Microprocessor 8085- interfacing- unit 5

Memory

Memory is the place where we store the data and retrieve data. The performance of a computer system depends upon the size of primary memory (RAM). memory is of type

1. Primary memory /Volatile memory (RAM) / main memory
2. Secondary memory / Non-volatile memory

Primary memory : this memory is internal memory of the computer. RAM and ROM both form part of primary memory.

Random access memory (RAM) : this is otherwise called as primary memory. The storage in this is temporary as it disappears from RAM as soon as the power to the computer is switched off. So they are called volatile memory also.

Read Only memory-ROM:

What is an interface?

Interfacing is a technique to be used for connecting the microprocessor to external device, mainly memory as well as I/O devices.

In general , RAM or ROM is used for memory interfacing

Memory- is a digital IC which stores the data in binary form

Memory Interface

Memory is made up of semiconductor material used to store the programs and data. The memory is broadly divided into

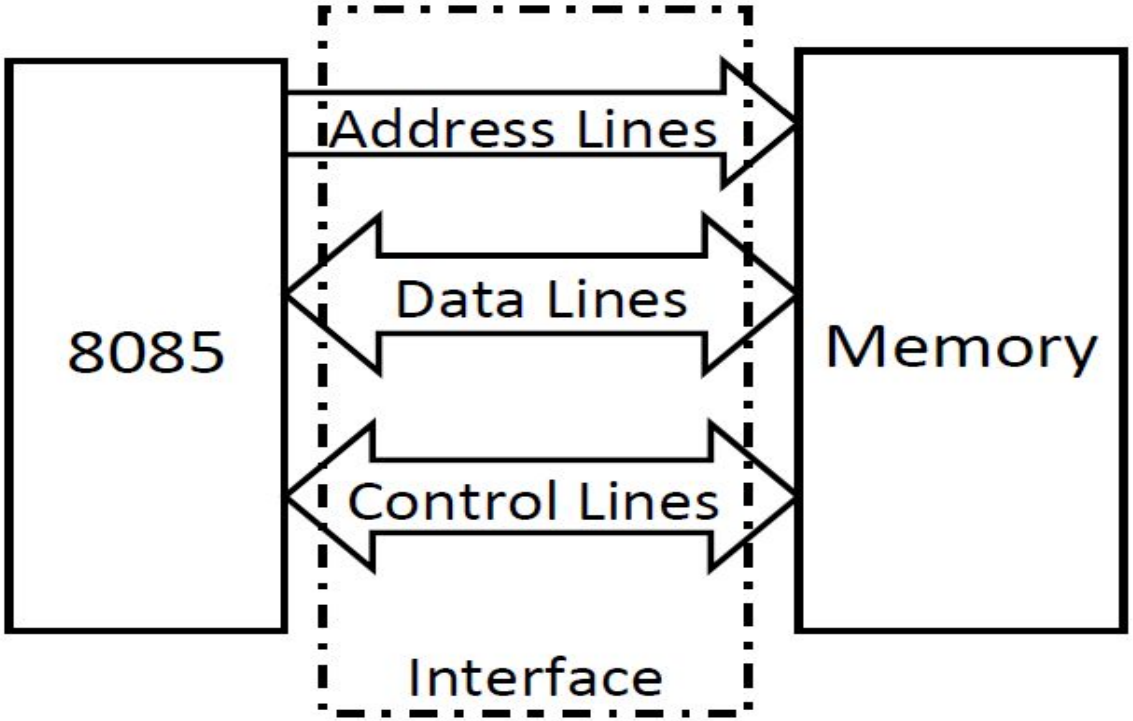
primary or main memory

- RAM and ROM are examples
- Microprocessor uses it in storing a program temporarily and executing the program
- The speed of this type of memory is fast
- Volatile

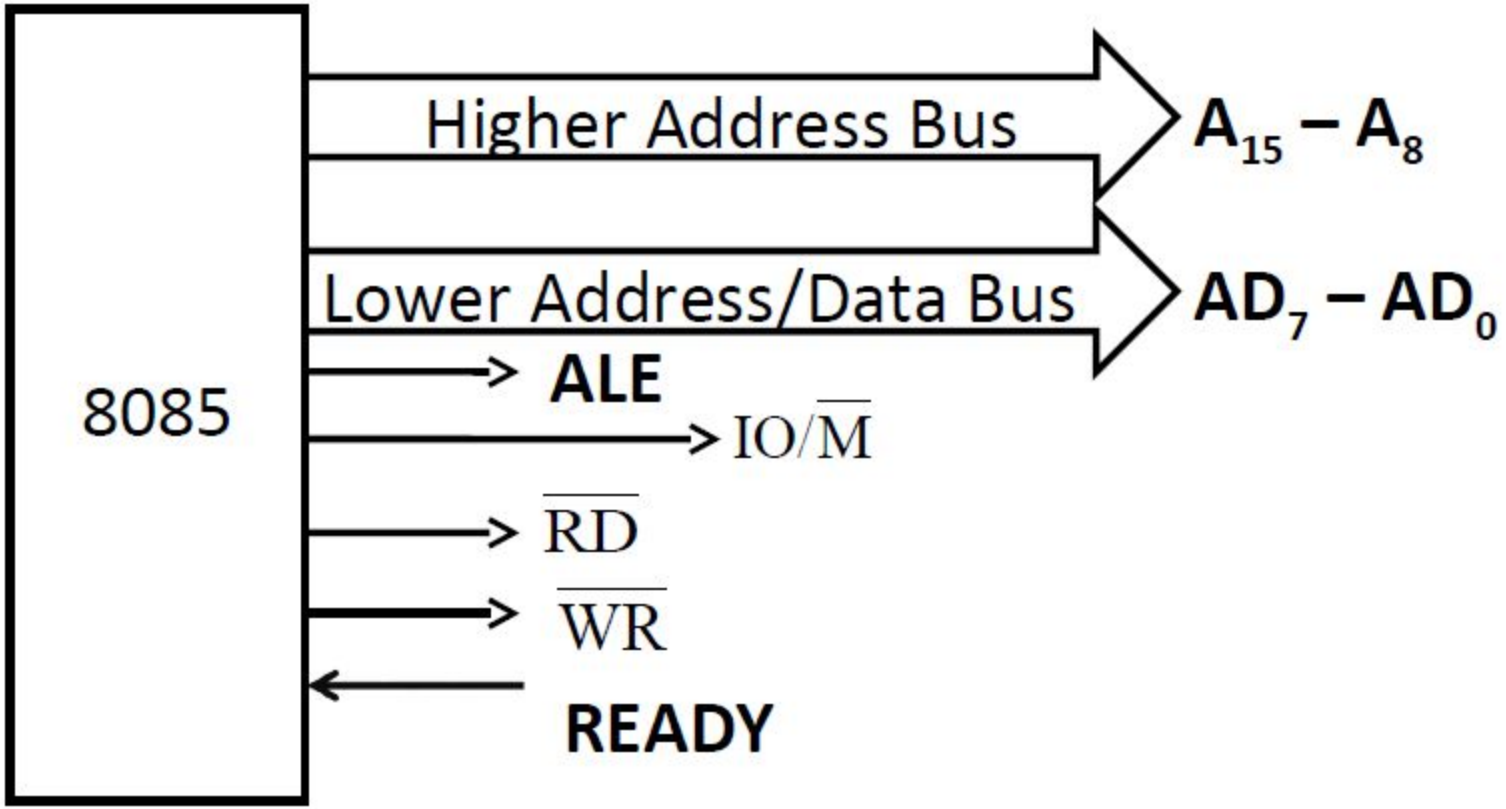
Secondary memory

- These are bulk storage
- Examples are floppy, hard disk, CD-ROM, magnetic tape, pen drive etc
- Slower and sequential access in nature
- non-volatile

Example Block Diagram



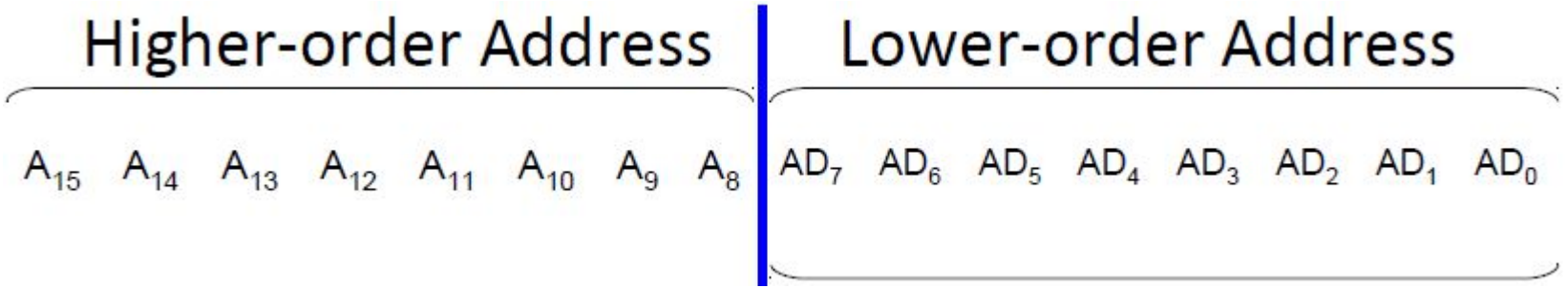
8085 interfacing pins



Address Bus of 8085

Address Bus

- 16 bit
- Used to address memory and I/O devices



Data Bus

- 8-bit
- Used to transfer instructions and data

Address bus

Higher order address bus (A15-A8)

- The higher order address bus is a unidirectional bus
- It carries most significant 8-bits of a 16-bit address of memory or I/O device.
- Address remains on lines as long as operation is not completed.

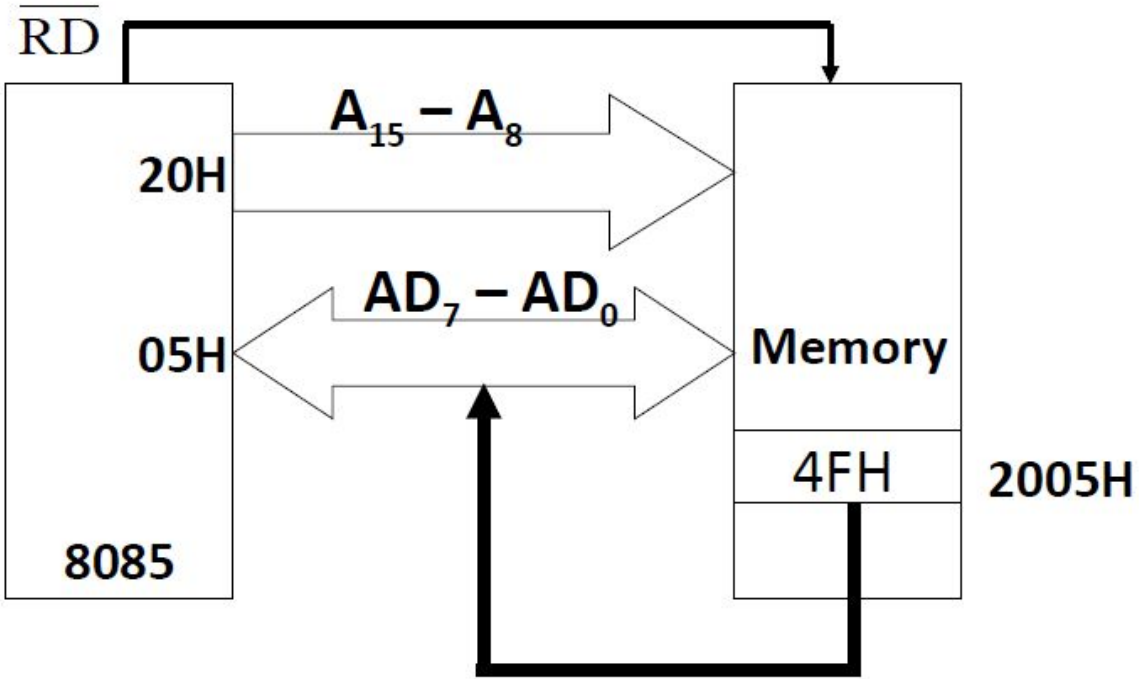
Lower order address / Data bus (AD7-AD0)

- bidirectional
- During first clock cycle, it serves as a least significant 8-bits of memory/ IO address
- For second and third clock cycles it acts as data bus and carries data.

Demultiplexing Address / Data lines

Lower order address bus & data bus are multiplexed on the same lines i.e AD0 to AD7

Demultiplexing refers to separating Address & Data signals for read / write operations.



Memory chip - R/W static memory

2048 registers and each register can store 8 bits.

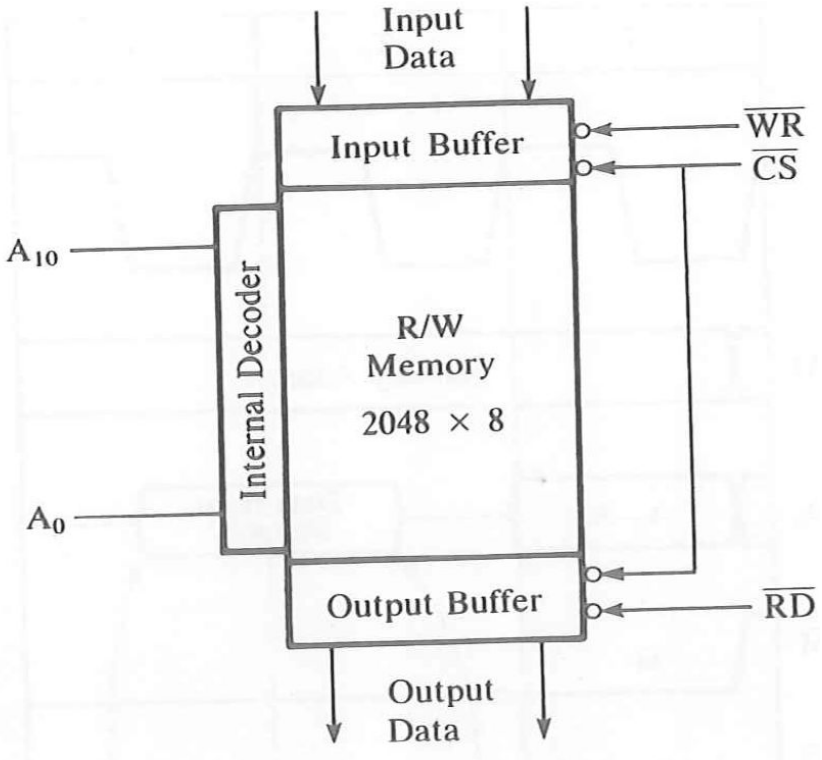
11 address lines A_{10}-A_0

2^{11}=2048

One chip select ($\overline{CS}$)

Two control lines $\overline{RD}$,- to enable read buffer

- $\overline{WR}$ - To enable write buffer



Memory chip - EPROM

4096 registers and each register can store 8 bits.

12 address lines A_{11}-A_0

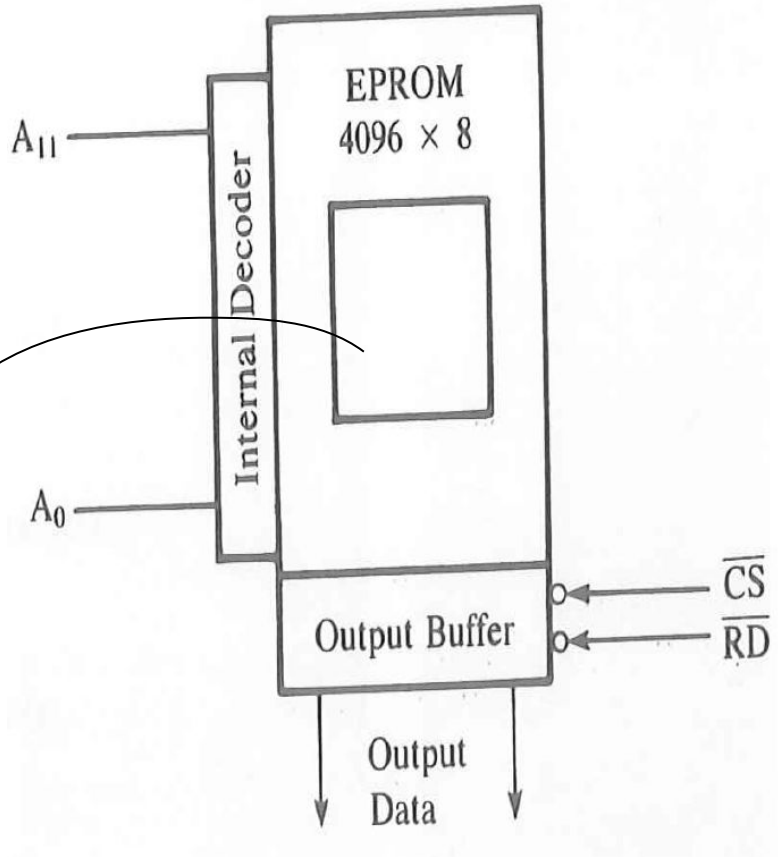
2^{12}=4096

One chip select ($\overline{CS}$)

One read control lines $\overline{RD}$,- to enable read buffer

Quartz window

-



Basic concepts in memory interfacing

The primary function is that microprocessor should be able to read from and write into a given register of a memory chip. To perform this , the microprocessor should

1. Be able to select the chip
2. Identify the register
3. Enable the appropriate buffer.

8085 timing for execution of the instruction MVI A,32H

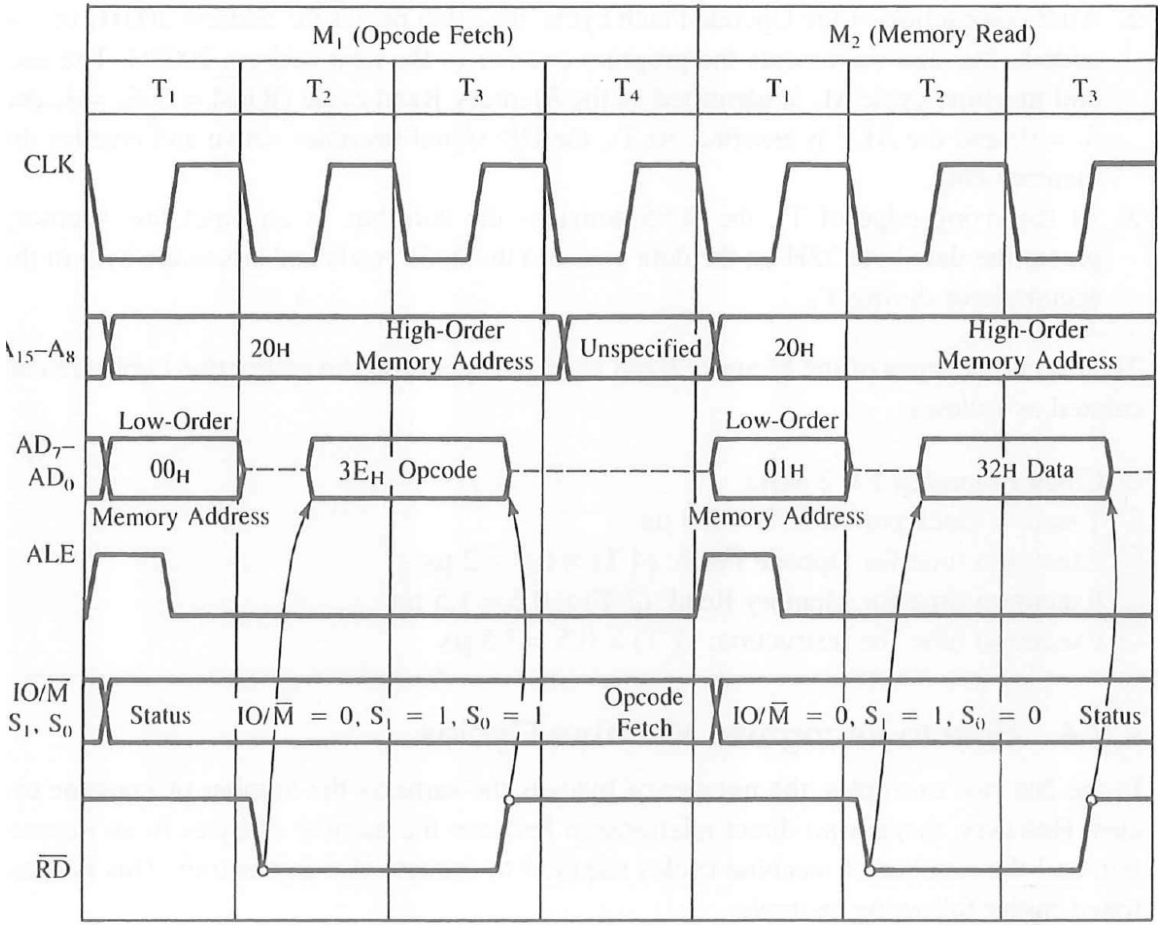
8085 places 16-bit address on address bus. For the memory chip in previous slide only 11 address lines are required to identify 2048 registers. Therefore A_{10} - A_0 is connected to memory chip.

2. The remaining address lines (A_{15}-A_{11}) is decoded to generate chip select($\overline{CS}$)

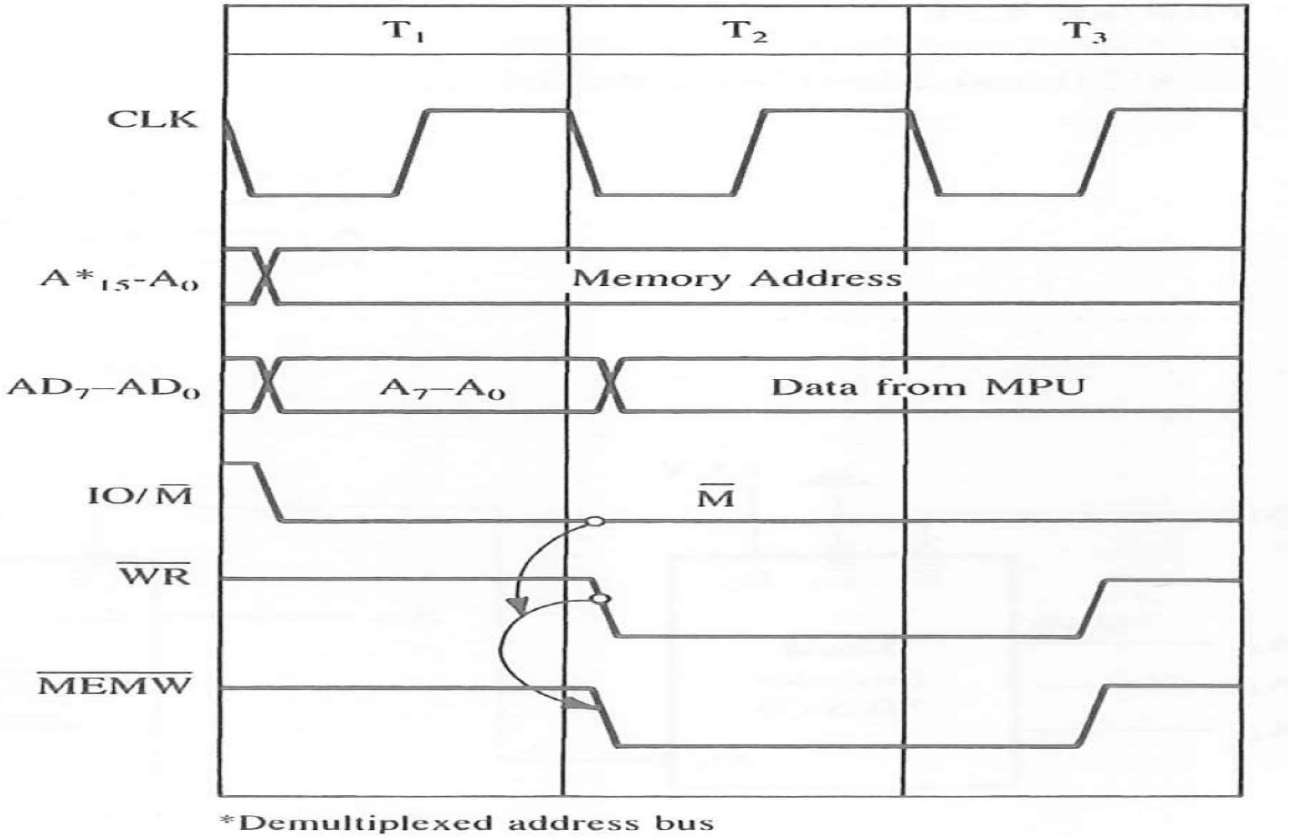
3. Two signals $IO\overline{M}$ and $\overline{RD}$

Indicate memory read operation

4. Memory places the data byte during T2 of M2



Timing of memory write cycle



*Demultiplexed address bus

Memory interface

To interface memory with microprocessor , the steps are

1. Connect the required address lines of the address bus to the address lines of the memory chip
2. Decode the remaining address lines of the address bus to generate the Chip select signal and connect the signal to select the chip
3. Generate control signals $\overline{MEMR}$ and $\overline{MEMW}$

Address decoding

The process of address decoding should result in identifying a register for a given address.

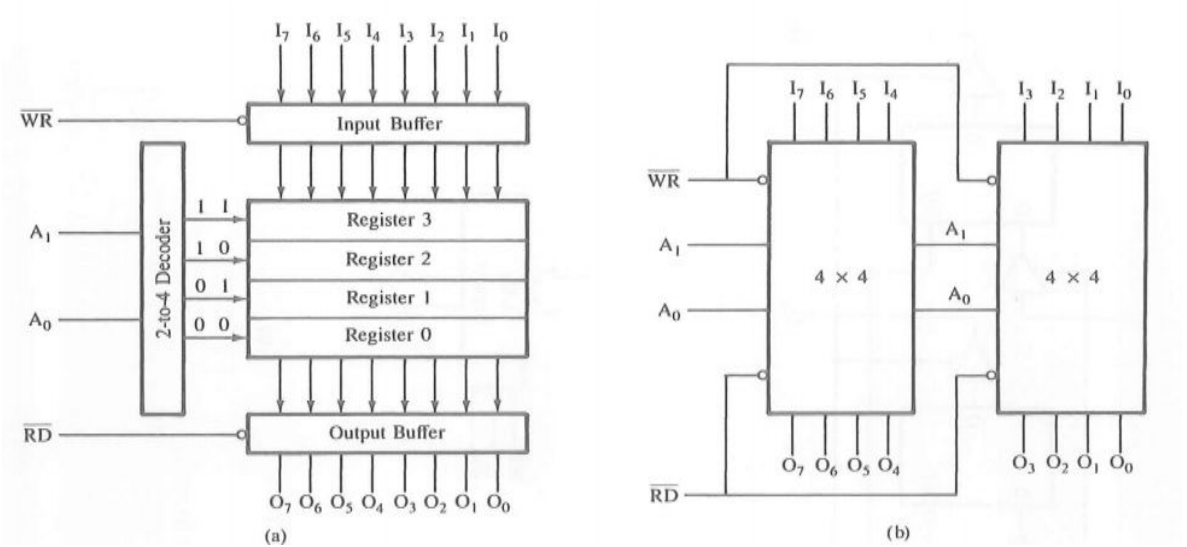
A unique pulse should be generated for a given address

For example the memory chip given in slide 11, 12 address lines are connected to memory chip (A_{11}-A_0) the remaining four address lines (A_{15}-A_{12}) must be decoded. This can be done by using only NAND gates or by 3-to-8 decoder

4 x 8 - bit register

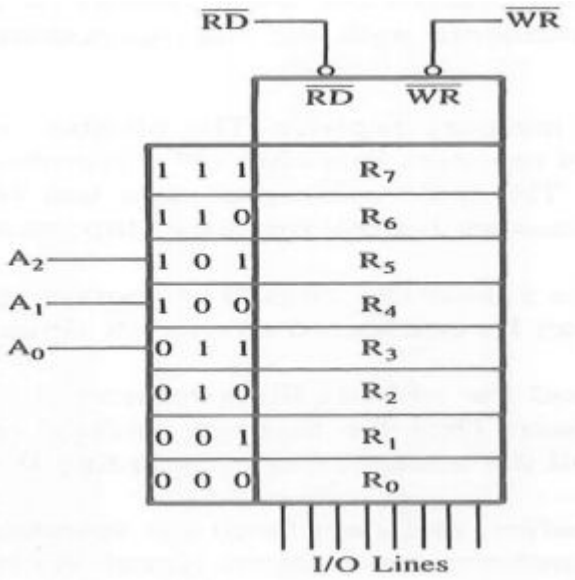
Four registers with eight cells are arranged in sequence. To write and read from any register, a specific register should be identified. We have 2-to-4 decoder to do that

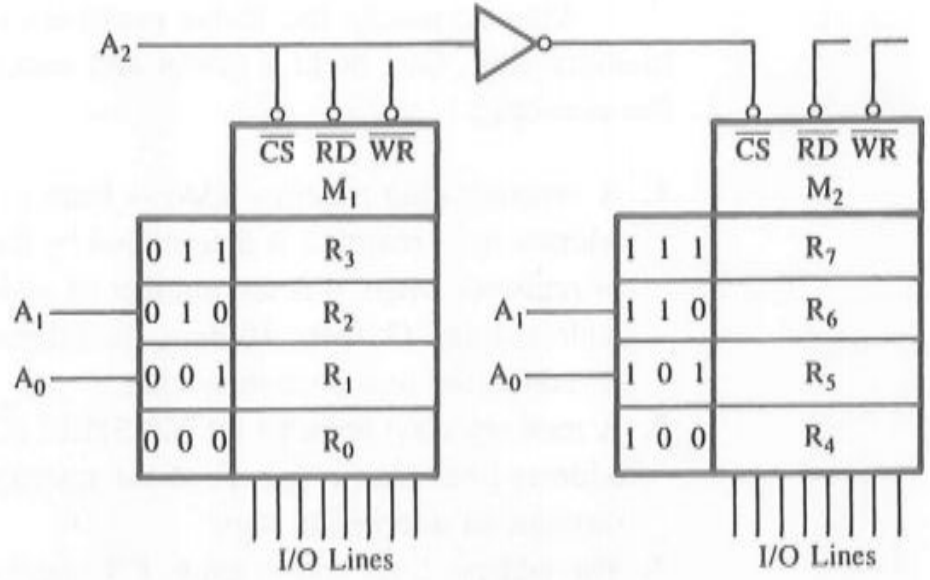
As 4 registers has to be identified we need 2 address lines with combinations (00,01,10,11) to identify R0,R1,R2,R3



If two chips are there how to identify. Other than 2 lines to address individual registers we need one more address line which select between these two. The concept of chip select gives an idea of expanding memory.

Two memory chips with four registers each and chip select





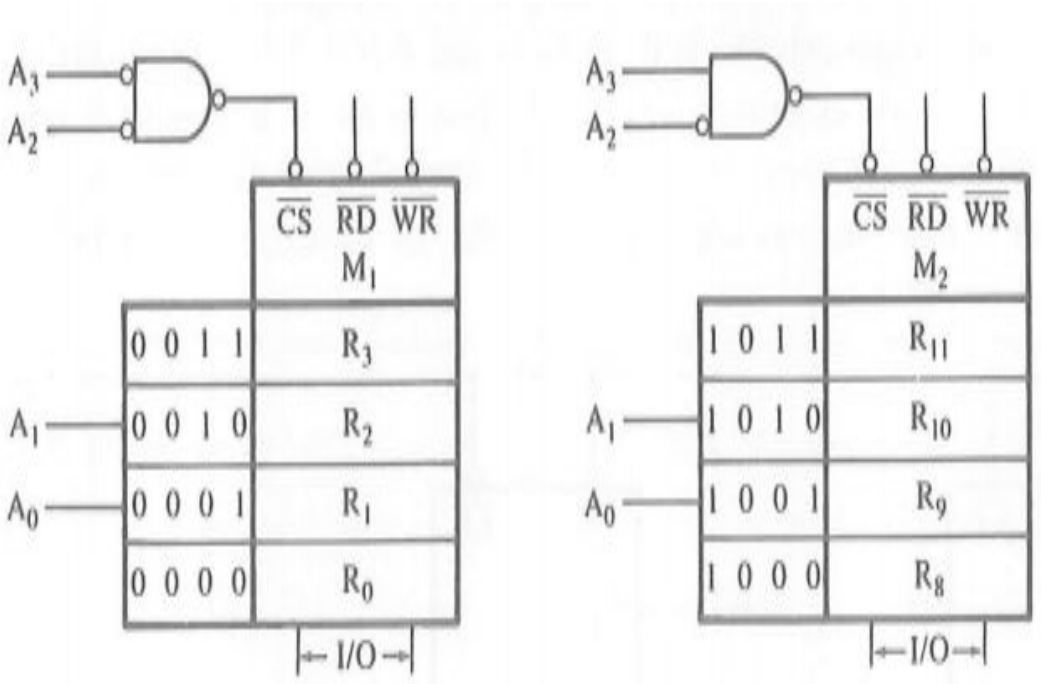
Addressing eight registers with four address lines

Assume we have four address lines and two memory chips with four registers each.

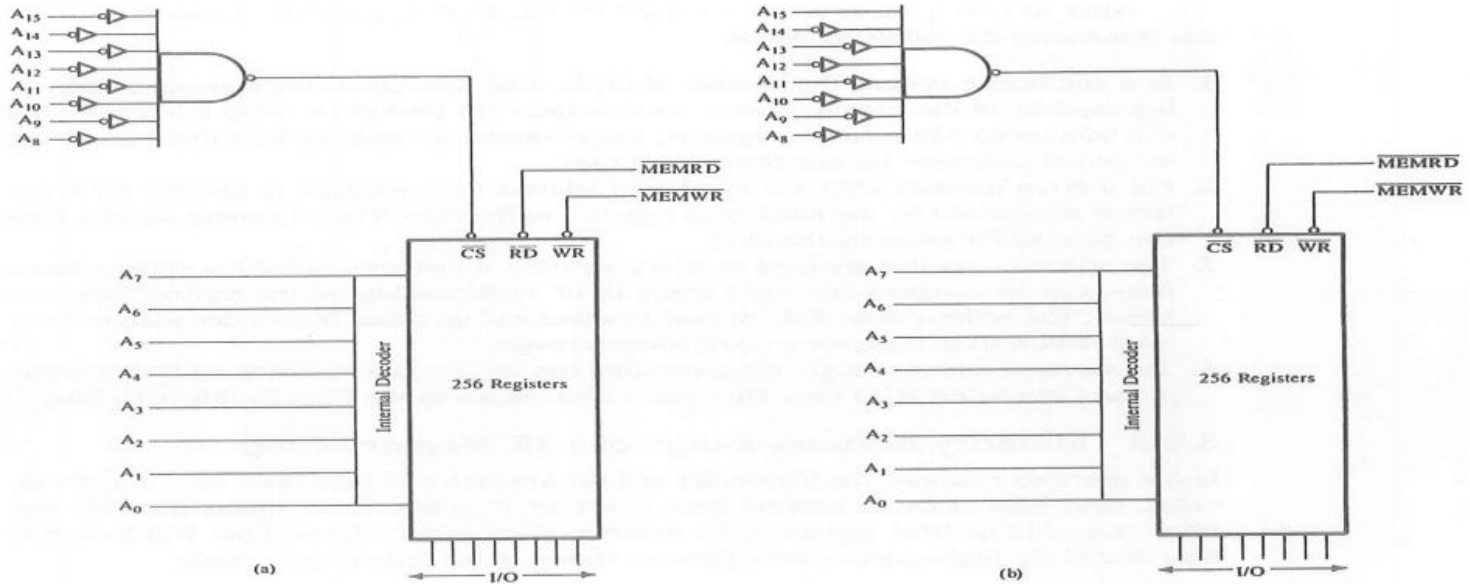
We need only three address lines. What can be done with fourth line.

Memory chip M1 is selected when A3 and A2 are =0 address ranging from 0000 to 0011 (0 to 3)

Memory chip M2 is selected when A3=1 and A2=0 address ranging from 1000 to 1011(8 to B)



Illustrate the memory address range of the chip with 256 bytes of memory



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								

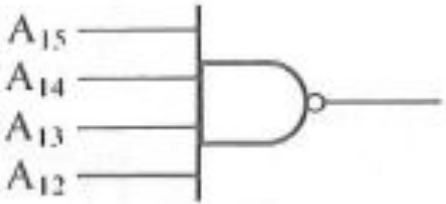
If the inverter on line A15 is removed , then address range is 8000H to 80FFH

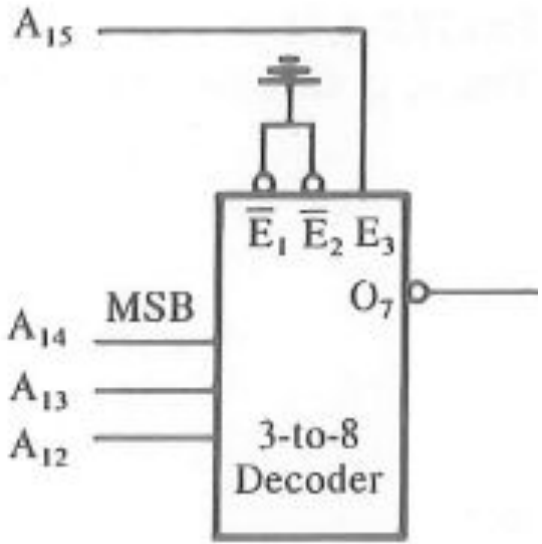
A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	
1	0	0	0	0	0	0	0	= 80H

Address decoding using NAND and 3-to-8 decoder

The output of the NAND gate goes active and selects the chip only when all address lines A_{15}-A_{12} are at logic 1.

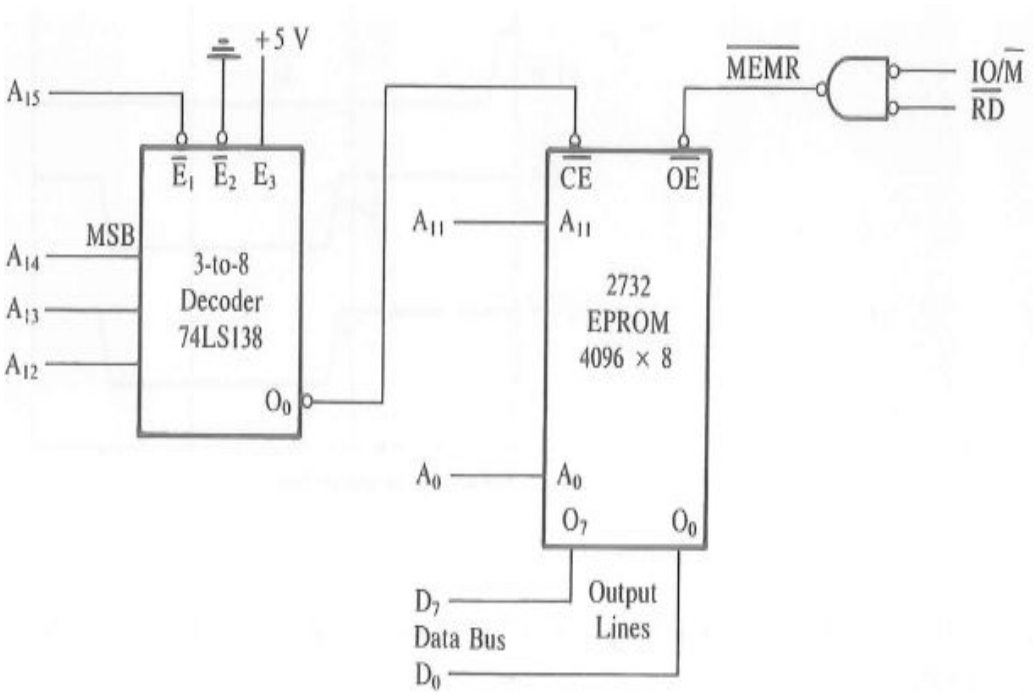
The same result can be obtained using O_7, of 3-t-8 decoder





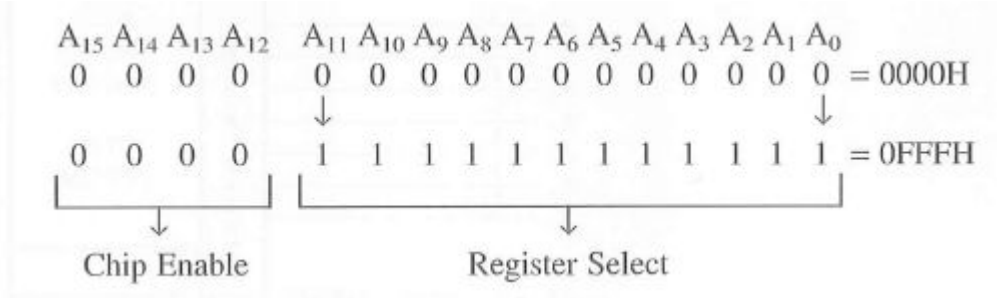
Interfacing example- 3-to-8 decoder to interface 2732 EPROM

1. The 8085 address lines A_{11}-A_0 are connected to pins A_{11}-A_0 of the memory chip to address 4096 registers
2. The decoder is used to decode four address lines A_{15}-A_{12}. The output O_0 of the decoder is connected to chip enable low. The chip enable low is asserted only when the address on A_{15}-A_{12} is 0000. A_{15}(low) enables the decoder and the input 000 asserts the output O_0
3. For this PROM, one control signal MEMR active low is needed. It is connected to OE low to enable the output buffer.



Address decoding and memory addresses

The address range of this memory chip can be obtained as follows:



The address lines A_{15}-A_{12} =0000 to assert Chip Enable

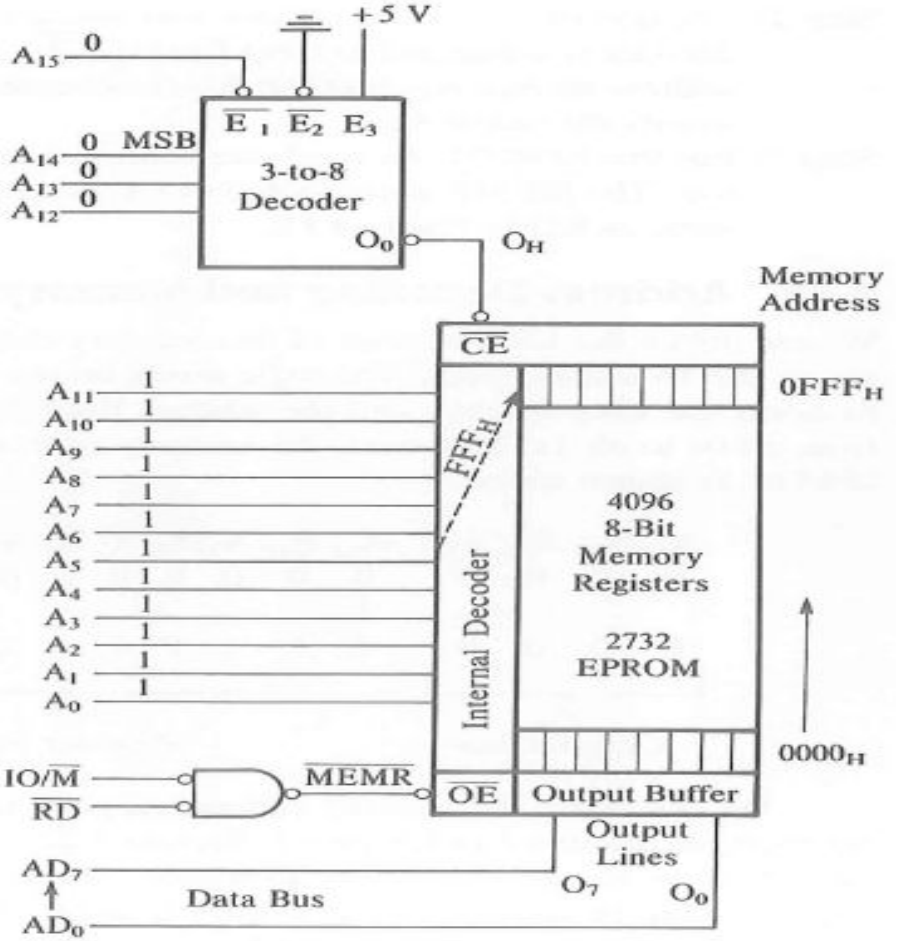
The address line A_{11}-A_0 = 0 to 1

The chip's 4096 bytes of memory can be viewed as 16 pages with 256 lines each.

0000-0FFFH (if the higher order is taken as 00-0F (16 lines) (if lower order is taken as 00-FF (256 lines)

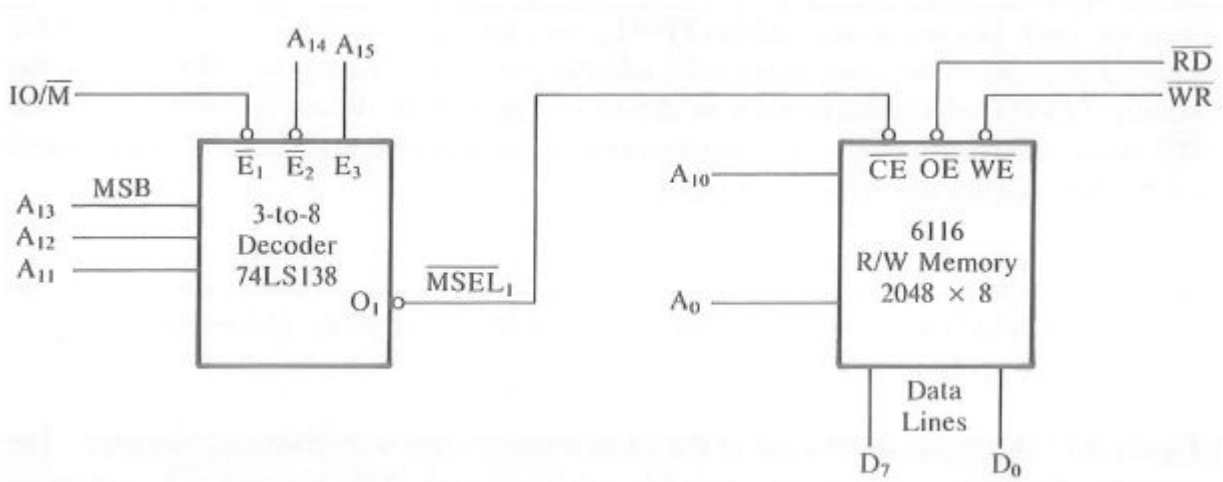
Address decoding and reading from memory

Reading data from memory 0FFFH



Memory interfacing

Example : analyse the interfacing circuit and find its memory address range



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	
1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	= 8800H
					1	1	1	1	1	1	1	1	1	1	1	= 8FFFH

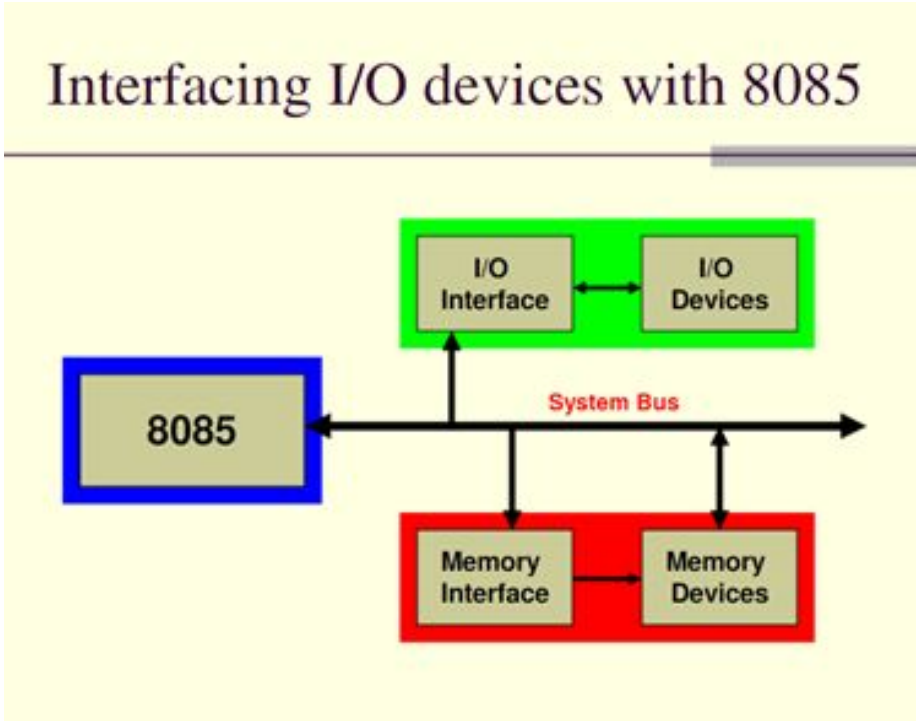
$MSEL_1$

Example for memory address range

Interfacing I/O devices

Techniques for I/O Interfacing

- Memory-mapped I/O
- Peripheral-mapped I/O



8085 communication with I/O devices

Important steps:

1. Identify the I/O device (with address)

   Memory-mapped I/O(16 bit address)

   Peripheral-mapped I/O (8-bit address)

2. Generate Timing & Control signals

3. Data Transfer takes place
4.

2. Generate Timing & Control Signals

Memory-mapped I/O

Reading Input : $IO\overline{M}$ =0, $\overline{RD}$ =0

Write to output $IO\overline{M}$ =0, $\overline{WR}$ =0

Peripheral-mapped I/O

Reading Input : $IO\overline{M}$ =1, $\overline{RD}$ =0

Write to output $IO\overline{M}$ =1, $\overline{WR}$ =0

Peripheral I/O instructions

IN instruction

IN 8-bit

Example : IN 01H

Inputs data from input device into the accumulator

2-byte instruction

OUT instruction

OUT 8-bit

Example : OUT 01H

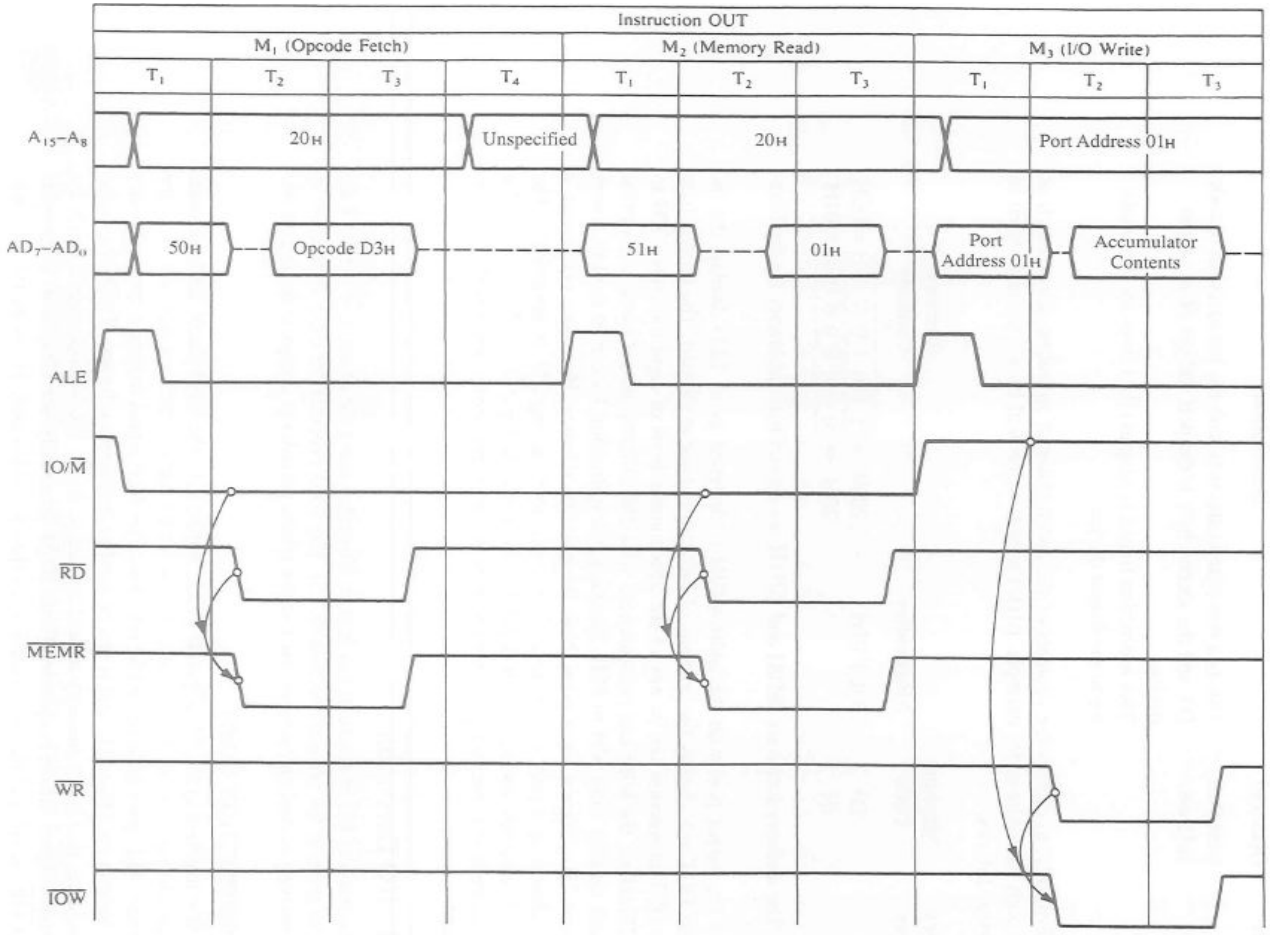
Outputs the contents of accumulator to an output device

2-byte instruction

Timing diagram of OUT instruction

Consider the instruction

Mem	code	mnemonics
2050	D3	OUT 01H
2051	01	



In M1- opcode fetch- 8085 places the high-order memory address 20H on A15-A8 and 50H in lower order AD7-AD0. ALE goes low indicating that it is a memory-related operation.
At T2- the MP sends the $\overline{RD}$ control signal, which is combined with $IO\overline{M}$ to generate $\overline{MEMR}$ and fetches the instruction code D3 using data bus. On decoding ,MP finds that the instruction is 2-byte.
In M2- MP places the next address, 2051H, on the address bus and gets the device address 01H via data bus.
In M3- the device address 01H is placed in low-order (AD7-AD0) as well as in high-order A15-A8. The $IO\overline{M}$ goes high indicating it is an I/O operation
T2-, the accumulator contents are placed on data bus, followed by control signal $\overline{WR}$
By ANDing the $IO\overline{M}$ and $\overline{WR}$ signals $\overline{IOW}$ signal is generated to enable output device.

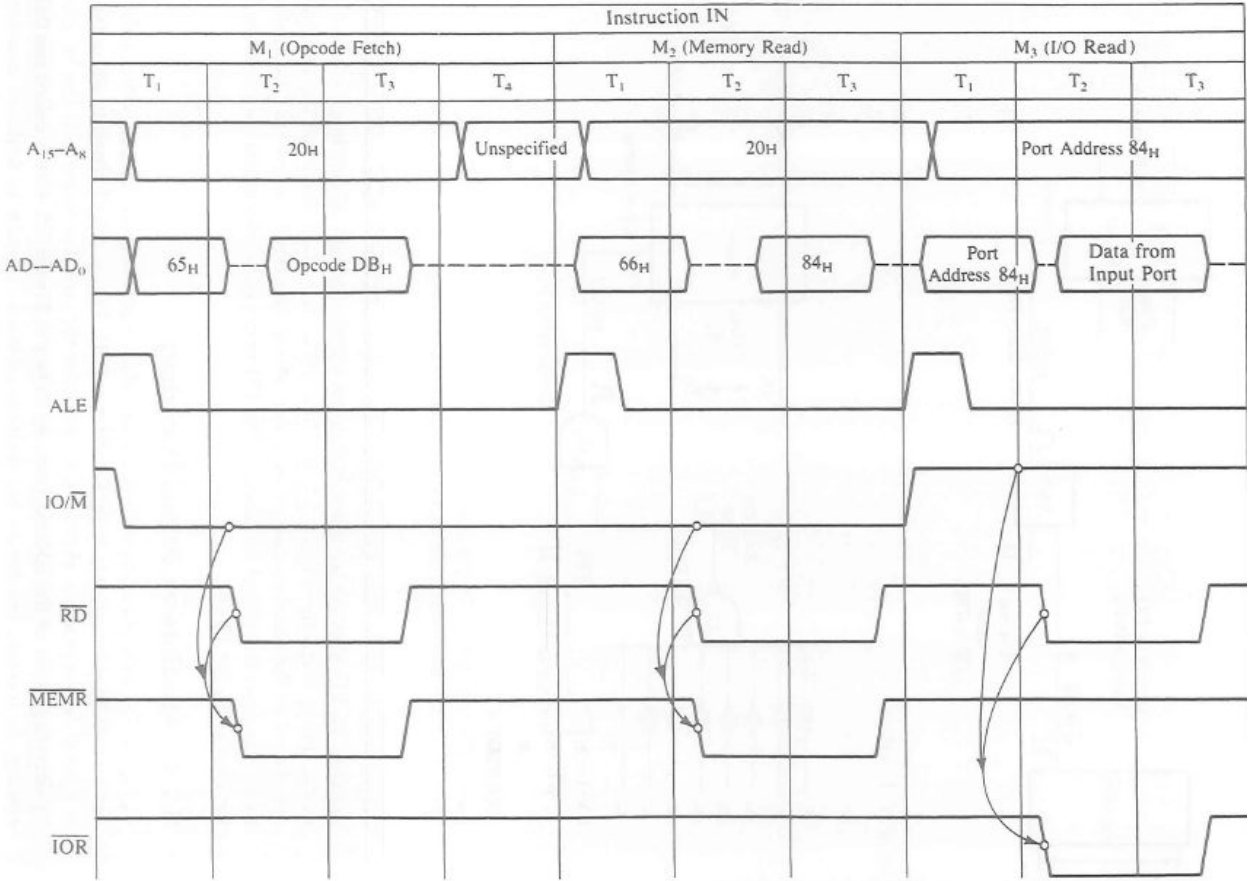
The information necessary is obtained at T2 and T3 of M3 cycle

Timing diagram of IN instruction

Example : consider the instruction **IN 84H**

Consider the instruction

Mem	code	mnemonics
2065	DB	IN 84H
2066	84	



In M1- opcode fetch- 8085 places the high-order memory address 20H on A15-A8 and 65H in lower order AD7-AD0. ALE goes low indicating that it is a memory-related operation.
M1 and M2- are identical to OUT instruction.

In M3- the device address 84H is placed in low-order (AD7-AD0) as well as in high-order A15-A8 and asserts which $\overline{RD}$ used to generate $\overline{IOR}$
The $\overline{IOR}$ enables the input port, and the data from input port are placed on the data bus and transferred to the accumulator.

Device selection and Data transfer

The objective of interfacing is to get information or a result out of the or into the processor and store it or display it. The OUT instruction serves that purpose.

During M3 cycle of the OUT instruction the MP places that information on the data bus.

If the data bus is connected to the latch, the information can be caught and displayed via LED or printer.

The questions are

1) When should we enable the latch to catch that information?
2) What should be the address of that latch?

The answer is at M3 cycle. The latch should be enabled when $IO\overline{M}$ is high and $\overline{WR}$ is active low.

The address of an output port is also on the address bus during M3.

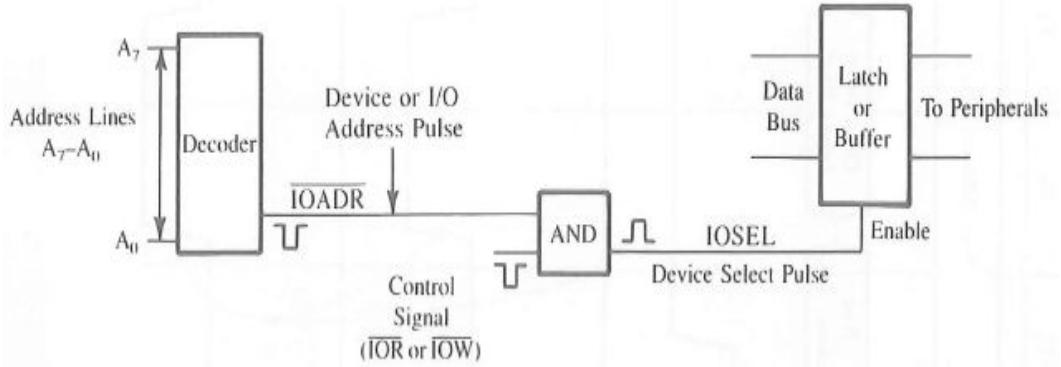
The steps to be done for device selection and data transfer

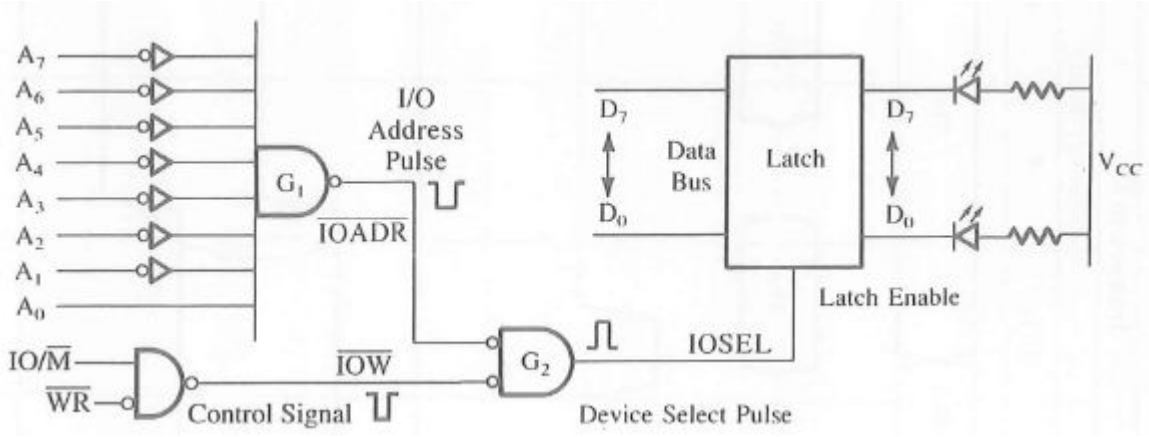
1. Decode the address bus to generate a unique pulse corresponding to the device address on the bus. This is called the device address pulse or I/O address pulse.
2. Combine (AND) the device address pulse with the control signal to generate a device select(I/O select) pulse that is generated only when both signals are asserted.
3. Use the I/O select pulse to activate the interfacing device (I/O port)

Block diagram of I/O interface

Decode logic for LED output port

The figure shows a decoding circuit for the output device with address 01H. Line A0 is connected directly, and lines A7-A1 are connected through the inverters. When the address bus carries address 01H, gate G1 generates a low pulse, otherwise, the output remains high. Gate G2 combines output of G1 and the control signal $\overline{IOW}$ to generate I/O select pulse when both signals are low. The content of accumulator are placed on data bus and are available for few microseconds.



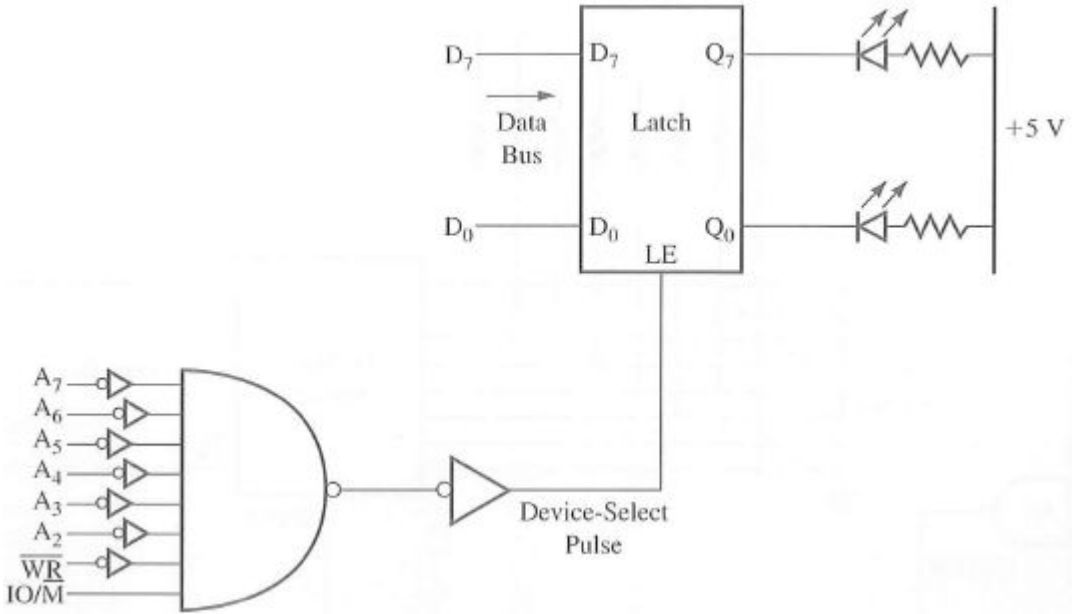


Absolute vs Partial Decoding

When all eight address lines are decoded to generate a unique output pulse, we call it as **absolute decoding.**

Partial decoding : when only few address lines are used for generating unique output pulse, we say it as partial decoding. Because of this a device has multiple addresses

In the figure the address lines A1 and A0 are not connected. And they are replaced by $IO\overline{M}$ and $\overline{WR}$ signals.

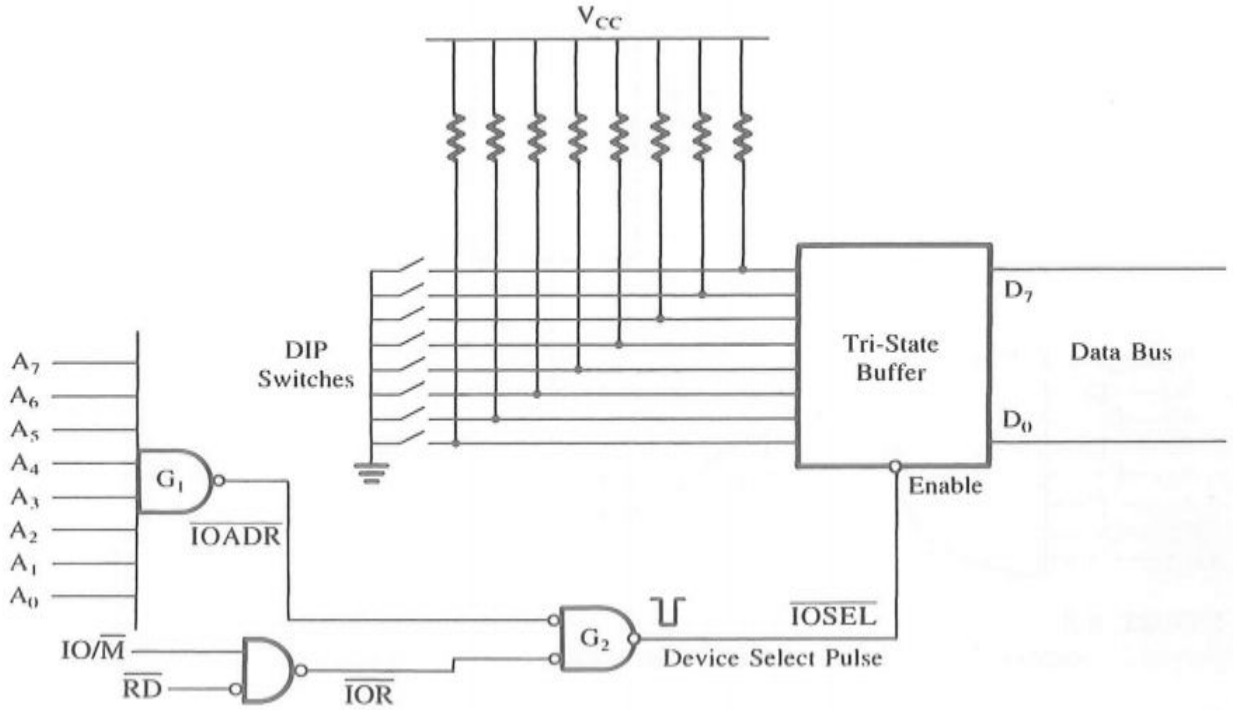


Decode logic for a Dip-Switch input port

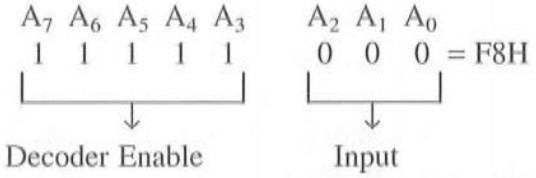
A7-A0=1 (FFH)

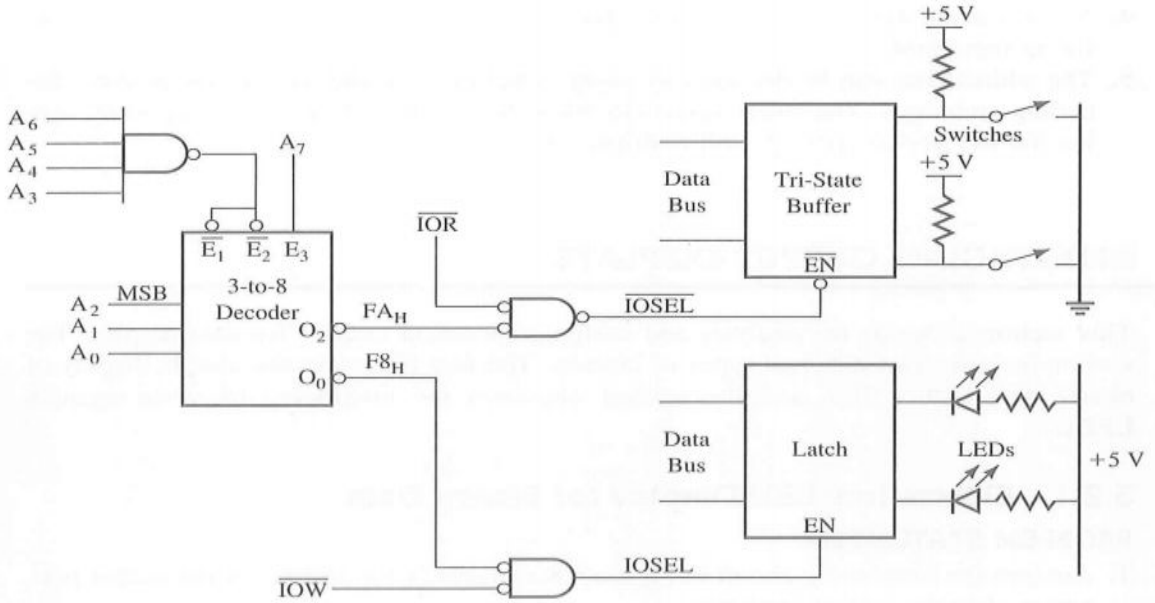
G1=0 + $\overline{IOR}$ = G2

IN FFH, G2 generates device select pulse to enable tri-state buffer.



Interfacing I/Os using decoders





Memory-mapped I/O

- 8085 uses its 16-bit address bus to identify a memory location
- Memory address space : 0000H to FFFFH
- 8085 needs to identify I/O devices also
- I/O devices can be interfaced using addresses from memory space
- 8085 treats such an I/O device as a memory location

This is called Memory-mapped I/O

Peripheral-mapped I/O

- 8085 has a separate 8-bit addressing scheme for I/O devices
- I/O address space : 00H to FFH

This is called Peripheral-mapped I/O or I/O-mapped I/O

Memory-mapped I/O

To transfer data between MPU and I/O devices, memory-related instructions (LDA, STA etc) and memory control signals ($\overline{MEMR}$ $\overline{MEMW}$) are used

The microprocessor communicates with an I/O device as if it were one of the memory locations. In the below example if an output device , instead of a memory register is connected at this address , the accumulator contents will be transferred to the output device. This is called memory-mapped I/O technique.

Memory Address	Machine Code	Mnemonics
2050	32	STA 8000H
2051	00	
2052	80	

Comparison of memory-mapped I/O with peripheral I/O

Characteristics	Memory-Mapped I/O	Peripheral I/O
1. Device address	16-bit	8-bit
2. Control signals for Input/Output	$\overline{MEMR}$/$\overline{MEMW}$	$\overline{IOR}$/$\overline{IOW}$
3. Instructions available	Memory-related instructions such as STA; LDA; LDAX; STAX; MOV M,R: ADD M; SUB M; ANA M: etc.	IN and OUT
4. Data transfer	Between any register and I/O	Only between I/O and the accumulator
5. Maximum number of I/Os possible	The memory map (64K) is shared between I/Os and system memory	The I/O map is independent of the memory map; 256 input devices and 256 output devices can be connected
6. Execution speed	13 T-states (STA,LDA) 7 T-states (MOV M,R)	10 T-states
7. Hardware requirements	More hardware is needed to decode 16-bit address	Less hardware is needed to decode 8-bit address
8. Other features	Arithmetic or logical operations can be directly performed with I/O data	Not available