Subject Name: Microprocessor and Microcontroller

Subject Code: EC502

Contact: 3P

Credits: 3

Prepared by: Dr. Kaushik Roy, Mr. Subhadeep Biswas

Module 1

8085 Microprocessor:

Introduction to Microcomputer based system:

Microprocessor is a digital device on a chip which can fetch instructions from a memory, decode and execute them i.e. performs certain arithmetic and logical operations, accept data from input device, and send results to output devices. Therefore, a microprocessor interfaced with memory and Input/ Output devices forms a Microcomputer. Basically, there are five building blocks of a digital computer namely:

Input Unit Through this unit data and instructions are fed to the memory of the computer. The basic purpose of this unit is to read the data into the machine. The program from the memory is read into the machine along with the input data which are required to solve or compute the problem by the machine. The typical devices which are used for this purpose are keyboards, paper tape reader and toggle switches etc.

Memory Unit The memory unit of a digital computer consists of devices which are capable of storing information. The memory of a computer is usedfor storing two distinct type of information such as data to be processed by the computer and program through which the result of the desired problem is obtained. Computer program and data are stored in the Memory Unit. This usually consists of chips of both ROMs (Read Only Memories) and RAMs (Random AccessMemories) either bipolar or MOS.
Arithmetic and This unit is used for performing arithmetic operations such as Logical Unit Addition, Subtraction, Multiplications, division and other logical (ALU) operations on the data. The control unit guides ALU which of the operations are to be performed. The sequence of the instructions is controlled by thecontrol unit.
Control Unit The control unit performs the most important function in a computer. It controls all other units and also controls the flow of data from one unit to another for performing computations. It also sequences the operations. It instructs all the units to perform the task in a particular sequence with the help of clock pulses.
Output Unit After processing of the data in the Arithmetic and Logical Unit, the results are displayed to the output world through this unit. The CRTs (Cathode Ray Tubes), LEDs (Light Emitting Diodes) and Printer etc. form the output unit.
In a computer system ALU and Control Unit are combined in one unit called Central Processing Unit (CPU). The block diagram of a computer is shown in figure 1

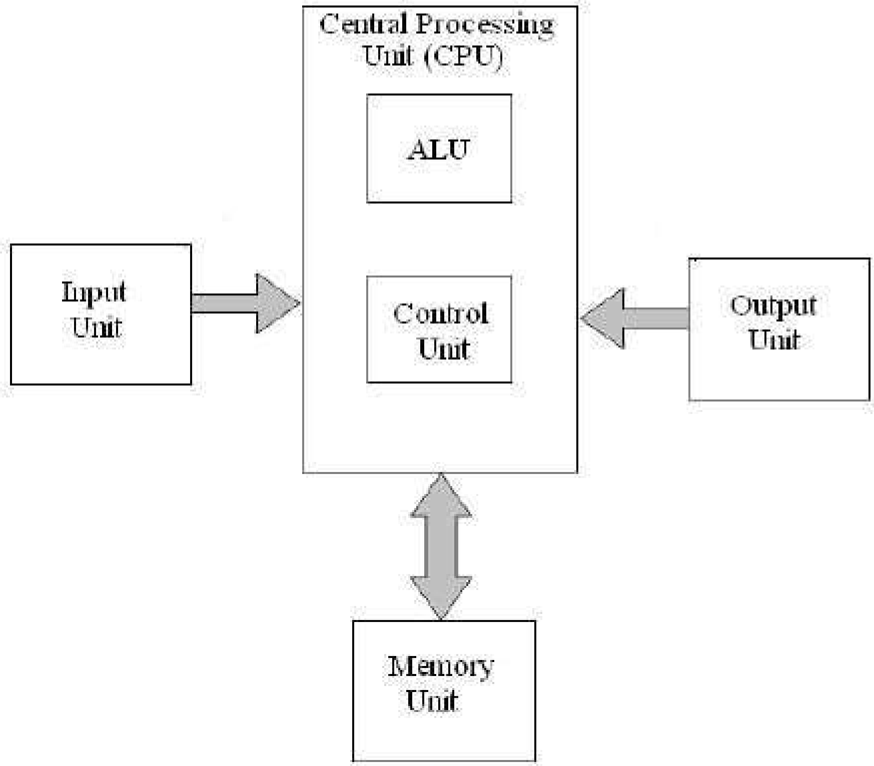


Fig. 1

The Central Processing Unit is analogous to the human brain as all the decisions as per the instructions are made by CPU. All other parts are also controlled by this unit. A microprocessor is an integrated circuit designed for use as Central Processing Unit of a computer. The CPU is the primary and central player in communicating with devices such as memory, input and output. However, the timing of communication process is controlled by the group of circuits called control unit. The term 'Microprocessor' came into existence, in 1971, when the Intel Corporation of America, developed the first microprocessor (INTEL-4004) which is a 4-bit microprocessor (μ p). A microprocessor is a programmable digital electronic component that incorporates the functions of a Central Processing Unit (CPU) on single semi-conducting Integrated Circuits (ICs). As such, a system with microprocessor as an integral part is termed as a microprocessor based system. When a computer is microprocessor based, it is called a microcomputer (μ c).

A microprocessor is specified by its 'Word Size', e.g. 4-bit, 8-bit, 16-bit etc. By the term 'word size" means the number of bits of data that is processed by the microprocessor as a unit. For example, an 8bit microprocessor performs various operations on 8-bit data. It also specifies the width of the data bus. As discussed above, a microcomputer consists of input/ output devices, and memory, in addition to microprocessor which acts its CPU. In fact CPU is commonly referred to microprocessor (μ p). Microprocessors made possible the advent of the microcomputer in the mid-1970s. Before this period, electronic CPUs were typically made from bulky discrete switching devices. Later on smallscale integrated circuits were used to design the CPUs. By integrating the processor onto one or very few large-scale integrated circuit package (containing the equivalent of thousands or millions of discrete transistors), the cost of processor was greatly reduced.

The evolution of microprocessors has been known to follow Moore's law when it comes to steadily increasing performance over the years. This law suggests that the complexity of an integrated circuit, with respect to minimum component cost, doubles in every 18 months. This dictum has generally proven true since the early 1970's. This lead to the dominance of microprocessors over every other form of computer; every system from the largest mainframes to the smallest hand held computers now uses a microprocessor as its core.

The microprocessor based systems play significant role in the every functioning of industrialized societies. The microprocessor can be viewed as a programmable logic device that can be used to control processes or to turn on/off devices. So the microprocessor can be viewed as a data processing unit or a computing unit of a computer. The microprocessor is a programmable integrated device that has computing and decision making capability similar to that of the central processing unit (CPU) of a computer. Nowadays, the microprocessor is being used in a wide range of products called microprocessor based products or systems. The microprocessor communicates and operates in the binary numbers 0 and 1, called bits. Each microprocessor has a fixed set of instructions in the form of binary pattern called a machine language. However, it is difficult for human beings to communicate in the language of 0s and 1s. Therefore, the binary instructions are given abbreviated names, called mnemonics, which form the assembly language for a given microprocessor.

Evolution of Microprocessor and microcontrollers and their advantages and disadvantages:

An English Mathematician Charles Babbage was the first man to propose the basic principle of modern computers from 1792-1871. He gave the concept of a programmable machine having computer similar to modern digital computers. He is, therefore, known Father of Modern Computers. In 1930s successful general purpose mechanical computer was developed. Before this, mechanical calculators were built to perform simple mathematical operations such as addition, subtraction, multiplication and division. Improvement continued and in 1944 Prof. H. Aiken developed a first practical electro-mechanical digital computer in collaboration with IBM. This computer was known as HAWARD MARK-I. It was in large size (51' long and 8'high) and weighing about 2 tons. The punch cards ware used to input the data in the computer. During the development of the HAWARD MARK-I Computer, Konard Joos of Germany was busy in developing another computer based on 0's and 1's rather than decimal numbers. So he developed a computer making use of relays (on-off for
1's and 0's) during 1936-44. Joos also developed a language for the computer. The giant machines during 1940-50 were thus designed using relays and vacuum tubes. In 1945, John J. Mauchy and J. Presper Eckert of University of Pennsylvania developed first electronic computer ENIAC (Electronic Numerical Integrator and Calculator). It was too huge weighing 30 tons and occupied an area 30'X50' and made use of 18000 vacuum tubes, more than 30000 resistors, 10000 capacitors and 6000 switches. It took 200 µ S for addition, and 3mS for 10-digit multiplication. It had separate memory for program and data. It used 20 electronic accumulators for memory.

Each accumulator stored signed 10-digit decimal number. A number of computers using vacuum tubes were developed during 1940-55. The main drawback with the ENIAC was the life of the vacuum tube components, which required the frequent maintenance. The invention of semiconductor transistors in 1948 at Bell Laboratories leads further development of computers. The use of semiconductor transistors could not only reduced the size of computers but also increased its capability to a great extent. This leads reduction in cost. Further, the invention of Integrated circuits in 1958 by Jack Kilby ofTexas Instrument made a revolution in electronic circuitry. The use of ICs made the sizeof computers very small and became more versatile in functions. Finally, the advent of IC technology leads to the development of first microprocessor (INTEL 4004) in 1971 at Intel = Corporation by an engineer Marcian E. Hoff. It was a 4-bit microprocessor – a programmable controller on a chip. This was called the first generation microprocessor. It was fabricated using Pchannel MOSFET technology and had an instruction set of 45 different instructions. It addressed 4096 four-bit wide memory locations. The Pchannel MOSFET technology gave low cost but low speed not compatible with TTL (Transistor5 Transistor Logic) technology. It has to use at least 30 ICs to form a system. As INTEL 4004 had very small number of instructions, it could be used in limited applications such as early video games and small microprocessor-based controllers. Seeing microprocessor as a viable product, Intel Corporation released the 8008 microprocessor – an extended 8 bit version of the 4004 microprocessor in 1972. Soon a variety of microprocessors was released by different manufactures. A few first generation microprocessors are listed in table 1.

Table 1

4-bit Microprocessors	8-bit Microprocessors
INTEL 4004 INTEL 4040 FAIRCHILD PPS-25 ROCKWELL PPS-4 NATIONAL IMP-4	INTEL 8008 NATIONAL IMP-8 ROCKWELL PPS-8 AMI 7200 MOSTEK 5065

NMOStechnology

which offered faster speed, higher density and still better reliability. In the year 1974, an 8-bit microprocessor INTEL 8080 was developed using NMOS technology. It requires only two additional devices to design a functional CPU. It is much faster than 8008 and has more instructions than 8008 that facilitates the programming. The 8080 was compatible with TTL, whereas the 8008 was not directly compatible. The 8080 microprocessor could also address four times more memory (64K bytes) than the 8008 microprocessor (16K bytes). INTEL Corporation in 1977 developed another 8bit microprocessor 8085 which was proved to be a better version than 8080. The execution time of two 8bit numbers is 2.0 μ S for 8085 whereas it is 1.3 μ S for 8080. The main advantages of the 8085 were its internal clock generator, internal system controller, and higher clock frequency. Some of the important second generation microprocessors are given in table 2.

Table 2

8-bit Microprocessors	12-bit Microprocessors
INTEL 8080 INTEL 8085 FAIRCHILD F8 MOTOROLA M6800 ZILOG Z-80 SIGNETICS 2650	INTERSIL 6100 TOSHIBA TLCS-12

The advantages of second generation microprocessor are given below:

i) Larger chip size (170 X 200 mils),
(ii) 40 pins,
(iii) More number of on-chip decoded timing signals,
(iv) Ability to address larger memory space,
(v) Ability to address more I/O Ports,
(vi) More powerful instruction set,

(vii) Faster operation,
(viii) Better Interrupt handling capabilities.

Third generation microprocessors were introduced in 1978. These were 16-bit microprocessors, designed using HMOS (High Density MOS) technology. These microprocessors offered better speed and higher packing density than NMOS. Some important third generation microprocessors are given in table 3.

Table 3

16-bit Microprocessors		
INTEL 8086 INTEL 8088 INTEL 80186 INTEL 80286	MOTOROLA-68000 MOTOROLA-68010 NATIONAL NS-16016 TEXAS INSTRUMENT-TMS-99000	INTERSIL 6100 TOSHIBA TLCS-12 ZILOG Z-8000

In 1978, 16-bit INTEL 8086 microprocessor of 64 pins was introduced and in 1979 other 16-bit microprocessor 8088 was developed. In addition to the other performances, these µPs contain multiply/divide/arithmetic hardware. The memory addressing capabilities has been increased to very large i.e., 1MB to 16MB through a variety of flexible and powerful addressing mode. The other characteristics of third generation are given below: (i) These microprocessors were 40/48/64 pins, (ii) High speed and very strong processing capability,

(iii) Easier to program,
(iv) Allow for dynamically re-locatable programs,
(v) Size of internal registers were 8/16/32 bits,
(vi) These µPs had the multiply/divide/arithmetic hardware,
(vii) Physical memory space was from 1 to 16 Mega-bytes (MB),
(viii) Flexible 10 port addresses,
(ix) More powerful interrupt and hardware capabilities, (x) Segmented address and virtual memory features.

Fourth generation microprocessors of 32 bits were introduced in the form of 80386 in 1985 and 80486 in 1989. The instruction set of the 80386 microprocessor was upward compatible with the earlier 8086, 8088 and 80286 microprocessors. However, the 80486 is an improved version of the 80386 microprocessor. The 80386 executes many instructions in 2 clock cycles while the 80486 executes in one clock cycle. These microprocessors are of low power version of HMOS technology.

Some important fourth generation microprocessors are given in table 4.

Table 4

32-bit Microprocessors	
INTEL 80386 INTEL 80486 NATIONAL NS16022 MOTOROLA MC 88100	MOTOROLA M-68020 MOTOROLA M-68030 BELLMAC-32

Fifth generation microprocessor was introduced by INTEL Corporation in 1993 in the form of PENTIUM with 64 data bus. The Pentium was similar to the 80386 and 80486 microprocessor. The two introductory versions of the Pentium operated with a clock frequency of 60 MHz and 66 MHz and a speed of 110 MIPS (Million Instructions Per Second). With better and more advanced technologies, the speed of µPs has increased tremendously. The old 8085 of 1977 executed 0.5 million instruction/sec. (0.5 MIPS), while the 80486 executes 54 million instruction per sec. The Pentium Pro Processor is the Sixth generation microprocessor introduced in 1995 having better architecture but more in size. The Pentium Pro Microprocessor contains 21 million transistors, 3 integer units as well as a floating unit to increase the performance of most software. The basic clock frequency is 150 MHz and 166 MHz.

Architecture of 8085 Microprocessor:

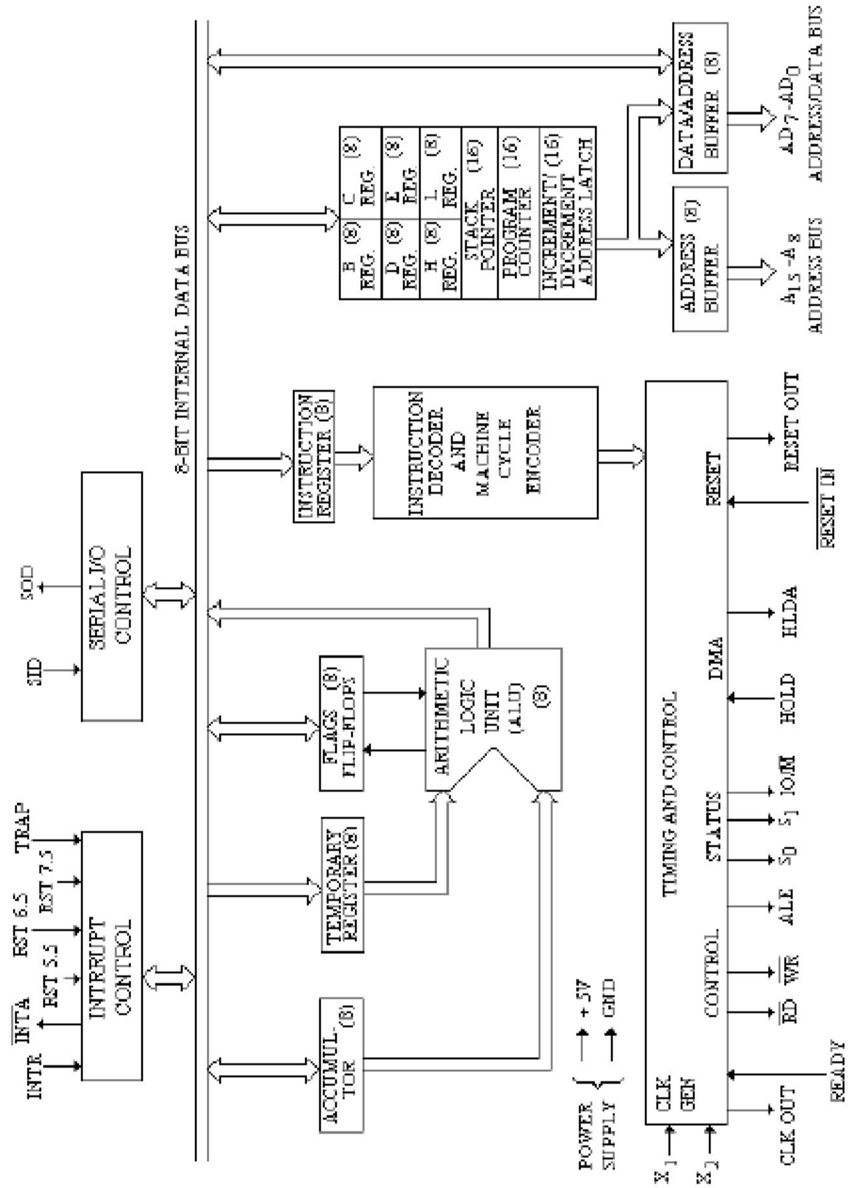
Intel 8085, an 8-bit NMOS microprocessor is available in the form of 40 Pin dual in line IC package. It is fabricated on a single LSI chip. It operates on +5 V d.c. supply. The clock speed used in this microprocessor is about 3 MHZ. General Purpose 8-bit microprocessor is capable of addressing up to 64 K bytes (i.e. 2^{16} □□65536 bytes) of memory. The functional block diagram is shown in figure 5.1. The main functional components of 8085A microprocessor are as given below: (i) Register Section

(ii) Arithmetic and Logic Unit

(iii) Timing and Control Section

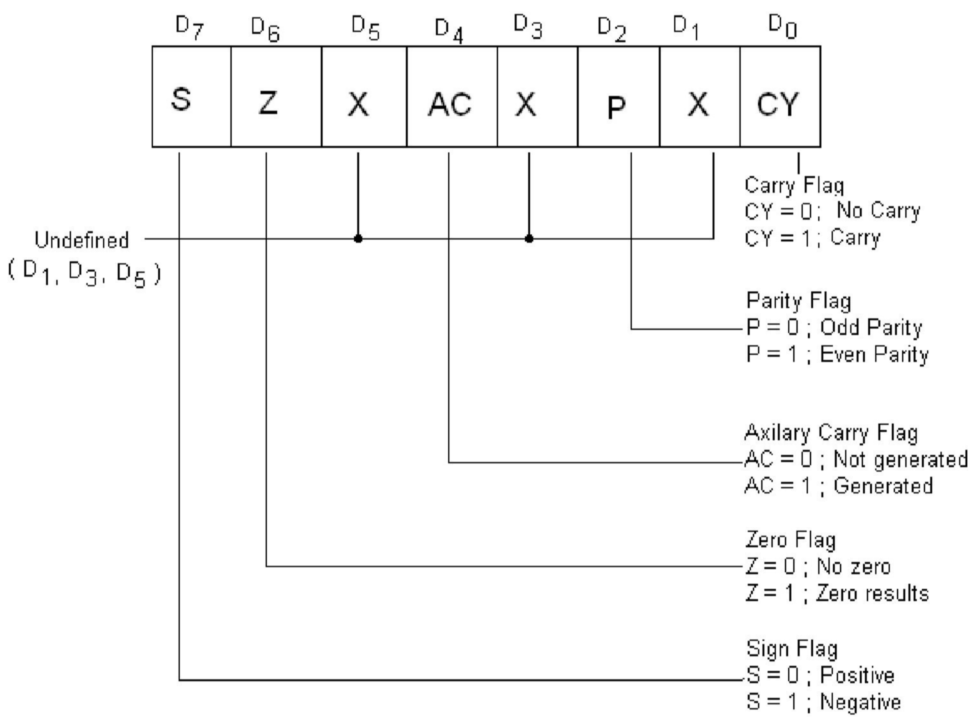
(iv) Interrupt Control

(v) Serial Input / Output Control



Register Section
The 8085 microprocessor contains eight addressable 8-bit registers namely:
A (Accumulator) register
F Flag register (Flag flip-flops)
B register
C register
D register
E register H register
L register
Out of these registers B, C, D, E, H and L registers are 8-bit general purpose registers. These registers can either be used as single register or a combination of two registers as 16 bit register pair.

As discussed in SAP-III computers, the valid register pairs are B-C, D-E or H-L register pairs. The higher order byte of 16 bit data is stored in first register (B in B-C register pair), and low order byte in the second register (C in B-C register pair).
The H-L register pair can also be used for register indirect addressing; since this register pair can also function as data pointer.
The large number of general purpose registers gives more flexibility and ease in the programming. However, the general purpose registers are limited as registers occupy more space on the silicon chip. Beside these general purpose registers, the 8085 has remaining two 8-bit registers Accumulator (A) and Flag (F) as special purpose registers and two 16 bit registers namely Program counter (PC) and stack pointer (SP). Accumulator (A)
As discussed in preceding chapters, accumulator is a 8 bit buffer register extensively used in arithmetic, logic, load and store operations as well as in input / output instructions. All the arithmetic and logical operations are performed on the accumulator contents; i.e. one of the operand is always taken into the accumulator. Flag (F) Register
It is an 8-bit register associated with the execution of instructions in the microprocessor. Out of the 8 bits of flag register, 5 bits contains significant information in terms of status flags. The five flags are:
(i) Sign flag (S)
(ii) Zero flag (Z)
(iii) Carry flag (CY)
(iv) Parity flag (P)
(v) Auxiliary Carry flag (AC)
All the flags except the Auxiliary carry (AC) flag have been discussed in SAP-III computers. The bit positions reserved for these flags in the flag register (F) is shown in figure



(i) Sign flag (S) The sign flag is set (S = 1), if the result of the operation of the instruction is negative (MSB of the result is 1); otherwise it is reset (S = 0) for the positive result (MSB is zero).
(ii) Zero flag (Z) The zero flag is set (Z = 1) if the result of the operation of the instruction is zero otherwise this flag is reset (Z = 0). i.e. Z = 1 if the result is zero, and Z = 0 if the result is not zero. (iii)

Carry flag (CY) The carry flag is set to 1, if there exist a carry (or borrow) to the highest order bit (non-existent 9_{th} position) as a result of the execution of addition or subtraction instructions. If there is no carry (or borrow) to the higher order bit, the carry flag is reset. i.e. CY = 1 if there is a carry to the highest order bit (or overflow), and CY = 0 if there is no carry to the highest order bit (or no overflow).

(iv) Parity flag (P) After an arithmetic and logic operation, if the result has even number of 1s, then parity bit is set. If on the other hand the result has odd number of 1s, the parity flag is reset. i.e. P = 1, if the result has even number of 1s, and P = 0, if the result has odd number of 1s.

(v) Auxiliary carry (AC) This is a new flag in 8085 microprocessor. This flag (AC) is set to 1, if there is an overflow at bit 3 of the accumulator. AC flag is used in BCD arithmetic. This is illustrated as given below:

As shown in figure 5.2, the five bits in flag register are defined. The three bits are undefined. The Accumulator and 8 bits (including three undefined bits) of flag register form a Program Status Word (PSW). The accumulator and flag registers are treated as a 16 bit unit for stack operation.

Program Counter – 16 bit register

The program counter is a 16 bit register. It is used to send 16 bit address to fetch the instruction from the memory. It acts as a pointer which indicates the address of the next instruction to be fetched and executed. The program counter is updated after an instruction has been fetched by the processor. If an instruction is one byte instruction, then the program counter will be updated by one (i.e. PC = PC + 1). Similarly, for two and three byte instructions, the program counter will be updated by two (i.e. PC = PC + 2) or three (i.e. PC = PC + 3) locations respectively.

Stack Pointer – 16 bit register

The stack is an area of RAM (random access memory or read / write memory) in which temporary information is stored. It is stored on First-In-Last-Out (FILO) basis. An address in the RAM area is assigned to the stack pointer where the first information is stored as the first stack entry. This is done by initializing stack pointer by an instruction. Higher stack entries are made at the progressively decreasing addresses.

5.1.2 Address Buffer and Address-Data Buffer

It has already been discussed that 8085 requires 8-bit data bus and 16-bit address bus, as the memory address is of 16 bits. More number of IC pins are required if separate address and data bus are introduced. To restrict the number of pins of 8085 to only 40, lower address lines A_0-A_7 and data lines D_0-D_7 are used in multiplexed mode. The multiplexed lines are designed as Address/Data Bus (AD_0AD_7). So whenever 16-bit address is transmitted by the microprocessor 8 MSBs of the address lines are sent on the Address Bus (A_{15}-A_8) and 8 LSBs of the lines are sent on the Address/Data Bus (AD_7- AD_0). The 8 LSBs of the address are then latched either into memory or external latch so that the complete address remains available for further operation. The 8-bit Address/Data Bus will now be free for the data transmission.

5.1.3 Arithmetic and Logical Unit (ALU)

The arithmetic and logical unit (ALU) basically consists of accumulator (A), flag register (F) and a temporary register (which is inaccessible by the programmer or user). This unit carries out the arithmetic and logic calculations of the data stored in general purpose registers or in memory locations. The arithmetic operations are ADD, SUB, compare, increments, decrements and complements etc.; while logical operations are AND, OR, XOR and Rotate. The result of these operations could be placed in the accumulator or elsewhere through the internal bus. Arithmetic operations are usually carried out in 2's complement adder / subtrator discussed in the preceding chapter. For these operations, ALU receives the control signals from the timing and control unit.

5.1.4 Timing and Control Unit

This unit consists of the following sections:
1. Instruction Register and Decoder
2. Control signals
1. Instruction Register and Decoder
As discussed in the preceding chapter, the CPU fetches an instruction from the memory for its execution. This instruction can be of 1-3 byte long. The first byte contains the op code of instruction which basically specifies the nature of operation to be performed indicating the length of the instruction. The first byte (op code of the instruction) transferred to 8-bit instruction register through the internal bus of the CPU, becomes available at the instruction decoder. The decoder decodes the op code and directs the control unit to produce the necessary control signals.
2. Control Signals
Following are the control signals of 8085 microprocessor needed for the operation of CPU.
(i) X_1, X_2 and CLK Out
Two pins X_1 and X_2 are provided to be externally connected to a quartz crystal. The clock signal of fixed frequency is generated through the internal circuitry of the processor. The frequency at which the microprocessor 8085 works is half of the crystal frequency. The quartz crystal of 6.144 MHz is used in this processor. This gives the clock frequency of 3.072 MHz (half of the crystal frequency) of 50% duty cycle. The clock period is of about 320 nsec. The output of the clock frequency is also available at CLK out terminal. (ii) Address Latch Enable (ALE)
The 16 bit address bus is basically divided into two sets. The most significant bits A_7-A_{15} of the address bus are used separately and the least significant bits of the address AD_0-AD_7 are time multiplexed with the bits of bidirectional data bus (D_0-D_7). The AD_0- AD_7 bus serves the dual purpose as they can be used as low-order address bus as well as bidirectional data bus at different times. This is used as address bus, during the first clock cycle of the machine cycle involving memory; and during the remaining clock cycle of the machine cycle, it acts as the data bus. This is accomplished by address latch enable (ALE) signal provided in the processor. During the first clock cycle of the machine cycle ALE is high which enables the lower 8-bit of the address to be latched either into the memory or external latch. (iii) D R (Read) Signal

This is an active low signal to be connected to memory read input (output enable signal to memories) or to input / output read signal to enable input / output buffer.
(iv) R W (Write) Signal
Similar to read signal (D R), write signal (R W) is also active low. This signal is used to write to the memory or input / output devices.
(v) IO/ M (Input Output / Memory)
This signal IO/Mdistinguishes that the address and data is meant for either I/O devices or memory. Whenever this signal is high (1), microprocessor will communicate to the I/O devices and whenever it is low (0), microprocessor will communicate to the memory.
(vi) Status Signals (S_0, S_1)
The status signals (S_0, S_1) along with IO/M signal indicate the type of machine cycle in progress. The type of machine cycle are op code fetch cycle, memory read cycle, memory write cycle, I/O read cycle or I/O write cycle.
Various types of status codes are given in table

Machine Cycle	$IO/\overline{M}$	S_1	S_0
Op code fetch Cycle	0	1	1
Memory Read Cycle	0	1	0
Memory Write Cycle	0	0	1
I/O Read Cycle	1	1	0
I/O Write Cycle	1	0	1
INTR Acknowledge	1	1	1
Halt	Hi-Z	0	0

(vi) Hold and HLDA

HOLD and HLDA (Hold Acknowledge) signals are used for DMA (Direct Memory Access) operation. In a microprocessor, the data transfer between I/O devices and memory will take place through the microprocessor. The involvement of the processor slows down the data transfer between I/O devices and memory. The transfer of data directly from I/O devices to memory without involvement of microprocessor is called DMA. The DMA will save the time as CPU relinquishes the control of Buses. In this way DMA transfers the large amount of data in a relatively short time. The HOLD and HLDA signals are used in the operation. Whenever HOLD signal is high, CPU temporarily relinquishes its operation by floating the address, data and control buses; and DMA operation is started. A high HLDA (Hold Acknowledge) signal is also sent to DMA controller, indicating that CPU has received the hold request. Whenever the data transfer is complete, then the control to CPU is returned back by sending a HOLD signal.Further the HLDA signal goes low. (viii) READY signal (Input)

Some peripheral devices connected to 8085 microprocessor operate at much slower speed than the processor. To synchronize the speed of CPU and peripheral devices or to slow down the speed of 8085, the READY signal is used. If the READY signal is high the peripheral device is ready and the processor can complete the data transfer. If this signal is low the microprocessor waits (by generating a number of NOP T-states) till it goes high.

(ix) and RESET OUT

The signal may be low from the operator Reset button or from the processor. When the signal is low, the CPU will reset the program counter, instruction register and other circuits. It also sends a high RESET OUT. The RESET OUT signal goes to peripheral devices to reset or initialized. When signal goes high and

RESET OUT goes low, the data processing may begin.

Interrupt Control

Sometimes it is necessary to interrupt the execution of the main program. For this an interrupt request is obtained from the I/O devices. After receiving the interrupt request (INTR), processor temporarily stops what it was doing and attends to the I/O device.INTA is an interrupt acknowledge signal which is sent by the microprocessor after INTR signal is received. After the work of the I/O device is complete it returns to what it was doing earlier. Basically 8085A has five hardware interrupts namely: INTR

RST 5.5

RST 6.5

RST 7.5 and

TRAP If two or more of these interrupts are active at the same time, the 8085 takes them in order of priority level. The priority levels of these interrupts are given in table 5.3

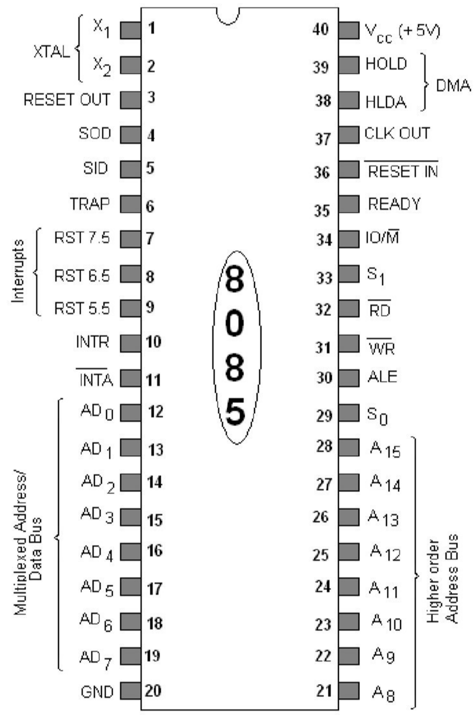
Interrupts	Priority
TRAP	1
RST 7.5	2
RST 6.5	3
RST 5.5	4
INTR	5

Serial I/O Control

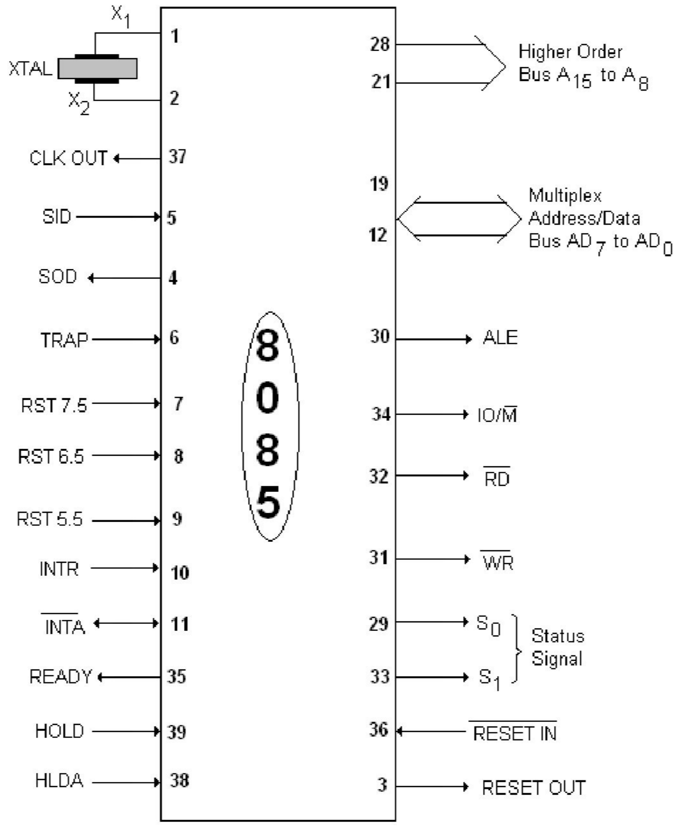
Serial input / output control circuit incorporated in this microprocessor is used for the data transmission. For this purpose two pins SID and SOD are provided in the serial input/output control unit. The SID (Serial Input Data) terminal receives the serial data stream from an input device, the control unit converts serial data stream to parallel data before it is used by the computer. After the conversion 8-bit parallel data is stored in the accumulator. Similarly, SOD (Serial Out Data) terminal outputs the 8-bit parallel available with the accumulator into serial form to the peripheral device connected with the computer

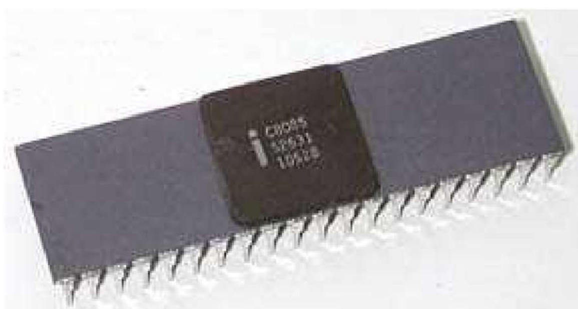
PIN DESCRIPTION OF 8085

The pin details and logical schematics of the 40 pin dual line package (DIP) IC 8085 are shown in figures 5.3(a) and (b) respectively. Fig. 5.3 (c) shows the shape of the microprocessor. The descriptions of various pins of the microprocessor are given below:



8085
XTAL
X1
X2
RESET OUT
SOD
SID
TRAP
Interrupts
RST 7.5
RST 6.5
RST 5.5
INTR
INTA
Multiplexed Address/ Data Bus
AD0
AD1
AD2
AD3
AD4
AD5
AD6
AD7
GND
Vcc (+5V)
HOLD
HLDA
DMA
CLK OUT
RESET IN
READY
IO/M
S1
RD
WR
ALE
S0
A15
A14
A13
A12
A11
A10
A9
A8
Higher order Address Bus
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
40
39
38
37
36
35
34
33
32
31
30
29
28
27
26
25
24
23
22
21



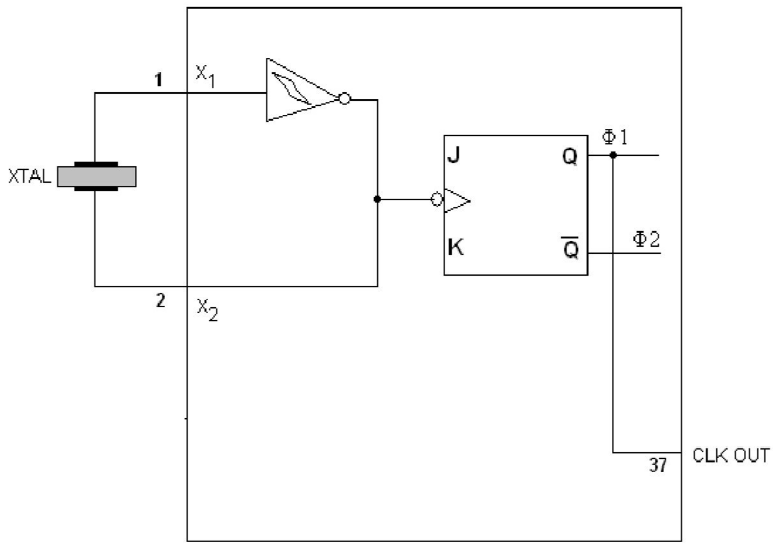


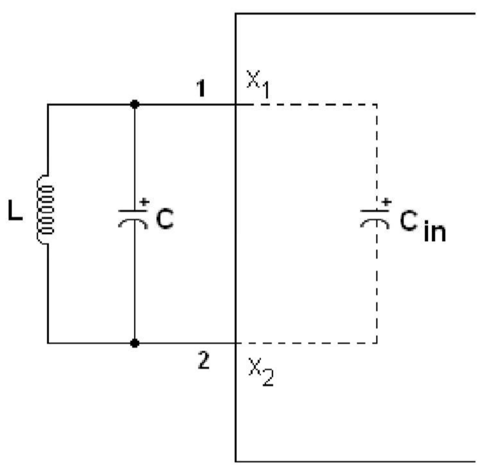
PIN NOS. 1 and 2:

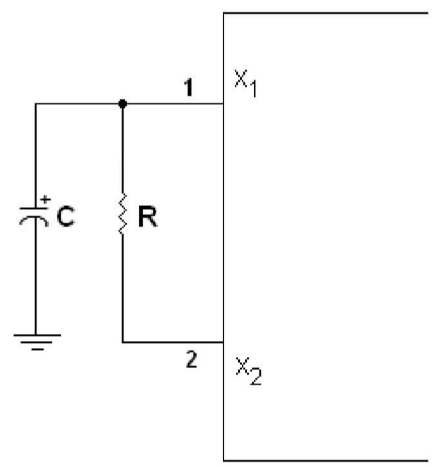
These X_1 and X_2 pins are to be connected to an external quartz crystal, L-C or R-C network which drives the internal clock generator. The clock signal of appropriate frequency is determined when a quartz crystal is connected to the on-chip oscillator as shown in figure 5.4(a). The oscillator output from the Schmitt trigger drives a flip-flop which divides the frequency by a factor of two. The circuit produces two clock signals Φ1 (CLK) and Φ2 (CLK) to derive the internal circuit of the microprocessor. A 6.25 MHz crystal is used to provide 3.125 MHz internal clock frequency. Generally, quartz crystal is used for the On-chip oscillator for the accurate and stable clock frequency, though a parallel resonant L-C circuit may be used for the frequency determining network as shown in figure 5.4(b). The network produces a signal whose frequency tolerance is about □10%. The component values may be chosen from the following formula:

$$f = \frac{1}{2\pi\sqrt{L(C + C_{in})}}$$

The input capacitance Cin is approximately 15 Pf. To minimize the variations in frequency, it is recommended to choose C as 30 Pf.







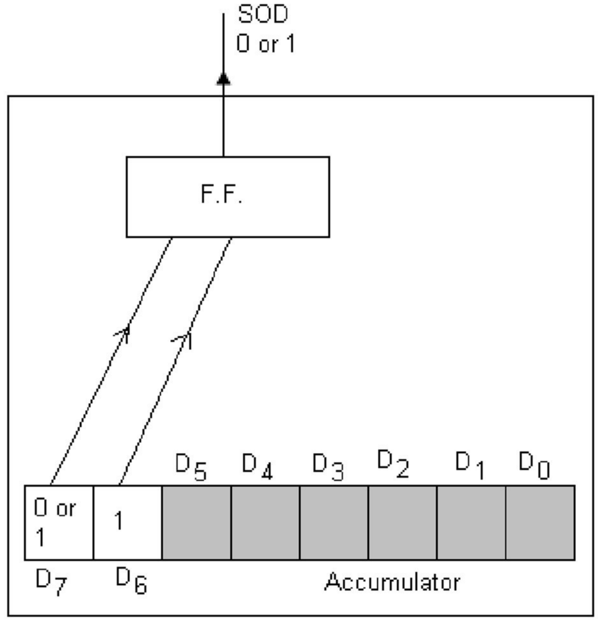
An R-C network may also be used as the frequency determining network for the on-chip oscillator of the microprocessor as shown in figure 5.4(c). The driving frequency generated by this circuit is approximately 3MHz. It is not recommended to use the frequencies higher or lower than this.

PIN NO. 3

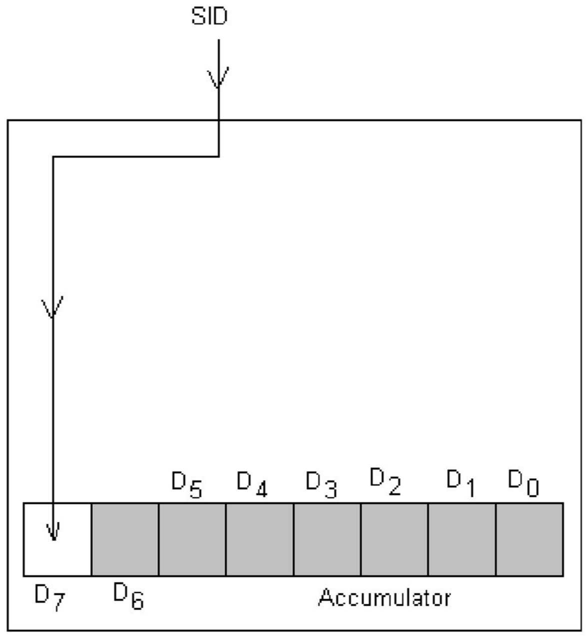
This is RESET OUT signal, which indicates that CPU is being reset. When it is high, system is reset. The signal is synchronized to the processor clock and lasts for an integral number for clock periods. When the RESET OUT signal goes low, the processing begins.

PIN NOS. 4 and 5

Pin Nos. 4 and 5 indicate SOD (Serial Out Data) and SID (Serial In Data) terminals respectively. These pins are associated with Serial Input/Output control unit for 8085 microprocessor. As already discussed these pins are used for the serial data transmission. The SOD output pin can deliver a serial data stream to a peripheral device. On executing SIM (Set Interrupt Mask) instruction, if bit D_6 is set to 1, the content of D_7 bit (set or reset) of the accumulator is latched on the SOD pin as shown in figure 5.5(a).



The data on the SID line (PIN 5) loads into accumulator at bit D_7 whenever a RIM instruction is executed as shown in figure 5.5(b).



,

PIN NOS. 6 to 11

The interrupt control unit of the microprocessor contains these pins. The Pins 6 to 11 are restart interrupts named as: TRAP (Pin No.6) I Priority

RST 7.5 (Pin No. 7) II Priority

RST 6.5 (Pin No. 8) III Priority

RST 5.5 (Pin No. 9) IV Priority

INTR (Pin No. 10) V Priority

The TRAP has the highest priority and INTR has the lowest priority. The priority level is of importance if two or more interrupts become active at the same time. The TRAP is non-maskable interrupt. It is both edge and level sensitive.

The interrupts (TRAP, RST 7.5, RST 6.5 and RST 5.5) are also called vector interrupts, as each interrupt has fixed memory location (vector location) for the transfer of control from the normal execution of the routine. The vector locations of these interrupts are given in table 5.4. As soon as any of these pins 6 to 10 are active (high), the internal circuit of 8085 stops the normal execution of program and the program control is transferred to the corresponding memory location (vector location).

Interrupts	Memory locations
TRAP	0024 H
RST 7.5	003C H
RST 6.5	0034 H
RST 5.5	002C H

INTR (Pin No. 10) is a general purpose interrupt and has the lowest priority. As soon as Pin No. 10 is high, the microprocessor stops the execution of normal program and after completing the instruction at hand, it goes to CALL instruction. The INTR is enabled or disabled by the instructions ET (Enable Interrupts) or DI (Disable Interrupts) respectively.

The Pin No. 11 is an Interrupt Acknowledge (INTA) signal. A low (logic 0) to this pin indicate that the microprocessor has acknowledged the request from the peripheral device. It is also used to activate the interrupt controller.

PIN NOS. 12 to 19

Pin Nos. 12 to 19 (AD_0-AD_7) form bi-directional multiplexed Address/Data Bus. The least significant 8 bits of the memory address (or I/O Address) appear on the bus during the first T-states of a machine cycle. It then becomes the data bus during the next T-states. PIN NO. 20

Pin No. 20 is the ground terminal.

PIN NOS. 21 to 28

The Pin Nos. 21 to 28 (A8-A15) form unidirectional most significant 8 bits of memory address or 8 bits of the I/O address. It remains in the high impedance state during HOLD, HALT and RESET modes.

PIN NOS. 29 to 33

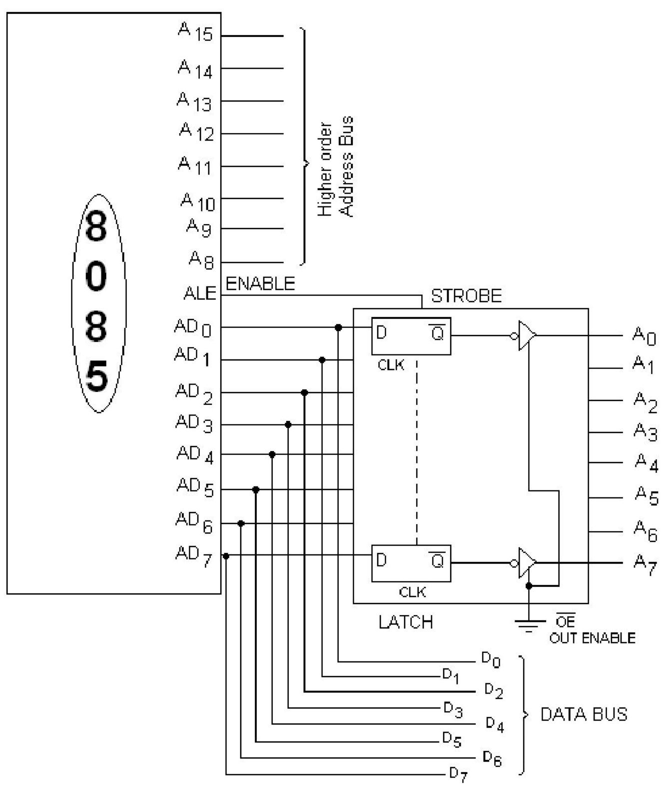
The Pin Nos. 29 to 33 labled as S_0 and S_1 respectively are known as status signals. These status signals along with IO/M signal indicate the various operations as indicated in table 5.5.

Machine cycle	$IO/\overline{M}$	Status S_1	S_0	Control signals
Op code Fetch	0	1	1	$\overline{RD} = 0$
Memory Read	0	1	0	$\overline{RD} = 0$
Memory Write	0	0	1	$\overline{WR} = 0$
I/O Read	1	1	0	$\overline{RD} = 0$
I/O Write	1	0	1	$\overline{WR} = 0$
Interrupt Ack.	1	1	1	$\overline{INTA} = 0$
HALT	HI-Z	0	0	
HOLD	HI-Z	X	X	$\overline{RD}, \overline{WR} = Z$
RESET	HI-Z	X	X	$\overline{INTA} = 1$

HI-Z = High Impedance State
X = Unspecified

PIN NO. 30

The Pin No. 30 is known as ALE (Address Latch Enable) terminal. When this signal is high the information carried on the multiplexed address/data bus (AD_0-AD_7) is the lower 8 bits of the address. It also enables the low order address (AD_0-AD_7) from the multiplexed address/data bus to latch either into the memory or the external latch. The ALE signal separates the low order address and data from the multiplexed Address/data Bus. This is illustrated in figure 5.6.



PIN NOS. 31, 32 and 34

The Pin Nos. 31 and 32 are the two control signals WR (Write bar) and RD (Read bar) respectively. The pin 34 carries IO/M signal which is one of the status signals. The other status signals are S0 and S1 discussed earlier. A low WR signal generated by the microprocessor sends (writes) data into I/O devices or memory. Similarly, a low RD signal generated by the microprocessor reads (receives) the data from the I/O devices or memory locations. The IO/M signal indicates whether the address on the address bus is meant for I/O devices. However, a low to this signal indicates that the address on the address bus is meant for memory location. The RD ,WR and IO/M signals function together. PIN NO. 35

The Pin No. 35 is known as READ signal which forces the microprocessor to wait till the data become available from the memory or input/output devices. This signal is needed to synchronize the speed of the microprocessor with I/O devices or memory as the memory or I/O devices are not as fast as the microprocessor. When the READ signal is low, the microprocessor waits till the READY signal is 1. As soon as READY signal is 1, the microprocessor knows that the data are available from the memory or I/O devices. PIN NO. 36

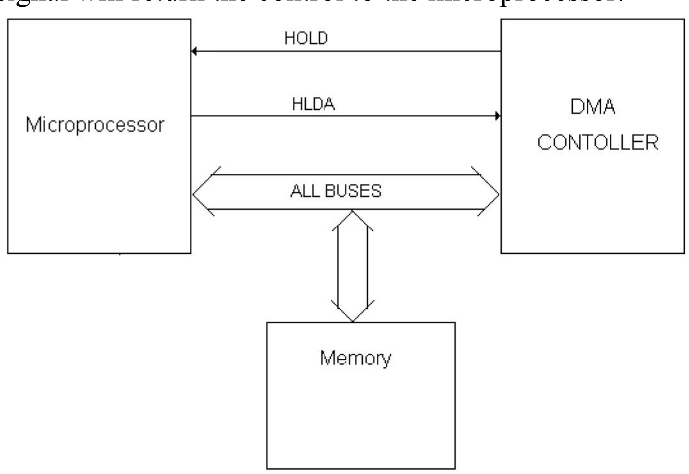
This pin is signal. This input carrying signal may be operated by the operator using the RESET button provided externally or it may be operated directly from the other source. When this signal is low (momentarily), the CPU will reset the program counter, instruction register, all interrupts (except TRAP) are disabled, SOD signal becomes low and Data, address and control buses are floated. When this signal goes high, the data processing begins.

PIN NO. 37

This pin carries CLK OUT signal. It is derived from the on-chip oscillator, which goes to peripherals to synchronize their timings.

PIN NOS. 38-39

The Pin Nos. 38 and 39 are the HOLD and HLDA (Hold Acknowledge) signals respectively. These signals are used in DMA (Direct Memory Access) operations. As shown in figure 5.7, when any I/O device indicates that the data are ready for DMA transfer, a high HOLD signal is sent by the DMA controller to the 8085 microprocessor. It is in fact a request signal from the DMA controller to the microprocessor. The microprocessor then sends a high signal to DMA controller indicating that the microprocessor has received the request from the I/O devices and will relinquish the address, data and control bus after completing the current instruction. The DMA controller thus carries out the data transfer. A low HOLD signal will return the control to the microprocessor.



PIN NO. 40

The pin 40 is +VCC, which is to be externally connected to +5 volt d.c. supply.

INSTRUCTION SET OF 8085 MICROPROCESSOR

The 8085 includes all the instructions of SAP-III. In addition there are few more instructions which will be discussed below. These new instructions were not considered in SAP-3 because of its architecture.

LHLD address (Loads the H-L pair direct)

This instruction loads the H-L pair direct with two bytes already stored in two consecutive memory locations starting at the specified memory address. The contents stored in the memory location whose address is given with the instruction will be loaded to the L-register; and the contents stored in the next memory location (address + 1) will be loaded to the H-register. i.e. $[L] \leftarrow [M_{address}]$ and $[H] \leftarrow [M_{address+1}]$

For example, let 2A H is stored in the memory location 2100 H and 2B H is stored in the memory location 2101 H, then after the execution of the instruction LHLD 2100 H , the L-register will have 2A H and H-register will have 2B H. None of the flags is affected with this instruction.

SHLD address (Stores the H-L pair direct)

This instruction does the reverse operation of LHLD. The instruction SHLD address stores the contents of L-register to memory location whose address is given with the instruction; and the contents of Hregister are stored in the next consecutive memory location (address + 1). i.e. $[M_{address}] \leftarrow [L]$ and $[M_{address+1}] \leftarrow [H]$

No flag is affected with this instruction too.

For example, if $[L] = 3A$ H and $[H] = 3B$ H, then after the execution of the instruction SHLD 2200 H will result. $[M_{2200\,H}] = 3A$ H and $[M_{2201\,H}] = 3B$ H

LDAX rp (Loads the Accumulator Indirect)

This instruction loads the accumulator, the contents already stored in the memory location addressed by the register pair (rp). Here rp represents B-C or D-E register pair. The H-L register pair is not included in this instruction. i.e. $[A] \leftarrow [M_{[rp]}]$

The possible combinations of the instruction are:

LDAX B

LDAX D

No flag is affected with the execution of this instruction. For example, if [D] = 25 H, [E] = 00 H and $M_{2500\,H}$ = 34 H, then after the execution of the instruction LDAX D, the accumulator will have:

$[A] \leftarrow [M_{2500H}]$ i.e. A = 34 H

It is worth while to mention that the instruction LDAX H does not exist, because the contents stored in the memory location addressed by H-L register pair may be loaded to accumulator by the instruction MOV A, M.

STAX rp (Stores the Accumulator Indirect)

The STAX rp instruction does the reverse operation of LDAX rp. This instruction stores the accumulator contents in the memory location addressed by the register pair (rp). Here too rp represents B-C or D-E register pair. The H-L register pair is not included in this instruction. i.e. □ □ □ □ □A□ □$_{rp}$ M

The possible combination of this instruction are:

STAX B

STAX D

No flag is affected with the execution of this instruction. For example, if B = 21 H, C = 00 H and A = 3A H, then after the execution of the instruction STAX B, 3A H will be stored in the memory location 2100 H.

i.e. □M □ □3AH $_{2100H}$ □ □

The combination STAX H is not included in this instruction as MOV A, M performs the same operation.

XCHG (Exchange the contents of H-L register with D-E register)

This is one byte instruction and no operand is needed with it. It exchanges the contents of H and L register with D and E registers respectively. i.e. □H□ □ □D□ □and □L□ □ □E□ □

For example: If H = 25 H , L = 32 H and D = 12 H, E = 1B H then after the execution of XCHG instruction, we have: H = 12 H, L = 1B H and D = 25 H, E = 32 H

The instruction XCHG is generally used to keep track of more than one memory location at a time without using LDAX and STAX instructions.

Let us write a program to add two numbers stored in memory locations 2100 H and 2201 H without using LDAX and STAX instructions. The answer is to be loaded in the memory location 2100 H.

LXI H, 2201 H ; Loads H = 22 H and L = 01 H

LXI D, 2100 H ; Loads D = 21 H and E = 00 H

MOV A, M ; □ □ □ □ □ □ □ □ □$_{H}$ A M$_{2201}$ □ □

XCHG ; H = 21 H, L = 00 H and D = 22 H, E = 01 H

ADD M ; □ □ □ □ □ □ □ □ □ □ □ □ □$_{H}$ A A M$_{2100}$ □ □ □ □

MOV M, A ; □M □ □ □A□ □$_{H}$

□ □$_{2100}$ HLT

In this program no LDAX and STAX instructions are used. DAA

(Decimal Adjust the Accumulator)

The DAA is one byte instruction and no operand is needed with this instruction. It adjusts the accumulator to packed BCD (Binary Coded Decimal) after addition of two BCDs. In other words, after addition of two hexadecimal numbers if this instruction is used then the result in decimal form is obtained. For this Auxiliary Carry Flag (AC) and Carry Flag (CY) take care of this instruction. It functions in two steps:

1. If the lower nibble (lower 4-bits) of the accumulator is greater than 9 or Auxiliary carry flag is set, then it adds 06 H to the accumulator.
2. Subsequently, if the higher nibble (higher 4-bits) of the accumulator is now greater than 9 or the carry flag (CY) is set, it adds 60 H to the accumulator. All the flags are affected with this instruction.

PCHL (Copies H-L to PC)

This is one byte instruction and no operand is needed with this instruction. It copies the contents of Hregister to high-order byte of the program counter (PC) and the contents of L-register to low order byte of the program counter.

□PC□ □ □HL□ □i.e. □PCH□ □ □H□ □and □PCL□ □ □L□ □

For example if PC = 2106 H and HL = 2500 H then after the execution of the instruction PCHL will result:

PC = 2500 H

No flag is affected with the instruction.

This instruction is basically an unconditional jump instruction as is evident from the above example.

SPHL (Copies HL to Stack Pointer SP)

This is also one byte instruction as no operand is used. The SPHL instruction copies the contents of Hregister to high order byte stack pointer (SP) and the contents of L-register to low order byte of stack pointer (SP).

□SP□ □ □HL□ □i.e. □SPH□ □ □H□ □and □SPL□ □ □L□ □

None of the flags is affected with this instruction. For example if H = 23 H and L = 45 H and SP = 2501 H then after the execution of the instruction SPHL will result: SP = 2501 H

This is another way of initialization the stack pointer.

XTHL (Exchanges the top of the stack with H-L register pair)

The XTHL is one byte instruction and does not require any operand. The top byte of the stack is exchanged with L-register and next byte of the stack is exchanged with Hregister. i.e.

□□□□□□□□SP L □□M and □□□□□□□□SP 1 H M □□□□ For example if H = 21 H L = 02 H and M_{SP} =1A H M_{SP+1} = 2C H as shown in figure 5.8(a), then after the execution of the instruction XTHL will result: H = 2C H L = 1A H and M_{SP} = 02 H M_{SP+1} = 21 H as shown in figure 5. 8(b). No flag is affected with the instruction.

TIMING DIAGRAM FOR 8085 INSTRUCTIONS

The working of 8085 microprocessor can best be understood by considering the timing diagrams of its few instructions. The graphical representation depicting the necessary steps carried out in a machine cycle is known as Timing Diagram. As is well known that the total time required for the execution of an instruction is the time required to fetch and execute an instruction. i.e. Instruction Cycle = Instruction fetch + Instruction execution

The instruction cycle may be of one, two or three bytes. During the fetch cycle, an instruction of the program (op code) is extracted from the memory locations and copied in the instruction register (IR) of C.P.U.

The op code of the instruction copied in the instruction register (IR) is decoded during the execution cycle to perform the specific activities.

It may further be mentioned that an instruction cycle may take one to five machine cycles. Generally, first machine cycle known as op code fetch cycle, has either four or six T-states and the remaining machine cycles called execution cycles, have two or three T-states. A T-state represents one clock cycle.

The NOP is the shortest instruction which takes only one machine cycle with four T-states. However, the CALL is the longest instruction which takes five machine cycles with 18 T-states.

Here the timing diagrams of a few instructions will be discussed.

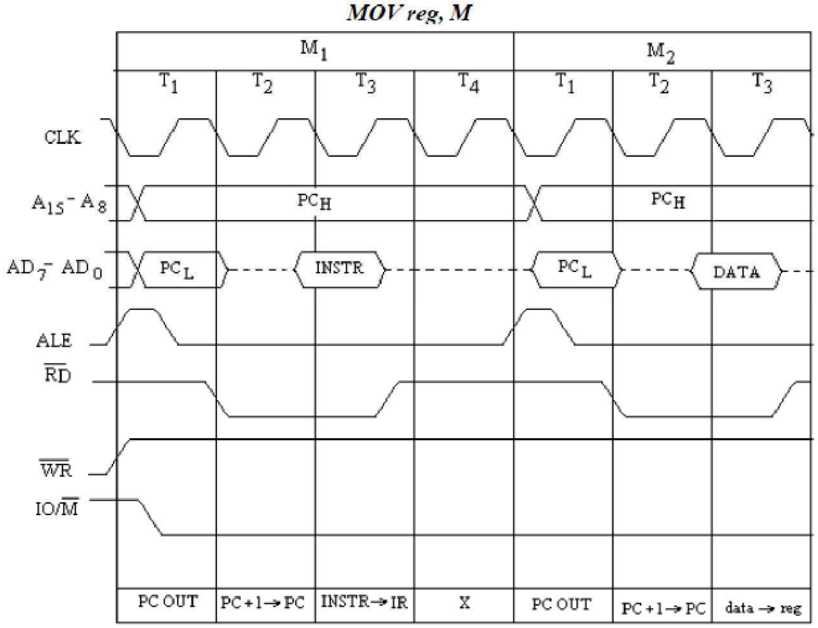
5.4.1 Timing Diagram of MOV reg, M The timing diagram of the instruction

MOV reg, M is shown in figure 5.9.

This is an indirect read instruction and takes two memory cycles (M_1 and M_2). The first machine cycle (M_1) is known as instruction fetch cycle, during which the op code of the instruction is fetched from the memory. This machine cycle takes four Tstates. The second machine cycle M_2 is known as the execution cycle during which the data from the memory location addressed by H-L register pair is transferred to the given register. The second machine takes three states.

During the first T-state (T_1) of M_1 cycle, microprocessor sends the address of the memory location where the op code of the instruction MOV reg, M is stored, to the address lines. The high order byte of the PC (PCH) is placed on A_8-A_{15} lines and it stays on till T_4. The low order byte of PC (PCL) is placed on the address data lines (AD_0-AD_7) which stays on only during T_1-state. For this purpose ALE (Address Latch Enable) signal gives a positive pulse midway through first T-state (T_1), which latches the low order byte of the address into the memory chips. The IO/M signal goes low at the beginning of T_1 state; this enables the peripheral chips for a memory operation rather than input/output operation. It is customary to represent the address lines by double sided waveforms as the address bits may be high or low (ref. fig. 5.9).

MOV reg, M



During the second T-state (T_2) of this op code fetch cycle, program counter (PC) is incremented (PC = PC + 1). The address disappears from the address data bus (AD_7- AD_0) at the beginning of T_2 state. This is shown by dashed line indicating the data on the bus is invalid or meaningless. At the beginning of T_2 state RD signal goes low and remains low till the middle of the T_3 state.

During the third T-state (T_3) of M_1 machine cycle, the op code of the instruction is read out from the memory which is sent to the instruction register (IR) i.e. INSTR □□IR .

The fourth T-state (T_4) of M_1 machine cycle is denoted by X which indicates that this T-state is needed for the instruction decoding and other internal operations before the execution cycle. Now the second machine cycle M_2 known as execution cycle starts. During the first T-state of this execution cycle the contents of H-L register pair are placed on the address bus and address data bus. Basically, it performs the same operation as the T_1-state of M_1 cycle. During T_2 and T_3 states of M_2 machine cycle the data is read from the memory and copied in the specified register.

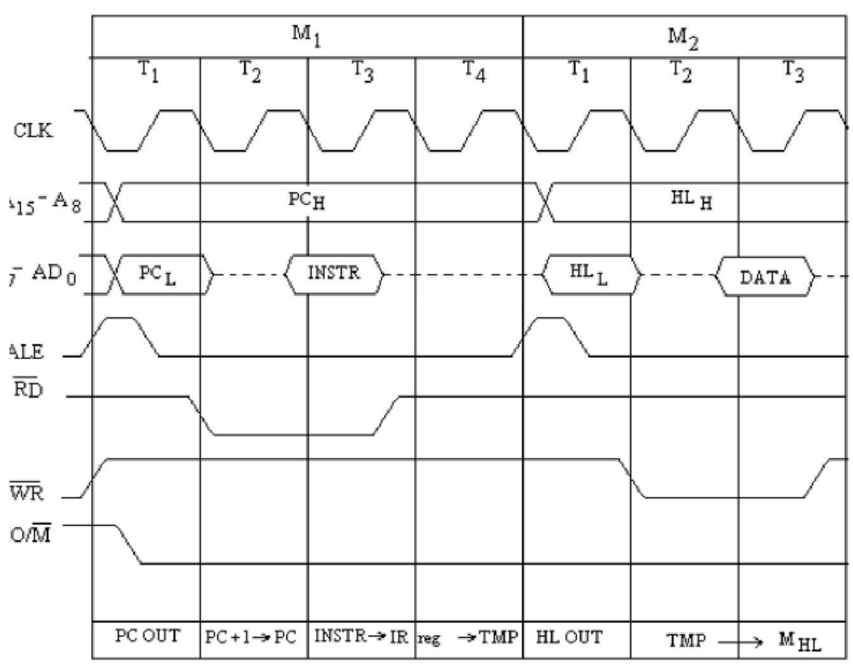
This completes the instruction MOV reg, M. It may be noted that this instruction needs two machine cycles with 7 T-states.

5.4.2 Timing Diagram of MOV M, reg

This is an indirect write instruction. The timing diagram of MOV M, reg instruction is shown in figure 5.10. The first three states of memory fetch cycle M_1 are the same as for MOV reg, M. During fourth T-state (T_4) of M_1, the contents of specified register are copied in the temporary register. During the T_1-state of second machine cycle M_2, the contents of H-L pair are placed on the address and address data bus. As usual during this state ALE sends a high pulse. During T_2 and T_3 states of machine cycle M_2, contents of temporary register are transferred (copied) to the specified address. Since it is memory write instruction, so at the beginning of T_2-state WR signal activates (becomes low) and it remains low till half way through its T_3-state.

This instruction also takes two machine cycles with 7 T-states.

MOV M, reg



Module 2

1. Addition of Two 8 bit numbers and the result is 8 bit

Opcode	Operand	Comments
LXI	2501H	ADDRESS OF 1'ST NO.GET 2501H IN HL pair
MOV	A,M	Move the Content of 2501 to the acc.
INX	H	Increments HL pair by 1
ADD	M	Add 1st and 2nd numbers
STA	2503H	Store the sum in memory location 2503H.
HLT		End the program

2. Addition of two 8 bit no. and sum is 16 bits

Opcode	Operand	Comments
LXI	H,2501H	ADDRESS OF 1'ST NO.is stored in HL pair.
MVI	C,00H	MSBs of sum in register C is initially given the value 00.
MOV	A,M	Move the contain of 2501H to Acc.
INX	H	Increment HL pair by 1.
ADD	M	Add 1st &2nd No.
JNC	L1	If carry is not present? Go to label L1
INR	C	If present C is increment by 1
L1 STA	2503 H	LSB of sum is store in memory location2503H.

MOV	A,C	The MSB of sum are copied to ACC
STA	2504 H	The MSB of sum is stored in memory location 2504H
HLT		Stop the Program.

3. Find The Larger Of The Two Numbers

Opcode	Operand	Comments
LXI	H,2501H	Address of the first No. in H-L pair is obtained..
MOV	A,M	First No. is copied in to the accumulator.
INX	H	Address of the2nd No. in H-L pair is obtained
CMP	M	2^{ND} Number is compare with the 1^{st} NO.
JNC	LOOP	If 2^{nd} No. not greater than 1^{st} no. go to loop
MOV	A,M	If it is greater it is moved from memory ACC.
Loop STA	2503H	Store the larger No. In the accumulator.
HLT		Stop the program

4. Find The Smaller Of The Two Numbers

Opcode	Operand	Comments
LXI	H,2501H	Address of the first No. in H-L pair is obtained..
MOV	A,M	First No. is copied in to the accumulator.
INX	H	Address of the2nd No. in H-L pair is obtained
CMP	M	2^{ND} Number is compare with the 1^{st} NO.
JC	LOOP	If 1^{st} No. not greater than 2^{nd} no. go to loop

MOV	A,M	If it is greater it is moved from memory ACC.
Loop STA	2503H	Store the larger No. In the accumulator.
HLT		Stop the program

5. Arrange the Data array in Ascending order

Opcode	Operand	Comments
ST:LXI	H8501H	Set up H-L as a memory pointer for bytes.
MVI	D 00H	Clear register D to set up a flag.
MVI	C,04H	Set up counter
CHECK MOV	A,M	Get data byte
INX	H	Point to next byte.
CMP	M	Compare bytes
JC	NB	If a next byte do not Exchange, Move the content memory to the accumulator
MOV	B,M	Get second byte for exchange.
MOV	M,A	Store first byte in second location.
DCX	H	Point to first location
MOV	M,B	Store second byte in first location
INX	H	Go for next comparison.
MVI	D,01H	Load 1 in d as reminder for Exchange.
NB:DCR	C	Decrement counter.
JNZ	CHECK	If counter≠0 go back
MOV	A,D	Get flag bit in A
RRC		Place flag bit D0 in carry.
JC	ST	If flag is 1 exchange occurred start next.
HLT		End of program

6. 8 Bit Multiplication

Opcode	Operand	Comments
MVI	D,00	Initialize register D to 00
MVI	A,00	Initialize Accumulator content to 00
LXI	H,4150	
MOV	B,M	Get the first number in B - reg
INX	H	Increment H
MOV	C,M	Get the second number in C- reg.
LOOP: ADD	B	Add content of A - reg to register B.
JNC NEXT		Jump on no carry to NEXT.
INR	D	Increment content of register D
NEXT: DCR	C	Decrement content of register C.
JNZ LOOP		Jump on no zero to address
STA	4152	Store the result in Memory
MOV	A, D	Move D to A
STA	4153	Store the MSB of result in Memory
HLT		Terminate the program

7. Block of Data Transfer from one set of locations to another

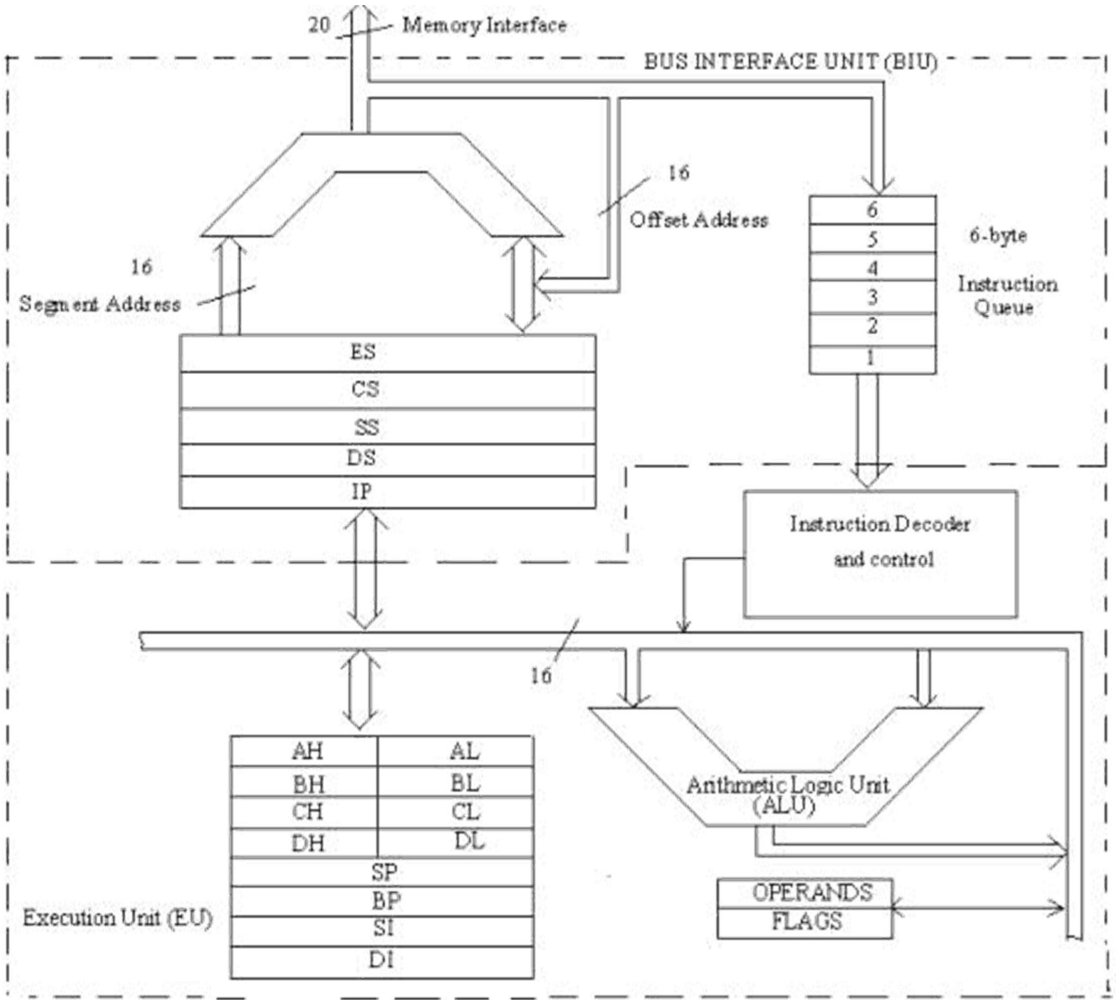
Opcode	Operand	Comments
LXI	D,4500	Load destination address in DE pair
LXI	H,4100	Load the count in HL pair
MOV	C,M	Copy the count to register C
LOOP: INX	H	Increment memory
MOV	A,M	Copy element to Accumulator
STAX	D	Store the element to the address in the DE pair
INX	D	Increment destination address
DCR	C	Decrement count
JNZ	LOOP	Jump on non-zero to the label LOOP
HLT		Program ends

Module 3

8086 Architecture

Intel 8086 is a 16 bit integer processor. It has 16-bit data bus and 20-bit address bus. The lower 16-bit address lines and 16-bit data lines are multiplexed (AD0-AD15). Since 20-bit address lines are available, 8086 can access up to 2 20 or 1 Giga byte of physical memory.

The basic architecture of 8086 is shown below.



The internal architecture of Intel 8086 is divided into two units, viz., Bus Interface Unit (BIU) and Execution Unit (EU).

The Bus Interface Unit (BIU) generates the 20-bit physical memory address and provides the interface with external memory (ROM/RAM). As mentioned earlier, 8086 has a single memory interface. To speed up the execution, 6-bytes of instruction are fetched in advance and kept in a 6byte Instruction Queue while other instructions are being executed in the Execution Unit (EU). Hence after the execution of an instruction, the next instruction is directly fetched from the instruction queue without having to wait for the external memory to send the instruction. This is called pipe-lining and is helpful for speeding up the overall execution process. 8086's BIU produces the 20-bit physical memory address by combining a 16-bit segment address with a 16bit offset address. There are four 16-bit segment registers, viz., the code segment (CS), the stack segment (SS), the extra segment (ES), and the data segment (DS). These segment registers hold the corresponding 16-bit segment addresses. A segment address is the upper 16-bits of the starting address of that segment. The lower 4-bits of the starting address of a segment is always zero. The offset address is held by another 16-bit register. The physical 20-bit address is calculated by shifting the segment address 4-bit left and then adding that to the offset address.

General Purpose Registers

8086 CPU has 8 general purpose registers, those are:

I. AX - the accumulator register (divided into AH / AL)
II. BX - the base address register (divided into BH / BL)
III. CX - the count register (divided into CH / CL)
IV. DX - the data register (divided into DH / DL) V. SI - source index register
VI. DI - destination index register
VII. BP - base pointer
VIII. SP - stack pointer

Segment Registers

CS - points at the segment containing the current program
DS - generally points at segment where variables are defined ES
- extra segment register, it's up to a coder to define its usage SS
- points at the segment containing the stack.

Although it is possible to store any data in the segment registers, this is never a good idea. The segment registers have a very special purpose - pointing at accessible blocks of memory.

Segment registers work together with general purpose register to access any memory value. For example if we would like to access memory at the physical address 12345h (hexadecimal), we should set the DS = 1230h and SI = 0045h. This is good, since this way we can access much more memory than with a single register that is limited to 16 bit values.
CPU makes a calculation of physical address by multiplying the segment register by 10h and adding general purpose register to it (1230h * 10h + 45h = 12345h):

```
 12300
+ 0045
 -----
 12345
```

Note:

- The address formed with 2 registers is called an effective address.
- By default BX, SI and DI registers work with DS segment register.
  BP and SP work with SS segment register.
- Other general purpose registers cannot form an effective address. BX can form an effective address, BH and BL cannot.

Special Purpose Registers

IP - the instruction pointer.
Flags Register - determines the current state of the processor.

IP register always works together with CS segment register and it points to currently executing instruction. Flags Register is modified automatically by CPU after mathematical operations, this allows to determine the type of the result, and to determine conditions to transfer control to other parts of the program. Generally these registers cannot be accessed directly.

<u>Flag Register</u>

A 16 flag register is used in 8086. It is divided into two parts : Condition code or status flags and Machine control flags.

The condition code flag register is the lower byte of the 16-bit flag register. The condition code flag register is identical to 8085 flag register, with an additional overflow flag.The control flag register is the higher byte of the flag register. It contains three flags namely direction flag(D), interrupt flag (I) and trap flag (T).

S- Sign Flag : This flag is set, when the result of any computation is negative.

Z- Zero Flag: This flag is set, if the result of the computation or comparison performed by the previous instruction is zero.

P- Parity Flag: This flag is set to 1, if the lower byte of the result contains even number of 1's.

C- Carry Flag: This flag is set, when there is a carry out of MSB in case of addition or a borrow in case of subtraction.

T- Tarp Flag: If this flag is set, the processor enters the single step execution mode.

I- Interrupt Flag: If this flag is set, the maskable interrupt are recognized by the CPU, otherwise they are ignored.

D- Direction Flag: This is used by string manipulation instructions. If this flag bit is ‘0’, the string is processed beginning from the lowest address to the highest address, i.e., auto incrementing mode. Otherwise, the string is processed from the highest address towards the lowest address, i.e., auto incrementing mode.

AC-Auxilary Carry Flag: This is set, if there is a carry from the lowest nibble, i.e, bit three during addition, or borrow for the lowest nibble, i.e, bit three, during subtraction.

O- Over flow Flag: This flag is set, if an overflow occurs, i.e, if the result of a signed operation is large enough to accommodate in a destination register. The result is of more than 7-bits in size in case of 8-bit signed operation and more than 15-bits in size in case of 16-bit sign operations, then the overflow will be set.

Memory Segmentation

The memory in an 8086 based system is organized as segmented memory. In this scheme, the complete physically available memory may be divided into a number of logical segments. Each segment is 64K bytes in size and is addressed by one of the segment registers. The 16-bit contents of the segment register actually point to the starting location of a particular segment.To address aspecific memory location within a segment, we need an offset address. The offset address is also 16-bit long so that the maximum offset value can be FFFFH, and the maximum size of any segment is thus 64K locations.The CPU 8086 is able to address 1Mbytes of physical memory. The complete 1Mbyte memory can be divided into 16 segments, each of 64Kbytes size. The addresses of the segments (segment base address) may be assigned as 0000H to F000H respectively. The offset address values are from 0000H to FFFFH so that the physical addresses range from 00000H to FFFFFH. In the above case, the segments are called non-overlapping segments. The nonoverlapping segments are in some cases, however, the segments may be overlapping as Suppose a segment starts at a particular address and its maximum size can be 64Kbytes. But, if another segment starts before this 64Kbytes location of the first segment, the two segments are said to be overlapping segments. The area of memory from the start of the second segment to the possible end of the first segment is called as overlapped segment area. The size of address bus of 8086 is 20 and is able to address 1 Mbytes () of physical memory.The compete 1 Mbytes memory can be divided into 16 segments, each of 16 Kbytes size.The addresses of the segment may be assigned as 0000H to F000H respectively.The offset values are from 0000H to FFFFFH.If the segmentation is done as per above mentioned way, the segments are called non-overlapping segments.

In some cases segment may overlap also. Suppose a segment starts at a particular address and its maximum size can go up to 64 Kbytes. But if another segment starts before this 64 Kbytes location of the first segment, the two segments are said to be overlapping segment.The area of memory from the start of the second segment to the possible end of the first segment is called as overlapped segment.

The main advantages of the segmented memory scheme are as follows:

a) Allows the memory capacity to be 1 Mbyte although the actual addresses to be handled are of 16-bit size
b) Allows the placing of code data and stack portions of the same program in different parts (segments) of memory, for data and code protection.
c) Permits a program and/ or its data to be put into different areas of memory each time program is executed, ie, provision for relocation may be done.

Non-overlapping segments and overlapping segments are shown in the figure below.

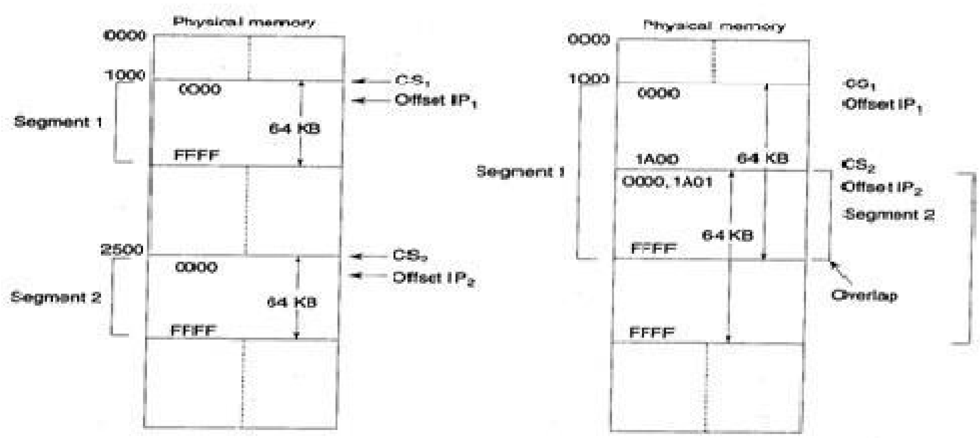
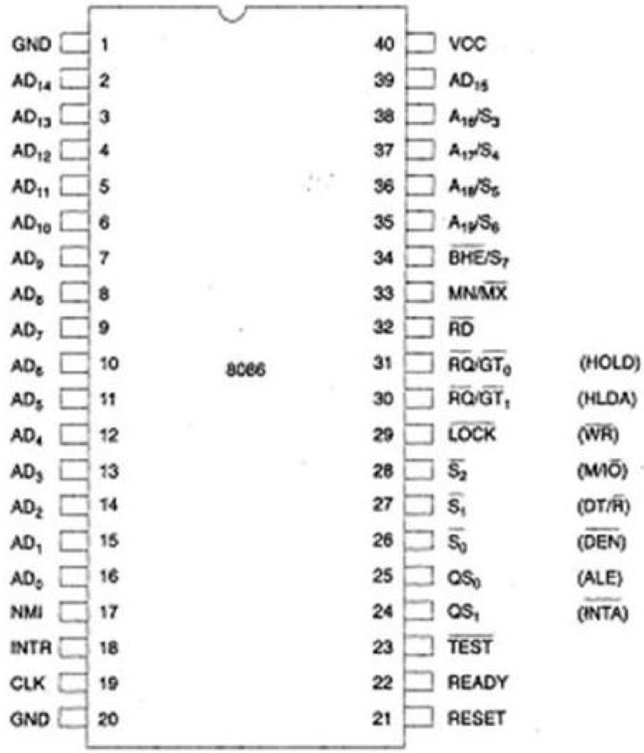


Fig: Non-overlapping segment and Overlapping Segment

8086 Pin Diagram

8086 was the first 16-bit microprocessor available in 40-pin DIP (Dual Inline Package) chip. Let us now discuss in detail the pin configuration of a 8086 Microprocessor.

Here is the pin diagram of 8086 microprocessor −



Let us now discuss the signals in detail −

Power supply and frequency signals

It uses 5V DC supply at V_{CC} pin 40, and uses ground at V_{SS} pin 1 and 20 for its operation.

Clock signal

Clock signal is provided through Pin-19. It provides timing to the processor for operations. Its frequency is different for different versions, i.e. 5MHz, 8MHz and 10MHz.

Address/data bus

AD0-AD15. These are 16 address/data bus. AD0-AD7 carries low order byte data and AD8AD15 carries higher order byte data. During the first clock cycle, it carries 16-bit address and after that it carries 16-bit data.

Address/status bus

A16-A19/S3-S6. These are the 4 address/status buses. During the first clock cycle, it carries 4bit address and later it carries status signals.

S7/BHE

BHE stands for Bus High Enable. It is available at pin 34 and used to indicate the transfer of data using data bus D8-D15. This signal is low during the first clock cycle, thereafter it is active.

Read

It is available at pin 32 and is used to read signal for Read operation.

Ready

It is available at pin 32. It is an acknowledgement signal from I/O devices that data is transferred. It is an active high signal. When it is high, it indicates that the device is ready to transfer data. When it is low, it indicates wait state.

RESET

It is available at pin 21 and is used to restart the execution. It causes the processor to immediately terminate its present activity. This signal is active high for the first 4 clock cycles to RESET the microprocessor.

INTR

It is available at pin 18. It is an interrupt request signal, which is sampled during the last clock cycle of each instruction to determine if the processor considered this as an interrupt or not.

NMI

It stands for non-maskable interrupt and is available at pin 17. It is an edge triggered input, which causes an interrupt request to the microprocessor.

TEST

This signal is like wait state and is available at pin 23. When this signal is high, then the processor has to wait for IDLE state, else the execution continues.

MN/MX

It stands for Minimum/Maximum and is available at pin 33. It indicates what mode the processor is to operate in; when it is high, it works in the minimum mode and vice-aversa.

INTA

It is an interrupt acknowledgement signal and id available at pin 24. When the microprocessor receives this signal, it acknowledges the interrupt.

ALE

It stands for address enable latch and is available at pin 25. A positive pulse is generated each time the processor begins any operation. This signal indicates the availability of a valid address on the address/data lines.

DEN

It stands for Data Enable and is available at pin 26. It is used to enable Transreceiver 8286. The transreceiver is a device used to separate data from the address/data bus.

DT/R

It stands for Data Transmit/Receive signal and is available at pin 27. It decides the direction of data flow through the transreceiver. When it is high, data is transmitted out and vice-a-versa.

M/IO

This signal is used to distinguish between memory and I/O operations. When it is high, it indicates I/O operation and when it is low indicates the memory operation. It is available at pin 28.

WR

It stands for write signal and is available at pin 29. It is used to write the data into the memory or the output device depending on the status of M/IO signal.

HLDA

It stands for Hold Acknowledgement signal and is available at pin 30. This signal acknowledges the HOLD signal.

HOLD

This signal indicates to the processor that external devices are requesting to access the address/data buses. It is available at pin 31.

QS_1 and QS_0

These are queue status signals and are available at pin 24 and 25. These signals provide the status of instruction queue. Their conditions are shown in the following table −

QS_0	QS_1	Status
0	0	No operation

0	1	First byte of opcode from the queue
1	0	Empty the queue
1	1	Subsequent byte from the queue

S_0, S_1, S_2

These are the status signals that provide the status of operation, which is used by the Bus Controller 8288 to generate memory & I/O control signals. These are available at pin 26, 27, and 28. Following is the table showing their status −

S_2	S_1	S_0	Status
0	0	0	Interrupt acknowledgement
0	0	1	I/O Read
0	1	0	I/O Write
0	1	1	Halt
1	0	0	Opcode fetch

1	0	1	Memory read
1	1	0	Memory write
1	1	1	Passive

LOCK

When this signal is active, it indicates to the other processors not to ask the CPU to leave the system bus. It is activated using the LOCK prefix on any instruction and is available at pin 29.

RQ/GT_1 and RQ/GT_0

These are the Request/Grant signals used by the other processors requesting the CPU to release the system bus. When the signal is received by CPU, then it sends acknowledgment. RQ/GT_0 has a higher priority than RQ/GT_1.

Addressing Modes

The different ways in which a source operand is denoted in an instruction is known as addressing modes. There are 8 different addressing modes in 8086 programming −

Immediate addressing mode

The addressing mode in which the data operand is a part of the instruction itself is known as immediate addressing mode.

Example

```
MOV CX, 4929 H, ADD AX, 2387 H,  MOV AL, FFH
```

Register addressing mode

It means that the register is the source of an operand for an instruction.

Example

```
MOV CX, AX   ; copies the contents of the 16-bit AX register into
             ; the 16-bit CX register),
ADD BX, AX
```

Direct addressing mode

The addressing mode in which the effective address of the memory location is written directly in the instruction.

Example

```
MOV AX, [1592H], MOV AL, [0300H]
```

Register indirect addressing mode

This addressing mode allows data to be addressed at any memory location through an offset address held in any of the following registers: BP, BX, DI & SI.

Example

```
MOV AX, [BX]  ; Suppose the register BX contains 4895H, then the contents
            ; 4895H are moved to AX

ADD CX, {BX}
```

Based addressing mode

In this addressing mode, the offset address of the operand is given by the sum of contents of the BX/BP registers and 8-bit/16-bit displacement.

Example

```
MOV DX, [BX+04], ADD CL, [BX+08]
```

Indexed addressing mode

In this addressing mode, the operands offset address is found by adding the contents of SI or DI register and 8-bit/16-bit displacements.

Example

```
MOV BX, [SI+16], ADD AL, [DI+16]
```

Based-index addressing mode

In this addressing mode, the offset address of the operand is computed by summing the base register to the contents of an Index register.

Example

```
ADD CX, [AX+SI], MOV AX, [AX+DI]
```

Based indexed with displacement mode

In this addressing mode, the operands offset is computed by adding the base register contents. An Index registers contents and 8 or 16-bit displacement.

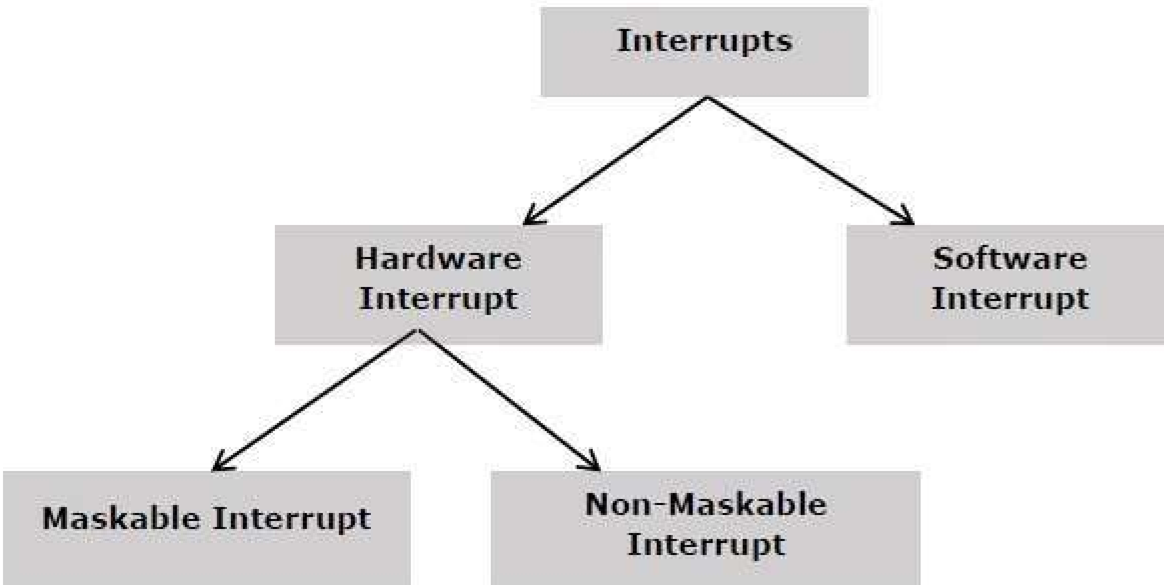
Example

```
MOV AX, [BX+DI+08], ADD CX, [BX+SI+16]
```

Interrupt

Interrupt is the method of creating a temporary halt during program execution and allows peripheral devices to access the microprocessor. The microprocessor responds to that interrupt with an ISR (Interrupt Service Routine), which is a short program to instruct the microprocessor on how to handle the interrupt.

The following image shows the types of interrupts we have in a 8086 microprocessor −



Hardware Interrupts

Hardware interrupt is caused by any peripheral device by sending a signal through a specified pin to the microprocessor.The 8086 has two hardware interrupt pins, i.e. NMI and INTR. NMI is a non-maskable interrupt and INTR is a maskable interrupt having lower priority. One more interrupt pin associated is INTA called interrupt acknowledge.

NMI

It is a single non-maskable interrupt pin (NMI) having higher priority than the maskable interrupt request pin (INTR)and it is of type 2 interrupt.

When this interrupt is activated, these actions take place −

Completes the current instruction that is in progress.

- Pushes the Flag register values on to the stack.

- Pushes the CS (code segment) value and IP (instruction pointer) value of the return address on to the stack.
- IP is loaded from the contents of the word location 00008H.
- CS is loaded from the contents of the next word location 0000AH.
- Interrupt flag and trap flag are reset to 0.

INTR

The INTR is a maskable interrupt because the microprocessor will be interrupted only if interrupts are enabled using set interrupt flag instruction. It should not be enabled using clear interrupt Flag instruction.The INTR interrupt is activated by an I/O port. If the interrupt is enabled and NMI is disabled, then the microprocessor first completes the current execution and sends ‘0’ on INTA pin twice. The first ‘0’ means INTA informs the external device to get ready and during the second ‘0’ the microprocessor receives the 8 bit, say X, from the programmable interrupt controller.

These actions are taken by the microprocessor − First completes the current instruction.

- Activates INTA output and receives the interrupt type, say X.
- Flag register value, CS value of the return address and IP value of the return address are pushed on to the stack.
- IP value is loaded from the contents of word location X × 4
- CS is loaded from the contents of the next word location.
- Interrupt flag and trap flag is reset to 0

Software Interrupts

Some instructions are inserted at the desired position into the program to create interrupts. These interrupt instructions can be used to test the working of various interrupt handlers. It includes −

INT- Interrupt instruction with type number

It is 2-byte instruction. First byte provides the op-code and the second byte provides the interrupt type number. There are 256 interrupt types under this group.

Its execution includes the following steps −

- Flag register value is pushed on to the stack.

- CS value of the return address and IP value of the return address are pushed on to the stack.
- IP is loaded from the contents of the word location ‘type number’ × 4 CS is loaded from the contents of the next word location.
- Interrupt Flag and Trap Flag are reset to 0

The starting address for type0 interrupt is 000000H, for type1 interrupt is 00004H similarly for type2 is 00008H and ……so on. The first five pointers are dedicated interrupt pointers. i.e. − TYPE 0 interrupt represents division by zero situation.

- TYPE 1 interrupt represents single-step execution during the debugging of a program.
- TYPE 2 interrupt represents non-maskable NMI interrupt.
- TYPE 3 interrupt represents break-point interrupt.
- TYPE 4 interrupt represents overflow interrupt.

The interrupts from Type 5 to Type 31 are reserved for other advanced microprocessors, and interrupts from 32 to Type 255 are available for hardware and software interrupts.

INT 3-Break Point Interrupt Instruction

It is a 1-byte instruction having op-code is CCH. These instructions are inserted into the program so that when the processor reaches there, then it stops the normal execution of program and follows the break-point procedure.

Its execution includes the following steps −

- Flag register value is pushed on to the stack.
- CS value of the return address and IP value of the return address are pushed on to the stack.
- IP is loaded from the contents of the word location 3×4 = 0000CH
- CS is loaded from the contents of the next word location.
- Interrupt Flag and Trap Flag are reset to 0

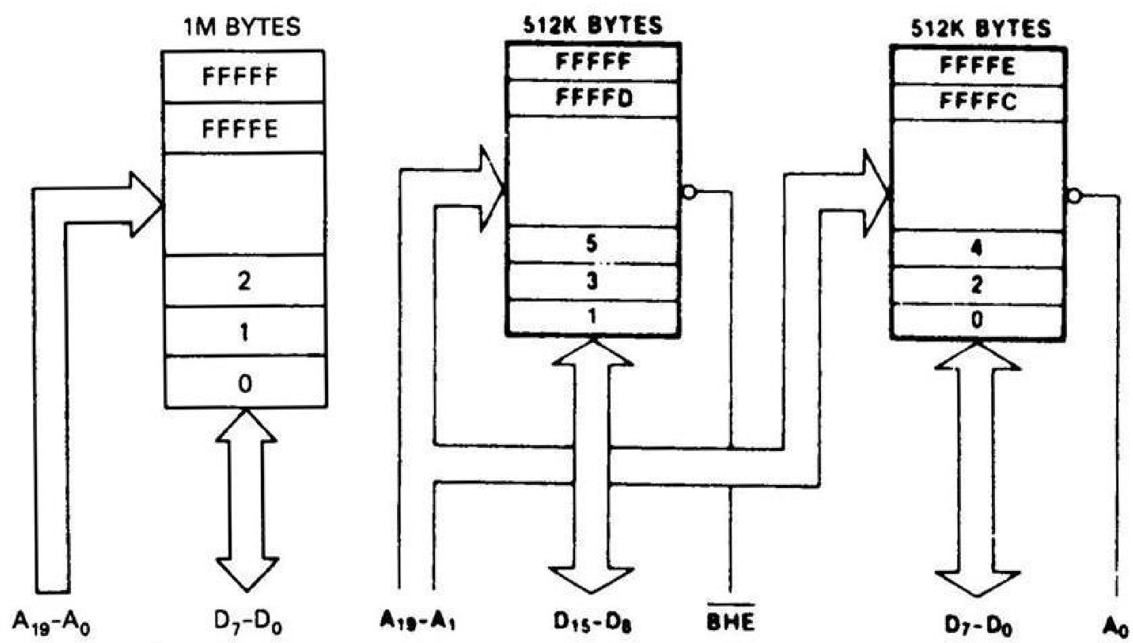
INTO - Interrupt on overflow instruction

It is a 1-byte instruction and their mnemonic INTO. The op-code for this instruction is CEH. As the name suggests it is a conditional interrupt instruction, i.e. it is active only when the overflow flag is set to 1 and branches to the interrupt handler whose interrupt type number is 4. If the overflow flag is reset then, the execution continues to the next instruction.

Its execution includes the following steps −.

- Flag register values are pushed on to the stack.
- CS value of the return address and IP value of the return address are pushed on to the stack.
- IP is loaded from the contents of word location 4×4 = 00010H
- CS is loaded from the contents of the next word location.
- Interrupt flag and Trap flag are reset to 0

Memory Interfacing



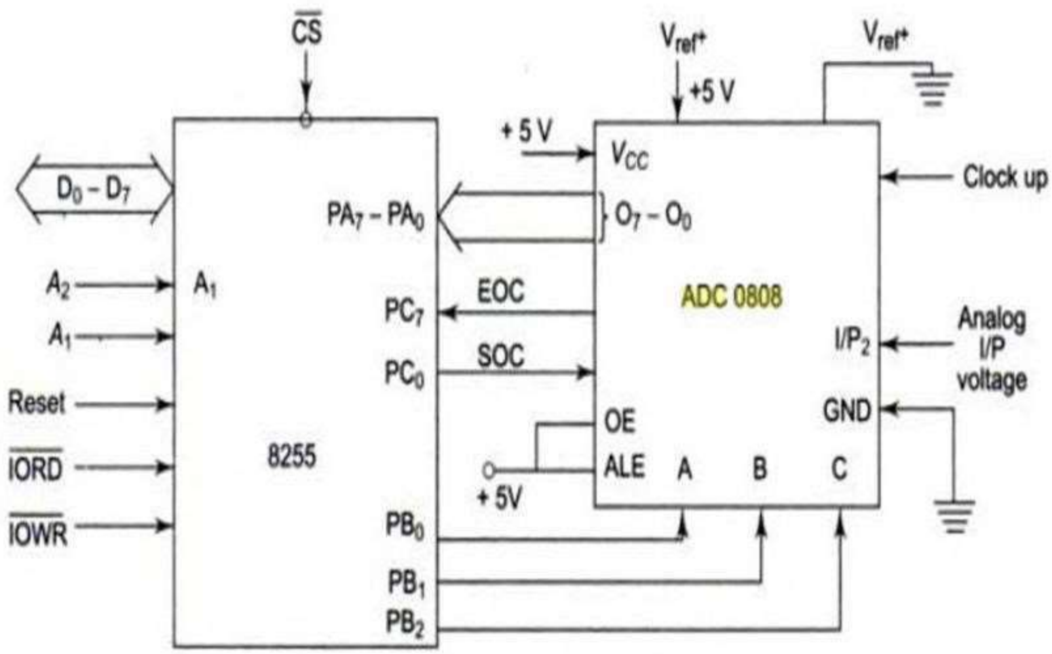
The general procedure of static memory interfacing with 8086 is described as follows:

1.Arrange the available memory chips so as to obtain16-bit data bus width.
2.The upper 8-bit bank is called odd address bank and lower 8-bit bank is called even address bank.
3.Connect available address lines of memory chips with those of microprocessor and also connect read and write inputs to corresponding processor control signals.
4.Connect 16-bit data bus of the memory bank with that of the microprocessor 8086.
5.The remaining address lines of BHE and A0are used for decoding the required chip select signals for the odd and even memory banks.
6.The chip select of memory is derived from the O/P of the decoding.

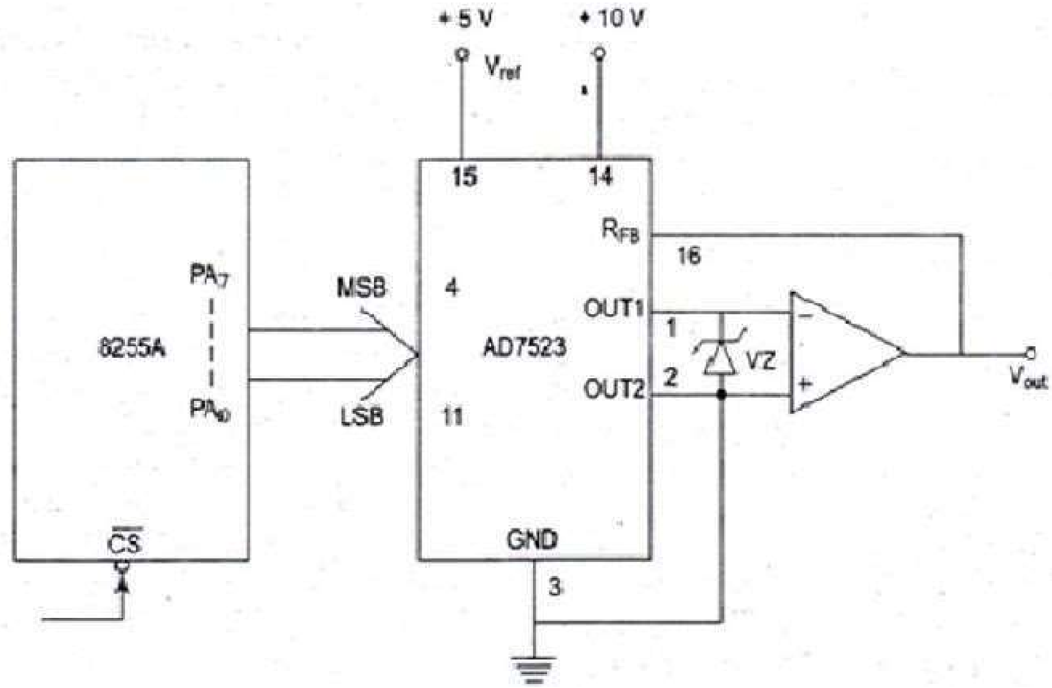
ADC and DAC Interfacing

In most of the cases, the PIO 8255 is used for interfacing the analog to digital and digital to analog converters with a microprocessor. We have already studied 8255 interfacing with 8086 as an I/O port in interfacing chapter. This section we will only emphasize the interfacing techniques of analog to digital and digital to analog converters with 8255.

ADC Interfacing with 8086



DAC Interfacing with 8086



Instruction Set

The 8086 microprocessor supports 8 types of instructions −

- Data Transfer Instructions
- Arithmetic Instructions
- Bit Manipulation Instructions
- String Instructions
- Program Execution Transfer Instructions (Branch & Loop Instructions)
- Processor Control Instructions
- Iteration Control Instructions
- Interrupt Instructions

Data Transfer Instructions

These instructions are used to transfer the data from the source operand to the destination operand. Following are the list of instructions under this group −

Instruction to transfer a word

- MOV − Used to copy the byte or word from the provided source to the provided destination.
- PPUSH − Used to put a word at the top of the stack.

- POP − Used to get a word from the top of the stack to the provided location.
- PUSHA − Used to put all the registers into the stack.
- POPA − Used to get words from the stack to all registers.
- XCHG − Used to exchange the data from two locations.
- XLAT − Used to translate a byte in AL using a table in the memory.

Instructions for input and output port transfer

- IN − Used to read a byte or word from the provided port to the accumulator.
- OUT − Used to send out a byte or word from the accumulator to the provided port.

Instructions to transfer the address

- LEA − Used to load the address of operand into the provided register.
- LDS − Used to load DS register and other provided register from the memory
- LES − Used to load ES register and other provided register from the memory.

Instructions to transfer flag registers

- LAHF − Used to load AH with the low byte of the flag register.
- SAHF − Used to store AH register to low byte of the flag register.
- PUSHF − Used to copy the flag register at the top of the stack.
- POPF − Used to copy a word at the top of the stack to the flag register.

Arithmetic Instructions

These instructions are used to perform arithmetic operations like addition, subtraction, multiplication, division, etc.

Following is the list of instructions under this group −

Instructions to perform addition

- ADD − Used to add the provided byte to byte/word to word.

ADC − Used to add with carry.

INC − Used to increment the provided byte/word by 1.

- AAA − Used to adjust ASCII after addition.
- DAA − Used to adjust the decimal after the addition/subtraction operation.

Instructions to perform subtraction

- SUB − Used to subtract the byte from byte/word from word.
- SBB − Used to perform subtraction with borrow.
- DEC − Used to decrement the provided byte/word by 1.
- NPG − Used to negate each bit of the provided byte/word and add 1/2's complement.
- CMP − Used to compare 2 provided byte/word.
- AAS − Used to adjust ASCII codes after subtraction.
- DAS − Used to adjust decimal after subtraction.

Instruction to perform multiplication

- MUL − Used to multiply unsigned byte by byte/word by word.
- IMUL − Used to multiply signed byte by byte/word by word.
- AAM − Used to adjust ASCII codes after multiplication.

Instructions to perform division

- DIV − Used to divide the unsigned word by byte or unsigned double word by word.
- IDIV − Used to divide the signed word by byte or signed double word by word.
- AAD − Used to adjust ASCII codes after division.
- CBW − Used to fill the upper byte of the word with the copies of sign bit of the lower byte.
- CWD − Used to fill the upper word of the double word with the sign bit of the lower word.

Bit Manipulation Instructions

These instructions are used to perform operations where data bits are involved, i.e. operations like logical, shift, etc.

Following is the list of instructions under this group −

Instructions to perform logical operation

- NOT − Used to invert each bit of a byte or word.
- AND − Used for adding each bit in a byte/word with the corresponding bit in another byte/word.
- OR − Used to multiply each bit in a byte/word with the corresponding bit in another byte/word.
- XOR − Used to perform Exclusive-OR operation over each bit in a byte/word with the corresponding bit in another byte/word.
- TEST − Used to add operands to update flags, without affecting operands.

Instructions to perform shift operations

- SHL/SAL − Used to shift bits of a byte/word towards left and put zero(S) in LSBs.
- SHR − Used to shift bits of a byte/word towards the right and put zero(S) in MSBs.
- SAR − Used to shift bits of a byte/word towards the right and copy the old MSB into the new MSB.

Instructions to perform rotate operations

- ROL − Used to rotate bits of byte/word towards the left, i.e. MSB to LSB and to Carry Flag [CF].
- ROR − Used to rotate bits of byte/word towards the right, i.e. LSB to MSB and to Carry Flag [CF].
- RCR − Used to rotate bits of byte/word towards the right, i.e. LSB to CF and CF to MSB.
- RCL − Used to rotate bits of byte/word towards the left, i.e. MSB to CF and CF to LSB.

String Instructions

String is a group of bytes/words and their memory is always allocated in a sequential order.

Following is the list of instructions under this group − REP −
Used to repeat the given instruction till CX ≠ 0.

REPE/REPZ − Used to repeat the given instruction until CX = 0 or zero flag ZF = 1.

- REPNE/REPNZ − Used to repeat the given instruction until CX = 0 or zero flag ZF = 1.
- MOVS/MOVSB/MOVSW − Used to move the byte/word from one string to another.
- COMS/COMPSB/COMPSW − Used to compare two string bytes/words.
- INS/INSB/INSW − Used as an input string/byte/word from the I/O port to the provided memory location.
- OUTS/OUTSB/OUTSW − Used as an output string/byte/word from the provided memory location to the I/O port.
- SCAS/SCASB/SCASW − Used to scan a string and compare its byte with a byte in AL or string word with a word in AX.
- LODS/LODSB/LODSW − Used to store the string byte into AL or string word into AX.

Program Execution Transfer Instructions (Branch and Loop Instructions)

These instructions are used to transfer/branch the instructions during an execution. It includes the following instructions −

Instructions to transfer the instruction during an execution without any condition

- CALL − Used to call a procedure and save their return address to the stack.
- RET − Used to return from the procedure to the main program.
- JMP − Used to jump to the provided address to proceed to the next instruction.

Instructions to transfer the instruction during an execution with some conditions

- JA/JNBE − Used to jump if above/not below/equal instruction satisfies.
- JAE/JNB − Used to jump if above/not below instruction satisfies.
- JBE/JNA − Used to jump if below/equal/ not above instruction satisfies.

- JC − Used to jump if carry flag CF = 1
- JE/JZ − Used to jump if equal/zero flag ZF = 1
- JG/JNLE − Used to jump if greater/not less than/equal instruction satisfies.

-
- JGE/JNL − Used to jump if greater than/equal/not less than instruction satisfies.
- JL/JNGE − Used to jump if less than/not greater than/equal instruction satisfies.
- JLE/JNG − Used to jump if less than/equal/if not greater than instruction satisfies.
- JNC − Used to jump if no carry flag (CF = 0)
- JNE/JNZ − Used to jump if not equal/zero flag ZF = 0
- JNO − Used to jump if no overflow flag OF = 0
- JNP/JPO − Used to jump if not parity/parity odd PF = 0
- JNS − Used to jump if not sign SF = 0
- JO − Used to jump if overflow flag OF = 1
- JP/JPE − Used to jump if parity/parity even PF = 1
- JS − Used to jump if sign flag SF = 1

Processor Control Instructions

These instructions are used to control the processor action by setting/resetting the flag values.

Following are the instructions under this group −

- STC − Used to set carry flag CF to 1
- CLC − Used to clear/reset carry flag CF to 0
- CMC − Used to put complement at the state of carry flag CF.
- STD − Used to set the direction flag DF to 1
- CLD − Used to clear/reset the direction flag DF to 0
- STI − Used to set the interrupt enable flag to 1, i.e., enable INTR input.
- CLI − Used to clear the interrupt enable flag to 0, i.e., disable INTR input.

Iteration Control Instructions

These instructions are used to execute the given instructions for number of times. Following is the list of instructions under this group −

- LOOP − Used to loop a group of instructions until the condition satisfies, i.e., CX = 0
LOOPE/LOOPZ − Used to loop a group of instructions till it satisfies ZF = 1 & CX = 0

  LOOPNE/LOOPNZ − Used to loop a group of instructions till it satisfies ZF = 0 & CX = 0

- JCXZ − Used to jump to the provided address if CX = 0

Interrupt Instructions

These instructions are used to call the interrupt during program execution.

- INT − Used to interrupt the program during execution and calling service specified.
- INTO − Used to interrupt the program during execution if OF = 1
- IRET − Used to return from interrupt service to the main program

Module 4

1. Addition:

MOV AX,05

MOV BX,03

ADD AX,BX

MOV SI,8000

MOV[SI],AX

HLT 2. Subtraction: MOV AX,05 MOV

BX,03

SUB AX,BX

MOV SI,8000

MOV[SI],AX

HLT 3.

Multiplication: MOV AX,05 MOV BX,03

MUL BX

MOV SI,8000

MOV[SI],AX

HLT

4. Division:

MOV AX,05

MOV BX,03

DIV BX

MOV SI,8000

MOV[SI],AX

HLT

5. Block of data transfer: MOV AX, 2000H MOV

DS, AX

```
MOV AX, 3000H
MOV ES, AX
MOV SI, 0000H
MOV DI, 0000H
MOV CX, 000AH
CLD
REP: MOVSB
MOV AH, 4CH
INT 21H
```

6. Smallest/Largest no finding:

```
ASSUME CS:CODE,DS:DATA
MOV AX,DATA
MOV DS,AX
MOV CL,04H
MOV AX,0000H
MOV AL,DS:[SI]
INC SI
UP: CMP AL,DS:[SI]
JC DOWN          (JNC for largest no)
MOV AL,DS:[SI]
DOWN:INC SI
DEC CL
JNZ UP
MOV SM,AL
```

INT 3H

7. Ascending/descending order:

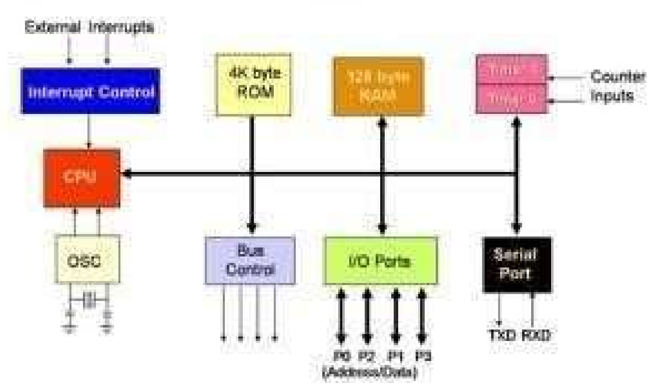
```
ASSUME CS:CODE, DS:DATA

START:  MOV AX,DATA

MOV DS,AX

MOV BX,SIZ

DEC BX

OUTLUP: MOV CX,BX

MOV SI,0H

INLOOP: MOV AL,A[SI]

INC SI

CMP AL,A[SI]

JB GO_ON      (JA for Descending order)

XCHG AL,A[SI]

MOV A[SI-1],AL

GO_ON:  LOOP INLOOP

DEC BX

JNZ OUTLUP

INT 3
```

Module 5

The 8051 Microcontroller was designed in 1980's by Intel. Its foundation was on Harvard Architecture and was developed principally for bringing into play in Embedded Systems.

8051 Microcontroller Architecture:

Microcontroller 8051 block diagram is shown below. Let's have a closer look on features of 8051 microcontroller design:



Block Diagram of 8051 Microcontroller CPU (Central Processor Unit):

As you may be familiar that Central Processor Unit or CPU is the mind of any processing machine. It scrutinizes and manages all processes that are carried out in the Microcontroller. User has no power over the functioning of CPU. It interprets program printed in storage space (ROM) and carries out all of them and do the projected duty. CPU manages different types of registers in 8051 microcontroller.

Pin Configuration:

Pins 1 – 8:- recognized as Port 1. Different from other ports, this port doesn't provide any other purpose. Port 1 is a domestically pulled up, quasi bi directional Input/output port.

Pin 9:- As made clear previously RESET pin is utilized to set the micro-controller 8051 to its primary values, whereas the micro-controller is functioning or at the early beginning of application. The RESET pin has to be set elevated for two machine rotations.

Pins 10 – 17:- recognized as Port 3. This port also supplies a number of other functions such as timer input, interrupts, serial communication indicators TxD & RxD, control indicators for outside memory interfacing WR & RD, etc. This is a domestic pull up port with quasi bi directional port within.

Pins 18 and 19:- These are employed for interfacing an outer crystal to give system clock.

Pin 20:- Titled as Vss – it symbolizes ground (0 V) association.

Pins- 21-28:- recognized as Port 2 (P 2.0 – P 2.7) – other than serving as Input/output port, senior order address bus indicators are multiplexed with this quasi bi directional port.

Pin- 29:- Program Store Enable or PSEN is employed to interpret sign from outer program memory.

Pin-30:- External Access or EA input is employed to permit or prohibit outer memory interfacing. If there is no outer memory need, this pin is dragged high by linking it to Vcc. Pin-31:- Aka Address Latch Enable or ALE is brought into play to de-multiplex the address data indication of port 0 (for outer memory interfacing). Two ALE throbs are obtainable for every machine rotation. Pins 32-39: recognized as Port 0 (P0.0 to P0.7) – other than serving as Input/output port, low order data & address bus signals are multiplexed with this port (to provide the use of outer memory interfacing). This pin is a bi directional Input/output port (the single one in microcontroller 8051) and outer pull up resistors are necessary to utilize this port as Input/output.

Pin-40: termed as Vcc is the chief power supply. By and large it is +5V DC.

Each chip also contains:

- four 8-bit input/output (I/0) ports
- serial interface
- 64K external code memory space
- 64K external data memory space
- Boolean processor
- 210 bit-addressable locations
- 4us multiply/divide
- On chip Code memory-4K ROM
- On chip data memory-128 bytes
- Timers-2

8051 Memory Organisation

Internal ROM

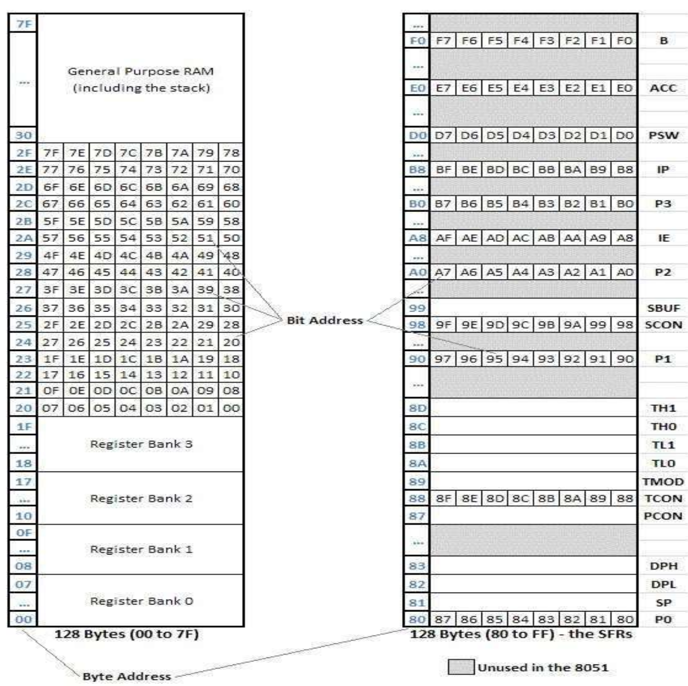
The 8051 has 4K (4096 locations) of on-chip ROM. This is used for storing the system program. $2^{12} = 4096$, therefore the internal ROM address bus is 12 bits wide and internal ROM locations go from 000H to FFFH.

Internal RAM

There are 256 bytes of internal RAM on the 8051. $2^8 = 256$, therefore the internal RAM address bus is 8 bits wide and internal RAM locations go from 00H to FFH.

The first 128 locations (00H to 7FH) of internal RAM are used by the programmer for storing data while the second 128 locations (80H to FFH) are the Special Function Registers (SFRs) which we will deal with later.

The diagram below is a summary of the 8051 on-chip RAM.



Register Banks

There are four register banks from 00H to 1FH. On power-up, registers R0 to R7 are located at 00H to 07H. However, this can be changed so that the register set points to any of the other three banks (if you change to Bank 2, for example, R0 to R7 is now located at 10H to 17H).

Bit-addressable Locations

The 8051 contains 210 bit-addressable locations of which 128 are at locations 20H to 2FH while the rest are in the SFRs. Each of the 128 bits from 20H to 2FH have a unique number (address) attached to them, as shown in the table above. The 8051 instruction set allows you to set or reset any single bit in this section of RAM.

With the general purpose RAM from 30H to 7FH and the register banks from 00H to 1FH, you may only read or write a full byte (8 bits) at these locations. However, with bit-addressable RAM (20H to 2FH) you can read or write any single bit in this region by using the unique address for that bit. We will later see that this is a very powerful feature.

Special Function Registers (SFRs)

Locations 80H to FFH contain the special function registers. As you can see from the diagram above, not all locations are used by the 8051 (eleven locations are blank). These extra locations are used by other family members (8052, etc.) for the extra features these microcontrollers possess.

Also note that not all SFRs are bit-addressable. Those that are have a unique address for each bit.

We will deal with each of the SFRs as we progress through the course, but for the moment you should take note of the accumulator (ACC) at address E0H and the four port registers at addresses 80H for P0, 90h for P1, A0 for P2 and B0 for P3.

We will later see how easy this makes reading from and writing to any of the four ports.

The Missing Registers

The two registers not shown in the table above are the instruction register and the program counter. These two very important registers are not placed in RAM because it is not necessary to make them directly accessible to the programmer. The instruction register gets its data from the execution cycle while the program counter is manipulated by the operation of these instructions. Introduction to 8051 Microcontroller Instruction Set

Writing a Program for any Microcontroller consists of giving commands to the

Microcontroller in a particular order in which they must be executed in order to perform a specific task. The commands to the Microcontroller are known as a Microcontroller's Instruction Set.

Just as our sentences are made of words, a Microcontroller's (for that matter, any computer) program is made of Instructions. Instructions written in a program tell the Microcontroller which operation to carry out.

An Instruction Set is unique to a family of computers. This tutorial introduces the 8051 Microcontroller Instruction Set also called as the MCS-51 Instruction Set.

As the 8051 family of Microcontrollers are 8-bit processors, the 8051 Microcontroller Instruction Set is optimized for 8-bit control applications. As a typical 8-bit processor, the 8051 Microcontroller instructions have 8-bit Opcodes. As a result, the 8051 Microcontroller instruction set can have up to $2^8 = 256$ Instructions.

A Brief Look at 8051 Microcontroller Instructions and Groups

Before going into the details of the 8051 Microcontroller Instruction Set, Types of Instructions and the Addressing Mode, let us take a brief look at the instructions and the instruction groups of the 8051 Microcontroller Instruction Set (the MCS51 Instruction Set).

The following table shows the 8051 Instruction Groups and Instructions in each group. There are 49 Instruction Mnemonics in the 8051 Microcontroller Instruction Set and these 49 Mnemonics are divided into five groups.

DATA TRANSFER	ARITHMETIC	LOGICAL	BOOLEAN	PROGRAM BRANCHING
MOV	ADD	ANL	CLR	LJMP
MOVC	ADDC	ORL	SETB	AJMP
MOVX	SUBB	XRL	MOV	SJMP
PUSH	INC	CLR	JC	JZ
POP	DEC	CPL	JNC	JNZ

XCH MUL RL JB CJNE XCHD DIV RLC JNB DJNZ

DA A RR JBC NOP

RRC ANL LCALL SWAP ORL ACALL

CPL RET

RETI

JMP

8051 Addressing Modes

What is an Addressing Mode?

An Addressing Mode is a way to locate a target Data, which is also called as Operand. The 8051 Family of Microcontrollers allows five types of Addressing Modes for addressing the Operands. They are:

- Immediate Addressing
- Register Addressing
- Direct Addressing
- Register – Indirect Addressing
- Indexed Addressing

Immediate Addressing

In Immediate Addressing mode, the operand, which follows the Opcode, is a constant data of either 8 or 16 bits. The name Immediate Addressing came from the fact that the constant data to be stored in the memory immediately follows the Opcode.
The constant value to be stored is specified in the instruction itself rather than taking from a register. The destination register to which the constant data must be copied should be the same size as the operand mentioned in the instruction.

Example: MOV A, #030H

Here, the Accumulator is loaded with 30 (hexadecimal). The # in the operand indicates that it is a data and not the address of a Register.

Immediate Addressing is very fast as the data to be loaded is given in the instruction itself.

Register Addressing

In the 8051 Microcontroller Memory Organization Tutorial, we have seen the organization of RAM and four banks of Working Registers with eight Registers in each bank.

In Register Addressing mode, one of the eight registers (R0 – R7) is specified as Operand in the Instruction.

It is important to select the appropriate Bank with the help of PSW Register. Let us see a example of Register Addressing assuming that Bank0 is selected.

Example: MOV A, R5

Here, the 8-bit content of the Register R5 of Bank0 is moved to the Accumulator.

Direct Addressing

In Direct Addressing Mode, the address of the data is specified as the Operand in the instruction. Using Direct Addressing Mode, we can access any register or onchip variable. This includes general purpose RAM, SFRs, I/O Ports, Control registers.

Example: MOV A, 47H

Here, the data in the RAM location 47H is moved to the Accumulator.

Register Indirect Addressing

In the Indirect Addressing Mode or Register Indirect Addressing Mode, the address of the Operand is specified as the content of a Register. This will be clearer with an example.

Example: MOV A, @R1

The @ symbol indicates that the addressing mode is indirect. If the contents of R1 is 56H, for example, then the operand is in the internal RAM location 56H. If the contents of the RAM location 56H is 24H, then 24H is moved into accumulator.

Only R0 and R1 are allowed in Indirect Addressing Mode. These register in the indirect addressing mode are called as Pointer registers.

Indexed Addressing Mode

With Indexed Addressing Mode, the effective address of the Operand is the sum of a base register and an offset register. The Base Register can be either Data Pointer (DPTR) or Program Counter (PC) while the Offset register is the Accumulator (A).

In Indexed Addressing Mode, only MOVC and JMP instructions can be used. Indexed Addressing Mode is useful when retrieving data from look-up tables.

Example: MOVC A, @A+DPTR

Here, the address for the operand is the sum of contents of DPTR and Accumulator.

NOTE: Some authors and textbooks add few other Addressing Modes like Absolute Addressing Mode, Relative Addressing Mode and Long Addressing Mode.

Types of Instructions in 8051 Microcontroller Instruction Set

Before seeing the types of instructions, let us see the structure of the 8051 Microcontroller Instruction. An 8051 Instruction consists of an Opcode (short of Operation – Code) followed by Operand(s) of size Zero Byte, One Byte or Two Bytes.
The Op-Code part of the instruction contains the Mnemonic, which specifies the type of operation to be performed. All Mnemonics or the Opcode part of the instruction are of One Byte size.
Coming to the Operand part of the instruction, it defines the data being processed by the instructions. The operand can be any of the following:

- No Operand
- Data value
- I/O Port
- Memory Location
- CPU register

There can multiple operands and the format of instruction is as follows:

MNEMONIC DESTINATION OPERAND, SOURCE OPERAND

A simple instruction consists of just the opcode. Other instructions may include one or more operands. Instruction can be one-byte instruction, which contains only opcode, or two-byte instructions, where the second byte is the operand or three byte instructions, where the operand makes up the second and third byte.

Based on the operation they perform, all the instructions in the 8051 Microcontroller Instruction Set are divided into five groups. They are:

- Data Transfer Instructions
- Arithmetic Instructions
- Logical Instructions
- Boolean or Bit Manipulation Instructions
- Program Branching Instructions

We will now see about these instructions briefly.

Data Transfer Instructions

The Data Transfer Instructions are associated with transfer with data between registers or external program memory or external data memory. The Mnemonics associated with Data Transfer are given below.

- MOV
- MOVC
- MOVX
- PUSH
- POP
- XCH
- XCHD

The following table lists out all the possible data transfer instruction along with other details like addressing mode, size occupied and number machine cycles it takes.

Mnemonic	Instruction	Description	Addressing Mode	# of Bytes	# of Cycles
MOV	A, #Data	A ← Data	Immediate	2	1
	A, Rn	A ← Rn	Register	1	1
	A, Direct	A ← (Direct)	Direct	2	1
	A, @Ri	A ← @Ri	Indirect	1	1
	Rn, #Data	Rn ← data	Immediate	2	1
	Rn, A	Rn ← A	Register	1	1
	Rn, Direct	Rn ← (Direct)	Direct	2	2
	Direct, A	(Direct) ← A	Direct	2	1
	Direct, Rn	(Direct) ← Rn	Direct	2	2
	Direct1, Direct2	(Direct1) ← (Direct2)	Direct	3	2
	Direct, @Ri	(Direct) ← @Ri	Indirect	2	2
	Direct, #Data	(Direct) ← #Data	Direct	3	2
	@Ri, A	@Ri ← A	Indirect	1	1
	@Ri, Direct	@Ri ← Direct	Indirect	2	2
	@Ri, #Data	@Ri ← #Data	Indirect	2	1
	DPTR, #Data16	DPTR ← #Data16	Immediate	3	2
MOVC	A, @A+DPTR	A ← Code Pointed by A+DPTR	Indexed	1	2
	A, @A+PC	A ← Code Pointed by A+PC	Indexed	1	2
	A, @Ri	A ← Code Pointed by Ri (8-bit Address)	Indirect	1	2
MOVX	A, @DPTR	A ← External Data Pointed by DPTR	Indirect	1	2
	@Ri, A	@Ri ← A (External Data 8-bit Addr)	Indirect	1	2
	@DPTR, A	@DPTR ← A (External Data 16-bit Addr)	Indirect	1	2
PUSH	Direct	Stack Pointer SP ← (Direct)	Direct	2	2
POP	Direct	(Direct) ← Stack Pointer SP	Direct	2	2
XCH	Rn	Exchange ACC with Rn	Register	1	1
	Direct	Exchange ACC with Direct Byte	Direct	2	1
	@Ri	Exchange ACC with Indirect RAM	Indirect	1	1
XCHD	A, @Ri	Exchange ACC with Lower Order Indirect RAM	Indirect	1	1

Arithmetic Instructions

Using Arithmetic Instructions, you can perform addition, subtraction, multiplication and division. The arithmetic instructions also include increment by one, decrement by one and a special instruction called Decimal Adjust Accumulator.

The Mnemonics associated with the Arithmetic Instructions of the 8051 Microcontroller Instruction Set are:

- ADD
- ADDC
- SUBB
- INC
- DEC
- MUL
- DIV
- DA A

The arithmetic instructions has no knowledge about the data format i.e. signed, unsigned, ASCII, BCD, etc. Also, the operations performed by the arithmetic instructions affect flags like carry, overflow, zero, etc. in the PSW Register.

All the possible Mnemonics associated with Arithmetic Instructions are mentioned in the following table.

Mnemonic	Instruction	Description	Addressing Mode	# of Bytes	# of Cycles
ADD	A, #Data	A ← A + Data	Immediate	2	1
	A, Rn	A ← A + Rn	Register	1	1
	A, Direct	A ← A + (Direct)	Direct	2	1
	A, @Ri	A ← A + @Ri	Indirect	1	1
ADDC	A, #Data	A ← A + Data + C	Immediate	2	1
	A, Rn	A ← A + Rn + C	Register	1	1
	A, Direct	A ← A + (Direct) + C	Direct	2	1
	A, @Ri	A ← A + @Ri + C	Indirect	1	1
SUBB	A, #Data	A ← A – Data – C	Immediate	2	1
	A, Rn	A ← A – Rn – C	Register	1	1
	A, Direct	A ← A – (Direct) – C	Direct	2	1
	A, @Ri	A ← A – @Ri – C	Indirect	1	1
MUL	AB	Multiply A with B (A ← Lower Byte of A*B and B ← Higher Byte of A*B)	--	1	4
DIV	AB	Divide A by B (A ← Quotient and B ← Remainder)	--	1	4
DEC	A	A ← A – 1	Register	1	1
	Rn	Rn ← Rn – 1	Register	1	1
	Direct	(Direct) ← (Direct) – 1	Direct	2	1
	@Ri	@Ri ← @Ri – 1	Indirect	1	1
INC	A	A ← A + 1	Register	1	1
	Rn	Rn ← Rn + 1	Register	1	1
	Direct	(Direct) ← (Direct) + 1	Direct	2	1
	@Ri	@Ri ← @Ri + 1	Indirect	1	1
	DPTR	DPTR ← DPTR + 1	Register	1	2
DA	A	Decimal Adjust Accumulator	--	1	1

Logical Instructions

The next group of instructions are the Logical Instructions, which perform logical operations like AND, OR, XOR, NOT, Rotate, Clear and Swap. Logical Instruction are performed on Bytes of data on a bit-by-bit basis. Mnemonics

associated with Logical Instructions are as follows:

- ANL
- ORL
- XRL

- CLR
- CPL
- RL
- RLC
- RR
- RRC
- SWAP

The following table shows all the possible Mnemonics of the Logical Instructions.

Mnemonic	Instruction	Description	Addressing Mode	# of Bytes	# of Cycles
ANL	A, #Data	A ← A AND Data	Immediate	2	1
	A, Rn	A ← A AND Rn	Register	1	1
	A, Direct	A ← A AND (Direct)	Direct	2	1
	A, @Ri	A ← A AND @Ri	Indirect	1	1
	Direct, A	(Direct) ← (Direct) AND A	Direct	2	1
	Direct, #Data	(Direct) ← (Direct) AND #Data	Direct	3	2
ORL	A, #Data	A ← A OR Data	Immediate	2	1
	A, Rn	A ← A OR Rn	Register	1	1
	A, Direct	A ← A OR (Direct)	Direct	2	1
	A, @Ri	A ← A OR @Ri	Indirect	1	1
	Direct, A	(Direct) ← (Direct) OR A	Direct	2	1
	Direct, #Data	(Direct) ← (Direct) OR #Data	Direct	3	2
XRL	A, #Data	A ← A XRL Data	Immediate	2	1
	A, Rn	A ← A XRL Rn	Register	1	1
	A, Direct	A ← A XRL (Direct)	Direct	2	1
	A, @Ri	A ← A XRL @Ri	Indirect	1	1
	Direct, A	(Direct) ← (Direct) XRL A	Direct	2	1
	Direct, #Data	(Direct) ← (Direct) XRL #Data	Direct	3	2
CLR	A	A← 00H	--	1	1
CPL	A	A ← A	--	1	1
RL	A	Rotate ACC Left	--	1	1
RLC	A	Rotate ACC Left through Carry	--	1	1
RR	A	Rotate ACC Right	--	1	1
RRC	A	Rotate ACC Right through Carry	--	1	1
SWAP	A	Swap Nibbles within ACC	--	1	1

Boolean or Bit Manipulation Instructions

As the name suggests, Boolean or Bit Manipulation Instructions will deal with bit variables. We know that there is a special bit-addressable area in the RAM and some of the Special Function Registers (SFRs) are also bit addressable.

The Mnemonics corresponding to the Boolean or Bit Manipulation instructions are:

- CLR
- SETB
- MOV

- JC
- JNC
- JB
- JNB
- JBC
- ANL
- ORL
- CPL

These instructions can perform set, clear, and, or, complement etc. at bit level. All the possible mnemonics of the Boolean Instructions are specified in the following table.

Mnemonic	Instruction	Description	# of Bytes	# of Cycles
CLR	C	C ← 0 (C = Carry Bit)	1	1
	Bit	Bit ← 0 (Bit = Direct Bit)	2	1
SET	C	C ← 1	1	1
	Bit	Bit ← 1	2	1
CPL	C	C ← $\overline{\text{C}}$	1	1
	Bit	Bit ← $\overline{\text{Bit}}$	2	1
ANL	C, /Bit	C ← C. $\overline{\text{Bit}}$ (AND)	2	1
	C, Bit	C ← C . Bit (AND)	2	1
ORL	C, /Bit	C ← C + $\overline{\text{Bit}}$ (OR)	2	1
	C, Bit	C ← C + Bit (OR)	2	1
MOV	C, Bit	C ← Bit	2	1
	Bit, C	Bit ← C	2	2
JC	rel	Jump is Carry (C) is Set	2	2
JNC	rel	Jump is Carry (C) is Not Set	2	2
JB	Bit, rel	Jump is Direct Bit is Set	3	2
JNB	Bit, rel	Jump is Direct Bit is Not Set	3	2
JBC	Bit, rel	Jump is Direct Bit is Set and Clear Bit	3	2

Program Branching Instructions

The last group of instructions in the 8051 Microcontroller Instruction Set are the Program Branching Instructions. These instructions control the flow of program logic. The mnemonics of the Program Branching Instructions are as follows.

- LJMP
- AJMP
- SJMP
- JZ
- JNZ
- CJNE

- DJNZ
- NOP
- LCALL
- A CALL
- RET
- RETI
- JMP

All these instructions, except the NOP (No Operation) affect the Program Counter (PC) in one way or other. Some of these instructions has decision making capability before transferring control to other part of the program.

The following table shows all the mnemonics with respect to the program branching instructions.

Mnemonic	Instruction	Description	# of Bytes	# of Cycles
ACALL	ADDR11	Absolute Subroutine Call PC + 2 → (SP); ADDR11 → PC	2	2
LCALL	ADDR16	Long Subroutine Call PC + 3 → (SP); ADDR16 → PC	3	2
RET	--	Return from Subroutine (SP) → PC	1	2
RETI	--	Return from Interrupt	1	2
AJMP	ADDR11	Absolute Jump ADDR11 → PC	2	2
LJMP	ADDR16	Long Jump ADDR16 → PC	3	2
SJMP	rel	Short Jump PC + 2 + rel → PC	2	2
JMP	@A + DPTR	A + DPTR → PC	1	2
JZ	rel	If A=0, Jump to PC + rel	2	2
JNZ	rel	If A ≠ 0, Jump to PC + rel		
CJNE	A, Direct, rel	Compare (Direct) with A. Jump to PC + rel if not equal	3	2
	A, #Data, rel	Compare #Data with A. Jump to PC + rel if not equal	3	2
	Rn, #Data, rel	Compare #Data with Rn. Jump to PC + rel if not equal	3	2
	@Ri, #Data, rel	Compare #Data with @Ri. Jump to PC + rel if not equal	3	2
DJNZ	Rn, rel	Decrement Rn. Jump to PC + rel if not zero	2	2
	Direct, rel	Decrement (Direct). Jump to PC + rel if not zero	3	2
NOP		No Operation	1	1

PARALLEL AND SERIAL ADC

This section will explore interfacing of both parallel and serial ADC chips to microcontrollers. First, we describe the ADC0804 chip, then show how to interface it to the 8051. Then we examine the ADC0808/0809 and ADC0848 characteristics and show how to interface them to the 8051. At the end of this section, we describe the serial ADC chip MAX 1112 and program it in both in C and Assembly.

ADC devices

Analog-to-digital converters are among the most widely used devices for data acquisition. Digital computers use binary (discrete) values, but in the physical world everything is analog (continuous). Temperature, pressure (wind or liquid), humidity, and velocity are a few examples of physical

quantities that we deal with every day. A physical quantity is converted to electrical

(voltage, current) signals using a device called a transducer. Transducers are also referred to as sensors. Sensors for temperature, velocity, pressure, light, and many other natural quantities produce an output that is voltage (or current). Therefore, we need an analog-to-digital converter to translate the analog signals to digital numbers so that the microcontroller can read and process them. An ADC has n-bit resolution where n can be 8, 10, 12, 16 or even 24 bits. The higherresolution ADC provides a smaller step size, where step size is the smallest change that can be discerned by an ADC. This is shown in Table. In this chapter we examine several 8-bit ADC chips. In addition to resolution, conversion time is another major factor in judging an ADC. Conversion time is defined as the time it takes the ADC to convert the analog input to a digital (binary) number. The ADC chips are either parallel or serial. In parallel ADC, we have 8 or more pins dedicated to bringing out the binary data, but in serial ADC we have only one pin for data out. Serial ADCs are discussed at the end of this section.

Table : Resolution vs. Step Size for ADC

n-bit	Number of steps	Step Size (mV)
8	256	5/256 = 19.53
10	1024	5/1024 = 4.88
12	4096	5/4096 = 1.2
16	65536	5/65536 = 0.076

Notes: $V_{cc} = 5$ V

Step size (resolution) is the smallest change that can be discerned by an ADC.

Module 6

Structure of the 8051 Microcontroller Assembly Language

The Structure or Syntax of the 8051 Microcontroller Assembly Language is discussed here. Each line or statement of the assembly language program of 8051 Microcontroller consists of three fields: Label, Instruction and Comments.

The arrangement of these fields or the order in which they appear is shown below. [Label:] Instructions [//Comments]

NOTE: The brackets for Label and Comments mean that these fields are optional and may not be used in all statements in a program.

Before seeing about these three fields, let us first see an example of how a typical statement or line in an 8051 Microcontroller Assembly Language looks like.

TESTLABEL: MOV A, 24H ; THIS IS A SAMPLE COMMENT

In the above statement, the "TESTLABEL" is the name of the Label, "MOV A, 24H" is the Instruction and the "THIS IS A SAMPLE COMMENT" is a Comment.

Label

The Label is programmer chosen name for a Memory Location or a statement in a program. The Label part of the statement is optional and if present, the Label must be terminated with a Colon (:).

An important point to remember while selecting a name for the Label is that they should reduce the need for documentation.

Instruction

The Instruction is the main part of the 8051 Microcontroller Assembly Language Programming as it is responsible for the task performed by the Microcontroller. Any Instruction in the Assembly Language consists of two parts: Op-code and Operand(s).

The first part of the Instruction is the Op-code, which is short for Operation Code, specifies the operation to be performed by the Microcontroller. Op-codes in Assembly Language are called as Mnemonics. Op-codes are in binary format (used in Machine Language) while the Mnemonic (which are equivalent to Op-codes) are English like statements.

The second part of the instruction is called the Operand(s) and it represents the Data on which the operation is performed. There are two types of Operands: the Source Operand and the Destination Operand. The Source Operand is the Input of the operation and the Destination Operand is where the result is stored.

Comments

The last part of the Structure of 8051 Assembly Language is the Comments. Comments are statements included by the developer for easier understanding of the code and is used for proper documentation of the Program.

Comments are optional and if used, they must begin with a semicolon (;) or double slash (//) depending on the Assembler.

The following statements will show a few possible ways of using Label, Instruction and Comments.

Label without instruction and comment: LABEL:
Line with Label and Instruction: LABEL: MOV A, 22H
Line with Instruction and Comment: MOV A, 22H ; THIS IS A COMMENT
Line with Label and Comment: LABEL: ; THIS IS A COMMENT
Line with only Comment: ; THIS IS A COMMENT

8051 Microcontroller Assembly Language Directives

Assembly Language Directives are not the instructions to the 8051 Microcontroller Assembler even though they are written in the Mnemonic field of the program. Assembly Language Directives are actually instructions to the Assembler and directs the Assembler Program what to do during the process of Assembling.

The Assembly Language Directives do not have any effect on the contents of the 8051 Microcontroller Memory (except DB and DW directives).

These Directives are dependent on the Assembler Program and in case of ASM51 Assembler, the following are the categories of Directives.

Assembler State Control	ORG, END, USING
Symbol Definition	SEGMENT, EQU, SET, DATA, IDATA, XDATA, BIT, CODE
Storage Initialization	DS, DBIT, DB, DW
Program Linkage	PUBLIC, EXTERN, NAME
Segment Selection	RSEG, CSEG, DSEG, ISEG, BSEG, XSEG

We will now see about few of the important and frequently used Assembly Language Directives.

ORG – Set Origin

The 8051 Microcontroller Assembly Language Program will start assembling from the Program Memory Address 0000H. This is also the address from which the 8051 Microcontroller will start executing the code.

In order place the Program and Data anywhere in the Address Space of the 8051 Microcontroller, you can use the ORG Directive.

Examples

ORG 0000H ; Tells the Assembler to assemble the next statement at 0000H LJMP MAIN

; Code Memory at 0000H. Jump to MAIN.

ORG 000BH ; Tells the Assembler to assemble the next statement at 000BH MAIN: NOP

; Code Memory at 000BH. MAIN starts here.

DB – Define Byte

The DB Directive is used to define a Byte type variable. Using this directive, you can define data in Decimal, Binary, HEX or ASCII formats. There should be a suffix of 'B' for binary and 'H' for HEX. The ASCII Characters are placed in single quotation marks (like 'string').

Examples

```
ORG 0000H
DB 10              ; Define Byte 10 (Decimal) and store at 0000H
DB 30H             ; Define Byte 30 (HEX) and store at 0001H
DB 'STRING'        ; Define String 'STRING' and store at 0002H to 0007H
DB 00001111B       ; Define Byte 00001111 (Binary) and store at 0008H
DB 1234H           ; Define Byte 34 (HEX) and store at 0009H. Only lower byte is
accepted as DB can allocate only a Byte of Memory.
```

DW – Define Word

The Define Word (DW) Directive is used to include a 16-bit data in a program. The functionality of DW is similar to that of DB except that DW generates 16-bit values.

EQU – Equate

Using the EQU Directive, you can associate a Symbol (or Label) with a Value.

Examples

```
TMP EQU #30        ; Assigns the value #30 to the name TMP  RED_LED
EQU P1.0      ; P1.0 is defined as RED_LED
```

END

The END Directive is used to stop the assembling process. This should be the last statement in the program. END Directive cannot have a Label and the statements beyond END will not be processed by the Assembler.

Example

```
ORG 0000H
MOV A, 20H
MOV R0, #30
END
```

Examples of 8051 Microcontroller Assembly Language Programming Example 1

The following is a simple Assembly Language for 8051 Microcontroller which copies the data from R0 of Bank0 to R0 of Bank3.

```
ORG 00H
MOV R0, #33H
MOV A, R0
```

SETB PSW.3
SETB PSW.4
MOV R0, A END

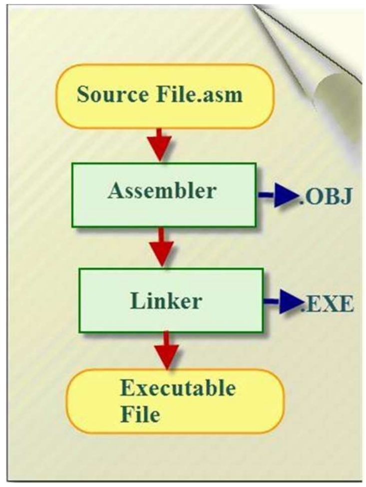
Example 2

In the next example, you can Toggle the LEDs ON and OFF (Blinking LEDs) that are connected to PORT1 of the 8051 Microcontroller.

```
ORG 00H                         ; Assembly Starts from 0000H.
 ; Main Program
 START:      MOV P1, #0XFF              ; Move 11111111 to PORT1.
             CALL WAIT  ; Call WAIT
             MOV A, P1  ; Move P1 value to ACC
             CPL A                ; Complement ACC
             MOV P1, A  ; Move ACC value to P1
             CALL WAIT  ; Call WAIT
             SJMP START           ; Jump to START
 WAIT:       MOV R2, #10          ; Load Register R2 with 10 (0x0A)
 WAIT1:      MOV R3, #200               ; Load Register R3 with 10 (0xC8)
 WAIT2:      MOV R4, #200               ; Load Register R4 with 10 (0xC8)
             DJNZ R4, $  ; Decrement R4 till it is 0. Stay there if not 0.
             DJNZ R3, WAIT2       ; Decrement R3 till it is 0. Jump to WAIT2 if not 0.
             DJNZ R2, WAIT1       ; Decrement R2 till it is 0. Jump to WAIT1 if not 0.
             RET            ; Retu rn to Main Program
 END                            ; End Assembly
```

8051 Programming in Assembly Language

The assembly language is a fully hardware related programming language. The embedded designers must have sufficient knowledge on hardware of particular processor or controllers before writing the program. The assembly language is developed by mnemonics; therefore, users cannot understand it easily to modify the program.

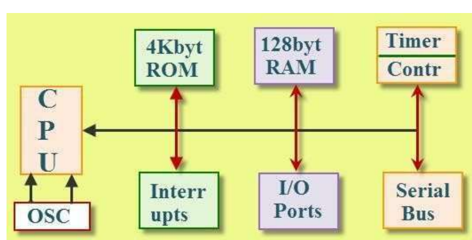


8051 Programming in Assembly Language

Assembly programming language is developed by various compilers and the “keiluvison” is best suitable for microcontroller programming development. Microcontrollers or processors can understand only binary language in the form of ‘0s or 1s’; An assembler converts the assembly language to binary language, and then stores it in the microcontroller memory to perform the specific task.

8051 Microcontroller Architecuture

The 8051 microcontroller is the CISC based Harvard architecture, and it has peripherals like 32 I/O, timers/counters, serial communication and memories. The microcontroller requires a program to perform the operations that require a memory for saving and to read the functions. The 8051 microcontroller consists of RAM and ROM memories to store instructions.



8051 Microcontroller Architecuture

A Register is the main part in the processors and microcontrollers which is contained in the memory that provides a faster way of collecting and storing the data. The 8051 assembly language programming is based on the memory registers. If we want to manipulate data to a processor or controller by performing subtraction, addition, etc., we cannot do that directly in the memory, but it needs registers to process and to store the data. Microcontrollers contain several types of registers that can be classified according to their instructions or content that operate in them.

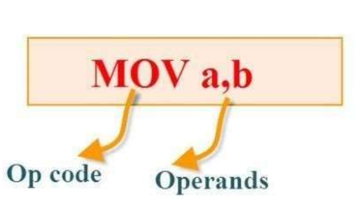
8051 Microcontroller Programs in Assembly Language

The assembly language is made up of elements which all are used to write the program in sequential manner. Follow the given rules to write programming in assembly language.

Rules of Assembly Language

- The assembly code must be written in upper case letters
- The labels must be followed by a colon (label:)
- All symbols and labels must begin with a letter
- All comments are typed in lower case
- The last line of the program must be the END directive

The assembly language mnemonics are in the form of op-code, such as MOV, ADD, JMP, and so on, which are used to perform the operations.



Op-code: The op-code is a single instruction that can be executed by the CPU. Here the op-code is a MOV instruction.

Operands: The operands are a single piece of data that can be operated by the op-code. Example, multiplication operation is performed by the operands that are multiplied by the operand.

Syntax: MUL a,b;

The Elements of an Assembly Language Programming:

- Assembler Directives
- Instruction Set
- Addressing Modes

END

Addressing Modes:

The way of accessing data is called addressing mode. The CPU can access the data in different ways by using addressing modes. The 8051 microcontroller consists of five addressing modes such as:

- Immediate Addressing Mode Register Addressing Mode
- Direct Addressing Mode

- Indirect Addressing Mode
- Base Index Addressing Mode Immediate Addressing Mode:

In this addressing mode, the source must be a value that can be followed by the '#' and destination must be SFR registers, general purpose registers and address. It is used for immediately storing the value in the memory registers.

Syntax:

MOV A, #20h //A is an accumulator register, 20 is stored in the A//

MOV R0,#15 // R0 is a general purpose register; 15 is stored in the R0 register//

MOV P0, #07h //P0 is a SFR register;07 is stored in the P0//

MOV 20h,#05h //20h is the address of the register; 05 stored in the 20h// Ex:

MOV R0, #1

MOV R0, #20 //R0 <—R0[15]+20, the final value is stored in R0//

Register Addressing Mode:

In this addressing mode, the source and destination must be a register, but not general purpose registers. So the data is not moved within the general purpose bank registers.

Syntax:

MOV A, B; // A is a SFR register, B is a general purpose register//

MOV R0, R1 //Invalid instruction, GPR to GPR not possible//

EX:

MOV R0, #02h

MOV A, #30h

ADD R0, A //R0<—R0+A, the final value is stored in the R0 register//

Direct Addressing Mode

In this addressing mode, the source or destination (or both source and destination) must be an address, but not value.

Syntax:

MOV A,20h // 20h is an address; A is a register// MOV 00h, 07h // both are addressed of the GPS registers// Ex:

MOV 07h,#01h

MOV A, #08h

ADD A,07h //A<—A+07h the final value is stored in A//

Indirect Addressing Mode:

In this addressing mode, the source or destination (or destination or source) must be a indirect address, but not a value. This addressing mode supports the pointer concept. The pointer is a variable that is used to store the address of the other variable. This pointer concept is only used for R0 and R1 registers.

Syntax:

MOVR0, #01h //01 value is stored in the R0 register, R0 address is 08h//

MOV R1, #08h//R1 is the pointer variable that stores address (08h) of R0 //

MOV 20h,@R1 //01 value is stored in the 20h address of the GP register//

Indirect Addressing Mode

Base Index Addressing Mode:

This addressing mode is used to read the data from the external memory or ROM memory. All addressing modes cannot read the data from the code memory. The code must read through the DPTR register. The DPTR is used to point the data in the code or external memory.

Syntax:

MOVC A, @A+DPTR //C indicates code memory//

MOCX A, @A+DPTR // X indicate external memory//

EX: MOV A, #00H //00H is stored in the A register//

MOV DPTR, #0500H //DPTR points 0500h address in the memory//

MOVC A, @A+DPTR //send the value to the A register//

MOV P0, A //date of A send to the PO registrar// Instruction Set:

The instruction set is the structure of the controller or processor that provides commands to the controller to guide the controller for processing data. The instruction set consists of instructions, native data types, addressing modes, interrupt registers, exceptional handling and memory architecture. The 8051 microcontroller can follow CISC instructions with Harvard architecture.

In case of the 8051 programming different types of CISC instructions include:

- Data Transfer Instruction set
- Sequential Instruction Set
- Arithmetic Instruction set
- Branching Instruction set
- Loop Instrcution Set
- Conditional Instruction set
- Unconditional Instruction set
- Logical Instruction set Boolean Instruction set Arithmetic Instruction Set:

The arithmetic instructions perform the basic operations such as:

- Addition
- Multiplication
- Subtraction
- Division

Addition:

ORG 0000h

MOV R0, #03H // move the value 3 to the register R0// MOV A, #05H // move the value 5 to accumulator A// Add A, 00H // addA value with R0 value and stores the result inA// END

Multiplication:

ORG 0000h

MOV R0, #03H // move the value 3 to the register R0// MOV A, #05H // move the value 5 to accumulator A//

MUL A, 03H // Multiplied result is stored in the Accumulator A //
END

Subtraction:

ORG 0000h

MOV R0, #03H   // move the value 3 to register R0//

MOV A, #05H   // move the value 5 to accumulator A//

SUBB A, 03H   // Result value is stored in the Accumulator A //

END

Division:

ORG 0000h

MOV R0, #03H   // move the value 3 to register R0//

MOV A, #15H   // move the value 5 to accumulator A//

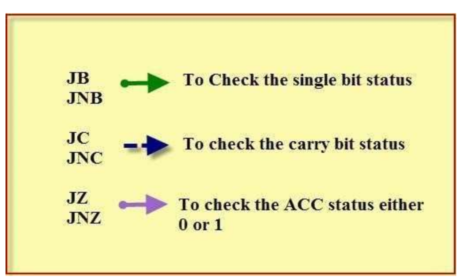
DIV A, 03H   // final value is stored in the Accumulator A // END

MNEMONICS	DESCRIPTION	BYTES
ADD A,Rr	A+Rr-->A	1
ADD A,@Rp	A+(Rp)-->A	2
ADD A, #N	A+N-->A	1
SUB A,Rr	A-Rr-->A	2
SUB A,@Rp	A-(Rp)-->A	1
SUB A, #N	A-N-->A	2
DEC A	A-1-->A	1
DEC Rr	Rr-1-->	2
DEC ADD	(ADD)-1-->ADD	1
INC A	A+1-->A	2
INC Rr	Rr+1-->	1
INC ADD	(ADD)+1-->ADD	2
DIV AB	A/B-->AB	1
MUL AB	A*B-->AB	2

Conditional Instructions

The CPU executes the instructions based on the condition by checking the single bit status or byte status. The 8051 microcontroller consists of various conditional instructions such as:

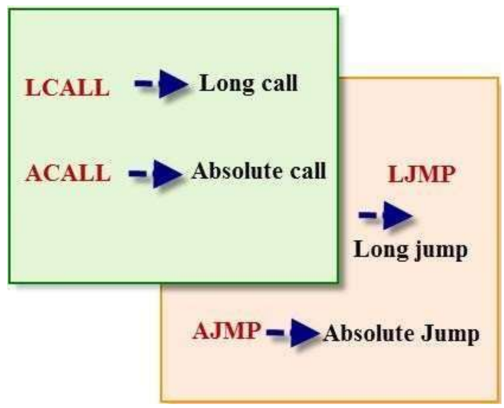
- JB —>Jump below
- JNB —> Jump if not below
- JC —> Jump if Carry
- JNC —>Jump if not Carry JZ —>Jump if Zero
- JNZ —> Jump if not Zero



Call and Jump Instructions:

The call and jump instructions are used to avoid the code replication of the program. When some specific code used more than once in different places in the program, if we mention specific name to code then we could use that name anywhere in the program without entering a code for every time. This reduces the complexity of the program. The 8051 programming consists of call and jump instructions such as LCALL, SJMP.

- LCALL
- ACALL
- SJMP
- LJMP



Loop Instructions:

The loop instructions are used to repeat the block each time while performing the increment and decrement operations. The 8051 microcontroller consist two types of loop instructions:

- CJNE —> compare and jump if not equal
- DJNZ —> decrement and jump if not zero

Logical Instruction Set:

The 8051 microcontroller instruction set provides the AND, OR, XOR, TEST, NOT and Boolean logic instructions for set and clears the bits based on the need in the program.

Mnemonics	Description	Bytes
ANL A, Rr	A AND Rr-->A	1
ANL A, ADD	A AND(ADD)-->A	2
ORL A, Rr	A OR Rr-->A	1
ORL A, ADD	A OR(ADD)-->A	2
XRL A, Rr	A XOR Rr-->A	1
XRL A, ADD	A XOR(ADD)-->A	2
CLR A	00-->A	1
CPL A	A bar-->A	2

1. Syntax:

MOV A, #20H /00100000/

MOV R0, #03H /00000101/

ORL A, R0 //00100000/00000101=00000000//

2. Syntax:

MOV A, #20H /00100000/

MOV R0, #03H /00000101/

ANL A, R0

3. Syntax:

MOV A, #20H /00100000/

MOV R0, #03H /00000101/

XRL A, R0

Shifting Operators

The shift operators are used for sending and receiving the data efficiently. The 8051 microcontroller consist four shift operators:

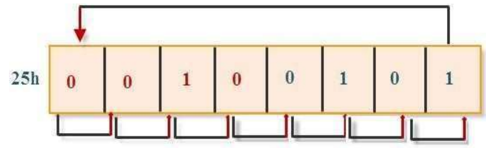
- RR —> Rotate Right
- RRC —>Rotate Right through carry
- RL —> Rotate Left
- RLC —>Rotate Left through carry Rotate Right (RR):

In this shifting operation, the MSB becomes LSB and all bits shift towards right side bit-by-bit, serially.

Syntax:

MOV A, #25h

RR A



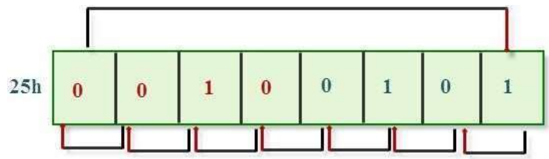
Rotate Left (RL):

In this shifting operation, the MSB becomes LSB and all bits shift towards Left side bit-by-bit, serially.

Syntax:

MOV A, #25h

RL A



RRC Rotate Right through Carry:

In this shifting operation, the LSB moves to carry and the carry becomes MSB, and all the bits are shift towards right side bit by bit position.

Syntax:

MOV A, #27h

RRC A

RLC Rotate Left through Carry:

In this shifting operation, the MSB moves to carry and the carry becomes LSB and all the bits shift towards left side in a bit-by-bit position.

Syntax:

MOV A, #27h

RLC A

Basic Embedded C Programs:

The microcontroller programming differs for each type of operating system. There are many operating systems such as Linux, Windows, RTOS and so on. However, RTOS has several advantages for embedded system development. Some of the Assembly level programming examples are given below.

LED blinking using with 8051 microcontroller:

- Number Displaying on 7-segment display using 8051 microcontroller
- Timer/Counter calculations and program using 8051 microcontroller
- Serial Communication calculations and program using 8051 microcontroller LED programs with 8051 Microcontrller

1. WAP to toggle the PORT1 LEDs

```
ORG 0000H
TOGLE: MOV P1, #01    //move 00000001 to the p1 register//
CALL DELAY    //execute the delay//
MOV A, P1        //move p1 value to the accumulator//
CPL A        //complement A value //
MOV P1, A        //move 11111110 to the port1 register//
CALL DELAY    //execute the delay//
SJMP TOGLE
DELAY: MOV R5, #10H    //load register R5 with 10//
TWO:        MOV R6, #200    //load register R6 with 200//
ONE:       MOV R7, #200    //load register R7 with 200//
DJNZ R7, $    //decrement R7 till it is zero//
DJNZ R6, ONE    //decrement R7 till it is zero//
DJNZ R5, TWO    //decrement R7 till it is zero//
RET            //go back to the main program //
END
```

Timer/Counter Calculations and Program using 8051 Microcontroller:

The delay is the one of the important factors in the application software development. The timers and counters are hardware components of the microcontroller, that are used in many applications to provide the accurate time delay with count pulses. Both the tasks are implemented by the software technique.

1. WAP to calculate the 500us time delay.

```
MOV TMOD, #10H        //select the timer mode by the registers//
MOV TH1, #0FEH        // store the delay time in higher bit//
```

```
MOV TL1, #32H        // store the delay time in low bit//
JNB TF1, $           //decrement the value of the timer till it is zero//
CLR TF1              //clear the timer flag bit//
CLR TR1              //OFF the timer//
```

2. WAP to toggle the LEDs withthe 5 sec time delay

```
ORG 0000H
RETURN: MOV PO, #00H
ACALL DELAY
MOV P0, #0FFH
ACALL DELAY
SJUMP RETURN
DELAY:   MOV R5, #50H      //load register R5 with 50//
DELAY1: MOV R6, #200    //load register R6 with 200//  DELAY2:
MOV R7, #229    //load register R7 with 200//
DJNZ R7, $    //decrement R7 till it is zero//
DJNZ R6, DELAY2//decrement R6 till it is zero//
DJNZ R5, DELAY1//decrement R5 till it is zero//
RET            //go back to the main program //
END
```

3. WAP to count the 250 pulses using mode0 count0 Syntax:

```
ORG 0000H
MOV TMOD, #50H        //select the counter//
MOV TH0, #15        //move the counting pulses higher bit//
MOV TH1, #9FH        //move the counting pulses, lower bit//
SET TR0            //ON the timer//
JNB  $           //decrement the count value till zero//
CLR TF0           //clear the counter, flag bit//
CLR TR0           //stop the timer//
END
```

Serial Communication Programming Using 8051 Microcontroller:

Serial communication is commonly used for transmitting and receiving the data. The 8051 microcontroller consist of UART/USART serial communication and the signals are transmitted and received by Tx and Rx pins. The UART communication transfers the data bitby-bit serially. The UART is a half-duplex protocol that transfers and receives the data, but not at the same time.

1. WAP to transmit the characters to the Hyper Terminal

```
MOV SCON, #50H      //set the serial communication//
MOV TMOD, #20H      //select the timer mode//
MOV TH1, #-3        //set the baud rate//
SET TR1             //ON the timer//
MOV SBUF, #'S'      //transmit S to the serial window //
JNB TI, $           //decrement value of the timer till it is zero//
CLR RI              // clear receive interrupt //
CLR TR1             //clear timer//
```

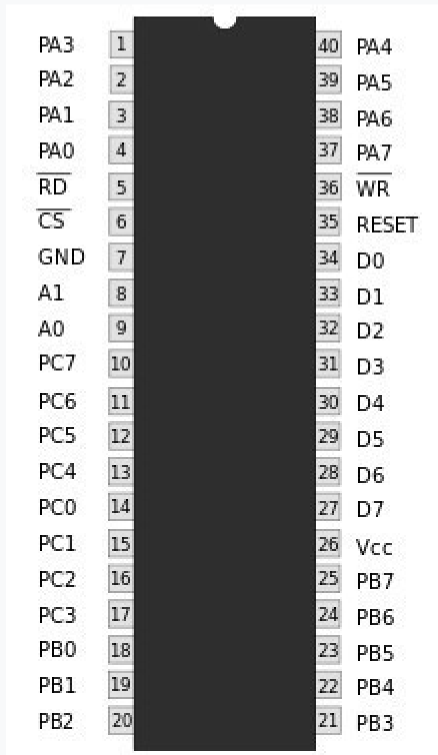
2. WAP to transmit the Receive the character by the Hyper Terminal

```
MOV SCON, #50H      //set the serial communication//
MOV TMOD, #20H      //select the timer mode//
MOV TH1, #-6        //set the baud rate//
SET TR1             //on the timer//
MOV SBUF, #'S'      //transmit S to the serial window //
JNB RI, $           //decrement value of timer till it is zero//
CLR RI              // clear receive interrupt //
MOV P0, SBUF        //send the SBUF register value to the port0// CLR TR1
//clear timer//
```

Module 7

Intel 8255

Intel D8255



Pinout of 8255

The Intel 8255 (or i8255) Programmable Peripheral Interface (PPI) chip was developed and manufactured by Intel in the first half of the 1970s for the Intel 8080 microprocessor. The 8255 provides 24 parallel input/output lines with a variety of programmable operating modes.

The 8255 is a member of the MCS-85 Family of chips, designed by Intel for use with their 8085 and 8086 microprocessors and their descendants[1]. It was first available in a 40pin DIP and later a 44-pin PLCC packages.[2] It found wide applicability in digital processing systems and was later cloned by other manufacturers. The 82C55 is a CMOS version for higher speed and lower current consumption.

The functionality of the 8255 is now mostly embedded in larger VLSI processing chips as a subfunction. The 8255 is still made but mostly used to expand the I/O of microcontrollers.

Function

The 8255 gives a CPU or digital system access to programmable parallel I/O.[4] The 8255 has 24 input/output pins.[5] These are divided into three 8-bit ports (A, B, C).[6] Port A and port B can be used as 8-bit input/output ports. Port C can be used as an 8-bit input/output port or as two 4-bit input/output ports or to produce handshake signals for ports A and B.

The three ports are further grouped as follows:

---

1. Group A consisting of port A and upper part of port C.
2. Group B consisting of port B and lower part of port C.

Eight data lines (D0–D7) are available (with an 8-bit data buffer) to read/write data into the ports

or control register under the status of the RD (pin 5) and WR (pin 36), which are active-low signals for read and write operations respectively. Address lines A_1 and A_0 allow to access a data register for each port or a control register, as listed below:

A_1	A_0	Port selected
0	0	port A
0	1	port B
1	0	port C
1	1	control register

The control signal chip select CS (pin 6) is used to enable the 8255 chip. It is an active-

low signal, i.e., when CS = 0, the 8255 is enabled. The RESET input (pin 35) is connected to the RESET line of system like 8085, 8086, etc., so that when the system is reset, all the ports are initialized as input lines. This is done to prevent 8255 and/or any peripheral connected to it from being destroyed due to mismatch of ports. As an example, consider an input device connected to 8255 at port A. If from the previous operation, port A is initialized as an output port and if 8255 is not reset before using the current configuration, then there is a possibility of damage of either the input device connected or 8255 or both, since both 8255 and the device connected will be sending out data.
The control register (or the control logic, or the command word register) is an 8-bit register used to select the modes of operation and input/output designation of the ports.[7]

Operational modes of 8255

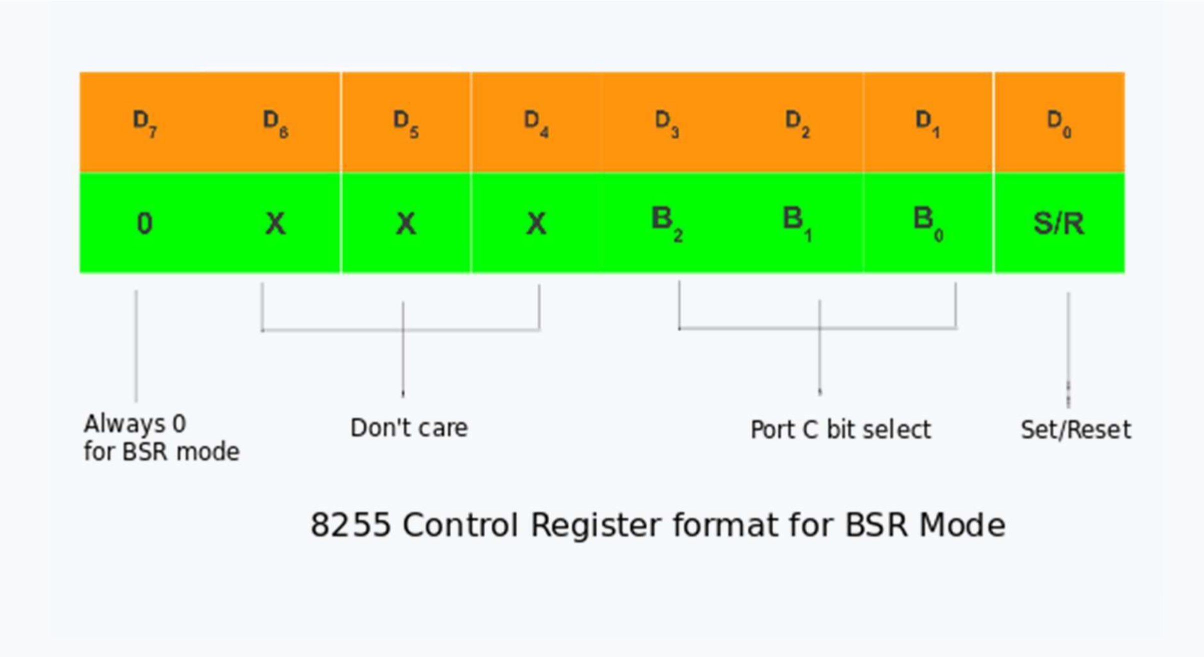
There are two basic operational modes of 8255:

- Bit Set/Reset mode (BSR mode). □ Input/Output mode (I/O mode).

The two modes are selected on the basis of the value present at the D_7 bit of the control word register. When $D_7 = 1$, 8255 operates in I/O mode, and when $D_7 = 0$, it operates in the BSR mode.

Bit Set/Reset (BSR) mode

The Bit Set/Reset (BSR) mode is available on port C only. Each line of port C (PC_7 - PC_0) can be set or reset by writing a suitable value to the control word register. BSR mode and I/O mode are independent and selection of BSR mode does not affect the operation of other ports in I/O mode.[8]



8255 BSR mode

- D_7 bit is always 0 for BSR mode.
- Bits D_6, D_5 and D_4 are don't care bits.
- Bits D_3, D_2 and D_1 are used to select the pin of Port C. Bit D_0 is used to set/reset the selected pin of Port C.

Selection of port C pin is determined as follows:

D3	D2	D1	Bit/pin of port C selected
0	0	0	PC_0

0	0	1	PC_1
0	1	0	PC_2
0	1	1	PC_3
1	0	0	PC_4
1	0	1	PC_5
1	1	0	PC_6
1	1	1	PC_7

As an example, if it is needed that PC_5 be set, then in the control word,

1. Since it is BSR mode, D_7 = '0'.
2. Since D_4, D_5, D_6 are not used, assume them to be '0'.
3. PC_5 has to be selected, hence, D_3 = '1', D_2 = '0', D_1 = '1'.
4. PC_5 has to be set, hence, D0 = '1'.

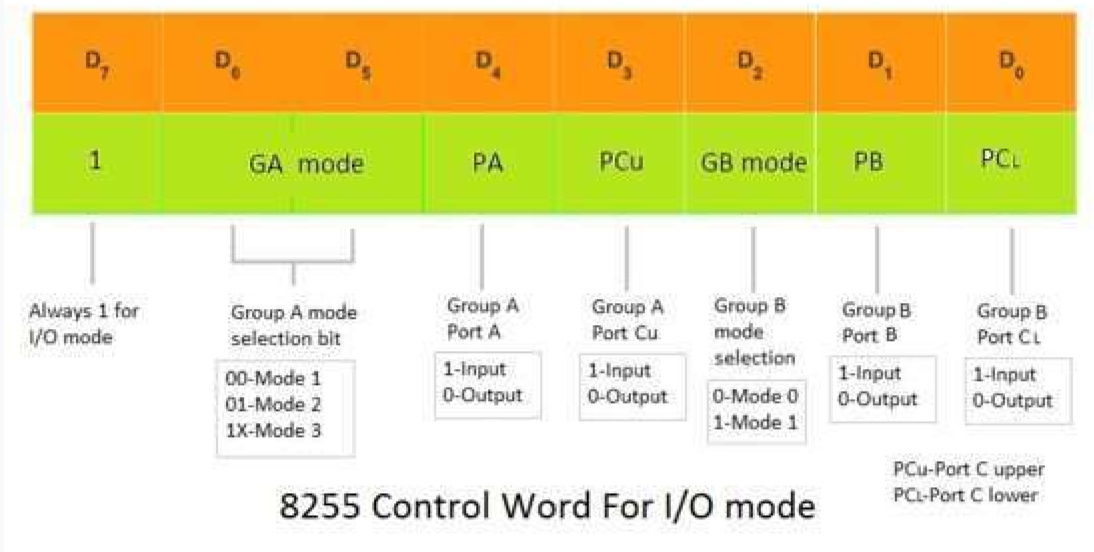
Thus, as per the above values, 0B (Hex) will be loaded into the Control Word Register (CWR).

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	1	0	1	1

Input/Output mode

This mode is selected when D_7 bit of the Control Word Register is 1. There are three I/O modes:[9]

1. Mode 0 - Simple I/O
2. Mode 1 - Strobed I/O
3. Mode 2 - Strobed Bi-directional I/O



I/O Control Word Format

Control Word format

- D_0, D_1, D_3, D_4 are assigned for port C lower, port B, port C upper and port A respectively. When these bits are 1, the corresponding port acts as an input port. For e.g., if $D_0 = D_4 = 1$, then lower port C and port A act as input ports. If these bits are 0, then the corresponding port acts as an output port. For e.g., if $D_1 = D_3 = 0$, then port B and upper port C act as output ports.
- D_2 is used for mode selection of Group B (port B and lower port C). When $D_2 = 0$, mode 0 is selected and when $D_2 = 1$, mode 1 is selected.
- D_5 & D_6 are used for mode selection of Group A (port A and upper port C). The selection is done as follows:

D_6	D_5	Mode
0	0	0
0	1	1

1	X	2

- As it is I/O mode, $D_7 = 1$.

For example, if port B and upper port C have to be initialized as input ports and lower port C and port A as output ports (all in mode 0):

1. Since it is an I/O mode, $D_7 = 1$.
2. Mode selection bits, D2, D5, D6 are all 0 for mode 0 operation.
3. Port B and upper port C should operate as Input ports, hence, $D_1 = D_3 = 1$.
4. Port A and lower port C should operate as Output ports, hence, $D_4 = D_0 = 0$.

Hence, for the desired operation, the control word register will have to be loaded with "10001010" = 8A (hex).

Mode 0 - simple I/O

In this mode, the ports can be used for simple I/O operations without handshaking signals. Port A, port B provide simple I/O operation. The two halves of port C can be either used together as an additional 8-bit port, or they can be used as individual 4-bit ports. Since the two halves of port C are independent, they may be used such that one-half is initialized as an input port while the other half is initialized as an output port.

The input/output features in mode 0 are as follows:

1. Output ports are latched.
2. Input ports are buffered, not latched.
3. Ports do not have handshake or interrupt capability.
4. With 4 ports, 16 different combinations of I/O are possible.

'Latched' means the bits are put into a storage register (array of flip-flops) which holds its output constant even if the inputs change after being latched.

The 8255's outputs are latched to hold the last data written to them. This is required because the data only stays on the bus for one cycle. So, without latching, the outputs would become invalid as soon as the write cycle finishes.
The inputs are not latched because the CPU only has to read their current values, then store the data in a CPU register or memory if it needs to be referenced at a later time. If an input changes while the port is being read then the result may be indeterminate.

Mode 0 – input mode

In the input mode, the 8255 gets data from the external peripheral ports and the CPU reads the received data via its data bus.

- The CPU first selects the 8255 chip by making CS low. Then it selects the desired port using A_0 and A_1 lines.

- The CPU then issues an RD signal to read the data from the external peripheral device via the system data bus.

Mode 0 - output mode

In the output mode, the CPU sends data to 8255 via system data bus and then the external peripheral ports receive this data via 8255 port.

- CPU first selects the 8255 chip by making CS low. It then selects the desired port using A_0 and A_1 lines.
- CPU then issues a WR signal to write data to the selected port via the system data bus. This data is then received by the external peripheral device connected to the selected port.

Mode 1 - Strobed Input/output mode

When we wish to use port A or port B for handshake (strobed) input or output operation, we initialise that port in mode 1 (port A and port B can be initilalised to operate in different modes, i.e., for e.g., port A can operate in mode 0 and port B in mode 1). Some of the pins of port C function as handshake lines.

For port B in this mode (irrespective of whether is acting as an input port or output port), PC0, PC1 and PC2 pins function as handshake lines.

If port A is initialised as mode 1 input port, then, PC3, PC4 and PC5 function as handshake signals. Pins PC6 and PC7 are available for use as input/output lines.

The mode 1 which supports handshaking has following features:

1. Two ports i.e. port A and B can be used as 8-bit i/o ports.
2. Each port uses three lines of port c as handshake signal and remaining two signals can be used as i/o ports.
3. Interrupt logic is supported.
4. Input and Output data are latched. Input Handshaking signals

1. IBF (Input Buffer Full) - It is an output indicating that the input latch contains information.
2. STB (Strobed Input) - The strobe input loads data into the port latch, which holds the information until it is input to the microprocessor via the IN instruction.
3. INTR (Interrupt request) - It is an output that requests an interrupt. The INTR pin becomes a logic 1 when the STB input returns to a logic 1, and is cleared when the data are input from the port by the microprocessor.
4. INTE (Interrupt enable) - It is neither an input nor an output; it is an internal bit programmed via the port PC4(port A) or PC2(port B) bit position.

Output Handshaking signals

1. OBF (Output Buffer Full) - It is an output that goes low whenever data are output(OUT) to the port A or port B latch. This signal is set to a logic 1 whenever the ACK pulse returns from the external device.

2. ACK (Acknowledge)-It causes the OBF pin to return to a logic 1 level. The ACK signal is a response from an external device, indicating that it has received the data from the 82C55A port.

3. INTR (Interrupt request) - It is a signal that often interrupts the microprocessor when the external device receives the data via the signal. this pin is qualified by the internal INTE(interrupt enable) bit.

4. INTE (Interrupt enable) - It is neither an input nor an output; it is an internal bit programmed to enable or disable the INTR pin. The INTE A bit is programmed using the PC6 bit and INTE B is programmed using the PC2 bit.

Mode 2 - Strobed Bidirectional Input/Output mode

Only port A can be initialized in this mode. Port A can be used for bidirectional handshake data transfer. This means that data can be input or output on the same eight lines (PA0 - PA7). Pins PC3 - PC7 are used as handshake lines for port A. The remaining pins of port C (PC0 - PC2) can be used as input/output lines if group B is initialized in mode 0 or as handshaking for port B if group B is initialized in mode 1. In this mode, the 8255 may be used to extend the system bus to a slave microprocessor or to transfer data bytes to and from a floppy disk controller.
Acknowledgement and handshaking signals are provided to maintain proper data flow and synchronisation between the data transmitter and receiver. Interfacing 8255 with 8085

Description Step
1:

a) Lower order of 8-bit address A0-A7 is separated from AD0-AD7 using address latch/buffer (Ex: IC 74373) and ALE signal.

b) The separated address lines A0-A7 are connected to A0-A7 input pins of 8255 and the separated data bus D0-D7 are connected to D0-D7 pins of 8255.

c) Reset out of 8085 is connected to reset pin of 8255.

Step 2:□ 8255 does not have internal (separate) control logic generator, hence the IO/M(bar), RD(bar) and WR(bar) control signals are not connected directly to 8255. These pins are 1st given to decoder and decoded using 3:8 decoder (Ex: IC 74138).

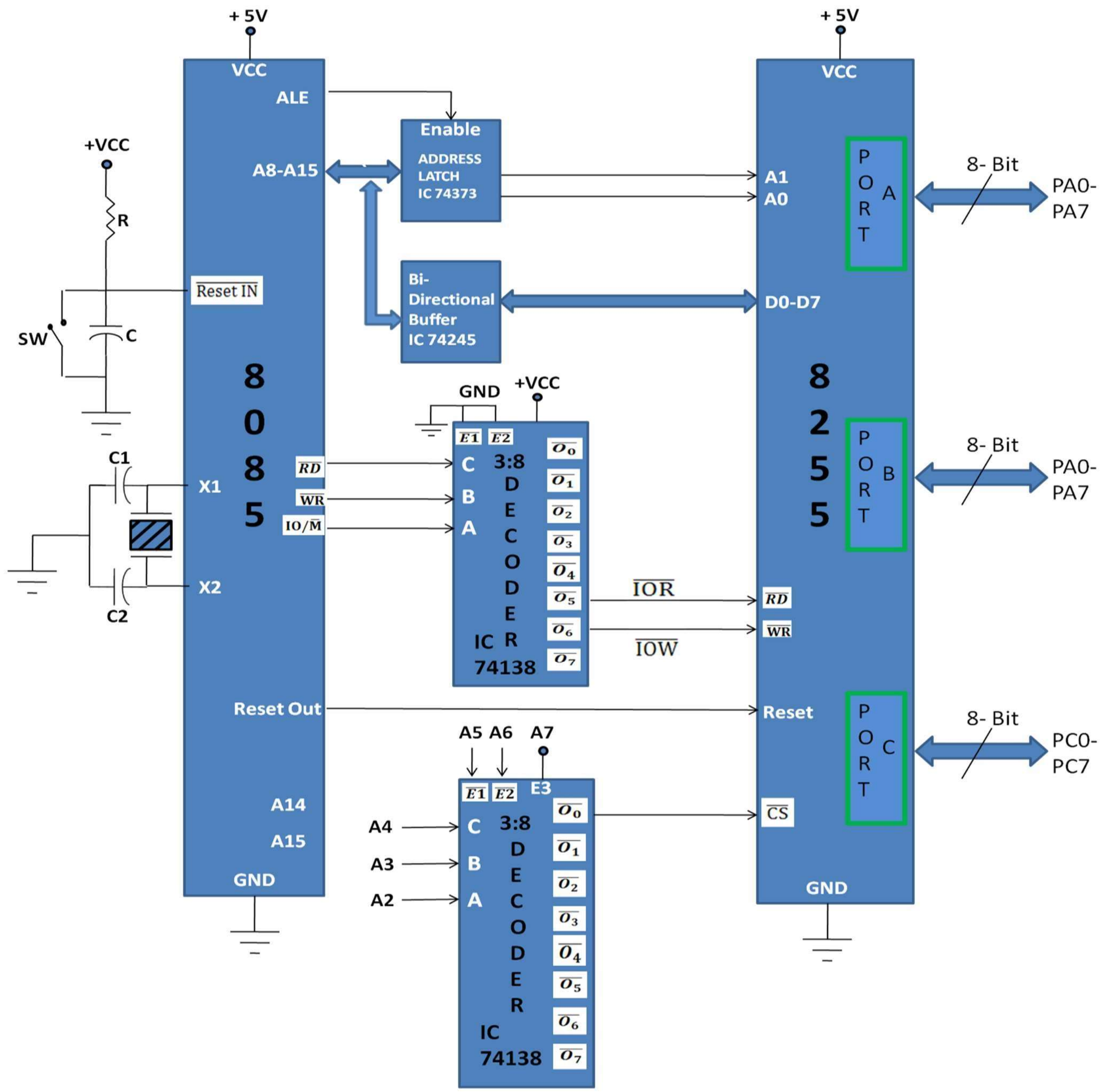
□ The generated control signals IOR(bar) and IOW(bar) are connected to RD(bar) and WR(bar) input of 8155.

Step 3:□ An active low signal of chip select logic is obtained decoding remaining address lines of lower order addresses A2- A7.

- Chip select logic and IO port address for this interfacing circuit are as :

Chip select address lines	Address lines to select port	HEX address	Selected I/O						
A7	A6	A5	A4	A3	A2	A1	A0		
1	0	0	0	0	0	0	0	80H	PORT A
1	0	0	0	0	0	0	1	81H	PORT B
1	0	0	0	0	0	1	0	82H	PORT C
1	0	0	0	0	0	1	1	83H	Chip select register

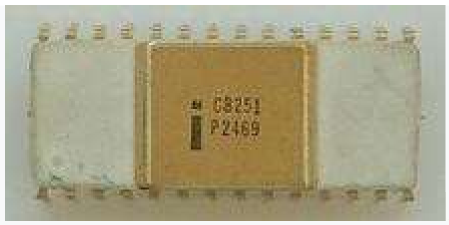
Interfacing Diagram



8251A programmable communication interface

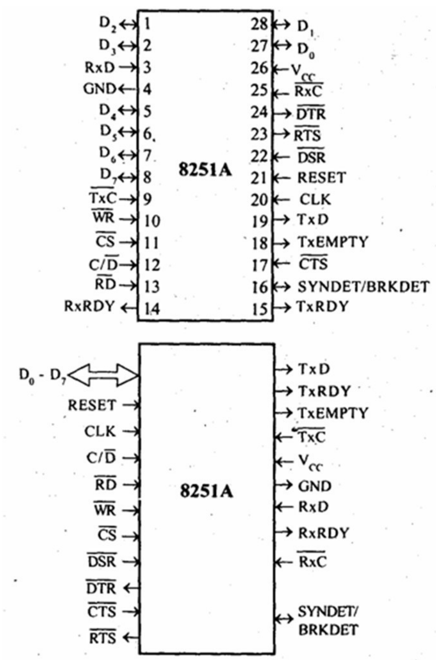
The 8251A is the industry standard Universal Synchronous/Asynchronous Receiver/Transmitter (USART) designed for data communications with microprocessor families such as MCS-48, 80, 85, and iAPX 86, 88. The 8251A is used as a peripheral device and is programmed by the CPU to operate using virtually any serial data transmission technique presently in use (including IBM "bisync"). The USART accepts data characters from the CPU in parallel format and then converts them into a continuous serial data stream for transmission. Simultaneously, it can receive serial data streams and convert them into parallel data characters for the CPU. The USART will signal the CPU whenever it can accept a new character for transmission or whenever it has received a character for the CPU. The CPU can read the complete status of the USART at any time. These include data transmission errors and control signals such as SYNDET, TxEMPTY. The chip is fabricated using Intel's high performance HMOS technology.

Intel 8251

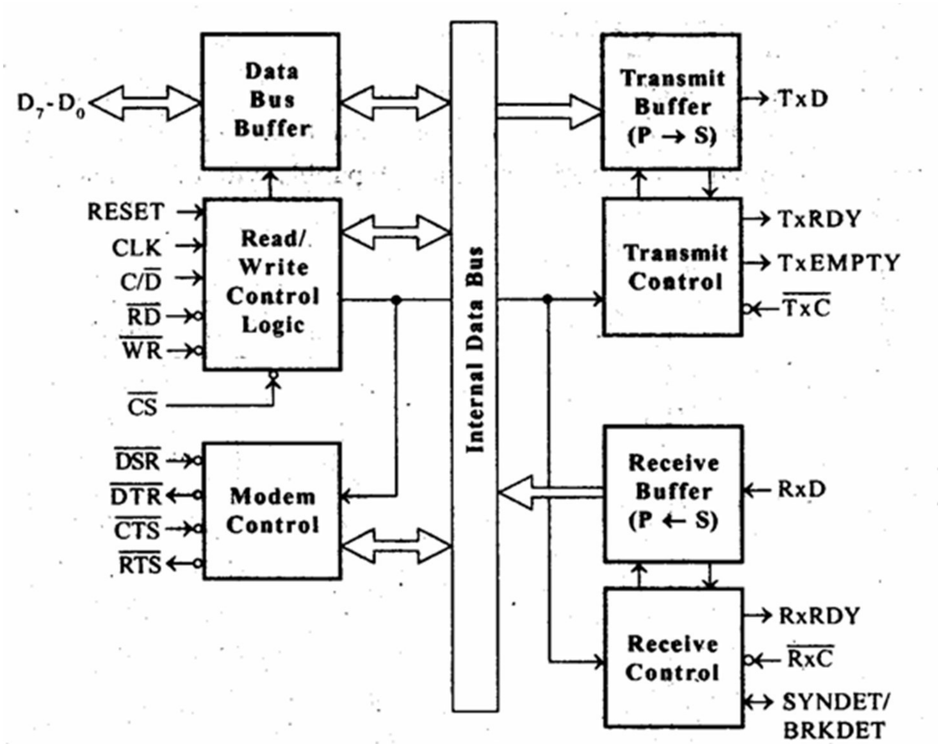


Intel C8251

8251-Programmable Communication Interface (8251-



Pin	Description
D_0-D_7	Parallel data
$C/\overline{D}$	Control register or Data buffer select
$\overline{RD}$	Read control
$\overline{WR}$	Write control
$\overline{CS}$	Chip Select
CLK	Clock pulse (TTL)
RESET	Reset
$\overline{TxC}$	Transmitter Clock
TxD	Transmitter Data
$\overline{RxC}$	Receiver Clock
RxD	Receiver Data
RxRDY	Receiver Ready
TxRDY	Transmitter Ready
$\overline{DSR}$	Data Set Ready
$\overline{DTR}$	Data Terminal Ready
SYNDET/ BRKDET	Synchronous Detect / Break Detect
$\overline{RTS}$	Request To Send Data
$\overline{CTS}$	Clear To Send Data
TxEMPTY	Transmitter Empty
V_{cc}	Supply (+5V)
GND	Ground (0 V)



It includes 5 sections

Read/write control logic: The control logic interfaces the chip with the processor, determines the functions of the chip according to the control word in its register, and monitors the data flow Transmitter:

The transmitter section converts a parallel word received from the processor into serial bits and transmits them over the TxD line to a peripheral Receiver:

The transmitter section receives serial bits from a peripheral, converts them into a parallel word, and transfers the word to the µP data bus system:

This bidirectional register can be addressed as an input and an output port when the pin is low.

Control/Data pin (C/D') : When this signal is high, the control register or the status register is addressed; when it is low, the data buffer is addressed. The control register and the status register are differentiated by RD' and WR' and signals respectively.

modem control:

The modem control is used to establish data communication through modems over telephone lines.

Features :

- The 8251 is a programmable chip designed for synchronous and asynchronous serial data communication
- The INTEL 8251 is the industry standard Universal Synchronous/Asynchronous Receiver/Transmitter (USART) designed for data communications
- The 8251 is used as a peripheral device and is programmed by the CPU to operate using virtually any serial data transmission technique
- The USART accepts data characters from the CPU in parallel format and then converts them into a continuous serial data stream for transmission
- Simultaneously, it can receive serial data streams and convert them into parallel data character for the CPU

Control Word Register

This 16-bit register for a control word consists of two independent bytes; the first byte is called Mode Instruction (Word) and the second byte is called the Command Instruction (Word).

This register can be accessed as an output port when the C/D' pin is high

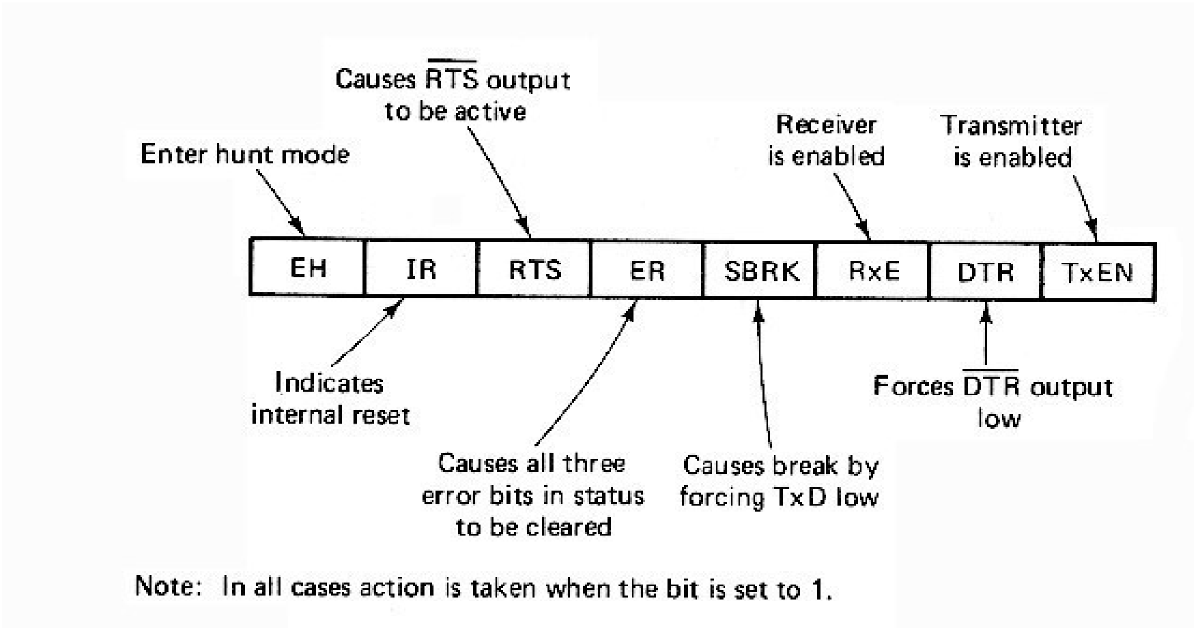
Status Register

This input register checks the ready status of a peripheral. This register is addressed as an input port when the pin C/D'is high. It has the same port address as the control register

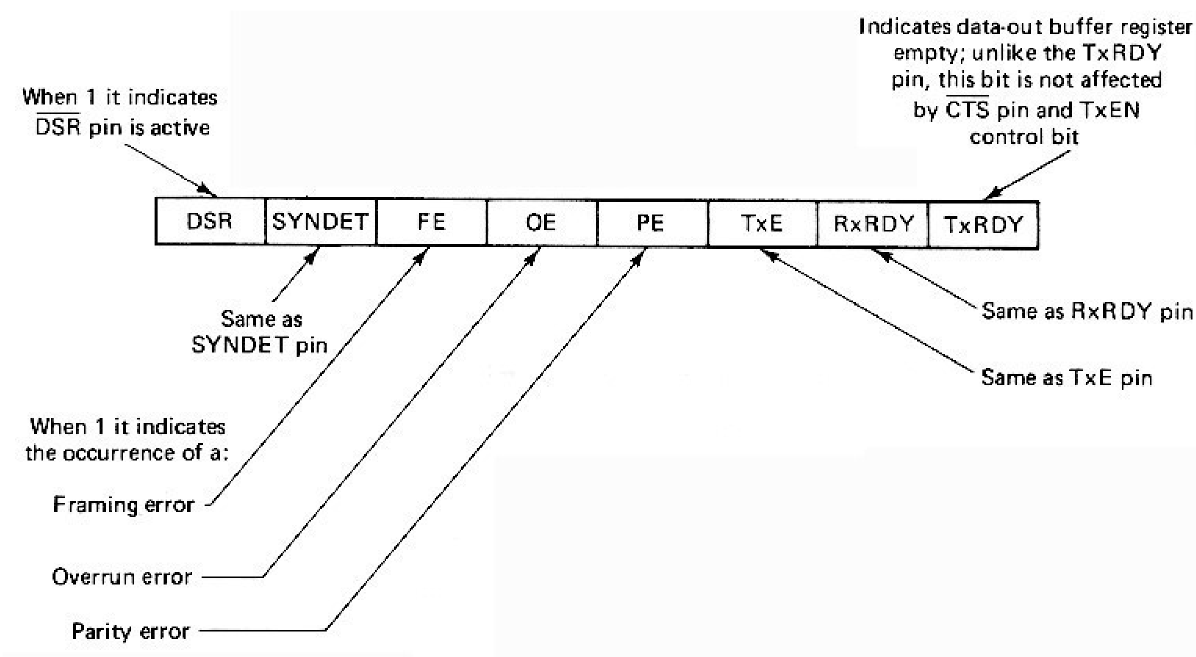
Initialization of 8251

To implement serial communication, 8085 must inform 8251 of all the details, such as mode, baud, stop bits, parity etc. Therefore prior to data transfer, a set of control words must be loaded into 16bit control register of the 8251. In addition, 8085 must check the readiness of a peripheral by reading the status register. The control words are divided into two formats: mode word and command word.

Control Word



Status word



Example 1

- A program sequence which initializes the mode register and gives a command to enable the transmitter and begin an asynchronous transmission of 7-bit characters followed by an even-parity bit and 2 stop bits is:

  MOV AL,11111010B OUT 51H,AL MOV AL,00110011B
  OUT 51H,AL

Example 2

- A program sequence which initializes the mode register and gives a command to enable the transmitter and begin an asynchronous transmission of 7-bit characters followed by an even-parity bit and 2 stop bits is:

MOV AL,11111010B OUT 51H,AL MOV AL,00110011B OUT 51H,AL

Intel 8253 - Programmable Interval Timer

The Intel 8253 and 8254 are Programmable Interval Timers (PTIs) designed for microprocessors to perform timing and counting functions using three 16-bit registers. Each counter has 2 input pins, i.e. Clock & Gate, and 1 pin for "OUT" output. To operate a counter, a 16-bit count is loaded in its register. On command, it begins to decrement the count until it reaches 0, then it generates a pulse that can be used to interrupt the CPU. Features of 8253

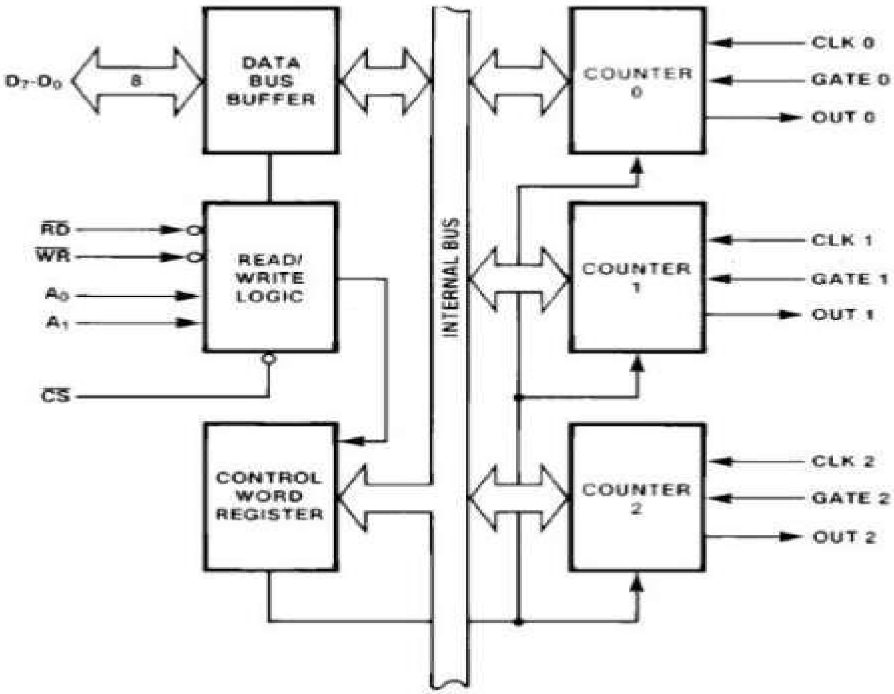
The most prominent features of 8253/54 are as follows −

It has three independent 16-bit down counters.

- It can handle inputs from DC to 10 MHz.
- These three counters can be programmed for either binary or BCD count.
- It is compatible with almost all microprocessors.
- 8254 has a powerful command called READ BACK command, which allows the user to check the count value, the programmed mode, the current mode, and the current status of the counter.

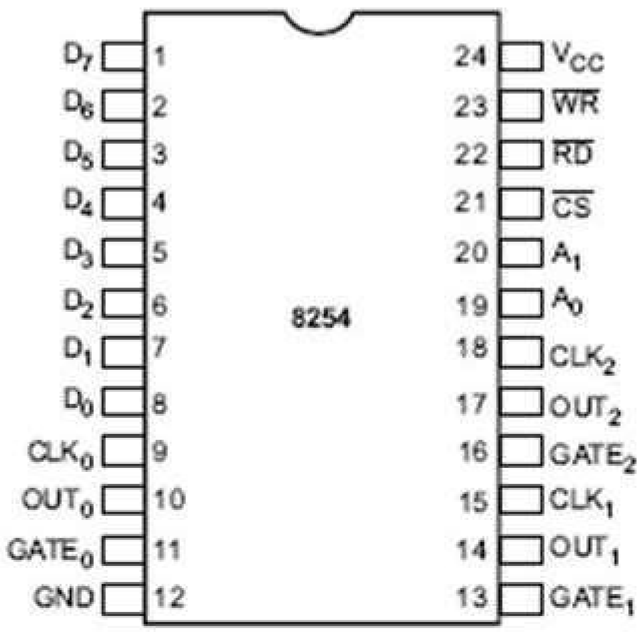
8253 Architecture

The architecture of 8254 looks as follows −



8253 Pin Description

Here is the pin diagram of 8254 −



In the above figure, there are three counters, a data bus buffer, Read/Write control logic, and a control register. Each counter has two input signals - CLOCK & GATE, and one output signal - OUT.

Data Bus Buffer

It is a tri-state, bi-directional, 8-bit buffer, which is used to interface the 8253/54 to the system data bus. It has three basic functions −

- Programming the modes of 8253/54.
- Loading the count registers.
- Reading the count values.

Read/Write Logic

It includes 5 signals, i.e. RD, WR, CS, and the address lines A_0 & A_1. In the peripheral I/O mode, the RD and WR signals are connected to IOR and IOW, respectively. In the memory mapped I/O mode, these are connected to MEMR and MEMW.

Address lines A_0 & A_1 of the CPU are connected to lines A_0 and A_1 of the 8253/54, and CS is tied to a decoded address. The control word register and counters are selected according to the signals on lines A_0 & A_1.

A_1	A_0	Result
0	0	Counter 0
0	1	Counter 1
1	0	Counter 2
1	1	Control Word Register
X	X	No Selection

A_1	A_0	RD	WR	CS	Result
0	0	1	0	0	Write Counter 0
0	1	1	0	0	Write Counter 1
1	0	1	0	0	Write Counter 2
1	1	1	0	0	Write Control Word
0	0	0	1	0	Read Counter 0
0	1	0	1	0	Read Counter 1
1	0	0	1	0	Read Counter 2
1	1	0	1	0	No operation
X	X	1	1	0	No operation
X	X	X	X	1	No operation

Control Word Register

This register is accessed when lines A_0 & A_1 are at logic 1. It is used to write a command word, which specifies the counter to be used, its mode, and either a read or write operation. Following table shows the result for various control inputs.

Counters

Each counter consists of a single, 16 bit-down counter, which can be operated in either binary or BCD. Its input and output is configured by the selection of modes stored in the control word register. The programmer can read the contents of any of the three counters without disturbing the actual count in process.

8253/54 can be operated in 6 different modes. In this chapter, we will discuss these operational modes.

Mode 0 ─ Interrupt on Terminal Count

- It is used to generate an interrupt to the microprocessor after a certain interval.
- Initially the output is low after the mode is set. The output remains LOW after the count value is loaded into the counter.
- The process of decrementing the counter continues till the terminal count is reached, i.e., the count become zero and the output goes HIGH and will remain high until it reloads a new count.
- The GATE signal is high for normal counting. When GATE goes low, counting is terminated and the current count is latched till the GATE goes high again.

Mode 1 – Programmable One Shot

- It can be used as a mono stable multi-vibrator.
- The gate input is used as a trigger input in this mode.
- The output remains high until the count is loaded and a trigger is applied.

Mode 2 – Rate Generator

- The output is normally high after initialization.
- Whenever the count becomes zero, another low pulse is generated at the output and the counter will be reloaded.

Mode 3 – Square Wave Generator

- This mode is similar to Mode 2 except the output remains low for half of the timer period and high for the other half of the period.

Mode 4 − Software Triggered Mode

- In this mode, the output will remain high until the timer has counted to zero, at which point the output will pulse low and then go high again.
- The count is latched when the GATE signal goes LOW.
- On the terminal count, the output goes low for one clock cycle then goes HIGH. This low pulse can be used as a strobe.

Mode 5 – Hardware Triggered Mode

- This mode generates a strobe in response to an externally generated signal.
- This mode is similar to mode 4 except that the counting is initiated by a signal at the gate input, which means it is hardware triggered instead of software triggered.
- After it is initialized, the output goes high.
- When the terminal count is reached, the output goes low for one clock cycle.

Module 8

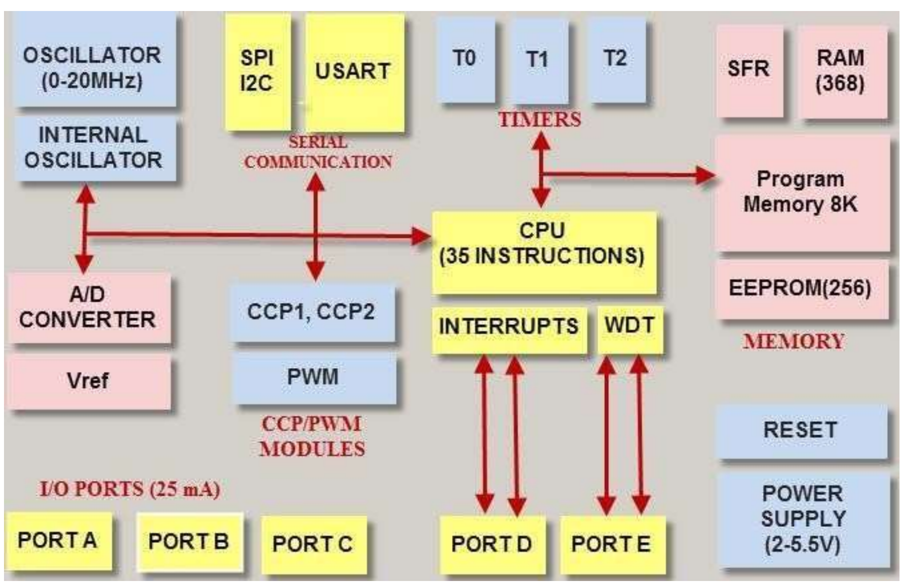
PIC (Programmable Interface Controllers)

PIC (Programmable Interface Controllers) microcontrollers are the worlds smallest microcontrollers that can be programmed to carry out a huge range of tasks. These microcontrollers are found in many electronic devices such as phones, computer control systems, alarm systems, embedded systems, etc. Various types of microcontrollers exist, even though the best are found in the GENIE range of programmable microcontrollers. These microcontrollers are programmed and simulated by a circuit-wizard software.

Every PIC microcontroller architecture consists of some registers and stack where registers function as Random Access Memory(RAM) and stack saves the return addresses. The main features of PIC microcontrollers are RAM, flash memory, Timers/Counters, EEPROM, I/O Ports, USART, CCP (Capture/Compare/PWM module), SSP, Comparator, ADC (analog to digital converter), PSP(parallel slave port), LCD and ICSP (in circuit serial programming) The 8bit PIC microcontroller is classified into four types on the basis of internal architecture such as Base Line PIC, Mid Range PIC, Enhanced Mid Range PIC and PIC18

Architecture of PIC Microcontroller

The PIC microcontroller architecture comprises of CPU, I/O ports, memory organization, A/D converter, timers/counters, interrupts, serial communication, oscillator and CCP module which are discussed in detailed below.



Architecture of PIC Microcontroller

CPU (Central Processing Unit)

It is not different from other microcontrollers CPU and the PIC microcontroller CPU consists of the ALU, CU, MU and accumulator, etc. Arithmetic logic unit is mainly used for arithmetic operations and to take logical decisions. Memory is used for storing the instructions after processing. To control the internal and external peripherals, control unit is used which are connected to the CPU and the accumulator is used for storing the results and further process.

Memory Organization

The memory module in the PIC microcontroller architecture consists of RAM (Random Access Memory), ROM (Read Only Memory) and STACK.

Random Access Memory (RAM)

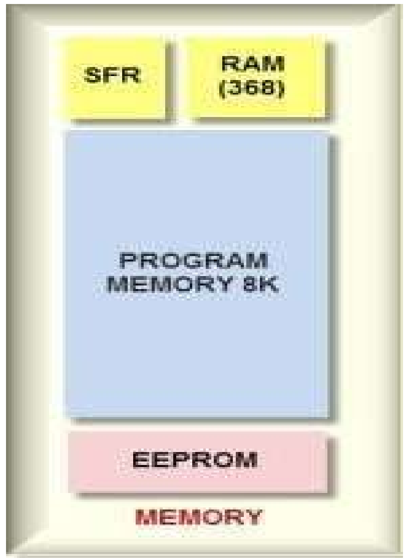
RAM is an unstable memory which is used to store the data temporarily in its registers. The RAM memory is classified into two banks, and each bank consists of so many registers. The RAM registers are classified into two types: Special Function Registers (SFR) and General Purpose Registers (GPR).

- General Purpose Registers (GPR)

These registers are used for general purpose only as the name implies. For example, if we want to multiply two numbers by using the PIC microcontroller. Generally, we use registers for multiplying and storing the numbers in other registers. So these registers don't have any special function,- CPU can easily access the data in the registers.

- Special Function Registers

These registers are used for special purposes only as the name SFR implies. These registers will perform according to the functions assigned to them , and they cannot be used as normal registers. For example, if you cannot use the STATUS register for storing the data, these registers are used for showing the operation or status of the program. So, user cannot change the function of the SFR; the function is given by the retailer at the time of manufacturing.



Memory Organization

Read Only Memory (ROM)

Read only memory is a stable memory which is used to store the data permanently. In PIC microcontrollerarchitecture, the architecture ROM stores the instructions or program, according to the program the microcontroller acts. The ROM is also called as program memory, wherein the user will write the program for microcontroller and saves it permanently, and finally the program is executed by the CPU. The microcontrollers performance depends on the instruction, which is executed by the CPU.

Electrically Erasable Programmable Read Only Memory (EEPROM)

In the normal ROM, we can write the program for only once we cannot use again the microcontroller for multiple times. But, in the EEPROM, we can program the ROM multiple times.

Flash Memory

Flash memory is also programmable read only memory (PROM) in which we can read, write and erase the program thousands of times. Generally, the PIC microcontroller uses this type of ROM.

Stack

When an interrupt occurs, first the PIC microcontroller has to execute the interrupt and the existing process address. Then that is being executed is stored in the stack. After completing the execution of the interrupt, the microcontroller calls the process with the help of address, which is stored in the stack and get executes the process.

I/O Ports

- The series of PIC16 consists of five ports such as Port A, Port B, Port C, Port D & Port E.

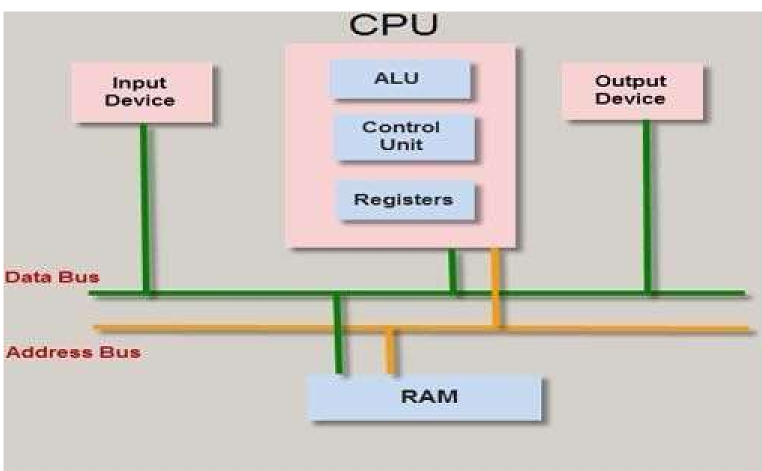
- Port A is an 16-bit port that can be used as input or output port based on the status of the TRISA (Tradoc Intelligence Support Activity) register.
- Port B is an 8- bit port that can be used as both input and output port.
- Port C is an 8-bit and the input of output operation is decided by the status of the TRISC register.
- Port D is an 8-bit port acts as a slave port for connection to the microprocessor BUS.
- Port E is a 3-bit port which serves the additional function of the control signals to the analog to digital converter.

BUS

BUS is used to transfer and receive the data from one peripheral to another. It is classified into two types such as data bus and address.

Data Bus: It is used for only transfer or receive the data.

Address Bus: Address bus is used to transmit the memory address from the peripherals to the CPU. I/O pins are used to interface the external peripherals; UART and USART both are serial communication protocols which are used for interfacing serial devices like GSM, GPS, Bluetooth, IR , etc.



PIC Microcontroller Applications

The PIC microcontroller projects can be used in different applications, such as peripherals, audio accessories, video games, etc. For better understanding of this PIC microcontroller, the following project demonstrates PIC microcontroller’s operations.

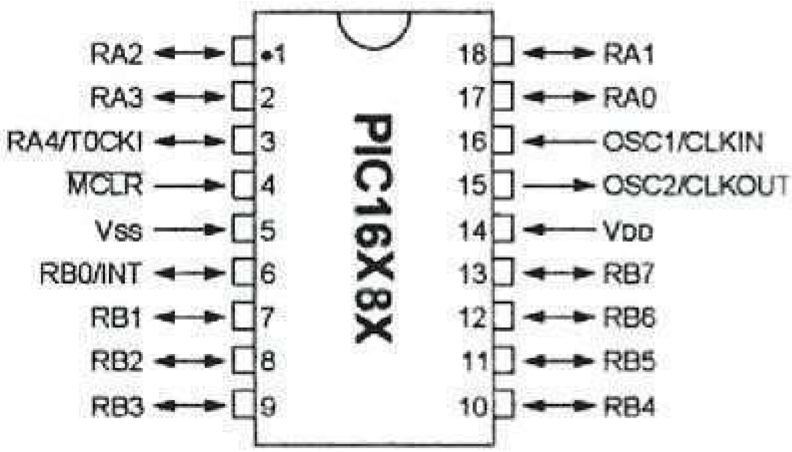
Advantages of PIC Microcontroller:

- PIC microcontrollers are consistent and faulty of PIC percentage is very less. The performance of the PIC microcontroller is very fast because of using RISC architecture.

- When comparing to other microcontrollers, power consumption is very less and programming is also very easy.
- Interfacing of an analog device is easy without any extra circuitry Disadvantages of PIC Microcontroller:
- The length of the program is high due to using RISC architecture (35 instructions)
- One single accumulator is present and program memory is not accessible

The 16F84 Pins

Below is a diagram showing the pin-outs of the PIC 16F84. We will go through each pin, explaining what each is used for.



RA0 To RA4

RA is a bidirectional port. That is, it can be configured as an input or an output. The number following RA is the bit number (0 to 4). So, we have one 5-bit directional port where each bit can be configured as Input or Output.

RB0 To RB7

RB is a second bidirectional port. It behaves in exactly the same way as RA, except there are 8 - bits involved.

VSS And VDD

These are the power supply pins. VDD is the positive supply, and VSS is the negative supply, or 0V. The maximum supply voltage that you can use is 6V, and the minimum is 2V.

OSC1/CLK IN And OSC2/CLKOUT

These pins is where we connect an external clock, so that the microcontroller has some kind of timing.

MCLR

This pin is used to erase the memory locations inside the PIC (i.e. when we want to re-program it. In normal use it is connected to the positive supply rail.

INT

This is an input pin which can be monitored. If the pin goes high, we can cause the program to restart, stop or any other single function we desire. We won't be using this one much.

T0CK1

This is another clock input, which operates an internal timer. It operates in isolation to the main clock. Again, we won't be using this one much either.

MULTIPLE CHOICE QUESTIONS

1. Which interrupt has the highest priority?

a) INTR b) TRAP c) RST6.5

2. In 8085 name the 16 bit registers?

a) Stack pointer b) Program counter c) a & b 3.

Which of the following is hardware interrupts?

a) RST5.5, RST6.5, RST7.5 b) INTR, TRAP c) a & b

4. What is the RST for the TRAP?

a) RST5.5 b) RST4.5 c) RST4

5. What are level Triggering interrupts?

a) INTR&TRAP b)RST6.5&RST5.5 c)RST7.5&RST6.5

6. Which interrupt is not level sensitive in 8085?

a) RST6.5 is a raising edge-trigging interrupt.

b) RST7.5 is a raising edge-trigging interrupt.

c) a & b.

7. What are software interrupts?

a) RST 0 - 7 b) RST 5.5 - 7.5 c) INTR, TRAP

8. Which stack is used in 8085?

a) FIFO b) LIFO c) FILO

9. Why 8085 processor is called an 8 bit processor?

a) Because 8085 processor has 8 bit ALU.

b) Because 8085 processor has 8 bit data bus.

c) a & b.

10. What is SIM?

a) Select Interrupt Mask b) Sorting Interrupt Mask c) Set Interrupt Mask. 11. RIM

is used to check whether, ______

a) The write operation is done or not

b) The interrupt is Masked or not

c) a & b

12. What is meant by Maskable interrupts?

a) An interrupt which can never be turned off.

b) An interrupt that can be turned off by the programmer.

c) none

13. In 8086, Example for Non maskable interrupts are

a) Trapb) RST6.5 c) INTR

14. What does microprocessor speed depends on?

a) Clock b) Data bus width c) Address bus width

15. Can ROM be used as stack?

a) Yes b) No c) sometimes yes, sometimes no

16. Which processor structure is pipelined?

a) all x80 processors b) all x85 processors c) all x86 processors

17. Address line for RST3 is?

a) 0020H b) 0028H c) 0018H

18. In 8086 the overflow flag is set when

a) The sum is more than 16 bits

b) Signed numbers go out of their range after an arithmetic operation

c) Carry and sign flags are set Key:

1 C ,2 C, 3 C ,4 B ,5 B, 6 B,7 A ,8 B,9 A ,10 C ,11 B,12 B,13 A ,14 C ,15 B,16 C,17 C ,18 B

19. Programmable peripheral input-output port is other name for

a) serial input-output port
b) parallel input-output port
c) serial input port
d) parallel output port

Ans:b

20. Port C of 8255 can function independently as
    a) input port
    b) output port
    c) either input or output ports
    d) both input and output ports

Ans : c

21. All the functions of the ports of 8255 are achieved by programming the bits of an internal register called
    a) data bus control
    b) read logic control
    c) control word register
    d) none of the mentioned

Ans:c

22. The data bus buffer is controlled by
    a) control word register
    b) read/write control logic
    c) data bus
    d) none of the mentioned

Ans:b

23. The register that may be used as an operand register is
    a) Accumulator
    b) B register
    c) Data register
    d) Accumulator and B register

Ans:d

24. The register that can be used as a scratch pad is
    a) Accumulator
    b) B register
    c) Data register

d) Accumulator and B register

Ans:b

25. The registers that contains the status information is
    a) control registers
    b) instruction registers
    c) program status word
    d) all of the mentioned

Ans:c

26. The architecture of 8051 consists of
    a) 4 latches
    b) 2 timer registers
    c) 4 on-chip I/O ports
    d) all of the mentioned

Ans:d

27. The register that provides control and status information about counters is a) IP
    b) TMOD
    c) TSCON
    d) PCON

Ans b

28. Which of the following is an 8-bit register?
    a) PSW(Program Status Word)
    b) TCON(Timer Control Register)
    c) Accumulator
    d) All of the mentioned

Ans:d

29. The higher and lower bytes of a 16-bit register DPTR are represented respectively as
    a) LDPTR and HDPTR
    b) DPTRL and DPTRH
    c) DPH and DPL
    d) HDP and LDP

Ans:c

30. Among the four groups of register banks, the number of groups that can be accessed at a time is a) 1
  b) 2
  c) 3
  d) all the four

Ans: a

31. If RS1=1, RS0=0, then the register bank selected is
  a) register bank 0
  b) register bank 1
  c) register bank 2
  d) register bank 3

Ans c

32. The idle mode can be terminated by
  a) PRESET
  b) CLEAR
  c) Interrupt
  d) Interrupt or reset

And:d

MODEL QUESTION

1.a) What is the use of ALE pin?

Ans:-The AD7– AD0 lines of 8085 are serving a dual purposeand that they need to be demultiplexed to get all the information.The high order bits of the address remain on the bus for three clock periods. However, the low order bits remain for only one clock periodand they would be lost if they are not saved externally. Also, notice that the low order bits of the address disappear when they are needed most. To make sure we have the entire address for the full three clock cycles, we will use an external latchto save the value of AD7– AD0 when it is carrying the address bits. We use the ALE signal to enable this latch.

1.b) Mention the purpose of SID and SOD lines.

Ans: SID (Serial input data line): It is an input line through which the microprocessor accepts serial data. SOD (Serial output data line): It is an output line through which the microprocessor sends output serial data.

1.c) Explain the difference between a JMP instruction and CALL instruction.

Ans: A JMP instruction permanently changes the program counter. A CALL instruction leaves information on the stack so that the original program execution sequence can be resumed.

1.d) How many address lines in a 4096 x 8 EPROM CHIP?

Ans: 12 address lines.

1.e) What is meant by Wait State?

Ans:-This state is used by slow peripheral devices. The peripheral devices can transfer the data to or from the microprocessor by using READY input line. The microprocessor remains in wait state as long as READY line is low. During the wait state, the contents of the address, address/data and control buses are held constant.

1.f) Write steps involved to fetch a byte in 8085

Ans:-

i. The PC places the 16-bit memory address on the address bus.

ii. The control unit sends the control signal RD to enable the memory chip. iii. The byte from the memory location is placed on the data bus.

iv. The byte is placed in the instruction decoder of the microprocessor and the task is carried out according to the instruction.

1.g) Define instruction cycle, machine cycle and T-state

Ans:-Instruction cycle is defined, as the time required completing the execution of an instruction. Machine cycle is defined as the time required completing one operation of accessing memory, I/O or acknowledging an external request. Tcycle is defined as one subdivision of the operation performed in one clock period.

1.h) Explain the different instruction formats with examples

Ans:-The instruction set is grouped into the following formats

- One byte instruction -MOV C,A
- Two byte instruction -MVI A,39H
- Three byte instruction -JMP 2345H

1.i) What are the features used mode 1 in 8255?

Ans:-Two groups – group A and group B are available for strobed data transfer.

1. Each group contains one 8-bit data I/O port and one 4-bit control/data port.
2. The 8-bit data port can be either used as input or output port. The inputs and outputs both are latched.
3. Out of 8-bit port C, PC0-PC2 is used to generate control signals for port B and PC3=PC5 are used to generate control signals for port A. The lines PC6, PC7 may be used as independent data lines.

1.j) What is the output modes used in 8279?

Ans: 8279 provides two output modes for selecting the display options:

1.Display Scan

In this mode, 8279 provides 8 or 16 character-multiplexed displays those can be organized as dual 4-bit or single 8-bit display units.

2.Display Entry

8279 allows options for data entry on the displays. The display data is entered for display from the right side or from the left side.

2.a) Explain priority interrupts of 8085.

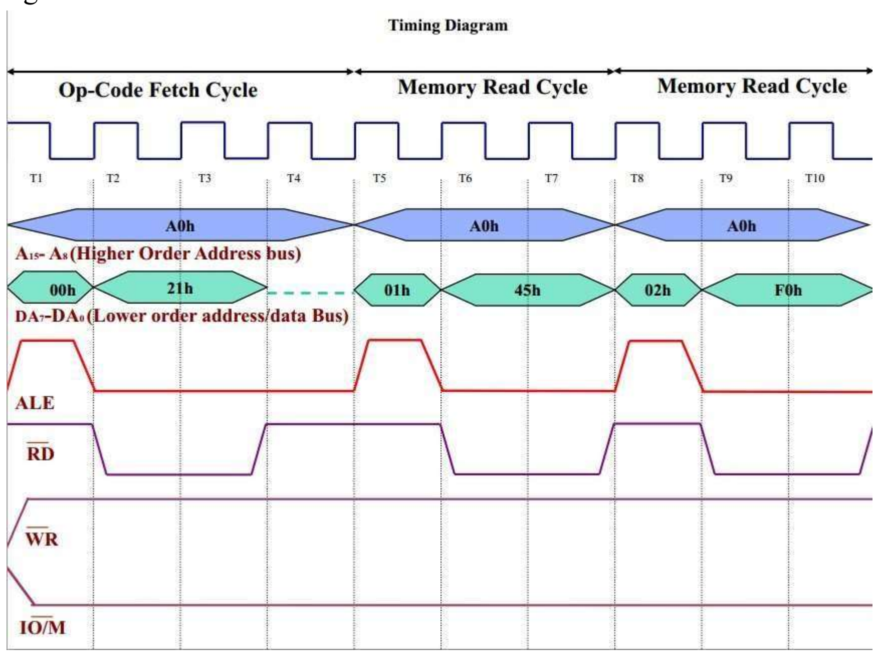
Ans:-The 8085 microprocessor has five interrupt inputs. They are TRAP, RST 7.5, RST 6.5, RST 5.5, and INTR. These interrupts have a fixed priority of interrupt service. If two or more interrupts go high at the same time, the 8085 will service them on priority basis. The TRAP has the highest priority followed by RST 7.5, RST 6.5, RST 5.5. The priority of interrupts in 8085 is shown in the table.

TRAP	1
RST 7.5	2
RST 6.5	3

RST 5.5	4
INTR	5

2.b) Draw timing diagram for LXI A, F045H.

Ans:- Corresponding code is 21 45 F0.



3.a) How information is saved on the stack of 8085?

Ans:-Information is saved on the stack by PUSHing it on. It is retrieved from the stack by POPing it off.

- The 8085 provides two instructions: PUSH and POP for storing information on the stack and retrieving it back. Both PUSH and POP work with register pairs ONLY. The PUSH Instruction

- PUSH B

– Decrement SP
– Copy the contents of register B to the memory location pointed to by SP – Decrement SP
– Copy the contents of register C to the memory location pointed to by SP During pushing, the stack operates in a "decrement then store" style.
– The stack pointer is decremented first, then the information is placed on the stack.

3.b) Explain Vectored and non-vectored interrupts of 8085.

Ans:-An interrupt vectoris a pointer to where the ISR is stored in memory.

• All interrupts (vectored or otherwise) are mapped onto a memory area called the Interrupt Vector Table (IVT).

– The IVT is usually located in memory page 00 (0000H - 00FFH).

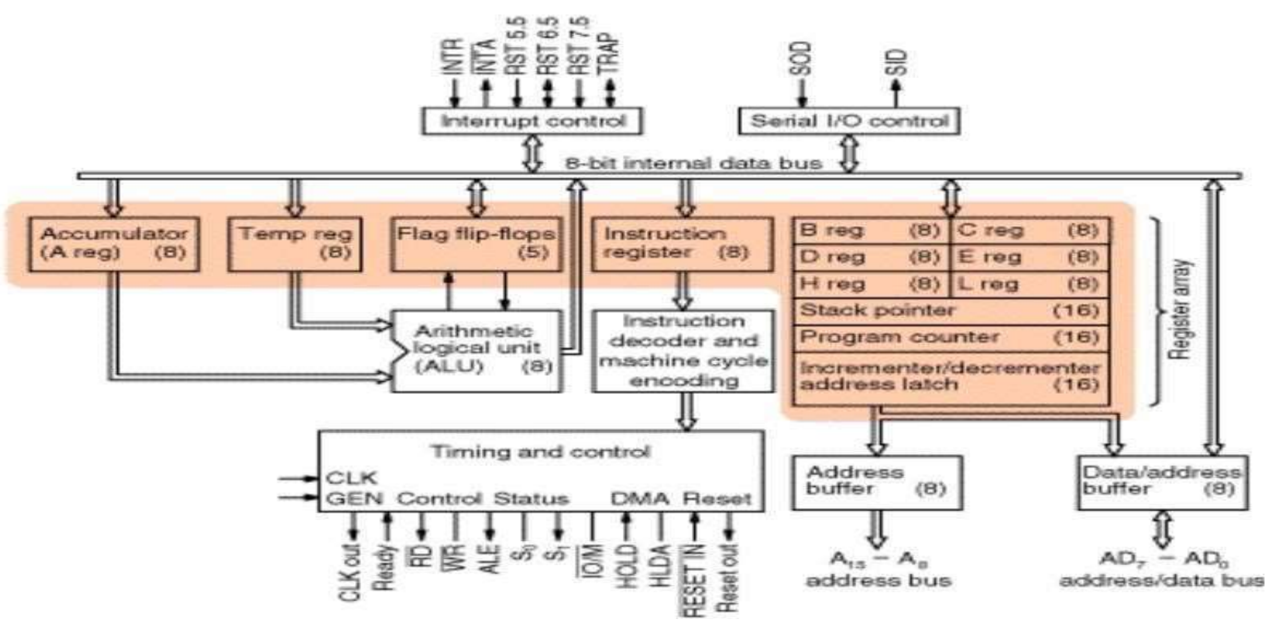
– The purpose of the IVT is to hold the vectors that redirect the microprocessor to the right place when an interrupt arrives.

– The IVT is divided into several blocks. Each block isused by one of the interrupts to hold its "vector. The 8085 Non-Vectored Interrupt Process

1. The interrupt process should be enabled using the EIinstruction.
2. The 8085 checks for an interrupt during the execution of every instruction.
3. If there is an interrupt, the microprocessor will complete the executing instruction, and start a RESTART sequence.
4. The RESTART sequence resets the interrupt flip flop and activates the interrupt acknowledge signal(INTA).
5. Upon receiving the INTA signal, the interrupting device is expected to return the op-code of one of the 8 RST instructions.
6. When the microprocessor executes the RST instruction received from the device, it saves the address of the next instruction on the stack and jumps to the appropriate entry in the IVT.
7. The IVT entry must redirect the microprocessor to the actual service routine.
8. The service routine must include the instruction EIto re-enable the interrupt process.
9. At the end of the service routine, the RETinstruction returns the execution to where the program was interrupted.

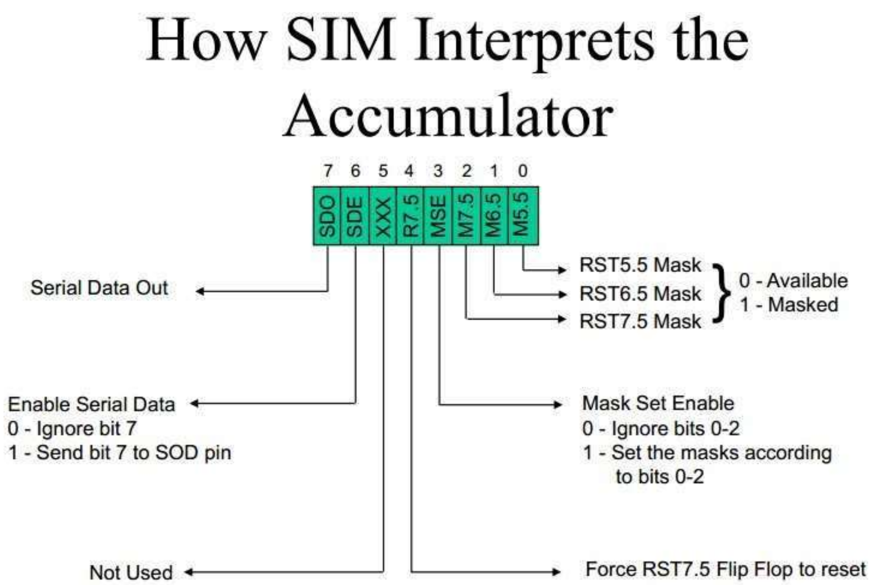
4.a) Draw a neat and clean block diagram of 8085.

Ans:-



Microprocessor consists of:
– Control unit: control microprocessor operations.
– ALU: performs data processing function.
– Registers: provide storage internal to CPU.
– Interrupts
– Internal data bus
The ALU
• In addition to the arithmetic & logic circuits, the ALU includes the accumulator, which is part of every arithmetic & logic operation.
• Also, the ALU includes a temporary register used for holding data temporarily during the execution of the operation. This temporary register is not accessible by the programmer. • Registers
– General Purpose Registers
• B, C, D, E, H & L (8 bit registers)
• Can be used singly
• Or can be used as 16 bit register pairs – BC, DE, HL
• H & L can be used as a data pointer (holds memory address) – Special Purpose Registers
• Accumulator(8 bit register) – Store 8 bit data
– Store the result of an operation
– Store 8 bit data during I/O transfer The Internal Architecture
• We have already discussed the general purpose registers, the Accumulator, and the flags.
• The Program Counter (PC)– This is a register that is used to control the sequencing of the execution of instructions.
– This register always holds the address of the next instruction.
– Since it holds an address, it must be 16 bits wide. The Internal Architecture
• The Stack pointer– The stack pointer is also a 16-bit register that is used to point into memory. –
The memory this register points to is a special area called the stack.
– The stack is an area of memory used to hold data that will be retreived soon. – The stack is usually accessed in a Last In First Out (LIFO) fashion.

4.b) Give format of SIM instruction.



Ans:-

5.a) Distinguish between the memories mapped I/O peripheral I/O?

Ans:

Memory Mapped I/O	Peripheral Mapped I/O
16-bit device address	8-bit device address
Data transfer between any between general-purpose register and I/O port.	Data is transfer only accumulator and I.O port
The memory map (64K) is of shared between I/O device and system memory.	The I/O map is independent the memory map; 256 input device and 256 output device can be connected
More hardware is required to decode 16-bit address	Less hardware is required to decode 8-bit address
Arithmetic or logic operation can be directly performed with I/O data	Arithmetic or logical operation cannot be directly performed with I/O data

5.b) Calculate delay of following routine:

MVI B, 10H LOOP2: MVI C, FFH LOOP1: DCR C JNZ LOOP1 DCR B
JNZ LOOP2

Ans:-

- Delay of inner loop
  - T_{O1} = 7 T-States
    - MVI C, FFH instruction
  - T_{L1} = (255 X 14) - 3 = 3567 T-States
    - 14 T-States for the DCR C and JNZ instructions repeated 255 times (FF = 255) minus 3 for the final JNZ

- Delay of outer loop
  - T_{O2} = 7 T-States
    - MVI B, 10H instruction
  - T_{L1} = (16 X (14 + 3574)) - 3 = 57405 T-States
    - 14 T-States for the DCR B and JNZ instructions and 3574 T-States for loop1 repeated 16 times ($10_{16} = 16_{10}$) minus 3 for the final JNZ.
  - T_{Delay} = 7 + 57405 = 57412 T-States

- Total Delay
  - T_{Delay} = 57412 X 0.5 µSec = 28.706 mSec

6) Explain the pipeline architechture of 8086.

Ans: 8086 architechture with BIU & EU interface

7) 8051 was developed using which technology?

Ans. Intel's original MCS-51 family was developed using NMOS technology, but later versions, identified by a letter C in their name (e.g., 80C51) used CMOS technology and consume less power than their NMOS predecessors. This made them more suitable for battery-powered devices.

8) Why 8051 is called 8 bit microcontroller?

Ans.The Intel 8051 is an 8-bit microcontroller which means that most available operations are limited to 8 bits.

9) What is the width of data bus? Ans.8-bit data bus

10) What is the width of address bus? Ans.16-bit address bus

11) List out the features of 8051 micro controller? Ans

- 40 Pin IC.
- 128 bytes of RAM.
- 4K ROM.
- 2 Timers (Timer 0 and Timer 1).
- 32 Input/ Output pins.
- 1 serial port.
- 6 Interrupts (Including Reset).

12) On-chip RAM is also called _____ memory? Ans.Direct memory

13) What location code memory space and data memory space begins? Ans.At location 0x00 for internal or external memory

14) How Much on chip RAM is available?
Ans.128 bytes of RAM (from 0x00 to 0x7F) and can be used to store data.

15) With 12 MHz clock frequency how many instructions (of 1 machine cycle and 2 machine cycle) it can execute per second?
Hint:A cycle is, in reality, 12 pulses of the crystal. That is to say, if an instruction takes one machine cycle to execute, it will take 12 pulses of the crystal to execute. Since we know the crystal is pulsing 11,059,000 times per second and that one machine cycle is 12 pulses, we can calculate how many instruction cycles the 8051 can execute per second:

11,059,000 / 12 = 921,583
This means that the 8051 can execute 921,583 single-cycle instructions per second. Since a large number of 8051 instructions are single-cycle instructions it is often considered that the 8051 can execute roughly 1 million instructions per second, although in reality it is less–and, depending on the instructions being used, an estimate of about 600,000 instructions per second is more realistic.

16) How much total external data memory that can be interfaced to the 8051? Ans.64K data memory

References:

1. Microprocessor architecture, programming and application with 8085 – R. Gaonkar

2. Microprocessors and microcontrollers - N. Senthil Kumar, M. Saravanan and Jeevananthan

3. Microprocessors & interfacing – D. V. Hall
4. https://www.tutorialspoint.com/microprocessor
5. http://www.nptel.ac.in
6. http://www.expertsmind.com
7. http://www.hobbyprojects.com
8. https://www.elprocus.com/8051-microcontroller-architecture-and-applications
9. https://www.tutorialspoint.com/embedded_systems/es_microcontroller.htm
10. https://www.slideshare.net/yayavaram/8051-microcontroller-notes
11. https://www.edgefx.in/8051-microcontroller-architecture
12. https://www.robomart.com/blog/introduction-to-8051-microcontroller
13. The 8051 Microcontroller and Embedded Systems,Muhammed Ali Mazidi,J.G.Mazidi
14. https://www.intorobotics.com/8051-microcontroller-programming-tutorialssimulatorscompilers-and-programmers
15. http://microcontrollerslab.com/pic-microcontroller-assembly-language
16. https://www.elprocus.com/simple-steps-burn-program-into-microcontroller
17. https://www.elprocus.com/basics-and-structure-of-embedded-c-program-withexamplesfor-beginners