MICROPROCESSOR

CSC 405

Subject Incharge

Dakshata Panchal

Assistant Professor

email: dakshatapanchal@sfit.ac.in

CSC 405 Microprocessor

Module 3

Peripherals and their interfacing with 8086

Contents as per syllabus

- Memory Interfacing - RAM and ROM
- Decoding Techniques – Partial and Absolute
- 8255-PPI – Block diagram, Functional PIN Diagram, CWR, operating modes, interfacing with 8086.
- 8253 PIT - Block diagram, Functional PIN Diagram, CWR, operating modes, interfacing with 8086.
- 8257-DMAC – Block diagram, Functional PIN Diagram Register organization, DMA operations and transfer modes

Main Memory

- Interfaces directly to microprocessor
- **Characteristics:**
  - Any location can be accessed at random
  - Each byte has a unique address
  - Data is read or written in one machine cycle
- Types – RAM and ROM

Main Memory

RAM

- Volatile
- Read and Write
- Every bit is a flip-flop.
- It loses data when power is cut off.

ROM

- Nonvolatile
- Read Only
- Made of floating gate transistors.
- Special hardware (UV light) is required to write, so it cannot lose data when power is cut off.

Types of ROMs

1. **Mask-Programmable ROMs**
   - Works like truth table generator
   - Diode connections are programmed at the factory according to truth table provided by user
   - E.g., home appliances

2. **Field-Programmable ROMs (PROM)**
   - Fusible-link PROM
   - UV-light Erasable PROM (EPROM)
   - Electrically Erasable PROM (EEPROM)
   - E.g., computers, phones, etc.

EPROMs

- Can be programmed by user
- Can be erased using UV light
- EPROM series by Intel are
- Each chip contains
  - *n* Address pins
  - 8 Data pins
  - Chip Enable ($\overline{CE}$)
  - Output Enable ($\overline{OE}$)

EPROM chips

IC Name	Memory size (in Bytes)	Size in Kilo Bits
2704	512B	4 Kb
2708	1KB	8 Kb
2716	2KB	16 Kb
2732	4KB	32 Kb
2764	8KB	64 Kb
27128	16KB	128 Kb

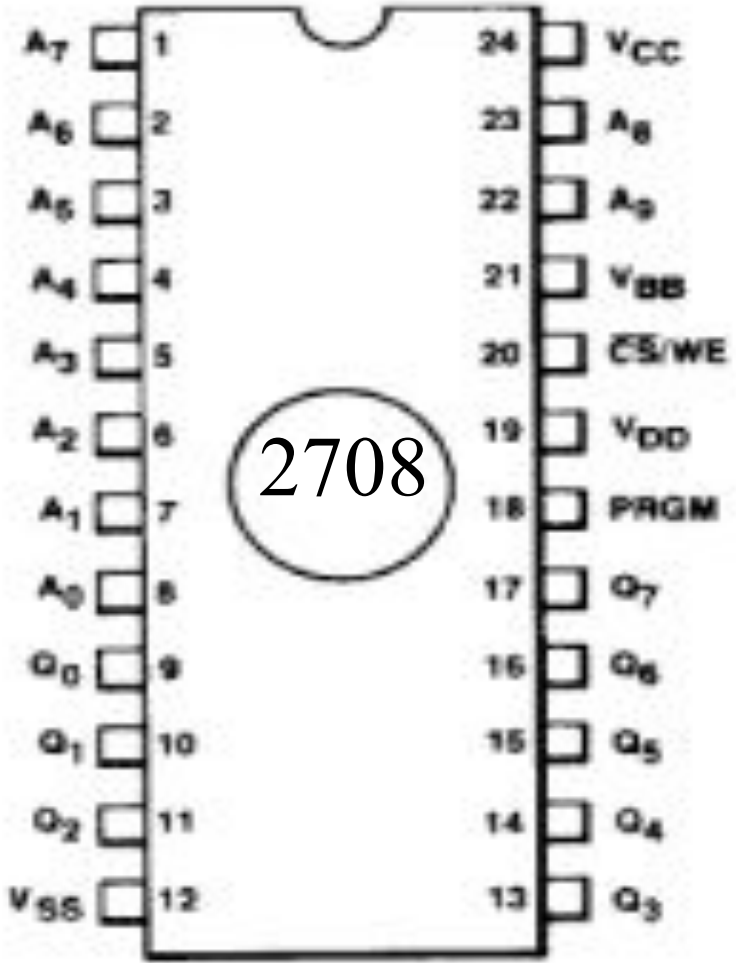
Calculation:

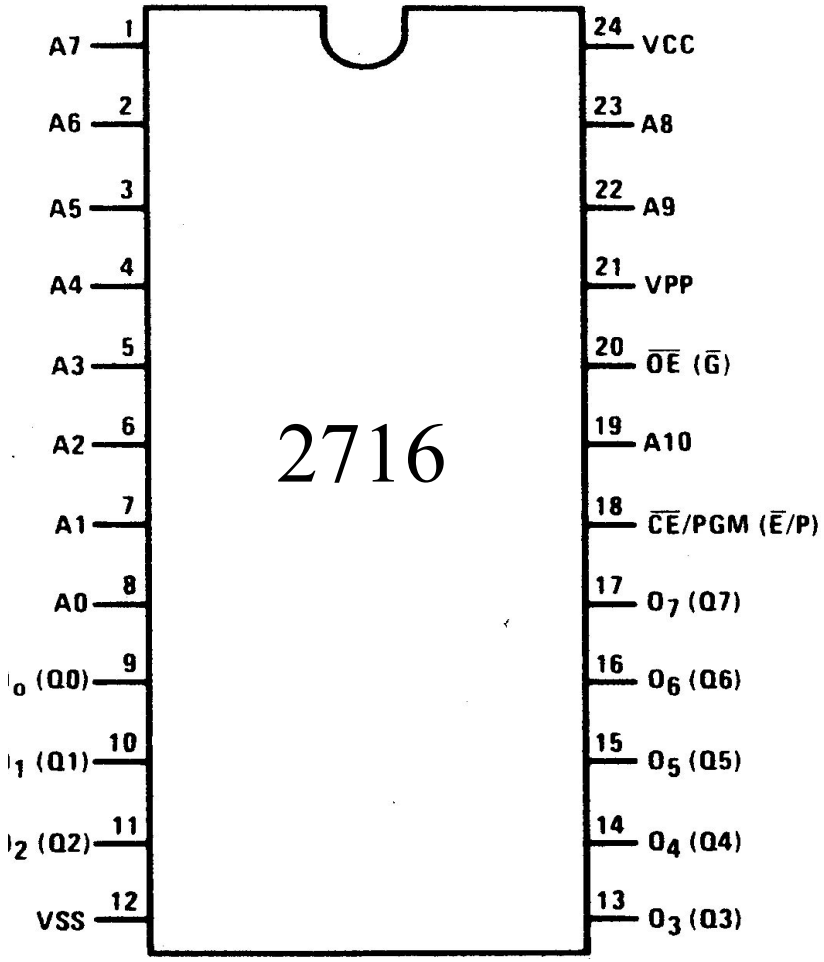
27xx → xx/8 = size in KB (xx is size in Kilo bits)

Example:

2764 → 64/8 = 8 KB (64 Kilobits)

System Designing with 8086





Types of RAM

1. **Static RAM (SRAM)**

2. **Dynamic RAM (DRAM)**

SRAM

- 4-6 transistors are connected to form a simple RS flip-flop
- **Advantages:**
  - Ease of interfacing
  - Fast access times
  - No timing circuits required
- **Disadvantage:**
  - Low-bit-density component : requires six transistor per bit
- Example: cache memory

DRAM

- Requires only a single bit select transistor and a capacitor
- **Advantages:**
  - High-bit-density component
  - Less power requirement
- **Disadvantage:**
  - Necessitates a refresh operation once every 2ms
  - Complex interfacing – requires a DRAM Controller (IC8203).
- Example: RAM chips

RAM chips

IC Name	Memory size (in Bytes)	Size in Kilo Bits
6208	1KB	8 Kb
6216	2KB	16 Kb
6232	4KB	32 Kb
6264	8KB	64 Kb
62128	16KB	128 Kb
62256	32KB	256 Kb

Calculation:

62xx → xx/8 = size in KB (xx is size in Kilo bits)

Example:

6264 → 64/8 = 8 KB (64 Kilobits)

Microprocessor System

- Includes memory and I/O devices
- Can communicate only with one device at a time (because of common bus)and hence requires address decoders
- Every location, on every device has a unique address.

How??

Address Decoding

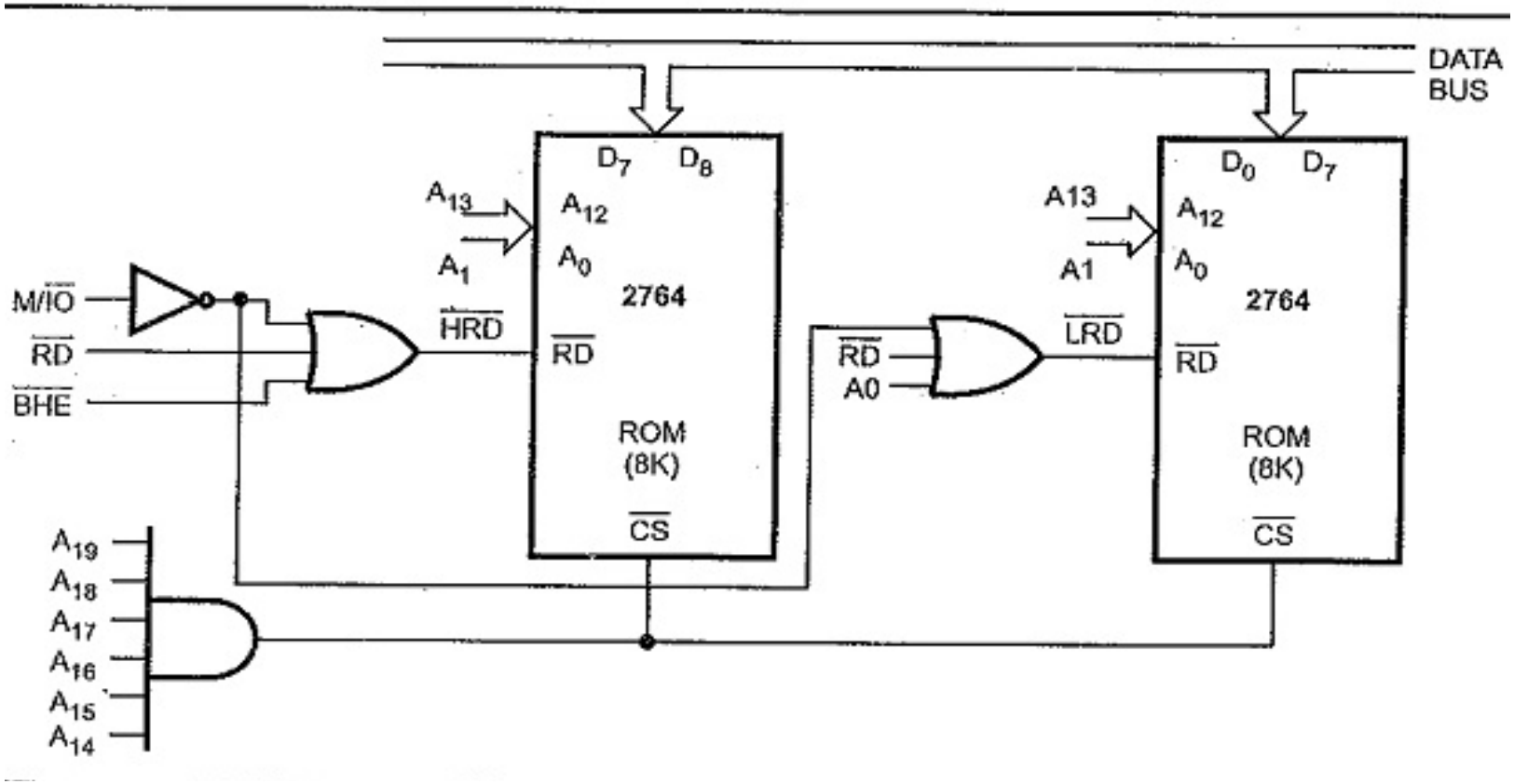
- 8086 can access up to 1 MB addresses
- But we may not use this entire range some address lines are unused
- **Address Decoder:** examines address lines and enables memory
- 3 decoding techniques:
  - Absolute (Full) Decoding
  - Linear(Partial) Decoding
  - Block Decoding

Absolute (Full) Decoding

- Memory chip is selected only for a specified logic level on address lines
- **Problem1:** Design an absolute address for 16K EPROM interface for 8086 such that the memory begins at 80000H
- **Problem2: :** Design an absolute address for 8K EPROM interface for 8086 such that the memory begins at FE000H

System Designing with 8086

Absolute Decoding



Partial (Linear) Decoding

- In small systems , much of the memory space is unused and hence don't require absolute decoding
- In this technique, some of the address lines are simply ignored.
- For 16K example, A18-A14 are ignored

Partial (Linear) Decoding

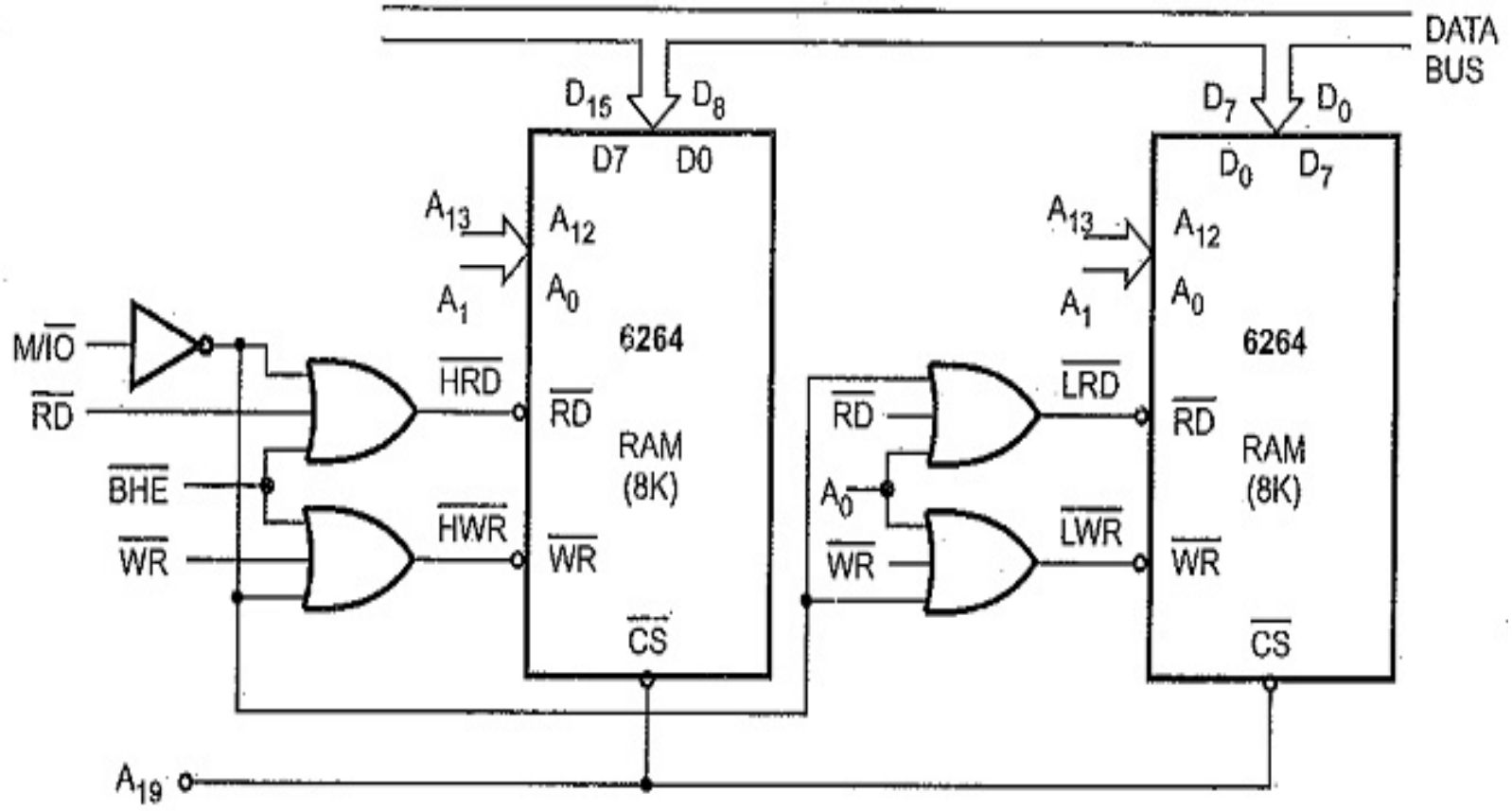
- Here chip will be enabled whenever A19 is high.
- This correspond to any address from 80000H to FFFFFH
- **Drawback:** The same location is accessed for addresses 80000,84000,88000 …(multiple addresses or shadow addresses)

  (A18-A14 are ignored 2^5 (32) addresses for a location)

  Uses : It is used in embedded systems, where memory configuration is fixed

System Designing with 8086

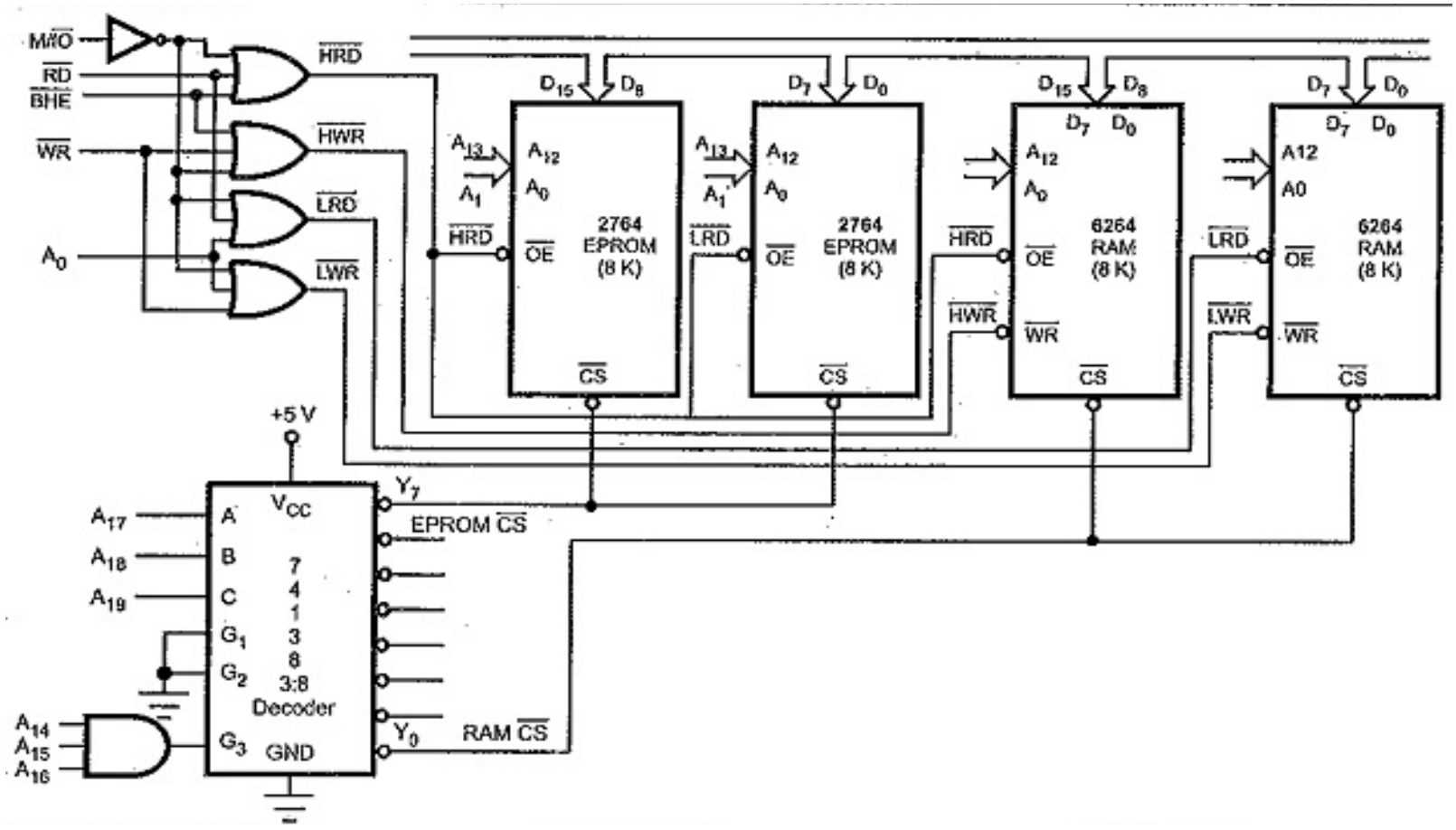
Linear Decoding



Block Decoding

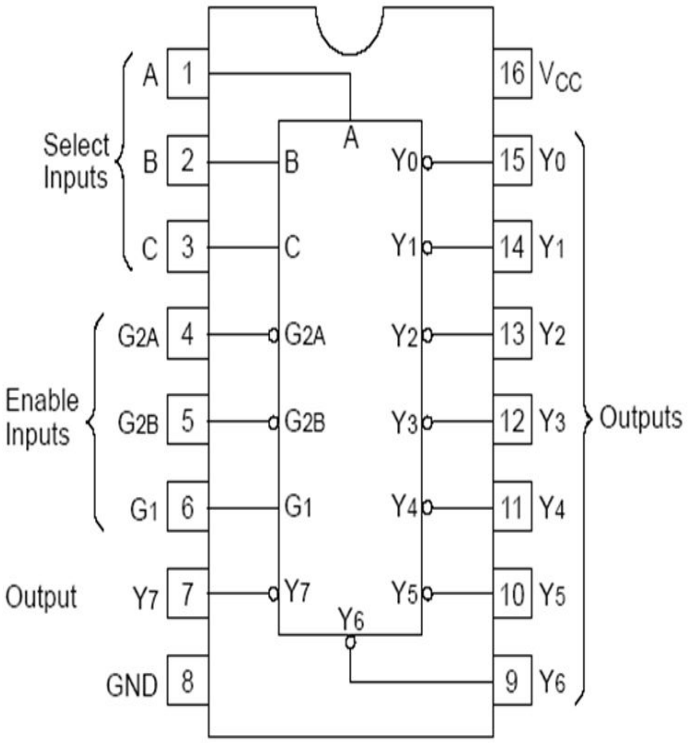
- A special decoder IC is used to generate chip select signal for each of the several blocks of memory .
- E.g. 74138 (3:8 Decoder)

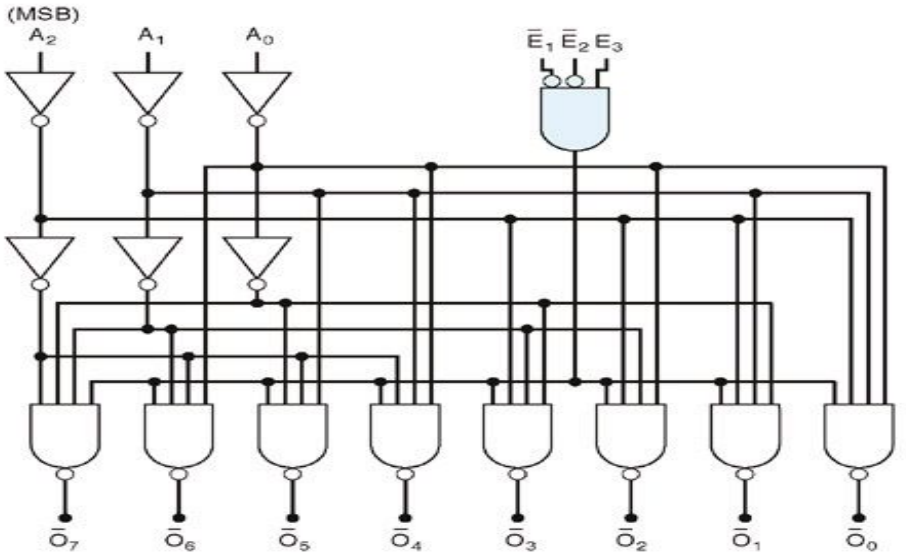
Block Decoding



System Designing with 8086

74138

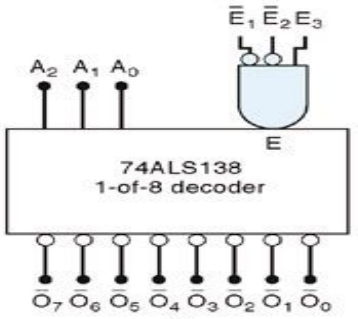




(a)

$\bar{E}_1$	$\bar{E}_2$	E_3	Outputs
0	0	1	Respond to input code $A_2A_1A_0$
1	X	X	Disabled – all HIGH
X	1	X	Disabled – all HIGH
X	X	0	Disabled – all HIGH

(b)



(c)

Exercise 1

Design an 8086 based system with the following specifications

1. 8086 in minimum mode
2. 64KB EPROM
3. 64KB RAM

Draw the complete schematic of the design indicating address map

RAM Address Decoding Table

M/m	Address	A19	A18	A17	A16	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
RAM Even	00000H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0FFFEH	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
RAM Odd	00001H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
	0FFFFH	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

RAM Address Decoding Table

M/m	Address	A19	A18	A17	A16	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
ROM	F0000H	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Even	FFFFEH	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
ROM	F0001H	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Odd	FFFFFH	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Example 2

Design an 8086 based system with the following specifications

1. 8086 working at 8 MHz in maximum mode
2. 32 KB EPROM using 16 KB chips
3. 16 KB SRAM using 8 KB chips

Explain design along with memory address map **(MAY 15)**

System Designing with 8086

Given

- Processor working at 8 MHz
  - Oscillating frequency of crystal = 8x3 = 24MHz
- EPROM - 32KB using 16KB chips
  - No of chips required = 32/16 = 2 chips (1 set)
- SRAM - 16KB using 8KB chips
  - No of chips required = 16/8 = 2 chips (1 set)

EPROM

- EPROM – Ending address is FFFFFH
- 16KB requires 14 address lines
  - 16KB = 2^{10} (1KB) * 2^4 (16) = 10+4 = 14 lines
- Since 14 lines are required, set A0-A13 to 1 and remaining lines 0. We get, 03FFFH (16KB)
- Starting addr of EPROM = Ending addr – size

  Starting addr of EPROM = FFFFFH – 03FFFH

  Starting addr of EPROM = FC000H

SRAM

- SRAM – Starting address is 00000H
- 8KB requires 13 address lines
  - 8KB = 2^{10} (1KB) * 2^{3} (8) = 10+3 = 13 lines
- Since 13 lines are required, set A0-A12 to 1 and remaining lines 0. We get, 01FFFH (8KB)
- Ending addr of RAM = Starting addr + size

  Ending addr of RAM = 00000H + 01FFFH

  Ending addr of RAM = 01FFFH

Address Lines

- EPROM requires 14 address lines
  - A1-A14 will be used
- SRAM requires 13 address lines
  - A1-A13 will be used
- A0 and BHE are used for selecting odd or even bank

Memory		Starting Address	Ending Address
RAM (set 1)	Even Bank	00000 H	01FFE H
	Odd Bank	00001 H	01FFF H
ROM (set 1)	Even Bank	FC000 H	FFFFE H
	Odd Bank	FC001 H	FFFFF H

Address Decoding Table

Memory Chip			Address																	
			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
RAM (set 1)	E	SA=00000H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	B	EA=03FFEH	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0
	O	SA=00001H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
	B	EA=03FFFH	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
ROM (set 1)	E	SA=F8000H	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	B	EA=FFFFEH	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
	O	SA=F8001H	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
	B	EA=FFFFFH	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Example 3

Design an 8086 based system with the following specifications

1. 8086 working at 10 MHz in minimum mode
2. 32 KB EPROM using 8 KB chips
3. 16 KB SRAM using 4 KB chips

Explain design along with memory address map **(DEC 14)**

System Designing with 8086

Given

- Processor working at 10 MHz
  - Oscillating frequency of crystal = 10x3 = 30MHz
- EPROM - 32KB using 8KB chips
  - No of chips required = 32/8 = 4 chips (2 sets)
- SRAM - 16KB using 4KB chips
  - No of chips required = 16/4 = 4 chips (2 sets)

EPROM 1

- EPROM set 1 – Ending address is FFFFFH
- 8KB requires 13 address lines
  - 16KB = 2^{10} (1KB) * 2^{3} (8) = 10+3 = 13 lines
- Since 14 lines are required, set A0-A12 to 1 and remaining lines 0. We get, 01FFFH (16KB)
- Starting addr of EPROM = Ending addr – size

  Starting addr of EPROM = FFFFFH – 01FFFH

  Starting addr of EPROM = FE000H

EPROM 2

- EPROM set 2 – Ending address is FDFFFH
- 8KB requires 13 address lines
  - 16KB = 2^{10} (1KB) * 2^3 (8) = 10+3 = 13 lines
- Since 14 lines are required, set A0-A12 to 1 and remaining lines 0. We get, 01FFFH (16KB)
- Starting addr of EPROM = Ending addr – size

  Starting addr of EPROM = FDFFFH – 01FFFH

  Starting addr of EPROM = FC000H

SRAM 1

- SRAM set 1 – Starting address is 00000H
- 8KB requires 13 address lines
  - 8KB = 2^{10} (1KB) * 2^{2} (4) = 10+2 = 12 lines
- Since 13 lines are required, set A0-A11 to 1 and remaining lines 0. We get, 00FFFH (8KB)
- Ending addr of RAM = Starting addr + size

  Ending addr of RAM = 00000H + 00FFFH

  Ending addr of RAM = 00FFFH

SRAM 2

- SRAM set 2 – Starting address is 01000H
- 8KB requires 12 address lines
  - 8KB = 2^{10} (1KB) * 2^{2} (4) = 10+2 = 12 lines
- Since 13 lines are required, set A0-A11 to 1 and remaining lines 0. We get, 00FFFH (8KB)
- Ending addr of RAM = Starting addr + size

  Ending addr of RAM = 01000H + 00FFFH

  Ending addr of RAM = 01FFFH

System Designing with 8086

Memory Chip			Address																			
			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
RAM (set 1)	E	SA=00000H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	B	EA=00EFEH	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0
	O	SA=00001H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
	B	EA=00FFFH	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1
RAM (set 2)	E	SA=01000H	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
	B	EA=01FFEH	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0
	O	SA=01001H	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1
	B	EA=01FFFH	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
ROM (set 2)	E	SA=FC000H	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	B	EA=FDFFEH	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	0
	O	SA=FC001H	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
	B	EA=FDFFFH	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
ROM (set 1)	E	SA=FE000H	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
	B	EA=FFFFEH	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
	O	SA=FE001H	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1
	B	EA=FFFFFH	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

8086 Designing (example discussed)

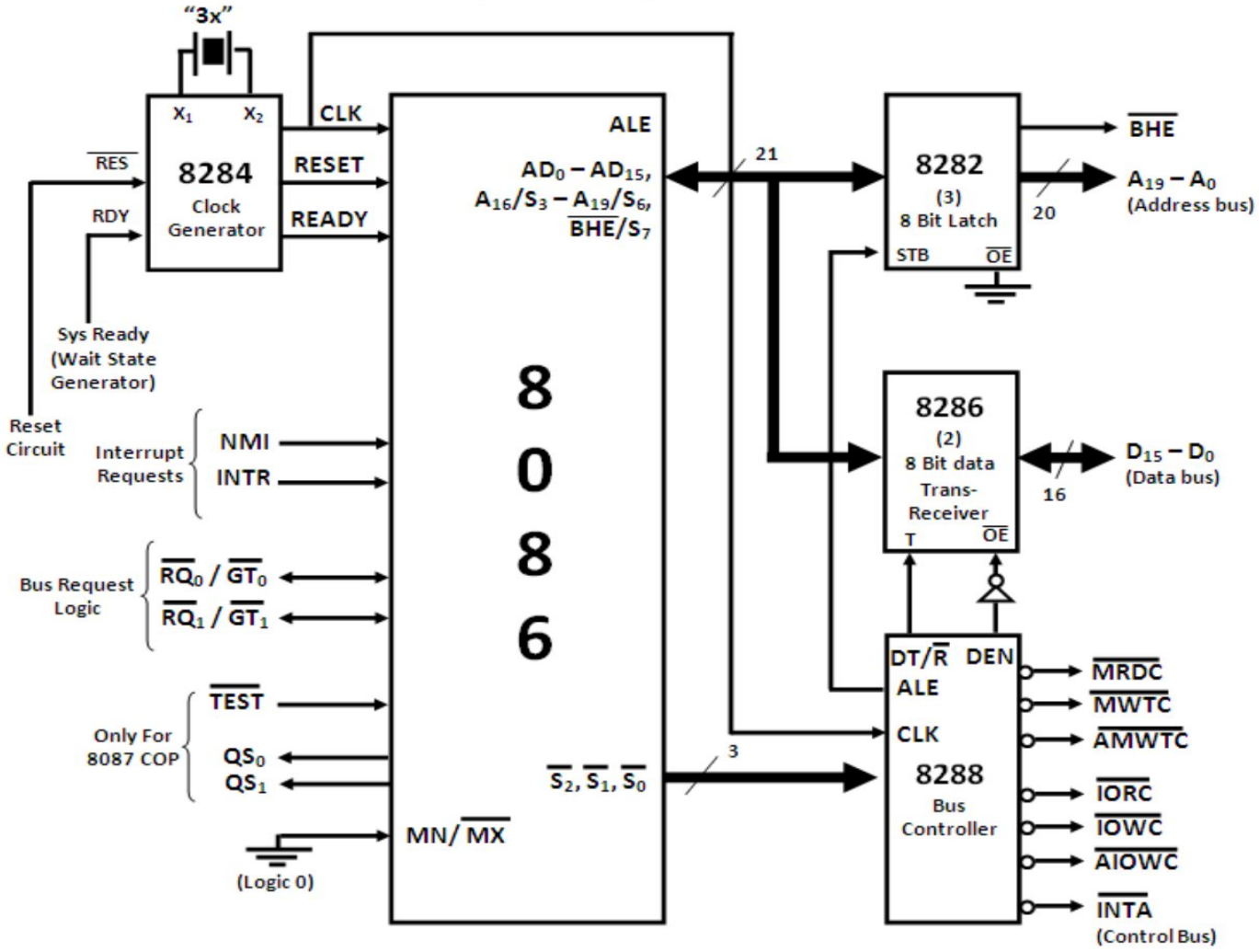
Design an 8086 based system with the following specifications

1. 8086 working at 6 MHz in minimum mode
2. 32 KB EPROM using 16 KB chips
3. 128 KB RAM using 32KB chips

Explain design along with memory address map **(DEC 14)**

8086 Designing (example discussed)

Soln: Show 8086 max mode config with a crystal of 18 MHZ.



8086 Designing (example discussed)

Memory Calculations:

EPORM:

Required = 32 KB, Available = 16 KB

No. of chips = 2 chips.

Starting address of EPROM is calculated as:

FFFFFH – (Space required by total EPROM of 32 KB)

```
  F F F F F H
-   7 F F F H
  F 8 0 0 0 H
```

Size of a single EPROM chip = 16 KB

$= 16 \times 1KB = 2^4 \times 2^{10}$
$= 2^{14}$
= 14 address lines

= **(A 14 ... A1)**

RAM:

Required = 128 KB, Available = 32 KB

No. of chips = 4 chips.

Starting address of RAM is: 00000H

Size of a single RAM chip = 32 KB

$= 32 \times 1 \text{ KB} = 2^5 \times 2^{10}$
$= 2^{15}$
= 15 address lines

= **(A15 ... A1)**

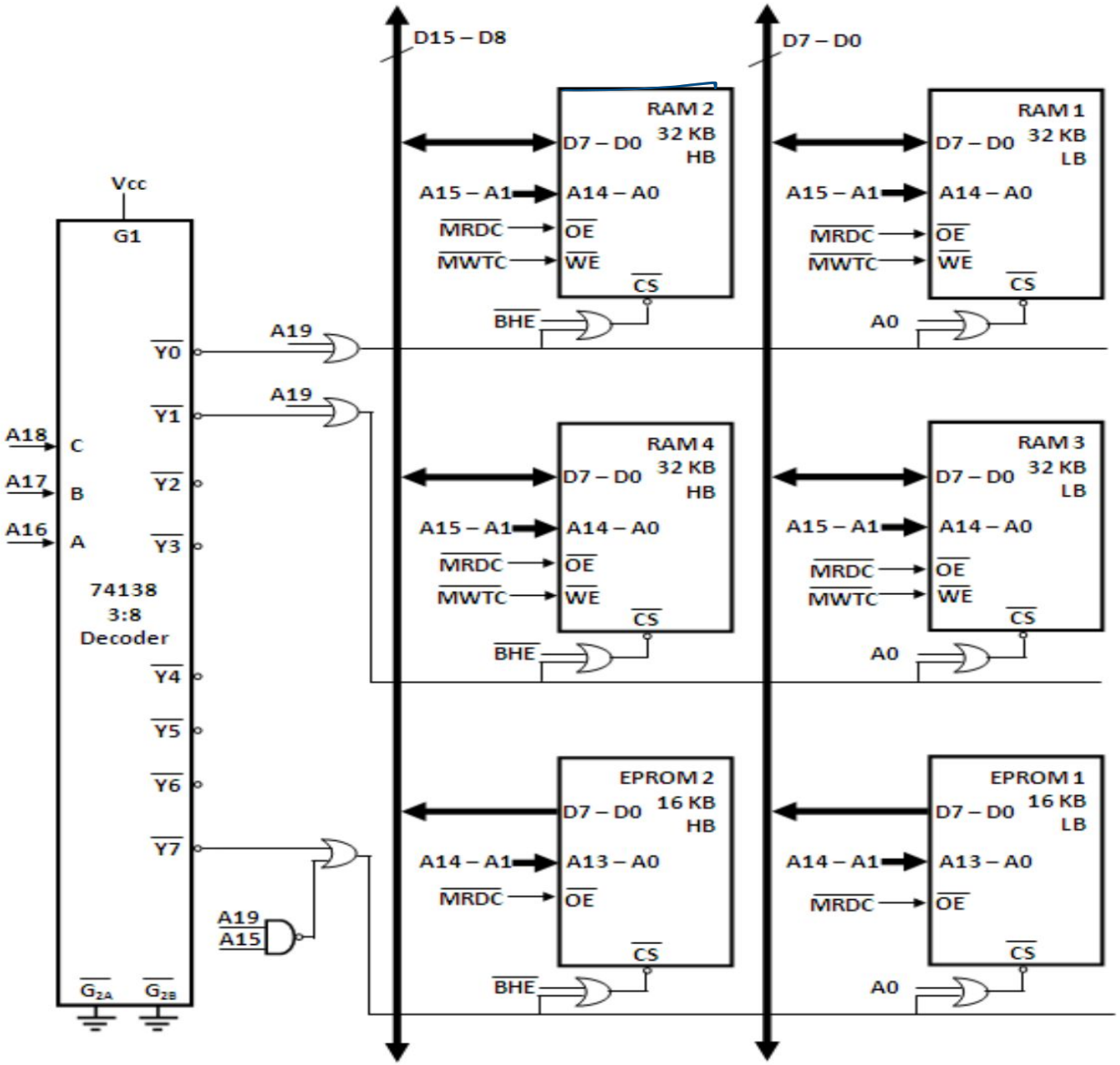
8086 Designing (example discussed)

Memory Map

Memory Chip	Address Bus																				Memory Address
	A19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	A0	
RAM 1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	00000H
(LB)	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0FFFEH
RAM 2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	00001H
(HB)	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0FFFFH
RAM 3	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10000H
(LB)	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1FFFEH
RAM 4	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	10001H
(HB)	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1FFFFH
EPORM 1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	F8000H
(LB)	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	FFFFEH
EPORM 2	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	F8001H
(HB)	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	FFFFFH

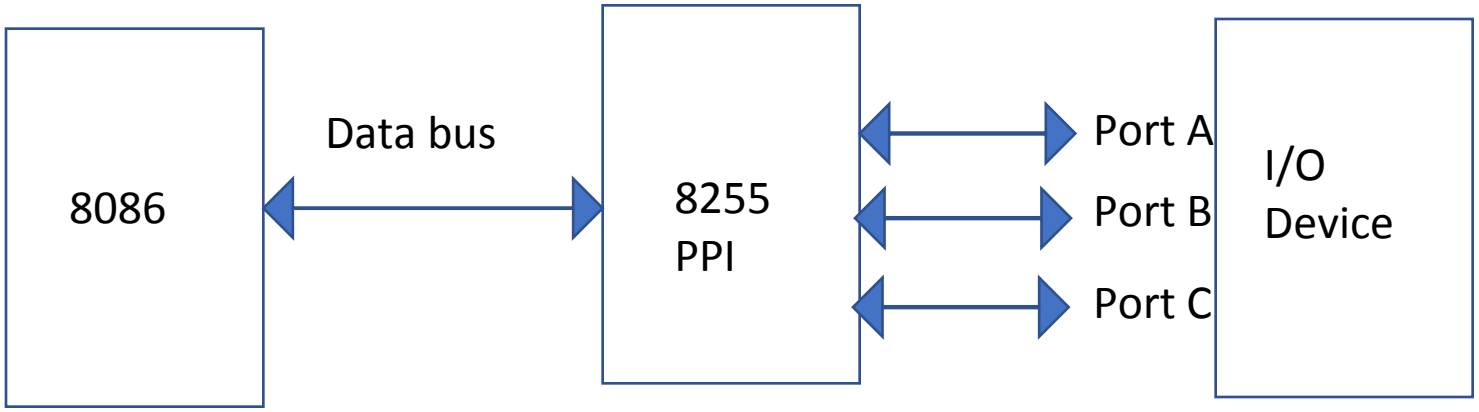
8086 Designing (example discussed)

Memory Diagram



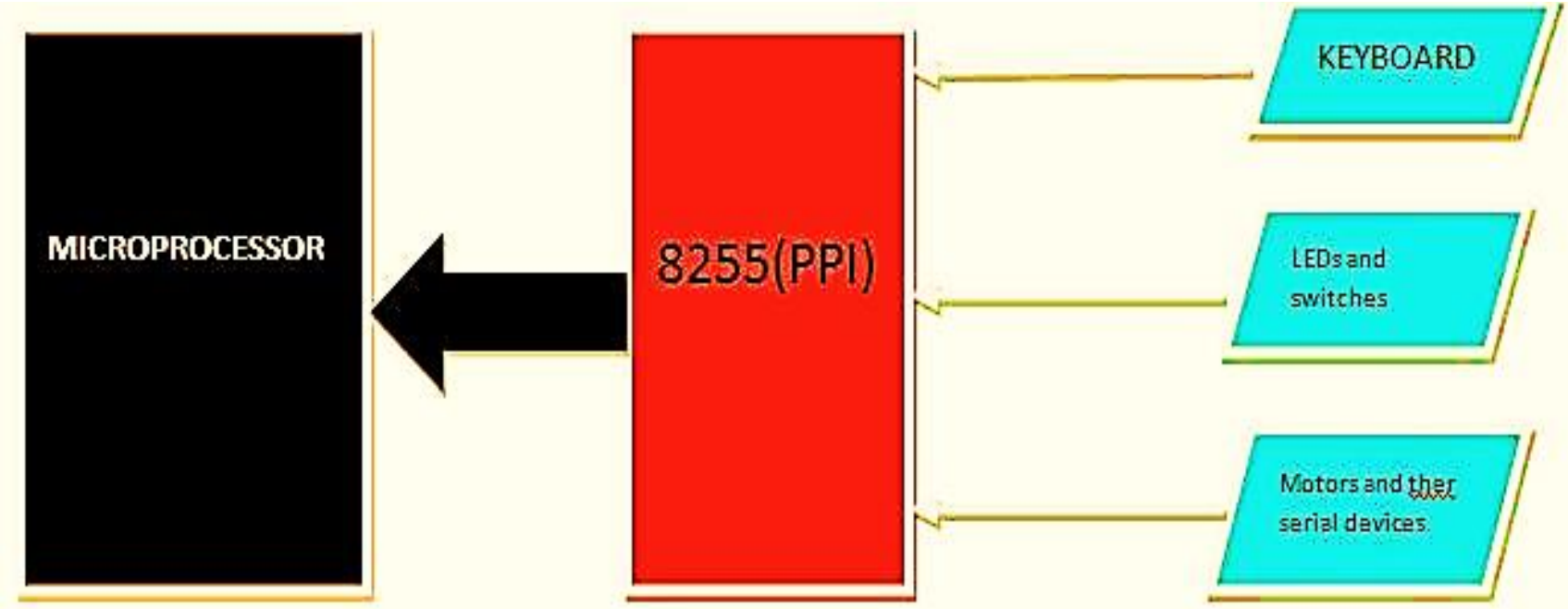
Peripheral Interface

- It provides an interface between Processor and I/O devices



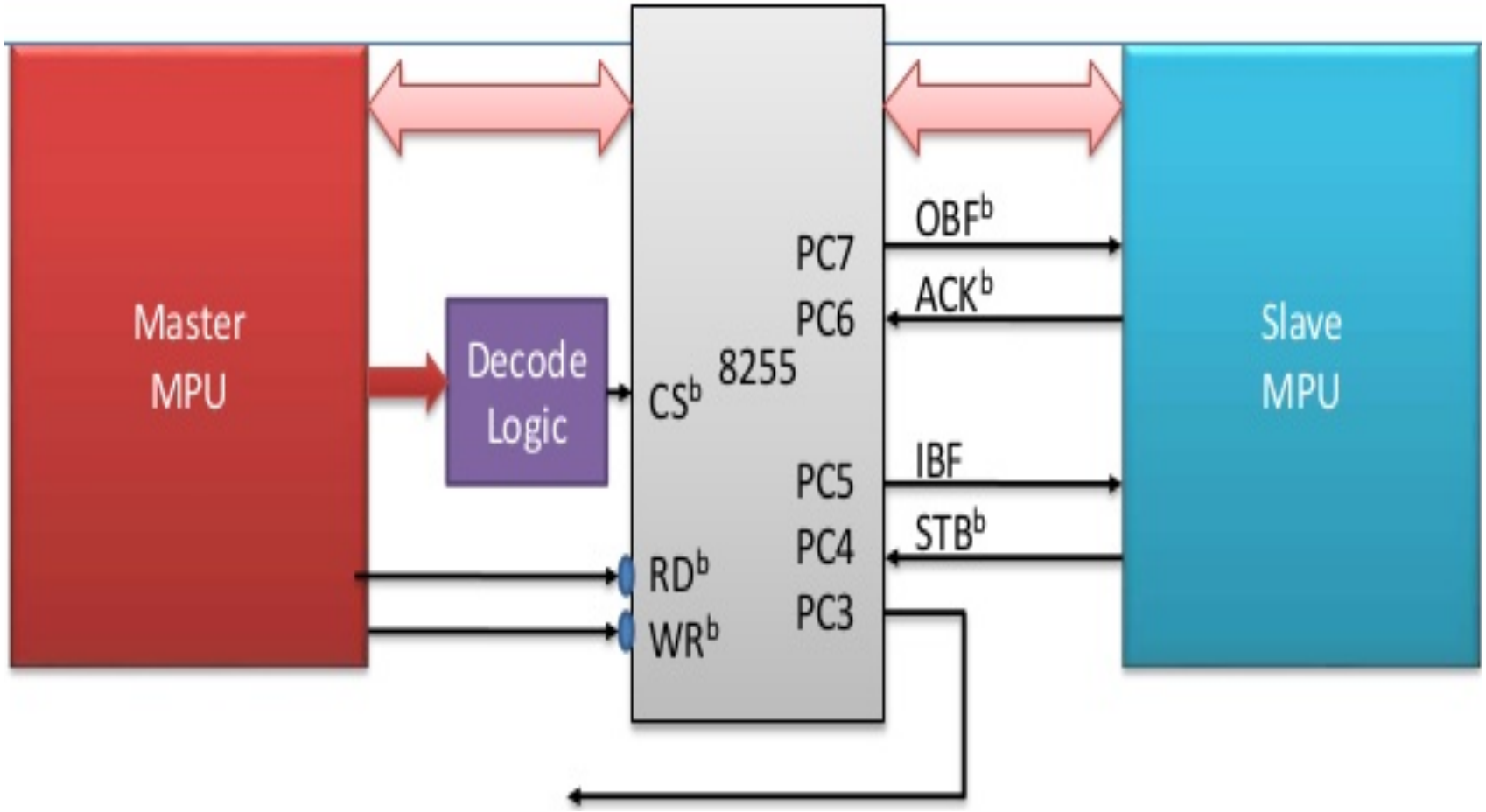
Peripheral Interface

- Controls **flow of data** between a microcomputer and its peripherals



8255-Programmable Peripheral Interface

Can be used to transfer data between 2 MPUs



8255-Programmable Peripheral Interface

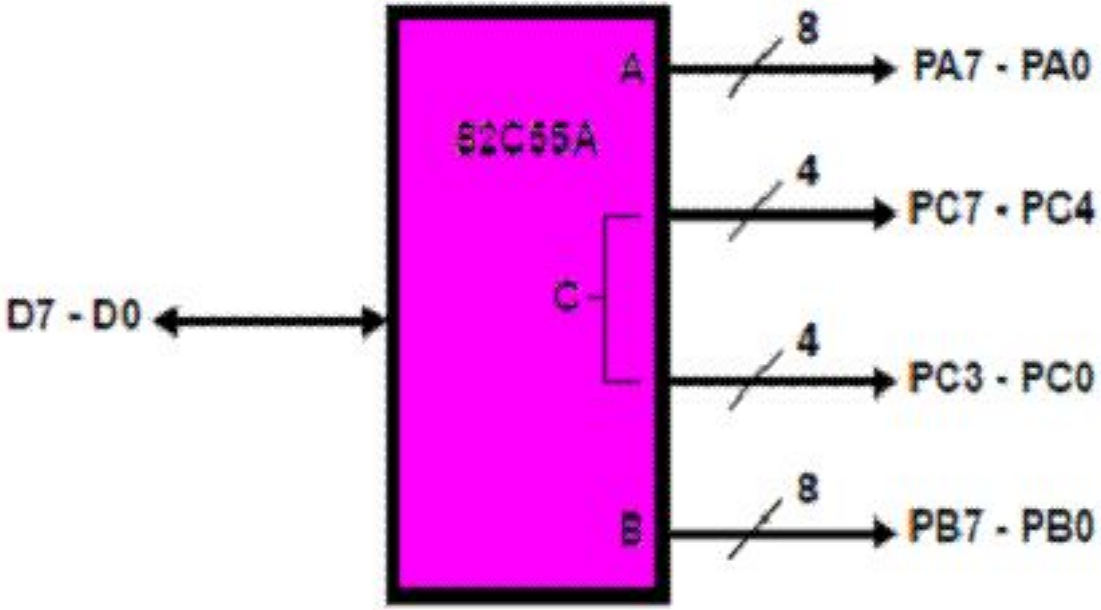
- It is a general purpose parallel I/O interfacing device.
- It can be programmed to transfer data under various conditions.
- It provides 24 I/O lines organized as three 8-bit ports labeled **A**, **B** and **C**.
- Ports A and B can be programmed as an 8-bit input or output port.

8255-Programmable Peripheral Interface

- The eight bits of port **C** can be used as one 8-bit port, or two 4-bit ports
- Port C = **C_{UPPER} (C_U)** and **C_{LOWER} (C_L)**
- The functions of these ports are defined by writing a control word in control register

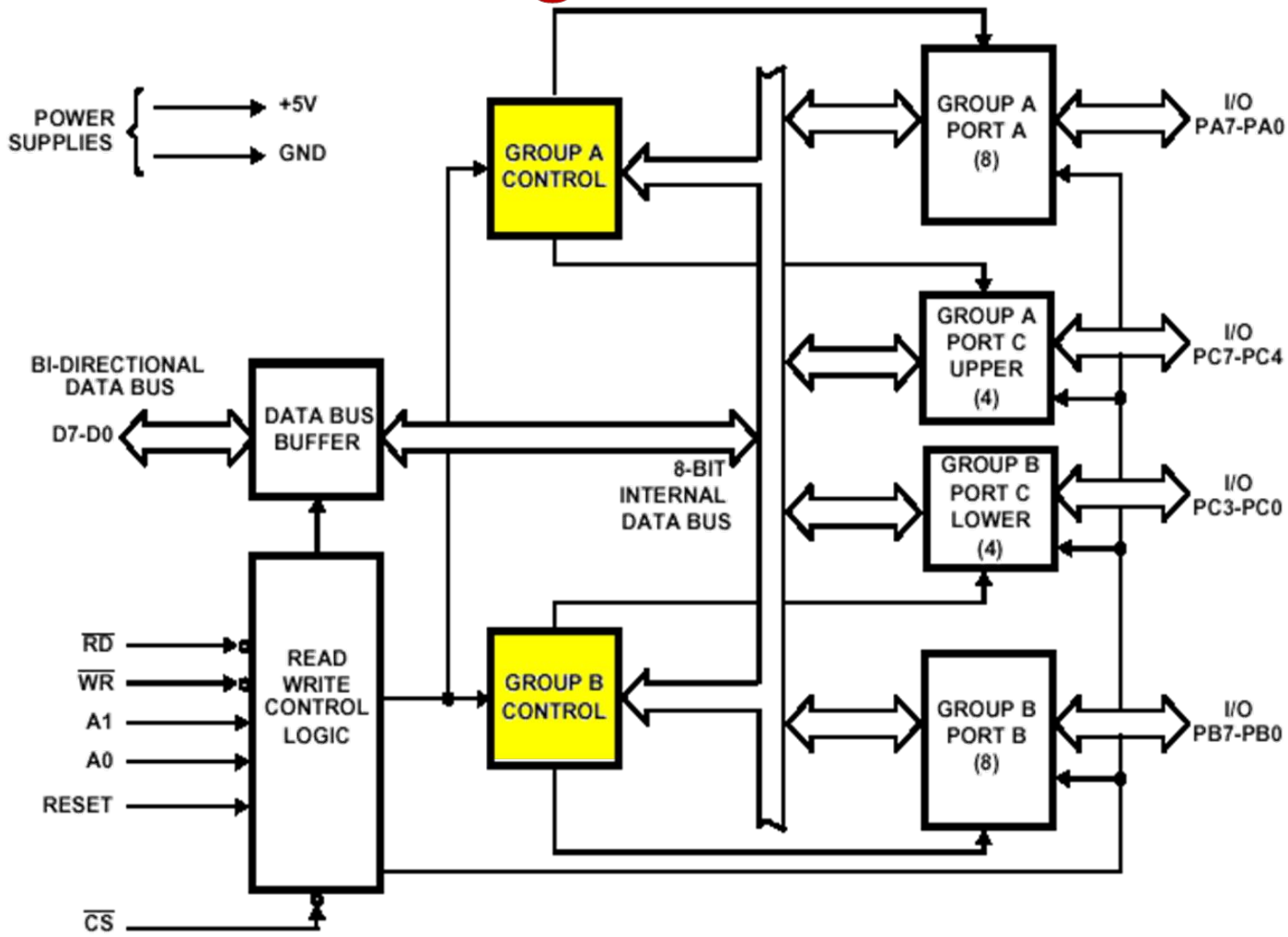
8255-Programmable Peripheral Interface

8255 Port Organization



8255-Programmable Peripheral Interface

8255 Block Diagram



8255 Block diagram

- It consists of
  - two 8-bit ports,
  - two 4-bit ports,
  - data bus buffer and
  - control logic

8255-Programmable Peripheral Interface

Control Logic

- $\overline{RD}$ - It enables the READ operation
- $\overline{WR}$ - It enables the WRITE operation
- RESET – This signal clears control register and sets all ports in input mode
- $\overline{CS}$ - This signal select the chip and A_1 and A_0 specify one of the I/O ports or control register as follows

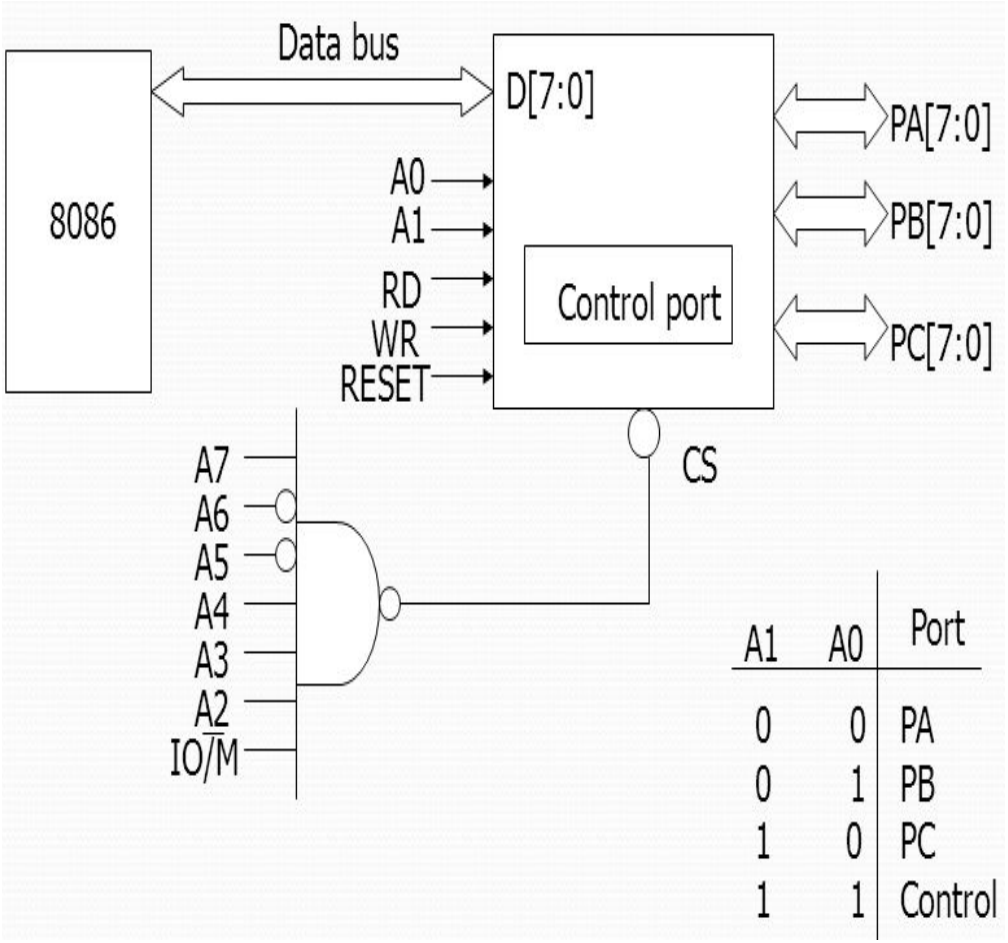
8255-Programmable Peripheral Interface

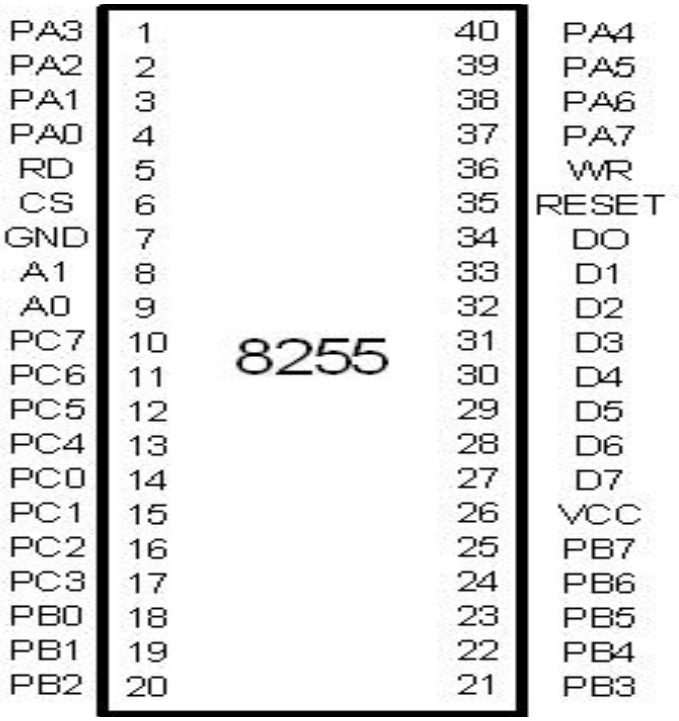
Port Selection

$\overline{CS}$	A_1	A_0	Selected
0	0	0	Port A
0	0	1	Port B
0	1	0	Port C
0	1	1	Control Register
1	X	X	8255 is not selected

8255-Programmable Peripheral Interface

Interfacing with 8086 with Pin Diagram



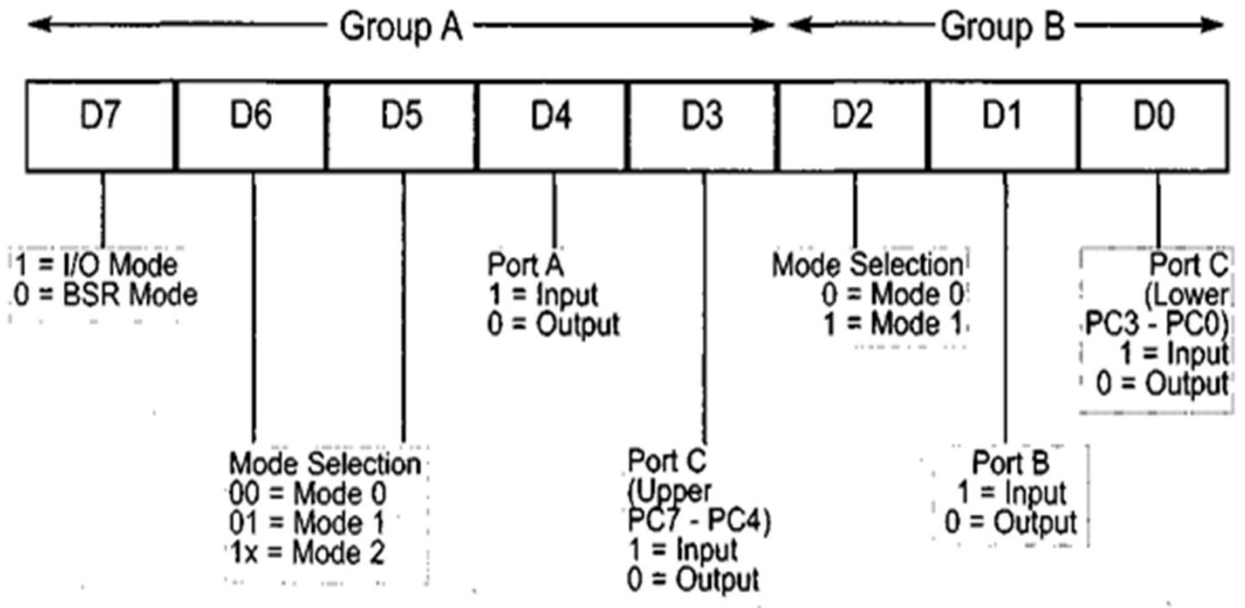


40-pin DIP
(Dual Inline Package)

Control Word

- The contents of this register specify an I/O function for each port

Control Word of 8255



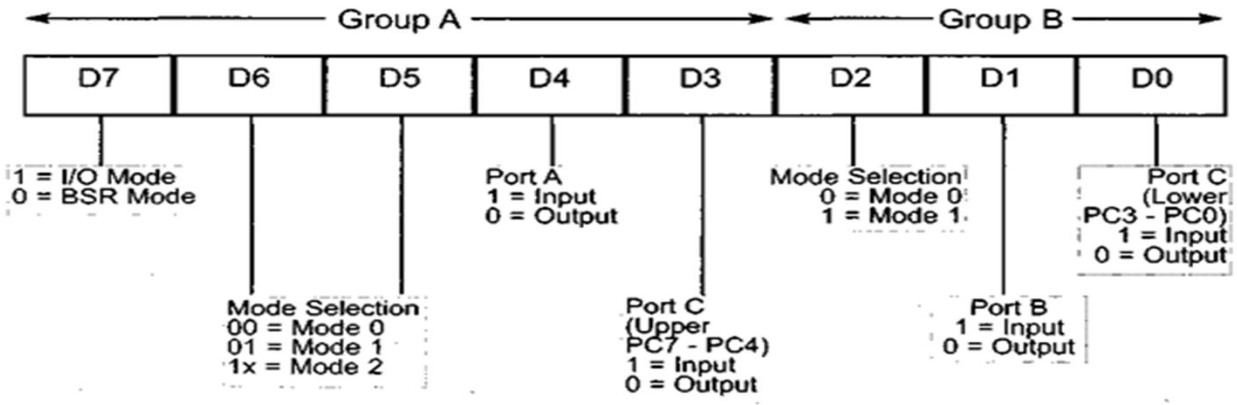
8255-Programmable Peripheral Interface

Example

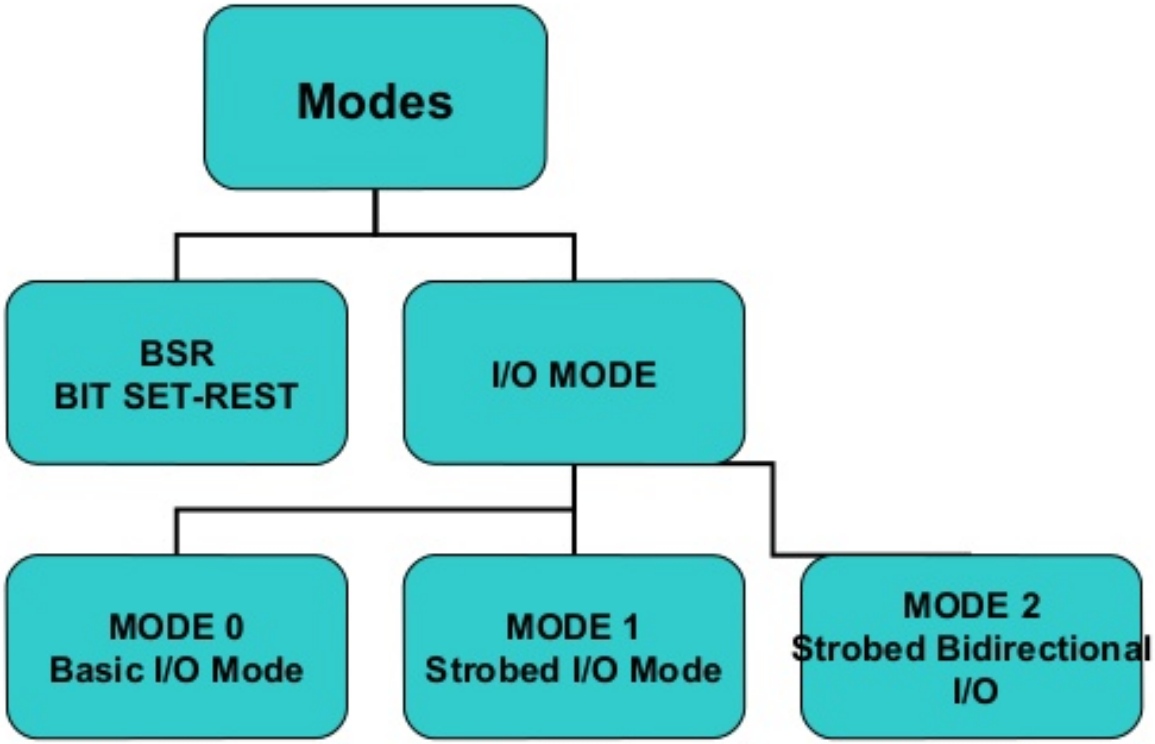
Find the control word , if 8255 is in mode 0 and port A is an output port and port B and C are inputs (It is in I/O Mode)

1	0	0	0	1	0	1	1

Ans : CWR = 8B H



Operating Modes of 8255

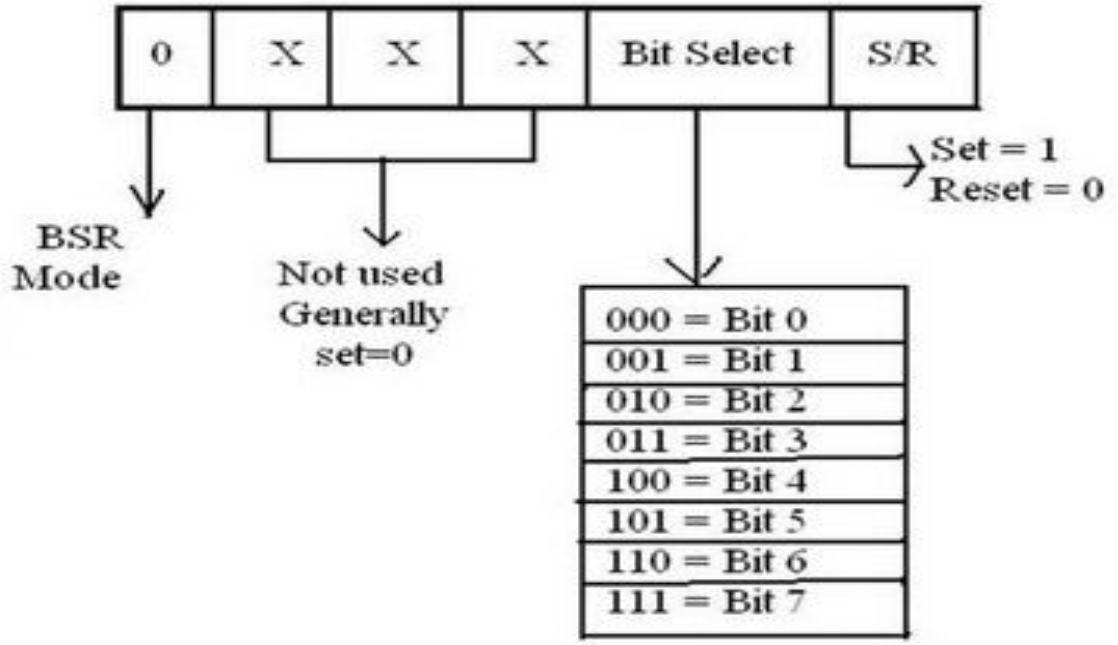


BSR Mode

- It is **concerned with port C** where each bit can be set and reset by writing appropriate control word
- D7 = 0 is recognized as a BSR control word
- It does not alter any previously transmitted control word with D7 = 1
- I/O operations of Ports A and B are not affected by BSR mode
- In BSR mode, individual bits of port C can be used for applications such as ON/OFF switch

8255-Programmable Peripheral Interface

BSR Control Word

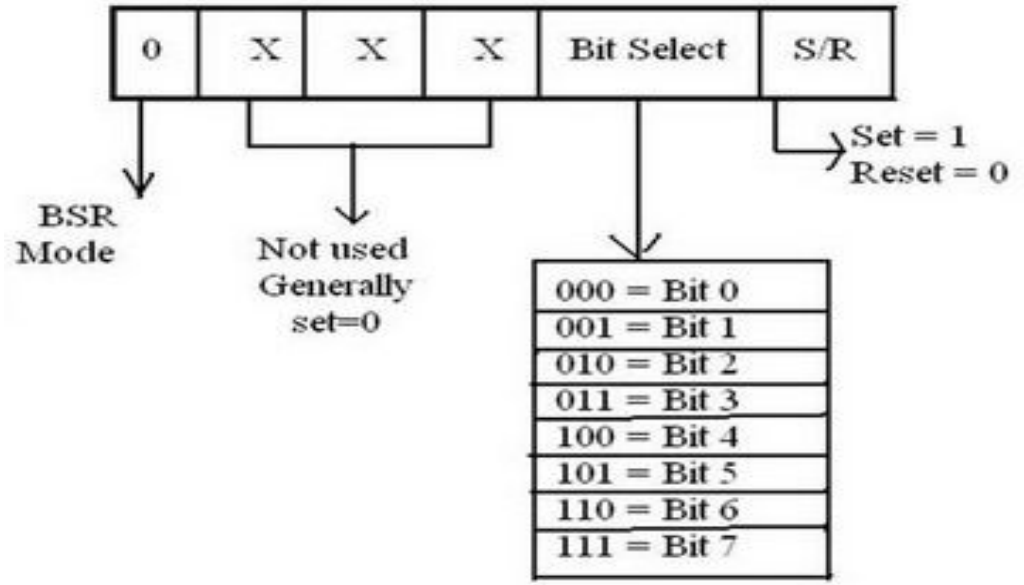


Example

Find the control word , if 8255 is in working in BSR mode, and bit 5 has to be set to 1

0	0	0	0	1	0	1	1

Ans : CWR = 0B H



Mode 0 : Simple Input or Output

- In this mode, ports A and B are used as two simple 8-bit I/O ports and port C as two 4-bit I/O ports
- Each port can be programmed to function simply an input port or output port
- The input/output features in Mode 0 are
  - Outputs are latched
  - Inputs are not latched
  - Ports do not have handshake or interrupt capability

Mode 1 : Input or Output with Handshake

- In mode 1, the handshake signals are exchanged between MPU and peripherals prior to data transfer
- The features of this mode are

1. Ports A and B function as 8-bit I/O ports.
2. Each port uses 3 lines from port C for handshake signals. The remaining 2 lines of port C can be used for I/O .
3. Both input and output data are latched
4. Interrupt logic is supported.

8255-Programmable Peripheral Interface

Mode 1 : Input Control Signals

- For handshaking port A uses PC_3, PC_4 and PC_5 and port B uses PC_0, PC_1 and PC_2 .
- These signals are as follows:

- Can be erased using UV light
- EPROM series by Intel are
- Each chip contains
  - n Address pins
  - 8 Data pins
  - Chip Enable ($\overline{CE}$)
  - Output Enable ($\overline{OE}$)

- This signal is generated by a peripheral device to indicate that it has transmitted a byte of data
- In response to this input, 8255 generates IBF and INTR.

8255-Programmable Peripheral Interface

IBF : Input Buffer Full

- This signal is an acknowledgement by 8255 to indicate that the latch has received the data
- This is reset when MPU reads data

INTR : Interrupt Request

- This is an output signal that may be used to interrupt MPU.
- This signal is generated if STB, IBF and INTE are all at logic 1.
- This is reset by the falling edge of RD signal

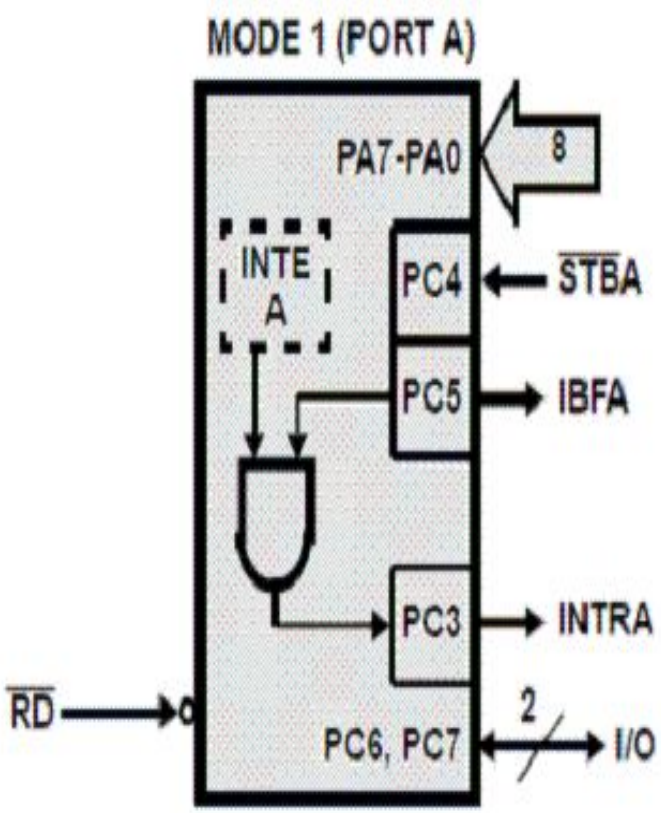
8255-Programmable Peripheral Interface

INTE : Interrupt Enable

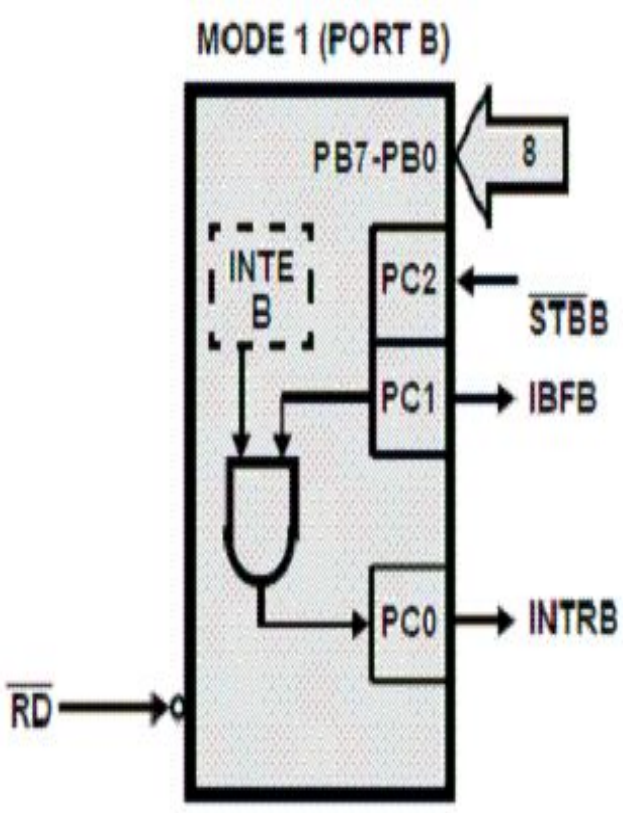
- This is an internal flipflop used to enable or disable the generation of the INTR signal
- The $INTE_A$ is enabled or disabled through PC4 and $INTE_B$ through PC2

8255-Programmable Peripheral Interface

Mode 1 Input Configuration: Port A



Mode 1 Input Configuration: Port B



Programming 8255

- 8255 can be programmed using
- Status Check I/O
  - In this method , the MPU continues to check data status through IBF line until it goes high.
  - The disadvantage is that the MPU is tied up in the loop.
- Interrupt I/O
  - In this case, the INTR line is used.

Status Word

- It will be placed in accumulator if Port C is read.

D7	D6	D5	D4	D3	D2	D1	D0
I/O	I/O	IBF_A	$INTE_A$	$INTR_A$	$INTE_B$	IBF_B	$INTR_B$

8255-Programmable Peripheral Interface

Mode 1 : Output Control Signals

- Can be erased using UV light
- EPROM series by Intel are
- Each chip contains
  - n Address pins
  - 8 Data pins
  - Chip Enable ($\overline{CE}$)
  - Output Enable ($\overline{OE}$)

- It goes low when MPU writes data into the output latch of 8255
- This signal indicates peripheral that new data is ready to be read.
- It goes high again after 8255 receives an $\overline{ACK}$ from peripheral

- Can be programmed by user.
- Can be erased using UV light
- EPROM series by Intel are
- Each chip contains
  - n Address pins
  - 8 Data pins
  - Chip Enable ($\overline{CE}$)
  - Output Enable ($\overline{OE}$)

- This is an input signal from a peripheral when it receives data from 8255 ports

Mode 1 : Output Control Signals

- Can be erased using UV light
- EPROM series by Intel are
- Each chip contains
  - n Address pins
  - 8 Data pins
  - Chip Enable ($\overline{CE}$)
  - Output Enable ($\overline{OE}$)

- This is an output signal and it is set by the rising edge of ACK signal
- This signal can be used to interrupt MPU to request next data
- The INTR is set when $\overline{OBF}, \overline{ACK}$ and INTE are all one and reset by the falling edge of $\overline{WR}$

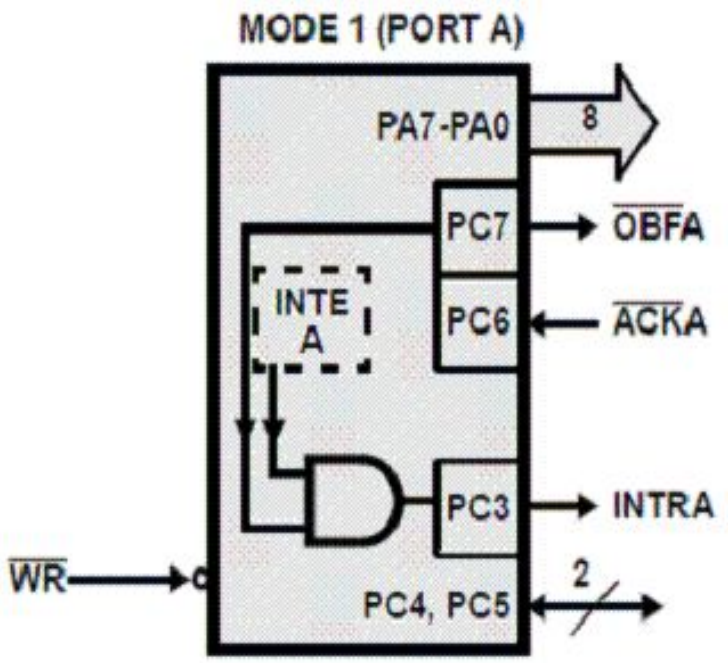
Mode 1 : Output Control Signals

4. INTE - Interrupt Enable

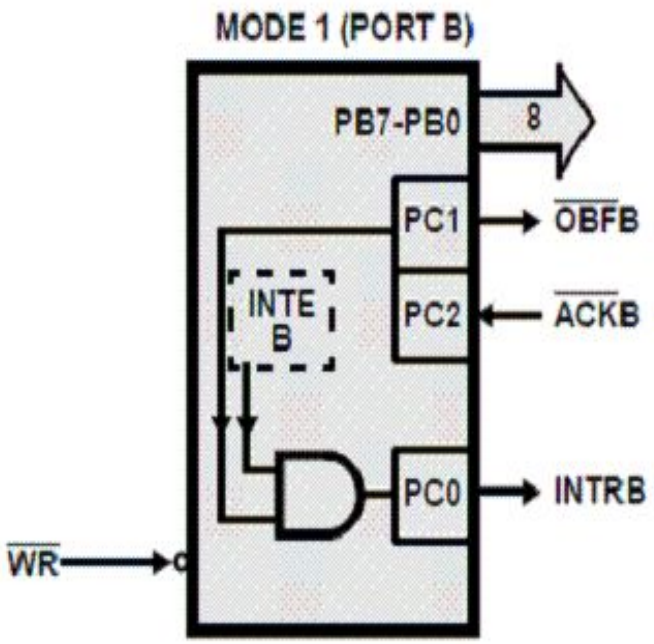
- This is an internal flipflop to a port and needs to be set to generate INTR signal
- The two flipflops $INTE_A$ and $INTE_B$ are controlled by bits PC6 and PC2 respectively through BSR mode

8255-Programmable Peripheral Interface

Mode 1 : Output Configuration : Port A



Mode 1 : Output Configuration : Port B



8255-Programmable Peripheral Interface

Status Word

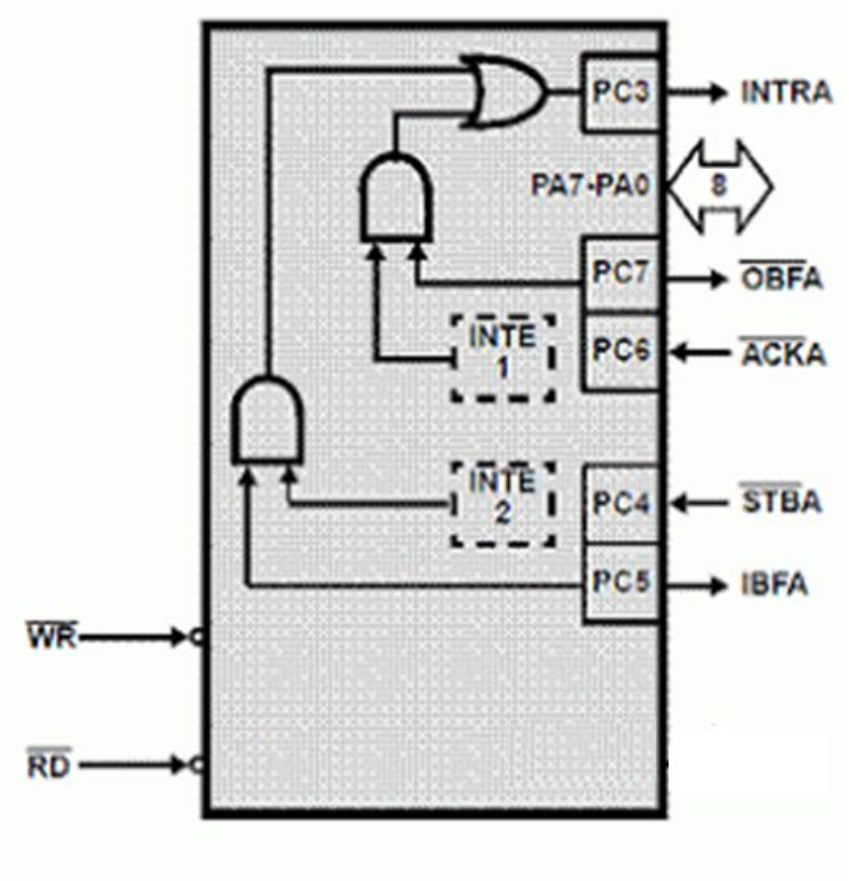
It will be placed in accumulator if Port C is read.

D7	D6	D5	D4	D3	D2	D1	D0
$\overline{OBF_A}$	$INTE_A$	I/O	I/O	$INTR_A$	$INTE_B$	$\overline{OBF_B}$	$INTR_B$

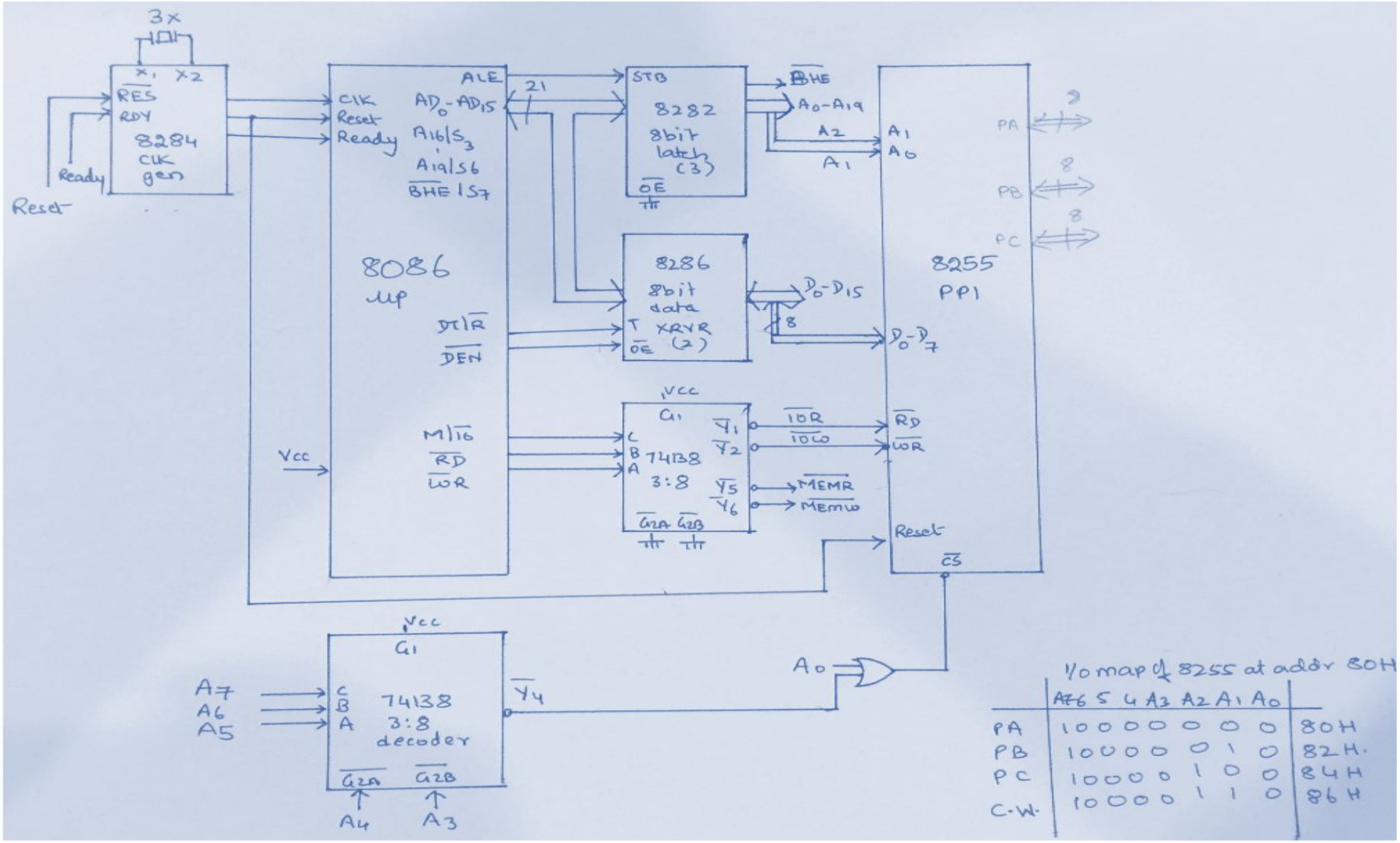
Mode 2 : Bidirectional Data Transfer

- In this mode, Port A is a bidirectional port and Port B either in mode 0 or mode 1
- Port A uses 5 signals from port C as handshake signals.
- The remaining 3 signals from port C can be used either as simple I/O or as handshake for port B.

Mode 2 –Port A Control Pins



8255 interfacing with 8086



I/O Interfacing

- I/O Devices can be interfaced in 2 ways

1. I/O Mapped I/O:

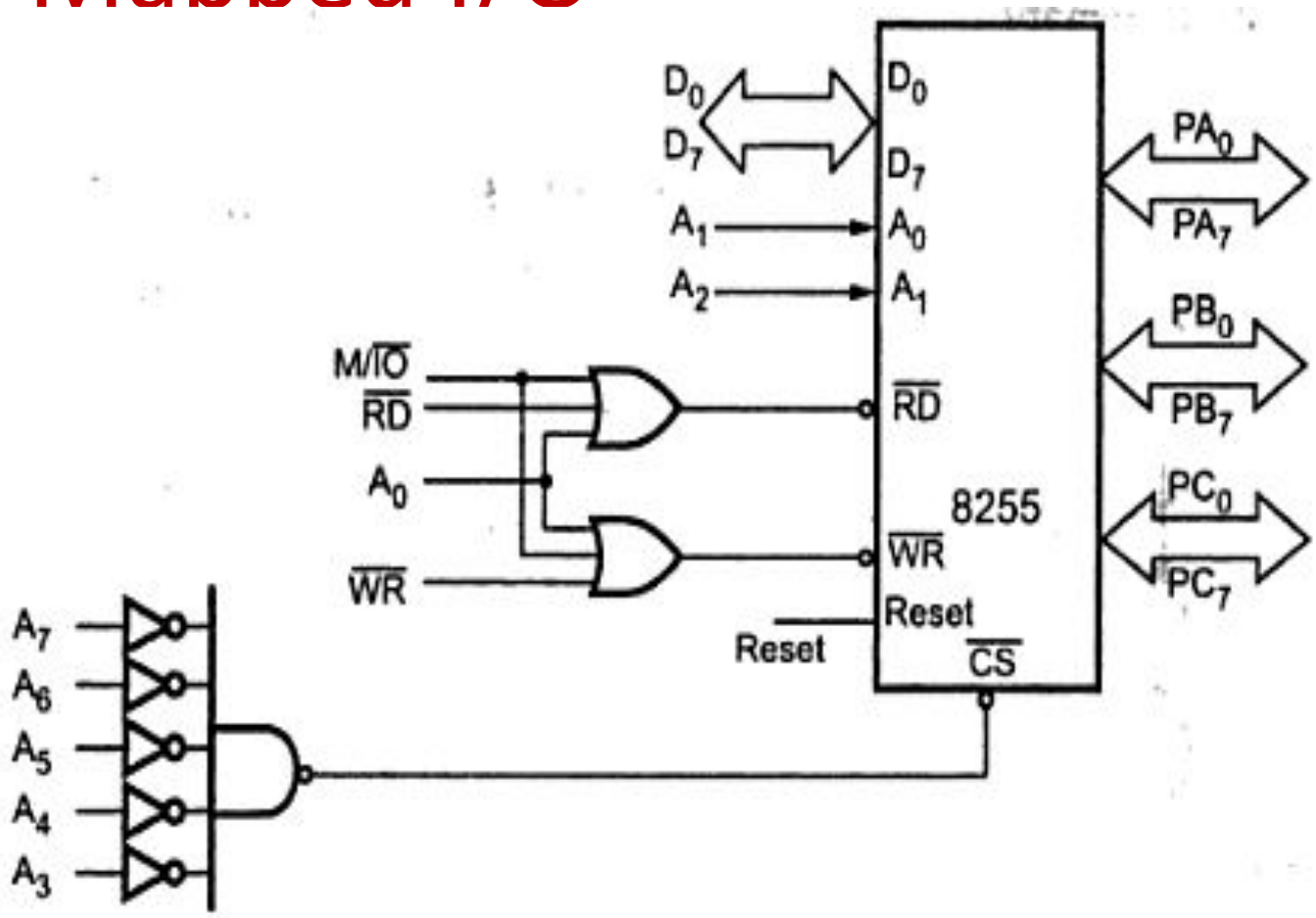
- $\overline{M} / IO$ signal is low
- IN and OUT instructions are used
- Direct : 256 (2^8) , Indirect : 65536 (2^{16}) addresses are used

2. Memory Mapped I/O:

- $M / \overline{IO}$ signal is high
- All 20 address lines are used

8255-Programmable Peripheral Interface

I/O Mapped I/O



8255-Programmable Peripheral Interface

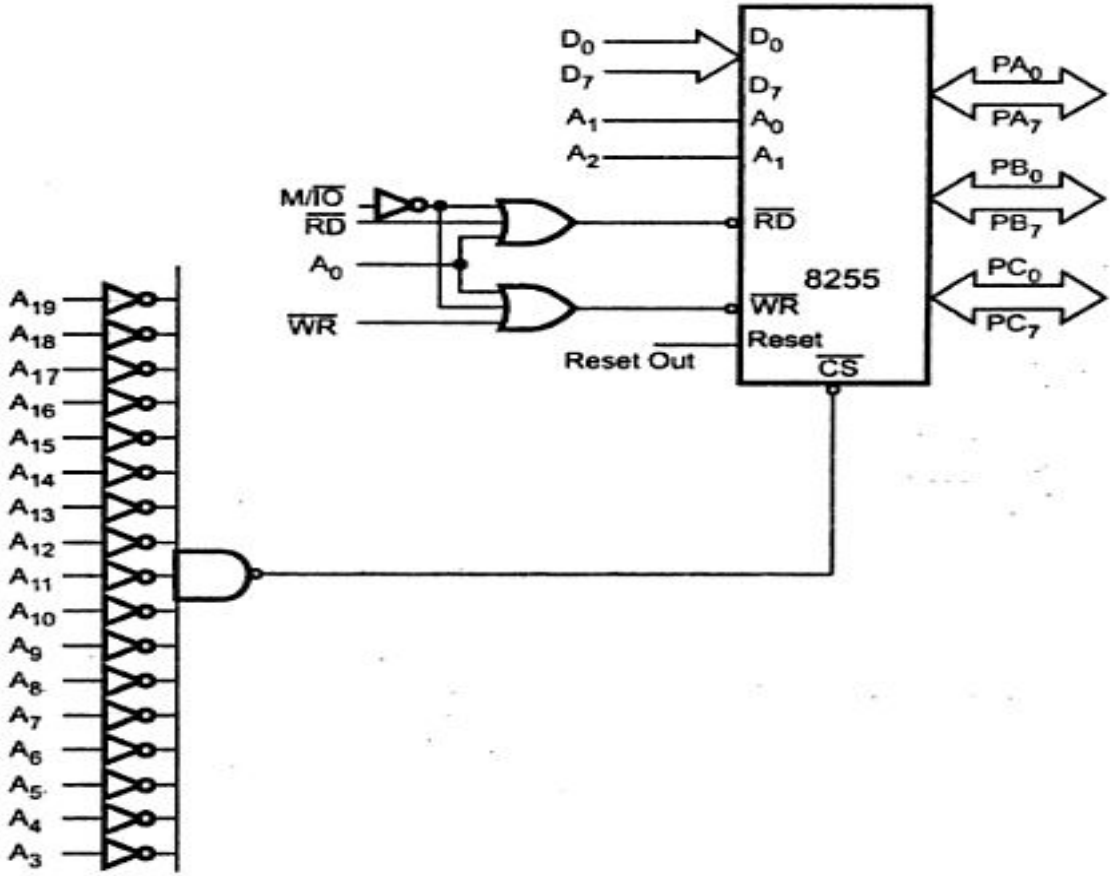
I/O Mapped I/O

I/O Map :

Port / control Register	Address lines								Address
	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
Port A	0	0	0	0	0	0	0	0	00H
Port B	0	0	0	0	0	0	1	0	02H
Port C	0	0	0	0	0	1	0	0	04H
Control register	0	0	0	0	0	1	1	0	06H

8255-Programmable Peripheral Interface

Memory Mapped I/O



Memory Mapped I/O

I/O Map :

Register	Address
Port A	00000H
Port B	00002H
Port C	00004H
Control register	00006H