Microprocessor: A microprocessor is a multipurpose, programmable, clock-driven, register base electronic device, that reads binary instruction from storage device called memory accept binary data and input and process data according to those instruction and provide result as output. Microprocessor application can be classified as:

Reprogrammable system:

- In this microprocessor is used for computing and data processing.
- Capable of handing large data, storage devices such as disks and CD Rom and peripherals devices such as printers. Eg microcomputer.

Embedded system:

- In this case microprocessor is a part of final product and is not available for reprogramming to end uses.
- Eg washing machine, traffic light controller, Automatic testing machine.

Evolution of microprocessor:

4004-introduced in 1971, first 4 bit up having memory addressing capability of 1 KB

- Consist of 16 pin with clock signal of 750 HZ

8008- introduced in 1972, 8 bit µp , 40 pin

8080- introduced in 1973 , 8 bit µp.

8085- introduce in 1976, 8 bit µp having addressing capability of 64kb,cosists of 40 pin with 3-6 MHZ clock signal.

8086 – introduce in 1978, 16 bit µP having addressing capability of 1 MB , consists of 40 pins with 5-10 MHZ clock signal.

8088- introduced in 1980, 8/16 bit µp with memory addressing capability of 1 MB, consists of 5-8 MHZ clock signal.

80286- introduced in 1982, 16 bit µp with memory addressing capacity of 16 MB, consists of 68 pin with 6-12.5 MHZ clock signal.

80386 – introduce in 1985 , 32 bit µ p with 4 GB memory addressing capability. Consists of 132 pins with 22 to 33 MHZ clock signal.

80486- introduced in 1989, 32 bit µp with 4 GB memory addressing capacity, consists of 168 pin with 26-100 MHZ clock signal.

Pentium:- introduced in 1993, 32 bit up with 4 GB of memory addressing capacity consists of 168 pins with 100 and 150 MHZ.

Pentium pro, Pentium II , and Pentium III, was developed each with 32 bit word length having 150-1000 MHZ clock signal.

Calculator verses computer:-

- Calculator are dedicated devices and can perform arithmetic and logical operations only. Also calculator had to be operated manually for each and every steps to be followed to solve a problem.

  The computer are programmable in the sense that all state steps needed to solve a problem are fed into main memory as program. Than the computer works on the previous fed program.
- calculator has very less memory capacity in comparison to the computer.
- A calculator has a very small size in comparison to the computer.

Microcomputer:-

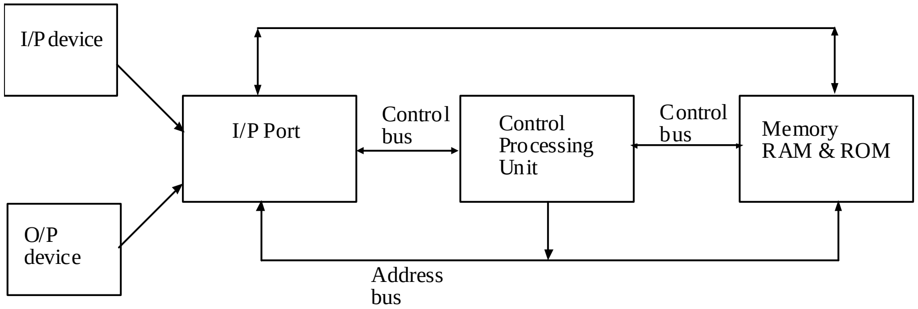
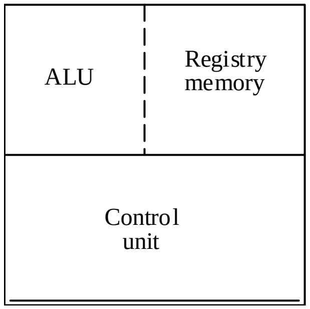


Fig. Block diagram of microcomputer

Microprocessor:



Microcomputer: A small computer that contains microprocessor is microcomputer. They range from 4-bit words that can address a few thousand bytes of memory to 32 bit words and can address billions of bytes of memory. In microcomputer cpu is single integrated circuit called microprocessor. The block diagram of microcomputer is shown above. The major parts are central processing unit, memory, and input output ports. Each of these part are connected with each other through address bus , data bus , and control bus.

Memory: This consists of a mixture of ram and Rom. It may also have magnetic floppy disk, magnetic hard disk or optical disk. It's function are :

1. Store the binary codes for the sequences of instruction and then write a program from that sequence of instruction for the computer.
2. Store the binary coded data with which the computer is going to work.

I/P port: The i/p section allows the computer to take in data from the outside world or send data to the outside world. Eg keyboard, video display terminals, printers, modems, etc. The physical devices used to interface the computer buses to external systems are called ports. Two ports are available i/p port example keyboard, mouse. O/p port example monitor , printer.

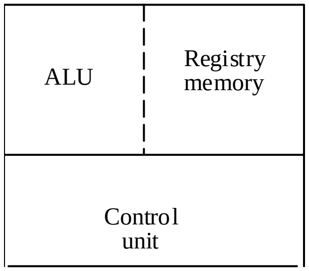
Central processing unit: The cpu control the operation of computer. Cpu fetches binary coded instruction from memory. Decode the instruction into a series of actions and carries out these actions in a sequence of steps. It also contains the instruction pointer register which hold the address of the next instruction or data item to be fetched from memory.

Address bus: The address bus consists of 16,20,24 or 32 parallel signal lines. On these lines the CPU sends out the address of the memory location that is to be written to or form.

Data bus: The data bus consist of 8, 16 or 32 parallel signal lines and are bidirectional that CPU can reads data in from memory and send data out to memory on these lines. ……device in a system will have ……out connected to the data bus but only one device at a time has its out enable.

Control bus: The control bus consists of 4-10 parallel signal lines. The CPU sends out signal on the control bus to enable the o/p of the address memory device. Control bus signal are memory read, write, i/p read, o/p write.

Microprocessor:



ALU: This area of microprocessor perform various function on data. The ALU performs arithmetic operation like addition subtraction and logical operation like And, OR, X-OR.

Register array: This area of microprocessor consists of various register identified by B,C, D,E, H, L. These register are used to temporary store the data during the execution of a program.

Control Unit: This area provides the timing and control signal to all the operations in the microcomputer. It contains the flow of data between the microprocessor memory and peripheral.

Stored Program concept: The task of entering and altering the programs for the ENIAC (electronic numerical integrator and computer) was extremely tedious. The programming concept could be faciliated if the program could represented in a form suitable for storing in memory along side the data. Than a computer could get it's instruction by reading them form the memory and a program could be set or altered by setting the values of a portion of memory . This approach is known stored program concept.

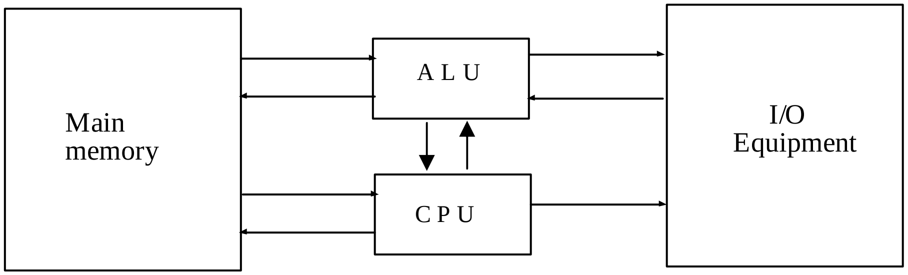


Fig: Von-Numann Architecture.

Main memory is used to store both data and instruction ALU is capable for performing Arithmetic and logical operation binary data. The program control unit(cpu) interprets the instruction in memory and causes them to be a execute. The input/output unit and helps inputting data and getting results.

The memory of Von-Neumann machine consists of thousand storage location called words of 40 binary digits(bits). Both data and instruction are stored in it.

The storage locations of control unit and ALU are called registers. The various registers of this model are MBR, MAR, IR, IBR, PC, AC.

Memory Buffer Register: It consist of a word to be stored in memory or is used to receives a memory or is used to receive a word from memory.

Memory address Register: It contain the address in memory of the world to be written from or read into the MBR.

IR(Instruction register): Contain the 8 bit upcode (operation code) instruction being executed.

IBR(instruction buffer register): It is used to temporarily hold the instruction from a word in memory.

PC (program counter): It contain address of next instruction to be fetched from memory.

Ac (Accumulator) and MQ(multiplier quotient): They are employed to temporarily hold operands and results of ALU operations.

Harvard Architecture:

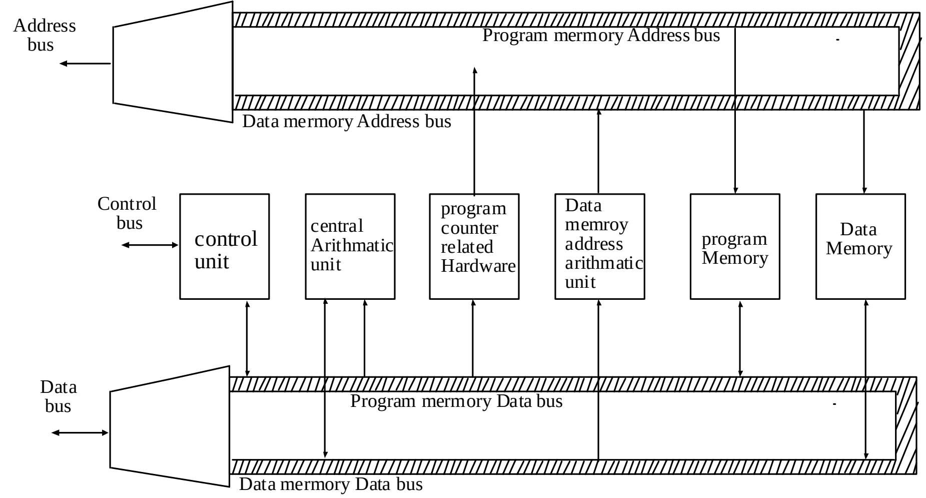


Fig. Block diagram of the Harvard architecture based µP

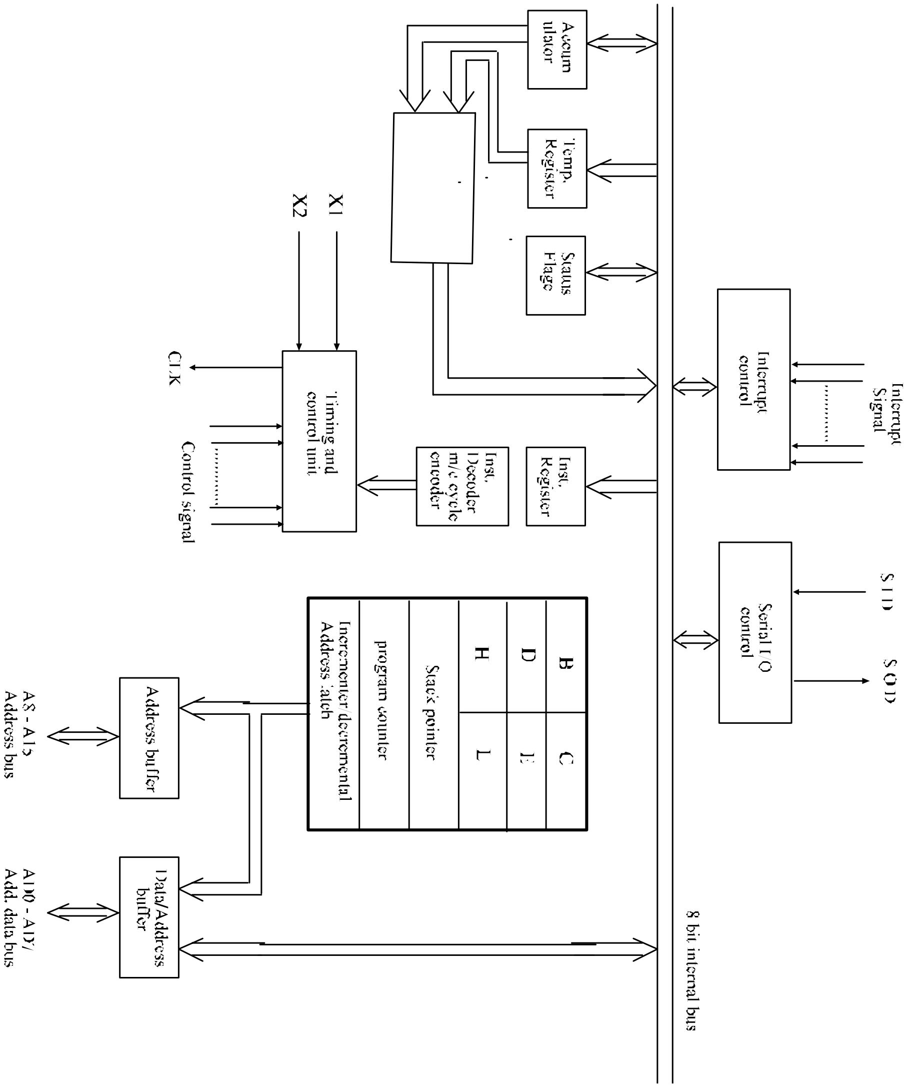
Harvard Architecture based computer consist so separate memory spaces for the programs (or instruction) and data. Each memory space has it's own address and data bus. Thus both instruction and data can fetch from memory concurrently.

From the figure it is seen that there are two data and two and address buses for the program and data memory spaces respectively. The program memory data bus and data memory data are multiplexed to form single data bus where as program memory Address and data memory address are multiplexed to form single address bus. Hence there are two blocks of ram chip. One for program memory and another for data memory space.

Data memory address arithmetic unit generates data memory address. The data memory address bus carries the memory address of data where as program memory address bus carries the memory address of the instruction.

Central arithmetic logic unit consists of the ALU, multiplier, Accumulator, etc. The program counter is used to address program memory. Pc always contains the address of next instruction to be fetched. Control unit control the sequence of operations to be executed. The data and control bus are bidirectional where as address bus is unidirectional.

Internal Architecture of 8085:



Intel 8085 is a 8 bit general purpose microprocessor capable of addressing upto 64kB of memory. It is a 40 pin ic package fabricated on a single LSI using an NMOS technology. It's clock speed is about 3 MHZ and uses a single 5v DC supply.

The internal structure of 8085 is shown in figure. It consists of three main section.

1. Register array.
2. Arithmetic and logic unit.
3. Timing and control unit.

Register array: The 8085 has both 8 bit and 16 bit registers. It has 8 addressable 8 bit registers and three 16 bit registers. These registers can be classified as a.. General purpose register b. Special purpose register.

a. **General purpose register:** The 8085 has 6 general purpose registers to store 8 bit data during program execution. B,C, D ,E, H, L are 8 bit registers and can be used singly or 16 bit register pairs. BC, DE, HL. When used in register pairs , the high order byte resides in the 1st register that is B when BC is as register pair and low order byte in second(ie c when BC is used). The register pair Hl besides it's possible use as to independent registers functions as a data pointer. It can hold memory addresses that are referred to in a number of instructions which use register indirect addressing.

b. **Special purpose register:**

Accumulator: It is a 8 bit register used in arithmetic logic load and store operations as well in input output instructions.

Flag Register: It is 8 bit register in which the bits carry significant information in the form of flags.

S- Sing Flag
Z- zero flag
AC- Auxiliary carry flag
D- Parity flag
CY- Carry flag

SZ	X	AC	X	P	X	CY

Temporary register: It is 8 bit register not accessible to the programmer while executing the instruction. The 8085 places the date into temporary register for a brief period.

Program counter: The program counter acts as a pointer to the next instruction to be executed and always contains 16 bit address of the memory location of the next instruction. The program counter is updated by the processor and points to the next instruction after the processor has fetched the instruction.

Stack pointer: The stack is an area of read write memory in which temporary information is stored in first in last out basis. The stack pointer holds the address of last byte written on to the stack.

Instruction register and decoder: These are not accessible to the programmer after fetching an instruction from memory the processor load it in the instruction register. This instruction is decoded by the decoder and the sequence of events are established for the execution of instruction.

Arithmetic and logic unit: The ALU performs the computing function. It includes the accumulator, the temporary register, arithmetic and logic circuits and the flag register. The temporary register is used to store hold data during an arithmetic/ logic operation. The result is stored in the accumulator and the flags are set or cleared according to the result of operation.

Timing and control unit: This unit synchronizes all the microprocessor operations with the clock. The clock is symmetrical square wave signal that drives the cpu. The control circuitary and all the operations are driven by the clock signals.

Interrupt: The 8085 has 5 interrupt signal that can be used to interrupt a program execution. The various interrupt signals are, INTR, RST 5.5, RST 6.5, RST 7.5, TRAP.

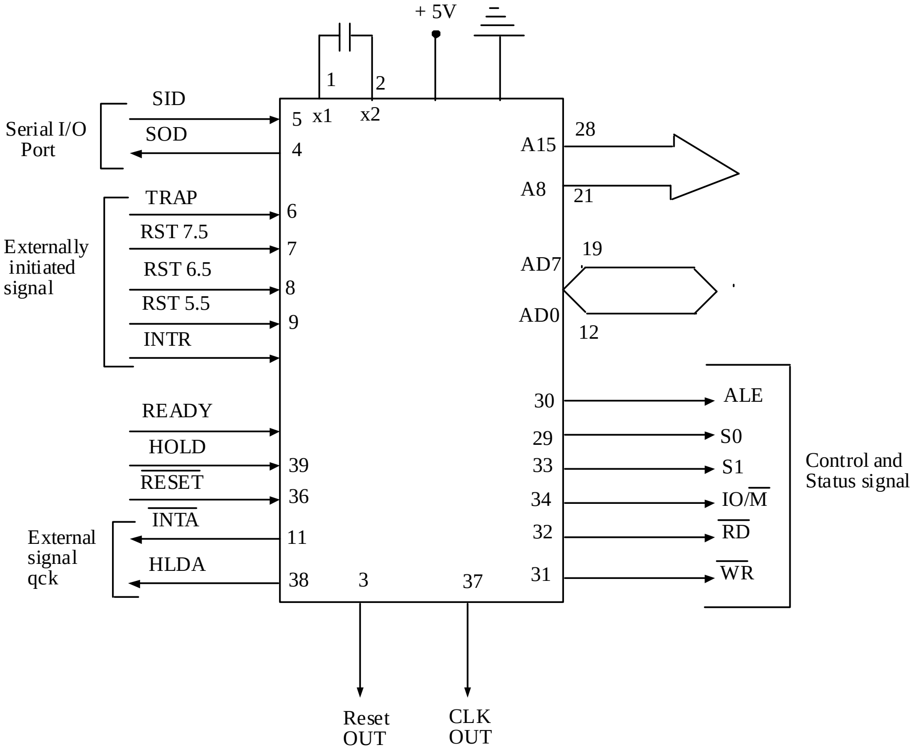
Serial I/p port: The 8085 has two signals to implement the serial transmission SID and SOD. This two signals are used to transmitting the data serially.

Data and address bus: The 8085 has 8 bit data bus and and hence 8 bit of data can be transmitted parallel from an to a microprocessor . The 8085 has 16 bit address bus as memory address are of 16 bit.

- The 8 bit significant bits of the address are transmitted by AD bus. The AD bus transmits the data and address at different moments. At a particular moment it transfers either data or address.
- The 8 most significant bit of the address are transmitted by address bus A bus.
- First of all the 16 bit data is transmitted by the microprocessor MSB on the A bus and 8 LSB on the

AD bus. Thus the effective 16 bit bus is used for 16 bit address. Then data is transfer via AD bus.

8085 Microprocessor unit pin details:



Intel 8085 contains 40 pins as shown in figure which has

- 8 unidirectional address pins (A_8 to A_{15})
- 8 bidirectional multiplexed address/data pins (AD_0 to AD_7)
- 11 control output pins
- 11 control input pins.
- Two power supply pins +5v and ground.
- $A_8 – A_{15}$ (pin 21 – 28) output
- Does the work of carrying the 8 MSB of the address data.

$AD_0 – AD_7$:

- [pins 12 – 19] I/O
- It carrys both data and address.
- It carries lower address bits as well as it can be used as data bus as it can be used as data bus.
- Carries data of 8 bit.

ALE (Address latch enable):

- Pin 30, output pin
- It goes high during first clock cycle of a machine cycle.
- When high, $AD_0 – AD_7$ is used as address bus.

IO/$\overline{M}$:

- pin 34, o/p pin
- Distinguishes whether the address is fir memory or I/O
- When high the operation is performed between I/O and µP
- When low the operation is performed between memory and µp.

S_0 , S_1 :

- Pin 29, 33, o/p pin
- These are status signals and indicates the type of operation performed.

S_0	S_1	Operation
0	0	HALT
0	1	READ
1	1	WRITE
1	1	FETCH (bring info. From the Memory to µP)

$\overline{RD}$:

- Pin 32, o/p pin
- Controls READ operation.
- When it goes low, the selected memory or I/O device is read.

$\overline{WR}$:

- Pin 31, o/p pin
- A low indicates a write operation being performed into the selected memory or I/P device.

READY:

- Pin 35, i/p pin
- It is used to sense whether a peripheral is ready to transfer data or not.
- If READY is high, the peripheral is ready.
- It is low, the µp waits till it goes high.

HOLD (when high):

- pin 39, i/p pin
- It indicates the another device is requesting the use of buses. Having received a HOLD request the µP stops the use of the buses as soon as the current instruction is completed. The processor regains the bus after the removal of the HOLD singnal.

HLDA:

- Pin 38, 0/p pin.
- A signal for HOLD ack.
- It indicates that the HOLD request has been received.
- After the removal of a HOLD request the HLDA goes low.

INTR:

- pin 10 , input pin
- It is an interrupt request signal.
- When it goes high the program counter does not increment its content. The µP suspects its normal sequence of instruction at hand it goes to the CALL instruction.

$\overline{\text{INTA}}$:

- pin 11, o/p
- The µP sends as interrupt ack after INTR is received.

RST 5.5, RST 6.5, RST 7.5 & TRAP:

- Pin 9,8 , 7 & 6, input pin
- These are interrupt signals.
- When interrupt is recognized, the next instruction is executed from a fixed location.
- RST 7.5,6.5,5.5 are ……..interrupt.

$\overline{\text{RESET IN}}$:

- pin 36, input pin.
- When signal on this pin is low, the µP is reset.

RESET OUT:

- pin 3, output pin.
- This signal indicates that µP is being reset.
- This signal can be used to reset other devices.

X_1, X_2 :

- Pin 1, 2 , i/p pin.

- These are terminals to connect to an external crystal oscillator which drives on internal circuitary of the µP to produce a suitable clock for the operation of µP.

CLK:

- pin 37, output pin.
- It is a clock output for user which can be used for other digital ICs
- Its frequency is same at which processor operates.

SID:

- pin 5, input pin.
- It is a data line for serial i/p.
- The SID signal can be used to i/p the SID pin to the most significant bit of the accumulator.

SOD:

- Pin 4, o/p pin.
- It is a data line for serial o/p.
- It can be used to o/p the most significant bit of the accumulator.

VCC:

- pin 40, input pin.
- +5v dc supply.

V_{ss} :

- Pin 20, input pin.
- Ground.
-

Microprocessor instruction:

Register transfer language(RTL): The internal network organization of a digital computer is defined by specifying.

- The set of register it contains and their functions.
- The sequence of micro operation performed on the binary information stored in the registers.
- The control that initiates the sequence of micro operation.

The symbolic notation used to describe the micro operation transfer among register is called register transfer language .

Fetch and execute cycle:

Eg: Mov

If the next instruction in the pc is Mov A,B then

Fetch cycle:

T1: MAR- pc

T2: MBR – M

T3: IR- (MBR)

PC- Pc+1

Execute cycle:

T1: MAR-(IR (address of B))

T2: MBR-(B)

T3: MAR-(IR(address of A))

T4: A- PC

In fetch cycle the last two operations are included within a single time unit T3 as the increment of pc takes place immediately after the content of MBR gets transferred into instruction registers. The representation, T1, T2, T3 represent time units and are of equal duration operations performed within this single unit is called micro operation.

8085 instruction Description:

Instruction: It is command given to a computer to perform specific operation on given data. The entire group of instructions called the instruction set determines what functions the micro processor can perform.

It has two parts

Opcode : Specifies what operation to be performed.

Operand: Specifies where to perform the operation.

opcode	operand

Instruction of a computer: The computer can be used to perform a specific task only by specifying the necessary steps needed to complete the task. The collection of such ordered steps forms a program of a computer. The program is thus collection form of such steps called instruction.

Categories of instruction:

1. Data Transfer instruction.
2. Arithmetic instruction.
3. Logical instruction.
4. Branch instruction.
5. Machine control instruction.

1. Data transfer instruction: This instruction copies data from one location called source to another location called destination without modifying the content of the source. The various types of data transfer instruction are :

a. Between register: eg MOV A, B , MOV C,A
b. Specific data byte two a register or memory location.
   Eg. MVI B,32H
   LDA ,2500H , LDAX B
c. Between a memory location and register.
   Eg. LXI B,2000H
   LXI- load immediate at location.
d. Between an input/output device and location.
   Eg. IN 02H , OUT PORT 1

2 **Arithmetic instruction:** These instruction perform arithmetic operations such as addition, subtraction, increment, and decrement.

i. **Addition**: Any 8 bit number or the content of register or the content of memory location can be added to the content of accumulator and the sum is stored in the accumulator. No two other 8 bit register can be added directly.
   Add B,
   ADD M
   ADI 06H

ii. **Subtraction**: Any 8 bit number or the content of register or can be subtracted from the content of accumulator and result is stored in the accumulator.
   eg Sub B, Sub M, SBI 06H

3. Increament/Decreament : The 8 bit contents of register or memory location can be increased or decreased by 1.

INR B
DCR B
INX H
DCX H

4. Logical instruction: The instruction of this group perform logical operation like AND, OR, Compare, rotate etc.

CMP –
RAL-
ANA-
ORA-

5. **Branch instruction:** List instruction alters the sequence of program execution either conditionally or unconditionally.

Jump: These instruction test for certain condition and altered the program sequence when the condition is met. Eg JC, JZ, JNC, JNZ

Call, return and restart: These instruction change a sequence of program either by calling a subroutine or returning from subroutine.

6. **Machine controlled operation or instruction:** This instruction control machine function such as Holt, Not Stop, End Operation. Eg: HLT, NOP, EI

Addressing modes: Each instruction requires certain data in which it has to operate. There are various technique to specify data for instructions. These technique are called addressing mode.

1. **Direct addressing mode:** In this mode of addressing the address of the operand(data) is given in the instruction it self. Eg. STA 2400H, IN 02H
2. **Immediate addressing mode**: In this addressing mode the operand is specified within the instruction itself . eg. AVI A, 06L , LXI H, 2500H
3. **Register addressing mode**: The operand are in general purpose register. The upcode specifies the address of the registers in addition to the operation to be performed. MOV A, B , MOV B, ADD B, SUB C
4. **Register indirect addressing mode**: In this mode of addressing the address of operand is specify by the register pair as a pointer for operand. Eg, LXI H, 2500H , MOV A,M , ADD M
5. **Implied addressing mode:** There are certain instruction which operate on the content of a accumulator such instruction do not require the address of operand. Eg, CMA, RLA, RAR, NOP, HLT

Instruction cycle: The necessary steps that the cpu carries out to fetch an instruction and necessary data from the memory and to execute it constitute and instruction cycle it is defined as the time required to complete the execution of an instruction.

An instruction cycle consists of fetch cycle and execute cycle. In fetch cycle CPU fetches upcode from the memory . The necessary steps which are carried out to fetch an upcode from memory constitute a fetch cycle. The necessary steps which are carried out to get data if any from the memory and to perform the specific operation specified in an instruction constitute and execute cycle . The total time required to execute an instruction given by IC = Fc+ Ec
The 8085 consists of 1-6 machine cycles or operations.

Fetch cycle: The first byte of an instruction is it's upcode . The program counter keeps the memory address of the next instruction to be executed in the beginning of fetch cycle the content of the program counter ,which is the address of the memory location where upcode is available , is send to the memory. The memory places the upcode on the data bus so as to transfer it to CPU. The entire process takes 3 clock cycle.

Execute cycle/Operation:- The upcode fetched from the memory goes to IR from the IR it goes to the decoder which decodes instruction. After the instruction is decoded execution begins.

- If the operand is in general purpose register ,execution is performed immediately. I,e in one clock cycle.
- If an instruction is contains data or operand address , then CPU has to perform some read operations to get the desired data.
- In some instruction write operation is performed. In write cycle data are sent from the CPU to the memory of an o/p device.
- In some cases execute cycle may involve one or more read or write cycle or both.

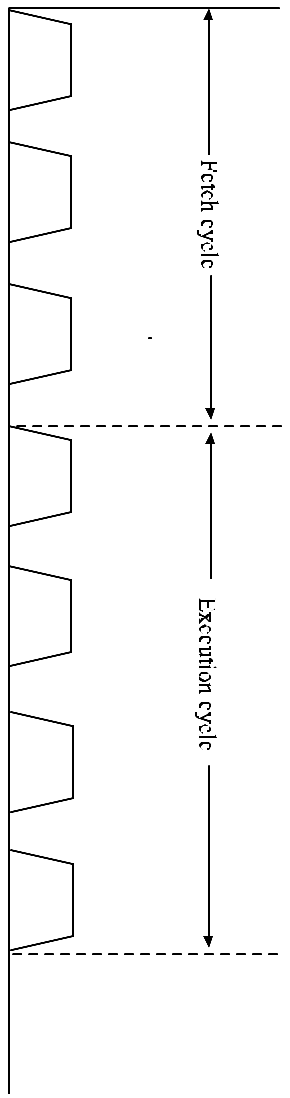


Fig. inst. Cycle showing FC, EC, IC

Fig. inst. Cycle showing FC, EC, IC

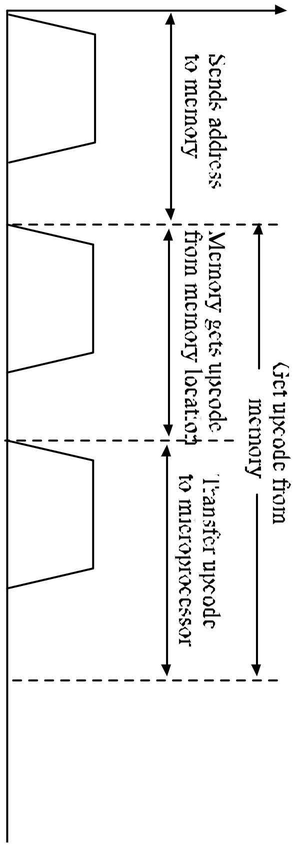

Sends address to memory
Memory gets upcode from memory location
Transfer upcode to microprocessor
Get upcode from memory

Fig. Fetch cycle

Fetch execution overlap:

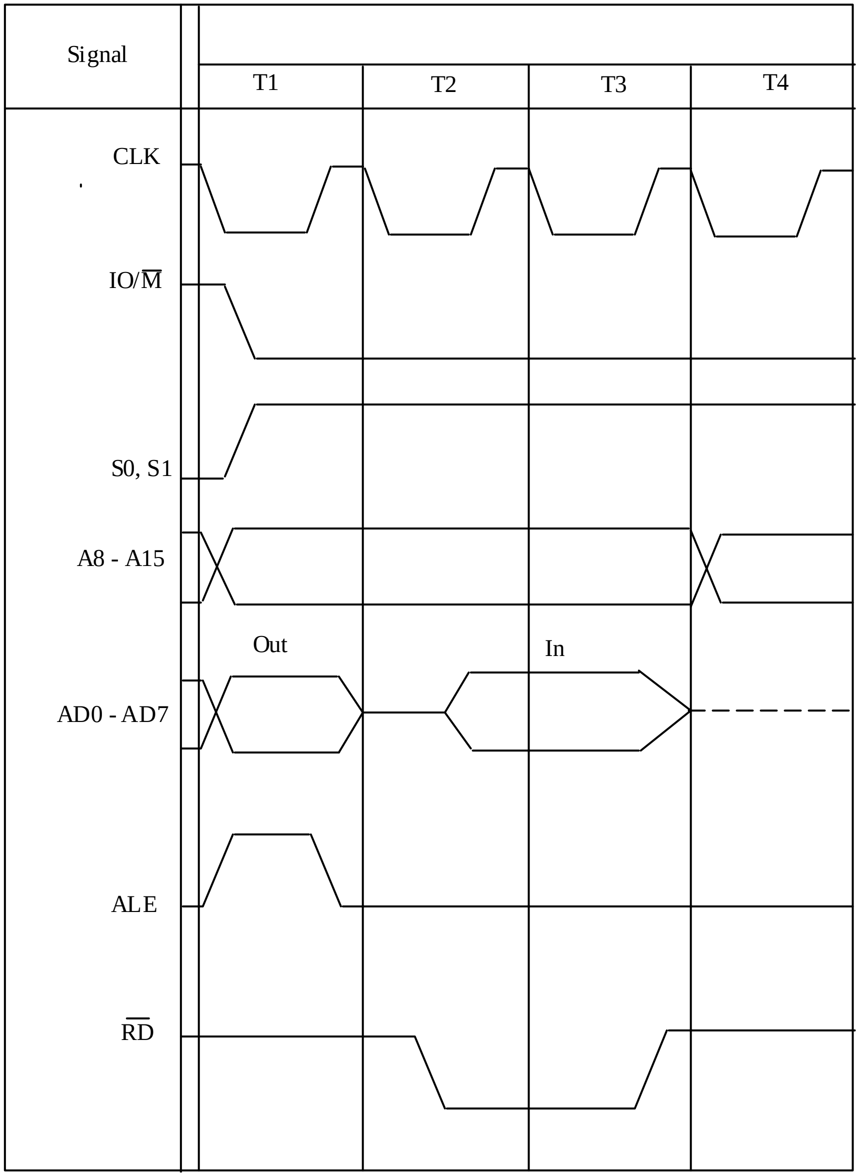
	t1	t2	t3
I1	F	E1	
I2		F2	E2

Machine cycle: It is defined as the time required to complete one operation of accessing memory , i/p, o/p or acknowledging and external request. This cycle may consists of 3 to 6 T states.

T-states: It is defined as one sub division of the operation performed in one clock period. These sub division are internal states synchronized with system clock and each T states precisely equal to one clock period.

Timing diagram: The necessary steps which are carried in a machine cycle can represented graphically. Such graphical representation is called timing diagram.

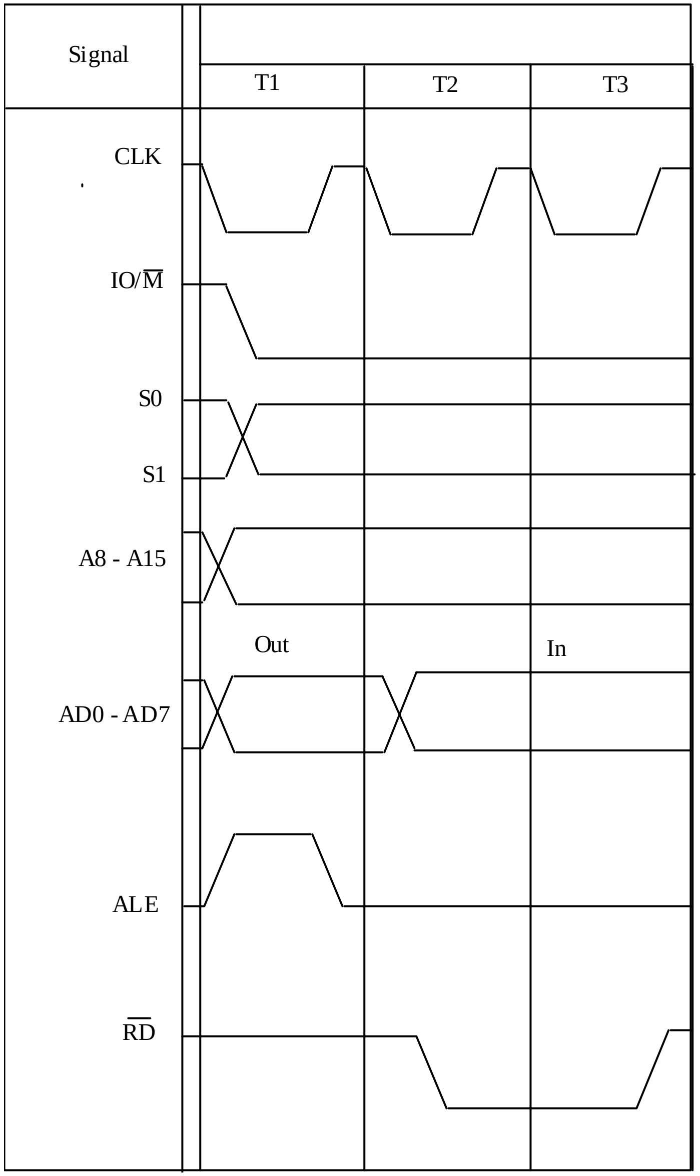
Timing Diagram:



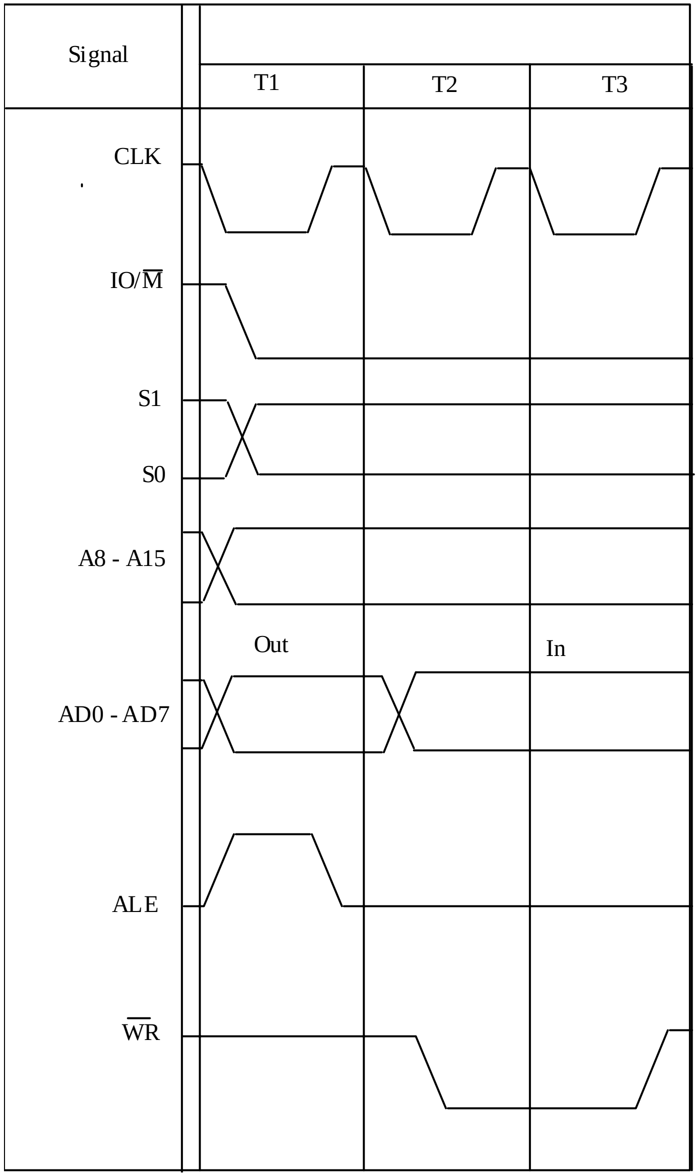
- The 8085 puts a low in the IO/$\overline{M}$ lines of the system indicating memory operation.
- The 8085 sets $S_o = 1$ and $S_1 = 1$ on the system bus indicating memory fetch operations.
- The 8085 places the pc high byte on the A8 to A15 lines and the PC low bytes on the AD0- AD7 lines of the system bus. The 8085 also sets ALE signal to high as soon as ALE signal goes low the AD_0 to AD_7 lines are used as data lines for reading the upcode.
- At the beginning of T2 the 8085 puts the $\overline{RD}$ lines to low indicating a read operations. After some time the 8085 loads the upcode into the instruction registers.
- During T4 clock period the 8085 decodes the instructions.

Timing diagram for memory read cycle:

I/O read:



I/O write:



General Control signal:

IO/$\overline{M}$	$\overline{RD}$	$\overline{WR}$	Signals generated
0	0	1	MEMR
0	1	0	MEMW
1	0	1	IOR
1	1	0	IOW

MOV A, B = 1MC – opcode fetch – 4 T state
MVI A, 32H = 2MC – opcode fetch, 1 memory read – 7 T states
STA 2050H- 4 MC- opcode fetch, 2 memory read , 1 memory write.
MOV A, M- 2 MC- 1 opcode fetch, 1 memroy write.
IN 84H – 3 MC – opcode fetch, 1 memory read, i/o read.
OUT 02H – 3 MC – opcode fetch, 1 memory read, 1 i/o write.
JMP 2050 – 3 MC – opcode fetch, 2 memory read. (when condition satisfied)

- 2 MC- opcode fetch, 1 memory read. (When condition unsatisfied)

ADI 12H- 2MC- opcode feth, 1 memory read.

Assembly language Programming: Among three different programming language of the computer , assembly language programming represents the programming level in between the machine language and the high level language. Since machine language program consists of either binary or hexadecimal opcodes, programming microprocessor with it is difficult because one must deal with numbers programs in assembly and high level languages are represented by instructions that use English language type statement. Thus it is more convenient to write programs in assembly language then in machine language. However the microprocessor can only understands the binary numbers and hence a translator is used to convert assembly or high level program into binary machine language so that microprocessor can execute the program.

Linker: A large program is generally divided into smaller programs known as modules. It is easier to develop to test and debug smaller program.

A linker is a program which links smaller program together to form a large program while developing a program , subroutines which are stored in the library file are frequently used in the program. The linker links these subroutines with the main program. It converts object codes into executable form.

Assembler: It is a program that translates assembly language mnemonics or source code into binary code or object code. This translation requires that the source program be written strictly according to the specified syntax of the assembler.

Assembly language format:

Label	**opcode**	**operand**	**comments**
jump	MOV	A, B	

A typical assembly language format is divided into four parts called fields. They are level, opcode, operand and comments.

- The assembler statements have free field format which means that no of blanks can be left between the fields.
- Comments are optional but help for good documentation.
- A level for and instruction is also optional but it's use greatly facilitates specifying the jump location.

eg. Label opcode operand comments
Start: LXI sp, 20FFH ; initialize the stack pointer.

These fields are separated by delimeters . Typical delimeters use are

Space- between each field
Comma- between two operand
Colon- After label
Semicolon- Before comment

1.Write a program to perform the following :
a). Load the no: 1BH in D
b) Load the no. B5H in B
C). Increment the content of B by 1.
d). Decrement the content of D by 1.
e). Subtract the content of D from the content of B.
f). Display the result at OUT port 1.

```
MVI D,1BH
MVI B,B5H
INR B
DCR D
MOV A, B
SUB D
OUT PORT 1
HLT
```

2 Wap to load the data byte in the register C. Mask the high- order bits (D7-D4) and display the low order bits (D3-D0) at outport. Exclusive-OR the result with 57H and display at OUT PORT2.

```
Solution:  MVI C, A8H
           MOV A, C
           ANI OFH
           OUT PORT1
           XRI 57H
           OUT PORT 2
            HLT
```

3. Wap to load the byte 8EH in register D and F7H in register E. Mask the higher order bits (D7 – D4) from both the data bytes , EX-OR the low order bit (D3-D0) and display the answer.

```
Solution:           MVI D, 8EH
                    MOI A,  D
                    ANI
                    MVI D, 8EH
                    MVI E, F7H
                    MOV A, D
                    ANI OFH
                    MOV D, A
                    MOV A, E
                    ANI OFH
                    XRA D
                    OUT PORT 1
                    HLT
```

4. Write a program to load two unsigned nos in register B and C respectively . Subtract c from B. If the result in 2's complement convert the result in absolute magnitude and display it port 1. Otherwise display the result.

```
Solution:        MVI B, byte 1
                 MVI C, byte 2
                 MOV  A, B
                 SUB C
                  JNC           label 1
                  CMA
                   ADI  01H
   Label 1        OUT PORT 1
                   HLT
```

5. Write an ALP to do the following:

a) Load A with byte 1.

b) Load B with byte 2.

c) Compare the equality of the contents of A and B

d) If two nos . are equal , display 01 otherwise display 00H at port 1.

```
Solution:         MVI A, byte 1
                  MVI B, byte 2
                  SUB B
                  JNZ   loop
                  MVI A, 01H
                  OUT PORT 1
                  HLT
  Loop            MVI A, 00H
                  OUT PORT
                   HLT
```

6. The following block of data is stored in memory location from CO55 to C05A H. Transfer the entire block of data to the locations C080 to C085 H in reverse order.

Data: 22, A5, B2, 99, 7F, 37

```
Solution:    LXI H, C055 H
             LXI D, C085 H
             MOV B, 06H
 Next         MVI A, M
             STAX D
             INX H
             DCX D
             DCR B
             JNZ   next
        HLT.
```

7. Write a program to find larger of two nos. 1st no in C001 and 2nd no in C002 and result in C003 H.

```
Solution:   LXI H, C001 H
            MOV  A, M
            INX H
            CMP M
             JNC     loop
             MOV  A, M
 Loop        STA  C003 H
             HLT
```

8. Write an ALP to find the smallest no in a data array. Data from location C000H to C005 H.

```
Solution:      LXI H, C000H
               MVI C, 06H
               MOV A, M
               DCR C
 Loop          INX H
               CMP M
                JC          loop1
               MOV A, M
 Loop1          DCR C
                JNZ      loop
                STA   C0C0 H
                 HLT.
```

9. Write an ALP to multiply two nos: eg 05 H × 08 H

```
Solution:      MVI   A, 00H
               MVI  B, 08H
               MVI  C, 05H
     Loop       ADD  B
                DCR  C
                JNZ         loop
                STA  C000H
                HLT
```

10. Write an ALP for the following addition. $1^2+2^2+3^2+4^2+5^2+6^2+7^2+8^2+9^2$

```
 Solution:   MVI A, 00H
             MVI B, 09H
    Loop 1   MOV C, B
    Loop 2   ADD  B
             DCR  C
             JNZ       loop2
             DCR  B
             JNZ       loop 1
             OUT PORT 1
             HLT
```

11. Write an ALP to count the no of 1 in the given string '10100110' and display the result at COCOH

```
Solution:     MVI A, A6 H
              MVI B, 00H
              MVI C, 08H
  Loop1       RAL
              JNC           loop2
              INR B
  Loop2       DCR C
              JNZ           loop 1
          MOV  A, B
          STA  COCOH
           HLT
```

12. The following datas are stored in memory location starting from C0B0 to C0B9 H . Take a test no. 48. Find out how many times the no 48 is repeated. Display the result at C0C0H .

DADA: 12, 23, 34, 45, 48, 56, 48, 67, 48, 89.

```
Solution:         LXI  H, C0B0 H
                   MVI B, 00H
                  MVI C, 0A H
 Loop 1            MOV  A, M
                  CPI  48 H
                  JNZ           loop2
                  INR  B
  Loop 2          INX H
                  DCR  C
                  JNZ           loop1
```

```
              MOV  A,B
              STA  COCO H
              HLT
```

13.8 bit multiplication , product is 16 bit . The multiplicand is loaded in the two consecutive memory locations 2501 and 2502 H . The multiplier is stored in 2053 H. Store the product in 2504 and 2505 H.

```
Solution:            LHLD  2501 H
                     XCHG
                      LDA   2503 H
                      LXI   H, 0000
                      MVI  C, 08
  Loop1              DAD  H
                      RAL
                      JNC        loop2
                      DAD  D
  Loop 2              DCR  C
                       JNZ        loop1
                      SHLD  2504 H
                       HLT
```

14. Write an ALP to divide two nos. The dividend is in C001 and divisor is in C002. Store the quotient in C0C0 H and remainder in C0C1.

```
 Solution:      LXI H, C001 H
              MOV A,M
              INX H
              MOV B,M
              MVI C, 00H
      Loop1         CMP B
              JC  Loop2
              INR C
              SUB B
              JNZ    Loop 1
 Loop2        STA  COC1 H
              MOV  A, C
              STA  COCO  H
              HLT
```

15. To arrange 54 , EB, 85, A8 & 99 in descending order. These numbers are stored in the memory location 2501 to 2505 H. The count = 05 is restored in 2500 H. Results are to be stored in 2601 to 2605 H.

Solution:

```
              LXI D, 2601 H
              LXI H, 2500 H
              MOV B, M
        Start CALL       Subroutine 1
              STAX D
              CALL       Subroutine  2
              INX  D
              DCR B
              JNZ    start
              HLT
Subroutine 1:
              LXI  H, 2500 H
              SUB  A
Loop1:        INX H
              CMP M
              JNC loop 2
              MOV  A, M
Loop2:        DCR  C
              JNZ   Loop 1
              RET
Subroutin2:
              LXI   H, 2500H
              MOV  C, M
```

```
Loop1       INX  H
            CMP  M
            JZ          Loop2
            DCR   C
            JNZ    Loop1
Loop2       MVI  A,  00H
            MOV  M, A
            RET
```

Note: Subroutine 1 gives the largest number of array. Subroutine 2 find the largest number and replace it by 00.

Counter and delay:

Timing delay using one register:

```
        MVI  C, FFH  ---------- 7 T state
 Loop   DCR  C     -------------- 4 T state
        JNZ  Loop  --------  10  or  7 T state
```

Consider a micro computer with 2 MHZ frequency

Clock period , $T = 1/f = ½ = 0.5$ µ sec

Delay for inst. Outside the loop T_o = No of T_s state × T

$= 7 \times 0.5$

$= 3.5$ µ sec

Delay for inst inside the loop, T_L = No of T state × T * (N_{10})

$= (14\times 0.5\times 10^{-6}\times 255)$

$= 1785$ µ sec

Now $T_{LA} = T_L - 3\times 0.5$

$= 1785 - 1.5$

$= 1.7835$ ms

$T_D = T_0 + T_L$

$= 3.5 + 1785$

$= 1788.5$

$= 1.7885$ ms

Time delay for register pair:

```
LXI  B, 2384 H ………………. 10 T state
Loop  DCX B …………………. 6 Tstate
MOV A, C …………………….. 4 T state
ORA B  ………………………… 4 T state
JNZ  loop ………………………. 10/7
```

Delay calculation:

T_0 = T state * T

$= 10 \times 0.5 \times 10^{-3}$

$= 109$ ms

T_L = No of T state * T * (N_{10})

$= 24 * 0.5 * 90.92$

Total Delay = $T_0 + T_L$

=

Time delay using a loop within a loop:

```
 MVI  B, 38 H  …….. 7 T
Loop2  MVI  C, FFH ………7 T
Loop 1 DCR  C……..4 T
 JNZ  loop1 ------10/7 T
 DCR B ------ 4 T
 JNZ  loop2 ------ 10/7 T
```

Delay calculation:

$T_o = 7 * 0.5 = 3.5$ µ sec

$T_{L1} = 14 * 0.5 * 255 – 3*0.5$

$T_{L2} = (T_{L1} + 21*0.5)_{N10} – 3$

Total dealy = $T_o + T_{L2}$

Q. Write a program to count continuously in hexadecimal from FFH to 00H in a system with a 0.5 µ s clock period. Use register C to set up a 1 ms delay between each count and display the number at one of the o/p ports.

```
      MVI  B, 00H
Next  DCR  B
      MVI  C, count
Delay DCR C
      JNZ  delay
      MOV A, B
      OUT port 1
      JMP  Next
```

Delay calculation:

$T_L =$ T state * T* (T_{10})
$= 14 \times 0.5 \times$ count
$= 7 \times$ count µ s
$T_o = 35 \times T$
$= 35 \times 0.5$
$= 17.5$ µ s

$T_D = T_L + T_o$
1 ms = $7 *$ count $* 10^{-6} + 17.5 * 10^{-6}$

Count = 140.35
= 8C H

Q. Write a program to generate a continuous square wave with the period 500 µ sec. Assume the system clock period is 325 µ sec and use bit D_o to output the square wave.

Solution:

```
        MVI  D, AA
ROTATE  MOV A, D
        RLC
        MOV  D, A
        ANI  01 H
        OUT PORT 1
        MVI  B, count
Delay   DCR  B
        JNZ   delay
        JMP   ROTATE
```

Delay calculation:
$T_L = 14 * 325 * 10^{-9}$ * count or 14 * 325 * (count - 1) + 11 T-state * 325
$T_0 = 46 * 325 * 10^{-9}$
$T_D = T_L + T_0$
250 = $(52.4)_{10}$ = 34 H

Types of Assemblers:

1. One pass assembler
2. Two pass assembler

One pass assembler:

This assembler goes through the assembly language programs one and translate the assembly language program into machine language program .

This assembler has the problem of defining forward references this means that a jump instruction using an address that appears latter in the program must be defined by the program after the program is assembled

Two pass assembler:

This assembler scan the assembly language twice. In the 1st pass, the first memory location is determined from the 'ORG' statement and the counter knows the location i,e counter is initialized . Then the assembler scans each instruction and records location in the address column. The address location counter keeps tracks of the byte in the program. The assembler also generates a symbol table in the 1st pass. When it comes across the label , it records the label and location.

In the second pass each instruction is examined and menominocs and label are replaced by the machine code.

```
 e.g  PORT 0              EQU         00H
      PORT 1              EQU         01H
                    ORG         2000H
      Start,        IN PORT  0
                OUT PORT 1
                JMP start
                END
```

When this program is assembled by one pass assembled the program is converted into it's machine code in the 1st pass only.

In case of two pass assembler the task is shown as below.

Pass 1:

Address	Machine code	Label	Opcode	Operand	Symbol table
2000		start	IN	PORT 0	PORT 0 00H PORT 1 01H Start 2000H
2002			OUT	PORT 1	
2004			JMP	start	

Pass 2:

```
      2000          D3 00
      2002          D3  01
      2004          C3  00,20
```

The assembler Directives:

Assembler directive are the instruction to the assembler concerning the program being assemble l they are also called pseudo instruction or pseudo upcode. They define the way according to which micro computer is directed to perform a specific task. These instruction are neither translated into machine code nor assigned any memory location in the object file. Few assembler directive are explained below.

ORG (Origin): The pseudo instruction lets the programmer place the program anywhere in the memory. Internally the assembler maintains a program counter type registor called address counter . This counter maintains the address of the next instruction or the data to be processed e.g

ORG 7000 H

```
        MVI   A, 02
```

The 8085 assembler will generate the following code for this statement

```
        7000          3E
        7001          02
```

EQU (equate): This pseudo instruction assigns a value in it's operand field to an address in it's label field. This allows the user to assign numeric value to a symbolic name. The user can then use the symbolic value in the program instead of it's numeric value.

```
                START   EQU 0200H
```

This assign the value 0200H to the label start.

```
                2000------Jump
                2001------00
                2002------02
```

PORT A EQU 40H: Here PORT A is assign value 40 H.

DB (Define byte):

This pseudo instruction is used to initialized a area byte by byte. Assembled bytes of data are stored in successive memory location until all values are stored.

```
      e.g  ORG  2000H
Data: DB  20H, 30H, 40H, 50H
```

DW(Define word): This pseudo instruction is used to assign a 16 bit value to two memory location. e.g ORG 7000H, START DW 4AC2 H

$$\begin{pmatrix}7000 & C2\\ 7001 & 4A\end{pmatrix}$$

This will assign C2 to location 700H and 4A to 700H and 4A to 7001 H location. It is assumed that assembler will assign low byte first (C2) and high byte (4A).

END: This pseudo instruction is used to define the end of assembly. The HLT instruction suggest the end of a program but that does not mean that it is the end of the assembly.

Bus structure and memory devices:

1. Data bus
2. Address bus
3. Control bus.

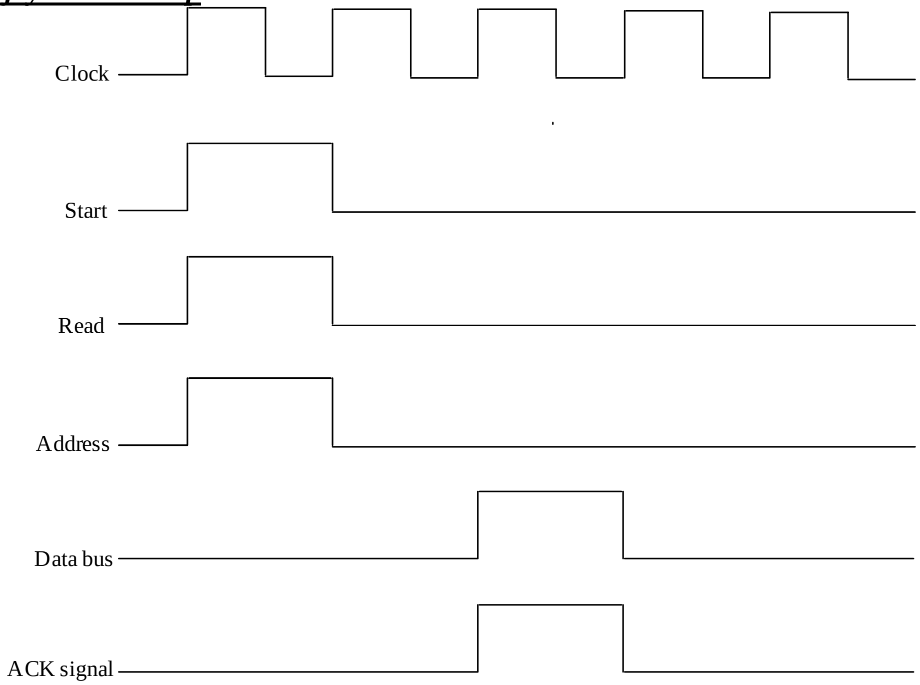
Data bus: Data bus provides a path for monitoring data between the system modules. The data bus consist of number of separate line 8, 16, 32, or 64. The number of lines is referred as a width of data bus. Since each line can carry only one bit at a time. The number of lines determine how many bits can be transmitted at a time. the width of the bus is key factor in determining the overall system performance.

Address bus: The address bus are used designate the source or destination of the data on data bus . Examples: if the CPU requires to read a word (8, 16 or 32) bits of data from memory, it puts the address of the desire word on the address bus. The width of the address bus determines the maximum possible memory capacity of the system. The address bus is also used to address input output ports. The higher order bits are used to select a particular modules on the bus and the lower order bits select memory location or input / output port within the module.

Control bus: The control bus is a group of lines used to control the access to and the use of data and address bus since data and address buses are shared by all components of micro processor system control signal transmit command and timing information between the system module. The timing signal indicate the validity of data and address information where as command signal specify operation to be performed. Some of the control signals are :

- Memory read.
- Memory write.
- Input/ output write.
- Transfer acknowledgement.
- Bus request.
- Bus grant.
- Interrupt request.
- Interrupt ack.

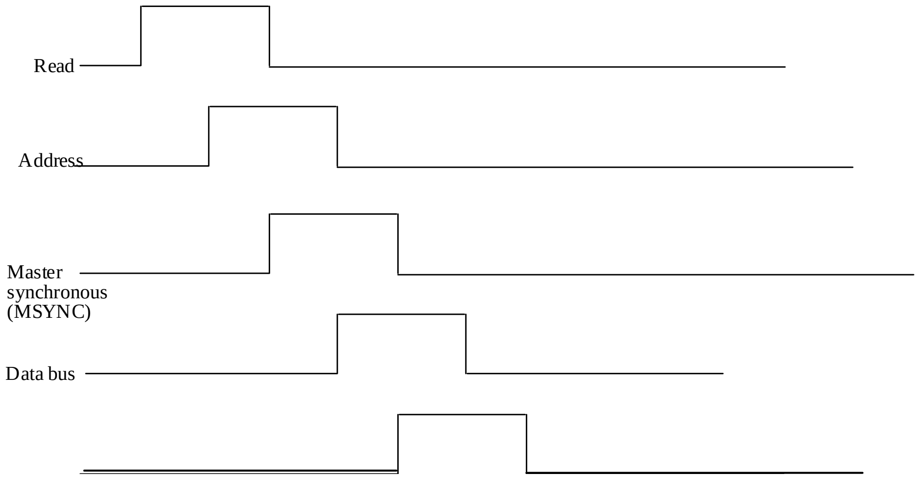
Synchronous Bus: In synchronous bus the occurance of the events on the bus is determined by a clock. The clock transmits a regular sequences of zeroes and one of equal duration. A single zero-one or 1-0 transmission is called clock cycle and defines a time slot. All other devices on the bus can read the clock live and all events starts at the beginning of clock cycle.



The above diagram describes the following steps.

- The CPU issues a START signal to indicate the presence of address of control information on bus.
- Then it issues memory read signal and places the memory address on the address bus.
- The addressed memory module recognizes the address and often a delay of one clock cycle it places the data signals on the buses.

Asynchronous bus: In an asynchronous bus the timing is maintained in a such a way that occurrence of one events on the bus follows and depends on the occurance of the previous events.



- The CPU places the memory read and the address signals on the bus.
- After allowing for these two signals to stabilized, it issues master synchronous signal (M SYNC) to indicate the presence of valid address of control signals on the bus.
- The addressed memory module responds with the data and the slave synchronous (SSYNC).

Memory:

A memory unit is a collection of storage cells together with associated circuits needed to transfer information in out of storage. The memory stores binary information in groups of bits called words. A memory is a group of 1's and 0's and may represent a number, an instruction code one or more alphanumeric characters or any other binary-coded information. A group of 8 bits is called a byte. The capacity of memories in commercial computers is usually stated as the total number of bytes that can be stored.

For 8 bit micro-computer system memory word and memory byte are the same. A memory word is identified by an address. The 8 bit microprocessor uses 16 bit address to access memory word. This provides a maximum of 2^{16} = 65536 memory addresses ranging from 0000H to FFFF H. Thus , the memory capacity for this Micro-computer system is 64K.

Memory can be classified as non-volatile memory or volatile memory. Non-volatile memory retains the stored data even when there is no power. On the other hand volatile memory losses its contents when the power is removed. Few examples are:

Non volatile: ROM, PROM, EPROM, Floppy disk, hard disk etc.
Volatile: RAM (static dynamic), CCD (Charge coupled devices etc).

In broad sense, a microcomputer's memory system can be divided into three groups;

i. Processor memory

ii. Primary memory
iii. Secondary memory.

Processor Memory:

Processor memory refers to a set of microprocessor registers. They are used to hold temporary results when a computation is in progress Although use of such registers enhances the execution speed, the cost involved in the approach forces a micro-computer designer to include only a few registers include the processor. In 8085 we have registers like A, B, C, D, E, H, L, SP, PC etc to store data temporarily.

Primary Memory:

It is the storage area where all programs are executed. The microprocessor can directly access only those items that are stored in the primary memory. Hence, all programs and data must be within the primary memory prior to execution. Usually, the size of the primary memory is much larger than that of processor memory and its operating speed is much slower than processor's registers.

Primary memories can be divided into two main groups:

1. Read only memory (ROM)
2. Random Access memory. (RAM)

Read only memory (ROM):

ROM is a non volatile memory and can be read only. It is used to store data and programs that are not to be altered. Among other things ROM is needed for storing an initial program called boot strap loader. The bootstrap loader is a program whose function is to start the computer software operating when power is turned on. Since RAM is volatile, its contents are destroyed when power is turned off. The contents of ROM remain unaltered after power is turned off and on again. The startup of a computer consists of turning the power on and starting the execution of an initial program. Thus when power is turned on, the hardware of the computer sets the program counter to the first address of the bootstrap loader. The bootstrap program loads a portion of the operation of the operating system from disk to main memory and control is then transferred to the operating system, which prepares the computer for general use.

The principle on which ROM is based can be explained with the help of figure below. This ROM is constructed with diodes.

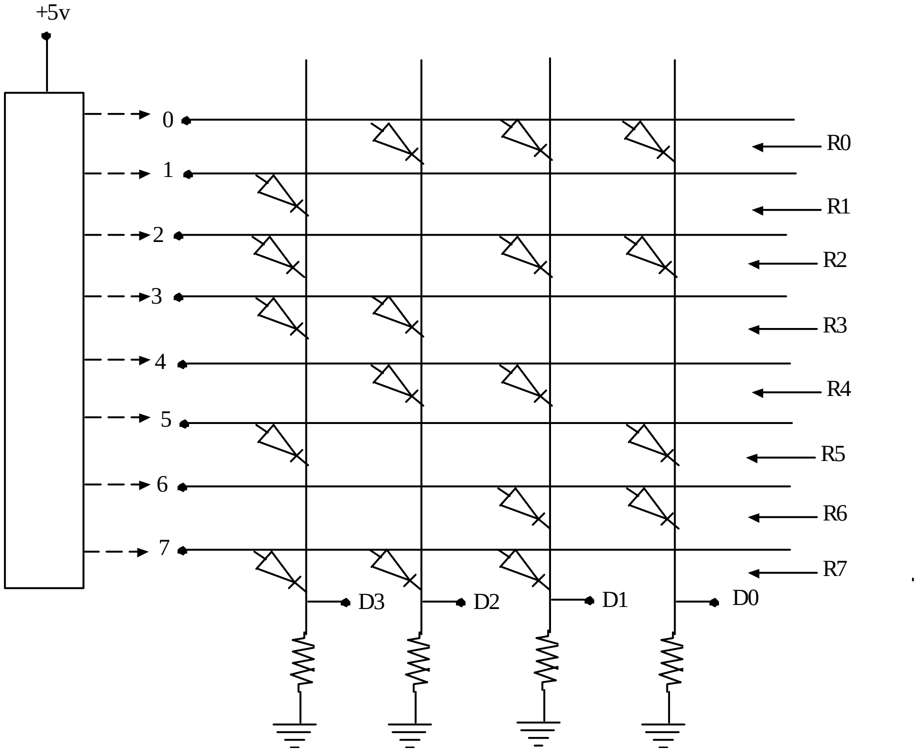


Fig. simple diode ROM

Here, each horizontal row is a register or memory location. The R_o register contains three diodes, the R_1 register has one diode, and so on. The output of the ROM is the word.

$$D = D_3D_2D_1D_0$$

In switch position 0, a high voltage turns on the diodes in R_0 register; all other diodes are off. This means that a high output appears at D_2, $D_{1,}$ D_0. Therefore the word stored at memory location 0 is

$$D = 0111$$

When the switch is moved to position 1. The diode in the register R1 conducts; all other diodes are off. Thus the word stored at memory location 1 is

$$D = 1000$$

Similarly, datas in other registers are stored with integrated circuits manufacturer stores the word at the time of fabrication. The words are permanently stored once the diodes are wired in place. In fabrication ROMs the manufacturer may use bipolar transistors or MOSFETS. But the idea is still basically the same, the transistors or MOSFETs act like the diodes in the figure above. ROMs can be classified as below:

Masked ROM: In this ROM, a bit pattern is permanently stored. Memory manufactures are generally equipped to do this.

PROM (Programmable ROM): This memory has nichrome or polysilicon wires arranged in a matrix . These wires can be funcitionally viewed as diode or fuse. This memory can be programmed by the user with a special PROM programmer that selectively burns the fuses (applying high current) according to the bit patters to be stored. The process is known as 'burning the PROM', and the information stored is permanent.

EPROM (Erasable PROM): EPROM uses MOSFETS. Data is stored with PROM programmer. Later data can be erased with ultraviolet light. The light posses through a quarts windows in C package on the chip, where it releases stored charges. The effect is to wipe out the stored contents. In other word, the EPROM is ultraviolet-light-erasable and electrically programmable. Once the chip is programmed the window is covered with opaque tape to avoid accidental erasing.

EEPROM (Electrically erasable PROM): This is non-volatile like EPROM but does not require U-V rays to be erased. It can be completely erased or have certain byes changed, using electrical pulses. Writing to EEPROM is slower than writing to RAM, so it can not be used in high speed circuits.

FLASH MEMORY: This is a modified EEPROM. The difference is the erasure procedure. EEPROM can be erased at a register level, but flash memory must be erased either in its entirety or at the sector (block) level.

Random Access memory OR Read/Write memory: It is a volatile memory and is used in store programs and datas for immediate use of the processor. Data can be readily written into and read from a RAM at any selected address in any sequence. Two types of read/write memories are available:

- Static RAM
- Dynamic RAM

Static RAM: This memory is made up of flip flops; and stores the bits as voltage. Each memory cell requires six transistors. This memory is more expensive and power consuming than dynamic memory. It is called 'static' because the information doesn't need a constant update. These memories are commonly used for cache memory. This types of memory is very fast with access time is 15 to 30 nanoseconds but is physically bulky.

Dynamic RAM (DRAM) : Dynamic random-access memory is an improvement over the expensive and bulky SRAM. DRAM uses a different approach to store data. Information is stored as charges in a very small capacitor. If a charge exists in a capacitor, it's interpreted as 1. The absence of a charge will be interpreted as 0.

Because DRAM uses capacitors there is a chance of leakage of charge. Thus it needs to use a constant refresh signal to keep the information in the memory (every few millisecond). DRM technology allows several memory units, called cells to be packed at very high density. Therefore, these chips can hold very large amount of information. Most PCs today use DRAM. Access time for DRAM is 80 nanoseconds or more, slower than SRAM, and two or three times faster than ROM.

The memory can be classified even as: random access memory and sequential access memory.

The term random-access memory simply means a memory system in which any memory location can be accessed as easily as any other. Most of today's microprocessor memory system are random access. It includes RAM and ROMs both. Originally RAM, was used to refer to random access memory but now it is used to designate a read/write memory.

Sequential-access memory means a memory in which to access a memory location you must access every memory location between the presents memory location you are addressing and the memory location that you want to read from a or write to. One can't jump to the desired location Sequential access is mostly used for mass storage of data. For example, information recorded on a reel of magnetic tape is stored in sequential access form.

Some mass storage device such as floppy disk use a combination of random access and sequential access. Two major semiconductor technologies are used in integrated-circuit memory devices, which are used to build memory system for today's memory. There are bipolar and MOS (metal-oxide semiconductor) technology.

Bipolar memories: They are used with microprocessor systems. The advantage of bipolar memories is then very fast access time. They have a number of disadvantages when compared with MOs memories. They draw a great deal of power, and there are fewer memory bits for the same silicon chip size. The fabrication is much more complicated and thus are much more expensive than Mos memories. For these reasons bipolar memory is used only for applications which can afford its great cost.

MOS Memory: It is the most common microcomputer memory. The Mos memory cells are structurally simple and economical. The MOS cell needs an area of about 25% to

50% of the area required by the bipolar cell. Hence they can be more densely packed. Mos cell consume much less power than bipolar cells.

Today’s RAM cell are, in common of two types depending upon the storage elements. MOS technology is applied in both cases. These are

- Static RAM cell
- Dynamic RAM cell.

Static RAM Cell:

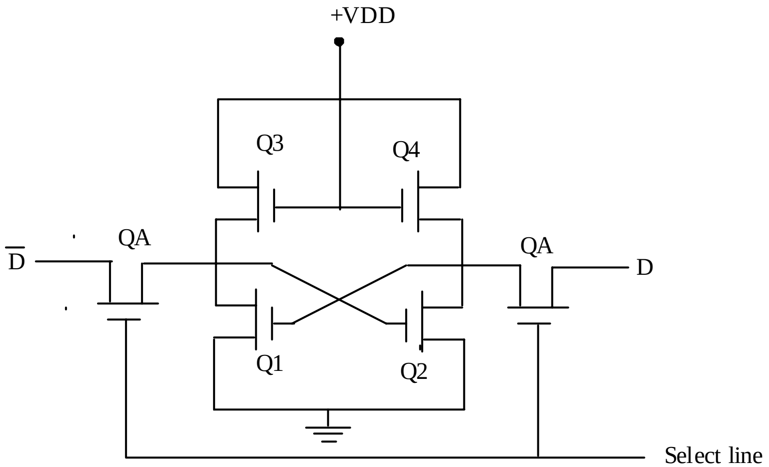


Fig. static RAM cell

- The figure above shows a NMOS static memory cell which consists of six NMOS transistors.
- The transistors Q_A, is called the access transistors are used to access the data during the read and write operation.
- The transistors Q3 and Q4 are pull up load transistors and their cross connection with Q2 and Q1 respectively forms a flip flop.
- The D and D’ are data lines. There is external driving circuitry which forces D and D’ to opposite values. When reading the bit from the cell the D and D’ serve as output lines, while writing a bit on the cell they serve as forced input.
- Before reading or writing on cell, the cell must be selected, and the select line is used for the purpose.

Working on Cell:

- The access transistor Q_A’s are turned on via the row select line (high).
- The data lines are forced into a state with D high and D’ low.
- The MOSFET Q1 will be turned on and Q2 off.
- When forcing signals are removed Q1 will continue to hold D’ low (i,e Q1 will be on) and will keep Q2 off and D output high.
- The forced state is thus self-sustaining and stable.

A similar stable state exists with D low and D’ high. For either cases, the stored data will be hold by the flip-flop until it is either changed by new forcing singles. Or until the power to the ckt is removed.

Reading from Cell:

- Q_A’s are turned on via select line (high).
- The stored data appears on D’ and D respectively from Q1 and Q2.

Dynamic RAM Cell:

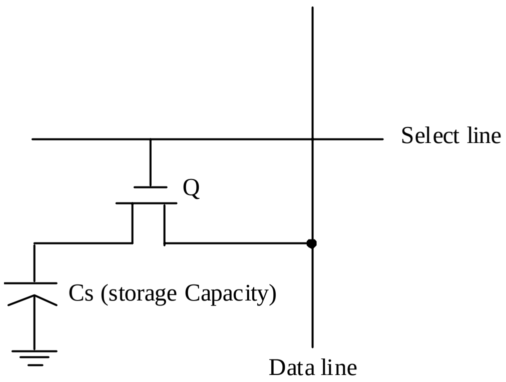


Fig. Dynamic RAM cell

The figure above shows a dynamic RAM cell which consists of one MOSFET and one capacitor.

- The capacitor is to store the bit.
- The transistor acts as a switch.

Write Operation:

- The data line is high (to store 1)
- The select line is high.
- The MOSFET is turned ON and the capacitor is charged.
- When data and selection line goes low the MOSFET open (turns off) and the capacitor retains its charge.

Read Operation:

- The select line is high.
- The data in the capacitor is then received at the data line.

The advantage of this types of cell are:

1. Very simple , allows very large memory arrays to be constructed on a chip at a low cost per bit.
2. Power consumption is low.

The disadvantage is :

- The storage capacitor cannot hold its charge over an extended period of time and losses the stored data bit unless its charge is refreshed periodically.
- This process of refreshing requires additional memory circuitry and complicates the operation of the dynamic RAM.

Interfacing Devices:

These devices are semiconductor chips that are needed to connect peripherals to the bus system. Several type of interfacing devices are necessary to interconnect the component of a bus oriented system. The commonly used devices are tri-state , buffer, encoder, decoder and latches.

Tri- state devices: In general a logic device has two states: logic 1 and logic 0. The tri-state device has a third state too, high impedance state. The devices has a third line (other than input and output) called enable. When this line is activated, the tri-state device functions the same way as ordinary logic devices. When the third line is disabled, the logic device goes into the high impedance state as if it were disconnected from the system. Ordinarily , current is required to drive a device in logic 0 and logic 1 states. In the high impedance state 1 practically no current is drawn from the system.

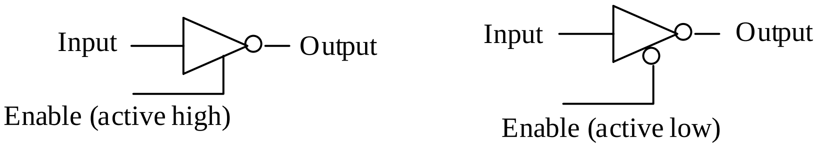


Fig. tri-state buffer

In a micro-compurter systems, peripherals are connected in parallel between the address bus and data bus. However because of tri-state interfacing devices, peripherals do not load the system buses. The microprocessor communicates with one device at a time by enabling the tri-state line of the interfacing device. Tri-state logic is critical to proper functioning of the microcomputer.

Buffer: The buffer is a logic ckt that amplifies the current or power. It has one input line and one output line. The logic level of the output is same as hat of the input; logic 1 input provides logic 1 output. The buffer is used primarily to increase the driving capability of a logic circuit. It is also known as driver.

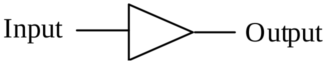
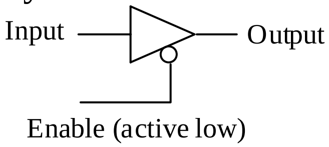


Fig. A buffer

Tri-state buffer: This is the buffer with a third line enable to activate the device. When the line is activated it acts as an ordinary buffer when disabled the buffer goes into high impedance state. This buffer in commonly used to increase the driving capability of the data bus and the address bus.



As the address bus is unidirectional, this device is commonly used as a driver. The octal buffer 74LS244 is a typical example of a tri-state buffer.

The data bus of a μc is bi-directional , therefore it requires a buffer that allows data to flow in both directional. A combination of two buffers in different direction combines to form a bi-directional buffer. The bi-directional buffer 47LS245 is a typical example and is used commonly as a driver for the data bus.

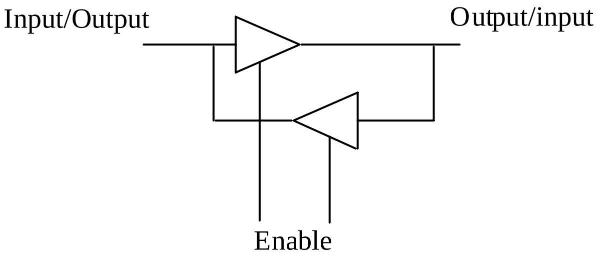
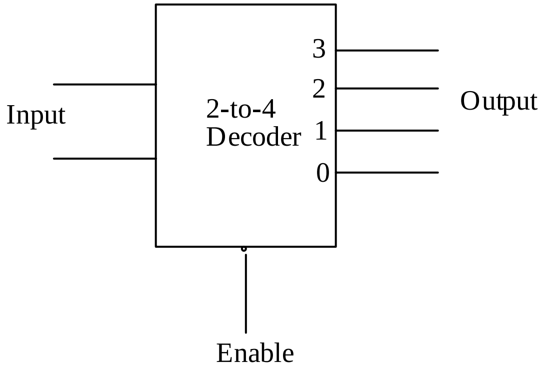
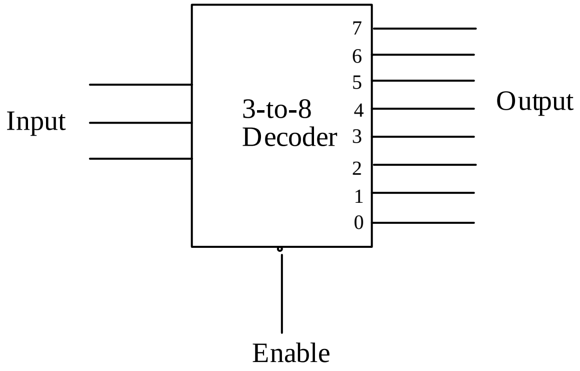


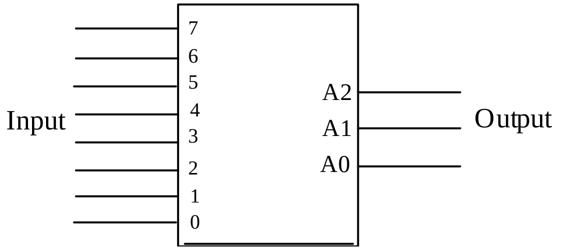
Fig. Bidirectional buffer

Decoder: The decoder is a logic ckt that identifies each combination of the signals present to its input. If the input to a decoder has n lines, the decoder will have 2^n output lines. For example if the input has two binary lines the output lines would be 4. The two lines can assume four combinations of input signals. – 00, 01, 10, 11 – with each combination identified by the output lines 0 to3. If the input is 11, the output logic 3 will be at logic 1 and otherwise will remain at logic 0 . This is called decoding . Various types of decodes are available; for example 3 to 8 , 4 to 16 , etc. In general decoders have enable lines too. The decoder will not function unless enable lines are activated.





Encoder: The encoder is a logic ckt that provide the appropriate code(binary, BCD etc) as output for each input signal. The process is the reverse of decoding . Figure below shows an 8 to 3 encoder; it has eight active low inputs and three output lines.



When 0 goes low , the output is 000, when the input line 5 goes low the output is 101. Encoder are commonly used with keyboards. For each key pressed, the corresponding binary code is placed on the data bus.

Latches: A latch is used commonly to interface output devices. When the µPU sends an output, data are available on the data bus for only few micro-seconds ; therefore a latch is used to hold data for display.

In its simples form, a latch is a D flip flop. We are having other latches(flip flops) like; RS flip flop, JK flip flop and Master slave flip flop etc.

Flip flop even form a cell of memory.

Internal structure of memory:

The internal structure of a 8 × 8 memory is shown below. Every memory unit has the similar types of structure.

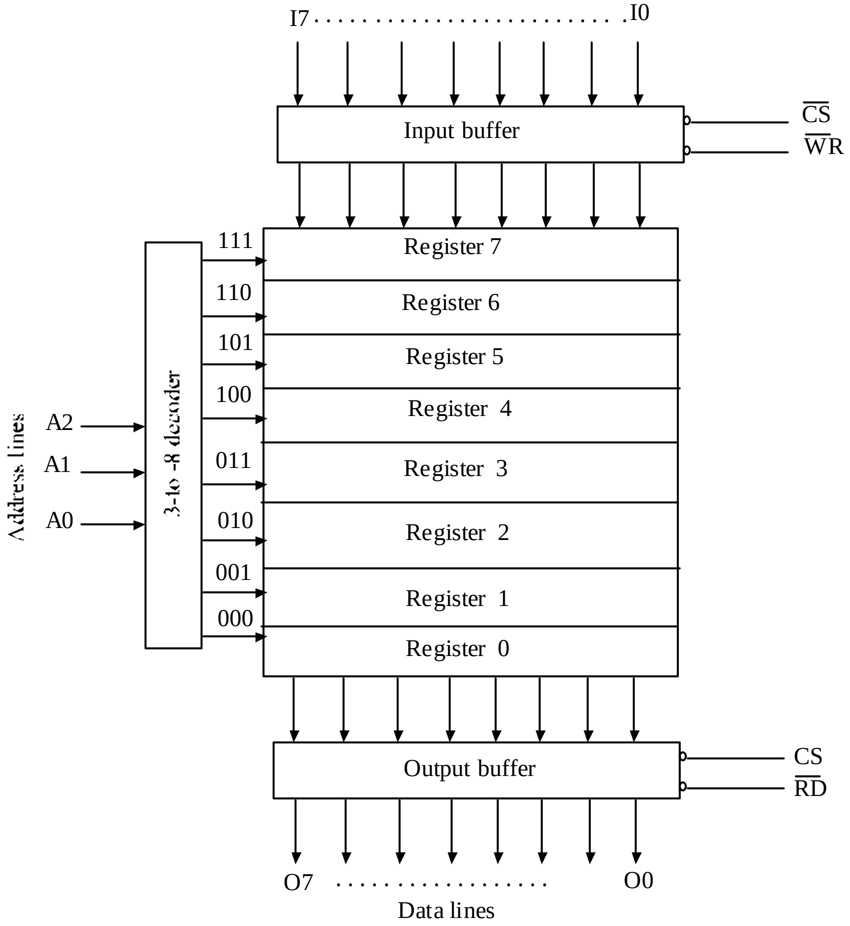


Fig. A 8×8 memory unit

Internally a memory consists of:

~ address decoder
~ input buffer
~ output buffer
~ registers

With

~ address lines
~ data lines
~ $\overline{RD}$, $\overline{WR}$, $\overline{CS}$ control lines.

The number of address lines will be determined by the memory capacity. The number of data lines will be determined by memory size. For example for memory capacity 1k× 8 will have 10 address lines and data lines. 2k × 4 chip will have 11 address lines and 4 data lines. For 2^n k × m memory capacity the number of address lines = n and number of data lines = m.

Let's consider the 8 × 8 memory device with 8 register , a 3 to 8 decoder, an input buffer and an outpur buffer. The device will have 3 address lines and eight data lines. It will also have control lines $\overline{RD}$, $\overline{WR}$ and $\overline{CS}$.

- To write an 8 bit word the µp places the register address on the three address line e.g to write in the register 7, µp places 111 on the address lines.
- The decoder decodes the address and selects the register 7.
- Then the µp places the data on the data bus and sends the active low $\overline{WR}$ control signal.
- The control signal enables the input buffer and data are placed in the selected register.
- To read from this memory, the process is similar to that of write operation except that output buffer is enabled with $\overline{RD}$ active low signal.
- The remaining address lines of the µp address bus are used to select the chip ($\overline{CS}$).

Basic concept in memory interfacing:

The primary function of memory interfacing is that two microprocessor should be able to read from and write into a given register of a memory chip. To perform this operation the micro processor should.

1. Be able to select the chip
2. Identify the register.
3. Enable the appropriate buffer.

Let us consider a RAM chip (6.6) to be interfaced with 8085 µP. The interfacing ckt is shown below:

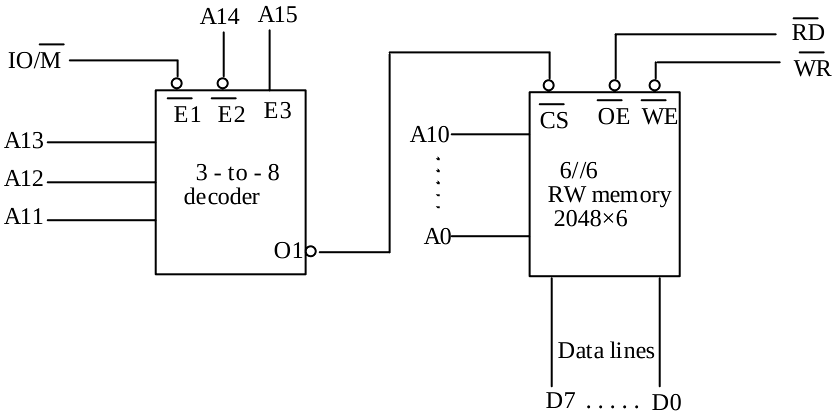


Fig. Interfacing R/W memory

For interfacing the chip with 8085 µp we just need and additional 3 to 8 decoder to select the chip. The register inside the chip can be identified by the internal address decoder and input or output buffers can be enabled by the control signals $\overline{RD}$ or $\overline{WR}$.

The memory chip 2048 × 8 requires 11 address lines to identify the 2048 register. Therefore the lower address lines $A_{10} – A_0$ form the µP are connected to the chip. The remaining address line ($A_{15} – A_{11}$) should be decoded to generate a chip select ($\overline{CS}$) signal unique for that chip.

The remaining five lines are connected to the decoder (3 to 8 decoder) as shown is the figure above.

- The decoder is enabled by $IO/\overline{M}$ signal is addition to address lines A_{15} and A_{14}
- The output O_1 of the decoder is connected to $\overline{CS}$ of the memory chip.
- The input line lines to the decoder are A_{15}, A_{12}, and A_{11}. These activate the output O_1 to select the memory chip.
- Thus to select the memory chip we must have the output O_1 to selected by the 3 to 8 decoder, which means we need $A_{13} = 0$

$$A_{12} = 0$$
$$A_{11} = 1$$

- Also to enable the 3 to 8 decoder we need $IO/\overline{M}$ low and $A_{14} = 0$, $A_{15} = 1$

Thus the chip is selected by '10001' at lines $A_{15} – A_{11}$ of the address bus. And hence the range of address for the memory chip would be

10001 00000000000 = 8800 H
10001 11111111111 = 8FFF H

Thus for interfacing the 6116 (RAM chip) the – 8 data lines of µp are connected to 8- data lies of the chip.

- The lower address lines $A_{10} – A_0$ are connected to that of the chip.
- The higher address lines $A_{15} – A_{11}$ are connected to the 3-to-8 line decoder.
- The chip is selected by the output O_1 of the 3-to-8 line decoder being connected to the $\overline{CS}$ pin of the chip.

Similarly the EPROM can be interfaced with 8085. The only difference would be, there would be no $\overline{WR}$ lines connection, as this memory is not used for write operation. The interfacing is as shown below.

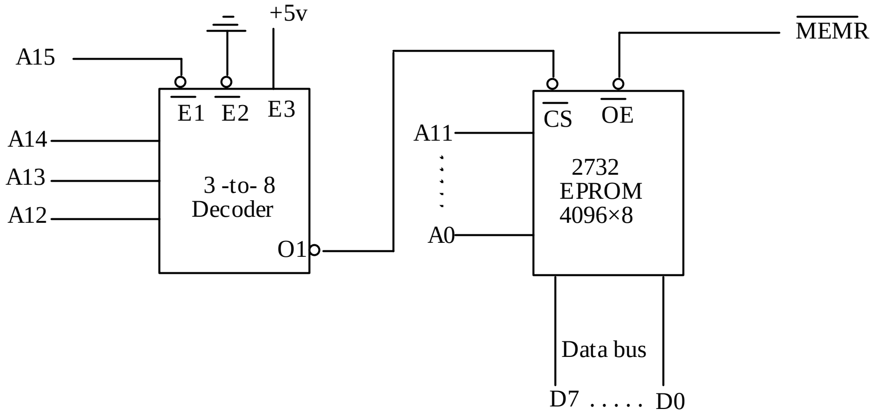


Fig. interfacing 2732 EPROM

Four interrupts TRAP, RST 7.5, 6.5, 5.5 are automatically vectored (transferred) to specific locations on without any external hardware. They do not require $\overline{INTA}$ signal or an input port; the necessary hardware is already implemented inside the 8085. These interrupts and their call locations are:

	Call locations
TRAP	0024 H
RST 7.5	003C H
RST 6.5	0034 H
RST 5.5	002CH

The TRAP has the highest priority, followed by RST 7.5, 6.5, 5.5 and INTR. Figure below shows the schematic diagram of 8085 Interrupts.

I/O interface:

Input and output interfaces provide a method for transferring information between internal storage and external I/O devices. Peripherals connected to a computer need special communication links for interfacing them with central processing unit. The purpose of the communication link is to resolve the difference that exist between the processor and each peripheral. The major differences are:

1. Peripherals are electromechanical and electromagnetic devices and their manner of operation is different form the operation of the CPU and memory, which are electronic devices. Therefore a conversion of signal value may be required.
2. The data transfer rate of peripherals is usually slower than the transfer rate of the CPU, and consequently a synchronization mechanism may needed.
3. Data codes and formats in peripherals differ from the word format in the CPU and memory.
4. The operating modes of peripherals are different from each other and each must be controlled so as not to disturb the operation of the other peripherals connected to the CPU.

To resolve these differences computes systems include special hardware components between the CPU and peripherals to supervise and synchronize all inputs and outputs transfers. These components are called interface units because they interface between the processor bus and peripheral devices. The two major types of I/O interface are:

1. Serial interface.
2. Parallel interface.

Serial interface: A serial interface exchanges data with the peripheral in serial mode, where data are transmitted one bit at a time along a simple communication link. Serial transmission is slow but inexpensive to implement as far as the number of wires is connected.

The function of serial I/O interface is to deal with the data on the bus in the parallel mode and to communicate with the connected device in the serial mode. If the bus has n data lines the serial i/o interface accepts n bits of datas simultaneously form the bus . These n bits are put to the I/O devices one bit at a time requiring n time slot for transmission. The reverse process takes place during reception of data form the device.

Parallel interface: Some I/O devices can handle data at speeds that can not be supported with serial interfaces so such a case a parallel interface must be used, where n bits of the data are handled simultaneously by the bus an on the links to the device. This achieves a faster interchange of data but becomes expensive due to the need of multiple wires. If any I/O devices particularly those requiring high data transfer rate use this arrangement.

Communication can be broadly defined as an unilateral or bipolar transfer of meaningful data between two points through a mediums. Communication can be in

1. Simplex mode
2. Half duplex mode
3. Full duplex mode.

The simplex mode: This mode allows the transmission of data over a single channel in one direction only. The data can be transmitted only from point A to point B not form point B to point A as shown in figure below. Physically two wire are required for transmission of data. One is signal wire and other is ground return wire.

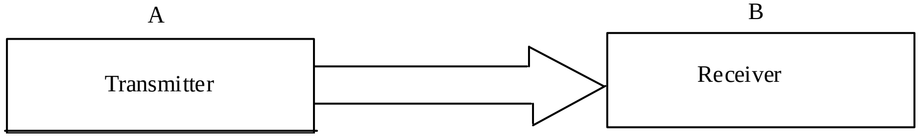


Fig. Simplex mode

The half duplex mode: This mode is also known as two way alternate transmission. This mode allows the transmission of data over a single channel in both directions but not simultaneously. In this mode too, just two wires are required, but additional circuitry is needed to determine and establish the direction of flow of information..

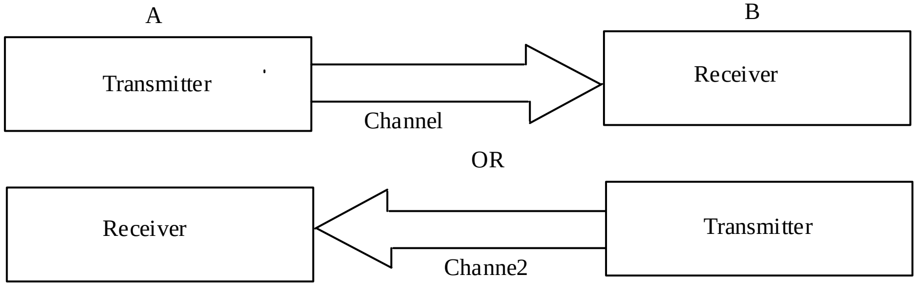


Fig. Half duplex mode

The full duplex mode: This mode allows the transmission of data in both directions simultaneously but on two separate channels. Two signal wires and two ground return wires are required in this mode. A disadvantage of this mode is the additional cost of the second channel. It is however possible to offer simultaneously two way communication on a single channel using complex circuitry.

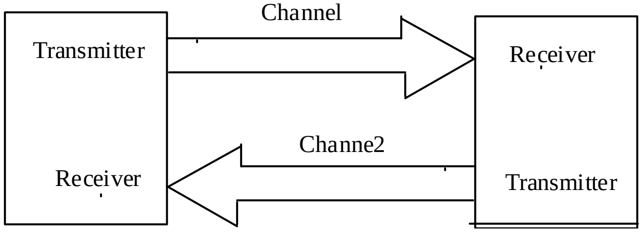


Fig. Full duplex mode

Method of communication:

Parallel data transfer: When a word of n bits is to be transmitted in parallel each bit is transmitted on a separate line along with a common ground line with respect to which the status of each line is measured. Thus, a channel comprises of (n+1) lines.

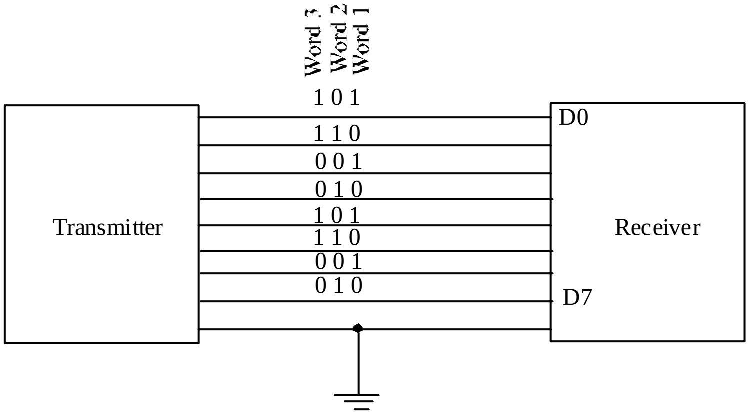


Fig. parallel data transfer

Here, the time required to transfer one word is equal to the time taken to transmit a bit. Parallel data transmission is impractical over long distances because of prohibitive cost of installing a large number of lines.

Serial data transfer: In serial data transfer, each bit of the word is sent in succession, one at a time over a single pair of wires. A parallel to serial converters is used to convert the incoming parallel data to serial form and then the data is sent out with the lest significant bit D_o first and most significant bit D_7 coming last of all. If the bit rate is retained after the parallel to serial conversion, the time taken to transmit a word in serial data transmission will be n times more than the time taken in parallel data transmission. If the word in the above example were to sent serially, the data on the channel will appear as in figure below.

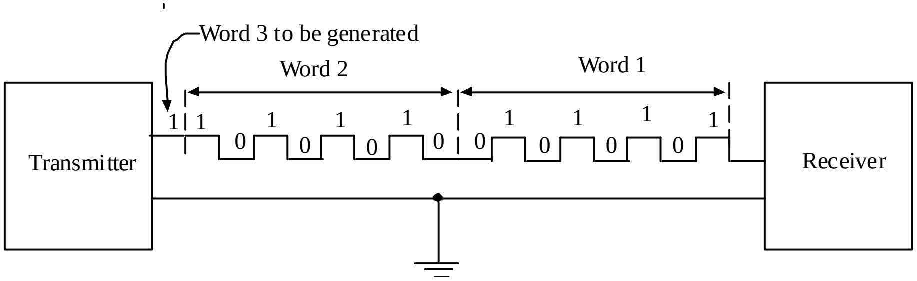


Fig. serial data transfer

There are tow types of serial transfers. They are:

1. Asynchronous serial data transfer.
2. Synchronous data transfer.

Asynchronous transfer:

In this types of transmission, the receiving device does not 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 unit must be formatted. In other words, each data unit must contain 'a start bit' and 'stop bit(or bits)' indicating the beginning and the end of each data unit . In asynchronous transmission the data message is sent one word at a time.

- When no datas are sent over the time it is maintained at an idle value; a logic 1.
- Start bit is a logic 0.
- The stop bit is a logical 1.

Both the transmitter and receiver are given separate clocking signals. It is not essential that they both be of exactly the same frequency. The data is sent out of the transmitter synchronous with its own clock input and the data is similarly received and assembled at the receiver.

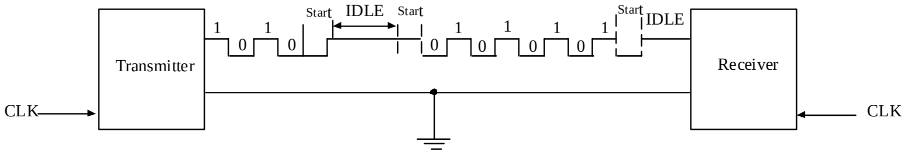


Fig. Asynchronous communication.

When a character is to be transmitted, the transmitter first sends out a low bit. This transition is perceived by the receiver and it gets ready to receive the data.

The transmitter then sends out the word bit by bit, one after the other, synchronous with its clock.

The receiver assembles the data one by one synchronous with its clock. The receiver must known beforehand the transmission 'Baud Rate'(bits per seconds) for proper assembling at its end. After all the bit of a character are sent, the transmitter sends out a stop bit which is the idle value (logic 1) to indicate the end of transmission.

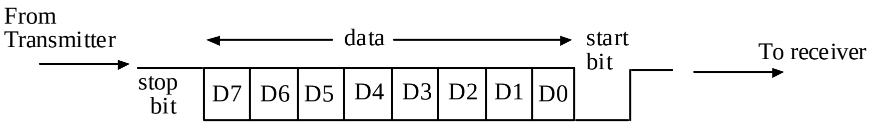


Fig. Transmission format for Asynchronous transfer

Asynchronous communication is a start-stop type of communication and is used where the source of data may not be providing a steady stream of new characters. The data thus comes to the receiver at unevenly spaced intervals without reference to a master clock, hence the name Asynchronous.

Asynchronous transmission: Synchronous communication is used for transferring large amount of data at a stretch without frequent start or stops. In synchronous systems to , the line is maintained at the idle value when no data is being transmitted. The transmission begins with a block header which is a predetermined pattern of bits. The receiver identifies the pattern and gets ready to receive the characters.

The transmitter sends the data character by character, bit by bit. After sending all the characters, the transmitter sends another pattern of bits to indicate the end of transmission.

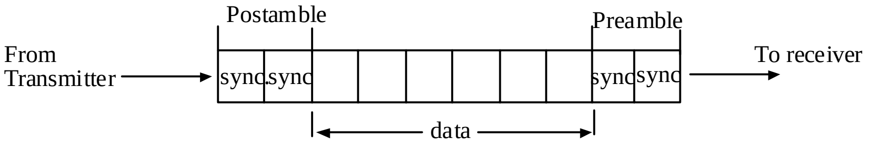


Fig. Transmission format for synchronous transfer

This format is generally used for high speed transmission (more then 20 kbps). Here the receiver and transmitter are synchronized by a master clock i.e both function depending upon the same clock signal.

Baud Rate: The rate at which the bits are transmitted (bits per seconds) is called a baud; technically, however it is defined as the number of signal changes/ second. Each piece of equipment has its own band requirements.

Bus Interface standards: The microcomputer is a bus oriented system where by subsystems or peripherals are interconnected through the bus architecture. Peripheral and computers are designed and manufactured by various manufactures. Therefore, a common understanding must exit , among various manufacturing and user groups, that can ensure compatibility among different equipment. When this understanding is defined and generally accepted in industry (and by users) , it is known as a standard. In the field of electronics, these standards are generally defined by professional organizations such as IEEE; EIAC (Electronic industries Association or the standards are forced upon the industry either by a dominant manufacturer or by common practice. Various bus standards are defined till now. We would be dealing RS 232C and GPIB here.

RS 232C: This interface standard is most widely used standard for serial communication between microcomputers and peripheral devices. The interface, defined by EIA, relates essentially to two types of equipment. The first is known as data terminal equipment. While the second is referred as data communication equipment (DTE). The data terminal equipment (e.g a microcomputer) is capable of sending and/or receiving data via the serial interface. The data communication equipment on the other hand is generally through of as a device which can facilitate serial data communications (e.g modems).

- RS 232C works in a negative logic. The standard specifies that 'the logic one' level is a voltage between -3 and -15 v and 'the logic zero' is a voltage between +3 and +15 V. The commonly used voltages are +12v and -12V.
- The transmission line normally used a twisted pair of shielded wire with a line capacitance of more than 1200PF and no less than 300Pf. The standard specifies the line length to 50 feets only.
- The standard describes the function of 25 signal and hansake pins for serial data transfer. It also specifies that the DTE connector should be male and the DCE connector should be female. Usually 9 pin and 25 pin connectors are available.

Among the 25 pins of RS232C the independent pin numbers are:

Pin no.	**Signals**	**Functions**
2	Transmit data, TXD	Output , transmit data from DTE to DCE
3	Receive data RXD	Input, DTE receive data from DCE
4	request to send, RTS	General purpose output from DTE

5 Clear to send,CTS Input to DTE, used as handshake
6 Data set ready, DSR Input to DTE, indicate that DCE is ready
7 Singal ground,GND Common reference bet. DTE and DCE
8 Data carrier detect, DCD Used by DTE to disable data reception
20 Data terminal ready, DTR Output means DTE is ready.

The main problem with –RS-232C is that it can only transfer data reliably about 50ft (16.4m) at its maximum rate of 20k band. If longer lines are used the transmission rate has to be drastically reduced. For higher rate of transfers and for longer distance we have another standards defined.

Standards	**Speed**	**Distances**	**Voltage Range**
RS 232C	20k band	50ft	±15V
RS-422A	10Mbaud at 40ft	4000ft	±7
	100 Kbaud at 4000ft		
RS-485A	100K baud at 30ft		
	1 kbaud at 4000ft	4000ft	±12V

RS 232C interface with DTE and DCE: The figure below shows a interfacing with minimum lines.

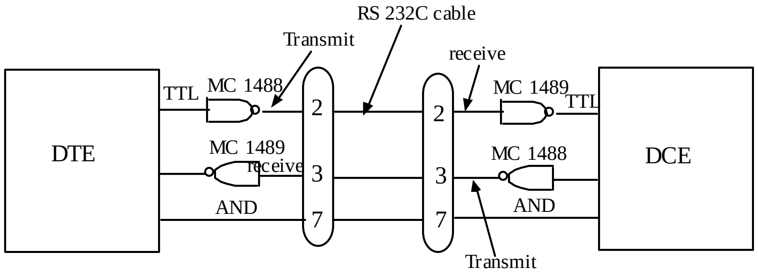


Fig. RS 232C interface

The signaling in RS-232C is not compatible with the TTL logic level. For TTL 0 v to 0.2V is considered a logic 0 and 3.4 v to 5v as logic 1. But RS-232C works in a negative logic -3 to -15v considered as logic 1 and +3 to +15v as logic 0. Because of this incompatibility of the data lines with the TTL logic, voltage translators called line drivers and line receivers are required to interface TTL logic with RS-232C signals.

The line driver MC 1488 converts logic 1 into approx -9V and logic 0 into +9v. Before it is received by the DCE it is again converted by the line receiver MC 1489 into TTL-Compatible logic.

The minimum interface required both a computer and a peripheral device requires three lines; pin 2,3 and 7. These lines are defined in relation to the DTE; the terminal transmits on pin 2 and receives as pin 3. On the other hand the DCE transmits on pin 3 and receives on pin 1. Pin 7 is ground pin.

GPIB (General purpose instrumentation bus) ~ IEE-488 standard

~ HPIB (Hewlette puckard interface bus)

This is the most widely used industrial bus format. It was developed to facilitate interface of computers with various instruments as – printers, tape recorders, digital voltmeters, frequency counters, function generators etc. Some of the features of this bus are:

1. Data transfer among the interconnected devices in digital format.
2. Allows 8 bit parallel bi-directional communication.
3. Fifteen devices may be connected to one continuous bus.
4. Total transmission path is limited to 20 metres or two meters per device.
5. Data transfer rate on any single line is limited to 1M-byte/sec
6. The bus as 24 signals: eight bidirectional data lines.
   : eight control lines.
   : eight ground lines.

This standard describes four types of devices which may be interface to GPIB.

a. **Listners:** These devices can receive data band control signals from other devices connected to the bus but are not capable of generating data. e.g printers.
b. **Talkers:** These devices are only capable of placing data on the bus and can not receive data. Only one talker can be active at a given time but fourteen devices can listen at a time. Example for talkers are: scanners, tape readers, voltmeters etc.

Talkers/ listners: Talkers and listners can both send and receive data to/from the bus. A digital multimeters is a typical example. Data is sent to it in order to change ranges and returned to the bus in the form of digitized readings of voltage, current and resistances.

Controller: controllers are used to supervise the flow of data on the bus and provide processing facilities. The controller within an IEE-488 system is a microcomputer.

I/O ports addressing: A number of microprocessor have distinct memory and I/O address spaces. These processors generally have special instruction to access a port. A separate control line is used to distinguish between I/O and memory operations. In 8085 IO/$\overline{M}$ line is used for this purpose. The address of the I/O port (XXH) is put on both the lower and upper eight lines of the address bus during T_1 of the machine cycle. I/O device can be interfaced to an CO85A system in two ways: 1. I/O mapped I/O mode
2.Memory mapped I/O mode.

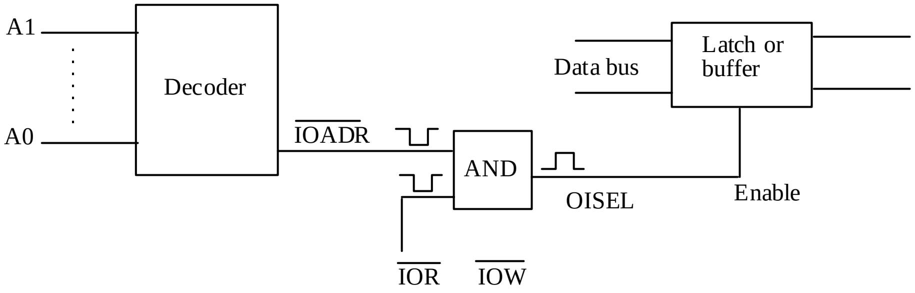
I/O mapped I/O mode (standard I/O): In this mode, the IO/$\overline{M}$ signal of the 8085A is used to distinguish between I/O read/write and memory Read/write operations. The I/O device an be accessed during IN and OUT instructions (a 1 byte address is specified in the instruction).

Memory mapped I/O mode: In this mode the IO/M signal output is not used to distinguish between memory and I/O devices. The microprocessor communicates with an I/O device as if it were one of the memory location. Hence the I/O devices are address as memory.

Interfacing: The objective of interfacing an output device is to get information or a result out of the processor and to display it, similarly the input device is interfaced to get information into the processor. To do that three common steps are undertaken.

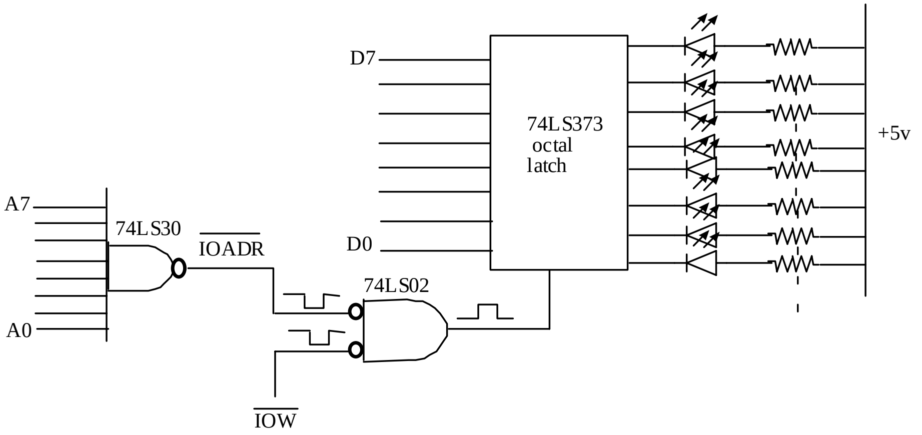
1. Decode the address bus to generates a unique pulse corresponding to the device address on the bus; this is called I/O address pulse.
2. Combine (AND) the device address pulse with the control signal to generate a device select (I/O select) pulse that is generated only when both signals are applied.
3. Use the I/O select pulse to active the interfacing device (I/O ports)

The diagram below illustrates the steps:



The address lines $A_7 – A_0$ are connected to a decoder which will generate a unique pulse corresponding to each address on the address lines. This pulse is combined with the control signal to generate a device select pulse. Which is used to enable an output latch or an input buffer.

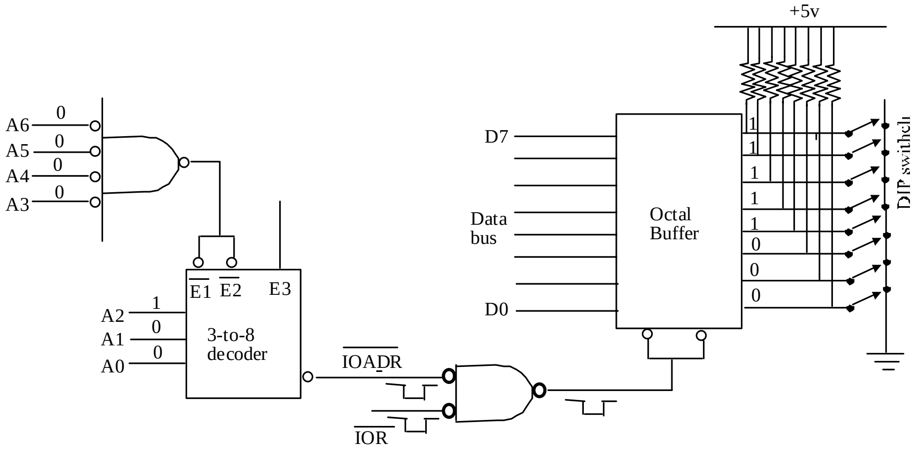
Interfacing output display: The figure below shows an output interfacing ckt for LED display.



Here:

- The address bus $A_7 – A_0$ is decoded by using an 8 input NAND gate.
- The output of the NAND gate goes low only when the address line carry the address FF H.
- The output of the NAND gate combines with the microprocessor control signal $\overline{IOW}$ in a NOR gate (connected as a negative AND). The output of NOR gate (74LS02) goes high to generate on I/O select pulse when both inputs are low(or both signals are asserted).
- Mean while the contents of the accumulator have been put on the data bus.
- The I/O select pulse is used to activate the latch and data are latched and displayed on the diodes.(LEDs)

Interfacing an Input device (DIP switches): The figure below shows an input interfacing ckt for DIP switches.



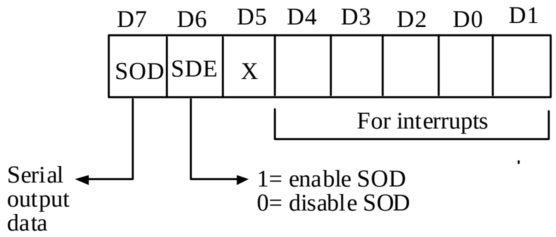
Here:

- The address bus $A_7 – A_0$ is decoded using an decoder.
- The output of the decoder (O_4) goes low only when the address line carry the address 84H.
- The output of the decoder combines with the µP control signal $\overline{IOR}$ in a negative NAND gate. The output of this gate goes low to generate an I/O select pulse when both inputs are low.
- The I/O select pulse is used to activate the buffer and the data from the DIP switches (F8H) is put on the data.

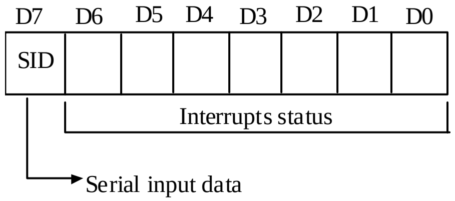
The 8085- serial I/O lines: SOD and SID:

- The 8085 µP has two pins specially designed for software controlled serial I/O. One is called SOD (serial output data) and the other is called SID(serial input data).
- Data transfer is controlled through two instructions: SIM and RIM

SOD: The instruction SIM is necessary to output data serially from SOD line. It can be interpreted for serial output as below:



SID: The instruction RIM is used to input serial data through the SID line. It can be interpreted for serial input as below.



SID and SOD lines eliminates the use of input and output port in the software controlled serial I/O. SID is a 1-bit input port and SOD is a 1-bit output port.

Interrupt:

An interrupt is a signal that a peripheral board sends to the central processor in order to request attention. In response to an interrupt, the processor stops what it is currently doing and executes a service routine. When the execution of the service routine is terminated, the original process may resume its previous operation.

The interrupt is initiated by an external device and is asynchronous, meaning that it an be initiated at any time without reference to system clock. However , the response to an interrupt request is directed or controlled by the microprocessor.

Interrupts are primarily issued on:

~ initiation of I/O operation.
~ completion of I/O operation.
~ occurrence of hardware or software errors.

Process of interrupt operation:

1. The I/O unit issues an interrupt signal to the processor. An interrupt signal from I/O is the request for exchange of data with the processor.
2. The processor finishes execution of the current instruction before responding to the interrupt.
3. The processor testes for an interrupt, determines that there is one, and sends an acknowledgement signal to the device that issued the interrupt. After receiving this acknowledgement, the device retains its interrupts signal.
4. The processor now begins to transfer the control to the routine which serves the interrupt request from the device. This routine is called 'Interrupt service routine' and it resides at a specified memory location.

For this process, the CPU needs to save information needed to reassume the current program at the point of interrupt. The minimum information required is i) the status of the processor, which contained by the processor status word (PSW) and ii) the location of the next instruction to be executed which is contained by the program counter (PC), these all are pushed onto the stack.

5. The processor then loads the program counter with the entry location of the interrupt service routine that will respond to this interrupt. Once the program counter has been loaded, the control is transferred to the interrupt handler program.
6. The fundamental requirement of the interrupt service routine is that it should begin by saving the contents of all the registers on the stack(as state of the main program should be safe).

Suppose the user program is interrupted after the instruction at location N. The contents of all the registers plus the address of the next instruction are saved on the stack. The stack pointed is updated and the programs counter is updated to point to the beginning of the interrupt service routine.

7. The interrupt handler now proceeds to process the interrupt. This will include an examination of status information relating to the I/O operation or the other event that caused an interrupt. It may also involve sending additional commands or acknowledgement to the I/O unit.
8. When interrupt processing is complete the saved register's value (of the main program) are retrieved from the stack and restored to the register.
9. The final function is to restore the PSW and program counter values from the stack. As a result the next instruction to be executed will be from the previously interrupted main program.

Types of interrupt:

There are three major types of interrupts that cause a break in the normal execution of a program. They can be classified as:

1. External interrupts
2. Internal interrupts
3. Software interrupts.

External interrupts: External interrupts are initiated via the microprocessor's interrupt pins by external devices such I/O devices, timing device, circuit monitoring the power supply etc. Causes of these interrupts may be; I/O device requesting transfer of data, I/O device finished transfer of data, elasped time of an event, or power failure. Timeout interrupt may result from a program that is an endless loop and thus exceeded its time allocation. Power failure interrupt may have as its service routine a program that transfers the complete state of the CPU into a non-destructive memory in few milliseconds before power ceases.

External interrupts can be further divided into two types:

1. Maskable interrupt.
2. Non-maskable interrupt.

Maskable interrupt: A maskable interrupt is one which cab be enabled or disabled by executing instructions such as EI (enable interrupts)and DI (Disable interrupt). If the microprocessor's 'interrupt enable flip flop' is disabled, it ignores a maskable interrupt. In 8085, the 1 byte instruction EI sets the interrupt enable flip flop and enables the interrupt process. Similarly the 1 byte instruction DI resets the interrupt enable flip flop and disables the interrupt process. No maskable interrupts are recognized by the processor when the interrupt is disabled.

Non maskable interrupt: This type of interrupt cannot be enabled or disabled by instructions. This types has higher priority over the maskable interrupt. This means that if both the makable and non maskable interrupts are activated at the same time, then the processor will service the non-maskable interrupt first. In 8085 TRAP is an example of non maskable interrupt.

Internal interrupt: Internal interrupt arise form illegal or erroneous use of an instruction or data. Cause of this interrupt may be: register overflow, attempt to divide by zero, an invalid operation code, stack overflow etc. These error conditions usually occur as a result of premature termination of the instruction execution. These are even termed as exceptions.

The difference between internal and external interrupt is that the internal interrupt is initiated by some exceptional conditions caused by the program itself rather than by an external events. Internal interrupts are synchronous with the program, while external interrupts are asynchronous. If the program is return, the internal interrupt will occur in the same place each time. External interrupts depends on external conditions that are independent of the program being executed at the time.

Software interrupt: External and internal interrupts are initiated from signal that occur in the hardware of the cpu. A software interrupt is initiated by executing an instruction. Software interrupt is a special call instruction that behaves like an interrupt rather than a subroutine call. It can be used by the programmer to initiate an interrupt procedure at any desired point in the program. The most common use of software interrupt is associated with a supervisor call instruction. This instruction provides means for switching from a CPU user mode to the supervisor mode. Certain operations in the computer may be assigned to the supervisor mode only, as for example, a complex input or output transfer procedure. In 8085 the instruction like RST0, RST1, RST2, RST3…..etc. causes a software interrupt.

Interrupt priority: Data transfer between the CPU and an I/O device is initiated by the CPU. However , the CPU cannot start the transfer unless the device is ready to communicate with the CPU. The readiness of the device can be determined from an interrupt signal. The CPU responds to the interrupt request by storing the return address form PC into a memory stack and then the program branches to a service routine that process the required transfer.

In micro-computer a number of I/O device are attached to the processor, with each device being able to originate an interrupt request. The first task of the interrupt system is to indemnify the source of the interrupt. There is also the possibility that several sources will request service simultaneously. In this case the system must also decide which device to service first.

An interrupt priority is a system that established a priority over the various sources to determine which condition is to be serviced first when two or more request arrive simultaneously. The system may also determine which conditions are permitted to interrupt the computer while another interrupt is being serviced. Higher-priority interrupt levels are assigned to request which, if delayed or interrupted, could have serious consequences. Device with higher speed transfers such as magnetic disks are given high priority, and slow devices such as keyboards receive low priority. When two devices interrupt the processor at the same time, the processor service the device, with the higher priority first.

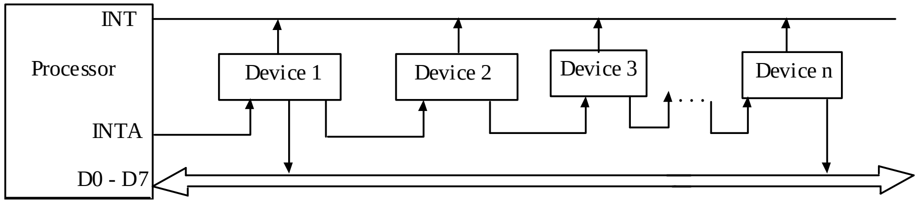
There are mainly two ways of servicing multiple interrupts. These are:

1. Polled interrupt.
2. Chained (Vectored) interrupt.

Polled interrupt: Polled interrupt are handled using software and are therefore slower compared to vectored (hardware) interrupts. In this method there is one common branch address for all interrupts. The program that takes care of interrupts begins at the branch address and polls the interrupts sources in sequence. The order in which they are tested determines the priority of each interrupt. The highest priority source is tested first, and if its interrupt signal is on, control branches to a service routine for this source. Other wise the next lower priority source is tested, and so on. Thus, the initial service routine for all interrupts consists of a program that test the interrupt sources in sequence and branches to one of many possible service routines.

Polled interrupts are very simple . But or large number of devices, the time required to poll each device may exceed the to service the device. In such case, the faster mechanism called chained interrupt is used.

Chained interrupt: This is hardware concept of handling the multiple interrupts. In this technique, the devices are connected in a chain fashion as shown in figure below for setting up the priority system.



Here the device with he highest priority placed in the fist position, followed by lower priority devices. Suppose that one or more devices interrupt the processor at a time. In response, the processor saves its current status and then generates an interrupt acknowledge (INTA) signal to the highest priority device, which is device 1 in our case. If this device has generated the interrupt it will accept the INTA signal from the processor; otherwise, it will pass INTA on to the next device until the INTA is accepted by the interrupting device.

Once accepted, the device provides a means to the processor for finding the interrupt address vector using external hardware. Usually the requesting device responds by placing a word on the data lines. With the help of hardware it generates interrupts vector address. This word is referred to as vector, which the processor used as a pointer to the appropriate device service routine.

This avoids the need to execute a general interrupt service routine first. So this technique is also referred to as vectored interrupts.

Interrupts of 8085: The 8085 has five interrupts:

i. TRAP
ii. RST 7.5
iii. RST 6.5
iv. RST 5.5
v. INTR

The four interrupts TRAP, RST 7.5, 6.5,5.5 are automatically vectored (transferred) to specific locations on without any external hardware. They do not require $\overline{INTA}$ signal or an input port; the necessary hardware is already implemented inside the 8085. These interrupts and their call locations are:

TRAP	Call location
TRAP	0024 H
RST 7.5	003 CH
RST 6.5	0034 H
RST 5.5	002C H.

The TRAP has the highest, followed by RST 7.5, 6.5, 5.5 and INTR. Figure below shows the schematic diagram of 8085 interrupts.

INTR:

This interrupt is maskable. It can be enabled by instruction EI and can be disabled by instruction DI. The INTR interrupt requires external hardware to transfer program sequence to specific CALL locations. There are 8 numbers of CALL-Locations for INTR interrupt. The hardware circuit generate RST codes for this purpose and places that on the data bus externally.

When the microprocessor is executing a program, it checks the INTR line (when interrupt enable flip flop is enabled using EI instruction) during the execution of each instruction. If the line is high and the interrupt is enabled, the microprocessor completes the currents instruction, disabled the interrupt enable flip flop and sends a $\overline{INTA}$ signal. The processor does not accept any interrupt requests until the interrupt flip flop is enabled again.

The signal $\overline{INTA}$ is used to insert a Restart (RST) instruction, (it saves the memory address of the next instruction to the stack. The program is transferred to the cal location.). The RST instruction and their call locations are :

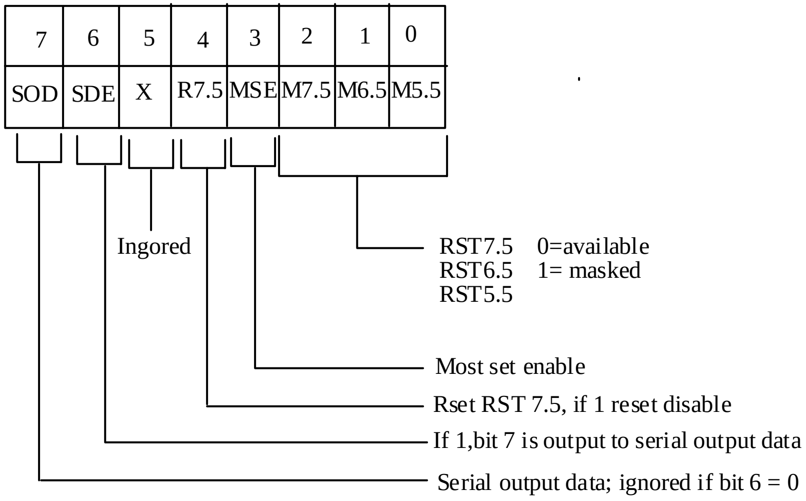
Instruction	Hex-code	Call location
RST 0	C7	0000
RST 1	CF	0008
RST 2	D7	0010
RST 3	DF	0018
RST 4	E7	0020
RST 5	EF	0028
RST 6	F7	0030
RST 7	FF	0038

- bit D_3 is a control bit and should be 1 for bits D_0 , D_1, and D_2 to be effective.
- Logic 0 on D_0, D_1, and D_2 will enable the corresponding interrupts and logic 1 will disable the interrupts.

ii) The second function is to reset RST 7.5 flip flop. Bit D_4 is additional control for RST 7.5

- If D_4 =1, RST 7.5 is reset. This is used to ignore RST 7.5 without servicing it.

iii) The third function is to implement serial I/O. Bit D_7 and D_6 are used for serial I/O and do not effect the interrupts.


7
6
5
4
3
2
1
0
SOD
SDE
X
R7.5
MSE
M7.5
M6.5
M5.5
Ingored
RST7.5 0=available
RST6.5 1= masked
RST5.5
Most set enable
Rset RST 7.5, if 1 reset disable
If 1,bit 7 is output to serial output data
Serial output data; ignored if bit 6 = 0

Assuming that the task to be performed is written as a subroutine at the specified location the processor performs a task. This service routine includes the instruction EI to enable the interrupt again and RE-instruction to retrieve the memory address where the program has interrupted. Then the execution goes to the main program again.

TRAP:

It is a non maskable interrupt. It has the highest priority among the interrupt signal. It need not be enabled and it cannot be disabled. When this interrupt is triggered the program control is transferred to the location 0024 H without any external hardware or the interrupt enable instruction. TRAP is generally used for such critical events as power failure and emergency shut off.

RST 7.5, 6.5, 5.5: These interrupts are maskable and are enabled by software using instructions EI and SIM (set interrupt mask). The execution of the instruction SIM enables/disables the interrupts according to the bit pattern of the accumulator.

SET Interrupt Mask (SIM) instruction:

- This is 1 byte instruction.
- Can be used for three different functions:

i). One function is set mask for RST 7.5, 6.6 and 5.5 interrupts. This instruction reads the content of the accumulator and enables or disables the interrupts.

Pending interrupts:

When one interrupt request is being served, other interrupt may occur resulting in a pending request. When more than one interrupts occur. Simultaneously the interrupts having higher priority is served and the interrupts with lower priority remain pending. The 8085 has an instruction RIM using which the programmer can know the current status of pending interrupts. This instructions gives the current status of only maskable interrupts.

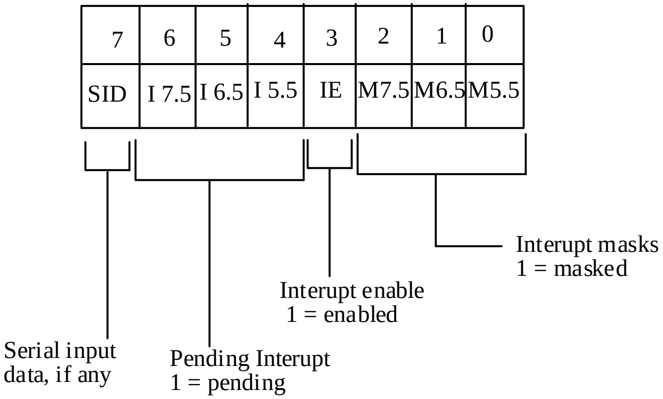
Instruction RIM:

- Read interrupt Mask.
- 1 byte instruction.
- Can be used for the followings.

a. To read interrupt mask. This instruction loads the accumulator with 8-bits indicating the current status of the interrupts.

b. To identify the pending interrupts. Bits D_4, D_5, and D_6 identify the pending interrupts .

c. To receive serial data. Bit D_7 is used to receive serial data.



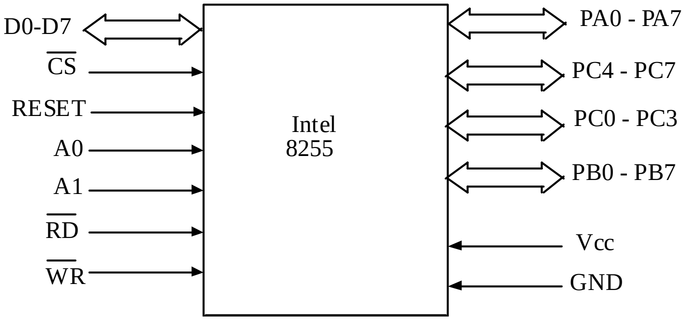
7	6	5	4	3	2	1	0
SID	I 7.5	I 6.5	I 5.5	IE	M7.5	M6.5	M5.5

I/O interface:

8255 programmable peripheral interface (PPI):

A programmable peripheral interface is a multipart device. The part may be programmed in a variety of ways as required by the programmer. The device is very useful for interfacing peripheral devices. It has 3 8- bit ports, namely port A, port B and port C. The port C has been further divide into two of 4-bit ports, and port C upper and port C lower. Thus a total of 4 ports are available, two 8-bit ports and two 4-bit ports. Each part can be programmed either as an i/p port or an o/p port.

$PA_0 – PA_7$ - 8 pins of port A
$PB_0 – PB_7$ - 8 pins of port B
$PC_0 – PC_3$ - 4 pins of port C_{lower}
$PC_4 – PC_7$ - 4 pins of port $C_{upper.}$



Operating modes of 8255:

8255 has 3 modes of operation which are selected by s/w.

Mode 0 - Simple input/output
Mode 1 – Strobed input/output
Mode 2 – Bidirectional port.

Mode 0: In this mode of operation, a port can be operated as a simple o/p or i/p port. Each of the four ports of 8255 can be programmed to be either an i/p or o/p port.

Mode 1: Mode 1 is strobed input/output mode of operation. The port A and port B are designed to operate in this mode of operation. When port A and port B are programmed in mode 1, six pins of port c are used for their control. PC_0,PC_1 and PC_2 are used for control of port B which can be used as input or output port. If the port A is operated as an input I/P or O/P port If the port A is operated as an input port. PC_3, PC_4, and PC_5 are used for its control. The remaining pins of port C, i.e PC_6 and PC_7 can be used as either input or output. When port A is operated as an output port, pins PC_3, PC_6 and PC_7 are used for its control. The pins PC_4 and PC_5 can be used either as input or output.

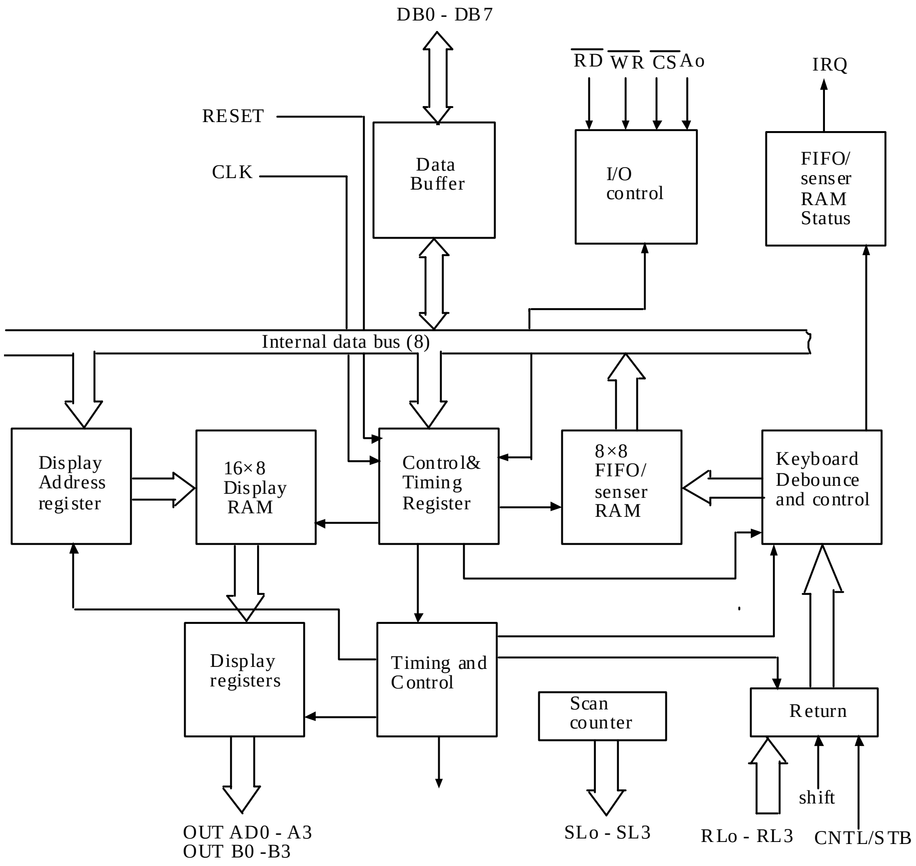
Mode 2: Mode 2 is strobed bidirectional mode of operation. In this mode port A can be programmed to operate as a bidirectional port. The mode 2 operation is only for port A. When port A is programmed in mode 2, the port B can be used in either mode 1 or mode 0. For mode 2 operation PC_3 to PC_7 are used for the control of port A.

8279 (Programmable keyboard/display interface): 8279 is a programmable keyboard interfacing device. It has been designed for the purpose of 8-bit µP . It has two sections namely keyboard section and display section. The function of the keyboard section is to interface the keyboard which is used as i/p device for the µP. The purpose of the display section is to drive alphanumeric displays or indication lights. Some important features are: simultaneous - keyboard display operations, scanned sensor mode, scanned. Keyboard mode, 8 – character keyboard FIFO, 2 – key lock out or N – key roll over with contact debouched, single 16-character display, dual 8 or 16 numerical display, programmable scan timing and mode programmable from CPU.

8279 block diagram:

Keyboard section: This section has eight lines (RL_0 – RL_7)that can be connected to 8 columns of a keyboard, plus two additional lines shift and CNTL/STB (control/strobe). The keys are automatically debouched and the keyboard can operate in two modes; two-key lockout or N-key rollover. In two-key lockout mode, if two keys are pressed almost simultaneously only the first key is recognized. In the N-key rollover mode, simultaneous keys are recognized and their codes are stored in the internal buffer. It can also be set up so that no key is recognized until only one key remains pressed.

This section also includes 8 × 8 FIFO RAM, that store keyboard entries and provides IRQ (interrupt request). Signal when FIFO is not empty.



Scan section: The scan section has scan counter and 4 scan lines (SL_0 – SL_3). These 4 scan lines can be decoded using a 4- to – 16 decoder to generate 16 lines for scanning.

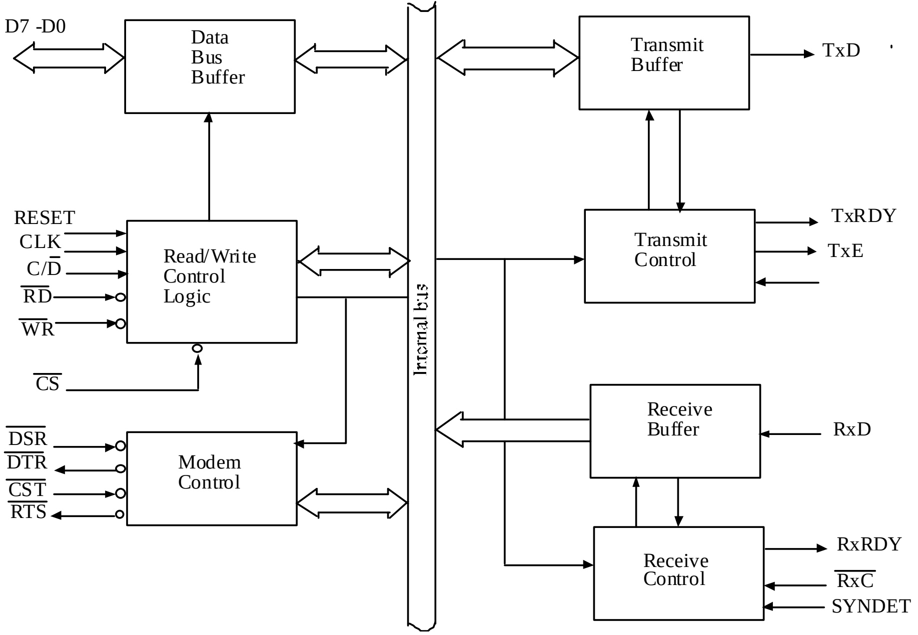
Display section: The display section has eight output lines divided into two groups A_0 – A_3 and B_0 –B_3. These lines can be used, either as a group of eight lines or as two groups of four, in conjunction with the scan line, for a multiplexed display. The display can be blanked by using the $\overline{BD}$ line. This section includes 16 × 8 display RAM.

MPU Interface section: This section includes eight bidirectional data lines (DB_0 –DB_7), one interrupt request (IRQ) line, and 6 lines for interfacing , including the buffer address line (A_0) .

When A_0 is high, signals are interpreted as control words or status; when A_0 is low, signals are interpreted as data. The IRQ line goes high whenever data entries are stored in the FIFO indicating the availability of data.

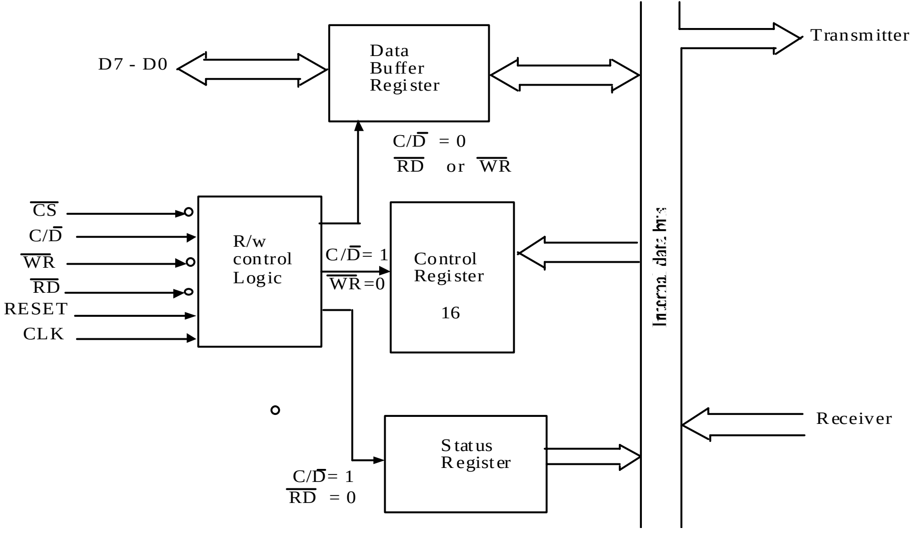
8251A programmable communication Interface:.

The 8251A is a programmable chip designed for synchronous and asynchronous serial data communication, packed in a 28-pin DIP. Figure shows the block diagram of 8251A. It includes 5 sections: Read/write control logic, Transmitter, Receiver, Data bus buffer, and Modem control.



The control logic interfaces the chip with the MPU. The transmitter section converts a parallel word received from MPU into serial bits and transmit them over the TXD line to a peripheral. The receiver section receives serial bits from a peripheral, converts them into a parallel word and transfers back to the MPU. The modern control is used to establish data communication through modems over telephone lines.

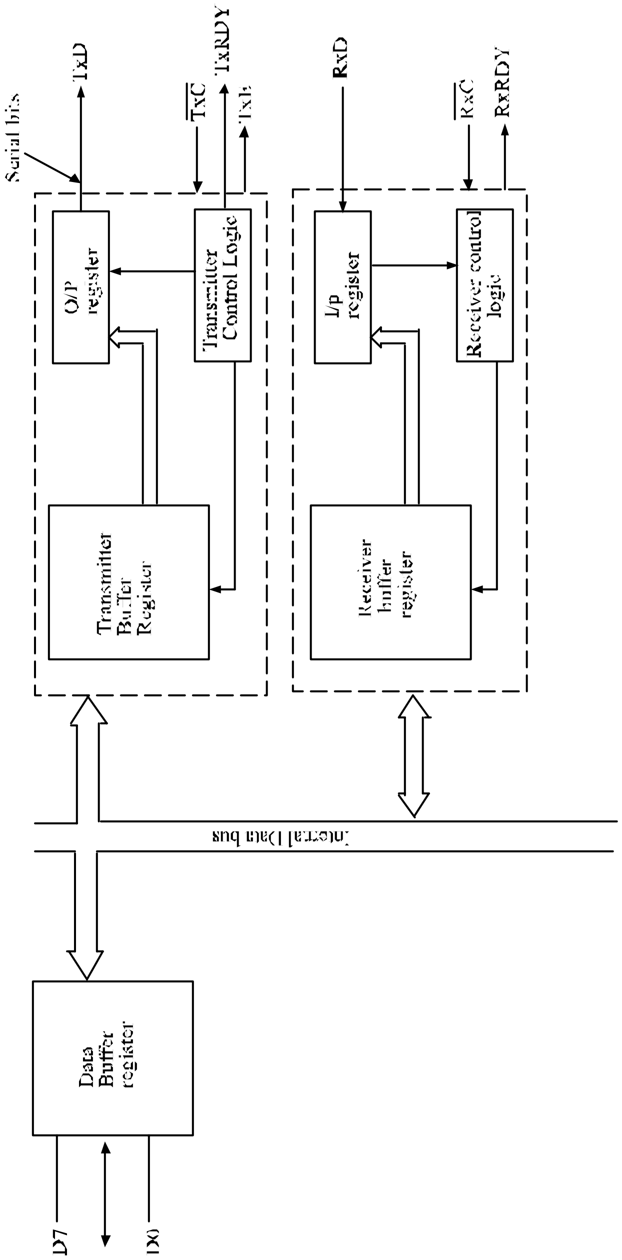
Read/ control logic and registers: This section includes R/W control logic , six i/p signals, and 3 buffer registers: data register, control register and status registers.



$\overline{CS}$ - Chip select: when signal goes low, 8251A is selected by MPU for communication.
$C/\overline{D}$ - control/ data: When this signal is high , control register or the status register is selected. When low data buffer is addressed.

$\overline{CS}$	$C/\overline{D}$	$\overline{RD}$	$\overline{WR}$	Function
0	1	1	0	MPU writes inst into control reg
0	1	0	1	MPU reads status from status reg.
0	0	1	0	MPU o/ps data to the data buffer
0	0	0	1	MPU accepts data from data buffer.
1	X	X	X	USART is not selected

Transmitter Section:



The transmitter accepts parallel data from the MPU and converts into serial data. It has two register: a buffer register to hold eight bits and an o/p register to convert eight bits into a stream of serial bits. The MPU writes a byte in the buffer register; whenever the o/p register is empty, the contents of the buffer register are transmitted to the o/p register. This section transmits data on the TxD pin with the start and stop bits.

Receiver section: The receiver accepts serial data on the RxD line from a peripheral and converts them into parallel data. This section has two registers; the receiver i/p register and the buffer register. When RxD line goes low, the i/p register accepts the data and loads it into the buffer register. Subsequently, the parallel byte is transferred to the MPU when requested.