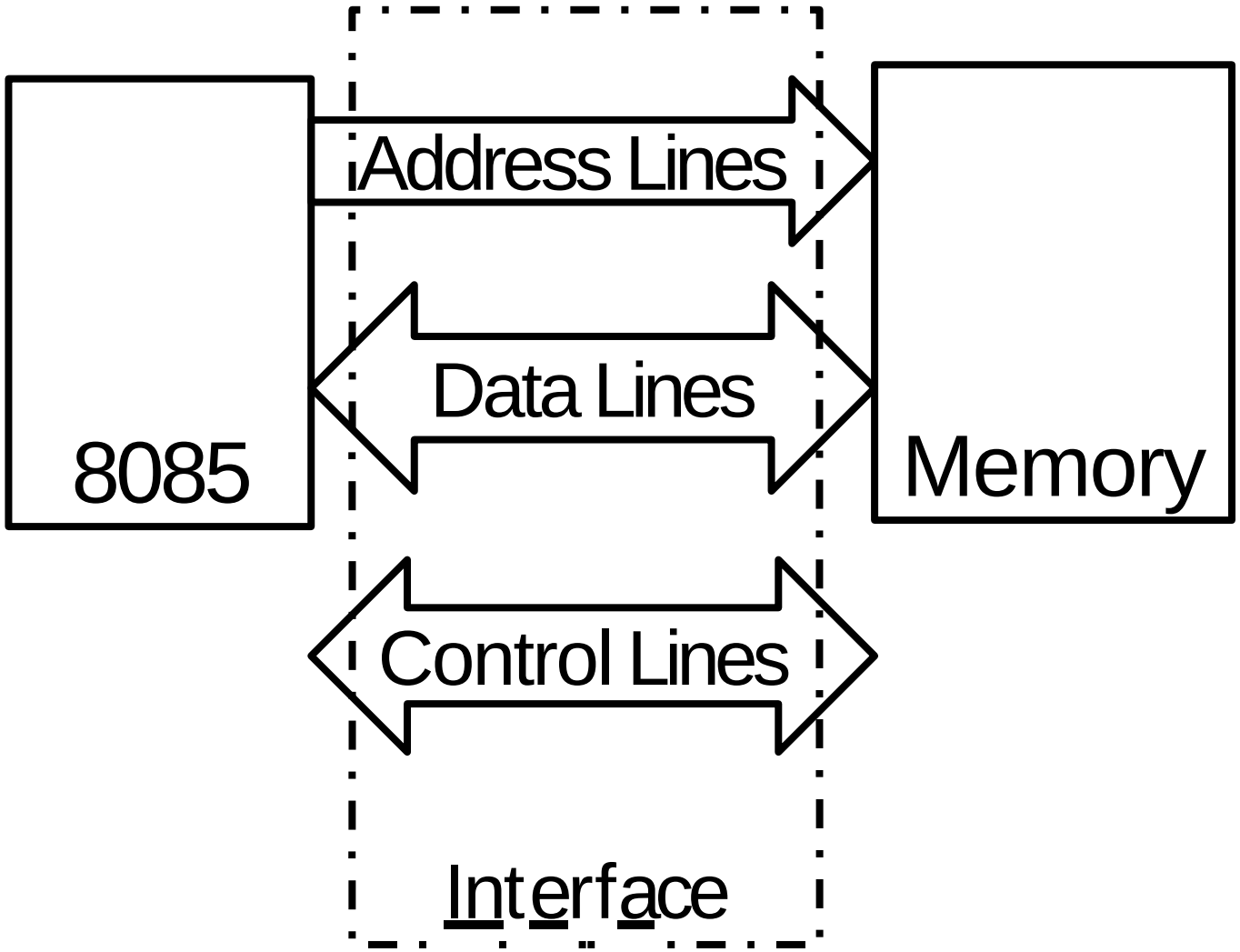
Memory and I/O Interfacing with 8085 Microprocessor

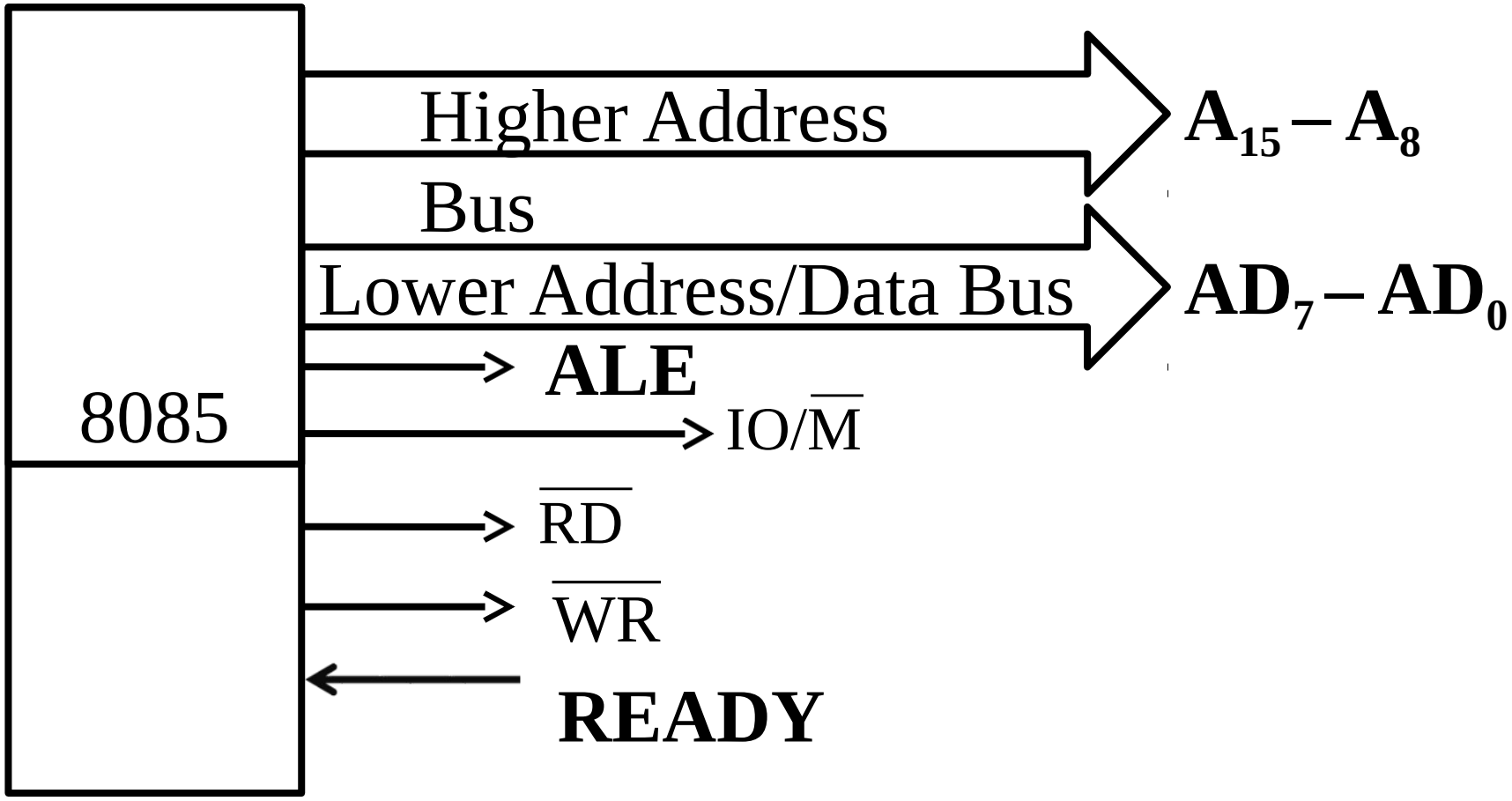
What is an Interface

- An **interface** is a concept that refers to a point of interaction between components, and is applicable at the level of both hardware and software.
- This allows a component, (such as a **graphics card** or an **Internet browser**), to function independently while using interfaces to communicate with other components via an **input/output system** and an **associated protocol**.

Example Block Diagram

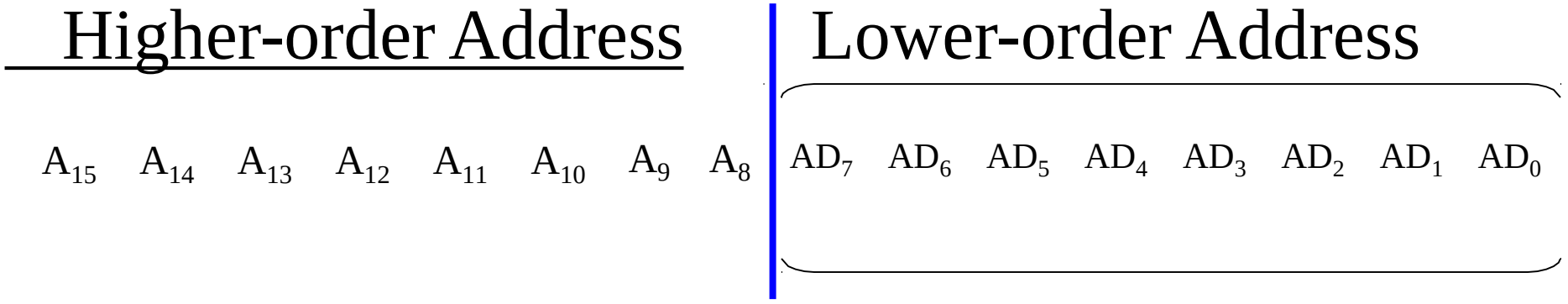


8085 Interfacing Pins



Address Bus of 8085

- Address Bus
  - Used to address memory & I/O devices
  - 8085 has a 16-bit address bus



- Data Bus
  - Used to transfer instructions and data
  - 8085 has a 8-bit data bus

Higher Order Address Bus

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

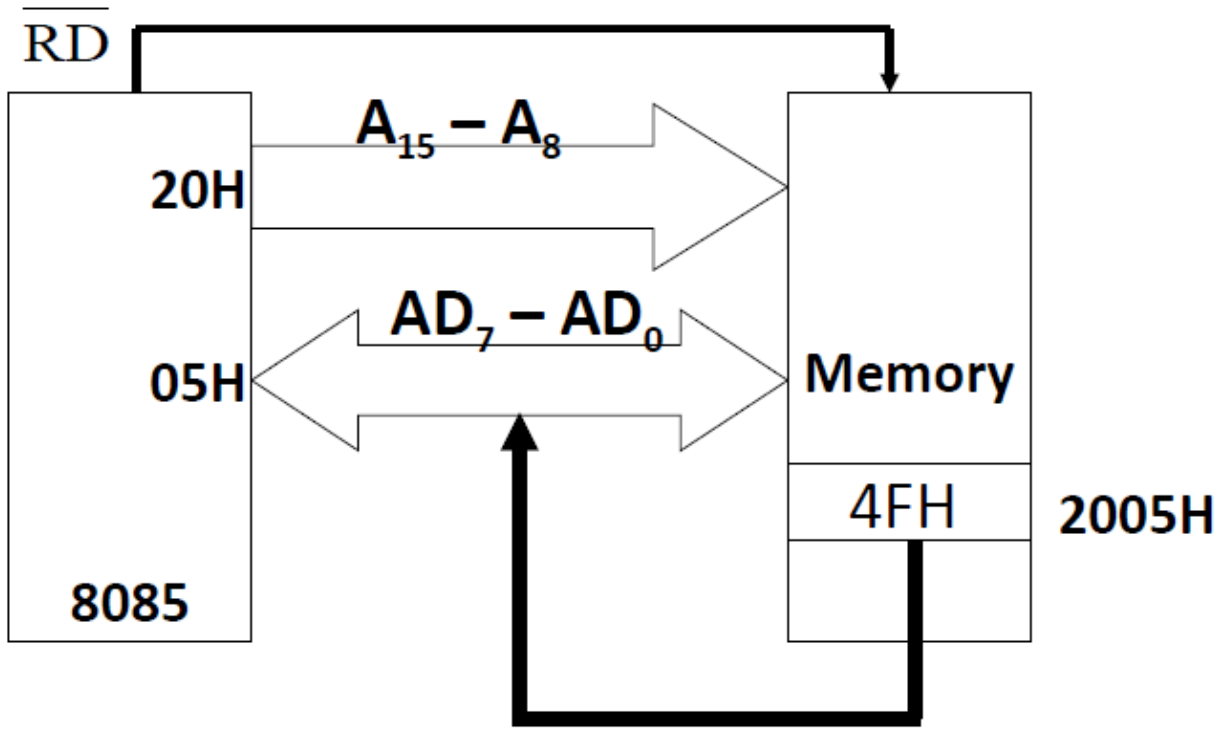
Lower Order Address/Data Bus

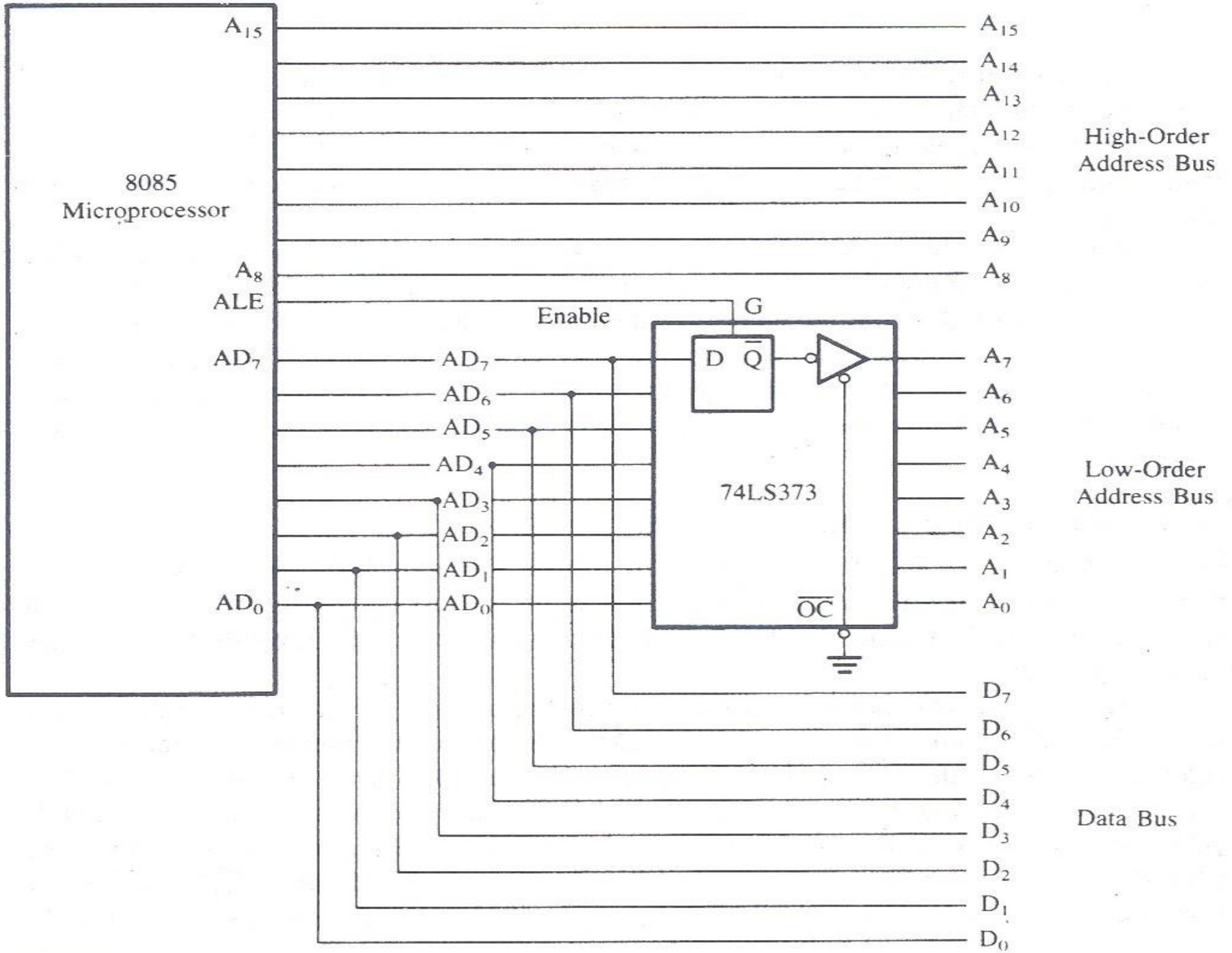
- This bus is bidirectional and works on time division multiplexing between address and data.
- During first clock cycle, it serves as a least significant 8-bits of memory/ IO address.
- For second and third clock cycles it acts as data bus and carries data.

Demultiplexing Address/Data Lines

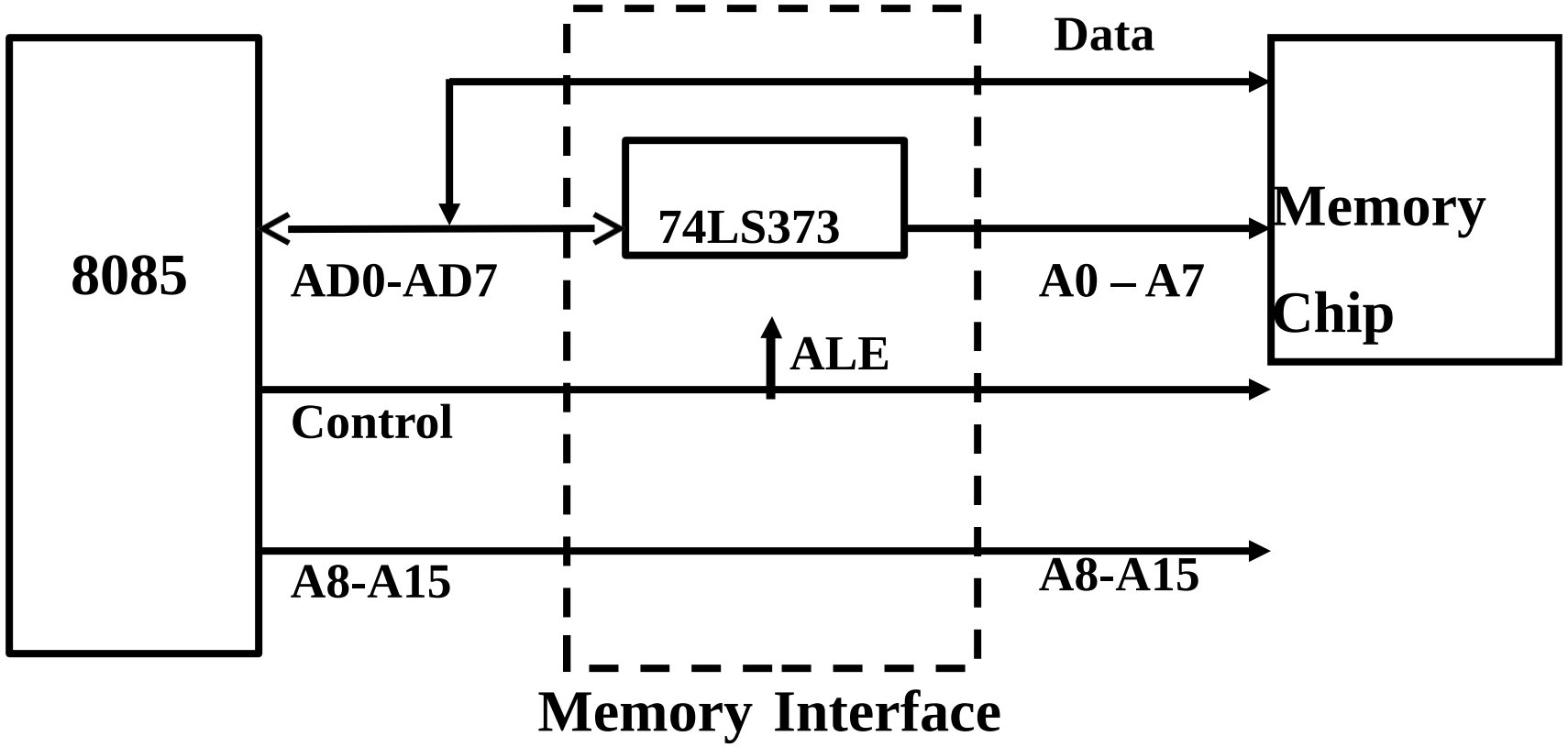
- 8085 identifies a memory location with its 16 address lines, (AD0 to AD7) & (A8 to A15)
- 8085 performs data transfer using its data lines, AD0 to AD7
- Lower order address bus & Data bus are multiplexed on same lines i.e. AD0 to AD7.
- Demultiplexing refers to separating Address & Data signals for read/write operations.

Need for Demultiplexing…

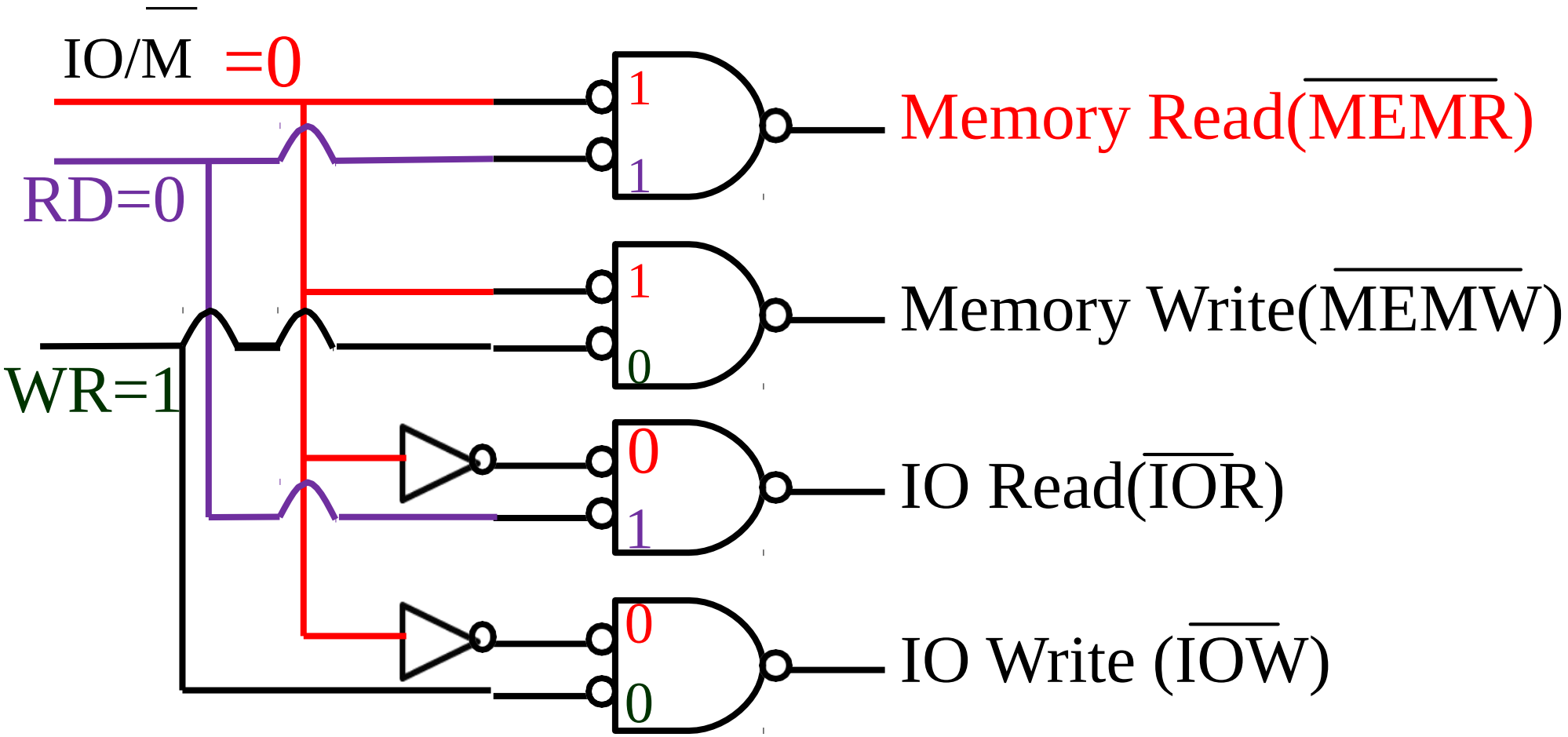



8085
Microprocessor
A_{15}
A_8
ALE
AD_7
AD_0
Enable
G
D
Q
74LS373
$\overline{OC}$
AD_7
AD_6
AD_5
AD_4
AD_3
AD_2
AD_1
AD_0
A_{15}
A_{14}
A_{13}
A_{12}
A_{11}
A_{10}
A_9
A_8
A_7
A_6
A_5
A_4
A_3
A_2
A_1
A_0
D_7
D_6
D_5
D_4
D_3
D_2
D_1
D_0
High-Order
Address Bus
Low-Order
Address Bus
Data Bus

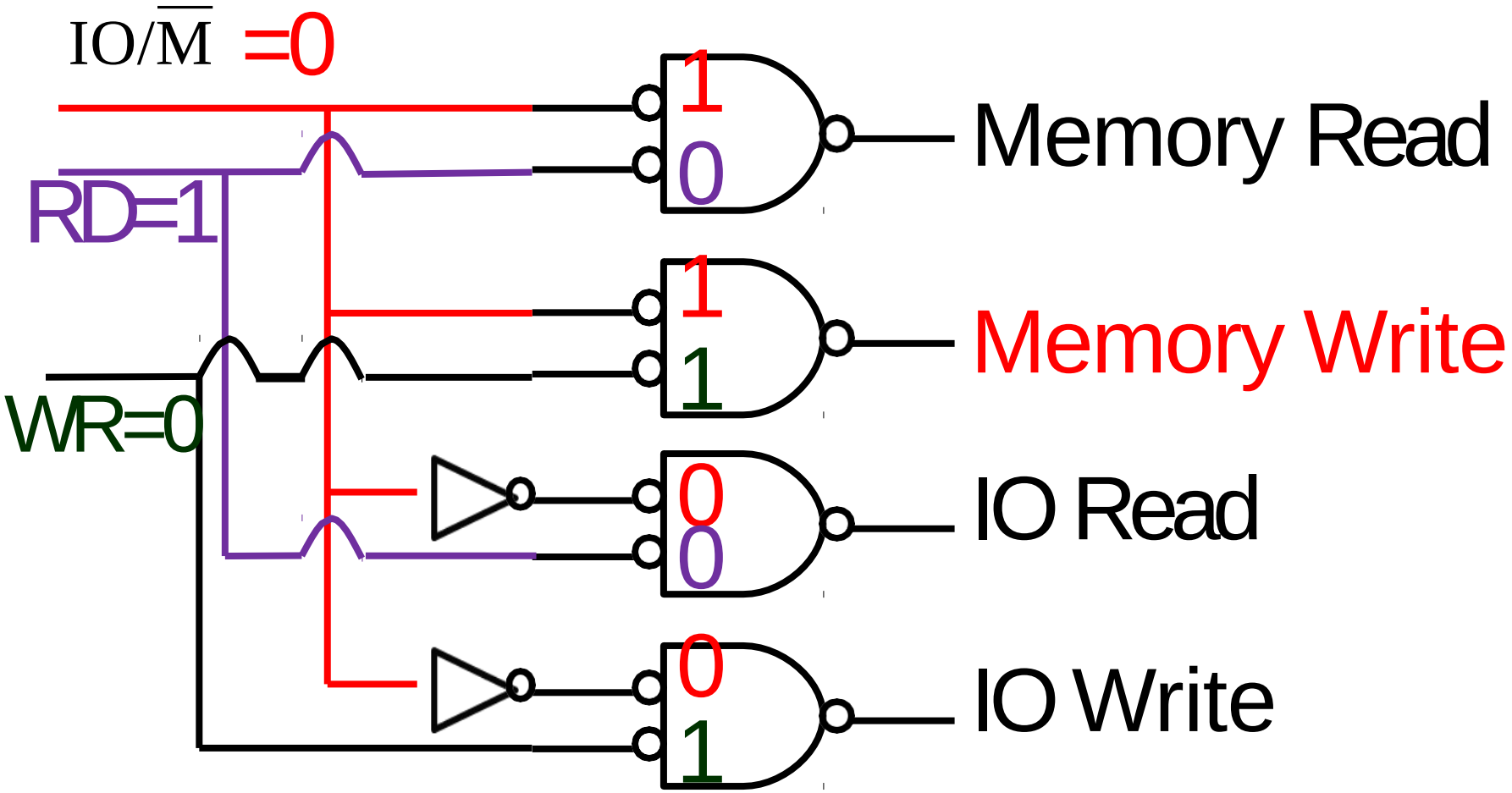
Demultiplexing Address/Data Lines



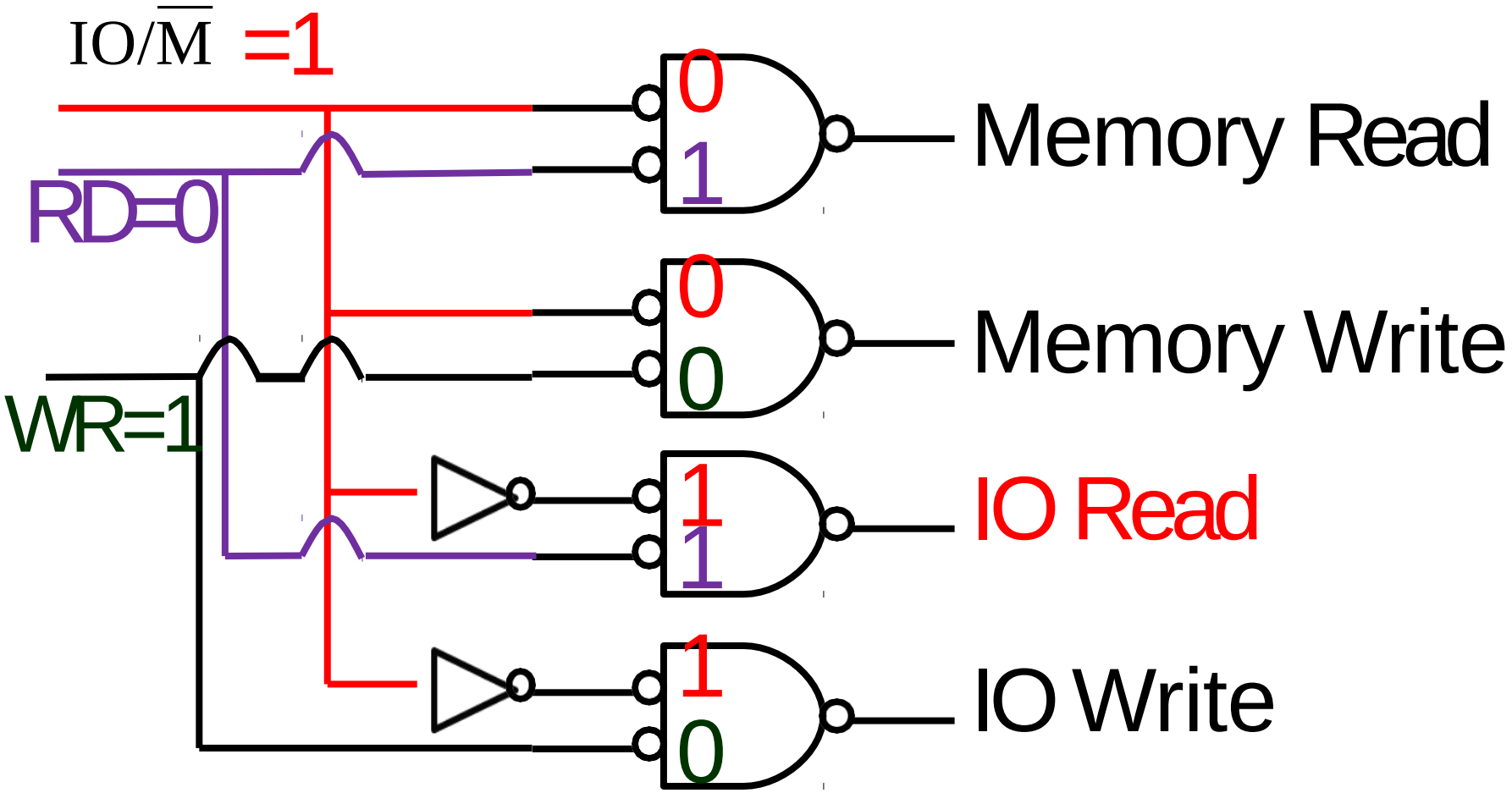
Generating Control Signals



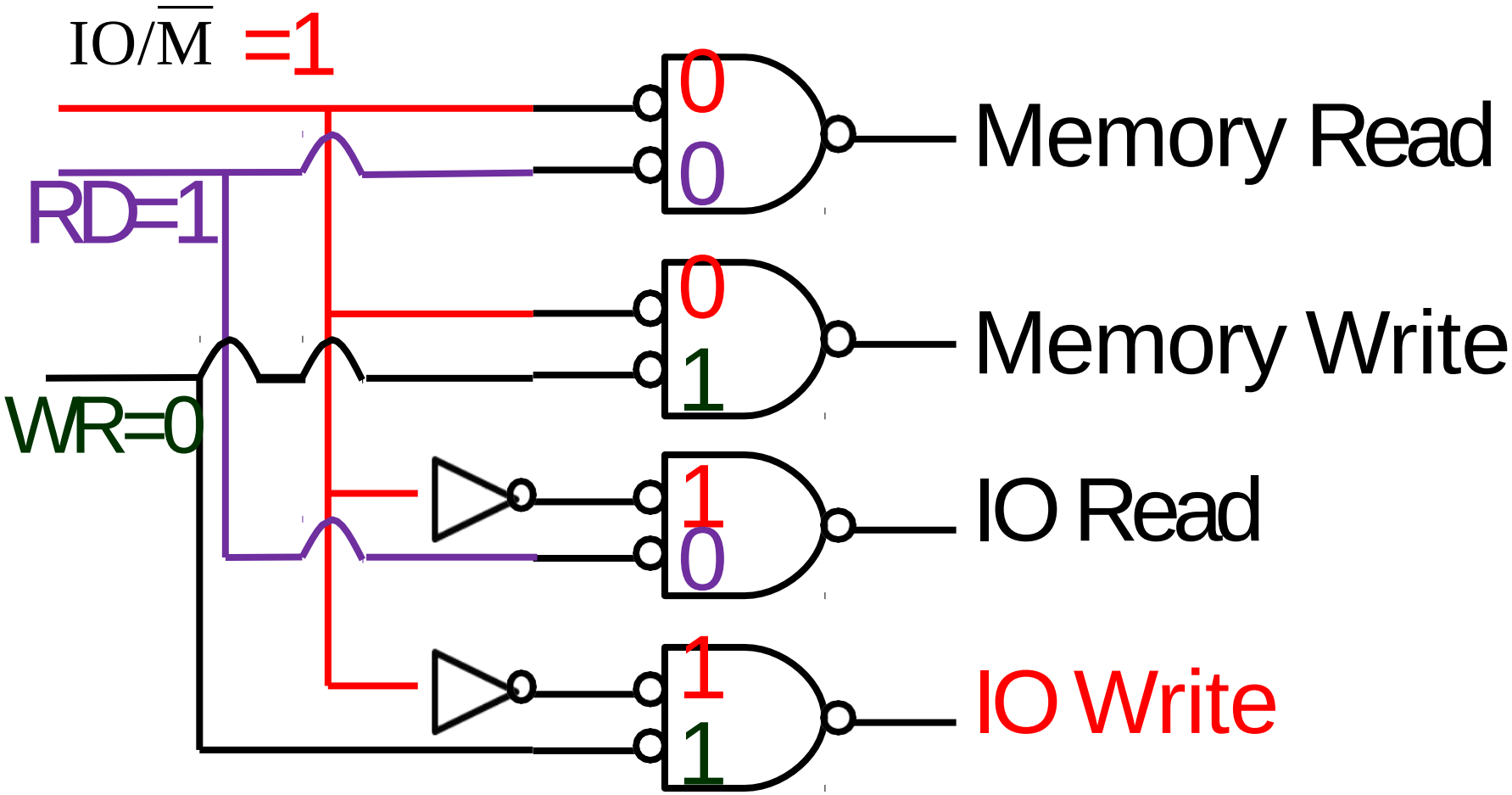
Generating Control Signals



Generating Control Signals



Generating Control Signals



Memory Interface

- The memory is made up of semiconductor material used to store the programs and data. The types of memory is,
  - Primary or main memory
  - Secondary memory

Primary Memory

- RAM and ROM are examples of this type of memory.
- Microprocessor uses it in storing a program temporarily (commonly called loading) and executing a program.
- Hence the speed of this type of memory should be fast.

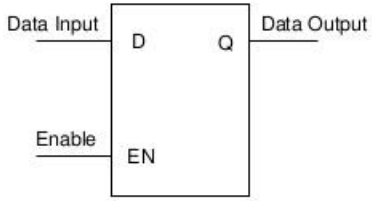
Secondary Memory

- These are used for bulk storage of data and information.
- The main examples include Floppy, Hard Disk, CD-ROM, Magnetic Tape etc.
- Slower and Sequential Access Nature.
- Non-volatile nature.

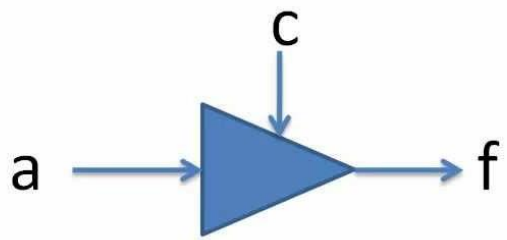
Memory element

The Basic Memory Element

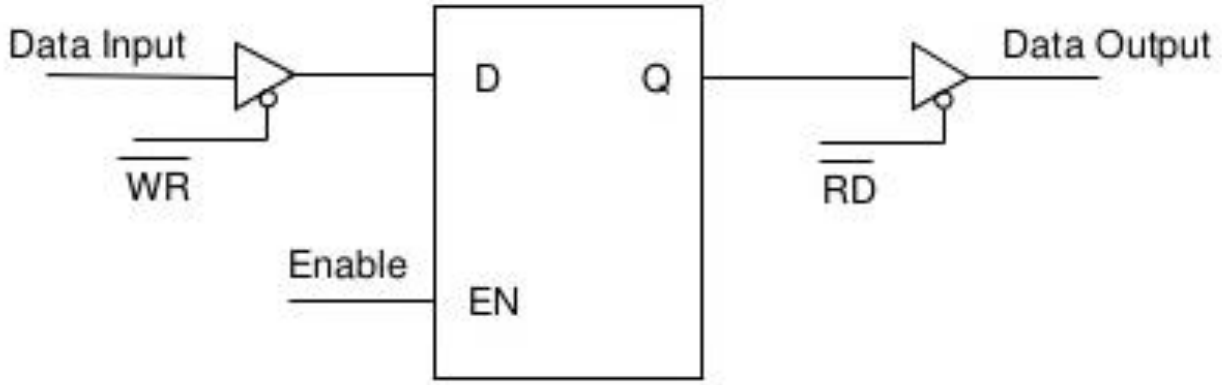
- The basic memory element is similar to a D latch.
- This latch has an input where the data comes in. It has an enable input and an output on which data comes out.

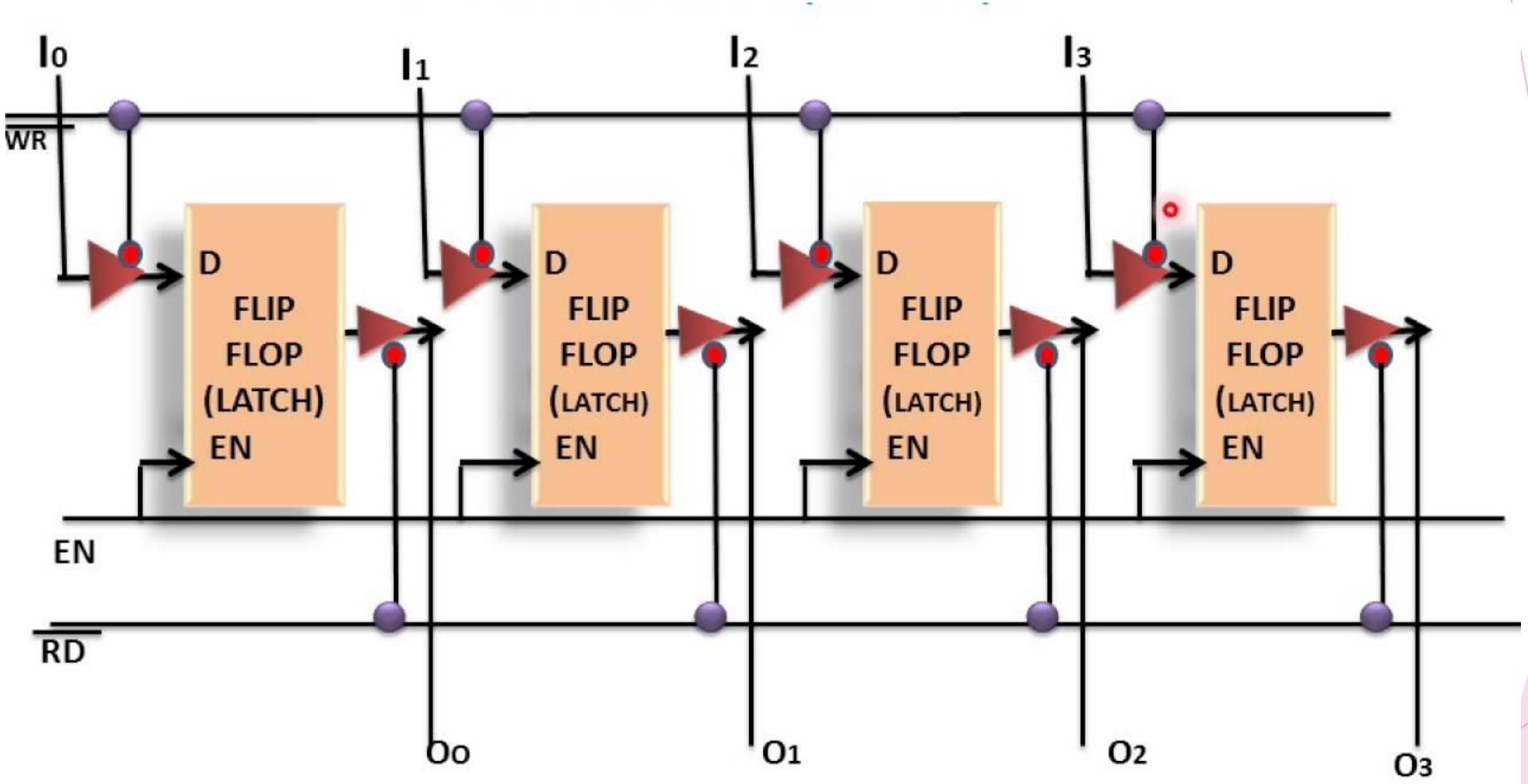


Tri-State Buffer



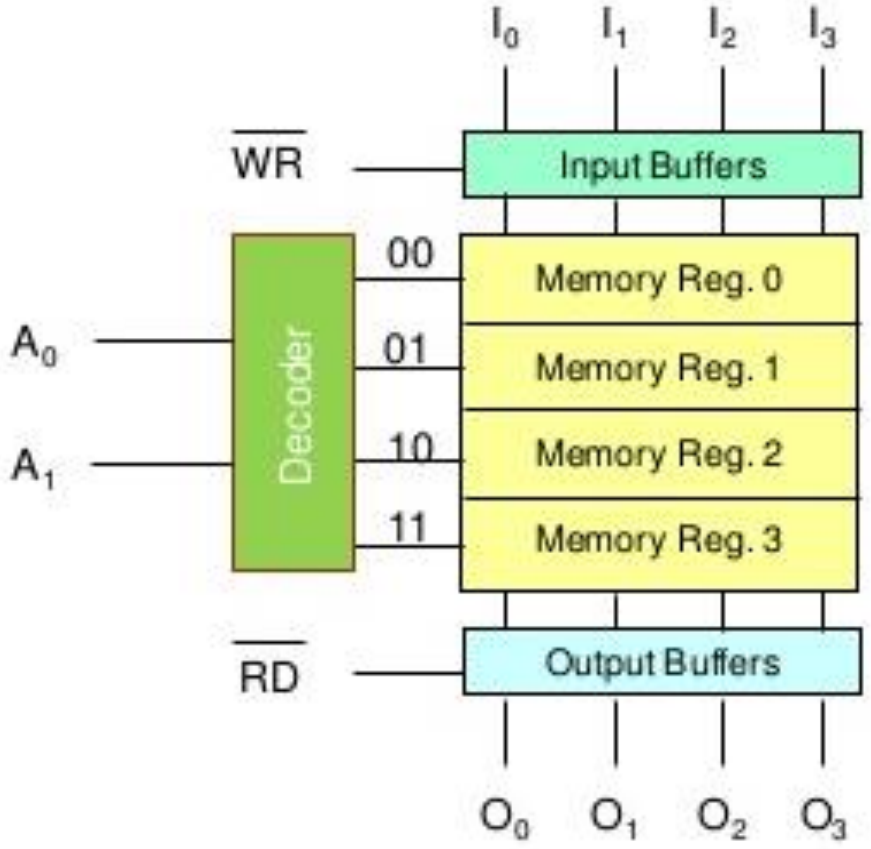
c	a	f
0	0	Z
0	1	Z
1	0	0
1	1	1




I0
I1
I2
I3
WR
D
FLIP
FLOP
(LATCH)
EN
D
FLIP
FLOP
(LATCH)
EN
D
FLIP
FLOP
(LATCH)
EN
D
FLIP
FLOP
(LATCH)
EN
EN
RD
Oo
O1
O2
O3

A group of Memory Registers

- If we represent each memory location (Register) as a block we get the following

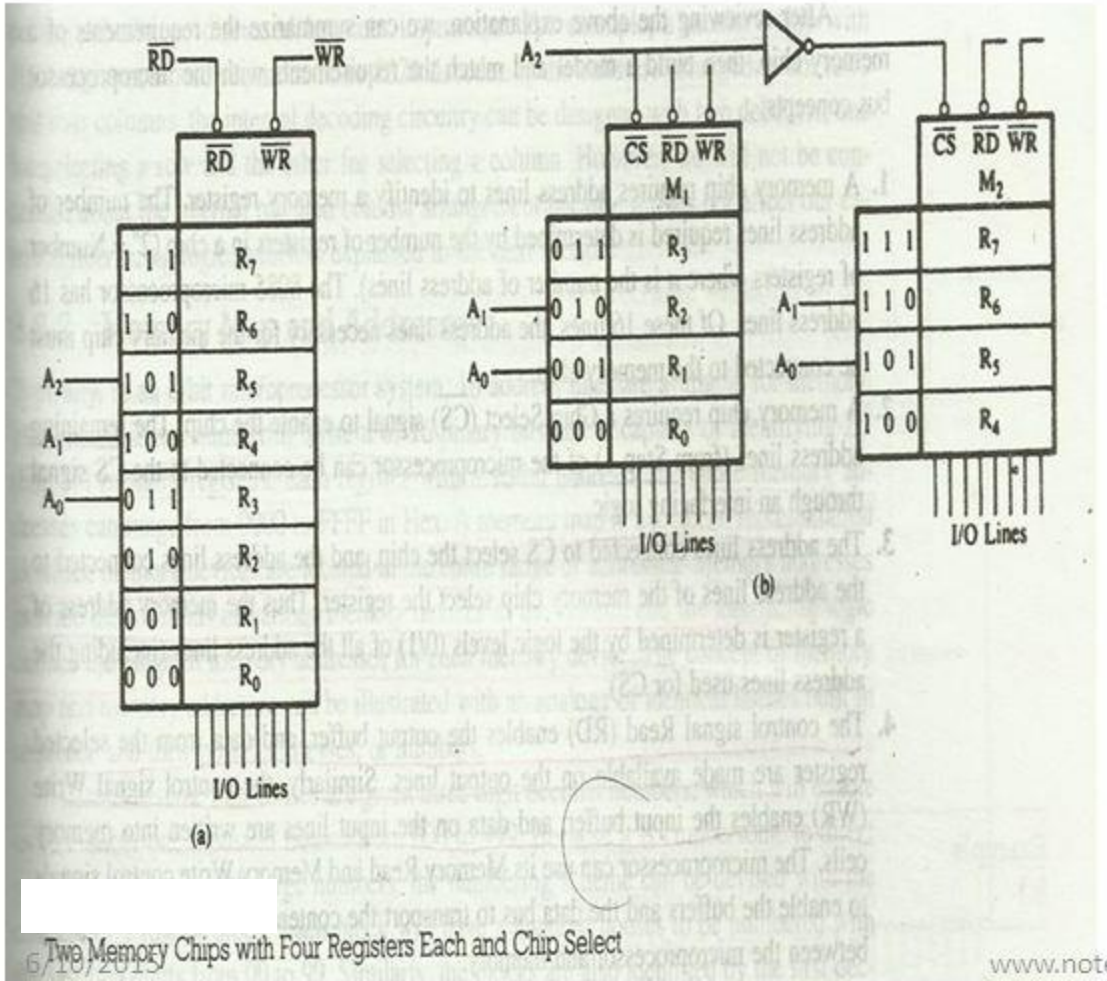


INPUT (ADDRESS LINES) A_0	A_1	OUTPUT (REG. SELECT)
0	0	REG . 0
0	1	REG . 1
1	0	REG . 2
1	1	REG . 3

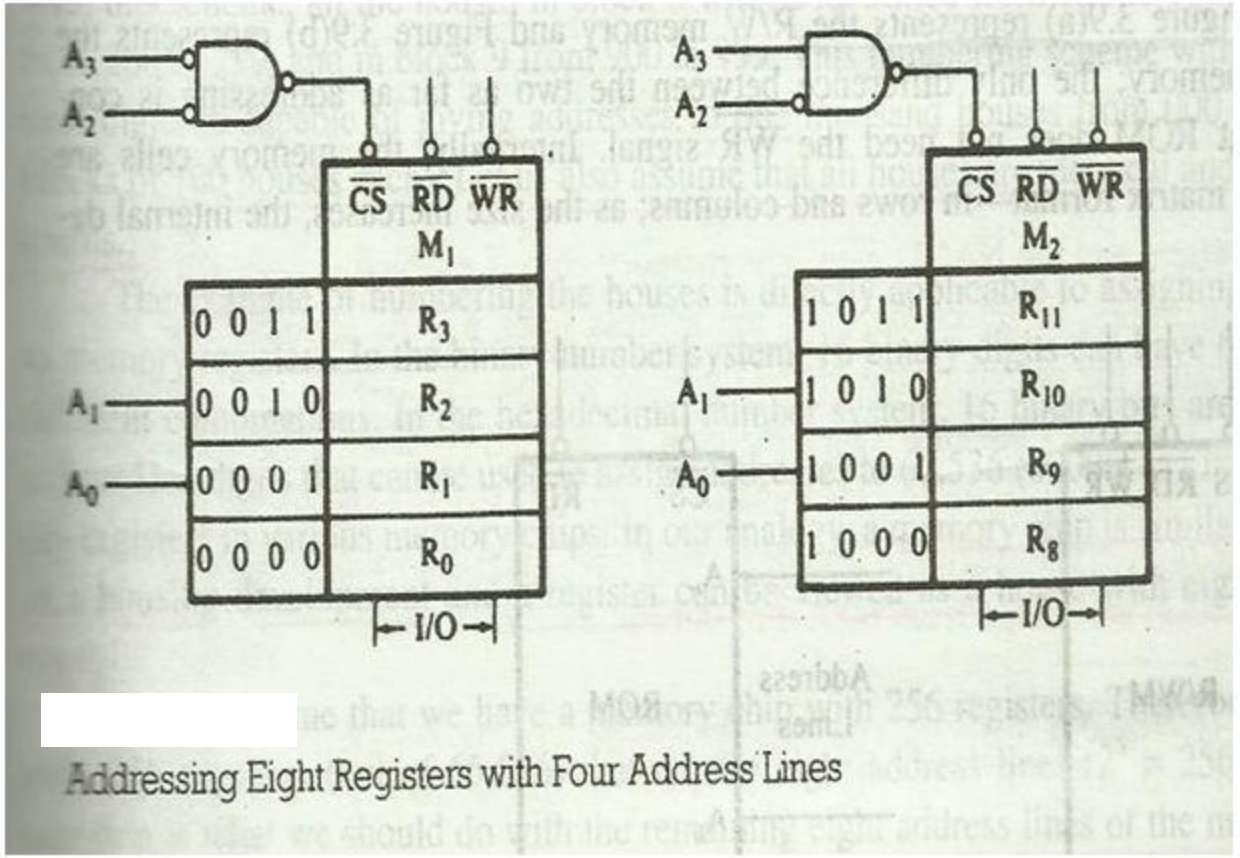
4 X 4 MEMORY :- It indicates, there are 4 register location each of 4-bits.....i.e. 4 X 4.

Expanding Memory

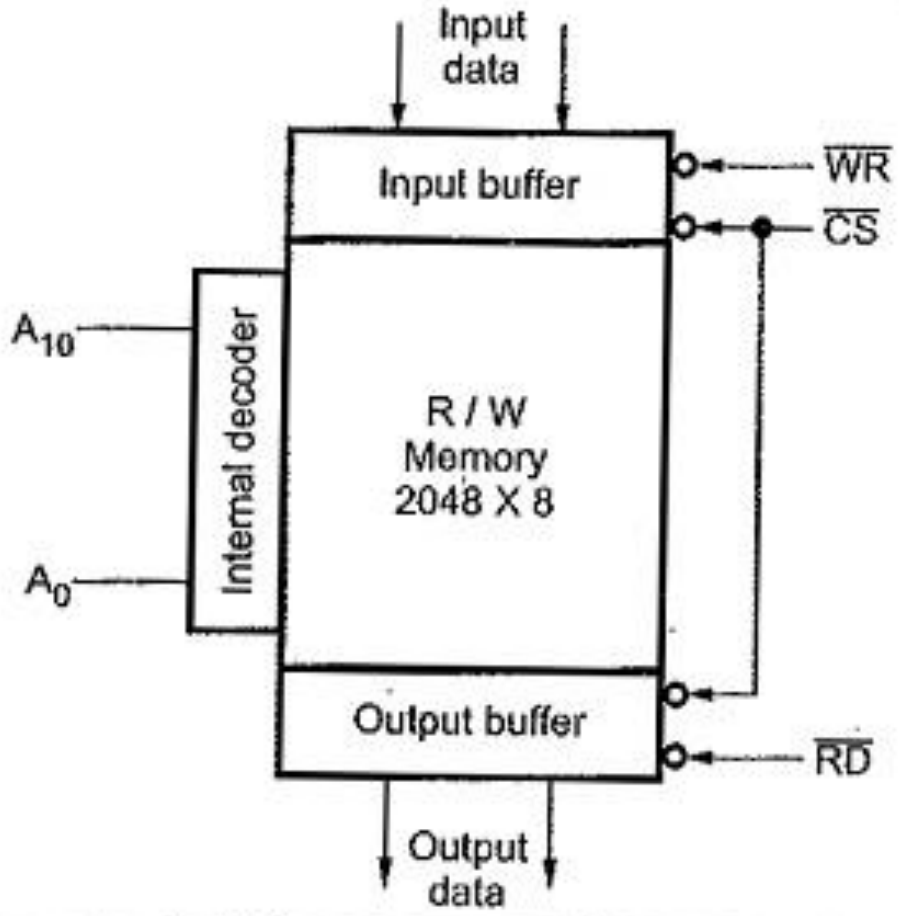
The concept of chip select signal gives us more flexibility in designing chips and allows us to expand memory size by using multiple chips.



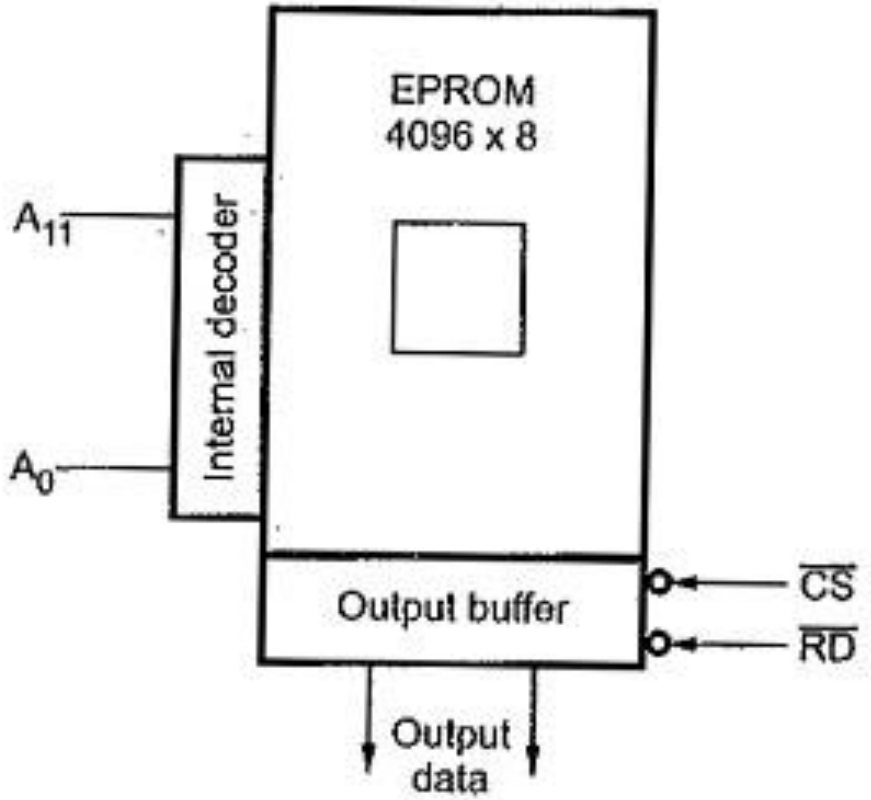
Two Memory Chips with Four Registers Each and Chip Select



Addressing Eight Registers with Four Address Lines

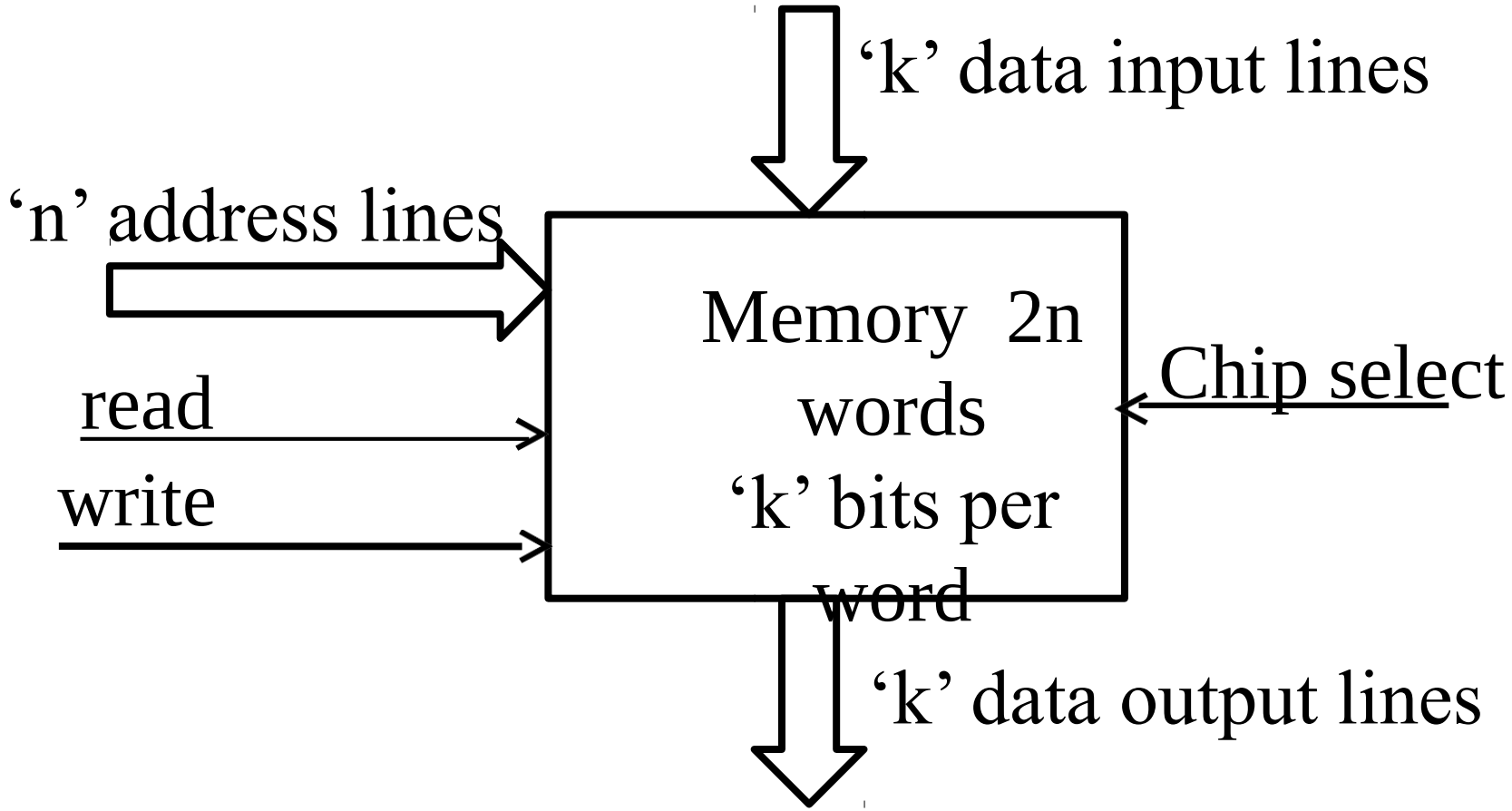


(a) Logic diagram for RAM

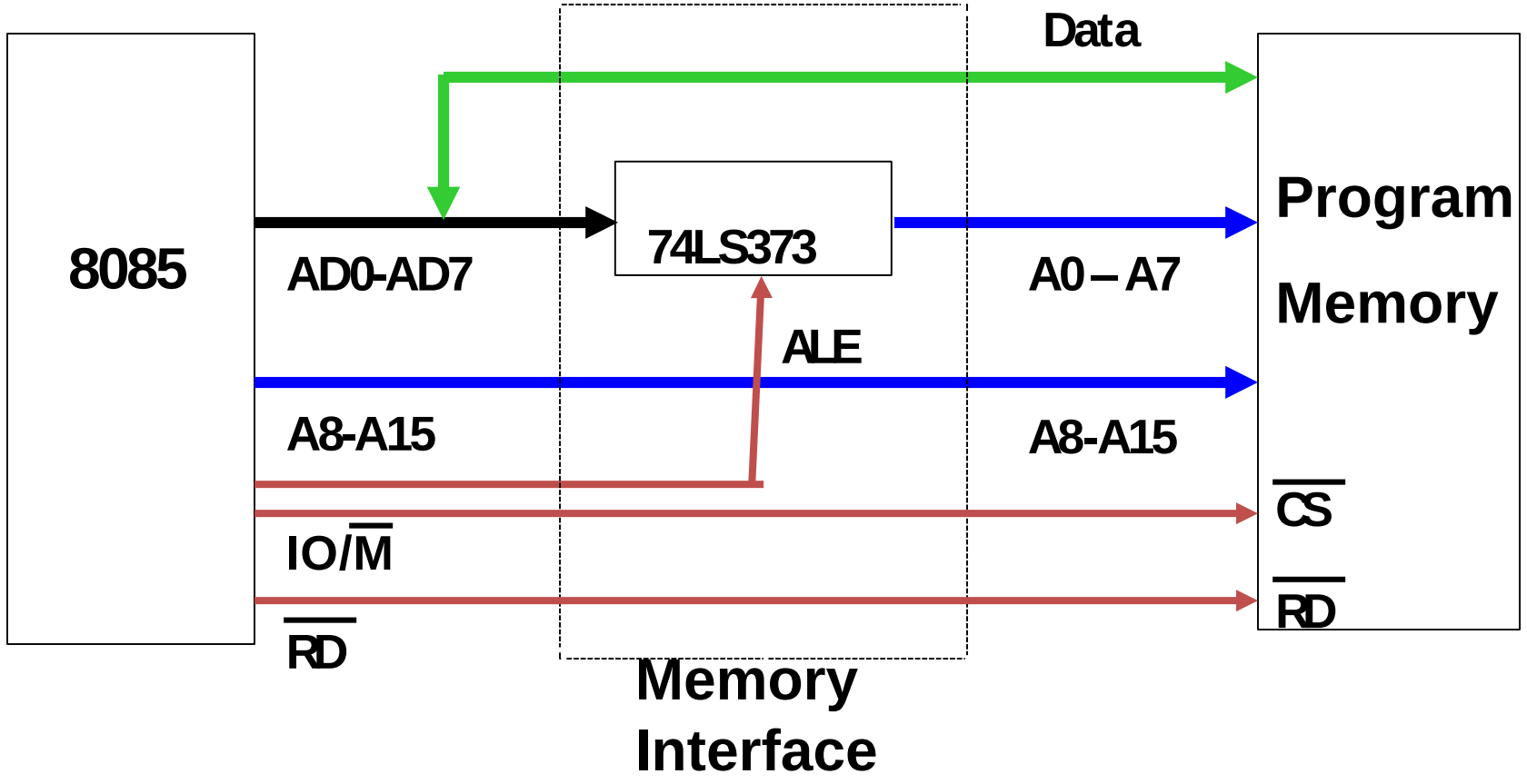


(b) Logic diagram for EPROM

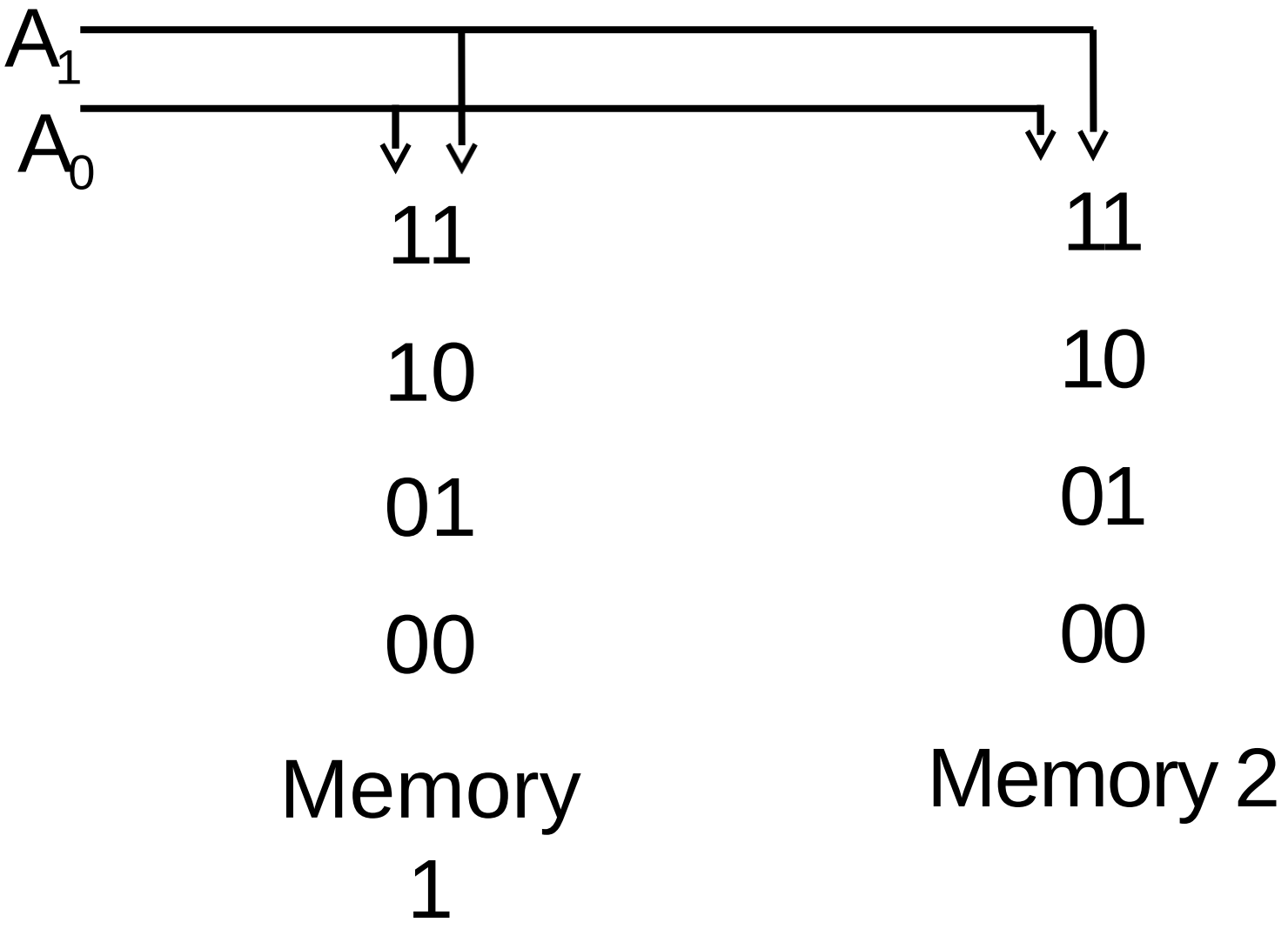
Memory Chip


‘k’ data input lines
‘n’ address lines
Memory 2n words
‘k’ bits per word
read
write
Chip select
‘k’ data output lines

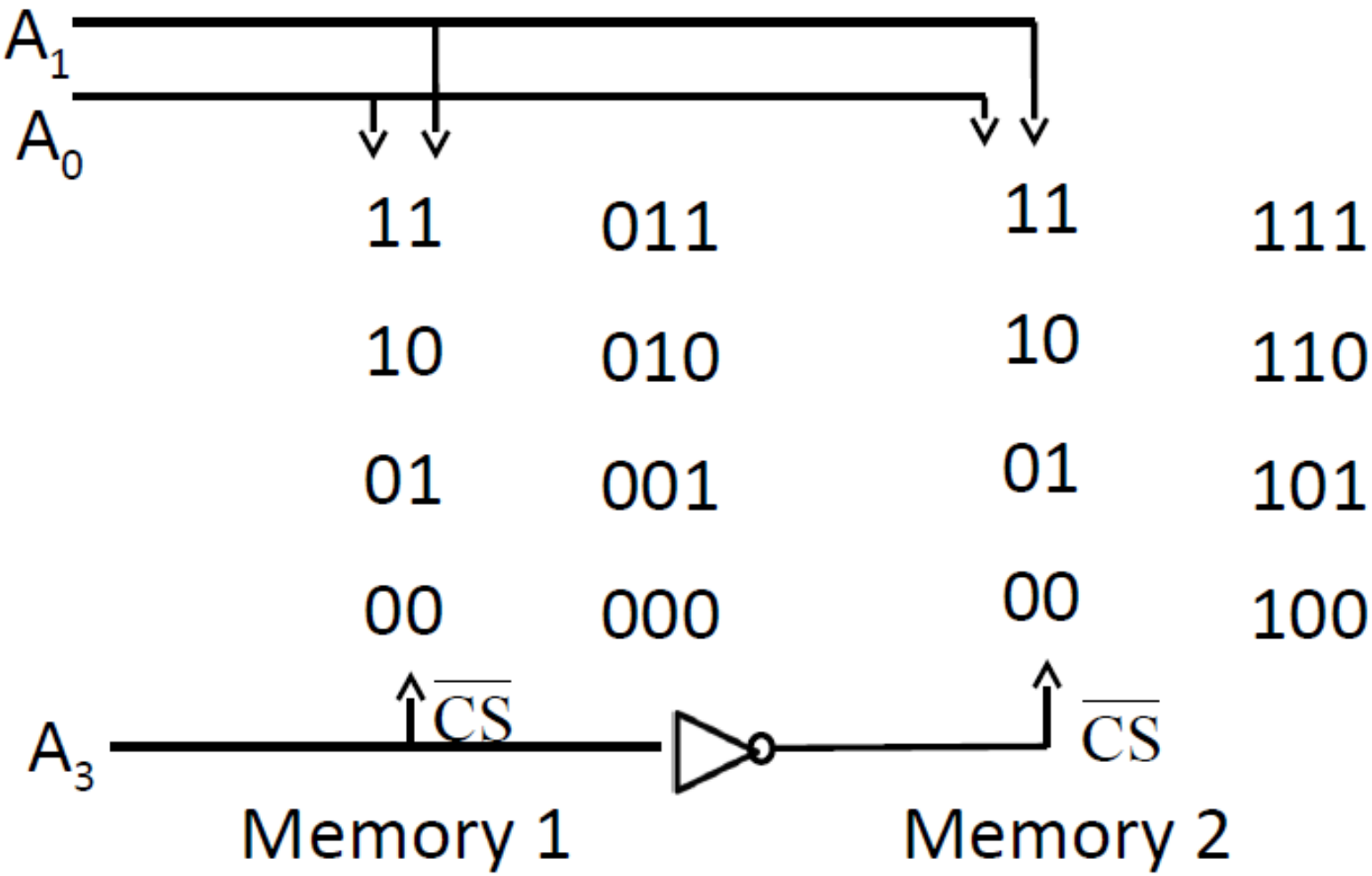
8085 Interfacing with Memory chips



Interface with two memorychips



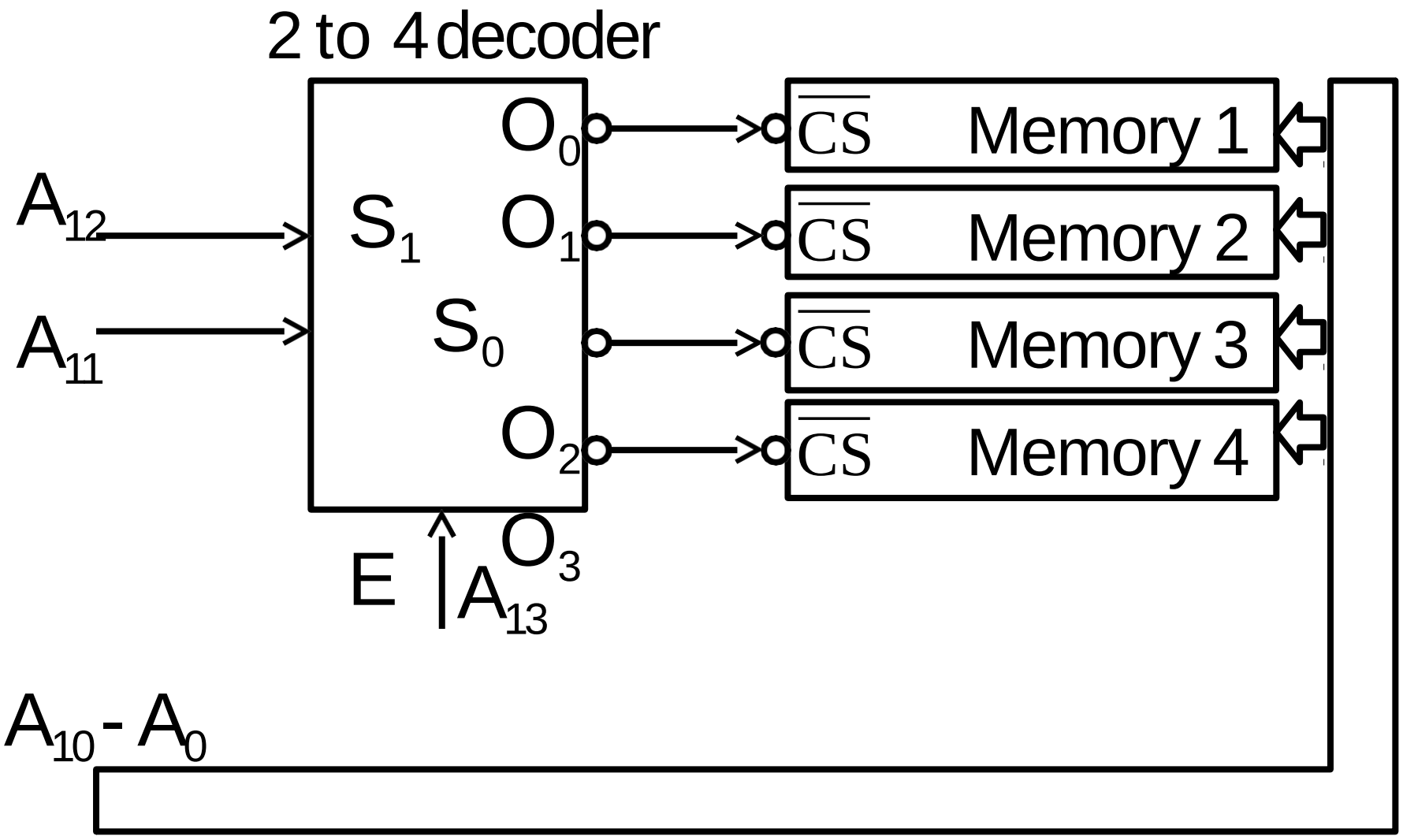
Interface with two memory chips



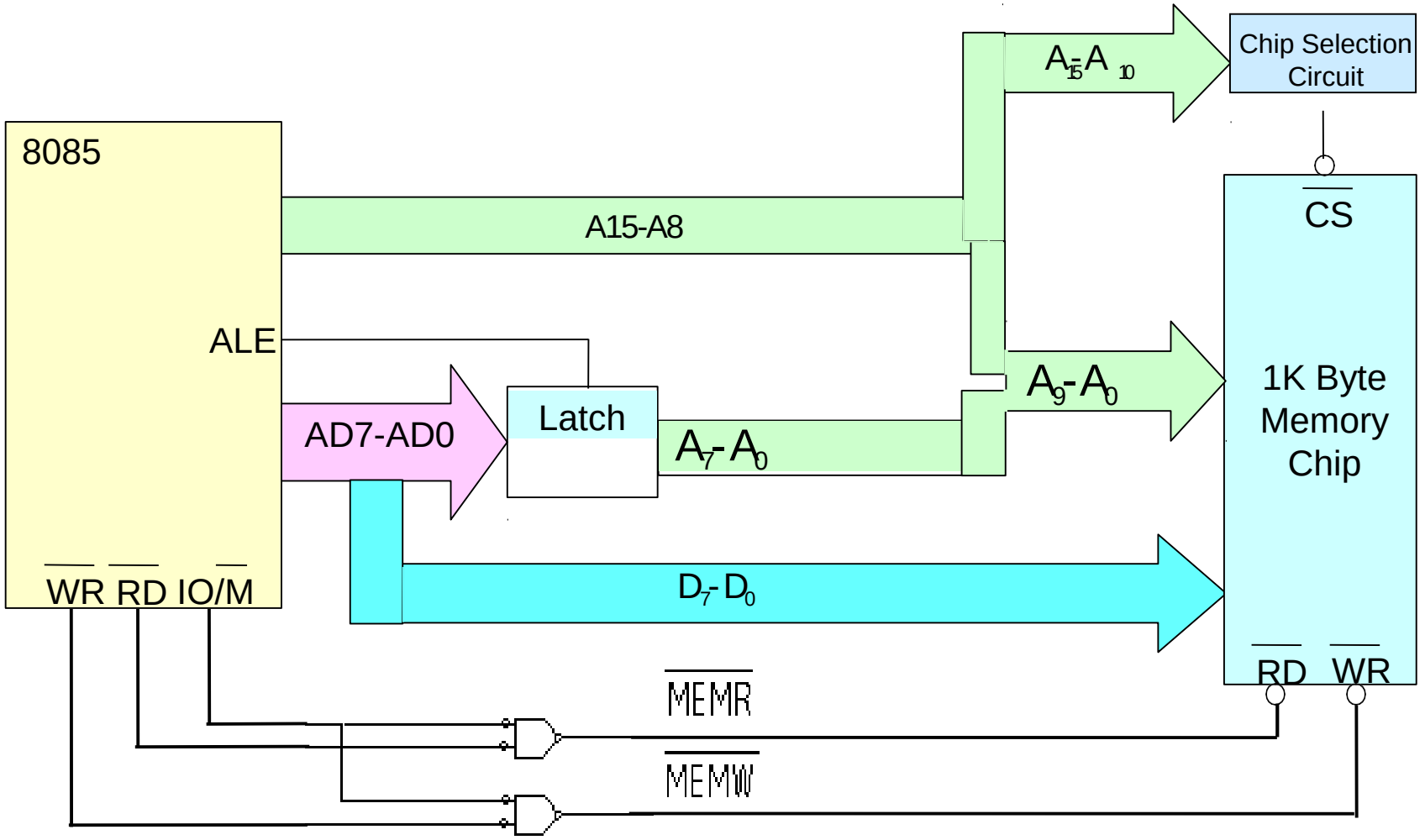
Interface with Multiple Chips

- In case of multiple chips simple circuit like NOT gate will not work.
- In this case normally decoder circuits like 3-to-8 decoder circuit 74LS138 are used.
- These circuit are called address decoders.

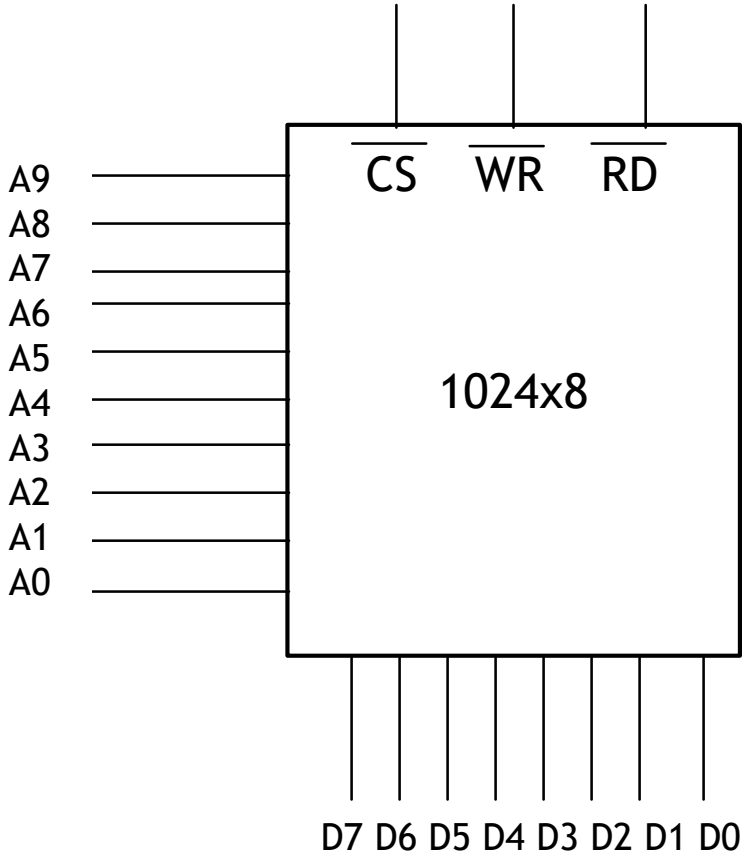
Address decoders

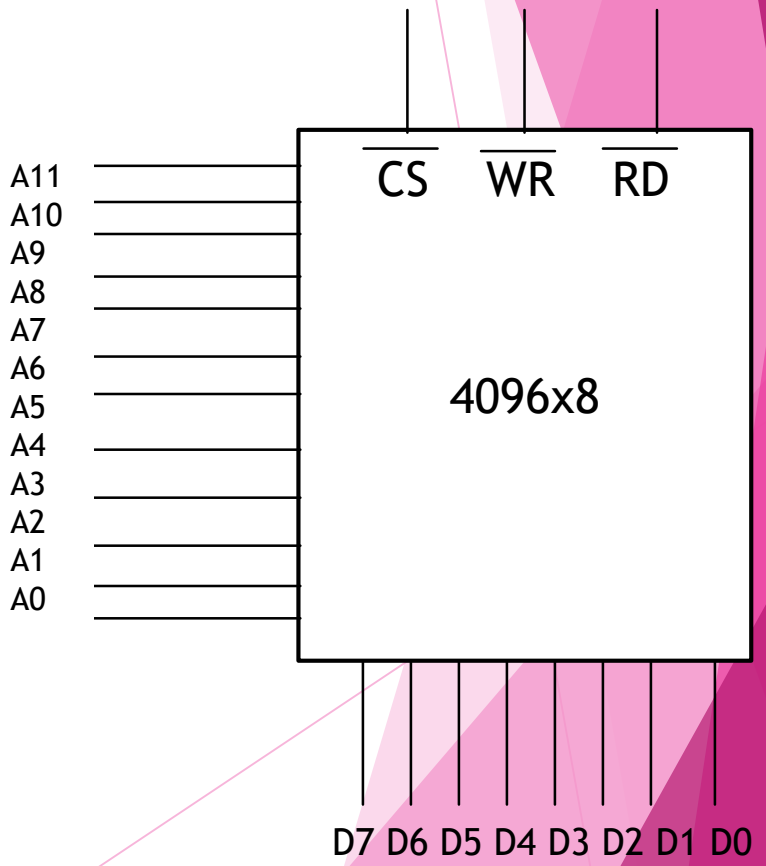


The Overall Picture

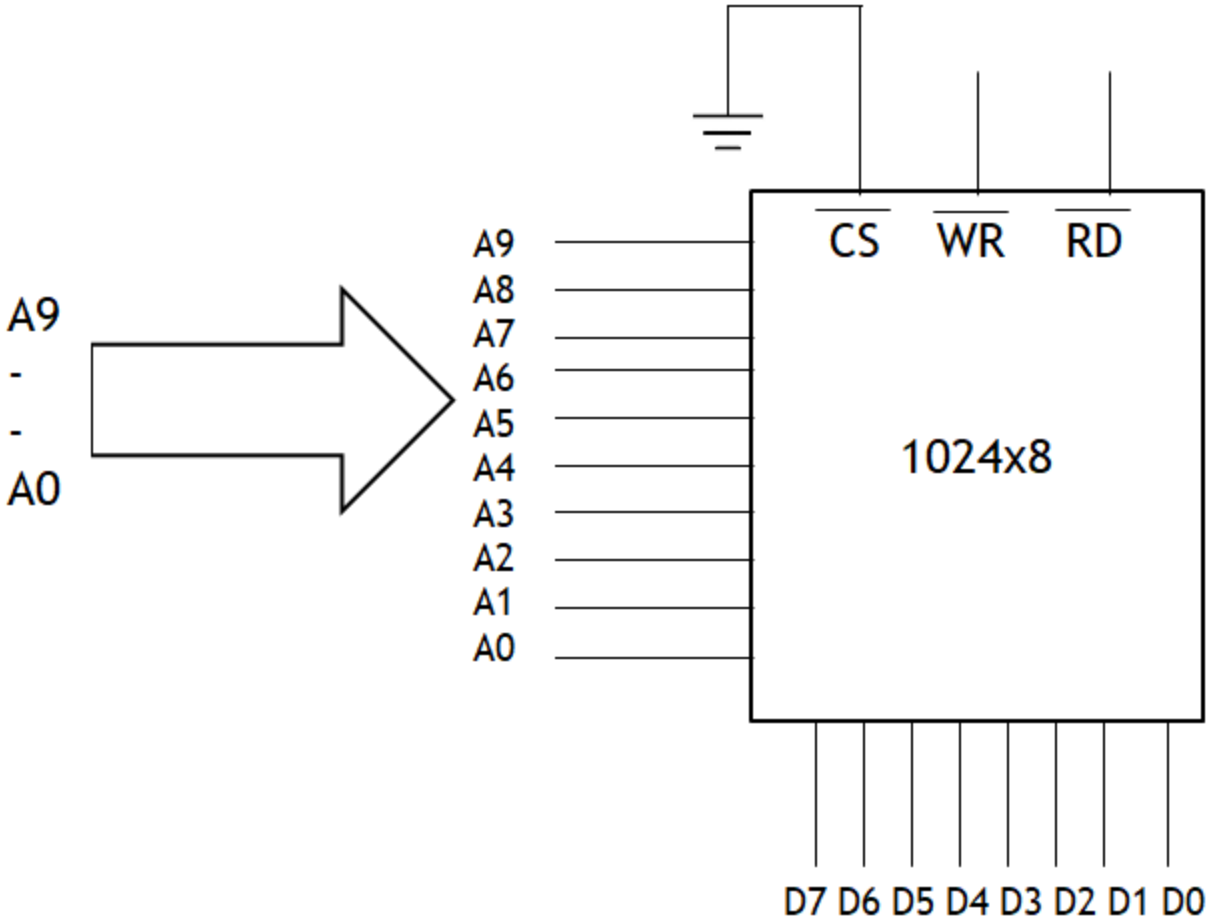


1024x8 MEMORY





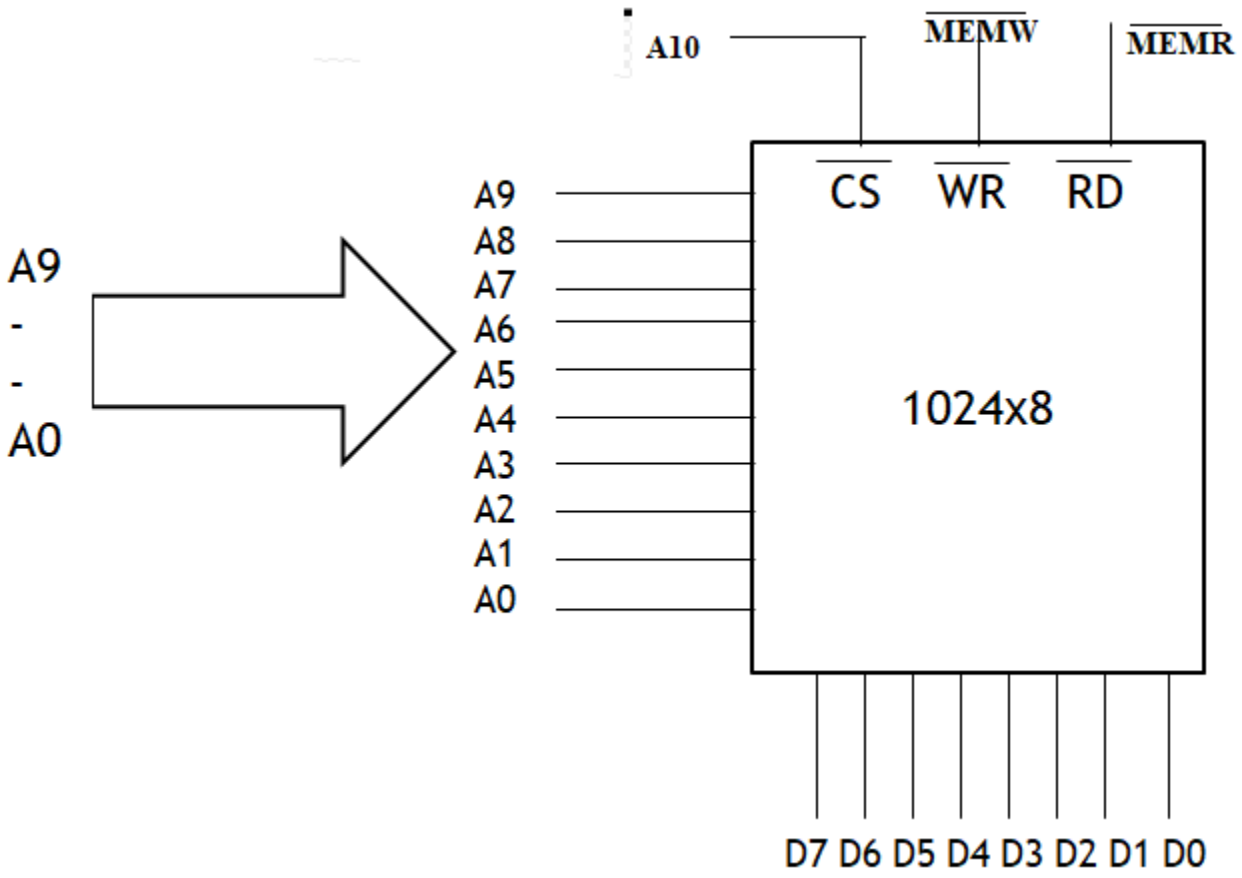
1024x8 MEMORY with 10 address lines



Address range

A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	
0	0	0	0	0	0	0	0	0	0	000
1	1	1	1	1	1	1	1	1	1	3FF

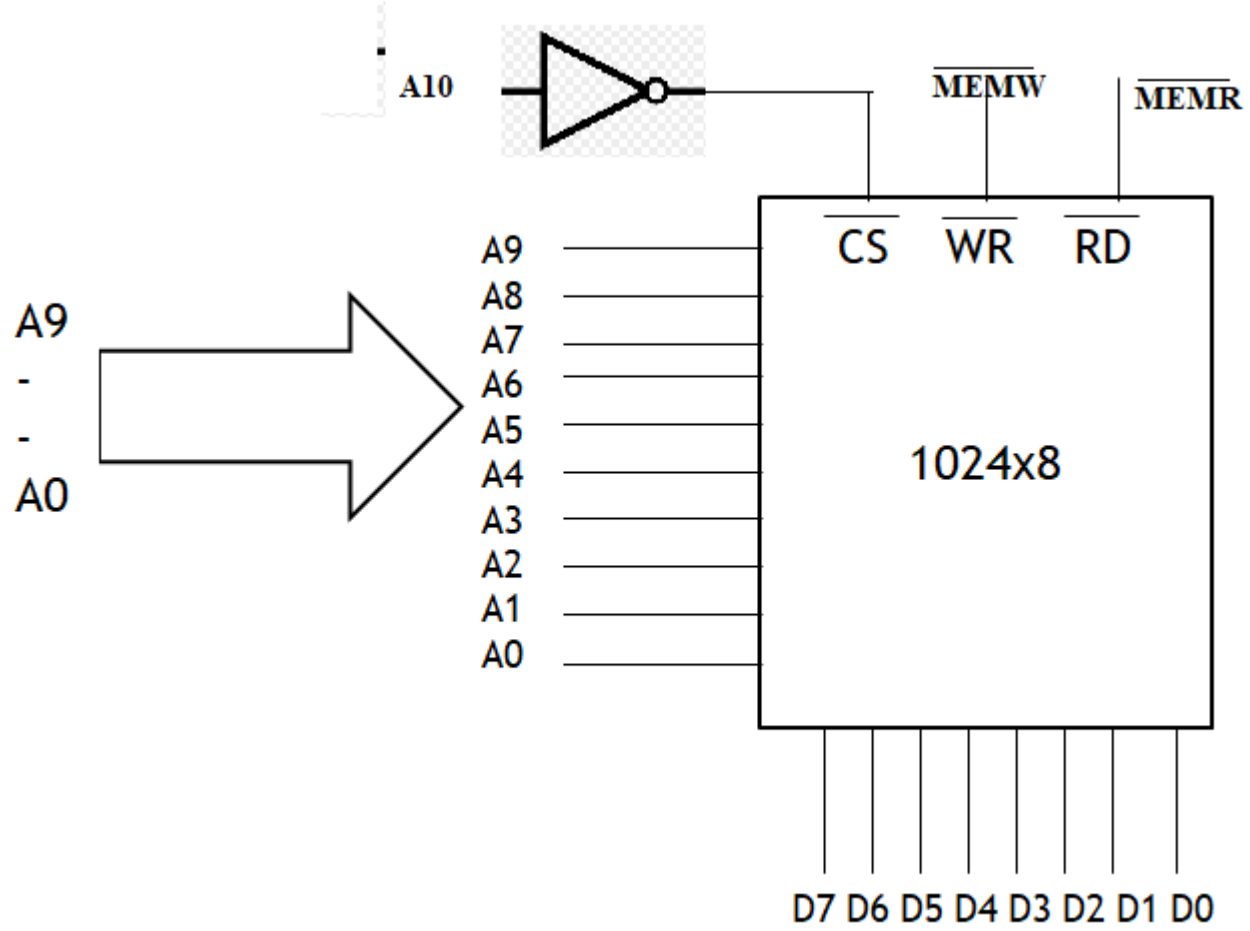
1024x8 MEMORY with 11 address lines



Address range

A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	hexadecim
0	0	0	0	0	0	0	0	0	0	0	000
0	1	1	1	1	1	1	1	1	1	1	3FF

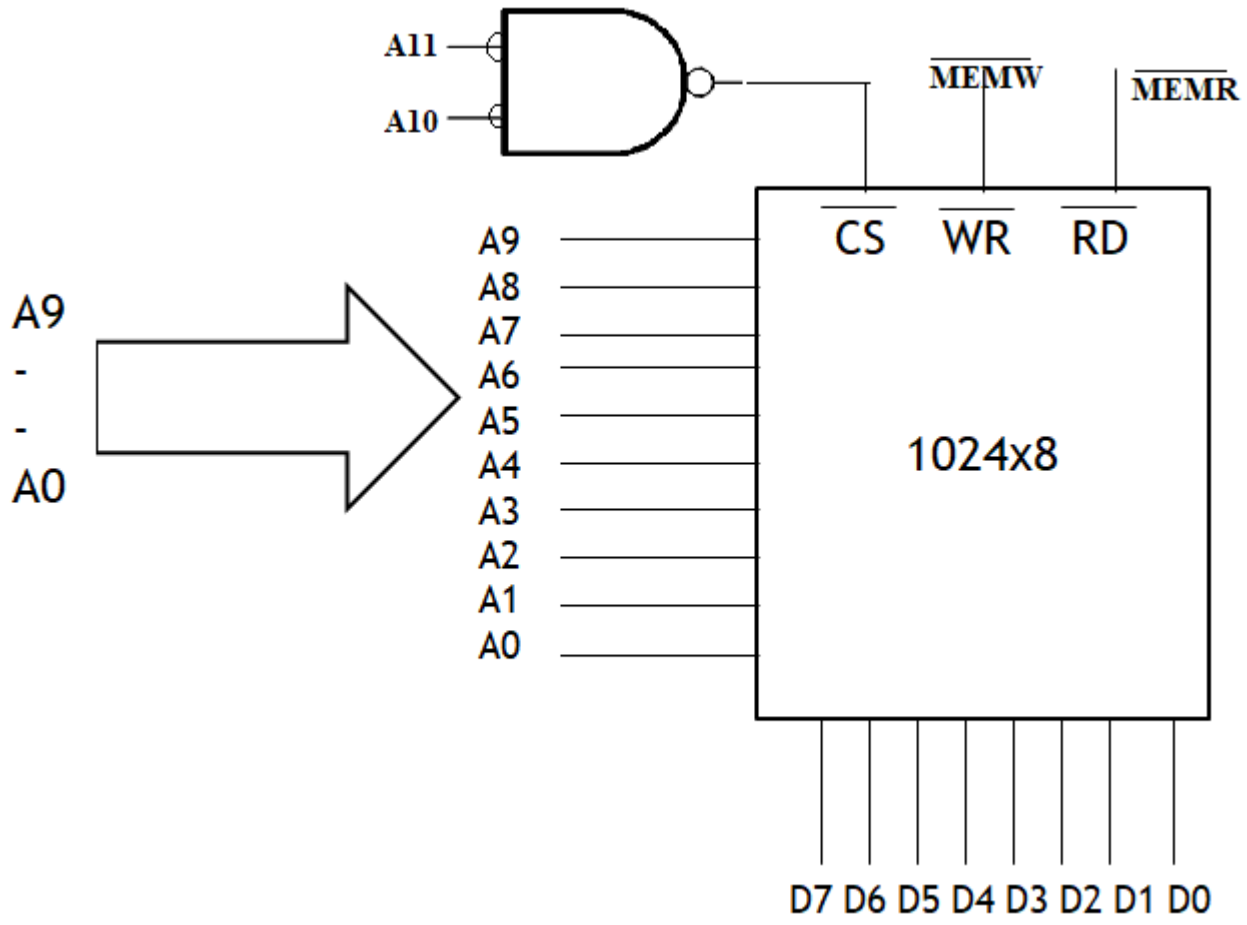
1024x8 MEMORY with 11 address lines



Address range

A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	
1	0	0	0	0	0	0	0	0	0	0	400
1	1	1	1	1	1	1	1	1	1	1	7FF

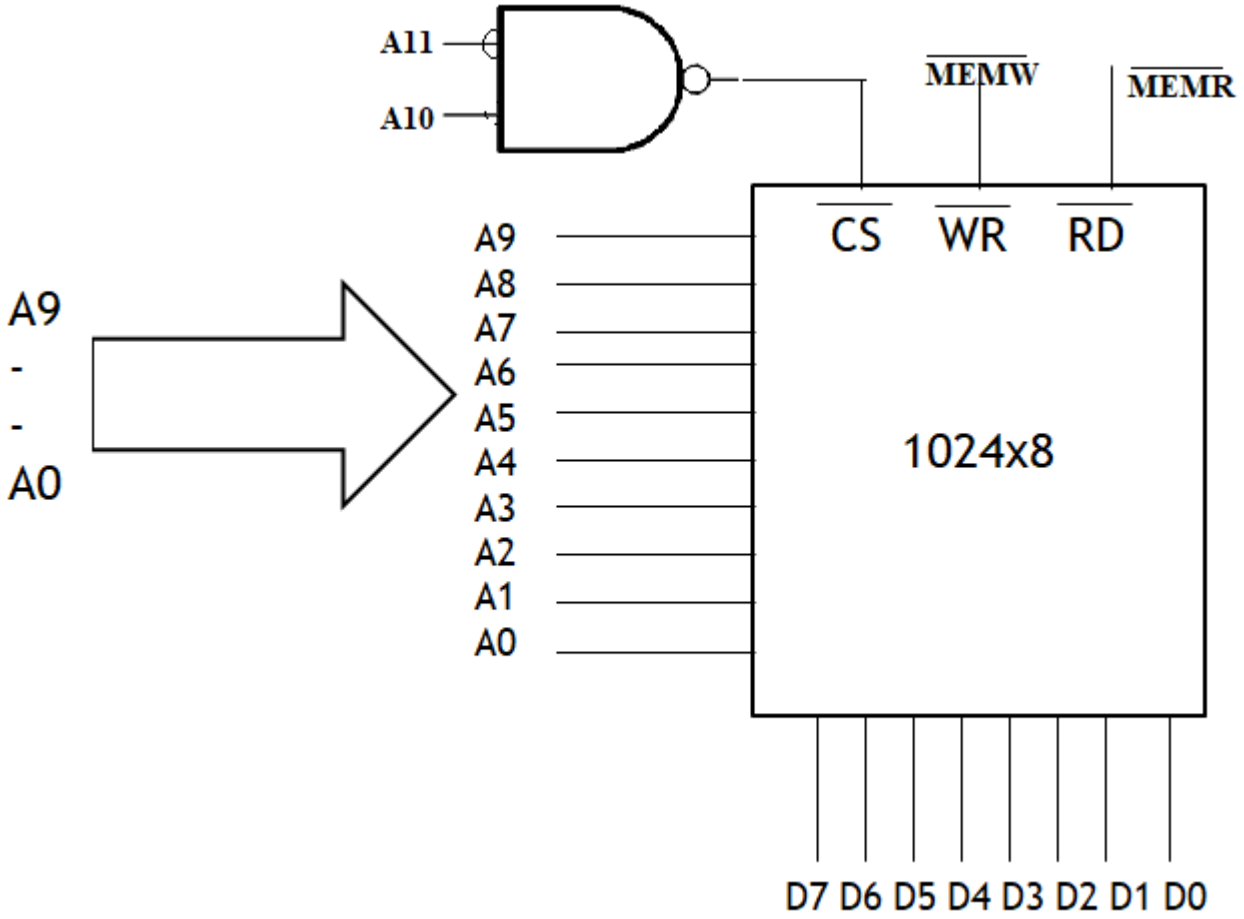
1024x8 MEMORY with 12 address lines



Address range

A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	
0	0	0	0	0	0	0	0	0	0	0	0	000
0	0	1	1	1	1	1	1	1	1	1	1	3FF

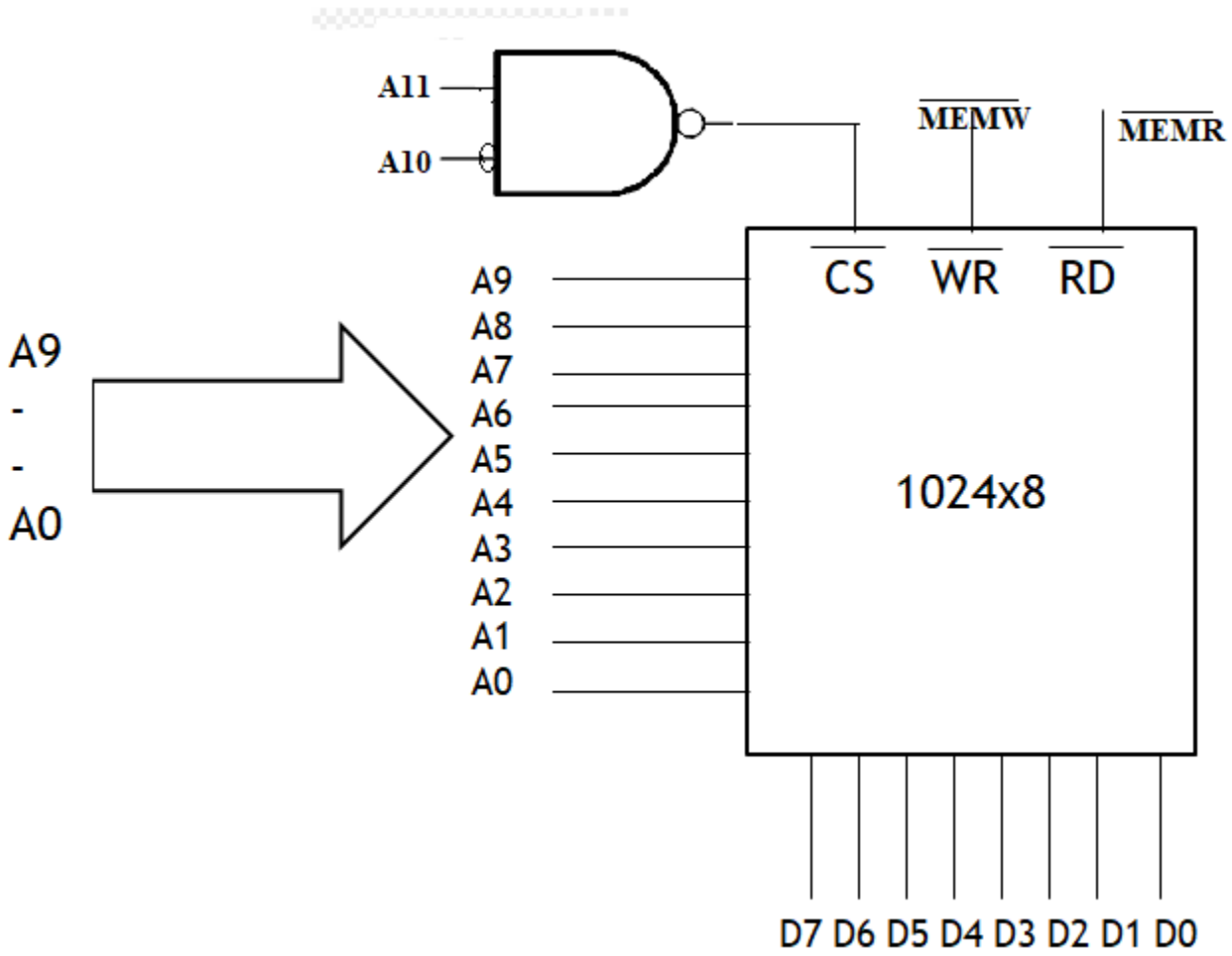
1024x8 MEMORY with 12 address lines



Address range

A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	
0	1	0	0	0	0	0	0	0	0	0	0	400
0	1	1	1	1	1	1	1	1	1	1	1	7FF

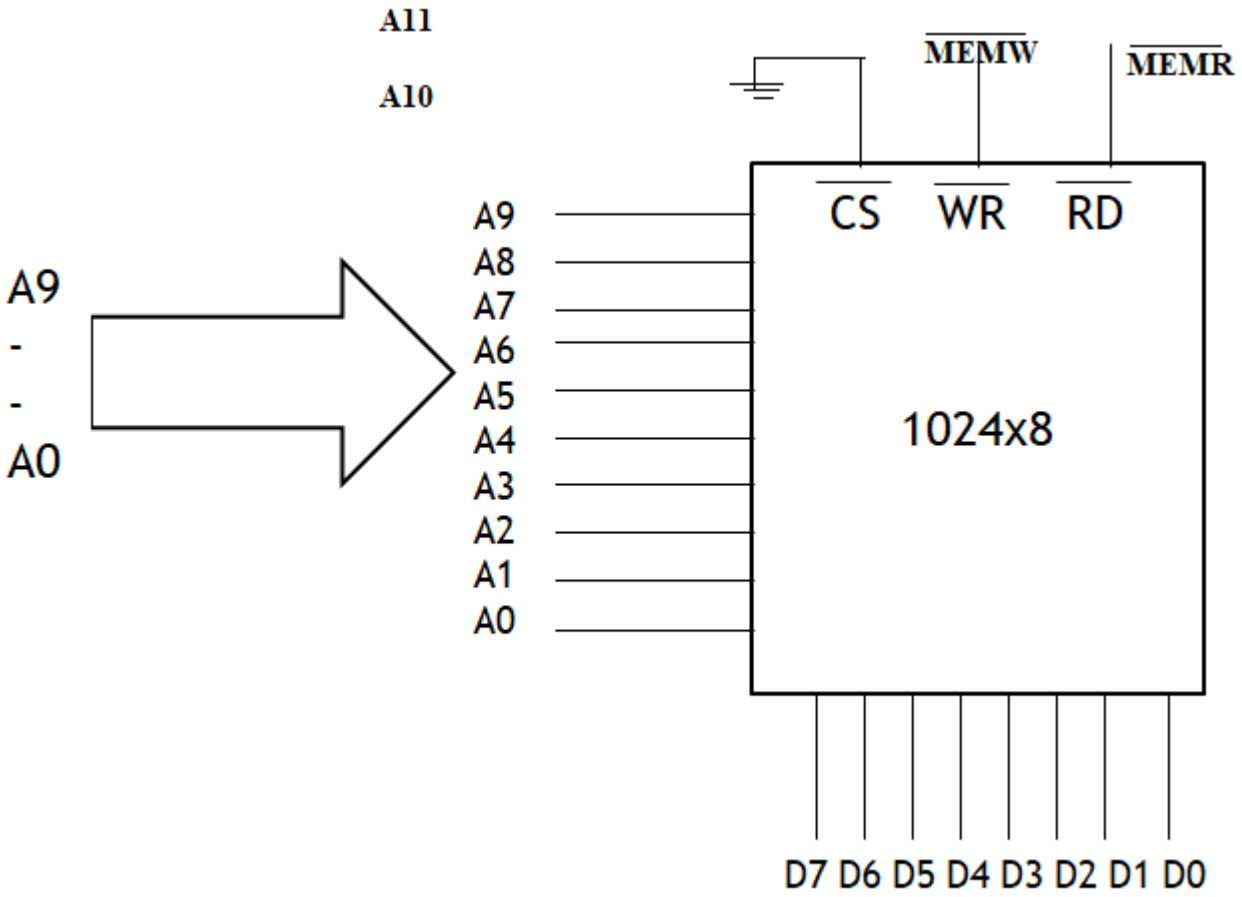
1024x8 MEMORY with 12 address lines



Address range

A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	
1	0	0	0	0	0	0	0	0	0	0	0	800
1	0	1	1	1	1	1	1	1	1	1	1	DFF

1024x8 MEMORY with 12 address lines



Address range

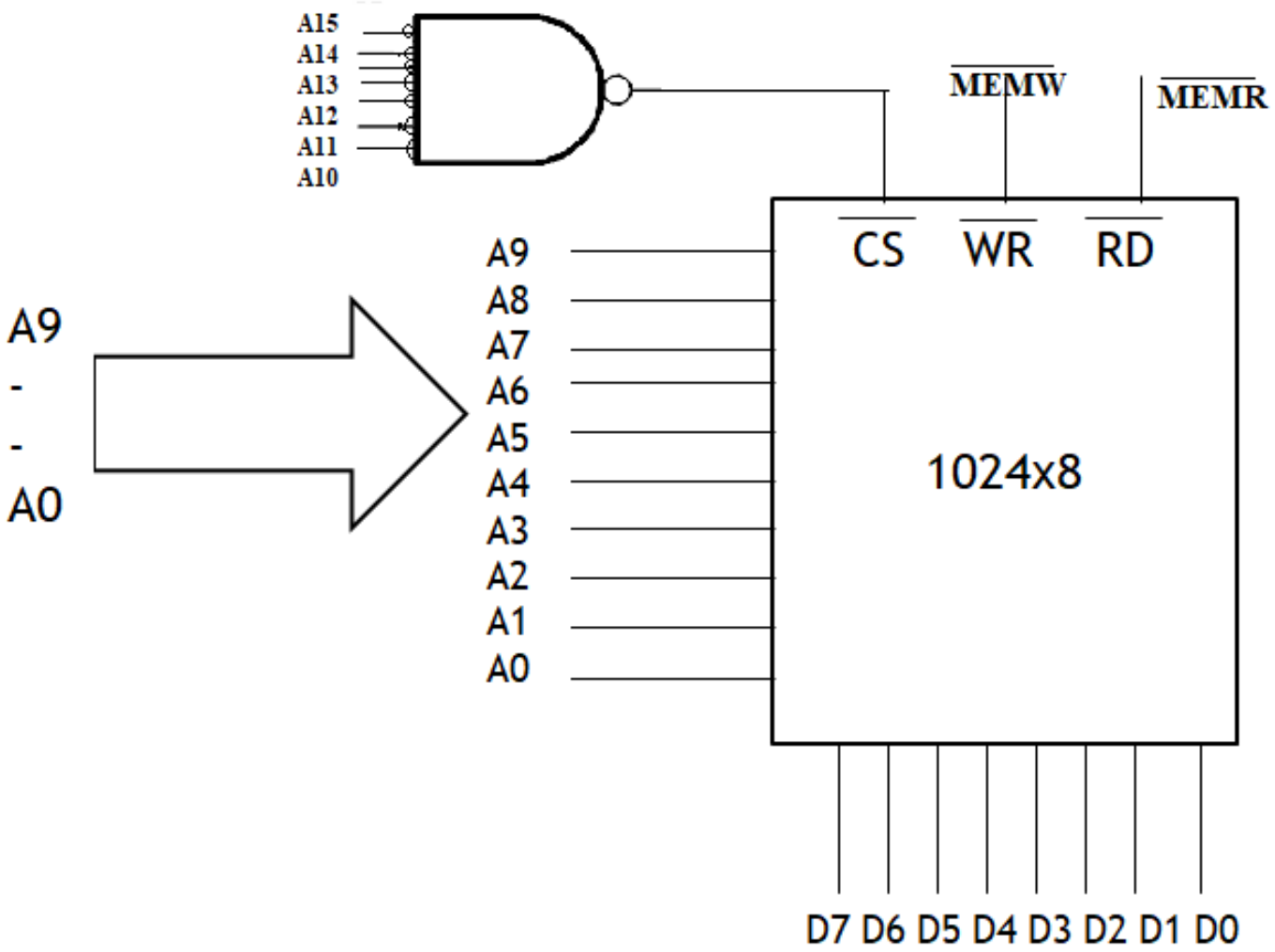
A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	
0	0	0	0	0	0	0	0	0	0	0	0	000
0	0	1	1	1	1	1	1	1	1	1	1	3FF

A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	
0	1	0	0	0	0	0	0	0	0	0	0	400
0	1	1	1	1	1	1	1	1	1	1	1	7FF

A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	
1	0	0	0	0	0	0	0	0	0	0	0	800
1	0	1	1	1	1	1	1	1	1	1	1	DFF

A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	
1	1	0	0	0	0	0	0	0	0	0	0	A00
1	1	1	1	1	1	1	1	1	1	1	1	FFF

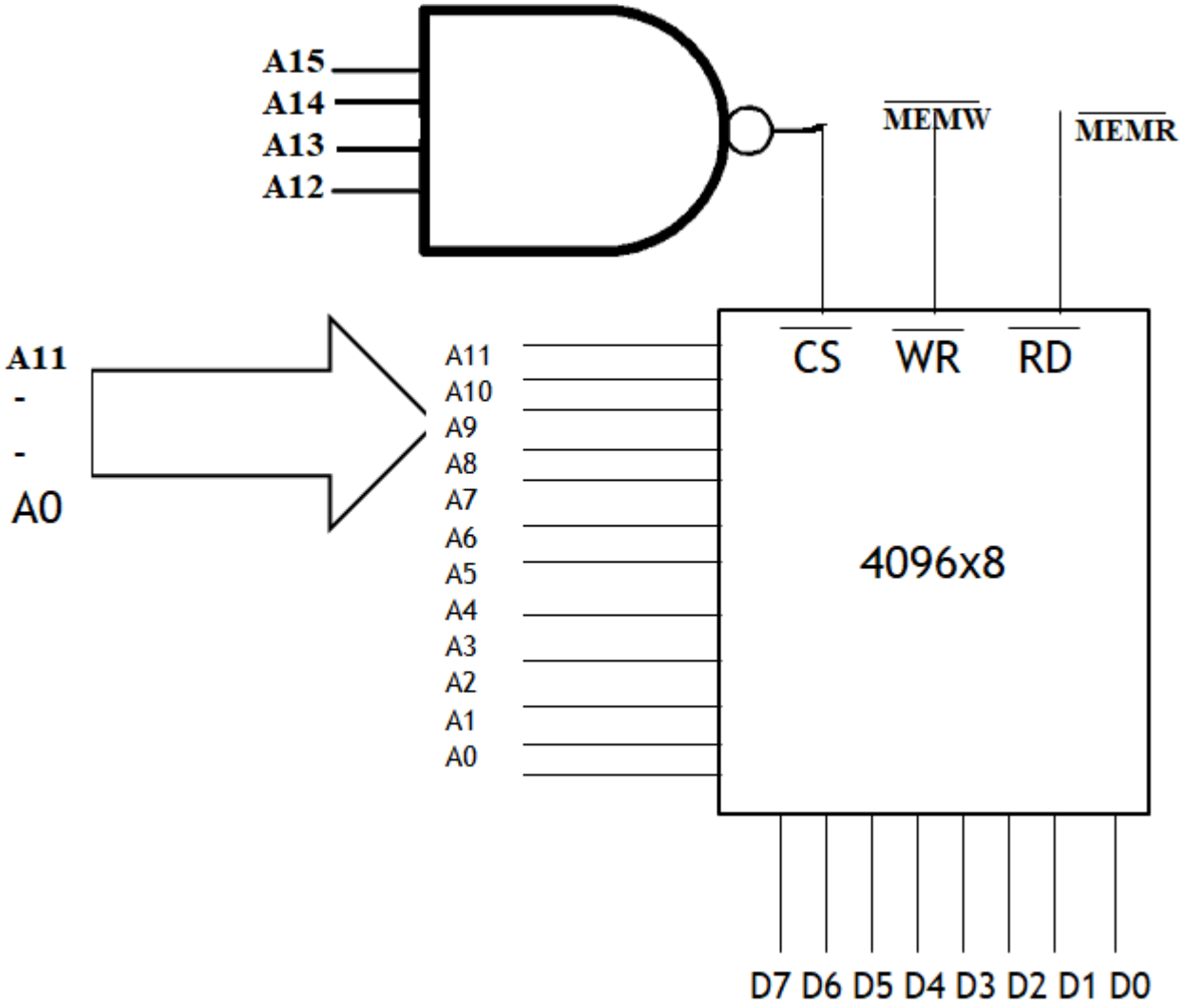
1024x8 MEMORY with 16 address lines



Address range

A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000
0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	03FF

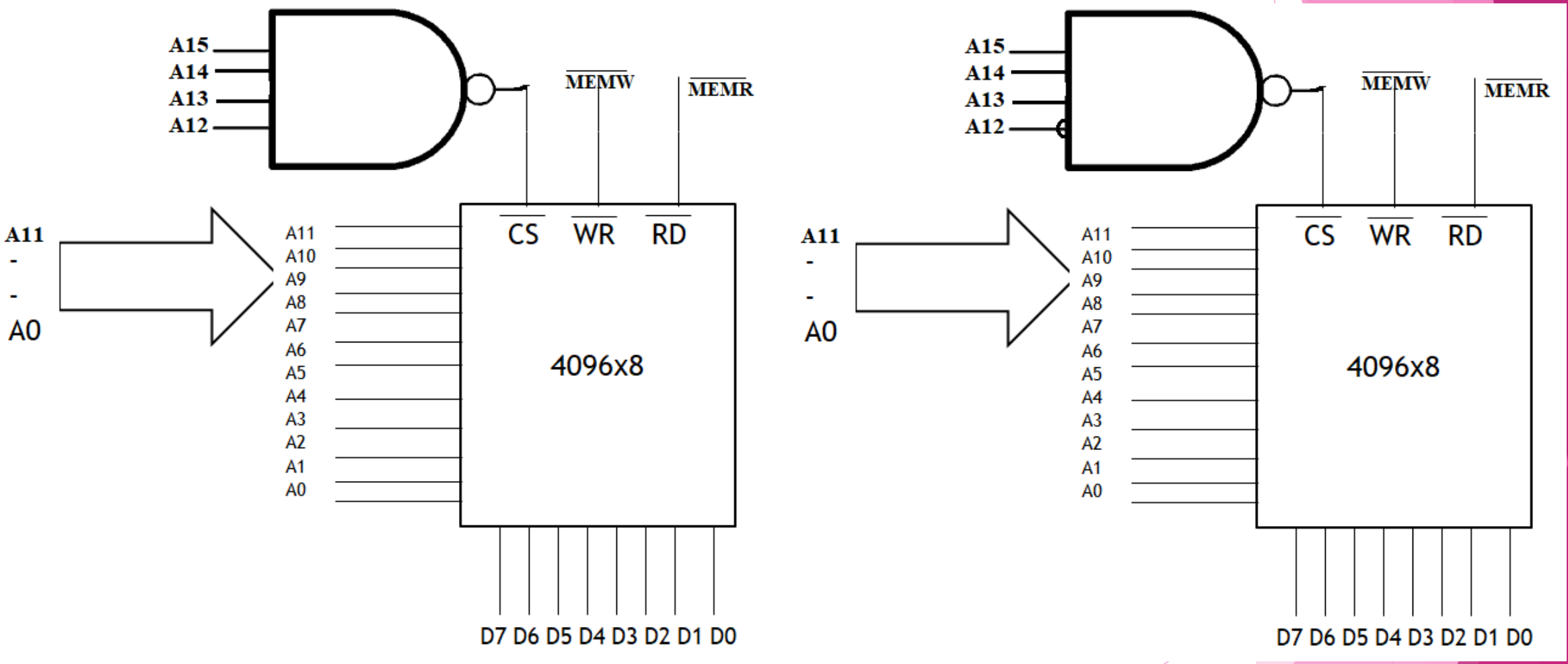
INTERFACING OF 4Kx8 MEMORY WITH 8085



Address range

A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	F000
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	FFFF

INTERFACING TWO OF 4Kx8 MEMORY WITH 8085



Address range

A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	F000
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	FFFF

A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	E000
1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	EFFF

Example 3.1 Illustrate the memory address range of chip with 256 bytes of memory as shown in figure 3.10 a and explain how the range can be changed by modifying the hardware of the Chip Select CS(active low) line shown in

Example 3.1 **Illustrate the memory address range of chip with 256 bytes of memory as shown in figure 3.10 a and explain how the range can be changed by modifying the hardware of the Chip Select CS(active low) line shown in figure 3.10 b**

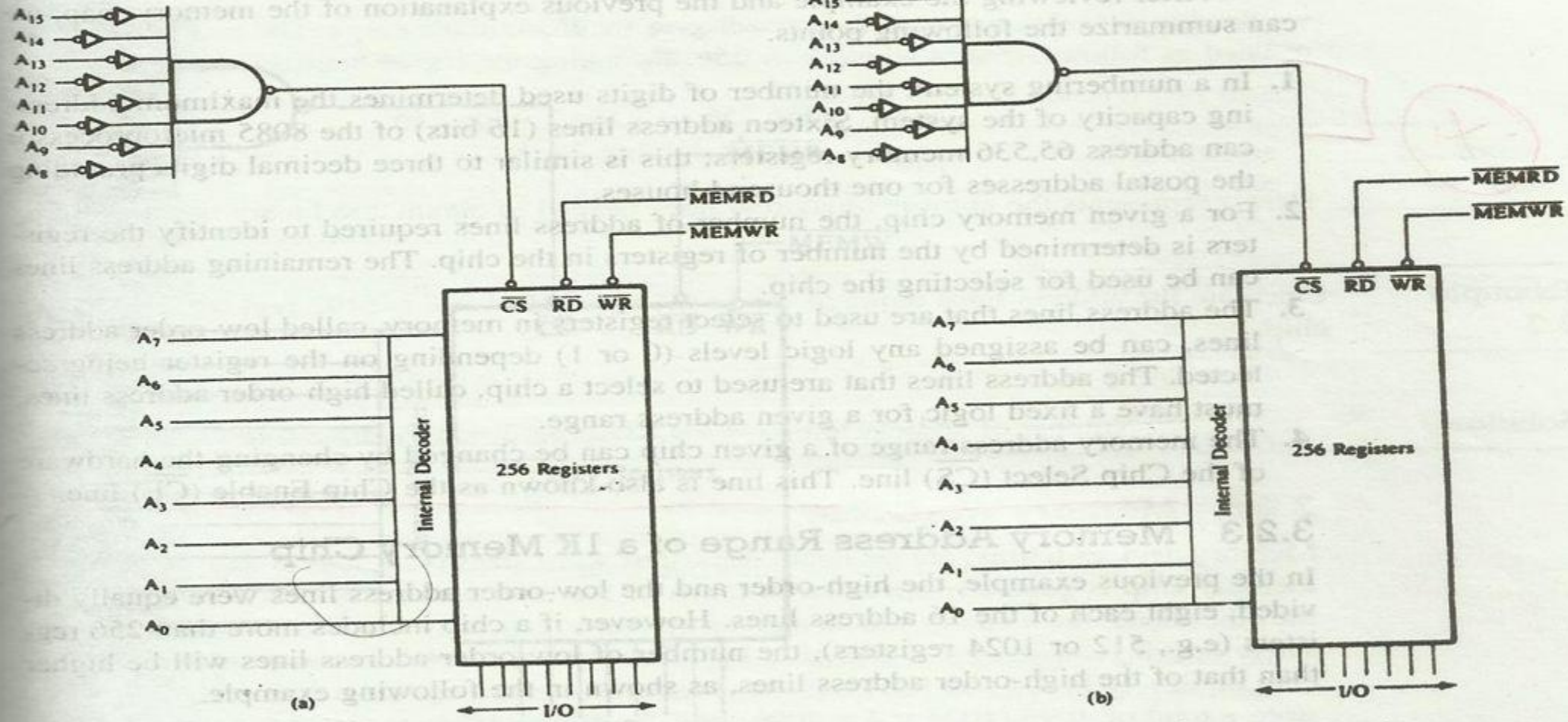


FIGURE 3.10
Memory Maps: 256 Bytes of Memory

Example 3.2. Explain the memory address range of 1K (1024 X 8) memory shown in fig 3.11 and explain the changes in the addresses if the hardware of the CS(active low) line modified.

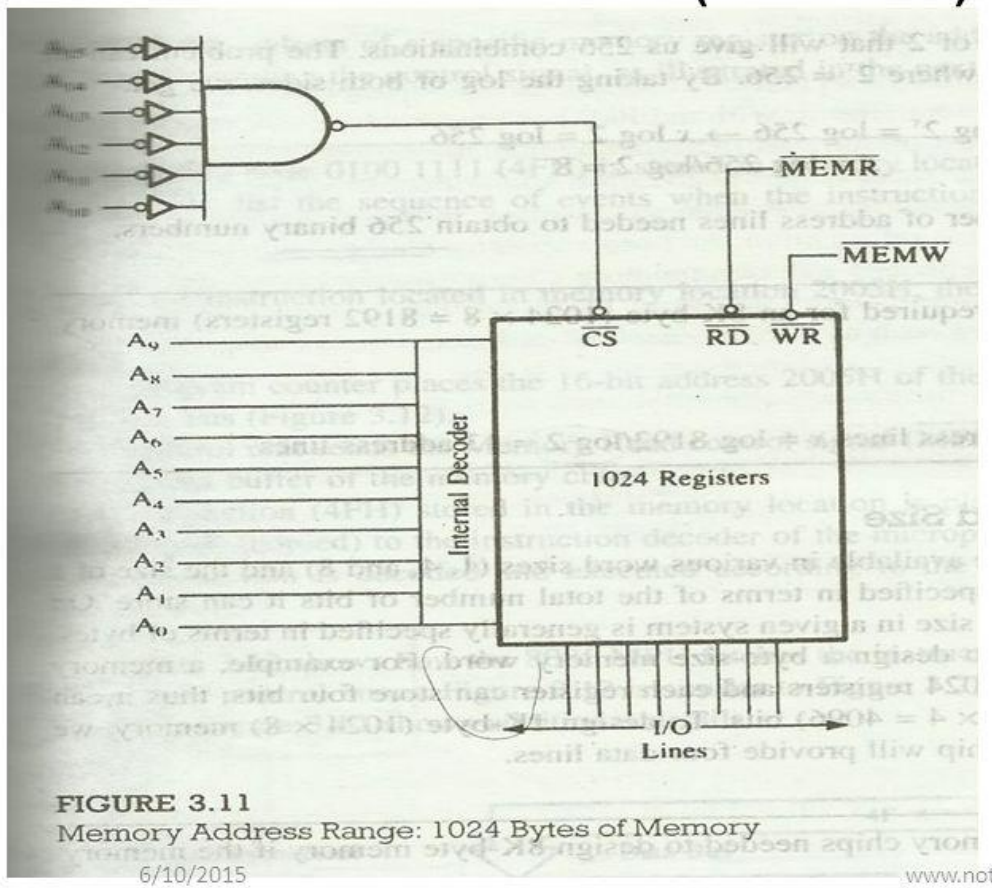
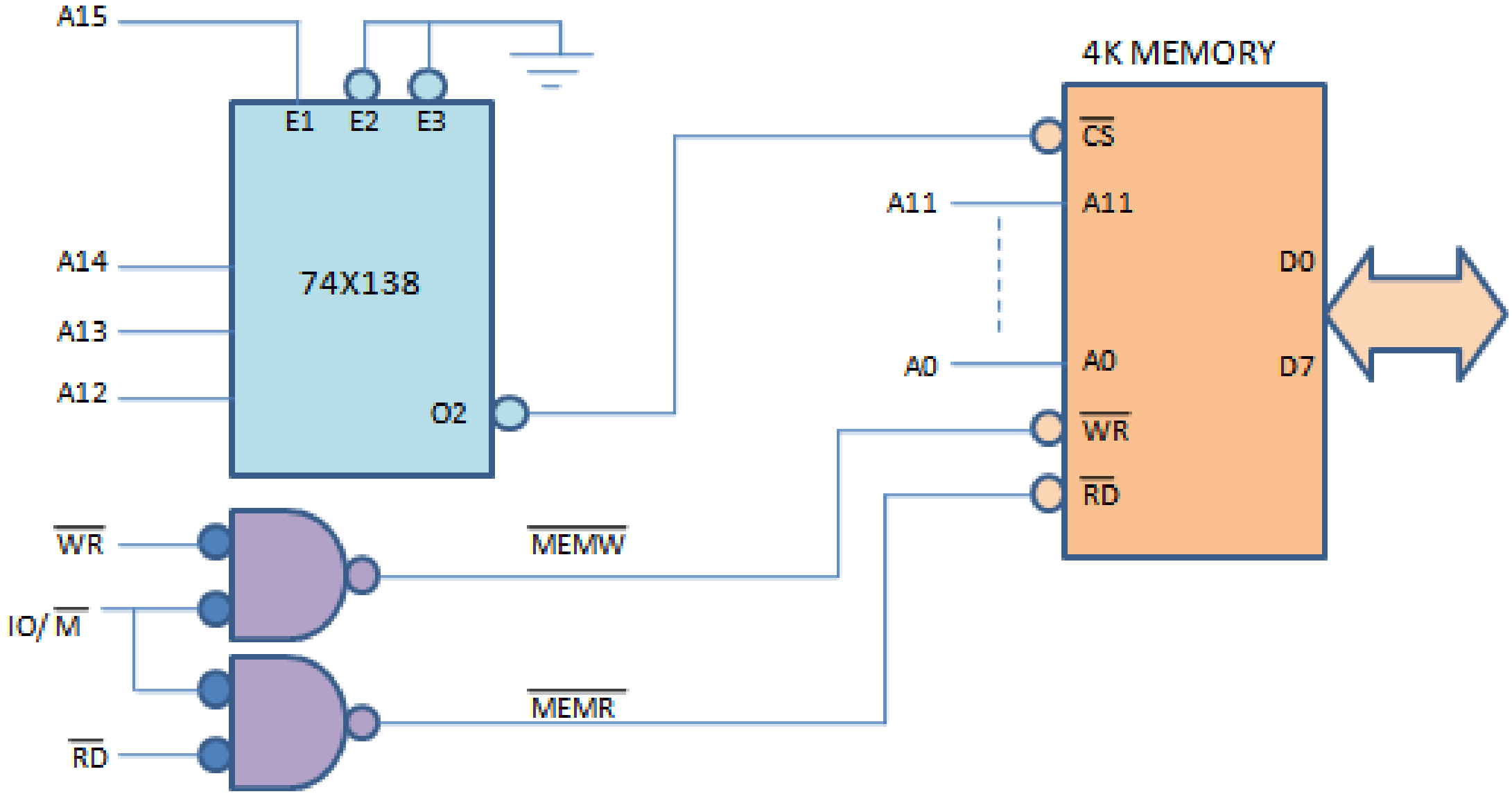


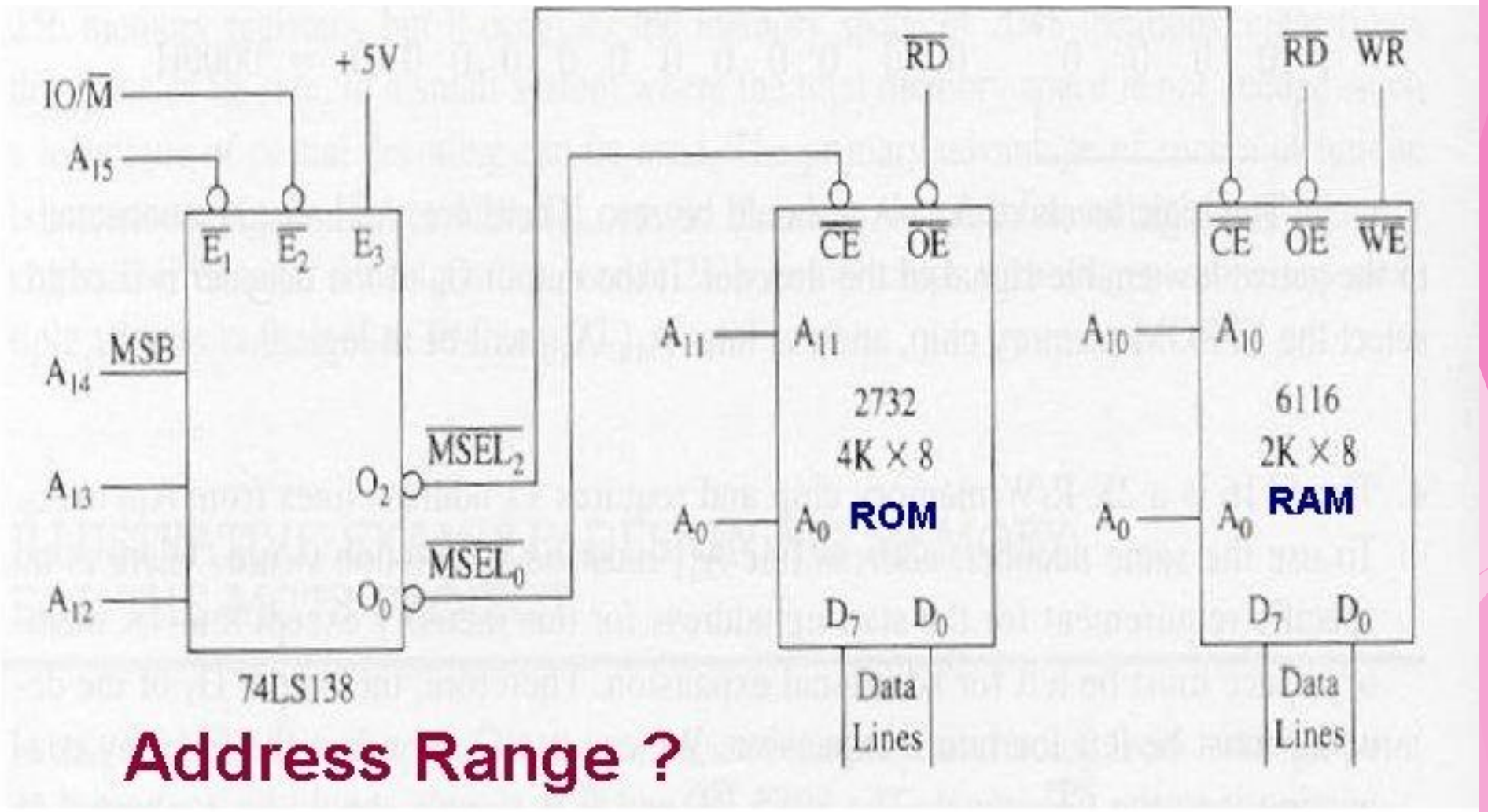
FIGURE 3.11
Memory Address Range: 1024 Bytes of Memory

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 Select Logic						1	1	1	1	1	1	1	1	1	1	= 03FFH

The memory addresses range from 0000H to 03FFH.


A15
E1
E2
E3
74X138
A14
A13
A12
O2
4K MEMORY
CS
A11
A11
D0
A0
A0
D7
WR
RD
WR
IO/ M
RD
MEMW
MEMR

Address Decoding



Address Range ?

Types of Address Decoding

- There are two types of address decoding techniques
  - Exhaustive (Absolute) Decoding
  - Partial Decoding

Exhaustive Decoding

- In this type of scheme all the 16 bits of the 8085 address bus are used to select a particular location in memory chip.
- Advantages:
  - Complete Address Utilization
  - Ease in Future Expansion
  - No Bus Contention, as all addresses are unique.
- Disadvantages
  - Increased hardware and cost.
  - Speed is less due to increased delay.

Partial Decoding

- In this scheme minimum number of address lines are used as required to select a memory location in chip.
- Advantages:
  - Simple, Cheap and Fast.
- Disadvantages:
  - Unutilized space & **fold back** (multiple mapping).
  - Bus Contention.
  - Difficult future expansion.

Interfacing I/O Devices

- Using I/O devices data can be transferred between the microprocessor and the outside world.
- This can be done in groups of 8 bits using the entire data bus. This is called parallel I/O.
- The other method is serial I/O where one bit is transferred at a time using the SID and SOD pins on the Microprocessor.

Types of Parallel Interface

- There are two ways to interface 8085 with I/O devices in parallel data transfer mode:
  - **Memory Mapped IO**
  - **IO Mapped IO**

Memory Mapped IO

- It considers them like any other memory location.
  - They are assigned a 16-bit address within the address range of the 8085.
  - The exchange of data with these devices follows the transfer of data with memory. The user uses the same instructions used for memory.

IO Mapped IO

- It treats them separately from memory.
  - I/O devices are assigned a “port number” within the 8-bit address range of 00H to FFH.
  - The user in this case would access these devices using the **IN** and **OUT** instructions only.
  - **IN 8-bit input device address**

  e.g.

  IN 08H

  Data will transfer to Accumulator from input device.
  - **OUT 8-bit output device address**

  e.g.

  OUT 01H

  Data will transfer from Accumulator to output device.

IO mapped IO V/s Memory Mapped IO

Memory Mapped IO

IO is treated as memory. 16-bit addressing.

More Decoder Hardware.

Can address 2^{16}=64k locations.

Less memory is available.

- **IO Mapped IO**
- IO is treated IO. 8- bit addressing.
- Less Decoder Hardware.
- Can address 2^8=256 locations.
- Whole memory address space is available.

IO mapped IO V/s Memory Mapped IO

Memory Mapped IO

- Memory Instructions are used.
- Memory control signals are used.
- Arithmetic and logic operations can be performed on data.
- Data transfer b/w register and IO.

IO Mapped IO

- Special Instructions are used like IN, OUT.
- Special control signals are used.
- Arithmetic and logic operations can not be performed on data.
- Data transfer b/w accumulator and IO.

The interfacing of output devices

- Output devices are usually slow.
- Also, the output is usually expected to continue appearing on the output device for a long period of time.
- Given that the data will only be present on the data lines for a very short period (microseconds), it has to be latched externally.

The interfacing of output devices

- To do this the external latch should be enabled when the port's address is present on the address bus, the IO/M signal is set high and WR is set low.
- The resulting signal would be active when the output device is being accessed by the microprocessor.
- Decoding the address bus (for memory-mapped devices) follows the same techniques discussed in interfacing memory.

Interfacing of Input Devices

- The basic concepts are similar to interfacing of output devices.
- The address lines are decoded to generate a signal that is active when the particular port is being accessed.
- An IORD signal is generated by combining the IO/M and the RD signals from the microprocessor.

Interfacing of Input Devices

- A tri-state buffer is used to connect the input device to the data bus.
- The control (Enable) for these buffers is connected to the result of combining the address signal and the signal IORD.

Instruction stores in Memory Address

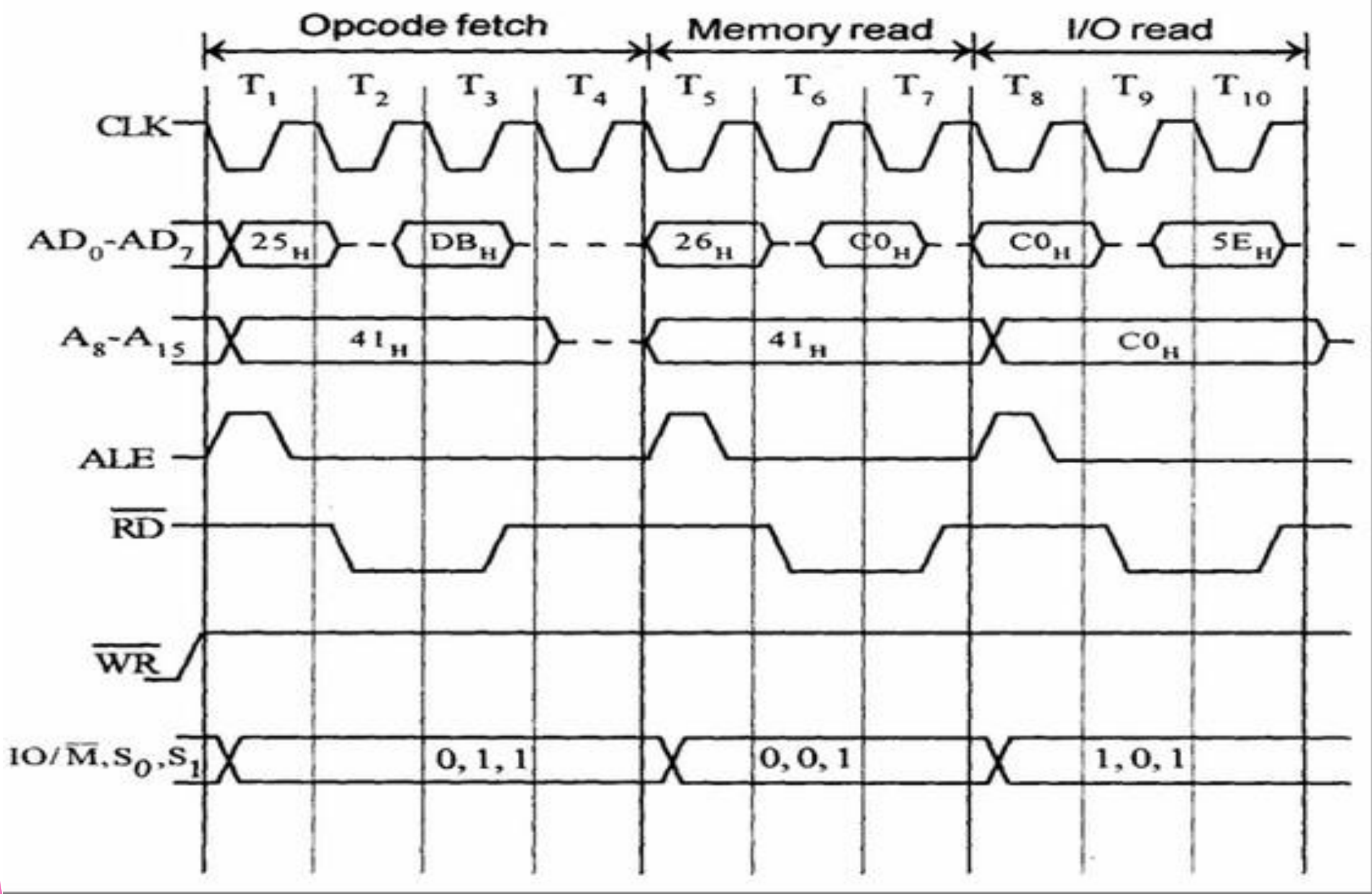
Address	Mnemonics	OpCode
2030	IN 80H	DB
2031		80

Timing diagram for IN 41H

- Fetching the Opcode DBH from the memory 2030H.
- Read the port address C0H from 2031H.
- Read the content of port C0H and send it to the accumulator.
- Let the content of port is 5EH.

It require 3 m/c cycles 10 T states

opcode fetch cycle (4T)
memory read cycle (3T)
I/O read cycle (3T)


Opcode fetch
Memory read
I/O read
T1
T2
T3
T4
T5
T6
T7
T8
T9
T10
CLK
AD0-AD7
25H
DBH
26H
C0H
C0H
5EH
A8-A15
41H
41H
C0H
ALE
RD
WR
IO/M, S0, S1
0, 1, 1
0, 0, 1
1, 0, 1

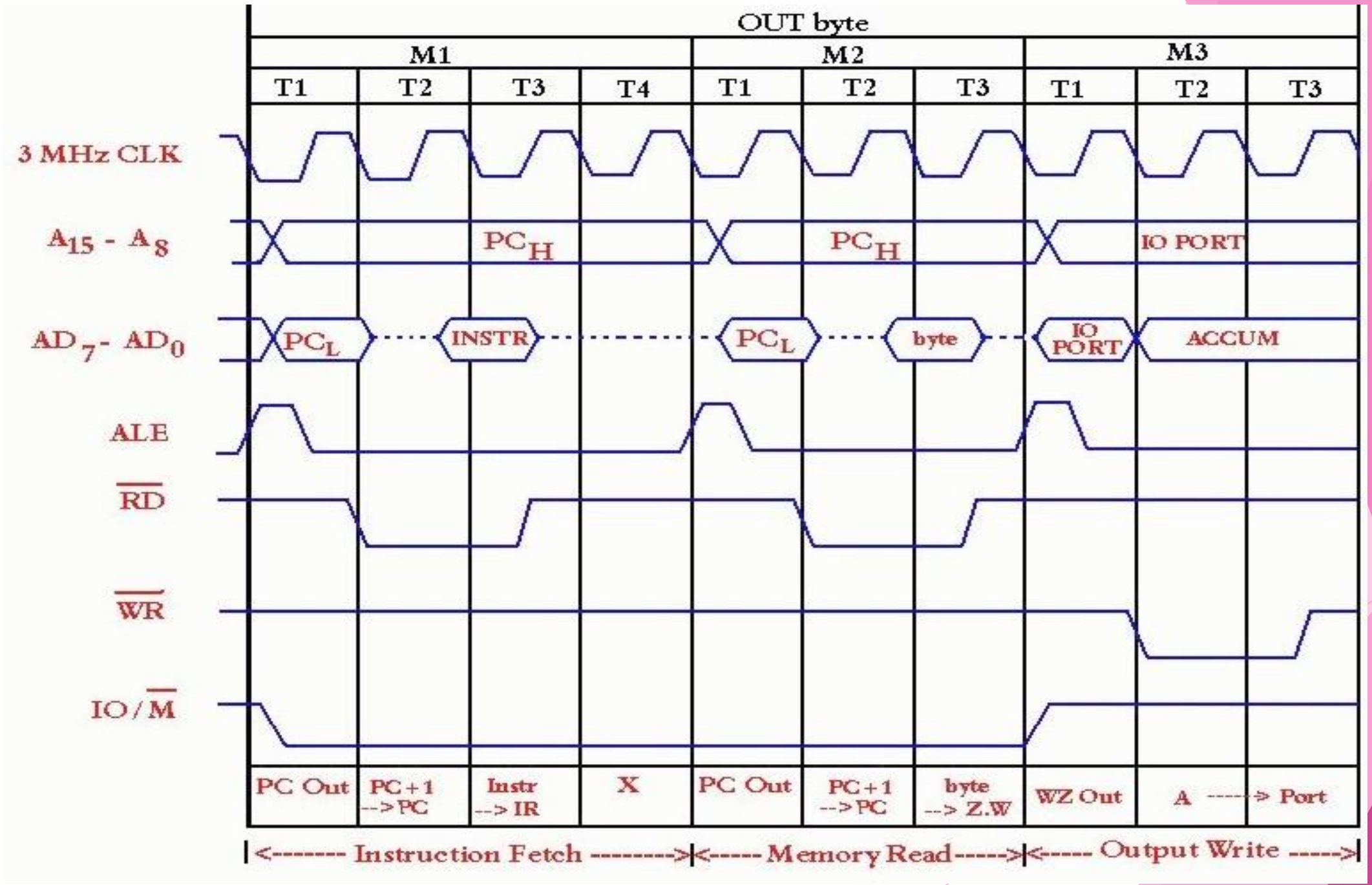
OUT instruction

Machines Cycles(10T):

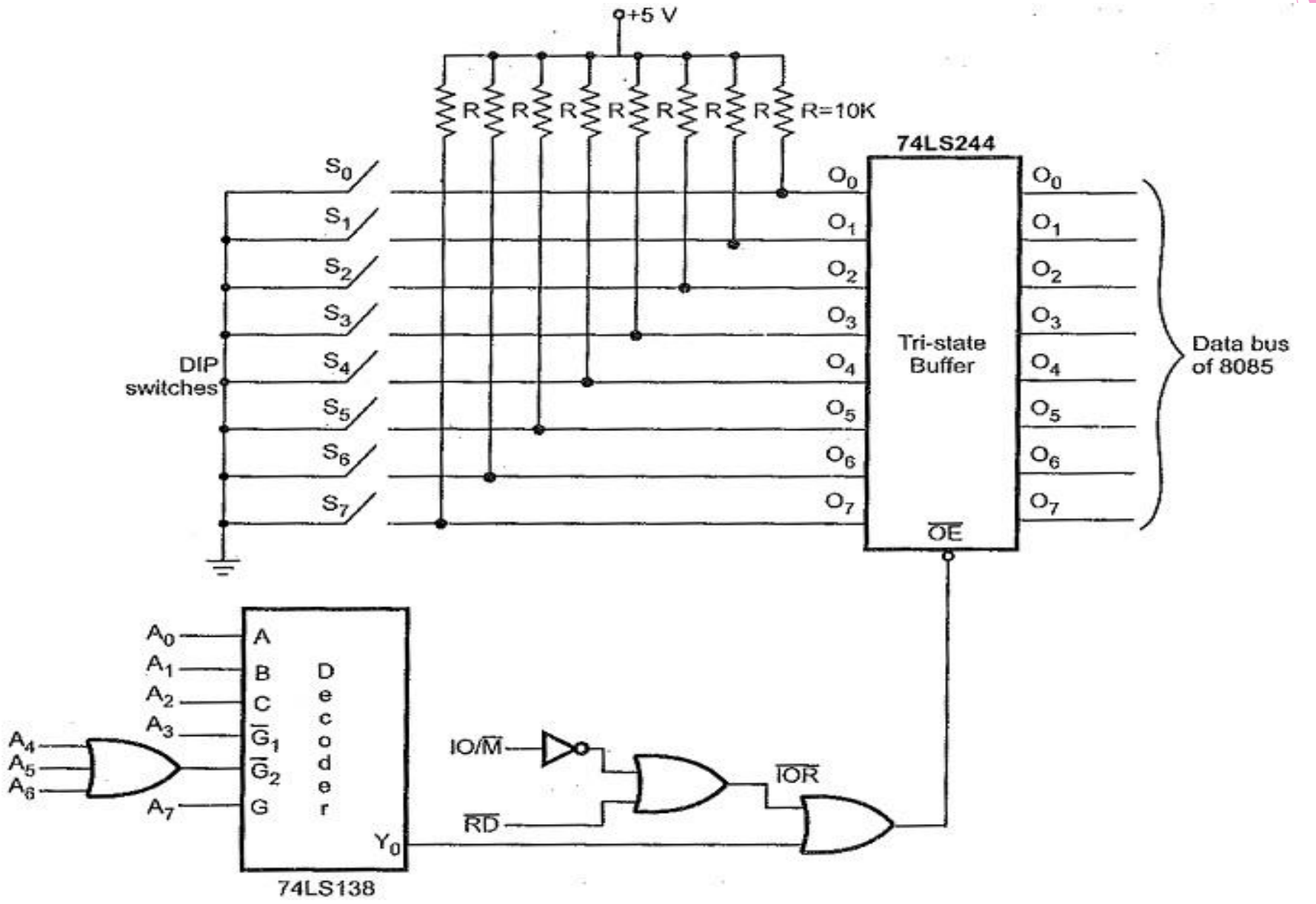
1.instruction fetch cycle(4T)

2.memory read cycle (3T)

3.IO write cycle (3T)


OUT byte
M1
M2
M3
T1
T2
T3
T4
T1
T2
T3
T1
T2
T3
3 MHz CLK
A15 - A8
PCH
PCH
IO PORT
AD7 - AD0
PCL
INSTR
PCL
byte
IO PORT
ACCUM
ALE
RD
WR
IO/M
PC Out
PC+1 --> PC
Instr --> IR
X
PC Out
PC+1 --> PC
byte --> Z.W
WZ Out
A ----> Port
<------- Instruction Fetch -------->
<----- Memory Read ----->
<----- Output Write ----->|

Interfacing of Input Switches

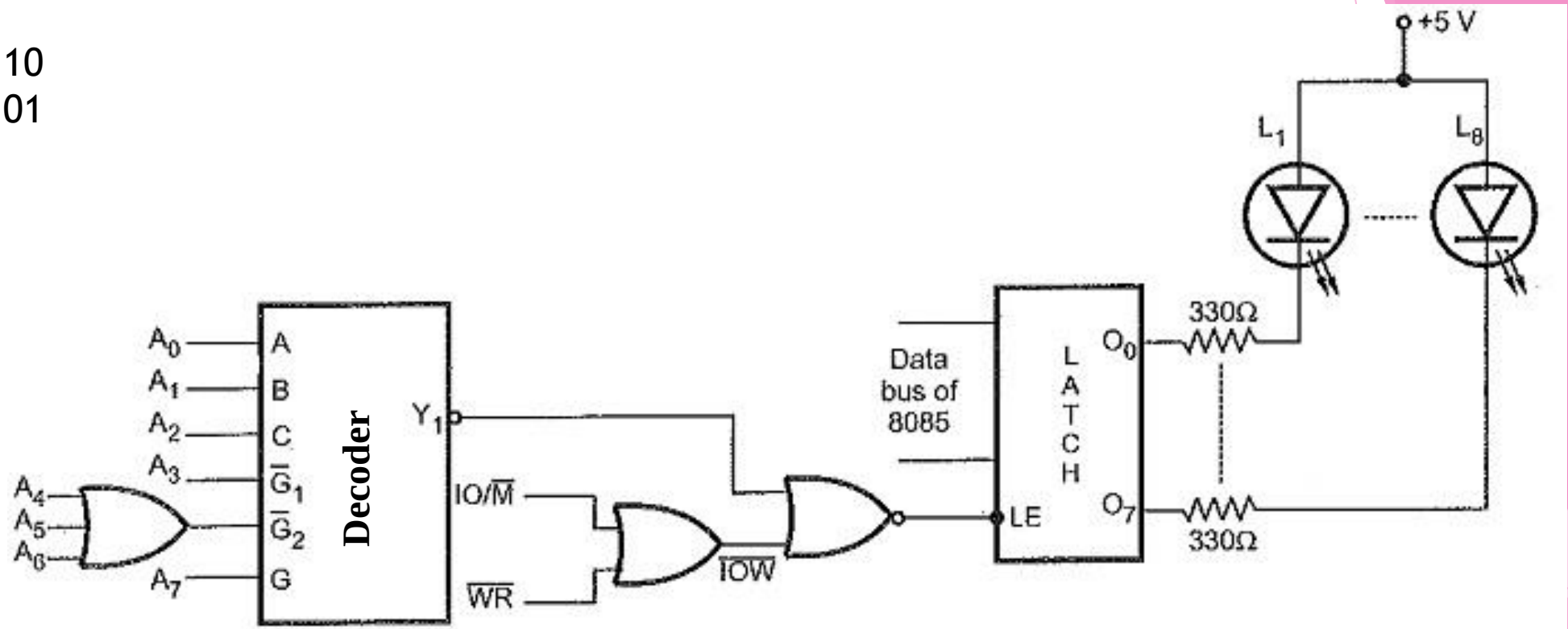


Circuit Diagram to Interface Input Port

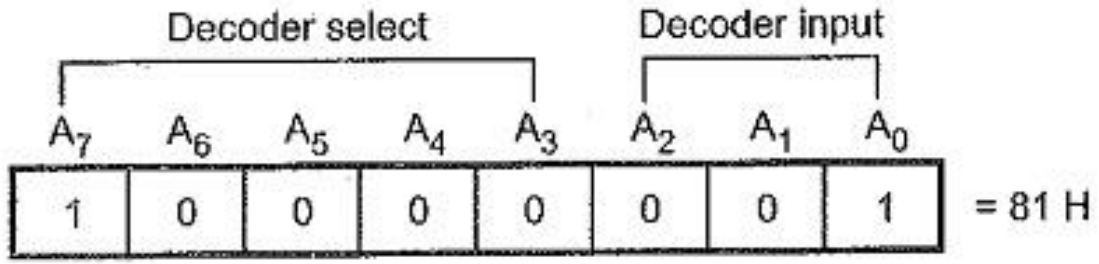
IN 80H

Interfacing of Output LEDs

11111110
11111101

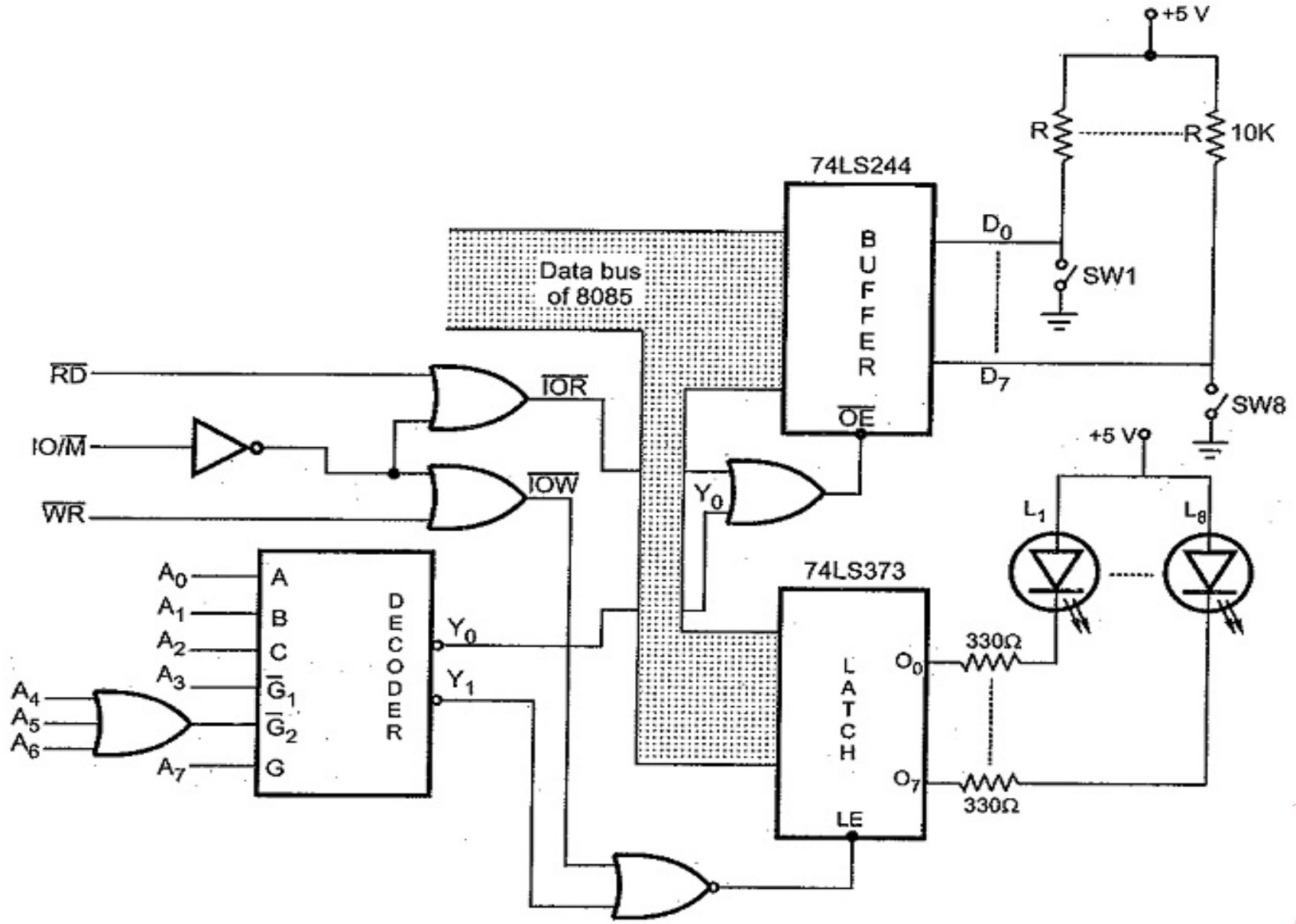


OUT 81H



Circuit Diagram to Interface Output Port

Interfacing of Input Switches and Output LEDs



Interfacing Using I/O Mapped I/O

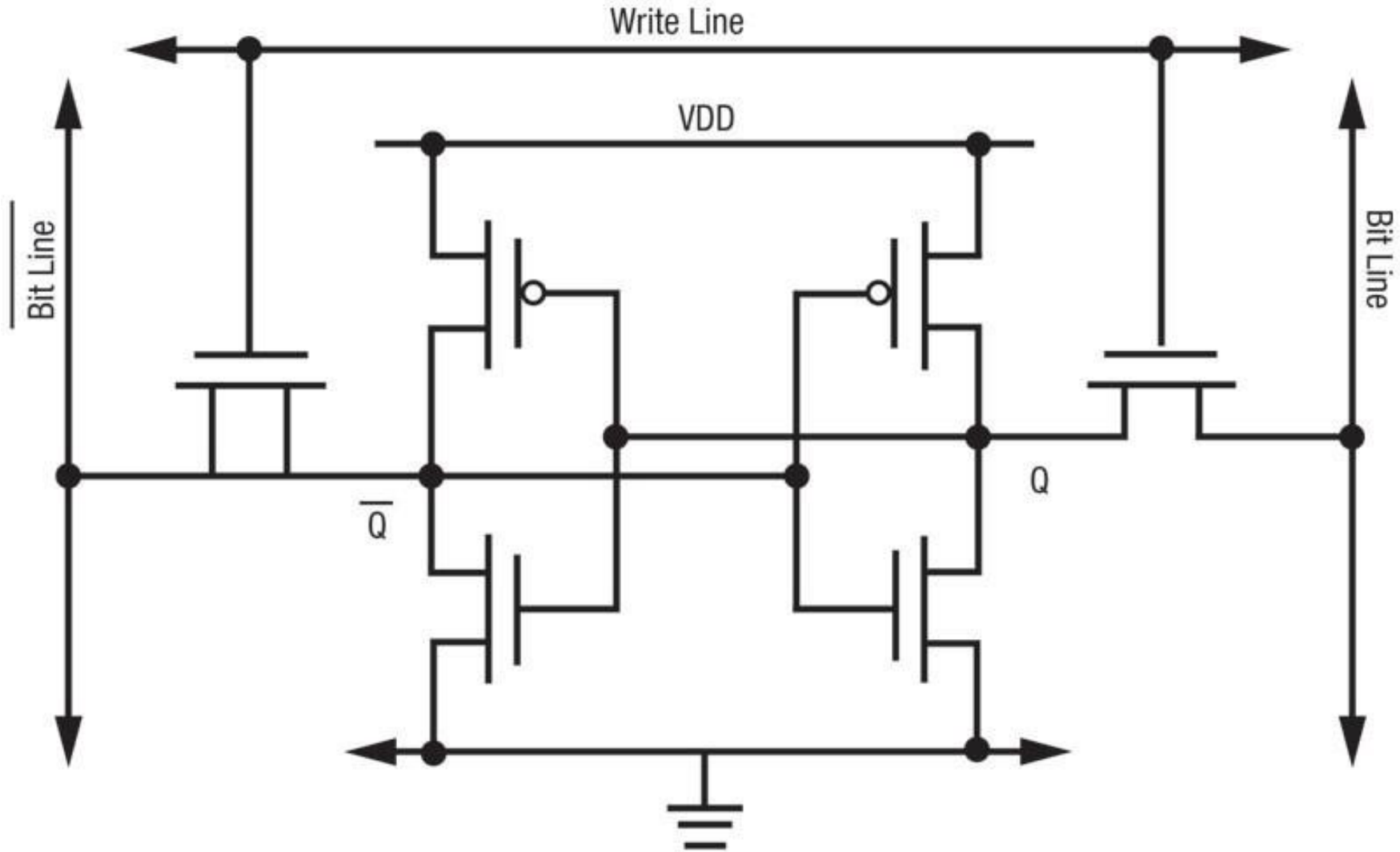
Basic RAM Cell

- RAM is a type of computer memory that can be accessed randomly i.e. any location can be accessed any time within chip.
- It is most common type of memory found in computers, printers etc.
- It is basically of two types:
  - SRAM
  - DRAM

SRAM

- SRAM stands for Static Random Access Memory.
- This memory is made up of flip-flops and stores the bit as a voltage.
- Each cell requires 6 transistors hence chip has low density but high speed.
- More expensive and consumes more power.
- Often known as cache memory in high speed PCs.

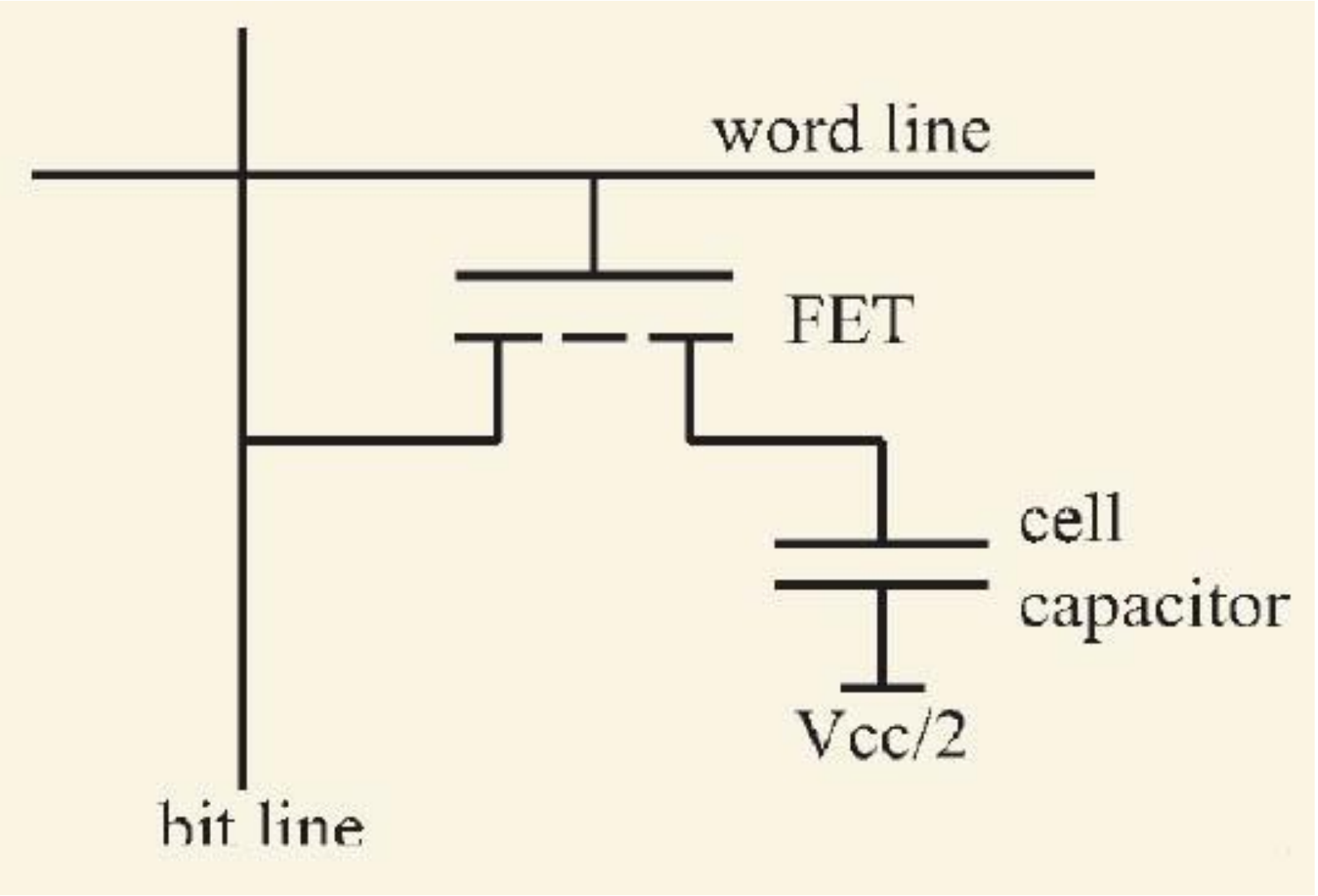
Basic SRAM Cell



DRAM

- DRAM stands for Dynamic Random Access Memory.
- This memory is made up of MOS transistor gates and it stores the bit as charge.
- High density, low power consumption, cheap as compared to SRAM.
- Due to leakage of charge requires frequent refreshing and hence extra circuitry.

Basic DRAM



ROM

- ROM is a read only memory.
- It retains the information even if power is turned off.
- It contains permanently stored instructions that help in staring up of a computer e.g. BIOS or Basic Input Output System.
- These are of following three basic types

– PROM, EPROM, EEPROM

PROM

- The Programmable Read Only Memory can be programmed only once in its lifetime.
- Information once stored can not be erased.
- Requires special hardware circuit to program it.

EPROM

- Stands for Erasable Programmable Read Only Memory.
- These ROMs can be erased and programmed again and again.
- Can be erased with UV light or electricity.
- Main disadvantage is that it takes 15 to 20 minutes to erase it.

EEPROM

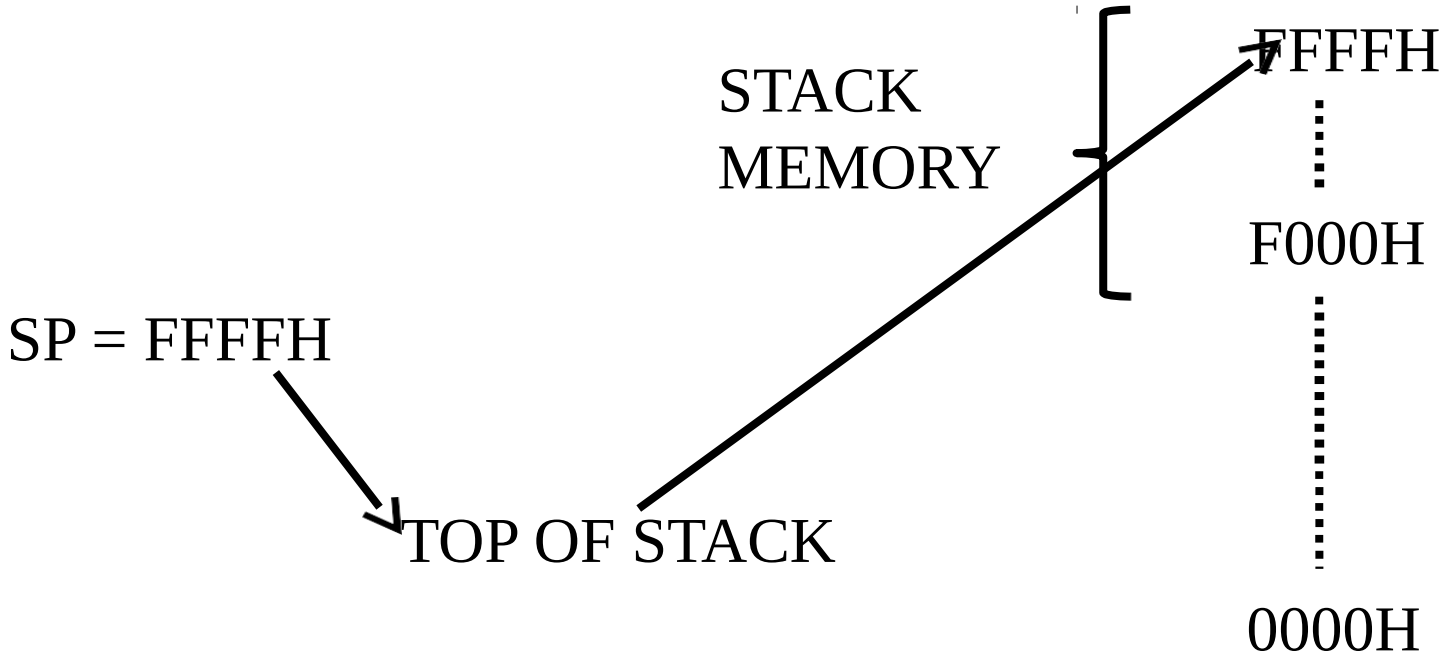
- Stands for Electrically Erasable Programmable Read Only Memory.
- Information can be erased electrically at register level rather than erasing entire information.
- It requires lesser erasing time.

Stack

- It is a part of memory, reserved in RAM, used to temporarily store information during execution of program.
- Starting address of stack is loaded in “Stack Pointer (SP)” (a 16-bit register).
- The address pointed to by SP is known as “Top of Stack”, which is always an empty memory location.

Stack Initialization

- Stack can be defined anywhere in RAM.
- But generally it initialized from highest (end) address of RAM to avoid any data loss.



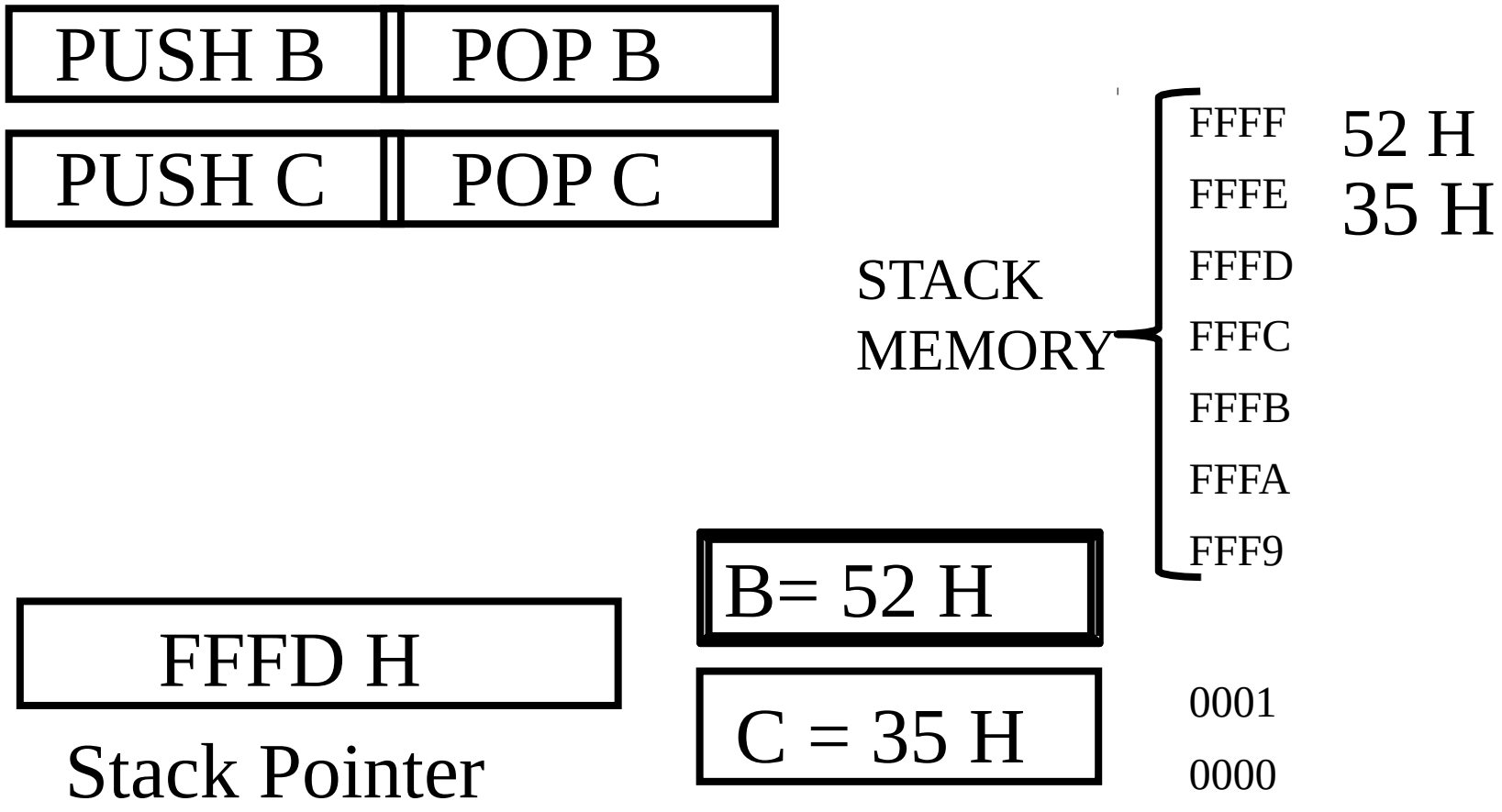
Size of Stack Memory

- Theoretically there is no limitation on the size of stack memory.
- Practically the size of stack memory is limited to the availability of free RAM.
- As RAM is used to store temporarily program and data during execution, hence only free RAM can be used as stack.

Storing Data on Stack

- Stack is Last-In-First-Out (LIFO) type of memory.
- When information is stored on stack, the Stack Pointer register decrements to point to lower empty address.
- When information is read from stack, the Stack Pointer register increments to point to higher empty address.

Animation Stack Memory



Advantages of Stack

- Address is always in Stack Pointer, need not be part of instruction, therefore, stack access is always faster.
- Stack instructions are short with only one operand.
- Used to save important data before branch instruction e.g. jump or interrupt instruction.

THANKS