Er. Yadvendra Sharan
M.Tech. (Hons.), B.Tech.
Assistant Professor
Department of Electronics & Communication Engineering
Phonics Group of Institutions, Roorkee.
Phone: +91-8273990016
yadvendra.sharan@yahoo.com

Subject: (TEE-404) Microprocessors & Its Application

Unit II – 8-bit Microprocessors (Intel-8085)

THE COMPUTER: TOP LEVEL STRUCTURE

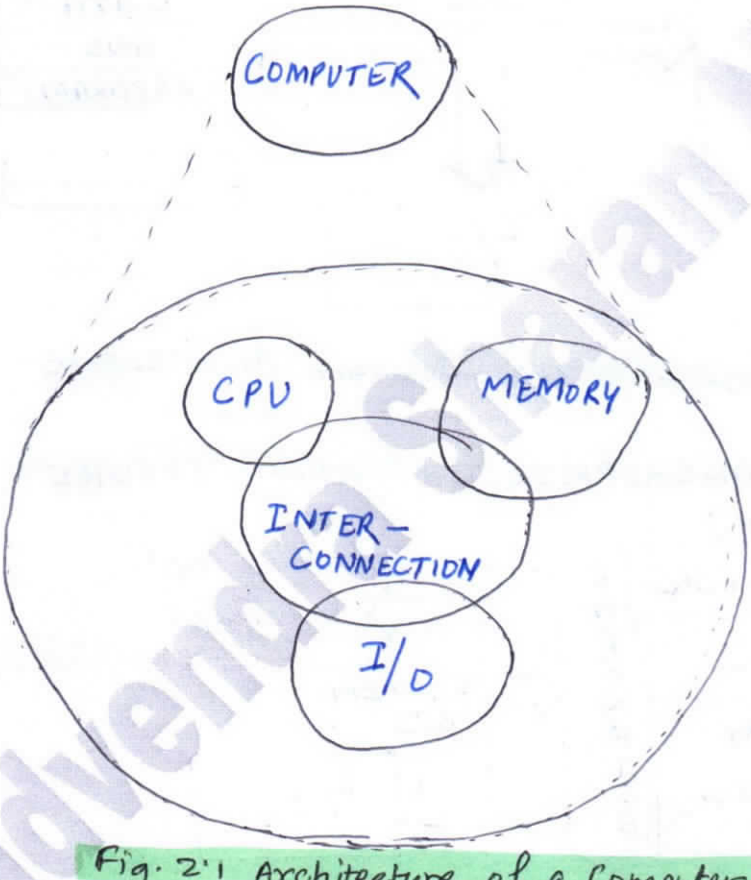


Fig. 2.1 Architecture of a Computer System

VON NEUMANN ARCHITECTURE

- also known as von Neumann Model and Princeton Architecture.
- is a computer architecture (First Draft of a Report on the EDVAC)
- Basic functions of von Neumann Model is given as below.
  - Both Data and Instruction are stored in R/W Memory
  - The Contents of R/W memory accessed by Location
  - Instructions are accessed and executed sequentially.

NOTE! CPU is nothing but a microprocessor or microprocessor is single chip CPU.

Unit II – 8-bit Microprocessors (Intel-8085)

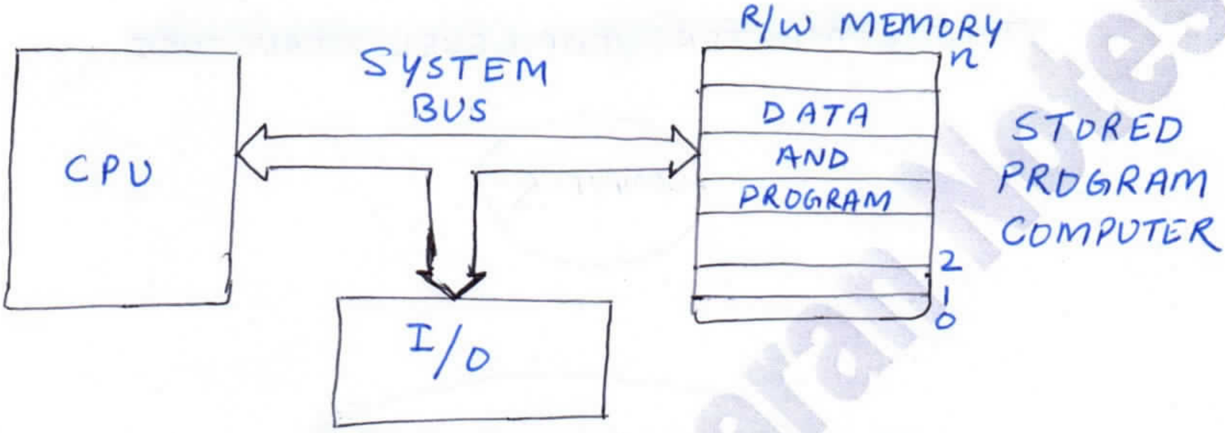
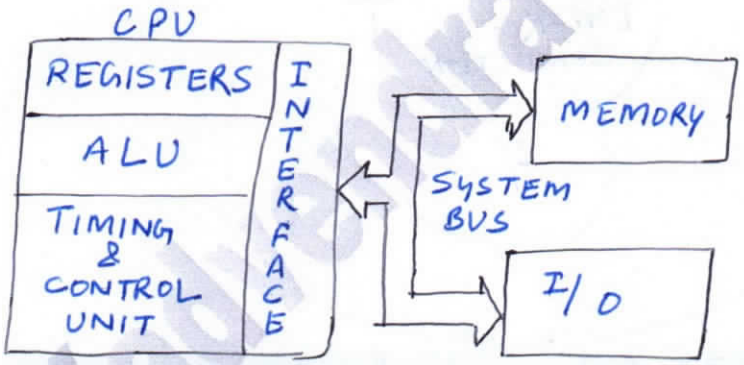


Fig 2.2 Von Neumann Architecture

MICROPROCESSOR ARCHITECTURE



MP → CPU-ON-A-CHIP

Fig 2.3 Organization of Computer

Microprocessor has following Sections

1- Register Section

2 - ALU Section

3 - Interface Section

4 - Timing and control Unit Section

Unit II – 8-bit Microprocessors (Intel-8085)

REGISTER SECTION

- Registers are set of flipflops which are used to store data(instruction data, Result etc.)
- Number of flipflops are depend upon size of data size of instruction, size of address.
- In case of 8085 most of registers are 8-bit
- Registers are broadly categorized as follows

• GENERAL PURPOSE REGISTERS

- Can be used to store data, to store address.
- By general purpose we mean that these registers have not been explicitly used to store particular type of information.
- Number of general purpose registers and number of bits in different registers will decide its functionality.
- Number of general purpose registers which are available inside a up will decide.

a. size of the program (No. of instructions)
Because intermediate results did not be transfered to main memory as a consequence more the no. of general purpose registers it will be possible to write a program with small no. of instruction.

b. Execution Time

c. Ease of Programming

- SPECIAL PURPOSE REGISTERS

- used for some specific function and given as below.

I. Accumulator [to store one of the operand]

II. Program counter [PC]
- Keep trakcks of the memory location
- hold the memory address of the next instruction to be executed.

III. STATUS Registers [will reflect the outcome of instruction execution]

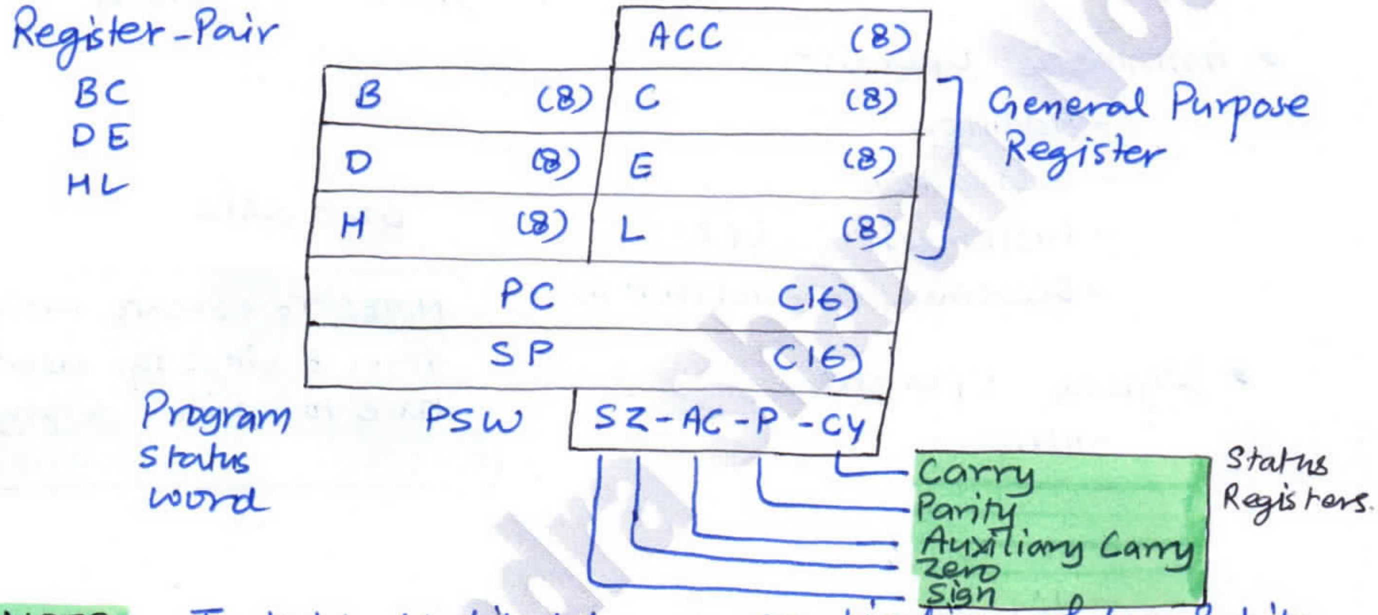
For example: overflow, sign, parity, zero result etc.

IV. Stack Pointer [SP]
- used to implement very important data structure. or data structure stack.
- The stack is a sequence of memory locations set aside by a programmer to store/retrieve the contents of accumulator, flags, program counter and general purpose registers during the execution of a program.
- In most of the up stack is implemented with the help of stack pointer.

- Apart from these user accessible registers, some registers are not user accessible registers which are used to perform some useful function
  a. MAR [Memory address Register]
  b. MDR [Memory Data Register]
  c. Instruction Register
  d- TEMP Registers

Unit II – 8-bit Microprocessors (Intel-8085)

REGISTER ORGANIZATION OF 8085



NOTE: To hold 16-bit data a combination of two 8-bit registers can be employed. The combination of two 8-bit registers is known as a register-pair

NOTE:

I- MAR - usedto hold the address before it comes out the address bus

II- MDR - used as buffer register to store the data which is accessed through the system bus.

III- IR - used to hold instruction which is being decoded and executed.

IV- TEMP REG. [to hold intermediate result during an arithmatic / logical operation.

Unit II – 8-bit Microprocessors (Intel-8085)

ARITHMETIC AND LOGIC UNIT [ALU]

-perform some arithmetic and logical operations

- Arithmetic Operations
  - Addition
  - Subtraction
  - Increment (add 1)
  - Decrement (subtract 1)

8bit data.

NOTE: To execute more than 8-bit data user have to write a program

- Logical Operations
  - AND
  - OR
  - ExOR
  - NOT
  - CLEAR
  - COMPARE
  - SHIFT/ROTATE

INTERFACE SECTION

40-Pins

- is decided by technology and commercial aspects.
- Interfaces broadly categorized as follows.

- MEMORY AND I/O CONTROL LINES

R/$\overline{W}$ → Read or write operation

IO/$\overline{M}$ → CPU is execuhing IO instruction or memory inst.

READY/WAIT -used for interfacing slow memory or I/O devices.

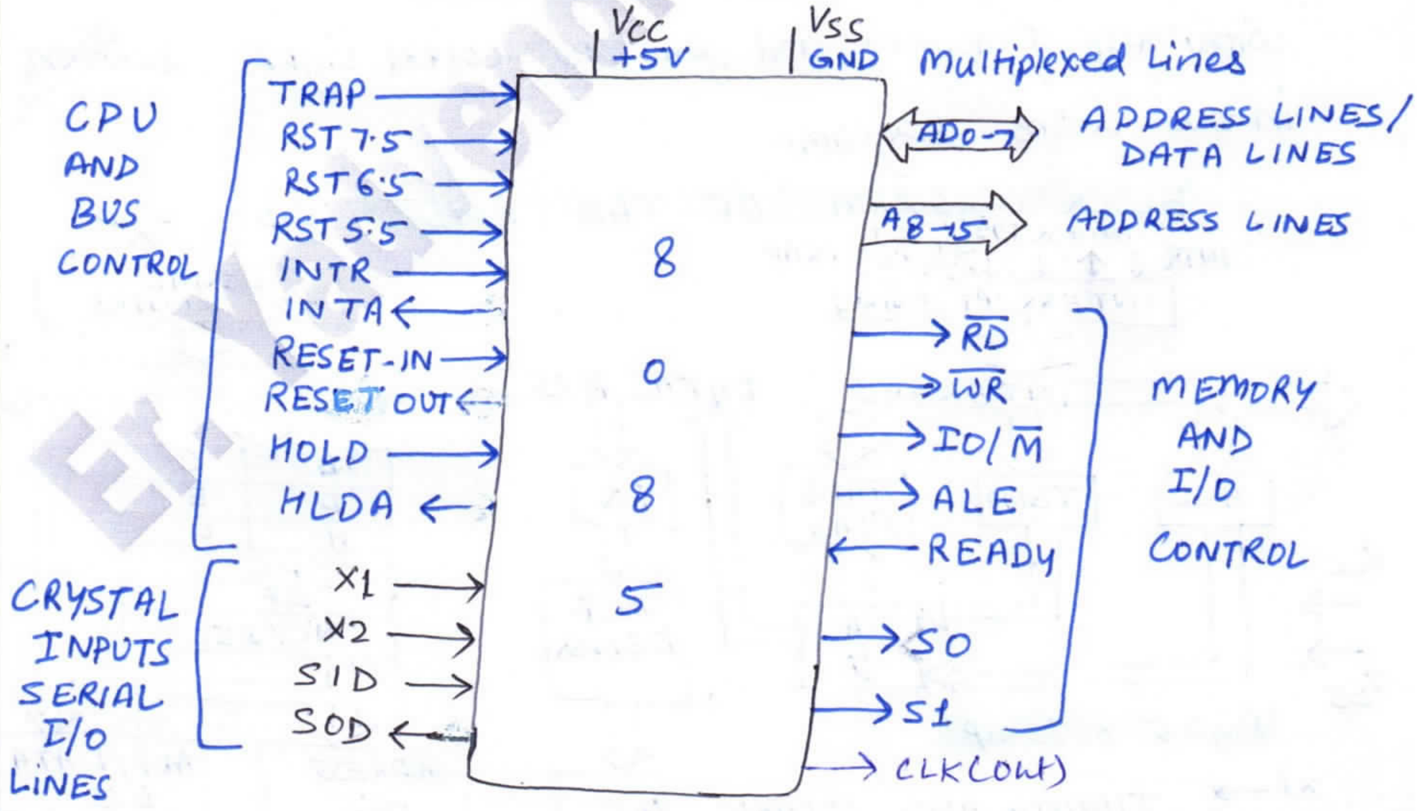
ALE (Address latch Enable) whather lines are holding address or data.

STATUS LINES -
ADDRESS LINES -
DATA LINES -

INTERFACE SECTION (Contd..)

- CPU AND BUS CONTROL LINES
  - RESET
  - INTERRUPTS
  - BUS REQUEST / BUS GRANT LINES
- UTILITY LINES
  - Power supply Lines [Vcc GND]
  - Clock Lines
  - I/O LINES

8085 INTERFACE LINES



TIMING AND CONTROL UNIT

- Coordinates and control all internal units.
- also coordinates with external world.
- Timing and control unit is Finite State Machine (FSM)

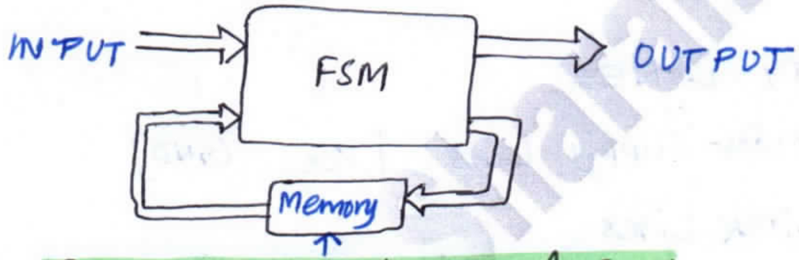


Fig 2.4 Block diagram of FSM.

- Control units will receive some input and decode this with help of memory and generate control signal as shown in the above diagram.

BLOCK DIAGRAM OF THE 8085

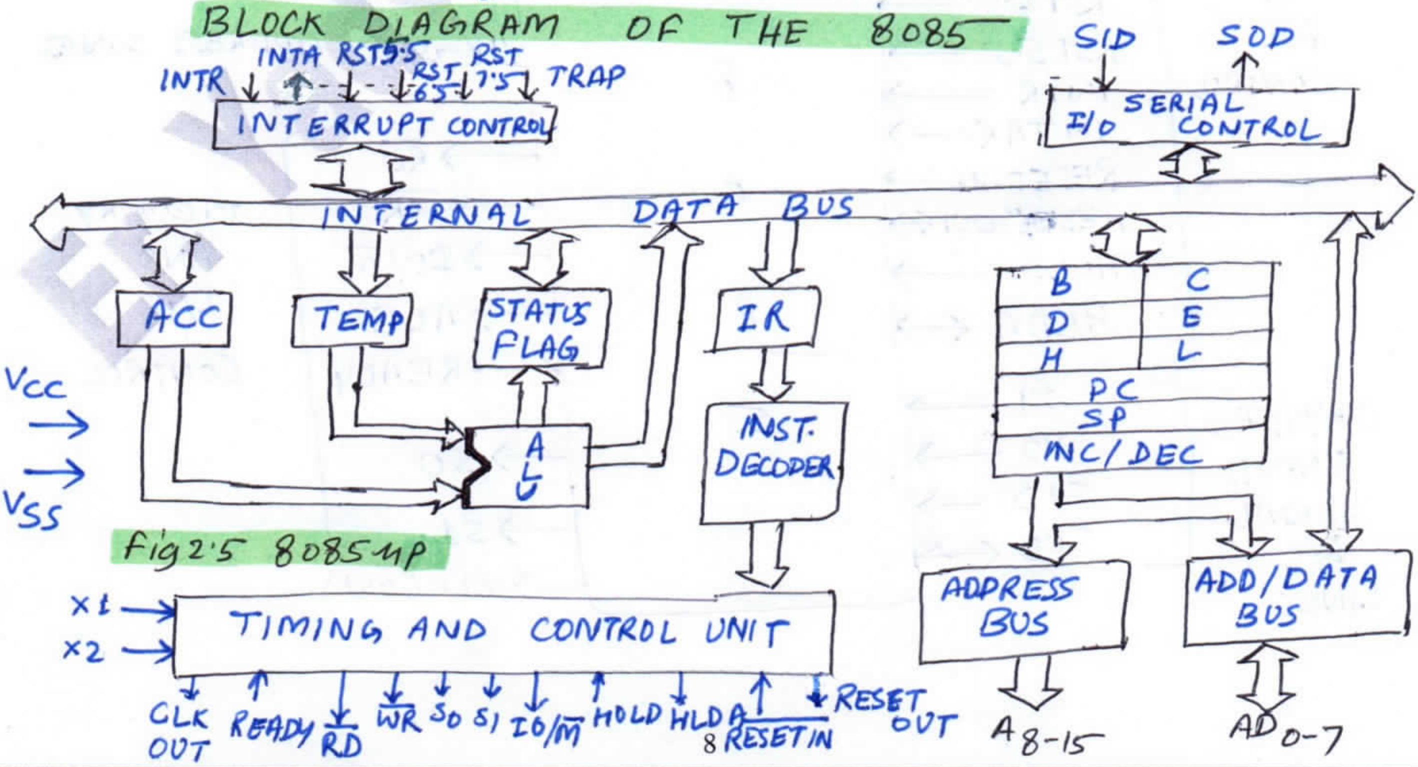


Fig 2.5 8085 μP

TIMING AND CONTROL UNIT

- If ALU is brain of CPU than timing and control unit is the heart of the CPU.
- Coordinates and control the subsystems within CPU and also outside the CPU.
- Control unit can be designed in two ways.
  - Hardwired
  - Microprogrammed.

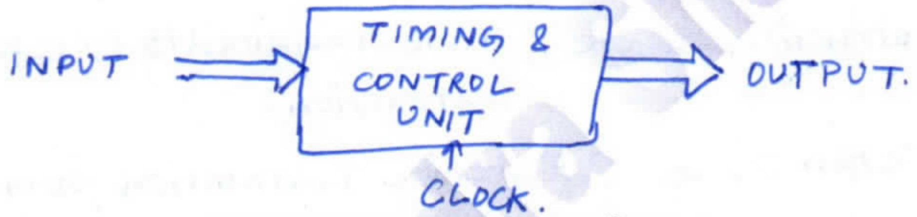


Fig 2·6 Control unit.

CLOCK.

- It can be single Phase or Two Phase clock.

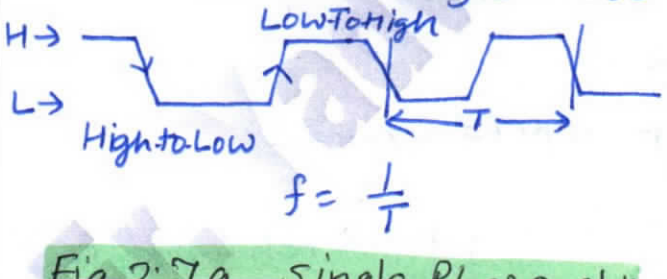


Fig 2·7a Single Phase cLK

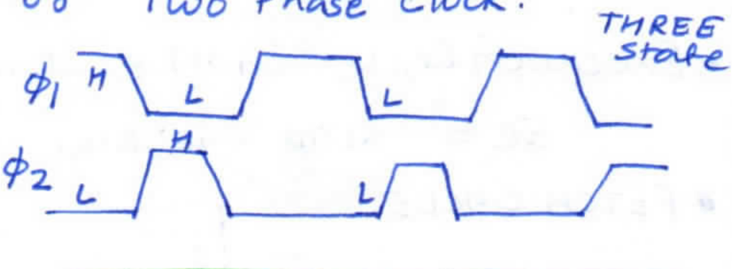


fig 2·7 b Two Phase CLK.

- One method is to generate clock externally and then apply to the μp.
- Alternative method is that clock circuitory is inside the μp but some components (crystal) which can not be fabricated within the IC has to be kept outside the IC

Unit II – 8-bit Microprocessors (Intel-8085)

CLOCK (Contd..)

- Under the control of the clock insts are sequencial fetch& executed.

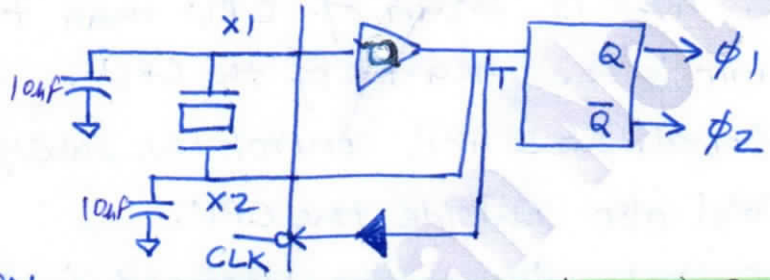


Fig 2·8 Clock circuitory in 8085

- INSTRUCTION EXECUTION

$$IC = FC + EC$$

IC- Instruction Cycle (Time required to execute one Instruction)

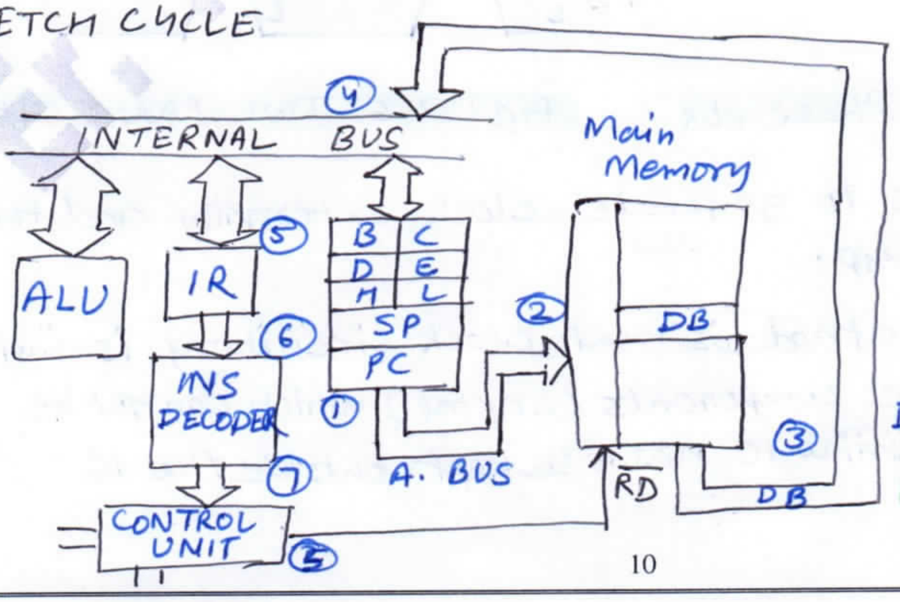
FC → Fetch Cycle (Fetching Instruction from the main memory then decoding it)

EC → Execution Cycle (Time Required to execute the instruction)

- Execution Cycle involve some

EC = READ CYCLE(S) + WRITE CYCLE(S)

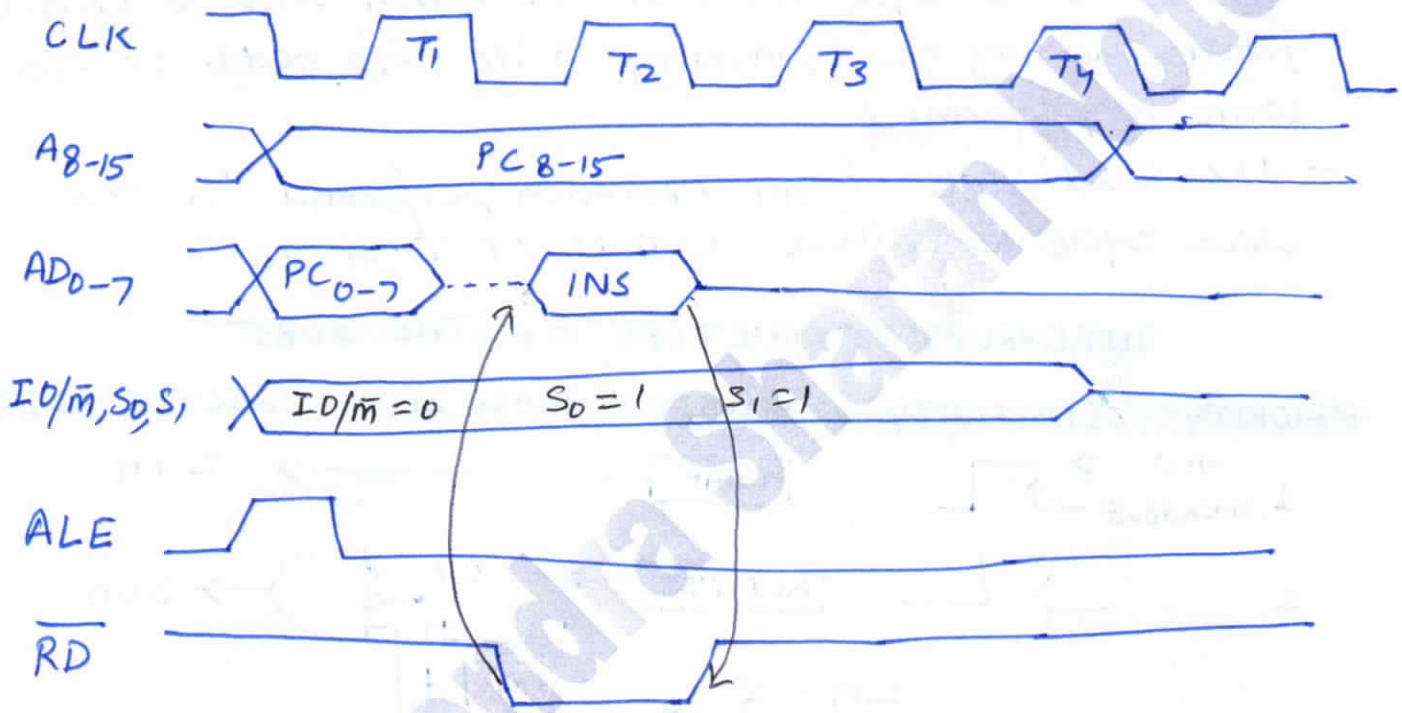
- FETCH CYCLE



Fetch Cycle is executed as the specified order: ①②③④⑤⑥⑦

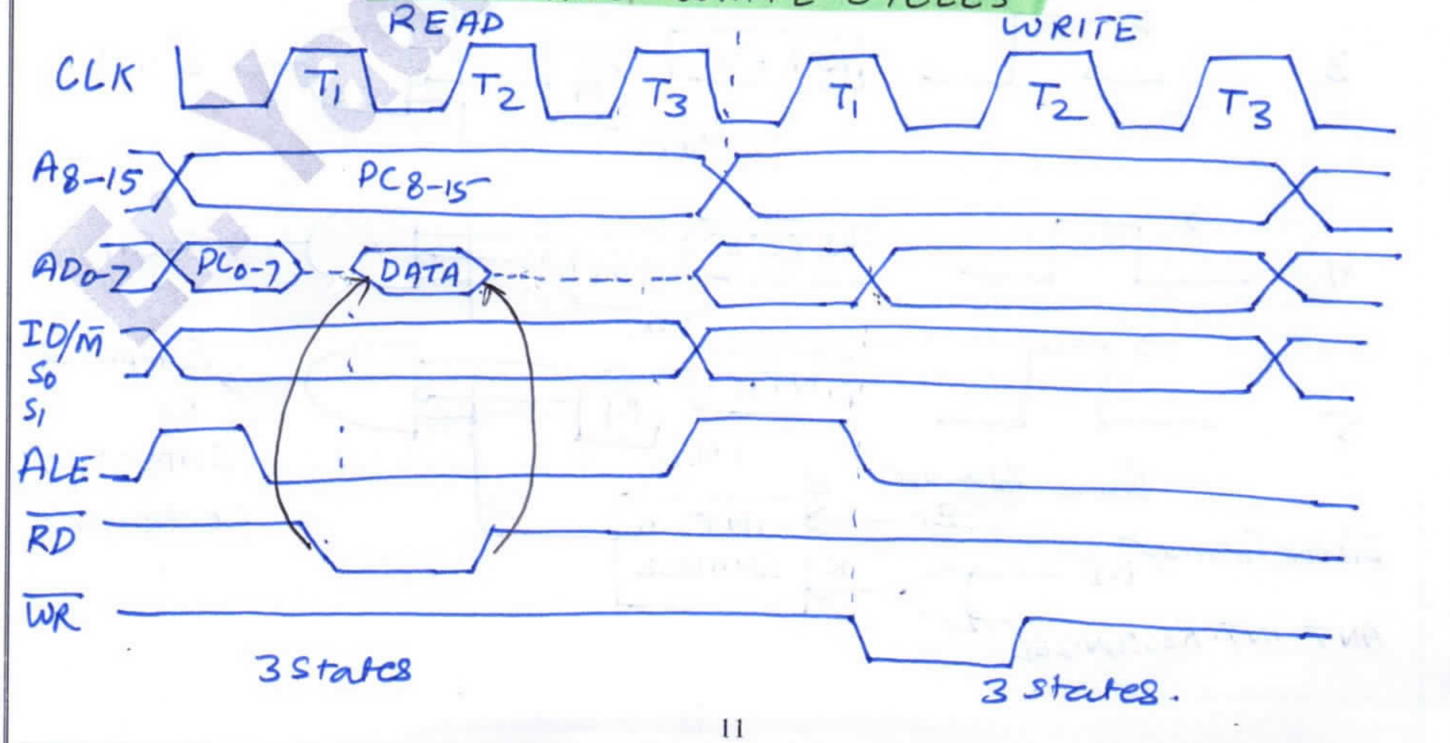
Fig 2·9 Fetch Cycle Execution

FETCH CYCLE TIMING



ONE MACHINE CYCLE 4 STATES

READ AND WRITE CYCLES

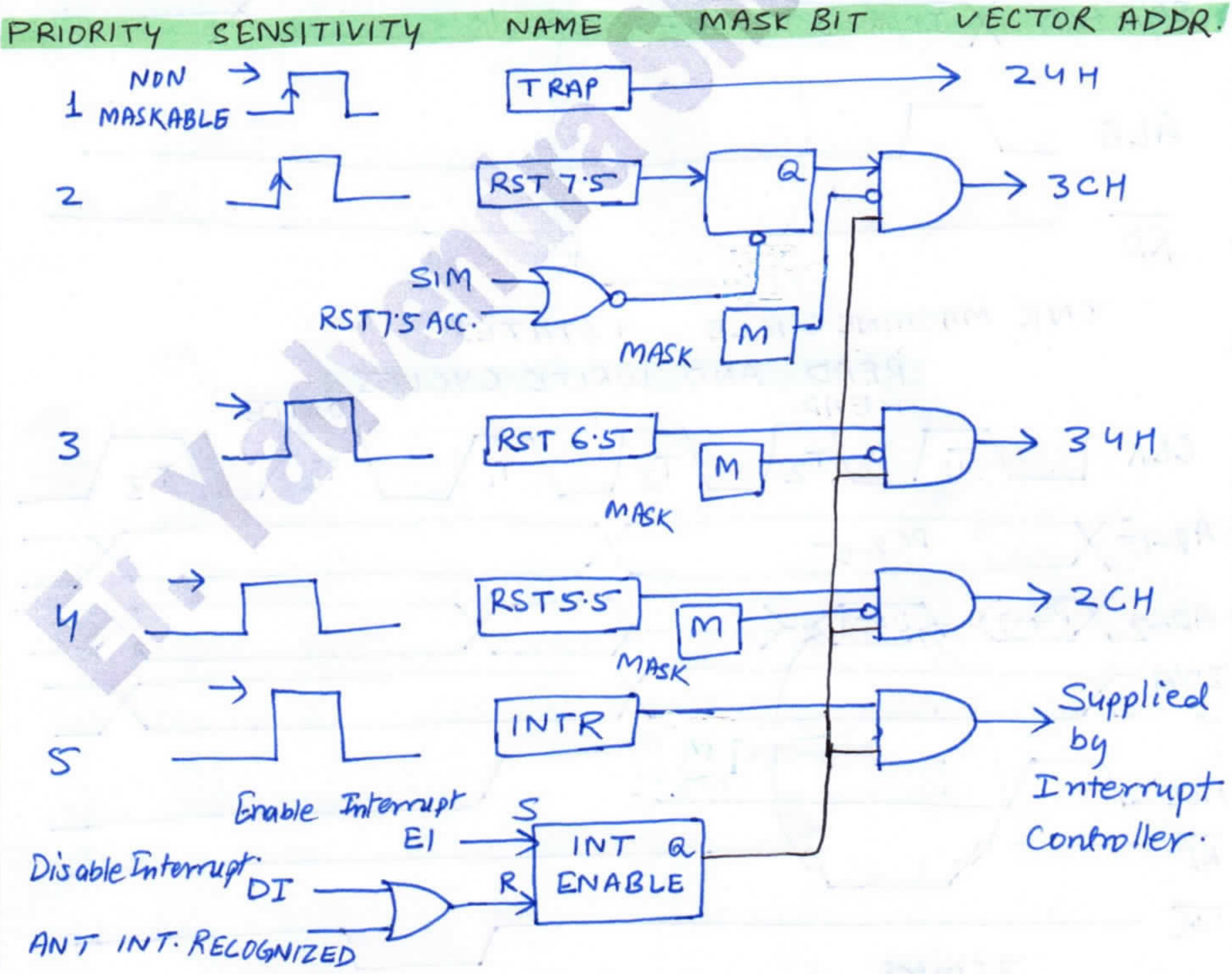


Unit II – 8-bit Microprocessors (Intel-8085)

Machine Cycle and State

- In a machine cycle one basic operation such as opcode fetch, memory read, memory write, I/o read or I/o write is performed.
- One subdivision of an operation performed in one clock cycle is called a state or T-state.

INTERRUPT STRUCTURE OF THE 8085



INTERRUPT STRUCTURE OF THE 8085

- when an interrupt line goes high processor completes its current instruction and saves program counter on the stack.
- It also reset Interrupt Enable flipflop before taking up ISS (Interrupt Service Subrautine) (stored on vector Address or fixed address) so that the occurrence of further interrupts by other devices is prevented during the execution of ISS.
- All the interrupts except TRAP are disabled by resetting the interrypt Enable flipflop.
- The interrupt which can be masked off (i.e. made ineffective) are called maskable interrupts. Masking is done by software.
- An interrupt for which hardware automatically transfers the program to a specific memory location is known as vectored interrupt.
- Other external device Interrupt Controller will provide the address where the CPU will jump whenever. INTR interrupt activated.
- INTR is used along with a special device interrupt controller hardware.

Unit II – 8-bit Microprocessors (Intel-8085)

BUS CONTROL RESPONSE

HOLD ← DMA Request

HLDA → DMA Grant   Then microprocessor leaves the controlover the buses as soon as the current machine cycle is complete and external device take control of the buses.

RESET RESPONSE

- When RESET is applied CPU will brought back to the predefined state.
- In 8085, a-It resets the PC to zero.
  b-it also resets interrupt enable and HLDA FF.
  c-it also resets the IR.
  d-it also resets the machine state FF
  e-it also reset the machine cycle FF
  f-it sets RST 7.5 MASK BIT, RST 6.5 MASK BIT
  RST 5.5 MASKBIT

STATE TRANSITION DIAGRAM

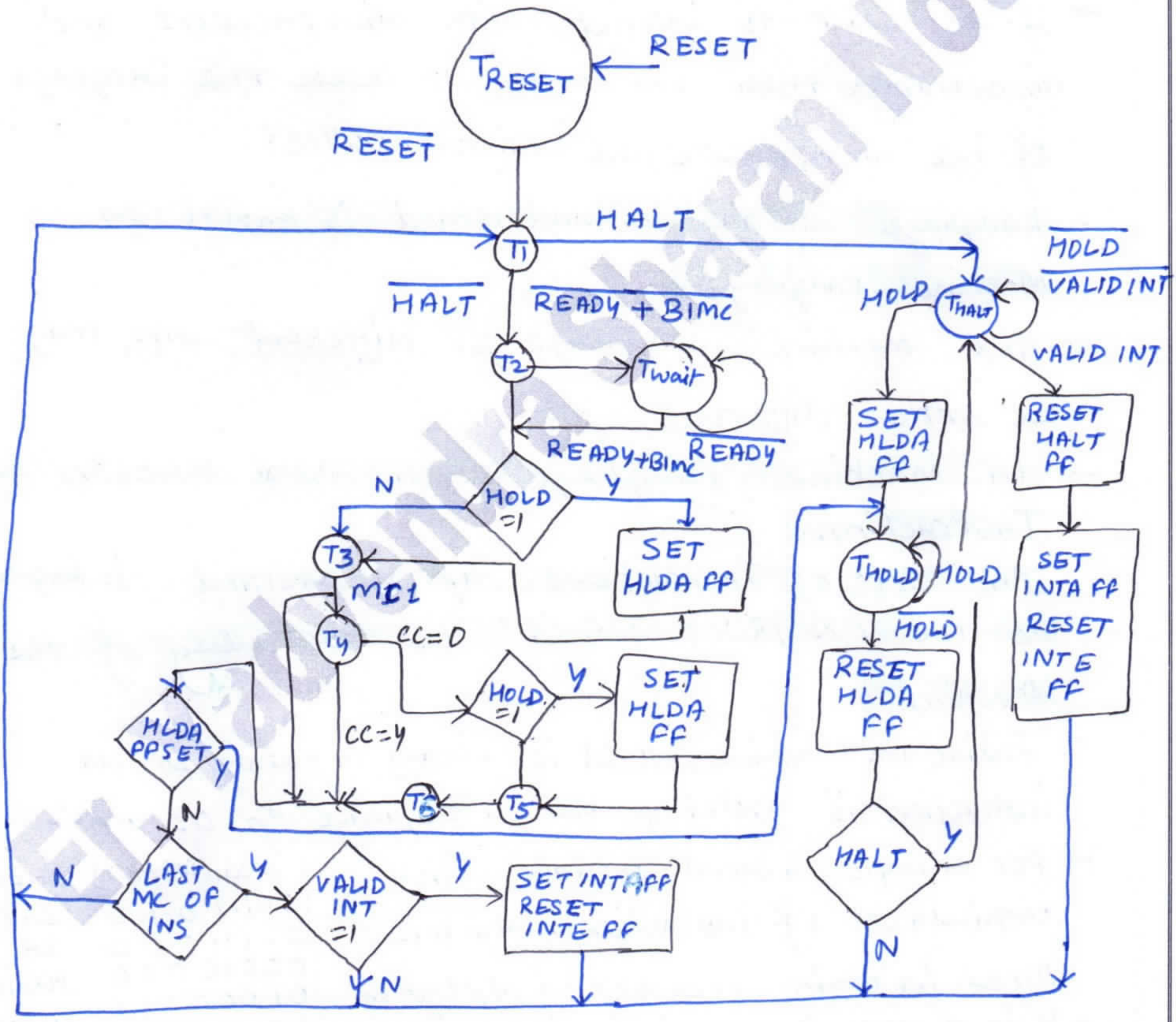


Fig 2.10 state transition diagram of 8085 A up.

Unit II – 8-bit Microprocessors (Intel-8085)

INSTRUCTION SET

THE VOCABULARY OF THE MACHINE

- If we want to interact with microprocessors and microcontroller then we have to learn the language of the microprocessors and microcontroller.
- Language µp and µC understand is known as Machine language.
- And machine language for different µps may be little different.
- In machine language each word known as Instruction.
- And set of Instructions µp understand is known as the "INSTRUCTIONSET" the vocabulary of the machine.
- Inside the machine, word is binary in nature so an instruction is nothing but a sequence of 0's and 1's..
- For example, a small program which comprises a set of instructions which is stored in main memory of µp system.
- Main memory is organized in terms of bytes (=8bits)
- Each memory location stores 8bits and instructions are stored one after the another memorylocation as shown in Fig 2'11

Main Memory	HEX
00111010	3A (H)
00111010	3A
00000000	Machine 00
01000111	Language 47
00111010	3A
00111011	3B
00000000	00
10000000	80
00110010	32
00111100	3C
00000000	00
01110110	76

Fig 2'11 Example of Program

Unit II – 8-bit Microprocessors (Intel-8085)

INSTRUCTION SET (Contd...)

- Machine Language is difficult to understand so for human interaction some steps are taken.
- One simple step toward this direction is to represent these instruction in "HEX NUMBER"
- HEX represent is also not good enough although it is represented in terms of some characters.
- So for better interaction these instruction are represented with the help of "MNEMONICS". In which each instruction is represented with the help of a symbol which specifies a particular operation.
- Requirement of proper programming.
    - WELL DEFINED FORMAT
    - DATA TYPES
    - ADDRESSING MODES

Main Memory	HEX	Mnemonics
00111010	3A H	LDA 003A (H) which represents Load Acc. with content of memory loc. 003A H
00111010	3A H	
00000000	00 H	
01000111	47 H	MOV B, A
00111010	3A H	LDA 003B H
00111011	3B H	
00000000	00 H	
10000000	80 H	ADD B
00110010	32 H	STA 003C H
00111100	3C H	
00000000	00 H	
01110110	76 H	HLT

Machine Language

Same thing may represent an opcode and data. so that a WELL DEFINED FORMAT is required.

Unit II – 8-bit Microprocessors (Intel-8085)

DATA AND ITS REPRESENTATION

- Something which conveys some meaning is data.
- Data is of two types
  - a- Non Numeric
  - b- Numeric

NonNumeric Data

- A sequence of characters
- can be represented by some code in computer by ASCII Code.
  - ASCII – American Standard code for Information Interchange.
- 7 bit in length, 128 different characters.
- All the alphabets AtoZ, lower and Upper Case, all the numbers (0-9), other characters (printable & Nonprintable)
- to store non numeric data we use ASCII code inside the uP.

Numeric Data.

-First of all we have to understand the number representation.

DECIMAL NUMBER SYSTEM

0 – 9

n- digit number

$a_{n-1} a_{n-2} \cdots a_1 a_0 \quad 9 \le a_i \le 0$

NOTE
can not be used inside uP because everything should be in the form 0's & 1's.

$N = a_{n-1} r^{n-1} + a_{n-2} r^{n-2} + \cdots + a_1 r^1 + a_0$

where r → radix

for Decimal Number System $r = 10$

For Example: $1949 = 1\times10^3 + 9\times10^2 + 4\times10 + 9$

Sign Representation
+ Positive No.
– Negative No.

Unit II – 8-bit Microprocessors (Intel-8085)

DATA AND ITS REPRESENTATION (Contd..)

BINARY NUMBER SYSTEM

SIGN MAGNITUTE RADIX = 2

$a_{n-1}r^{n-1} + a_{n-2}r^{n-2} + \ldots + a_1r^1 + a_0$ $a_i \rightarrow 0, 1$ $r = 2$

For Example

$(25)_{10} \equiv (11001)_2$

$25 = 1\times2^4 + 1\times2^3 + 0\times2^2 + 0\times2^1 + 1$

Sign Representation	
0	Positive
1	Negative

1's Complement Form (8-bit)

+25 ≡ 00011001

−25 ≡ 11100110

2's Complement Form (8bits)

+25 ≡ 00011001

−25 ≡ 11100110

$$\begin{array}{r} 11100110 \\ +\quad 1 \\ \hline 11100111 \\ \hline \end{array}$$

which is very popular inside µp field.

Advantages.

a- one zero (+0 & −0)

b - Able to perform Addition & subtraction with same hardware.

DECIMAL NUMBERS

- can be stored inside µp in following method:

XXXX 0001 — in Binary Coded Decimal Form UNPACKED

0100 ⋮ 0011 — PACKED

stored two BCD's

- 8 bit word D7 D6 - - - - - - - D0 (also known as LITTLE-ENDIAN convention.

D0 → LSB D7 → MSB D7 → sign Bit D6-D0 → 7 bit data

Unit II – 8-bit Microprocessors (Intel-8085)

INSTRUCTION FORMATS

OP CODE	SOURCE ADDR OP1	SOURCE ADDR OP2	DEST. ADDR.	ADDR. OF NEXT INS.

1 + 2 + 2 + 2 + 2 = 9 BYTES

- This will not only occupies large memory but it also takes longer time to execute the instruction.
- Basic objective of realistic instruction format is
  - o shorter size
  - o shorter Time to execute.
- To achieve this several techniques are followed.

1- To use PC (Program Counter)
ADDR. OF NEXT INS. is removed and PC will always hold the address of NEXT INSTRUCTION.
NOTE: some instructions where the next instruction to be executed is not in consecutive memory location. In such cases explicitly provide the address of the next instruction.

2- To use ACC (Accumulator)
To hold one of the operands
then SOURCE ADDR OP1 is not required as part of the instruction.

3- SOURCE AND DESTINATION ADDR same.
result is also stored in the ACC when one of the operands is stored in ACC.
then DEST ADDR did not provided as a part of INSTRUCTION.

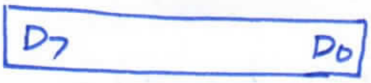
4- Use of General Purpose Registers | to store other operands in the GPRS.

Unit II – 8-bit Microprocessors (Intel-8085)

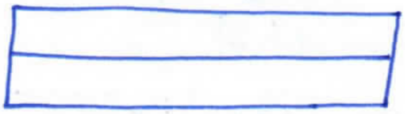
INSTRUCTION FORMATS OF 8085

- Broadly classified as follows:

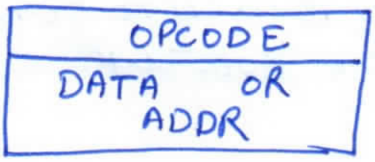
● By using Number of Bytes



SINGLE BYTE OPCODE

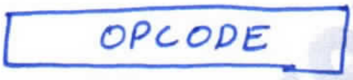


DOUBLE BYTE /TWO BYTE OPCODE DATA/ ADDR



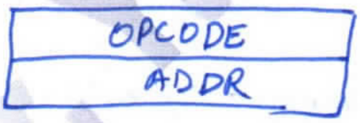
THREE BYTE

● By using NUMBER OF OPERAND ADDR

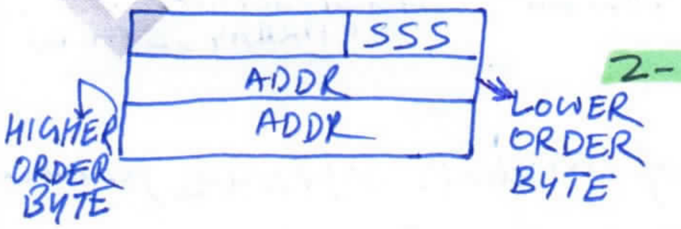


0-ADDR No need to explicitly specify the ADDR.

If the instruction is MONADIC in nature (only one operand is involved)



1-ADDR



2-ADDR One ADDR is specified in register {SSS -Source ADDR} another ADDR is speeified in two Bytes.

- According to Little Endian convention Lower order Byte store first.
- 2-ADDR Can be speeified by SINGLE/TWO/THREE BYTE.

Er. Yadvendra Sharan
M.Tech. (Hons.), B.Tech.
Assistant Professor
Department of Electronics & Communication Engineering
Phonics Group of Institutions, Roorkee.
Phone: +91-8273990016
yadvendra.sharan@yahoo.com

Subject: (TEE-404) Microprocessors & Its Application

Unit II – 8-bit Microprocessors (Intel-8085)

SUMMARY

DATA TYPES SUPPORTED BY 8085 μP

- 8-BIT 2's complement Integer Number.

D7 . . . . . D0
SIGN ... LSB

RANGE: −128 to +127
-2^{n-1} TO $+2^{n-1}-1$

PRECISION: $1:2^{8}$

- BCD NUMBERS IN Packed Form | XXXX ⁝ YYYY |
  - DAA (Decimal Adjust Accumulator) is used to convert Binarydata in BCD Packed Form.

INSTRUCTION FORMATS OF 8085 μP

- SINGLE BYTE
- TWO BYTE
- THREE BYTE

ADDRESSING MODES OF μPs

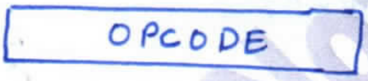
GOALS

- To access Large Address Space.
- Specify Address with smaller number of bits.
- Easy implementation of popular data structures. (ARRAY, STACK)
- Fast and simple
- Easy Implementation of Modern Software features (Position Independency, Reentrancy)

Unit II – 8-bit Microprocessors (Intel-8085)

COMMON ADDRESSING MODES

- INHERENT O-ADDRESS

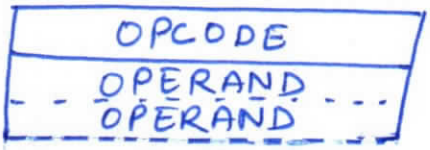


Only the opcode is provided and address is implecitly used.

e.g. MONADIC Instruction HLT, NOT/COMPLEMENT then no requirement of address

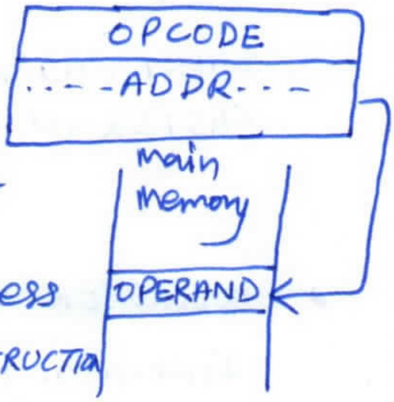
- IMMEDIATE

operand itself is provided as part of instruction then no address is required.



For CONSTANT DATA VALUES or INITIALIZATION OF SOME REG.

- DIRECT/ABSOLUTE Address of operand is provided.



– Most natural, Simple (does not involve any computation)

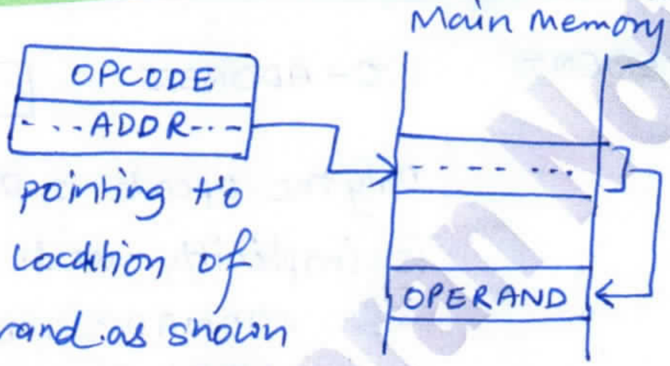
– Depending upon the size of address SINGLE/TWO/THREE BYTE INSTRUCTION is used.

EA = A

Effective Address is Memory Location provided in instruction.

COMMON ADDRESSING MODES (Contd...)

- INDIRECT



- SLOW
- FEEXIBLE

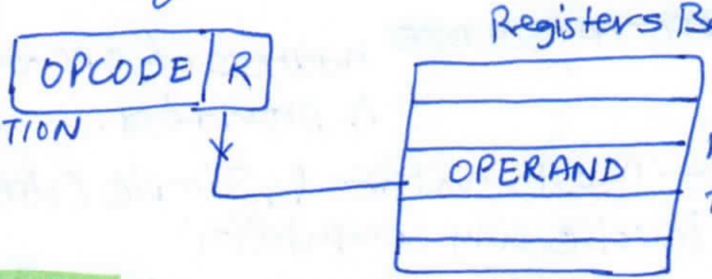
ADDR is pointing to memory location of the operand as shown in figure.

EA = (A)

Effective Address is content of memory Location provided as part of instruction.

- REGISTER
  - to specify address with smaller number of bits.
  - some of the operands canbe stored register itself instead of storing them in main memory.

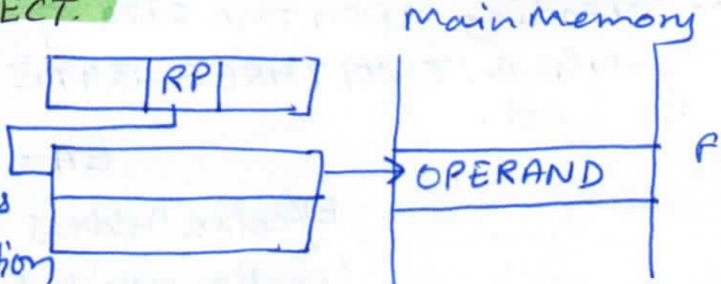
SHORT PROGRAM
FASTER IN EXECUTION



operand has to transfer to reg. bfore

- REGISTER INDIRECT.

Instruction has theADDR of Reg. Pair and RP has the memory location of operand asshown in figure.

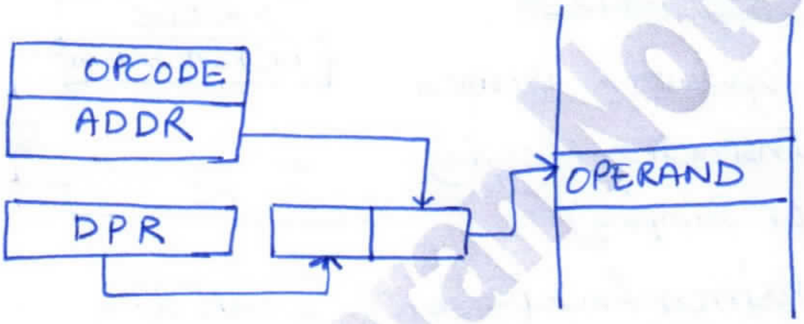


FLEXIBILITY

Unit II – 8-bit Microprocessors (Intel-8085)

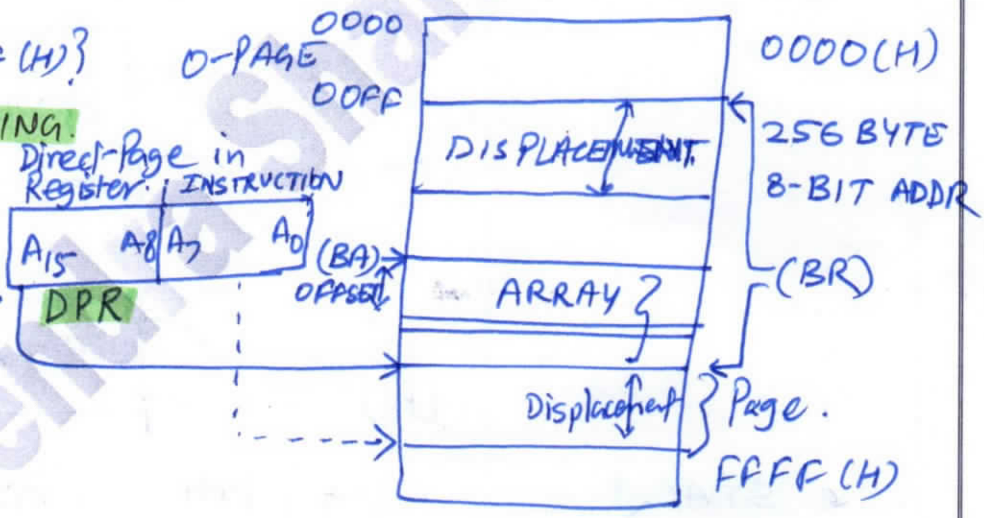
COMMON ADDRESSING MODES (Contd...)

- **PAGED**



- FAST
- without DPR still we can use PAGED ADDRESSING
- O-PAGE ADDR {From 0000(H) to 00FF (H)}

NOTE-PAGING

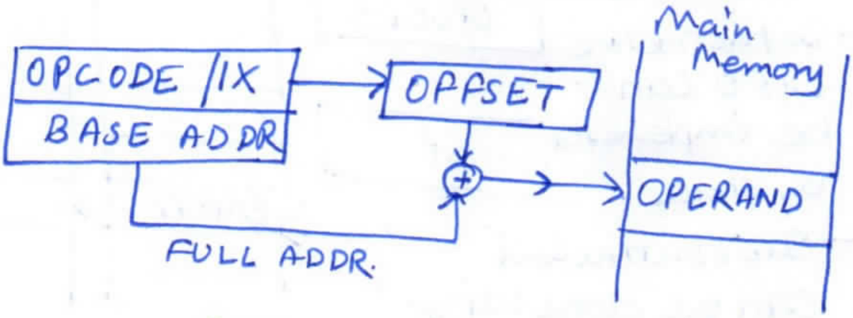


When DPR is not available Lower order BYTE has the ADDR on O-PAGE.

- **INDEXED**

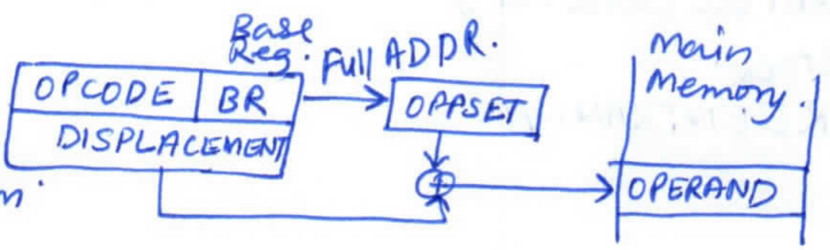
$EA = (IX) + BA$

- used to implement an ARRAY
- AUTO DEC/INC



- **BASED**

- Purpose of Based ADDR is different from Indexed ADDR.
- Commonly used for the purpose of Relocation.

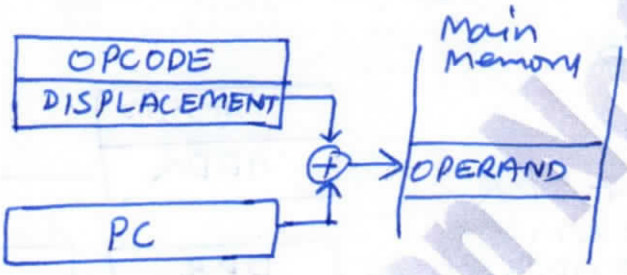


- **BASED-INDEXED** has the combined effect of Indexed and Based ADDR.

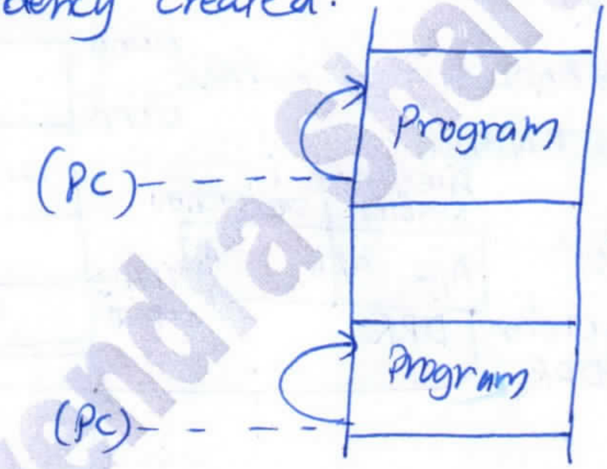
Unit II – 8-bit Microprocessors (Intel-8085)

COMMON ADDRESSING MODES (Contd...)

- RELATIVE

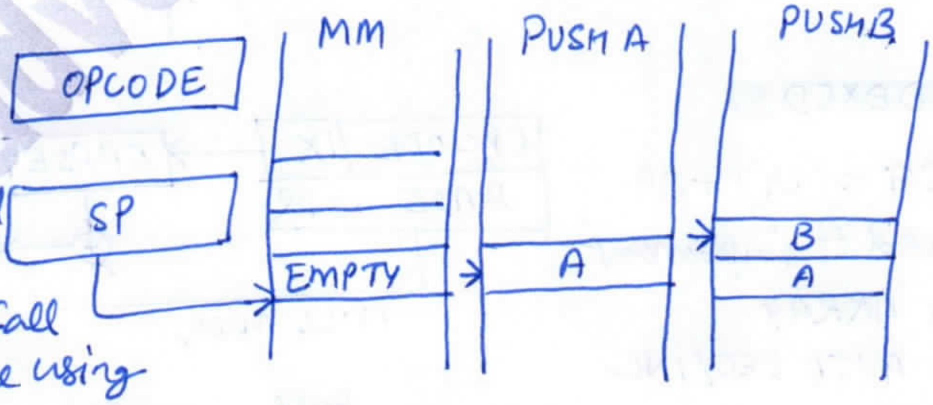


- Popular in Branch Instruction (Looping or Jumping)
- Position Independency created.



Program can be shifted using displacement and Looping is still present...

- STACK



- Datastructure LIFO can be implemented by stack.
- Subroutine Call can be done using stack.
- REENTRANCY

ADDRESSING MODES OF 8085

- Inherent
- Immediate
- Direct
- Register
- Register -Indirect
- Stack

Later on we will discuss how various addressing modes, Data type and instruction formats are combined to realize the INSTRUCTION SET of 8085.

INSTRUCTION SET DESIGN

Parameters involved in design of Instruction Set.

- Operations supported by the ALU
- Data Types
- Instruction Format
- Registers Available
- Addressing Modes

CLASSIFICATION OF INSTRUCTIONS

- Data Movement Instructions
  Instruction that move data between registers, between register and memory location and I/O transfer.
- Data Manipulation Instructions
  Depending upon the capability of ALU, operations it can perform.
- Program Manipulation Instructions
  with the help of these, user can change the flow of program.
- Status Manipulation Instructions
  which changes the status of the processor.
- Miscellaneous Instructions

Unit II – 8-bit Microprocessors (Intel-8085)

DATA MOVEMENT INSTRUCTIONS

- BETWEEN TWO REGISTERS

Example

01	DDD	SSS

Single Byte

DDD/SSS	REG.
111	A
000	B
001	C
010	D
011	E
100	H
101	L

MOV r_1, r_2 $[r_1] \leftarrow [r_2]$

CYCLES : 1 (Fetch Cycle)
STATES : 4

- IMMEDIATE OPERAND TO REG.

MVI r, data

00	DDD	110
DATA		

CYCLES: 2 (Fetch + Read Cycle)
STATES: 7

LXI rp, data

00	RP	0001
Lower Order Byte		
Higher Order Byte		

CYCLES: 3 (Fetch + 2 Read Cy(s))
STATES: 10

- BETWEEN A REG. AND A MEMORY LOCATION

MOV r, M

01	DDD	110

[r] ← [[H-L]] CYCLES: 2 (Fetch+Read)
STATES: 7

▶ It will use Register Indirect ADDR.

▶ The content of the memory location, whose address is in (H)(L) pair, is moved to register r

MOV M, r

01	110	SSS

[[H-L]] ← [r]

CYCLES: 2 (Fetch + Read Cycles)
STATES: 7

In case of Direct addressing.

Load ACC Direct

LDA addr.

[A] ← [(byte3)(byte2)]
addr

01 111 010
← ADDR →

CYCLES: 4 [Fetch+3Read]
STATES: 13

Store ACC Direct

STA addr
[Addr] ← [A]

01 110 010
← ADDR →

CYCLES: 4
STATES: 13

DATA MOVEMENT INSTRUCTIONS (Contd..)

→ special instruction.

Load HL pair direct — LHLD addr — [L]← [addr], [H]←[addr+1]
MACHINE CYCLE = 5 State = 16

Store HL pair direct — SHLD addr — [addr]←[L] [addr+1]←[H]
CYCLE = 5 State 16

Load ACC indirect — LDAX rp — [A]←[[rp]]
CYCLE = 2 State = 7

Store ACC indirect — STAX rp — [[rp]]←[A]
CYCLE = 2 State = 7

Exchange — XCHG — [H-L]↔[D-E]
CYCLE = 1 State = 4

→ Between ACC and I/O Location

IN Port

11011011
port addr.

CYCLE = 3
State = 10

OUT port

11100011
port addr

CYCLE = 3
STATE = 10

NOTE: μp is provided with single bus so on that single bus, the CPU can generate an address that can corresponds to either Memory location or IO Location. So no transfer of data takes place between a memory to IO so a special category of addressing mode (i.e. DMA mode is used)

Unit II – 8-bit Microprocessors (Intel-8085)

DATA MANIPULATION INSTRUCTIONS

ARITHMETIC INSTRUCTIONS

ADD	ADD r [A]←[A]+[r]	ADD M [A]←[A]+[[H L]]	ADI data [A]←[A]+data
with (CY)	ADC r [A]←[A]+[r]+[CS]	ADC M [A]←[A]+[H-L]+[CS]	ACI data [A]←[A]+data+[CS]
SUB	SUB r [A]←[A]-[r]	SUB M [A]←[A]-[H-L]	SUI data [A]←[A]-data
with borrow	SBB r [A]←[A]-[r]-[CS]	SBB M [A]←[A]-[[HL]]-[CS]	SBI data [A]←[A]-data-[CS]

	C= 1
	S= 4
	reg.
		C=2
		S=7
		reg. indirect
			C=2
			S=7
			immediate

INR	INR r
	[r]←[r]+1
		C=1
		S=4
			INR M
			[[H-L]]←[[HL]]+1
				C=3
				S=10

DCR	DCR r
	[r]←[r]-1
		C=1
		S=4
			DCR M
			[[HL]]←[[HL]]-1
				C=3
				S=10

INX rp C=1 S=6
[rp]←[rp]+1

DCX rp C=1 S=6
[rp]←[rp]-1

DAD rp C=3 S=10
[HL]←[HL]+[rpH, rpL]

- Special Instruction C=1 S=4

DAA (Decimal adjust Accumulator)
Instruction's Result is in hexadecimal and The DAA instruction operates on this result and gives the final result in decimal system

```
A =  37BCD    0011 0111
    +15BCD    0001 0101   ADD
    -----     ---------
     52BCD    0100 1100
                 4    C   (H)
              0100 1100   DAA
              0000 0110
              ---------
              0101 0010
     52BCD       5    2
```

DATA MANIPULATION INSTRUCTIONS (contd..)

LOGICAL INSTRUCTIONS

A	B	AND	EXOR	OR
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	0	1

ANA r	ANA M	ANI data
XRA r	XRA M	XRI data
ORA r	ORA M	ORI data

COMPARE

CMP r	CMP M	CPI data

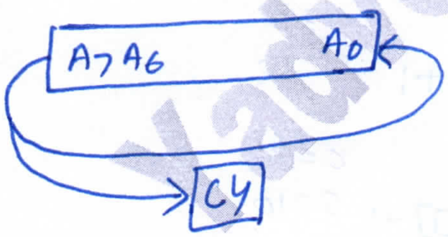
C=1
S=4
	C=2
	S=7
		C=2
		S=7

COMPLEMENT

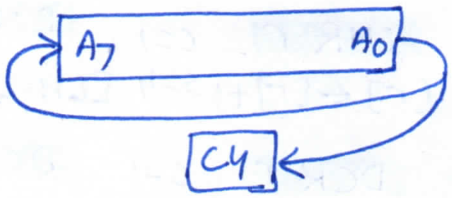
CMA [A] ← [$\bar{A}$]
C=1
S=4

ROTATE/SHIFT

RLC

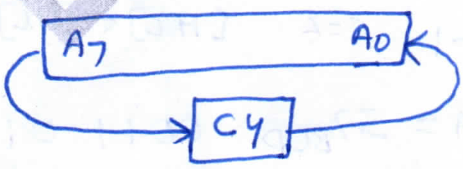


RRC

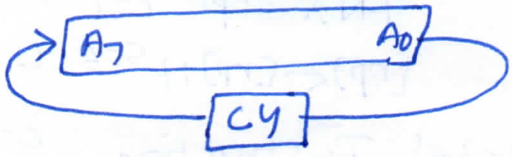


C=1
S=4

RAL



RAR



PROGRAM MANIPULATION INSTRUCTIONS

- Unconditional Jump] Branch Group
- Conditional Jump]
- Subroutine Call and Return
- Software Interrupts

BRANCH GROUP

This group change the normal sequence of the program (In sitnation when instruction has to come some other memory location different from provided by PC)

UNCONDITIONAL

JMP addr

11000011
ADDR
ADDR

[PC]←[byte3byte2]
Immediate Addr.
C=3
S=10

CONDITIONAL

Condition	CCC	Mnemonic
NZ (Z=0)	000	JNZaddr
Z (Z=1)	001	JZ addr
NC (CY=0)	010	JNC addr
C (CY=1)	011	JC addr
PO (P=0)	100	JPO addr
PE (P=1)	101	JPE addr
P (S=0)	110	JP addr
M (S=1)	111	JM addr

11 CCC 010
ADDR
ADDR

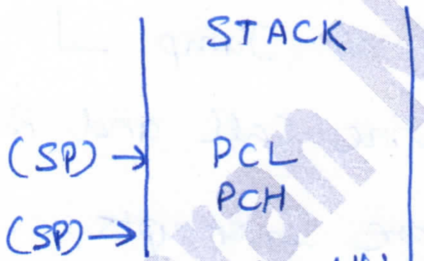
C=2/3
S=7/10

PROGRAM MANIPULATION INSTRUCTIONS

SUBROUTINE CALL AND RETURN

CALL addr

[[SP]-1] ← [PCH]
[[SP]-2] ← [PCL]
[SP] ← ([SP]-2)
[PC] ← addr
(byte 3)(byte 2)



11 00 1101
← ADDR →

CYCLE=5
STATES=18

Conditional Call

11 CCC 100
← ADDR →

C=2/5
S=9/18

Conditional CALL Instructions

CC addr   CNC addr   CZ addr   CNZ addr
CP addr   CM addr   CPE addr   CPO addr

RET (Return from Subroutine)

11 001000

CYCLE=3
STATES=10

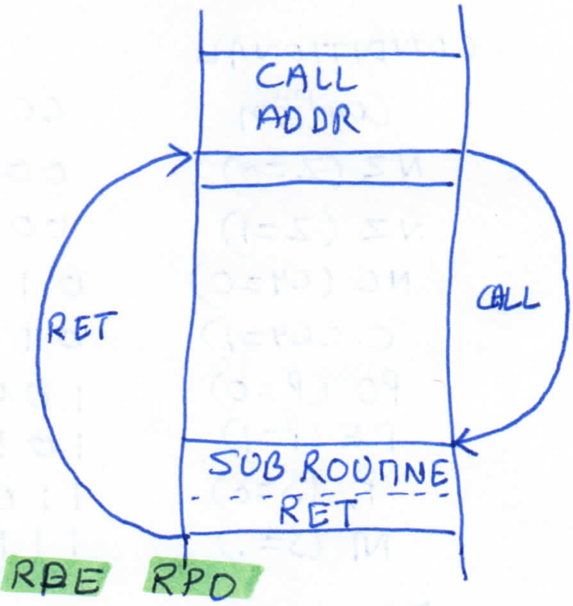
[PCL] ← [[SP]]
[PCH] ← [[SP]+1]
[SP] ← ([SP]+2)

Conditional RETURN

11 CCC000

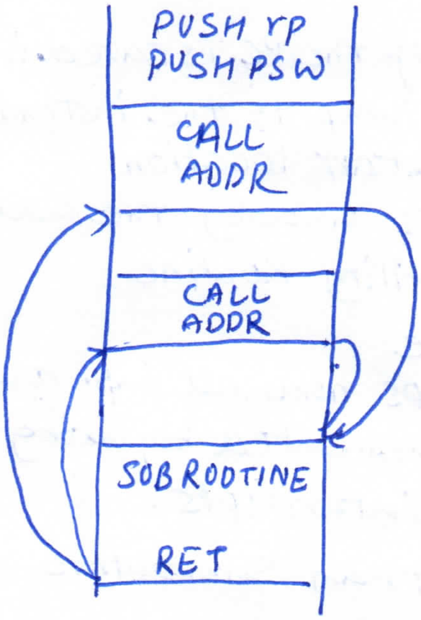
CYCLE=1/3
STATE=6/12

RNC RC RNZ RZ RP RM RPE RPO



Unit II – 8-bit Microprocessors (Intel-8085)

SUBROUTINE CALL AND RETURN (Contd..)



Nested Subroutine Call : size of stack

PUSH rp [[SP]-1] ← [rh]
[[SP]-2] ← [rl]
[SP] ← ([SP]-2)
C=3
S=12

PUSH PSW [[SP]-1] ← [A]
[[SP]-2] ← PSW
[SP] ← ([SP]-2)
C=3
S=12

POP rp [rl] ← [[SP]]
[rh] ← [[SP]+1]
[SP] ← [SP]+2]
C=3
S=10

POP PSW PSW ← [[SP]]
[A] ← [[SP]+1]
[SP] ← [SP]+2]
C=3
S=10

XTHL [L] ↔ [[SP]]
[H] ↔ [[SP]+1]
C=5
S=16

SPHL [H-L] ← [SP]
C=1
S=6

SOFTWARE INTERRUPT

HARDWARE INTERRUPTS : Commonly used for Interrupt driven Data Transfer

SOFTWARE INTERRUPTS : Essentially Instructions.
whenever a hardware Interrupt occurs the way the processor response the same way processor will behave whenever Interrupt Instruction is executed.

SOFTWARE INTERRUPTS (Contd..)

RST n $0 \le n \le 7$ one word CALL Instruction.

[[SP] -1] ← [PCH]
[[SP]-2] ← [PCL]
[SP] ← [[SP]-2]
[PC] ← 8×NNN

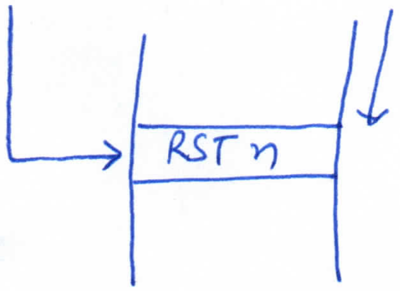
$\boxed{11 \quad NN \quad N \quad 111}$

CYCLE = 3
STATE = 12

The content of the PC is saved in the SP
The program jump to the instruction starting at restart location.
The location is ushally the address of an OS Utility Routine
For example
BIOS holds all imp. rowtine
BIOS is implemented by using software Interrupts.

- Software Interrupts are faster than subroutine
- Addressesare fixed or vector ADDR.
- Implementing Break Point, software Interrupts are used.

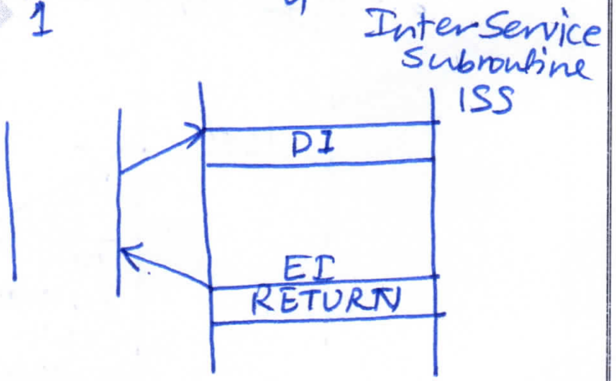
Instruction	NNN	Opcode	Restart location
RST 0	000	C7	0000
RST 1	001	CF	0008
RST 2	010	D7	0010
RST 3	011	DF	0018
RST 4	100	E7	0020
RST 5	101	EF	0028
RST 6	110	F7	0030
RST 7	111	FF	0038



Unit II – 8-bit Microprocessors (Intel-8085)

STATUS MANIPULATION INSTRUCTION

		CYCLE	STATE
• EI	1 1 1 1 1 0 1 1 Enable Interrupt	1	4
• DI	1 1 1 1 0 0 1 1 Disable Interrupt	1	4
• CMC	0 0 1 1 1 1 1 1 Complement the carry status	1	4
• STC	0 0 1 1 0 1 1 1 Set carry [CY]←1	1	4



MISCELLANEOUS INSTRUCTIONS

		CYCLE	STATES
• HLT Last Inst. Halt	0 1 1 1 0 1 1 0	1	4
• NOP	0 0 0 0 0 0 0 0 No operation	1	4
• RIM Read Interrupt Mask	0 0 1 0 0 0 0 0	1	4
• SIM Set Interrupt Mask	0 0 1 1 0 0 0 0	1	4

RIM:

SID	I7·5	I6·5	I·5·5	IE	M7·5	M6·5	M5·5

SIM: Mask bit can be set if MSE=1

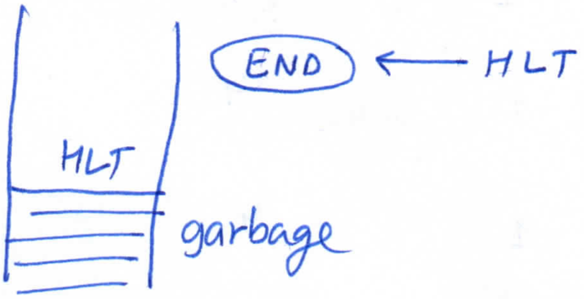
SOD	SOE	X	R7·5	MSE	M7·5	M6·5	M5·5

RIM & SIM are dual purpose Inst. used to provide serial mode of data transfer and to set or Read the Mask bit

NOTE

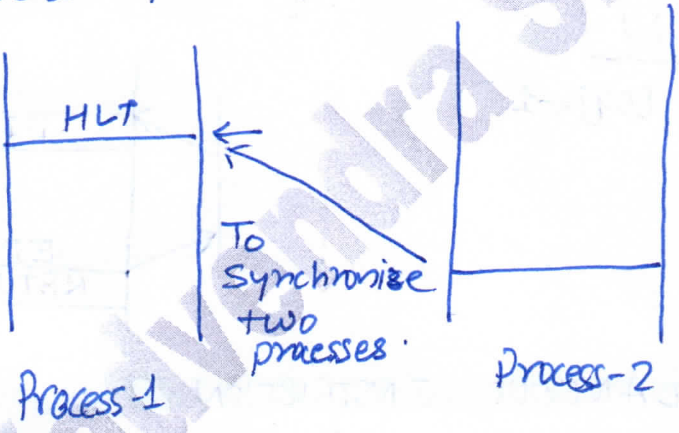
Use of Halt

- To use it as the last instruction



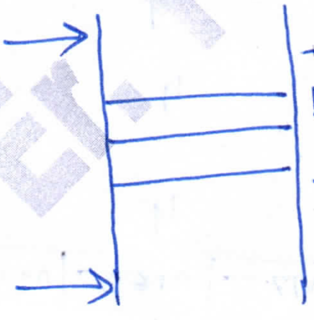
whenever HLT is executed no further instruction will executed and garbage codes at the end of program will not be executed

- Process Synchronization



processor will remain in HLT state untill a valid Interrupt so Halt is also called as wait for Interrupt.

Use of NOP : in Debugging & to provide Delay



- to replace unwanted Instruction after Debugging we can use NOP

- Similary we can also keep NOP instruction intentionally and afterDebugging it will provide facility to insert any instruction.

M7·5, M6·5, M5·5 } Mask bit   MSE — Mask Set Enable

R7·5 → Reset RST 7·5   SOE → Serial out Enable.   SOD → Serial output Data

I7·5, I6·5, I5·5 } Interrupt Pending status   X → Unused.

IE → Interrupt Enable Flag   SID → serial Input Data

Unit II – 8-bit Microprocessors (Intel-8085)

SYMBOLS AND ABBREVIATIONS.

Symbol / Abbreviation	Meaning
addr	16-bit address of the memory location
data	8-bit data
data 16	16-bit data
r, r_1, r_2	One of the registers A, B, C, D, E, H or L
A, B, C, D, E, L	8-bit register
A	Accumulator
H-L	Register pair H-L
B-C	Register pair B-C
DE	Register pair D·E
PSW	Program Status Word
M	Memory whose add. is in H-L pair
H	Appearing at the end of a group of digits specifies Hexadecimal e.g. 0038 H
rp	One of the register pairs B represents B-C pair D represents D-E pair H represents H-L pair SP represents 16-bit stack
rh	The high order register of a reg. pair
rl	The Low order reg. of a reg. pair
PC	Program Counter
CS	Carry Status.

SYMBOLS AND ABBREVIATIONS

Symbol / Abbreviation	Meaning
[ ]	The content of reg. identified within bracket
[[ ]]	The content of memory location whose address is in the reg. pair identified within brackets.
∧	AND operation
∨	OR operation
∀ or ⊕	Exclusive -OR
←	Move data in the direction of arrow
⇔	Exchange contents.

x