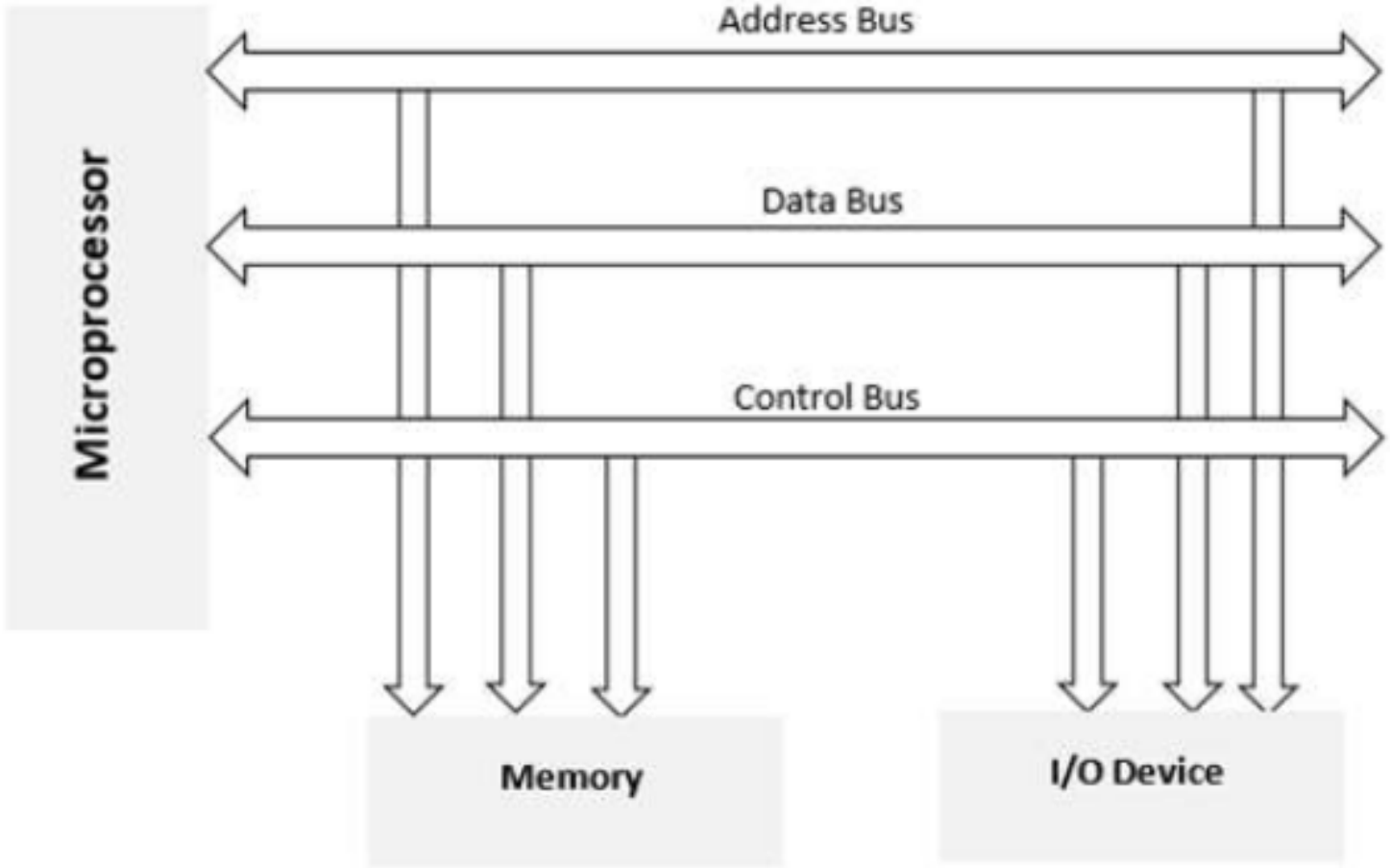
Memory and I/O interfacing

Department of Computer Science and Engineering(CSED), NIT Calicut.

Outline

Memory and I/O Interfacing

Several memory chips and I/O devices are connected to a microprocessor



Memory Interfacing

- Every microprocessor-based system has a memory system
- Two main types of memory: read-only memory (ROM) and random access memory (RAM) or read/write memory
- Read-only memory contains system software and permanent system data, while RAM contains temporary data and application software
- How to interface both memory types to the Intel family of microprocessors

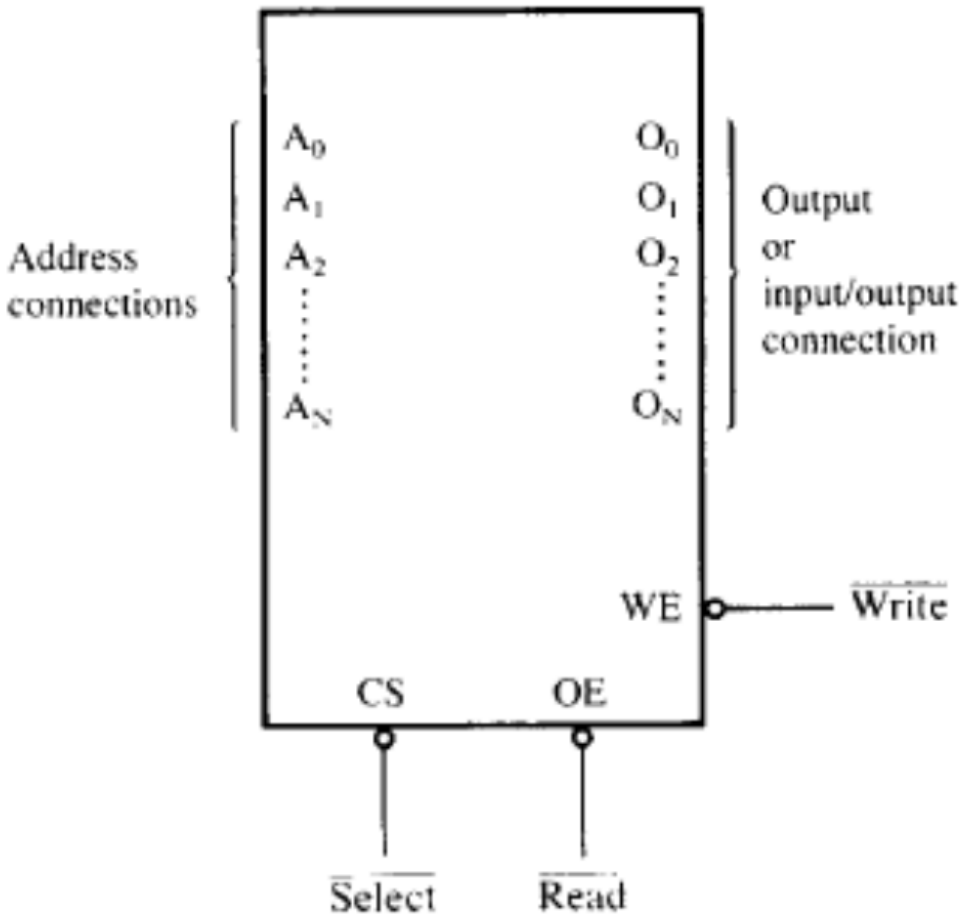
Memory Devices

- Read-only memory (ROM)
- Programmable Read-only memory (PROM)
- Erasable programmable Read-only memory (EPROM)
- Flash memory (EEPROM)
- Static random access memory (SRAM)
- Dynamic random access memory (DRAM)

Memory Pin Connections

- Pin connections common to all memory devices are:
  - the address inputs
  - data outputs or input/ outputs
  - some type of selection input, and
  - at least one control input used to select a read or write operation

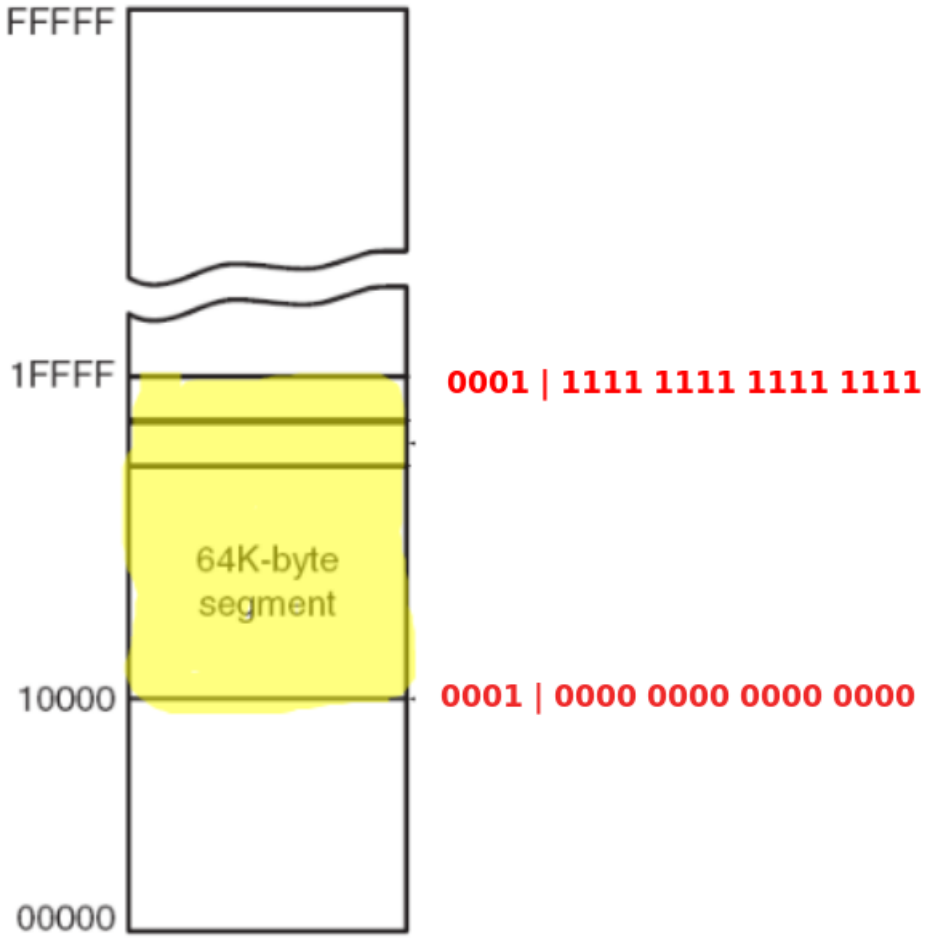
Memory Pin Connections



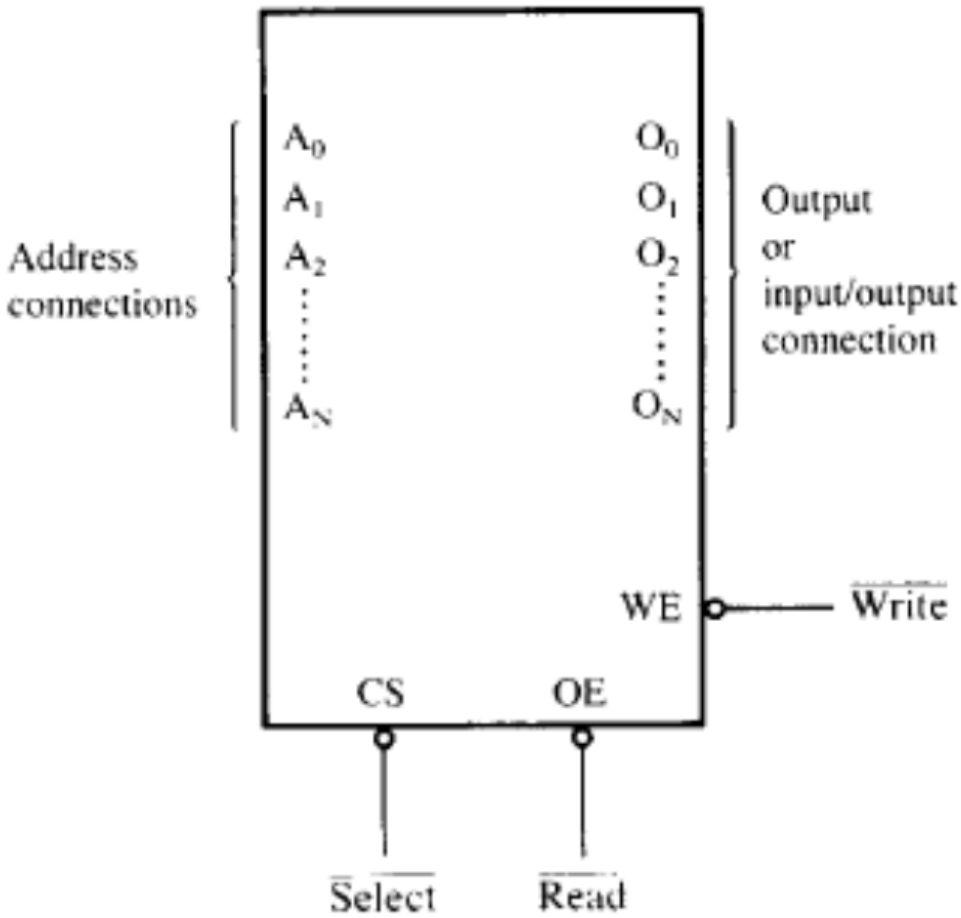
Address Connections

- All memory devices have address inputs that select a memory location within the memory device.
- Address inputs are almost always labeled from A_0 , the least significant address input, to A_n
- For example, a memory device with 10 address pins has its address pins labeled from A_0 to A_9
- The number of address pins found on a memory device is determined by the number of memory locations found within it
- A 1K memory (1024 memory locations) device has 10 address pins A_0 to A_9;
- It takes a 10-bit binary number (1024 different combinations) to select any single location on a 1024-location device
- A memory that starts at location 10000H and ends at location 1FFFFH is a ?-byte memory

A memory that starts at location 10000H and ends at location 1FFFFH is a ?-byte memory device



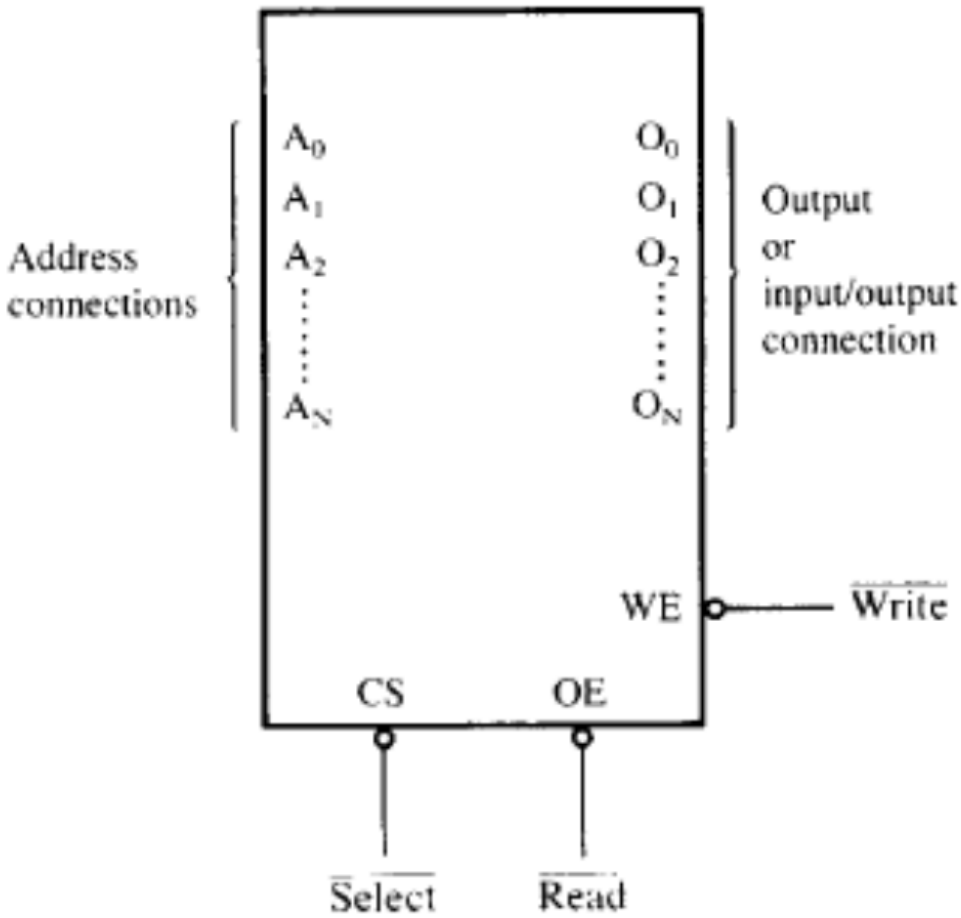
Memory Pin Connections



Data Connections

- All memory devices have a set of data outputs or input/outputs
- Data pins on memory devices are labeled D_0 through D_7 for an 8-bit-wide memory device
- For example, a memory device with 1K memory locations and 8 bits in each location is often listed as a 1K $\times$ 8 by the manufacturer.
- Memory devices are often classified according to total bit capacity.
- For example, a 1K $\times$ 8-bit memory device is sometimes listed as an 8K memory device, or a 64K $\times$ 4 memory is listed as a 256K device.

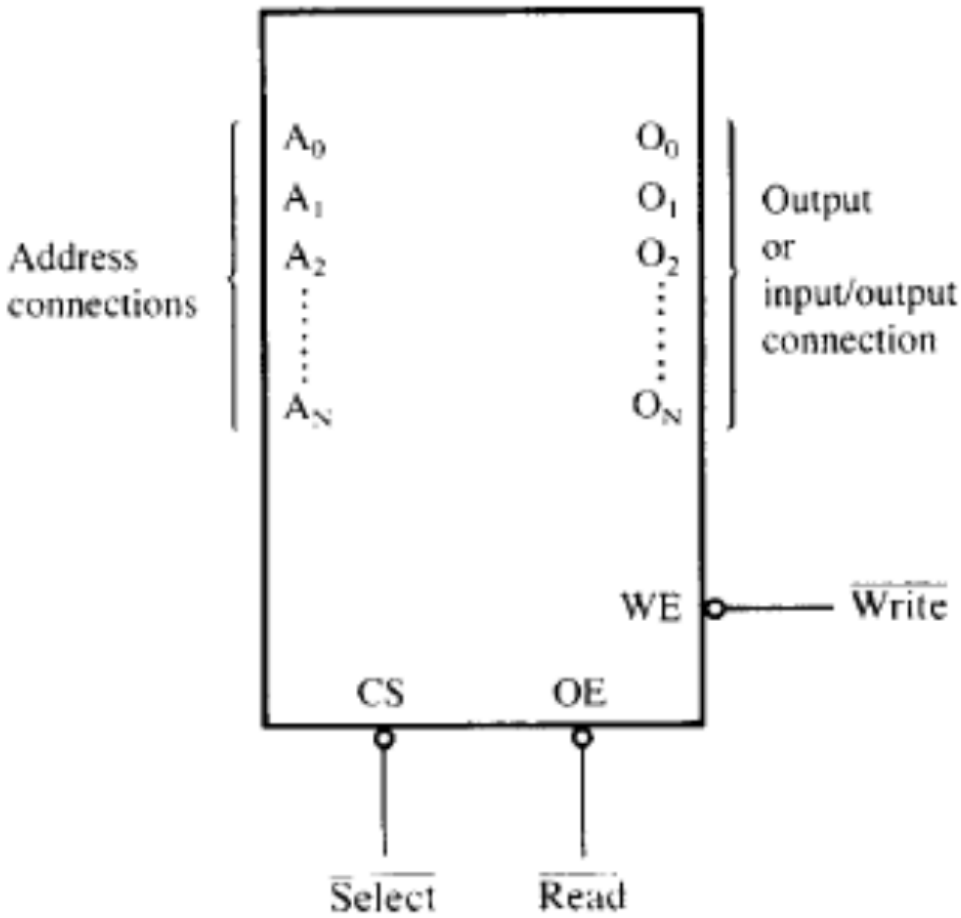
Memory Pin Connections



Selection Connections

- Each memory device has an input—sometimes more than one—that selects or enables the memory device.
- This type of input is most often called a chip select ($\overline{CS}$), chip enable ($\overline{CE}$), or simply select ($\overline{S}$) input.
- If the $\overline{CS}$, $\overline{CE}$, or $\overline{S}$ input is active (a logic 0, in this case, because of the overbar), the memory device performs a read or write operation; if it is inactive (a logic 1, in this case), the memory device cannot do a read or a write because it is turned off or disabled.
- If more than one $\overline{CS}$ connection is present, all must be activated to read or write data.

Memory Pin Connections



Control Connections

- All memory devices have some form of control input or inputs.
- A ROM usually has only one control input, while a RAM often has one or two control inputs.
- The control input most often found on a ROM is the output enable ($\overline{OE}$) or gate ($\overline{G}$) connection, which allows data to flow out of the output data pins of the ROM.
- If $\overline{OE}$ and the selection input ($\overline{CE}$) are both active, the output is enabled; if $\overline{OE}$ is inactive, the output is disabled
- A RAM memory device has either one or two control inputs.
- If there is only one control input, it is often called R/$\overline{W}$.
- This pin selects a read operation or a write operation only if the device is selected by the selection input ($\overline{CS}$).

Address decoding

- 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 of the processor, decoding is necessary.
- 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.

Why Decode Memory?

- Consider interfacing 8088 microprocessor to the 2716 EPROM
- The 8088 issues 20-bit addresses for a total of 1MB of memory address space.
- However, the BIOS on a 2716 EPROM has only 2KB of memory and 11 address pins
- This means that the microprocessor sends out a 20-bit memory address whenever it reads or writes data.
- Because the EPROM has only 11 address pins, there is a mismatch that must be corrected.

Why Decode Memory?

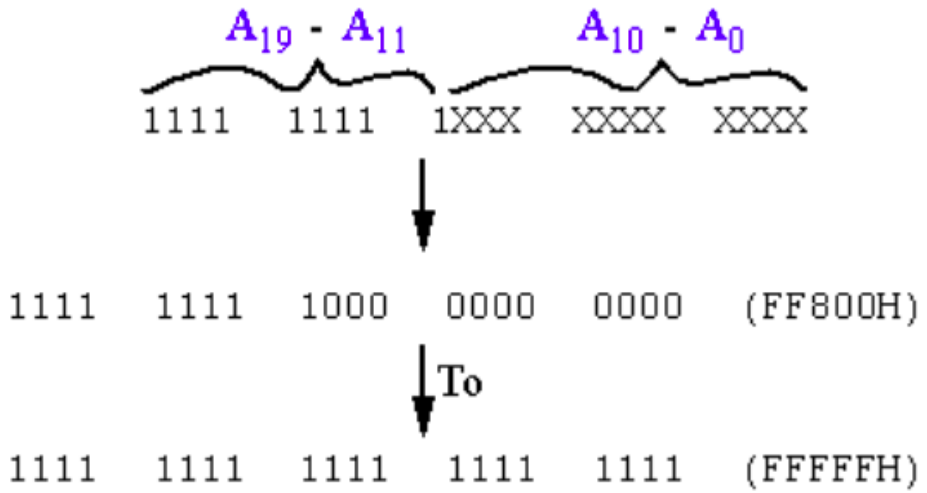
- A decoder can be used to decode the additional 9 address pins and allow the EPROM to be placed in any 2KB section of the 1MB address space.
- If only 11 of the 8088's address pins are connected to the memory, the 8088 will see only 2K bytes of memory instead of the 1M bytes that it "expects" the memory to contain.
- The decoder corrects the mismatch by decoding the address pins that do not connect to the memory component.

For decoding

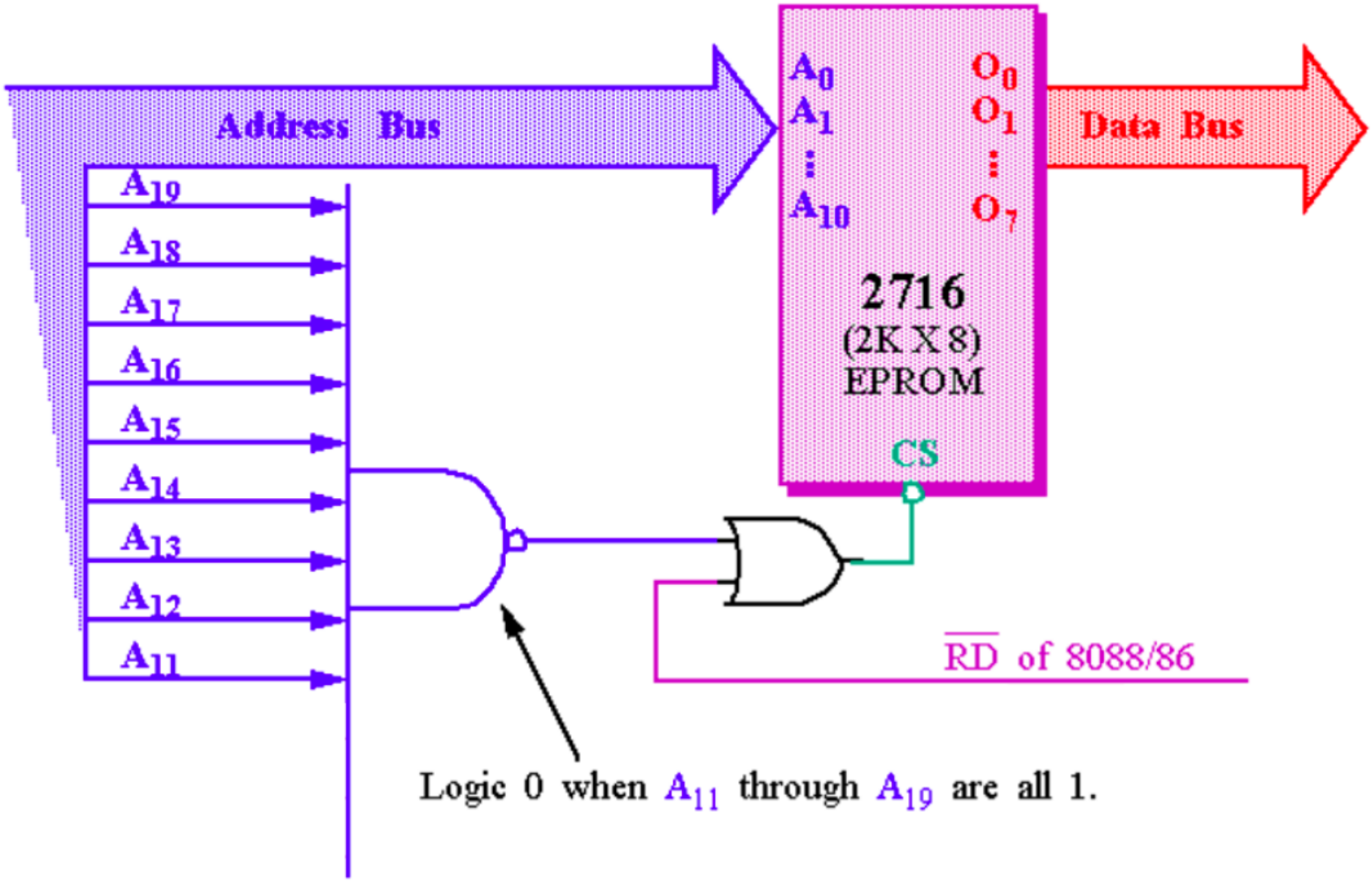
- Simple NAND Gate Decoder
- The 3-to-8 Line Decoder
- The Dual 2-to-4 Line Decoder
- PLD Programmable Decoders

Address space

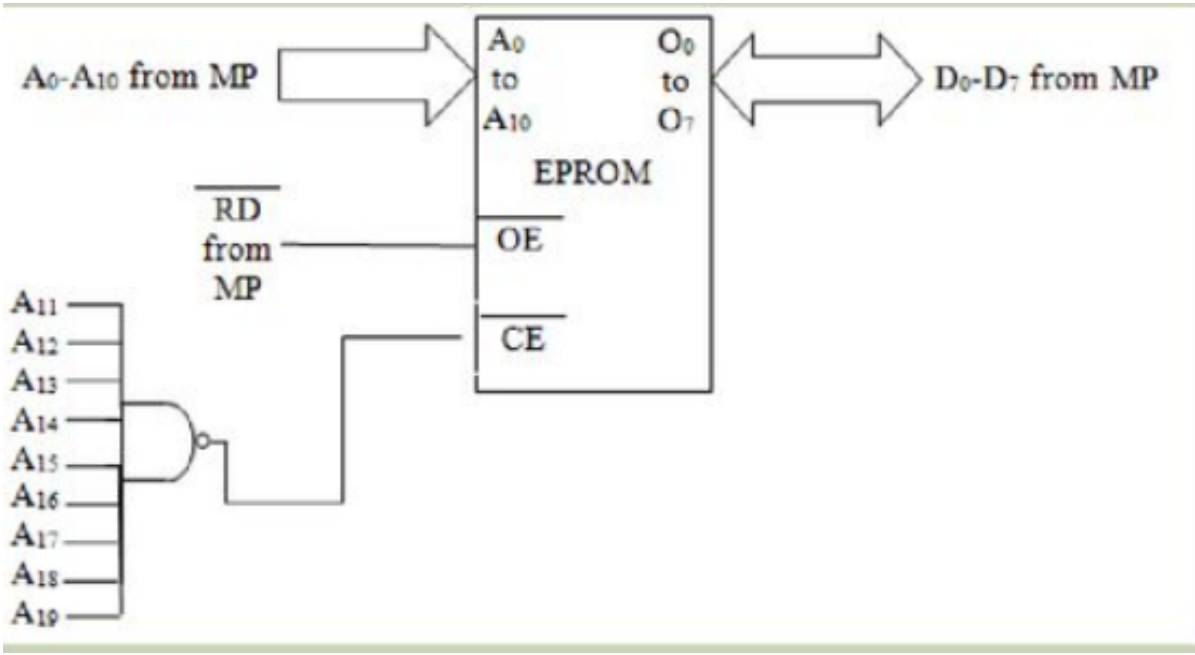
- *To determine the address range that a device is mapped into:*



Decoding (NAND)



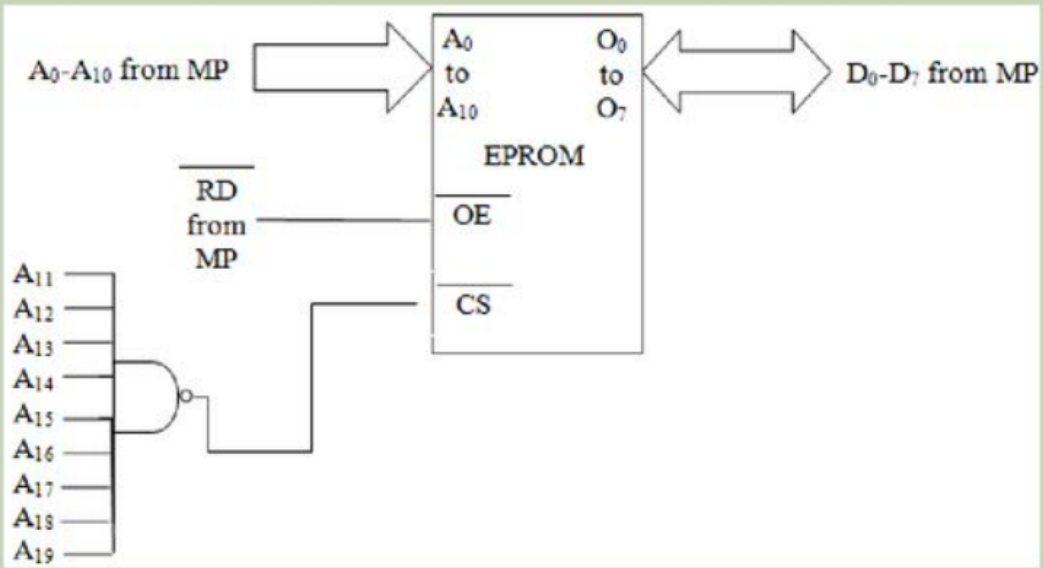
Decoding


A0-A10 from MP
A0
to
A10
O0
to
O7
D0-D7 from MP
EPROM
RD
from
MP
OE
CE
A11
A12
A13
A14
A15
A16
A17
A18
A19

Decoding

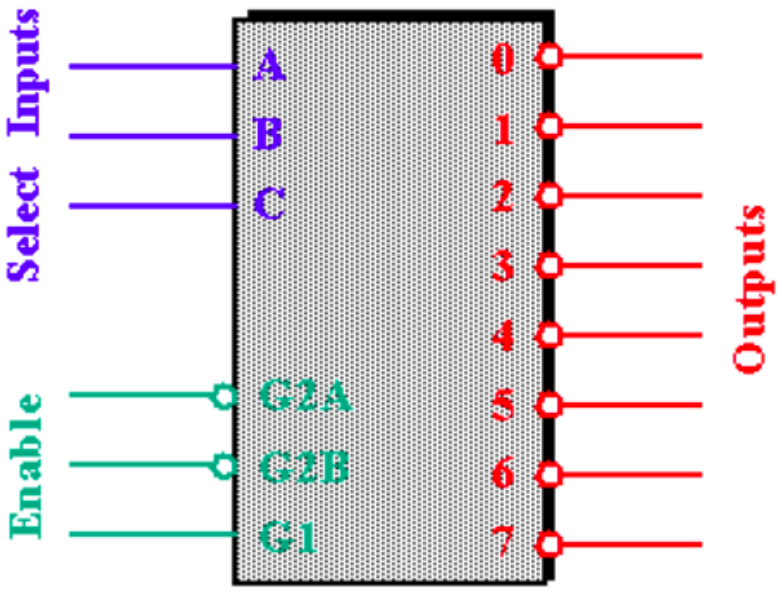
- Design a memory interface using a simple NAND gate decoder that selects a 2KB EPROM for memory location FF800H–FFFFFH.
- Solution:
- Memory Map

A19	A18	A17	A16	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	Address in Hex
1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	FF800H
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	FFFFFH



Decoding (Decoder)

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



Inputs						Output							
Enable			Select										
G2A	G2B	G1	C	B	A	0	1	2	3	4	5	6	7
1	X	X	X	X	X	1	1	1	1	1	1	1	1
X	1	X	X	X	X	1	1	1	1	1	1	1	1
X	X	0	X	X	X	1	1	1	1	1	1	1	1
0	0	1	0	0	0	0	1	1	1	1	1	1	1
0	0	1	0	0	1	1	0	1	1	1	1	1	1
0	0	1	0	1	0	1	1	0	1	1	1	1	1
0	0	1	0	1	1	1	1	1	0	1	1	1	1
0	0	1	1	0	0	1	1	1	1	0	1	1	1
0	0	1	1	0	1	1	1	1	1	1	0	1	1
0	0	1	1	1	0	1	1	1	1	1	1	0	1
0	0	1	1	1	1	1	1	1	1	1	1	1	0

Memory Interfacing

- Interface 512KB RAM to 8088 MP using 64KB RAM using 3:8 decoder with starting address of memory as 80000H. Clearly mention decoder logic and memory map.
- **Solution:**
- Calculate number IC (Chip) required
- Interface 512KB RAM using 64KB RAM so 8 IC of size 64KB
- Number of address lines required to connect each IC
- So for 64KB memory IC required 16 address lines i.e. $A_{15} - A_0$.
- Memory map to find address line to be connected to the decoder (NAND or 3:8 decoder) and also find the start and end memory address of each IC.

Memory Interfacing

A_{19}	A_{18}	A_{17}	A_{16}	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	Memory Address	
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	80000H	IC0
1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	8FFFFH	
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	90000H	IC1
1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	9FFFFH	
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	A0000H	IC2
1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	AFFFFH	
1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	B0000H	IC3
1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	BFFFFH	
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	C0000H	IC4
1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	CFFFFH	
1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	D0000H	IC5
1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	DFFFFH	
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	E0000H	IC6
1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	EFFFFH	
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	F0000H	
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	FFFFFH	IC7

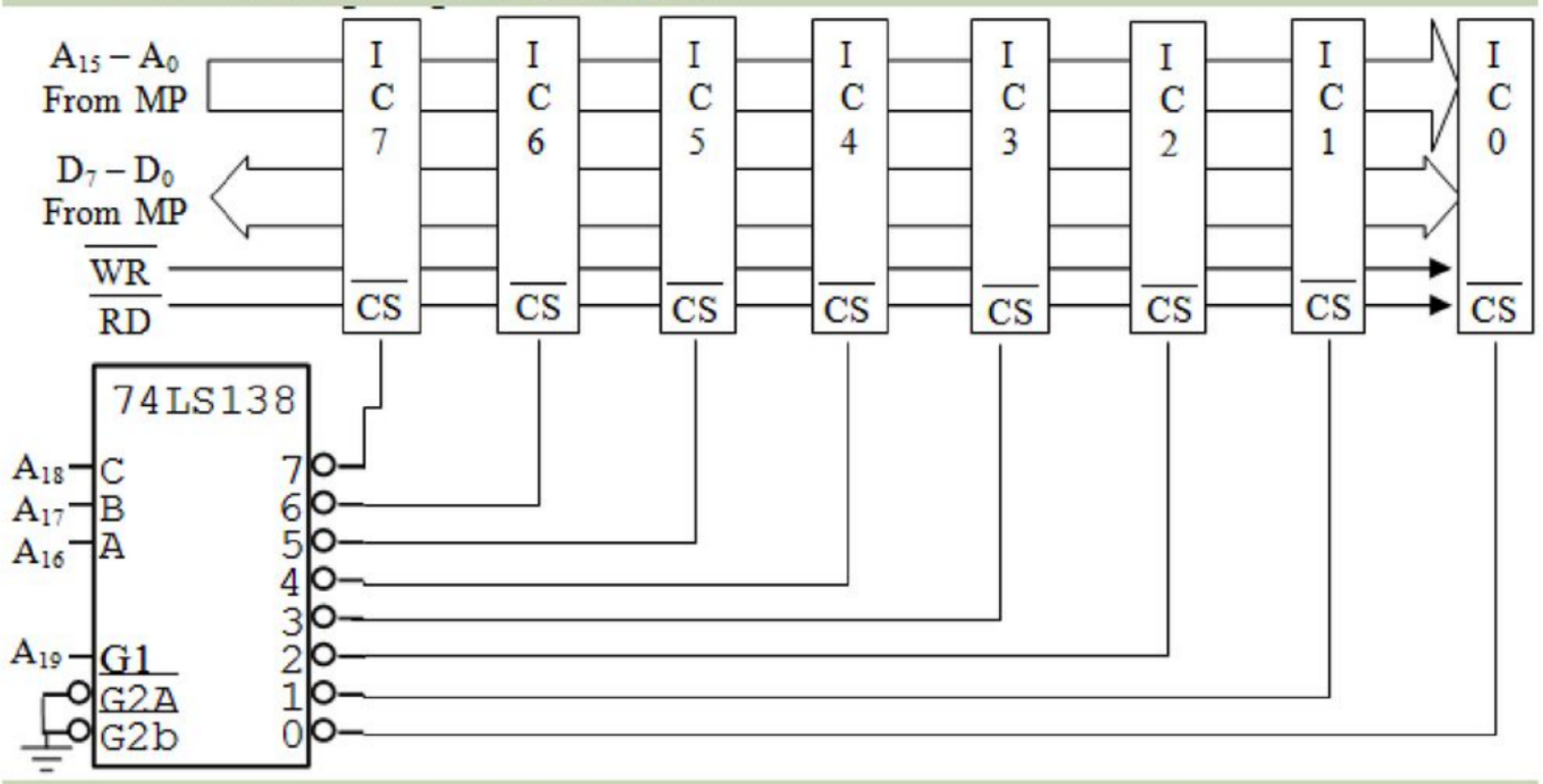
Connect to Decoder I/P
C,B,A

Connect to 64KB ICs

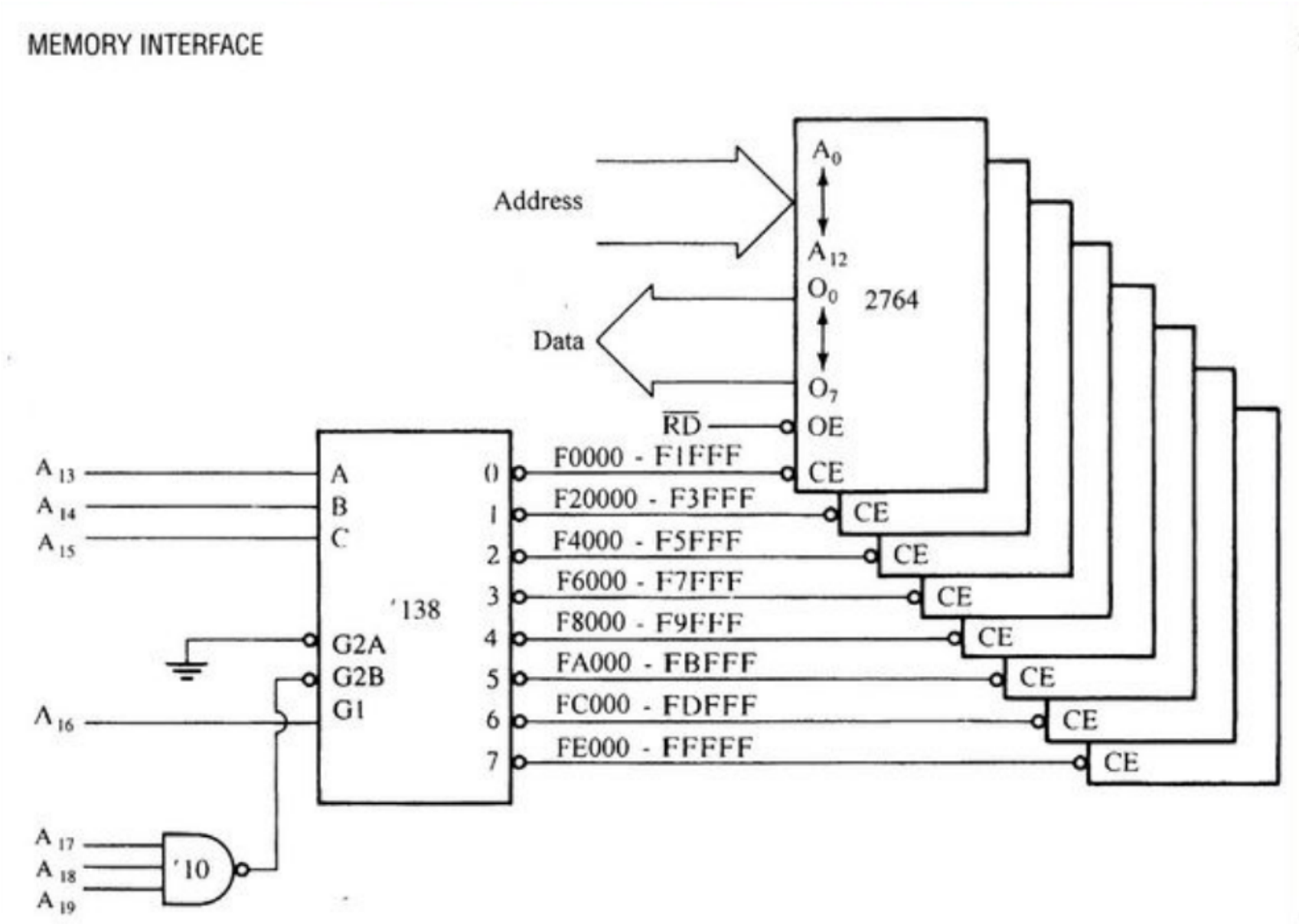
Connect to G1 of Decoder

Memory Interfacing

- Draw the interfacing diagram



Memory Interfacing



I/O interfacing

- A microprocessor is great at solving problems, but if it can't communicate with the outside world, it is of little worth.
- IO interfacing is the communication between various devices like keyboard, mouse, printer, etc
- Keyboard and displays are used as communication channel with outside world
- Therefore, it is necessary that we interface keyboard and displays with the microprocessor. This is called I/O interfacing.

I/O interfacing

- The instruction set contains one type of instruction that transfers information to an I/O device (OUT) and another to read information from an I/O device (IN).
- Instructions (INS and OUTS,found on all versions except the 8086/8088) are also provided to transfer strings of data between the memory and an I/O device.
- Instructions that transfer data between an I/O device and the microprocessor's accumulator (AL, AX, or EAX) are called IN and OUT.
- The I/O address is stored in register DX as a 16-bit I/O address

I/O interfacing

IN AL, DX	8	A byte is input into AL from the port addressed by DX
IN AX, DX	16	A word is input into AX from the port addressed by DX
OUT DX, AL	8	A byte is output from AL into the port addressed by DX
OUT DX, AX	16	A word is output from AX into the port addressed by DX

I/O interfacing

INSB	8	A byte is input from the port addressed by DX and stored into the extra segment memory location addressed by DI, then DI = DI ± 1
INSW	16	A word is input from the port addressed by DX and stored into the extra segment memory location addressed by DI, then DI = DI ± 2
OUTSB	8	A byte is output from the data segment memory location addressed by SI into the port addressed by DX, then SI = SI ± 1
OUTSW	16	A word is output from the data segment memory location addressed by SI into the port addressed by DX, then SI = SI ± 2

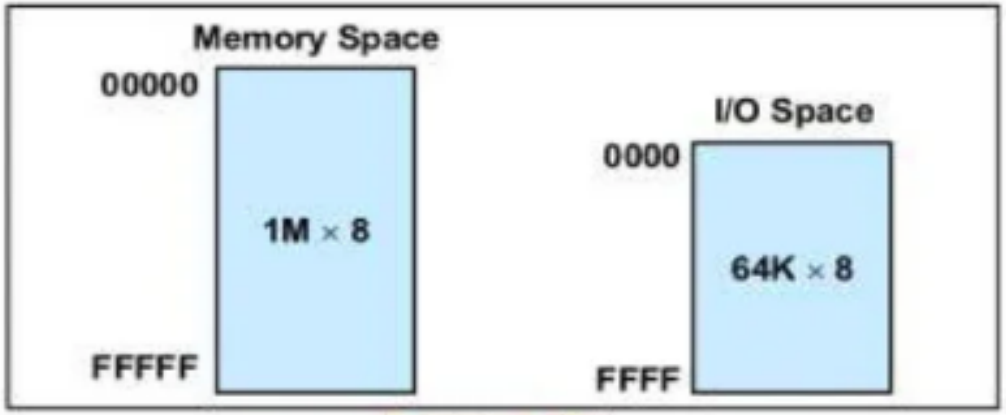
I/O interfacing

- Whenever data are transferred by using the IN or OUT instructions, the I/O address, often called a port number (or simply port), appears on the address bus.
- The external I/O interface decodes the port number in the same manner that it decodes a memory address.
- The INS and OUTS instructions address an I/O device by using the DX register, but do not transfer data between the accumulator and the I/O device as do the IN and OUT instructions.
- Instead, these instructions transfer data between memory and the I/O device.
- The memory address is located by ES:DI for the INS instruction and by DS:SI for the OUTS instruction.

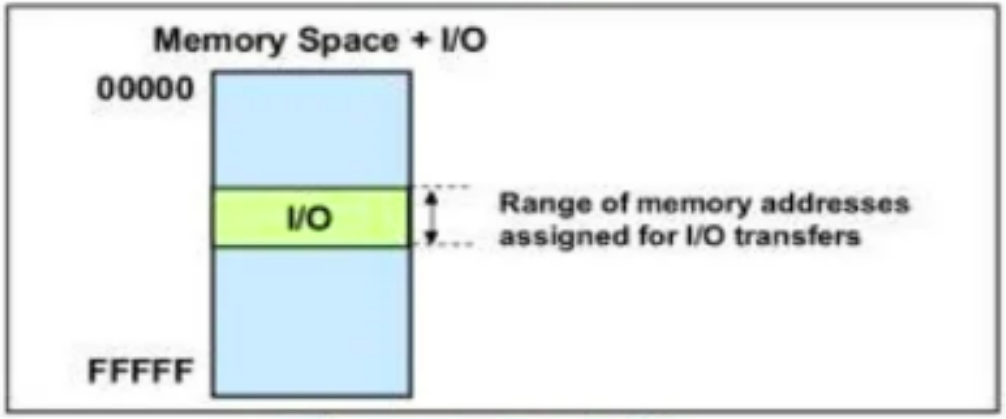
Isolated and Memory-Mapped I/O

- There are two different methods of interfacing I/O to the microprocessor: isolated I/O and memory-mapped I/O.
- In the isolated I/O scheme, the IN, INS, OUT, and OUTS instructions transfer data between the microprocessor's accumulator or memory and the I/O device.
- In the memory-mapped I/O scheme, any instruction that references memory can accomplish the transfer.

Isolated vs. Memory Mapped I/O



Isolated I/O



Memory-Mapped I/O

Isolated-Mapped I/O

- The most common I/O transfer technique used in the Intel microprocessor-based system is isolated I/O.
- The term isolated describes how the I/O locations are isolated from the memory system in a separate I/O address space.
- The addresses for isolated I/O devices, called ports, are separate from the memory. Hence, the user can expand the memory to its full size without using any of memory space for I/O devices.
- A disadvantage of isolated I/O is that the data transferred between I/O and the microprocessor must be accessed by the IN, INS, OUT, and OUTS instructions.
- Separate control signals for the I/O space are developed, which indicate an I/O read or an I/O write operation.

Memory-Mapped I/O

- Unlike isolated I/O, memory-mapped I/O does not use the IN, INS, OUT, or OUTS instructions.
- Instead, it uses any instruction that transfers data between the microprocessor and memory.
- A memory-mapped I/O device is treated as a memory location in the memory map.
- The main advantage of memory-mapped I/O is that any memory transfer instruction can be used to access the I/O device.
- The main disadvantage is that a portion of the memory system is used as the I/O map.
- This reduces the amount of memory available to applications.

I/O PORT ADDRESS DECODING

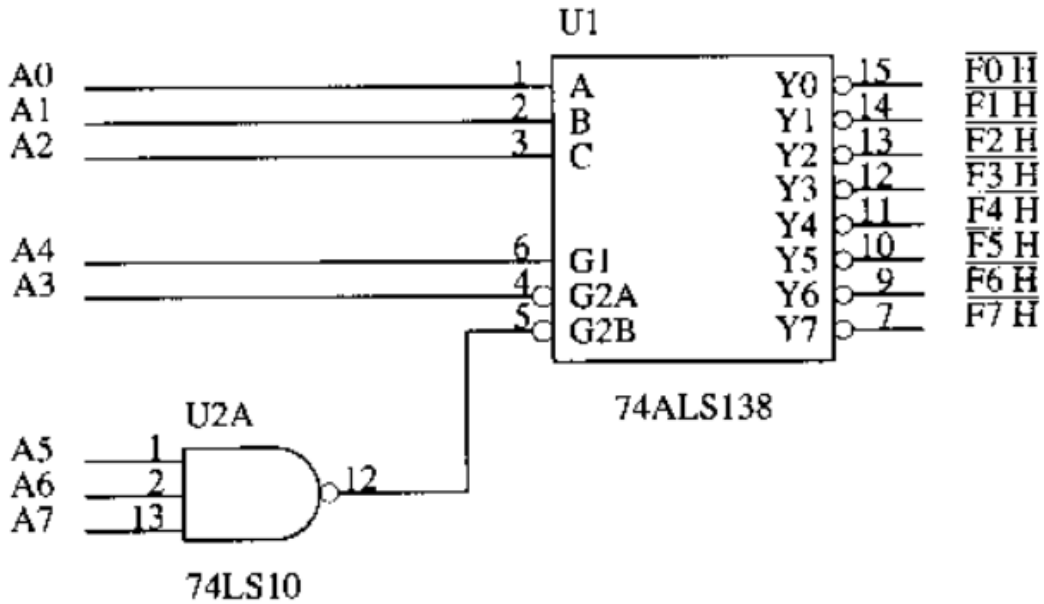
- I/O port address decoding is very similar to memory address decoding, especially for memory mapped I/O devices.
- we do not discuss memory-mapped I/O decoding because it is treated the same as memory
- The decision to use memory-mapped I/O is often determined by the size of the memory system and the placement of the I/O devices in the system
- The main difference between memory decoding and isolated I/O decoding is the number of address pins connected to the decoder.
- We decode A31–A0 , A23–A0 , or A19–A0 for memory, and A15–A0 for isolated I/O. Sometimes, if the I/O devices use only fixed I/O addressing, we decode only A7–A0.
- In the personal computer system, we always decode all 16 bits of the I/O port address.
- Another difference with isolated I/O is that $\overline{IORC}$ and $\overline{IOWC}$ activate I/O devices for a read or write operation

Decoding 8-Bit I/O Port Addresses

- The fixed I/O instruction uses an 8-bit I/O port address that appears on A15–A0 as 0000H–00FFH.
- If a system will never contain more than 256 I/O devices, we often decode only address connections A7–A0 for an 8-bit I/O port address.
- Thus, we ignore address connection A15–A8.
- Embedded systems often use 8-bit port addresses.
- Note that the DX register can also address I/O ports 00H–FFH.
- If the address is decoded as an 8-bit address, we can never include I/O devices that use a 16-bit I/O address.
- The personal computer never uses or decodes an 8-bit address.

Decoding 8-Bit I/O Port Addresses

FIGURE 11–10 A port decoder that decodes 8-bit I/O ports. This decoder generates active low outputs for ports F0H–F7H.




THANK YOU