CSE321 MICROPROCESSORS

8085 MPU

INTRODUCTION
NUMBER SYSTEMS

- Radix/Base: It is the total number of available digits in a number system
- Some number systems with their radices

Number system	Digits	Radix/Base
Binary	0,1	2
Octal	0,1,2,3,4,5,6,7	8
Decimal	0,1,2,3,4,5,6,7,8,9	10
Hexadecimal	0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F	15

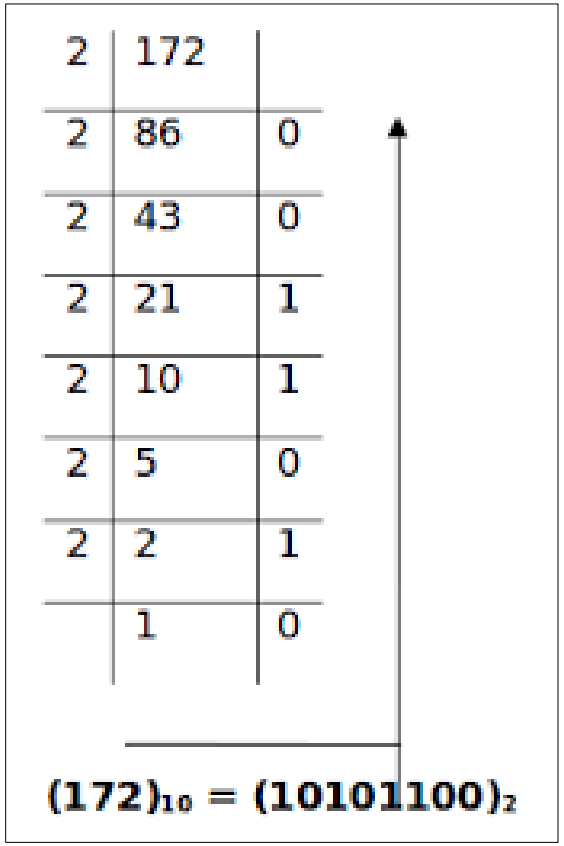
Number system conversions

1. Decimal (X_{10}) to X_2, X_8, X_{10}

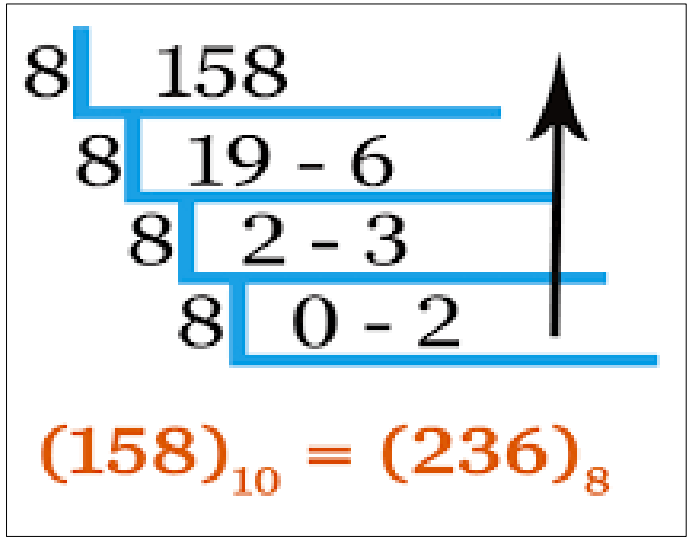
Steps involved:

1. Divide the decimal number with the base of the target number system
2. As a result of division, we get Quotient and Remainder, the first remainder is LSB of the converted bit
3. Continue division until the number is indivisible, the final remainder is the MSB
4. Write the converted bit from MSB through LSB

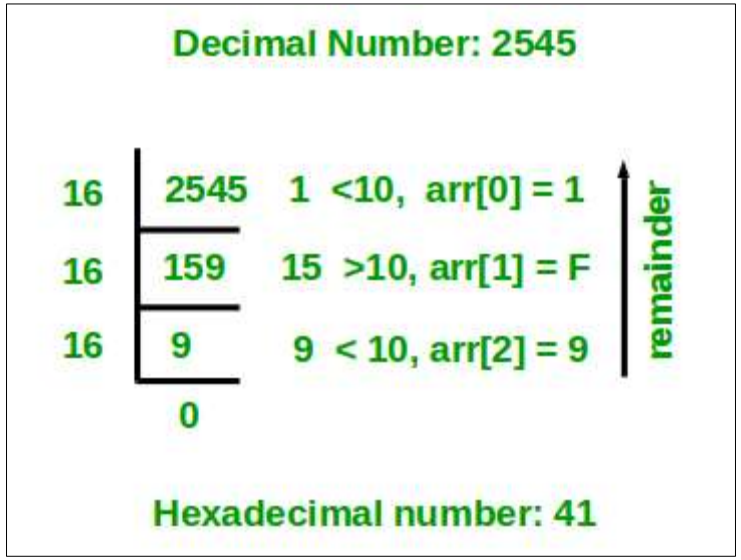
Examples for Decimal to other number systems



a. Decimal to binary conversion



b. Decimal to octal conversion

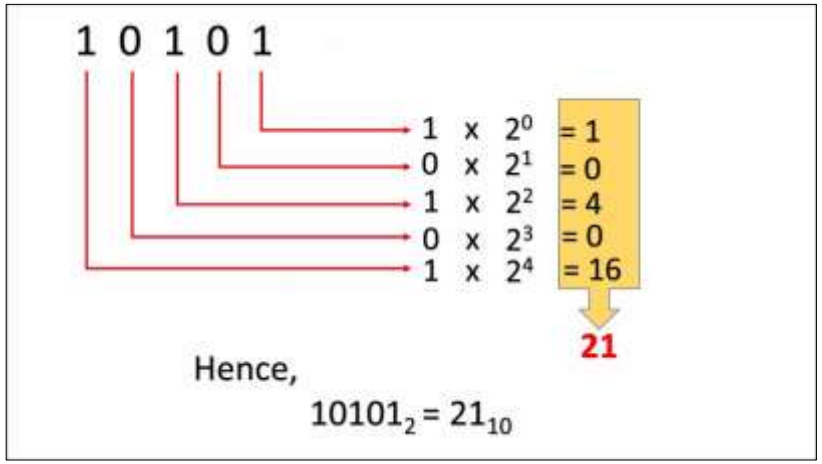


c. Decimal to hexadecimal conversion

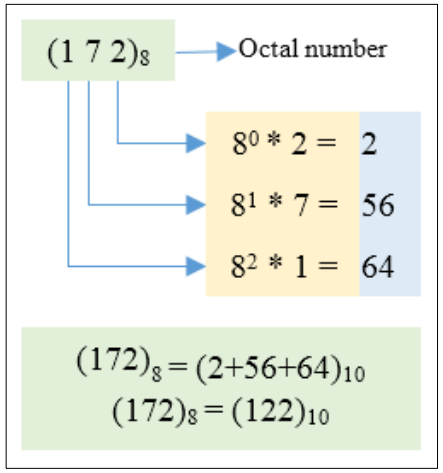
2. Any number to Decimal

Steps involved:

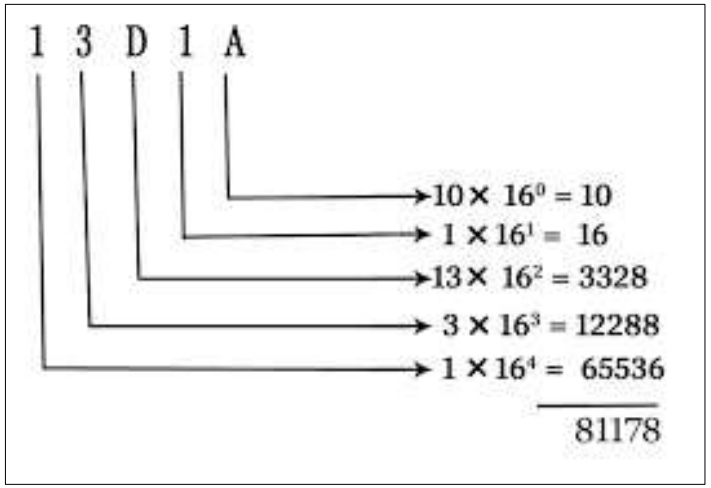
1. Multiply each digit of the given number with the base of the source number system raised to the power of place value of the corresponding digit
2. Do the sum of all product terms, the sum is the converted decimal number



a. Binary to Decimal conversion



b. Octal to Decimal conversion

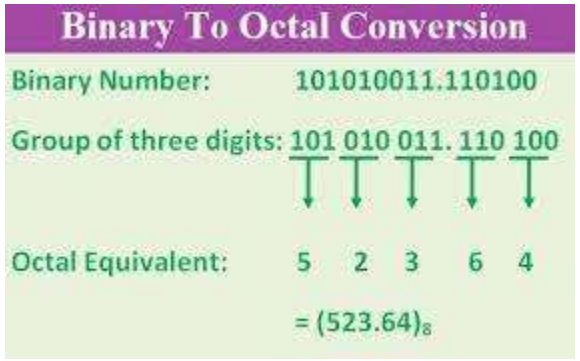


c. Hexadecimal to Decimal conversion

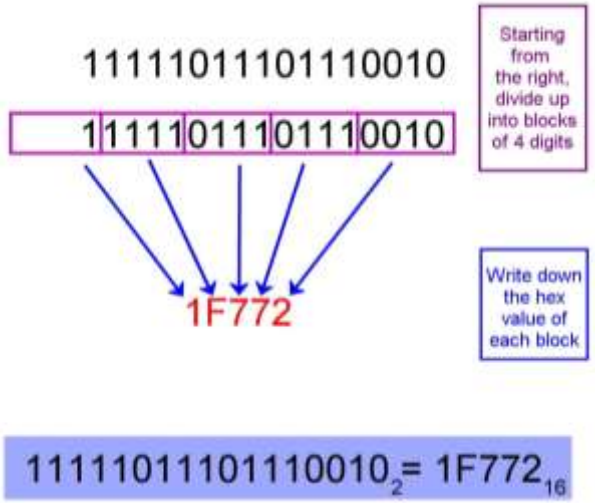
3. Binary to Octal & hexadecimal systems

Steps involved:

1. For binary to Octal conversion, group the given binary number from LSB in 3 digits, then write the corresponding value for that group
2. For binary to Hexadecimal conversion, group the given binary number from LSB in 4 digits, then write the corresponding value for that group.



a. Binary to Octal conversion

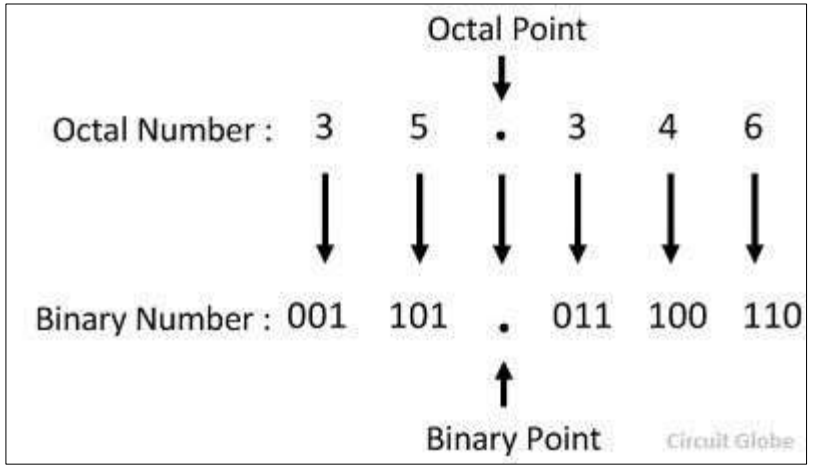


b. Binary to Hexadecimal conversion

4. Octal and Hexadecimal to binary number system

Steps involved:

1. For Octal to binary write each digit of the octal number in 3 digits as seen from the table below
2. For Hexadecimal to binary write each digit of the octal number in 4 digits as seen from the table below



a. Octal to binary conversion

Number	0	1	2	3	4	5	6	7
Binary	0000	0001	0010	0011	0100	0101	0110	0111
Hexadecimal	0	1	2	3	4	5	6	7

Number	8	9	10	11	12	13	14	15
Binary	1000	1001	1010	1011	1100	1101	1110	1111
Hexadecimal	8	9	A	B	C	D	E	F

Circuit Globe

b. Hexadecimal to binary conversion

5. Octal to Hexadecimal number system & vice-versa

Steps involved:

- *Octal to Hexadecimal:*
  1. There's no direct method, convert the octal to binary by writing each digit in 3 digits
  2. Convert binary to Hex by grouping as 4 and writing its hex equivalent

- *Hexadecimal to Octal:*

  There's no direct method, but there are two routes to do this,
  - Route 1: Via Decimal (Hex to decimal to octal)
  - Route 2: Via Binary (Hex to binary to octal)

5. Octal to Hexadecimal number system & vice-versa

Step 1:

Octal to Binary Conversion

7	5	2
111	101	010

So the binary equivalent is 111101010

Step 2:

Binary to Hex Conversion

0001	1110	1010
1	D	9

a. Octal to Hexadecimal conversion

Example:

Convert the hexadecimal number $5A_H$ into its octal equivalent.

Solution:

First convert the hexadecimal number into its decimal equivalent, then convert the decimal number into its octal equivalent.

5	A
16^1	16^0
16	1
80 + 10 = 90_{10}	

$8\overline{)90}$ r = 2 ← LSB

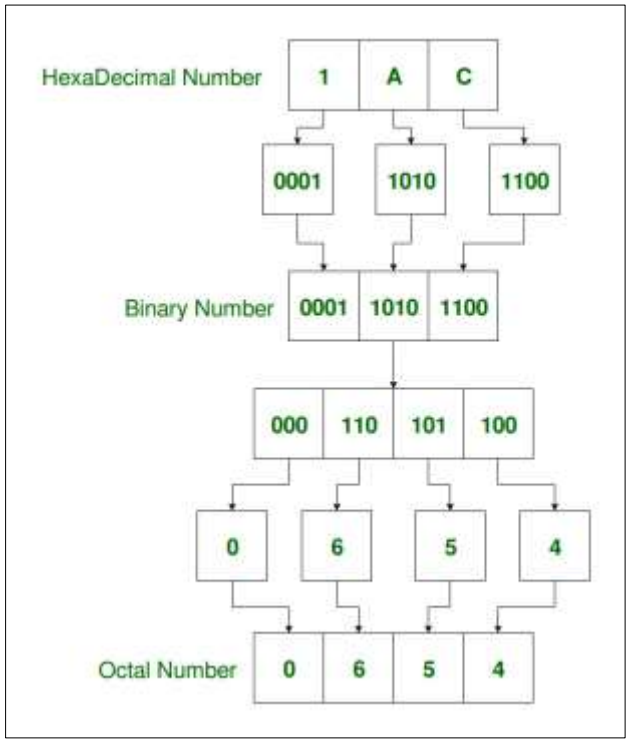
$8\overline{)11}$ r = 3

$8\overline{)1}$ r = 1 ← MSB

$\therefore 5A_H = 132_8$

20

b. Hex to Oct via decimal



c. Hex to Oct via binary

Basic Concepts of Microprocessors

- Differences between:
  - Microcomputer – a computer with a microprocessor as its CPU. Includes memory, I/O etc.
  - Microprocessor – silicon chip which includes ALU, register circuits & control circuits
  - Microcontroller – silicon chip which includes microprocessor, memory & I/O in a single package.

What is a Microprocessor?

- The word comes from the combination micro and processor.
  - Processor means a device that processes whatever. In this context processor means a device that processes numbers, specifically binary numbers, 0's and 1's.
    - To process means to manipulate. It is a general term that describes all manipulation. Again in this content, it means to perform certain operations on the numbers that depend on the microprocessor's design.

Definition (Contd.)

- Lets expand each of the underlined words:
  - Programmable device: The microprocessor can perform different sets of operations on the data it receives depending on the sequence of instructions supplied in the given program.
  - By changing the program, the microprocessor manipulates the data in different ways.
  - Instructions: Each microprocessor is designed to execute a specific group of operations. This group of operations is called an instruction set. This instruction set defines what the microprocessor can and cannot do.

Definition (Contd.)

– Takes in: The data that the microprocessor manipulates must come from somewhere.

- It comes from what is called "input devices".
- These are devices that bring data into the system from the outside world.
- These represent devices such as a keyboard, a mouse, switches, and the like.

Definition (Contd.)

- Numbers: The microprocessor has a very narrow view on life. It only understands binary numbers.
- A binary digit is called a bit (which comes from **b**inary dig**it**).
- The microprocessor recognizes and processes a group of bits together. This group of bits is called a "word".
- The number of bits in a Microprocessor's word, is a measure of its "abilities".

Definition (Contd.)

- Words, Bytes, etc.
  - The earliest microprocessor (the Intel 8088 and Motorola's 6800) recognized 8-bit words.
  - Later microprocessors (8086 and 68000) were designed with 16-bit words.
    - A group of 8-bits were referred to as a "half-word" or "byte".
    - A group of 4 bits is called a "nibble".
    - Also, 32 bit groups were given the name "long word".
  - Today, all processors manipulate at least 32 bits at a time and there exists microprocessors that can process 64, 80, 128 bits

Definition (Contd.)

ASIP

- Arithmetic and Logic Operations:
  - Every microprocessor has arithmetic operations such as add and subtract as part of its instruction set.
    - Most microprocessors will have operations such as multiply and divide.
    - Some of the newer ones will have complex operations such as square root.
  - In addition, microprocessors have logic operations as well. Such as AND, OR, XOR, shift left, shift right, etc.
  - Again, the number and types of operations define the microprocessor's instruction set and depends on the specific microprocessor.

INSTRUCTION SET OF 8085

1. Data transfer operations
2. Arithmetic instructions
3. Logical instructions
4. Branch instructions
5. Stack, I/O and machine instructions
6. Additional instructions

DATA TRANSFER INSTRUCTIONS

Mnemonics		Examples	Operation
1.1	MVI R,** 8-bit	MVI B, 4FH	Load 8-bit data (byte) in a register
1.2	MOV Rd, Rs**	MOV B, A MOV C, B	Copy data from source register Rs into destination register Rd
1.3	LXI Rp,** 16-bit	LXI B, 2050H	Load 16-bit number in a register pair
1.4	OUT 8-bit (port address)	OUT 01H	Send (write) data byte from the accumulator to an output device
1.5	IN 8-bit (port address)	IN 07H	Accept (read) data byte from an input device and place it in the accumulator
1.6	LDA 16-bit	LDA 2050H	Copy the data byte into A from the memory specified by 16-bit address
1.7	STA 16-bit	STA 2070H	Copy the data byte from A into the memory specified by 16-bit address
1.8	LDAX Rp	LDAX B	Copy the data byte into A from the memory specified by the address in the register pair
1.9	STAX Rp	STAX D	Copy the data byte from A into the memory specified by the address in the register pair

DATA TRANSFER INSTRUCTIONS

1.10	MOV R, M	MOV B, M	Copy the data byte into register from the memory specified by the address in HL register
1.11	MOV M, R	MOV M, C	Copy the data byte from the register into memory specified by the address in HL register

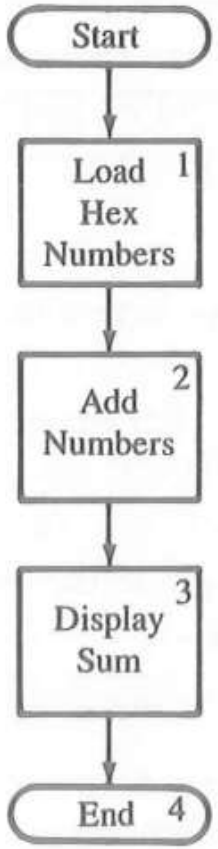
ARITHMETIC INSTRUCTIONS

Mnemonics		Examples	Operation
2.1	ADD R	ADD B	Add the contents of a register to the register to the contents of A
2.2	ADI 8-bit	ADI 37H	Add 8-bit data to the contents of A
2.3	ADD M	ADD M	Add the contents of memory to A; the address of memory is in HL register
2.4	SUB R	SUB C	Subtract the contents of a register from the contents of A
2.5	SUI 8-bit	SUI 7FH	Subtract 8-bit data from the contents of A
2.6	SUB M	SUB M	Subtract the contents of memory from A; the address of memory is in HL register
2.7	INR R	INR D	Increment the contents of a register
2.8	INR M	INR M	Increment the contents of memory, the address of which is in HL
2.9	DCR R	DCR E	Decrement the contents of a register
2.10	DCR M	DCR M	Decrement the contents of a memory, the address of which is in HL
2.11	INX Rp	INX H	Increment the contents of a register pair
2.12	DCX Rp	DCX B	Decrement the contents of a register pair

A SAMPLE PROGRAM FOR DOING BASIC OPERATIONS

To write an assembly language program, we need to translate the blocks shown in the flowchart into 8085 operations and then, subsequently, into mnemonics. By examining the blocks, we can classify them into three types of operations: Blocks 1 and 3 are copy op-

FIGURE 2.3
Flowchart: Adding Two Numbers



A SAMPLE PROGRAM FOR DOING BASIC OPERATIONS

Block 1:	MVI A,32H	Load register A with 32H
	MVI B,48H	Load register B with 48H
Block 2:	ADD B	Add two bytes and save the sum in A
Block 3:	OUT 01H	Display accumulator contents at port 01H
Block 4:	HALT	End

MEMORY OCCUPANCY OF THE ABOVE ALP

Mnemonics	Hex Code	
MVI A,32H	3E	2-byte instruction
	32	
MVI B,48H	06	2-byte instruction
	48	
ADD B	80	1-byte instruction
OUT 01H	D3	2-byte instruction
	01	
HLT	76	1-byte instruction

INSTRUCTION SET OF 8085: LOGICAL INSTRUCTIONS:

INSTRUCTION	OPERATION	MEANING	EFFECT ON FLAG REGISTER
ANA R	A= A ^ R	Logical AND on A and R contents and stored in A	CY=0, AC=1, S, P, and Z get affected
ANA M	A= A ^ M	Logical AND on A and on data present in address specified by HL and stored in A	
ANI 8b address	A= A ^ 8b data.	Logical AND on A and 8 bit data and stored in A	
ORA R	A= A ∨ C	Logical OR on A and R contents and stored in A	CY=0, AC=0, S, P, and Z get affected
ORI M	A= A ∨ M	Logical OR on A and on data present in address specified by HL and stored in A	
ORI 8b address	A= A ∨ 8b data.	Logical OR on A and 8 bit data and stored in A	
XRA R	A= A ⊕ R	Logical XOR on A and R contents and stored in A	
XRA M	A= A ⊕ M	Logical XOR on A and on data present in address specified by HL and stored in A	
XRA 8b address	A= A ⊕ 8b data	Logical XOR on A and 8 bit data and stored in A	

INSTRUCTION SET OF 8085: LOGICAL INSTRUCTIONS:

INSTRUCTION	OPERATION	Meaning	Effect on Flag register
CMP R	A= A-R	If A<R then, set carry flag to 1 and Z= 0; If A=R then, set carry flag to 0 and Z=1; If A>R then, set carry flag to 0 and Z=0.	
CMP M	A= A-M	If A<Data from address from HL pair, then, set carry flag to 1 and Z= 0; If A=Data from address from HL pair, then, set carry flag to 0 and Z=1; If A>Data from address from HL pair, then, set carry flag to 0 and Z=0.	
CPI 8b data	A= A-8b data	If A<Data, then, set carry flag to 1 and Z= 0; If A=Data, then, set carry flag to 0 and Z=1; If A>Data, then, set carry flag to 0 and Z=0.	
RLC	Rotate left w/o carry	Rotate the 8 bits in the given reg from MSB to LSB without considering the effect of carry	CY=MSB
RRC	Rotate right w/o carry	Rotate the 8 bits in the given reg from LSB to MSB without considering the effect of carry	CY= LSB
RAL	Rotate left with carry	Rotate the 8 bits in the given reg from MSB to LSB **THROUGH** carry	CY=MSB
RAR	Rotate right with carry	Rotate the 8 bits in the given reg from LSB to MSB **THROUGH** carry	CY= LSB

INSTRUCTION SET OF 8085: LOGICAL INSTRUCTIONS:

INSTRUCTION	OPERATION	Meaning	Effect on Flag register
CMA	Compliment Acc.	A= $\overline{A}$	No effect on flags
CMC	Compliment carry	C= $\overline{C}$	Only CY
STC	Set carry to '1'	C=1	Only CY=1

BRANCH INSTRUCTIONS:

JUMP
- Conditional
- Unconditional

CALL
- Conditional
- Unconditional

RETURN
- Conditional
- Unconditional

INSTRUCTION SET OF 8085: BRANCH INSTRUCTIONS:

- Conditional branching take place based on the condition of the flags like Carry, Zero etc.,

- Some conditions are:
  1. NZ– Non Zero
  2. Z– Zero (Z==1)
  3. C– Carry (CY==1)
  4. NC– No Carry (CY==0)
  5. PE– Parity is even (P==1)
  6. PO– Parity is odd (P==0)
  7. P– Sign bit is zero (S==0)
  8. M– Sign bit is minus (S==1)

JUMP	CALL (SP=SP-2)	RETURN (SP=SP+2)
JNZ	CNZ	RNZ
JZ	CZ	RZ
JC	CC	RC
JNC	CNC	RNC
JPE	CPE	RPE
JPO	CPO	RPO
JP	CP	RP
JM	CM	RM

STACK, I/O AND MACHINE INSTRUCTIONS:

INSTRUCTION	Meaning	Effect on Flag register
PUSH	Push two bytes of data into stack	No effect.
POP	Pop two bytes of data from top of the stack	
HTHL	Exchange HL pair contents with two bytes of data from top two rows of a stack	
SPHL	Move contents of HL pair into stack.	
IN	Input data into Accumulator	
OUT	Output data from Accumulator	
EI	Enable Interrupts	
DI	Disable Interrupts: {RST 7.5, 6.5, 5.5, & INTR will be disabled}	
HLT	Microprocessor is Halted.	
NOP	No operation.	

ADDITIONAL INSTRUCTIONS

INSTRUCTION	Meaning
DAA	Decimal Adjust Accumulator //Note: It is always preceded by ADD operation

EXAMPLE FOR DAA:

MVI A, 15H
MVI B, 07H
ADD B
DAA
HLT

0	0	0	1	0	1	0	1	Accumulator
0	0	0	0	0	1	1	1	B register
0	0	0	1	1	1	0	0	Acc after execution
1				C				Hex eq. of Acc after ex.
2				2				After BCD error correction

ADDITIONAL INSTRUCTIONS

INSTRUCTION	Meaning
SIM	Set Interrupt Mask

EXAMPLE FOR SIM:

SOD	SDE	X	R7.5	MSE	M 7.5	M 6.5	M 5.5	8 bit alignment
1	1	0	0	1	1	1	1	

Serial Output Data

Serial Data Enable

Reset 7.5

Mask Set Enable

Mask/Unmask interrupts 7.5, 6.5 and 5.5

0 for Unmasking

1 for Masking

Write a program to mask interrupts 7.5, 6.5 & serial output of logic 1

Program:

```
MVI A, CE
SIM
```

ADDITIONAL INSTRUCTIONS

INSTRUCTION	Meaning
RIM	Read Interrupt Mask

EXAMPLE FOR RIM:

SID	I_7	I_6	I_5	IE	$I_{7.5}$	$I_{6.5}$	$I_{5.5}$	8 bit alignment
1	1	1	1	1	1	1	1	Used to find the status of interrupts

If 1, represents Interrupts are pending

If 1, represents Interrupts are masked

Serial Data input

Interrupt Enable

SOFTWARE INTERRUPTS

INSTRUCTION	OP.CODE	ADDRESS
RST 0	C7 H	0000 H
RST 1	CF H	0008 H
RST 2	D7 H	0010 H
RST 3	DF H	0018 H
RST 4	E7 H	0020 H
RST 5	EF H	0028 H
RST 6	F7 H	0030 H
RST 7	FF H	0038 H

DESCRIPTION

- To store RST 5 we need to store EF H which is 1 byte so all these are 1 byte instructions
- RST are similar to CALL because it pushes the address of next instruction to stack before branching to interrupt service routine (ISR)
- Then after ISR completion it comes back to original program by popping the address from the stack

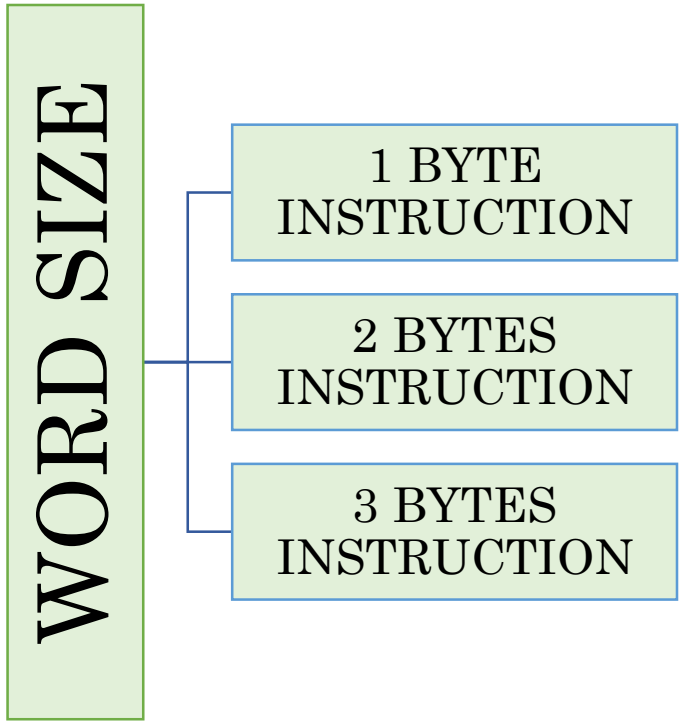
INSTRUCTION WORD SIZE

COMPONENTS OF AN INSTRUCTION

OPCODE + OPERAND = INSTRUCTION (MNEUMONICS)

MVI A, 05H

OPCODE OPERANDS



ONE BYTE INSTRUCTION

A 1-byte instruction includes the opcode and the operand in the same byte. For example:

Task	Opcode	Operand*	Binary Code	Hex Code
Copy the contents of the accumulator in register C.	MOV	C,A	0100 1111	4FH
Add the contents of register B to the contents of the accumulator.	ADD	B	1000 0000	80H
Invert (complement) each bit in the accumulator.	CMA		0010 1111	2FH

TWO BYTE INSTRUCTION

In a 2-byte instruction, the first byte specifies the operation code and the second byte specifies the operand. For example:

Task	Opcode	Operand	Binary Code	Hex Code	
Load an 8-bit data byte in the accumulator.	MVI	A,32H	0011 1110	3E	First Byte
			0011 0010	32	Second Byte
Load an 8-bit data byte in register B.	MVI	B,F2H	0000 0110	06	First Byte
			1111 0010	F2	Second Byte

These instructions would require two memory locations each to store the binary codes. The data bytes 32H and F2H are selected arbitrarily as examples.

THREE BYTE INSTRUCTION

In a 3-byte instruction, the first byte specifies the opcode, and the following two bytes specify the 16-bit address. Note that the second byte is the low-order address and the third byte is the high-order address. For example:

Task	Opcode	Operand	Binary Code	Hex Code*	
Load contents of memory 2050H into A.	LDA	2050H	0011 1010	3A	First Byte
			0101 0000	50	Second Byte
			0010 0000	20	Third Byte
Transfer the program sequence to memory location 2085H.	JMP	2085H	1100 0011	C3	First Byte
			1000 0101	85	Second Byte
			0010 0000	20	Third Byte

These instructions would require three memory locations each to store the binary codes.

OPCODE FORMAT

Code	Registers
000	B
001	C
010	D
011	E
100	H
101	L
111	A
110	Reserved for Memory-Related operation

Code	Register Pairs
00	BC
01	DE
10	HL
11	AF OR SP

OPCODE FORMAT

Some of the operation codes are identified as follows:

Function	Operation Code
1. Rotate each bit of the accumulator to the left by one position.	00000111 = 07H (8-bit opcode)
2. Add the contents of a register to the accumulator.	10000 SSS (5-bit opcode—3 bits are reserved for a register)

This instruction is completed by adding the code of the register. For example,

Add	:	10000
Register B	:	000
to A	:	Implicit
Binary Instruction:		$\underbrace{10000}_{\text{Add}}\ \underbrace{000}_{\text{Reg.B}}$ = 80H

In assembly language, this is expressed as

Opcode	Operand	Hex Code
ADD	B	80H

OPCODE FORMAT

3. MOVE (Copy) the content of register Rs (source) to register Rd (destination)	01 2-bit Opcode for MOVE	DDD Reg. Rd	SSS Reg. Rs

This instruction is completed by adding the codes of two registers. For example,

Move (copy) the content:		0 1
To register C	:	0 0 1 (DDD)
From register A	:	1 1 1 (SSS)
Binary Instruction	:	$\underbrace{0\ 1}_{\text{Opcode}}\ \underbrace{0\ 0\ 1\quad 1\ 1\ 1}_{\text{Operand}} \rightarrow 4FH$

In assembly language, this is expressed as

Opcode	Operand	Hex Code
MOV	C,A	4F

DATA FORMAT

1. ASCII code

- *Americal Standard Code for Information Interchange*
- *It is a 7-bit alphanumeric code that represents numbers, alphabets, and nonprintable characters like carriage return.*
- *ASCII (Extended): Has one extra bit (8 bits in total) that denotes graphical characters*

2. BCD code

- *Binary Coded Decimal, is a binary code for decimal numbers (0-9), so we need 4 bits to represent BCD numbers, numbers from 1010 to 1111 are invalid.*

DATA FORMAT

3. Signed integer

- *For a 8-bit processor like 8085, the MSB signifies the sign bit, if the MSB is 1 then it is a negative number, if it is 0 then it is a positive number.*
- *So the maximum possible positive number that 8085 can process is 0111 1111 (07H), likewise the negative numbers are 1000 0000 (80 H) to 1111 1111 (FF H)*

4. Unsigned integers

- *Here all the 8 bits are used to signify the magnitude of the number, FF H is the maximum processable 8-bit number by 8085*
- *However, 8085 can process data that are greater than 8-bits by dividing them in group of 8-bits*

ADDRESSING MODES OF 8085

1. Direct addressing mode
2. Register addressing mode
3. Register indirect addressing mode
4. Immediate addressing mode
5. Implicit addressing mode

Direct Addressing Mode

- In direct addressing, the address of the data is given in the instruction itself
- Example: STA 2400H
- Corresponding opcode for this instruction: 32, 00, 24
- Here 32: represents source is from Accumulator

  00: Represents lower order address

  24: Represents higher order address

Register Addressing Mode

- In this, operand is one of the general purpose registers
- Example: MOV A, B
- Corresponding opcode for this instruction: 01 111 000 (78 H)
- Here 01 denotes Move operation, 111 denotes Destination register (accumulator) and 000 denotes Source register

Register Indirect Addressing Mode

- In this, the address of the operand is denoted by a register pair
- Example:

  LXI H, 2500 H

  MOV A, M

- Address is loaded to HL pair first, then data in that address is moved to Accumulator

Immediate Addressing Mode

- In this, the operand is specified within the instruction itself
- Example:

  LXI H, 2500 H

  LDI A, FF H

- 16-bit value is loaded into HL pair (from first example)
- 8-bit value is loaded into Accumulator (from second example)

Implicit Addressing Mode

- These instructions operate on the content of the Accumulator
- Hence, they do not require the address of the operand or second operand
- Example:

  RAL

  RAR

  RLC

  RRC etc.,

Illustrative Program

Column 1	Column 2		Column 3
Instructions	Memory Addresses	Hex Code	Comments
LDA 2051H	2030	3A	Copy the first byte, 49H, from memory location 2051H into A
	2031	51	
	2032	20	
MOV B, A	2033	47	Save the first byte in B
LDA 2052H	2034	3A	Copy the second byte, 9FH, from memory location 2052H into A
	2035	52	
	2036	20	
SUB B	2037	90	Subtract 49H from 9FH and save the result in A
STA 2053H	2038	32	Save the result in memory location 2053H
	2039	53	
	203A	20	
HLT or RST7	203B	76/FF	End of the program
	2051	49	
	2052	9F	These data bytes must be manually loaded; they are not part of writing the program
	2053	00	

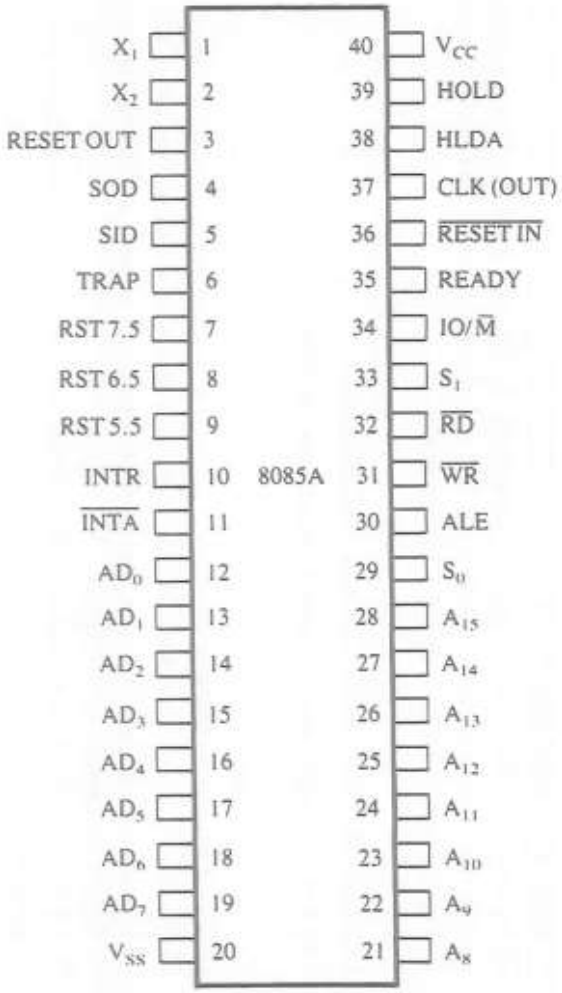
THE 8085 MPU

The term **microprocessing unit** (MPU) is similar to the term **central processing unit** (CPU) used in traditional computers. We define the MPU as a device or a group of devices (as a unit) that can communicate with peripherals, provide timing signals, direct data flow, and perform computing tasks as specified by the instructions in memory. The unit will have the necessary lines for the address bus, the data bus, and the control signals, and would require only a power supply and a crystal (or equivalent frequency-determining components) to be completely functional.

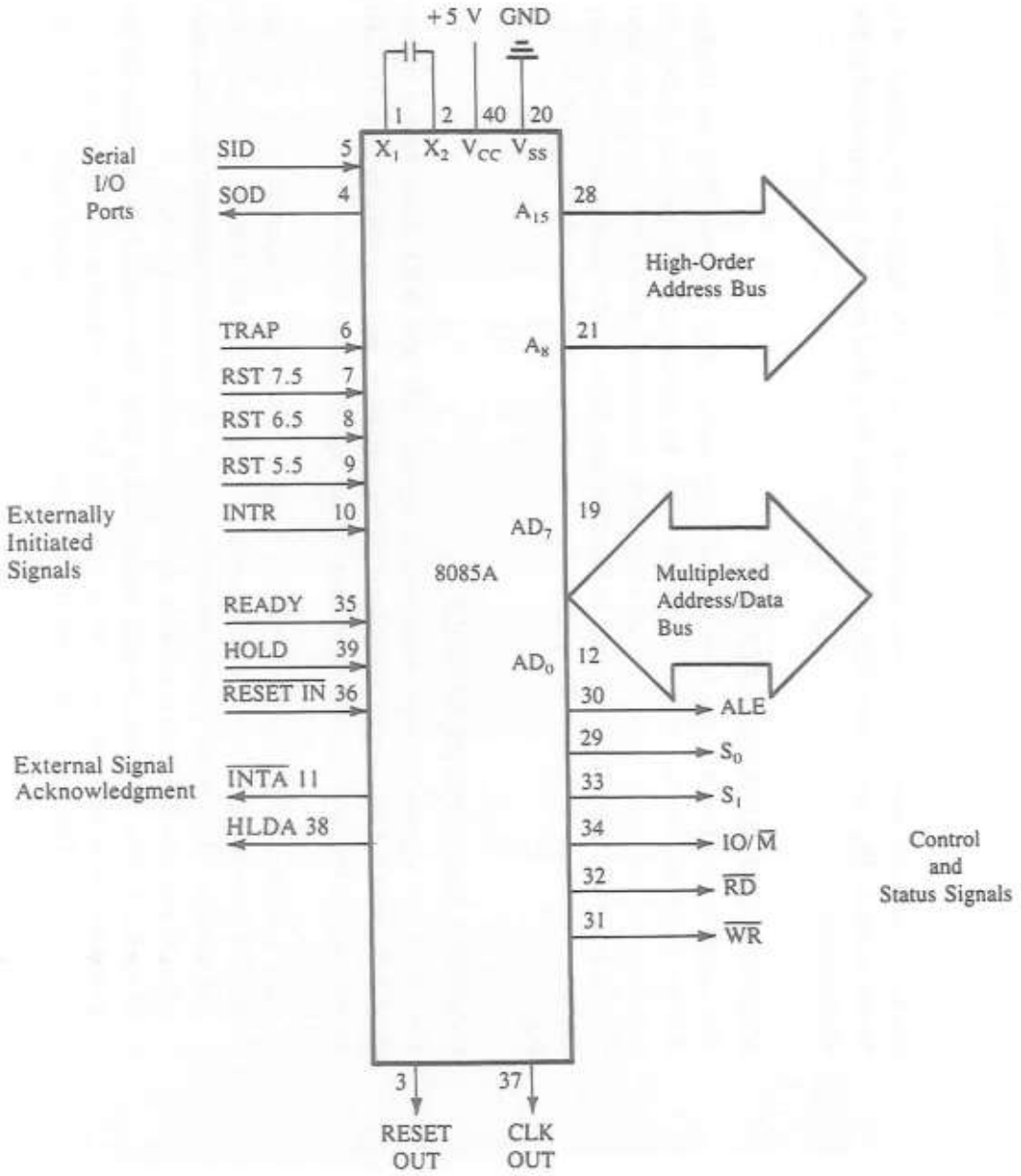
LIMITATIONS OF 8085 AS AN MPU

- The lower order address of 8085 is multiplexed (time-shared) with data bus and hence, needs to be demultiplexed
- Appropriate control signals need to be generated to interface memory and I/O devices

PIN DIAGRAM OF 8085 IC



8085 Pinout



8085 MP

1. Power requirement: +5V
2. Clock (frequency of operation): 3 MHz to 5 MHz (8085A v2)
3. IC: 40 pin IC
4. Memory: Capable of addressing 64K memory

Signal Classification

1. Address bus
2. Data bus
3. Control and status signals
4. Power supply and frequency signals
5. Externally initiated signals
6. Serial I/O signals

Address bus

- There are 16 address lines
- They are divided into two segments
- A_{15}-A_8 and AD_7-AD_0
- A_{15}-A_8: is the higher order address lines and they are unidirectional

Mutliplexed address/data lines

- AD_7-AD_0: is the lower order address as well as data lines (time-slotted); hence bidirectional
- During the execution of an instruction, initially this lower order address lines contain address, but later during execution they contain data

CONTROL AND STATUS SIGNALS

This group of signals includes two control signals ($\overline{RD}$ and $\overline{WR}$), three status signals ($IO/\overline{M}$, S_1, and S_0) to identify the nature of the operation, and one special signal (ALE) to indicate the beginning of the operation. These signals are as follows:

- ALE—Address Latch Enable: This is a positive going pulse generated every time the 8085 begins an operation (machine cycle); it indicates that the bits on AD_7–AD_0 are address bits. This signal is used primarily to latch the low-order address from the multiplexed bus and generate a separate set of eight address lines, A_7–A_0.
- $\overline{RD}$—Read: This is a Read control signal (active low). This signal indicates that the selected I/O or memory device is to be read and data are available on the data bus.
- $\overline{WR}$—Write: This is a Write control signal (active low). This signal indicates that the data on the data bus are to be written into a selected memory or I/O location.
- $IO/\overline{M}$: This is a status signal used to differentiate between I/O and memory operations. When it is high, it indicates an I/O operation; when it is low, it indicates a memory operation. This signal is combined with $\overline{RD}$ (Read) and $\overline{WR}$ (Write) to generate I/O and memory control signals.
- S_1 and S_0: These status signals, similar to $IO/\overline{M}$, can identify various operations, but they are rarely used in small systems. (All the operations and their associated status signals are listed in Table 4.1 for reference.)

TABLE 4.1
8085 Machine Cycle Status and Control Signals

Machine Cycle	Status $IO/\overline{M}$	Status S_1	Status S_0	Control Signals
Opcode Fetch	0	1	1	$\overline{RD} = 0$
Memory Read	0	1	0	$\overline{RD} = 0$
Memory Write	0	0	1	$\overline{WR} = 0$
I/O Read	1	1	0	$\overline{RD} = 0$
I/O Write	1	0	1	$\overline{WR} = 0$
Interrupt Acknowledge	1	1	1	$\overline{INTA} = 0$
Halt	Z	0	0	$\overline{RD}, \overline{WR} = Z$ and $\overline{INTA} = 1$
Hold	Z	X	X	
Reset	Z	X	X	

NOTE: Z = Tri-state (high impedance)
X = Unspecified

POWER SUPPLY AND CLOCK FREQUENCY

The power supply and frequency signals are as follows:

- ☐ V_{CC}: +5 V power supply.
- ☐ V_{SS}: Ground Reference.
- ☐ X_1, X_2: A crystal (or RC, LC network) is connected at these two pins. The frequency is internally divided by two; therefore, to operate a system at 3 MHz, the crystal should have a frequency of 6 MHz.
- ☐ CLK (OUT)—Clock Output: This signal can be used as the system clock for other devices.

EXTERNALLY INITIATED SIGNALS, INCLUDING INTERRUPTS

The 8085 has five interrupt signals (see Table 4.2) that can be used to interrupt a program execution. One of the signals, INTR (Interrupt Request), is identical to the 8080A microprocessor interrupt signal (INT); the others are enhancements to the 8080A. The microprocessor acknowledges an interrupt request by the $\overline{\text{INTA}}$ (Interrupt Acknowledge) signal. (The interrupt process is discussed in Chapter 12.)

In addition to the interrupts, three pins—RESET, HOLD, and READY—accept the externally initiated signals as inputs. To respond to the HOLD request, the 8085 has one signal called HLDA (Hold Acknowledge). The functions of these signals were previously discussed in Section 3.1.3. The RESET is again described below, and others are listed in Table 4.2 for reference.

- $\overline{\text{RESET}}$ $\overline{\text{IN}}$: When the signal on this pin goes low, the program counter is set to zero, the buses are tri-stated, and the MPU is reset.
- RESET OUT: This signal indicates that the MPU is being reset. The signal can be used to reset other devices.

TABLE 4.2
8085 Interrupts and Externally Initiated Signals

☐ INTR (Input)	Interrupt Request: This is used as a general-purpose interrupt; it is similar to the INT signal of the 8080A.
☐ $\overline{\text{INTA}}$ (Output)	Interrupt Acknowledge: This is used to acknowledge an interrupt.
☐ RST 7.5 (Inputs) RST 6.5 RST 5.5	Restart Interrupts: These are vectored interrupts that transfer the program control to specific memory locations. They have higher priorities than the INTR interrupt. Among these three, the priority order is 7.5, 6.5, and 5.5.
☐ TRAP (Input)	This is a nonmaskable interrupt and has the highest priority.
☐ HOLD (Input)	This signal indicates that a peripheral such as a DMA (Direct Memory Access) controller is requesting the use of the address and data buses.
☐ HLDA (Output)	Hold Acknowledge: This signal acknowledges the HOLD request.
☐ READY (Input)	This signal is used to delay the microprocessor Read or Write cycles until a slow-responding peripheral is ready to send or accept data. When this signal goes low, the microprocessor waits for an integral number of clock cycles until it goes high.

SERIAL I/O PORTS

The 8085 has two signals to implement the serial transmission: SID (Serial Input Data) and SOD (Serial Output Data). In serial transmission, data bits are sent over a single line, one bit at a time, such as the transmission over telephone lines. This will be discussed in Chapter 16 on serial I/O.

Microprocessor communication and bus timings

To fetch 4F from 2005 H

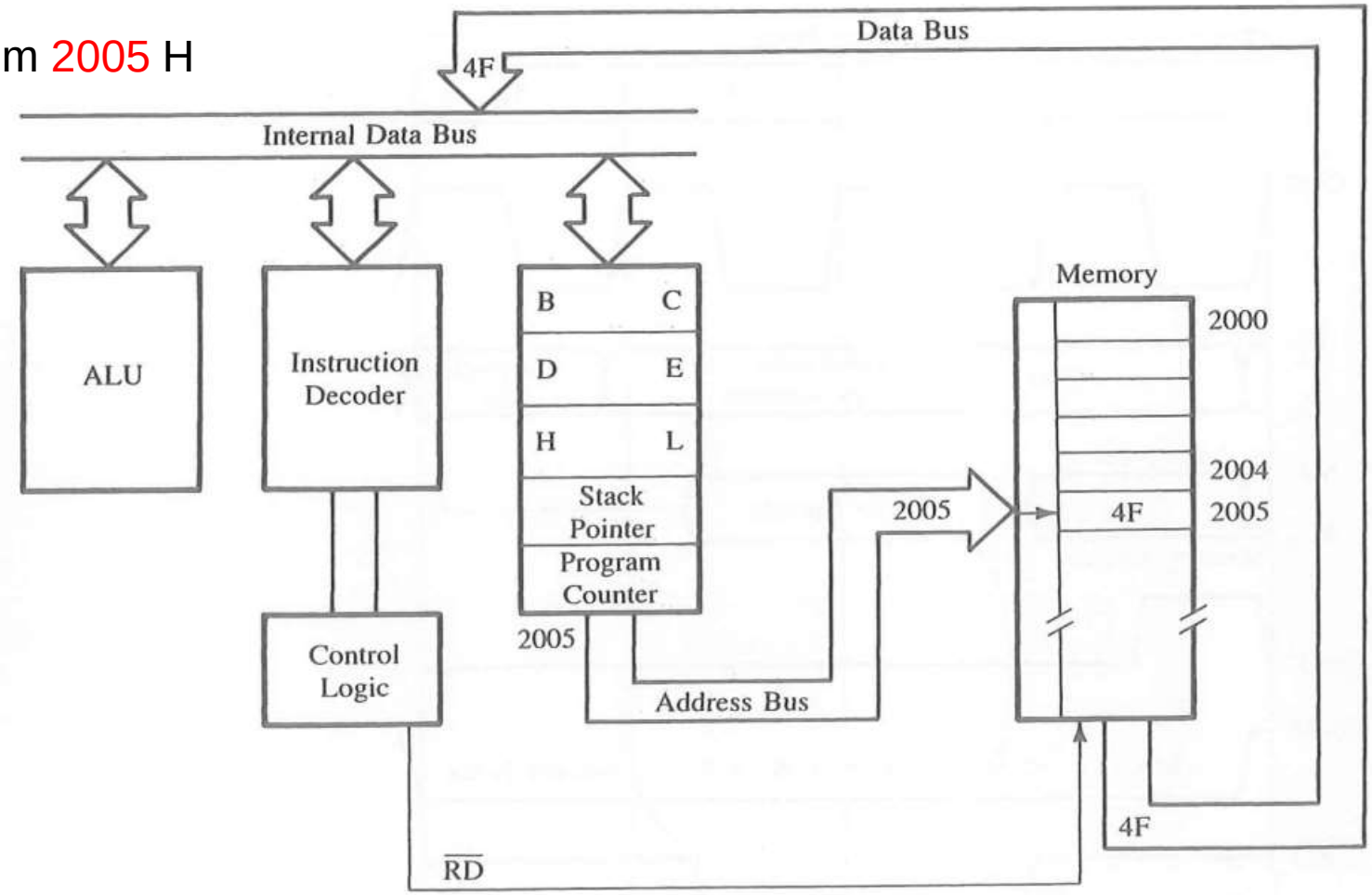


FIGURE 4.2
Data Flow from Memory to the MPU

Bus timings

To fetch 4F from 2005 H

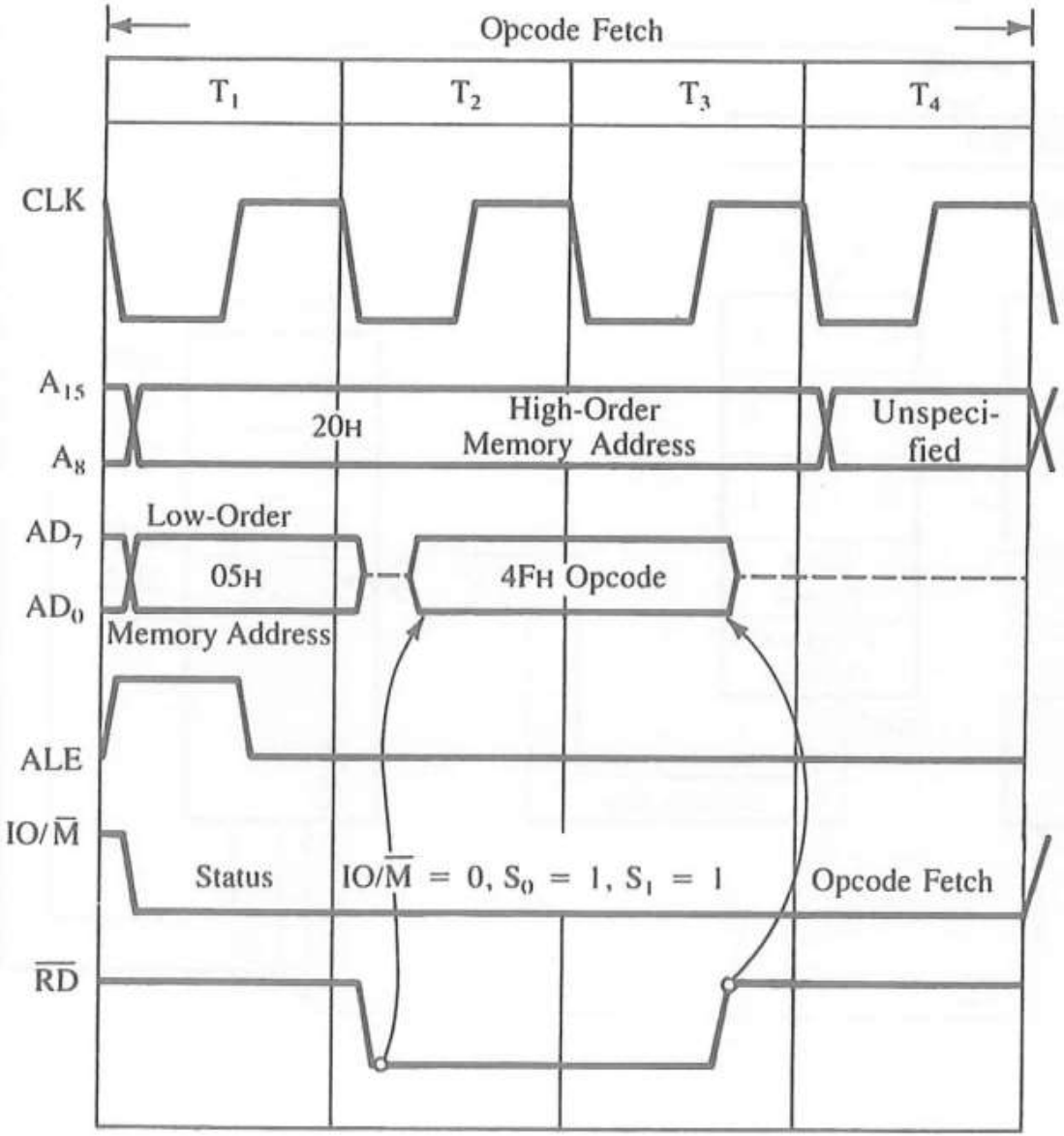
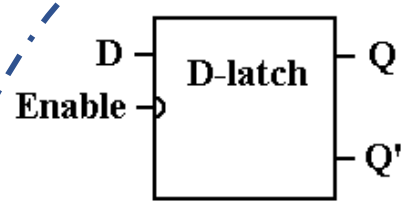


FIGURE 4.3
Timing: Transfer of Byte from Memory to MPU

Address demultiplexing

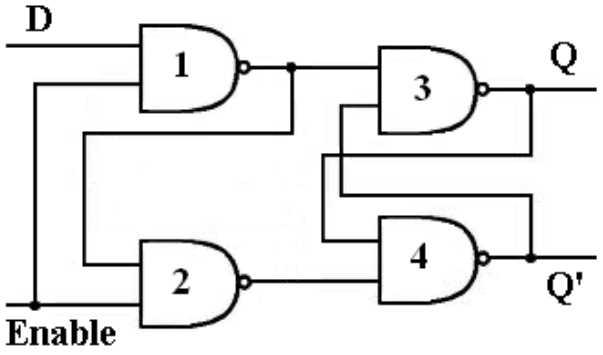
To fetch 4F from 2005 H



(a)

Enable	D	Q+	Q'+
1	1	1	0
1	0	0	1
0	1	Q	Q'
0	0	Q	Q'

(b)



(c)

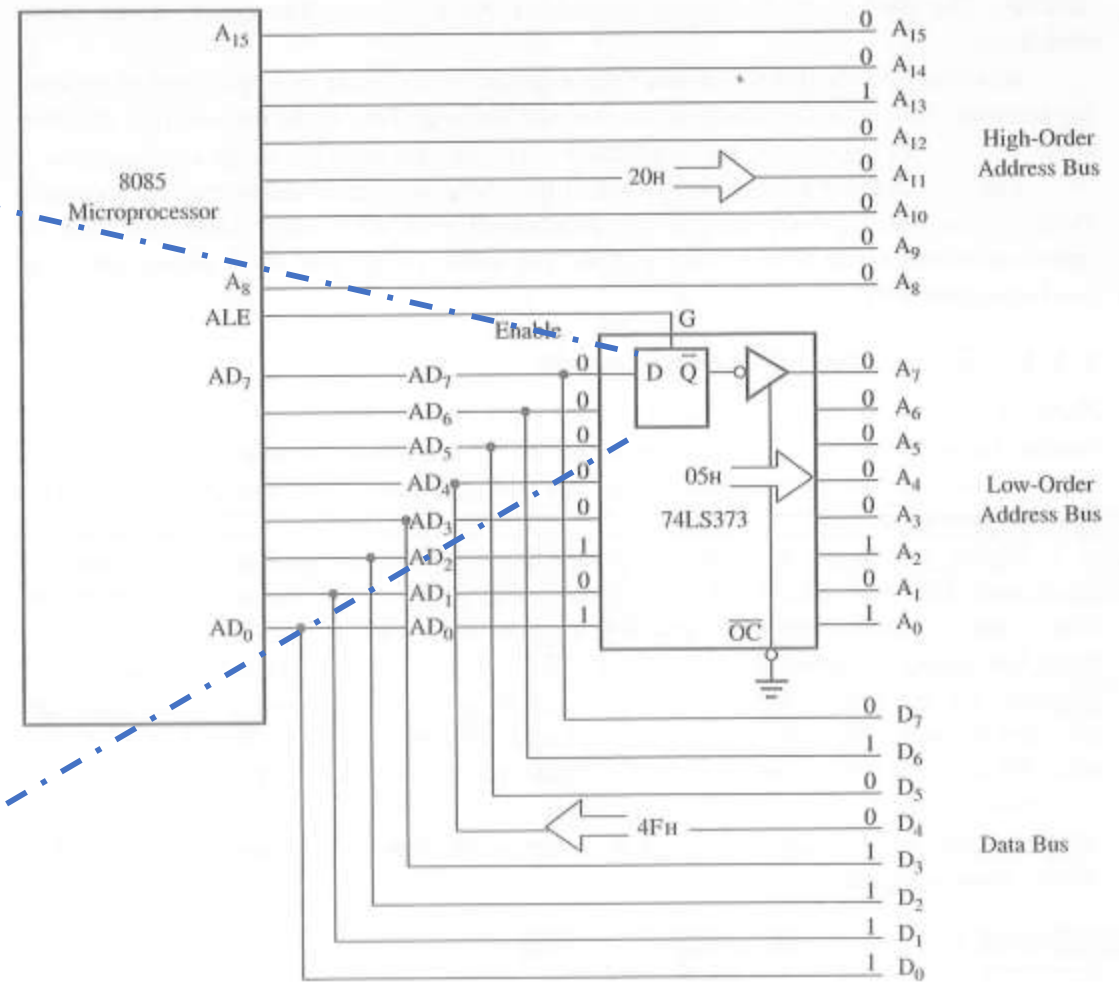
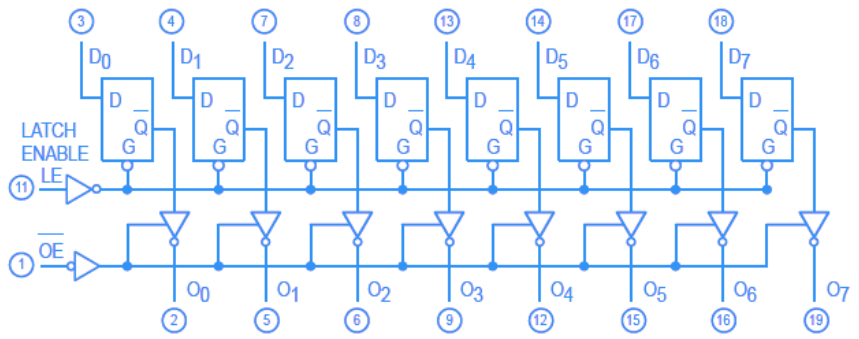


FIGURE 4.4
Schematic of Latching Low-Order Address Bus

Address demultiplexing

To fetch 4F from 2005 H



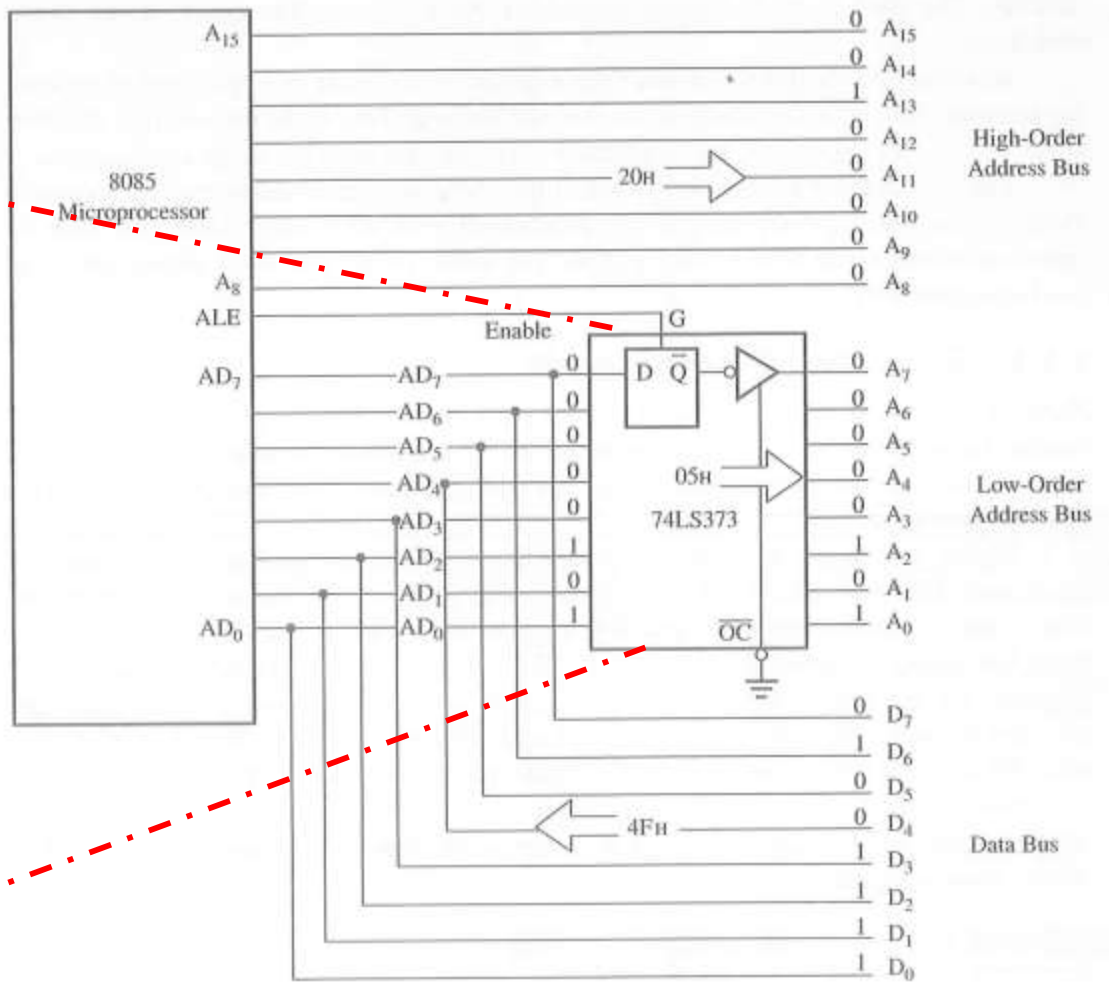
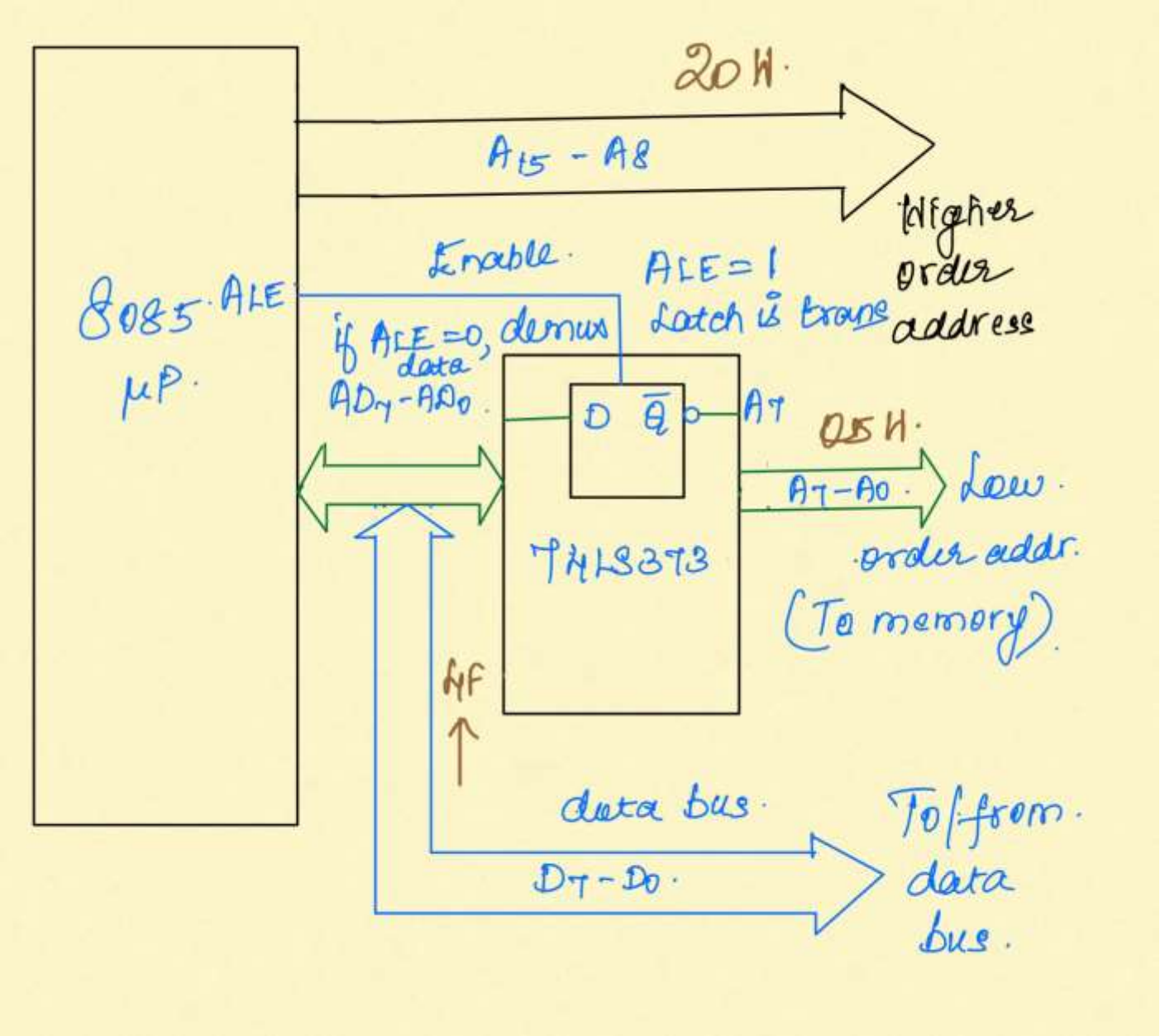


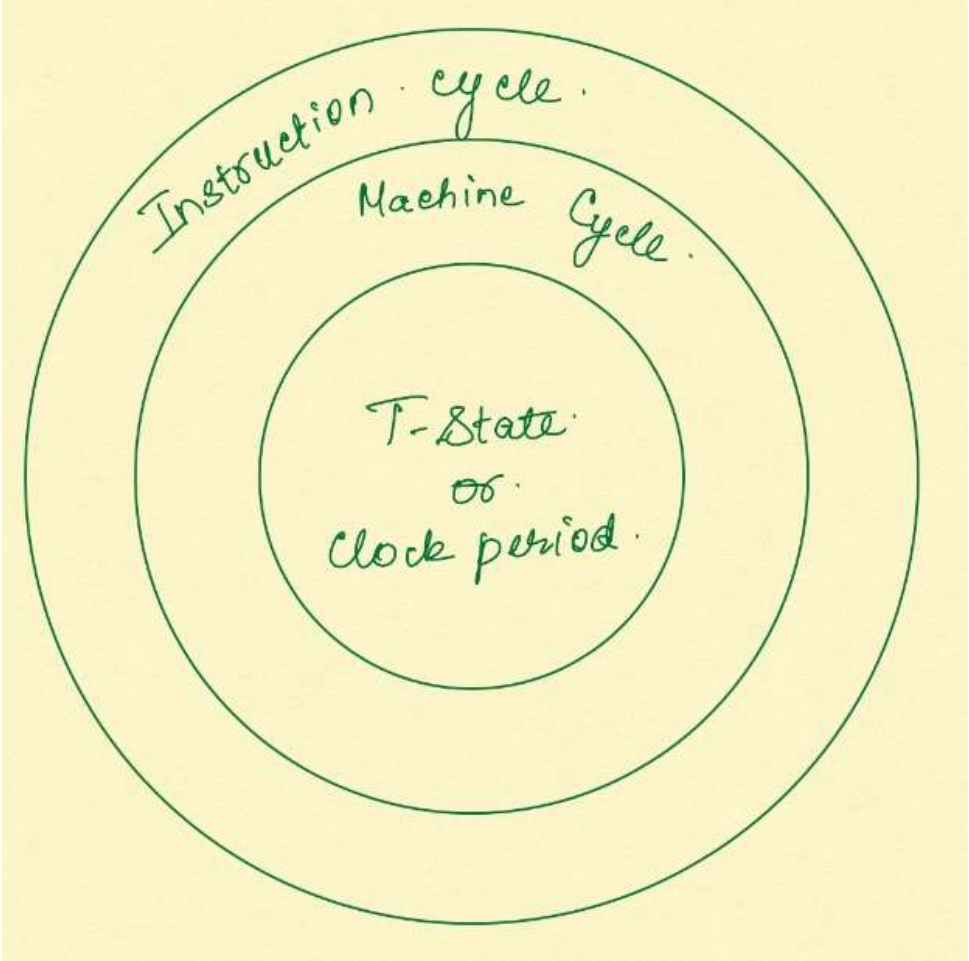
FIGURE 4.4
Schematic of Latching Low-Order Address Bus

Address demultiplexing

To fetch 4F from 2005 H



3 terms



T-state: A division of operation @ 1 clock cycle

Machine cycle: Time/Total t-states taken for executing one operation, say op.code fetching (4F)

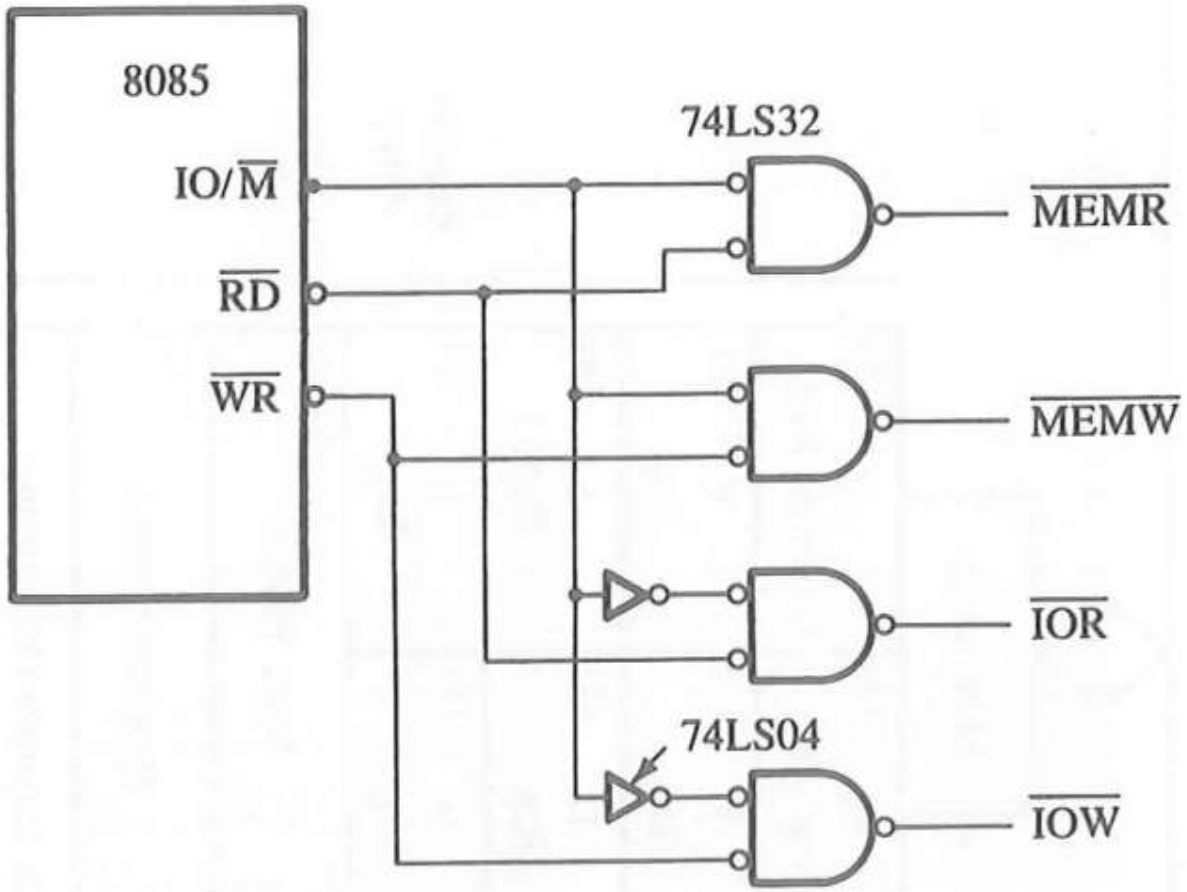
From memory, it take one or more T-states.

Instruction cycle: It is the time taken for completing one instruction, it may take many machine cycles.

Eg: “4F” MOV C, A.

Generation of control signals

FIGURE 4.5
Schematic to Generate Read/Write Control Signals for Memory and I/O



Generation of control signals

(Overall perspective)

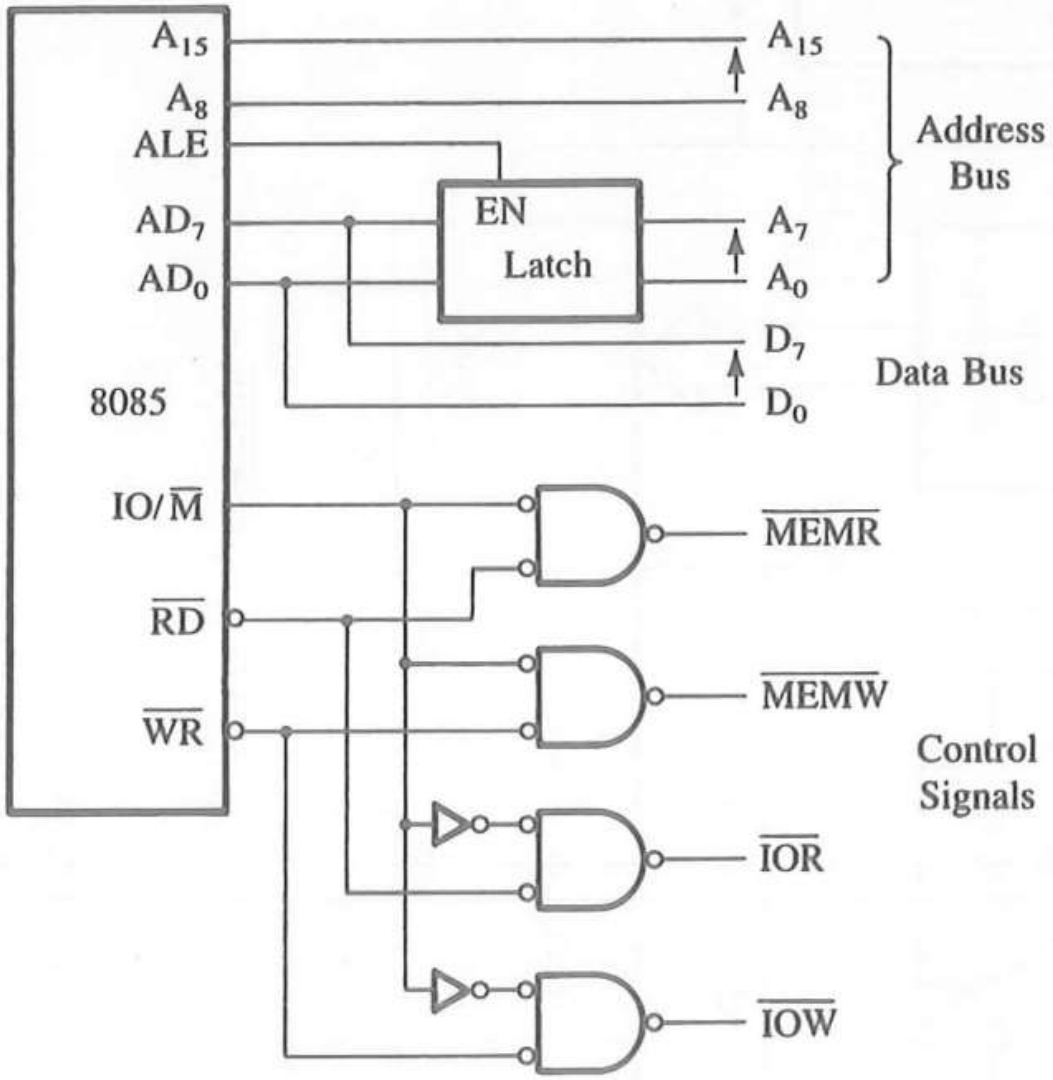


FIGURE 4.6
8085 Demultiplexed Address and Data Bus with Control Signals

Detailed look of 8085 MP

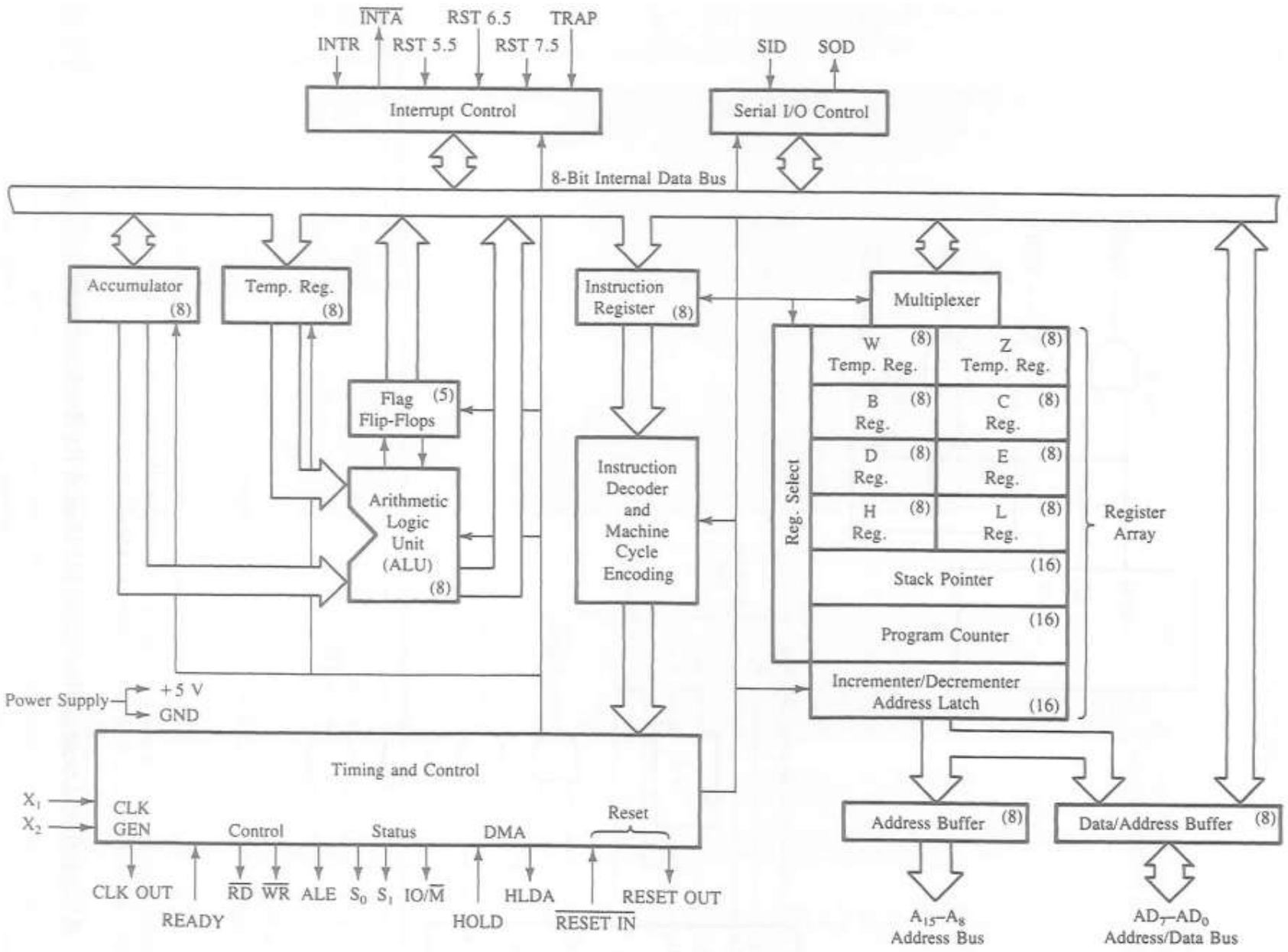


FIGURE 4.7
The 8085A Microprocessor: Functional Block Diagram
NOTE: The 8085A microprocessor is commonly known as the 8085.
SOURCE: Intel Corporation, *Embedded Microprocessors* (Santa Clara, Calif.: Author, 1994), pp. 1-11.

Detailed look of 8085 MP

1. ALU
2. Timing and control unit
3. Instruction register and decoder
4. Register array
5. Interrupt control
6. Serial I/O control

The ALU

1. It performs the computing functions
2. Includes, Accumulator, temporary register, arithmetic and logic circuits, and five flags
3. Results are usually stored in the accumulator, and the flags (flip-flops) are used to set/reset acc. to the result of operation
4. These flags generally represents data condition in accumulator with some exceptions

Flag register

1. ***S-Sign flag***: After arithmetic/logical operation if D_7 of Accumulator (usually) is 1, then, the S flag is set. Used with signed integers,
   - In a given byte if $D_7 = 0$ → positive number, if 1 → negative number
   - In arithmetic operations, in 8-bit integer representation, D_7 is reserved for sign, remaining 7 bits are used to compute the magnitude
   - This flag is irrelevant for unsigned integers. So, in unsigned integers, even if D_7 is 1 and S=1 it doesn't mean it is a negative number

Flag register

2. ***Z- Zero flag***: this flag is set when ALU operation results in Zero. Else, it is reset
   - This flag is modified by results in accumulator as well as in other registers
3. ***AC-Auxilary Carry flag:*** This flag is set when carry generated by digit D_3 is passed on to digit D_4. It is used for internal operations and is not available to programmer, to make a conditional jump
4. ***P-Parity flag:*** This flag is set when there are even number of 1s in the result of an arithmetic/logical operation, even though the magnitude of the number may be odd. If the result has odd number of 1s then it is reset

Flag register

5. ***CY- Carry flag:*** This flag is set when there is carry occurring in D_7 as a result of addition operation, else it is reset, it also serves as a borrow flag for subtraction

The bit positions reserved for these flags in the flag register are as follows:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
S	Z		AC		P		CY

Among the five flags, the AC flag is used internally for BCD arithmetic; the instruction set does not include any conditional jump instructions based on the AC flag. Of the remaining four flags, the Z and CY flags are those most commonly used.

TIMING AND CONTROL UNIT

This unit synchronizes all the microprocessor operations with the clock and generates the control signals necessary for communication between the microprocessor and peripherals. The control signals are similar to a sync pulse in an oscilloscope. The $\overline{RD}$ and $\overline{WR}$ signals are sync pulses indicating the availability of data on the data bus.

INSTRUCTION REGISTER AND DECODER

The instruction register and the decoder are part of the ALU. When an instruction is fetched from memory, it is loaded in the instruction register. The decoder decodes the instruction and establishes the sequence of events to follow. The instruction register is not programmable and cannot be accessed through any instruction.

REGISTER ARRAY

The programmable registers were discussed in the last chapter. Two additional registers, called temporary registers W and Z, are included in the register array. These registers are used to hold 8-bit data during the execution of some instructions. However, because they are used internally, they are not available to the programmer.

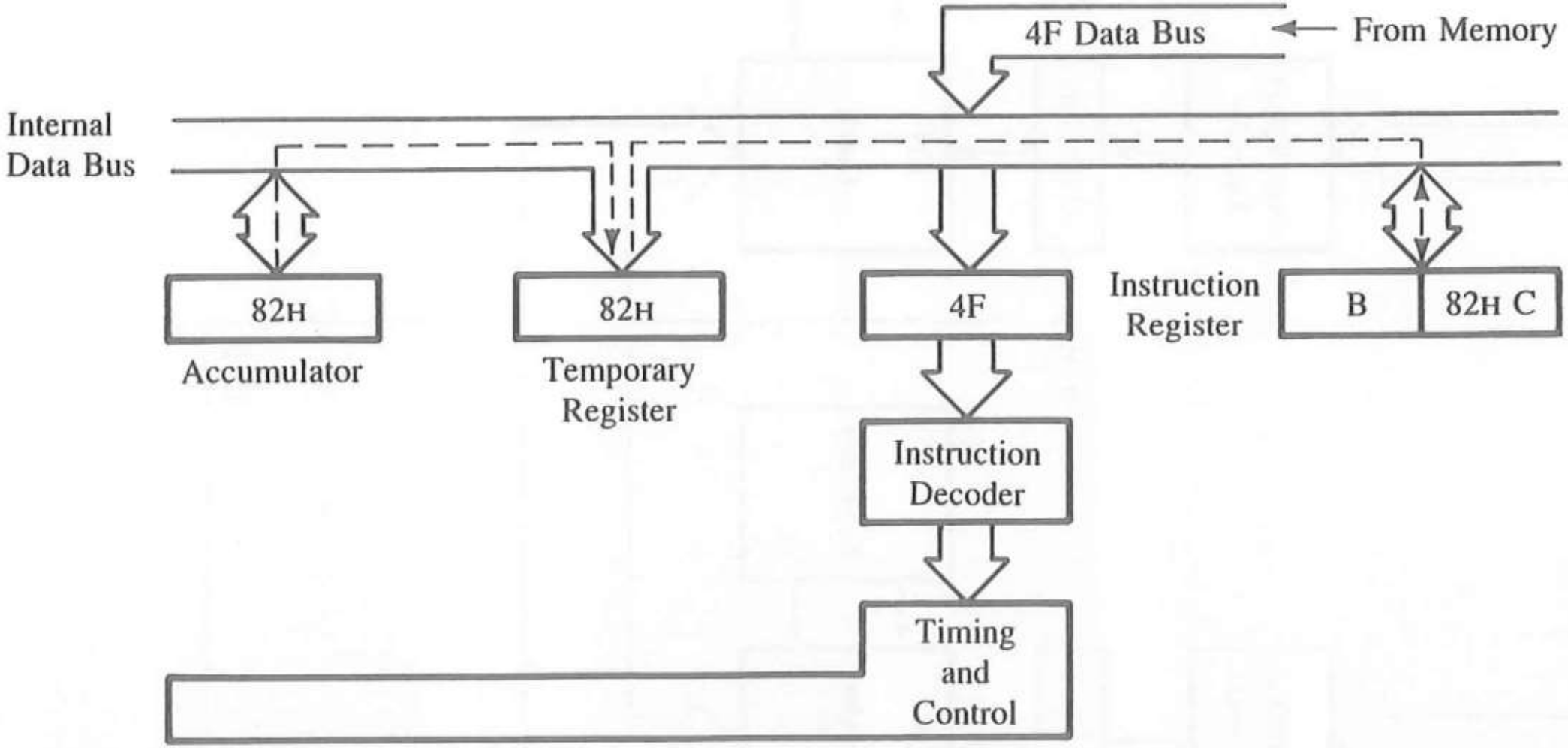


FIGURE 4.8
Instruction Decoding and Execution

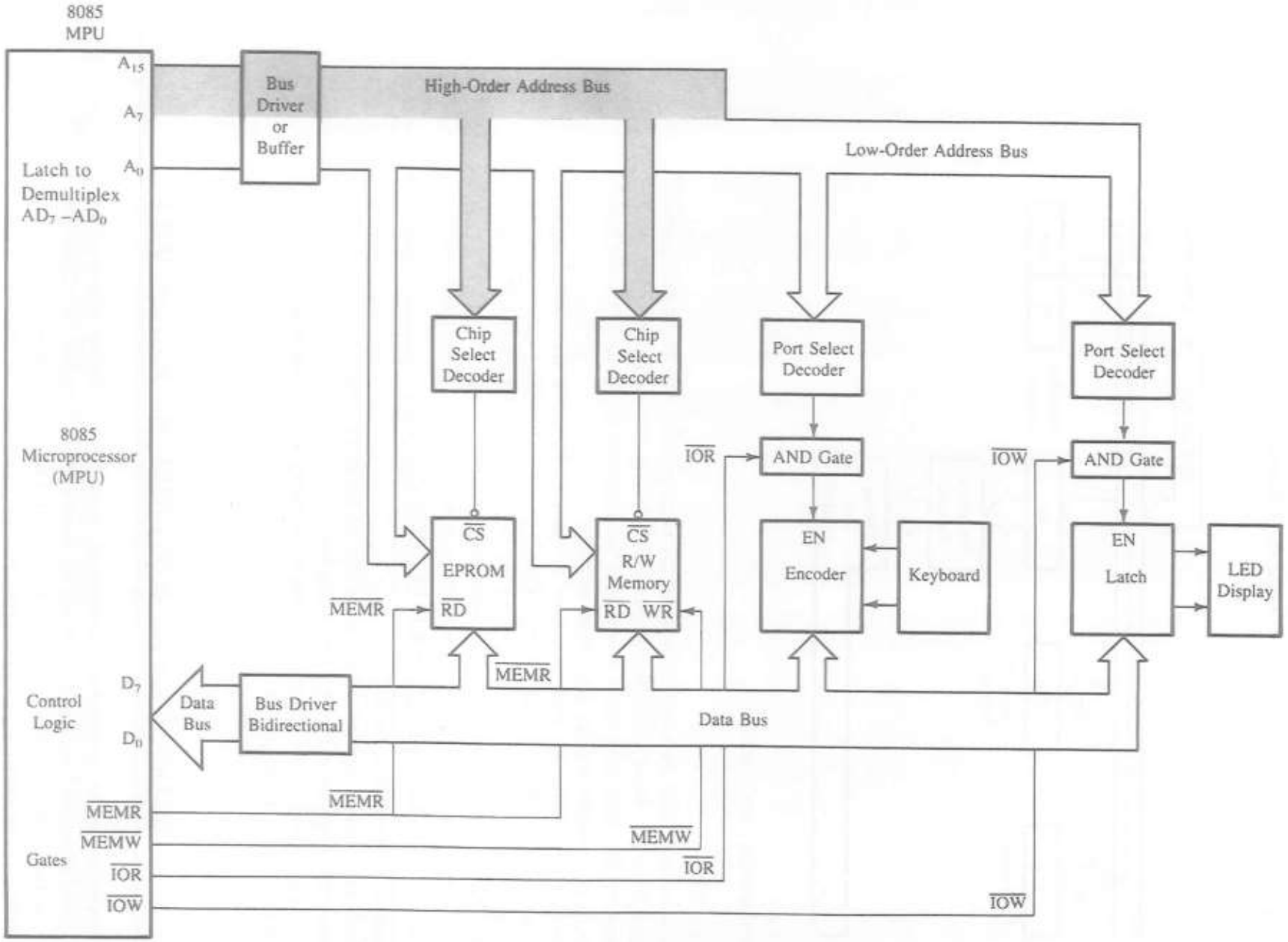


FIGURE 4.9
8085 Single-Board Microcomputer System

NOTE: The bus AD_7–AD_0 is demultiplexed using a latch and memory, and I/O control signals are generated using gates. Figure 4.6 is included in the MPU.

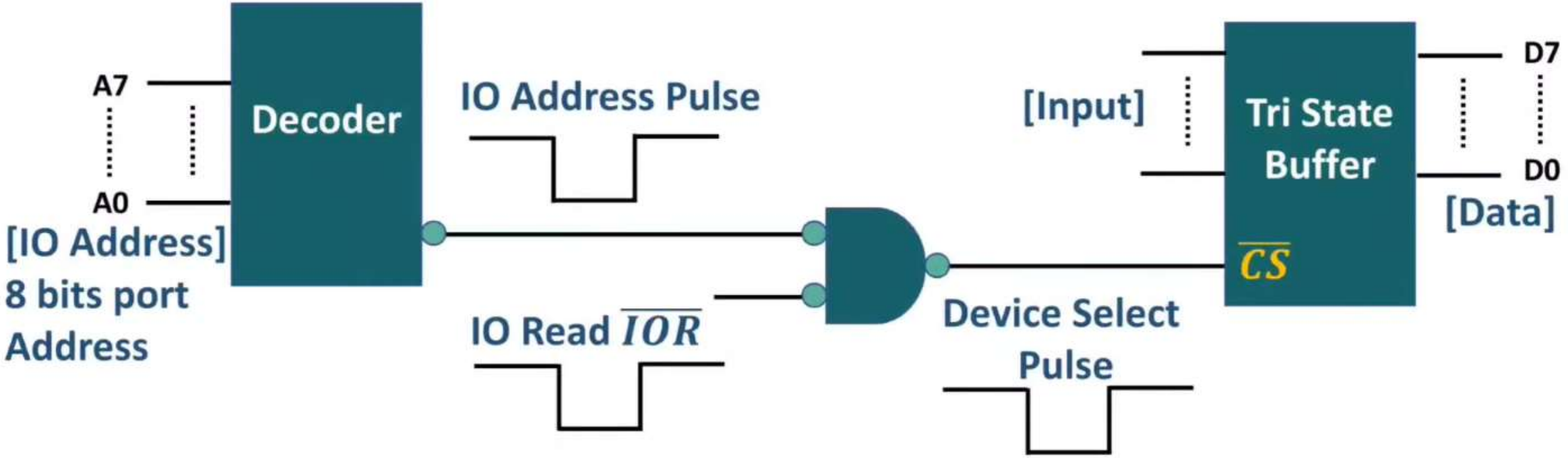
Peripheral Mapped IO in 8085

- Here IO Devices are treated as IO [Input Output Devices].
- It needs 8 bits addressing by A0 to A7 address lines of microprocessor 8085.
- So, it can address = 2^8 = 256 IO devices or Addresses.
- Compared to memory mapped IO, it needs less hardware for decoding of IO. The reason is memory mapped IO have 16 address lines for address and selection of IO.
- With IO mapped IO, read and write of IO is done by $\overline{IOR}$ and $\overline{IOW}$ control signals.
- In IO mapped IO, Data transfer is only possible with Accumulator and IO devices.
- With Microprocessor 8085, we have only IN and OUT instructions for IO mapped IO.

Blueprint of IO Interface with 8085

❖ INPUT Interface

- ❑ For INPUT of data we use IN Instruction with 8085.
- ❑ Example : IN [8 bits port]
- ❑ Accumulator will get content of Data lines D0 to D7 from Input Devices.



Absolute and Partial Decoding in 8085

- With Absolute decoding, we use all the address lines [A0-A7] for decoding of IO devices.
- With Partial decoding, few address lines are not used for decoding IO devices.
- With Absolute decoding, we have fixed address for decoding IO.
- With Partial decoding, multiple address may decode IO.
- Absolute and partial decoding may happen in case of Memory interface.

- Example of Absolute decoding.
- Port address is there with IN and OUT instruction as per all A0 to A7 lines.
- 5CH = 0101 1100 b
- Here, A7, A5, A1, A0 = 0 and A6, A4, A3, A2 = 1.
- So, device has fixed address.

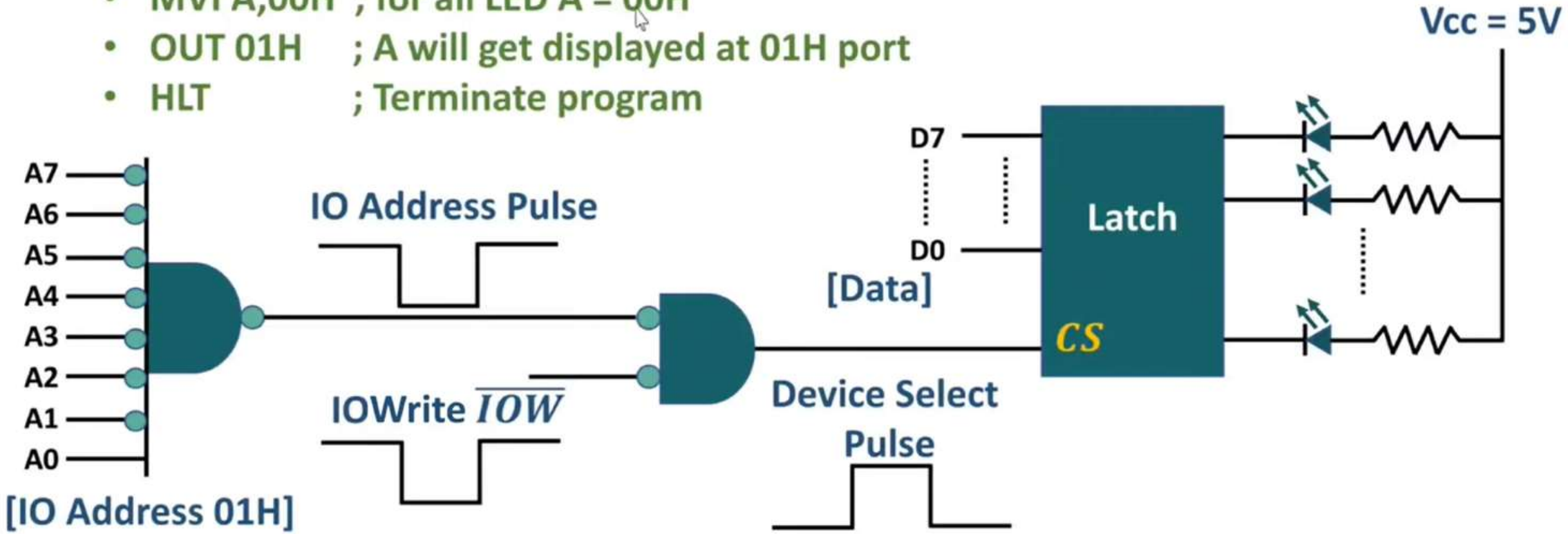
- Example of Partial decoding.
- Port address is there with IN and OUT instruction as A0 to A7 lines, but few address lines are not used.
- 0101 11XX b = 5CH, 5DH, 5EH and 5FH.
- Here, A1 and A0 are unused, so total four address may decode given IO.

LED Interfacing in 8085

❖ Interface 8 Common Anode CA LEDs to 8085 connected at port address 01H and write a program to turn ON all LEDs.

❑ Program

- MVI A,00H ; for all LED A = 00H
- OUT 01H ; A will get displayed at 01H port
- HLT ; Terminate program



Timing Diagram of OUT instruction in 8085

❖ OUT (8 bit Port Address = 05H) [Opcode =D3H]

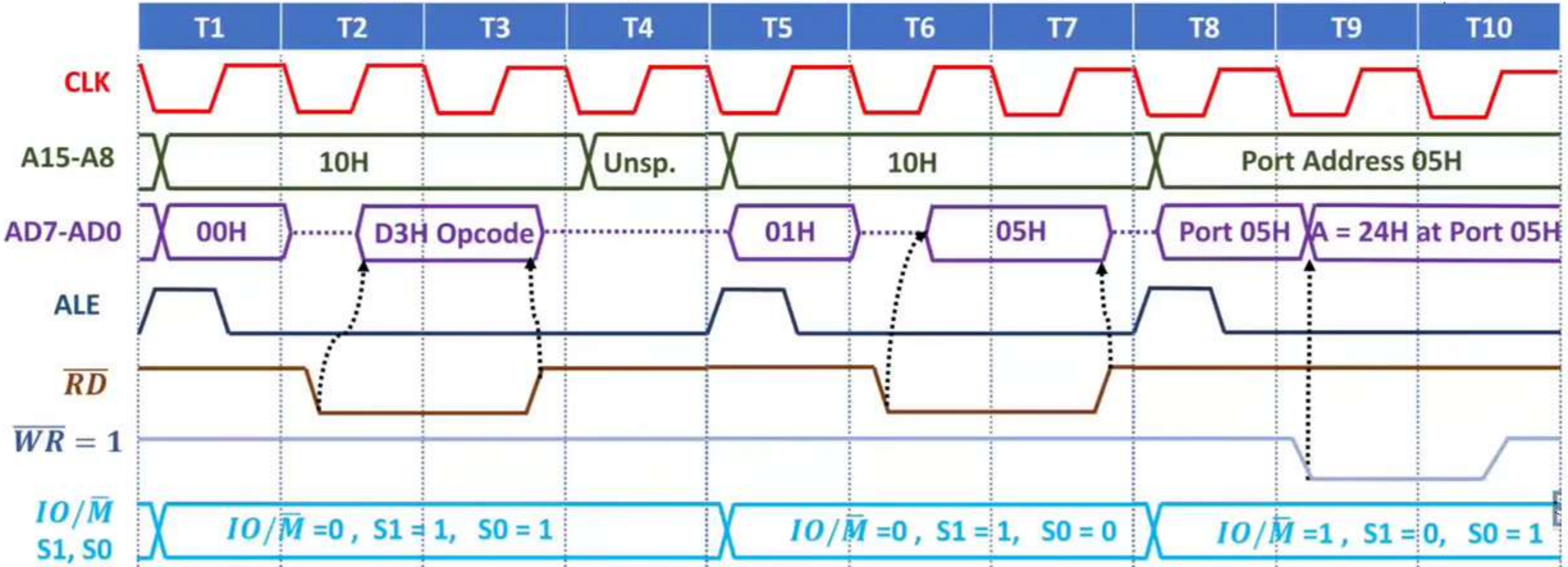
PC = 1000H
8085
A= 24H

Add	Data
1000H	D3H
1001H	05H

❖ OUT 05H executions needs opcode fetch, Memory Read & IO Write machine cycle.

❖ So, After instruction, A = 24H will get display at 05H port Address.

❖ OUT 05H [Opcode Fetch = D3H] ❖ Memory Read 05H from 1001H ❖ IO Write A at 05H Output Port



Timing Diagram of IN instruction in 8085

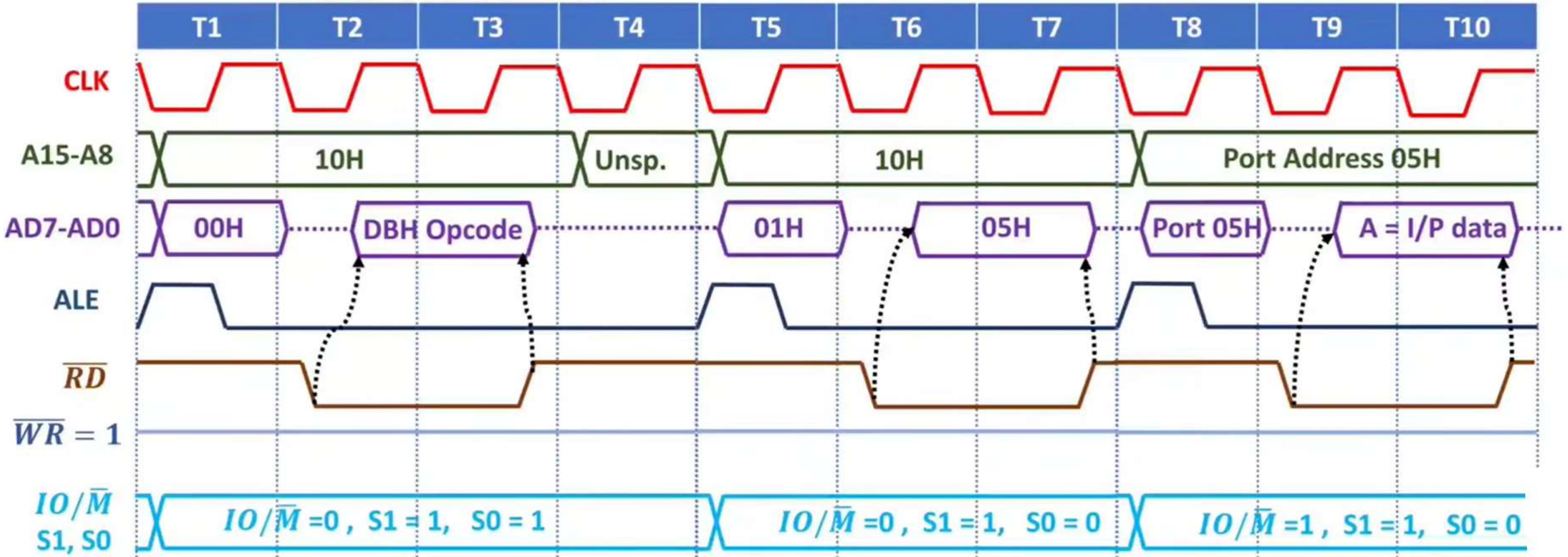
- IN (8 bit Port Address = 05H) [Opcode =DBH]

PC = 1000H
8085
A= 00H

Add	Data
1000H	DBH
1001H	05H

- IN 05H executions needs opcode fetch, Memory Read & IO Read machine cycle.
- So, After instruction, Accumulator will get copied from 05H port Address.

- IN 05H [Opcode Fetch = DBH]
- Memory Read 05H from 1001H
- IO Read into A from port add 05H

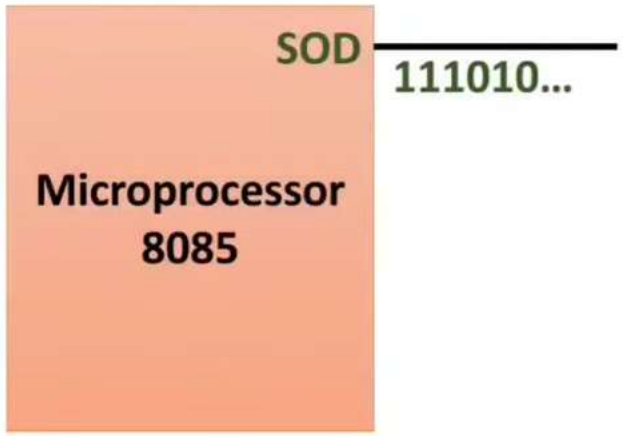


Memory Mapped IO and IO Mapped IO in 8085

Memory Mapped IO	IO Mapped IO
❖ Here IO devices treated as Memory	❖ Here IO devices treated as IO.
❖ 16 bits addressing ($A_0 - A_{15}$)	❖ 8 bits addressing ($A_0 - A_7$)
❖ It can address = 2^{16} = 64K Address	❖ It can address = 2^8 = 256 Address
❖ Number of devices = 64K = 65536	❖ Number of devices = 256
❖ Due to many devices we need more decoder hardware in Memory Mapped IO.	❖ Due to less devices we need less decoder hardware in IO Mapped IO.
❖ Due to limited Address lines (16) with 8085 available memory less with memory mapped IO.	❖ Available memory more with IO mapped IO as number of devices are less in IO mapped IO.
❖ Here we use $\overline{MEMR}$ and $\overline{MEMW}$ control signals.	❖ Here we use $\overline{IOR}$ and $\overline{IOW}$ control signals.
❖ Data transfer happens between any registers and IO.	❖ Data transfer only between Accumulator and IO.
❖ ALU operation is done with all registers.	❖ Not Available.
❖ Instructions Example: LDA XXXXH, STA XXXXH, MOV A,M	❖ Instructions Example: IN XXXXH, OUT XXXXH

Serial Communication by SIM Instruction

✓ Write a program to transmit data stored at 1000H memory location serially at a baud rate of 1000 as per UART standard.



Add	Data
0FFEH	
0FFFH	
1000H	Data
1001H	
1002H	

❖ **SIM Command**

SOD	SDE	-	R 7.5	MSE	M7.5	M6.5	M5.5

- ❑ To send 0, SIM command should be 0100 0000 = 40H
- ❑ To send 1, SIM command should be 1100 0000 = C0H
- ❑ Here SIM command is used by writing data on Accumulator and then write SIM instruction.

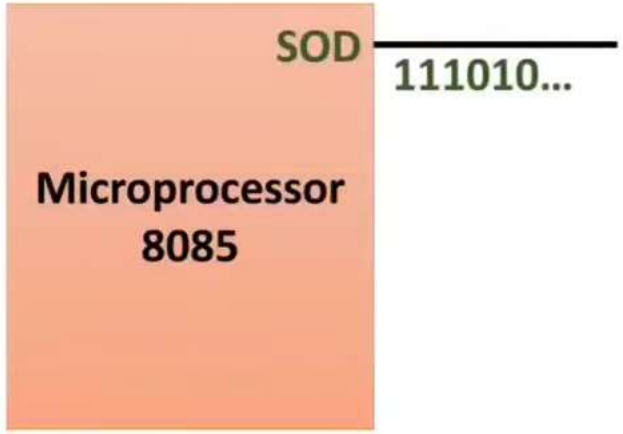
❖ **UART Standard**

0	8 bits data	1

```
LDA 1000H    ;taking data from 1000H
MOV B,A      ;making copy of data of A as we use A in program
MVI C,08H    ;Counter of sending data

MVI A,40H    ;Sending data 0 as per UART format
SIM
CALL Delay   ;1msec delay as per 1000 baud rate

MVI A,C0H    ;Sending data 1 as per UART format
SIM
CALL Delay   ;1msec delay as per 1000 baud rate
HLT
```

Serial Communication by SIM Instruction

✓ Write a program to transmit data stored at 1000H memory location serially at a baud rate of 1000 as per UART standard.



Add	Data
0FFEH	
0FFFH	
1000H	Data
1001H	
1002H	

❖ SIM Command

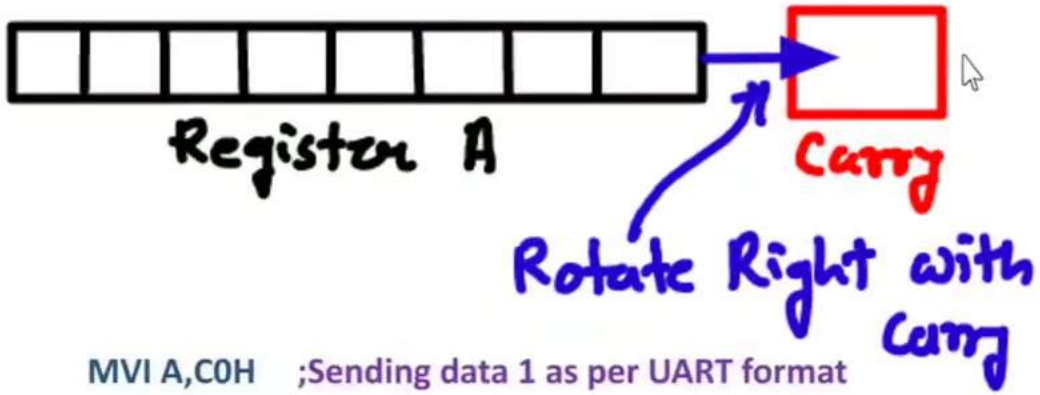
SOD	SDE	-	R 7.5	MSE	M7.5	M6.5	M5.5

- ❑ To send 0, SIM command should be 0100 0000 = 40H
- ❑ To send 1, SIM command should be 1100 0000 = C0H
- ❑ Here SIM command is used by writing data on Accumulator and then write SIM instruction.

❖ UART Standard

0	8 bits data	1

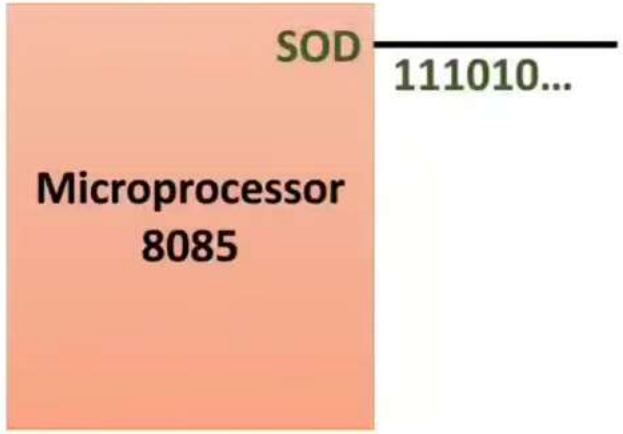
```
        LDA 1000H   ;taking data from 1000H
        MOV B,A     ;making copy of data of A as we use A in program
        MVI C,08H   ;Counter of sending data

        MVI A,40H   ;Sending data 0 as per UART format
        SIM
        CALL Delay  ;1msec delay as per 1000 baud rate

Loop:   MOV A,B     ;taking data from B
        RRC         ;Rotate A right with carry
```




```
        MVI A,C0H   ;Sending data 1 as per UART format
        SIM
        CALL Delay  ;1msec delay as per 1000 baud rate
        HLT
```

Serial Communication by SIM Instruction

✓ Write a program to transmit data stored at 1000H memory location serially at a baud rate of 1000 as per UART standard.



Add	Data
0FFEH	
0FFFH	
1000H	Data
1001H	
1002H	

❖ **SIM Command**

SOD	SDE	-	R 7.5	MSE	M7.5	M6.5	M5.5

- To send 0, SIM command should be 0100 0000 = 40H
- To send 1, SIM command should be 1100 0000 = C0H
- Here SIM command is used by writing data on Accumulator and then write SIM instruction.

❖ **UART Standard**

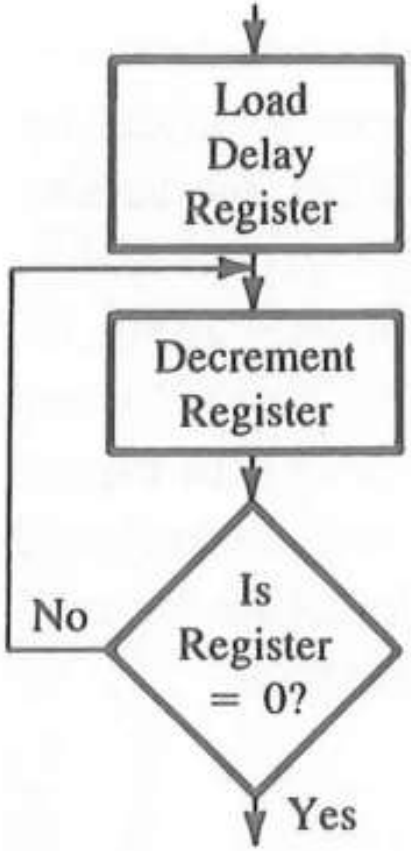
0	8 bits data	1

```
        LDA 1000H   ;taking data from 1000H
        MOV B,A     ;making copy of data of A as we use A in program
        MVI C,08H   ;Counter of sending data

        MVI A,40H   ;Sending data 0 as per UART format
        SIM
        CALL Delay  ;1msec delay as per 1000 baud rate

Loop:   MOV A,B     ;taking data from B
        RRC         ;Rotate A right with carry
        MOV B,A     ;making copy of data of A as we use A in program
        MVI A,C0H   ;Sending data 1 as per UART format
        JC ONE
        MVI A,40H   ;Sending data 0 as per UART format
ONE:    SIM
        CALL Delay  ;1msec delay as per 1000 baud rate
        DCR C       ;Counter in register C for 8 bit data
        JNZ loop

        MVI A,C0H   ;Sending data 1 as per UART format
        SIM
        CALL Delay  ;1msec delay as per 1000 baud rate
        HLT
```

```
        LDA 1000H    ;taking data from 1000H
        MOV B,A      ;making copy of data of A as we use A in program
        MVI C,08H    ;Counter of sending data

        MVI A,40H    ;Sending data 0 as per UART format
        SIM
        CALL Delay   ;1msec delay as per 1000 baud rate

Loop:   MOV A,B      ;taking data from B
        RRC          ;Rotate A right with carry
        MOV B,A      ;making copy of data of A as we use A in program
        MVI A,C0H    ;Sending data 1 as per UART format
        JC ONE
        MVI A,40H    ;Sending data 0 as per UART format
ONE:    SIM
        CALL Delay   ;1msec delay as per 1000 baud rate
        DCR C        ;Counter in register C for 8 bit data
        JNZ loop

        MVI A,C0H    ;Sending data 1 as per UART format
        SIM
        CALL Delay   ;1msec delay as per 1000 baud rate
        HLT

Delay:  MVI D,Count ;7T State
Again:  DCR D       ;4T State
        JNZ Again   ;10T/7T State
        RET         ;10T State
```

Td = 1 mSec = (17 + 14*Count) T – 3T
If T = 0.3333µSec for F = 6MHz
Then, Count = D5H = 213

Delay and counter design



Label	Opcode	Operand	Comments	T-states
	MVI	C,FFH	;Load register C	7
LOOP:	DCR	C	;Decrement C	4
	JNZ	LOOP	;Jump back to ; decrement C	10/7

Delay and counter design

The last column in Figure 8.2 shows the T-states (clock periods) required by the 8085 microprocessor to execute each instruction. (See Appendix F for the list of the 8085 instructions and their T-states.) The instruction MVI requires seven clock periods. An 8085-based microcomputer with 2 MHz clock frequency will execute the instruction MVI in 3.5 μs as follows:

$$\text{Clock frequency of the system } f = 2 \text{ MHz}$$
$$\text{Clock period } T = 1/f = 1/2 \times 10^{-6} = 0.5\ \mu\text{s}$$
$$\text{Time to execute MVI} = 7 \text{ T-states} \times 0.5$$
$$= 3.5\ \mu\text{s}$$

However, if the clock frequency of the system is 1 MHz, the microprocessor will require 7 μs to execute the same instruction. To calculate the time delay in a loop, we must account for the T-states required for each instruction and for the number of times the instructions are executed in the loop.

Delay and counter design

In Figure 8.2, register C is loaded with the count FFH (255_{10}) by the instruction MVI, which is executed once and takes seven T-states. The next two instructions, DCR and JNZ, form a loop with a total of 14 (4 + 10) T-states. The loop is repeated 255 times until register C = 0.

The time delay in the loop T_L with 2 MHz clock frequency is calculated as

$$TL = (T \times \text{Loop T-states} \times N10)$$

where T_L = Time delay in the loop

T = System clock period

N_{10} = Equivalent decimal number of the hexadecimal count loaded in the delay register

$$T_L = (0.5 \times 10^{-6} \times 14 \times 255)$$
$$= 1785\ \mu s$$
$$\approx 1.8\ ms$$

Delay and counter design

In most applications, this approximate calculation of the time delay is considered reasonably accurate. However, to calculate the time delay more accurately, we need to adjust for the execution of the JNZ instruction and add the execution time of the initial instruction.

The T-states for JNZ instruction are shown as 10/7. This can be interpreted as follows: The 8085 microprocessor requires ten T-states to execute a conditional Jump instruction when it jumps or changes the sequence of the program and seven T-states when the program falls through the loop (goes to the instruction following the JNZ). In Figure 8.2, the loop is executed 255 times; in the last cycle, the JNZ instruction will be executed in seven T-states. This difference can be accounted for in the delay calculation by subtracting the execution time of three states. Therefore, the adjusted loop delay is

$$\begin{aligned} T_{LA} &= T_L - (3 \text{ T-states} \times \text{Clock period}) \\ &= 1785.0\ \mu s - 1.5\ \mu s = 1783.5\ \mu s \end{aligned}$$

Now the total delay must take into account the execution time of the instructions outside the loop. In the above example, we have only one instruction (MVI C) outside the loop. Therefore, the total delay is

$$\text{Total Delay} = \begin{matrix}\text{Time to execute instructions} \\ \text{outside loop}\end{matrix} + \begin{matrix}\text{Time to execute} \\ \text{loop instructions}\end{matrix}$$

$$\begin{aligned} T_D &= T_O + T_{LA} \\ &= (7 \times 0.5\ \mu s) + 1783.5\ \mu s = 1787\ \mu s \\ &\approx 1.8\ \text{ms} \end{aligned}$$

Delay and counter design

8.1.2 Time Delay Using a Register Pair

The time delay can be considerably increased by setting a loop and using a register pair with a 16-bit number (maximum FFFFH). The 16-bit number is decremented by using the instruction DCX. However, the instruction DCX does not set the Zero flag and, without the test flags, Jump instructions cannot check desired data conditions. Additional techniques, therefore, must be used to set the Zero flag.

The following set of instructions uses a register pair to set up a time delay.

Label	Opcode	Operand	Comments	T-states
	LXI	B,2384H	;Load BC with 16-bit count	10
LOOP:	DCX	B	;Decrement (BC) by one	6
	MOV	A,C	;Place contents of C in A	4
	ORA	B	;OR (B) with (C) to set Zero flag	4
	JNZ	LOOP	;If result ≠ 0, jump back to LOOP	10/7

Delay and counter design

PROBLEM STATEMENT

Write a program to generate a continuous square wave with the period of 500 μs. Assume the system clock period is 325 ns, and use bit D_0 to output the square wave.

Delay and counter design

PROBLEM ANALYSIS

In this problem, the period of the square wave is 500 μs; therefore, the pulse should be on (logic 1) for 250 μs and off (logic 0) for the remaining 250 μs. The alternate pattern of 0/1 bits can be provided by loading the accumulator with the number AAH (1010 1010) and rotating the pattern once through each delay loop. Bit D_0 of the output port is used to provide logic 0 and 1; therefore, all other bits can be masked by ANDing the accumulator with the byte 01H. The delay of 250 μs can be very easily obtained with an 8-bit delay count and one register.

Illustrative program for delay using counter

PROGRAM

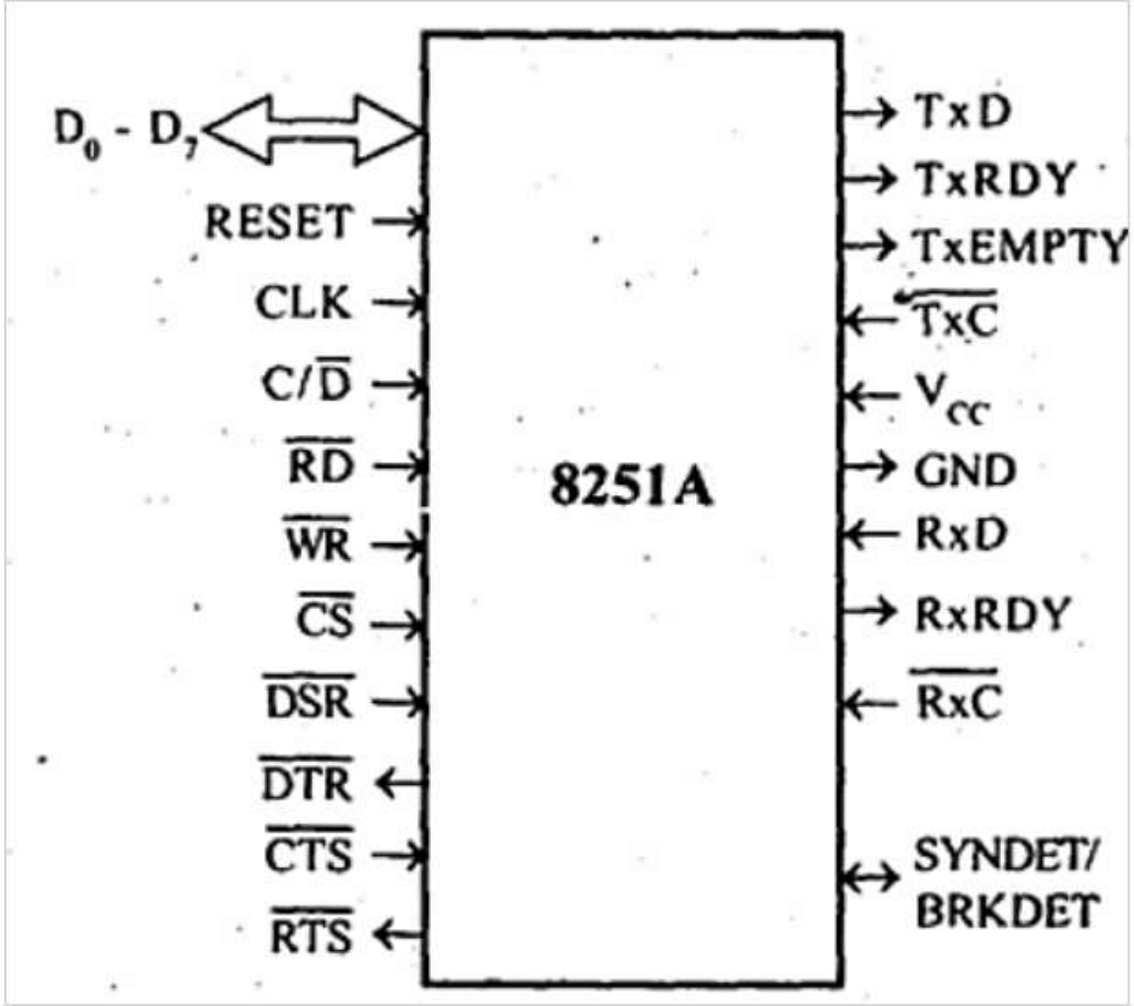
Memory Address HI-LO	HEX Code	Label	Mnemonics	Comments
XX00	16			
01	AA		MVI D,AA	;Load bit pattern AAH
02	7A	ROTATE:	MOV A,D	;Load bit pattern in A
03	07		RLC	;Change data from AAH to
				; 55H and vice versa
04	57		MOV D,A	;Save (A)
05	E6		ANI 01H	;Mask bits D_7–D_1
06	01			
07	D3		OUT PORT1	;Turn on or off the lights
08	PORT1			
09	06		MVI B,COUNT (7T)	;Load delay count for 250 μs
0A	COUNT			
0B	05	DELAY:	DCR B (4T)	;Next count
0C	C2		JNZ DELAY (10/7T)	;Repeat until (B) = 0
0D	0B			
0E	XX			
0F	C3		JMP ROTATE (10T)	;Go back to change logic level
10	02			
11	XX			

SERIAL COMMUNICATION USING USART (8251A)

USART AS A 28-PIN DIP PACKAGE

USART: Universal Synchronous Asynchronous Receiver Transmitter

The pin description of 8251A chip

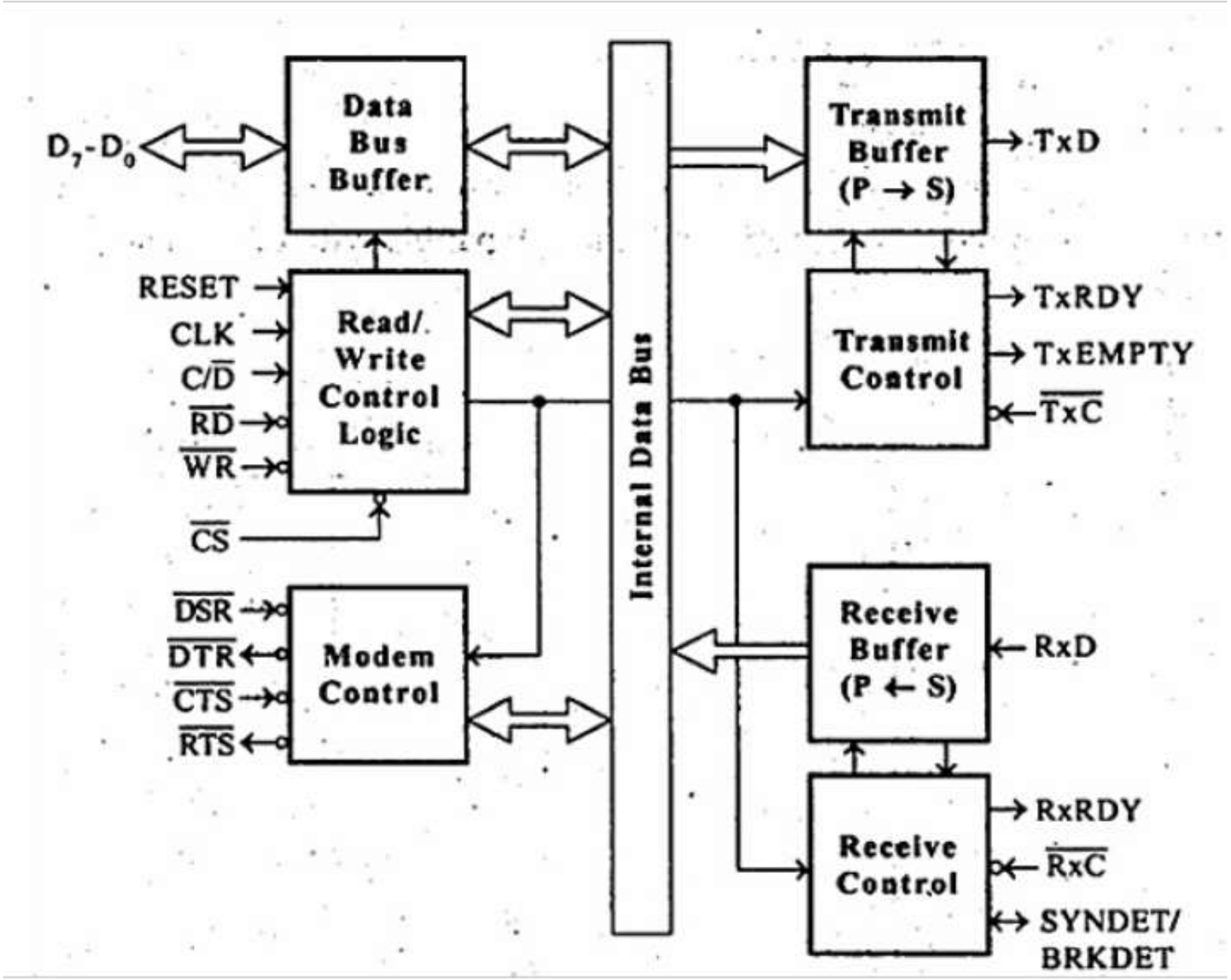


Pin description of the DIP package

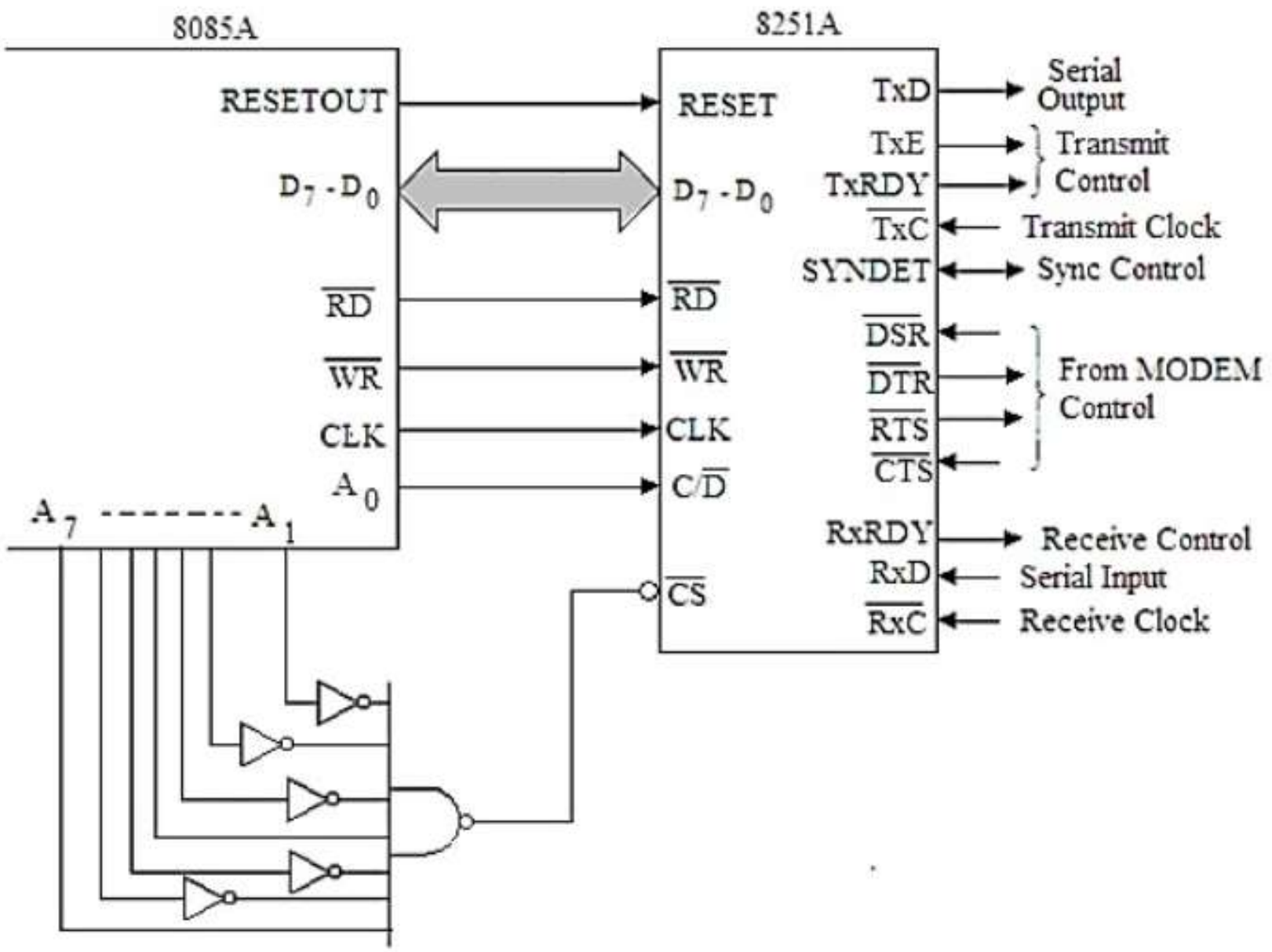
Pin	Description
D_0 - D_7	parallel data
C/$\overline{D}$	Control register or Data buffer select
$\overline{RD}$	Read Control
$\overline{WR}$	Write control
$\overline{CS}$	Chip Select
CLK	clock pulse
RESET	Reset
$\overline{TxC}$	Transmitter Clock
$\overline{TxD}$	transmitted data
$\overline{RxC}$	Receiver Clock
RxD	Receiver Data
RxRDY	Receiver Ready
TxRDY	Transmitter Ready
$\overline{DSR}$	Data Set Ready

$\overline{DTR}$	Data Terminal Ready
SYNDET/	Synchronous Detect/
BRKDET	DetectBreak
$\overline{RTS}$	Request to send Data
$\overline{CTS}$	Clear to send Data
TxEMPTY	Transmitter Empty
V_{cc}	V_{cc} (5V)
GND	Ground(0V)

Block diagram of USART (8251A)



Interfacing USART (8251A) with 8085A MPU



Mapping of 8251 with 8085A

The **C/$\overline{D}$** pin is used to select either control register or data register. This pin is connected to the A_0 pin of 8085. The $\overline{CS}$ pin of 8251 is attached to the output of an address decoder circuit. The address decoder uses A_7 to A_1 lines of the microprocessor. In this diagram the $\overline{CS}$ will be enabled when A_7 and A_4 is at logic 1, and all other lines are at logic 0.

From the following table, we can see how to read or write data word, read the status word and write control word.

A0	$\overline{RD}$	$\overline{WR}$	Task	Port Address
0	0	1	Read Data Word	90H
0	1	0	Write Data Word	90H
1	0	1	Read Status Word	91H
1	1	0	Write Control Word	91H

Programmable Peripheral Interface (PPI) IC8255

- PPI 8255 is a general purpose programmable I/O device designed to interface the CPU with its outside world such as ADC, DAC, keyboard etc
- It can be used with almost any microprocessor
- It consists of three 8-bit bidirectional I/O ports i.e. PORT A, PORT B and PORT C. We can assign different ports as input or output functions
- We can program it according to the given condition

12.2 PROGRAMMABLE PERIPHERAL INTERFACING CHIP 8255

8255 is a general purpose programmable peripheral interfacing chip. It is a 40-pin IC which operates on +5 V power supply. It is designed to work in two modes, i.e. IO modes and bit set reset (BSR) modes. It consists of three 8-bit IO ports named port A, port B, port C. port A and port B always behave as 8-bit ports, whereas, port C can be used as a single 8-bit IO port or as two 4-bit IO ports or we can use the individual pins of port C as individual lines. These three ports are divided into two groups, i.e. group A and group B. Group A consists of port A and port C upper (PC_U), whereas, group B consists of port B and port C lower (PC_L). Here port C upper (PC_U) means the signals PC_4, PC_5, PC_6 and PC_7 and port C lower (PC_L) means the pins PC_0, PC_1, PC_2, PC_3. In which mode these three ports operate is decided by the D_7 bit of the control word register (CWR). If the D_7 bit of the CWR is 0, then 8255 will operate in BSR mode and if the D_7 bit is 1, then it will operate in IO mode. Figure 12.1 shows the modes of 8255. The IO mode is further classified as

Mode 0 – Basic or simple input/output
Mode 1 – Strobed input/output
Mode 2 – Bidirectional bus

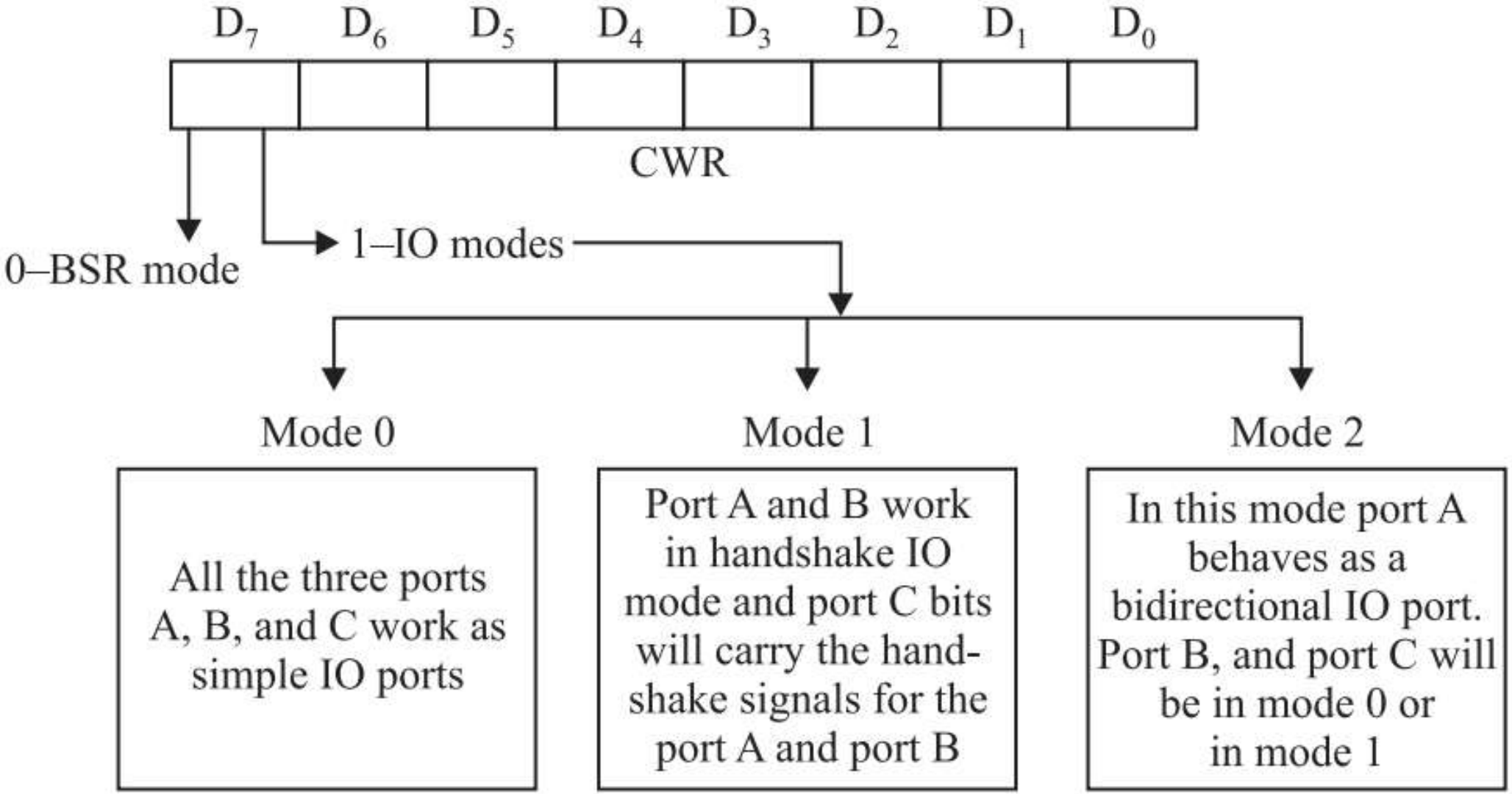
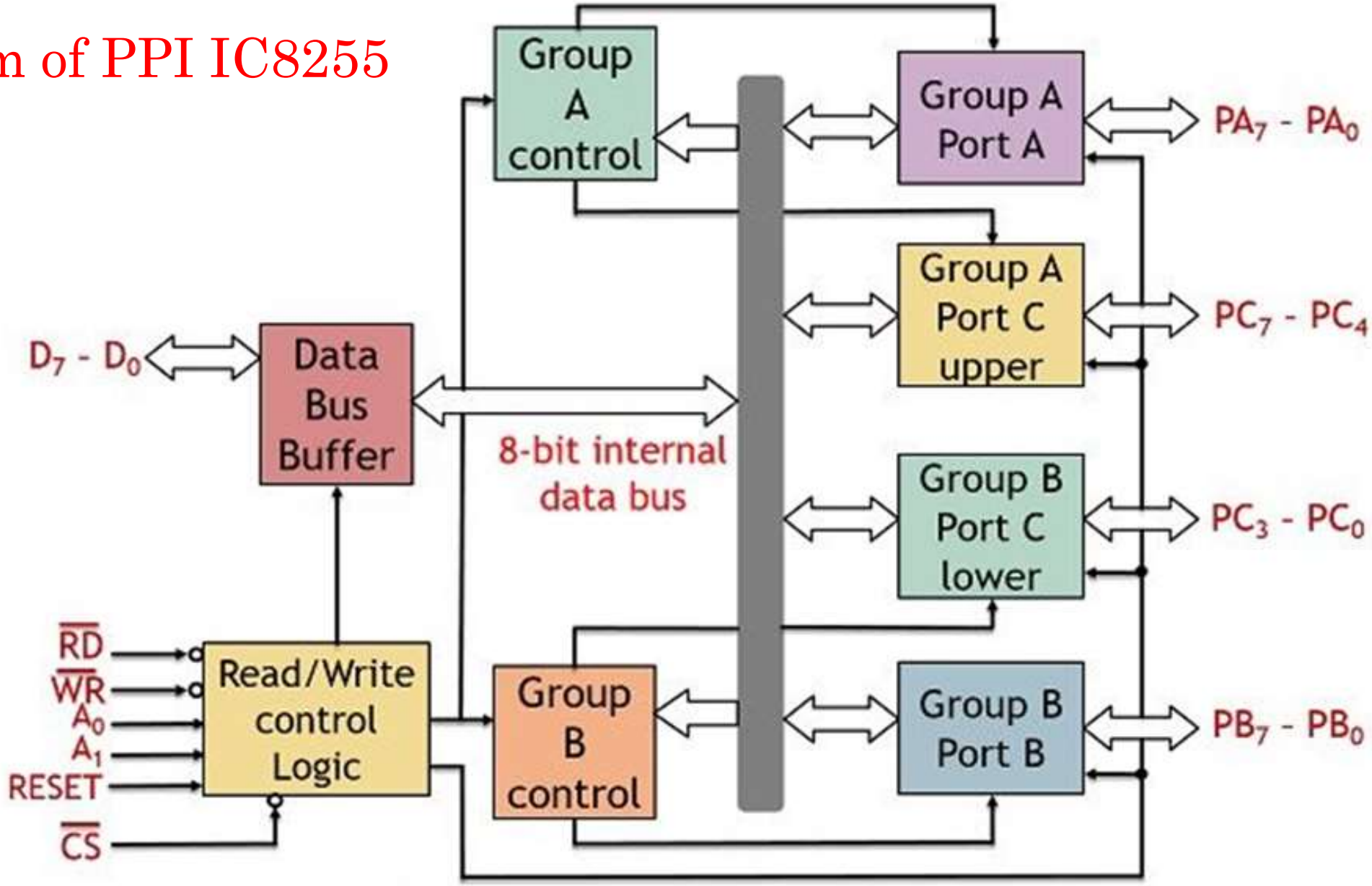


Figure 12.1 Basic modes of 8255.

Block diagram of PPI IC8255



Block diagram of PPI- IC8255

The block diagram of 8255 mainly consists of four parts as shown in Figure 12.2. These are:

1. Data bus buffer
2. Read/write control logic
3. IO ports
4. Group A and group B control logic.

Block diagram of PPI- IC8255

1. **Data bus buffer:** This is an 8-bit bidirectional tri-stated data bus buffer which is used to interface 8255 data bus with the D_0–D_7 data bus of the system bus. It is internally connected to the internal data bus of 8255 and the demultiplexed D_0–D_7 bus of the MPU. These lines are used to transfer the data to the IO ports or the control word to the CWR or the status word to the MPU. The direction of data flow will be decided by the Read or Write control signals. If the read control signal is asserted, the data will go to the MPU and if write control signal is asserted, then data will go from the MPU.
2. **Read/write control logic:** This unit of 8255 receives its signal from the control logic of MPU and the address bus of the system. The RD^{-} and WR^{-} are connected to the IOR^{-} and IOW^{-} signals or to the $MEMR^{-}$ and $MEMW^{-}$ signals depending upon whether the 8255 is interfaced in IO mapped IO technique or interfaced with Memory mapped IO technique respectively. The A_1 and A_0 are generally connected to the A_1 and A_0 address signals of the system. These two signals are used to decide the address of the three ports and the CWR. The remaining address lines of the system are used to select the 8255 chip through its CS^{-} signal. The reset line of 8255 is interfaced with the reset out signal of the clock generator of 8086.

Block diagram of PPI- IC8255

3. **IO ports:** This section of 8255 consists of the three IO ports, i.e. Port A, B, and Port C. All these ports are bidirectional in nature which are having internal output latch/ buffers and input buffers. These ports are divided into two groups, group A and group B as already discussed. The port C can be used as (1) simple IO, (2) handshake signals, and (3) status signal inputs. For handshake signals and status signals it is used in coordination with port A and port B. The bit set/reset capability is only provided by port C that means BSR mode is only for port C.
4. **Group A and group B control logic:** 8255 IO ports are divided into two groups. Each group is programmed through software. Each of the Control blocks (Group A and Group B) accepts commands from the Read/Write Control Logic, receives control words from the internal data bus and issues the proper commands to its associated ports. G_A control will control G_A ports and G_B control will control G_B ports.

About CWR (Control Word Register)

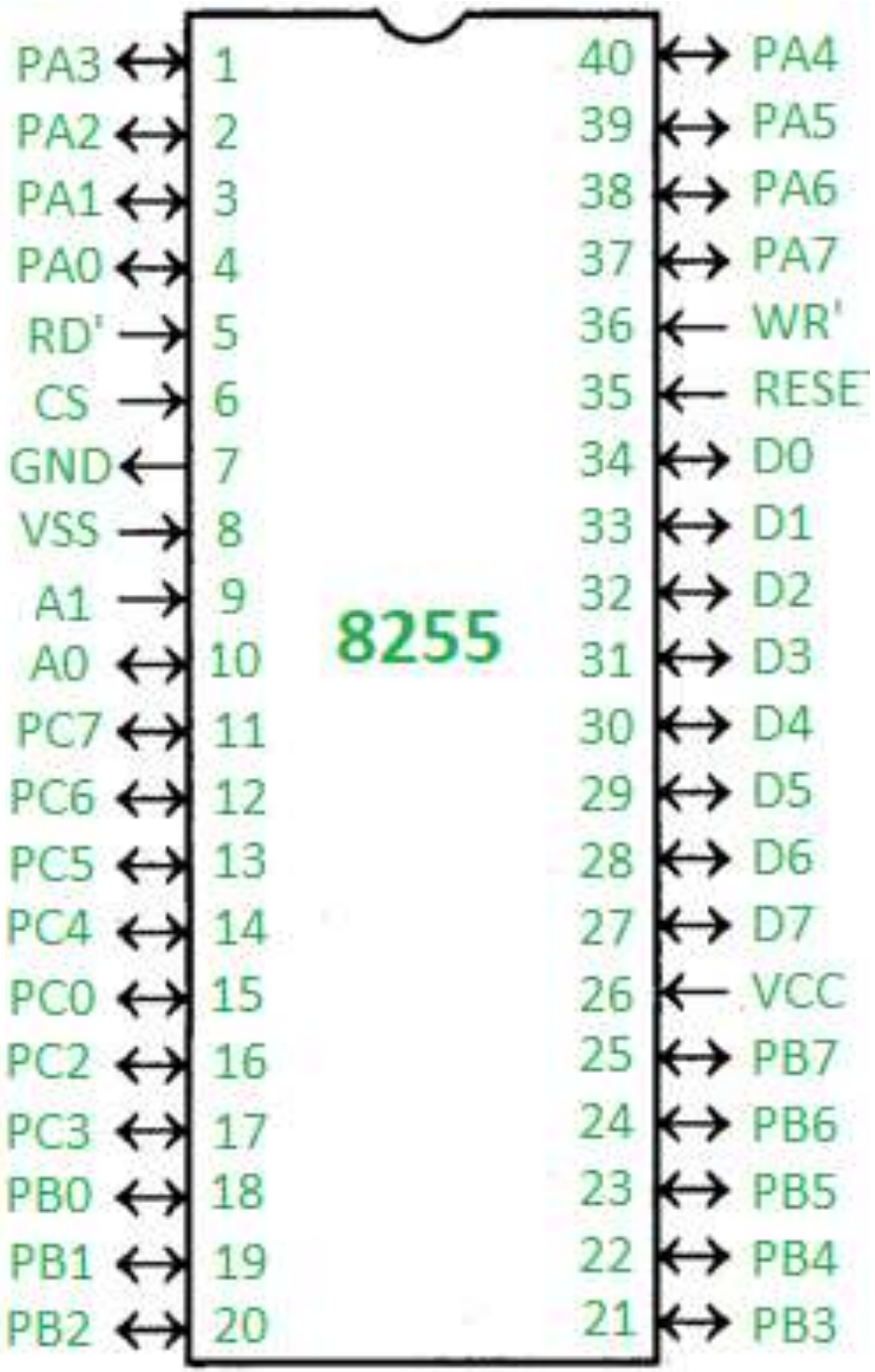
In essence, the processor sends outputs a control word to the 8255A. The control word contains information such as mode, bit set, bit reset, etc. that initializes the functional configuration of the 8255A.

The control word register can only be written into. No Read operation of the Control Word Register is allowed.

Pin diagram of PPI 8255

1. **D_0–D_7 (Data bus):** These are 8-bit bidirectional data bus, connected to system data bus for data transfer between 8086 and 8255. On these data lines 8086 will send control word data to initialize 8255, send data or read data.
2. **RD⁻ (Read):** This is for read operation. It is active low signal which is input to the 8255. A low on this input pin enables the 8255 to send the data or status information to the CPU on the data bus. In essence, it allows the processor to read from the 8255.
3. **WR⁻ (Write):** This is for write operation. It is active low signal which is input to the 8255. A low on the input pin enables the CPU to write data or control words into the 8255.
4. **RESET:** This signal is used to reset the 8255. A high on this input signal clears the control register and all ports (A, B, C) are set to the input mode. This signal is connected to the *reset out of 8085* and reset signal of the clock generator of 8086.
5. **Ports A, B, and C:** The 8255A contains three 8-bit ports (A, B, and C). All can be configured in a wide variety of functional characteristics by the system software but each has its own special features that further enhances the power and flexibility of the 8255A.

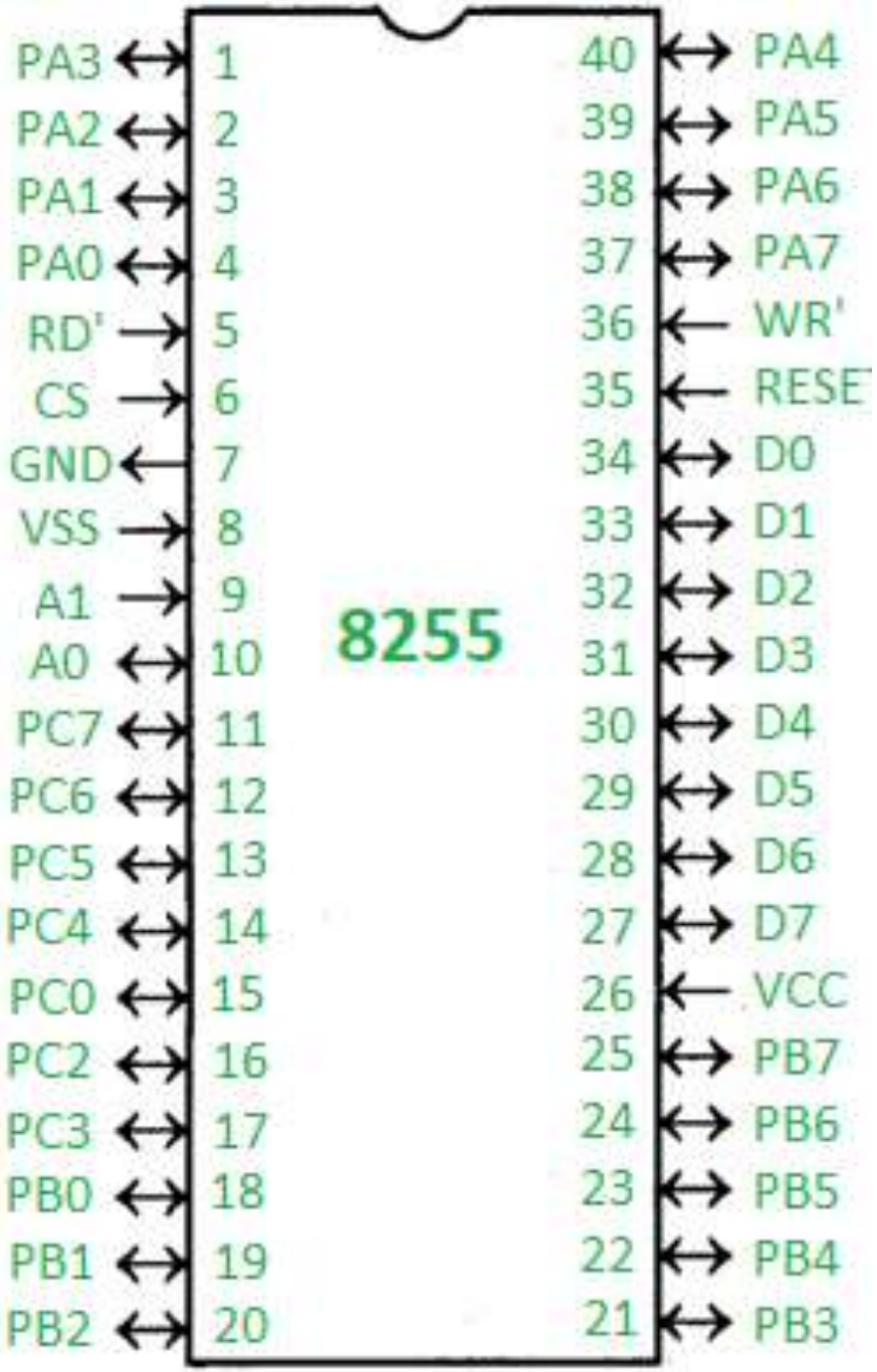
   Port A: One 8-bit data output latch/buffer and one 8-bit data input latch.
   Port B: One 8-bit data output latch/buffer and one 8-bit data input buffer.
   Port C: One 8-bit data output latch/buffer and one 8-bit data input buffer (no latch for input). This port can be divided into two 4-bit ports under the mode control. Each 4-bit port contains a 4-bit latch and it can be used for the control signal outputs and status signal inputs in conjunction with ports A and B.



Pin diagram of PPI 8255

6. **A_0 and A_1 (Address lines):** These two input signals were connected with the A_1 and A_0 address lines of the demultiplexed address lines. These two signals will give the addresses of the port addresses and the CWR as follows:

A_1	A_0	Port
0	0	P_A
0	1	P_B
1	0	P_C
1	1	CWR



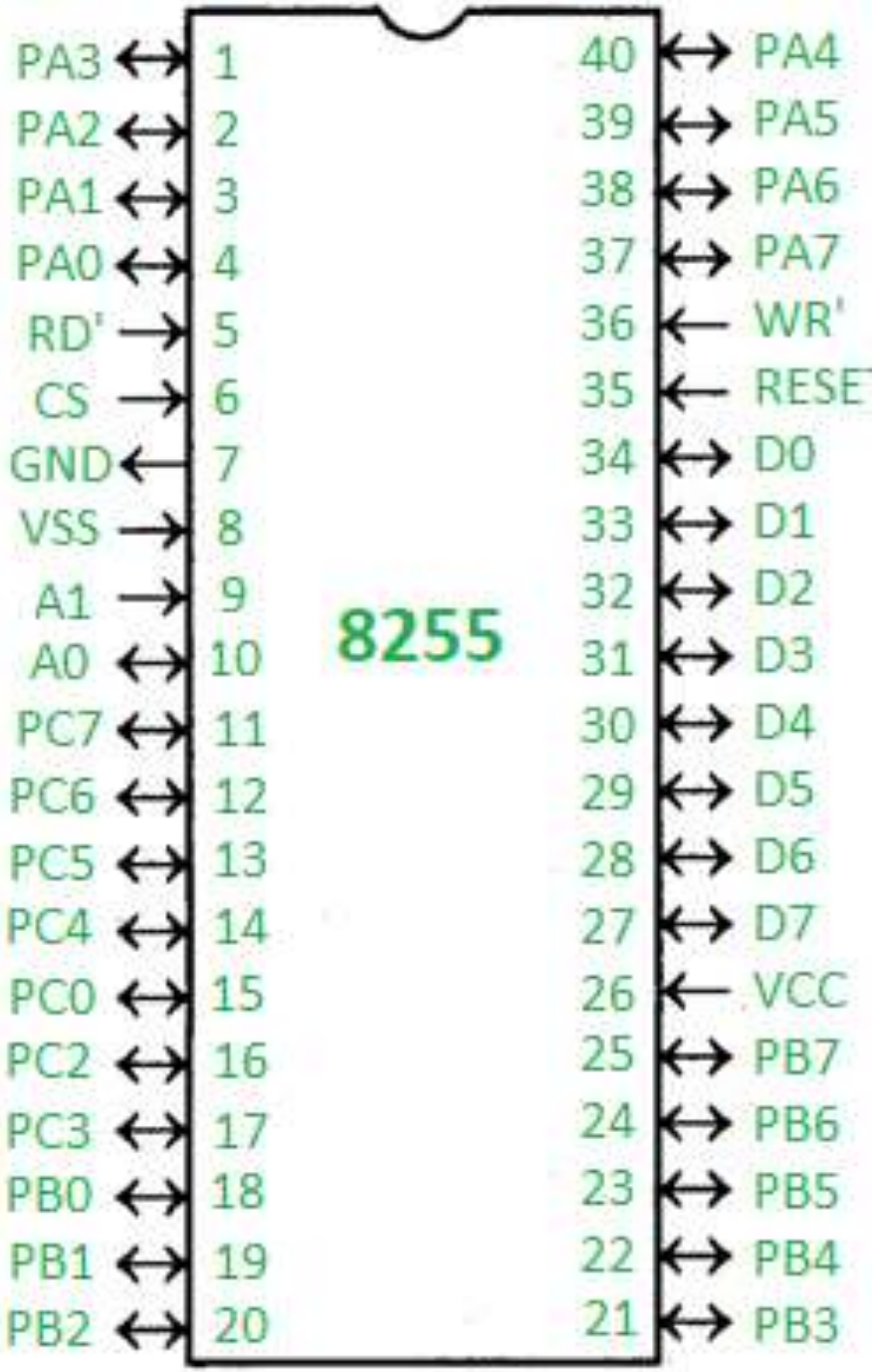
Pin diagram of PPI 8255

The input signals A_1–A_0, in conjunction with the RD^- and WR^- inputs and the chip select signal CS^- define the basic IO operations. These functions are tabulated in Table 12.1.

Table 12.1 IO Operation definitions

A_1	A_0	RD^-	WR^-	CS^-	Operation
0	0	0	1	0	Port A – data bus (input)
0	1	0	1	0	Port B – data bus (input)
1	0	0	1	0	Port C – data bus (input)
0	0	1	0	0	Data bus – port A (output)
0	1	1	0	0	Data bus – port B (output)
1	0	1	0	0	Data bus – port C (output)
1	1	1	0	0	Data bus – control (output)
X	X	X	X	1	Data bus – 3 state
1	1	0	1	0	Illegal condition
X	X	1	1	0	Data bus – 3 state

7. **CS^- (Chip select):** A low on this signal select the chip. The chip select signal generated by the A_2–A_7 address lines (in case of IO mapped IO) or A_2–A_{15} address lines (in case of Memory mapped IO) is connected to this signal. The combination of these address lines along with the A_1–A_0 generates the actual 8-bit or the 16-bit port addresses.



Operation description of PPI 8255

The 8255 chip have one control word register. This register is selected when $A_1 = A_0 = 1$, $CS^{-} = 0$ and $WR^{-} = 0$ that is a write operation. We prepare a control word according to the bits defined in the control word and load this bit pattern into the control word register to initialize 8255. The D_7 bit of the control word register defines whether this chip is to be used for IO modes or for the Bit Set–Reset (BSR) mode. If the D_7 bit is 0 then this chip will be initialized in BSR mode and then the remaining bit of the CWR will be defined as per BSR mode. If the D_7 bit is 1 then 8255 will work in IO mode and then the remaining bits of the CWR will define the IO operations. The D_6 to D_0 bits of the CWR will have different definitions for the two modes.

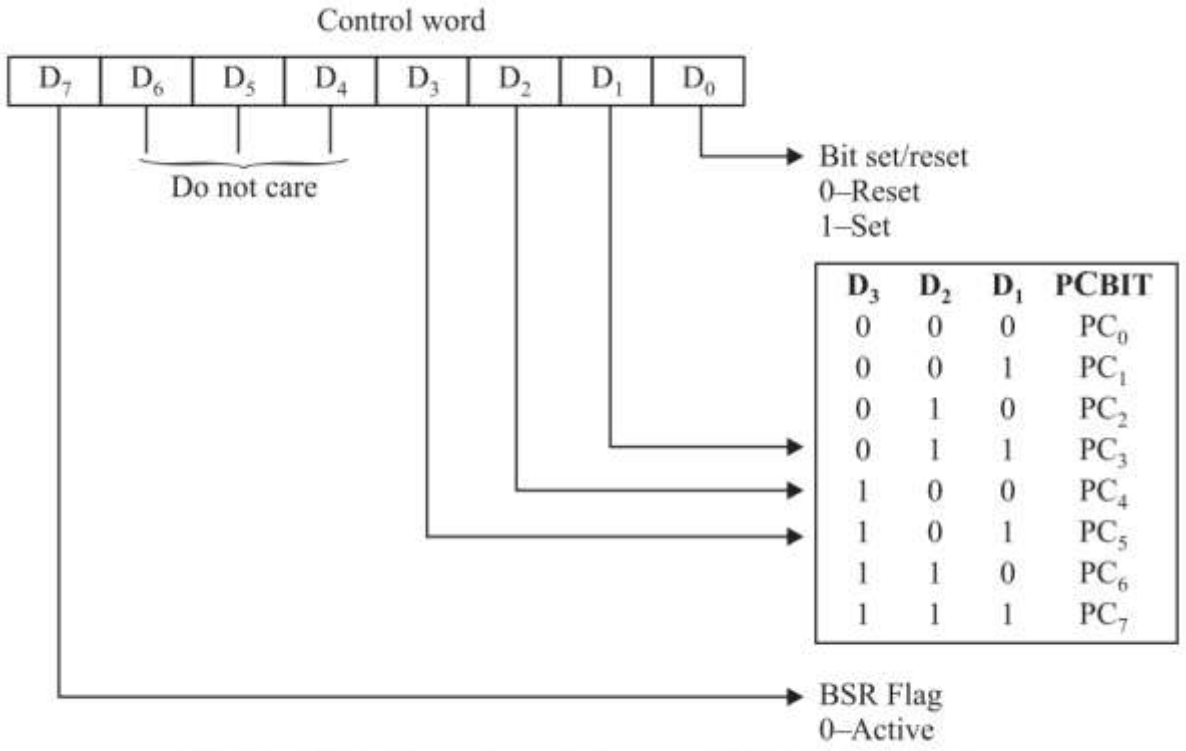
Operation description (BSR mode) of PPI 8255

12.3.1 Bit Set Reset (BSR) Mode

Any of the eight bits of port C can be set or reset using a single output instruction. This feature reduces software requirements in control-based applications.

The control word format of 8255 in BSR mode is shown in Figure 12.4. The bits D_3, D_2, and D_1 of this control word register will define which of the bit of the port C is to be set or reset. The D_0 bit of this CWR will state whether the respective bit of port C is to be set or reset. For example, if we want to reset the PC_5 bit of port C then $D_3D_2D_1 = 101$ and $D_0 = 0$. The complete CWR bit pattern to reset PC_5 will be 0XXX1010. A BSR mode will not affect an IO mode.

Operation description (BSR mode) of PPI 8255



D_3	D_2	D_1	PCBIT
0	0	0	PC_0
0	0	1	PC_1
0	1	0	PC_2
0	1	1	PC_3
1	0	0	PC_4
1	0	1	PC_5
1	1	0	PC_6
1	1	1	PC_7

Figure 12.4 Control word format of 8255 in bsr mode.

When port C is being used as status/control for port A or B, these bits can be set or reset by using the bit set/reset operation just as if they were data output port.

Operation description (IO mode) of PPI 8255

12.3.2 IO Mode

There are three basic modes of operation that can be selected by the systems software:

Mode 0 – Basic input/output
Mode 1 – Strobed input/output
Mode 2 – Bidirectional bus

When the reset input goes high, all ports will be set to the input mode (i.e. all 24 lines will be in the high impedance state). After the reset is removed, the 8255A can remain in the input mode with no additional initialization required. During the execution of the systems program any of the other modes may be selected using a single output instruction. This allows a single 8255 to service a variety of peripheral devices with a simple software maintenance routine.

The modes for ports A and port B can be separately defined, while port C is divided into two portions as required by the port A and port B definitions. All of the output registers, including the status flip-flops, will be reset whenever the mode is changed. Modes may be combined so that their functional definition can be tailored to almost any I/O stricture.

12.3.3 Control Word Register of 8255 in IO Mode

The control word register (CWR) format in IO mode is shown in Figure 12.5. The CWR is used to define or initialize the operation of the 8255. Its bits are used to define that in which mode the three ports will operate and the mode of operations of these ports. The functions of these bits are as follows:

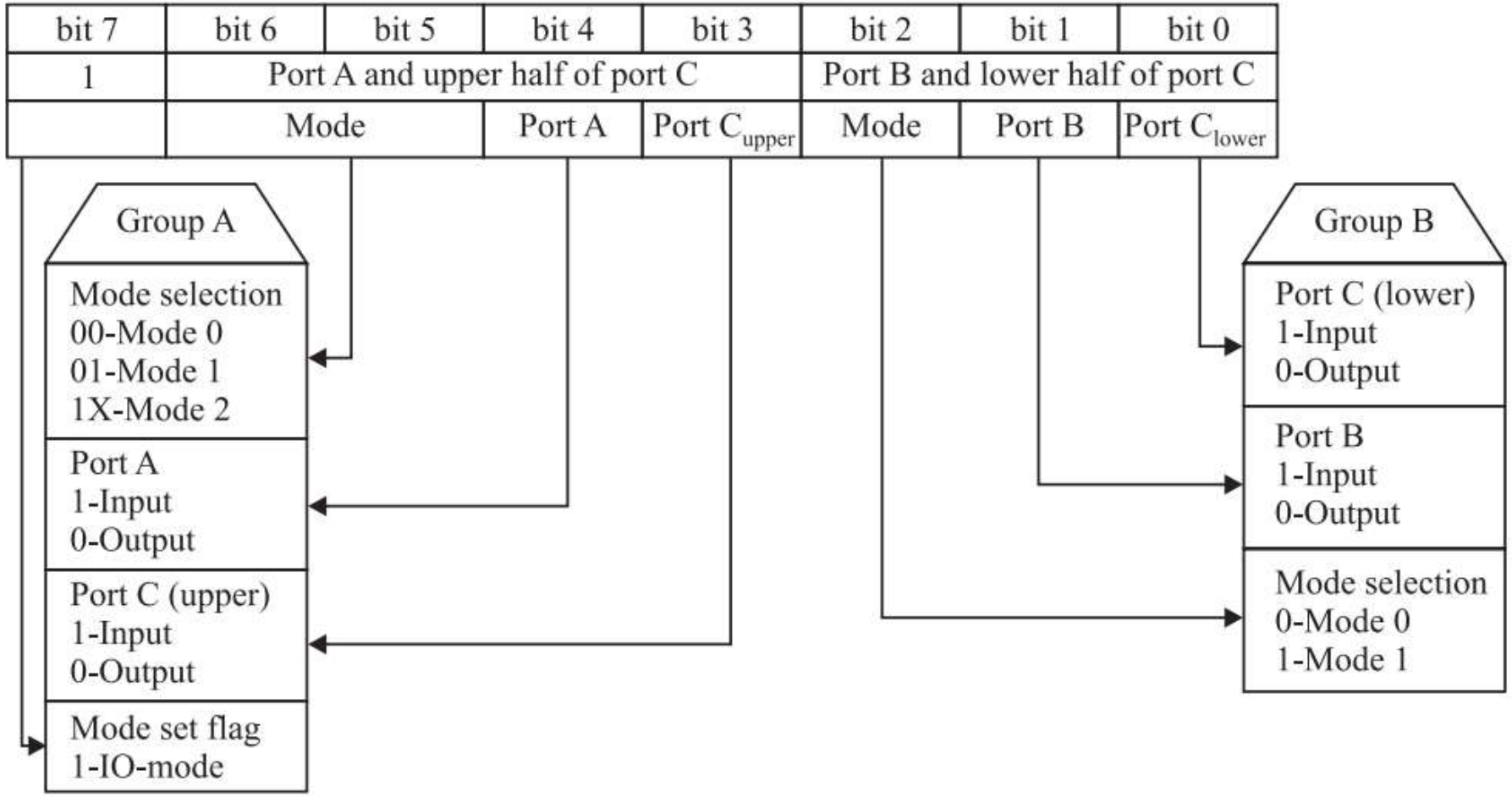


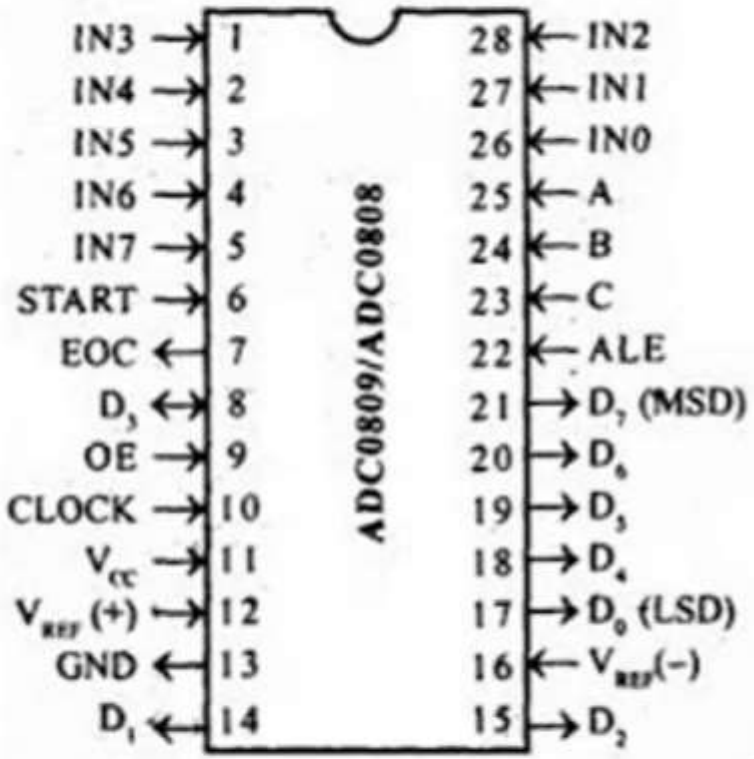
Figure 12.5 Control word register of 8255 in IO mode.

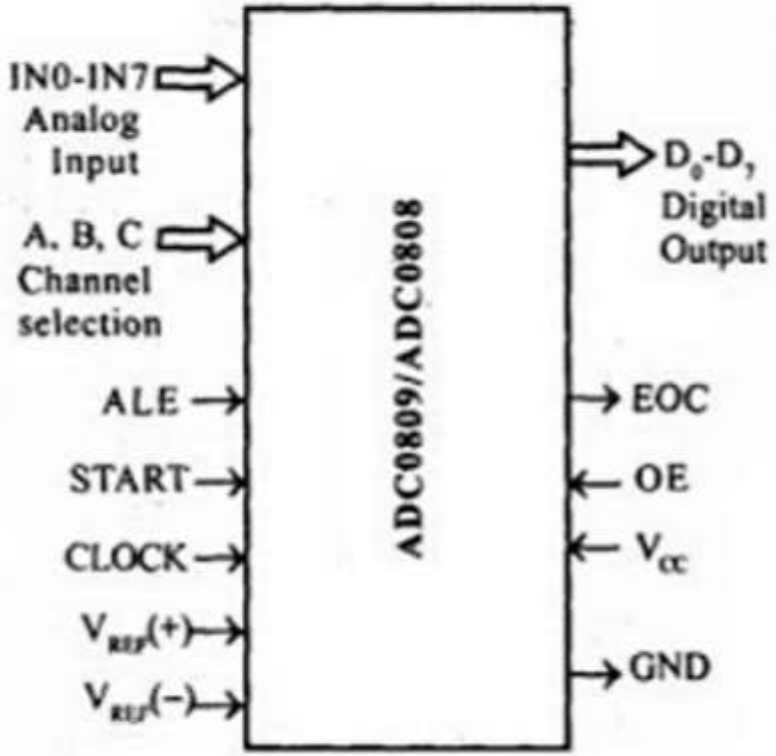
ADC 0808 interfacing: Intro to 0808/09 IC

- It is an input device which converts analog voltage to digital value
- ADC 0808 is an 8-bit ADC
- It is highly versatile device widely used in data acquisition system, µP based instrumentation system, control etc.,
- It operates on +5 V, its input range is 0 to 5V
- Its operation is based on successive approximation technique
- It typical conversion time is 100 µs with power consumption
- It consists of 8 multiplexed input channels (IN0-IN7) for analog inputs
- 8 channels are selected using 3 channel selection lines (A, B, C)
- In order to do this conversion and inputting data to µP, first address of the input channel (channel selection) is done after 50 ns. ALE signal is made high, after another 2.5 µs, Start of Conversion (SOC) line is made high
- After completion of conversion, End of Conversion (EOC) is passed which indicates the digital equivalent of the analog input given is available at the output port of ADC which is D0-D7

ADC IC0808 PIN diagram

Interface ADC 0808 with 8085 μP





LSD = Least Significant Digit, MSD = Most Significant Digit

Understanding the connectivity through 8255

Interface ADC 0808 with 8085 μP

8 bit ADC
+5V DC
8 Multiplexed input channels

Channels	C	B	A
IN0	0	0	0
IN1	0	0	1
IN2	0	1	0
IN3	0	1	1
IN4	1	0	0
IN5	0	0	1
IN6	1	1	0
IN7	1	1	1

SOC ← PC3

EOC

PC0

PC1

PC2

PC Lower - Output port

PC Upper - Input port

To take digital output to **μP**

↑

Port A – Input port

Port B – Not required (Let it be O/P)

Designating ports of 8255 & CWR generation

Port A – Input port

Port C upper – Input port

Port B – Output port

Port C Lower – Output port

MVI A, 98 H

OUT 03H

D7	D6	D5	D4	D3	D2	D1	D0
Modes of operatio n=1	Mode of Port A	Mode of Port A	Port A (I/O)	Port Cupper (I/O)	Mode of Port B	Port B (I/O)	Port C lower (I/O)
1	0	0	1	1	0	0	0

9 8

Starting, A/D conversion and reading digital data by µP

How to provide SOC pulse

PC3 make initially high then Low

Lets consider channel 0

	PC3	PC2	PC1	PC0
Initial :	1	0	0	0
Final :	0	0	0	0

```
MVI A, 08
OUT 02

MVI A, 00
OUT 02
```

To pass SOC pulse

Now ADC starts conversion

Need to check conversion completed or not

```
REPEAT :  IN, 02
          RAL
          JNC REPEAT
```

To check EOC reception

If completed read the digital value through Port A and save into any location

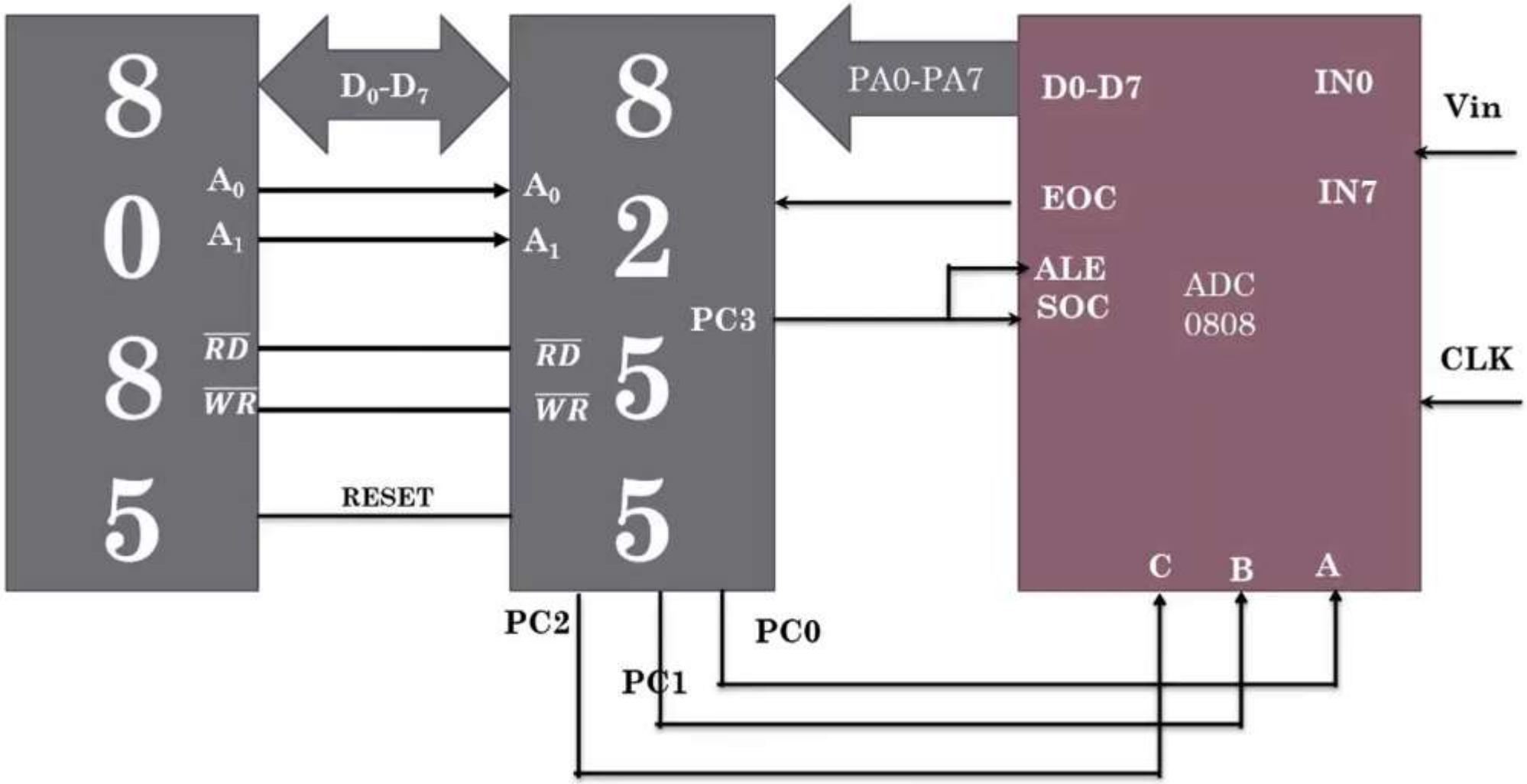
```
IN 00
STA 3000H
HLT
```

To read digital data through Port A of 8255

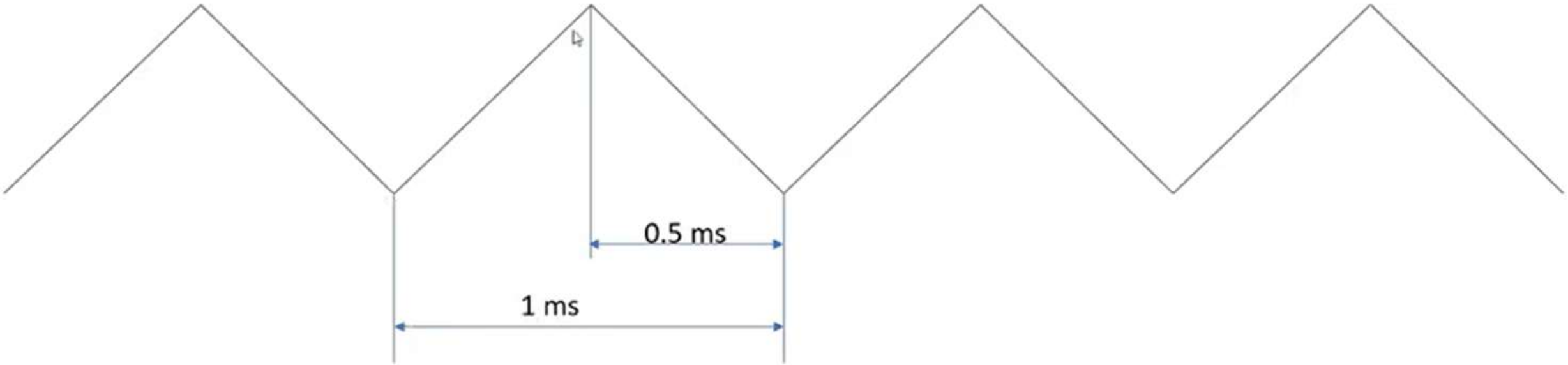
ADC 0808 interfacing: Complete program

```
        MVI A, 98 H     ;Assigning CWR data in Accumulator
        OUT 03H         ;Sending A data to CWR of 8255
        MVI A, 08H      ;Assigning SOC ON pulse data (for PC_LOWER) in acc.
        OUT 02H         ; Sending A to Port C (PC_LOWER)
        MVI A, 00H      ;Assigning SOC OFF pulse data (for PC_LOWER) in acc.
        OUT 02H         ;Sending the same to Port C (PC_LOWER)
REPEAT:IN, 02H          ;Checking for EOC reception using Port C (7th bit) of 8255
        RAL             ;Rotating left through carry to check the status of 7th bit
        JNC REPEAT      ;Waiting until EOC is received or proceeding to reading the digital data if EOC is received
        IN 00H          ;Reading the digital data from Port A of 8255
        STA 3000H       ;storing the read data in some memory location say 3000h
        HLT             ;halting the program
```

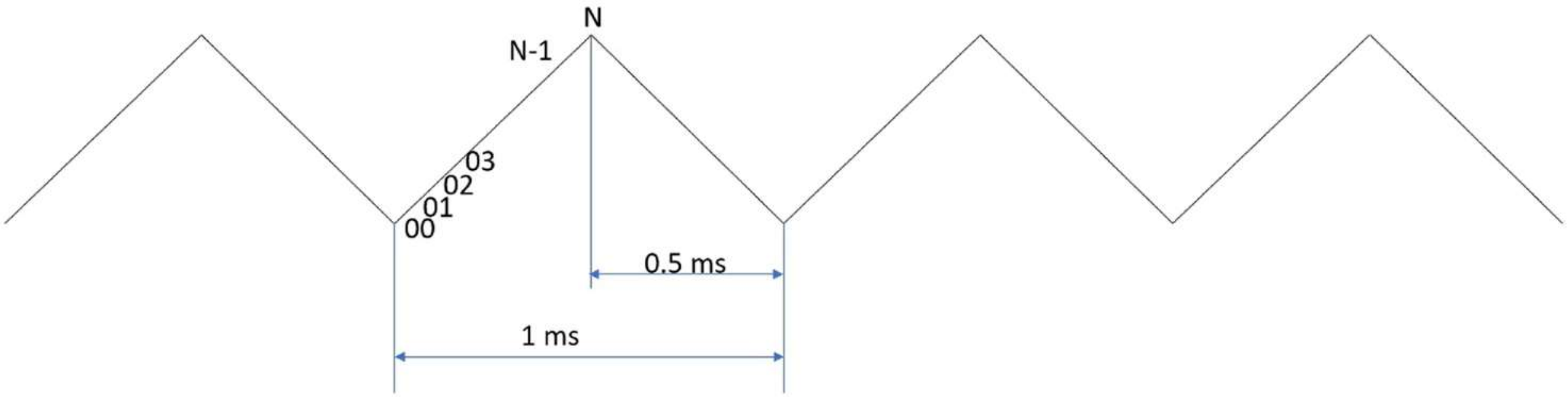
ADC 0808 interfacing: Interfacing diagram



Triangular waveform generation using DAC interfaced with 8085 MP



Triangular waveform generation using DAC interfaced with 8085 MP

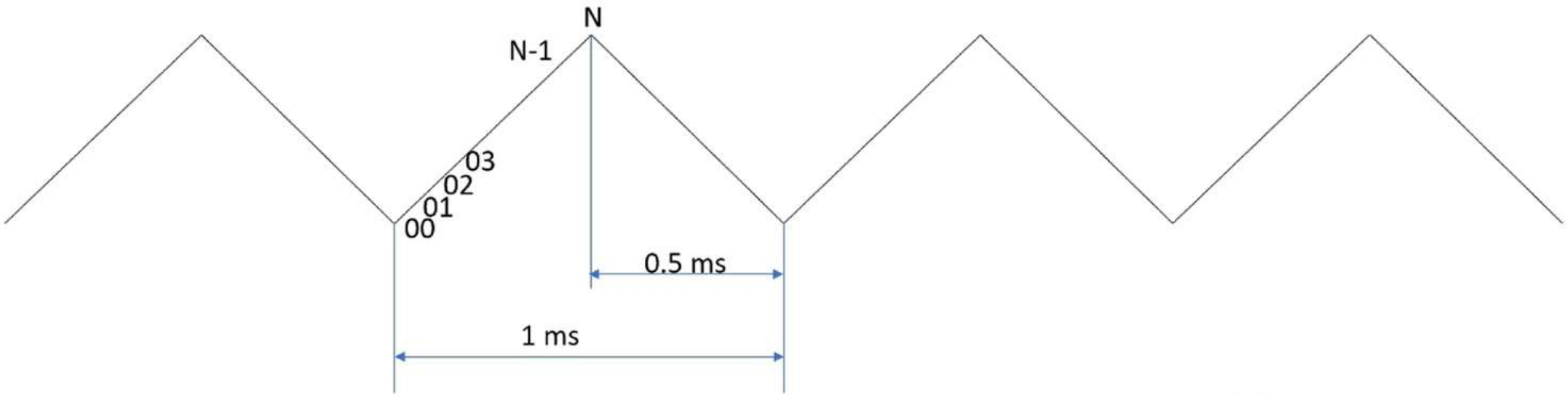


Instruction

Total T states

MVI A,00
OUT 30 H

Triangular waveform generation using DAC interfaced with 8085 MP

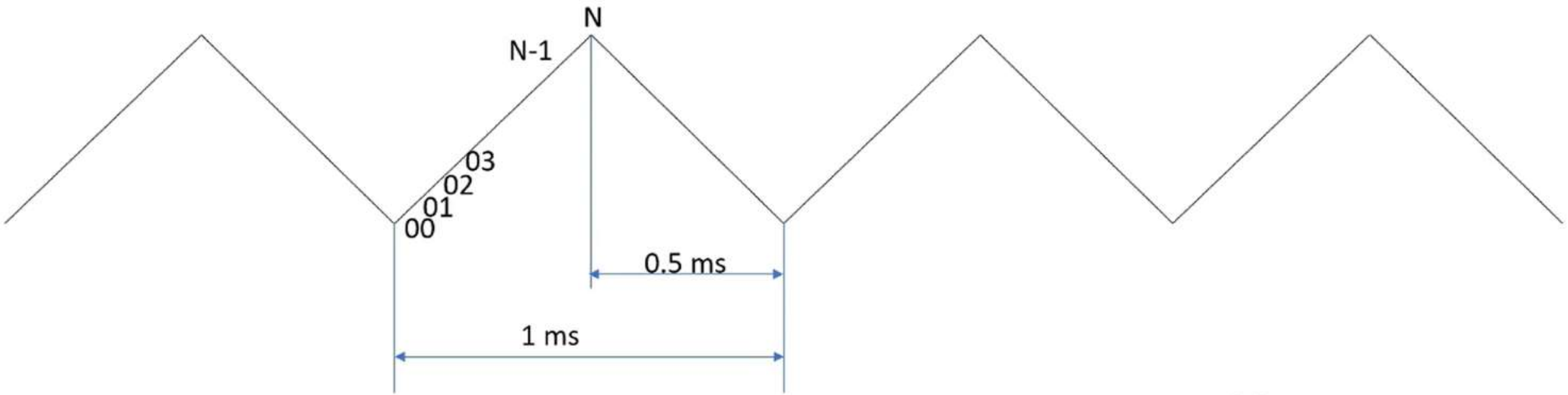


Instruction	Total T states
MVI A,00	1 * 7T
Display: OUT 30 H	N *10T
INR A	N *4T
CPI N	N *7T
JNZ Display	(N-1)*10T+7T
	(7+10N+4N+7N+10N-10+7)T

(31N+4)T = 0.5 ms

(31N+4)320*10^-9 = 0.5 ms

Triangular waveform generation using DAC interfaced with 8085 MP



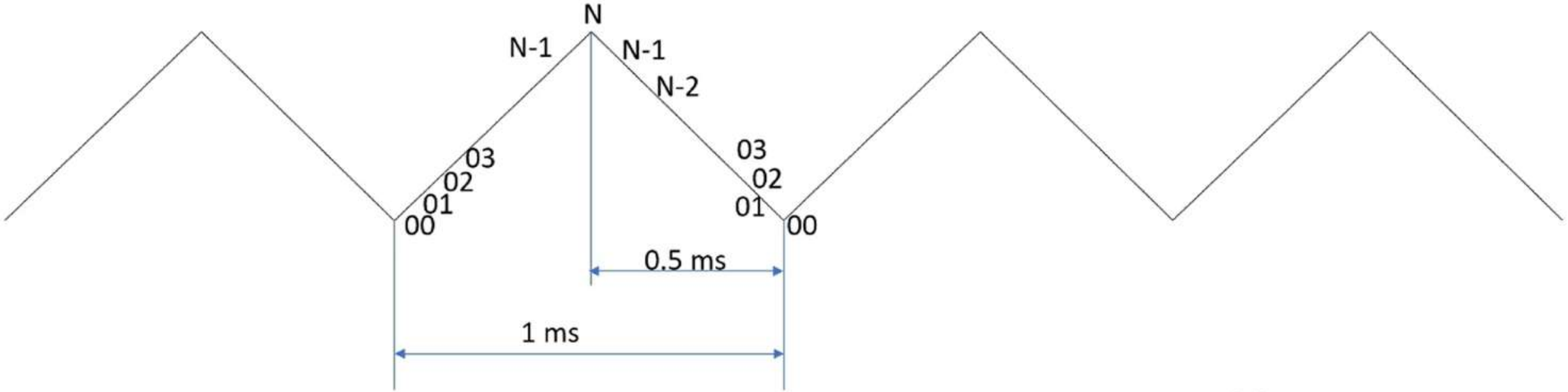
Instruction	Total T states
MVI A,00	1 * 7T
Display: OUT 30 H	N *10T
INR A	N *4T
CPI N	N *7T
JNZ Display	(N-1)*10T+7T
	(7+10N+4N+7N+10N-10+7)T

(31N+4)T = 0.5 ms

(31N+4)320*10^-9 = 0.5 ms

N =50d = 32H

Triangular waveform generation using DAC interfaced with 8085 MP



Instruction	Total T states
MVI A,00	1 * 7T
Display: OUT 30 H	N *10T
INR A	N *4T
CPI N	N *7T
JNZ Display	(N-1)*10T+7T
	(7+10N+4N+7N+10N-10+7)T

(31N+4)T = 0.5 ms

(31N+4)320*10^-9 = 0.5 ms

N =50d = 32H

Triangular waveform generation using DAC interfaced with 8085 MP

#1 : Load control word number to make all ports of 8255 as O/P port in A register

#2 : OUT control word format i.e. (A) in to control word register

#3 : Load A with 00H and out it into port A of 8255 (to DAC)

#4 : Increment A, out the value through port A of 8255 (to DAC)

#5 : Repeat Incrementing A , till (A) reaches to 32H

```
        MVI A,80H
        OUT 33H
        MVI A,00
Display+ : OUT 30 H
        INR A
        CPI 32H
        JNZ Display+
```

Triangular waveform generation using DAC interfaced with 8085 MP

#1 : Load control word number to make all ports of 8255 as O/P port in A register

```
MVI A,80H
```

#2 : OUT control word format i.e. (A) in to control word register

```
OUT 33H
```

#3 : Load A with 00H and out it into port A of 8255 (to DAC)

#4 : Increment A, out the value through port A of 8255 (to DAC)

#5 : Repeat Incrementing A , till (A) reaches to 32H

```
        MVI A,00
Display+ : OUT 30 H
        INR A
        CPI 32H
        JNZ Display+
```

#6 : Decrement A , till (A) reaches to 00H

```
Display-: OUT 30 H
        DCR A
        CPI 00H
        JNZ Display-
```

Triangular waveform generation using DAC interfaced with 8085 MP

#1 : Load control word number to make all ports of 8255 as O/P port in A register

#2 : OUT control word format i.e. (A) in to control word register

#3 : Load A with 00H and out it into port A of 8255 (to DAC)

#4 : Increment A, out the value through port A of 8255 (to DAC)

#5 : Repeat Incrementing A , till (A) reaches to 32H

#6 : Decrement A , till (A) reaches to 00H

#7 : Repeat from step 3

```
        MVI A,80H
        OUT 33H
REPEAT: MVI A,00
Display+ : OUT 30 H
        INR A
        CPI 32H
        JNZ Display+

Display-: OUT 30 H
        DCR A
        CPI 00H
        JNZ Display-

        JMP REPEAT
```

END OF 8085 MPU

THANK YOU