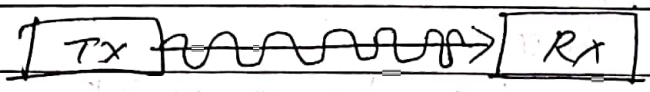
Chapter 5: Basic Data Communication, I/O interface.

Date
Page

Communication : It is the process of exchange of information between the terminals



Transmitter Receiver

-> The interfacing components required for serial I/O peripheral are the same as those for parallel I/O device.

-> The primary difference between parallel I/O & serial I/O is in the no. of lines used for data transfer.

-> The parallel I/O uses the entire data bus where serial I/O uses one data line.

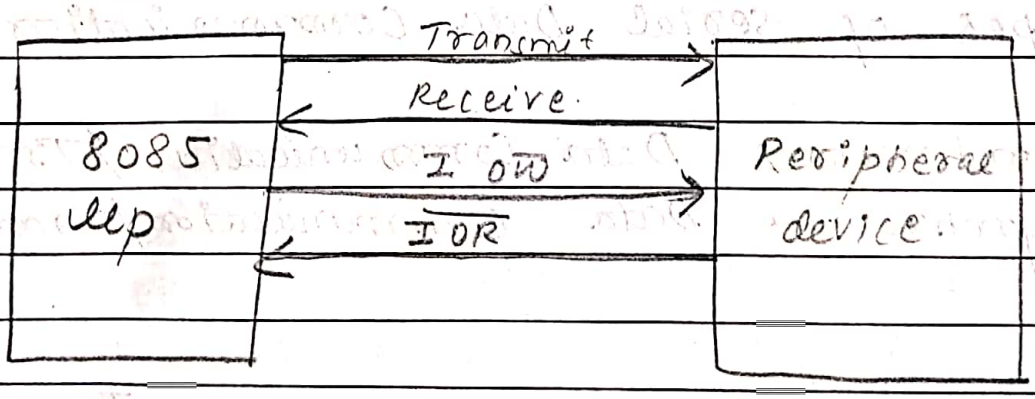


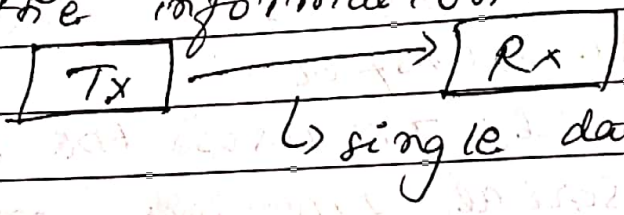
fig: I/O interfacing with processor.

Types of communication.

1) Parallel communication.
2) Serial communication.

1) In parallel communication, the transmitter section will use all the available data lines for sending the information parallelly to Receiver section.

2) In serial communication, the transmitter section will use only a single data line to transmit the information.

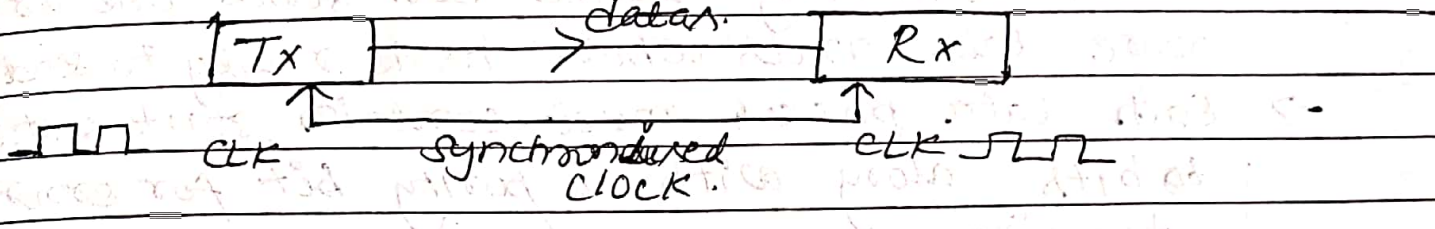


Types of serial Data Communication.

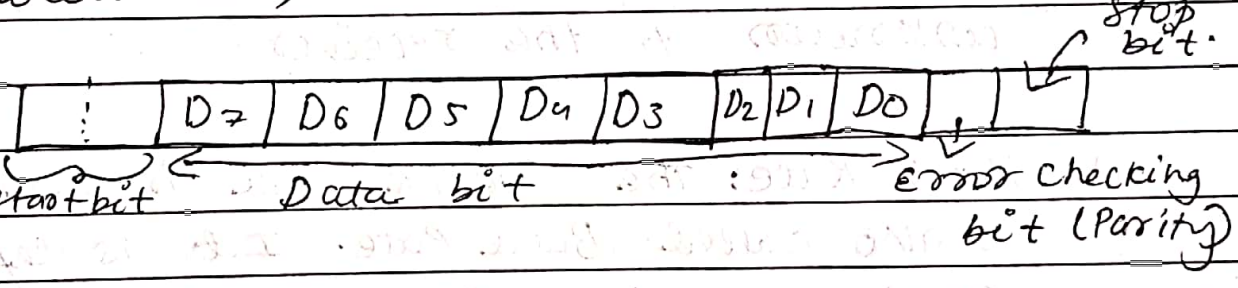
1) Synchronous Data Communication / Transmission.
2) Asynchronous Data communication / Transmission.

1) Synchronous Data communication: The basic feature of synchronous data transmission is that data is transmitted or received based on clock signal. At a specific rate of data transmission, the transmitting device sends a bit of each block pulse to interpret the data correctly, the transmitter must know the no of data to be transformed.

Usually one or two SYNC characters are ~~to be~~ ~~from~~ used to indicate the start of each synchronous data stream.



=> The data unit may contain 7 or 8 bits. It may also contain a parity bit. That can be shown as;



-> The synchronous receiver waits in a "hunt mode" while waiting for data. As soon as the transmitter sends one or two SYNC signals, the receiver starts accepting the data.

-> In synchronous transmission the 'Tx' should send data continuously to the receiver.

-> The data packet must contain ; 1) start bit. 2) stop bit. 3) Error checking bit. 4) Data bit.

2) Serial Asynchronous Data Communication:

⇒ In asynchronous data transmission, the receiving device doesn't need to be synchronized with the transmitting device.

→ The transmitting device can send one or more data units when it is ready to send.

→ Each data packet must contain start & stop bits along with a parity bit for error checking.

→ Each asynchronous serial data unit can be divided into equal time intervals. Hence total size of the data to be sent is unknown to the receiver.

Baud Rate: The serial data transmission rate is also called Baud Rate. It is defined as the number of bits transformed per second since each bit is transmitted over a duration of 1 interval,

$$\text{Baud Rate} = \frac{1}{\text{bit interval}} = \text{bits/sec.}$$

Serial communication can also be classified according to direction & simultaneity of data flow.

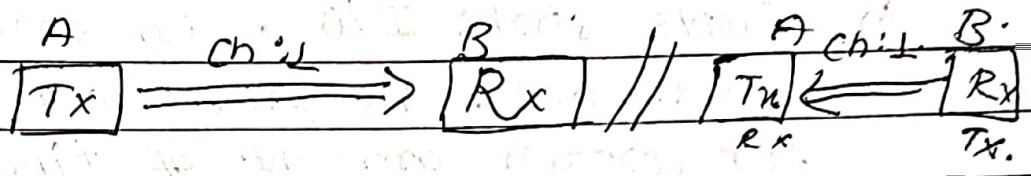
1) Simplex mode; transmission of data over "single channel" in one-direction only.

eg: from A → B

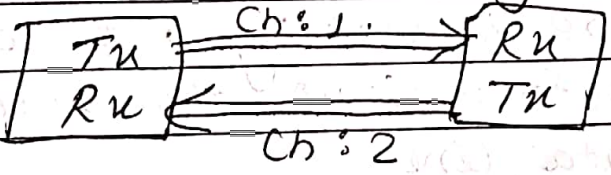
not B → A.

ii) Half - Duplex Mode: transmission of data over a single channel in both direction but not simultaneously.

eg:



iii) Full-duplex mode: transmission of data in both directions simultaneously over two channel.

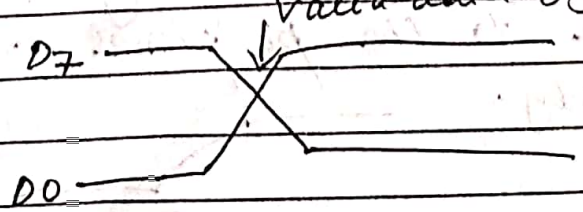


2) Parallel Data Communication / Transmission:

=> The process of sending data bits through the available 8-bit lines together is known as parallel communication.

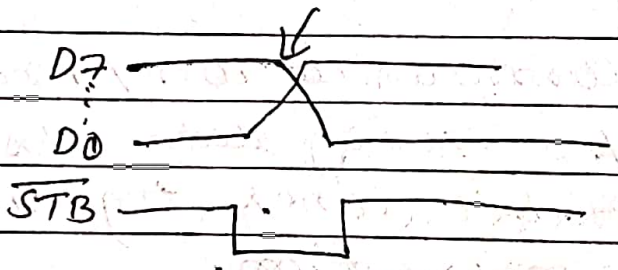
Methods of parallel communication:

(a) simple I/O; When you need to get digital data from a simple switches such as thermostat, thermocouple etc into the µp, all we need to do is to connect the input port line & read the input:



The crossed lined on the waveform represents the time at which new data byte becomes valid on the output lines of the port.

b) Simple strobe I/O ; In some cases, the valid data is present on the device only for certain amount of time. So, it must be read within that time.
For example; when a key is pressed in any circuit boards, the board sends out the code for the pressed key and sends out a strobe signal to indicate that valid data is present in data line.



c) Single Handshake I/O;
-> Handshake signals are used to synchronize slow peripherals with high speed ulps.
-> The peripheral device takes some parallel data and gives an STB (strobe) signal to ulps to indicate that it is ready of transmission.
-> The ulp detects the STB signal and sends an ACK signal to the peripheral device to indicate it is ready.

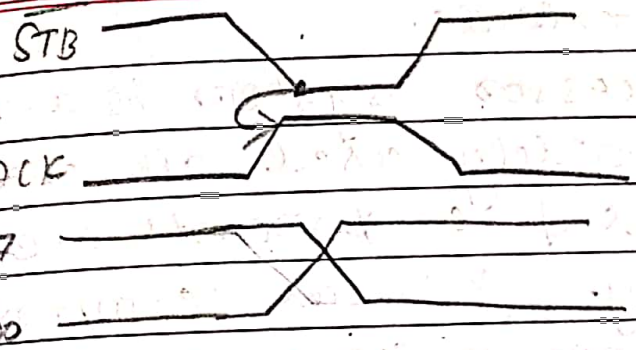


fig: Single Handshake I/O

iv) Double Handshake I/O;

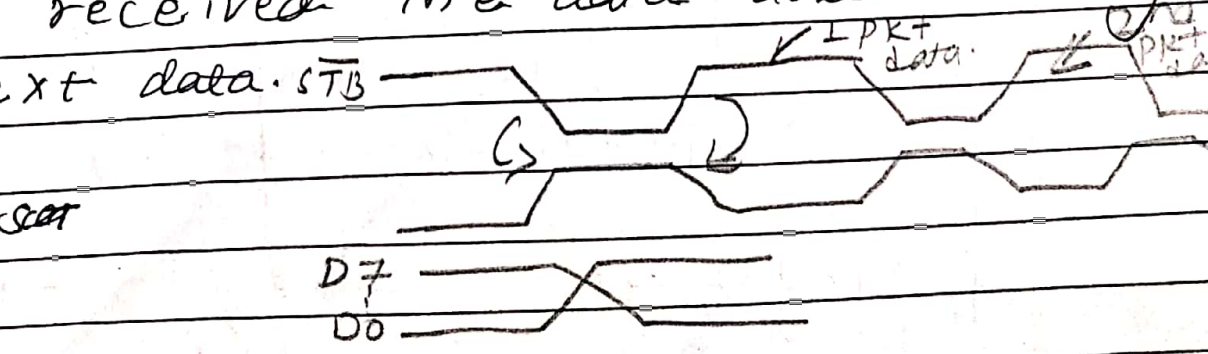
→ For data transfer where maximum coordination is required betn the sender & receiver, a double handshake is used.

→ The sending of peripheral device asserts the $\overline{STB}$ line low to indicate the receiving device whether it is ready or not for data reception.

→ The receiving system raises its ACK line high to indicate that it is ready.

→ The peripheral device sends a byte of data and raises its STB line high to indicate valid data is present.

→ When the receiving system reads the data its ACK line goes low to indicate that it has received the data and is ready for next data.



Bus Interface standards.

→ A microprocessor system is a bus oriented system widely used all over the world. Peripherals & components are manufactured by various companies. Therefore a common understanding must exist to ensure that there is a compatibility among different equipments. This is known as standard.

In the field of computer & electronics, these standard are generally defined by organizations such as IEEE (Institute of Electrical & Electronics Engineer)

RS-232 C.

→ Mostly used for serial communication between microcomputers & peripheral device.

→ The RS-232 C works on a negative logic. The standard specifies that "logic 1" is a voltage between -3 & -15V and "logic 0" is voltage between +3V & +15V.

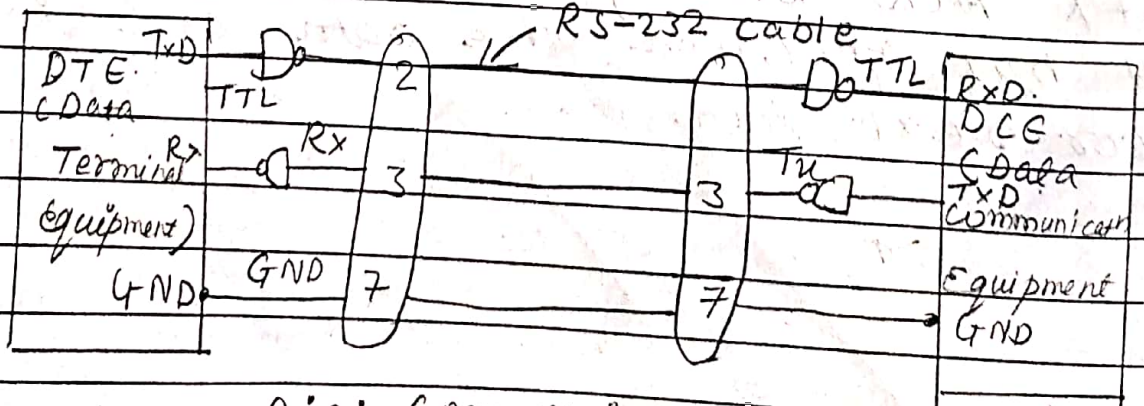


fig: Communication of DTE & DCE through RS-232 interface.

→ One of the most common uses of RS232C is the connection of peripheral terminals to computers..

→ MODEM and other devices used to send serial data are called data communication Equipment.

→ The computers that send or receive data are called data Terminal Equipment.

→ The RS-232 uses 25 pins (DB-25P) or 9 pins (DE-9P) standard where the 9 pins standard doesnot use all signal

→ For establishing the link between DTE & DCE, TTL to RS232C & RS-232C to TTL conversion is required

The property of RS232C is that it can transfer data reliably for 50ft. and with a maximum rate of 20K baud.

If longer lines are used, the transmission rate has to be reduced.

→ For higher rate of transfer & for longer distances we have to use other standards.

Standard	Speed	Distance	Voltage	Noise Immunity
• RS-232C	20K baud	50 ft	±15V	2V
• RS-422A	10 M baud	→40ft -	± 7 V	3.4V
	at 40 feet	4000ft		
	100K baud			
	at 4000ft.			
• RS - 423A	100K baud for	30ft - 4000ft	± 12V	1.8V.
	30ft &			
	1K baud			

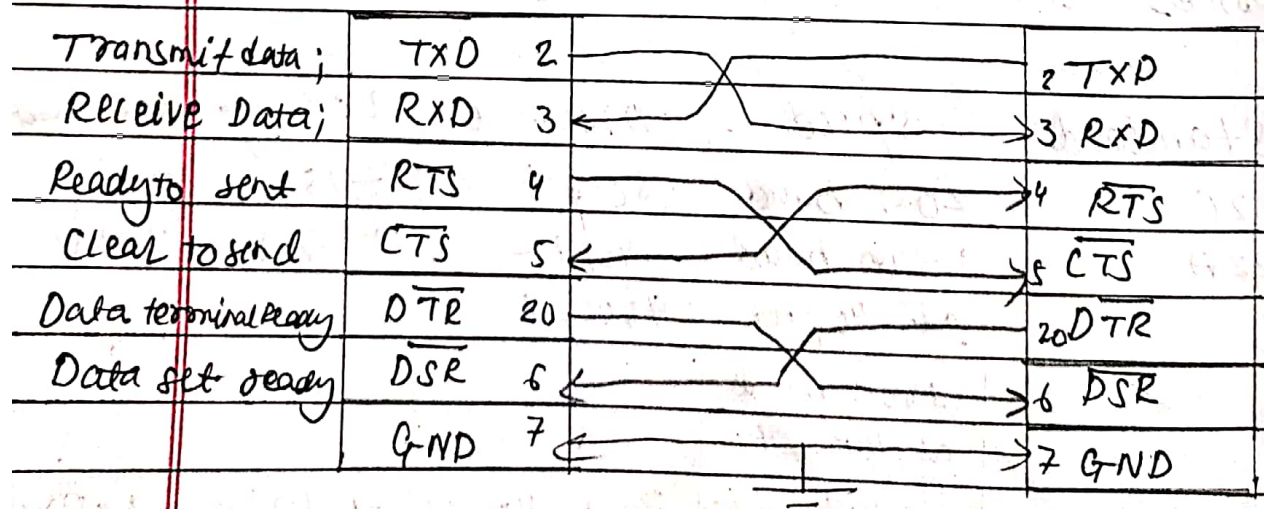
* DTR (Data Terminal Ready)
  -> After the terminal power is turned on, it asserts data terminal ready signal to tell the modem that it is ready.

* $\overline{DCD}$ (Data Carrier Detect); The modem will then assert its $\overline{DCD}$ signal to the terminal to indicate that it has established connection with the computer.

* TxD, RxD :- Receives the data through this line.
  L> sends the serial data to the modem.

* DSRD (Data Signal Rate Detect)
  -> It is used for switching to different baud rate.

Q) How do you connect two computers without using MODEM?



-> Most common connector is 9 pin RS232
-> They are straight through serial cable.

→ They are crossed cable used for data transfer
→ In remote locations where there were less availability of MODEMS this technique was used which was very cheap comparing to the case of MODEMS.

RTL (Register Transfer Language)
→ The symbolic notation used to describe the micro operations transfers among registers is called Register Transfer language. It is one of the ~~same~~ forms of hardware Description language (HDL)
→ RTL is a convenient tool for describing the internal organization of digital computers in precise manner.
→ It can also be used to facilitate the design process of digital systems.

Example: Let us consider the execution of instruction code MOV A, B.

Fetch cycle;
(i) The program counter (PC) contains the address of the instruction to be executed. Hence, it contains the address of memory location where the instruction MOV A, B resides.
In the first operation of fetch cycle, the contents of PC will be transferred to the (Memory Address Register (MAR)

The MAR then uses the address bus to transmit its contents. Let T_1 indicate the period of first operation.

T_1 ; MAR ← PC

ii) When the control unit issues the memory read signal, the contents of the memory location specified by MAR will be transferred to memory 'Buffer Register (MBR)'
Let T2 is the time period for this operation

T2 ; MBR ← [MAR]

iii) Finally the contents of MBR will be transmitted to the Instruction register (IR) and then the program counter (PC) will get incremented.
Let T_3 be the time period for this operation
Then

T_3 : IR ← MBR
PC ← PC + 1.

* Execute cycle
→ After the fetch cycle is completed, the execution starts. The execute cycle steps can be described as;

i) At the start of execution cycle, the instruction Register (IR) consists of instruction code for MOV A,B The address field of instruction specifies the address of the memory location A & B.

The first step is to obtain the data from the location B. For this, the address field of IR indicating the address of memory location will be transferred to address bus through MAR. Let 'T1' be the time taken;

T1 ; MAR ← (IR(Address of B))

ii) When control unit issues a memory read signal, the contents of Register B will be written to MBR. Let T2 be the time taken for this operation.

T2 ; MBR ← (contents of B

iii) Now, we need the memory location of A. For this the address field of IR indicating the address of A will be transferred to MR in time T3

T3 ; MAR ← (IR(Address of A))

iv) When the control unit issues the memory Read signal, the contents of MBR will be written to the memory location indicated by the contents of MAR in time T4.

T4 ; A ← MBR or T4 ; [MAR] ← MBR

Note; [MAR] = A.

Q. Write a program in 8086 to find the factorial of a number given by user.

```
-> dosseg
  • model small
  • stack 100h
  • data
      num db 5
      result db ?
  • code
    main proc
        mov ax @data
        mov ds, ax
          mov ax, num ; [ax]=5
          mov cx, num ; [cx]=5 ← counter
          dec cx ; [cx]=4
    loop: dec num ; [num]=4.
          mul num ; ax = 5x4x3x2x1
          dec cx
          JNZ loop
          mov result, ax ; [result]=—
          mov ax, 4c00h
          int 21h
    main endp
        end main.
```

5!
5x4x3x2x1
counter = 5
5x4x3x2x1

6 7 8 9 92

Q) Write a program in 8085 for the following type of addition.

$1^2 + 2^2 + 3^2 + \ldots + 9^2$

```
-> MVI A, 00 ;  (A) = 00
-> MVI B, 09H ; B = 09
   MOV C, B ;   (C) = 09
loop: ADD B ;   A ← 00 + 09 ← 09 ..... (81)
   DCR C ;      C = 08
   JNZ loop
   DCR B ; [B] = 08
   JNZ loop SKip
   STA D050
   RST 5 / HLT
```

Q) Write a program in 8085 to find the sum of even numbers from any 7 - input datas.

=> Soln:

```
MVI C, 07H : [C] = 07
MVI B, 00H : [B] = 00
LXI H, D000 ; H [D0] 00 L
   MOV A, M ; [A] = 01
```

i/p D000 : 01
D001 : 02
D002 : 03
D003 : 04
D004 : 05
D005 : 06
D006 : 07.

iii) If A > (Reg/M); CY->0 * [CMP] -> subtract
Z->0 (i) if A < (Reg/M) ; CY -> 1/2=0
ii) if A = (Reg/M) ; CY->0
Z->1

* WAP to find the smallest number among 3 numbers stored from memory location D000 - D002 & store the smallest number in memory location D050H.

Soln

i/p: D000 ; 02
D001 : 04
D002 : 01

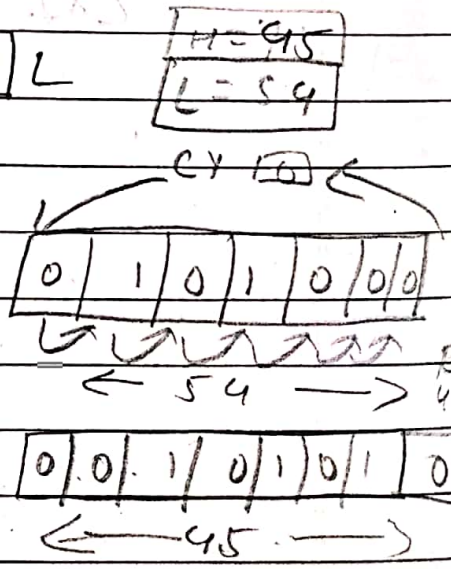
```
LXI H, D000; H [D0|00] L
MOV A, M ; [A] = 02
MVI C, 03 ; [C] = 03
loop2 : INX H ; H [D0|01]
CMP M
JC loop1 (Jump on carry)
-> MOV A, M
loop1 : DCR C ; C = [C-1]
JNZ loop2
STA D050 ; [D050] <- 01
RST 5
```

Q) WAP to find whether the given no is palindrome or not.

Soln

I/p : 4554

```
MVI C, 04 H
LXI H 4554 ; H [45|54] L
MOV A, L ; [A] = 54
loop: RRC
DCR C
JNZ loop
SUB H.
```



```
        JNZ loop
        SUB H
        JNZ SKIP
        LXI H, FFFF
        HLT / RST 5
SKIP :  LXI H, 0000
        HLT RSTS
```

Modes of operation in 8086 :

	Minimum Mode.	Maximum mode
i)	In minimum mode there can only be one processor in 8086.	i) In max. mode there can be multiple processors with 8086 like 8087 & 8089 co-processor.
ii)	MN/MX is 1 to indicate minimum mode (+5V)	ii) MN/MX is 0 to indicate max mode operation (ground)
iii)	ALE (Address latch enable) is given by 8086 as it is the only processor.	iii) ALE is given by 8288 (bus controller) as there are multiple processor.
iv)	Data Enable (DEN) & Data transmitted & Received (DT/R) is given by 8086.	iv) DT/R is given by 8288.
v)	Direct control signals M/IO, RD, WR is given by 8086 itself.	v) Here each processor generates status by signals. S_0, S_1, S_2.

vii) INTA is given by 8086 is response to interupts.	vi) INTA is given by bus controller.
ii) Circuit is simple	viii) Circuit is more complex
iii) Performance is slower	viii) Performance is much higher.
x) Less expensive to run.	ix) More expensive to run.

* Find the largest number among the block of data of size 0Ah and store the maximum value in location result in 8086.

soln

```
title Finding the max^m value.
dosseg
.model small
.stack 100h.
.data
        Array ab 12h, 03h, 17h, 50h, 49h,
                01h, 88h, 10h, 07, 70h
result :db?
.code
main proc
        mov ax, @data ; Initiative the data segment.
        mov ds,ax
```

```
mov si, offset Array
mov al, 00h
mov cx, 0Ah
cmp al, [si]
jnc loop1
    mov al, [si] -> al <- 88
loop1: ins si
    dec cx
    JNZ loop2
    mov result, al
    mov ax, 4c00h
    int 21h
    main endp
    end main.
```

Q) write a program to find the fibonacci series upto given nth terms.

0, 1, 1, 2, 3, 5, 8.

Acc 0050.

soln

```
MVI Au, 05 / LDA 0050
LXI H, 0050 H[00] 50] L
MOV B, A; [B] = [05] = of terms.
MVI D, 00 : [D] = 00
MVI C, 01 : [C] = 01
MVI A, 00 = [A] = 00

loop  MOV M, A; 0050; 00
      ADD C; A <- A+C; 00+01 = 01
      MOV D, C; [D] <- 01.
      MOV C, A; C[] <- 01.
```

Date ______
Page ______

```
MOV A,D ; [A] <- 01
    INX H
    DCR D ;     D = (n-1) terms
    JNZ loop.
```

Q) Write a program in 8086 to find the number of times the letter "O" exists in the string "Microprocessor" & store the count value.

=>
```
title: count letter
  dosseg
  .model small                      O -> 1 [ ]
  .stack 100h                       O -> 2 [ ]
  .data                             O -> 3 [ ]
     string db 'microprocessor','$'
     length dw $-string      <- length count hunxa
     total db
                                      SI -> m
  .code                              [I] -> 1
     main proc
        mov ax, @data
        mov ds, ax
        mov al, 00h
        mov SI, offset string.
        mov cx, length
  loop2: mov bh,[SI] ; m
        cmp bh, 'O'        ASCII value snga compare garchha
        JNZ loop.          loop janchha
        INC al ;           al : 01 -> 02
  loop  INC SI
        dec cx
        JNZ loop2
```

```
mov total ; al ; total : 03
mov ax , 4c00h
    int 21h
main endp
    end main.
```

Q. Registers BC pairs contains 2793H and register pair DE contains 3182H. Write an ALP in 8085 to add these 16-bit numbers and place the sum in memory location 2050H & 2051H.

Soln

```
LXI B, 2793 ; B [27] [93] C
LXI D, 3182H ; D [31] [82] E
MOV A,C : [A] = 93
ADD E ; A ← A+E
MOV L, A
MOV A, B ; [A] ← 27
ADC D
MOV H, A
SHLD 2050H
RST 5
```

Half word data carry

HL pair me load ho jaye memory bata.

H [BB] [AA]

* LHLD ; Load H & L Directly
Syntax ; LHLD 16-bit address
-> The contents of memory location pointed by the 16-bit address are copied the contents of next memory location in H

eg: Assume [2050] -> 44H
[2051] -> 77H.

LHLD 2050 ; [L] ← [2050]
[H] ← [2051].

H	L
77	44

mem ma load garne

HL pair bata

44	2050
77	2051

* SHLD ; Store H & L Register Direct
-> Assume ; H ← 01
L ← FF

Store the contents of H-L pair in memory location 2050 & 2051.

-> SHLD 2050 H.

H	L
01	FF

FF	2050
01	2051

RISC & CISC Architectures.

1) CISC Architecture

-> CISC stands for complex Instruction set computers machines based on these architectures have complex instructions.

-> most computers (personal) used today are based on this architecture.

Following are the characteristics of CISC machines.

- CISC machines have complex instructions which need a large cycle for execution.
- The transfer of data among the register is much faster than memory and processor.
- Pipelining is the proces of fetching one instruction when another instruction is executing in parallel. Due to complex instruction this feature cannot be heavily used in CISC architecture.
- CISC machines have large number of complex instructions based on multiple numbers of addressing modes.
- CISC machines processor doesnot consist of large number of registers due to large cost. so, these machines have to perform various memory read and write operation.
- CISC machines are preferable where the speed of processor is not the prime issue and

where general application are to be handled. Processors like 8085, 8086 etc are based on CISC processor.

* RISC Architecture:

-> RISC stands for Reduced instruction set computers. It focuses on small set of instructions which simplifies the hardware design and improves the processor performance.

-> The main features of RISC are;

- The number of instructions are minimized. i.e. less than 100
- The number of addressing modes are relatively low i.e. less than 3.
- RISC architecture promotes heavy pipelining.
- There are no micro-programs in RISC machines. Most instructions are directly executed by the hardware
- It supports high level language through appropriate section of instructions.
- Usually in scientific & research oriented tasks where the speed of processors is primary concern, RISC machines are used.
- RISC based system are Power PC from Apple, IBM super sparc from sun, 1860 from intel etc.

Difference between RISC and CISC.

	RISC	CISC
1)	Simple instructions taking one cycle.	1) Complex instructions taking multiple cycles.
2)	Most operations are registered to register with only LOAD and STORE operations.	2) Memory read & write operation from the main part for executing the instructions.
3.	Heavily pipelined for increasing performance.	3. Less pipelined.
4.	Multiple register sets	4. Single register set.
5.	Instructions have fixed format.	5. Instructions have variable format.
6.	Fewer number of instructions and addressing modes.	6. Large number of instructions and addressing modes.
7.	Instructions are executed by hardware.	7. Instn are interpreted by microprograms.
8.	It can be designed more quickly.	8. CISC processors have comparitively larger design cycle.

eg:-