Microprocessor- unit I

What is Microprocessor?

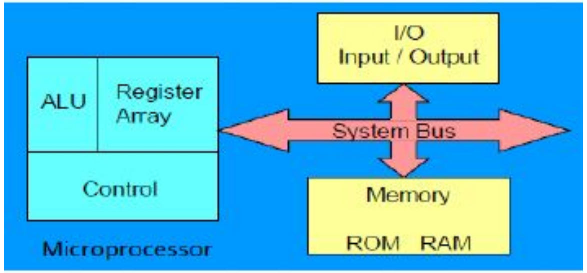
The microprocessor is a programmable integrated device that has computing and decision making capability similar to that of the central processing unit(CPU) of a computer.

The microprocessor is a multipurpose, programmable, clock-driven, register-based electronic device that reads binary instructions from a storage device called memory, accepts binary data as input and processes data according to those instructions, and provides results as output.

The microprocessor communicates and operates in binary numbers 0 and 1, called bits.

Each microprocessor has a fixed set of instructions in the form of binary patterns called a machine language. As it is difficult for the humans to communicate with 0s and 1s , the instructions in the form of mnemonics, which form the assembly language for a given microprocessor.

Microprocessor based system



ALU - Arithmetic and logical operations like add , subtraction etc
Register array - store temporary data during program execution
Control unit - provides necessary timing and control signals. It controls the flow of data between microprocessor and peripherals
Memory

- Stores information such as instructions and data in binary format(0 and 1)
- Sub-system of microprocessor-based system, sub-system includes the registers inside the microprocessors.

Read only Memory (ROM) : used to store programs that do not need alterations
Random Access Memory(RAM) : used to store programs that can read and altered like programs and data

Microprocessor based system .. cont..

- Input / output : communicates with the outside world. keyboard, printer etc
- System bus : Communication path between the microprocessor and peripherals (group of wires)

Semiconductor memory

With the invention of semiconductor technology, Integrated circuits plays a key role in designing microprocessor.

An entire circuit consisting of several transistors, diodes and resistors, logical gates can be designed in a single chip.

Small scale integration(SSI)- one logical gate in single chip

Medium -Scale Integration (MSI) – 100 logical gates in a single chip

Large Scale Integration (LSI) – 1000 gates in one chip

Very large scale integration (VLSI) – and Super Large scale integration (SLSI) –millions of gates in single chip

How does a microprocessor work?

- The instructions are stored sequentially in the memory.
- The microprocessor fetches the first instruction from memory, decodes it, and executes that instruction.
- The sequence is of machine cycle is
    - Fetch
    - Decode
    - Execute

The above sequence is continued until all instructions are performed

Machine language

1. The computer can understand only binary (0s and 1s) language(machine language). So the programmer has to communicate only in-terms of 0s and 1s.
2. But it is difficult to do so.
3. Thus the computer manufacturers have devised english-like words to represent the binary instructions of a machine.
4. These instructions are called as assembly language programs using these words.
5. The assembly language are specific to one machine cannot be executed or not transferable to other machine.
6. To overcome this general-purpose language or high-level language like C++, python are devised. So that these can be executed in any machine.
7. Microprocessor 8085 has 8 bit data so 2^8 = 256 combinations.

   For example :

   ➔ **0011 1100** – is an instruction that increments the number in accumulator by 1

   ➔ **1000 0000**– is an instruction that add the number in register B to the accumulator content , and keep the result in A

   So it is very difficult to remember and write.. For convenience, written in Hexadecimal code. For example **0011 1100 = 3C** $_H$ in hexa decimal

   and **1000 0000 = 80** in hexa decimal

Assembly Language

The program written in Hexa decimal is also difficult to follow. So there comes the mnemonic.

For example **0011 1100 = 3C** $_H$ in hexa decimal and **INR A** in assembly language
1000 0000 = 80 in hexa decimal and **ADD B** in assembly language
INR A - INR stands for Increment, A is accumulator.. This says that the content of accumulator is incremented by 1.
Similarly :
ADD B- ADD stands for addition and B represents content in register B. This says the addition of number in register B to the accumulator A content and store the result in A

So MP has 246 such bit pattern amounting to 74 different instruction for performing various operations.

Both machine language (binary) and assembly language are considered as low-level languages.

High Level Languages

How the english like words in high level languages are converted to binary languages of different microprocessors?

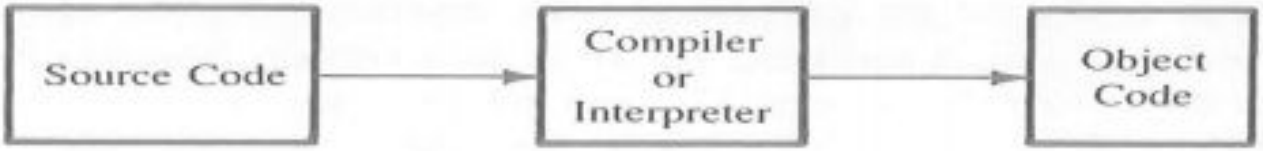
The answer is, through another program called compiler or interpreter.

These programs accept high-level language as their input (source code).

Then translates the source code into machine language for the respective microprocessor.

This translation is called the object code.

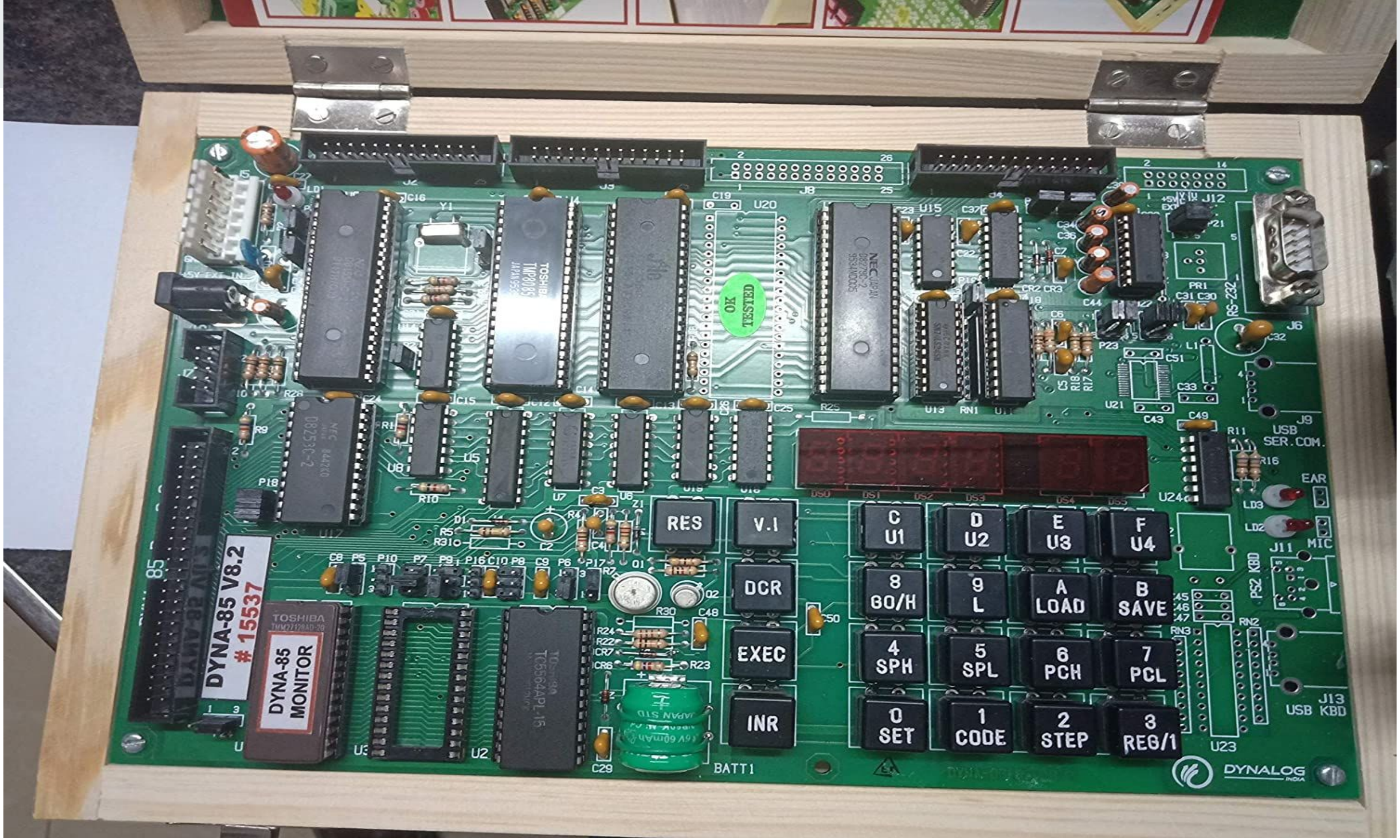
Trouble shooting (debugging) programs is easy in high-level language then in assembly language



8085 Microprocessor Architecture

8085 microprocessor is

- Developed by INTEL corp.
- General purpose 8-bit microprocessor
- 16 address lines , hence it can address $2^{16} = 65536$= 64 K memory
- 40 pins
- +5V single power supply
- - 3MHz to 5 MHz single-phase clock.
- It generates 8 bit I/O addresses, hence it can access $2^8 = 256$ I/O ports
- 5 hardware interrupts i.e TRAP, RST6.5, RST5.5, RST4.5 and INTR
- It provides DMA


DYNA-85 V8.2
# 15537
DYNA-85
MONITOR
TOSHIBA
TMP8085
RES
V.I
DCR
EXEC
INR
C U1
D U2
E U3
F U4
8 GO/H
9 L
A LOAD
B SAVE
4 SPH
5 SPL
6 PCH
7 PCL
0 SET
1 CODE
2 STEP
3 REG/1
OK TESTED
RS-232
USB SER.COM.
USB KBD
PS2 KBD
EAR
MIC
BATT1
DYNALOG INDIA

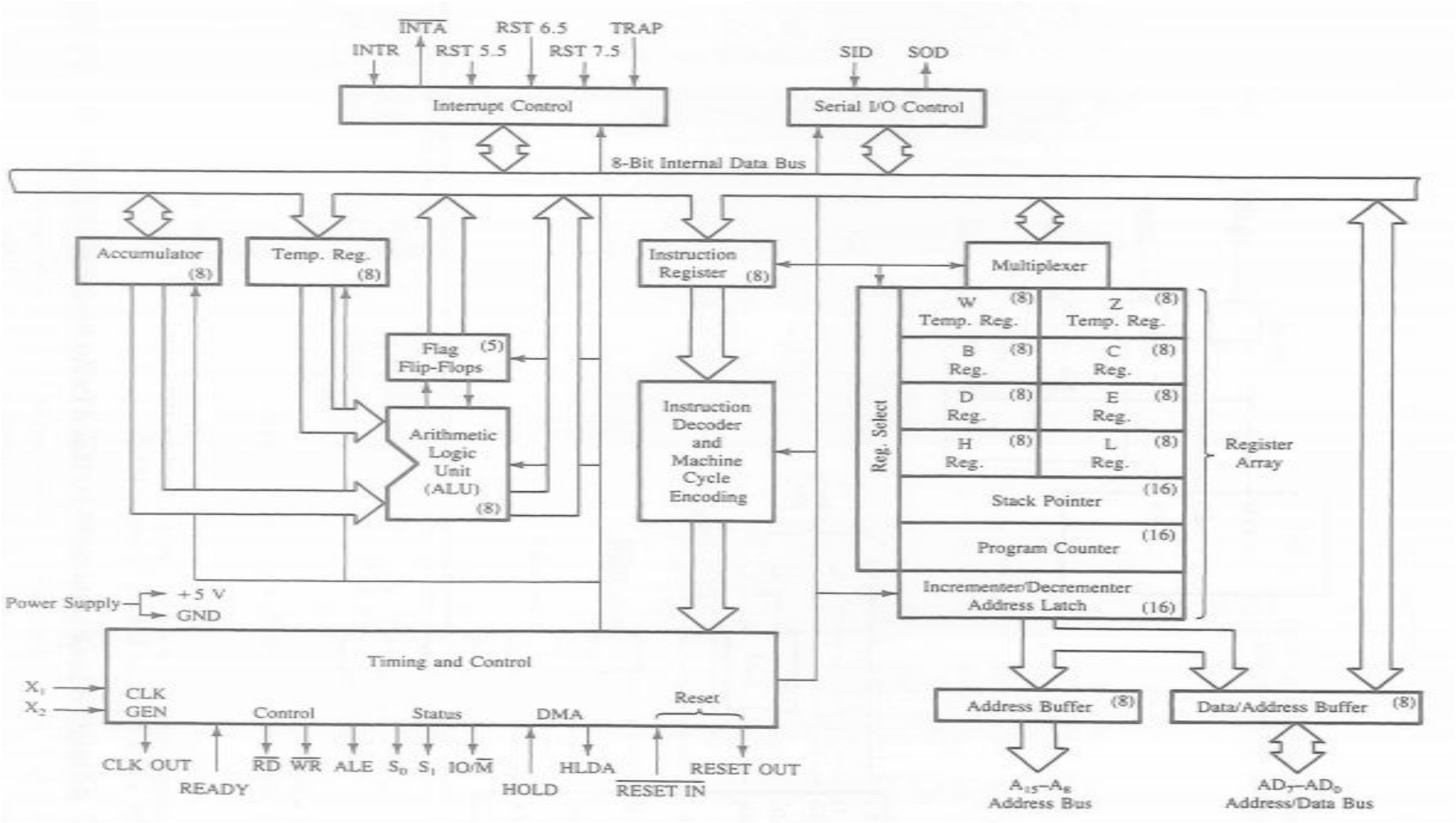
Latest microprocessor of intel

www.shutterstock.com · 437211229

Video of how a cpu works

💻 - See How a CPU Works

Internal Architecture (functional block diagram) of 8085



8085 Architecture .. cont..

8085 architecture consists of the following blocks:

1. Register Array
2. ALU & Logical Group
3. Instruction decoder and machine cycle encoder, Timing and control circuitry
4. Interrupt control group
5. Serial I/O control group

Slide 13

8085 Architecture .. cont..

1. Register Array : 14 registers (12- 8bit registers , 2 - 16bit registers)

   They are classified into 4 types as

(a) General purpose register[B,C,D,E,H,L]

(b) Special purpose register [Accumulator (A), Instruction Register and Flag register]

(c) Temporary Register [W, Z, temporary data register]

(d) Pointer Register or special purpose [PC, SP]

Slide 13

Register array- general purpose registers , special purpose register

General purpose registers

- B, C, D, E, H, L are 8 bit registers
- Can be used as 16 bit register pairs when combined as BC, DE, HL
- Used to store the intermediate data and result
- H and L can be used to hold data as well as memory address.

Special purpose registers(A, Instruction register, flag register)

Accumulator (A)

- 8 bit register
- All the ALU operation are performed with reference to the contents of Accumulator
- Result of an operation is stored in A
- Store 8 bit data during I/O transfer

Instruction Register:

- When an instruction is fetched from memory, it is loaded in IR. then transferred to the decoder for decoding
- It is not programmable and can not be accessed through any instructio.
- Used by MPU internally.

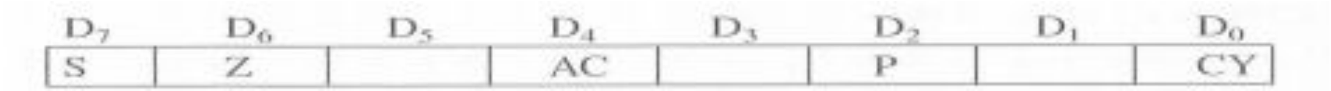
Slide 13

special purpose register - cont...

Flag Register (F)

- ❖ 8-bit register
- ❖ Indicates the status of the ALU operation
- ❖ ALU includes 5 flip flop, which are set or reset after an operation according to data conditions of the result in the accumulator.

The bit position of flag is given as:



D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
S	Z		AC		P		CY

S= sign bit

Z= zero bit

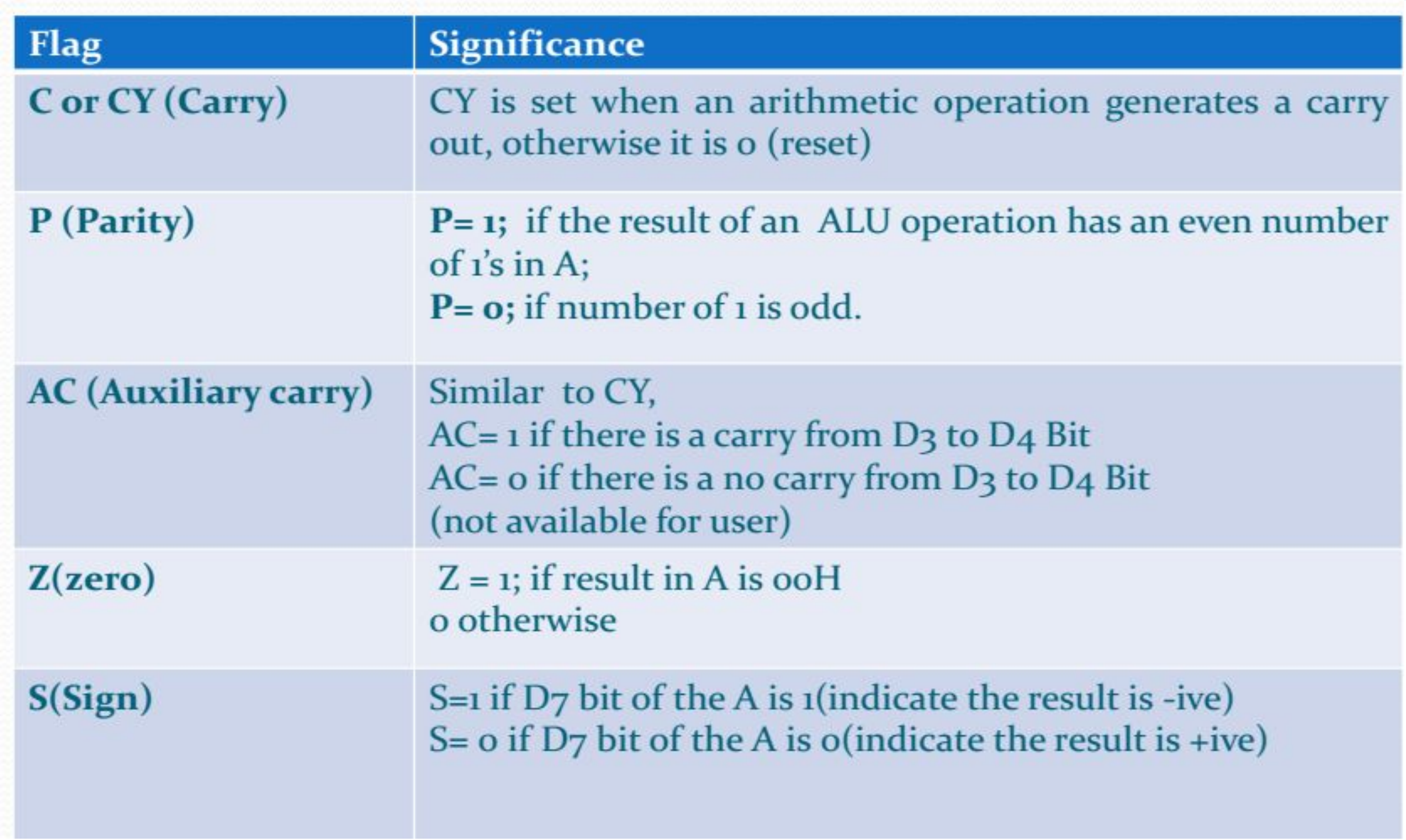
AC = Auxiliary carry

P = Parity

C = Carry

Slide 13

Flag register .. cont..



Flag	Significance
C or CY (Carry)	CY is set when an arithmetic operation generates a carry out, otherwise it is 0 (reset)
P (Parity)	**P= 1;** if the result of an ALU operation has an even number of 1's in A; **P= 0;** if number of 1 is odd.
AC (Auxiliary carry)	Similar to CY, AC= 1 if there is a carry from D3 to D4 Bit AC= 0 if there is a no carry from D3 to D4 Bit (not available for user)
Z(zero)	Z = 1; if result in A is 00H 0 otherwise
S(Sign)	S=1 if D7 bit of the A is 1(indicate the result is -ive) S= 0 if D7 bit of the A is 0(indicate the result is +ive)

Temporary Register

Temporary register**[W, Z, Temporary data register]**

W and Z register

- 8 bit
- Used to hold temporary addresses during the execution of some instructions

Temporary data register:

- 8 bit
- Used to hold temporary data during the ALU operations

Pointer register or special purpose [PC, SP]

Stack Pointer (SP)

- 16 bit register
- Holds the address of the data present at the top of stack memory
- Hold the content of PC when subroutines are used
- When there is a subroutine call or on an interrupt, i.e pushing the return address on a jump, and retrieving it after the operation is complete to come back to its original location

Program Counter (PC)

- 16 bit register.
- Used for execution of program
- Contain the address of the next instruction to be executed after fetching the instruction it is automatically incremented by 1

8085 Microprocessor Architecture.. cont

Instruction decoder and machine cycle encoder, Timing and control circuitry

(a) Instruction decoder and machine cycle encoder :

- Decodes the op-code stored in the Instruction Register (IR) and establishes the sequence of events to follow.
- Encodes it and transfer to the timing & control unit to perform the execution of the instruction.

(b) Timing and control circuitry

- works as the brain of the CPU
- For proper sequence and synchronization of all the operations of MP, this unit generates all the timing and control signals necessary for communication between microprocessor and peripherals.

8085 Microprocessor Architecture.. cont

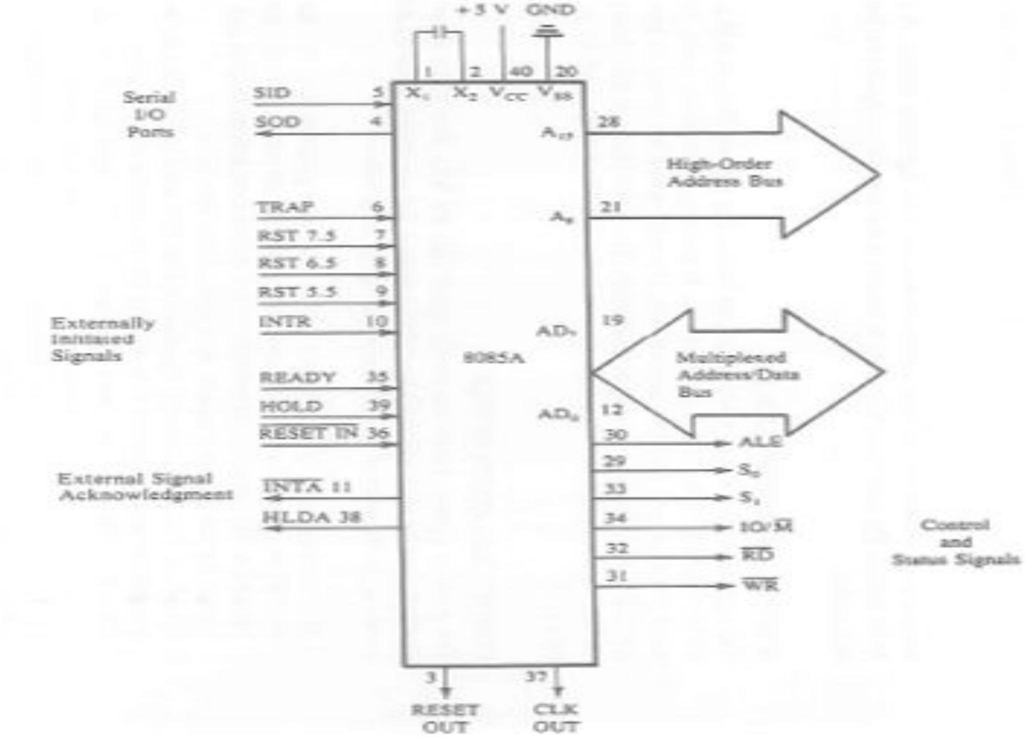
Interrupt control group

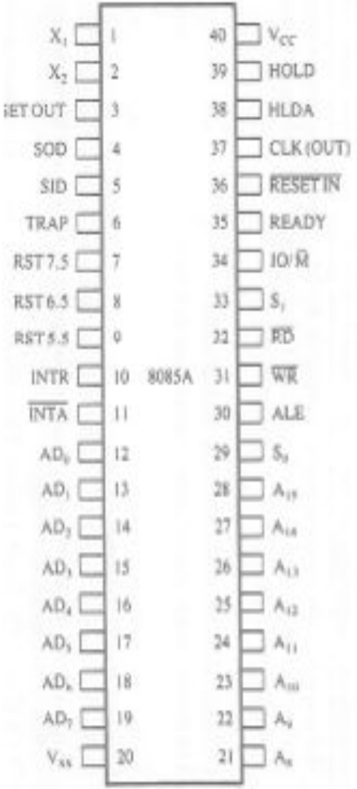
- ❖ Interrupt :- occurrence of an external disturbance
- ❖ After servicing the interrupt, 8085 resumes its normal working sequence
- ❖ Transfer the control to special routines
- ❖ Five interrupts: - TRAP, RST7.5, RST6.5, RST5.5, INTR
- ❖ In response to INTR, it generates INTA signal.

Serial I/O control group

- ❖ Data transfer on AD0-AD7 lines in parallel
- ❖ Under some condition the same will be used for serial transfer
- ❖ Serial data is entered through SID (Serial input data)
- ❖ Serial data is outputted on SOD

8085 pin configuration





8085 Pinout

Address bus and data bus

- 16 address lines divided into two segments as
  - A_{15}-A_8 - unidirectional and carries Most significant bit or higher order bits
  - AD_7-AD_0- bidirectional , carries least significant bit.- meant for data transfer and lower order address transfer
  - To save the number of pins lower order address pins are multiplexed with 8 bit data bus
  - During the execution of the instruction, these lines carry the address bits during the early part (T_1 state) , then during the late parts (T_2 state) of the execution they carry the 8 data bits.
-

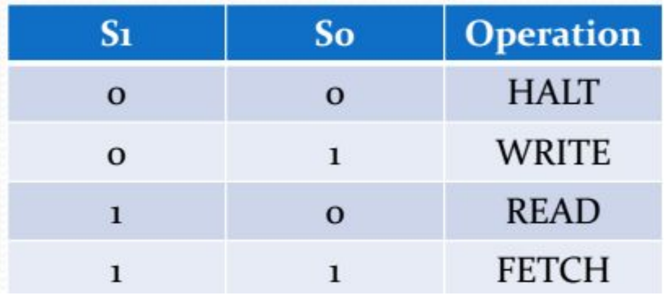
Status signals- ALE, S_1, S_0

ALE (Address Latch Enable) (pin 30)

- Used to demultiplex the address and data bus
- ALE - Address Latch Enable - this generated every time The 8085 begins an operation. It indicates bits on AD_7-AD_0 are address bits.
- If ALE=1 , the lines AD_0-AD_7 have address
- If ALE=0 , the lines AD_0-AD_7 have data

S1 and S0 (status signal)

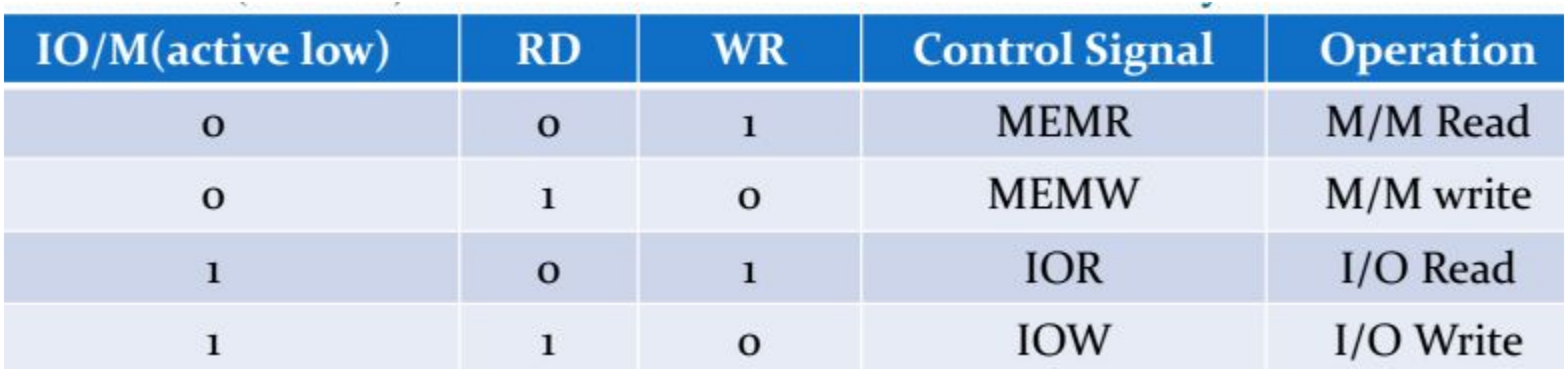
- Status signals to specify the kind of operation being performed.
-



S1	S0	Operation
0	0	HALT
0	1	WRITE
1	0	READ
1	1	FETCH

Control Signals

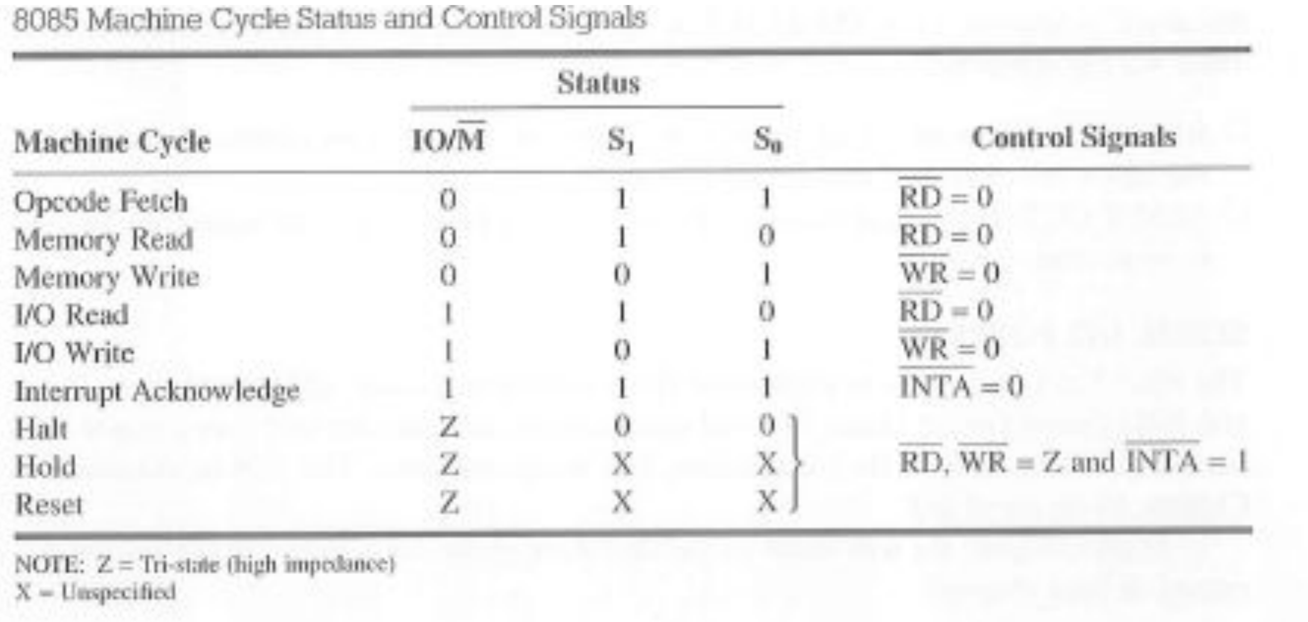
- **$\overline{RD}$- Read - . This is a read signal (active low)**
- Read memory or I/O device
- Indicated that data is to be read either from memory or I/P device and data bus is ready for accepting data from memory or I/O device.
- **$\overline{WR}$- Write (active low)**
- Write memory or I/O device
- Indicated that data on the data bus are to be written into selected memory or I/P device
- **$IO/\overline{M}$: Used to differentiate between I/O and memory operations.**
- Signal specifies that the read/write operation relates to whether memory or I/O device
- When (IO/M=1) the address on the address bus is for I/O device
- When (IO/M=0) the address on the address bus is for memory
-
-



IO/M(active low)	RD	WR	Control Signal	Operation
0	0	1	MEMR	M/M Read
0	1	0	MEMW	M/M write
1	0	1	IOR	I/O Read
1	1	0	IOW	I/O Write

Control and status signals ..cont..

V_{cc} - power supply
V_{SS} - Ground reference
$X_{1,}$ X_2 : A crystal is connected at these two pins.
CLK (OUT)- clock output



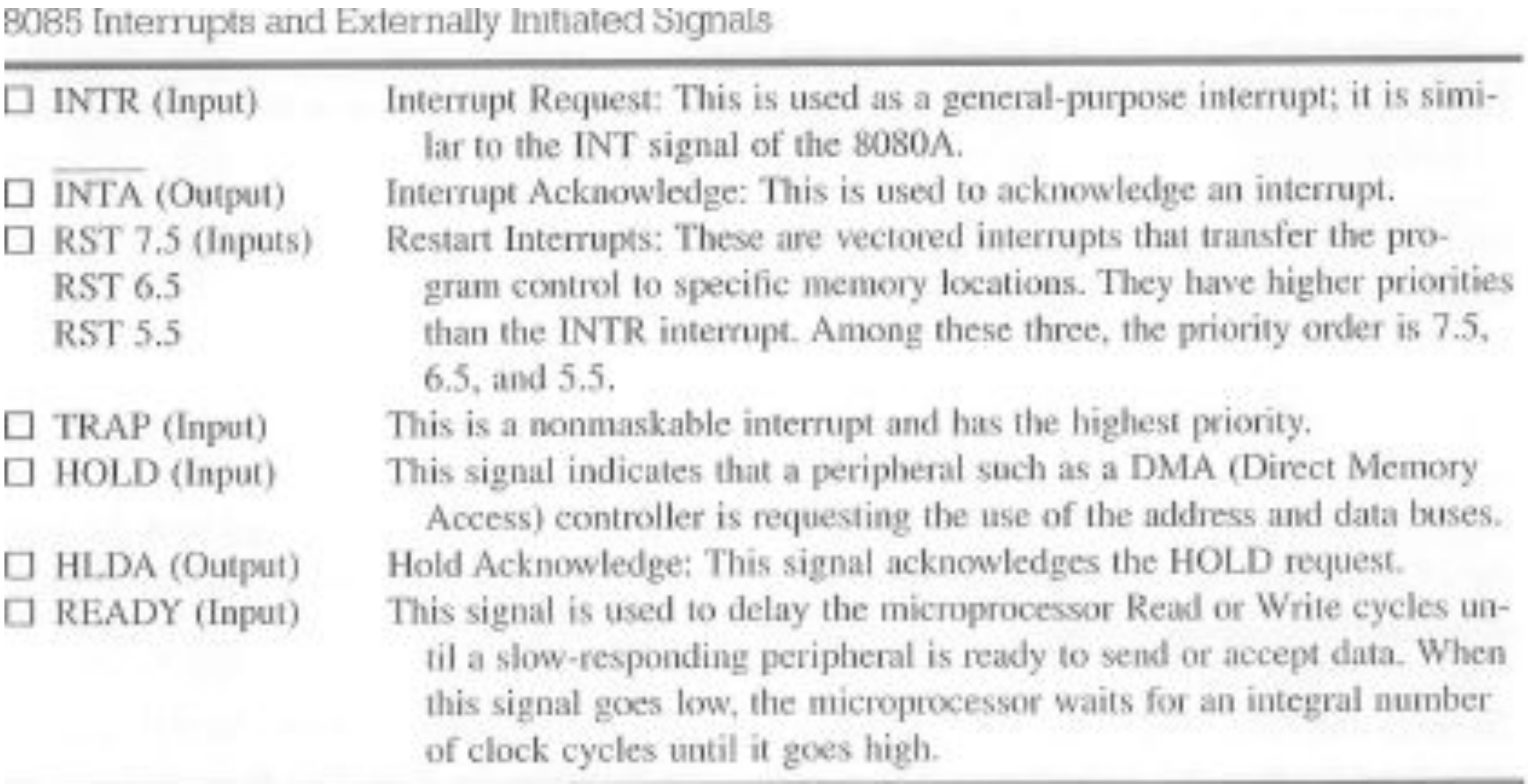
8085 Machine Cycle Status and Control Signals

	Status			
Machine Cycle	$IO/\overline{M}$	S_1	S_0	Control Signals
Opcode Fetch	0	1	1	$\overline{RD} = 0$
Memory Read	0	1	0	$\overline{RD} = 0$
Memory Write	0	0	1	$\overline{WR} = 0$
I/O Read	1	1	0	$\overline{RD} = 0$
I/O Write	1	0	1	$\overline{WR} = 0$
Interrupt Acknowledge	1	1	1	$\overline{INTA} = 0$
Halt	Z	0	0	$\overline{RD}$, $\overline{WR}$ = Z and $\overline{INTA}$ = 1
Hold	Z	X	X	
Reset	Z	X	X	

NOTE: Z = Tri-state (high impedance)
X = Unspecified

Externally initiated signals, including interrupts

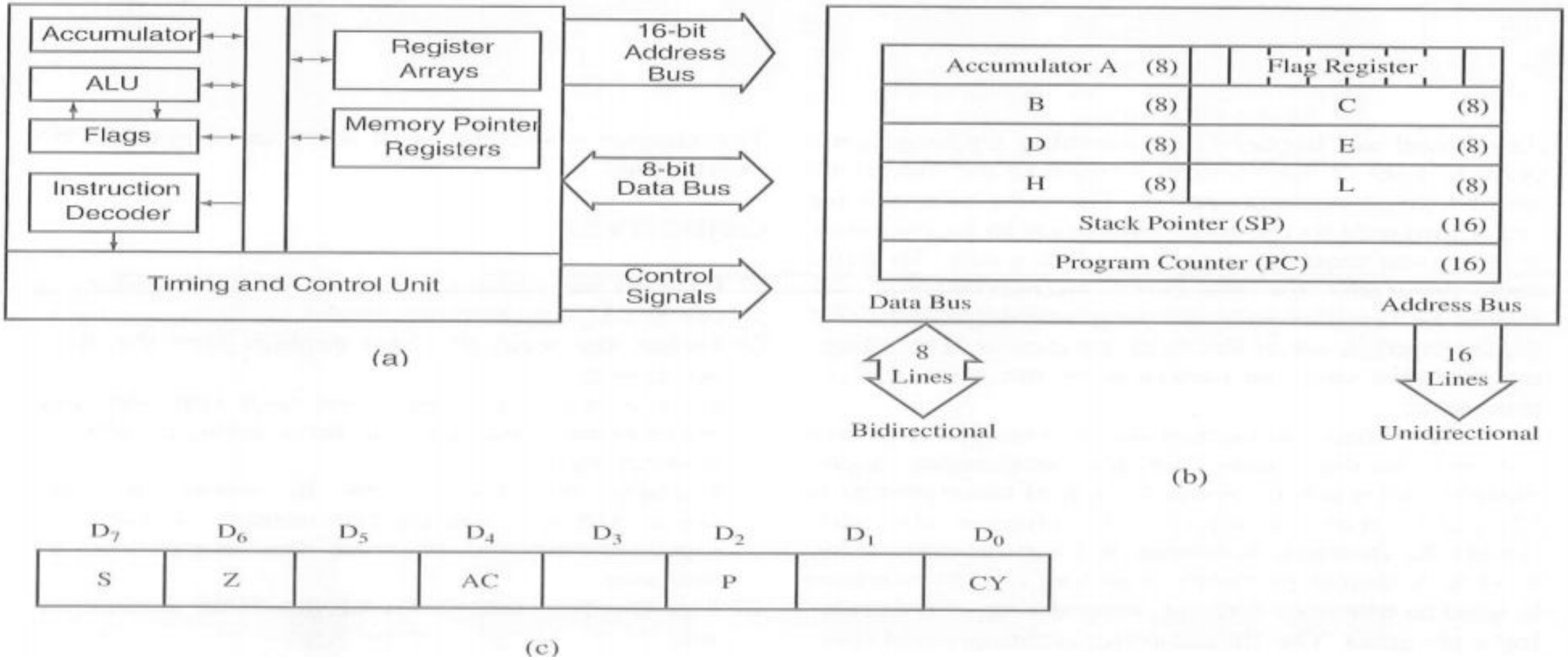
- 8085 has 5 interrupt signals
-



8085 Interrupts and Externally Initiated Signals

- □ INTR (Input) Interrupt Request: This is used as a general-purpose interrupt; it is similar to the INT signal of the 8080A.
- □ $\overline{\text{INTA}}$ (Output) Interrupt Acknowledge: This is used to acknowledge an interrupt.
- □ RST 7.5 (Inputs)
RST 6.5
RST 5.5 Restart Interrupts: These are vectored interrupts that transfer the program control to specific memory locations. They have higher priorities than the INTR interrupt. Among these three, the priority order is 7.5, 6.5, and 5.5.
- □ TRAP (Input) This is a nonmaskable interrupt and has the highest priority.
- □ HOLD (Input) This signal indicates that a peripheral such as a DMA (Direct Memory Access) controller is requesting the use of the address and data buses.
- □ HLDA (Output) Hold Acknowledge: This signal acknowledges the HOLD request.
- □ READY (Input) This signal is used to delay the microprocessor Read or Write cycles until a slow-responding peripheral is ready to send or accept data. When this signal goes low, the microprocessor waits for an integral number of clock cycles until it goes high.

The 8085 programming model



Accumulator A	(8)	Flag Register	
B	(8)	C	(8)
D	(8)	E	(8)
H	(8)	L	(8)
Stack Pointer (SP)			(16)
Program Counter (PC)			(16)

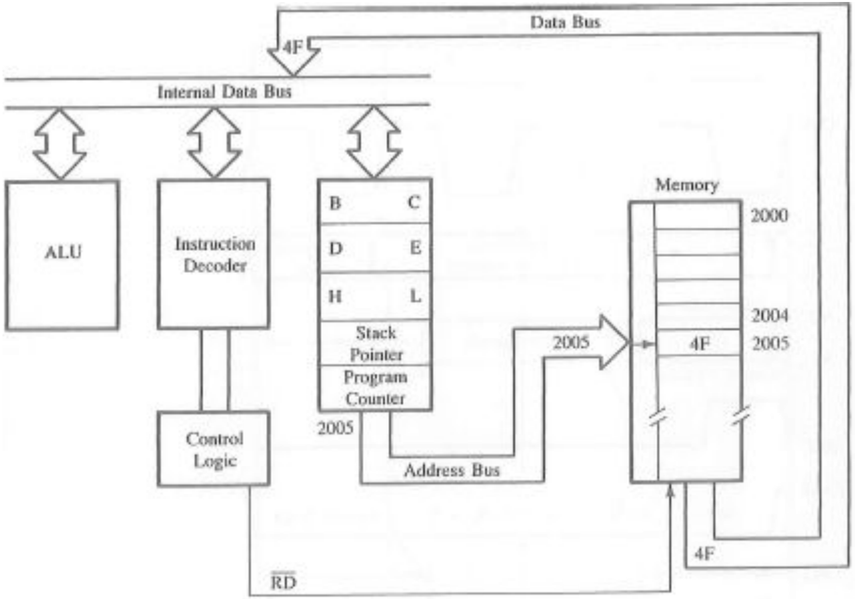
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
S	Z		AC		P		CY

Fig a) 8085 hardware model b)Programing Model c) Flag Register

Microprocessor communication and Bus timings

Illustrate the steps and the timing of data flow when the instruction code **0100 1111 (4FH - MOV C,A)** stored in location 2005H is being fetched.

1. The microprocessor places the 16-bit memory address from the program counter (PC) on the address bus. (T_1)
2. The control signal $\overline{RD}$ to enable the memory chip. (T_2)
3. The byte from the memory location is placed on the data bus.
4. The byte is placed in instruction decoder of MPU, and the task is carried out according to the instruction



TIMING DIAGRAM FOR : (4FH - MOV C,A)- TRANSFER OF BYTE from memory to MPU

step1:

At T_1 the high-order memory address 20H is placed on the address line $A_{15} - A_8$ and the low -order memory address 05H is placed on the bus ALE- goes high. $AD_7 - AD_0$

$IO/\overline{M}$ goes low,- stating this is a memory related operation

Step 2: : T_2

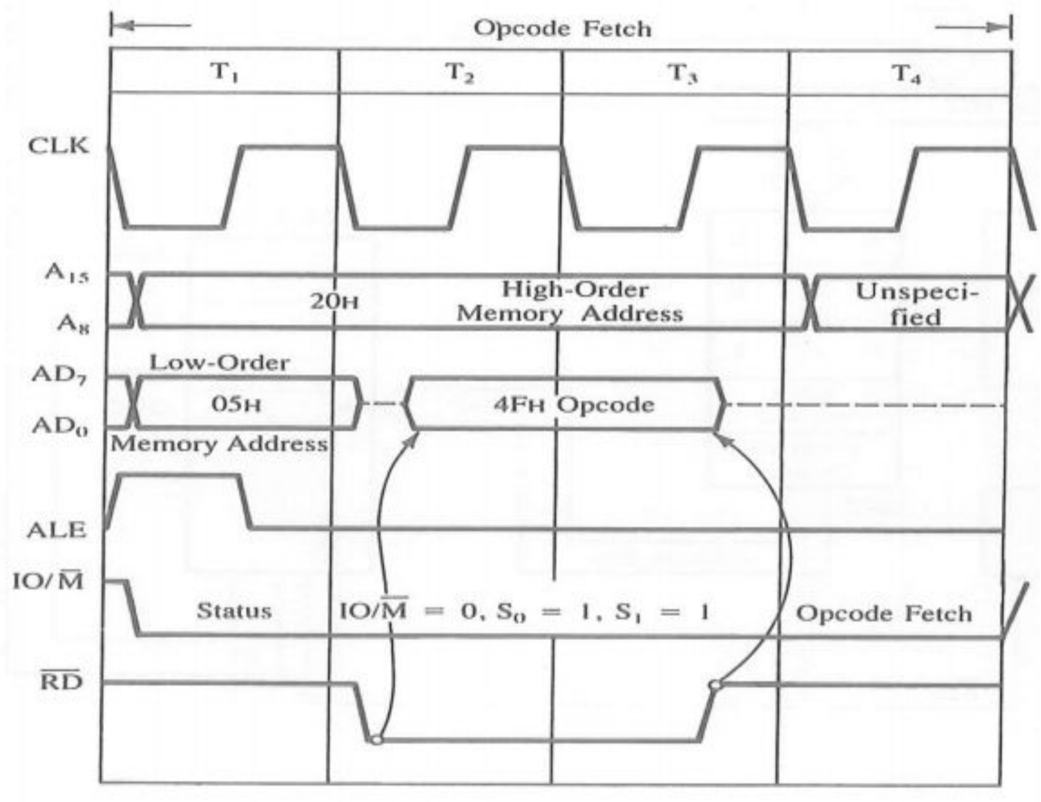
$\overline{RD}$ Is active during two clockperiods

Step 3: T_3

The instruction 4FH is placed on the bus $AD_7 - AD_0$ And transferred to MPU.

Step 4: - T_4

The task mentioned in instruction is carried over



Demultiplexing the bus $AD_7 - AD_0$

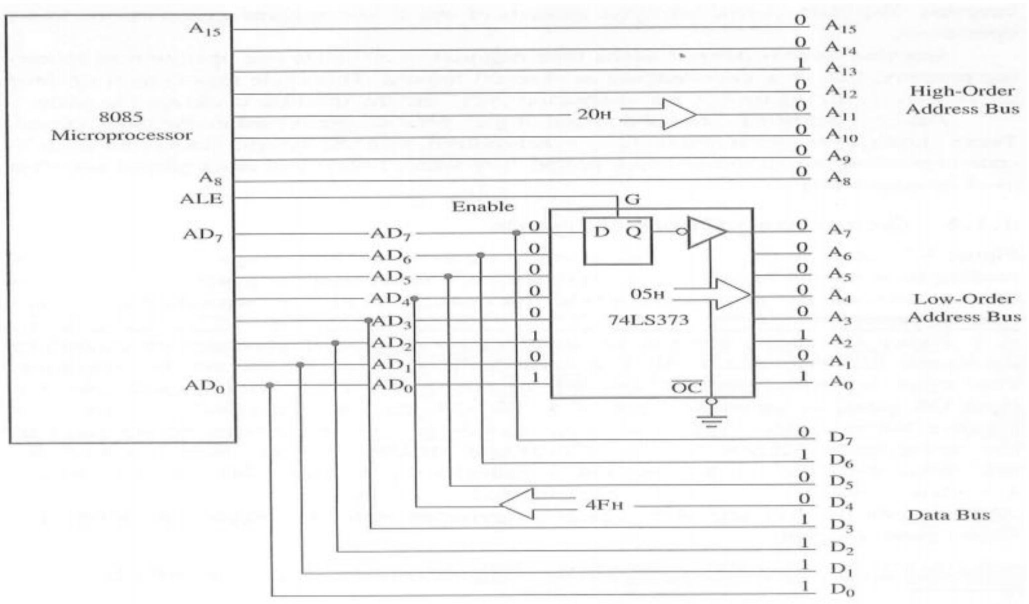
In the previous timing diagram, the address on the high-order bus (20H) remains on the bus for 3 clock periods. But the low-order address(05H) is lost after the first clock period. The address needs to be latched for identifying the memory address. If the bus$AD_7 - AD_0$ is used to identify memory location (2005H) , the address will change to 204FH after the first clock period.

The above situation brings the necessity of demultiplexing of address bus.

For that we go for ALE signal that used to latch a chip 74LS373.

At T1, the ALE goes high and the latch is Transparent: meaning output changes According to input data. So during T1 the Output of the latch is 05H. When the ALE goes low, the data byte 05H is latched Until next ALE, and the output of the latch Represents the low-order address bus $AD_7 - AD_0$ after the latching operation

Slide 31: TIMING DIAGRAM FOR : (4FH - MC



Instruction cycle, machine cycle, T-states

Instruction cycle :- the time required to complete the execution of an instruction. It consists of one to six machine cycle depending on the instruction type.
Machine cycle : the time required to complete one operation of accessing memory, I/O or acknowledging an external request. This may consists of three to six T-states.
T-state :- one subdivision of the operation performed in one clock period.

Generating Control Signals

Combining 3 control signals 4 different control signals can be generated as follows:

$\overline{RD}$ $\overline{WR}$ $IO/\overline{M}$

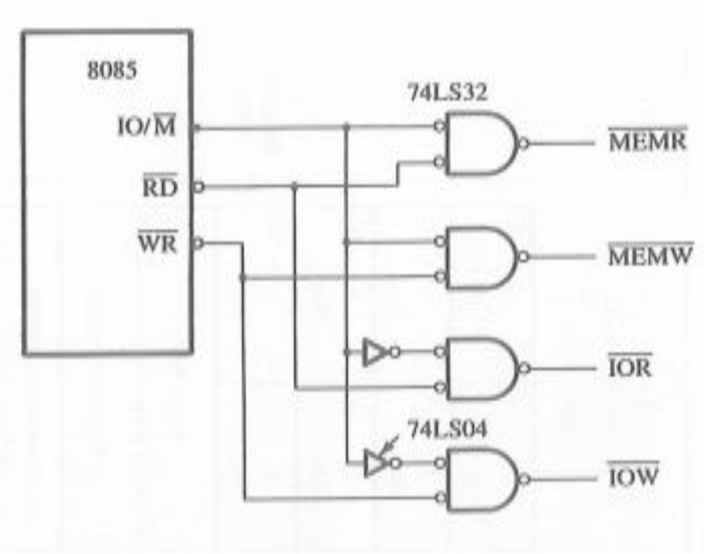
$\overline{MEMR}$ = Reading from memory

$\overline{MEMW}$ = writing into memory

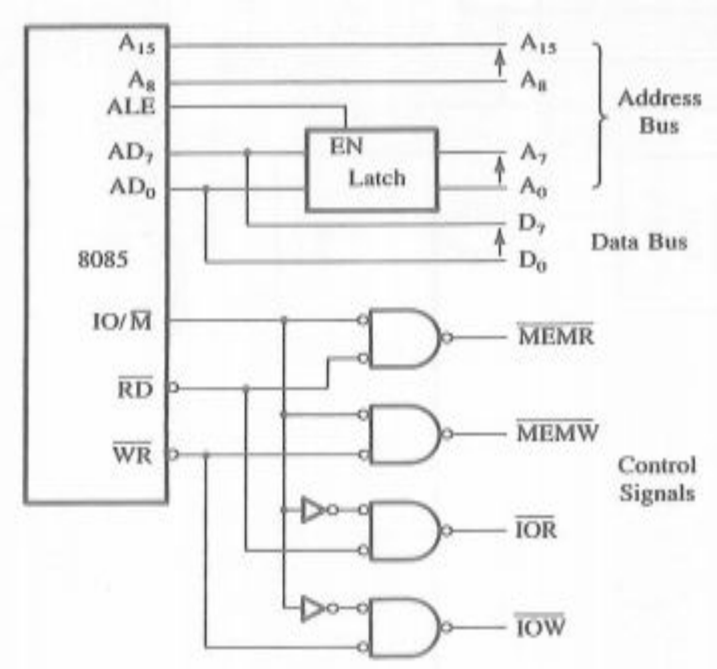
$\overline{IOR}$ = Reading from input port

$\overline{IOW}$ = Writing to an output port

Generating control signals.. cont..



Generate Read/Write control signals for Memory and I/O



8085 demultiplexed address and data bus with control signals

Microprocessor Architecture and its operations

The various operations can be classified in three categories

1. Microprocessor initiated operations
2. Internal operations
3. Peripheral operations

Microprocessor initiated operations and 8085 bus organization

The MPU performs primarily four operations

1. Memory read - reads data (or instructions) from memory
2. Memory write -Writes data (or instructions) into memory
3. I/O Read - Accepts data from input devices
4. I/O Write -Sends data to output devices

MPU in order to communicate with peripheral devices as well as memory, the MPU must do the following:

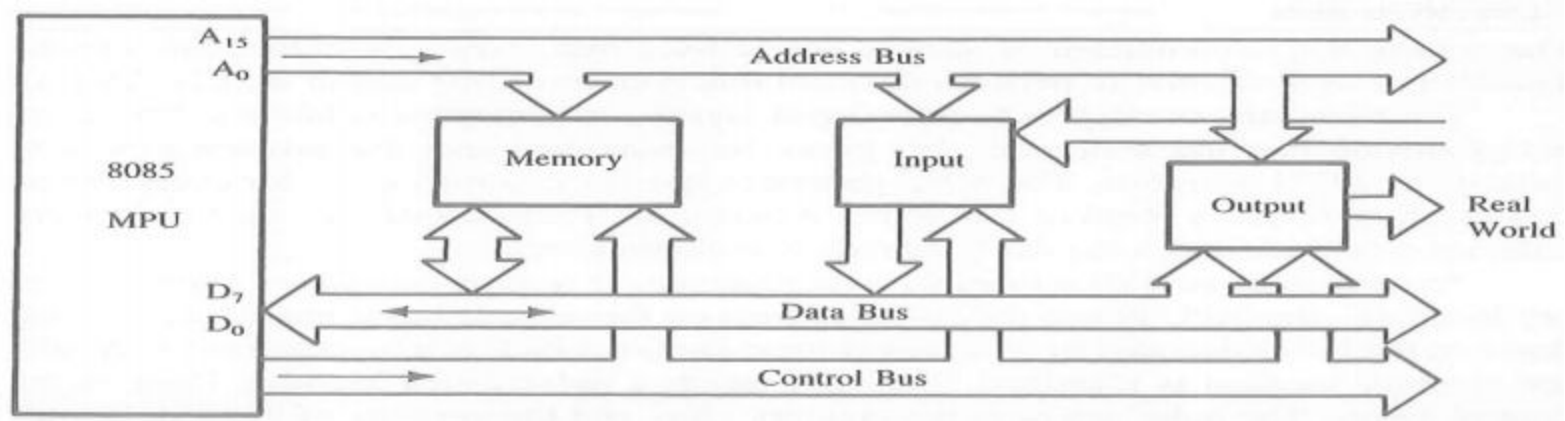
1. Identify the peripheral or the memory location (with its address)
2. Transfer binary information (data and instructions)
3. Provide timing or synchronization signals.

MPU performs these functions using three set of communication lines called buses:
Address bus, data bus and control bus

Address bus

Address bus- group of 16 lines A_0 - A_{15}

- Unidirectional - bits frow in one direction from MPU to peripheral devices, memory
- Perform the first function (identify peripheral or memory location)
- 2^{16}=65,536 memory locations can be identified (64K)



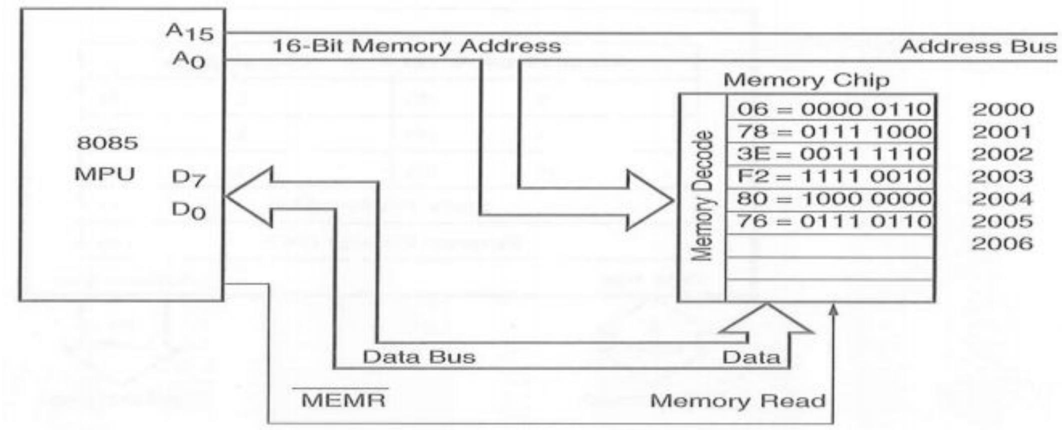
Data Bus

The data bus- group of 8 bit lines

- Bidirectional (data flow in both directions, i.e between MPU and memory, peripheral devices)
- This performs the second function- transferring binary information
- 2^8=256 - 00 to FF,
- Largest number that can appear is 1111 1111 to smallest 0000 0000

Control Bus

- Consists of various single lines that carry synchronization signals
- Performs the third function.
-
-



Internal Data Operations and the 8085 Registers

1. Store 8-bit data
2. Perform arithmetic and logical operations
3. Test for conditions
4. Sequence the execution of instructions
5. Store data temporarily during execution in the defined R/W memory locations called the stack.

Peripheral or Externally Initiated Operations

External devices can initiate the following operations

Reset :- all internal operations are suspended, PC is cleared.

Interrupt :- the microprocessor can be interrupted from the normal execution of instructions and asked to execute some other instructions called a service routine.

Ready :- if the signal at this READY pin is low, the microprocessor enters into a wait state.

Hold :- when the HOLD pin is activated by an external signal, the microprocessor relinquishes control of buses and allows the external peripheral to use them.

Instruction word size

The microprocessor is designed to execute 74 different instruction types

An instruction has two parts:

- Opcode - is a command such as ADD, MOV, AND etc
- Operand - object to be operated on, such as a byte or the contents of a register, memory location

8085 instruction set is classified into 3 categories by considering the length of the instructions.

- One byte 0010 1000 - 28H
- Two bytes 0010 1000 1001 0010 - 28H 92H
- Three bytes - 0010 1000 1001 0010 1011 0110 - 28H 92H B6H

2020	28
;	
2022	28
2023	92
;	
2030	28
2031	92
2032	B6

Instruction word size - one byte

One byte instructions

In 1-byte instruction, the opcode and the operand of an instruction are represented in one byte.

Operands are internal registers and coded into the instruction

One memory location is required to store the Instructions (single byte)

Example -1 - copy the contents of accumulator in register B

Instruction mnemonic - MOV B,A

Opcode - MOV

Operand- B,A

Hexcode - 47H

Binary code - 0100 0111

Example -2 - add the contents of accumulator to the contents of register B

Mnemonic- ADD B

Opcode- ADD

Operand- B

Hex Code- 80H

Binary code- 1000 0000

Example - 3 - invert (complement) each bit in the accumulator

M**nemonic- CMA**

Opcode- CMA

Operand- NA

Hex Code- 2FH

Binary code- 0010 1111

```
2000
2001
2002  47H
;
2010  80H
;
;
2020 2FH
```

Instruction word size - two byte

Two-byte instructions –

Two-byte instruction is the type of instruction in which the first 8 bits indicates the opcode and the next 8 bits indicates the operand.

Instructions require two memory locations to store in the memory

Example-1:

Task- Load the hexadecimal data 32H in the accumulator.

Mnemonic- MVI A, 32H

Opcode- MVI

Operand- A, 32H

Hex Code- 3E 32

Binary code- 0011 1110 0011 0010

Example-2:

Task- Load the hexadecimal data F2H in the register B.

Mnemonic- MVI B, F2H

Opcode- MVI

Operand- B, F2H

Hex Code- 06 F2

Binary code- 0000 0110 1111 0010

2000	
2001	
2002	3E
2003	32
;	
2010	06
2011	F2
;	
;	
2020	2FH

Instruction word size - three bytes

Three-byte instructions –

Three-byte instruction is the type of instruction in which the first 8 bits indicates the opcode and the next two bytes specify the 16-bit address. The low-order address is represented in second byte and the high-order address is represented in the third byte.

Instruction require three memory locations to store the single byte in the memory

Example-1:

Task- Load contents of memory 2010H in the accumulator.

```
Mnemonic- LDA 2010H
Opcode- LDA
Operand- 2010H
Hex Code- 3A
          10
          20
Binary code- 0011 1010
             0001 0000
             0010 0000
```

2000	
2001	
2002	3A
2003	10
2004	20
;	
2010	80H
;	
;	
2020	2FH

Instruction word size - three bytes ..cont..

Example-2:

Task- Transfer the program sequence to the memory location 2050H.

Mnemonic- JMP 2085H

Opcode- JMP

Operand- 2085H

Hex Code- C3 85 20

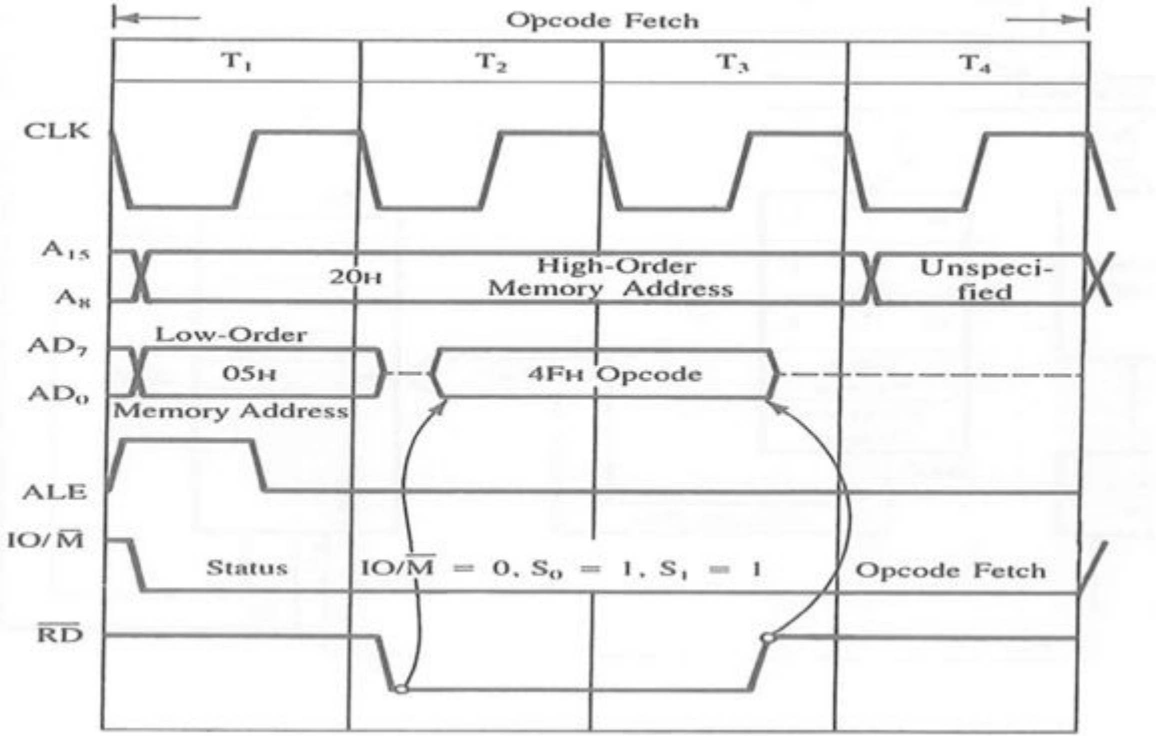
Binary code- 1100 0011 1000 0101 0010 0000

```
2000
2001
2002  C3
2003  85
2004   20
;
2010  80H
;
;
2020 2FH
```

Opcode fetch machine cycle

- Steps and timing involved in fetching instruction (1 byte) 0100 1111 (4F$_H$ – MOV C,A) stored in location 2005$_H$
-

2005   4F

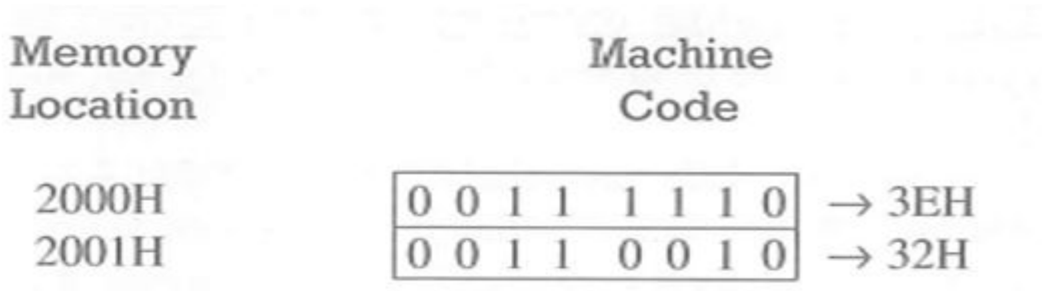


2 byte- Memory read Machine cycle

- In single byte we find the instruction to be opcode fetch as well as memory read.
- You can differentiate as opcode as well as memory read in 2 byte and 3 byte instructions only.

- Consider a 2 byte instruction (MVI A, 32_H). This will be stored in memory as follows

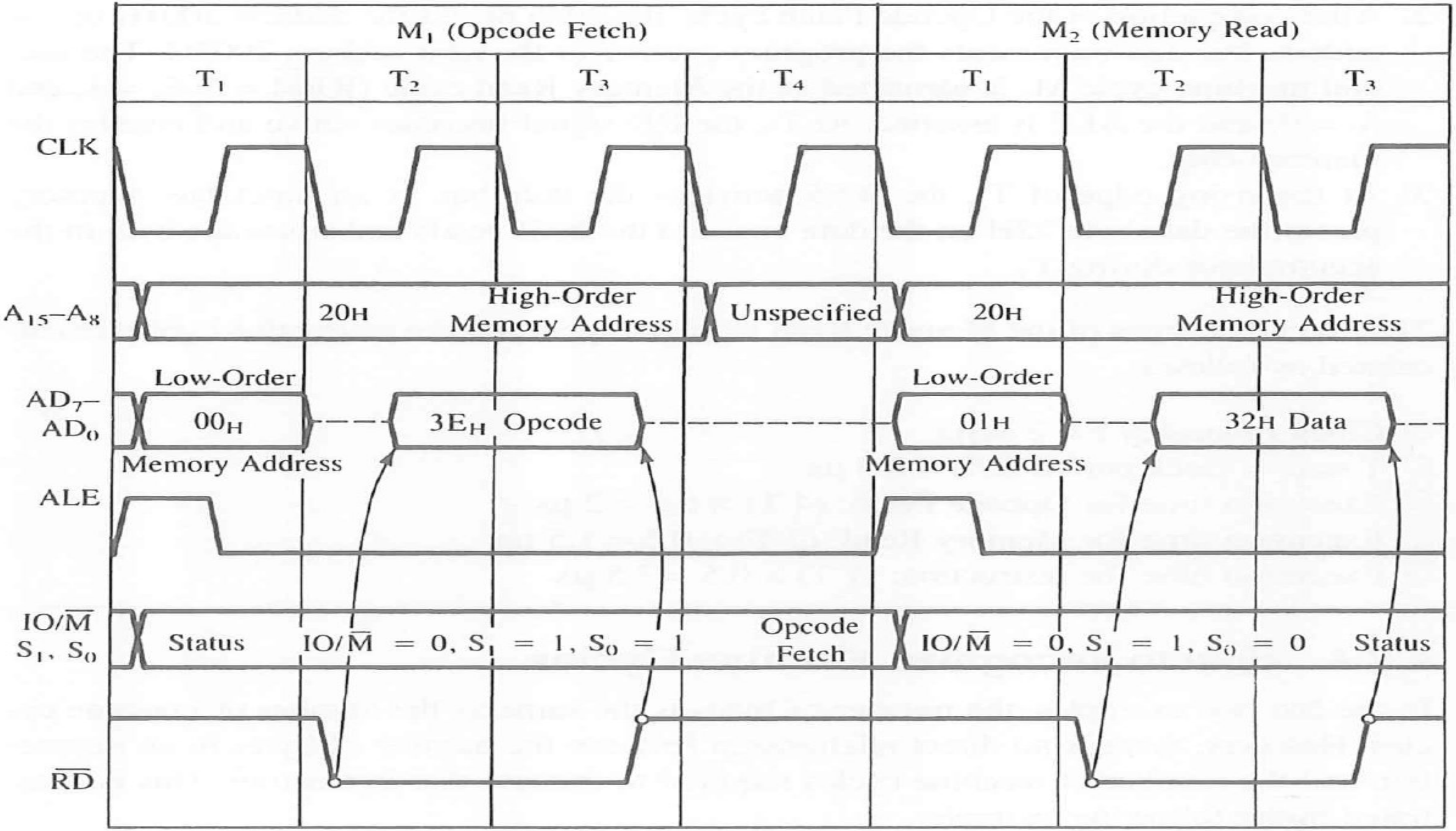
(MVI A, 32H) equal hexcode is = 0011 1110 (3EH) and 0011 0010 (32_H)



Memory Location	Machine Code	
2000H	0 0 1 1 1 1 1 0	→ 3EH
2001H	0 0 1 1 0 0 1 0	→ 32H

The instruction consists of two bytes: the opcode first (3E- MOV) and operand second (A, 32H).
So 8085 first fetch opcode (memory read), (Machine cycle 1- 4T state), secondly again to read operand from memory (memory read - machine cycle 2 - 3 T state) so totaly 7T state

2- byte Memory read Machine cycle cont...



Memory read Machine cycle cont...

The execution time of the Memory read machine cycle and the instruction cycle are calculated as follows

Clock frequency f= 2 MHz

T-state = clock period (1/f) = 0.5μs
Execution time for Opcode fetch : (4T) X 0.5 = 2 μs

Execution time for Memory Read (3) X 0.5 = 1.5 μs

Execution time for Instruction : (7 T) X 0.5 = 3.5 μs

3 -byte instruction Machine cycle

opcode	operand	bytes	Machine cycles	T-states	operation
STA	8000H	3	4	13	stores(writes) the content of accumulator in memory location 8000H

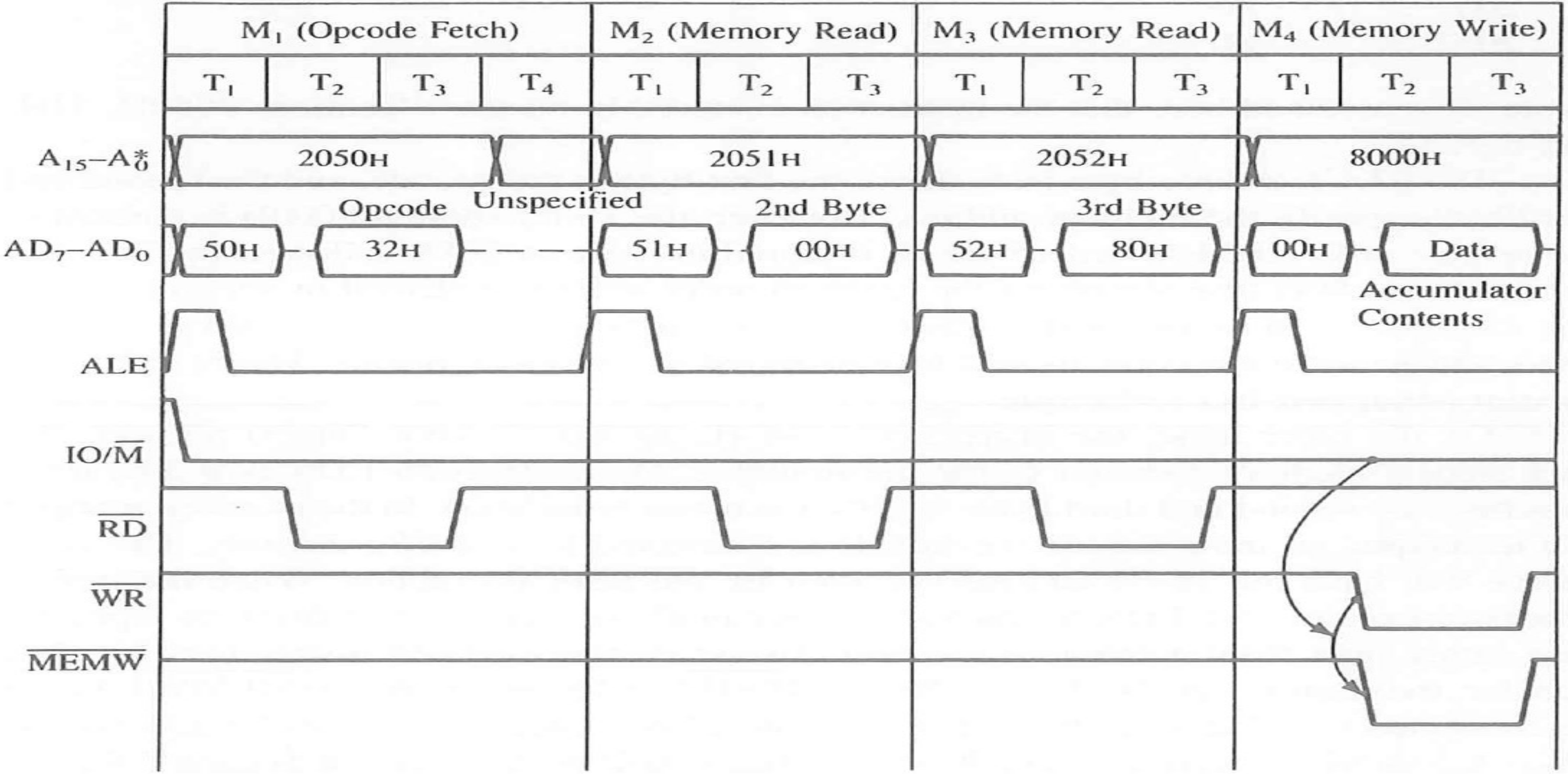
Memory address	Machine code	
2050 2051 2052	0011 0010 -> 32H 0000 0000 -> 00H 1000 0000-> 80H	Opcode Lower order address Higher order address

This is a 3 byte instruction, but needs four machine cycle.

1. The first operation is execution of an instruction is to fetch opcode.
2. The second and third are memory read to fetch operand
3. In the above instruction it stores (write) the content of accumulator in memory location, so it is memory write (thus 4th machine cycle)
4.

3 -byte instruction Machine cycle ..cont..

Timing diagram for instruction STA 8000H



Memory

The R/W memory is made of registers.

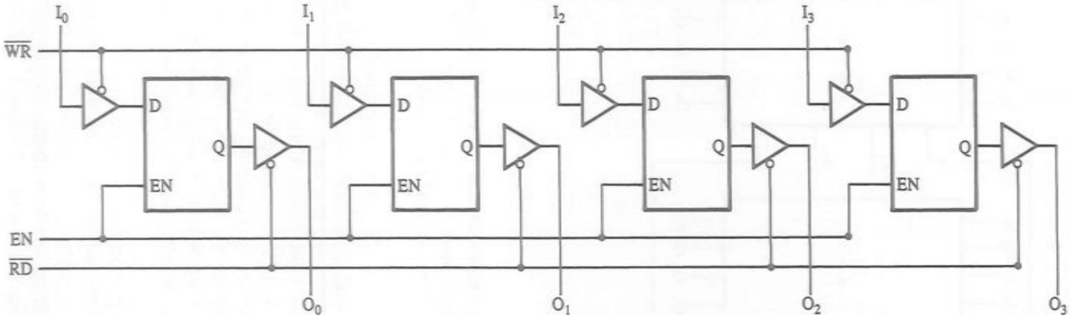
Registers are made of flip-flops where one flip-flop can store 1 bit of information.

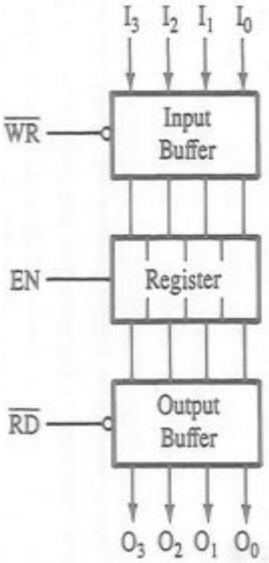
This is placed in memory chip.

To communicate with memory, the MPU should be able to

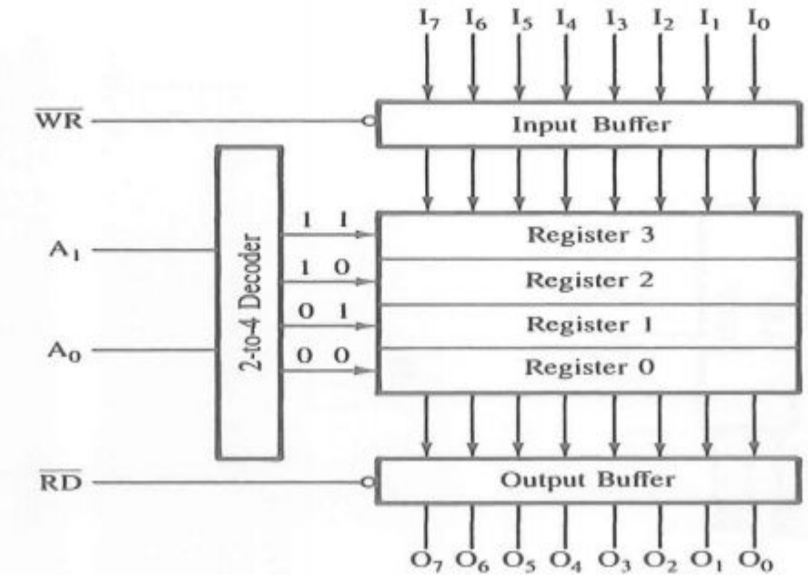
- Select the chip
- Identify the register
- Read from or write into the register

Consider an example of 4-bit register





4 x 8 bit register



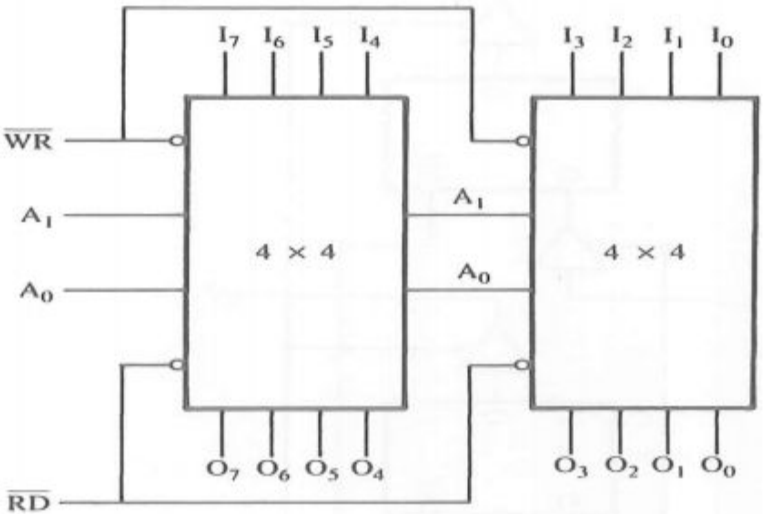
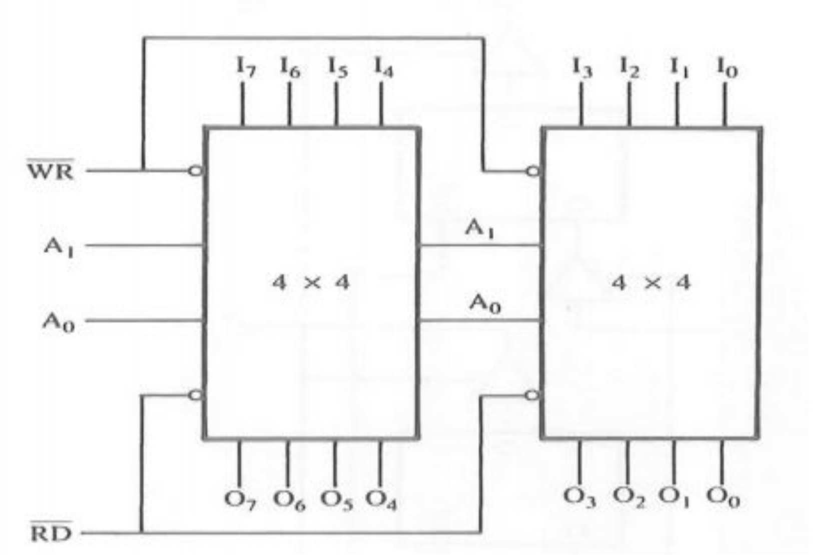
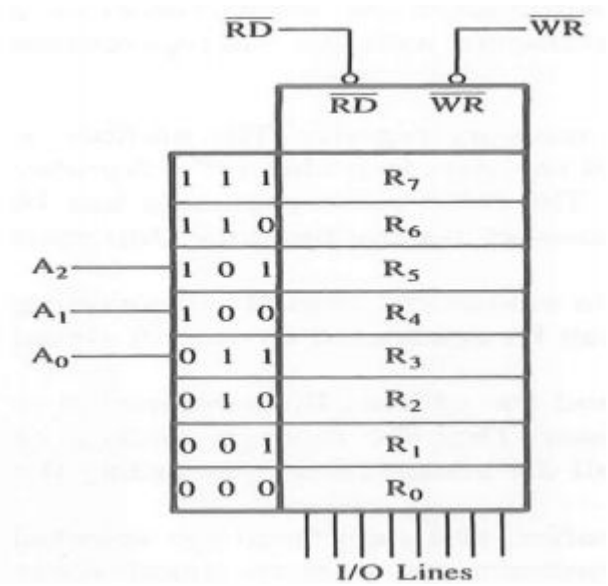


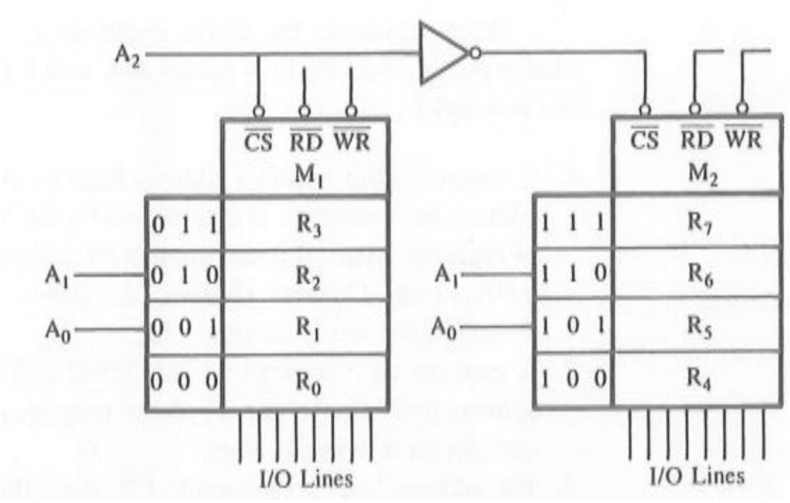
figure 1 - 4, 8 bit registers
Figure 2- 2 chips, each 4 register with 4 bits each
One can expand memory by adding more chips like this.

How to deal with memory more than one chip?



In the above figure 4+4= 8 registers are there. But this can address only one chip. Since 8 registers are there we need 3 address lines to access individual registers. In the above only 2 address lines are there. So a third address line can be used to select the chip. But how?





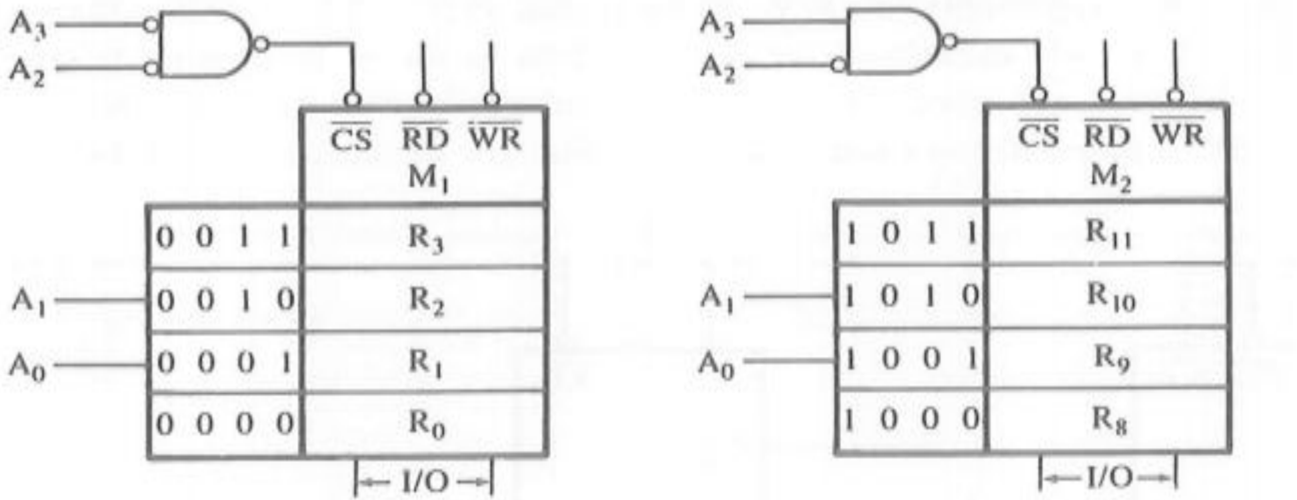
A2
$\overline{CS}$ $\overline{RD}$ $\overline{WR}$
M1
0 1 1
0 1 0
0 0 1
0 0 0
R3
R2
R1
R0
A1
A0
I/O Lines
$\overline{CS}$ $\overline{RD}$ $\overline{WR}$
M2
1 1 1
1 1 0
1 0 1
1 0 0
R7
R6
R5
R4
A1
A0
I/O Lines

In the second figure , we have two chips. A_0 and A_1 address lines are used to select individual registers in each chip. But in which chip? This is resolved using third address line (as there are 8 register we need minimum 3 address line 2^3=8) . this A_2 helps in selecting between chips. This is connected to chip select signal (CS=low) When A_2=0 chip M1 is selected. When A_2=1 chip M2 is selected.
By combining $A_2A_1A_0$ the memory range from 000 to 111

Addressing eight registers with four address lines

4 address lines are capable of addressing 16 registers. But we have only 8 registers. So the extra line also combined to select chip. Memory chip M1 is selected when A_3 and A_2 are both 0. (address range from 0000 to 0011)

Memory chip m2 is selected when A_3=1 and A_2=0. (address range from 1000 to 1011) this is called absolute or complete decoding



A_3
A_2
$\overline{CS}$ $\overline{RD}$ $\overline{WR}$
M_1
A_1
A_0
I/O
A_3
A_2
$\overline{CS}$ $\overline{RD}$ $\overline{WR}$
M_2
A_1
A_0
I/O

	M_1
0 0 1 1	R_3
0 0 1 0	R_2
0 0 0 1	R_1
0 0 0 0	R_0

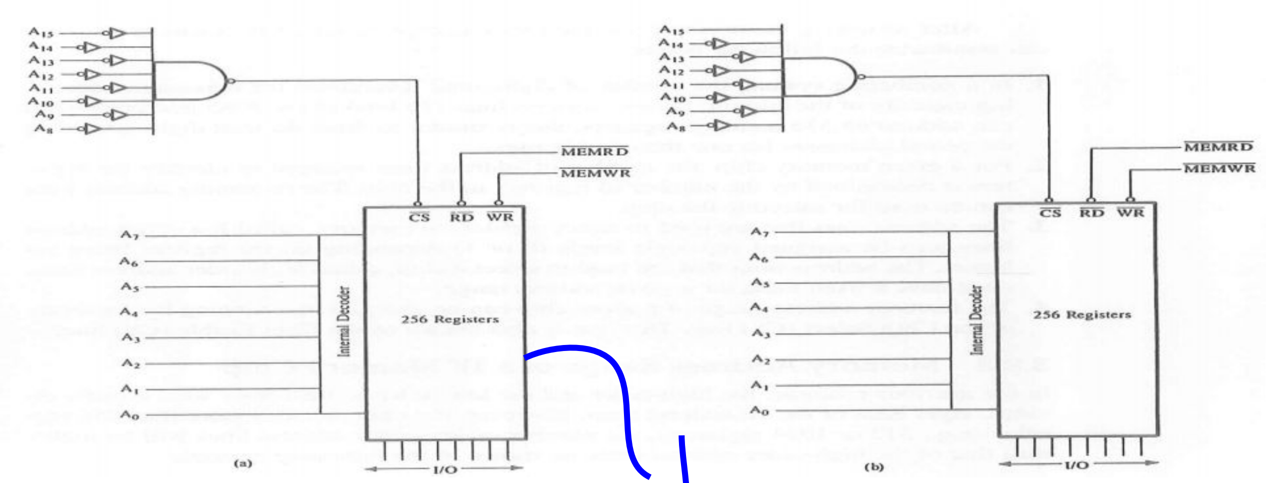
	M_2
1 0 1 1	R_{11}
1 0 1 0	R_{10}
1 0 0 1	R_9
1 0 0 0	R_8

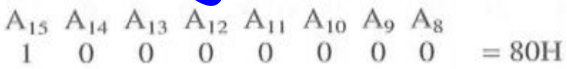
Memory map and addresses

8085 is a

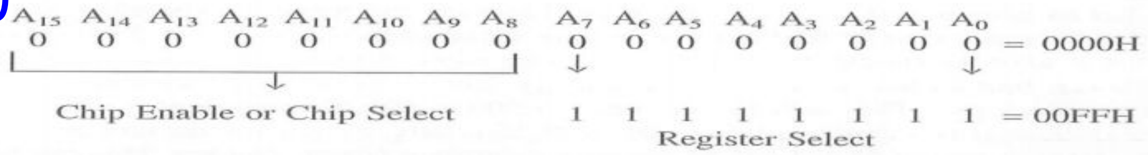
- 8 bit microprocessor (data)
- 16 bit address lines (2^{16}=65,536) memory registers
- The address range is from 0000 to FFFF

Example :- illustrate the memory address range of the chip with 256 bytes of memory. The size of memory can be expressed as 256 x 8





A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	
1	0	0	0	0	0	0	0	= 80H



A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	= 0000H
Chip Enable or Chip Select								1	1	1	1	1	1	1	1	= 00FFH
								Register Select								

The memory address for first figure range

From 0000H to 00FFH

For the second figure it ranges from 8000H t

80FFH

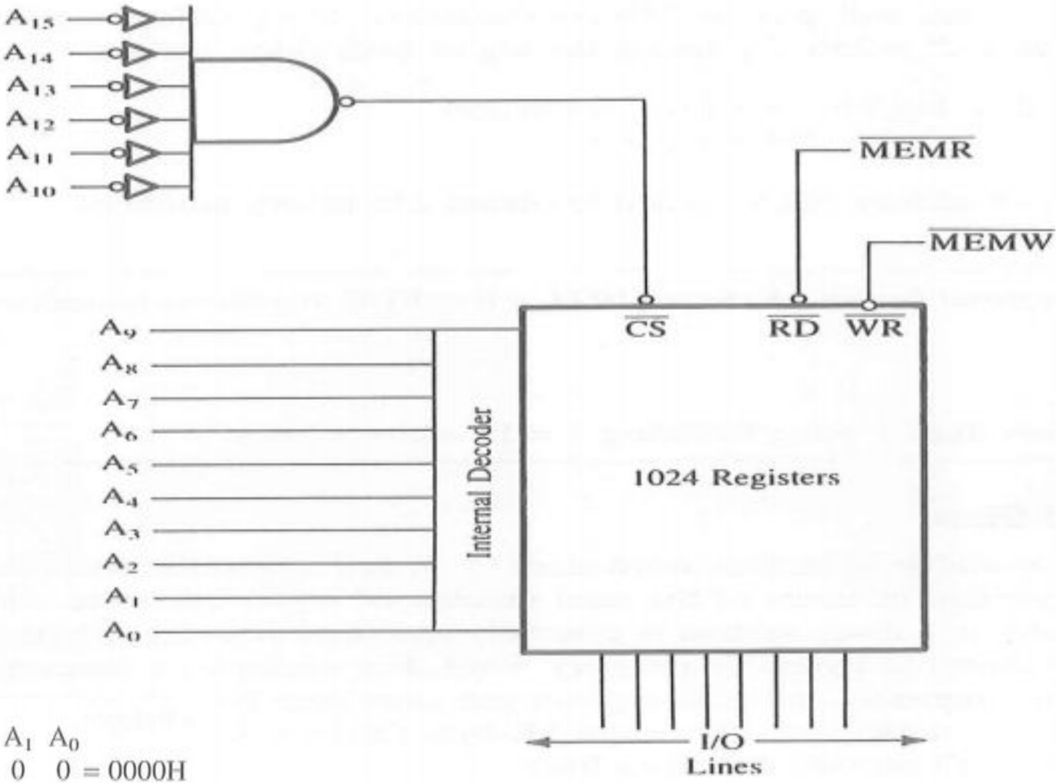
Memory map and addresses cont..

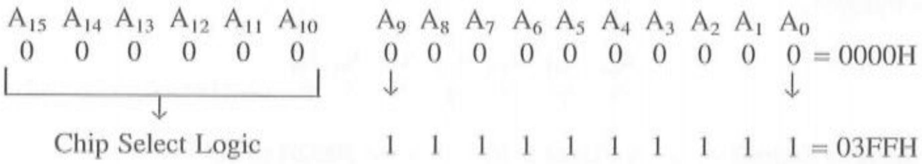
Example : explain the memory address range of 1K (1024 x 8) memory.

The memory chip has 1024 registers.

Therefore 10 address lines are (A_9 - A_0) are required to identify registers.

The remaining address lines(A_{15} - A_{10}) are used for chip select





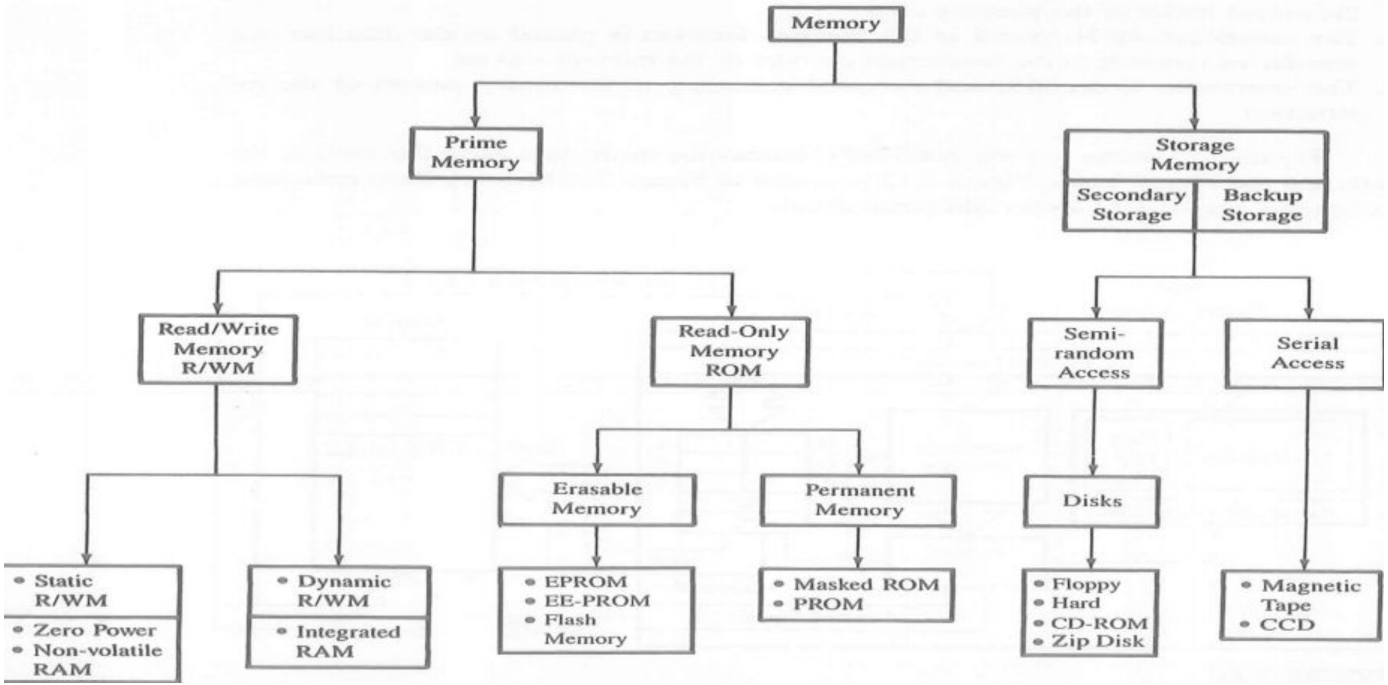
A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	= 0000H
↓						↓									↓	
Chip Select Logic						1	1	1	1	1	1	1	1	1	1	= 03FFH

Memory Classification

Memory classified into two:

Primary :- RAM, ROM

Secondary :- magnetic disk, tapes etc



RAM- Random access memory- volatile memory.

SRAM- static random access memory- made of flip-flops and store a bit as voltage. each memory cell requires six transistors. The memory chip has low density , but high speed.

DRAM - Dynamic RAM- made up of MOS transistor gates and store bit as charge. Advantage is high density, low power consumption and cheaper than SRAM. The disadvantage is chare leaks. Therefore , stored information needs to be read and written again every few milliseconds called refreshing of memory which adds additional cost.

ROM (Read only memory) :- non-volatile memory. This memory is used for programs and data that need not be altered. Five types of ROM are there- masked ROM, PROM, EPROM, EEPROM, and Flash Memory.

Mask ROM is a read-only memory whose contents are programmed by the integrated circuit manufacturer (rather than by the user). ... It is common practice to use rewritable non-volatile memory – such as UV-EPROM or EEPROM – for the development phase of a pro

Instruction Classification - based on functions

Instruction - is a binary pattern designed inside a microprocessor to perform a specific functions.
Instruction set - the entire group of instructions of a particular microprocessor.

8085 instruction set is broadly classified into five categories based on its functions

1. Data transfer operations
2. Arithmetic operations
3. Logical operations
4. Branching operations
5. machine -control operations

1. **Data transfer operations**

   These instructions copies data from one location called source to another location called destination. The locations are memory , registers, I/O devices.

   Types:

   Between registers, specific data byte to a register or memory location, between memory location and register, between I/O device and accumulator

Instruction classification- based on functions.. cont

2. Arithmetic operations

Addition, subtraction, increment, decrement

3. Logical operations

- ❖ AND , OR, Exclusive-OR
- ❖ Rotate - shift content of accumulate to left or right
- ❖ Compare - compare 8-bit data, or the contents of a register, or a memory location with content of accumulator
- ❖ Complement - complement the content of accumulator

4. Branching operations - alters the sequence of program execution either conditionally or unconditionally.

Jump - based on flag bits

Call, return, and restart

5. Machine control operations

Halt, Interrupt, or do nothing- instructions that control machine functions

Instruction classification- based on word size

An instruction has two parts

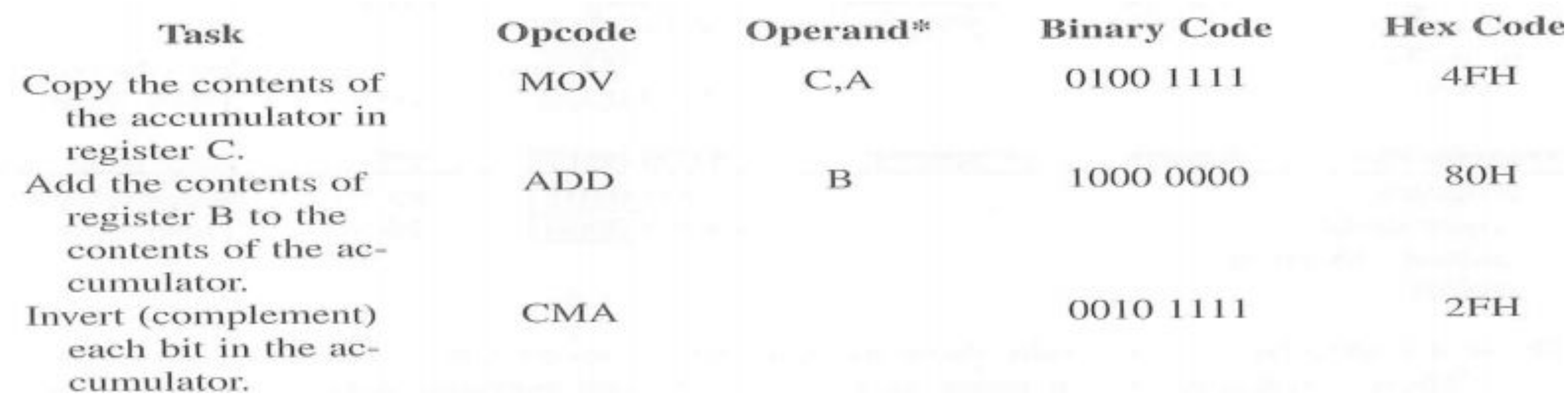
- **Operation code (op-code)** :- task to be performed
- **Operand** :- data to be operated.

Instruction word size- classified based on word size or byte size

1. 1-byte instructions
2. 2-byte instructions
3. 3-byte instructions

ONE-BYTE instructions

A 1-byte instruction includes the opcode and the operand in the same byte. Example

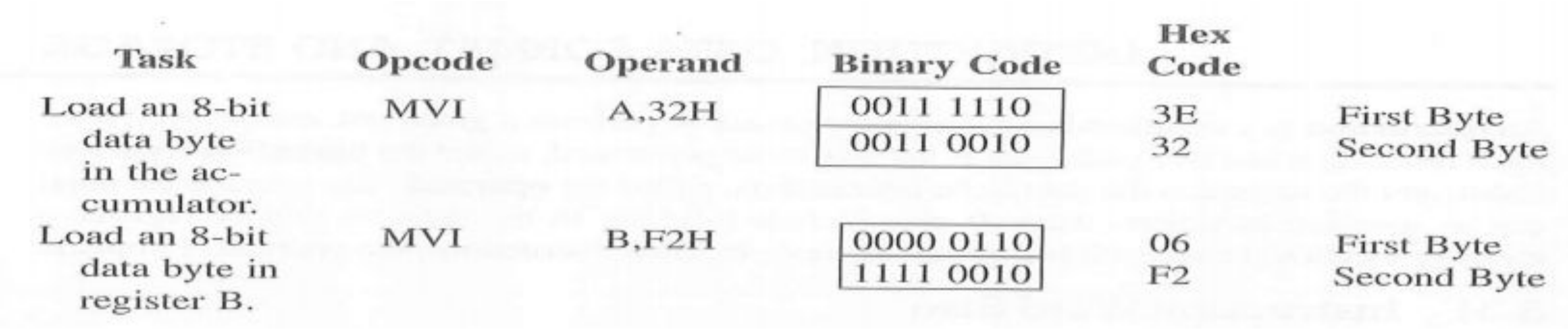


Task	Opcode	Operand*	Binary Code	Hex Code
Copy the contents of the accumulator in register C.	MOV	C,A	0100 1111	4FH
Add the contents of register B to the contents of the accumulator.	ADD	B	1000 0000	80H
Invert (complement) each bit in the accumulator.	CMA		0010 1111	2FH

Instruction classification- based on word size.. cont

Two-Byte Instructions

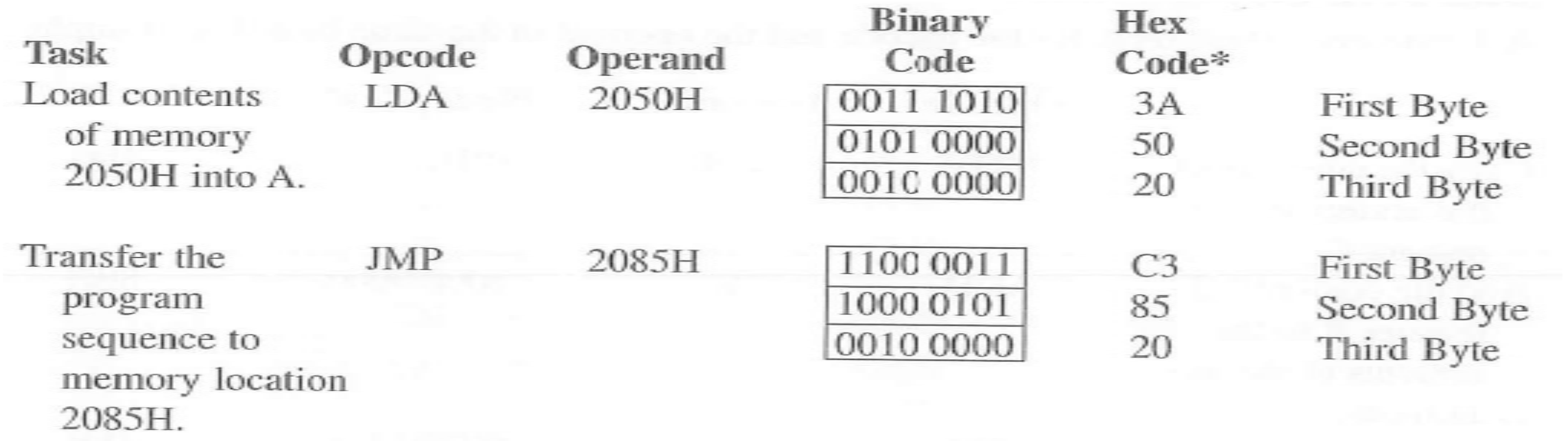
In this the first byte specifies the operation code and the second byte specifies the operand. Example



Task	Opcode	Operand	Binary Code	Hex Code	
Load an 8-bit data byte in the accumulator.	MVI	A,32H	0011 1110 0011 0010	3E 32	First Byte Second Byte
Load an 8-bit data byte in register B.	MVI	B,F2H	0000 0110 1111 0010	06 F2	First Byte Second Byte

These instructions require two memory locations each to store the binary codes,

Instruction classification- based on word size.. cont



Task	Opcode	Operand	Binary Code	Hex Code*	
Load contents of memory 2050H into A.	LDA	2050H	0011 1010	3A	First Byte
			0101 0000	50	Second Byte
			0010 0000	20	Third Byte
Transfer the program sequence to memory location 2085H.	JMP	2085H	1100 0011	C3	First Byte
			1000 0101	85	Second Byte
			0010 0000	20	Third Byte