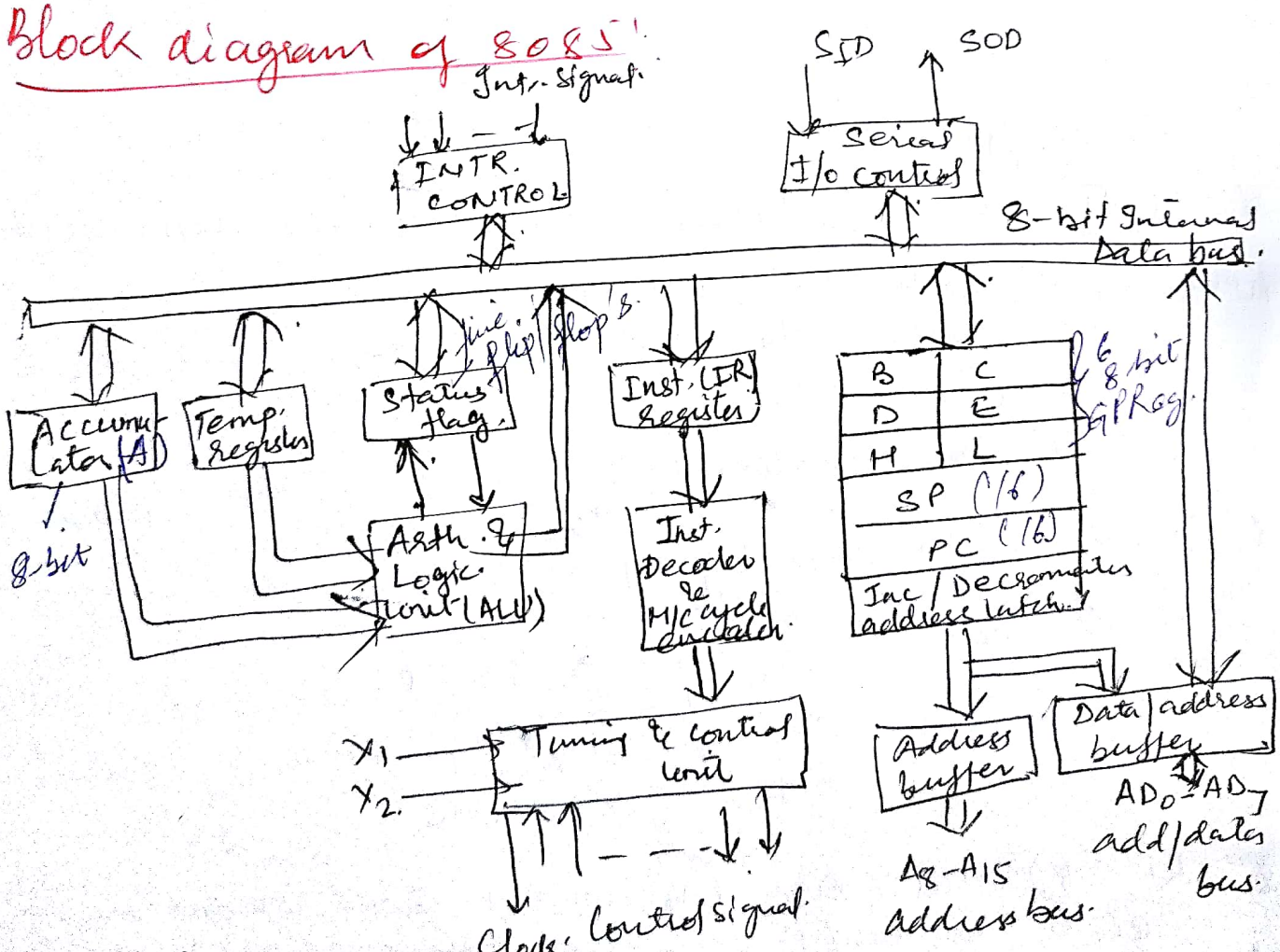
→ Controls entire operation of μP and peripherals connected to it

The 8-bit ADDRESS bus + 8-bit data bus form 16-bit address bus.

So. 2^{16} = 65536 = 64K, where 1K = 1024

means 2^{16} memory locations can be addressed directly by 8085. each memory location contains 1 byte of data.

Block diagram of 8085



⇒ Reg are used by µp for temp storage and manipulation of data and inst.

⇒ data remain in regis. till they are sent to memory or I/O devices.

→ 8085 is 8-bit µp.

→ data bus – 8-bit wide

→ add. bus – 16-bit wide as memory address are of 16-bits.

⇒ First of all 16-bit memory add. is transmitted by the µp, the 8 MSB's of the add. on the A-bus and 8-LSBs of the address on AD bus.

* $\overline{IO/M}$ (o/p) : It is a status signal which distinguishes whether the add. is for memory or I/O.
when high the add. on the add bus is for I/O device -
when low " " " memory.

* S_0, S_1 (o/p) : sent by µp to distinguish various types of operation

S_1	S_0	
0	0	→ Halt
0	1	→ write
1	0	→ Read
1	1	→ fetch

* $\overline{RD}$ (o/p) It is a signal to control read operation. When it goes low the selected memory or I/O device is read.

* $\overline{WR}$ (o/p) to control write operation. When it goes low the data on the data bus is written into the selected memory or I/O location.

Ready: A slow peripherals may be connected to µp through ready line. If 1, peripheral is ready if 0, µp waits till it goes high.

* Hold: indicates that another device is requesting for the use of add/data bus. Having received the Hold request the µp relinquishes the use of buses as soon as the current machine cycle is completed. Internal processing may continue. The processor regains the bus after the removal of Hold signal.

* HLDA: - signal for Hold acknowledgment.

* INTR: it has lowest priority.
- When it goes high the PC does not increment its contents.
→ µp suspends its normal sequence of instructions
→ µp acknowledges the interrupt signal and issues INTA signal.

RST 5.5, 6.5, 7.5, Trap: When an interrupt is recognised the next instruction is executed from a fixed location in the memory.

	location is picked up from which next inst
Trap	0024
RST 5.5	002C
6.5	0034
7.5	003C

These are restart interrupts they cause an internal restart to be automatically inserted. Each of them has programmable mask.
→ Trap has highest priority and non maskable.

Trap (highest)
RST 7.5
6.5
5.5
INTR (lowest)

* Reset in: It resets the PC to zero. It also resets interrupt enable and HLDA f/f's. It affect only IR. The CPU is held in reset cond. as long as reset is applied.
* Reset out: It indicates that CPU is reset.
* X_1, X_2: connected to ext. clk generator (crys. osc) produce a suitable clk for operation of µp.
* clk (o/p) clk o/p for user, which can be used for other digital IC's.
* SID (i/p) data line for serial i/p, the data on this line is loaded into 7th bit of the accumulator when RIM inst. is executed.
* SOD (o/p) :- 7th bit of the accumulator is o/p on SOD line when SIM inst. is executed.
* V_{CC} +5 volt supply.
* V_{SS} ground reference.

Intel 8085 instructions

→ Computer receives data from user + processes data + sends the result back to the user.

→ An inst is a command given to the comp. to perform a specified operation on given data.

Like ① MOV R1, R2 : The contents of reg. R2 is moved to reg R1, MOV B, A → (moves the content of reg. A to reg. B.)

② MVI R, data : MVI A, 05 , (moves 05 to Reg A.)

③ LDA addr : LDA 2400H (moves content of memory location 2400H to the accumulator)

opcode and operands:

↳ the first part of inst which specifies the task to be performed by the comp is called opcode:

→ 2nd part of inst is the data to be operated on, it is called operand (may be 8-bit or 16-bit address, data, internal reg or a reg or memory locations)

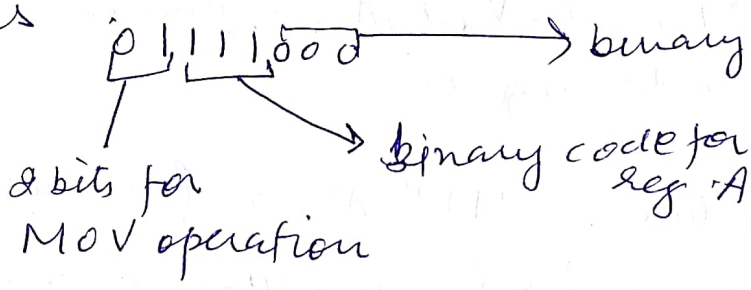
→ when operand is a register it is understood that the data is the contents of the register.

* Inst. word size: 8085 inst. are classified three types

1. 1-byte inst
2. 2-byte inst
3. 3-byte inst.

① MOV A, B : 78 H is the opcode for MOV A, B.

The opcode 78 H is written in binary form as



② ADD B: add contents of reg B to contents of acc.

→ 80H is the opcode for ADD B.

In this type of inst. it is assumed that the other operand is in the accumulator,

-80 H in binary form → 10000000

specify the operation to be performed ADD

3-bits for reg B for 8085 µP.

Two-byte inst. In this, the 1st byte of the inst is its opcode and 2nd byte is either data or address.

① MVI B, 05 : Move 05 to reg B.

[06, 05] → data which is to be moved to reg B.
↳ opcode for MVI B,

~~② IN 01 : Read data at port B.~~

* 3-byte inst: i) LXI H, 2400 H.

→ Load HL pair with 2400H

21, 00, 24 → in the code form.

↓ opcode for LXI H,

↳ 8 LSBs of data (2400H) which is loaded into reg L,

↳ 24 is 8 MSBs of data, which is loaded into reg. H.

(ii) LDA 2500H; get the content of the memory location 2500H into accumulator,

3A, 00, 25 → is the code form.

↓ opcode for LDA.

↳ 8 LSB's of the address of the mem. location (2500H)

↳ 8 MSBs of the address of memory location 2500H.

→ In this the data is content of memory location 2500H.

→ 3-byte inst is stored in 3-consecutive memory locations.

* Instruction cycle :- The necessary steps that CPU carries out to fetch an inst. and necessary data from the memory, and to execute it, constitute an IC.

$$IC = FC + EC.$$

In FC, a CPU fetches opcode (machine code of an inst.) from the memory. The necessary steps which are carried out to fetch an opcode from the memory, constitute a fetch cycle.

→ The necessary steps which are carried out to get data, if any, from the memory and to perform the specific operation specified in the inst. Constitute an execute cycle.

Lecture - 4th.

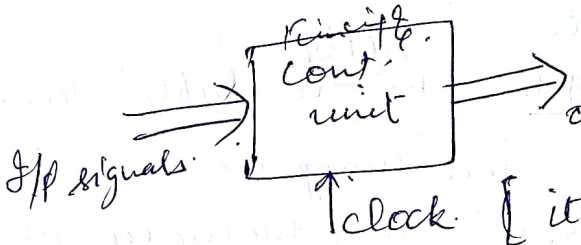
* ALU is brain of CPU, the timing & control unit is heart of system:-

* Timing and control unit:-

→ Coordinates and control the subsystems within CPU and outside the CPU (memory, I/O devices)

→ Control unit can be designed in two ways.

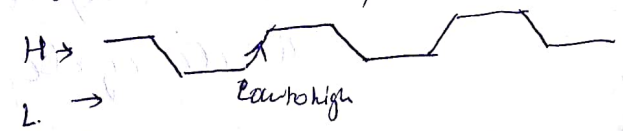
Hardwired — Microprogrammed



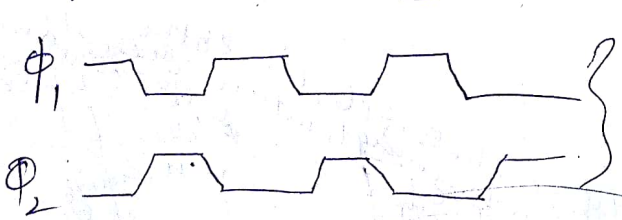
[it works under the clk]

How is generated in 8085?

* clock. It can be
  → single phase:
  → Two-phase. → will have repetition of signal.

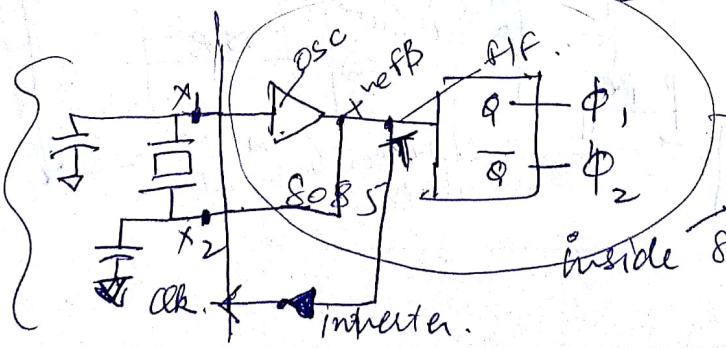


Level's



has two signal and

H-L
L-H
L-L
} Three states



ext to the chip

inside the chip 8085 two phase clk is generated

→ Under the cont. of the clk, the inst. are s
fetched from the main memory, one after the ot
and it is executed with in the CPU.

Instruction execution (fetch inst from main memory, decode it and then executing it)

$IC = fC + EC$ → no. of cycle, will depend on type of inst.

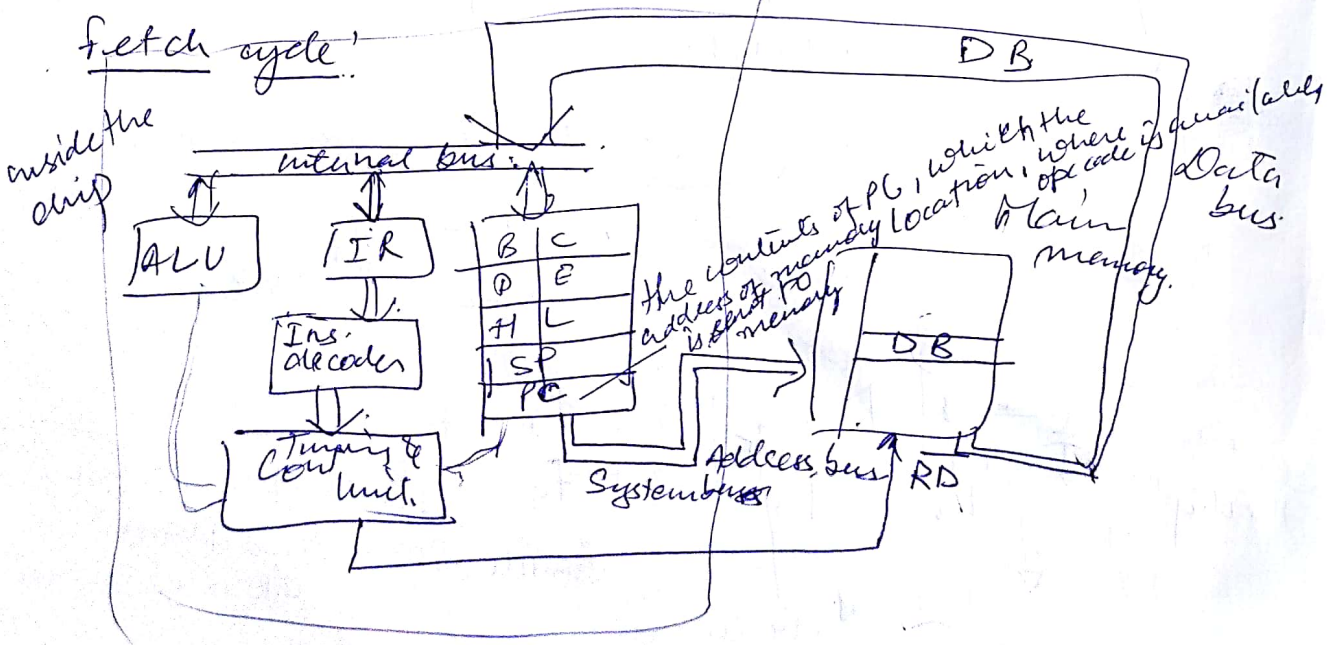
↓

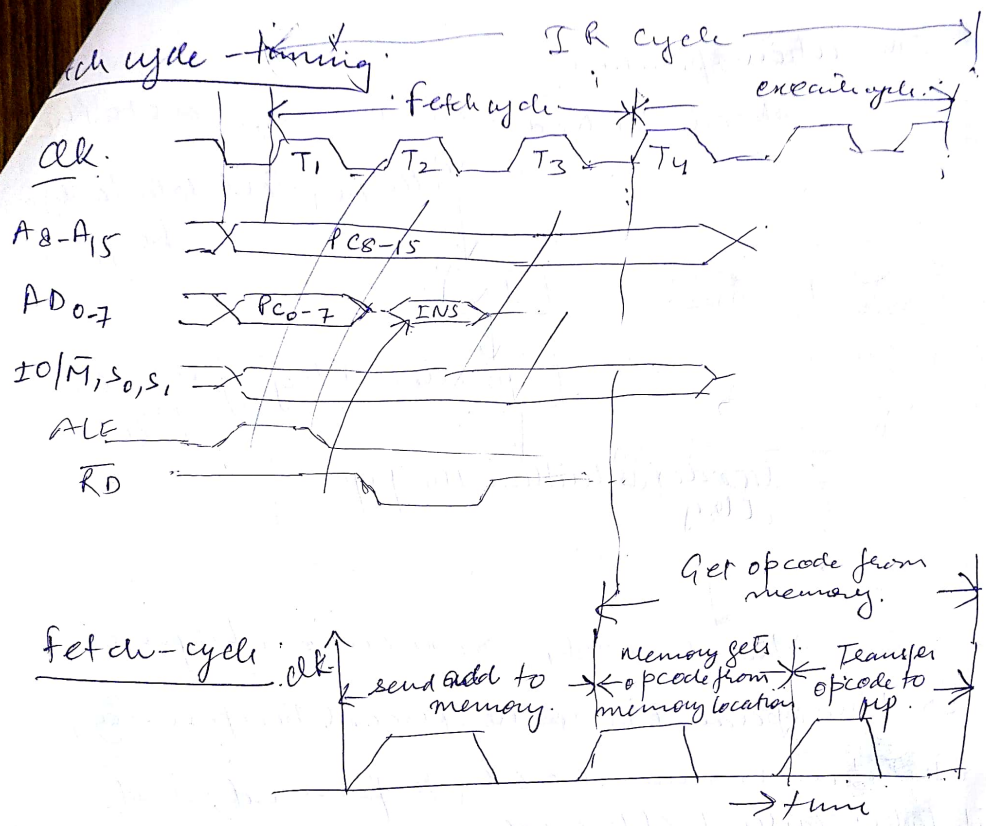
no. of cycle, ~~taken by~~ is usually fixed.

→ fetch cycle

→ execution cycle → ~~has~~ will be remarkable in time, because for execution, you may have to fetch some operands from main memory, or you have to write some data which is generated after execution, into the memory, so execution cycle involve → read cycle(s) / or more write cycle(s)

fetch cycle:





→. A slow memory may take more time.

→. In case of slow memory CPU has to wait till the memory sends the opcode.

→. The clk cycle for which CPU waits is called wait cycle.

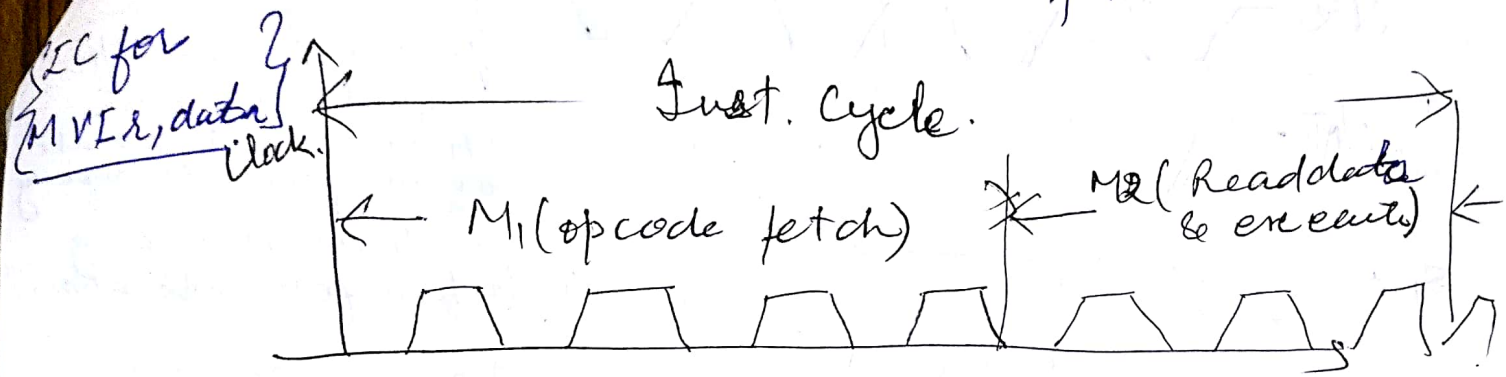
* execution operation!

opcode fetched from memory goes to the
↓
data register (data / addr. buffer)
↓
IR
↓
decoder circuitry — decodes the inst.
↓
decoder is within the µp - circuitry
↓
After decoded, → execution begins

→ if the operand is in the general purpose reg, execution is immediately performed, and it takes only 1 clk cycle,

→ but if an inst contains data or operand address which are still in the memory, the CPU has to perform some read operations to get the desired data. After receiving data,
↓
perform execution operation
↓
read cycle is similar to fetch cycle,
↓
For some inst write operation is performed In write cycle data are sent from the CPU to the memory or an o/p device.

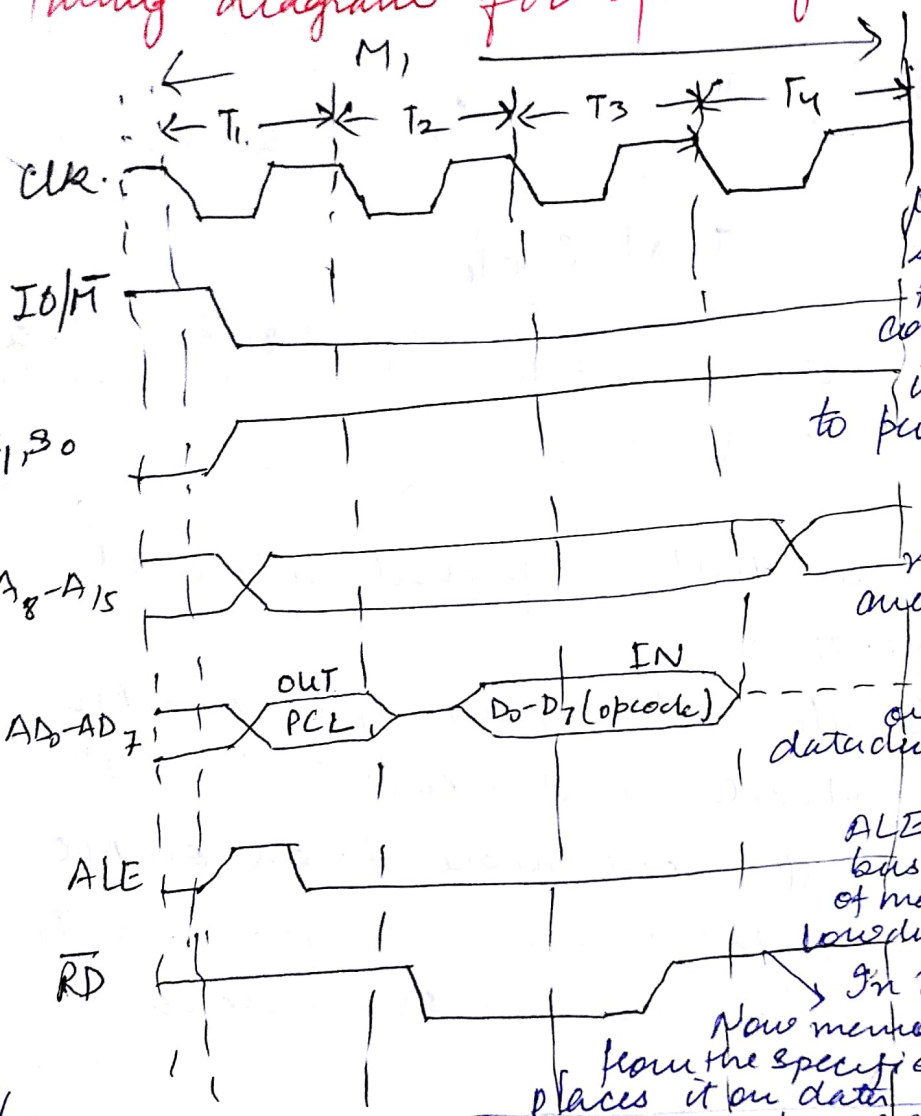
Machine Cycle: An IC consists of several MC's.



→ The opcode of an inst. is fetched in the 1st machine cycle of an IC.
→ Most of single byte inst. require only one MC to fetch the opcode and execute the inst.
→ Two or 3 byte inst. need more than one MC.
→ Additional MC are needed to read data from or to write data into the memory or I/O devices.

* **Timing diagram for opcode fetch cycle**:
→ In fetch cycle, the µP fetches the opcode of an inst. from the memory.
→

Timing diagram for opcode fetch operation.



µp issues a low IO/M signal to indicate that it wants to make comm. with the memory

$S_0, S_1 = 1$ indicate it is going to perform fetch operation

8-bits MSBs of the mem. add. are send over A-bus.

8 LSB's of mem. add. over AD bus; it carries data during T_2 and T_3.

ALE is high, when AD bus is used to carry 8 LSB's of mem. add, it becomes low during T_2 and T_3.

In T_2, µp make $\overline{RD}$ low, Now memory gets the opcode from the specified mem. location & places it on data bus.

→ During T_1, the µp sends out the add. of the memory location where opcode is available.

→ 16-bit memory add. is sent through add. bus (A) and AD.

During T3, the opcode is placed in IR, the memory is disabled when $\overline{RD}$ goes high during T_3.

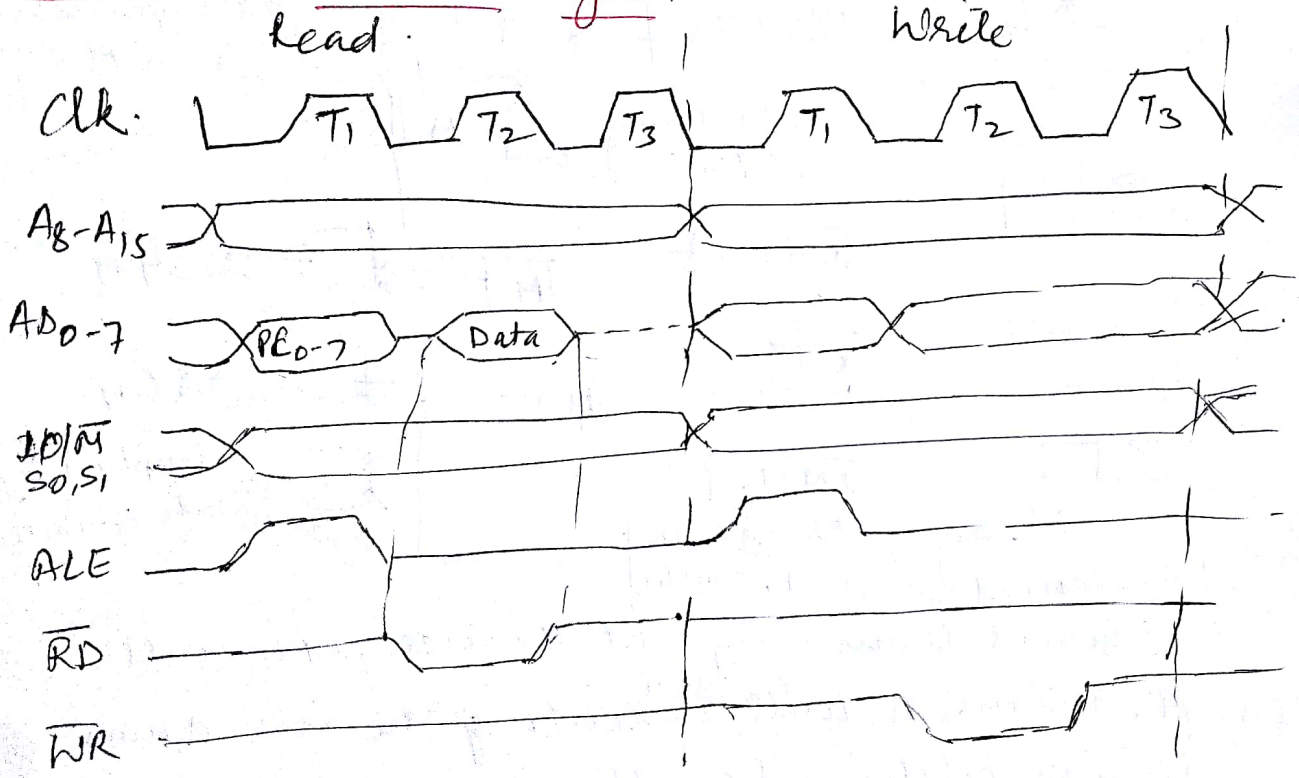
→ fetch cycle is completed by T_3.

→ opcode is decoded in T_4.

→ The µp examines READY signal in T_2, if it is high, the µp enters into T_3 state, if it is low, the µp inserts a wait state in between T_2 and T_3.

If inst is 1-byte long, it take 1 Machine cycle
→ 2,3-byte inst. takes more than one MC,
to read data or add from memory or I/O devices
or to write data into the memory or I/O devices

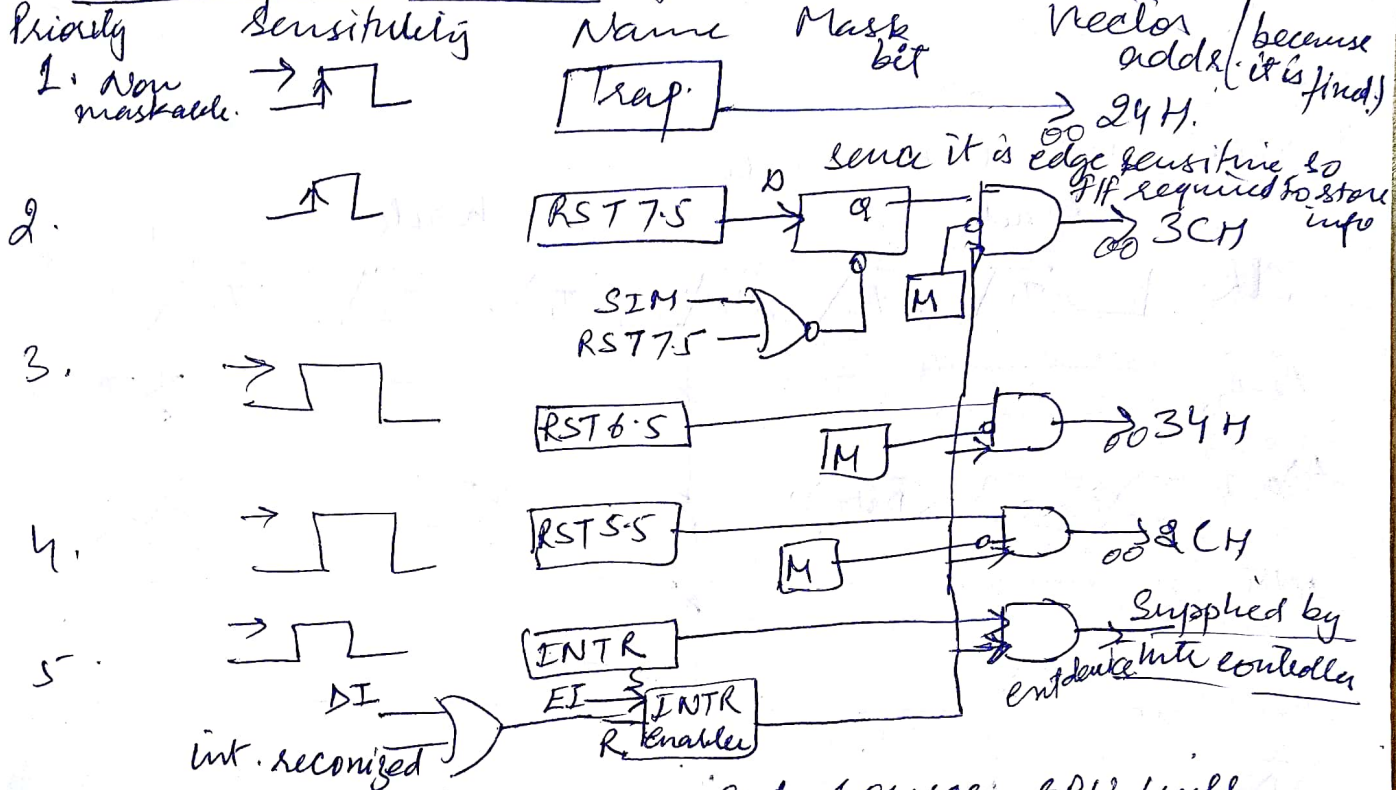
* Read and write cycle



I/O Read :- IO/M = 1
S0, S1 = 0, 1

* µp is provided with various CPU and bus control signals and one µp. CPU cont. signal is interrupt for example: Interrupt signals.

Interrupt structure of 8085. (Hardware Interrupts)



Trap: if it comes from any ext. source, CPU will interrupt, means, it will start executing the inst it was executing or fetching the inst, it will complete its process and after that it will branch to a particular mem. location, that is (0024H), it is fixed, whenever Trap occurs, PC will initialize by this 0024H and what will happen to the program it was executing, that PC value is stored in the stack, so this location 0024 is essentially the address of the interrupt service subroutine, that mean ISS corresponding to trap will start from 0024H, and

ce this subroutine & execution is over then it will
um from that subroutine and → the PC
contents will be initialized to recovered from
stack and PC value will be initialized
by that value. and this is how processor
will respond.

→ Other interrupts respond acc. to priority, which has highest priority.
→ A prog can disable all these interrupts by instruction setting several flag bits. by using DI
→ All can be enable by EI inst.
→ Interrupt can selectively enabled or disabled by using SIM instruction and mask bits.

→ whenever interrupt occur, interrupt will be activated if EI and M bits are activated.

There are some more bus cont. signals - HOLD HLDA

HOLD ← DMA request
HLDA → DMA grant (direct memory access is granted to the outside world).

ALL { add lines / Data lines / cont. lines } are tristated, means system bus cont will be handled by ext DMA controller.

* Reset response:
CPU is brought to a predefined state.
It initializes some P/P's, registers.
In 8085, when reset button is pressed, it clears PC, IR,

→ it resets the machine state flip flops
→ machine cycle ff
→ sets certain ff's like rst 7.5 mask bit
rst 6.5 mask bit
rst 5.5 mask bit