Microprocessors

Von Neumann Architecture

Its key principle is the **stored-program concept**, where **instructions and data share the same memory and bus system**.

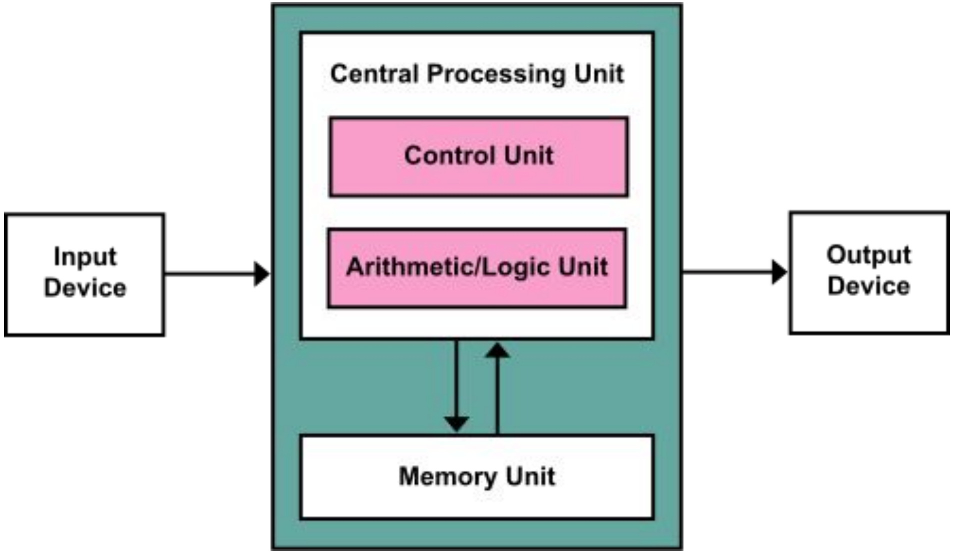
- **Single memory space** for both instructions and data.
- **Single bus** for data and instructions
- **Sequential execution**: Fetch → Decode → Execute cycle.

Advantages:

- Simpler design, cost-effective.
- Flexible for general-purpose computing.

Limitations:

- Performance bottleneck due to shared bus.
- Vulnerable to certain security issues (e.g., code injection).



Intel x86 family (8086, 80286, 80386, etc.)
ARM processors (in most general-purpose configurations)
General-purpose CPUs in PCs, laptops, servers.

Harvard Architecture

The **Harvard architecture** stores **instructions and data in separate memory units**, each with its own bus.

- **Separate instruction and data memory**.
- **Independent buses** → allows simultaneous fetching of instructions and data.
- Often used in **real-time systems** and **embedded applications**.

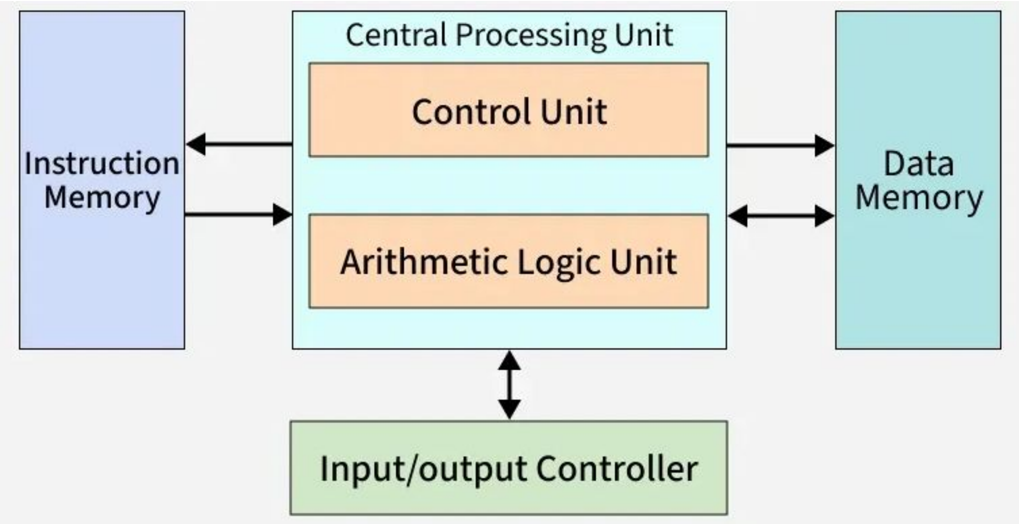
Advantages:

- Faster execution (parallel access).
- Predictable performance (ideal for embedded systems).

Disadvantages:

- More complex and expensive to implement.
- Less flexible for general-purpose computing.

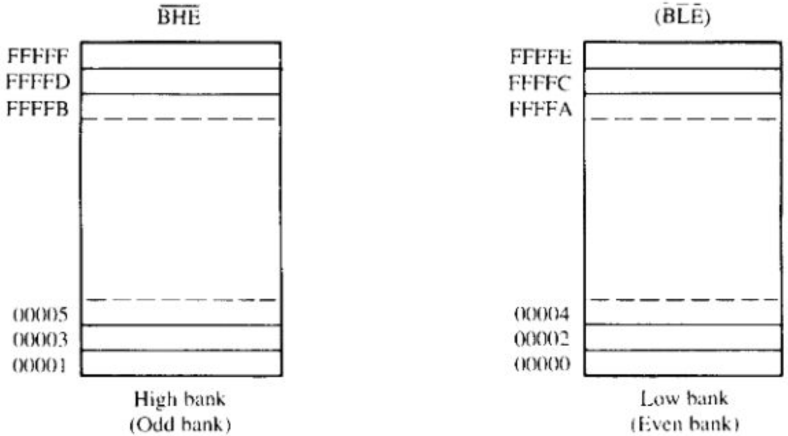
Microcontrollers: PIC, AVR, ARM Cortex-M series.
DSPs: Texas Instruments TMS320, Motorola DSPs.
Embedded systems: Automotive, IoT devices.



Memory banking

The 8086 must be able to write data to any 16-bit location—or any 8-bit location. This means that the 16-bit data bus must be divided into two separate sections (or banks) that are 8 bits wide so that the microprocessor can write to either half (8-bit) or both halves (16-bit).

FIGURE 10–27 The high (odd) and low (even) 8-bit memory banks of the 8086/80286/80386SX microprocessors.



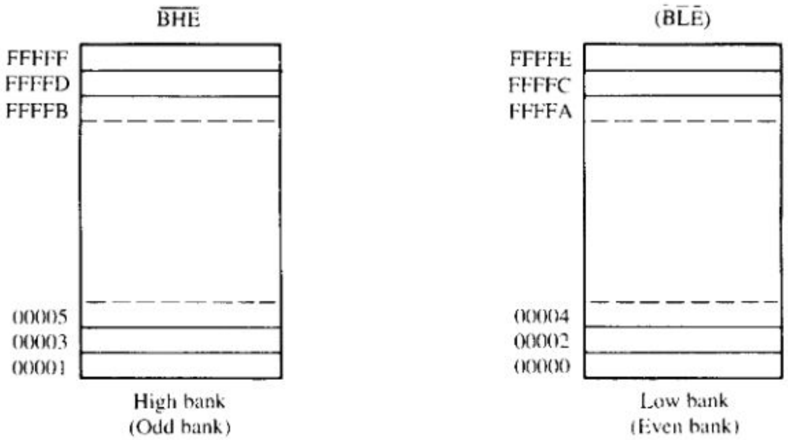
Note: A_0 is labeled $\overline{BLE}$ (Bus low enable) on the 80386SX.

low bank holds all the even-numbered memory locations
high bank holds all the odd-numbered memory locations.

- BLE’ also call A_0

Memory banking

FIGURE 10–27 The high (odd) and low (even) 8-bit memory banks of the 8086/80286/80386SX microprocessors.



Note: A_0 is labeled $\overline{BLE}$ (Bus low enable) on the 80386SX.

TABLE 10–2 Memory bank selection using $\overline{BHE}$ and $\overline{BLE}$ (A_0).

$\overline{BHE}$	$\overline{BLE}$	*Function*
0	0	Both banks enabled for a 16-bit transfer
0	1	High bank enabled for an 8-bit transfer
1	0	Low bank enabled for an 8-bit transfer
1	1	No bank enabled

Memory Banking

1 MB Memory [A19 – A0]

512 KB [ODD BANK]

- It is also called higher bank
- These memory bank is selected when $\overline{BHE} = 0$
- Range of Address is

$\overline{\text{BHE}}$ →

00001H
00003H
.....
FFFFFH

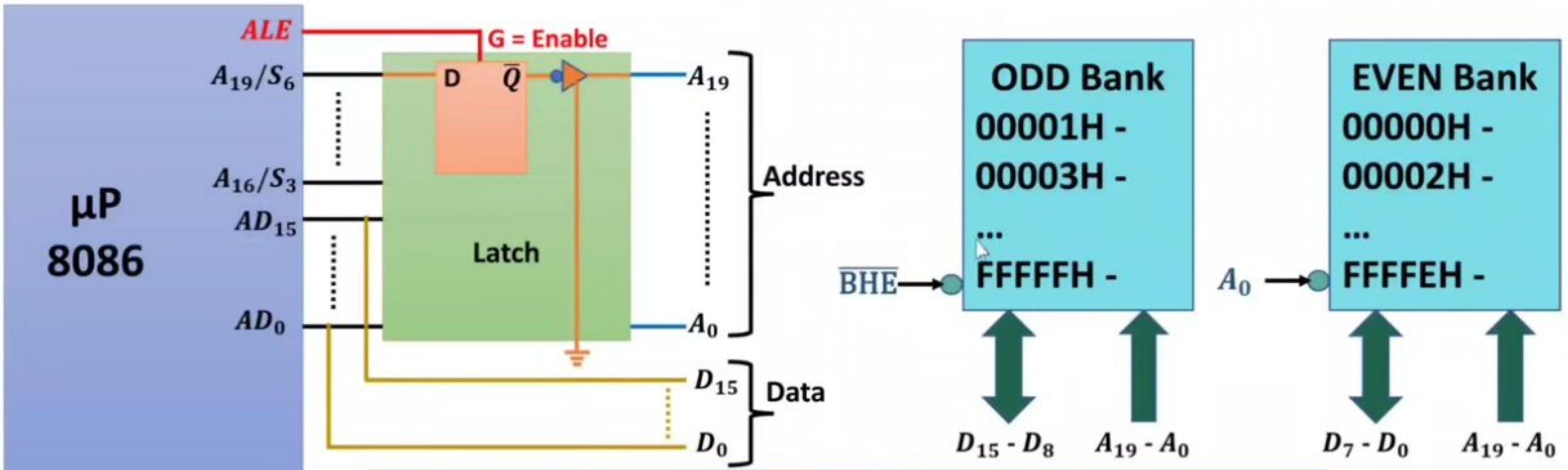
512 KB [EVEN BANK]

- It is also called Lower bank
- These memory bank is selected when $A_0 = 0$
- Range of Address is

A_0 →

00000H
00002H
.....
FFFFEH

Memory Banking



A0	$\overline{BHE}$	Machine Cycle	Data	Operation
0	0	1	D15-D0	Read a Word from Even Add
0	1	1	D7-D0	Read a Byte from Even Add
1	0	1	D15-D8	Read a Byte from Odd Add
1	0	2	D15-D8	Read a Word from Odd Add

Memory interfacing

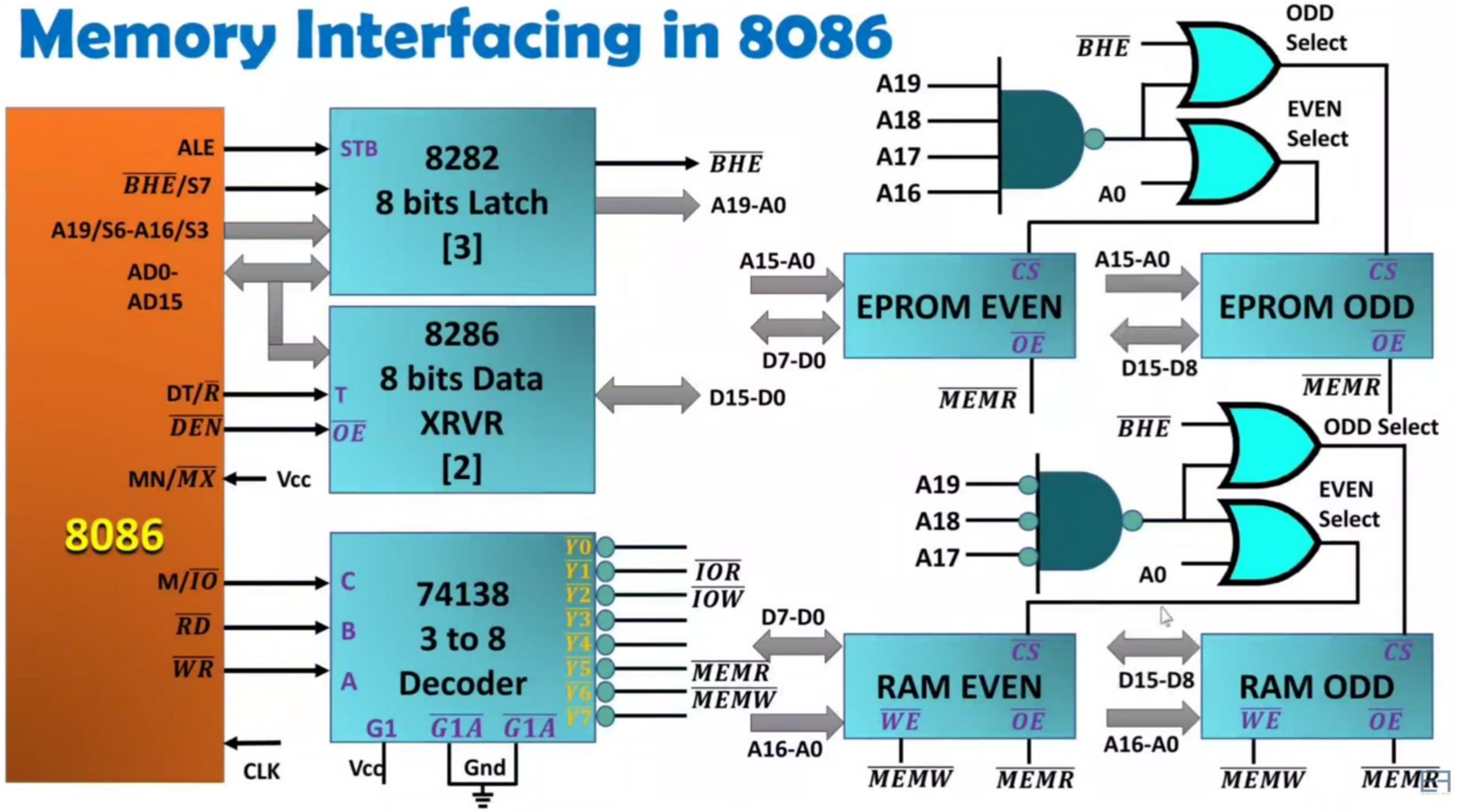
- Design following system with 8086 microprocessor.
  1. 8086 working in minimum mode with 8MHz.
  2. 64KB EPROM using 32KB EPROM
  3. 128KB RAM using 64KB RAM

- In Memory interfacing, we need to interface four categories of lines:
- Address Lines
- Data Lines
- Control Lines
- Chip Select

- 64KB EPROM using 32KB EPROM
- Numbers of Chips = 2 chips of 32KB EPROM [one chip for even address and another for odd address]
- Address lines for 64KB Address = $2^{10} \times 2^{6} = 2^{16}$
- So it needs 16 address lines.
- Data Lines for 64KB = 8 [For Byte, it is 8 bits]
- Control Lines for 64KB EPROM = Memory Read
- 128KB RAM using 64KB RAM
- Numbers of Chips = 2 chips of 64KB RAM [one chip for even address and another for odd address]
- Address lines for 128KB Address = $2^{10} \times 2^{7} = 2^{17}$
- So it needs 17 address lines.
- Data Lines for 128KB = 8 [For Byte, it is 8 bits]
- Control Lines for 128KB RAM = Memory Read & Memory Write.

Memory IC	A19	A18	A17	A16	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	Address
EPROM 1 [Lower Byte]	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	0	FFFFEH
EPROM 2 [Higher Byte]	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	F0001H
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	FFFFFH
RAM 1 [Lower Byte]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	00000H
	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1FFFEH
RAM 2 [Higher Byte]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	00001H
	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1FFFFH

Memory Interfacing in 8086



I/O interface

Microprocessors interact with external devices such as keyboards, displays, and sensors through Input/Output (I/O) operations. There are two primary methods for interfacing I/O devices:

1. Isolated I/O (I/O Mapped I/O)

I/O devices are assigned a separate address space distinct from the memory address space. The CPU uses special instructions to communicate with these devices.

- **Separate Address Space:** I/O devices are accessed using a dedicated I/O address range (e.g., 0x0000 to 0x00FF).
- **Special Instructions:** Uses IN and OUT instructions to read/write data.
- **Limited Address Range:** Typically supports 256 or 65,536 I/O ports depending on architecture.
- **No Memory Overlap:** Memory and I/O addresses are completely independent.

I/O interface

1. Isolated I/O (I/O Mapped I/O) - example

```
MOV DX, 03F8h    ; Load COM1 port address into DX
MOV AL, 'A'      ; Load ASCII character 'A' into AL
OUT DX, AL       ; Send 'A' to the serial port
```

Advantages:

- Efficient use of memory space.
- Simple decoding logic for I/O devices.
- Clear separation between memory and I/O operations.

- **Disadvantages:**

- Requires special instructions.
- Limited number of I/O ports.

I/O interface

1. Memory-Mapped I/O

In memory-mapped I/O, I/O devices are assigned addresses within the same address space as memory. The CPU accesses them using standard memory instructions.

- **Shared Address Space:** I/O devices are treated like memory locations.
- **Standard Instructions:** Uses instructions like MOV, LDA, STA, etc.
- **Flexible Addressing:** Can use full memory range for I/O devices.
- **No Need for Special Instructions:** Easier for compilers and high-level languages.

I/O interface

1. Memory-Mapped I/O - example

```
MOV AX, [1234h] ; Read from memory-mapped I/O device at address 1234h

MOV [1234h], AX ; Write to memory-mapped I/O device
```

Advantages:

- Easier integration with software tools.
- Allows complex data structures and buffers.
- More I/O devices can be supported.

- **Disadvantages:**

- Reduces available memory space.
- Requires more complex address decoding.

I/O interface

TABLE 11–1 Input/Output instructions.

Instruction	*Data Width*	*Function*
IN AL, p8	8	A byte is input into AL from port p8
IN AX, p8	16	A word is input into AX from port p8
IN EAX, p8	32	A doubleword is input into EAX from port p8
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
IN EAX, DX	32	A doubleword is input into EAX from the port addressed by DX
INSB	8	A byte is input from the port addressed by DI 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 DI and stored into the extra segment memory location addressed by DI, then DI = DI ± 2
INSD	32	A doubleword is input from the port addressed by DI and stored into the extra segment memory location addressed by DI, then DI = DI ± 4
OUT p8, AL	8	A byte is output from AL into port p8
OUT p8, AX	16	A word is output from AL into port p8
OUT p8, EAX	32	A doubleword is output from EAX into port p8
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
OUT DX, EAX	32	A doubleword is output from EAX into the port addressed by DX
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
OUTSD	32	A doubleword is output from the data segment memory location addressed by SI into the port addressed by DX, then SI = SI ± 4

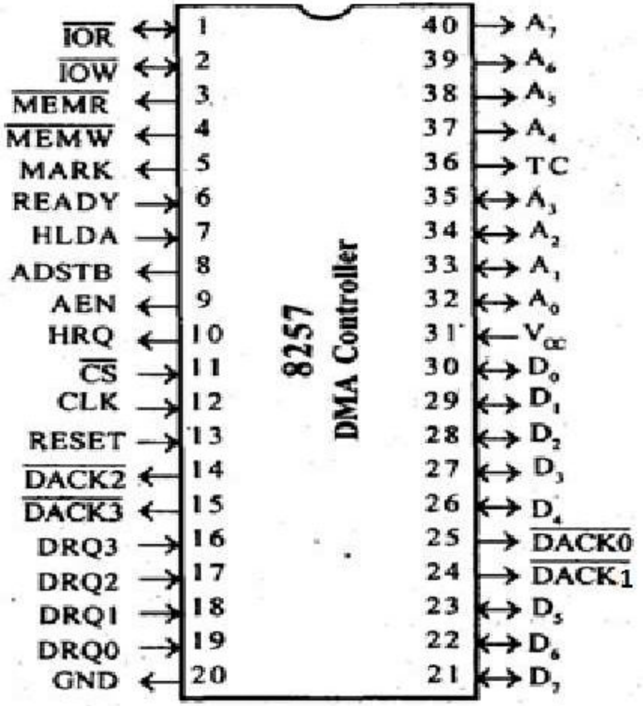
isolated I/O (I/O mapped I/O) and memory-mapped I/O

Memory Mapped IO	IO Mapped IO
❖ Here IO devices treated as Memory	❖ Here IO devices treated as IO.
❖ 20 bits addressing $(A_0 - A_{19})$	❖ 8 or 16 bits addressing $(A_0 - A_{15})$
❖ It can address = 2^{20} = 1MB Address	❖ It can address = 2^{16} = 64K Address
❖ Number of devices can be = 2^{20}	❖ Number of devices can be = 2^{16}
❖ Decoding is more complex as total 20 Address lines are used for full decoding.	❖ Decoding is Less complex as total 16 Address lines are used for full decoding.
❖ Decoding is Expensive.	❖ Decoding is cheaper.
❖ Here we use $\overline{MEMR}$ and $\overline{MEMW}$ control signals.	❖ Here we use $\overline{IOR}$ and $\overline{IOW}$ control signals.
❖ IO can be accessed by any memory instructions.	❖ IO can be accessed ONLY by IN and OUT Instructions.
❖ Data transfer happens between any registers and IO.	❖ Data transfer only between AX [AH & AL] and IO.
❖ Works slower due to more gates and circuits.	❖ Works faster due to less delay.

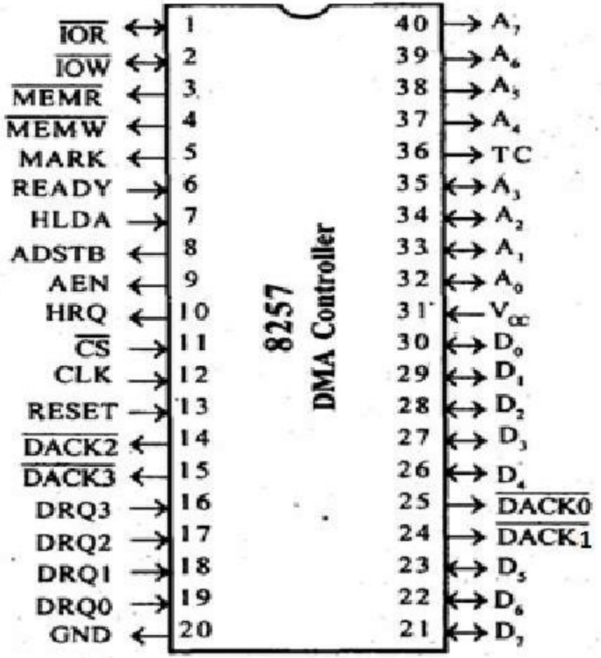
8257 Direct Memory Access

❖ Basics of 8257 Direct Memory Access

- ❑ 8257 is used to for high speed data transfer in between in IO devices and memory.
- ❑ Using Microprocessor data transfer is slow, as microprocessor have to execute instructions and it have to check interrupt as
- ❑ Using 8257, MPU releases the control of the buses to
- ❑ Here, Data transfer between memory and IO is bypassing MPU.
- ❑ To take control of Address and Data bus from MPU, HLDA control terminals which are used by DMA.



8257 Direct Memory Access



❖ **HOLD and HLDA**

❑ **HOLD :**

- This is active high signal input MPU.
- It gives request to MPU for address and data bus control.
- After receiving HOLD signal, MPU relinquishes the buses in following machine cycle.
- All buses are tri stated and HLDA sent out by MPU.
- MPU regains the control of buses after HOLD goes low.

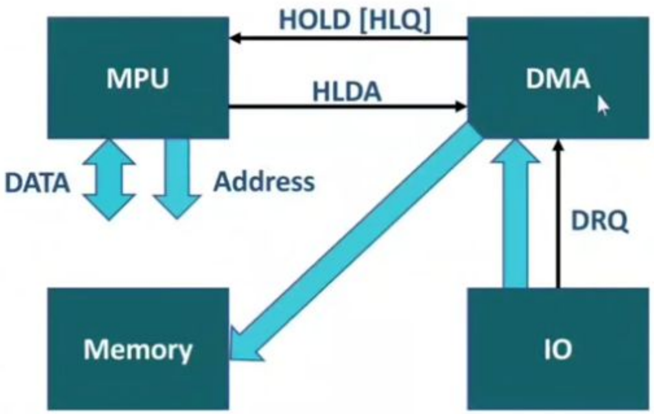
❑ **HLDA :**

- This is active high signal indicating that the MPU is relinquishing control of buses.

8257 Direct Memory Access

❖ Working of DMA

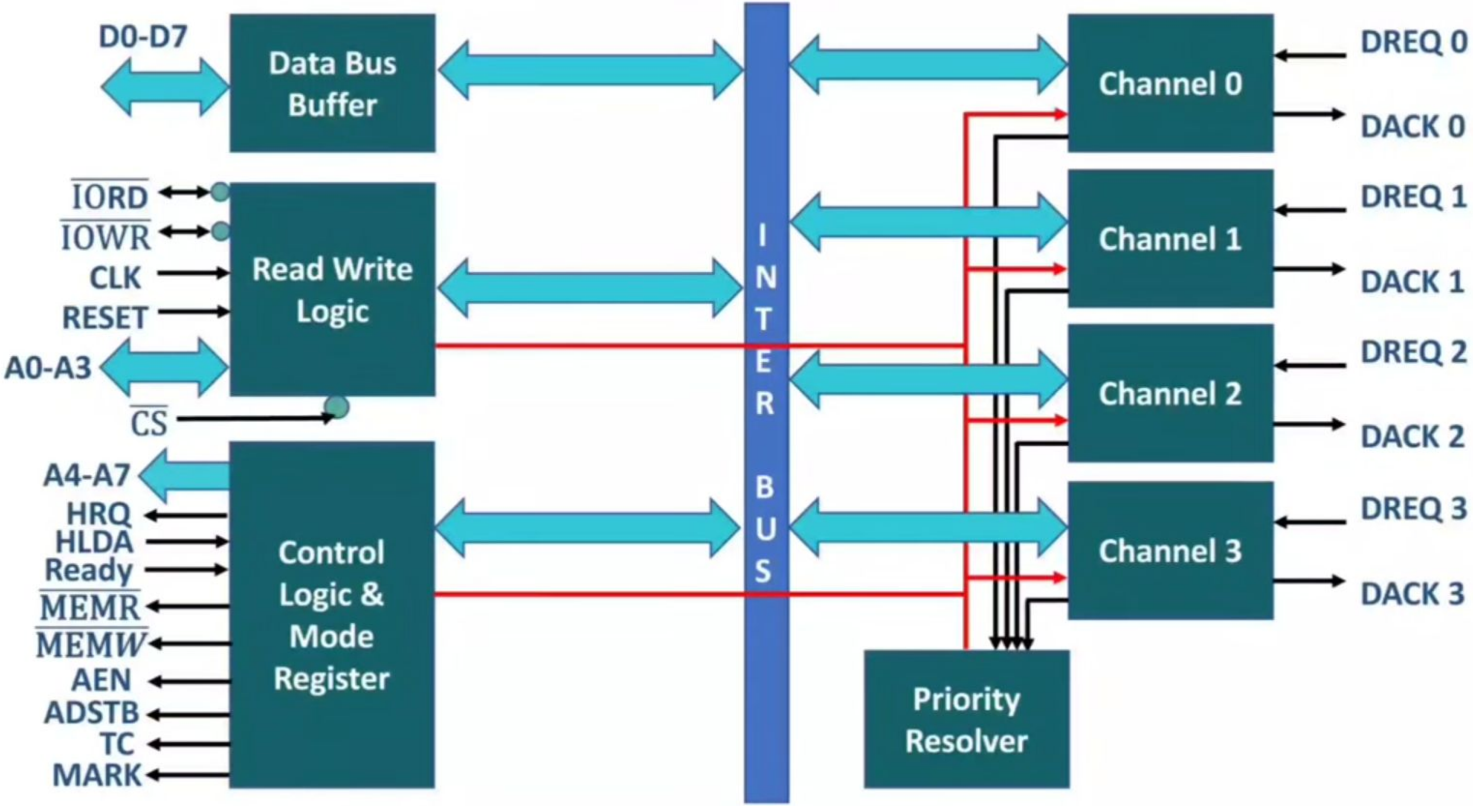
- ❑ If IO wants to send data to memory, then it will send request to DMA. [DRQ]
- ❑ To take control of system buses, DMA will send HOLD signal to MPU.
- ❑ To give control of address and data, MPU will give HLDA [HOLD Acknowledge]. Which indicates that now DMA is master of Buses. So now buses will be managed by DMA for memory and IO.



- ❑ Now data transfer can happen without involvement of MPU. Here now MPU don't need to execute instructions, so data exchange will be faster.
- ❑ Once HOLD signals goes low, MPU will take control of system buses and then MPU becomes master.

E

Block Diagram of 8257

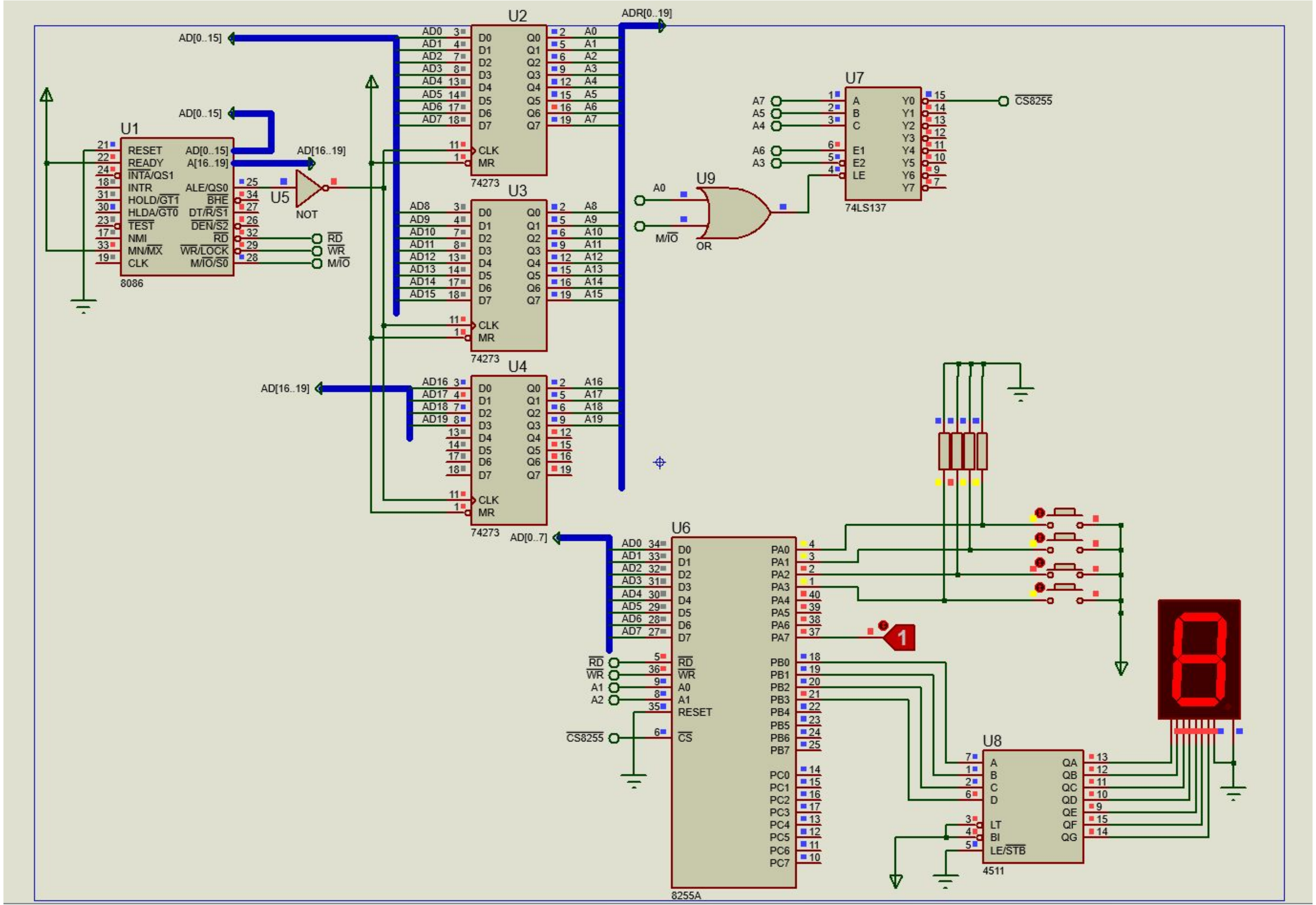


8255 programmable peripheral interface

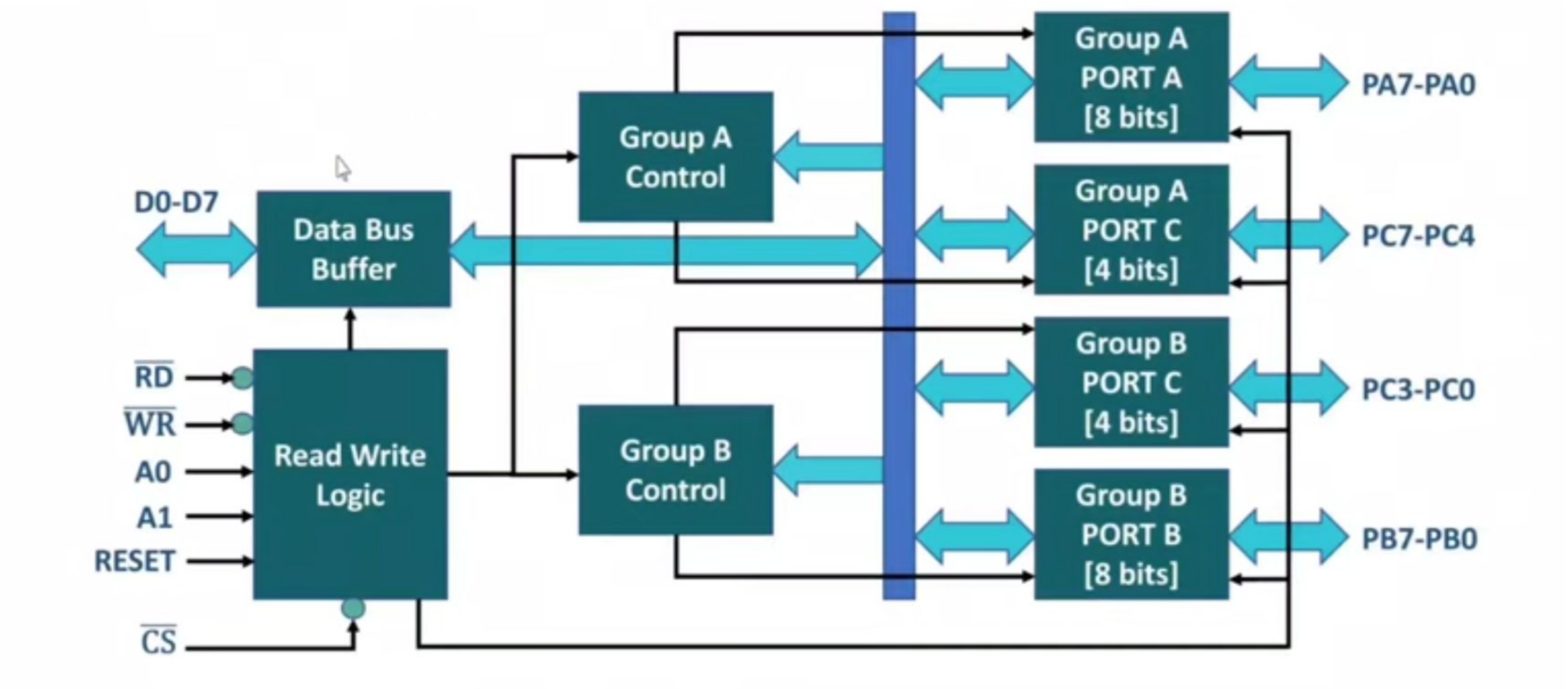
- 8255 is designed to work with various microprocessors like 8085, 8086 etc.
- 8255 is designed to increase capacity of Input Output Interface.
- 8255 has three 8bits bidirectional IO ports.
- 8255 has three IO modes of transfer data:
  - Simple IO mode
  - Handshake IO mode
  - Bidirectional handshake IO mode
- 8255 has BSR mode to alter individual bits of port C.

BSR mode (Bit Set/Reset Mode) is used to set or reset individual bits of the ports. This mode is particularly useful for controlling a single bit of **Port C**.

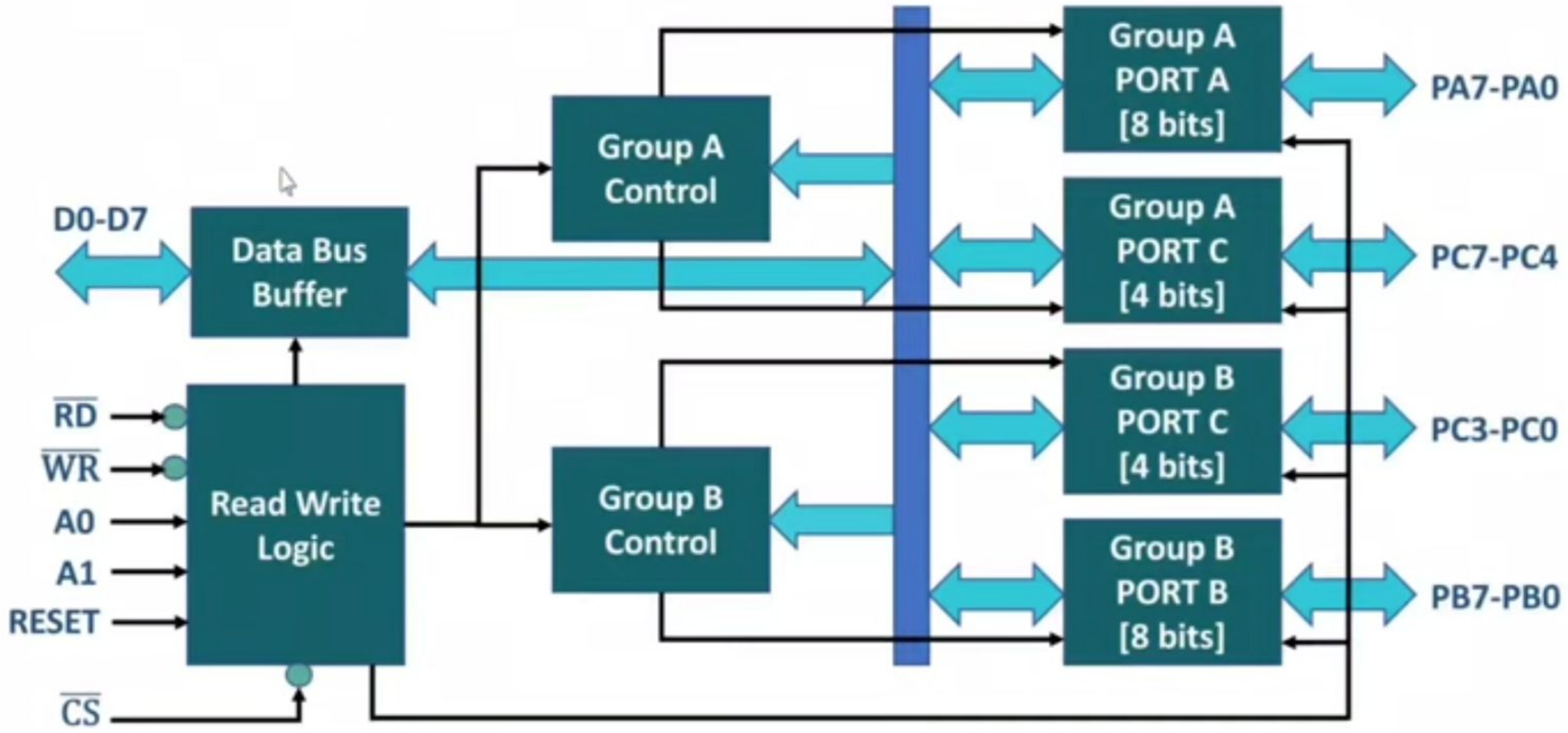
- Any bit of Port C can be set to 1 (set) or 0 (reset).
- This mode only affects Port C; other ports (Port A and Port B) are not influenced by this mode.


U1
8086
RESET
READY
INTA/QS1
INTR
HOLD/GT1
HLDA/GT0
TEST
NMI
MN/MX
CLK
AD[0..15]
A[16..19]
ALE/QS0
BHE
DT/R/S1
DEN/S2
RD
WR/LOCK
M/IO/S0
U5
NOT
RD
WR
M/IO
AD[0..15]
AD[16..19]
U2
74273
U3
74273
U4
74273
D0
D1
D2
D3
D4
D5
D6
D7
Q0
Q1
Q2
Q3
Q4
Q5
Q6
Q7
CLK
MR
ADR[0..19]
U7
74LS137
A7
A5
A4
A6
A3
A
B
C
E1
E2
LE
Y0
Y1
Y2
Y3
Y4
Y5
Y6
Y7
CS8255
U9
OR
A0
M/IO
U6
8255A
AD[0..7]
RD
WR
A0
A1
RESET
CS
A1
A2
CS8255
PA0
PA1
PA2
PA3
PA4
PA5
PA6
PA7
PB0
PB1
PB2
PB3
PB4
PB5
PB6
PB7
PC0
PC1
PC2
PC3
PC4
PC5
PC6
PC7
U8
4511
LT
BI
LE/STB
QA
QB
QC
QD
QE
QF
QG

Block diagram of 8255 programmable peripheral interface



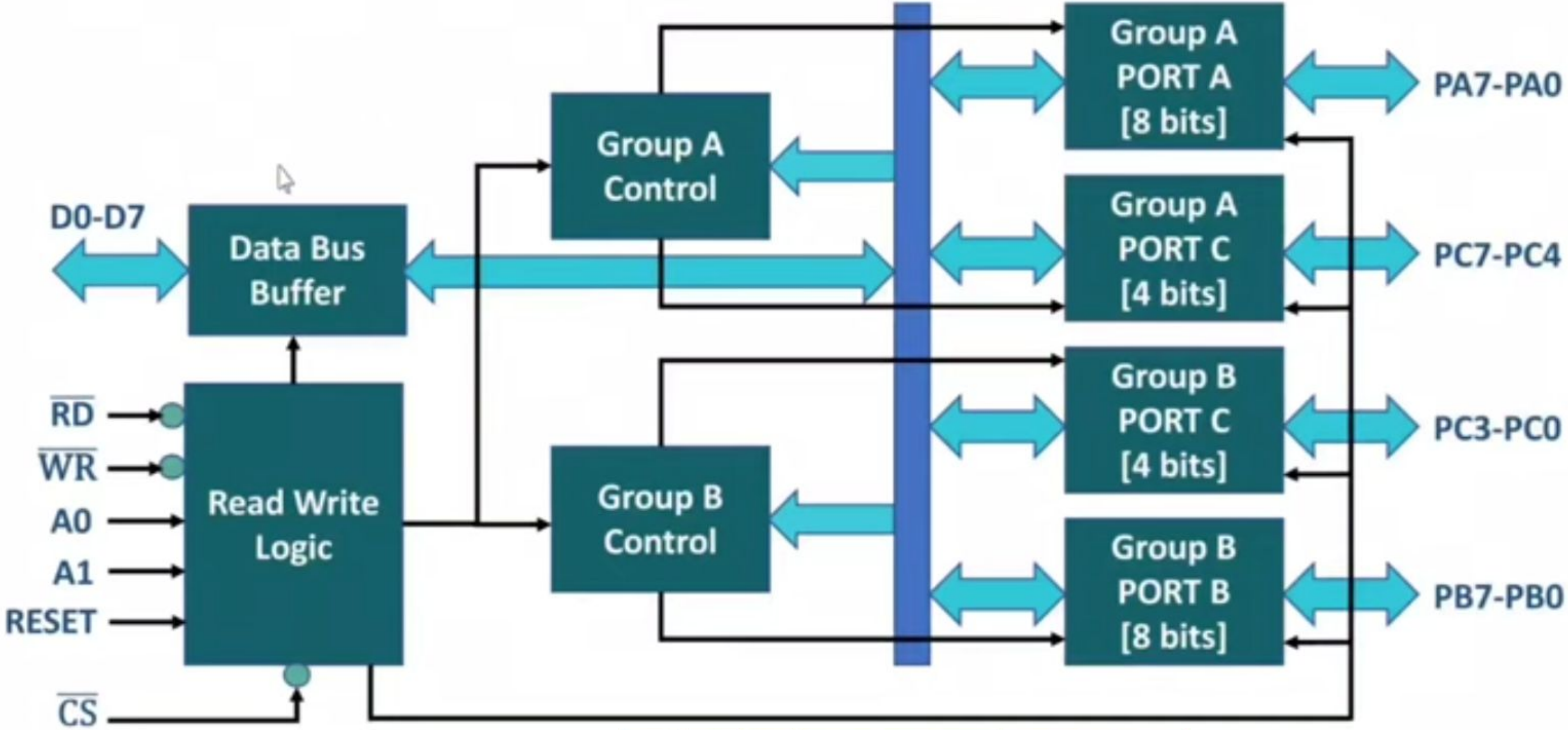
Block diagram of 8255 programmable peripheral interface



❖ Data Bus Buffer

- ❑ It has bidirectional data bus D0 – D7.
- ❑ D0 – D7 is interfaced with system data bus of Microprocessor.

Block diagram of 8255 programmable peripheral interface

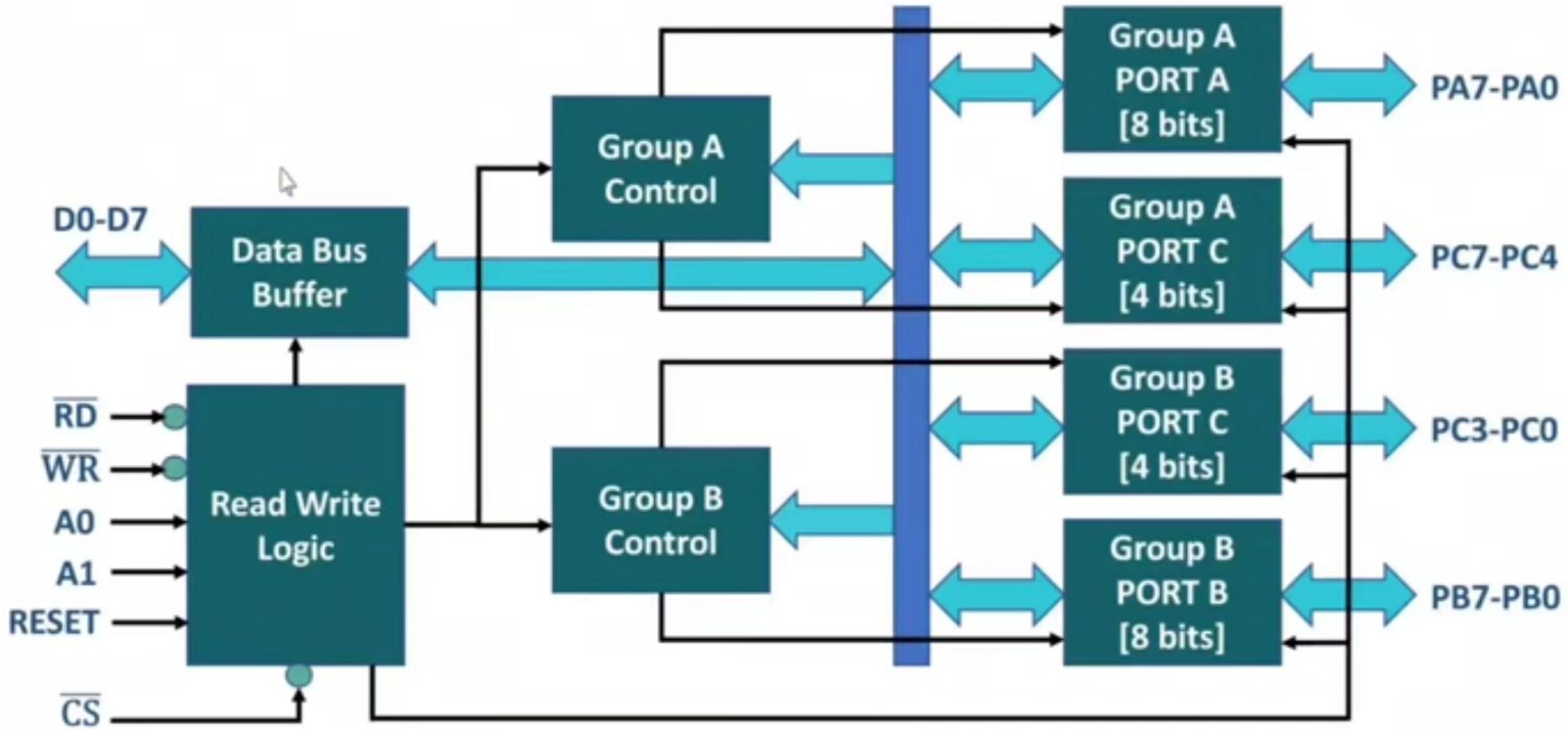


❖ Read Write Control Logic

- ❑ 8255 read and write data as per control signals $\overline{RD}$ and $\overline{WR}$ connected with microprocessor.
- ❑ RESET will reset 8255.
- ❑ A1 and A0 used to select port and control word.
- ❑ $\overline{CS}$ is used to select chip of 8255.

$\overline{CS}$	A1	A0	Selected	Sample Address
0	0	0	Port A	80H [1000 0000]
0	0	1	Port B	81H [1000 0001]
0	1	0	Port C	82H [1000 0010]
0	1	1	Control Register	83H [1000 0011]
1	X	X	8255 is not selected	

Block diagram of 8255 programmable peripheral interface



❖ Group A and Group B Control

- ❑ Group A control is used to control PORT A [PA7 – PA0] and UPPER PORT C [PC7 – PC4].
- ❑ Group B control is used to control PORT B [PB7 – PB0] and LOWER PORT C [PC3 – PC0].
- ❑ It takes control signals from control word and forwards it on respective ports.

Control word and modes of 8255

❖ Modes of 8255

- There are three 8 bits IO ports and works as follows:

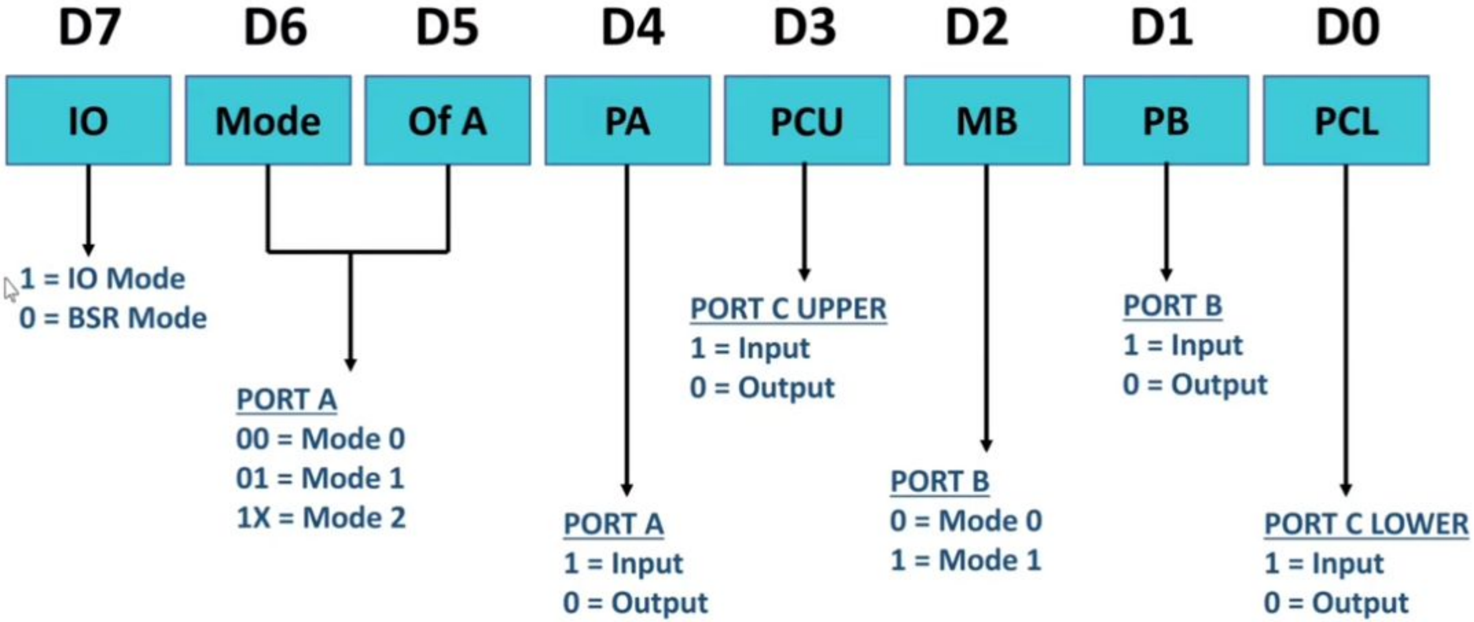
Port	MODE 0	MODE 1	MODE 2	BSR MODE
PORT A	YES	YES	YES	NO
PORT B	YES	YES	NO	NO
PORT C	YES	NO [HS]	NO [HS]	YES

- Here, Mode 0 is simple IO mode.
  - Output are latched and Input are not latched.
  - Port Do not have Interrupt handling capacity.
- Here, Mode 1 is IO mode with handshake.
  - Here each port uses three lines from port C as Handshake signals.
  - Here, Input and Output are latched.
  - Interrupt handling is supported.
- Here, Mode 2 is Bidirectional IO mode with handshake.
  - Here port A uses five lines from port C as Handshake signals.
  - Interrupt handling is supported.

Control word and modes of 8255

❖ Control Word

- ❑ 8255 has 8 bits of control word. It defines working of IO ports A, B and C.



Control word and modes of 8255

❖ BSR [Bit Set Reset] Mode of 8255

❑ BSR Mode only works with PORT C.

D7	D6	D5	D4	D3	D2	D1	D0
IO	X	X	X	BS-2	BS-1	BS0	S/R

IO:
1 = IO Mode
0 = BSR Mode

BIT Select			Bit
0	0	0	PC0
0	0	1	PC1
0	1	0	PC2
0	1	1	PC3
1	0	0	PC4
1	0	1	PC5
1	1	0	PC6
1	1	1	PC7

S/R:
1 = Set bit of Port C
0 = Reset bit of Port C

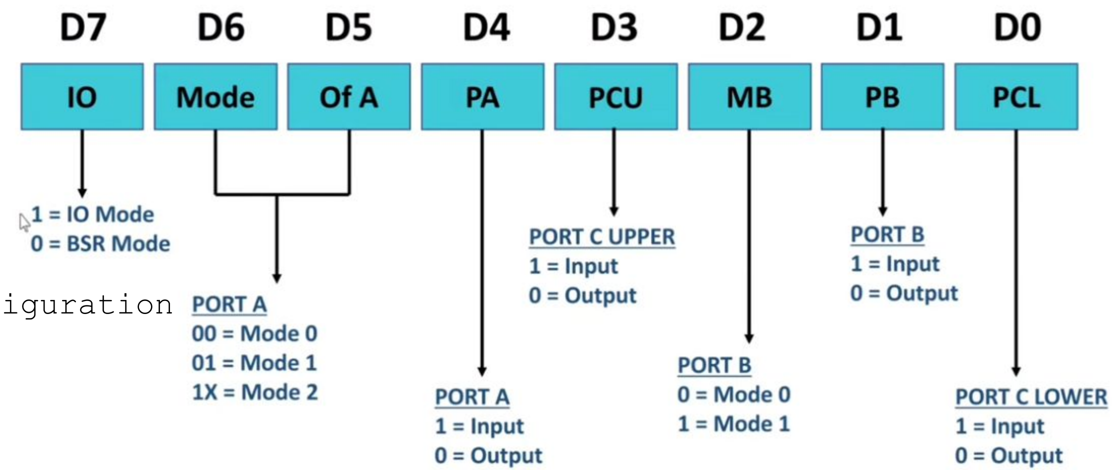
8255 generate control word

Example - 1: Write the control word to configure:

- Port A as input (Mode 1),
- Port B as output (Mode 0),
- Port C upper as input, and lower as output.

Control Word: 1011 1000 = B8h

```
MOV DX, 8003h ;Control register address
MOV AL, 0B8h  ;Control word for the configuration
OUT DX, AL    ;Send to 8255
```



8255 generate control word

Example - 2: Given an 8086 system with 8255 mapped at address 0x8000, write the assembly instructions to:

- Send `0x55` to Port A. (Port A Mode 0)
- Read from Port B. (Port B Mode 0)
- Set bit 3 of Port C using BSR mode.

- Port A: Mode 0, Output
- Port B: Mode 0, Input
- Port C (upper): Output
- Port C (lower): Input

Solution:

Control Register Setup

```
MOV DX, 8003h      ; Control register address
MOV AL, 83h        ; Control word → 10000011b
OUT DX, AL
```

Writing to Port A

```
MOV DX, 8000h      ; Port A address
MOV AL, 55h        ; Data (01010101b)
OUT DX, AL
```

Reading from Port B

```
MOV DX, 8001h      ; Port B address
IN AL, DX
```

Bit Set/Reset (BSR) Mode

```
MOV DX, 8003h     ; Control register
MOV AL, 07h       ; BSR control word
OUT DX, AL
```