Lecture -4-
Memory Interfacing

Memory Interfacing

- When we are executing any instruction, we need the microprocessor to access the memory for reading instruction codes and the data stored in the memory. For this, both the memory and the microprocessor requires some signals to read from and write to registers.
- The interfacing process includes some key factors to match with the memory requirements and microprocessor signals. The interfacing circuit therefore should be designed in such a way that it matches the memory signal requirements with the signals of the microprocessor.

Memory Interfacing

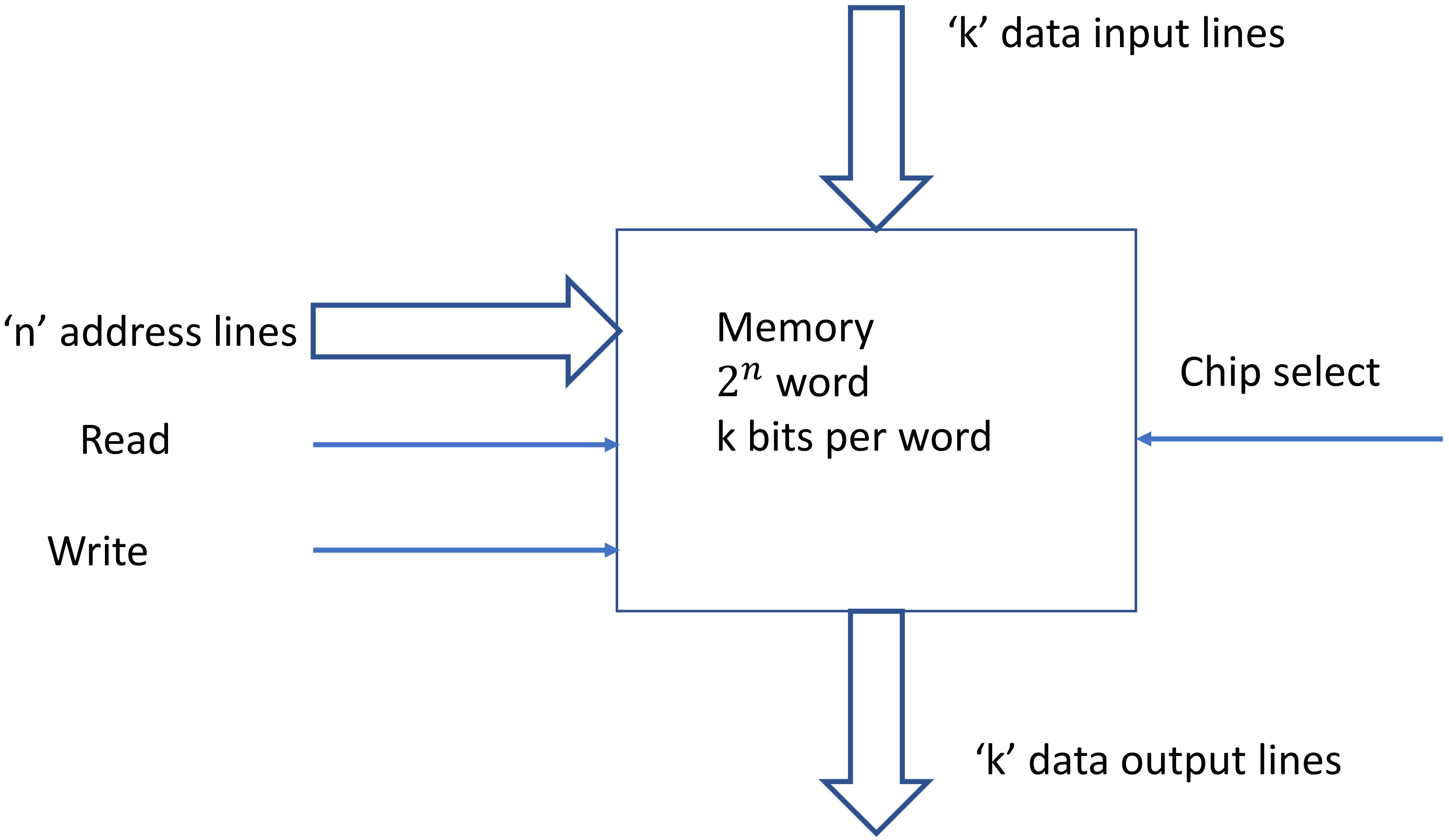


Figure (1)
Block diagram of memory

Memory Interfacing

- A general form diagram of ROM and RAM show in figure (1). Pin connections common to all memory devices are:
- Address connections: All memory devices have address inputs that select a memory location within the memory device. Address inputs are labeled ($A_0 - A_n$)
- Data connections: All memory devices have a set of data outputs or input/outputs. Today many of them have bi-directional common I/O pins.
- Selection connections: Each memory device has an input that selects or enables the memory device. This kind of input is most often called a chip select (CS) , chip enable (CE) or simply select (S) input.
- Control connections: The control input most often found on the ROM is the output enable (OE) or gate (G)this allows data to flow out of the output data pins of the ROM.

Minimum mode memory Interfacing

- Address bus & Data bus are multiplexed on same lines (AD0 to AD15).
- During first clock cycle, it serves as a memory/ IO address bus.
- For second and third clock cycles it acts as data bus and carries data.
- Demultiplexing refers to separating Address & Data signals for read/write operations.
- The control signals in minimum mode are listed in table below:-

$\overline{\boldsymbol{RD}}$	$\overline{\boldsymbol{WR}}$	$\boldsymbol{IO/\overline{M}}$	**Signal**
0	1	0	$\overline{MEMR}$
1	0	0	$\overline{ME\mathrm{MW}}$
0	1	1	$\overline{IOR}$
1	0	1	$\overline{IOW}$

Minimum mode memory Interfacing

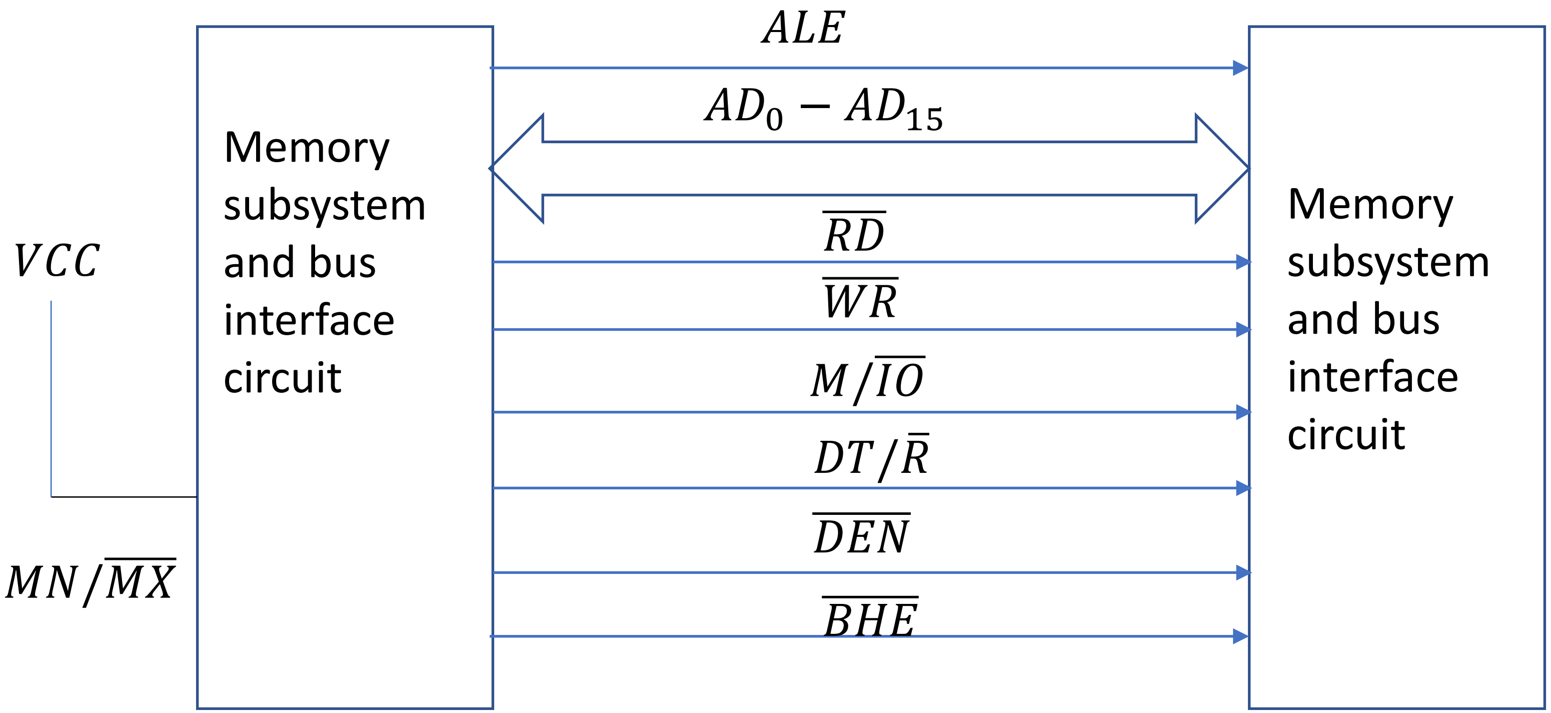
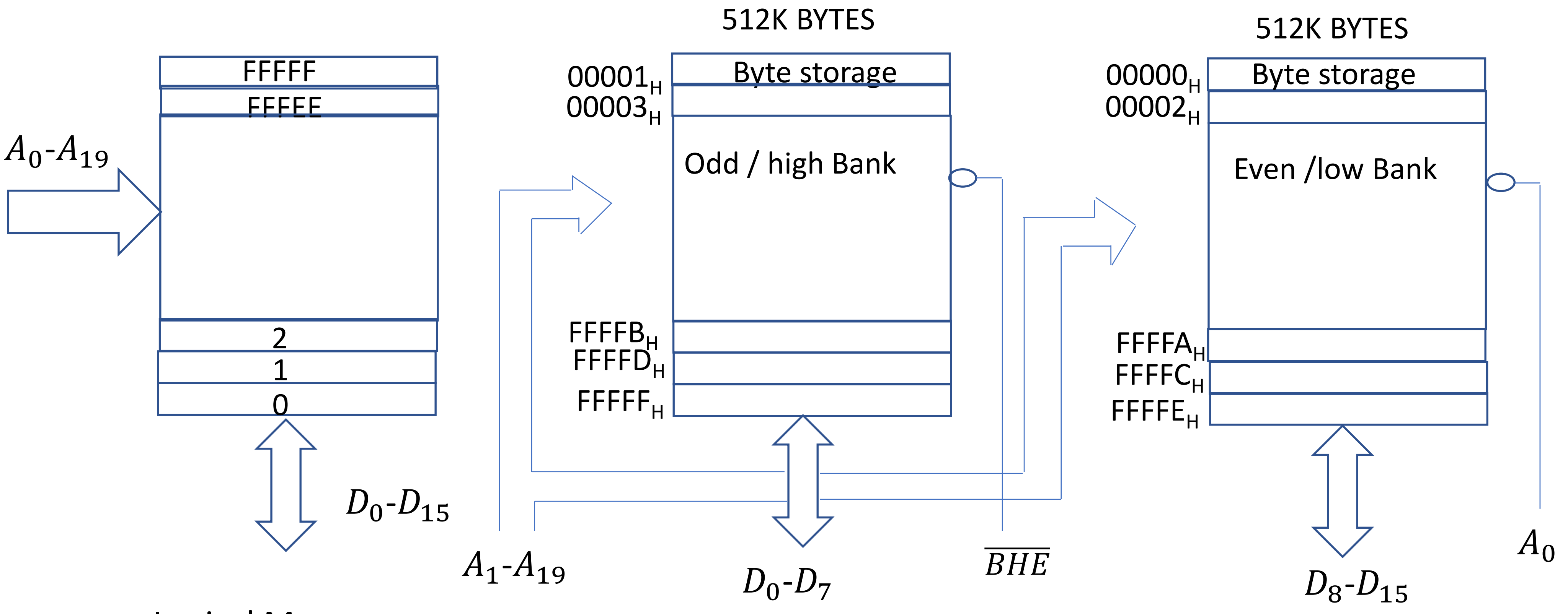


Figure (2) block diagram for minimum mode 8086 memory interface

8086 Memory Organization

- The memory address space of the 8086-based microcomputers has different logical and physical organizations.
- Logically, memory is implemented as a single 1M × 8 memory bank. The byte-wide storage locations are assigned consecutive addresses over the range from 00000H through FFFFFH.
- Physically, memory is implemented as two independent 512 Kbyte banks: the low (even) bank and the high (odd) bank.

8086 Memory Organization (a) Logical memory organization, and (b) Physical memory organization (high and low memory banks) of the 8086 microprocessor.



a-Logical Memory

b- Physical Memory

Figure (3)

8086 Memory Organization

- To distinguish between odd and even bytes, the CPU provides a signal called BHE (bus high enable).
- $\overline{BHE}$ and A_0 are used to select the odd and even byte, as shown in the table below.

$\overline{BHE}$	A_0	**Function**
0	0	Choose both odd and even memory bank
0	1	Choose only odd memory bank
1	0	Choose only even memory bank
1	1	None is chosen

8086 Memory Organization

- The data pins are typically bi-directional in read-write memories.
  - ❑The number of data pins is related to the size of the memory location. For example, an 8-bit wide (byte-wide) memory device has 8 data pins.
- Each memory device has at least one chip select (CS) or chip enable (CE) or select (S) pin that enables the memory device.
  - ❑This enables read and/or write operations.
  - ❑If more than one are present, then all must be 0 in order to perform a read or write.

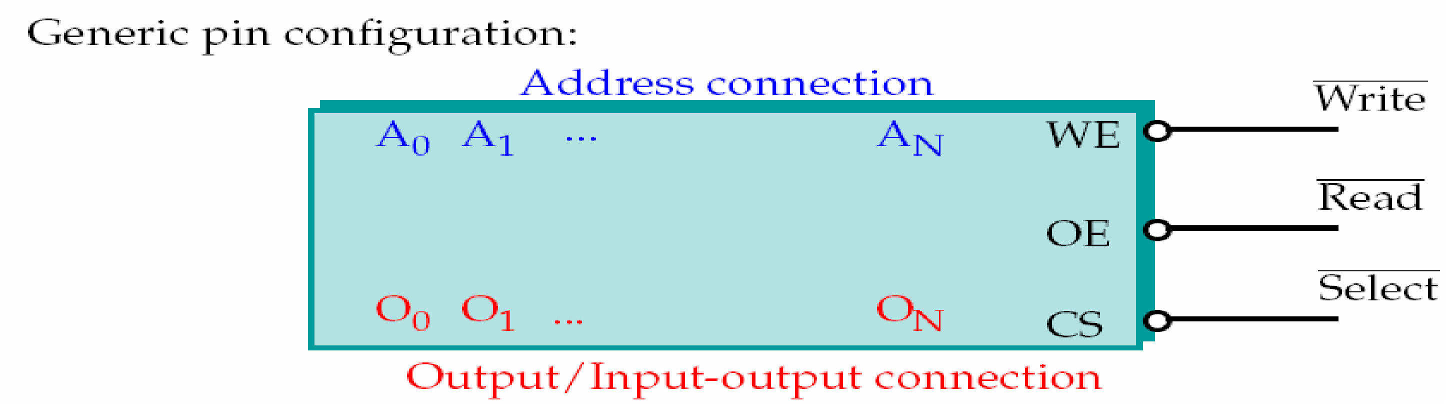

Generic pin configuration:
Address connection
A_0 A_1 ... A_N
WE
OE
CS
$\overline{\text{Write}}$
$\overline{\text{Read}}$
$\overline{\text{Select}}$
O_0 O_1 ... O_N
Output/Input-output connection

Figure (4)

Memory Interfacing

- Address Decoding
- In order to attach a memory device to the microprocessor, it is necessary to decode the address sent from the microprocessor. Decoding makes the memory function at a unique section or partition of the memory map.
- Without an address decoder, only one memory device can be connected to a microprocessor, which would make it virtually useless.

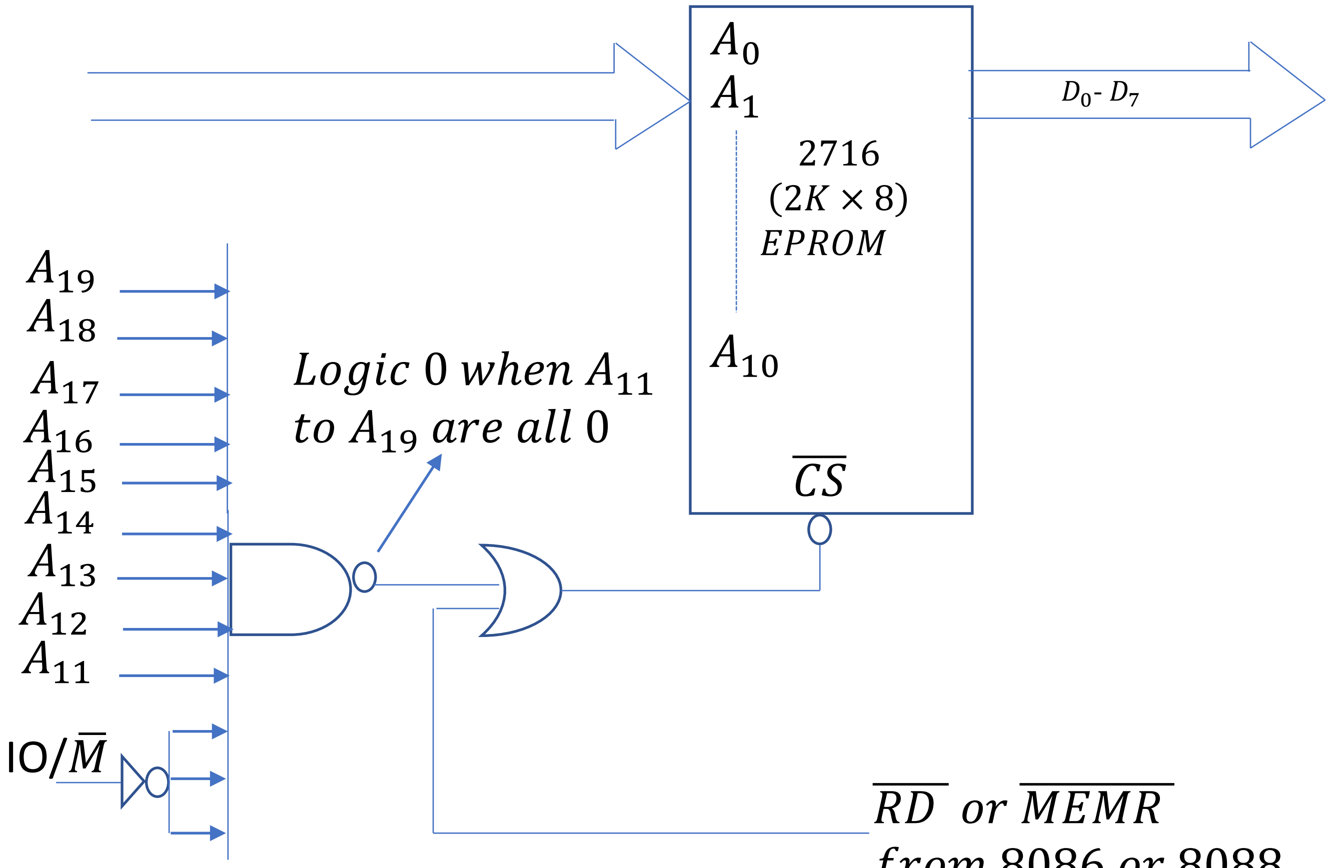
Memory Interfacing

- The processor can usually address a memory space that is much larger than the memory space covered by an individual memory chip.
- In order to splice a memory device into the address space off the processor, decoding is necessary.
- For example, the 8088 issues 20-bit addresses for a total off 1MB off memory address space.
- The BIOS on a 2716 EPROM has only 2KB of memory and 11 address pins.
- A decoder can be used to decode the additional 9 address pins and allow the EPROM to be placed in any 2KB section off the 1MB address space.
- The 8088 has 20 address connections and the 2716 EPROM has 11 connections.
- The 8088 sends out a 20-bit memory address whenever it reads or writes data..
- because the 2716 has only 11 address pins, there is a mismatch that must be corrected.
- The decoder corrects the mismatch by decoding address pins that do not connect to the memory component.

Memory Interfacing

- Simple NAND Gate Decoder
- When the 2K × 8 EPROM is used, address connections A10–A0 of 8088 are connected to address inputs A10–A0 of the EPROM.
- the remaining nine address pins (A19–A11) are connected to a NAND gate decoder.
- The decoder selects the EPROM from one of the 2K-byte sections of the 1M-byte memory system in the 8088 microprocessor.
- In this circuit a NAND gate decodes the memory address, as seen in the following Figure.
- A simple NAND gate decoder from FF800H to FFFFFH.
- To determine the address range that a device is mapped into :-

Memory Interfacing



$Logic\ 0\ when\ A_{11}$
$to\ A_{19}\ are\ all\ 0$

$\overline{RD}\ \ or\ \overline{MEMR}$
$from\ 8086\ or\ 8088$

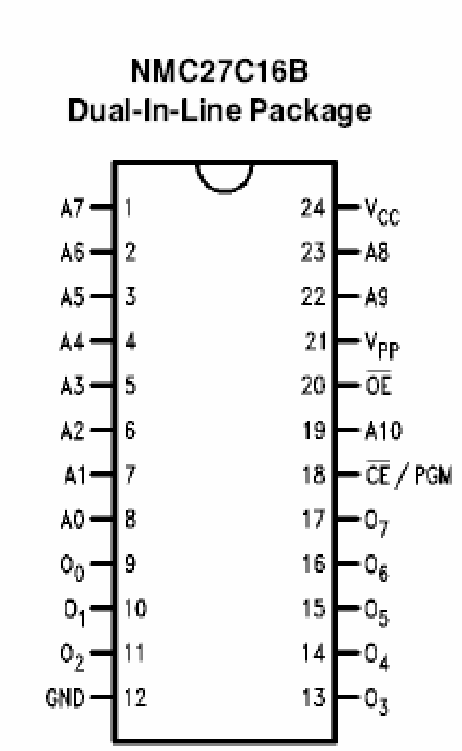


Figure (5)

Memory Interfacing

$1111 \;\; 1111 \;\; 1XXX \;\; XXXX \;\; XXXX$

$1111 \;\; 1111 \;\; 1000 \;\; 0000 \;\; 0000 \quad (FF800\,H)$

$1111 \;\; 1111 \;\; 1111 \;\; 1111 \;\; 1111 \quad (FFFFF\,H)$

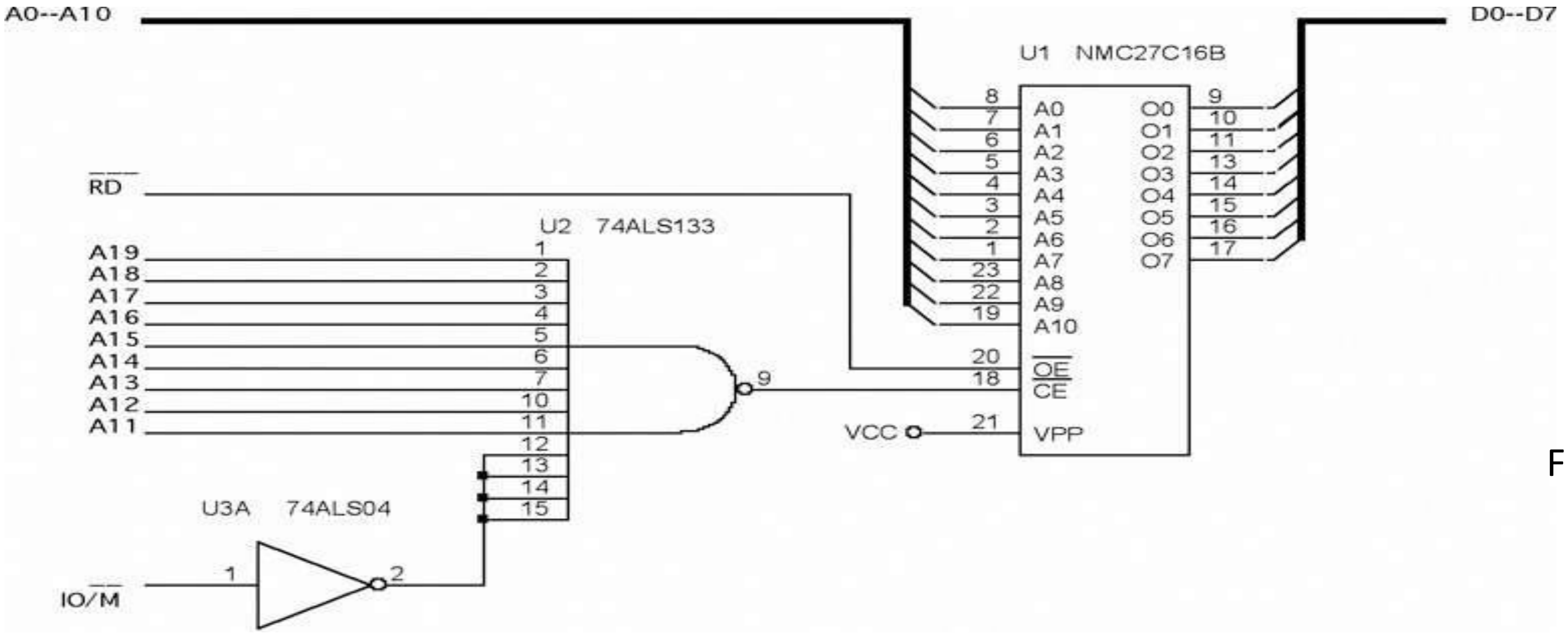
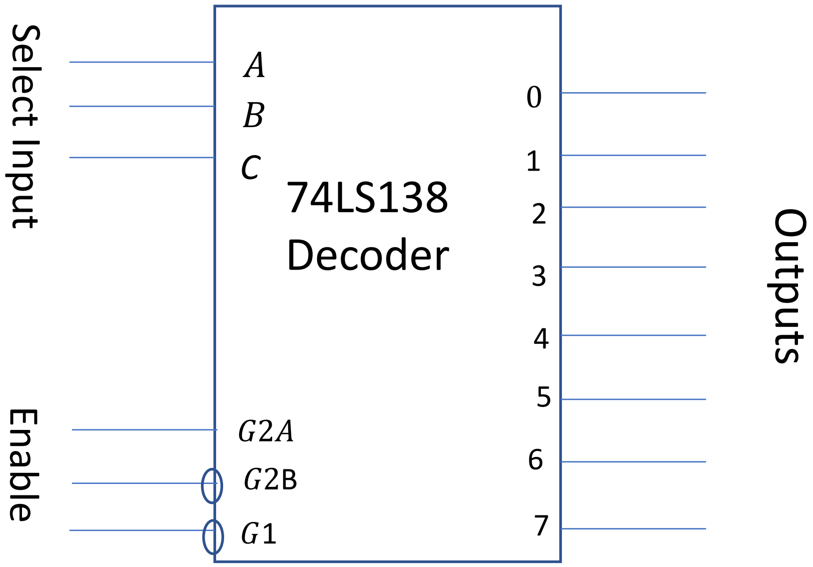


Figure (6)

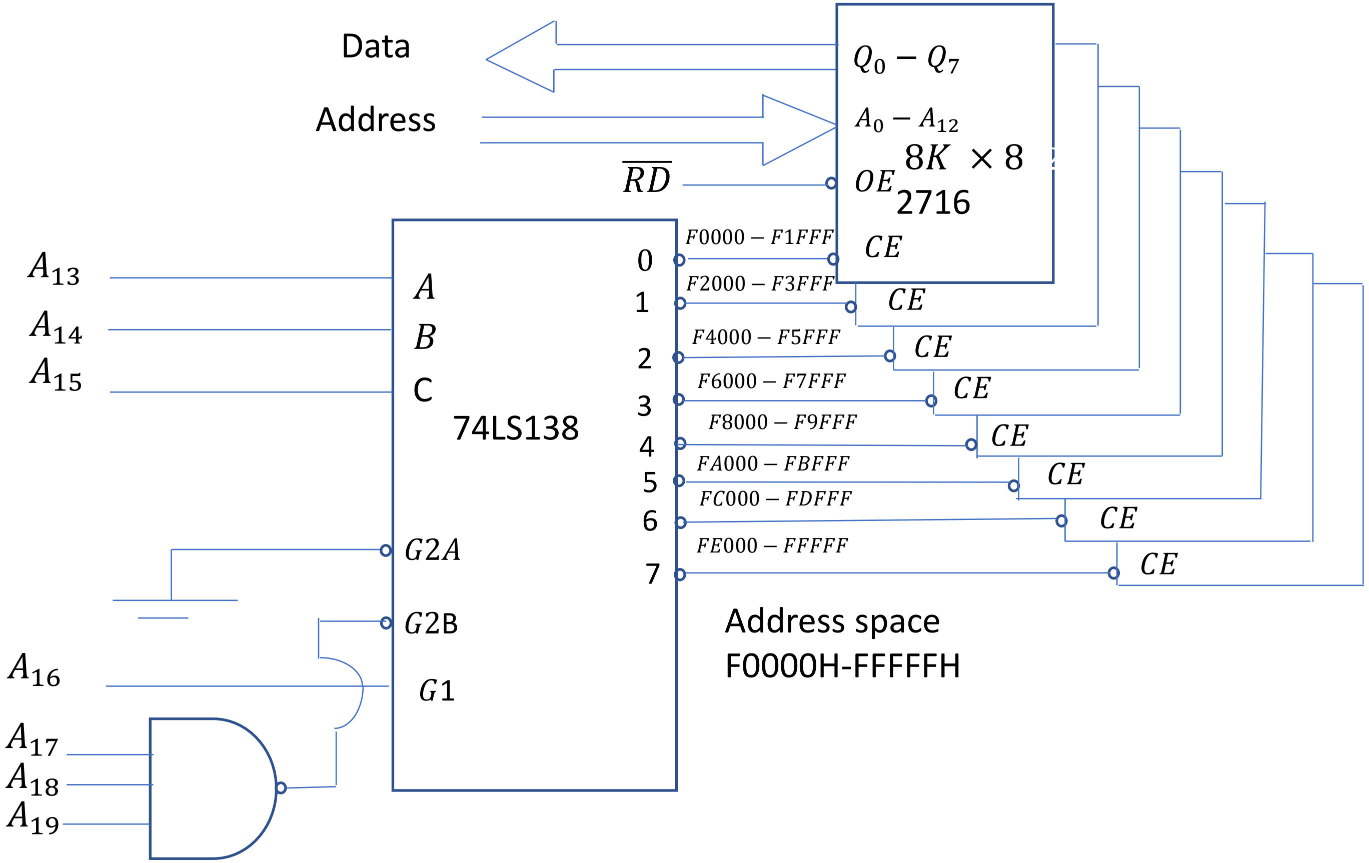
Memory Interfacing

- 3-to-8 Decoder (74LS138)



74LS138 3-to-8 line decoder Figure (7)

A circuit that uses eight 2764 EPROMs for a 64K × 8 section of memory in an 8088 microprocessor-based system. The addresses selected in this circuit are F0000H–FFFFFH.



A_{13} to A_{15} used to select a 2716

A_{16} to A_{19} used to enable the decoder

Module 0

1111 000X XXXX XXXX XXXX

1111 0000 0000 0000 0000 =F0000H

1111 0001 1111 1111 1111 =F1FFFH

Module 7

1111 111X XXXX XXXX XXXX

1111 1110 0000 0000 0000 =FE000H

1111 1111 1111 1111 1111 =FFFFFH

Figure (8)

Memory Interfacing

- In this circuit, a three-input NAND gate is connected to address bits A19–A17.
- When all three address inputs are high, the output off this NAND gate goes low and enables input G2B of the 74LS138.
- Input G1 is connected directly to A16.
- In order to enable this decoder, the first four address connections ((A19–A16)) must all be high.
- Address inputs C, B, and A connect to microprocessor address pins A15–A13.
- These three address inputs determine which output pin goes low and which EPROM is selected whenever 8088 outputs a memory address within this range to the memory system.

Memory Interfacing

- The 3 to 8 decoder
- $A_{19} \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots A_0$
- $1111\ \ XXXX\ XXXX\ XXXX\ XXXX$
- or
- $1111\ \ 0000\ \ 0000\ \ \ 0000\ \ 0000 = F0000H$
- to
- $1111\ \ 1111\ \ 1111\ \ 1111\ \ 1111\ \ = FFFFFH$

Memory Interfacing

The 3-to-8 Line Decoder (74LS138)

Example:

CBA

1111 000X XXXX XXXX XXXX

or

1111 0000 0000 0000 0000 = F0000H

to

1111 0001 1111 1111 1111 = F1FFFH

Memory Interfacing

The 3-to-8 Line Decoder (74LS138)

Example:

CBA

1111 001X XXXX XXXX XXXX

or

1111 0010 0000 0000 0000 = F2000H

to

1111 0011 1111 1111 1111 = F3FFFH

Memory Interfacing

- In many applications, the microcomputer system requirement for memory is greater than what is available in a single device. There are two basic reasons for expanding memory capacity:

  1. The byte-wide length is not large enough

  2. The total storage capacity is not enough bytes.

- Both of these expansion needs can be satisfied by interconnecting a number of ICs.

Example (1)

Implement 32Kx16 EPROM using 32KX8 EPROM?

Solution:-

$$32 \ \times 8 \, bits \ = 2^{15} \ \times 8 \rightarrow (A_0 - A_{14})$$

$$32 \ \times 16 \, bits \ = 2^{15} \ \times 16 \rightarrow (A_0 - A_{14})$$

Memory Interfacing

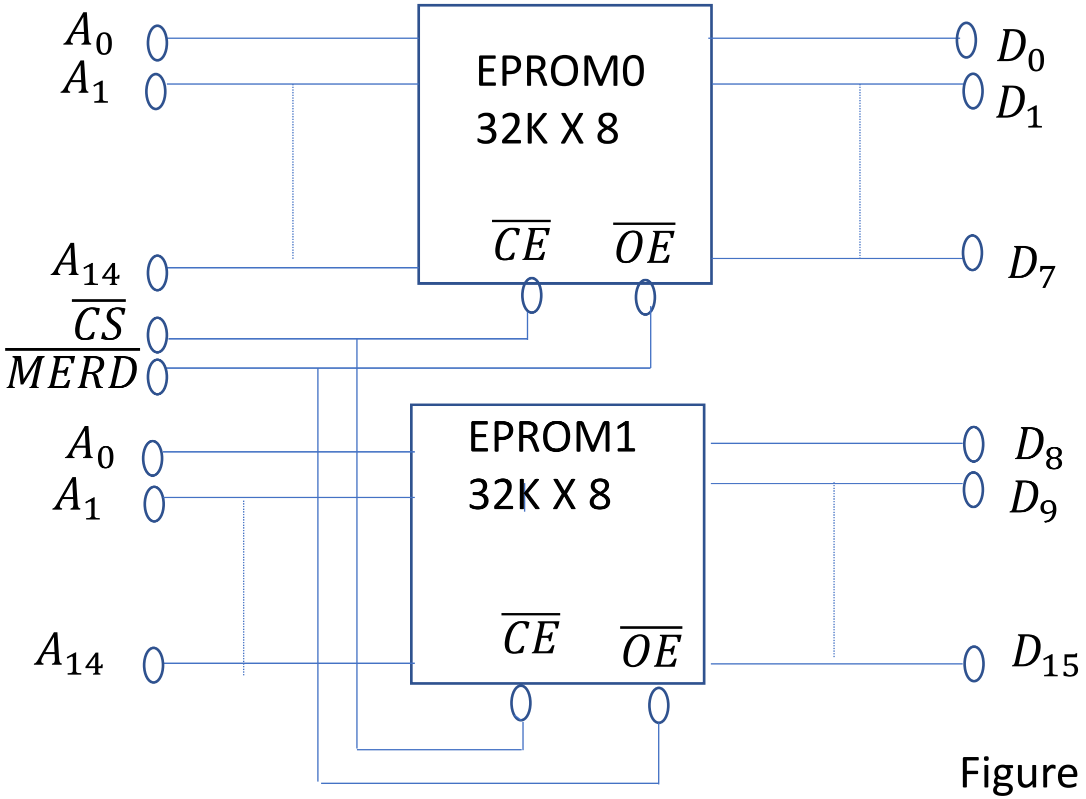


Figure (9)

Example (2)

If the memory chip size is 1024 X 4 bits, how many chips are required to make up 2K bytes of memory?

- $1024 \ \times 4 \ bits = 1K \times 8 = 2^{10} \times 4 \rightarrow (A_0 - A_9)$
- $2 \ K \ bytes \qquad = 2K \times 8 = 2^{11} \times 8 \rightarrow (A_0 - A_{10})$

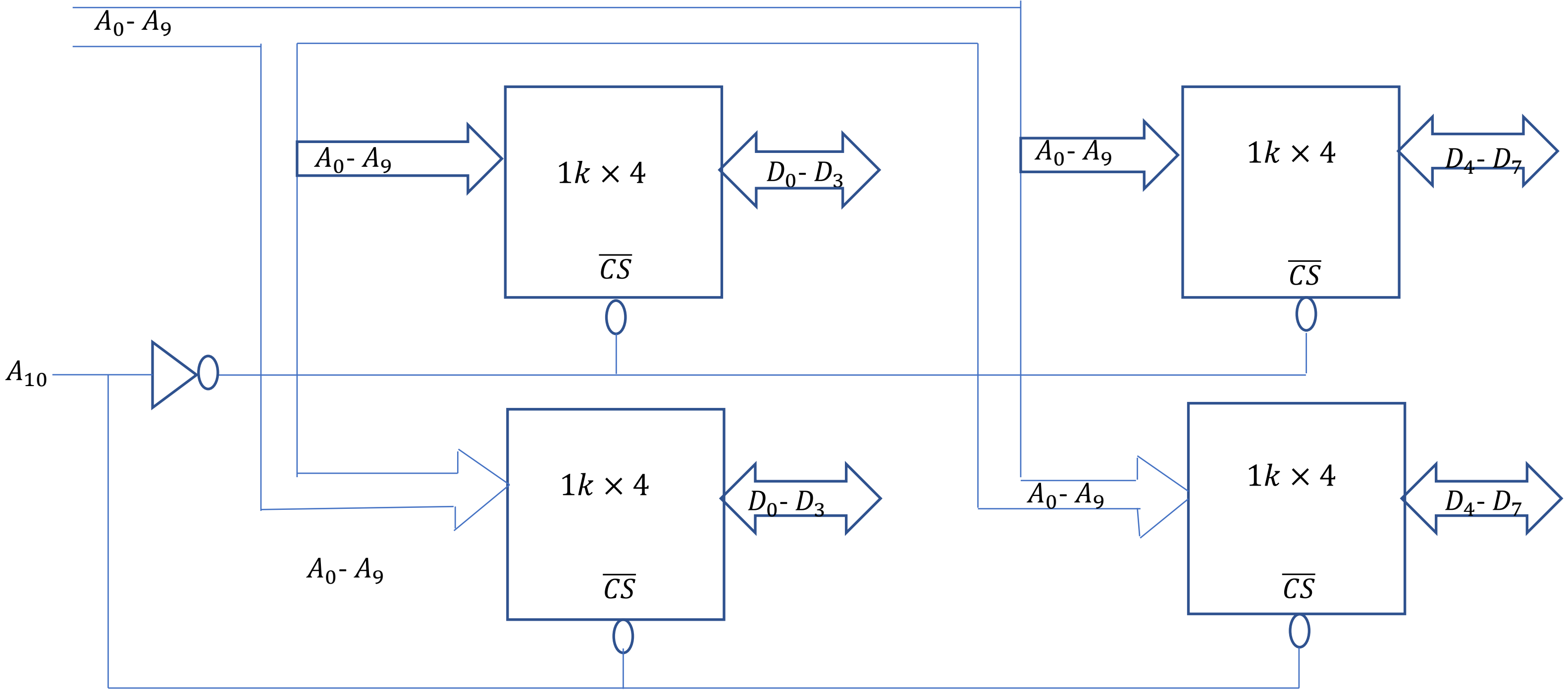


Figure (10)

Memory Interfacing

Example (3)

Show how to implement 64K× 16 EPROM using 32K×8 EPROM?

Solution:-

- $32\ K \times 8\ bits\ = 2^{15} \times 8 \rightarrow (A_0 - A_{14})$
- $64\ K \times 16\ bits\ = 2^{16} \times 8 \rightarrow (A_0 - A_{15})$

Memory Interfacing

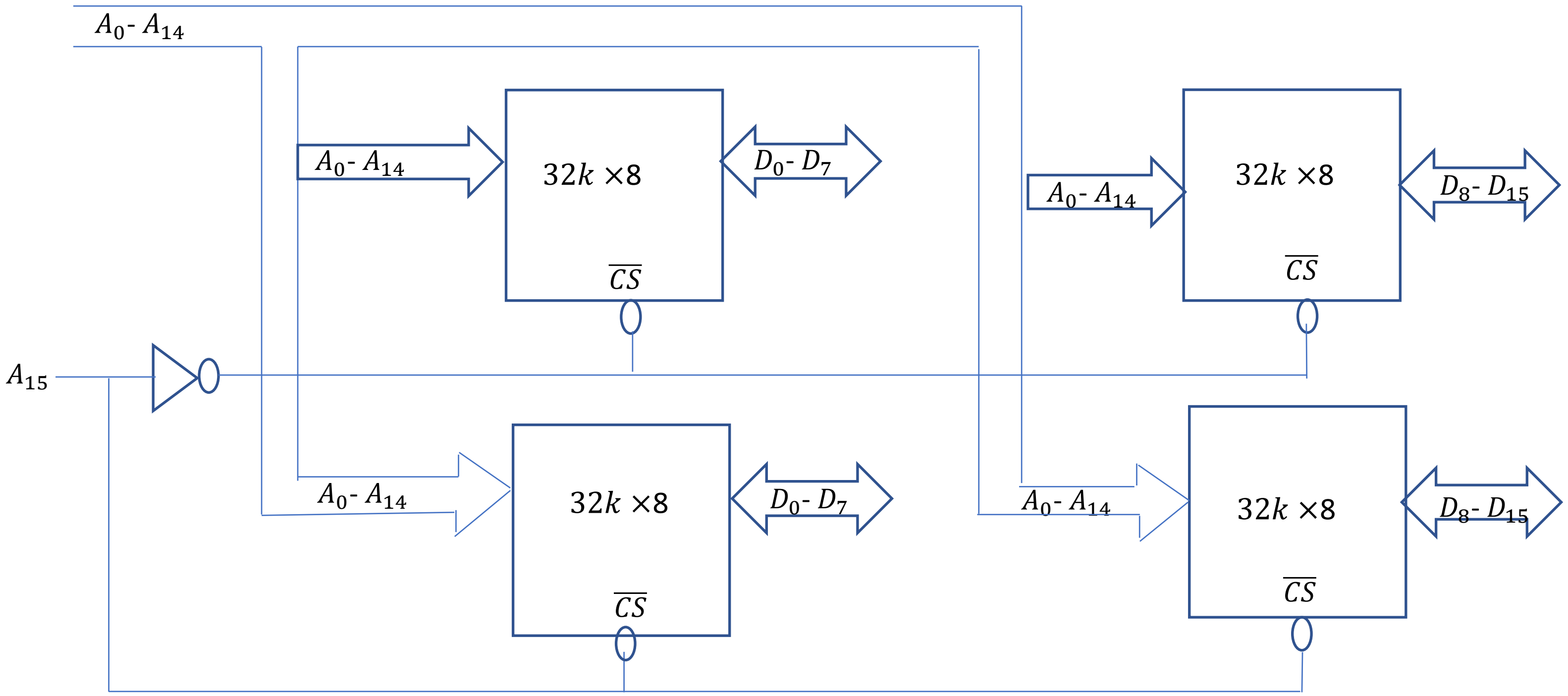


Figure (11)