MAILAM ENGINEERINGCOLLEGE
MAILAM, 604304
DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

Subcode/SubName: EE6502 MICROPROCESSORS AND MICROCONTROLLERS **Year/ Sem: III EEE / V**

UNIT-I - 8085 PROCESSOR

Hardware Architecture, pin outs – Functional Building Blocks of Processor – Memory organization – I/O ports and data transfer concepts– Timing Diagram – Interrupts.

Updated Question:

PART-A

Q.No.18	Page No: 4	(Nov/Dec 2017)
Q.No.60	Page No: 10	(Nov/Dec 2017)
Q.No.08	Page No: 02	(Apr/May 2018)
Q.No.61	Page No: 10	(Apr/May 2018)

PART-B

Q.No.02	Page No: 12	(Nov/Dec 2017)
Q.No.08	Page No: 31	(Nov/Dec 2017)
Q.No.02	Page No: 15	(Nov/Dec 2017)
Q.No.04	Page No: 19	(Apr/May 2018)
Q.No.07	Page No: 27	(Apr/May 2018)
Q.No. 09	Page No: 38	(Apr/May 2018)

TEXT BOOKS:

1. Krishna Kant, "Microprocessor and Microcontrollers", Eastern Company Edition, Prentice Hall of India, New Delhi , 2007.
2. R.S. Gaonkar, 'Microprocessor Architecture Programming and Application', with 8085, Wiley Eastern Ltd., New Delhi, 2013.
3. Soumitra Kumar Mandal, Microprocessor & Microcontroller Architecture, Programming & Interfacing using 8085,8086,8051,McGraw Hill Edu,2013.

REFERENCES:

1. Muhammad Ali Mazidi & Janice Gilli Mazidi, R.D.Kinely 'The 8051 Micro Controller and Embedded Systems', PHI Pearson Education, 5th Indian reprint, 2003.
2. N.Senthil Kumar, M.Saravanan, S.Jeevananthan, 'Microprocessors and Microcontrollers', Oxford,2013.
3. Valder – Perez, "Microcontroller – Fundamentals and Applications with Pic," Yeesdee Publishers, Tayler & Francis, 2013.

Prepared By
1. Mrs.J.Srivandhana.,AP/MCA
2. Mrs.T.Kala.,AP/MCA

Verified By
HOD/MCA

Approved By
Principal

PART A

1. What is a microcomputer?

A computer that is designed using a microprocessor as its CPU is called microcomputer.

2. What is Microprocessor? Give the power supply & clock frequency of 8085.

A microprocessor is a **multipurpose, programmable logic device** that reads **binary instructions** from a **storage device** called **memory**. **Accepts** binary data as **input** and **processes** data **according to those instructions** and **provides result** as **output**. The power supply of 8085 is +5V and clock frequency in 3MHz.

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.

3. List the components of microprocessor (single board microcomputer) based system

The microprocessor based system consist of microprocessor as CPU, semiconductor memories like EPROM and RAM, input device, output device and interfacing devices.

4. What are the basic units of a microprocessor?

The basic units or blocks of a microprocessor are **ALU**, an **array of registers** and **control unit.**

5. List few applications of microprocessor-based system.

It is used:

- For measurements, display and control of current, voltage, temperature, pressure, etc.
- For traffic control and industrial tool control.
- For speed control of machines.

6. Define bit, byte and word.

A digit of the binary number or code is called bit. Also, the bit is the fundamental storage unit of computer memory.

The 8-bit (8-digit) binary number or code is called **byte** and 16-bit binary number or code is called **word.** (Some microprocessor manufactures refer the basic data size operated by the processor as word).

7.Specify the size of data, address, and memory word and memory capacity of 8085 microprocessor. ***(April/May-2011)***

8085 has data lines (D_0-D_7)

16 address lines (A_0-A_{15})

Memory capacity = 2^{16} = 64 k Bytes of memory.

8. What are the functions of an accumulator?[Apr/May 2018]

The accumulator is the register associated with the ALU operations and sometimes I/O operations. It is an integral part of ALU. It holds one of data to be processed by ALU. It also temporarily stores the result of the operation performed by the ALU.

9. List the 16 – bit registers of 8085 microprocessor.

Stack pointer (SP) and Program counter (PC).

10. List the allowed register pairs of 8085.

- B-C register pair
- D-E register pair
- H-L register pair

11. Mention the purpose of SID and SOD lines.

- 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.

12. What happens to the 8085 processor when it is resetted?

When the 8085 processor is resetted it executes the first instruction at the OOOOH location. The 8085 resets (clears) instruction register, interrupt mask bits and other registers.

13. What is an Opcode and Operand?

The part of the instruction that specifies the operation to be performed is called the operation code or opcode.

The data on which the operation is to be performed is called as an Operand.

14. What is the function of IO/ signal in the 8085?

It is a status signal. It is used to differentiate between memory locations and I/O operations. When this signal is low (**IO/** = 0) it denotes the memory related operations. When this signal is high (**IO/** = 1) it denotes an I/O operation.

15. How many address lines in a 4096 x 8 EPROM CHIP?

12 address lines.

16. What is meant by polling?

Polling or device polling is a process which identifies the device that has interrupted the microprocessor.

17. What is meant by interrupt?

Interrupt is an external signal that causes a microprocessor to jump to a specific subroutine.

Definition: The meaning of 'interrupts' is to break the sequence of operation.While the cpu is executing a program,on 'interrupt' breaks the normal sequence of execution of instructions, diverts its execution to some other program called Interrupt Service Routine (ISR).After executing ISR , the control is transferred back again to the main program.Interrupt processing is an alternative to polling.

18. Explain priority interrupts of 8085. How many interrupts does 8085 have, mention them.[Nov/Dec 2017]

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.

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

19. What is the signal classification of 8085?

All the signals of 8085 can be classified into 6 groups

- Address bus
- Data bus
- Control and status signals
- Power supply and frequency signals
- Externally initiated signals
- Serial I/O ports

20. How clock signals are generated in 8085 and what is the frequency of the internal clock? *(May/June-14)*

The 8085 has the clock generation circuit on the chip but an external quartz crystal or L C circuit or RC circuit should be connected at the pins XI and X2. The maximum internal clock frequency of 8085A is 3.03 MHz

21. What are operations performed on data in 8085

The various operations performed are

- Store 8-bit data
- Perform arithmetic and logical operations
- Test for conditions
- Sequence the execution of instructions
- Store data temporarily during execution in the defined R/W memory locations called the stack

22. List the steps involved to fetch a byte in 8085?

- The PC places the 16-bit memory address on the address bus
- The control unit sends the control signal RD to enable the memory chip
- The byte from the memory location is placed on the data bus
- The byte is placed in the instruction decoder of the microprocessor and the task is carried out according to the instruction

23. What is meant by wait state?

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 the 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.

24. What do you mean by multiplexing the bus?

The signal lines AD7-AD0 is bidirectional. They are used as the lowered address bus as well as the data bus. In executing the instructions, during the earlier part of the cycle, these lines are used as the low order address bus. During the later part of the cycle, these lines are used as the data bus. This is known as multiplexing the bus,

25. Where is the READY signal used?

READY is an input signal to the processor, used by the memory or I/O devices to get extra time for data transfer or to introduce wait states in the bus cycles.

26. List the control and status signals of 8085 microprocessor and mention its need.
(Nov/Dec-2012)

There are two control signal (RD and WR), three status signals (IO/M, S1, S0). They are used to identify the basic type of internal operation done by the processor.

27. What are HOLD and HLDA and how it is used?

Hold and hold acknowledge signals are used for the Direct Memory Access (DMA) type of data transfer. The DMA controller place a high on HOLD pins in order to take control of the system bus. The HOLD request is acknowledged by the 8085 by driving all its tri-stated pins to high impedance state and asserting HLDA signal high.

28. Basic concepts in memory interfacing

The primary function of memory interfacing is that the microprocessor should be able to read from and write into a given register of a memory chip.

29. Why interfacing is needed for I/O devices?

Generally I/O devices are slow devices. Therefore the speed of I/O devices does not match with the speed of microprocessor. And so an interface is provided between system bus and I/O devices.

30. What is the drawback in memory mapped I/O?

When I/O devices are memory mapped, some of the addresses are allotted to I/O devices and so the full address space cannot be used for addressing memory (i.e., physical memory address space will be reduced). Hence memory mapping is useful only for small systems, where the memory requirement is less.

31. What does memory-mapping mean?

The memory mapping is the process of interfacing memories to microprocessor and allocating addresses to each memory locations.

32. How the 8085 processor differentiates a memory access (read/write) and I/O access (read/write)?

The memory access and I/O access is differentiated using 10 I M signal.

The 8085 processor asserts 10 IM low for memory read/write operation and 10 I M is asserted high for 1/0 read/write operation.

33. What is the need for system clock and how it is generated in 8085?

The system clock is necessary for synchronizing various internal operations or devices in the microprocessor and to synchronize the microprocessor with other peripherals in the system.

34. Why EPROM is mapped at the beginning of memory space in 8085system?

In 8085 microprocessor, after a reset, the program counter will have OOOOH address. If the monitor program is stored from this address then after a reset, it will be executed automatically. The monitor program is a permanent program and stored in EPROM memory.

If EPROM memory is mapped at the beginning of memory space, i.e., at OOOOH, then the monitor program will be executed automatically after a reset.

35. What is the need for timing diagram? *(April/May-15)*

The timing diagram provides information regarding the status of various signals, when a machine cycle is executed. The knowledge of timing diagram is essential for system designer to select matched peripheral devices like memories, latches, ports, etc., to form a microprocessor system.

36. What is Instruction cycle & machine cycle? How many machine cycles constitute one instruction cycle in 8085?

Instruction cycle

The sequence of operations that a processor has to carry out while executing the instructions is called as Instruction cycle. Each instruction cycle of a processor indium consists of a number of machine cycles.

Machine cycle

Machine cycle is defined as the time required to complete one operation of accessing memory, I/O, or acknowledging an external request. This cycle may consist of three to six T-states.

Each instruction of the 8085 processor consists of **one to five machine** cycles.

37. What is processor cycle (Machine cycle)?

The processor cycle or machine cycle is the basic operation performed by the processor. To execute an instruction, the processor will run one or more machine cycles in a particular order.

38. Define T-State.

T-State is defined as one subdivision of the operation performed in one clock period. These subdivisions are internal states synchronized with the system clock, and each T-State is precisely equal to one clock period.

39. What is fetch and execute cycle?

In general, the instruction cycle of an instruction can be divided into fetch and execute cycles. The fetch cycle is executed to fetch the opcode from memory. The execute cycle is executed to decode the instruction and to perform the work instructed by the instruction.

40. What is opcode fetch cycle?

The opcode fetch cycle is a machine cycle executed to fetch the opcode of an instruction stored in memory. Every instruction starts with opcode fetch machine cycle.

41. What operation is performed during first T -state of every machine cycle in 8085?

In 8085, during the first T -state of every machine cycle the low byte address is latched into an external latch using ALE signal.

42. What is ALE signal and READY signal? *(Nov/Dec-09) (Nov/Dec-14)*

The ALE (Address Latch Enable) is a signal used to de-multiplex the address and data lines, using an external latch. It is used to enable the external latch.

The READY signal of the 8085 microprocessor is sampled approximately one half clocks after the trailing edge of ALE and if not asserted, repeatedly one full clock cycle later until it is asserted.

43. What is the use of ALE?(OR)Mention the use of ALE. *(Nov/Dec-14)(Nov/Dec-15)*

The ALE is used to latch the lower order address so that it can be available in T2 and T3 and used for identifying the memory address. During T1 the ALE goes high, the latch is transparent, and the output changes according to the input data, so the output of the latch is the lower order address. When ALE goes low the lower order address is latched until the next ALE.

44. How many machine cycles does 8085 have, mention them

The 8085 have seven machine cycles. They are

- Opcode fetch
- Memory read
- Memory write
- I/O read
- I/O write
- Interrupt acknowledge
- Bus idle

45. Why status signals are provided in microprocessor?

The status signals can be used by the system designer to track the internal operations of the processor. Also, it can be used for memory expansion (by providing separate memory banks for program & data and selecting the bank using status signals).

46. Give the register organization of 8085

W (8) Temp. Reg	Z (8) Temp. Reg
B (8) Register	C (8) Register
D (8) Register	E (8) Register
H (8) Register	L(8) Register
Stack Pointer (16)	
Program Counter (16)	

47. What is interrupt I/O?

If the I/O devices initiate the data transfer through interrupt then the I/O is called interrupt driven I/O.

48. When the 8085 processor checks for an interrupt?

In the second T -state of the last machine cycle of every instruction, the 8085 processor checks whether an interrupt request is made or not.

49. What is interrupt acknowledge cycle?

The interrupt acknowledge cycle is a machine cycle executed by 8085 processor to get the address of the interrupt service routine in-order to service the interrupt device.

50. How the interrupts are affected by system reset?

Whenever the processor or system is resetted, all the interrupts except TRAP are disabled. In order to enable the interrupts, El instruction has to be executed after a reset.

51. What is Software interrupts?

The Software interrupts are program instructions. These instructions are inserted at desired locations in a program. While running a program, if software interrupt instruction is encountered then the processor executes an interrupt service routine.

52. What is TRAP? *(May/June -2012)*

The TRAP is non-Maskable interrupt of8085. It is not disabled by processor reset.

53. What is the difference between Hardware and Software interrupt?

The Software interrupt is initiated by the main program, but the Hardware interrupt is initiated by an external device. In 8085, the Software interrupt cannot be disabled or masked but the Hardware interrupt except TRAP can be disabled or masked

54. What is vectored and Non- Vectored interrupt?

When an interrupt is accepted, if the processor control branches to a specific address defined by the manufacturer then the interrupt is called vectored interrupt. In Non-vectored interrupt there is no specific address for storing the interrupt service routine. Hence the interrupted device should give the address of the interrupt service routine.

55. List the Software and Hardware interrupts of 8085?

Software interrupts: RST 0, RSTl, RST 2, RST 3, RST 4, RST 5, RST 6 and RST 7. Hardware interrupts: TRAP, RST 7.5, RST 6.5, RST 5.5 and INTR.

or after reorganization of interrupt.

56. What is masking and why it is required?

Masking is preventing the interrupt from disturbing the current program execution. When the processor is performing an important job (process) and if the process should not be interrupted then all the interrupts should be masked or disabled. In processor with multiple 'interrupts, the lower priority interrupt can be masked so as to prevent it from interrupting, the execution of interrupt service routine of higher priority interrupt.

57. When the 8085 processor accept hardware interrupt?

The processor keeps on checking the interrupt pins at the second T-state of last Machine cycle of every instruction. If the processor finds a valid interrupt signal and if the interrupt is unmasked and enabled then the processor accepts the interrupt. The acceptance of the interrupt is acknowledged by sending an OOA signal to the interrupted device.

58. When the 8085 processor will disable the interrupt system?

The interrupts of 8085 except TRAP are disabled after anyone of the following operations

- Executing El instruction.

- System or processor reset.
- After reorganization (acceptance) of an interrupt

59. How the vector address is generated for the INTR interrupt of 8085?

For the interrupt INTR, the interrupting device has to place either RST opcode or CALL opcode followed by l6-bit address. IRST opcode is placed then the corresponding vector address is generated by the processor. In case of CALL opcode the given l6-bit address will be the vector address.

60. Define Flags of 8085? *(May/June-14)*[Nov/Dec 2017]

The different flags and their positions in flag register are as shown in Fig. 4.6.

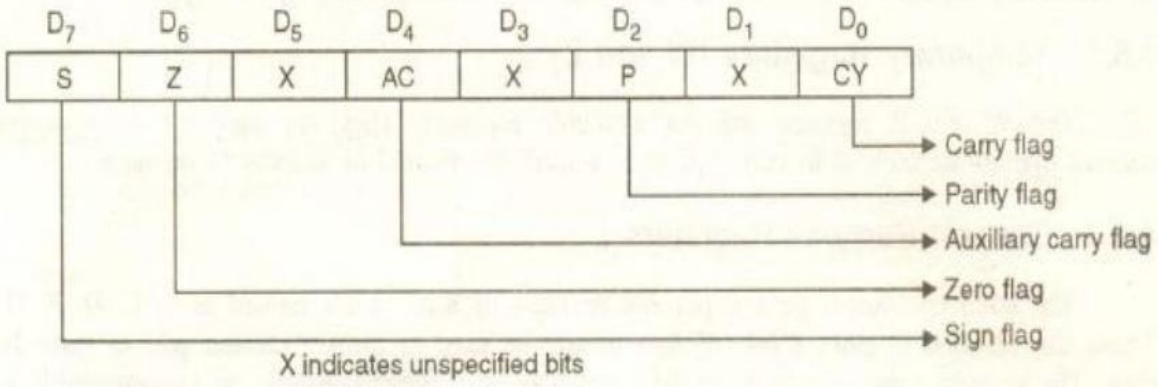


Fig. 4.6 : 8085 flag register

The flag register in the 8085 microprocessor.
- The flag register in 8085 is an 8-bit register which contains 5 bit positions.
- These five flags are of 1bit F/F and are known as **zero, sign, carry, parity and auxiliary carry**.
- For **sign flag** if the result of an MSB operation is 1 then it is set else it is reset.
- The **zero flag** is set of the result of an instruction is zero.
- The **auxiliary carry flag** is used for BCD operations, not free to the programmer.
- The **carry flag** is used for carrying and borrowing in case of addition and subtraction operations.
- The **parity flag** is used for results containing an even number of one's.

61. Difference between memory mapped I/O and peripheral I/O? [Apr/May 2018]

MEMORY MAPPEED I/O	PERIPHERAL I/O
16-bit device address	8-bit device address
The data transfer between any general-purpose register and I/O port	The data transfer only between accumulator and I/O port
The memory map(64kb)is shared between I/O device and system memory	The I/O map is independent of the memory map,256 input device and 256 output device
More hardware is required to decode 16-bit address	Less hardware is required to decode 8-bit address

62. Define the functions of parity flag and zero flag in 8085? *(May/June-2012)*

Zero flag: if result is 0 than it is set condition i.e.1 otherwise it is reset i.e. is 0.

Parity flag: if number of 1's in the answer is even than it is set condition i.e.1 otherwise it is reset i.e. is 0.

63. What is the use of Stack Pointer(SP)? (Nov/Dec-2015)

Stack Pointer (SP): The stack is a reserved area of the memory in the RAM where temporary information may be stored. A 16-bit stack pointer is used to hold the address of the most recent stack entry.

64. Write an 8085 assembly program to add two digit BCD numbers in memory locations 5000H and 5001H and store the result in memory location 5002H.(Nov 2016)

```
MOV A, 5000H
INR A
ADD E
DAA
STA 2300H
MOV A, H
ADC D
DAA
STA 2301H
HLT
```

65. List out the machine cycles for executing the instruction MVI A,34 H (Nov -16)

This is a 2 byte instruction so it requires 2 machine cycles to fetch the instruction

1. Op code fetch and 2. Memory read

66. Why data bus is bi-directional? (Apr/May -17)

The data bus is bidirectional because it takes the data from peripherals & also give the data to peripherals.

67. List out the machine cycles of 8085 microprocessor.(Apr/May -17)

The 8085 microprocessor has seven basic machine cycles. They are

- Opcode fetch cycle (4T)
- Memory read cycle (3 T)
- Memory write cycle (3 T)
- I/O read cycle (3 T)
- I/O write cycle (3 T)
- Interrupt acknowledge cycle
- Bus idle cycle

68. List out the uses of Microprocessors?

1. Calculators
2. Accounting system
3. Games machine

4. Complex Industrial Controllers
5. Traffic light Control
6. Data acquisition systems
7. Multi user, multi-function environments
8. Military applications
9. Communication systems**.**

69) List out the Advantages of Microprocessor.

- Computational/Processing speed is high & Simplifies system design
- Intelligence has been brought to systems
- Automation of industrial process and office automation
- Flexible.
- Compact in size.
- Maintenance is easier.

70) Why address bus of Intel 8085 is unidirectional and data bus is bidirectional? [Apr/May 2016]

Why address bus is unidirectional?

The address is an identification number used by the microprocessor to identify or access a memory location or I / O device. It is an output signal from the processor. Hence the address bus is unidirectional.

Why data bus is bi-directional?

The microprocessor has to fetch (read) the data from memory or input device for processing and after processing, it has to store (write) the data to memory or output device. Hence the data bus is bi-directional.

71) What is the function of NOP instruction in 8085.

NOP operation in 8085 is used to insert a delay of 4 T-states when microprocessor is communicating with slow peripheral devices. When enabling interrupts via the EI instruction, the interrupts are enabled after the successful execution of the next instruction.

72) What is a stack pointer register, describe briefly.

-The Stack pointer is a sixteen bit register used to point at the stack.
- In read write memory the locations at which temporary data and return addresses are stored is known as the stack.
- In simple words stack acts like an auto decrement facility in the system.
- The initialization of the stack top is done with the help of an instruction LXI SP.
- In order to avoid program crashes a program should always be written at one end and initialized at the other.

73) What is the use of DAD and DAA instruction in 8085 microprocessor? [Apr/May 2016]

The DAD instruction (Double Add) allows 16-bit addition between the HL register pair and any one of the BC, DE, HL, or SP register pairs, putting the result in HL. It takes a single operand which may be B, D, H, or SP. The carry flag is set to indicate overflow.

The DAA instruction (Decimal Adjust Accumulator) allows conversion of the 8-bit accumulator value to Binary Coded Decimal (BCD). If the low-order 4 bits of the accumulator are greater than 9, or the auxiliary carry flag is set, 6 is added to the low-order 4 bits of accumulator, then if the high-order 4 bits of the accumulator are greater than 9, or the carry flag is set, 6 is added to the high-order 4 bits of the accumulator.

74) What differences can you state between the HLT and HOLD states?

- The Hold is a hardware input whereas HLT is a software instruction.
- When the HLT state is executed the processor simply stops and the buses are driven to tri state. No form of acknowledgement signal is given out by the processor.
- In case of HOLD the processor goes into hold state but the buses are not driven to tri state.
- When the processor goes into the HOLD state it gives out an HLDA signal. This signal can be made to use by other devices.

75) Differentiate between 8085 and 8086 processors.

8085	8086
8 bit microprocessor	16-bit microprocessor
16-bit address bus	20-bit address bus
General purpose regi. Size are of 8 -bit	General purpose regi. Size are of 16 -bit
Instruction queue is not present	6 byte Instruction queue is present
Segmentation is not supported	Segmentation of memory is supported
Doses not support pipeline architecture	support pipeline architecture
Maximum clock frequency is 3MHz	Maximum clock frequency is 8MHz

PART B

1. List out the features of 8085 microprocessor.

The features of Intel 8085 microprocessor are as follows:

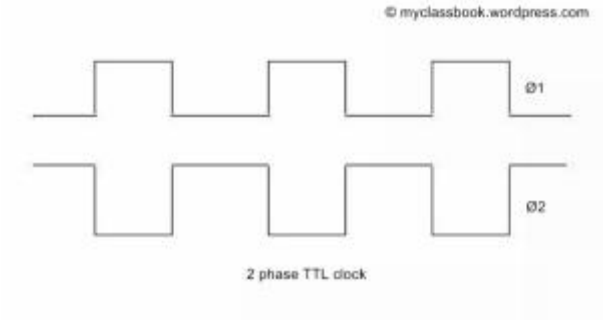
1) 8085 microprocessor is an 8 bit microprocessor. I.e. it can accept or provide 8 bit data simultaneously.

2) 8085 microprocessor is a single chip, NMOS device implemented with 6200 transistors.

3) 8085 microprocessor requires a single +5V DC power supply.

4) 8085 microprocessor provides on chip clock generator, therefore there is no need of external clock generator, but it requires external tuned circuit like LC, RC or crystal.

5) 8085 microprocessor requires two phase, 50% duty cycle, TTL clock. These clock signals are generated by an internal clock generator (refer following figure).



2 Phase TTL clock

6) The maximum clock frequency of 8085 microprocessor is 3MHz where as minimum clock frequency is 500 KHz.

7) 8085 provides 74 instructions with the following addressing modes:

- register
- direct
- immediate
- indirect
- implied.

8) The data bus is multiplexed with the address bus, hence it requires external hardware to separate data lines from address lines (this is one of the disadvantage of 8085).

9) 8085 microprocessor provides 16 address lines, therefore it can access 2^16 = 64K bytes of memory.

10) It generates 8 bit I/O address, hence it can access 2^8 = 256 input ports and 256 output ports.

11) It performs the following arithmetic and logical operations.

- 8 bit, 16 bit binary addition
- 2 digit BCD addition
- 8 bit binary subtraction
- logical AND, OR, EXOR
- complement and shift operations.

12) 8085 microprocessor has five hardware interrupts: TRAP, RST 5.5, RST 6.5, RST 7.5, INTR

13) The hardware interrupt capability of 8085 microprocessor can be increased by providing external hardware.

14) 8085 microprocessor has capability to share its bus with external bus controller (direct memory access controller); for transferring large amount of data from memory to I/O and vice versa.

15) 8085 microprocessor provides one accumulator, one flag register, 6 general purpose registers and two special purpose registers.
16) It provides status for advanced control signals. (Advanced control signals are used in large systems).
17) 8085 microprocessor can be used to implement three chip microcomputer (8085, 8155, 8355)
18) 8085 microprocessor provides two serial I/O lines which are SOD and SID; it means, serial peripherals can be interfaced with 8085 microprocessor directly.

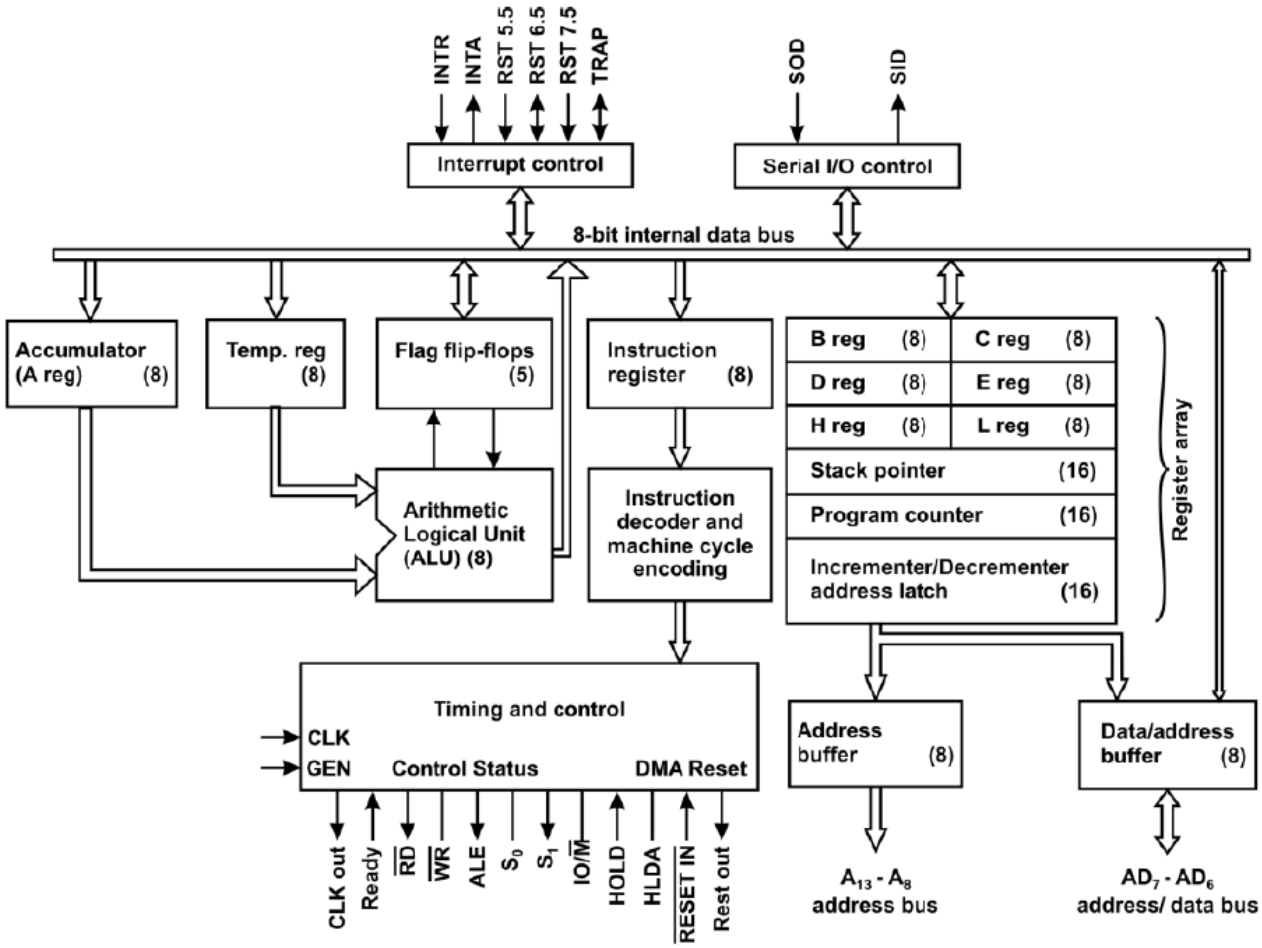
2) Explain with neat sketch about the architecture of 8085.(OR) Explain the architecture of 8085 microprocessor with block diagram. (Nov/Dec-09, April/May-11, May/June-12, Nov/Dec-15, April/May-15, Nov/Dec -16)

Architecture of 8085 microprocessor

8085 consists of various units and each unit performs its own functions. The various units of a microprocessor are listed below

1. Accumulator
2. Arithmetic and logic Unit
3. General purpose register
4. Program counter
5. Stack pointer
6. Temporary register
7. Flags
8. Instruction register and Decoder
9. Timing and Control unit
10. Interrupt control
11. Serial Input/output control
12. Address buffer and Address-Data buffer

8085 Architecture



1) Accumulator

- Accumulator is nothing but a **register** which can **hold 8-bit data**. Accumulator aids in storing two quantities.
- The data to be processed by arithmetic and logic unit is stored in accumulator.
- It also stores the result of the operation carried out by the Arithmetic and Logic unit.
- The accumulator is also called an 8-bit register.
- The accumulator is connected to Internal Data bus and ALU (arithmetic and logic unit).
- The accumulator can be used to **send or receive data** from the Internal Data bus.

2) Arithmetic and Logic Unit

There is always a need to perform arithmetic operations like +, -, *, / and to perform logical operations like AND, OR, NOT etc. So there is a necessity for creating a separate unit which can perform such types of operations. These operations are performed by the Arithmetic and Logic Unit (ALU). ALU performs these operations on 8-bit data.

But these operations cannot be performed unless we have an input (or) data on which the desired operation is to be performed. So from where do these inputs reach the ALU? For this purpose accumulator is used. ALU gets its Input from accumulator and temporary register. After processing the necessary operations, the result is stored back in accumulator.

3) General Purpose Registers

Apart from accumulator **8085 consists of six special types of registers called General Purpose Registers.** These general purpose registers are used to hold data like any other registers. The general purpose registers in 8085 processors are B, C, D, E, H and L. Each register can **hold 8-bit data**. Apart from the above function these registers can also be used to work in **pairs to hold 16-bit data**. They can work in pairs such as **B-C, D-E and H-L** to store 16-bit data. The H-L pair works as a memory pointer. A memory pointer holds the address of a particular memory location. They can store 16-bit address as they work in pair.

4) Program Counter and 5) Stack Pointer

Program counter is a **special purpose register**. Consider that an instruction is being executed by processor. As soon as the ALU finished executing the instruction, the processor looks for the next instruction to be executed. So, there is a necessity for holding the address of the next instruction to be executed in order to save time.

This is taken care by the program counter. A program counter stores the address of the next instruction to be executed. In other words the program counter keeps track of the memory address of the instructions that are being executed by the microprocessor and the memory address of the next instruction that is going to be executed.

Program counter is a 16-bit register. **Stack pointer** is also a 16-bit register which is used as a memory pointer. A stack is nothing but the portion of RAM (Random access memory). **So does that mean the stack pointer points to portion of RAM? Yes**. Stack pointer maintains the address of the last byte that is entered into stack. Each time when the data is loaded into stack, Stack pointer gets decremented. Conversely it is incremented when data is retrieved from stack.

6) Temporary Register:

As the name suggests this register acts as a temporary memory during the arithmetic and logical operations. Unlike other registers, this temporary register can only be accessed by the microprocessor and it is completely inaccessible to programmers. Temporary register is an 8-bit register. In the next article let us discuss about the FLAGS.

7) Flags(Nov/Dec 2017)

Flags are nothing but a group of individual Flip-flops. The flags are mainly associated with arithmetic and logic operations. The flags will show either a logical (0 or 1) (i.e.) a set or reset depending on the data conditions in accumulator or various other registers. A flag is actually a latch which can hold some bits of information. It alerts the processor that some event has taken place.

The different flags and their positions in flag register are as shown in Fig. 4.6.

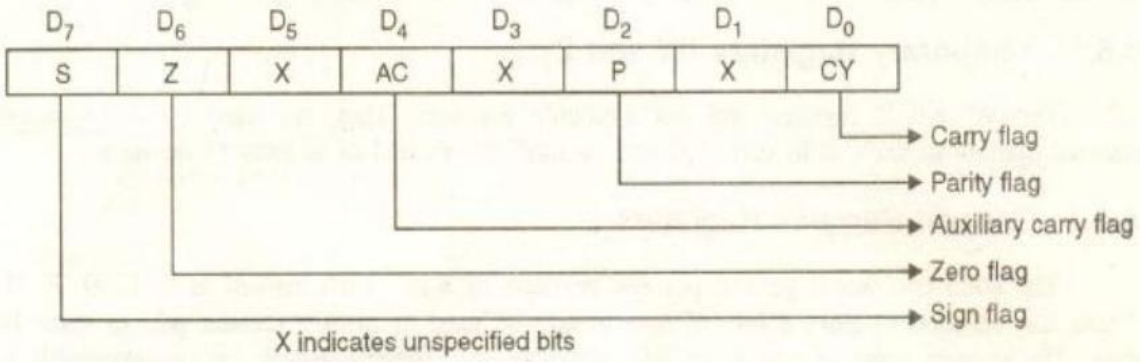


Fig. 4.6 : 8085 flag register

(1) **CY** **Carry flag :** If an operation performed in ALU, generates a carry from D_7 bit, the CY flag is SET. It works as 9th bit for addition and as borrow flag for subtraction. If there is no carry/borrow, out of MSB bit, i.e. D_7, of the result, CY flag is RESET.

(2) **AC** **Auxiliary carry flag :** If an operation performed in ALU generates a carry from lower nibble (i.e. D_0 to D_3) to upper nibble (i.e. D_4 to D_7) the AC flag is set i.e. a carry given by D_3 bit to D_4 is a AC flag.

This is not a general purpose flag, it is only used internally by microprocessor to perform binary to BCD conversion. It is not available for programmer for any decision making.

(3) **Z** **Zero flag :** If an operation in ALU results in zero, the zero flag is SET. If the result is not zero, the zero flag is RESET.

(4) **S** **Sign flag :** In sign magnitude format, the sign of a number is indicated by MSB bit. If MSB bit = 0, the number is positive and if MSB bit = 1, the number is negative. In 8085 MSB bit is D_7 bit. The sign flag is exact replica of D_7 bit of the result. If $D_7 = 1$, the flag is set and if $D_7 = 0$, the flag is reset. This flag can be used to perform operation on signed numbers.

(5) **P** **Parity flag :** This bit is used to indicate the parity of the result. If the result contain even number of 1's this flag is set. If the result contains odd number of 1's this flag is reset. i.e. by insertion of flag bit microprocessor maintains odd parity for result.

Why are they called flags?

The possible solution is from the small flags which are found on the mail boxes in America. The small flag indicates that there is a mail in the mail box. Similarly this denotes that an event has occurred in the processor. Intel processors have a set of 5 flags.

- Carry flag
- Parity flag
- Auxiliary carry flag
- Zero flag
- Sign flag

Consider two binary numbers. For example: 1100 0000 & 1000 0000

When we add the above two numbers, a carry is generated in the most significant bit. The number in the extreme right is least significant bit, while the number in extreme left is most significant bit. So a ninth bit is generated due to the carry. So how to accommodate 9th bit in an 8 bit register? For this purpose the Carry flag is used. The carry flag is set whenever a carry is

generated and reset whenever there is no carry. But there is an auxiliary carry flag? What is the difference between the carry flag and auxiliary carry flag?

Let's discuss with an example. Consider the two numbers given below

0000 0100, 0000 0101 .

When we add both the numbers a carry is generated in the third bit from the least significant bit. This sets the auxiliary carry f lag. When there is no carry, the auxiliary carry flag is reset. So whenever there is a carry in the most significant bit Carry flag is set. While an auxiliary carry flag is set only when a carry is generated in bits other than the most significant bit. Parity checks whether its even or add parity. This flag returns a 0 if it is odd parity and returns a 1 if it is an even parity. Sometimes they are also called as parity bit which is used to check errors while data transmission is carried out. Zero flag shows whether the output of the operation is 0 or not. If the value of Zero flag is 0 then the result of operation is not zero. If it is zero the flag returns value

1. Sign flag shows whether the output of operation has positive sign or negative sign. A value 0 is returned for positive sign and 1 is returned for negative sign. Instruction Register and Decoder Instruction register is 8-bit register just like every other register of microprocessor. Consider an instruction.
2. The instruction may be anything like adding two data's, moving a data, copying a data etc. When such an instruction is fetched from memory, it is directed to Instruction register. So the instruction registers are specifically to store the instructions that are fetched from memory. There is an Instruction decoder which decodes the information present in the Instruction register for further processing.

8) Timing and Control Unit

Timing and control unit is a very important unit as it synchronizes the registers and flow of data through various registers and other units. This unit consists of an oscillator and controller sequencer which sends control signals needed for internal and external control of data and other units. The oscillator generates two-phase clock signals which aids in synchronizing all the registers of 8085 microprocessor.

Signals that are associated with Timing and control unit are:

Control Signals: READY, ALE,
Status Signals: S0, S1, IO/ $\overline{M}$
DMA Signals: HOLD, HLDA
RESET Signals: RESET IN, RESET OUT

9) Interrupt Control

As the name suggests this control interrupts a process. Consider that a microprocessor is executing the main program. Now whenever the interrupt signal is enabled or requested the microprocessor shifts the control from main program to process the incoming request and after the completion of request, the control goes back to the main program. For example an Input/output device may send an interrupt signal to notify that the data is ready for input. The microprocessor temporarily stops the execution of main program and transfers control to I/O device. After collecting the input data the control is transferred back to main program.

Interrupt signals present in 8085 are:

1. INTR
2. RST 7.5
3. RST 6.5
4. RST 5.5
5. TRAP

Of the above four interrupts TRAP is a NON-MASKABLE interrupt control and other three are maskable interrupts. A non-maskable interrupt is an interrupt which is given the highest priority in the order of interrupts. Suppose you want an instruction to be processed immediately, then you can give the instruction as a non-maskable interrupt. Further the non-maskable interrupt cannot be disabled by programmer at any point of time.Whereas the maskable interrupts can be disabled and enabled using EI and DI instructions. Among the maskable interrupts RST 7.5 is given the highest priority above RST 6.5 and least priority is given to INTR.

10) Serial I/O control

The input and output of serial data can be carried out using 2 instructions in 8085.

- SID-Serial Input Data
- SOD-Serial Output Data

Two more instructions are used to perform serial-parallel conversion needed for serial I/O devices. SIM RIM

11) Address buffer and Address-Data buffer

The contents of the stack pointer and program counter are loaded into the address buffer and address-data buffer. These buffers are then used to drive the external address bus and address-data bus. As the memory and I/O chips are connected to these buses, the CPU can exchange desired data to the memory and I/O chips. The address-data buffer is not only connected to the external data bus but also to the internal data bus which consists of 8 -bits. The address data buffer can both send and receive data from internal data bus. Address bus and Data bus:

We know that 8085 is an 8-bit microprocessor. So the data bus present in the microprocessor is also 8-bits wide. So 8-bits of data can be transmitted from or to the microprocessor. But 8085 processor requires 16 bit address bus as the memory addresses are 16-bit wide. The 8 most significant bits of the address are transmitted with the help of address bus and the 8 least significant bits are transmitted with the help of multiplexed address/data bus. The eight bit data bus is multiplexed with the eight least significant bits of address bus.

12) Address Bus/Data Bus:

The address/data bus is time multiplexed. This means for few microseconds, the 8 least significant bits of address are generated, while for next few seconds the same pin generates the data. This is called Time multiplexing. But there are situations where there is a need to transmit both data and address simultaneously. For this purpose a signal called ALE (address latch enable) is used. ALE signal holds the obtained address in its latch for a long time until the data

is obtained and so when the microprocessor sends the data next time the address is also available at the output latch. This technique is called Address/Data demultiplexing.

3) Register structure of 8085.Explain the registers of 8085 microprocessor[Apr/may-17]

The 8085 registers are classified as:

1. General Purpose Registers
2. Temporary Registers a) Temporary data register b) W and Z registers
3. Special Purpose Registers a) Accumulator b) Flag registers c) Instruction register
4. Sixteen Bit Registers a) Program Counter (PC) b) Stack Pointer (SP)

1. General Purpose Registers

B, C, D, E, H and L are 8-bit general purpose registers can be used as a separate 8-bit registers or as 16-bit register pairs, BC, DE, and HL. When used in register pair mode, the high order byte resides in the first register (i.e. in B when BC is used as a register pair) and the low order

byte in the second (i.e. in C when BC is used as a register pair).HL pair also functions as a data pointer or memory pointer. These are also called scratchpad registers, as user can store data in them.

2. Temporary Registers

a) Temporary Data Register: The ALU has two inputs. One input is supplied by the accumulator and other from temporary data register. The programmer cannot access this temporary data register. However, it is internally used for execution of most of the arithmetic and logical instructions.

b) **W and Z Registers**: W and Z registers are temporary registers. These registers are used to hold 8-bit data during execution of some instructions. These registers are not available for programmer, since 8085 uses them internally.

3) Special Purpose Registers

a) Register A (Accumulator):

It is a tri-state eight bit register. It is extensively used in arithmetic, logic, load, and store operations, as well as in, input/output (I/O) operations.

Most of the times the result of arithmetic and logical operations is stored in the register A. Hence it is also identified as accumulator.

b) Flag Register: It is an 8-bit register, in which five of the bits carry significant information in the form of flags: S (Sign flag), Z (Zero flag), AC (Auxiliary Carry flag), P (Parity flag), and CY (Carry flag), as shown in Fig. 1.3.

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

Fig. 1.3 Flag register

S-Sign flag:
After the execution of arithmetic or logical operations, if bit D7 of the result is 1, the sign flag is set. In a given byte if D7 is 1, the number will be viewed as negative number. if D7 is 0, the number will be considered as positive number.

Z-Zero flag: The zero flag sets if the result of operation in ALU is zero and flag resets if result is non zero.
AC-Auxiliary Carry flag: This flag is set if there is an overflow out of bit 3 i.e., carry from lower nibble to higher nibble (D3 bit to D4 bit). This flag is used for BCD operations and it is not available for the programmer.
P-Parity flag: After an arithmetic or logical operation if the result has an even number of ones, i.e. even parity, the flag is set. If the parity is odd, flag is reset.
CY-Carry flag: This flag is set if there is an overflow out of bit 7. The carry flag also serves as a borrow flag for subtraction.
c) Instruction Register: In a typical processor operation, the processor first fetches the opcode of instruction from memory. The CPU stores this opcode in a register called the instruction register.

4. Sixteen Bit Registers

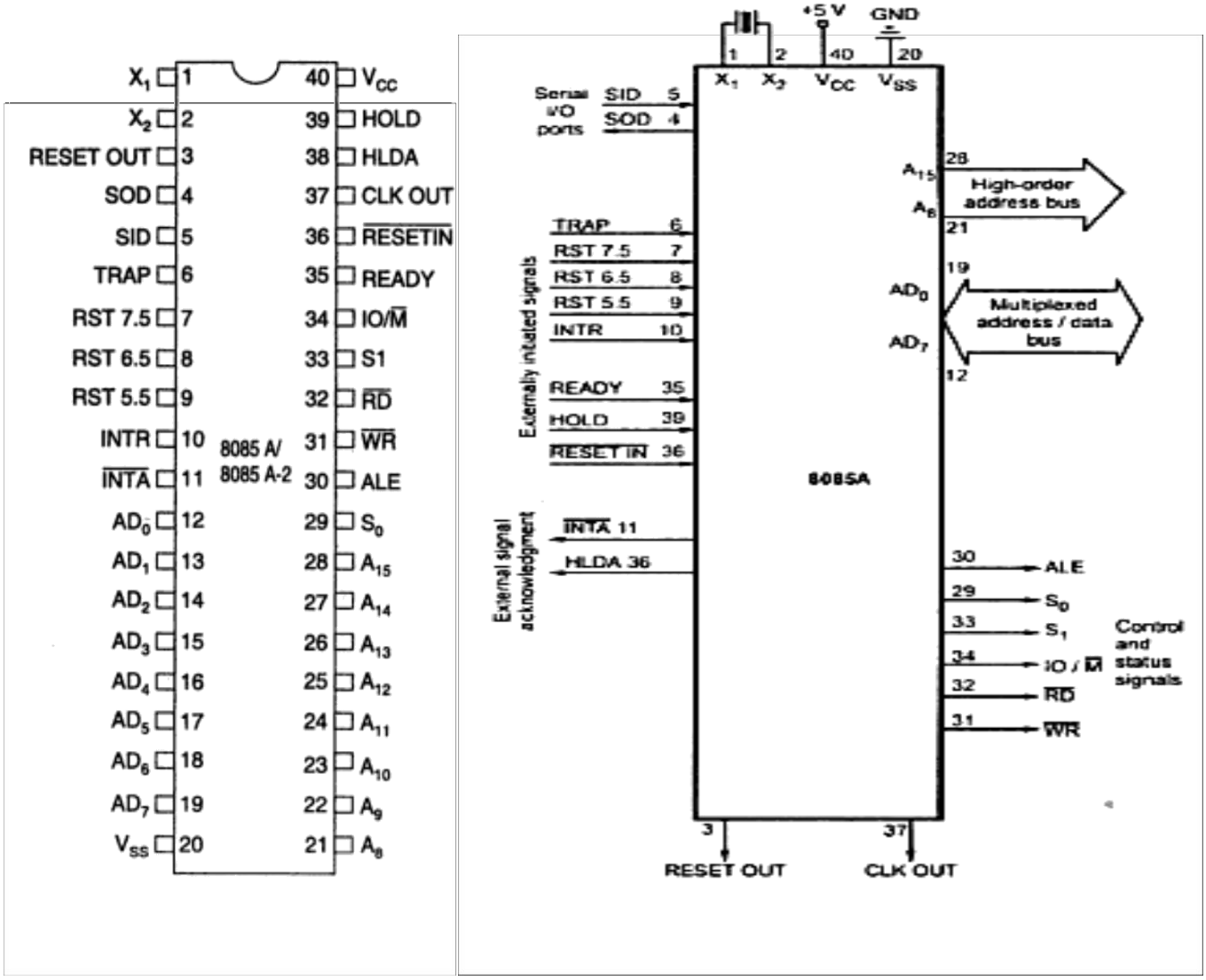
a) Program Counter (PC): Program is a sequence of instructions. Microprocessor fetches these instructions from the memory and executes them sequentially. The program counter is a special purpose register which, at a given time, stores the address of the next instruction to be fetched. Program Counter acts as a pointer to the next instruction.

How processor increments program counter depends on the nature of the instruction

b) Stack Pointer (SP): The stack is a reserved area of the memory in the RAM where temporary information may be stored. A 16-bit stack pointer is used to hold the address of the most recent stack entry.

4) Draw the pin configuration (signals) of 8085 microprocessor and explain it in detail. (May/June-12)(Nov/Dec-14(Or) With Pin diagram explain 8085 microprocessor.(Apr/May-17)[Apr/May 2018]

The 8085 microprocessor is available on a 40 pin dual in line package (DIP). The pin configuration is shown



a. Power supply and frequency signals.
b. Data bus and address bus.
c. Control bus.
d. Interrupt signals.
e. Serial I/O signals.
f. DMA signals.
g. Reset signals.

a) Power Supply and Frequency Signals

i) **VCC**: It requires a single +5 V power supply.
ii) **VSS**: Ground reference.

iii) **X1 and x2**: A tuned circuit like LC, RC or crystal is connected at these TWO pins. The internal clock generator divides oscillator frequency by 2, therefore, to operate a system at 3 MHz, the crystal of tuned circuit must have a frequency of 6 MHz

iv) **CLK OUT**: This signal is used as a system clock for other devices. Its frequency is half the oscillator frequency.

b) Data Bus and Address Bus

A) AD_O to AD_7: The 8 bit data bus (D_0–D7) is multiplexed with the lower half (A0–A7) of the 16 bit address bus.

B) A8 to A15: The upper half of the 16 bit address appears on the address lines A8to A15.

C) Control and Status Signals

Control Pins – RD, WR These are active low Read & Write pins

Status Pins – ALE, IO/M (active low), S1, S0

ALE (Address Latch Enable)-Used to de-multiplex AD7-AD0
IO/M – Used to select I/O or Memory operation
S1,S0 – Denote the status of data on data bus

d) Interrupt Signals

The 8085 has five hardware interrupt signals. RST 5.5, RST 6.5, RST 7.5, TRAP and INTR. The microprocessor recognizes interrupt requests on these lines at the end of the current instruction execution.

The (Interrupt Acknowledge) signal is **used** to indicate that the processor has acknowledged an INTR interrupt.

e) Serial I/O Signals

A) SID (Serial Input Data): This input signal is used to accept serial data bit by bit from the external device.

B) SOD (Serial O/P Data): This is an output signal which enables the transmission of serial data bit by bit to the external device.

f) DMA Signal

HOLD: This signal indicates that **another master is** requesting for the **use** of address bus, data bus and control bus.

HLDA: This active high signal is used to acknowledge HOLD request.

g) Reset Signals

A low on this pin

1. Internal sets the program counter to zero (0000H).
2. Resets the interrupt enable and HLDA flip-flops.
3. Tri-states the data bus, address bus and control bus (Note: Only during RESET is active).
4. Affects the contents of processor's registers randomly.

RESET OUT: This active high signal indicates that processor is being reset. This signal is synchronized to the processor clock and it can be used to reset other devices connected in the system.

5. Explain Memory interfacing in 8085 microprocessor (Or) Draw the interfacing diagram to interface with 8085with 2 KB RAM and 4 KB EPROM (Nov/Dec-16, Apr/May - 17)

The memory interfacing requires to:

- Select the chip
- Identify the register
- Enable the appropriate buffer.

Microprocessor system includes memory devices and I/O devices. It is important to note that microprocessor can communicate (read/write) with only one device at a time, since the data, address and control buses are common for all the devices. In order to communicate with memory or I/O devices, it is necessary to decode the address from the microprocessor. Due to this each device (memory or I/O) can be accessed independently. The following section describes common address decoding techniques.

Address Decoding Techniques:

- Absolute decoding/Full Decoding
- Linear decoding/Partial Decoding

Absolute decoding

In absolute decoding technique, all the higher address lines are decoded to select the memory chip, and the memory chip is selected only for the specified logic levels on this high-order address lines; no other logic levels can select the chip. Fig. 5.1 shows the memory interface with absolute decoding. This addressing technique is normally used in large memory systems.

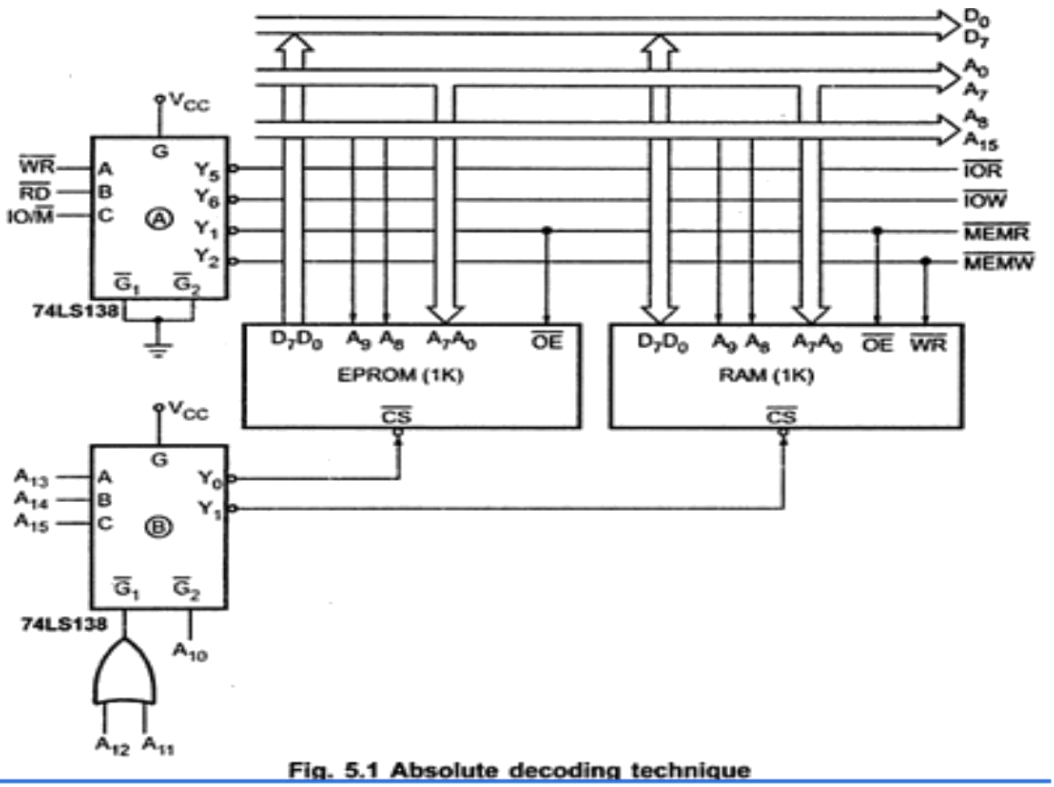


Fig. 5.1 Absolute decoding technique

Memory Map :

Memory ICs	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	Address
Starting address of EPROM	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000H
End address of EPROM	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	03FFH
Starting address of RAM	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	2000H
End address of RAM	0	0	1	0	0	0	1	1	1	1	1	1	1	1	1	1	23FFH

Table 5.1

Linear decoding

In small systems, hardware for the decoding logic can be eliminated by using individual high-order address lines to select memory chips. This is referred to as linear decoding. Fig. 5.2 shows the addressing of RAM with linear decoding technique. This technique is also called partial decoding. It reduces the cost of decoding circuit, but it has a drawback of multiple addresses (shadow addresses).

Memory Map :

Memory ICs	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	Address
Starting address of EPROM	0	X	X	X	X	X	0	0	0	0	0	0	0	0	0	0	0000H
End address of EPROM	0	X	X	X	X	X	1	1	1	1	1	1	1	1	1	1	03FFH
Starting address of RAM	1	X	X	X	X	X	0	0	0	0	0	0	0	0	0	0	8000H
End address of RAM	1	X	X	X	X	X	1	1	1	1	1	1	1	1	1	1	83FFH

Table 5.2

Full Address Decoding	Partial Address Decoding
1. All higher address lines are decoded to select the memory or I/O device.	1. Few higher address lines are decoded to select the memory or I/O device.
2. More hardware is required to design decoding logic.	2. Hardware required to design decoding logic is less and sometimes it can be eliminated.
3. Higher cost for decoding circuit.	3. Less cost for decoding circuit.
4. No multiple addresses.	4. It has a disadvantage of multiple addresses (shadow addresses).
5. Used in large systems.	5. Used in small systems.

Table 5.3

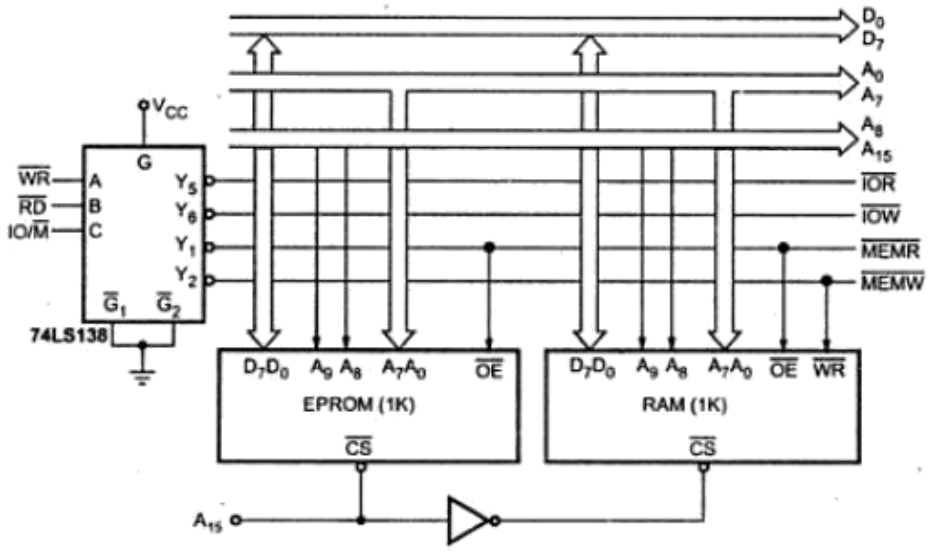


Fig. 5.2 Linear decoding

Fig. 5.2 shows the addressing of RAM with linear decoding technique. A15 address line is directly connected to the chip select signal of EPROM and after inversion it is connected to the

chip select signal of the RAM. Therefore, when the status of A_15 line is 'zero', EPROM gets selected and when the status of A15 line is 'one' RAM gets selected. The status of the other address lines is not considered, since those address lines are not used for generation of chip select signals.

6.Explain I/O Interfacing Techniques of microprocessor 8085

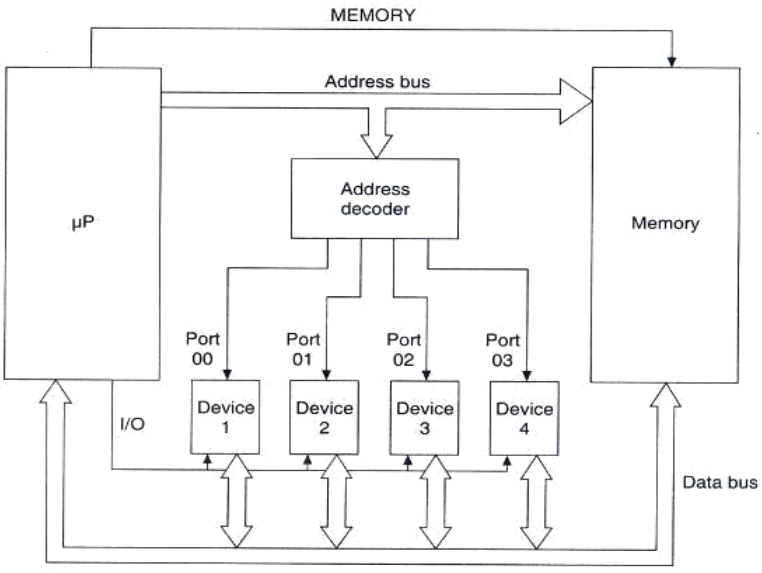
I/O Mapped I/O Interface

The I/O devices are identified by port numbers, and memory locations are identified by addresses. The memory read/write operations and I/O read/write operations are performed by different software instructions.

Whether the read/write operations are being performed on memory or I/O, or in other words whether the information on address and data lines is meant for a memory location or an I/O device—this identification is done by separate signals.

Thus, when read from an I/O device instruction is executed, the I/O signal is ON and the address on the address bus is decoded as the port number and an I/O device is selected.

I/O Mapped I/O Interface



In case of read/write from memory, the MEMORY signal is ON and a particular location of memory is selected. Because of separate memory and I/O signals, there is no confusion between device address (i.e. port number) and memory address. This is called I/O mapped I/O interface since I/O devices are treated separately from memory.

Memory Mapped I/O Interface

Consider the case except that the signals I/O and MEMORY are not present. Now when a read memory instruction is executed, there is no MEMORY signal to indicate that the address bus contains the memory location address.

If this memory location address is the same as that of a port number of an I/O device, an I/O device will also get selected together with the memory read operation being performed. Thus, there will be confusion between memory location and I/O device having the same address and port number.

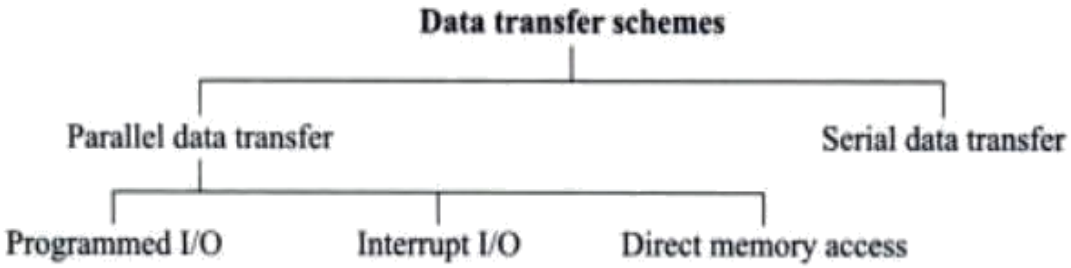
To read from device 1 (Port no. 00) memory location 00 will also get selected. However, if some memory locations is satisfied for the sake of I/O devices, this problem would not arise. It means that the I/O addresses (port numbers) and the memory addresses will not be the same. If the memory starts from address 04 onwards, then there would not be any problem.

The memory read/write instructions are quite versatile and powerful in general. If a microprocessor has more than one register apart from ACC (the 8085 has six registers other than ACC), then the memory operations can be performed using any of these registers. These instructions automatically become valid for I/O devices if connected in this fashion. The I/O read/write instructions in I/O-mapped I/O, normally require the transfer from an I/O port to accumulator. The same data is then subsequently transferred to the other register, thus wasting one instruction.

The memory read/write instructions employ various powerful addressing modes like indexed, indirect, base register addressing, etc. which is not true in the case of I/O read/write instructions. Thus, more compact and more efficient handling of I/O devices can be achieved if they are interfaced to microprocessor in this way. Since an I/O device is treated as memory location (identified by memory address), this interface is called memory mapped I/O.

7. Explain in brief about Data Transfer Schemes of 8085.

Data transfer schemes depend heavily on the environment (on-line or off-line processing), type of I/O device (capable of parallel or serial data transfer, synchronous or asynchronous) and the application. Data transfer schemes may be categorized as shown below.



7.1 Parallel Data Transfer

7.1.1 Programmed I/0

In programmed I/O, the data transfer is controlled by the user program being executed. Depending on the type of the device, data transfer may be synchronous or asynchronous. Synchronous data transfer is used when the I/0 device matches in speed with the microprocessor.

The microprocessor issues the read/write instruction addressing the *device* whenever data transfer is required. The actual data transfer takes place in one clock *cycle.*

When the I/O device speed and the microprocessor speed do not match, i.e. when the I/O device is slower than the microprocessor, asynchronous data transfer is used. In this mode of data transfer, the microprocessor checks the status of the device. If the device is not ready, the microprocessor continuously checks the status of the device till it becomes ready. The data transfer instruction is then issued by the microprocessor (Figure 2.20).

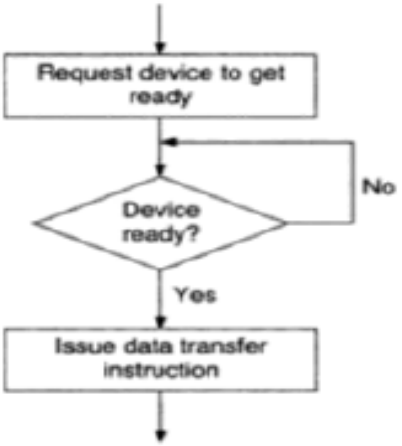


Figure 2.20 Asynchronous data transfer.

7.1.2 INTERRUPT I/O

The data transfer scheme in Figure 2.19 is quite inefficient, since the microprocessor is kept busy for the slower I/O device. The remedy to this problem is to allow the microprocessor to do its job when the device is getting ready and when the device is ready, the microprocessor can transfer the data. This can be achieved through interrupt.

When the interrupt signal is present, it should suspend the current job. The current status of the suspended job should be stored so that the microprocessor can restart the suspended job from the same point. The stack is *used* to store the status of the suspended job.

The microprocessor services the interrupt request by executing an Interrupt Service Routine. The interrupt operation is explained in Figure 2.21.

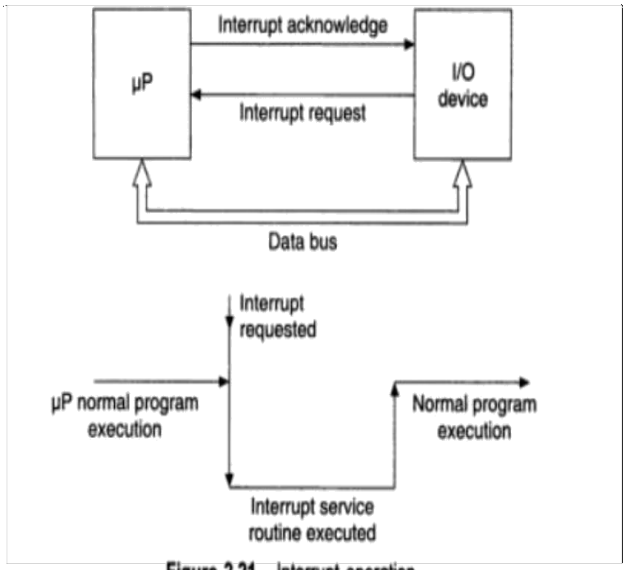


Figure 2.21 Interrupt operation.

Following is the operation sequence for interrupt operation.

(a) Normal program execution by microprocessor.
(b) The microprocessor initiates the device through a code/signal (e.g. start convert signal to initiate ADC conversion).
(c) The device when ready to send the data sends an interrupt signal on one of the interrupt pins.
(d) The microprocessor checks the validity of the interrupt request by checking whether
- The interrupt system is enabled.
- The particular interrupt is not disabled.
- Any higher priority interrupt is not pending or being processed.

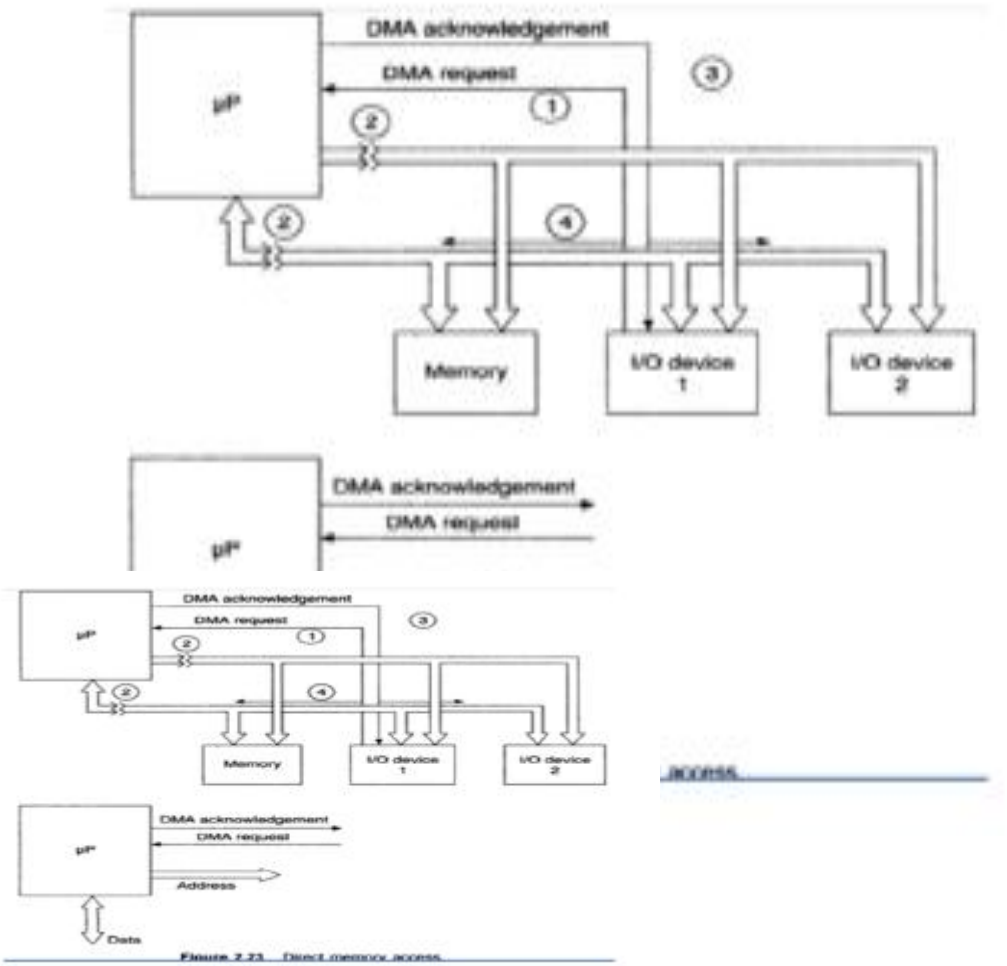
(e) If an interrupt request is valid, the microprocessor
- Completes the current instruction execution.
- Saves the Status Register and Program Counter (PC) in stack.
- Issues the interrupt acknowledgement signal.

- Determines the address of the interrupt servicing routine and stores the starting address in PC. The program thus branches to Interrupt Servicing Routine.

The last instruction of Interrupt Servicing Routine is 'Return'. When this instruction is executed, the PC and the Status Register are loaded back from stack. Thus normal program execution is resumed.

7.1.3 Direct memory access

In programmed I/O and interrupt I/O, data is transferred to the memory through the accumulator. This process is quite uneconomical for bulk data transfer, when the I/0 device matches the speed of the microprocessor. In such cases the device is allowed to transfer the data directly to memory, bypassing the microprocessor.



Following is the operation sequence in case of Direct Memory Access.

1. The microprocessor checks for DMA request signal once in each machine cycle.
2. The I/0 device sends the signal on DMA Request pin.
3. The microprocessor tristates the address, data and control buses.
4. The microprocessor sends the acknowledgement signal to the I/O device on DMA Acknowledgement pin.
5. The I/O device uses the bus system to perform the data transfer operation on memory.
6. On completion of data transfer, the I/O device withdraws the DMA request signal.
7. The microprocessor continuously checks the DMA request signal. When the signal is withdrawn, the microprocessor regains the control of buses and resumes normal operation.

7.2 Serial Data Transfer

Some devices like CRI receive and transmit data in serial mode. The data transfer between two processors will be in serial mode. The data is transferred bit by bit on a single line. This minimizes the number of interconnecting wires.

8. Draw the timing diagram for opcode fetch, Memory Read & Write, I/O Read & Write of 8085 processor. ***(Nov/Dec-09, April/May-11, May/June-12)(Nov/Dec-14)(Nov/Dec-15)(Or)***
Draw the timing diagram for I/O read & write Machine Cycles. (Nov 2016)[Apr/May 2018]

Timing Diagram is a graphical representation. It represents the execution time taken by each instruction in a graphical format. The execution time is represented in T-states.

Instruction Cycle:

The time required to execute an instruction is called instruction cycle.

Machine Cycle:

The time required to access the memory or input/output devices is called machine cycle.

T-State:

- The machine cycle and instruction cycle takes multiple clock periods.
- A portion of an operation carried out in one system clock period is called as T-state.

MACHINE CYCLES OF 8085:

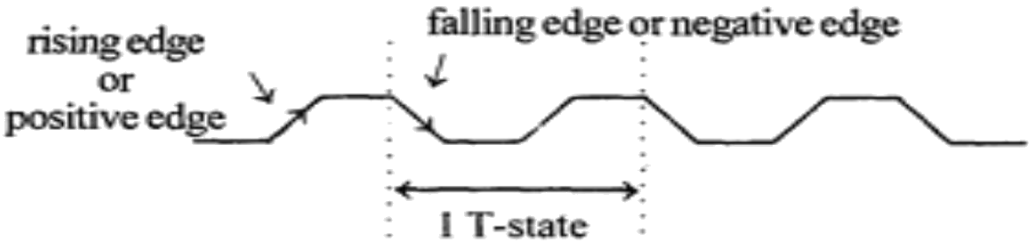
The 8085 microprocessor has 5 (seven) basic machine cycles. They are

1. Opcode fetch cycle (4T)
2. Memory read cycle (3 T)
3. Memory write cycle (3 T)
4. I/O read cycle (3 T)
5. I/O write cycle (3 T)
6. Interrupt acknowledge cycle
7. Bus idle cycle

Each instruction of the 8085 processor consists of one to five machine cycles, i.e., when the 8085 processor executes an instruction, it will execute some of the machine cycles in a specific order.

The processor takes a definite time to execute the machine cycles. The time taken by the processor to execute a machine cycle is expressed in T-states.

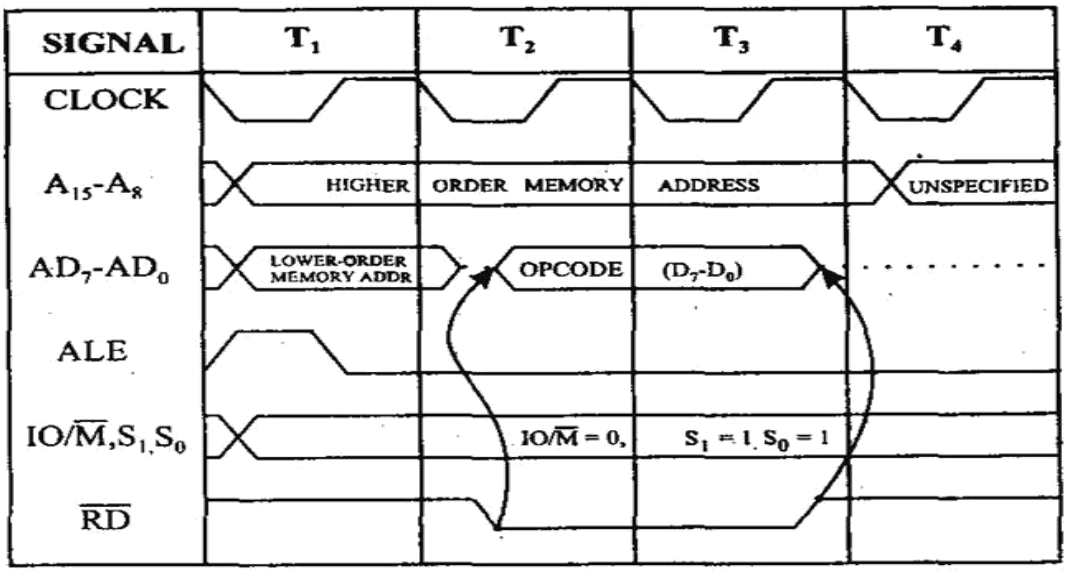
One T-state is equal to the time period of the internal clock signal of the processor. The T-state starts at the falling edge of a clock.



Time period, T = 1/f ; where f = Internal clock frequency

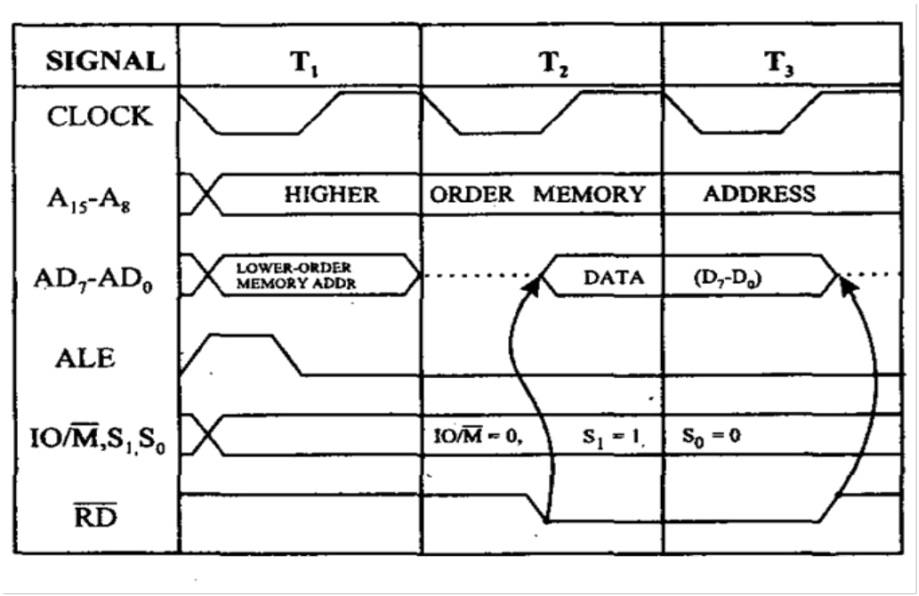
Opcode fetch machine cycle of 8085:

- Each instruction of the processor has one byte opcode.
- The opcodes are stored in memory. So, the processor executes the opcode fetch machine cycle to fetch the opcode from memory.
- Hence, every instruction starts with opcode fetch machine cycle.
- The time taken by the processor to execute the opcode fetch cycle is 4T.
- In this time, the first, 3 T-states are used for fetching the opcode from memory and the remaining T-states are used for internal operations by the processor.



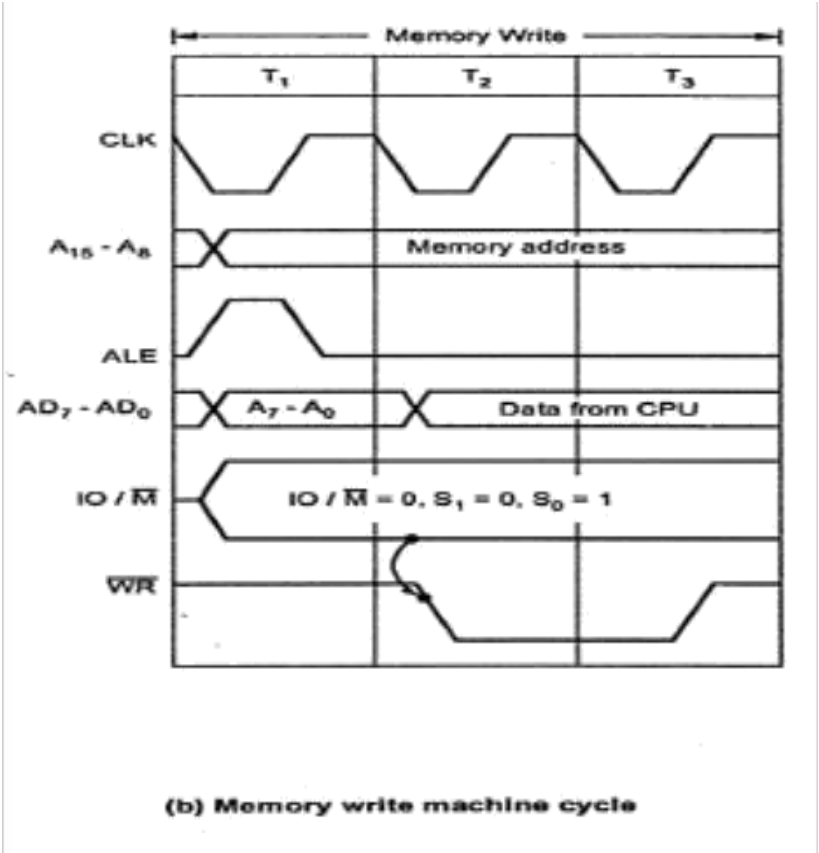
Memory read machine cycle of 8085:

- The memory read machine cycle is executed by the processor to read a data byte from memory.
- The processor takes 3T states to execute this cycle.
- The instructions which have more than one byte word size will use the machine cycle after the opcode fetch machine cycle.



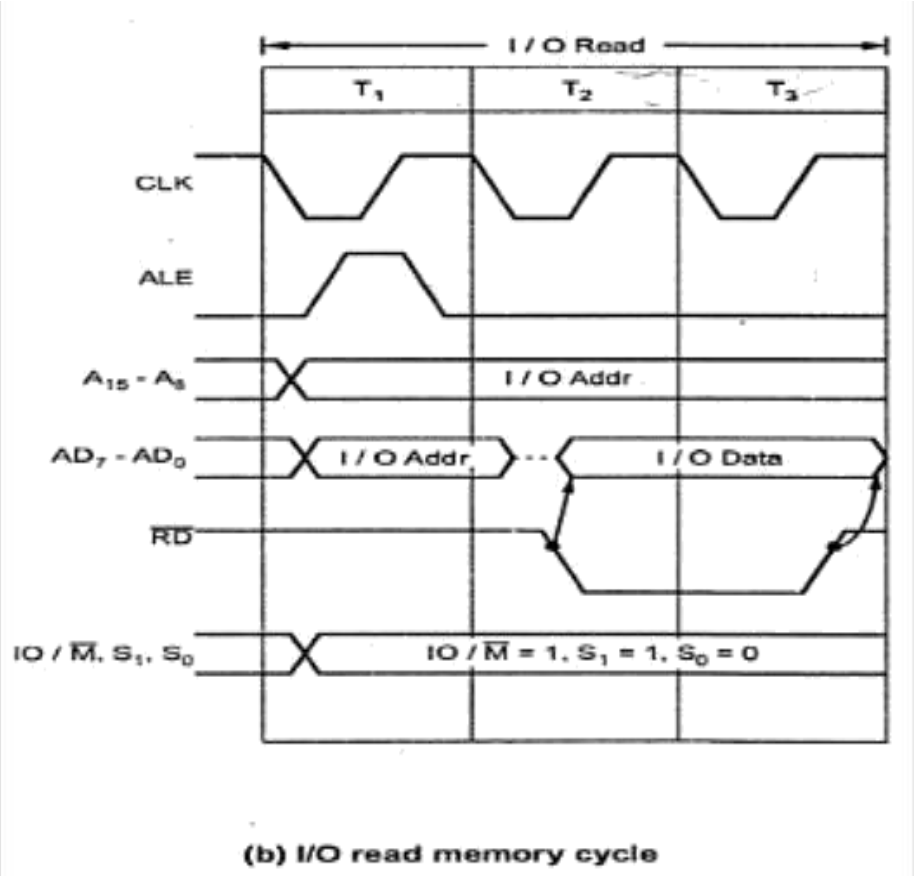
Memory write machine cycle of 8085:

The 8085 executes the memory write *cycle* to store the data into data memory or stack memory. The length of this machine cycle is 3T states (T_I - T_3). In this machine cycle, processor places the address on the address lines from the stack pointer or general purpose register pair and through the write process, stores the data into the addressed memory location



(b) Memory write machine cycle

I/O read cycle of 8085:

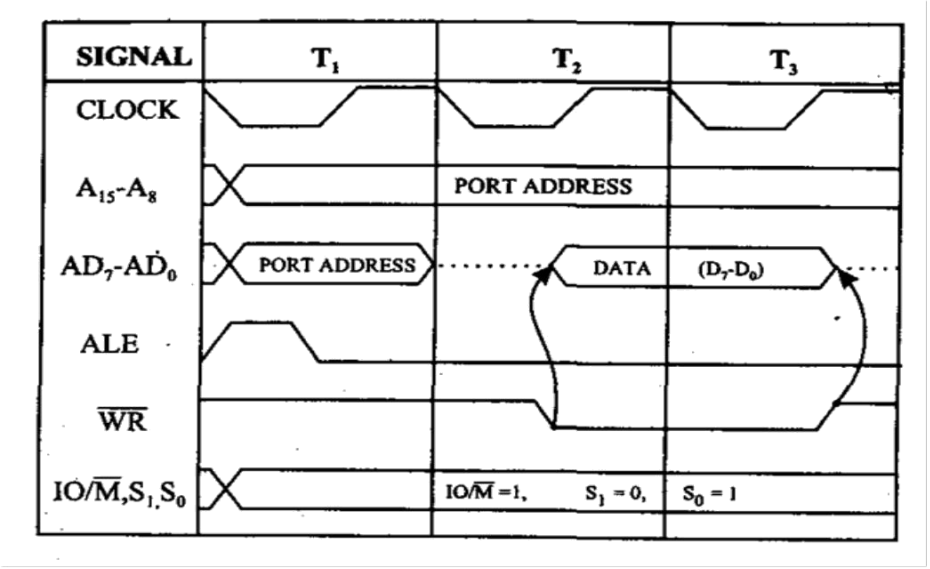


(b) I/O read memory cycle

The I/O read and I/O write machine cycles are similar to the memory read and memory write machine cycles, respectively, except that the IO/M signal is high for I/O read and I/O write machine cycles. High 10/M signal indicates that it is an I/O operation.

I/O write cycle of 8085:

- The I/O write machine cycle is executed by the processor to write a data byte in the I/O port or to a peripheral, which is I/O, mapped in the system.
- The processor takes 3T states to execute this machine cycle.



9. Explain in detail about the interrupt structure of 8085 processor. (Nov/Dec-09, May/June-12, April/May-15,Nov/Dec-15,Apr/May -17)[Nov/Dec 2017][Apr/May 2018]

Interrupt is signals send by an external device to the processor, to request the processor to perform a particular task or work. Mainly in the microprocessor based system the interrupts are used for data transfer between the peripheral and the microprocessor.

The processor will check the interrupts always at the 2nd T-state of last machine cycle. If there is any interrupt it accept the interrupt and send the INTA (active low) signal to the peripheral.

The vectored address of particular interrupt is stored in program counter. The processor executes an interrupt service routine (ISR) addressed in program counter. It returned to main program by RET instruction.

Types of Interrupts: It supports two types of interrupts.

- Hardware
- Software

Software interrupts:

- The software interrupts are program instructions. These instructions are inserted at desired locations in a program.
- The 8085 has eight software interrupts from RST 0 to RST 7. The vector address for these interrupts can be calculated as follows.
- Interrupt number * 8 = vector address
- For RST 5.5 * 8 = 40 = 28H
- Vector address for interrupt RST 5 is 0028H

The Table shows the vector addresses of all interrupts.

Interrupt	Vector address
RST 0	0000_H
RST 1	0008_H
RST 2	0010_H
RST 3	0018_H
RST 4	0020_H
RST 5	0028_H
RST 6	0030_H
RST 7	0038_H

Hardware interrupts:

- An external device initiates the hardware interrupts and placing an appropriate signal at the interrupt pin of the processor.
- If the interrupt is accepted then the processor executes an interrupt service routine.

The 8085 has five hardware interrupts

(1) TRAP (2) RST 7.5 (3) RST 6.5 (4) RST 5.5 (5) INTR

Interrupt	Vector address
RST 7.5	$003C_H$
RST 6.5	0034_H
RST 5.5	$002C_H$
TRAP	0024_H

TRAP:

- This interrupt is a non-maskable interrupt. It is unaffected by any mask or interrupt enable.
- TRAP bas the highest priority and vectored interrupt.
- TRAP interrupt is edge and level triggered. This means hat the TRAP must go high and remain high until it is acknowledged.
- In sudden power failure, it executes a ISR and send the data from main memory to backup memory.
- The signal, which overrides the TRAP, is HOLD signal. (i.e., If the processor receives HOLD and TRAP at the same time then HOLD is recognized first and then TRAP is recognized).
- There are two ways to clear TRAP interrupt.
    1. By resetting microprocessor (External signal)
    2. By giving a high TRAP ACKNOWLEDGE (Internal signal)

RST 7.5:

- The RST 7.5 interrupt is a Maskable interrupt.
- It has the second highest priority.
- It is edge sensitive. ie. Input goes to high and no need to maintain high state until it recognized.
- Maskable interrupt. It is disabled by,
    1. DI instruction
    2. System or processor reset.
    3. After reorganization of interrupt.
- Enabled by EI instruction.

RST 6.5 and 5.5:

- The RST 6.5 and RST 5.5 both are level triggered. . ie. Input goes to high and stay high until it recognized.
- Maskable interrupt. It is disabled by,
    1. DI, SIM instruction
    2. System or processor reset.
    3. After reorganization of interrupt.
- Enabled by EI instruction.
- The RST 6.5 has the third priority whereas RST 5.5 has the fourth priority.

INTR:

- INTR is a maskable interrupt. It is disabled by,
    1. DI, SIM instruction
    2. System or processor reset.
    3. After reorganization of interrupt.
- Enabled by EI instruction.
- Non- vectored interrupt. After receiving INTA (active low) signal, it has to supply the address of ISR.
- It has lowest priority.
- It is a level sensitive interrupts. ie. Input goes to high and it is necessary to maintain high state until it recognized.

The following sequence of events occurs when **INTR signal goes high.**

1. The 8085 checks the status of INTR signal during execution of each instruction.

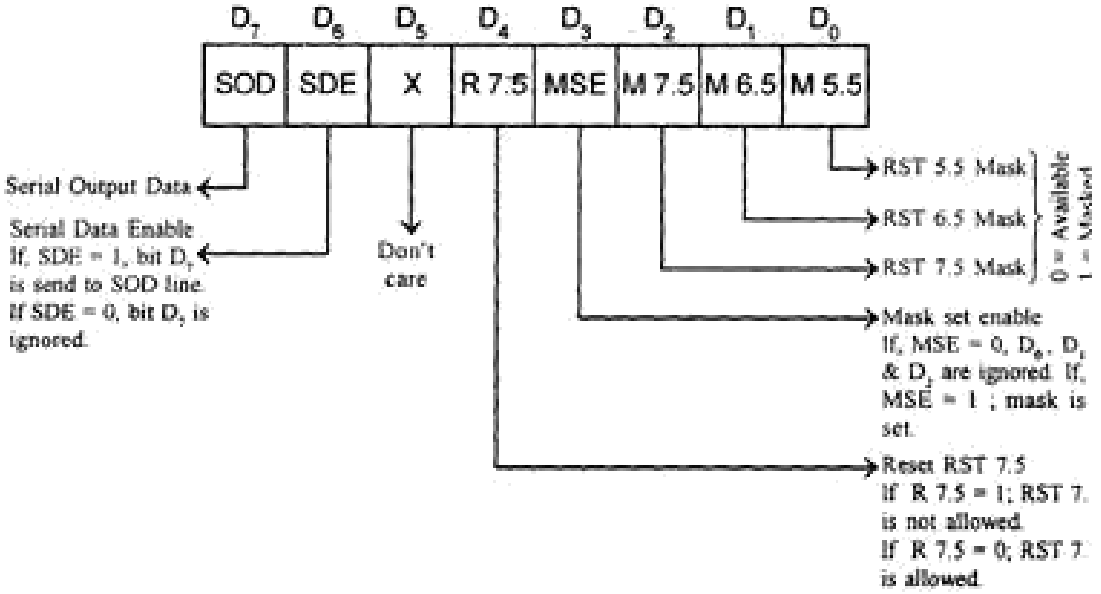
2. If INTR signal is high, then 8085 complete its current instruction and sends active low interrupt acknowledge signal, if the interrupt is enabled.

3. In response to the acknowledge signal, external logic places an instruction OPCODE on the data bus. In the case of multi byte instruction, additional interrupt acknowledge machine cycles are generated by the 8085 to transfer the additional bytes into the microprocessor.

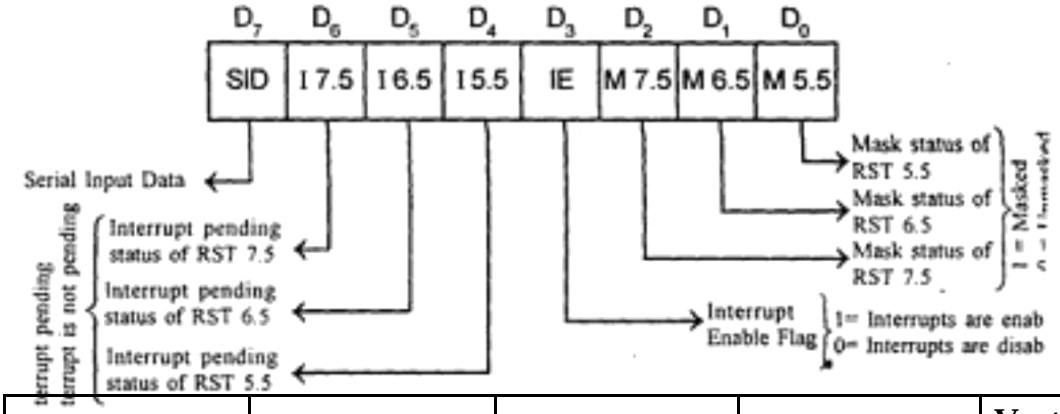
4. On receiving the instruction, the 8085 save the address of next instruction on stack and execute received instruction.

SIM and RIM for interrupts:

- The 8085 provide additional masking facility for RST 7.5, RST 6.5 and RST 5.5 using SIM instruction.
- The status of these interrupts can be read by executing RIM instruction.
- The masking or unmasking of RST 7.5, RST 6.5 and RST 5.5 interrupts can be performed by moving an 8-bit data to accumulator and then executing SIM instruction.
- The format of the 8-bit data is shown below.



- The status of pending interrupts can be read from accumulator after executing RIM instruction.
- When RIM instruction is executed an 8-bit data is loaded in accumulator, which can be interpreted as shown in fig.



Interrupt type	Trigger	Priority	Maskable	Vector Address
TRAP	Edge and Level	1st	No	0024H
RST 7.5	Edge	2nd	Yes	003CH
RST 6.5	Level	3rd	Yes	0034H
RST 5.5	Level	4th	Yes	002CH
INTR	Level	5th	Yes	-

8085 – Demo Programs

Now, let us take a look at some program demonstrations using the above instructions − Adding Two 8-bit Numbers

1) Write a program to add data at 3005H & 3006H memory location and store the result at 3007H memory location.

Problem demo −

(3005H) = 14H
(3006H) = 89H

Result −

14H + 89H = 9DH

The program code can be written like this −

```
LXI H 3005H   : "HL points 3005H"
MOV A, M      : "Getting first operand"
INX H         : "HL points 3006H"
ADD M         : "Add second operand"
INX H         : "HL points 3007H"
MOV M, A      : "Store result at 3007H"
HLT           : "Exit program"
```

Exchanging the Memory Locations

2) **Write a program to exchange the data at 5000M& 6000M memory location**.

```
LDA 5000M   : "Getting the contents at5000M location into accumulator"
MOV B, A    : "Save the contents into B register"
LDA 6000M   : "Getting the contents at 6000M location into accumulator"
STA 5000M   : "Store the contents of accumulator at address 5000M"
MOV A, B    : "Get the saved contents back into A register"
STA 6000M   : "Store the contents of accumulator at address 6000M"
```

Arrange Numbers in an Ascending Order

3) Write a program to arrange first 10 numbers from memory address 3000H in an ascending order.

```
MVI B, 09            :"Initialize counter"
START                 :"LXI H, 3000H: Initialize memory pointer"
MVI C, 09H           :"Initialize counter 2"
BACK: MOV A, M    :"Get the number"
INX H                 :"Increment memory pointer"
CMP M                  :"Compare number with next number"
JC SKIP                :"If less, don't interchange"
JZ SKIP                :"If equal, don't interchange"
MOV D, M
MOV M, A
DCX H
MOV M, D
INX H                 :"Interchange two numbers"
SKIP:DCR C            :"Decrement counter 2"
JNZ BACK              :"If not zero, repeat"
DCR B                :"Decrement counter 1"
JNZ START
HLT                   :"Terminate program execution"
```

Anna University Questions

PART-A

1.Specify the size of data, address, and memory word and memory capacity of 8085 microprocessor**.** ***(April/May-2011)***

2. How clock signals are generated in 8085 and what is the frequency of the internal clock? ***(May/June-14)***

3. List the control and status signals of 8085 microprocessor and mention its need? ***(Nov/Dec-2012)***

4. What is the need for timing diagram? ***(April/May-15)***

5. What is ALE signal and READY signal? ***(Nov/Dec-09) (Nov/Dec-14)***

6. What is the use of ALE? ***(Nov/Dec-14)***

7. What is TRAP? ***(May/June -2012)***

8. Define Flags of 8085? ***(Nov/Dec-14)***

9. Define the functions of parity flag and zero flag in 8085? ***(May/June-2012)***

10.What is the use of stack pointer?**(Nov/Dec-15)**

11.Mention the use of ALE**(Nov/Dec-15)**

12. Write an 8085 assembly program to add two digit BCD numbers in memory locations 5000H and 5001H and store the result in memory location 5002H.(Nov 2016)

13. List out the machine cycles for executing the instruction MVI A,34 H (Nov -16)

14. Why data bus is bi-directional? (Apr/May -17)

15.List out the machine cycles of 8085 microprocessor.(Apr/May -17)

16.what are the flags avaible in 8085 processor?[Nov/Dec 2017]

17.what are the interrupts available in 8085?[Nov/Dec 2017]

18. List the features of Accumulator.[Apr/May 2018]

19. Write the difference between standard I/O and Memory Mapped I/O.[Apr/May 2018]

PART-B

1. Explain with neat sketch about the hardware architecture of 8085. ***(Nov/Dec-09, April/May-11, May/June-12, Nov/Dec-12, April/May-15,Nov/Dec-15,16)*** [Nov/Dec 2017]

2. Draw the pin configuration of 8085 microprocessor and explain it in detail. ***(May/June-12) (May/June-14,17)*** (or) Draw the signal present in 8085 microprocessor and explain it in detail (or) Explain the function 8085 signals? ***(Nov/Dec-14) [Apr/May 2018]***

3 Distinguish between I/O mapped I/O and Memory mapped I/O of 8085 ***(Nov/Dec-09)*** *(P.NO :)*

4. Draw the timing diagram for opcode fetch, Memory Read & Write, I/O Read& Write of 8085 processor. ***(Nov/Dec-09, April/May-11, May/June-12,Nov/Dec-15)***

5. Explain in detail about the interrupt structure of 8085 processor. ***(Nov/Dec-09, May/June-12, April/May-15,Nov/Dec-15)[Nov/Dec 2017][Apr/May 2018]***

6.Draw and Explain the Flag register of 8085 in brief.[Nov/Dec 2017]

MAILAM ENGINEERINGCOLLEGE
MAILAM, 604304
DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

Sub.Code/Sub.Name: EE6502 MICROPROCESSORS AND MICROCONTROLLERS Year/Sem: III / V

UNIT-II - PROGRAMMING OF 8085 PROCESSOR

Instruction format and addressing modes – Assembly language format – Data transfer, data manipulation& control instructions – Programming: Loop structure with counting & Indexing – Look up table - Subroutine instructions - stack.

Updated Questions:[Nov/Dec 2017 & Apr/May 2018]

PART-A

Q.No.05	Page No: 02	(Nov/Dec 2017)
Q.No.08	Page No: 03	(Nov/Dec 2017)
Q.No.04	Page No: 02	(Apr/May 2018)
Q.No.12	Page No: 03	(Apr/May 2018)

PART-B

Q.No.01	Page No: 08	(Nov/Dec 2017)
Q.No.11	Page No: 32	(Nov/Dec 2017)
Q.No.02	Page No: 08	(Nov/Dec 2017)
Q.No.21	Page No: 40	(Nov/Dec 2017)
Q.No.01	Page No: 06	(Apr/May 2018)
Q.No.03	Page No: 09	(Apr/May 2018)

TEXT BOOKS:

1. Krishna Kant, "Microprocessor and Microcontrollers", Eastern Company Edition, Prentice Hall of India, New Delhi , 2007.
2. R.S. Gaonkar, 'Microprocessor Architecture Programming and Application', with 8085, Wiley Eastern Ltd., New Delhi, 2013.
3. Soumitra Kumar Mandal, Microprocessor & Microcontroller Architecture, Programming & Interfacing using 8085,8086,8051,McGraw Hill Edu,2013.

REFERENCES:

1. Muhammad Ali Mazidi & Janice Gilli Mazidi, R.D.Kinely 'The 8051 Micro Controller and Embedded Systems', PHI Pearson Education, 5th Indian reprint, 2003.
2. N.Senthil Kumar, M.Saravanan, S.Jeevananthan, 'Microprocessors and Microcontrollers', Oxford,2013.
3. Valder – Perez, "Microcontroller – Fundamentals and Applications with Pic," Yeesdee Publishers, Tayler & Francis, 2013.

Prepared By
1. Mrs.J.Srivandhana.,AP/MCA
2. Mrs.T.Kala.,AP/MCA

Verified By
HOD/MCA

Approved By
Principal

PART-A

1. What is an instruction?

An instruction is a binary pattern entered through an input device to command the microprocessor to perform that specific function

2. How does the microprocessor differentiate between data and instruction?

When the first machine code of an instruction is fetched and decoded in the instruction register, the microprocessor recognizes the number of bytes required to fetch the entire instruction.

For example MVI A, Data, the second byte is always considered as data. If the data byte is omitted by mistake whatever is in that memory location will be considered as data & the byte after the "data" will be treated as the next instruction.

3. List the different instruction formats with examples. (Apr/May-17)

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

4. List out the five categories of the 8085 instructions. Give examples of the instructions for each group (or) how are the 8085 instructions classified according to the functional categories? (Nov/Dec 11)[APR/MAY 2018]

- Data transfer group – MOV, MVI, LXI.
- Arithmetic group – ADD, SUB, INR.
- Logical group –ANA, XRA, CMP.
- Branch group – JMP, JNZ, CALL.
- Stack I/O and Machine control group – PUSH, POP, IN, HLT.

5. What is the use of addressing modes, mention the different types. (Or) Classify the addressing modes of 8085. (May/June 12)(Nov/Dec 13) (Nov/Dec 16)[Nov/Dec 2017]

The various formats of specifying the operands are called addressing modes, it is used to access the operands or data. The different types are as follows

- Immediate addressing
- Register addressing
- Direct addressing
- Indirect addressing
- Implicit addressing

6. How many operations are there in the instruction set of 8085 microprocessor?

There are 74 operations in the 8085 microprocessor.

7. Explain LDA, STA and DAA instructions

LDA copies the data byte into accumulator from the memory location specified by the 16-bit address. STA copies the data byte from the accumulator in the memory location specified by 16- bit address. DAA changes the contents of the accumulator from binary to 4-bit BCD digits.

8. Explain the difference between a JMP instruction and CALL instruction.[Nov/Dec 2017]

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.

9. Explain the purpose of the I/O instructions IN and OUT.

The IN instruction is used to move data from an I/O port into the accumulator. The OUT instruction is used to move data from the accumulator to an I/O port. The IN & OUT instructions are used only on microprocessor, which use a separate address space for interfacing.

10. What is the difference between the shift and rotate instructions?

A rotate instruction is a closed loop instruction. That is, the data moved out at one end is put back in at the other end. The shift instruction loses the data that is moved out of the last bit locations.

11. List the four instructions which control the interrupt structure of the 8085 microprocessor.

- DI (Disable Interrupts)
- EI (Enable Interrupts)
- RIM (Read Interrupt Masks)
- SIM (Set Interrupt Masks)

12. Define stack and explain stack related instructions. (Nov/Dec 13) [APR/MAY 2018]

The stack is a group of memory locations in the R/W memory that is used for the temporary storage of binary information during the execution of the program. The stack related instructions are PUSH & POP

13. Why do we use XRA A INSTRUCTION?

The XRA A instruction is used to clear the contents of the Accumulator and store the value 00H.

14. Compare CALL and PUSH instructions. (Nov/Dec 11)

CALL	PUSH
When CALL is executed the microprocessor automatically stores the 16-bit address of the instruction next to CALL on the stack.	The programmer uses the instruction PUSH to save the contents of the register pair on the stack.
When CALL is executed the stack pointer is decremented by two.	When PUSH is executed the stack pointer register is decremented by two.

15. Compare RET and POP (Nov/Dec 11)

RET	POP
RET transfers the contents of the top two locations of the stack to the PC	POP transfer the contents of the top two locations of the stack to the specified register pair.
When RET is executed the SP is incremented by two	When POP is executedthe SP is incremented by two
Has 8 conditional RETURN instructions	No conditional POP instructions

16. State the functions of given 8085 instructions: JP, JPE, JPO, JNZ (Apr/May 11)

JP Jump if plus (SF = 0)

JPE Jump if parity is even (PF = 1)

JPO Jump if parity is odd (PF = 0)

JNZ Jump if non zero (ZF = 0)

17. Why do we need look – up table? (Nov/Dec 11) (Nov/Dec 14)

Look up table is a memory location placed in the register of a microprocessor. It is used to transform the input data into a more desirable output format.

18. Where is the READY signal used? (Nov / Dec-09)

READY is an input signal to the processor, used by the memory or the input/output devices to get extra time for data transfer or to introduce wait states in the bus cycles.

19. What are HOLD and HLDA?

They are acknowledging signals, used for the Direct Memory Access (DMA) type of data transfer.

20. What is Subroutine?

Subroutine is a group of instructions written separately from the main program. Normally, if certain instructions have to be executed more than one time in the main program, those instructions can form one subroutine.

21. What are the instructions used for subroutine? (Nov/Dec 13) (April/May-15)

CALL (call a subroutine) and RET (Return from subroutine to main program).

22. What is a recursive procedure?

A recursive procedure is a procedure, which calls itself. Recursive procedures are used to work with complex data structures called trees. if the procedure is called with N=3, then the N is decremented by 1 after each procedure CALL and the procedure is called until N=0.

23. How to access subroutine within the main program procedure?

i) Accessed by CALL & RET instruction
ii) Machine code of instruction is put only once in the memory
iii) With procedures less memory is required
iv) Parameters can be passed in registers, memory location or stack

24. How is PUSH B instruction executed? Find the status after the execution. (Apr/May 11)

During PUSH and POP operation, stack pointer register gives the address of memory where the information is to be stored or to be read. The stack pointer's contents are automatically manipulated to point to stack top. The memory location currently pointed by stack pointer is called **top of stack**

25. What is the use of branching instructions? Give example. (May/June 12)

These instructions allow the 8085 to change the sequence of the program, either unconditionally or under certain test conditions. These instructions include branch instructions, subroutine call and return instructions and restart instructions.

26. What is the different machine control instructions used in 8085 microprocessor? (Nov/Dec 13)

1. EI enable interrupts.
2. DI disable interrupts.
3. NOP No operation is performed.
4. HLT This instruction halts the processor.
5. SIM This instruction masks the interrupts as desired. It also sends out serial data through the SOD pin. For this instruction command byte must be loaded in the accumulator.
6. RIM Read interrupt mask. It is used to check whether the interrupt is masked or not. It is also used to read data from SID line.

27. What is the similarity and difference between subtract and compare instructions? (May/June 14)

Similarity: Both the subtraction and comparison are performed by subtracting two data in the ALU and the flags are altered depending upon the result,

Difference: After the subtract operation, the result is stored in the destination register/memory, but after the compare operation the result is discarded**.**

28. What is NOP? State its importance. (May/June 14)

The NOP is a dummy instruction; it neither achieves any result nor affects any CPU register. This is used for producing software delay and reserve memory spaces for future software modifications.

29. What is PSW?

PSW- Program Status Word. The flag register and accumulator together is called PSW. Flag register is a low order register. The accumulator is a high order register.

30. What is the function of rotate instructions? (Nov/Dec-14)

- A rotate instruction is a closed loop instruction.
- That is, the data moved out at one end is put back in at the other end.
- Examples : RLC, RRC,RAR, RAL

31. List any two data manipulation instructions? (April/May-15)

The data manipulation instruction in microprocessor is usually divided into three basic types:

- Arithmetic instructions
- Logical and Bit manipulation instructions
- Shift instructions.

32. List the different instruction format in 8085.(Apr/May-17)

The 8085A instruction set consists of one, two and three byte instructions.

33. Explain the functioning of CMP instruction (Nov/Dec-2015)

CMP r

This instruction subtracts the contents of the specified register from contents of the accumulator and sets the condition flags as a result of the subtraction. It sets zero flag if A = r and sets any flag if A < r. Operation : A – r

CMP M

This instruction subtracts the contents of the memory location specified by **HL** register pair from the contents of the accumulator and sets the condition flags as a result of subtraction. It sets zero flag if A = M and sets carry flag if A < M. The HL register pair is used as a memory pointer.

Operation: A – M

34.What is the function of CALL instruction. ***(Nov/Dec 16)***

A CALL instruction leaves information on the stack so that the original program execution sequence can be resumed.

35. Write an 8085 program to swap higher and lower nibble of the contents of accumulator.(Apr/May-17)

```
MOV A,#DATA
SWAP A
```

36.What do you mean by Looping, Counting and Indexing?

Looping: In this tech the program is instructed to execute certain set of instructions repeatedly to execute a particular task number of times.

Counting: This tech allows programmer to count how many times the ins of instruction are executed.

Indexing: This tech allows programmer to point or refer the data stored in sequential memory location one by one.

37. What is the significance of 'XCHG' and 'SPHL' instructions?

XCHG-Exchange the contents of HL register pair with DE register pair ie the contents of register H are exchanged with the contents of register D and the contents of register L are exchanged with the contents of register E SPHL-store the contents of HL register pair to the stack pointer. The contents of H register provide the higher order address and the contents of L register provide the low order address. The contents of H and L registers are not altered.

38.What is meant by immediate addressing mode?

In an immediate addressing mode 8 or 16 bit data can be specified as a part of instruction. _I' indicates the immediate addressing mode. Eg; MVI A, 20 H

39. What is meant by register addressing mode?

The register addressing mode specifies the source operand, destination operand, or both to be contained in an 8085 registers. This results in faster execution, since it is not necessary to access memory locations for operand. Eg : MOV A, B

40. What is meant by direct addressing mode?

The direct addressing mode specifies the 16 bit address of the operand within the instruction itself. The 2nd and 3rd bytes of the instruction contain this 16 bit address.

Eg: LDA 2050 H

41. If flags are individual flip flops, can they be observed on an oscilloscope?

No, they cannot be observed on an oscilloscope. The flag register is internal to the microprocessor. However they can be tested through conditional branch instruction and they can be examined by storing them on the stack memory.

PART-B QUESTIONS AND ANSWERS

1. Describe the instruction format and addressing modes of 8085 microprocessor. *(Apr-12)(May/June 12)(Nov/Dec 14) (April/May 15)(Nov/Dec 16) (April/May 17)[Nov/Dec 2017] [APR/MAY 2018]*

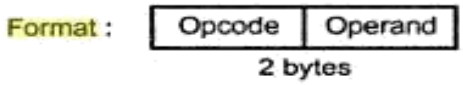
Instruction Formats

The 8085A instruction set consists of one, two and three byte instructions. The first byte is always the opcode; in two-byte instructions the second byte is usually data; in three byte instructions the last two bytes present address or 16-bit data.

1. One byte instruction

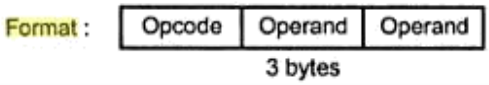
For Example: MOV A, B whose opcode is **78H** which is one byte. This instruction copies the contents of B register in A register.

2. Two byte instruction



For Example: MVI B, 02H. The opcode for this instruction is 06H and is always followed by a byte data (02H in this case). This instruction is a two byte instruction which copies immediate data into **B** register.

3. Three byte instruction



For Example: JMP 6200H. The opcode for this instruction is C3H and is always followed by 16 bit address (6200H in this case). This instruction is a three byte instruction which loads 16 bit address into program counter.

Addressing modes of 8085 processor.

Every instruction of a program has to operate on a data. The method of specifying the data to be operated by the instruction is called Addressing. The 8085 has the following 5 different types of addressing.

1. Immediate Addressing
2. Direct Addressing
3. Register Addressing
4. Register Indirect Addressing

5. Implied Addressing

1. Immediate Addressing:

- In immediate addressing mode, the data is specified in the instruction itself. The data will be a part of the program instruction.
- EX. MVI B, 3EH - Move the data 3EH given in the instruction to B register; LXI SP, 2700H.

2. Direct Addressing:

- In direct addressing mode, the address of the data is specified in the instruction. The data will be in memory. In this addressing mode, the program instructions and data can be stored in different memory.
- EX. LDA 1050H - Load the data available in memory location 1050H in to accumulator; SHLD 3000H

3. Register Addressing:

- In register addressing mode, the instruction specifies the name of the register in which the data is available.
- EX. MOV A, B - Move the content of B register to A register; SPHL; ADD C.

4. Register Indirect Addressing:

- In register indirect addressing mode, the instruction specifies the name of the register in which the address of the data is available. Here the data will be in memory and the address will be in the register pair.
- EX. MOV A, M - The memory data addressed by H L pair is moved to A register. LDAX B.

5. Implied Addressing:

- In Implied addressing mode, the instruction itself specifies the data to be operated.
- EX. CMA - Complement the content of accumulator; RAL

2. How Instructions of 8085 can be classified? [APR/MAY 2018]

The instructions provided by the 8085 can be categorized into five different groups based on the nature of function of the instructions.

- Data transfer operations
- Arithmetic operations
- Logical operations
- Branch operations and
- Stack, Input/Output and Machine control operations

2.1 Data Transfer Operations

The data transfer instructions load given data into register, copy data from register to register, copy data from register to memory location, and vice versa. In other words we can say that

data transfer i*n*structions copy data from source to destination. Source can be data or contents of register or contents of memory location whereas destination can be register or memory location. These instructions do not affect the flag register of the processor.

2.2 Arithmetic Operations

The arithmetic instructions provided by 8085 perform addition, subtraction, increment and decrement operations.

Addition: Any 8-bit number, or the contents of a register, or the contents of a memory location can be added to the contents of the accumulator and the resulted sum is stored in the accumulator. The resulted carry bit is stored in the carry flag. In 8085, no two other registers can be added directly, i.e. the contents of 8 and C registers cannot be added directly. To add two 16-bit numbers the 8085 provides DAD instruction. It adds the data within the register pair to the contents of the HL register pair and resulted sum is stored in the HL register pair.

Subtraction: Any 8-bit number, or the contents of a register, or the contents of a memory location can be subtracted from the contents of the accumulator and the result is stored in the accumulator. The resulted borrow bit is stored in the carry flag. In 8085, no two other registers can be added directly.

Increment/Decrement: The 8085 has the increment and decrement instructions to increment and decrement the contents of any register, memory location or register pair by 1.

2.3 Logical Operations

The logical instructions provided by 8085 perform logical, rotate, compare and complement operations.

Logical: Using logical instructions, any 8-bit number, or the contents of a register, or of a memory location can be logically ANDed, ORed, or Exclusive-ORed with the contents of the accumulator and the result is stored in the accumulator. The result also affects the flags according to definition of flags. For example, the zero result sets the zero flag.

Rotate: These instructions allow shifting of each bit in the accumulator either left or right by I bit position.

Compare: Any 8-bit number, or the contents of a register, or the contents of a memory location can be compared for equality, greater than, or less than, with the contents of the accumulator.

Complement: The result of accumulator can be complemented with this instruction. It replaces all 0s by 1s and all 1s by 0s.

2.4 Branching Operations

These instructions allow the 8085 to change the sequence of the program, either unconditionally or under certain test conditions. These instructions include branch instructions, subroutine call and return instructions and restart instructions.

2.5 Stack, Input/output and Machine Control Operations

These instructions control the stack operations, input/output operations and machine operations. The stack instructions allow the transfer of data from register pair to stack memory and from stack memory to the register pair. The input/output instruction allows the transfer of 8-bit data to input/output port. On the other hand machine instructions control the machine operations such as interrupt, halt, or do nothing.

Notations used in the explanation of instructions. These are:

Notation	**Meaning**
M	Memory location pointed by **HL** register pair
R	8-bit register
rp	16-bit register pair
rs	Source register
rd	Destination register
addr	16-bit address/8-bit address

3. With suitable example, discuss about instruction sets of 8085 microprocessor. (Nov/Dec 13)(Apr/May 11) [NOV/DEC 2017]

3.1 Data Transfer Group (or) Explain data transfer instruction in detail.

1. MVI r, data (8)

This instruction directly loads a specified register with an 8-bit data given within the instruction. Operation : r ←8-bit data (byte)

Example

MVI B, 60H: This instruction will load 60H directly into the B register.

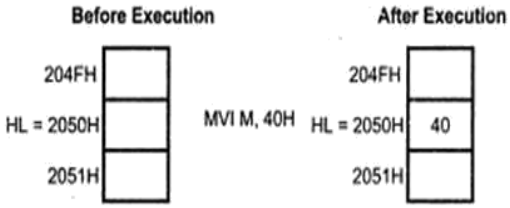
2. MVI M, data (8)

This instruction directly loads an 8-bit data given within the instruction into a memory location. Operation : M←byte or (HL) ←byte

Example:

H =20H and L =50H

MVI M, 40H: This instruction will load 40H into memory whose address is 2050H.



3. MOV rd, rs

This instruction copies data from the source register into destination register. The contents of the source register remain unchanged after execution of the instruction. Operation : rd ←rs

Example: A =20H

MOV B, A: This instruction will copy the contents of register A (20H) into register B.

4. MOV M, rs

This instruction copies data from the source register into memory location (HL register pair).

Operation: (HL) ← rs

Example: if HL = 2050H, B=30H.

MOV M, B: This instruction will copy the contents of B register (30H) into the memory location whose address is specified by HL (2050H).

5. MOV rd, M [NOV/DEC 2017]

This instruction copies data from memory location into destination register. The contents of the memory location remain unchanged.

Operation : rd← (HL)

Example : HL=2050H, contents at 2050H memory location= 40H

MOV C, M: This instruction will copy the contents of memory location pointed by HL register pair (40H) into the C register.

6. LXI rp, data (16)

This instruction loads immediate 16 bit data specified within the instruction into register pair or stack pointer. The rp is 16-bit register pair such as BC, DE, HL or 16-bit stack pointer.

Operation: rp←data (16)

Example

LXI B, 1020H: This instruction will load 10H into B register and 20H into C register.

7. STA addr

This instruction stores the contents of A register into the memory location whose address is directly specified within the instruction.

Operation : (addr) ← A

Example : A = 50H

STA 2000H : This instruction will, store the contents of A register (50H) to memory location 2000H.

8. LDA addr [NOV/DEC 2017]

This instruction copies the contents of the memory location into the accumulator. The contents of the memory location remain unchanged.

Operation: A ← (addr)

Example : (2000H) = 30H

LDA 2000H: This instruction will copy the contents of memorylocation 2000H i.e. data 30H into the A register.

9. SHLD addr [NOV/DEC 2017]

This instruction stores the contents of L register in the memory location given within the instruction and contents of H register at address next to it. This instruction is used to store the contents of H and L. registers directly into the memory.

Operation : (addr) ←L and (addr + I) ← H

Example **H=**30H, L= 60H

SHLD 2500H: This instruction will copy the contents of L registerat address 2500H and the contents of H register at address 2501H.

10. LHLD addr

This instruction copies the contents of the memory location given within the instruction into the L register and the contents of the next memory location into the H register.

Operation: L ← (addr), H ← (addr + 1)

Example**:** (2500H) = 30H, (2501H) = 60H

LHLD2500H; This instruction will copy the contents of memory location 2500H i.e. data 30H into the L register and the contents at memory location 2501H i.e. data 60H into the H register.

11. STAX rp

This instruction copies the contents of accumulator into the memory location.

Operation: (rp) ←A

Example**:** BC = 1020H, A = 50H

STAX B; This instruction will copy the contents of A register (50H) to the memory location specified by BC register pair (1020H).

12. LDAX rp

This instruction copies the contents of memory location into the accumulator.

Operation: A← (rp)

Example : DE= 2030H, (2030H) =80H

LDAX D: This instruction will copy the contents of memory location specified by DE register pair (80H) into the accumulator.

13. XCHG [NOV/DEC 2017]

This instruction exchanges the contents of the register H with thatof D and of L with that of E.

Operation H ↔D and L ↔ E

Example DE = 2040H, HL= 7080H
XCHG This instruction will load the data into registers as follows H =20H, L=40H, D=70H and E= 80H

3.2 Arithmetic Group (or) Explain the arithmetic Instructions of 8085 in detail.

1. ADD r
This instruction adds the contents of the specified register to the contents of accumulator and stores result in the accumulator.
Operation: A ←A + r

Example: A=20H, C =30H.
ADD C; This instruction will add the contents of C register, i.e. data 3011 to the contents of accumulator, i.e. data 20H and it will store the result 50H in the accumulator.

2. ADD M
This instruction adds the contents of the memory location pointed by HL register pair to the contents of accumulator and stores result in the accumulator. This instruction affects all flags.
Operation A←A+M

Example: A = 20H, HL =2050H, (2050H) = 10H
ADD M; This instruction will add the contents of memory location pointed by HL register pair, 2050H i.e. data 10H to the contents of accumulator i.e. data 20H and it will store the result, 30H in the accumulator.

3. ADI data (8)
This instruction adds the 8 bit data given within the instruction to the contents of accumulator and stores the result in the accumulator.
Operation: A ← A + data (8)

Example**:** A = 50H
ADI 70H ; This instruction will add 70H to the contents of the accumulator (50H) and it will store the result in the accumulator (C0H).

4. ADC r
This instruction adds the contents of specified register to the contents of accumulator with carry.
Operation A←A+r+CY

Example : Carry flag = 1, A = 50H, C= 20H
ADC C ; This instruction will add the contents of C (20H)register to the contents of accumulator (50H) with carry (1) and it will store result, 71H (50H+20H+1=71H) in the accumulator.

5. ADC M

This instruction adds the contents of memory location pointed by HL register pair to the contents of accumulator with carry and stores the result in the accumulator.

Operation: A ← A + M + CY

Example : Carry flag = 1, HL = 2050H, A = 20H, (2050H) =30H.

ADC M ; *This* instruction will add the contents of memory location pointed by HL register pair, 2050H, i .e. data 30H to the contents of accumulator, i.e. data 20H with carry flag (1) . It will store the result (30+20+1=51H) in the accumulator.

6. ACI data (8)

This instruction adds 8 bit data given within the instruction to the contents of accumulator with carry and stores result in the accumulator.

Operation : A← A+ data (8) + CY

Example: A = 30H, Carry flag = 1

ACI 20H ; This instruction will add 20H to the contents of accumulator, i.e. data 30H with carry (1) and stores the result, 51H (30 + 20 + 1 = 51H) in the accumulator.

7. DAD rp [NOV/DEC 2017]

This instruction adds the contents of the specified .register pair to the contents of the **HL** register pair and stores the result in the HL register pair.

Operation: HL←HL + rp

Example: DE= 1020H, HL=2050H

DAD D: This instruction will add the contents of DE register pair, 1020H to the contents of HL register pair, 2050H. It will store the result, 3070H in the HL register pair.

8. SUB r

This instruction subtracts the contents of the specified register from the contents of the accumulator and stores the result in the accumulator.

Operation A←A-r

Example A=50H, B=30H

SUB B; This instruction will subtract the contents of D register (30H) from the contents of accumulator (50H) and stores the result (20H) in the accumulator.

9. SUB M

This instruction subtracts the contents of the memory location pointed by HL register pair from the contents of accumulator and stores the result in the accumulator.

Operation: A ←A-M

Example: HL=1020H, A= 50H, (1020H) = 10H

SUB M ; This instruction will subtract the contents of memory location pointed by HL register pair, 1020H, i.e. data 10H from the contents accumulator, i.e. data 50H and stores the result (40H) in accumulator.

10. SUI data (8)

This instruction subtracts an 8 bit data given within the instruction from the contents of the accumulator and stores the result in the accumulator.

Operation: A ← A - data (8)

Example**:** A = 40H,

SUI 20H; This instruction will subtract 20H from the contents of accumulator (40H). It will store the result (20H) in the accumulator.

11. SBB r

This instruction subtracts the specified register contents and borrows flag from the accumulator contents. Operation: A←A-r-CY

Example : Carry flag= 1, C= 20H, A= 40H

SBB C ; This instruction will subtract the contents of C register (20H) and carry flag (1) from the contents of accumulator(40H) .It will store the result(40H - 20H - 1 =1FH) in the accumulator.

12. SBB M

This instruction subtracts the contents of memory location from the contents of accumulator and borrows flag and stores the result in the accumulator.

Operation: A← A-M-CY

Example: Carry flag = 1, HL= 2050H, A = 50H, (2050H) = 10H.

SBB M ;This instruction will subtract the contents of memory location; pointed by HL register pair, 2050H, i.e. data 10H and borrow (Carry flag = 1) from the contents of accumulator (50H) and stores the result 3FH in the accumulator (50 -10-1-3F).

13. SBI data (8)

This instruction subtracts 8 bit data given within the instruction and borrows' flag from the contents of accumulator and stores the result in the accumulator.

Operation: A ←A-data (8)-CY

Example: Carry flag = 1, A= 50H

SBI 20H; This instruction will subtract 2011 and the carry flag(1) from the contents of the accumulator (50H). It will store the result (50H – 20H - 1 = 2FH) in the accumulator.

14. DAA [NOV/DEC 2017]

This instruction adjusts accumulator to packed BCD (Binary Coded Decimal) after adding two BCD numbers. Instruction works as follows:
1. If the value of the low - order four bits (D_3-D_0) in the accumulator is greater than 9 or if auxiliary carry flag is set, the instruction adds 6 (06) to the low-order four bits.
2. If the value of the high-order four bits (D_7-D_4) in the accumulator is greater than 9 or if carry flag is set, the instruction adds 6(60) to the high-order four bits.

15. INR r

This instruction increments the contents of specified register by 1.The result is stored in the same register. Operation : r←r+ 1

Example: B=10H
INR B; This instruction will increment the contents of B register (10H) by one and stores the result (10 + 1 = 11H) in the *same* i.e. B register.

16. INR M

This instruction increment the contents of memory location pointed by HL register pair by 1. The result is stored at the same memory location.
Operation : M ← M + 1

Example: HL= 2050H, (2050H) = 30H
INR M ;This instruction will increment the contents of memory location pointed by HL register pair, 2050H, i.e. data 30H by one. It will store the result (30 + 1 = 11H) at the same place.

17. INX rp

This instruction increments the contents of register pair by one. The result is stored in the same register pair. Operation rp←rp + 1
Example: HL= 10FFH
INX H; This instruction will increment the contents of HL register pair (10FFH) by one. It will store the result (10FF+ 1 = 1100H) in the same i.e. HL register pair.

18. DCR r

This instruction decrements the contents of the specified register by one. It stores the result in the same register.
Operation: r ← r - 1
Example: E = 20H
DCR E: This instruction will decrement the contents of E register (20H) by one. It will store the result (20 - 1 =1FH) in the same, i.e. E register.

19. DCR M

This instruction decrements the contents of memory location pointed by HL register pair by 1. The result is stored in the same memory location.
Operation: M←M– 1

Example: HL = 2050H, (2050H) = 21H
DCR M ; This instruction will decrement the contents of memory location pointed by HL register pair, 2050H, i.e. data 21H by one. It will store the result (21 - 1 = 20H) in the same memory location.

20. DCX r*p*
This instruction decrements the contents of register pair by one. The result is stored in the same register pair. Only higher order register is to be specified within the instruction.
Operation rp←rp–1

Example: DE= 1020H
DCX D; This instruction will decrement the contents of DE register pair (1020H) by one and store the result (1020 - 1 = 101FH) in the same, DE register pair.

3.3 Logic Group (or) Explain logical instructions used in 8085 with examples. *(Nov/Dec 16)*

1. ANA r : This instruction logically ANDs the contents of the specified register with the contents of accumulator and stores the result in the accumulator.
Operation :A←A^r

Example :A= 10101010 (AAH), B= 00001111 (0FH)
ANA B; This instruction will logically AND the contents of B register with the contents of accumulator. It will store the result (0AH) in the accumulator.

2. ANA M
This instruction logically ANDs the contents of memory location pointed by HL register pair with the contents of accumulator. The result is stored in the accumulator.
Operation: A← A^M

Example : A = 01010101 = (55H), HL = 2050H (2050H) → 10110011 =(B3H)
ANA M ; This instruction will logically AND the contents of memory location pointed by HL register pair (B3H) with the contents of accumulator (55H). It will store the result 11H in the accumulator

3. ANI data
This instruction logically ANDs the 8 bit data given in the instruction with the contents of the accumulator and stores the result in the accumulator.
Operation: A← A ^ data (8)

4. XRA r

This instruction logically XORs the contents of the specified register with the contents of accumulator and stores the result in the accumulator.
Operation: A←A ⊕ r
Example : A= 1010 1010 (AAH), C= 0010 1101 (20H)
XRA C; This instruction will logically XOR the contents of C register with the contents of accumulator. It will store the result (87H) in the accumulator.

5. XRA M
This instruction logically XORs the contents of memory location pointed by HL register pair with the contents of accumulator.
Operation: A ←A ⊕ M

Example: A=0101 0101= (55H), HL= 2050H (2050H) →1011 0011= (B3H)

XRA M ; This instruction will logically XOR the contents of memory location pointed by HL register pair (2050H)i.e. data B3H with the contents of accmmulator (55H).It will store the result (E6H) in the accumulator.

6. XRI data
This instruction logically XORs the 8 bit data given in the instruction with the contents of the accumulator. Operation: A ← A ⊕ data

Example : A=10110011= (B3H)
XRI 39H ; This instruction will logically XOR the contents of accumulator (B3H) with 39H. It will store the result (8AH) in the accumulator.

7. ORA r
This instruction logically ORs the contents of specified register with the contents of accumulator and stores the result in the accumulator.
Operation: A← A v r

Example: A = 10101010 (AAH), B = 0001 0010 (12H)
ORA B; This instruction will logically OR the contents of B register with the contents of accumulator. It will store the result (BAH) in the accumulator.

8. ORA M
This instruction logically ORs the contents of memory location pointed by HL register pair with the contents of accumulator.
Operation: A← AvM

Example: A =0101 0101 = (55H) HL= 2050H (2050H) →10110011= (B3H)

ORA M; This instruction will logically OR the contents of memory location pointed by HL register pair (B3H) with the contents of accumulator (55H). It will store the result (F7H) in the accumulator.

9. ORI data

This instruction logically ORs the 8 bit data given in the instruction with the contents of the accumulator and stores the result in the accumulator.

Operation: A v data (8)

Example: A= 1011 0011 =
(B3H)

ORI 08H; This instruction will logically OR the contents of accumulator (B3H) with 08H. It will store the result (BBH) in the accumulator.

10. CMP r

This instruction subtracts the contents of the specified register from contents of the accumulator and sets the condition flags as a result of the subtraction. It sets zero flag if A = r and sets any flag if A < r.

Operation :A – r

Example: A 1011 1000 (B8H) and D=1011 1001 (B9H)

CMP D; This instruction will compare the contents of **D** register with the contents of accumulator. Here A<**D** so carry flag will set after the execution of the instruction.

11. CMP M

This instruction subtracts the contents of the memory location specified by **HL** register pair from the contents of the accumulator and sets the condition flags as a result of subtraction. It sets zero flag if A = M and sets carry flag if A < M. The HL register pair is used as a memory pointer.

Operation: A – M

Example: A = 1011 1000 (B8H), HL=2050Hand (2050H)=10111000 (B8H)

CMP M; This instruction will compare the contents of memory location (B8H) and the contents of accumulator. Here A=M so zero flag will set after the execution of the instruction.

12. CPI data

This instruction subtracts the 8 bit data given in the instruction from the contents of the accumulator and sets the condition flags as a result of subtraction. It sets zero flag if A = data and sets carry flag if A < data.

Operation: A - data (8)

Example: A = 1011 1010 (BAH)

CPI 30H ; This instruction will compare 30H with the contents of accumulator (BAH). Here A >***data*** so zero and carry both flags will reset after the execution of the instruction.

13. STC

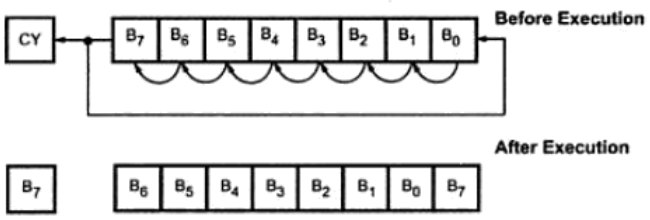
This instruction sets carry flag = 1
OperationCY←**1**
Example IF carry flag=0
STC : This instruction will set the carry flag=1

14. CMC
This instruction complements the carry flag.
Operation: CY←
Example: Carry flag = 1
CMC; This instruction will complement the carry flag i.e. carry flag = 0

15. CMA
This instruction complements each bit of the accumulator.
Operation: A ←
Example: A = 1000 1000 = 88H
CMA; This instruction will complement each bit of accumulator A = 0111 0111 = 77H

3.4 Rotate Group(or) Explain rotate group instruction

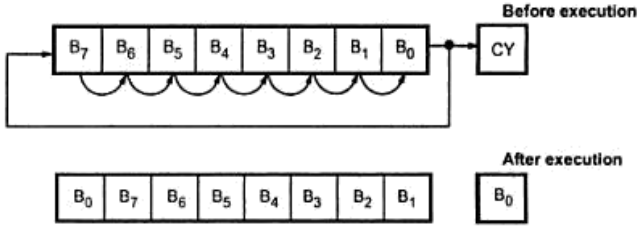
1. RLC
This instruction rotates the contents of the accumulator left by one position. Bit **B7** is placed in B_0 as well as in CY.
Operation



Example: A= 01010111 (57H) and CY= 1
RLC; After execution of the instruction the accumulator contents will be (1010 1110) AEH and carry flag will reset.

2. RRC
This instruction rotates the contents of the accumulator right by one position. Bit B_o is placed in **B7** as well as in CY.

Operation :

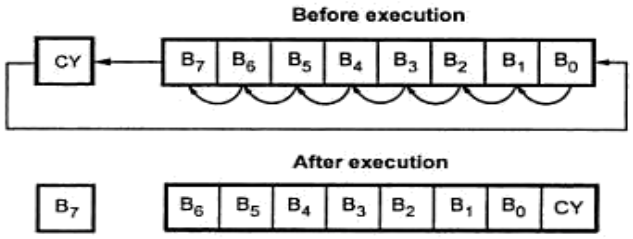


Example : A = 1001 1010 (9AH) and CY = 1

3. RAL

This instruction rotates the contents of the accumulator left by one position. Bit **B7** is placed in CY and CY is placed in B_o.

Operation :



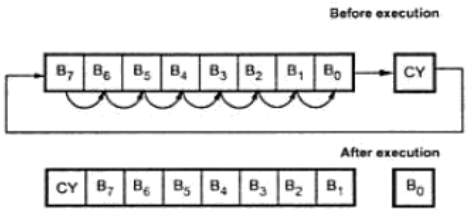
Example : A = 10101101 (ADH) and CY = 0

RAL; after execution of the instruction accumulator contents will be (0101 1010) 5AH and carry flag will set.

4. RAR

This instruction rotates the contents of the accumulator right by one position. Bit B_o is placed in CY and CY is placed in **B7.**

Operation :



Example : A = 1010 0011 (A3H) and CY = 0

RAR ; After execution of the instruction accumulator contents will be (0101 0001) 51H and carry flag will set.

3.5 Stack operation (or) Describe with a suitable example the operation of stack.

The stack is a portion of read/write memory set aside by the user for the purpose of storing information temporarily. When the information is written on the stack, the operation is called PUSH. When the information is read from stack, the operation is called POP.

The microprocessor stores the Information, much like stacking plates. Using this analogy of stacking plates it is easy to illustrate the stack operation.

Fig. 1.32 shows the stacked plates. Here, we realize that Hit is desired to take out the first stacked plate we will have to remove all plates above the first plate in the reverse order. This means that to remove first plate we will have to remove the third plate, then the second plate and

finally the first plate. This means that, the first information pushed on to the stack is the last information popped off from the stack. This type of operation is known as a first in, last out (FILO).

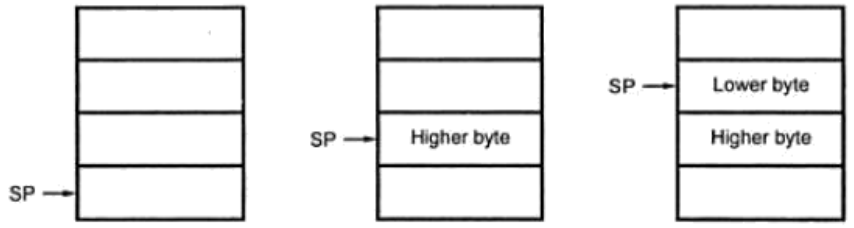
The special pointer register is called the stack pointer. During PUSH and POP operation, stack pointer register gives the address of memory where the information is to be stored or to be read. The stack pointer's contents are automatically manipulated to point to stack top. The memory location currently pointed by
stack pointer is called **top of stack**

Fig. 1.32 Stacked plates

1. PUSH rp

This instruction decrements stack pointer by one and copies the higher byte of the register pair into the memory location pointed by stack pointer. It then decrements the stack pointer again by one and copies the lower byte of the register pair into the memory location pointed by stack pointer.

Operation : SP ← SP – 1, (SP) ← rpH, SP ← SP – 1, (SP) ← rpL.



(a) Initial Position **(b) Decrements SP & stores higher byte** **(c) Decrements SP & stores lower byte**

Fig. 1.33 Steps involved in PUSH operation

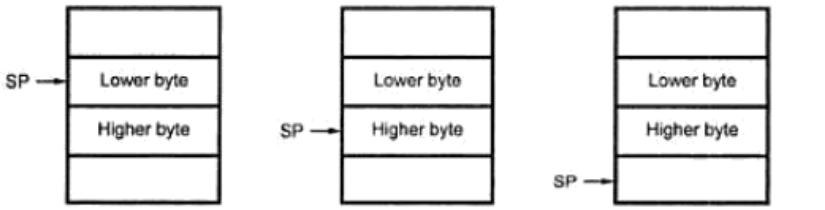
2. PUSH PSW This instruction decrements stack pointer by one and copies the accumulator contents into the memory location pointed by stack pointer. It then decrements the stack pointer again by one and copies the flag register into the memory location pointed by the stack pointer.

Operation : SP ← SP - 1
(SP) ← A
SP ←SP – 1
(SP)←Flag register

3. POP rp

This instruction copies the contents of memory location pointed by the stack pointer into the lower byte of the specified register pair and increments the stack pointer by one. It then copies the contents of memory location pointed by stack pointer into the higher byte of the specified register pair and increments the stack pointer again by one.

Operation : rpL ← (SP)
SP ← SP + 1
rpH ← (SP), SP ← SP + 1



(a) Initial position, (b) Increments SP and (c) Increments SP
Fig. 1.34 Steps involved in POP operation

4. POP PSW

This instruction copies the contents of memory location pointed by the stack pointer into the flag register and increments the stack pointer by one. It then copies the contents of memory location pointed by stack pointer into the accumulator and increments the stack pointer again by one.

Operation : Flag register ← (SP)
SP ←SP + 1
A ← (SP)
SP← SP + 1

5. SPHL

This instruction copies the contents of HL register pair into the stack pointer. The contents of H register are copied to higher order byte of stack pointer and contents of L register are copied to the lower byte of stack pointer. Operation: SP ←HL

Example: HL = 2500H
SPHL; This instruction will copy 2500H into stack pointer. So after execution of instruction stack pointer contents will be 2500H.

6. XTHL

This instruction exchanges the contents of memory location pointedby the stack pointer with the contents of L register and the contents of the next memory location with the contents of H register. This instruction does not modify stack pointer contents.

Operation : L ↔ (SP)
H ↔ (SP + 1)

3.6 Branch group (or) Explain about the Branch Group instructions in detail.

1. JMP addr

This instruction loads the PC with the address given within the instruction and resumes the program execution from this location.

Operation: **PC ←addr**

Example

JMP 2000H; This instruction will load PC with 2000H and processor will fetch next Instruction from this address.

2. Jcond addr This instruction causes a jump to an address given in the instruction if the desired condition occurs in the program before the execution of the instruction. The Table 1.5 shows the possible conditions for jumps.

Instruction code	Description	Condition for jump
JC	Jump on carry	CY = 1
JNC	Jump on not carry	CY = 0
JP	Jump on positive	S = 0
JM	Jump on minus	S = 1
JPE	Jump on parity even	P = 1
JPO	Jump on parity odd	P = 0
JZ	Jump on zero	Z = 1
JNZ	Jump on not zero	Z = 0

Table 1.5 Conditional jumps

3. CALL addr

The CALL instruction is used to transfer program control to a subprogram or subroutine. This instruction pushes the current PC contents onto the stack and loads the given address into the PC. Thus the program control is transferred to the address given in the instruction. Stack pointer is decremented by two.

Note: The stack is a part of read/write memory set aside for storing intermediate results and addresses.

Operation : $(SP - 1) \leftarrow PC_H$
$(SP - 2) \leftarrow PC_L$
$SP \leftarrow SP - 2$
$PC \leftarrow addr$

4. C cond addr

This instruction calls the subroutine at the given address if a specified condition is satisfied. Before call it stores the address of instruction next to the call on the stack and decrements stack pointer by two. The Table 1.6 shows the possible conditions for calls.

Instruction	Description	Condition for
CC	Call on *carry*	CV = 1
CNC	Call on not carry	CV = 0
CP	Call on positive	S = 0
CM	Call on minus	S = 1
CPE	Call on parity even	P = 1
CPO	Call on parity odd	P = 0
CZ	Call on zero	Z = 1

Operation : If condition true (SP - 1) ←PCH
(SP - 2) ← PCL
PC ←addr
else PC <— PC + 3

4. RET

This instruction pops the return addr (address of the instruction next to CALL in the main program) from the stack and loads program counter with this return address. Thus transfers program control to the instruction next to CALL in the main program.

Operation PCL← (SP)
PCH← (SP+1)
SP←SP+2

6. R condition

This instruction returns the control to the main program if the specified condition is satisfied. Table 1.7 shows the possible conditions for return.

Instruction code	Description	Condition for RET
RC	Return on carry	CY = 1
RNC	Return on not carry	CY = 0
RP	Return on positive	S = 0
RM	Return on minus	S = 1
RPE	Return on parity even	P = 1
RPO	Return on parity odd	P = 0
RZ	Return on zero	Z = 1
RNZ	Return on not zero	Z = 0

Table 1.7 Conditions for return

7. PCHL

This instruction loads the contents of HL register pair into the program counter. Thus the program control is transferred to the location whose address is in HL register pair.

Operation PC ← HL

Example HL = 6000H

PCHL; This instruction will load 6000H into the program counter

8. RST n

This instruction transfers the program control to the specific memory address as shown in Table 1.8.

Instruction code	Vector address
RST 0	0 × 8 = 0000H
RST 1	1 × 8 = 0008H
RST 2	2 × 8 = 0010H
RST 3	3 × 8 = 0018H
RST 4	4 × 8 = 0020H
RST 5	5 × 8 = 0028H
RST 6	6 × 8 = 0030H
RST 7	7 × 8 = 0038H

Table 1.8 Vector addresses for return instructions

3.7 Input/output instruction (or) Explain IN/OUT instructions in detail.

1. IN addr (8-bit)

This instruction copies the data at the port whose address is specified in the instruction into the accumulator. Operation: A ← (addr)

Example: Port address = 80H, data stored at port address 80H.**(80H) = 10H**

IN 80H: This instruction will copy the data stored at address 80H. i.e .data 10H in the accumulator.

2. OUT addr (8-bit) This instruction sends the contents of accumulator to the output port whose address is specified within the instruction.

Operation: (addr) ←

A Example: A = 40H

OUT 50H; This instruction will send the contents of accumulator (40H) to the output port whose address is 50H.

3.8 Machine Control Group instructions (or) Explain about Machine Control Group instructions with examples.

1. EI

This instruction sets the interrupt enable flip flop to enable interrupts. When the microprocessor is reset or after interrupt acknowledge, the interrupt enable flip-flop is reset. This instruction is used to re enable the interrupts.

Operation: IE (F/F)←1

2. DI

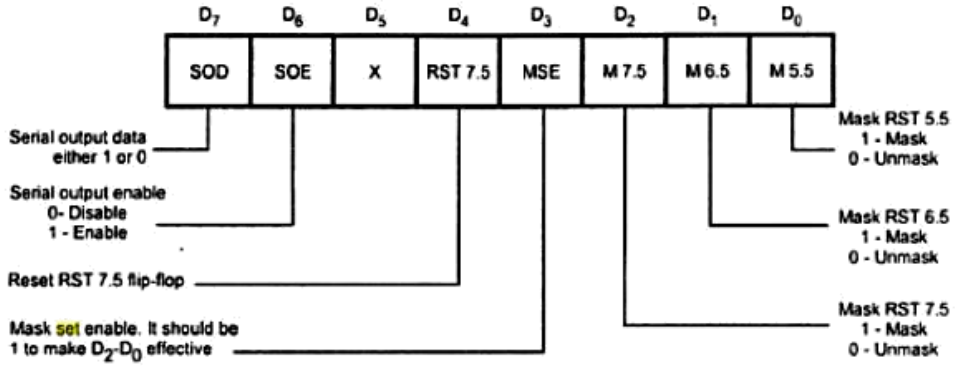
This instruction resets the interrupt enable flip-flop to disable interrupts. This instruction disables all interrupts except TRAP since TRAP is non-maskable interrupt (cannot be disabled. It is always enabled). Operation: IE (F/F) ←0

3. NOP: No operation is performed.

4. HLT: This instruction halts the processor.

5. SIM

This instruction masks the interrupts as desired. It also sends out serial data through the SOD pin. For this instruction command byte must be loaded in the accumulator.



Example : i) A = 0EH

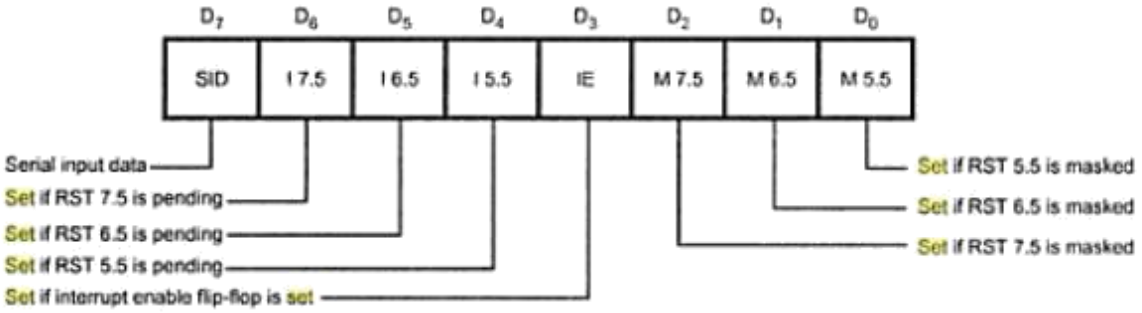
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
SOD	SOE	X	RST 7.5	MSE	M 7.5	M 6.5	M 5.5	Register A
0	0	0	0	1	1	1	0	= 0EH

6. RIM

This instruction copies the status of the interrupts into the accumulator. It also reads the serial data through the SID pin.

Example

RDA ; After execution of RIM instruction if the contents of accumulator are 4BH then we get following information.



D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
SID	I 7.5	I 6.5	I 5.5	IE	M 7.5	M 6.5	M 5.5	Register A
0	1	0	0	1	0	1	1	= 4BH

i.e. a) RST 7.5 is pending
b) RST 5.5 and RST 6.5 are masked
c) Interrupt Enable flip-flop is set
d) Serial input data is zero.

4. Explain about Looping, Counting and Indexing.

Looping: In this technique, the program is instructed to execute certain set of instructions repeatedly to execute a particular task number of times. For example, to add ten numbers stored in the consecutive memory locations we *have* to perform addition ten times.

Counting: This technique allows programmer to count how many times the instruction/set of instructions are executed.

Indexing: This technique allows programmer to point or refer the data stored in sequential memory locations one by one. Let us see the program loop to understand looping, counting and indexing.

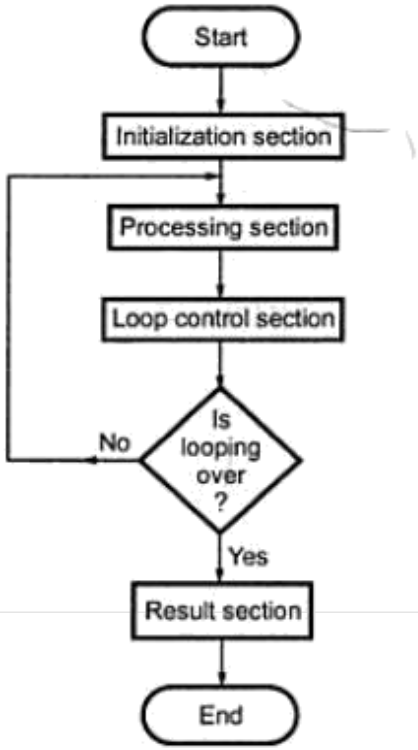
The program loop is the basic structure which forces the processor to repeat a sequence of instructions. Loops have four sections.

1 Initialization section. **2. Processing section.**

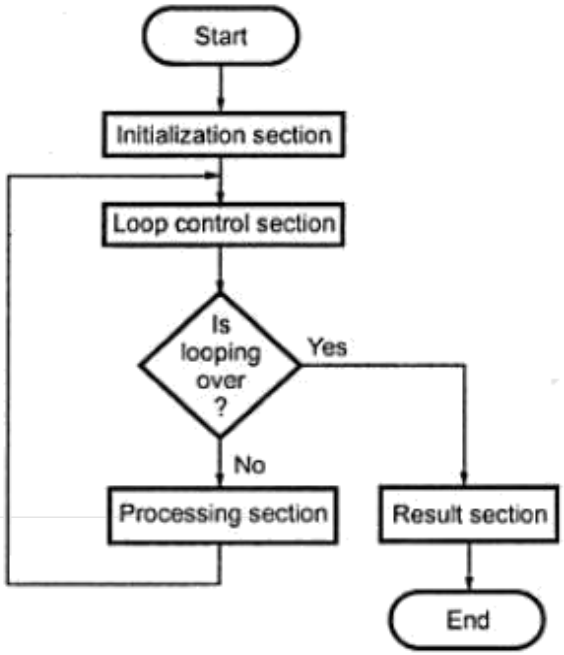
3. Loop control section **4. Result section.**

1. The initialization section establishes the starting values of
   - loop counters for counting how many times loop is executed,
   - address registers for indexing which give pointers to memory locations and
   - other variables
2. The actual data manipulation occurs in the processing section. This is the section which does the work.
3. The loop control section updates counters, indices (pointers) for the next iteration.
4. The result section analyzes and stores the results.

Flowchart



Flowchart 1



Flowchart 2

Note : The processor executes initialization section and result section only once, while it may execute processing section and loop control section many times. Thus, the execution time of the loop will be mainly dependent on the execution time of the processing section and loop control section. The flowchart 1 shows typical program loop. The processing section in this flowchart is always executed at least once. If you interchange the position ofthe processing and loop control section then it is possible that the processing section may not be executed at all, if necessary. Refer flowchart 2.

PROGRAMS

5. Write an ALU program for the addition of two 8 bit data.

LABEL	MNEMONICS	COMMENTS
	LXIH, 8200H	Set the pointer for Data
	MVIC, 00H	Clear C register account for Carry
	MOV A,M	Get the first data in A register
	INX H	Increment the HL pair for second data
	ADD M	Add the second data with accumulator
	JNC LOOP	Check for carry
	INR C	If carry=1, goto LOOP
LOOP	INX H	Increment the HL pair for sum

		value
	MOV M, A	Save the sum value in Memory
	INX H	Increment the HL pair for carry value
	MOV M, C	Save the carry value in Memory
	HLT	Halt

6. Write an ALU program for the subtraction of two 8 bit data.

LABEL	MNEMONICS	COMMENTS
	LDA, 8201H	Get the subtrahend in B register
	MOV B, A	Move the data in A register to B
	LDA, 8200H	Get the minuend in A register
	MVI C, 00H	Clear the C register for carry to indicate the sign
	SUB B	Subtract the data with A register
	JNC LOOP	Check for carry
	INR C	If carry=1, Increment the C-register
	CMA	Get the 2's Compliment for the difference
	ADD, O1H	Result in A register
LOOP	STA 8202H	Store the result in the memory
	MOV A, C	Move the C register to A register
	STA 8203H	Store the sign value in the memory
	HLT	Halt the program

7. Write an ALU program for the multiplication of two 8 bit data.(Nov/Dec-16)

LABEL	MNEMONICS	COMMENTS
	LXIH, 8200H	Set the pointer for Data
	MVIC, 00H	Clear C register account for Overflow
	XRA	Get the OR gate condition
	MOV B,M	Get the first data in B register
	INX H	Increment the HL pair for Second data
	MOV D,M	Get the first data in D register
REPT	ADD D	Add D register data with Accumulator
	JNC LOOP	Jump on no carry to Loop

		If carry = 1, Increment the C
	INR C	register
		Decrement the content in B
LOOP	DCR, B	register
	JNZ REPT	Repeat the addition
	INX H	Increment the HL pair
	MOV M, A	Store the low byte in the memory
	INX H	Increment the HL pair
	MOV M, C	Store the high byte in the memory
	HLT	Halt the program

8. Write an ALU program for the division of two 8 bit data. ***(Nov/Dec 15)(Apr/May-17)***

LABEL	**MNEMONICS**	**COMMENTS**
	LDA, 8201H	Set the pointer for dividend data
	MOV B, A	Move the data in A register to B
	LDA, 8200H	Get the divisor data in A register
	MVI C, 00H	Clear the C register for quotient value
LOOP 1	CMP B	Compare the content with B register
		If dividend is less than divisor go to LOOP
	JC LOOP	
		2
	SUB B	Subtract the divisor from dividend
		Increment the quotient by one for each
	INRC	
		subtraction
	JMP LOOP1	Jump to LOOP 1
LOOP 2	STA 8203H	Store the remainder in the memory
	MOV A, C	Move the C register to A register
		Store the Quotient value in the
	STA 8202H	memory
	HLT	Halt the program

9. Write an ALU program for sorting an array of data in ascending order. (May/June 12)

LABEL	**MNEMONICS**	**COMMENTS**
	LXIH 8200	Initialize the memory content

	MOV C, M	Move memory to C register
	DCR C	Decrement C register
Loop3	MOV B, C	Move C register to B register
	LXIH 8201	Load 8201 in HL pair
Loop2	MOV A, M	Move memory to accumulator
	INXH	Increment H register
	CMPM	Compare memory
	JC loop1	Jump to loop1 on carry
	MOV D, M	Move memory to D register
	MOV M, A	Move accumulator to memory
	DCX W	Decrement HL pair
	MOV M, D	Move D register to memory
	INXH	Increment HL register pair
Loop1	DCRC	Decrement HL pair
	JNZ loop2	Jump to loop 2 on non zero
	DCRC	Decrement HL pair
	JNZ loop3	Jump to loop 3 on non zero
	HLT	Stop the process

10. Write an ALU program for sorting an array of data in descending order.

LABEL	MNEMONICS	COMMENTS
	LXIH 8200	Initialise counter
	MOV C, M	Move memory to C reg
	DCR C	Decrement C reg
LOOP3	MOV B, C	Move C reg to B reg
	LXIH 8201	Load 8201 in HL pair
LOOP2	MOV A, M	Move memory to accumulator
	INXH	Increment H reg
	CMPM	Compare memory
	JNC LOOP1	Jump on no carry
	MOV D, M	Move memory to D reg
	MOV M, A	Move accumulator to memory
	DCX H	Decrement HL pair
	MOV M, D	Move D reg to memory
	DCX H	decrement HL reg pair
LOOP1	DCRB	Decrement HL pair
	JNZ LOOP2	Jump to loop 2 on HL memory
	DCRC	Decrement C reg
	JNC LOOP3	Jump loop 3 to memory

	HLT	Stop the process

11. Write an ALU program for finding the largest number in an array using 8085 processor? (April/ May-15,17) [APR/MAY 2018]

LABEL	MNEMONICS	COMMENTS
	MVIA, 00	Clear the accumulator
	MVIB, 05	Initialize the counter
	LXIH, 8200	Load HL pair
LOOP1	CMP M	Compare the memory with accumulator
	JNC LOOP2	Jump on no carry
	MOV, M	Move largest number to accumulator
LOOP2	INX H	Increment HL pair
	DCR B	Decrement B reg
	JNZ LOOP1	If B not equal to 0, jump to accumulator
	STA 8150	Store accumulator value in 8150
	HLT	Stop the program

12. Write an ALU program for finding the smallest number in an array using 8085 processor.(Apr/May 11)

LABEL	MNEMONICS	COMMENTS
	MVIA, FF	Move FF in the accumulator
	MVIB, 05	Initialize count with memory data
	LXIH, 8200	Load HL pair
LOOP1	CMPM	Compare the memory with accumulator
	JNC LOOP2	Jump 815C on array
	MOV A, B	Move smallest number accumulator
LOOP2	INX H	Increment number to accumulator
	DCR B	Decrement B reg
	JNZ LOOP1	Jump 8157 on nonzero
	STA 8250	Store accumulator value in 8250
	HLT	Stop the program

13. Write an ALU program for converting the ASCII to Hexadecimal using 8085 processor.

MNEMONICS	COMMENTS
LXIH8160	Point index memory
LXID8500	Permit index binary is to be

	hexadecimal
MOV A, M	Transfer type to A from HL pair
CALL BINARY	Call conversion sub routine
STAX D	-
HLT	Halt the program

SUBROUTINE PROGRAM:

LABEL	MNEMONICS	COMMENTS
BINARY	SUI, 30	-
	CPI DA	Compare accumulator with OA
	RC	Return if carry
	SUI, 07	-
	RET	Return to main program

14.. Write an ALU program for converting the Hexadecimal to ASCII using 8085 processor. ***(Nov/Dec 16)***

LABEL	MNEMONICS	COMMENTS
START	LXIH 8500	Pointer index to memory
	LXID 8160	Pointer index whether ASCII code
	MOV A,M	Move conten t memory to accumulator
	MOV B, A	Rotate right through
	RRC	-
	RRC	-
	RRC	-
	RRC	-
	Call ASCII	Call conversion location
RET	STAX D	Store in 8160 to next location
	INX D	Increment Opcode
	MOV A, B	Move B to acc
	Call ASCII	Call conversion location
	STAX D	Store second ASCII
	HLT	Halt the program

SUB ROUTINE PROGRAM

LABEL	MNEMONICS	COMMENTS
ASCII	ANI 0F	-
	CPI 0A	Compare the accumulator with 0A
	JC DEC	Jump to DEC when carry
	ADI 07	-
DEC	ADI 30	-
	RET	Return to main program

15. Write an ALU program to convert Decimal to Hexadecimal using 8085 processor.

LABEL	MNEMONICS	COMMENTS
Start	LDA8500	Load acc from 8500
	MOV B, A	Load B with A
	ANI OF	-
	MOV C, A	Move acc to C reg
	MOV A, B	Move B reg to acc
	RRC	Rotate right through OF acc
	RRC	4
	RRC	times and bring higher byte
	RRC	Mask Or the higher order
	ANI OF	byte
	MOV B, A	Load B with Acc
	XRA A	XOR acc with itself
	CMP B	Compare B
	JZ SKIP	-
Loop	ADI OA	Add OA to acc
	DCR B	Decrement to B
	JNZ LOOP	Jump if non zero to loop
Skip	ADDC	-
	STA 8160	Store the result
	HLT	Halt the program

16. Write an ALU program to convert Hexadecimal to Decimal using 8085 processor.

LABEL	MNEMONICS	COMMENTS
START	LDA 8500	Load acc from 8500
	MOV B, A	Load B with accumulator A
	ANI OF	-
	ADI 00	-
	DAA	-
	MOV D, A	Load D with accumulator A
	MOV A, B	Load B with accumulator A
	RRC	Rotate through carry
	RRC	and bring
	RRC	order
	RRC	Mask OF the higher order
	ANI OF	byte
	MOV E, A	Load E with accumulator A

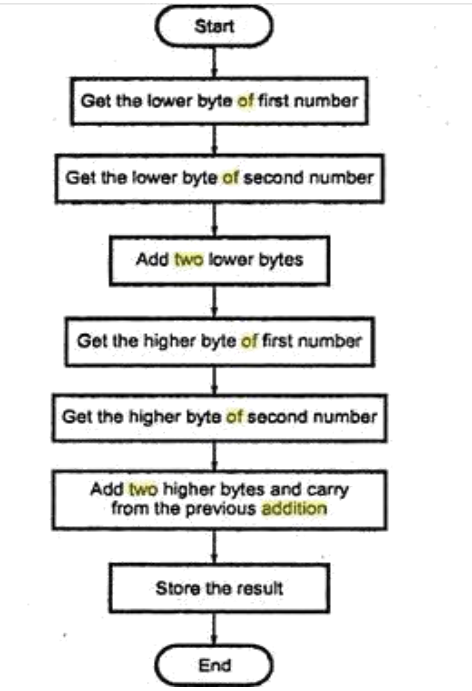
```
        XRA A       XOR accumulator with itself
        CMP E       Compare E
        JZ SKIP     -
LOOP    ADI 16      -
                    Decimal adjacent
        DAA         Accumulator
        DCR E       Decrement to E
        JNZ LOOP    -
SKIP    ADD D       -
                    Decimal adjacent
        DAA         accumulator
        STA 8160    Store the result
        HLT         Halt the program
```

17. Write an assembly language program with its output to add two 16 bit numbers using 8085. (Nov/Dec 13)

Add the 16-bit number in memory locations 2000H and 2001H to the 16-bit number in memory locations 2002H and 2003H. The most significant eight bits of the two numbers to be added are in memory locations 2001H and 2003H. Store the result in memory locations 2004H and 2005H with the most significant byte in memory location 2005H.

Sample problem
(2000H)=15H
(2001H)=1CH
(200311) =
B7H
(2003H)=5AH
Result = 1C15 + 5AB7H=
76CCH (2004H) =CCH
(2005H) = 76H
Source Program 1

```
LHLD 2000H      ; Get first 16-bit number in HL
XCHG            ; Save first 16-bit number in DE
LHLD 2002H      ; Get second 16-bit number in HL
MOV A, E        ; Get lower byte of the first number
ADD L           ; Add lower byte of the second number
MOV L, A        ; Store result in L register
MOV A, D        ; Get higher byte of the first number
ADC H           ; Add higher byte of the second number with carry
MOV H, A        ; Store result in H register
SHLD 2004H      ; Store 16-bit result in memory locations 2004H and 2005H.
HLT             ; Terminate program execution
```



Source program 2

```
LHLD 2000H     ; Get first 16-bit number
XCHG           ; Save first 16-bit number in DE
LHLD 2002H     : Get second 16-bit number in HL
DAD D          ; Add DE and HL
SHLD 2004H     ; Store 16 bit result in memory locations 2004H and 2005H.
HLT            ; Terminate program execution
```

In program 1 eight bit addition instructions are used (ADD and ADC) and (addition is performed in two steps. First lower byte addition using ADD instruction and then higher byte addition using ADC instruction. In program 2 16-bit addition instruction (DAD) is used.

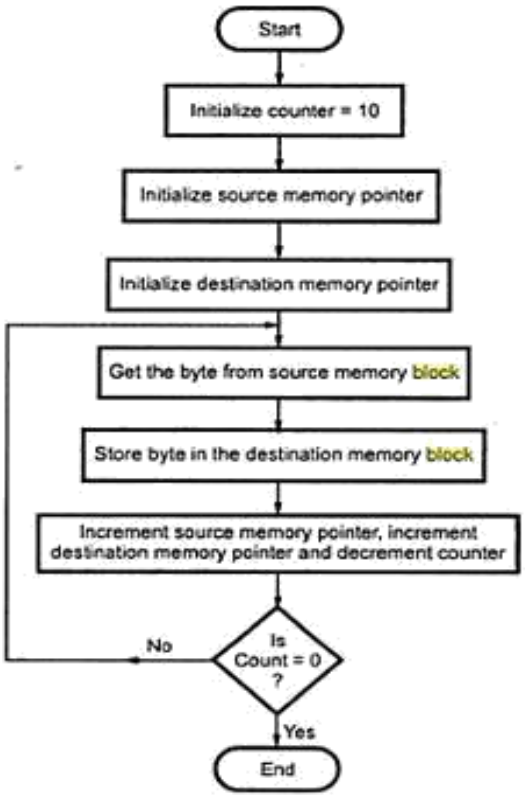
18 Data transfer from memory block B1 to memory block B2. (Nov/Dec 11)

Statement: Transfer ten bytes of data from one memory to another memory block Source memory block starts from memory location 2200H where as destination memory block starts from memory location 2300H.

Source program

```
        MVI C,0AH      ; Initialize counter
        LXI H, 2200H   ; Initialize source memory pointer
        LXI D, 2300H   ; Initialize destination memory pointer
BACK:   MOV A, M       ; Get byte from source memory block
        STAX D         ; Store byte in the destination memory block
        INX H          ; Increment source memory pointer
```

```
        INX D           ; Increment destination memory pointer
        DCR C           ; Decrement counter
        JNZ BACK        ; If counter ≠ 0 repeat
        HLT             ; Terminate program execution
```



19. Generate and display BCD up counter with frequency 1 Hz.(or)Write a program to count from 0 to 9 with one second delay between each count. At the count of 9, the counter should reset itself to 0 and repeat the sequence continuously. Assume the clock frequency is 1 MHz.(nov/dec 11)

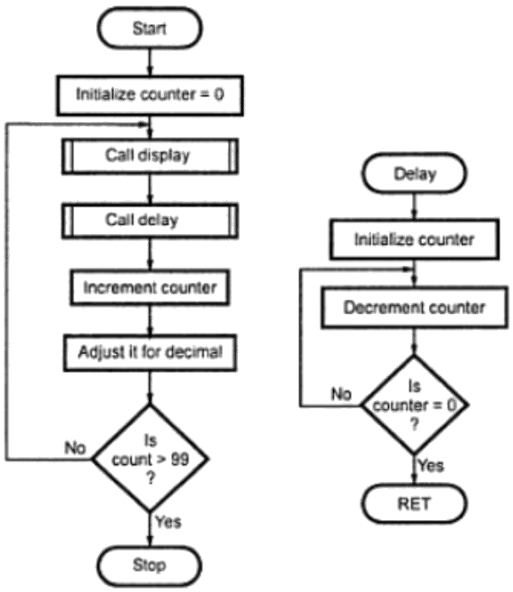
Statement: Write a program for displaying BCD up counter. Counter should count numbers from 00 to 99H and it should increment after every 1 sec. Assume operating frequency of 8085 equal to 3 MHz. Display routine is available. Solution:

```
        LXI SP, 27FFH       ; Initialize stack pointer
        MVI C, 00H          ; Initialize counter
BACK    : CALL Display      : Call display subroutine
        CALL Delay          ; Call delay subroutine
        MOV A, C
        ADI, 01             ; Increment counter
        DAA                 ; Adjust it for decimal
        MOV C.A             ; Store count
        CPI, 00             ; Check count is > 99
        JNZ BACK            ; If not, repeat
        HLT                 ; Stop
```

Delay Subroutine:

```
Delay:  MVI B, Multiplier-count     ; Initialize multiplier count
        LXI D, Initialize
BACK1:  Count
BACK:   DCX D                       ; Decrement count
        MOV A, E
        ORA D                       : Locally OR D and E
        JNZ BACK                    ; If result is not 0, repeat
                                    ; Decrement multiplier
        DCR B                       count
        JNZ BACKI                   ; If not zero, repeat
        RET                         ; Return to main program,
```

Flowchart:



Operating frequency : 3 MHz

$$\therefore \quad \text{Time for one T-state} = \frac{1}{3\ \text{MHz}} = 0.333\ \mu\text{sec}$$

$$\therefore \text{ Number of T-states required} = \frac{\text{Required time}}{\text{Time for 1 T-state}} = 3\times10^6$$

$$= \frac{1\ \text{sec}}{0.333\ \mu\text{sec}}$$

$$\text{Let us take multiplier count} = 3$$

$$\therefore \text{ Number of T-states required by inner loop} = \frac{3\times10^6}{3} = 1\times10^6$$

$$\therefore \quad 1\times10^6 = 10 + (\text{Count} - 1)\times 24 + 21$$

$$= 4166610$$

$$\text{Count} = 4166610 = \text{A2C2H}$$

20. Square Root of 8-bit Binary Number.(Apr/may 11)

Write an assembly language program to find the square root of an 8-bit binary number. The binary number is stored in the memory location 4200H if and store the square root in 4201H

Problem Analysis

The square-root can be computed by an iterative technique. First an initial value is assumed. Here the initial value of square root is taken as half the value of the given number. The new value of square root is computed by using an expression, XNEW = (X + Y/X)/2 where X is the initial value of the square root and Y is the given number. Then XNEW is compared with the initial value. If they are not equal, then the above process is repeated until X is equal to XNEW after taking XNEW as initial value, (i.e., X ← XNEW).

Algorithm

1. Load the given data (Y) in A-register.
2. Save the content of A-register in B-register.
3. Move 02H (divisor) to C-register.
4. Call DIV subroutine to get initial value of square root (X) in D-register.
5. Save the content of D-register (initial value X) in E register.
6. Move the given data (Y) from B-register to A-register.
7. Move the initial value (X) from D-register to C register.
8. Call DIV subroutine to get Y/X in D-register.
9. Move the Y/X available in D-register to A-register.
10. Add the value of X in E register to A-register to get X +Y/X in A-register.
11. Move 02H to C-register.
12. Call DIV subroutine to get new value of square root (XNEW) in D-register.
13. Compare X and XNEW.
14. If ZF =1, go to next step. If ZF = 0, go to step 5
15. Store the value of square root (A-register) in memory.
16. Stop.

Algorithm for subroutine DIV

1. Clear D-register.
2. Subtract the content of C-register (divisor) from the content of A-register (dividend).
3. Increment quotient (D-register).
4. Compare A-register and C-register.
5. If CF= 1, go to next step. If CF= 0 go to step 2.
6. Return to main program.

PROGRAM

```
        ORG 4100H     ; specify program starting address.
        LDA 4200H     ; Get the given data(Y) in A-register.
        MOV B, A      ; Save the data in B-register.
        MVI C, 02H    ; Get the divisor (O2H) in C-register.
        CALL DIV      ; Call division subroutine to get initialvalue(x)in the D-register.
REP:    MOV E,D       ;save the initial value in E-register.
        MOV A,B       ;Get the dividend(Y) in A-register.
        MOV C,D       ;Get the divisor(X) in C-register.
        CALL DIV      ;Call division subroutine to get Y/x in D.
        MOV A,D       ;Move Y/x in A-register.
        ADD E         ; Get ((Y/x)+x) in A-register.
        MVI C,02H     ;Get the divisor (02H) in C-register.
        CALL DIV      ;Call division subroutine to get XNEW in 0-register.
        MOV A,E       ;Get X in A-register.
        CMP D         ;Compare x and XNEW.
        JNZ REP       ;If XNEW is not equal to x, then repeat.
        STA 4201H     ;Save the square root in memory.
        HLT           ;Halt program execution.
```

DIVISION SUBROUTINE

```
DIV:  MVI D, 00H      ; Clear D-register for quotient.
NEXT: SUB C           ; subtract the divisor from dividend.
      INR D           ; Increment the quotient.
      CMP C           ; Repeat subtraction until the divisor
      JNC NEXT        ; is less than dividend.
      RET             ; Return to main program.
      END
```

Sample data

Input Data : 64_H

Output Data : $0A_H$

Memory address	Content
4200	64
4201	0A

21) Write An 8085 Program To Find The Avearage Of 10 Numbers And find the execution time of program. [APR/MAY 2018]

Aim:

To write an assembly language program to calculate the sum of datas using 8085 microprocessor kit.

Address	Label	Mnemonics	Hex Code	Comments
4100		LDA 4200	3A 00,42	Load the accumulator with number of values
4103		MOV B,A	4F	Move it from A to C
4104		LXI H, 4201	21,01,42	Load the starting address of data array
4107		SUB A	97	Intialise 'A' as 00
4108		MOV B,A	47	Intialise 'B' as 00
4109	Loop	ADD M	86	Add the previous sum with next data
410A		JNC Skip	D2, 0E, 41	Jump on if no carry
410D		INR B	04	Increment carry by one
410E	Skip	INX H	23	Increment pointer for next data
410F		DCR C	0D	Decrement 'C' by one
4110		JNZ Loop	C2, 09, 41	Jump if not zero
4113		STA 4400	32,00,44	Store the sum in accumulator
4116		MOV A,B	78	Move the value of carry to A from B
4117		STA 4401	32,01,44	Store the carry in memory
411A		HLT	76	End of program

Input

Input Address	Value
4200	04
4201	07
4202	09
4203	03
4204	04

Output

Output Address	Value
4400	17
4401	00

07 + 09 + 03 + 04 = 23

= 17 (in Hexa decimal)
(0F + 8 = 233)

```
0F   =   0000  1111
08   =   0000  1000
         ----------
         0001  0111
         1     7
```

Result:

The assembly language program for sum of datas was executed successfully using 8085 microprocessor kit.

22) Illustrate in brief about Lookup Table in 8085 microprocessor.

INTRODUCTION

Lookup tables are used for data conversion and also to access data that are in tabular form. Tabular data includes both numeric and alphabetic data in the form of character strings as well as in binary form. This experiment uses lookup tables to illustrate conversion from one code to another and also

to lookup character string data. It also illustrates how a jump table accesses various software as required by some applications.

OBJECTIVES

1. Use a lookup table to convert from one numeric code to another.
2. Locate and display character string data located in a lookup table.
3. Use a lookup to obtain jump or call addresses for computed jumps or calls.
4. Search a table for information.

PROCEDURE

A lookup table is a group of data organized so it is easily accessed without searching. An example lookup table may contain information such as 7 segment code for numeric displays. Or it may contain EBCDIC (extended binary coded decimal interchange code) that is converted to ASCII through a lookup table. Neither of these codes can be converted by using a numeric technique such as conversion from ASCII to BCD (subtract 30H).

Suppose that alphabetic data must be encrypted as a different code. A lookup table could be used to store the encryption codes. Such a lookup table and a procedure that converts ASCII to the encrypted code appears in Example 11 1. Note that the ASCII coded character in AL is converted to an entry from the table by using the XLAT (translate) instruction. The XLAT instruction adds the contents of AL to the contents of BX to form an address in the table. It then transfers a copy of the data at that address to the AL register. This software uses two encryption tables: the UTAB table encrypts uppercase letters and the LTAB table encrypts lowercase letters. The letters stored in the tables can be changed for different encryption codes. They could also be randomized occasionally for additional security. Of course this example may only be practical for developing anagrams or word scrambles for the newspaper. The XLAT instruction uses a segment override prefix to override the default segment (DS).

8085-program-for-**cube-of-a-number**-data-is-load-on-7000h-and-result-is-store-on-7001h-memory-location

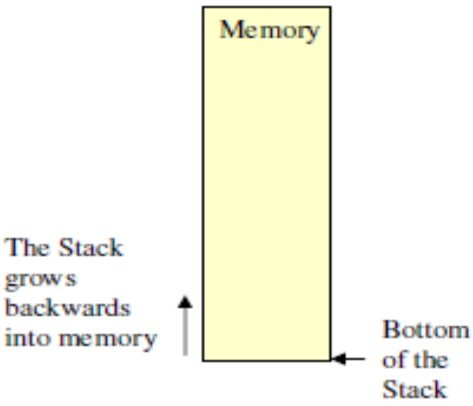
	LDA 7000H	3A,00,70
	MVI B 02H	06,02
	MOV C A	4F
	MOV D A	57
L2:	MVI A 00H	3E,00
L1:	ADD D	82
	DCR C	0D
	JNZ L1	C2,0B,70
	MOV C A	4F
	DCR B	05
	JNZ L2	C2,09,70
	MOV A C	79
	STA 7001H	32,01,70
	HLT	76

23) With an example briefly describe about Stack & Subroutine Instruction.

The stack is an area of memory identified by the programmer for temporary storage of information.

- The stack is a LIFO structure.
  - Last In First Out.
- The stack normally grows backwards into memory.

– In other words, the programmer defines the bottom of the stack and the stack grows up into reducing address range.



- Given that the stack grows backwards into memory, it is customary to place the bottom of the stack at the end of memory to keep it as far away from user programs as possible.
- In the 8085, the stack is defined by setting the SP (Stack Pointer) register.
  - LXI SP, FFFFH
- This sets the Stack Pointer to location FFFFH (end of memory for the 8085).
- The Size of the stack is limited only by the available memory

Saving Information on the Stack

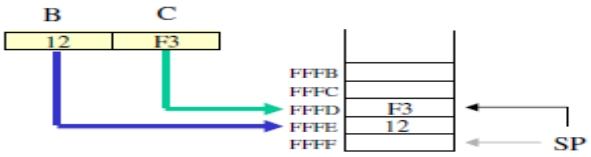
4

- 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

5

- PUSH B (1 Byte Instruction)
  - 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



The POP Instruction

6

- POP D (1 Byte Instruction)
  - Copy the contents of the memory location pointed to by the SP to register E
  - Increment SP
  - Copy the contents of the memory location pointed to by the SP to register D
  - Increment SP

Operation of the Stack

- 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.
- During poping, the stack operates in a "use then increment" style.
  - The information is retrieved from the top of the the stack and then the pointer is incremented.
- The SP pointer always points to "the top of the stack".

The PSW Register Pair

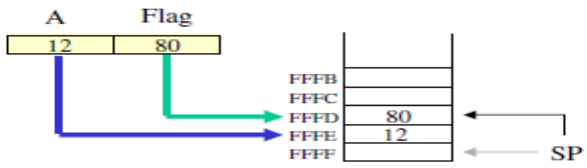
9

- The 8085 recognizes one additional register pair called the PSW (Program Status Word).
  - This register pair is made up of the Accumulator and the Flags registers.
- It is possible to push the PSW onto the stack, do whatever operations are needed, then POP it off of the stack.
  - The result is that the contents of the Accumulator and the status of the Flags are returned to what they were before the operations were executed.

PUSH PSW Register Pair

10

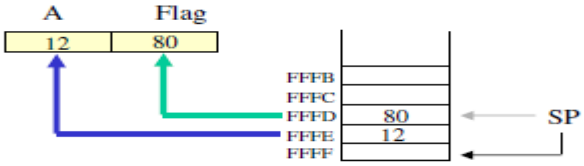
- PUSH PSW (1 Byte Instruction)
  - Decrement SP
  - Copy the contents of register A to the memory location pointed to by SP
  - Decrement SP
  - Copy the contents of Flag register to the memory location pointed to by SP



Pop PSW Register Pair

11

- POP PSW (1 Byte Instruction)
  - Copy the contents of the memory location pointed to by the SP to Flag register
  - Increment SP
  - Copy the contents of the memory location pointed to by the SP to register A
  - Increment SP



Modify Flag Content using PUSH/POP

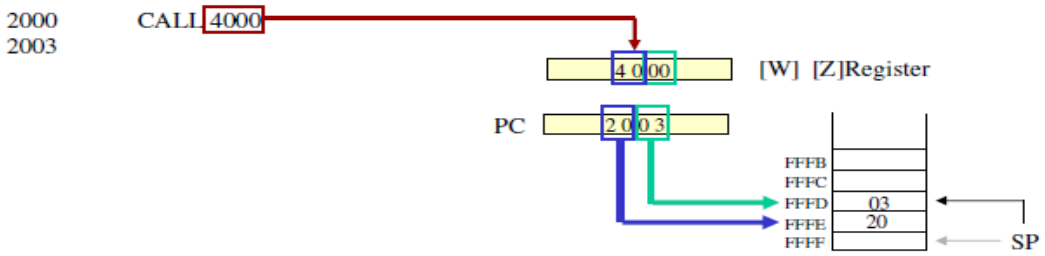
- Let, We want to Reset the Zero Flag
- 7 6 5 4 3 2 1 0
- 8085 Flag : S | Z | X |AC|X |P |X |Cy
- Program:
  - LXI SP FFFF
  - PUSH PSW
  - POP H
  - MOV A L
  - ANI BFH (BF$_H$= 1011 1111) * Masking
  - MOV L A
  - PUSH H
  - POP PSW

Subroutines

- A subroutine is a group of instructions that will be used repeatedly in different locations of the program.
  - Rather than repeat the same instructions several times, they can be grouped into a subroutine that is called from the different locations.
- In Assembly language, a subroutine can exist anywhere in the code.
  - However, it is customary to place subroutines separately from the main program.
- The 8085 has two instructions for dealing with subroutines.
  - The CALL instruction is used to redirect program execution to the subroutine.
  - The RET insutruction is used to return the execution to the calling routine.

The CALL Instruction

- CALL 4000H (3 byte instruction)
  - When CALL instruction is fetched, the MP knows that the next two Memory location contains 16bit subroutine address in the memory.



- MP Reads the subroutine address from the next two memory location and stores the higher order 8bit of the address in the W register and stores the lower order 8bit of the address in the Z register

– Pushe the address of the instruction immediately following the CALL onto the stack [Return address]

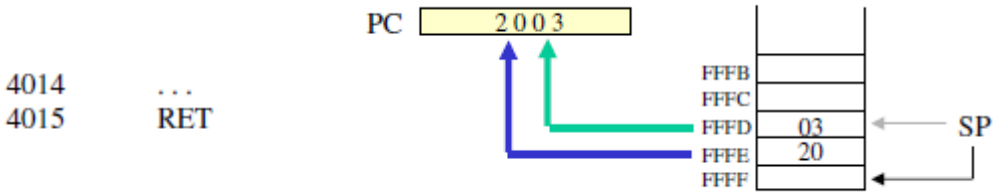
– Loads the program counter with the 16-bit address supplied with the CALL instruction from WZ register.

The RET Instruction

• RET (1 byte instruction)

– Retrieve the return address from the top of the stack

– Load the program counter with the return address.



Things to be considered in Subroutine

• The CALL instruction places the return address at the two memory locations immediately before where the Stack Pointer is pointing.

– You must set the SP correctly BEFORE using the CALL instruction.

• The RET instruction takes the contents of the two memory locations at the top of the stack and uses these as the return address.

– Do not modify the stack pointer in a subroutine.

You will loose the return address.

• Number of PUSH and POP instruction used in the subroutine must be same, otherwise, RET instruction will pick wrong value of the return address from the stack and program will fail.

Passing Data to a Subroutine

• Data is passed to a subroutine through registers.

– Call by Reference:

• The data is stored in one of the registers by the calling program and the subroutine uses the value from the register. The register values get modified within the subroutine. Then these modifications will be transferred back to the calling program upon returning from a subroutine

– Call by Value:

• The data is stored in one of the registers, but the subroutine first PUSHES register values in the stack and after using the registers, it POPS the previous values of the registers from the stack while exiting the subroutine.

i.e. the original values are restored before execution returns to the calling program.

• The other possibility is to use agreed upon memory locations.

– The calling program stores the data in the memory location and the subroutine retrieves the data from the location and uses it.

Cautions with PUSH and POP

• PUSH and POP should be used in opposite order.

• There has to be as many POP's as there are PUSH's.

– If not, the RET statement will pick up the wrong information from the top of the stack and the program will fail.

• It is not advisable to place PUSH or POP inside a loop.

Conditional CALL and RTE Instructions

• The 8085 supports conditional CALL and conditional RTE instructions.

– The same conditions used with conditional JUMP instructions can be used.

– CC, call subroutine if Carry flag is set.

– CNC, call subroutine if Carry flag is not set

– RC, return from subroutine if Carry flag is set

– RNC, return from subroutine if Carry flag is not set

Write a Program that will display FF and 11 repeatedly on the seven segment display. Write a 'delay' subroutine and Call it as necessary.

```
      C000: LXISP FFFF
      C003: MVIA FF
      C005: OUT 00
      C007: CALL 14 20
      C00A: MVIA 11
      C00C: OUT 00
      C00E: CALL 14 20
      C011: JMP 03 C0
DELAY: C014: MVIB FF
      C016: MVIC FF
      C018: DCR C
      C019: JNZ 18 C0
      C01C: DCR B
      C01D: JNZ 16 C0
      C020: RET
```

ANNA UNIVERSITY QUESTIONS

PART-A:

1. Why do we need look – up table? (Nov/Dec 11) (Nov/Dec 14)
2. What are the instructions used for subroutine? (Nov/Dec 13) (April/May-15)
3. List out the five categories of the 8085 instructions. Give examples of the instructions for each group (or) how are the 8085 instructions classified according to the functional categories? (Nov/Dec 11) [APR/MAY 2018]
4. What is the use of addressing modes, mention the different types. (May/June 12)(Nov/Dec 13)[Nov/Dec 2017]
5. Define stack and explain stack related instructions. (Nov/Dec 13) [APR/MAY 2018]
6. Compare CALL and PUSH instructions CALL PUSH. (Nov/Dec 11)
7. Compare RET and POP (Nov/Dec 11)
8. State the functions of given 8085 instructions: JP, JPE, JPO, JNZ (Apr/May 11)
9. Where is the READY signal used? (Nov / Dec-09)
10. How is PUSH B instruction executed? Find the status after the execution. (Apr/May 11)
11. What is the use of branching instructions? Give example. (May/June 12)
12. What is the different machine control instructions used in 8085 microprocessor? (Nov/Dec 13)
13. What is the similarity and difference between subtract and compare instructions? (May/June 14)
14. What is NOP? State its importance. (May/June 14)
15. What is the function of rotate instructions? (Nov/Dec-14)
16. List any two data manipulation instructions? (April/May-15)
17. How is time delay generated using subroutines?(Nov/Dec-15) 18.Explain the functioning of CMP instructions. (Nov/Dec-15)
19. Explain the difference between a JMP instruction and CALL instruction.[Nov/Dec 2017]

PART-B:

1. Describe the instruction format and addressing modes of 8085 microprocessor. (Apr/may 11)(may/June 12)(Apr/May-17)[Nov/Dec 2017]
2. Explain how instructions of 8085 can be classified?
3. Discuss in detail about the 8085 instruction set, explaining about the various types of operations. (or)With suitable example, discuss about 8085 microprocessor instructions used for data manipulation.
   (Apr/may 11)
   (Nov/Dec 13) Data transfer operations
   1. Arithmetic operations
   2. Logical operations(Nov/Dec 16)
   3. Branch operations and
   4. Stack, Input/output and Machine control operations

4. Describe with a suitable example the operation of stack. (May/June 12) 5. Describe with suitable examples the data transfer, loading and storing instructions.(May/june 12)
5. Explain the operations carried out when8085 executes the instruction:
   - i) MOV A,M (2M)
   - ii) XCHG (2M)
   - iii) DAD B (2M)
   - iv) DAA (3M)
   - v) LDA 6000 (2M)
   - vi) SHLD 4000 (2M)

PROGRAMS

1. Write an ALU program for the addition of two 8 bit data. (may/June 14)
2. Write an ALU program for the subtraction of two 8 bit data. .(may/June 14)
3. Write an ALU program for the multiplication of two 8 bit data.(Nov/Dec 11) (Nov/Dec-15,16)
4. Write an ALU program for the division of two 8 bit data. (Nov/Dec-15)(Apr/May-17)
5. Write an ALU program for sorting an array of data(unsigned number) in ascending order.(may/June 12) P.No:28
6. Write an ALU program for sorting an array of data in descending order.
7. Write an ALU program for finding the largest number in an array using 8085. (Apr/May-17)
8. Write an ALU program for finding the smallest number in an array using 8085 processor.(apr/may 11) P.No:30
9. Write an ALU program for converting the ASCII to Hexadecimal using 8085 processor.
10. Write an ALU program for converting the Hexadecimal to ASCII using 8085 processor. (Nov/Dec 16)
11. Write an ALU program to convert Decimal to Hexadecimal using 8085 processor.
12. Write an ALU program to convert Hexadecimal to Decimal using 8085 processor.
13. Write an assembly language program with its output to add two 16 bit numbers using 8085.(Nov/Dec 13) P.No:32
14. Sixteen bytes are stored in memory locations at XX50h to XX5Fh. Transfer the entire block of data to new memory locations starting at XX70h.(Nov/Dec 11) P.No:34
15. Write a program to count from 0 to 9 with one second delay between each count. At the count of 9, the counter should reset itself to 0 and repeat the sequence continuously. Assume the clock frequency is 1 MHz(Nov/Dec 11) P.No:34
16. Write an assembly language program based on 8085 microprocessor instruction set to find the square root of data from 1 to n using Lookup table.(Apr/may 11) P.No:36
17. Write An 8085 Program To Find The Avearage Of 10 Numbers And find the execution time of program. [APR/MAY 2018] P.No: 40

MAILAM ENGINEERINGCOLLEGE
MAILAM, 604304

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

Sub.Code/Sub.Name: EE6502 MICROPROCESSORS AND MICROCONTROLLERS Year / Sem: III/ V

UNIT-III- 8051 MICRO CONTROLLER

SYLLABUS

Hardware Architecture, pin outs – Functional Building Blocks of Processor – Memory organization – I/O ports and data transfer concepts– Timing Diagram – Interrupts-Comparison to Programming concepts with 8085.

Updated Question:

PART-A

Q.No.07	Page No: 03	(Nov/Dec 2017)
Q.No.02	Page No: 01	(Nov/Dec 2017)
Q.No.27	Page No: 06	(Apr/May 2018)
Q.No.34	Page No: 08	(Apr/May 2018)

PART-B

Q.No.09	Page No: 25	(Nov/Dec 2017)
Q.No.08	Page No: 23	(Nov/Dec 2017)
Q.No.01	Page No: 08	(Apr/May 2018)
Q.No.05	Page No: 16	(Apr/May 2018)

TEXT BOOKS:

1. Krishna Kant, "Microprocessor and Microcontrollers", Eastern Company Edition, Prentice Hall of India, New Delhi , 2007.
2. R.S. Gaonkar, 'Microprocessor Architecture Programming and Application', with 8085, Wiley Eastern Ltd., New Delhi, 2013.
3. Soumitra Kumar Mandal, Microprocessor & Microcontroller Architecture, Programming & Interfacing using 8085,8086,8051,McGraw Hill Edu,2013.

REFERENCES:

1. Muhammad Ali Mazidi & Janice Gilli Mazidi, R.D.Kinely 'The 8051 Micro Controller and Embedded Systems', PHI Pearson Education, 5th Indian reprint, 2003.
2. N.Senthil Kumar, M.Saravanan, S.Jeevananthan, 'Microprocessors and Microcontrollers', Oxford,2013.
3. Valder – Perez, "Microcontroller – Fundamentals and Applications with Pic," Yeesdee Publishers, Tayler & Francis, 2013.

Prepared By
1. Mrs.J.Srivandhana AP/MCA
2. Mrs.T.Kala AP/MCA

Verified By
HOD/MCA

Approved By
Principal

PART-A

1. What is a micro controller? Give examples.

Microcontrollers are designed as a single chip, which typically includes a MP, 64 bytes of R/W memory, 1K to 2 K bytes of ROM and several signal lines to connect I/O. These are complete micro computers on a chip also known as Microcontroller. Example: INTEL 8031, 8051 and 8097.

2. List the features of 8051 microcontroller? (May/June 2012) NOV/DEC 2017

The features are:

- 8 bit CPU with registers A (the accumulator) and B
- 16 bit Program Counter (PC) and Data Pointer (DPTR)
- 8 bit Program Status Word (PSW)
- 64K Program memory address space
- 64K Data memory address space,128 bytes of on chip data memory
- 32 I/0 pins for four 8 bit ports : Port 0, Port 1, Port 2, Port 3
- Two 16 bit timers / counters : T_0 and T_1,
- Full duplex UART: SBUF
- Two external and three internal interrupt sources
  - On chip clock oscillator.

3. Explain the operating mode 0 of 8051 serial ports?

- In this mode serial enters &exits through RXD, TXD outputs the shift clock.
- 8 bits are transmitted/received. 8 data bits (LSB first).
- The baud rate is fixed at 1/12 the oscillator frequency.

4. Explain the operating mode2 of 8051 serial ports?

- 11 bits are transmitted(through TXD)or received (through RXD):
- a start bit(0), 8 data bits(LSB first),a programmable 9th data bit,& a stop bite 1).
- ON transmit the 9th data bit (TB8 in SCON) can be assigned the value of 0 or 1.Or for eg:, the parity bit(P, in the PSW)could be moved into TB8.
- On receive the 9th data bit go into the RB8 in Special Function Register SCON, while the stop bit is ignored.
- The baud rate is programmable to either 1/32 or l/64th the oscillator frequency.

5. Explain the mode3 of 8051 serial ports?

- 11 bits are transmitted (through TXD) or received (through RXD)
- a start bit (0), 8 data bits (LSB first), a programmable 9th data bit & a stop bite 1).
- The baud rate in Mode3 is variable.
- In all the four modes, transmission is initiated by any instruction that uses SBUF as a destination register. Reception is initiated in Mode0 by the condition RI=0&REN=l. Reception is initiated in other modes by the incoming start bit if REN=1.

6. **Explain the interrupts of 8051 microcontroller? List the interrupts of 8051. (Nov/Dec 2013)(May/June 14) (Nov/Dec-15,16)**

The interrupts are:

External interrupt 0	IE0 0003H
Timer interrupt 0	TF0 000BH
External interrupt 1	IE1 0013H
Timer Interrupt 1	TF1 001BH
Serial Interrupt	
Receive interrupt	RI 0023H
Transmit interrupt	TI 0023H

7. **List the addressing modes of 8051? (Nov/Dec 2011)(Nov/Dec-14)(Apr/May-17) Nov/Dec-17**
   - Direct addressing
   - Immediate addressing
   - Register addressing
   - Index addressing
   - Register indirect addressing.
   - Bit addressing
   - Implicit addressing

8. **Write about CALL statement in 8051?**

   There are two subroutine CALL instructions. They are
   - LCALL (Long CALL)
   - ACALL (Absolute CALL)

Each increments the PC to the 1st byte of the instruction & pushes them in to the stack.

9. **Write about the jump statement?**

   There are three forms of jump. They are
   - LJMP(Long jump)-address 16
   - AJMP(Absolute Jump)-address 11
   - SJMP(Short Jump)-relative address

10. **Write a program to load accumulator, DPH, &DPL using 8051?**

```
MOV A, #30
MOV DPH, A
MOV DPL, A
```

11. **Write a program to find the 2's complement using 8051?**

```
MOV A, R0
CPL A
INC A
```

12. Write a program to add 2 8-bit numbers using 8051?

```
MOV A, #30H
ADD A, #50H
```

13. Write a program to swap two numbers using 8051?

```
MOV A, #DATA
SWAP A
```

14. Write a program to subtract 2 8-bit numbers &exchange the digits using 8051?

```
MOV A, #9F
MOV R0, #40
SUBB A, R0
SWAP A
```

15. Write a program to subtract the contents of R1 of Bank 0 from the contents of R0 of Bank 2 using 8051?

```
MOV PSW, #10
MOV A, R0
MOV PSW, #00
SUBB A, R1
```

16. Explain DJNZ instructions of Intel 8051 microcontroller?

- DJNZ Rn, rel (Decrement the content of the register Rn and jump if not zero.)
- DJNZ direct, rel (Decrement the content of direct 8-bit address and jump if not zero.)

19. State the function of RSI and RSO bits in the flag register of Intel 8051 microcontroller? (OR) Which bank of a register is being addressed at a particular time?

RS1, RS0 - Register bank select bits

20. Give the alternate functions for the port pins of port3?(April/may2011)

- RD - Read data control output.
- WR - Write data control output.
- Tl - Timer / Counter1 external input or test pin.
- T0 - Timer / Counter0 external input or test pin.
- INT 1 - Interrupt 1 input pin.
- TNT 0 - Interrupt 0 input pin.
- TXD - Transmit data pin for serial port in UART mode. RXD - Receive data pin for serial port in UART mode.

21. Specify the single instruction, which clears the most significant bit of B register of 8051, without affecting the remaining bits.

Single instruction, which clears the most significant bit of B register of 8051, without affecting the remaining bits, is CLR B.7

22. Name the special functions registers available in 8051. (May/June 2013)(Nov/Dec 14) (Apr/May-17)

ACC	Accumulator*
B	B Register*
PSW	Program Status Word*
SP	Stack Pointer
DPTR (Low)	Data Pointer Low
DPTR (High)	Data Pointer High
P0	Port 0*
PI	Port 1*
P2	Port 2*
P3	Port 3*
IP	Interrupt Priority control register *
IE	interrupt Enable control register *
TMOD	Timer/Counter Mode
TCON	Timer/Counter Control*
THO	(Timer/Counter) 0 High
TLO	(Timer/Counter) 0 Low
TH1	(Timer/Counter) 1 High
TL1	(Timer/Counter) 1 Low
SCON	Serial Control*
SBUF	Serial Data Buffer
PCON	Power Control

23. Explain the function of the pins PSEN & EA of 8051. (Nov/Dec-16)

PSEN stands for program store enable. In 8051 based system in which an external ROM holds the program code, this pin is connected to the OE pin of the ROM.

EA stands for external access.

- When the EA pin is connected to Vcc, program fetched to addresses 0000H through 0FFFH are directed to the internal ROM and program fetches to Addresses 1000H through FFFFH are directed to external *ROMI* EPROM.
- When the EA pin is grounded, all addresses fetched by program are directed to the external ROM/EPROM.

24. Explain the 16-bit registers DPTR and SP of 8051. (Apr/may 2011)

- **DPTR:** DPTR stands for data pointer. DPTR consists of a high byte (DPH) and a low byte (DPL). Its function is to hold a 16- bit address. It may be manipulated as a 16-bit data register or as two independent 8-bit registers. It serves as a base register in indirect jumps, lookup table instructions and external data transfer.
- **SP**: SP stands for stack pointer. SP is a 8- bit wide register. It is incremented before data is stored during PUSH and CALL instructions. The stack array can reside anywhere in on-chip RAM. The stack pointer is initialized to 07H after a reset. This causes the stack to begin at location 08H.

25. Write down the different operating modes for serial communication of 8051. (April/May-15)

Serial communication of 8051 operates under four modes. They are mode 0, mode 1, mode 2 and mode3 .SMO and SMI bits of SCON register specifies the mode.

26. Explain the register IE format of 8051.

- EA - Enable all control bit.
- ET2 - Timer 2 interrupt enable bit.
- ES - Enable serial port control bit.
- ETI - Enable Timer l control bit.
- EX 1 - Enable external interrupt 1 control bit. ETO - Enable Timer 0 control bit.
- EXO - Enable external interrupt 0 control bit.

26. Compare Microprocessor and Microcontroller. (April/May-18)

S.No.	Microprocessor	Microcontroller
1.	Microprocessor contains ALU, general purpose registers, stack pointer, program counter, and clock timing circuit and interrupt circuit.	Microcontroller contains the circuitry of microprocessor and in addition it has built-in ROM, RAM, I/0 devices, timers and counters.
2.	It has many instructions to move data between memory and CPU	It has one or two instructions to move data between memory and CPU.
3.	It has one or two bit handling instructions.	It has many bit handling instructions.
4.	Access times for memory and I/O devices are more.	Less access time for built-in memory and I/O devices.
5.	Microprocessor based system requires more hardware.	Microcontroller based system requires less hardware reducing PCB size and increasing the reliability.

28. How the RS -232C serial bus is interfaced to TTL logic device?

The RS-232C signal voltage levels are not compatible with TTL logic levels. Hence for interfacing TTL devices to RS- 232C serial bus, level converters are used. The popularly used level converters are MC 1488 & MC 1489 or MAX 232

29. What is 8051?

The 8051 is a standalone high performance microcontroller intended for use in sophisticated real time applications, such as instrumentation, industrial control, automobiles and computer peripherals.

30. Explain the operation of idle mode and power down mode?

- In idle mode, the CPU is turned off, whereas other devices like RAM and on-chip units remain active. Power down in idle mode is 15% of full power.

- In the power down mode, all on chip activities are suspended. The on-chip RAM continues to hold the data. The device draws only 10 micro ampere current.

31. What are the interrupt, timer and serial registers used in SFR?

INTERRUPT REGISTERS

IP	Interrupt Priority control register *
IE	lnterrupt Enable control register *
TIMER REGI STERS	
TMOD	Timer/Counter Mode
TCON	Timer/Counter Control*
THO	(Timer/Counter) 0 High
TLO	(Timer/Counter) 0 Low
TH1	(Timer/Counter) 1 High
TL1	(Timer/Counter) 1 Low
SERIAL COM MUNICATION REGISTERS	
SCON	Serial Control*
SBUF	Serial Data Buffer
PCON	Power Control

32. Write about program status word.

The Program Status Word (PSW) keeps the current status of the arithmetic and logical operations in different bits. The 8051 has four math flags mat respond automatically to the out of arithmetic and logic operations and 3 general purpose user flags that can be set 1 or 0 by the programmer as desired.

33. Write the function of TMOD register in 8051 microcontroller(Nov/Dec-15)

TMOD: Timer Mode Control Register

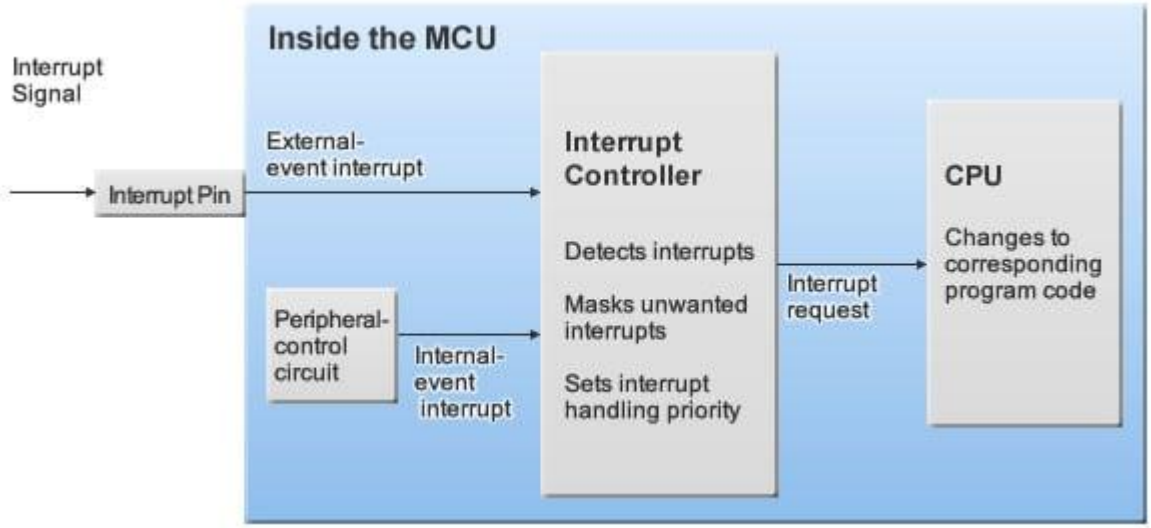
The TMOD SFR is used to control the mode of operation of both timers. Each bit of the SFR gives the microcontroller specific information concerning how to run a timer. The high four bits (bits 4 through 7) relate to Timer 1 whereas the low four bits (bits 0 through 3) perform the exact same functions, but for timer 0.

TxM1	**TxM0**	Timer Mode	Description of Mode
0	0	0	13-bit Timer.
0	1	1	16-bit Timer
1	0	2	8-bit auto-reload
1	1	3	Split timer mode

34. How the Microcontrollers respond to any interrupt request? APR/May 2018

Processing an Interrupt in the MCU

1. The main program is running.
2. An interrupt signal lets the MCU know that an event has occurred.
3. The MCU receives the interrupt signal, and suspends execution of the main program.
4. The MCU saves the current program execution state into its registers.
5. The MCU executes the interrupt routine corresponding to the received interrupt.
6. The MCU restores the saved program execution state.
7. Resume program execution.



Interrupt Processing Within the MCU

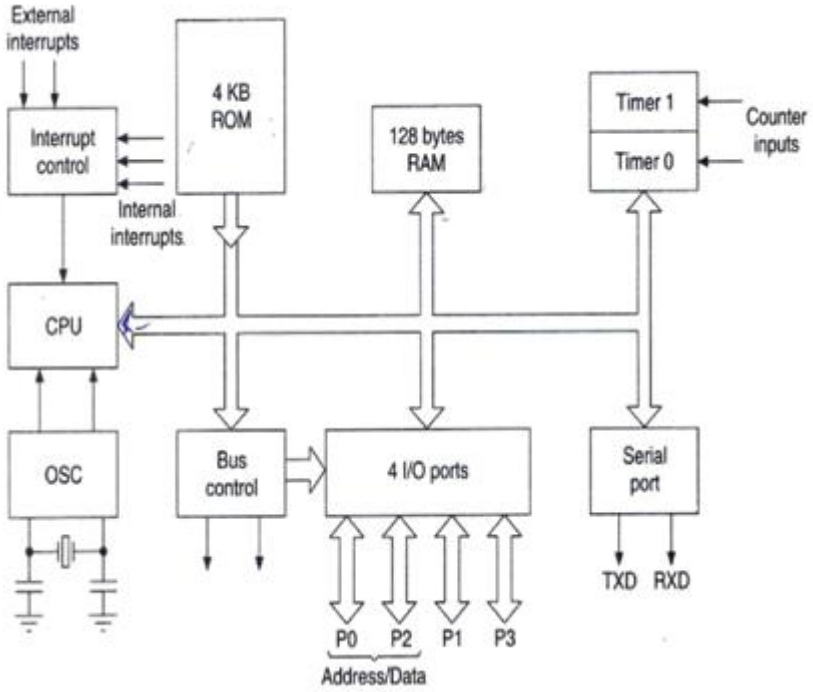
PART-B

1. Explain the Functional block diagram (or) architecture of 8051 microcontroller? (May/june2012, Nov/Dec-14, April/May-15 & 17) (April/May-18)

The features of the 8051are:

- 8 bit CPU with registers A (the accumulator) and B
- 16 bit Program Counter (PC) and Data Pointer (DPTR)
- 8 bit Program Status Word (PSW)
- 64K Program memory address space
- 64K Data memory address space,128 bytes of on chip data memory
- 32 I/0 pins for four 8 bit ports : Port 0, Port 1, Port 2, Port 3
- Two 16 bit timers / counters : T_0 and T_1,
- Full duplex UART: SBUF
- Two external and three internal interrupt sources
- On chip clock oscillator.

Functional Block Diagram of 8051 Microcontroller



Central Processing Unit:

The CPU is the **brain of the microcontrollers** reading user's programs and executing the expected task as per instructions stored there in. Its primary elements are

1. an Accumulator (ACC),
2. B register (B),
3. Stack pointer (SP),
4. Program counter (PC),
5. Program status word (PSW),
6. Data pointer register (DPTR) and
7. few more 8 bit registers.

1. Accumulator (ACC)

The accumulator performs **arithmetic and logic functions** on 8 bit input variables. Arithmetic operations include basic addition, subtraction, multiplication and division. Logical operations are AND, OR XOR as well as rotate, clear, complement etc.

Apart from the entire above, accumulator is responsible for conditional branching decisions and provides a temporary place in a data transfer operations within the device.

2. B Register (B)

B register is used in **multiply and divide operations**. During execution B register either keeps one of the two inputs or then retains a portion of the result. For other instructions it is used as **general purpose register**.

3. Stack Pointer (SP)

Stack Pointer (SP) is an 8 bit register. This pointer keeps **track of memory space** where the important register information is stored when the program flow gets into executing a subroutine. The stack portion may be **placed in anywhere in the on chip RAM**. But normally SP is initialized to 07H after a device reset and grows up from the location 08H. The SP is

automatically incremented or decremented for all **PUSH or POP** instructions and for all subroutine calls and returns.

4. Program Counter (PC)

The Program Counter (PC) is the 16 bit register **giving address of next instruction** to be executed during **program execution** and it always **points to the program memory space.**

5. Data Pointer Register (DPTR)

The Data Pointer Register (DPTR) is the **16 bit addressing register** that can be used to any 8 bit data from the data memory space. When it is not being used for this purpose, it can used as **two eight bit registers**, DPH and DPL.

6. Program Status Word (PSW)

The Program Status Word (PSW) keeps **the current status of the arithmetic and logical operations in different bits**. The 8051 has **four math flags** math respond automatically to the out of arithmetic and logic operations and **3 general purpose user flags** that can be set 1 or 0 by the programmer as desired.

The **math flags:** fire carry (C), auxiliary carry (AC), overflow (OF) and parity (P).
User flags are named flag 0 (F0), Register bank select bits RSO and K.

Input / Output Ports

8051 has 32 I/O pins configured as 4 eight bit parallel ports (P0, P1, P2 and P3). Each pin can **be used as an input or as an output** under the software control. These I/O pins can be **accessed directly by memory instructions** during program execution to get require flexibility.

Port 1 and Port 3 are available for **standard I/O** functions. **Port 3** pins has the additional functions : 2 external interrupt lines, 2 counter inputs, 2 serial port data lines and 2 timing control strobe lines.

Timers / Counters

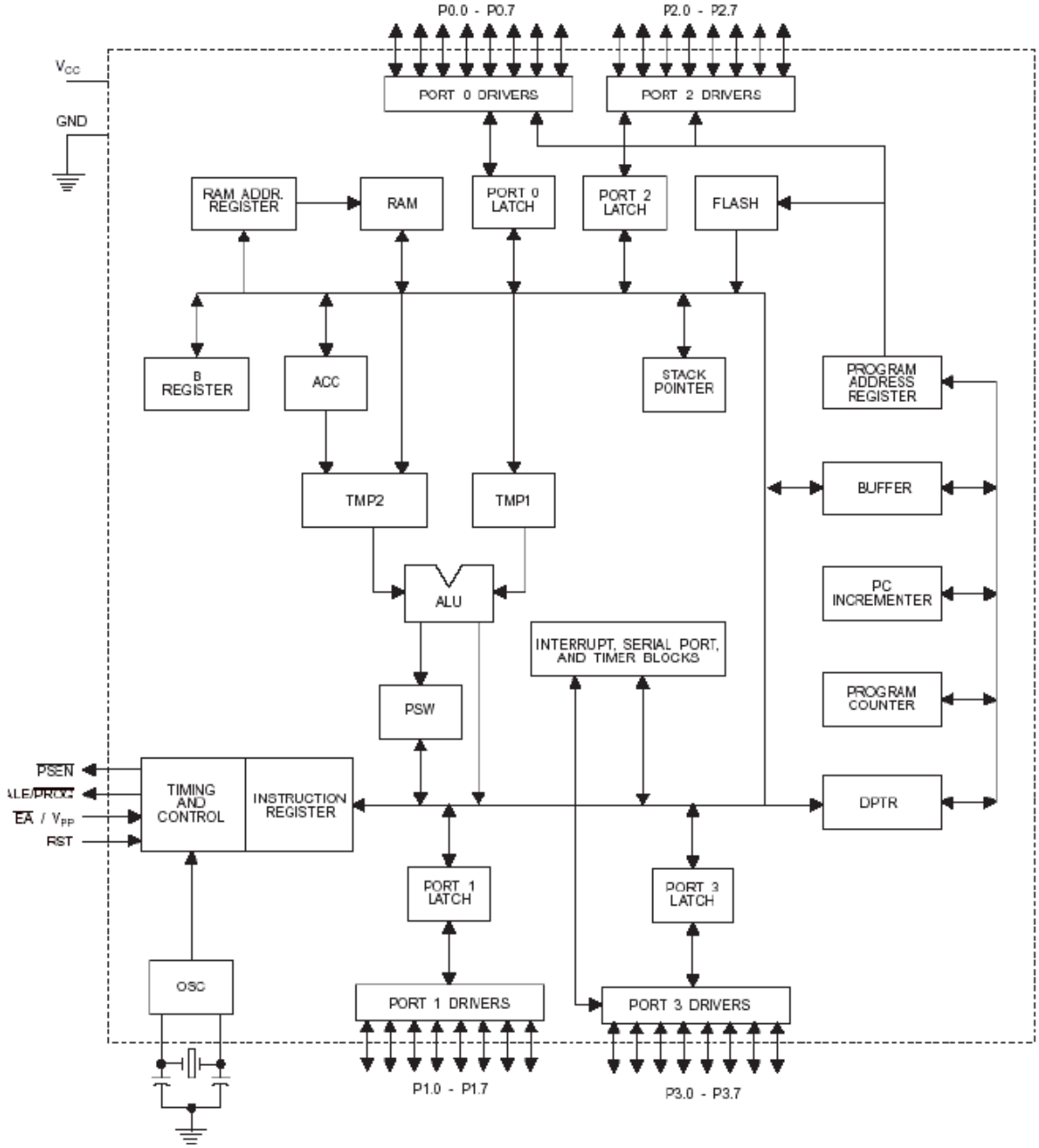
8051 has **two** 16 bit Timers / Counters, T0 and T1 capable of working in different modes. Each consists of a 'HIGH' byte and a 'LOW' byte which can be accessed under software. There is a mode control register (TMOD) and a control register (TCON) to configure these timers / counters in number of ways.

Serial Port

The 8051 has a high speed full duplex serial port which is software configurable in 4 basic modes:

- Shift register mode
- Standard UART mode
- Multiprocessor mode
- 9 bit UART mode. Full duplex means the data can go both ways at the same time.

Architecture of 8051



Interrupts

The 8051 has **five interrupt** sources: **One from the serial port (RI / TI)** when a transmission or reception operation is executed.

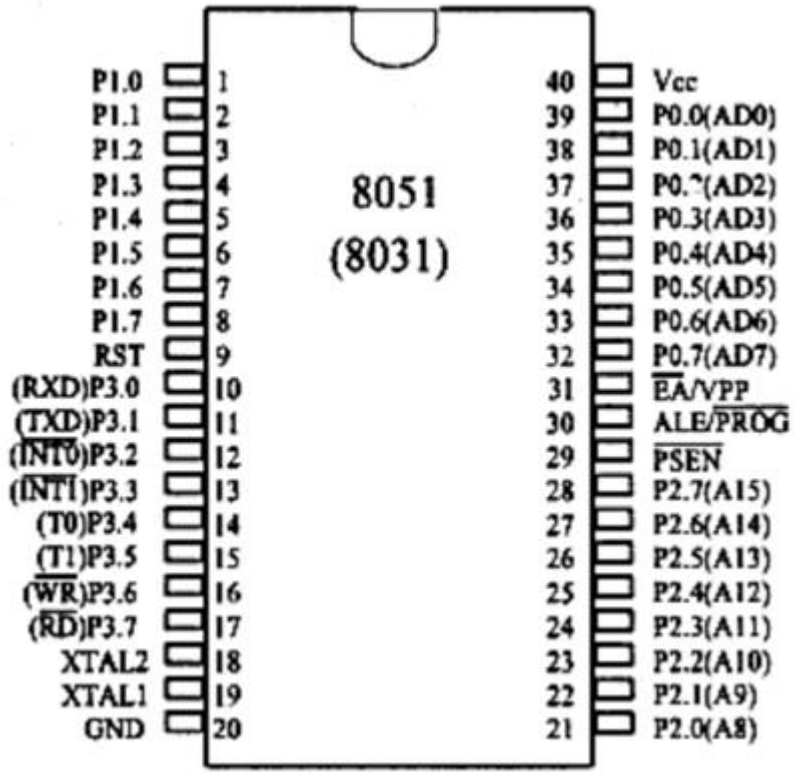
Two from the timers (TF0, TF1) when overflow occurs and **Two come from the two input pins INTO, INT1.** Each interrupt may be independently enabled or disabled to allow polling on same sources and each may be classified as high or low priority. These operations are selected by Interrupt Enable (IE) and Interrupt Priority (IP) registers.

Oscillator and Clock

The 8051 generates the clock pulses by which all internal operations are synchronized. Pins XTAL1 and XTAL 2 are provided for connecting a resonant network to form an oscillator. A quartz crystal is used for oscillator. The crystal frequency is the basic internal clock frequency of the microcontroller.

2. Explain in detail about the pin configuration (or) pin outs (or) signals of 8051 microcontroller?

Pin Description of 8051



Port 0(p0.0 to p0.7):
It is 8-bit bi-directional I/O port. It is bit/ byte addressable. During external memory access, it Functions as multiplexed data and low-order address bus AD0-AD7.

Port 1 (p1.0 to p1.7):
It is 8-bit bi-directional I/O port. It is bit/ byte addressable. When logic '1' is written into port latch then it works as input mode. It functions as simply I/O port and it does not have any alternative function.

Port 2 (p2.0 to p2.7):
It is 8-bit bi-directional I/O port. It is bit/ byte addressable. During external memory access it functions as higher order address bus (A8-A15).

Port 3(p3.0 to port 3.7):
It is 8-bit I/O port. In an alternating function each pins can be used as a special function I/O pin.

P3.0-RxD:
It is an Input signal. Through this I/P signal microcontroller receives serial data of serial communication circuit.

P3.1-TxD:
It is O/P signal of serial port. Through this signal data is transmitted.

P3.2- (INT0):
It is external hardware interrupt I/P signal. Through this user, programmer or peripheral interrupts to microcontroller.

P3.3-(INT1):
It is external hardware interrupt I/P signal. Through this user, programmer or peripheral interrupts to microcontroller.

P3.4- T0:
It is I/P signal to internal timer-0 circuit. External clock pulses can connects to timer-0 through this I/P signal.

P3.5-T1:
It is I/P signal to internal timer-1 circuit. External clock pulses can connects to timer-1 through this I/P signal.

P3.6-[WR (bar)]:
It is active low write O/P control signal. During External RAM (Data memory) access it is generated by microcontroller. when [WR(bar)]=0, then performs write operation.

P3.7-[RD (bar)]:
It is active low read O/P control signal. During External RAM (Data memory) access it is generated by microcontroller. when [RD(bar)]=0, then performs read operation from external RAM.

XTAL1 and XTAL2:
These are two I/P line for on-chip oscillator and clock generator circuit. A resonant network as quartz crystal is connected between these two pin. 8051 microcontroller also drives from external clock, then XTAL2 is used to drive 8051 from external clock and XTAL1 should be grounded.

[EA(bar)]/VPP:
It is and active low I/P to 8051 microcontroller. When (EA) = 0, then 8051 microcontroller access from external program memory (ROM) only. When (EA) = 1, then it access internal and external program memories (ROMS).

[PSEN(bar)]:
It is active low O/P signal. It is used to enable external program memory (ROM). When [PSEN(bar)]= 0, then external program memory becomes enabled and microcontroller read content of external memory location. Therefore it is connected to (OE) of external ROM. It is activated twice every external ROM memory cycle.

ALE: Address latch enable:
It is **active high O/P signal**. When it goes high, external address latch becomes enabling and lower address of external memory (RAM or ROM) latched into it. Thus it separates A0-A7 address from AD0-AD7. It provides properly timed signal to latch lower byte address. The ALE is activated twice in every machine cycle. If external RAM & ROM is not accessed, then ALE is activated at constant rate of 1/6 oscillator frequency, which can be used as a clock pulses for driving external devices.

RESET:
It is active high I/P signal. It should be maintained high for at least two machine cycle while oscillator is running then 8051 microcontroller resets.

3. Explain in detail about the 8051 interrupts? (May/june2013, Nov/Dec-14 & 16)

Interrupts are the events that temporarily suspend the main program, pass the control to the external sources and execute their task. It then passes the control to the main program where it had left off. 8051 has 5 interrupt signals, i.e. INT0, TFO, INT1, TF1, RI/TI.

Processing an Interrupt in the MCU

1. The main program is running.
2. An interrupt signal lets the MCU know that an event has occurred.
3. The MCU receives the interrupt signal, and suspends execution of the main program.
4. The MCU saves the current program execution state into its registers.
5. The MCU executes the interrupt routine corresponding to the received interrupt.
6. The MCU restores the saved program execution state.
7. Resume program execution.

The 8051 totally has **five interrupt sources**. They are.

1. INT0- External request from P3.2
2. Timer0-overflow from timer0 activates TF0
3. INT1-External request from P3.3
4. Timer1-Overflow from timer 1 activates TF1
5. Serial Port-Completion of transmission or reception of a serial frame activates TI or RI

Two interrupt registers are available in SFR namely IE & IP

1. IE- interrupt enable register
    - IE is used to enable or disable the interrupts
2. IP- interrupt priority register
    - IP is used to program the priority level

Priority interrupts:

S.No	Interrupts source	Priority level
1	External interrupt 0	Highest
2	Timer 0 overflow	Next highest priority
3	External interrupt 1	
4	Timer 1 overflow	
5	Serial port	Lowest priority

- External interrupt 0 has highest priority.
- Serial port has lowest priority.

These are scanned during each machine cycle.

- Priority is given during S6 of the machine cycle

- **Interrupt Control Register:-**

  i) **Interrupt enable Register:-**

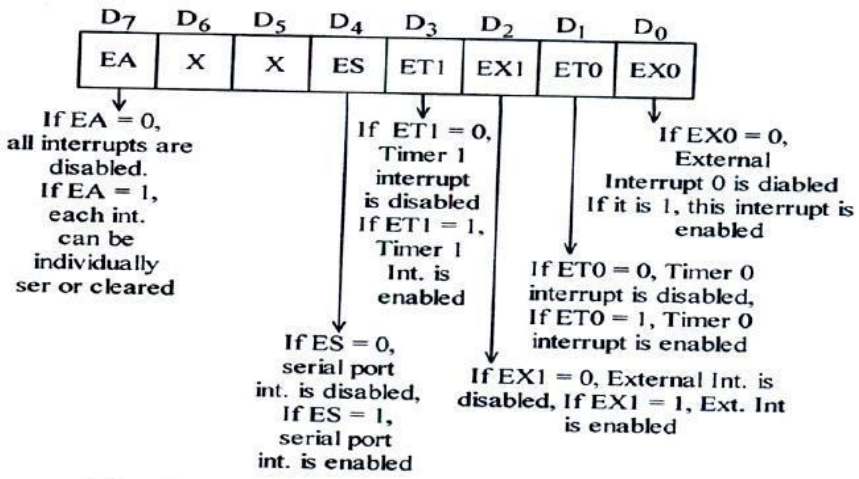


Fig: 5.13 – Bit pattern of Interrupt enable register

The bits of these registers can be set or cleared by the soft.

IE (Interrupt Enable) Register

This register is responsible for enabling and disabling the interrupt. EA register is set to one for enabling interrupts and set to 0 for disabling the interrupts. Its bit sequence and their meanings are shown in the following figure.

EA	-	-	ES	ET1	EX1	ET0	EX0

EA	IE.7	It disables all interrupts. When EA = 0 no interrupt will be acknowledged and EA = 1 enables the interrupt individually.
-	IE.6	Reserved for future use.
-	IE.5	Reserved for future use.
ES	IE.4	Enables/disables serial port interrupt.
ET1	IE.3	Enables/disables timer1 overflow interrupt.
EX1	IE.2	Enables/disables external interrupt1.
ET0	IE.1	Enables/disables timer0 overflow interrupt.
EX0	IE.0	Enables/disables external interrupt0.

IP (Interrupt Priority) Register

We can change the priority levels of the interrupts by changing the corresponding bit in the Interrupt Priority (IP) register as shown in the following figure.

- A low priority interrupt can only be interrupted by the high priority interrupt, but not interrupted by another low priority interrupt.
- If two interrupts of different priority levels are received simultaneously, the request of higher priority level is served.
- If the requests of the same priority levels are received simultaneously, then the internal polling sequence determines which request is to be serviced.

-	-	PT2	PS	PT1	PX1	PT0	PX0
bit7	bit6	bit5	bit4	bit3	bit2	bit1	

-	IP.6	Reserved for future use.
-	IP.5	Reserved for future use.
PS	IP.4	It defines the serial port interrupt priority level.
PT1	IP.3	It defines the timer interrupt of 1 priority.
PX1	IP.2	It defines the external interrupt priority level.
PT0	IP.1	It defines the timer0 interrupt priority level.
PX0	IP.0	It defines the external interrupt of 0 priority level.

5. Explain in detail about 8051 I/O ports (or) Explain the port operation of 8051(or) Explain various I/O ports and its functions of 8051 microcontroller (May/June 14,April/May-15,Nov/Dec-15,16) (April/May-18)

In 8051, totally **four ports** are available. These are $P_{0,}$ $P_{1,}$ $P_{1,}$ $P_{3.}$ These are 8bit pots. If the first 0 is written to a port, it will become output port. If the first 1 is written to a port it will become a input port.

PORT 0

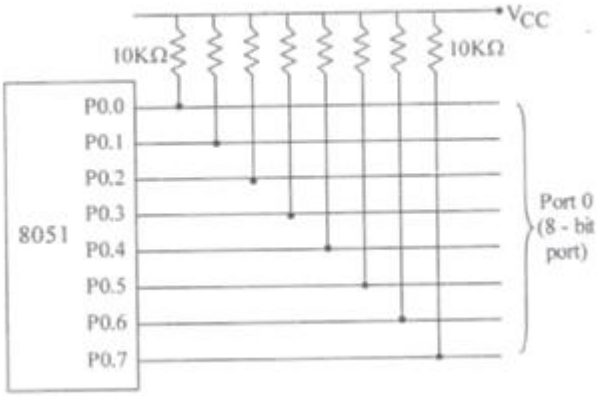
- It can be used as input or output port. Each pin must be connected externally to a 10kΩ pull-up resistors, because, P_0 is an open drain bidirectional I/O port.
- It can be used in two ways,
  1. It can be used as simple I/O port
  2. It can be used for external memory interface for lower order address and data bus

PORT 0 as input:

- If we want to use port 0 as input port then, the port is to be programmed by writing 1 to all the bits. By using the following program, port0 is configured as input port

```
MOV A, # 0FFh;                  /*A= 0FFh
MOV P0, A;                      /* P0 is used as input port by writing 1's to it
LOOP MOV A, P0 A ← P0,          /* get data from port
MOV P0, A;     P1 ← A,          /*send it to P1 SJMP
LOOP;
```

PORT 0 as output port, then 0 is written to a port

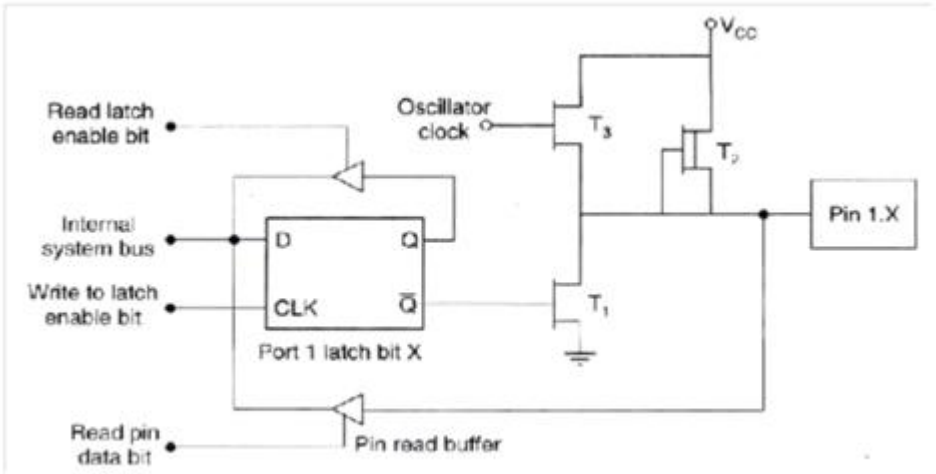


PORT 1

- It is not necessary to connect any pull-up resistors. Because, it is having pull-up resistors already

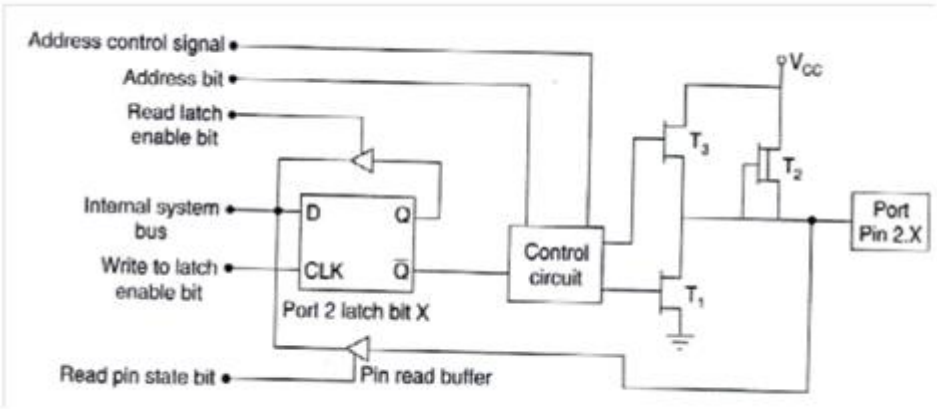
PORT1 As input: Upon rest, port1 is configured as input port.

- If **it is used as output port**, then 0 is written to a port. then, once again , if we want to use it as a input port, then 1 is written to all its bit



PORT2

- It is also a 8 bit port. It can be used as I/O. it is not necessary for any pull-up resistors

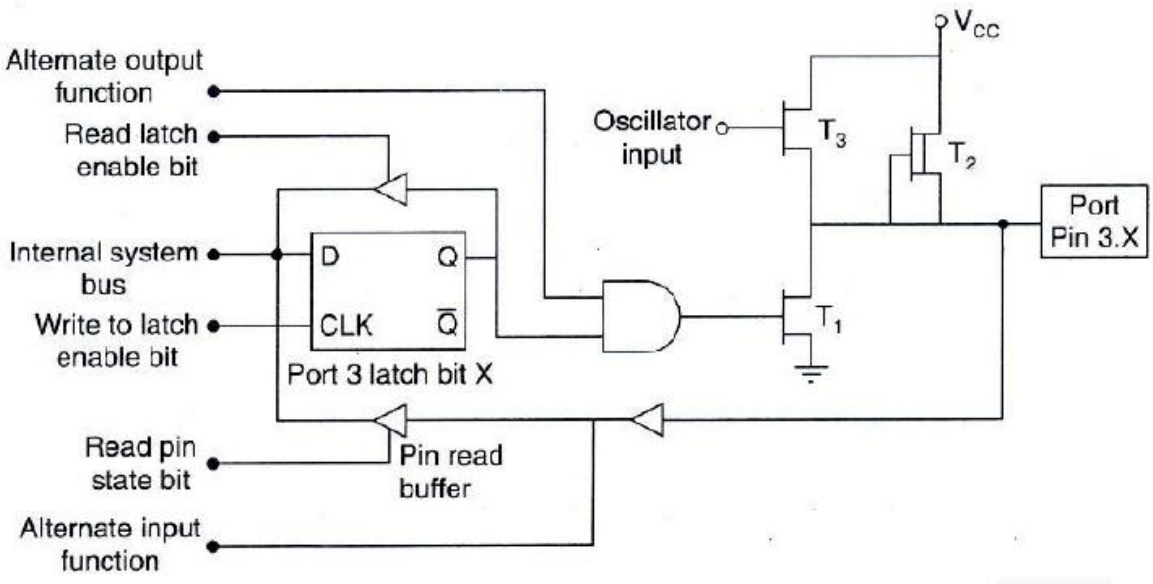


PORT2 As input:

```
        MOV A, #0FFh;    /*A= OFFh
        MOV P2, A;       /*P2 is configured as input port by setting port pins as 1's
LOOP    MOV A, P2        /*get the data from P2
        MOV P1, A;       /*send data to port1
        SJMP LOOP;
```

PORT3

- It can be used as input or output port.
- It is not necessary to connect any pull-up resistors



- Port3 pins can be used for doing alternate functions. These functions are given below.

P3.0	Serial data input (RXD)
P3.1	Serial data output (TXD)
P3.2	External interrupt 0 (INTO)
P3.3	External interrupt 1 (INT1)
P3.4	Timer 0 external input (TO)
P3.5	Timer 1 external input (Tl)
P3.6	External data memory write strobe (WR)
P3.7	External data memory read strobe (RD)

6. Explain detail about the timing diagram of 8051 microcontroller?(Nov/Dec-14)

The timing diagram of 8051 is represented as,

P_1 = phase 1

P_2 = phase 2

S_1 = State

- Each machine cycle consist of 6 states namely S_1 S_2 S_3…… S

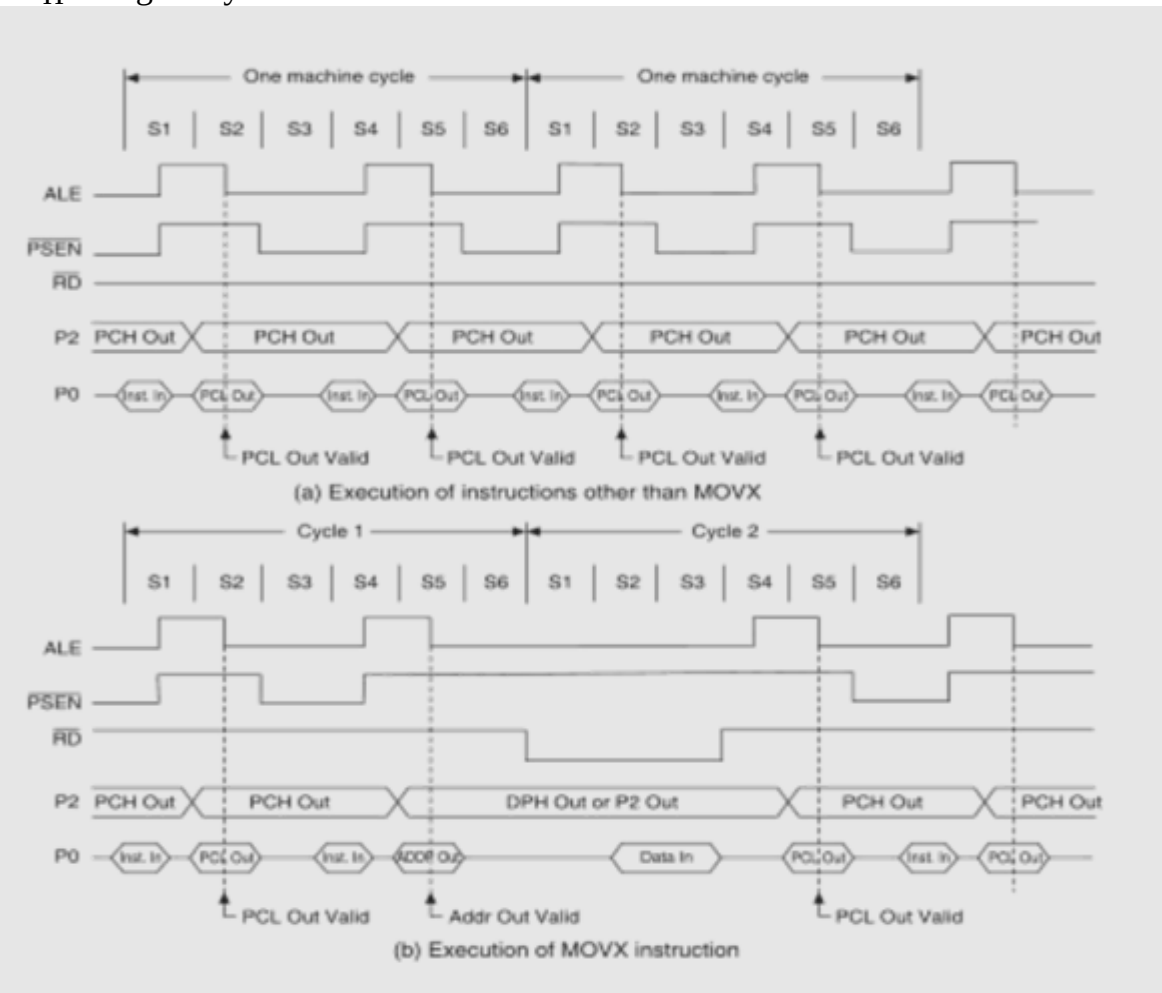
- Generally, arithmetic & logical operations take place during phase 1
- Internal register-to-register transfer take place during phase 2v
- Timing diagram for various instruction is shown below
- ALE signal is activated during $S_1 P_2$ & $S_2 P_1$ and its activated once during $S_4 P_2$ & $S_5 P_1$

P_0 = Port 0

P_2 = port 2

PC_L = Low byte of PC

PC_H = Higher byte of PC



The timing diagram of the MOVX instruction is shown above

- Instruction execution from external program memory is shown above in the timing diagram
- RD signal is also shown in the above timing diagram

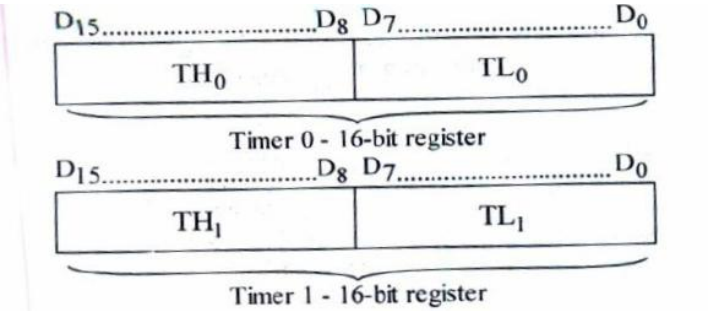
7. Explain in detail about 8051 timers/Counter? (Nov/Dec 2013)(Apr/May-17)

Two numbers of timers are available in 8051.

- These are timer 0 & timer 1.
- These are 16 bit timers. But, 8051 is a 8-bit micro controller. So, timer register is divided into 2 parts, namely TL_0, TH_0 & TL_1, and TH1.

1. TL_0 – lower byte of timer 0.
2. TH_0 – higher byte of timer 0.

3. TL_1 – lower byte of timer 1.
4. TH_1 – higher byte of timer 1.



TMOD register

Timer mod register is known as TMOD. It is used to set various timer operating modes for timer 0 and timer1.

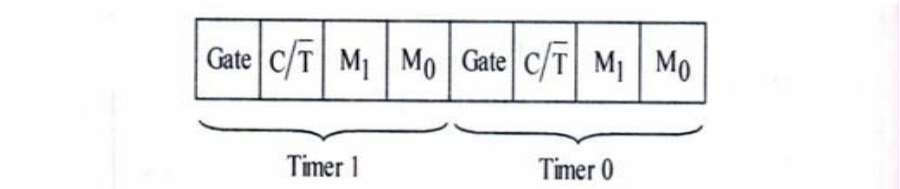


Fig: 5.16 - T MOD register bit pattern

M_1	M_0	Mode	Description
0	0	0	13-bit Timer mode. 8 bits of THX & 5 bits of TLX are used.
0	1	1	16 bit Timer mode
1	0	2	8 bit auto reload. THX has value to be reloaded to TLX whenever it overflows
1	1	3	Split timer mode

M_0, M_1 these bits are used to select mode. Separate M_0, M_1 will be available for timer 0 & timer1

C/T – counter / Timer

1. If C/*T* = 0 it is used as timer

2. If C/*T* = 1 it is used as counter

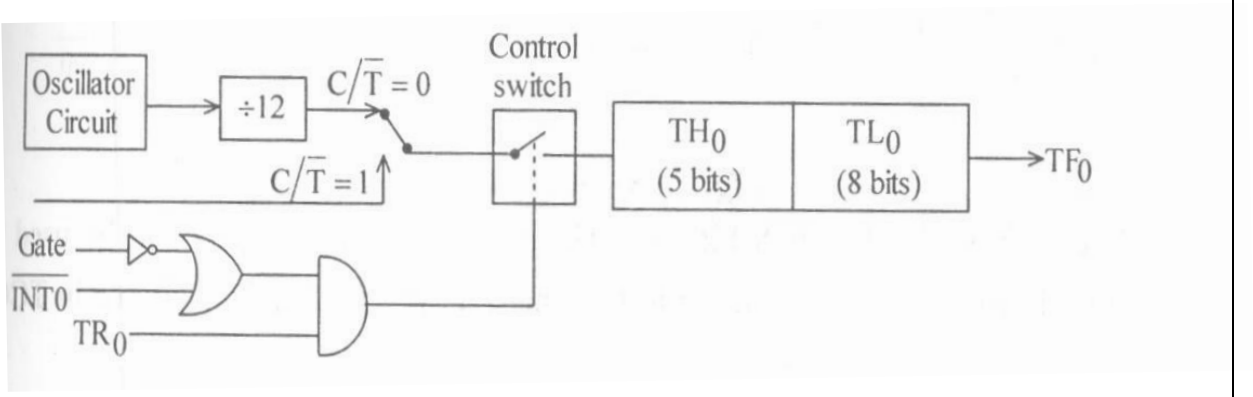
- Gate- if gate = 0, software way of controlling is enabled.
- The start and stop of the timer will be controlled by software using TR_0, TR_0.

E.g SETB TR_1
CLR TR_1
SETB TR_1
CLR TR_1

If gate = 1 hardware way of control is enable

Modes of timer

Mode 0:



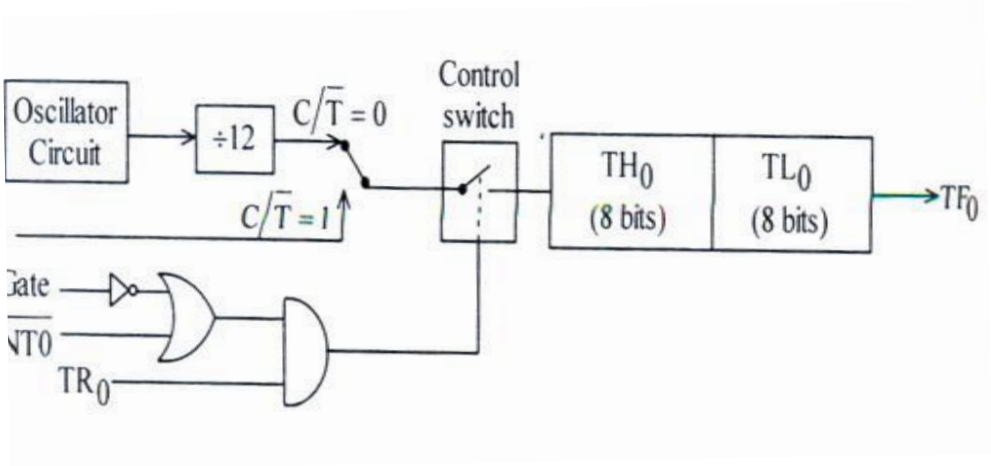
- Here TL_0 has 8 bits
- TH_0 has lower most 5 bits so 13 bit register is used.
- If C/*T* = 0 it acts as timer.
- The register is incremented as every machine cycle for that control switch is to be closed.
- The output of the AND gate to be high to close the switch.

Case 1: if TR_0 =1, gate = 0, INT0= 0

1. If TR_0 =1, (TR_0 = timer run control bit) timer0 is turned ON.
2. If gate=0 by using NOT gate, 1 is applied to OR gate. Here, INT0=0 so the output of the OR gate is 1. If TR_0 =1 then the output of the AND gate is 1 then the control switch is closed.
3. If C/*T* =1 then it is used as counter. So 8bits of TL_0 and 5bits of TH_0 from 13bit register is used for counting. Initially all bits are 1's if these 1's are changed into 0's then timer overflow flag is set (TF_0=1).
4. To operate timer1 in mode 0 then TH_0, TL_0 & TR_1 are used.

Case 2: if TR_0 =1, gate = 1, INT0= 1

1. If gate=1 then 0 is applied to OR gate. INT0=1 so the output of OR gate is 1, if TR_0 =1 then timer0 is turned ON. The same functions that are explained in case1 will be carried out.

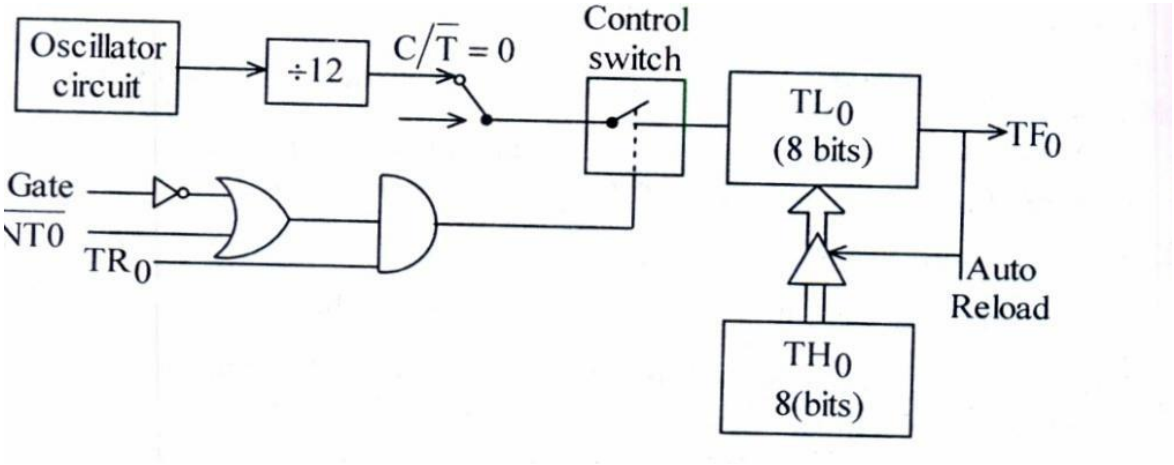


Mode 1:

It is same as mode 0 here **8bits of TL_0, and 8bits of TH_0** are used; it is also used for counting. If all 16 bits go from 1's to 0's then, TF_0 =1, TF_0 is available in TCON register.

Mode2:

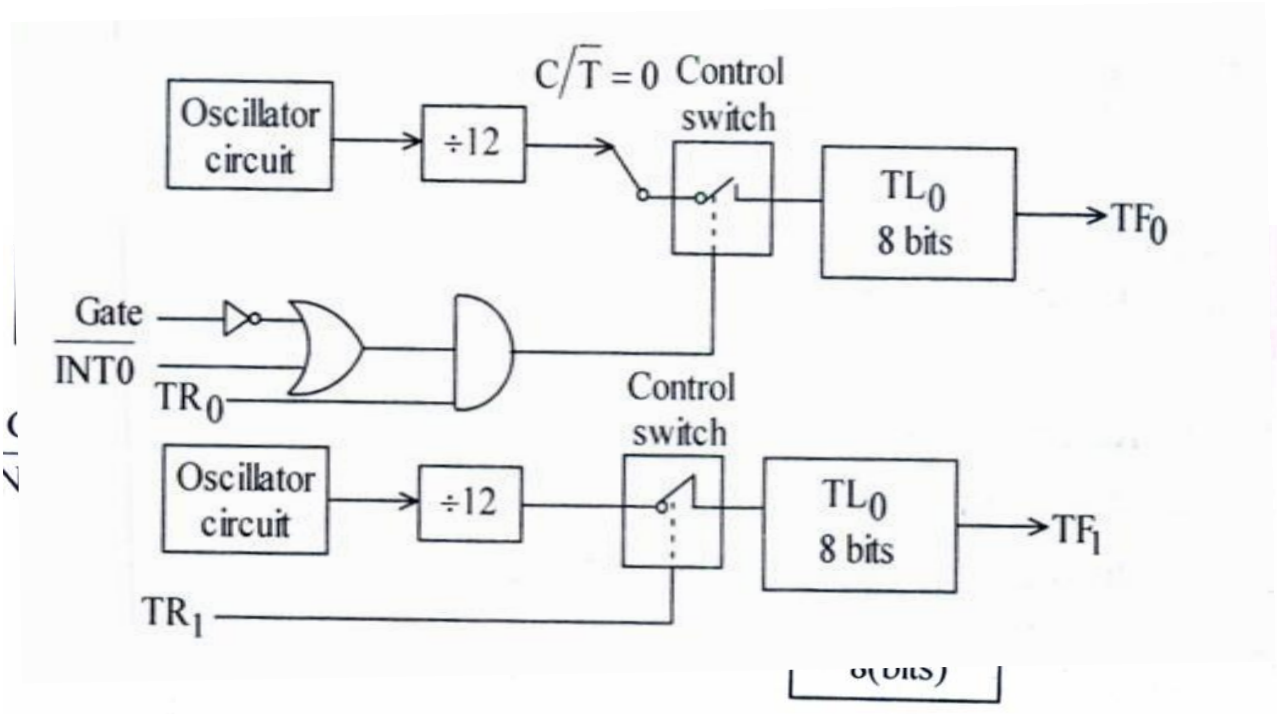
This mode is known as auto reload mode. As in previous modes, if the bits of TL_0 go from 1's to 0's then, timer overflow flag is set (TF_0=1). Then the content of TH_0 is loaded into TL_0 then the counting is started.



Mode3

Timer0 & timer1 can be operated in mode0, mode1&mode2. But mode3 is possible for timer0 and not for timer1. TL_0 & TH_0 became two separate counters.

TL_0 is controlled by the gate and timer overflow flag. TF_0 is set whenever all bits in TL_0 go from 1's to 0's.



T CON – Timer control register:

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

IT0; Interrupt 0 control bit: If IT_0=1, then, INT0 is triggered at falling edge. If IT_0=1, then, INT0 is low level triggered.

IE0; Interrupt 0 edge flag : If external interrupt edge is detected then it is set. If the interrupt is processed then it is reset.

IT1 ;(Interrupt 1 control bit): if IT_1=1, then, INT1 is triggered by falling edge. If IT_1=0, then, INT1 is low level triggered.

IE1 ;(Interrupt 1 edge flag) : If external interrupt edge is detected then it is set. If the interrupt is processed then it is reset.

TR0; Timer 0 run control bit: if TR_0=1 timer0 is ON. if TR_0=0 timer0 is OFF

TF0; Timer 0 overflow bit: if TF_0=1, timer overflow occur

TR1; Timer 1 run control bit : if TR_1=1 timer1 is ON. if TR_1=0 timer1 is OFF

TF1; Timer 1 overflow flag: if TF_1=1, timer overflow occur

8. What are the modes in which 8051 can be configured for serial transmission and explain? (Apr/may 2011) (Nov/Dec 2017)

The 8051 microcontroller chip has got all the circuitry in it for serial transmission. The RXD pin is used to receive the input serially and the TXD pin is used to transmit the data serially.

The serial communication is full duplex, meaning that the 8051 can receive and transmit at the same time. The receiving unit is buffered as well. Thus, the reception of the second byte or the frame data can start even before the first byte is received by the CPU.

The serial port in the 8051 can be configured into four different modes, namely mode 0 to mode 3, depending on the application.

Mode 0:

- In this mode, the 8051 receives and transmits through the RXD pin. The TXD outputs the shift clock.
- 8 bits of data are received or transmitted. While transmitting, the LSB of the byte is sent out first. Similarly, while receiving, the LSB is received first.
- The baud rate in this mode is constant and it is l/12th of the oscillator frequency.
- Once all the 8 bits of the data byte are transmitted, the Transmit Interrupt (TI) flag is set and an interrupt is generated. Similarly, after receiving all the 8 bits, the Receive Interrupt (RI) flag is set and an interrupt is generated.
- These TI and RI flags are nothing but the two bits in the SCON register. We shall later describe this register in detail.

Mode 1:

- In this mode, the 8051 transmits 10 bits of information through TXD and receives 10 bits of information through RXD.
- The first bit is the start bit followed by the 8 bits of data (LSB first) and then a stop bit (high). Once the stop bit is received, it means that the reception of one frame is complete, that is, a byte of data has come. This stop bit is loaded into a bit called RB8 in the SCON register .

• As in mode 0, here also an interrupt is generated once all the bits in a frame are received or transmitted. But unlike mode 0, here the baud rate is variable.
The stop bit and the start bit are automatically added by the 8051 CPU through hardware while transmitting the data.

Mode 2:

In this case, 11 bits are transmitted or received. This 11-bit frame is classified as shown below.

1. 1 bit for start
2. 8 bits for data
3. 1 bit can be programmed
4. 1 bit for stop

- The 9th data bit is programmable. This 9th bit is nothing but the TB8 bit in the SCON register. This bit can be used effectively while transmitting the data, i.e. with little software overhead; this bit can be used to send the parity of the byte that is to be transmitted.
- Transmitting: If we want to use the 9th bit as parity, we must load the TB8 with the parity of the byte that is to be transmitted. ACC has the byte to be transmitted.
- Receiving: On reception of a frame, the 9th bit goes to RB8 of the SCON register. The stop bit is ignored. The baud rate, in this mode, is programmable either l/32th or l/64th of the oscillator frequency.

Mode 3:

This mode is same as the mode 2 except for the baud rate. Here the baud rate is variable. Timer **1** is used as the baud rate generator.

Serial control register (SCON)

This SFR controls the operations of the serial port. This register is used to define the operating modes. This also receives the 9th bit and contains the transmit and receive interrupt flags as well.

SCON: Serial Control Register:

7	6	5	4	3	2	1	0
SMO	SMI	SM2	REN	TB8	RB8	Tl	Rl

SMO and SMI define the mode:

SMO	SMI	Mode	Description	Baud rate
0	0	0	Shift Register	1/12th Oscillator Frequency
0	1	1	8-bit UART	Variable
1	0	2	9-bit UART	1/64 Oscillator Frequency if SMOD 1/32 Oscillator Frequency if SMOD
1	1	3	9-bit UART	Variable

SM2: Special feature for multiprocessor communication in mode 2 and mode 3. In mode 2 and mode 3, if SM2 = **1,** the Rl will not be activated unless the 9th bit received (RB8) is **1.** In

mode **1,** if SM2 = **1,** the Rl will not be activated unless a valid stop bit is received. In mode **0,** SM2 should be reset.
REN: Enables the serial reception. If set, it enables the reception, otherwise the reception is disabled.
TB8: It is the 9th bit of the data that is to be transmitted. Set and reset by software. RB8: In modes 2 and 3, it is the 9th bit received. In mode **1,** if SM2 = **0,** RB8 is the stop bit that is received. In mode **0,** it is not used.

Power control register (PCON)

It is a special function register through which certain power control functions in CMOS version of the 8051 are implemented. In the HMOS version, all the bits of PCON except bit 7 are dummy.
Bit 7 is SMOD and is used in both CMOS and HMOS versions to double the baud rate in modes 1, 2 and 3. PCON is not bit addressable.

Baud rate

The baud rate, in mode 0, is fixed at 1/12th of the oscillator frequency. The baud rate in mode 2 is either l/64th or l/32th of the oscillator frequency depending on the bit value of SMOD in the special function register PCON.
If SMOD = 0, the baud rate is 1 64th of the oscillator frequency. If SMOD = 1, the baud rate is l/32th of the oscillator frequency.
For mode 1 and mode 3, the baud rates are variable. The baud rate is determined by the Timer 1 overflow rate, i.e.

$$\text{Baud rate} = \frac{\text{Timer 1 overflow rate}}{n}$$

where n is an integer and its value is either 32 or 16, i.e.
if SMOD = 1, then n = 16 else n = 32.
In this case, Timer 1 can be configured in any mode. It is advisable to keep Timer 1 in mode 2 which is also the auto reload 8-bit counter. Here, the overflow rate will depend upon the reload value in the TH1.

9. Explain the RAM structure of 8051 microcontroller.(Nov/Dec-16) (Nov/Dec 2017)

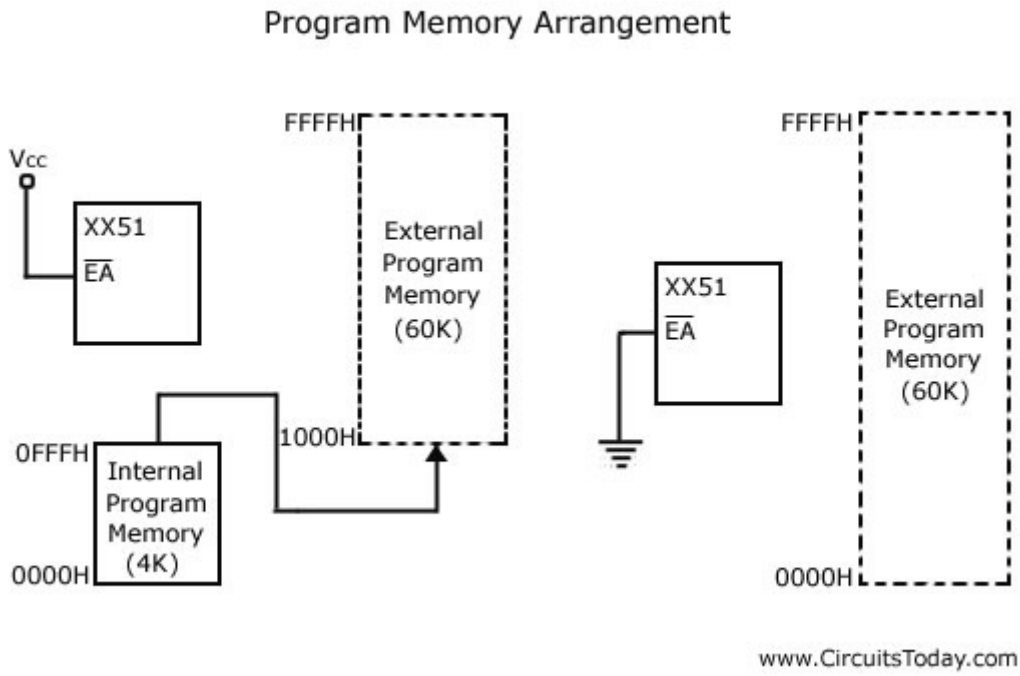
The organization of memory depends on two architecture types that are:

- **Von Neumann architecture:** The architecture also known as Von Neumann model or Princeton architecture comprises both, address as well as data memories, on a single unit.

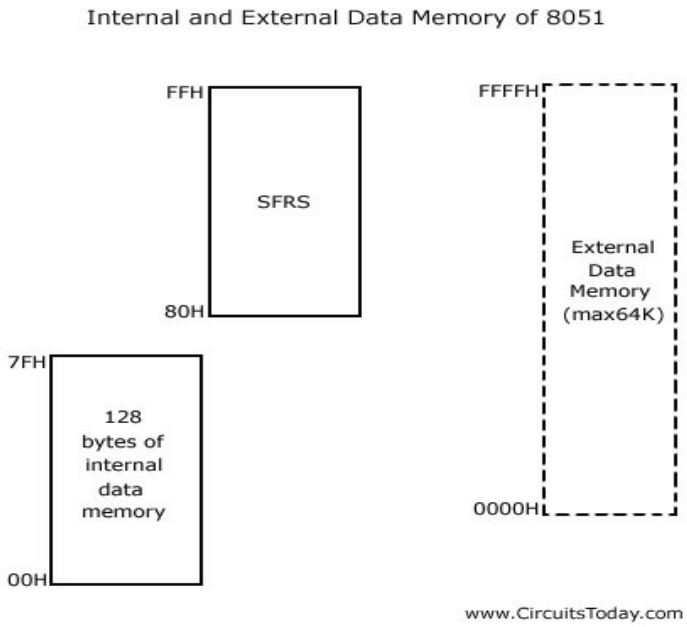
- **Harvard architecture:** In this architectural type, both the memories, address and data, are treated as separate units.

8051 Microcontroller's memory can be categorized into Program Memory and Data Memory, on the broad level. While, Program Memory is the ROM of 8051 Microcontroller; the Data Memory is RAM. Following is a brief description of the two memory types:

Program Memory (ROM): This memory type saves the executed code of a program permanently. Default size of the memory on chip is 4K; however, depending on the requirement, you can add external memory upto 60K in size. Thus, total Program Memory available on 8051 Microcontroller is 64K in size. The transfer process of executed program from default to external memory is automatic in nature. You must also note that handling of external memory is decided by the logic state of EA (External Access) pin.



- **Data Memory (RAM):** This memory type stores the code of executed program temporarily. While, internal data RAM is 128 bytes in size; it can have external memory upto 64K in size. Thus, total Data Memory available on 8051 Microcontroller is 64K + 128 bytes in size. Further, internal Data Memory RAM is divided in three parts, namely, Register Banks, Bit Addressable Area, and Scratch Pad Area.



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) .

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.

ANNA UNIVERSITY QUESTIONS

PART-A:

1. List the features of 8051 microcontroller**?(May/June 2012))**
2. Explain the interrupts of 8051 microcontroller**?(Nov/Dec2013)(May/June 2014)**
3. List the addressing modes of 8051**? (Nov/Dec 2011,Nov/Dec-14)**
4. Give the alternate functions for the port pins of port3**?(Apr/May2011)**
5. Explain the 16-bit registers DPTR and SP of 8051**.(Apr/May 2011)**
6. Name the special functions registers available in 8051**.(May/June 2013, Nov/Dec-14)**
7. Explain the function (purpose) of the pins and of 8051. **(May/June 14)(Nov-16)**
8. Write down the different operating modes for serial communication of 8051. **(Nov/Dec-14)**
9. List the interrupts of 8051(Nov/Dec-15,16)
10. Write the function of TMOD register in 8051 microcontroller(Nov/Dec-15)

PART-B

1. Explain the architecture of 8051 microcontroller**? (May/June2012,Nov/Dec-14, April/May-15, 17, 2018) or** Explain with block diagram, how to access external memory devices in an 8051 based system. **(Nov/Dec-17)**
2. Explain in detail about the 8051 interrupts**? (May/June2013,Nov/Dec-14&16,April/May15)**
3. Explain the I/O port operation of 8051?**(May/June 2014, April/May-15,2018) (Nov/Dec-15,16,**)
4. Explain detail about the timing diagram of 8051 microcontroller?**(Nov/Dec-14)**
5. Explain the Timers of 8051 microcontroller with appropriate diagram**(Nov/Dec-15)(Apr/May-17)**
6. Explain the RAM structure of 8051 microcontroller**.(Nov/Dec-16) (Nov/Dec 2017)**
7. Discuss in detail, the hardware and software support provided by 8051 for serial communication. **(Nov/Dec-17)**

MAILAM ENGINEERINGCOLLEGE
MAILAM, 604304

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

Sub.Code/Sub.Name: EE6502 MICROPROCESSORS AND MICROCONTROLLERS Year / Sem: III/ V

UNIT IV-PERIPHERAL INTERFACING

SYLLABUS

Study on need, Architecture, configuration and interfacing, with ICs: 8255, 8259, 8254,8237, 8251, 8279 ,- A/D and D/A converters &Interfacing with 8085& 8051.

Updated Question:

PART-A

Q.No.51	Page No: 11	(Nov/Dec 2017)
Q.No.52	Page No: 11	(Nov/Dec 2017)
Q.No.02	Page No: 02	(Apr/May 2018)
Q.No.50	Page No: 11	(Apr/May 2018)

PART-B

Q.No.01	Page No: 11	(Nov/Dec 2017)
Q.No.03	Page No: 21	(Nov/Dec 2017)
Q.No.01	Page No: 11	(Apr/May 2018)
Q.No.04	Page No: 23	(Apr/May 2018)

TEXT BOOKS:

1. Krishna Kant, "Microprocessor and Microcontrollers", Eastern Company Edition, Prentice Hall of India, New Delhi , 2007.
2. R.S. Gaonkar, 'Microprocessor Architecture Programming and Application', with 8085, Wiley Eastern Ltd., New Delhi, 2013.
3. Soumitra Kumar Mandal, Microprocessor & Microcontroller Architecture, Programming & Interfacing using 8085,8086,8051,McGraw Hill Edu,2013.

REFERENCES:

1. Muhammad Ali Mazidi & Janice Gilli Mazidi, R.D.Kinely 'The 8051 Micro Controller and Embedded Systems', PHI Pearson Education, 5th Indian reprint, 2003.
2. N.Senthil Kumar, M.Saravanan, S.Jeevananthan, 'Microprocessors and Microcontrollers', Oxford,2013.
3. Valder – Perez, "Microcontroller – Fundamentals and Applications with Pic," Yeesdee Publishers, Tayler & Francis, 2013.

Prepared By
1. Mrs.J.Srivandhana AP/MCA
2. Mrs.T.Kala AP/MCA

Verified By
HOD/MCA

Approved By
Principal

PART-A **UNIT-IV**

1. What are the signals used in input control signal and output control signals?

Input control signal

STB (strobe input)
IBF (Input buffer full)
INTR (Interrupt request)Output control signal
OBF (Output buffer full)
ACK (Acknowledge input)
INTR(Interrupt request)

2. What are the modes of operations used in 8253/8254? (Nov/Dec-14)(Nov/Dec-15)(Apr/May 17)

Each of the three counters of 8253 can be operated in one of the following six modes of operation.

i). Mode 0 (Interrupt on terminal count)
ii). Mode 1 (Programmable monoshot)
iii). Mode 2 (Rate generator)
iv). Mode 3 (Square wave generator) v).
Mode 4 (Software triggered strobe)
vi).Mode 5 (Hardware triggered strobe)

3. What are the different types of write operations used in 8253?

There are two types of write operations in 8253

i) Writing a control word register
ii) Writing a count value into a count register

The control word register contents are used for

i) Initializing the operating modes (mode 0-mode4)
ii) Selection of counters (counter 0- counter 2)
iii) Choosing binary /BCD counters
iv) Loading of the counter registers.

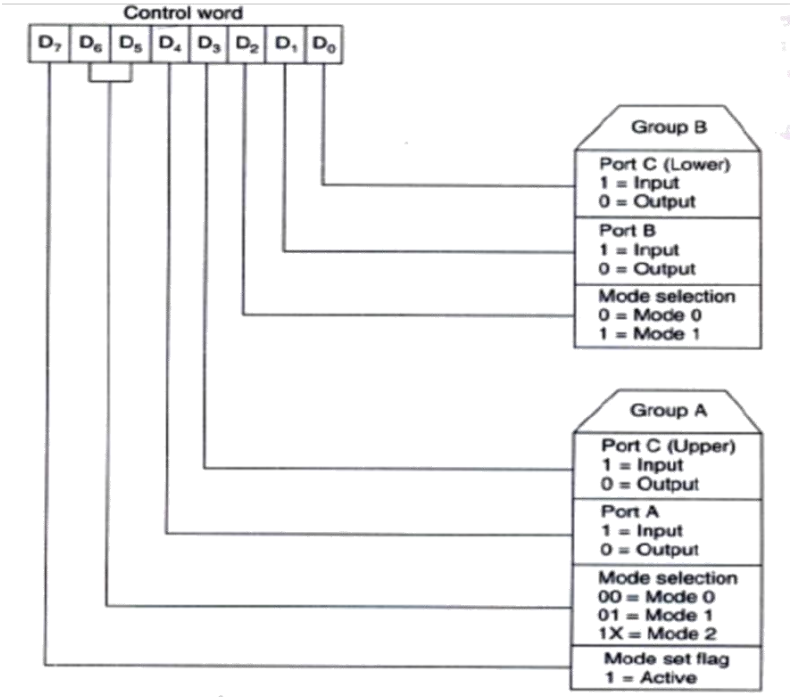
The mode control register is a write only register and the CPU cannot read its contents.

4. Give the different types of command words used in 8259?

The command words of 8259A are classified in two groups

i) Initialization command words (ICWs)
ii) Operation command words (OCWs)

5. Draw the command word format of 8255 in I/O mode. (Nov/Dec 16)



6. Define scan counter? (April/May 2011)

The scan counter has two modes to scan the key matrix and refresh the display. In the encoded mode, the counter provides binary count that is to be externally decoded to provide the scan lines for keyboard and display. In the decoded scan mode, the counter internally decodes the least significant 2 bits and provides a decoded 1 out of 4 scan on SL0-SL3.The keyboard and display both are in the same mode at a time.

7. 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.

8. 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.

9. What are the modes used in keyboard modes?

1. Scanned Keyboard mode with 2 Key Lockout.
2. Scanned Keyboard with N-key Rollover.
3. Scanned Keyboard special Error Mode.
4. Sensor Matrix Mode.

10. What are the modes used in display modes?

1. Left Entry mode

In the left entry mode, the data is entered from the left side of the display

unit. 2. Right Entry Mode.

In the right entry mode, the first entry to be displayed is entered on the rightmost display.

11. What is the use of modem control unit in 8251?

The modem control unit handles the modem handshake signals to coordinate the communication between the modem and the USART.

12. What is interfacing?

An interface is a shared boundary between the devices which involves sharing information. Interfacing is the process of making two different systems communicates with each other.

13. List the operation modes of 8255. (Nov/Dec 2013)

a) I/O Mode: i. Mode 0-Simple Input/Output.

ii. Mode 1-Strobed Input/Output (Handshake mode)

iii. Mode 2-Strobed bidirectional mode

b) Bit Set/Reset Mode.

14. What is meant by cascading in 8259? (Apr/May 17)

- The cascade pins (CAS0, CAS1 and CAS2) from the master are connected to the corresponding pins of the slave.
- For the slave 8259, the SP (low) / EN (low) pin is tied low to let the device know that it is a slave.
- The SP (low) / EN (low) pin can be used as input or output signal.
- In non-buffered mode it is used as input signal and tied to logic-I in master 8259 and logic-0 in slave 8259.
- In buffered mode it is used as output signal to disable the data buffers while data is transferred from 8259A to the CPU

15. What is a control word?

It is a word stored in a register (control register) used to control the operation of a program digital device.

16 . What is the purpose of control word written to control register in 8255?

The control words written to control register specify an I/O function for each I.O port. The bit D7 of the control word determines either the I/O function of the BSR function.

17. What is the size of ports in 8255?

Port-A : 8-bits

Port-B : 8-bits

Port-CU : 4-bits

Port-CL : 4-bits

18. Distinguish between the memories mapped I/O peripheral I/O?

Memory Mapped I/O	Peripheral MappedI/O
16-bit device address	8-bit device address
Data transfer between any general-purpose register and I/O port.	Data is transfer only between accumulator and I.O port
The memory map (64K) is shared between I/O device and system memory.	The I/O map is independent of 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

. **19. What is an USART? (Nov/Dec-14)**

USART stands for universal synchronous/Asynchronous Receiver/Transmitter. It is a programmable communication interface that can communicate by using either synchronous or asynchronous serial data.

20. What is I/O mapping?

The assignment of addresses to various I/O devices in the memory chip is called as I/O mapping.

21. What is the use of 8251 chip?

8251 chip is mainly used as the asynchronous serial interface between the processor and the external equipment.

22. What is 8279? Mention its features.

Programmable Keyboard/Display interface.

The main features of 8279 are as follows:

1. Simultaneous keyboard and display operation is provided.
2. Keyboard can be read in three modes.
   Scanned Keyboard Mode
   Scanned Sensor Mode strobed
   Input Entry Mode
3. It has an 8-character keyboard FIFO
4. Interrupt is generated when key is entered.
5. There is 8-dual or 16—Numerical Display 7. 16 x 8 display ram is displayed
6. The display can be in different modes.
   - Left Entry
   - Right Entry
7. It has programmable scan timing
8. It relieves the CPO from repetitive scanning of the keyboard and repetitive data Output

23. List the major components of the keyboard/Display interface.

a. Keyboard section b. Scan section

c. Display section d. CPU interface section

24. What is Key bouncing? (May/June 12)

Mechanical switches are used as keys in most of the keyboards. When a key is pressed the contact bounce back and forth and settle down only after a small time delay (about 20ms). Even though a key is actuated once, it will appear to have been actuated several times. This problem is called Key Bouncing. Microprocessor must wait until the key reach to a steady state; this is known as Key bounce.

25. Define HRQ?

The hold request output requests the access of the system bus. In non- cascaded 8257 systems, this is connected with HOLD pin of CPU. In cascade mode, this pin of a slave is connected with a DRQ input line of the master 8257, while that of the master is connected with HOLD input of the CPU.

26. What is the use of stepper motor?

A stepper motor is a device used to obtain an accurate position control of rotating shafts. A stepper motor employs rotation of its shaft in terms of steps, rather than continuous rotation as in case of AC or DC motor.

27. What is TXD & RXD?

TXD- Transmitter Data Output This output pin carries serial stream of the transmitted data bits along with other information like start bit, stop bits and priority bit.

RXD- Receive Data Input This input pin of 8251A receives a composite stream of the data to be received by 8251A.

28. What is swapping?

The procedure of fetching the chosen program segments or data from the secondary storage into the physical memory is called 'swapping'.

29. Write the function of crossbar switch?

The crossbar switch provides the inter connection paths between the memory module and the processor. Each node of the crossbar represents a bus switch. All these nodes may be controlled by one of these processors or by a separate one altogether.

30. What is a data amplifier?

Transceivers are the bi-directional buffers are sometimes they are called as data amplifiers. They are required to separate the valid data from the time multiplexed address data signal. They are controlled by 2 signals i.e DEN & DT/R.

31. What is the need for D/A converter? (May/June 12)

D/A converter are used to convert digital quantity to analog signal. D/A converter produces an o/p current or voltage proportional to digital quantity applied to its i/p. D/A converter are also used as a port 0

32. What are the internal registers available in 8259 PIC? (April/May-15)

The Internal Registers Available in 8259 PIC are as follows

1. Interrupt service register (ISR)
2. Priority resolver
3. Interrupt request register (IRR)
4. Interrupt mask register (IMR)

33. What are the commonly used ADC's and DAC's?

The different types of ADC are successive approximation ADC, counter type ADC flash type ADC, integrator converters and voltage to-frequency converters.

DAC R/2R ladder type

34. What is the use of handshaking signals?

It is used to data transfer method which is used when the speed of an I/O device does not match with the speed of the microprocessor. Asynchronous data transfer is also called as Handshaking.

35. What is the use of modem control unit in 8251?

The modem control unit handles the modem handshake signals to coordinate the communication between the modem and the USART.

36. What is the use of 8251 chip?

Intel's 8251A is a universal synchronous asynchronous receiver and transmitter compatible with Intel's Processors. This may be programmed to operate in any of the serial communication modes built into it. This chip converts the parallel data into a serial stream of bits suitable for serial

transmission. It is also able to receive a serial stream of bits and converts it in to parallel data bytes to be read by a microprocessor.

37. What are the different types of methods used for data transmission?

The data transmission between two points involves unidirectional or bi-directional transmission of meaningful digital data through a medium. There are basically there modes of data transmission

(a) Simplex
(b) Duplex
(c) Half Duplex

In simplex mode, data is transmitted only in one direction over a single communication channel. For example, a computer (CPU) may transmit data for a CRT display unit in this mode. In duplex mode, data may be transferred between two trans receivers in both directions simultaneously.

In half duplex mode, on the other hand, data transmission may take place in either direction, but at a time data may be transmitted only in one direction.

38. What are the functional types used in control words of 8251?

The control words of 8251A are divided into two functional types.

1. Mode Instruction control word
2. Command Instruction control word

Mode Instruction control word:-This defines the general operational characteristics of 8251A. Command Instruction control word:-The command instruction controls the actual operations of the selected format like enable transmit/receive, error reset and modem control.

39. What is the need for 8259 PIC? (Nov/Dec 2013)

By connecting priority Interrupt controller, it is possible to increase the interrupt handling capacity of the microprocessor. The 8259A is a commonly used priority interrupt controller, which is specifically designed for use with interrupt signals INTR and of Intel series.

40. What are the different peripheral interfacing used with 8085 microprocessor? (May/June 13)

8255 PPI, 8259 PIC, 8251 USART, 8279 Key board display controller and 8253 Timer/ Counter – Interfacing with 8085 - A/D and D/A converter interfacing.

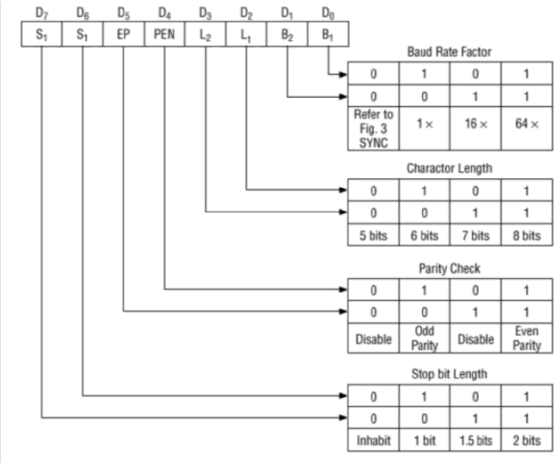
41. What are the applications of D/A converter interfacing with 8255? (May/June 12)

Microprocessor based process control system

Stepper motor interfacing

Centronic interfacing

42. Draw the 'Mode Word' format of 8251 USART. (Nov/Dec 2011)



43. State the use of ISR and PR registers in 8259 PIC. (Nov/Dec 2011)

Interrupt Service Register (ISR)

The Interrupt Service Register (ISR) stores all the levels that are currently being serviced.

Priority Resolver

The priority resolver determines the priorities of the bits set in the IRR. The bit corresponding to the highest priority interrupt input is set in the ISR during the input.

44. What are the different ways to end the interrupt execution in 8259 programmable Interrupt controller? (April/May 2011)

1. fully nested mode
2. special fully nested mode
3. Rotating priority mode
4. special masked mode and
5. Polled mode.

45. What are the output terminals in USART 8251? (May/June 13)

TXD (output terminal)

TXRDY (output terminal)

TXEMPTY (Output terminal)

RXRDY (Output terminal)

SYNDET/BD (Input or output terminal)

DTR (Output terminal)

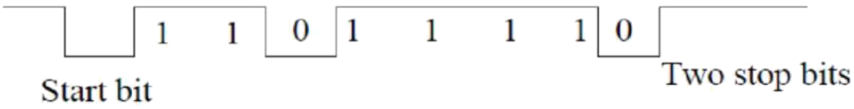
RTS (Output terminal)

46. What are the features of 8259? (May/June 14) (Nov/Dec 16)

1. It is programmed to work with either 8085 or 8086 processor.
2. It manages 8-interrupts according to the instructions written into its control registers.
3. In 8086 processor, it supplies the type number of the interrupt and the type number is programmable. In 8085 processor, the interrupt vector address is programmable. The priorities of the interrupts are programmable.
4. The interrupts can be masked or unmasked individually.
5. The 8259s can be cascaded to accept a maximum of 64 interrupts.

47. How data is transmitted in asynchronous serial transmission? (May/June 14)

Data to be transmitted is sent out one character at a time and the receiver end of the communication line synchronization is performed by examining synchronization bits that are included at the beginning and at the end of each character.



48. Write the Features of 8255A.

1. The 8255A is a widely used, programmable, parallel I/O device.
2. It can be programmed to transfer data under various conditions, from simple I/O to interrupt I/O.
3. It is compatible with all Intel and most other microprocessors.
4. It is completely TTL compatible.
5. It has three 8-bit ports: Port A, Port B, and Port C, which are arranged in two groups of 12 pins.
6. Its bit set/reset mode allows setting and resetting of individual bits of Port C.
7. The 8255 can operate in 3 I/O modes: (i) Mode 0, (ii) Mode 1, and (iii) Mode 2.

49. Distinguish between synchronous and asynchronous transmission? (April/May-15)

S.NO	Synchronous Data Transmission	Asynchronous Data transmission
1	In this transmission, both transmitter and receiver are synchronous with the common CLK signals	In this transmission, both transmitter and receiver are not synchronous with the common clock (CLK), they use separate clock
2	In this mode, the framing bits are sent along with block (at the beginning of block)	Framing bits are sent along with each character (at the beginning and end of the character)
3	The speed of the synchronous data transmission is higher than 20K bauds	The speed of the asynchronous data transmission is less than 20K bauds.
4	Framing information is SIM character	Framing information is start and stop bits
5	It is always implemented through hardware	It is implemented through software

50. How is DMA initiated? How the DMA operations perform in Microprocessor? (Apr/May 2018)

When the IO device needs a DMA transfer, it will send DMA request signal to the DMA controller. The DMA controller in turn sends a HOLD request to the processor. When the processor receives a HOLD request, it will drive its tri stated pins to high impedance state at the end for current instruction execution and send an acknowledge signal to the DMA controller. Now the DMA controller will perform DMA transfer

51. What is keyboard interfacing? (May/June 2012& Nov/Dec 2017) or How is keyboard interfaced with microprocessor?

Keyboard interfacing is interfacing an input device. Push button switches are used in simple keyboard interface one input line is required to interface one key and this number will increase with number of keys. It is in the form of matrix with rows and columns and at the intersection a switch is present.

52. Give the difference between maskable and non-maskable interrupts Nov/Dec 2017

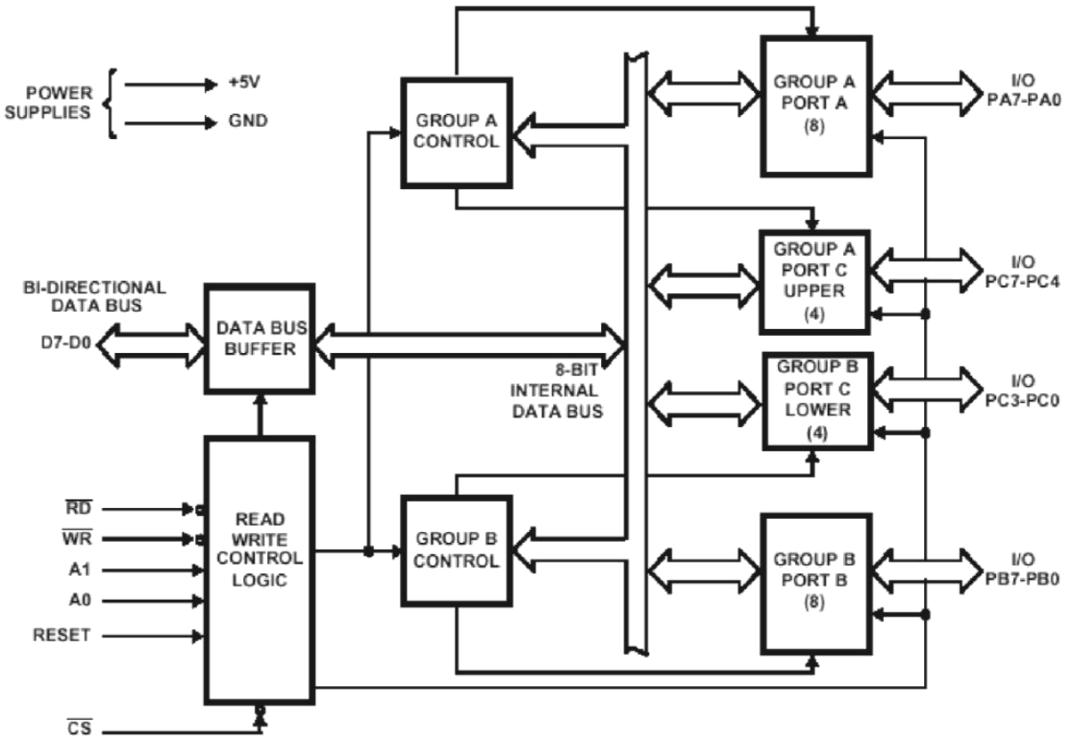
Maskable Interrupt: An Interrupt that can be disabled or ignored by the instructions of CPU are called as Maskable Interrupt. Eg: RST6.5,RST7.5,RST5.5 OF 8085 are maskable Interrupts.

Non-Maskable Interrupt: An interrupt that cannot be disabled or ignored by the instructions of CPU are called as Non-Maskable Interrupt.

PART-B

1. With neat sketch explain the functional block diagram of 8255 -Programmable peripheral Interface (PPI). (May/June 13, Nov/Dec-14, Nov/Dec-15 &17) (Apr/May 2017) (Apr/May 2018)

PROGRAMMABLE PERIPHERAL INTERFACE (PPI) - INTEL 8255



The 8255A block diagram

8255 PPI has 24 I/O pins distributed to four ports

1. Port A,
2. Port B,
3. Port C (upper) and
4. Port C (lower).

These ports are divided into two groups.

- Group A has port A and port C (upper) and
- Group B has port B and port C (lower).

While port A and port B have eight I/O lines each, port C (upper) and port C (lower) have four I/O lines each, thus accounting for 24 I/O lines.

D_7-D_0 account for data bus buffer, control logic, Group A and Group b controls

Group A control block controls port A and P_{C7}-P_{C4} while Group B control block control port B and P_{C3}-P_{C0}

- Port A: It can be programmed by mode 0, mode1, mode 2
- Port B: Programmed by mode 0 & mode 1
- Port C: Programmed by bit set/reset Operation.

Data Bus Buffer:

- It is a tri-state bi-directional buffer used to interface the internal data bus of 8255 to the system data bus.
- The instruction executed by microprocessors can read the data from the buffer or write the data into the buffer.

Control logic:

- The control logic block accepts control bus signals as well as inputs from the address bus and generates the commands to the individuals Group A and Group B.
- The control logic has six lines, they are ;
  - RD (Read): When the signal is low, MP reads data from selected I/O port of 8255.
  - WR (write): Signal is low, Microprocessor writes data
  - A0 and A1:

A1	A0	Selected ports/ Control Word Register
0	0	Port A
0	1	Port B

1	0	Port C
1	1	Control Word Register

- Reset: This is an active high signal; it class all the ports in the input mode.
- CS(Chip Select):

CS	Selected
0	8255 is selected
1	8255 is not selected

8255 can be operated in two basic modes.

- Bit set/reset mode (BSR mode) and
- I/O mode.

i) Bit set/reset mode

- This mode is applicable only for port C. This mode can set or reset a single bit in port C.
- The eight possible combinations of states of bit D3 D2 D1 in the bit set-Reset format determine particular bit in P_{C0}-P_{C7}, as per the status of bit D_O

BSR Control word format

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	**X**	**X**	**X**				**S/R**

D_7 **D_0**

1- I/O mode 1-Set

0- BSR mode 0-Reset

(D_3, D_2, D_1)- bit selection

000- Bit 0	P_{C0}	100 – Bit 4	P_{C4}
001- Bit 1	P_{C1}	101 – Bit 5	P_{C5}
010 – Bit 2	P_{C2}	110 – Bit 6	P_{C6}
011 – Bit 3	P_{C3}	111 – Bit 7	P_{C7}

ii) I/O mode is further divided into three modes of operation.

i. Simple I/O mode (mode 0)
ii. I/O mode (mode 1) and
iii. bidirectional (mode 2)

1. Simple I/O mode (mode 0)

Mode 0:

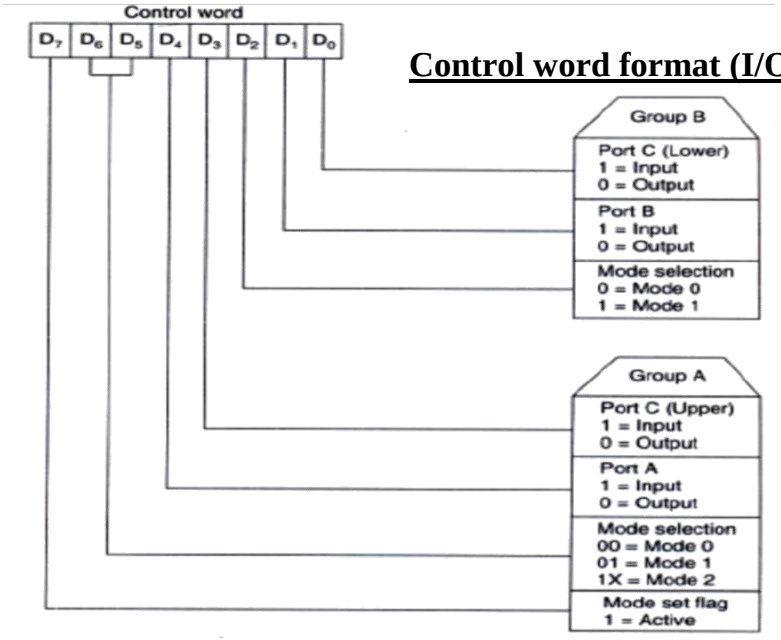
- There are two 8-bit ports (A and B) and two 4-bit ports (C (lower)) and (C (upper)).
- Any port can be an input port or an output port.
- Outputs are latched.
- Inputs are buffered.
- Ports do not have handshake.

2. Input/output (Mode 1)

- It provides means for transferring I/O data to or from a specified port in conjunction with strobes or handshaking signals. Port A and port B use the lines on port C for handshaking signals.
- There are two groups (group A and B).
  - Each group contains one 8-bit data port and one 4-bit control data port.
  - The 8-bit data port can be either an input port or an output port. Both inputs and outputs are latched.
  - The 4-bit port is used for control as well as for status of the 8-bit data port.

3 Bidirectional (Mode 2)

This functional configuration provides a means for communicating with a peripheral device or structure on a single 8-bit bus for both transmitting and receiving data. Handshaking signals are provided to maintain a proper bus flow discipline. Interrupt generation and enable/disable functions are also available.



Control word format (I/O mode)

Configuring and interfacing of 8255

The 8255 can be configured to work in one of the above modes by the microprocessor through a control word. This control word is sent by the microprocessor to the 8255. Ports A, B, C, and the control register port are addressed by A_0, A_1 pins as follows:

8255 port selection

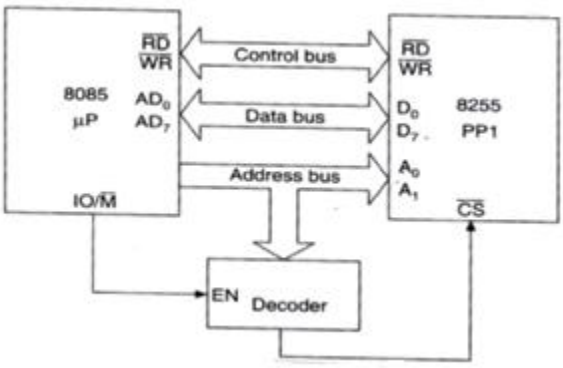
A_1	A_0	Selected ports
0	0	PORT A
0	1	PORT B
1	0	PORT C
1	1	Control register

While both the input and output operations are possible on A, B and C ports, the control register port can only be written into. No read operation of the control register is allowed.

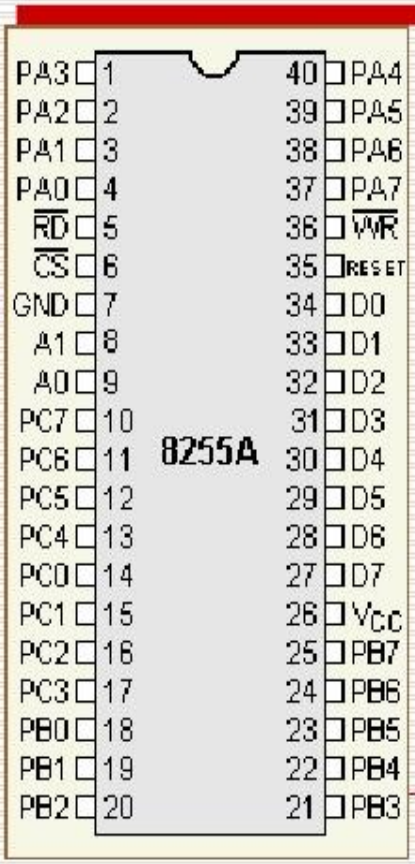
Interfacing the 8255

The address decoder generates the chip select signal; and are directly connected to the 8085 read/write control signals, and the RESET is connected to the general reset. In the present interfacing, the port numbers for A, B, C and control register ports are 0, 1, 2 and 3 respectively. The software instructions IN (Port No.) and OUT (Port No.) are used for data transfer.

Interfacing 8255 PPI to 8085 microprocessor

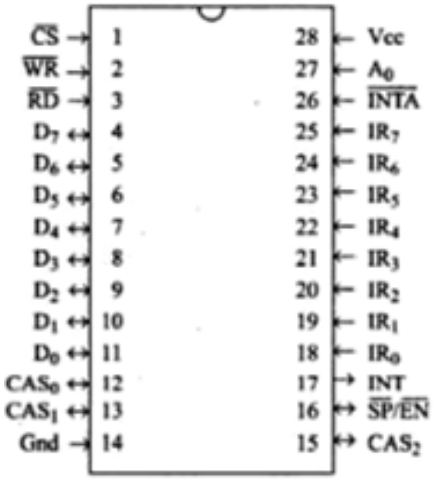


Pin diagram of 8255



D7 – D0	Data Bus
PA7 – PA0	Port A
PB7 – PB0	Port B
PC7 – PC0	Port C
CS	Chip Select
A0, A1	Address bits
RD	Read Input
WR	Write Input
RESET	Reset Input
Vcc	+5V
GND	0 Volts

2. With neat sketch explain the functional block diagram of programmable interrupt controller 8259. (May/June 12, April/May-15) (Nov/Dec 2016)



FUNCTIONAL BLOCK DIAGRAM OF 8259:

It has eight functional blocks. They are,

- Control logic
- Read Write logic
- Data bus buffer

UNIT-IV

4. Interrupt Request Register (IRR)
5. In-Service Register (ISR)
6. Interrupt Mask Register (IMR) 7. Priority Resolver (PR) 8. Cascade buffer.

Data Bus Buffer

The data bus buffer allows the 8085 to send control words to the 8259A and read a status word from the 8259A. The 8-bit data bus buffer also allows the 8259A to send interrupt opcode and address of the interrupt service subroutine to the 8085.

Read/Write Logic

The inputs control the data flow on the data bus when the device is selected by asserting its chip select input low.

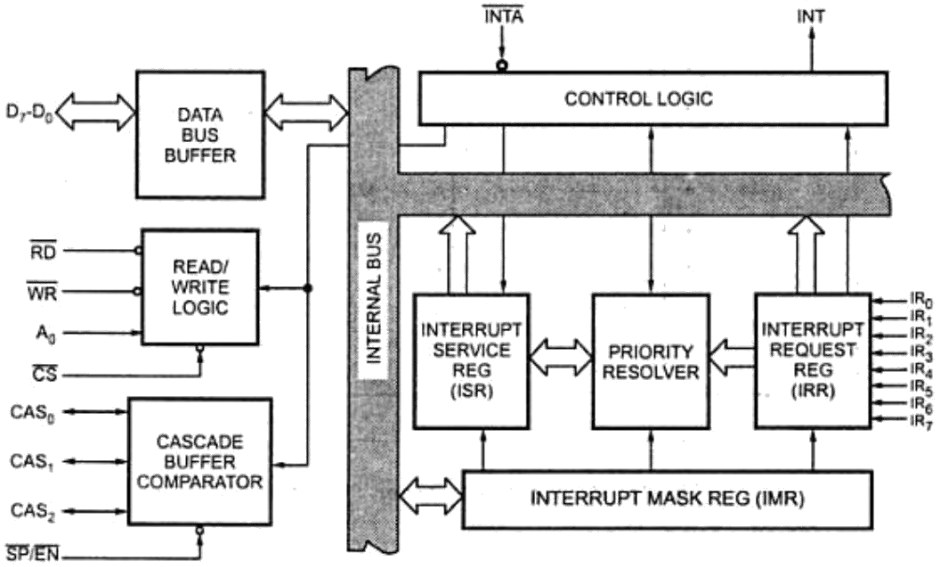


Fig. 15.2 Block diagram of 8259A

Control Logic

This block has an input and an output line. If the 8259A is properly enabled, the interrupt request will cause the 8259A to assert its INT output pin high. If this pin is connected to the INTR pin of an 8085 and if the 8085 Interrupt Enable (IE) flag is set, then this high signal will cause the 8085 to respond INTR as explained earlier.

Interrupt Request Register (1RR)

The IRR is used to store all the interrupt *levels* which are requesting the service. The eight interrupt inputs set corresponding bits of the Interrupt Request Register upon service request.

Interrupt Service Register (ISR)

The Interrupt Service Register (ISR) stores all the levels that are currently being serviced.

Interrupt Mask Register (IMR)

Interrupt Mask Register (IMR) stores the masking bits of the interrupt lines to be masked. This register can be programmed by an Operation Command Word (OCW). An interrupt which is masked by software will not be recognized and serviced even if it sets the corresponding bits in the IRR.

Priority Resolver

The priority resolver determines the priorities of the bits set in the IRR. The bit corresponding to the highest priority interrupt input is set in the ISR during the INTA input.

Cascade Buffer Comparator

This section generates control signals necessary for cascade operations. It also generates Buffer-Enable signals. The 8259 can be cascaded with other 8255 in order to expand the interrupt handling capacity to sixty-four levels. In such a case, the former is called a master, and the latter are called slaves. The 8259 can be set up as a master or a slave by the SP/EN pin.

CAS◦ - CAS

For a master 8259, the CAS_0-CAS2 pins are output pins, and for slave 8259s, these are input pins. When the 8259 is a master (that is, when it accepts interrupt requests from other 8259s), the CALL opcode is generated by the Master in response to the first INTA . The vector address must be released by the slave 8259. The master sends an identification code of three-bits to select one out of the eight possible slave 8259s on the CAS0-CAS_2 lines.

The slave 8259s accept these three signals as inputs (on their CAS◦ - CAS_2 pins) and compare the code sent by the master with the codes assigned to them during initialization. The slave thus selected (which had originally placed an interrupt request to the master 8259) then puts the address of the interrupt service routine during the second and third INTA pulses from the MN).

SP/EN (Slave Program /Enable Suffer)

The SP/EN signal is tied high for the master. However it is grounded for the slave.

In large systems where buffers are used to drive the data bus, the data sent by the 8259 in response to INTA cannot be accessed by the MPU (due to the data bus buffer being disabled). If an 8259 is used in the buffered mode (buffered or non-buffered modes of operation can be specified at the time of initializing the 8259), the SP/EN pin is used as an output which can be used to enable the system data bus buffer whenever the data bus outputs of 8259 are enabled (i.e. when it is ready to send data).

Control status word of 8259

First the 8259 should be programmed by sending Initialization Command Word (ICW) and Operational Command Word (OCW). These command words will inform 8259 about the following,

D0 : IC4: 0=no ICW4, 1=ICW4 required

D1 : SNGL: 1=Single PIC, 0=Cascaded PIC

D2 : ADI: Address interval. Used only in 8085, not 8086

1=ISR's are 4 bytes apart (0200, 0204, etc)

0=ISR's are 8 byte apart (0200, 0208, etc)

D3 : LTIM: level triggered interrupt mode: 1=All IR lines level triggered.0=edge triggered

D4-D7: A5-A7: 8085 only. ISR address lower byte segment.

The lower byte is of which A7, A6, A5 are provided by D7-D5 of ICW1 (if ADI=1), or A7, A6 are provided if ADI=0. A4-A0 (or A5-A0) are set by 8259 itself:

ICW1 (Initialisation Command Word One)

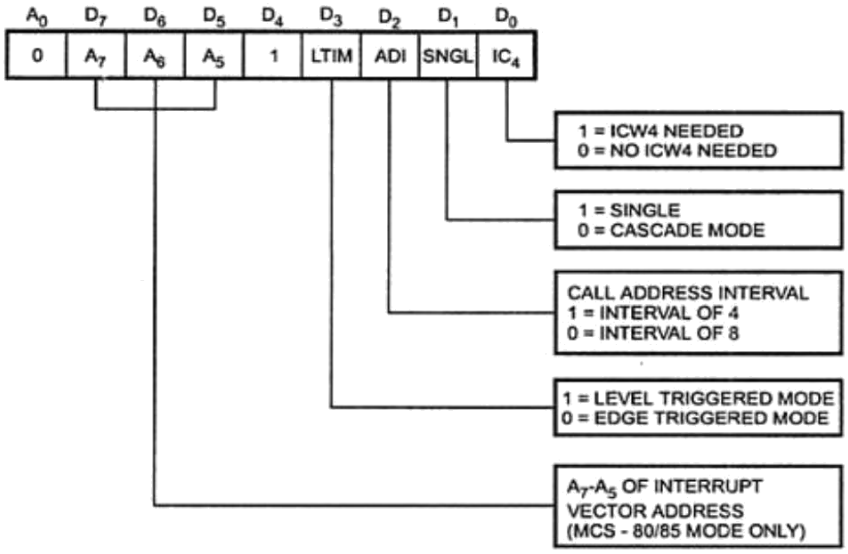


Fig. 15.4 Initialization command word 1 (ICW1)

ICW2 (Initialisation Command Word Two)

Higher byte of ISR address (8085), or 8 bit vector address (8086).

A0	D7	D6	D5	D4	D3	D2	D1	D0
1	A15	A14	A13	A12	A11	A10	A9	A8

ICW3 (Initialisation Command Word Three)

A0		D7	D6	D5	D4	D3	D2	D1	D0
1	Master	S7	S6	S5	S4	S3	S2	S1	S0
	Slave	0	0	0	0	0	ID3	ID2	ID1

- Master mode: 1 indicates slave is present on that interrupt, 0 indicates direct interrupt
- Slave mode: ID3-ID2-ID1 is the slave ID number. Slave 4 on IR4 has ICW3=04h (0000 0100)

ICW4 (Initialisation Command Word Four)

A0	D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	0	SFNM	BUF	M/S	AEOI	Mode

SFNM: 1=Special Fully Nested Mode, 0=FNM M/S: 1=Master, 0=Slave
AEOI: 1=Auto End of Interrupt, 0=Normal Mode: 0=8085, 1=8086

OCW1 (Operational Command Word One)

A0	D7	D6	D5	D4	D3	D2	D1	D0
1	M7	M6	M5	M4	M3	M2	M1	M0

IRn is masked by setting Mn to 1; mask cleared by setting Mn to 0 (n=0..7)

OCW2 (Operational Command Word Two)

A0	D7	D6	D5	D4	D3	D2	D1	D0
1	R	SL	EOI	0	0	L3	L2	L1

OCW3 (Operational Command Word Three)

A0	D7	D6	D5	D4	D3	D2	D1	D0
1	D7	ESMM	SMM	0	1	MODE	RIR	RIS

ESMM	SMM	Effect
0	X	No effect
1	0	Reset special mask
1	1	Set special mask

3. With neat sketch explain the functional block diagram of 8251 USART (16) (April/may 2011.May/June 13)) (April/may 2018)

- The 8251A is a programmable serial communication interface chip designed for synchronous and asynchronous serial data communication.
- It supports the serial transmission of data and having 28 pins DIP.

Read/Write control logic:

- The Read/Write Control logic interfaces the 8251A with CPU, determines the functions of the 8251A according to the control word written into its control register.
- It monitors the data flow.
- This section has three registers and they are control register, status register and data buffer.
- The active low signals ,, CS and C/ are used for read/write operations with these three registers.
- When C/ is high, the control register is selected for writing control word or reading status word.
- When C/ is low, the data buffer is selected for read/write operation.
- When the reset is high, it forces 8251A into the idle mode.
- The clock input is necessary for 8251A for communication with CPU and this clock does not control either the serial transmission or the reception rate.

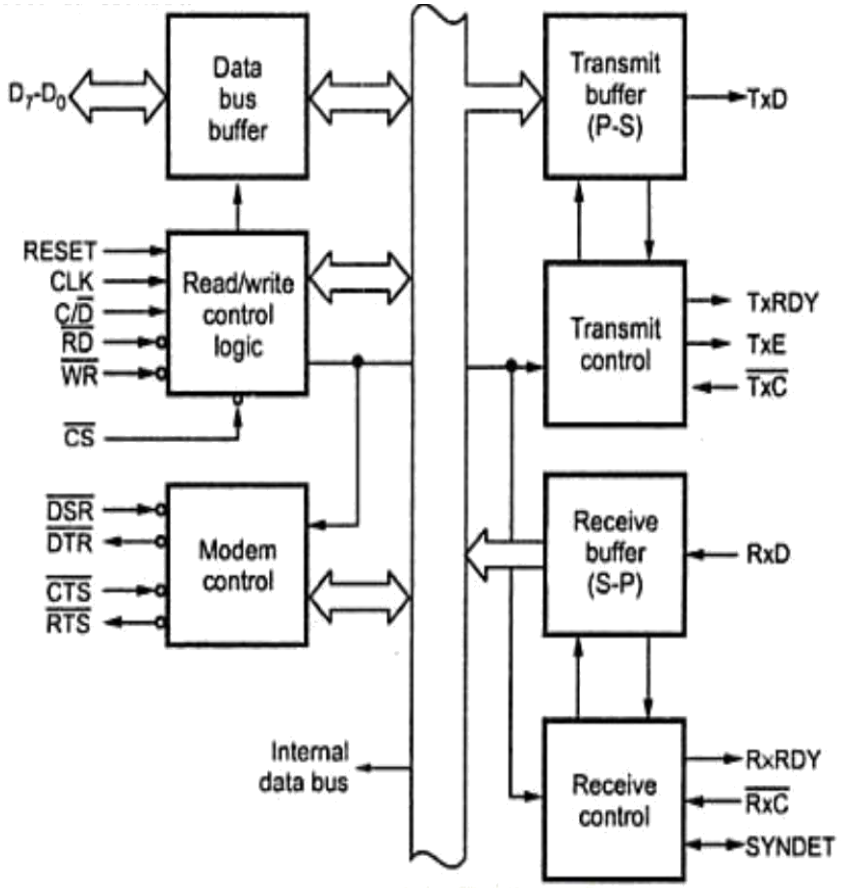


Fig. 7.2 Block diagram

Transmitter section:

- The transmitter section accepts parallel data from CPU and converts them into serial data.
- The transmitter section is double buffered, i.e., it has a buffer register to hold an 8-bit parallel data and another register called output register to convert the parallel data into serial bits.
- When output register is empty, the data is transferred from buffer to output register. Now the processor can again load another data in buffer register.
- If buffer register is empty, then TxRDY is goes to high.
- If output register is empty then TxEMPTY goes to high.
- The clock signal, TxC (low) controls the rate at which the bits are transmitted by the USART.
- The clock frequency can be 1,16 or 64 times the baud rate.

Receiver Section:

- The receiver section accepts serial data and convert them into parallel data
- The receiver section is double buffered, i.e., it has an input register to receive serial data and convert to parallel, and a buffer register to hold the parallel data.
- When the RxD line goes low, the control logic assumes it as a START bit, waits for half a bit time and samples the line again.
- If the line is still low, then the input register accepts the following bits, forms a character and loads it into the buffer register.
- The CPU reads the parallel data from the buffer register.
- When the input register loads a parallel data to buffer register, the RxRDY line goes high.
- The clock signal RxC (low) controls the rate at which bits are received by the USART.
- During asynchronous mode, the signal SYNDET/BRKDET will indicate the break in the data transmission.
- During synchronous mode, the signal SYNDET/BRKDET will indicate the reception of synchronous character.

MODEM Control:

- In 8251A the transmission and reception baud rates can be different or same.
- The device which requires serial communication with processor can be connected to this 9pin D-type connector using 9-core cable.
- The signals TxEMPTY, TxRDY and RxRDY can be used as interrupt signals to initiate interrupt driven data transfer scheme between processor and 8251 A.

4.With neat sketch explain the working of 8279 PROGRAMMABLE KEYBOARD/DISPLAYCONTROLLER. (Nov/Dec 2011, Nov/Dec 13.May/June 13, Nov/Dec-14, April/May-15,Nov/Decc-15) (Nov/Dec 2016) (Nov/Dec 2017)

KEYBOARD AND DISPLAY CONTROLLER (Intel 8279)

In the keyboard interface, the microprocessor scans the various keys, performs software debouncing and finally finds out the code of the depressed key.

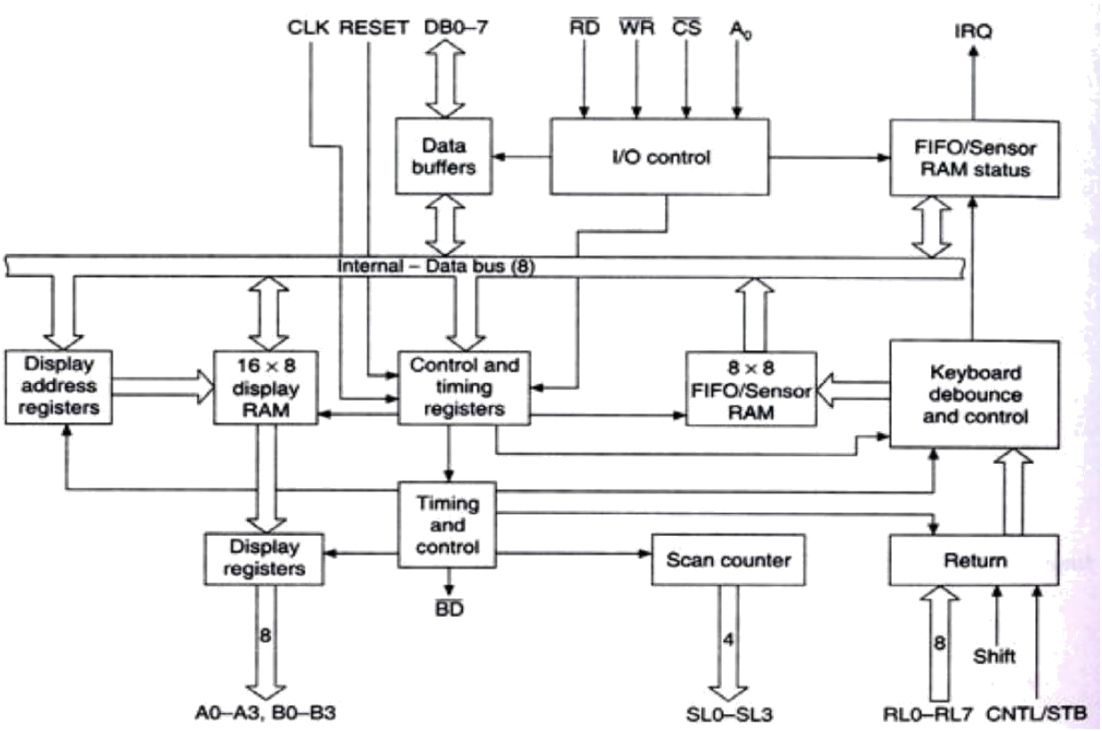
In the display interface, the microprocessor is busy in sending bit-by-bit information to the shift registers regarding the display of various segments.

The 8279 consists of two Sections

- Keyboard section and
- Display section.

- The keyboard section can interface to regular type writer style keyboards or random toggle or thumb switches.
- The display section drives the alphanumeric displays or a bank of indicator lights. Thus, the processor is relieved from scanning the keyboard or refreshing the display.

Internal block diagram of 8279



KEYBOARD SECTION

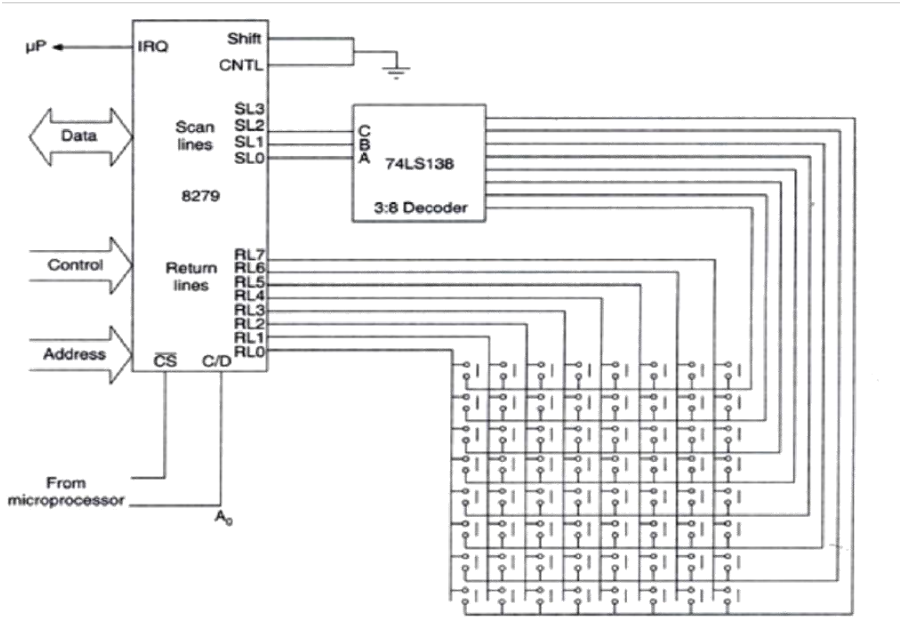
The 8279 offers four scan lines SL0-SL3, eight return lines RL0-RL7, a shift line and a control line. The scan lines are interfaced to the rows of the keyboard. The return lines can be directly connected to the eight columns of the keyboard, whereas the shift and control lines are connected to the shift and control keys.

There are two ways in which scan lines can be interfaced to the keyboard.

- Encoded Scan
- Decoded scan.

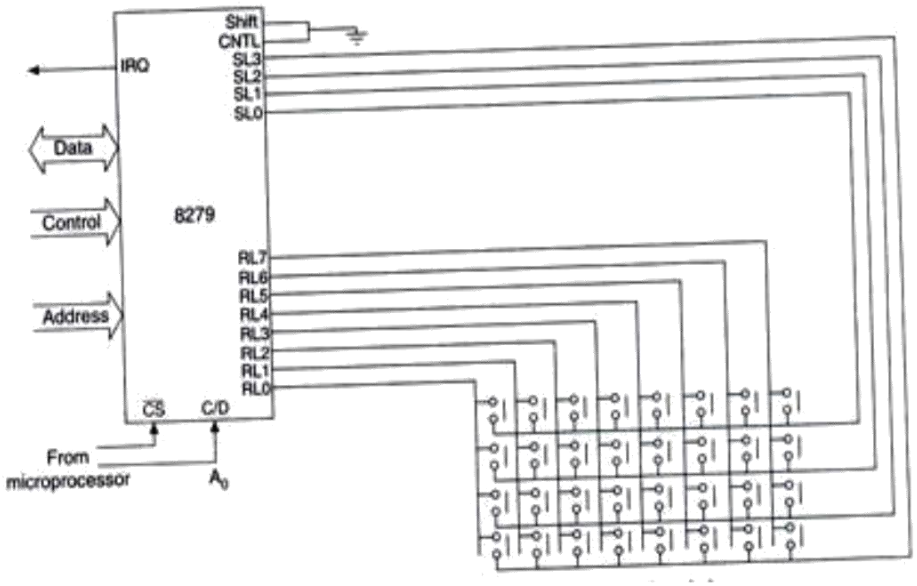
With the encoded scan, the row number is output on the three low-order scan lines and a 3:8 decoder is needed to send signals to the individual rows. This enables the interfacing of 8x8 keyboard.

Interfacing of 8 × 8 keyboard to 8279 chip- encoded scan



With the decoded scan, the 8279 provides row activation signals on four lines in sequence which can be directly interfaced to four rows of keyboard thus enabling the interfacing of 4 x 8 keyboard. In both the cases a key depression generates a 6-bit encoding of the key position, a shift bit and a control bit.

Interfacing of 8 × 8 keyboard to 8279 chip- decoded scan



This information is entered in the RAM resident in the 8279. The RAM serves as a simple queue, i.e. First In First Out (FIFO), meaning that the position code for various key depressions will be stored and transferred to the computer in order of their entry. When FIFO is empty and a key depression occurs, then the interrupt signal is sent to the processor on the IRQ line.

Types of debouncing

Every key depression should be validated before entering the information in FIFO. The 8279 has two types of debouncing:

- 2-key lockout and
- N-key rollover.

2-key lockout

If two or more keys are pressed simultaneously, the key which is released last is treated as a single depression.

N-key rollover

In case of N-key rollover, each key depression is treated independently from all the others, i.e. more than one key can **be** pressed **simultaneously** and recognized.

Interfacing sensor to 8279

The scan and return lines can be used to interface a matrix of switches. These switches have been named sensors and have only two states—ON and OFF

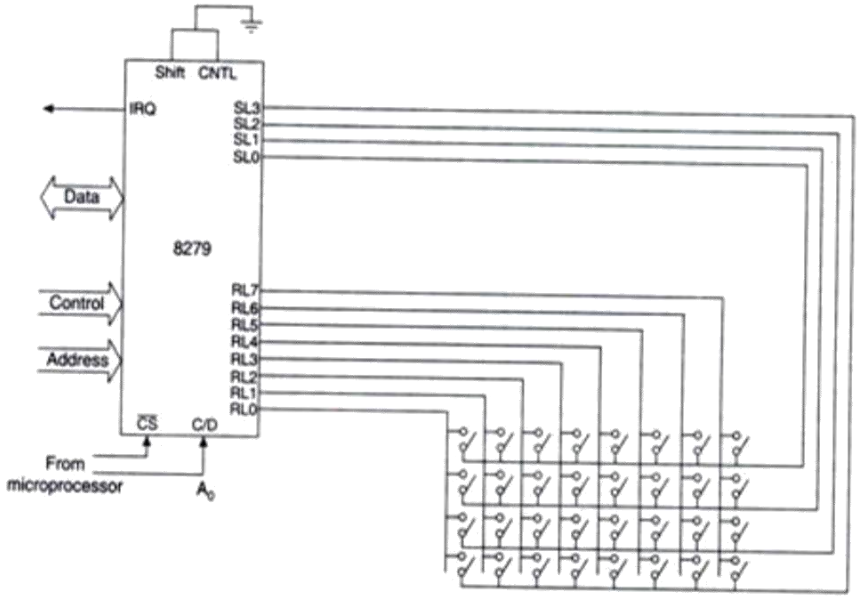
The data format

MSB **LSB**

RL7	RL6	RL5	RL4	RL3	RL2	RL1	RL0

This information is entered directly into the sensor RAM. In this way, the sensor RAM keeps on the image of the state of the switches in the sensor matrix. If a change in the sensor value is detected, the interrupt line (IRQ) goes high. There is no debouncing in this mode.

Interfacing of 8 × 4 sensor matrix to 8279 chip- decoded scan



Apart from the keyboard and the sensor matrix interfacing, these lines can also be used for general purpose strobed input. This data is also entered to FIFO from the return lines. The data entry is caused by the rising edge of the CNTL/STB line pulse.

The data format

MSB **LSB**

RL7	RL6	RL5	RL4	RL3	RL2	RL1	RL0

Data can come from the switch matrix or another encoded keyboard.

Possible modes in keyboard section are

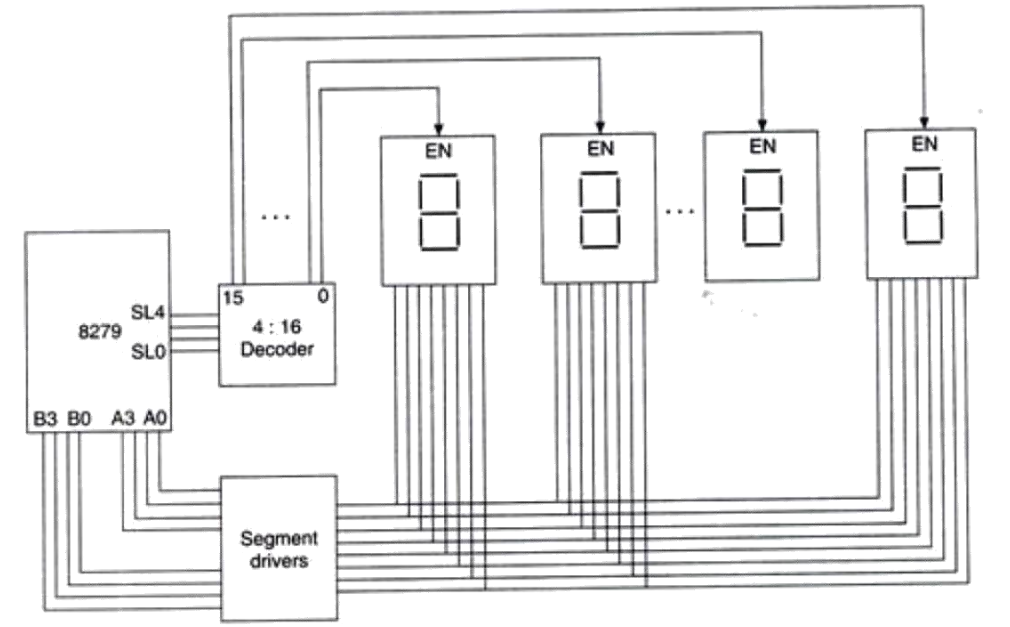
- Encoded Scan Keyboard- 2-key lockout Encoded Scan Keyboard- N-key rollover
- Decoded Scan Keyboard-2-key lockout
- Decoded Scan Keyboard-N-key rollover
- Encoded Scan Sensor Matrix
- Decoded Scan Sensor Matrix
- Strobed Input

DISPLAY SECTION

The display section provides a scanned display interface for LED, incandescent and other popular display technologies. It allows both numeric and alphanumeric segment displays as well as lamp indicators. The 8279 uses four **scan lines SL_0- SL_3,**two 4-bit output ports (A0-A3 and B0-B3), a 16 x 8 display RAM and a blank display ( ) line for its display operation.

The two 4-bit ports can be used independently or as one 8-bit port. For BCD type seven segment LED displays, only 4-bit codes are needed in the BCD form. The two 4-bit ports can be used independently for this.

Interfacing 7-segment LED display to the 8279 chip



The 16 x 8 display RAM can be organized into a dual 16 x 4 RAM. Thus the 8279 facilitates 8 or 16 characters multiplexed displays that can be organized as dual 4 bit or single 8 bit. Both right and left entry display formats are possible.

The display RAM can be loaded or read by the CPU. The read/write addresses are programmed by the CPU command. They can also set auto-increment after each read or write. The display RAM can be directly read by the CPU after the correct mode and address is set.

When the keyboard is in the decoded scan, the display will automatically be in the decoded scan, i.e. only the first 4 characters in the display RAM will be displayed. The output is used to blank the display during digit switching. The display-blanking command can be used to activate the line, i.e., to blank the display.

5. With neat sketch explain the working of 8253/8254 as PROGRAMMABLE TIMER/ COUNTER.(May/June 13, Nov/Dec 2011,Nov/Dec-15) (Nov/Dec 2016)

PROGRAMMABLE INTERVAL TIMERS INTEL 8253 AND INTEL 8254

The 8253 facilitates three 16-bit programmable counters on the same chip. The 8253, on receiving the command from the CPU, counts out the delay and interrupts the CPU at the end, thus reducing the software overheads.

Other functions which can be achieved by the 8253 (applications) are as follows:

- Event Counter
- Programmable Rate Generator
- Binary Rate Multiplier
- Real Time Clock
- Digital One-Shot
- Complex Motor Controller

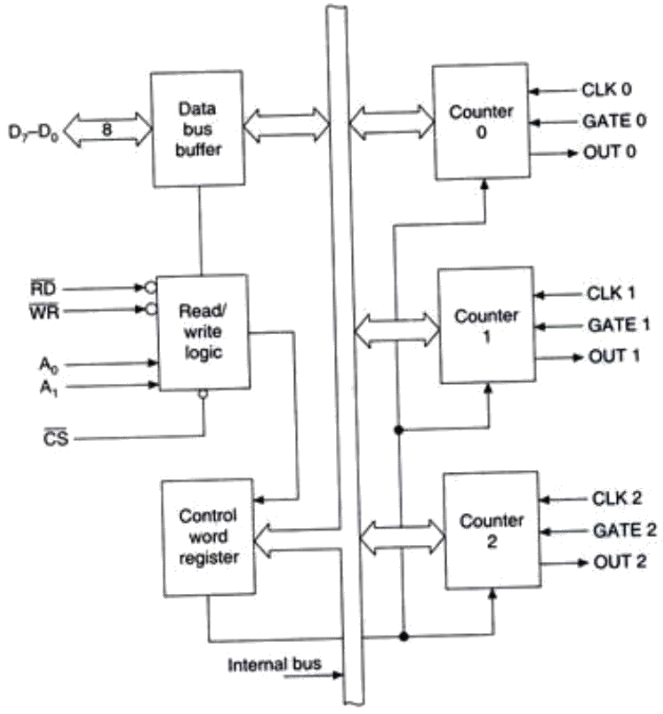
Each counter consists of a single, 16-bit programmable down counter which can operate either in binary or in BCD. Each counter has two inputs as CLK and GATE and one output as OUT.

The 8254 handles signals from dc to 10 MHz

frequency. Up to 8 MHz — Intel 8254

Up to 10 MHz — Intel 8254-2

Functional block diagram of 8253



Operating Modes

The counters are fully independent and each can have separate mode configurations and counting operations, binary or BCD.

Mode 0—interrupt on terminal count

- This mode is typically used for event counting.
- The GATE input is used to enable or disable the counting. When the GATE input is high, the counting is enabled and when it is low the counting is disabled.
- After the count is loaded into the selected count registers, the OUT remains low and the counter starts counting one cycle after the counter is loaded.
- On reaching the terminal count (i.e. zero), OUT will go high and remain high until the count register is reloaded or the control word is written.
- The OUT point can be connected to any interrupt request pin of the microprocessor.

Mode 1—programmable one-shot

- The GATE input is used to trigger the OUT.
- The OUT will be initially high and go low on the count following the rising edge at the GATE.
- When the terminal count is reached, the OUT goes high and remains high until the CLK pulse after the next trigger. Thus, the duration of one-shot pulse at OUT can be programmed through the count.

Mode 2—rate Generator

- The counter in this mode acts as divide-by-N counter. It is used to generate a real-time clock interrupt.
- The OUT will initially be high. When the count has decremented to l, the OUT goes low for one clock pulse.
- The OUT then goes high again, the counter reloads the initial count and the process is repeated.
- The GATE input, when low, disables the counting. When the GATE input goes high, the counter starts from the initial count.
- The OUT gives one pulse after every N clock pulses, thus achieving the rate generator function.

Mode 3—square wave generator

- It is similar to Mode 2 except that the OUT remains high for the first-half of the count and goes low for the other-half, thus generating a programmable square wave shape at OUT.
- Suppose n is the number loaded into the counter, then the OUT will be
- High for n/2 counts and low for n/2 counts if n = even
- High for (n + l)/2 counts and low for (n - l)/2 counts if n = odd.

The GATE input function is the same as that in Mode 2.

Mode 4—software triggered strobe

- The OUT is initially high and goes low for one clock period on the terminal count. Thus, a pulse can be generated by software.
- The GATE input when low disables the counting and when high, enables the counting.
- The difference between Mode 4 and Mode 2 is that in Mode 2, the OUT pulses are generated continuously after every N clock pulses (i.e. on terminal count but N is reloaded automatically), but in Mode 4, the OUT pulse is generated only once after N clock pulses (i.e.terminal count, but no automatic reloading).

Mode 5—hardware triggered strobe

- The OUT is initially high. The counter will start counting after the rising edge of the GATE input and the OUT will go low for one clock period when the terminal count is reached.

- The hardware circuit should trigger the GATE input to initiate the counting operation, thereby generating the OUT pulse.

OPERATION OF GATE INPUT IN DIFFERENT MODES

Mode	Low or going low	Rising	High
0	Disables counting	—	Enables counting
1	—	(1) Initiates counting (2) Resets output after the next clock	—
2	(1) Disables counting (2) Sets the immediately high output	Initiates counting	Enables counting
3	(1) Disables counting (2) Sets the output immediately high	Initiates counting	Enables counting
4	Disables counting	—	Enables counting
5	—	Initiates counting	—

6. Explain the Programming and the interfacing of 8253

In the 8253, the control words are used to program the counters with the desired mode and quantity information. These control words are sent to the Control Word Register for each counter to be programmed.

The format of the control word is as follows:

D_7	D_6	D_5	D_4	D_3	D_2	D,	D_0
SC_1	SC_0	RW_1	RW_0	M_2	M_1	M_0	BCD

(a)The SC_1 and SC_0 bits select one of the three counters of the 8253

(SC_1, SC_0=00—counter 0, 01—counter 1, 10—counter 2, 11—illegal)

In case of the 8254, when (SC_1, SC_0=11) then it is a read back command which is different from that of the 8253.

(b) The M2, Ml and M0 bits select the mode

(M2, Ml, M0 = 000—Mode 0, 001—Mode 1,X10—Mode 2, X11—Mode 3, 100—Mode 4, 101—Mode 5), X = Don't care

(c) The BCD bit selects the BCD counting or the Binary counting (BCD = 0—Binary counting 16bit, 1—Binary coded decimal (BCD) counter (4 decades))

(d) The RW_1, RW_0 bits identify the Read/Write operation for the count as follows:

RW_1	RW_0	Operation performed
0	0	Counter latching operation. Used when the programmer wants to read the contents of any counter "on the fly", i.e. without disturbing the counting.
0	1	Read/Write the least significant byte only.
1	0	Read/Write the most significant byte only.
1	1	Read/Write the least significant byte first and then the most significant byte.

The actual order of programming is quite flexible. The control words can be written in any sequence of the counter selection. The loading of the count register with the actual count value, however, must be done in the actual sequence specified by the RW_0, RW_1 bits in the control word for any counter.

That means:

(a) For each counter, the control word must be written before the initial count is written.

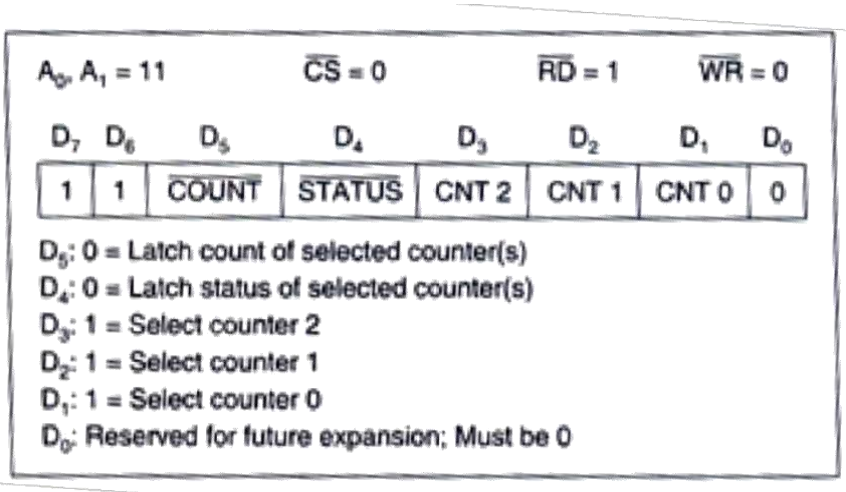
(b) The initial count must follow the count format, i.e. RW_0 , RW_1 = 01—least significant byte only, RW_0 , RW_1 = 11—least significant byte followed by the most significant byte.

If a counter is programmed to read/write two byte counts, it must be ensured that the program does not transfer the control between writing the first byte and the second byte to another routine which also writes into the same counter. Otherwise, an incorrect value will get loaded to the counter. The contents of the counter can be read by a simple I/O read operation. However, counting should be inhibited before this, by the GATE input. The other method is the read "on the fly" (also known as Counter Latch Command), i.e. read without disturbing the counting process. To achieve this, the control register is loaded with a special control word as shown below:

D_7	D_6	D_5	D_4	D_3	D_2	D,	D_0
SC_1	SC_0	0	0	X	X	X	X

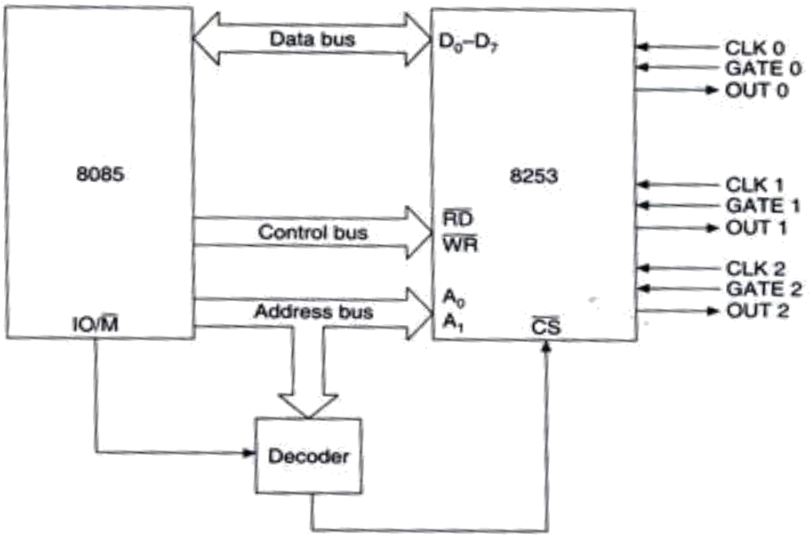
The SC_1 and SC_0 bits specify the counter. The contents of the counter are latched into a storage register associated with the counter in the 8253. Using a simple I/O read operation, the storage register, i.e. the contents of counter, can be read.

Another possible method for reading the counter is to use the Read Back command. This is specific to the 8254 and is not present in the 8253. By this command, the user can check the count value, the programmed mode and the current state of the OUT pin and the NULL count flag of the selected counters. The command is written into the control word register. Read back command format

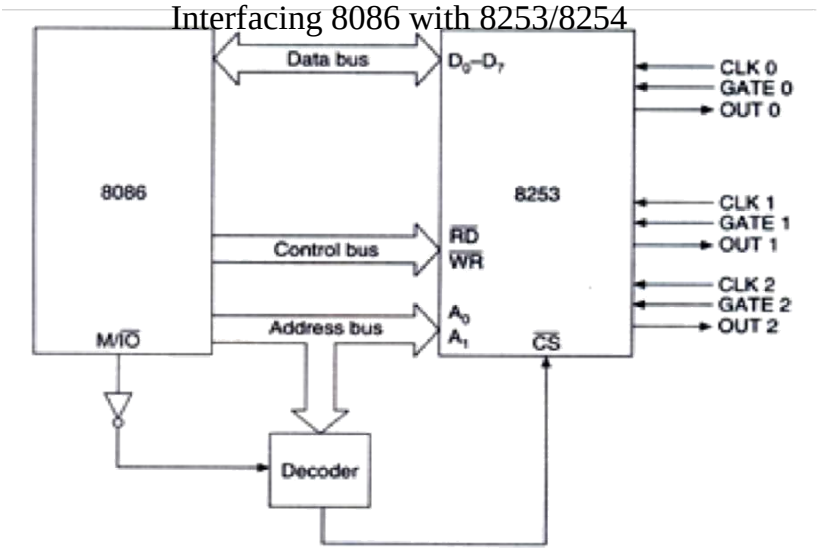


The Read Back command may be used to latch the outputs of either one or more counters by setting the COUNT bit D_5 = 0 and selecting the desired counters. This single command is functionally equivalent to multiple Counter Latch commands. The latched count of each counter is held until it is read or the counter is reprogrammed.

Interfacing 8085 with 8253/8254



Interfacing 8086 with 8253/8254

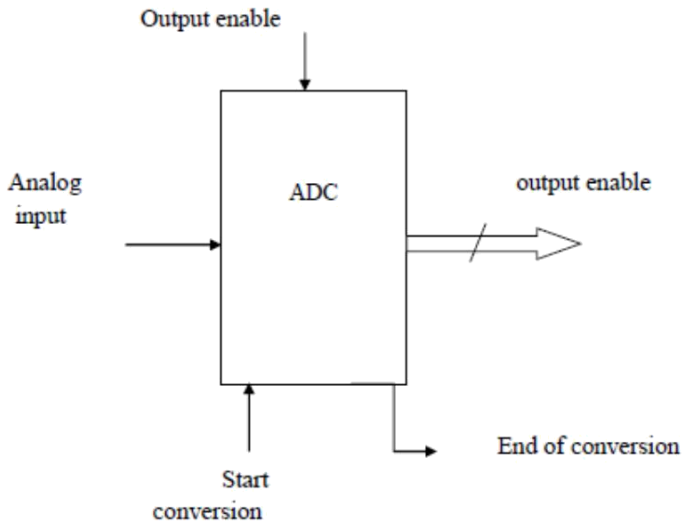


7. Explain how the ADC interfacing with 8085 (April/May 2011, May/June 12, Nov/Dec 2011, Nov/Dec-14, April/May-15)

ANALOG-TO-DIGITAL CONVERTER

The analog-to-digital converter (ADC) takes an analog signal as input and presents an equivalent digital signal as output. The basic techniques include—ramp, integrating ramp, successive approximation, etc. Out of these, the successive approximation technique is quite prevalent. The whole operation is carried out by the following signals.

- Start Convert
- End of Conversion
- Output Enable



The control logic sends the Start Convert signal when the analog signal is stable on the analog input line. The ADC converts the analog signal to digital and sends the End of Conversion signal on completion of conversion.

To read the output from digital output lines, the control logic should send the Output Enable signal to ADC, which will then enable the lines of the digital output. In many ADCs the digital output is sent to digital output lines by ADC along with the End of Conversion signal. Thus the Output Enable is not used in such cases.

The analog-to-digital converter (ADC) deals with the following signals.

Input voltage	Input analog voltage. Input to ADC.
Start Convert	Command to start conversion. Input to ADC
End of Conversion	Information that conversion is complete and the output can be read. Output signal from ADC.
Digital output	Digital value of analog input derived from conversion. Output from ADC.

Interfacing microprocessor to ADC

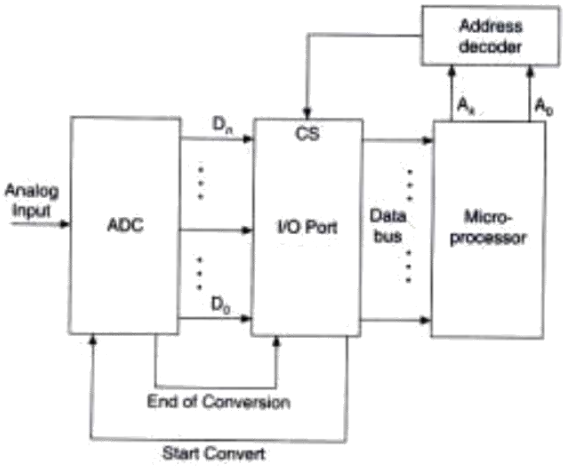
A microprocessor can be interfaced to ADC in any one of the following three modes.

Asynchronous Mode

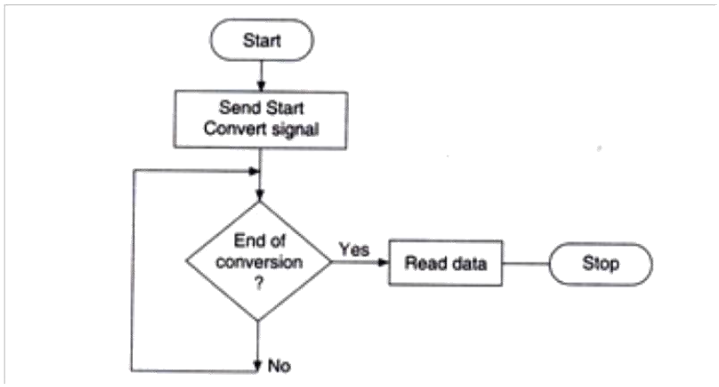
- The microprocessor starts the conversion process by sending the Start Convert signal to the ADC.
- The ADC takes the analog input and converts it to digital.
- When the conversion is complete, the ADC outputs the End of Conversion signal to the microprocessor.
- On receiving this signal, the microprocessor reads the data present at the output of ADC.

The microprocessor continuously checks for the End of Conversion signal after the Start Convert signal is output.

Microprocessor interface to ADC (Asynchronous mode)



Flowchart (Asynchronous mode)



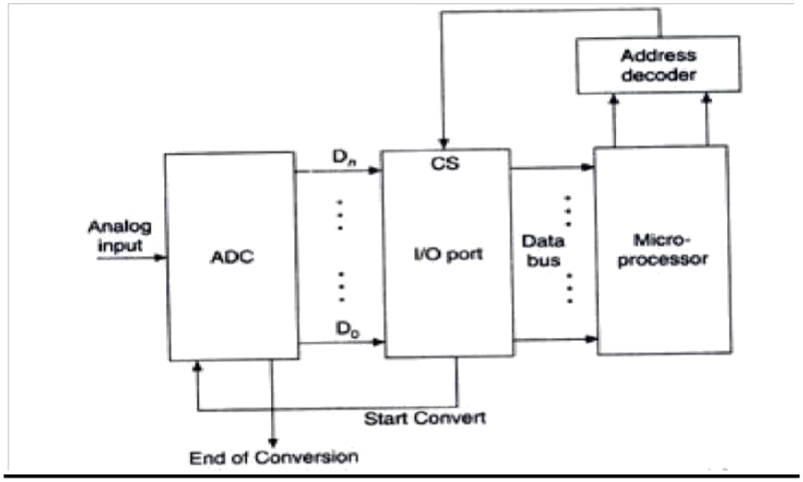
Disadvantage:

It is evident that in the asynchronous mode, the microprocessor wastes time by waiting for the End of Conversion signal. The conversion time for an ADC is up to a few milliseconds. Thus, for every conversion, this much time of the microprocessor will be wasted.

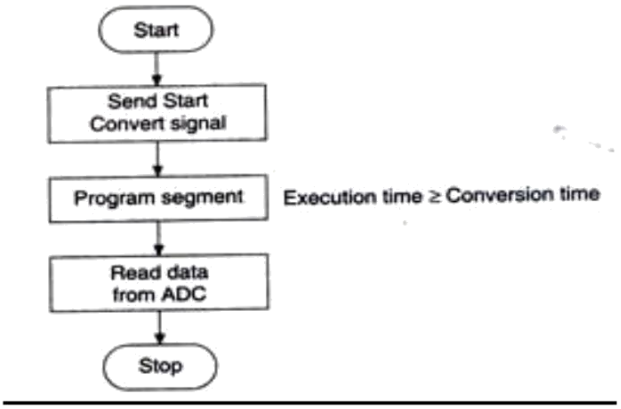
Synchronous Mode

The microprocessor issues the Start Convert signal to initiate the conversion process. Since the conversion time is known, the microprocessor, then, executes certain instructions so that the execution time of the instructions is greater or equal to the conversion time. The microprocessor will then read the data from ADC directly.

Microprocessor interface to ADC (Synchronous mode)



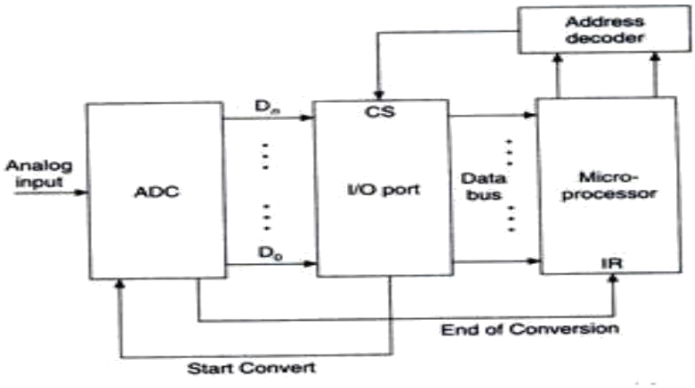
Flowchart (Synchronous mode)



Interrupt Mode

Many users, while accepting the benefits of the synchronous interface, find it inconvenient to write the program segment whose execution time is nearly equal to the conversion time. In a system containing 50 channels, 50 such different program segments will have to be written. The interrupt mode is used to avoid this.

Microprocessor interface to ADC (Interrupt mode)



- In this mode, the End of Conversion signal is physically connected to one of the interrupt inputs of the microprocessor.
- The microprocessor initiates the conversion process by sending the Start Convert signal to the ADC.
- The ADC completes the conversion and sends the End of Conversion signal.
- Since the End of Conversion signal is connected to one of the interrupt inputs of the microprocessor, the microprocessor is interrupted.
- The microprocessor suspends its program execution, saves the status and then executes the Interrupt Service Routine to service the interrupt caused.
- The interrupt service routine, in the present case, contains instructions to read and store the digital data from ADC.

It is clear that the interrupt mode is most advantageous for the interfacing of the microprocessor to the ADC. However, it uses one interrupt input of the microprocessor.

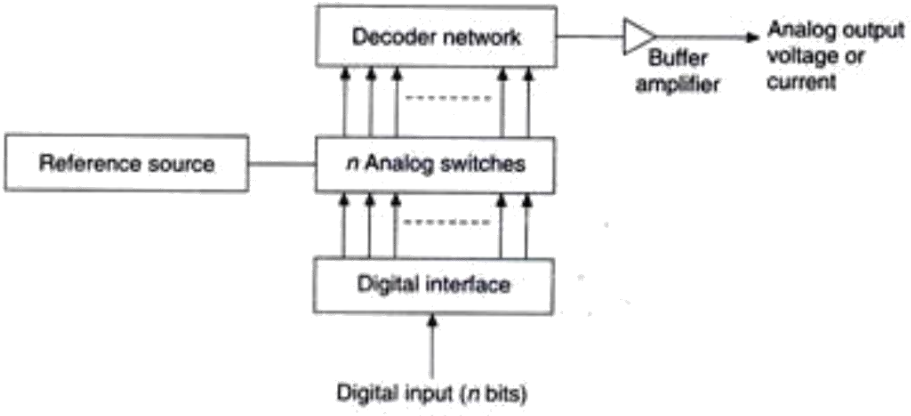
8. Explain how the DAC interfacing with 8085 May/June 12, Nov/Dec 2011

DIGITAL-TO-ANALOG CONVERTER

Digital-to-analog converters (DACs) translate digital information into equivalent analog voltage or current.

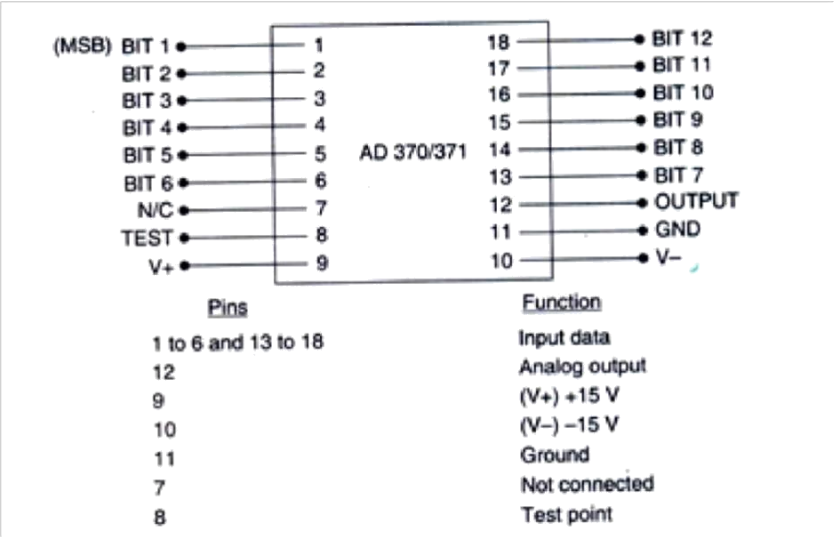
The DAC works on a very simple principle of decoder resistance network. The main constituents of DAC are the decoder network, an analog switch for each digital input, a buffer amplifier and the necessary control logic. The digital interface provides the storage of input data word and the control of the analog switch position. The analog switches under the control of digital interface either connect the reference source to the decoder network input terminal or ground the terminal.

DIGITAL-TO-ANALOG CONVERTER



The buffer amplifier provides impedance buffering and current driving capability. It is generally required because the output impedance of the decoder network is relatively high.

DAC AD 370/371 Pin diagram

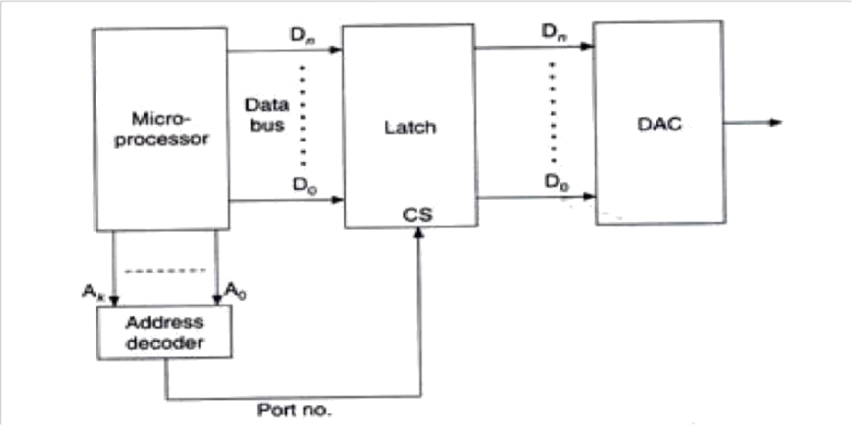


It is quite easy to interface a digital-to-analog converter to a microprocessor. If an n bit microprocessor is to be interfaced to an m-bit DAC, the following three possibilities may arise.

Case (i) n = m:

The output of the microprocessor can be directly connected to DAC through a latch. The microprocessor outputs the 8-bit digital value to the specified latch which is connected to the DAC. The software interface consists of instructions to load the digital data to the latch.

Microprocessor interface to DAC (n=m)

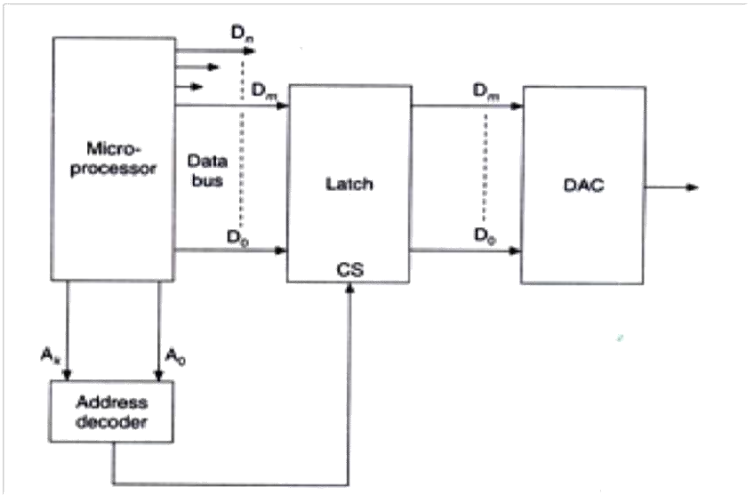


Case (ii) n > m:

This case is abnormal. However, if one is forced to connect a higher-bit *(n)* microprocessor to a lower-bit *(m)* DAC, only then, the lower *m* bits of the microprocessor data bus are connected to the DAC latch.

For DAC, the microprocessor behaves like an m-bit microprocessor . The programmer must ensure that the data is represented only in *m* bits, otherwise the higher-order bits may be lost, resulting in loss in accuracy. The software consists of loading data into the latch.

Microprocessor interface to DAC (n>m)



Case (iii) n < m:

Often it happens that 16-bit calculations are required to be performed on an 8-bit microprocessor. This is achieved by using double precision. If this result is to be output on a 16-bit DAC, then the 16-bit DAC will have to be interfaced to an 8-bit microprocessor.

Since DAC works continuously without any start/stop pulse, the two bytes should be loaded to DAC at the same instant. If the two bytes of the16-bit data are loaded one by one, then the analog value at the output will not represent the 16- digital value. Thus, interfacing a lower bit *(n)* processor to a higher bit *(m)* DAC is quite tricky.

First the lower-order n-bit data are stored in latch 1. The higher-order *(m - n)* bits are then stored in latch 2. The m-bit data are then stored simultaneously in latch 3 and latch 4 using the same pulse. Latch 3 and latch 4 are connected to DAC directly.

The software involves storing the data in various latches. The latch 3 and latch 4 are selected simultaneously and thus the DAC gets all the *m* bits of data at the same instant.

Microprocessor interface to DAC (n<m)

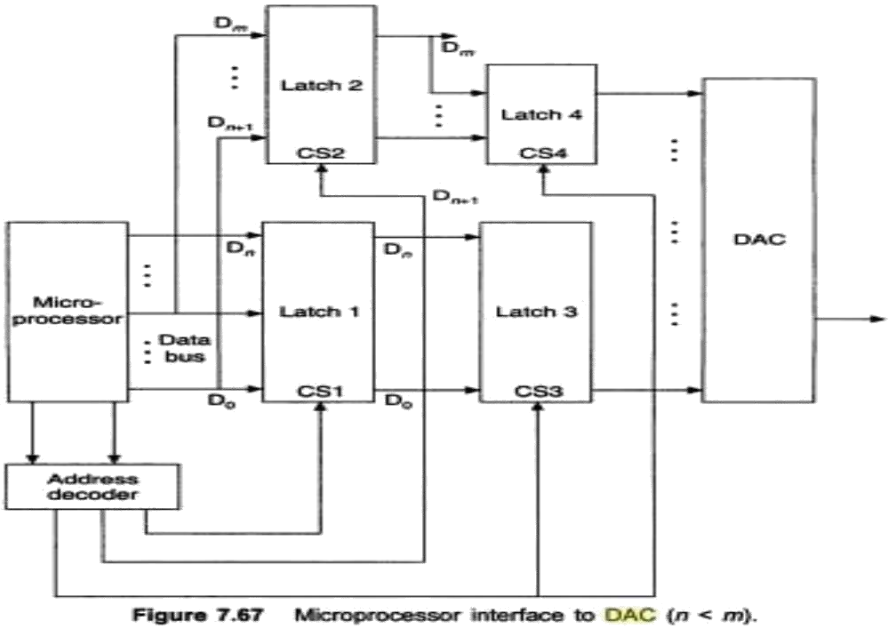


Figure 7.67 Microprocessor interface to DAC ($n < m$).

9. **Explain in detail about the 8237/57 DMA controller? (Apr/May 2017)**

DIRECT MEMORY ACCESS:

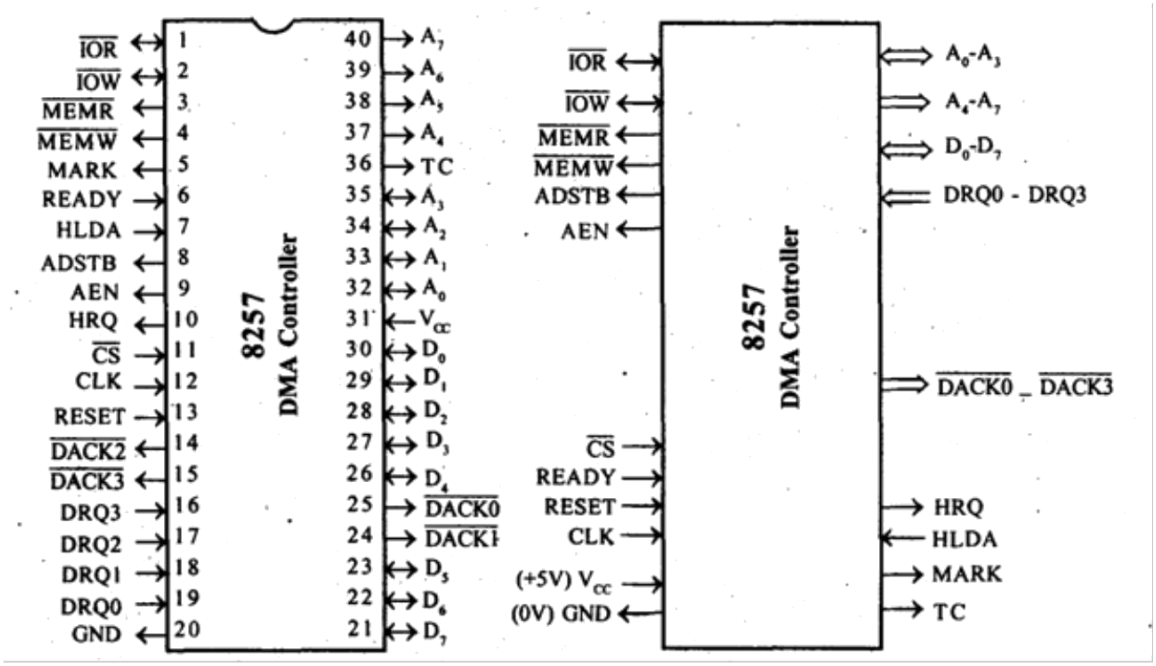
The ability of an I/O sub system is to transfer data to and from a memory sub system which is used for high speed data transfer.

Ex: Data transfer between a floppy disk and memory.

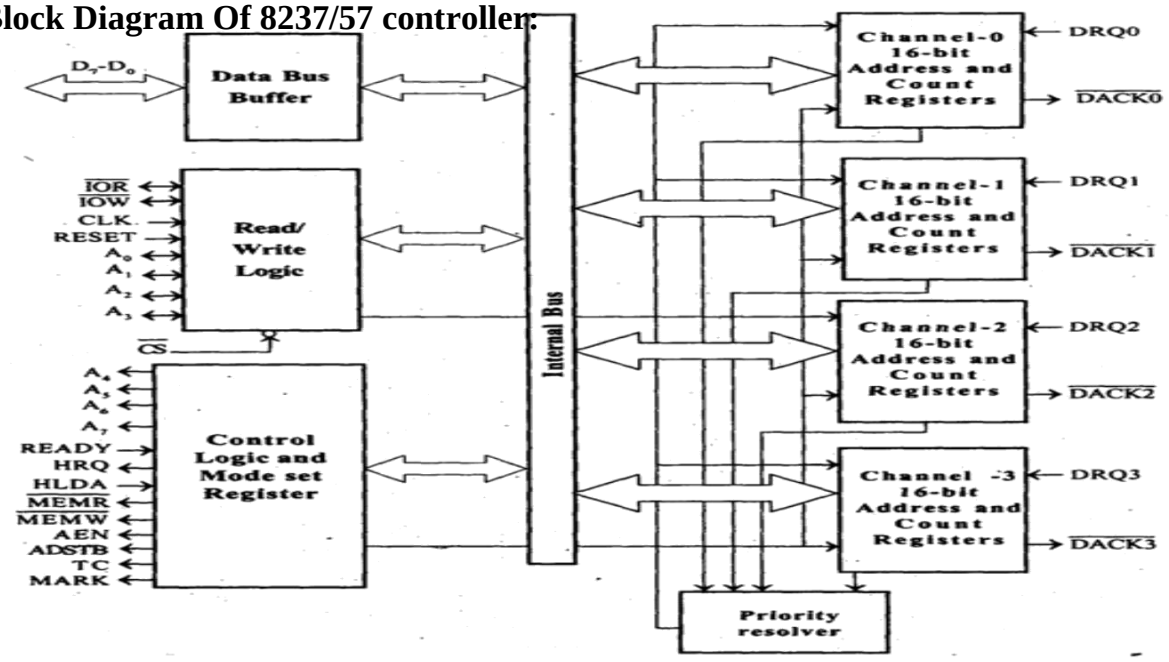
DMA CONTROLLER:

It is a device that can control data transfer between an I/O subsystem and a memory subsystem without the help of CPU.

Pin Diagram of 8237/57



Block Diagram Of 8237/57 controller:



Data bus buffer:

- It is a tri-state, bidirectional, 8 bit buffer which interfaces the 8257 to the system data in the slave mode; it is used to transfer data between microprocessor and internal registers.
- In master mode, it is used to send higher byte address (A8-A15) on the data bus.

Read/write logic:

- When the microprocessor is programming or reading one of the internal registers of the read/write logic accepts the I/O read (IOR) or low signal.
- Decodes least significant four address bits (A0-A7) and either writes the contents of the data bus addressed register or places the contents of the addressed register onto data bus.
- During DMA cycles the Read/write logic generates the I/O read and memory write or I/O write and memory read signals IOR control the data transfer between peripheral and memory device.

DMA channels:

The 8257 provides four identical channels labeled CH0, CH1, CH2 and CH3. Each channel has two-16 bit registers.

- DMA address register
- Terminal count register

a. DMA address register:

- It specifies the address of the first memory location to be accessed.
- It is necessary to load valid memory address in the DMA address register before channel is enabled.

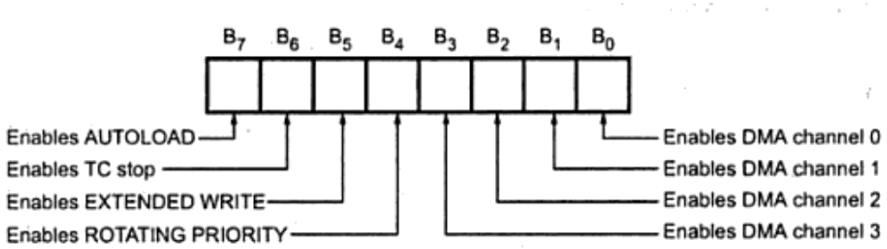
b. Terminal count register:

- The value loaded into the low order 14 bits of TCR specifies the number of DMA cycles minus one (N-1) before TC output is activated.
- Therefore, for N number of desired DMA cycles it is necessary to load the value N-1 into the low order 14 bits of TCR.
- MSB 2 bits specify the type of operation to be performed.

CONTROL LOGIC:

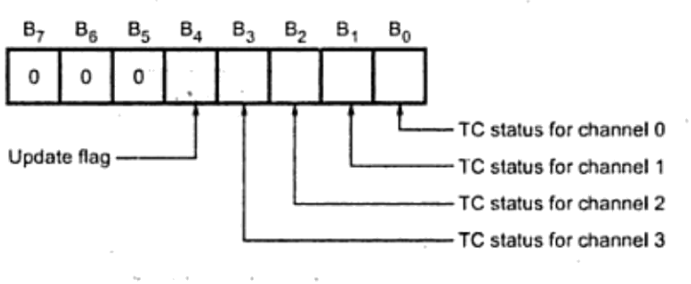
- It controls the sequence of operations during all DMA cycles by generating the appropriate control signals and the 16 bit address that specified the memory location to be accessed.
- It consists of mode set register and status register.
- Mode set register is programmed by the CPU to configure 8257 whereas the status register is read by CPU to check which channels have reached a terminal count condition and status of update flag.

Mode set register:



- LSB 4 bits are the enable 4 DMA channels.
- MSB 4 bits are the enable autoload, TC stop, extended write, rotating priority modes and terminal count registers.
- It is cleared by RESET input, this disabling all options, inhibiting all channels and preventing bus conflicts on power-up.

STATUS REGISTER:



- It indicates which channels have reached a terminal count condition and includes the update flag.
- The Tc status bit=1, terminal count has been reached for that channel.
- Tc bit remains set until the status register is read or the 8257 is reset.
- Update flag =1, 8257 is executing update cycle
- In update cycle 8257 lad parameters in channel 3 to channel 2.

PRIORITY RESOLVER:

- It resolves the peripherals request. It can be programmed to work into two modes, either fixed mode or rotating priority mode.

PREVIOUSLY ASKED ANNA UNIVERSITY QUESTIONS

PART A

1. What are the different ways to end the interrupt execution in 8259 programmable Interrupt controller? April/May 2011
2. What is the function of Scan section in 8279 programmable keyboard/Display Controller? April/May 2011
3. What are the applications of D/A converter interfacing with 8255?May/June 12
4. What is keyboard interfacing? May/June 12
5. Draw the 'Mode Word' format of 8251 USART. Nov/Dec 2011.
6. State the use of ISR and PR registers in 8259 PIC. Nov/Dec 2011
7. What are the output terminals in USART 8251? May/June 13
8. What are the different peripheral interfacing used with 8085 microprocessor? May/June 13
9. What is the need for 8259 PIC? Nov/Dec 2013
10. What are the basic modes of operations of 8255? Nov/Dec 2013
11. What are the features of 8259?May/June 14
12. How data is transmitted in asynchronous serial transmission? May/June 14
13. What are the modes of operations used in 8253? (Nov/Dec-14)
14. What is an USART? (Nov/Dec-14)
15. What are the internal registers available in 8259 PIC? (April/May-15)
16. Distinguish between synchronous and asynchronous transmission? (April/May-15)
17. Write the control value for 8255 PPI when PORT A and PORT B are inputs in simple I/O mode (Nov/Dec-15)
18. What are the working modes of 8254 timer(Nov/Dec-15)
19. How is DMA initiated? How the DMA operations perform in Microprocessor? (Apr/May 2018)
20. What is keyboard interfacing? (May/June 2012& Nov/Dec 2017) or How is keyboard interfaced with microprocessor?
21. Give the difference between maskable and non-maskable interrupts Nov/Dec 2017
22. How is DMA initiated? How the DMA operations perform in Microprocessor? (Apr/May 2018)

PART B

1. Explain the operating modes of 8255 programmable peripheral interface. (8) Nov/Dec 2011,Nov/Dec 13 (or) Explain the operation of 8255 PPI Port A programmed as input and output in Mode 1 with necessary handshaking signals. (8) April/May 2011, May/June 14. (or) Draw and explain the functional block diagram of 8255 PPI.(8) May/June 13, Nov/Dec-14,Nov/Dec-15
2. With functional block diagram, explain the operation and programming of 8251 USART in detail. (16) April/May 2011 , May/June 13, May/June 12
3. Draw and explain the functional block diagram of 8259 programmable interrupt controller. May/June 12, Nov/Dec-14

4. Show and explain the ADC interfacing with 8085 microprocessor. (8) (or) Why do we need A/D converter and D/A converter? Draw and describe the interfacing of A/D and D/A converter interfacing to 8085 microprocessor. (8) May/June 12, Nov/Dec 2011, April/May2011, Nov/Dec-14, April/May-15
5. Describe the comparison of I/O mapped and memory mapped I/O interfacing. May/June 12
6. Draw the logical block diagram of 8279 keyboard display controller and explain. Nov/Dec 2011, Nov/Dec 13. May/June 13, Nov/Dec-14, April/May-15,Nov/Dec-15
7. Draw the control word of 8253/8254 timer/counter and explain the operating modes of 8253/8254 timer/counter. (8) Nov/Dec 2011, May/June 14(or) Draw and explain the functional block diagram of 8253/8254 timer. (8) May/June 13,Nov/Dec-15
8. Draw the block diagram of 8255 (PPI) and explain its various operating modes Nov/Dec 2017& April/May-18
9. With a neat diagram, explain the internal architecture of keyboard and display controller IC-8279. Nov/Dec 2017
10. Draw the functional diagram of 8251 and explain its block in detail. April/May-18

MAILAM ENGINEERINGCOLLEGE

MAILAM, 604304

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

Subcode/SubName: EE6502 MICROPROCESSORS AND MICROCONTROLLERS **Year/ Sem: III / V**

UNIT V MICRO CONTROLLER PROGRAMMING & APPLICATIONS 9

Data Transfer, Manipulation, Control Algorithms& I/O instructions – Simple programming exercises- key board and display interface – Closed loop control of servo motor- stepper motor control – Washing Machine Control.

Updated Question: PART A &PART B

Q.No.26	Page No: 06	(Nov/Dec 2017)
Q.No.18	Page No: 04	(Nov/Dec 2017)
Q.No.27	Page No: 06	(Apr/May 2018)
Q.No.28	Page No: 06	(Apr/May 2018)
Q.No.07	Page No: 26	(Nov/Dec 2017)
Q.No.10	Page No: 43	(Nov/Dec 2017)
Q.No.06	Page No: 23	(Apr/May 2018)
Q.No.08	Page No: 30	(Apr/May 2018)

TEXT BOOKS:

1. Krishna Kant, "Microprocessor and Microcontrollers", Eastern Company Edition, Prentice Hall of India, New Delhi , 2007.
2. R.S. Gaonkar, 'Microprocessor Architecture Programming and Application', with 8085, Wiley Eastern Ltd., New Delhi, 2013.
3. Soumitra Kumar Mandal, Microprocessor & Microcontroller Architecture, Programming & Interfacing using 8085,8086,8051,McGraw Hill Edu,2013.

REFERENCES:

1. Muhammad Ali Mazidi & Janice Gilli Mazidi, R.D.Kinely 'The 8051 Micro Controller and Embedded Systems', PHI Pearson Education, 5th Indian reprint, 2003.
2. N.Senthil Kumar, M.Saravanan, S.Jeevananthan, 'Microprocessors and Microcontrollers', Oxford,2013.
3. Valder – Perez, "Microcontroller – Fundamentals and Applications with Pic," Yeesdee Publishers, Tayler & Francis, 2013.

Prepared By

1. Mrs.J.Srivandhana.,AP/MCA

2. Mrs.T.Kala.,AP/MCA

Verified By

HOD/MCA

Approved By

21/6/18

Principal

PART-A

1. Write a program using 8051 assembly language to change the data 55h stored in the lower byte of the data pointer register to AAH using rotate instruction?

```
MOV DPL,#55H
MOV  A,DPL
RL A
LABEL : SJMP Label
```

2. Explain the contents of the accumulator after the execution of the following program segments?

```
MOV A,#3CH
MOV R4,#66H
ANL A,R4
A 3C
R4 66
A 24
```

3. Write a program to load accumulator a,DPH and DPL with 30H?

```
MOV A,#30
MOV  DPH,A
MOV DPL,A
```

4. Write a program to perform multiplication of 2 nos using 8051? [Nov/Dec-2013]

```
MOV A,#data 1
MOV B,#data 2
MUL AB
MOV DPTR,#5000
MOV @DPTR,A(lower value)
INC DPTR
MOV A,B
MOVX@DPT
R,A
```

5. Write a program to mask the 0 th&7 th bit using 8051?

```
MOV A,#data
ANL A,#81
MOV DPTR,#4500
MOVX @DPTR,A
LOOP SJMP LOOP
```

6. Write about CALL statement in 8051?(Nov/Dec-14)

There are two subroutine CALL instructions. They are

- LCALL(Long CALL)
- ACALL(Absolute CALL)

Each increments the pc to the 1 st byte of the instruction & pushes them in to the stack.

7. **Write about the jump statement? [Nov/Dec-2011]**

There are three forms of jump. They are

LJMP(Long-jump)- address 16
AJMP(Absolute jump)-address 11
Sjmp(short jump)-relative address

8. Write a program to load accumulator DPH & DPL using 8051? [April/May-2011]

MOV A,#30
MOV DPH,A
MOV DPL,A

9. Write a program to find 2's complement using 8051? [May/June 2016] MOV A,R0
CPL A
INC A

10. Write a program to add two 8-bit numbers using 8051?

MOV A,#30H
ADD A,#50H

11. Write a program to swap two numbers using 8051?

MOV A,#data
SWAP A

12. Write a program to subtract two 8-bit numbers &exchange the digits using 8051?

MOV A,#9F
MOV R0,#40
SUBB A,R0
SWAP A

13. Write a program to subtract the contents of R1 of bank 0from the contents of R0 of bank 2 using 8051?

MOV PSW,#10
MOV A,R0
MOV PSW,#00

SUBB A,R1

14. What are the tasks involved in keyboard interfacing? [May/June-2012]

The task involved in keyboard interfacing are sensing a keyboard interfacing are sensing a key actuation, de bouncing the key and generatingkeycodes(decoding the key).these task are performed software if the keyboard is interfaced through ports and they are performed by hardware if the keyboard is interfaced through 8279.

15. How a keyboard matrix is formed in keyboard interface ? [May/June 2016]

The return lines RL0 to RL7 of 8279 are used to form the columns of keyboard matrix.in decoded scan the scan lines SLO to SL3 of 8279 are used to form the rows of keyboard matrix. In encoded scan mode, the output lines of external decoder are used as rows of keyboard matrix.

16. What is scanning in keyboard and what is scan time?

The process of sending a zero to each row of a keyboard matrix and reading the columns for key actuation is called scanning. the scan timeis the time taken by the processor to scan all the rows one by one starting from first row and coming back to the first row.

17. What is scanning in display and what is the scan time?

In display devices the process of sending display codes to 7-segment LED'S to display the led's one by one is called scanning. The scan time is the time taken to display all the 7-segment LED'S one by one, starting from first LED and coming back to the first LED again.

18. Write about program status word.(May/June 14)(April/May-15)(Nov/Dec-15)[Nov/Dec 2017]

The Program Status Word (PSW) keeps the current status of the arithmetic and logical operations in different bits. The 8051 has four math flags mat respond automatically to the out of arithmetic and logic operations and 3 general purpose user flags that can be set 1 or 0 by the programmer as desired.

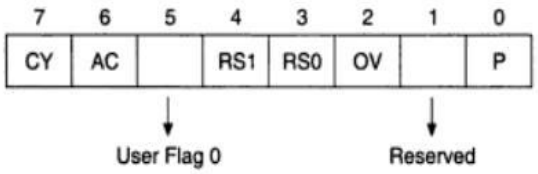


Figure 10.5 The format of the Program Status Word.

19. State the functions performed by JBC and CJNE instructions in 8051 microcontroller. (May/June 14)(Nov/Dec-14)

JBC bit, rel (Jump if bit is set and clear bit)

If the indicated bit is a 1, branch to the address indicated; otherwise, proceed with the next instruction. In either case, clear the designated bit. No flags are affected.

CJNE <destbyte>, <src.byte>, rel (Compare and jump if not equal)

The CJNE compares the magnitudes of the first two operands, and branches if their values are not equal. The carry flag is set, if the unsigned integer value of <dest-byte> is less than the unsigned integer value of <src-byte>; otherwise, the carry is cleared. Neither operand is affected.

The first two operands allow four addressing mode combinations—the accumulator may be compared with any directly addressed byte or immediate data and any indirect RAM location, or the working register can be compared with an immediate data.

CJNE A, direct, rel
CJNE A, #data, rel
CJNE Rn, #data, rel
CJNE @Ri, #data, rel (where Ri = RO or R1)

20. Give some example of input devices to microprocessor-based systems? [April/May-2011][April/May-15]

The input devices used in the microprocessor- based system are keyboards, DIP switches, ADC, floppy disc, etc.

21. List out the difference between MOV and MOVX instruction(Nov/Dec-15)

- MOV performs a bit or a byte transfer from the source operand to the destination operands.

MOV <dest.bit>, <scr.bit> (Move bit data)

- The Boolean variable indicated by second operand is copied into the location specified by
- the first operand one of the operand must be carry flag.
- The MOVX instructions transfer data between the accumulator and a byte of the external
- data memory.
- • MOVX <dest-byte>,<src-byte> (Move external)

The MOVX instructions transfer data between the accumulator and a byte of the external data memory.

22. What is the use of PSW? (Nov/Dec-16)

•The PSW includes the instruction address, condition code, and other fields. In general, the PSW is used to control instruction sequencing and to hold and indicate the status of the system in relation to the program currently being executed.

23.Mention any four data transfer instructions of 8051 microcontroller. (Nov/Dec-16)

(i)MOV A, Rn
(ii)MOV A, direct
(iii)MOV A,#data
(iv)MOV A,@ Ri

23. Explain the function of DJNZ instructions? (April/May- 2017)

DJNZ(Decrement and jump if not zero)

The DJNZ decrements by 1, the contents of the location indicated and branches to the address indicated by the second operand if the resulting value is zero.

The location may be a register or a directly addressed byte and no flags are affected.

25. What is meant by bit oriented instructions (April/May- 2017)

Similar to logical instructions, bit oriented instructions perform logic operations. The difference is that these are performed upon single bits.

Example:

```
CLR  C
SET   B
CPL C
```

26.What is Baud rate?[Nov/Dec 2017]

Baud rate

The baud rate, in mode 0, is fixed at 1/12th of the oscillator frequency. The baud rate in mode 2 is either l/64th or l/32th of the oscillator frequency depending on the bit value of SMOD in the special function register PCON.

If SMOD = 0, the baud rate is 1 64th of the oscillator frequency. If SMOD = 1, the baud rate is l/32th of the oscillator frequency.

For mode 1 and mode 3, the baud rates are variable. The baud rate is determined by the Timer 1 overflow rate, i.e.

$$\text{Baud rate} = \frac{\text{Timer 1 overflow rate}}{n}$$

where n is an integer and its value is either 32 or 16

27.What is use of data pointer register? [Apr/May 2018]

The Data Pointer (DPTR) is the 8051's only user-accessable 16-bit (2-byte) register. The Accumulator, "R" registers, and "B" register are all 1-byte values.

DPTR, as the name suggests, is used to point to data. It is used by a number of commands which allow the 8051 to access external memory. When the 8051 accesses external memory it will access external memory at the address indicated by DPTR.

While DPTR is most often used to point to data in external memory, many programmers often take advantge of the fact that it's the only true 16-bit register available. It is often used to store 2-byte values which have nothing to do with memory locations.

28.What is the advantage of closed loop control system for interfacing?[Apr/May 2018]

An advantage of the closed - loop control system is the fact that the use of feedback makes the system response relatively insensitive to external disturbances and internal variations in system parameters.

It is thus possible to use relatively inaccurate and inexpensive components to obtain the accurate control of a given plant, whereas doing so is impossible in the open -loop case.

From the stability point of view, the open - loop control system is easier to build because system stability is not a major problem. On the other hand, stability is a major problem in the closed - loop control system, which may tend to overcorrect errors that can cause oscillations of constant or changing amplitude.

29. List the different types of 8051 instructions. [Apr 2010, Nov 2011]

The different types of 8051 instructions are:

a. Logical Instructions
b. Arithmetic Instructions
c. Data transfer Instructions
d. Branch Instructions
e. Jump & CALL Instruction

30. What are the various operations performed by Boolean variable instructions of 8051? [Apr 2010, Nov 2011]

Boolean variable instructions perform the following operations

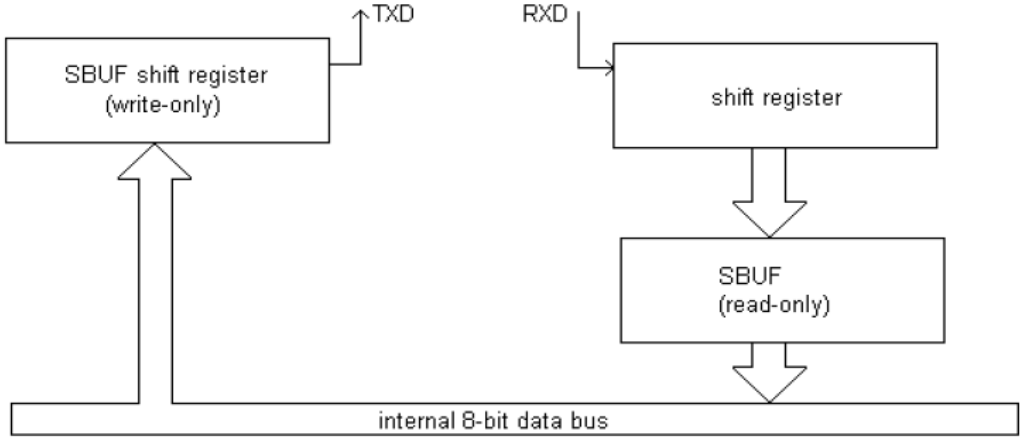
BOOLEAN OPERATOR MNEMONICS

- AND(AND logical) ANL
- OR(OR logical) ORL
- NOT(COMPLEMENT) CPL
- XOR(Exclusive OR logical) XRL

31. What is the operation of the given 8051 microcontroller instructions XRL A, direct? [Apr 2011]

XOR each bit of A with the same bit of the direct RAM address and the result is stored in A (Acc).

32. Draw the flow chart for the programming of serial port of 8051.[Apr 2012]



33. What are the applications of 8051 microcontroller? [Apr 2012]

The applications of 8051 microcontroller are:

- Automobile
- Aeronautics
- Mobile communication
- Robotics
- Remote sensing etc.,

34. How is the pulse generated from microcontroller for stepper motor control.[Apr 2013]

To cause the stepper to rotate, we have to send a pulse to each coil in turn. The 8051 does not have sufficient drive capability on its output to drive each coil, so there are a number of ways to drive a stepper, Stepper motors are usually controlled by transistor or driver IC like ULN2003.

Driving current for each coil is then needed about 60mA at +5V supply.

35. Why do we need opto-isolator circuit between microcontroller and the stepper motor? [Nov 2011]

Opto-isolator are widely used to isolate the stepper motor's EMF voltage and keep it form damaging the digital microcontroller system.

36. Why interfacing is needed for 1/0 devices? [Nov 2009]

Generally I/O devices are slow devices. Therefore the speed of I/O devices does not match with
the speed of microprocessor. And so an interface is provided between system bus and I/O devices.

37. What is the operation carried out when 8051 executes the instruction MOVC A, @ A +DPTR? [Nov 2009]

This instruction loads the accumulator from the contents of program memory whose address is
given by the sum of the contents of accumulator and contents of DPTR register (A) ‹—((A) + (DPTR))

38. Explain PUSH and POP instructions in 8051.

PUSH-The stack pointer is incremented by one. The contents of the indicated variable Is then copied into the internal RAM location addressed by the stack pointer. POP-Reverse of PUSH operation.

39. How many ports are bit addressable in an 8051 microcontroller?

In 8051 there are many bit-addressable registers such as A (ACC), B, SCON,PCON, TCON, p0, p1, p2, p3.

40. Write a program to find the 2's complement using 8051?

```
MOV A, R0
CPL A
INC A
```

41. Explain rotate instructions of 8051.

RL A,
RLC A,
RR A,
RRC A

42. Explain DAA instruction of 8051.

Decimal adjust accumulator for addition bytes.

43. What does the mnemonics "LCALL" and "ACALL" stands for?

There are two subroutine CALL instructions. They are LCALL (Long CALL) ACALL (Absolute CALL). Each increments the PC to the 1st byte of the instruction & pushes them in to the stack.

44. What are the uses of PWM in motor control using microcontroller?

The speed of the dc motor depends on the applied voltage. The average applied dc voltage and power can be varied using a technique called pulse width modulation. In this technique the dc power supply is not a voltage of fixed amplitude ie it is a pulsating

45. Calculate the reload value of timer1 for achieving a baud rate of 4800in 8051 for a crystal
frequency of 11.0592MHz?

$$TH = 256 - \frac{k * \text{Oscillatory frequency}}{384 * \text{Baud rate}}$$
$$= 256 - 1*11.0592*10E6 / 384*4800$$
$$= 250 = FAH$$

46. List the features of ADC0804?

i) 8-bit successive approximation ADC
ii) Access time is 135ns
iii) Conversion time is 100µs
iv) It has an on chip clock generator
v) It does not require any zero adjustment
vi) It operates on single 5V power supply.

47. Give the PSW setting for making register bank 2 as default register bank in 8051microcontroller .

```
MOV PSW, #10 ; SELECT BANK 2
MOV A, R0 ; (A) ‹— (R0) FROM BANK 2
MOV PSW, #00 ; SELECT BANK 0
CLR C ; CLEAR CARRY
SUBB A, R1 ; A ‹— A- (R1) FROM BANK 0
```

The above program is to subtract the contents of R1 of BANK0 from the contents of RO of Bank 2.

PART-B

1. Explain in detail about the data transfer instructions?[April/May-2011]

Data transfer instruction

It is divided into three classes

1. General purpose transfers
2. Accumulator-specific transfer
3. Address object transfer

General purpose transfers

- MOV performs a bit or a byte transfer from the source operand to the destination operands.
- PUSH increments the SP register and then transfer a byte from the source operand to the stack elements, currently addressed by SP.
- POP transfers a byte operand from the stack element addressed by the SP register to the destination operand and then decrements SP.

MOV <dest.byte>, <scr.byte> (Move byte variable)

The byte variable indicates by the second operand is copied into the location specified by the first operand. The source byte is not affected. No other register or flag is affected.

MOV A, Rn
(A) ← (Rn) MOV A,direct
MOV A, @Ri MOV A, #data
MOV Rn, A
MOV Rn, direct MOV Rn, #data

Content of a register in register bank cannot be transferred to another register in register bank, since the instruction MOV Rm, Rn is not present.

MOV <dest.bit>, <scr.bit> (Move bit data)

The Boolean variable indicated by second operand is copied into the location specified by the first operand one of the operand must be carry flag.

MOV C, bit

(C)← (bit)
MOV bit, C

PUSH direct (push onto stack)

The stack pointer is incremented by one. The contents of indicated variable are then copied into the internal RAM location addressed by stack pointer. The instruction can be used to modify the contents of PSW in case the stack location coincides with SFR.

(SP)←(SP)+1

((SP)) ← (direct)

POP direct (pop from stack)

The content of internal RAM location addressed by stack pointer are read, and the stack pointer is decremented by one. The value read is then transferred to the directly addressed byte indicated.

(Direct) ← (SP)
(SP)← (SP)-1
Eg:
POP DPTR

Accumulator-specific transfer

This transfers operations are provided:

- XCH exchanges the byte source operand with register A
  XCHD exchanges the low-order nibble of the byte source operand with the low-order nibble of register A.
- MOVX performs a byte move between the external data memory and register A.
- MOVC performs the move of a byte from the program memory to register A.

XCH A, <byte> (Exchange accumulator with byte variable)

Exchange the byte source operand with register A. the source operand can use register direct or register indirect addressing.

XCH A, Rn
(A)→(Rn)
(B) XCH A,
direct XCH
A, @ R0

XCHD A, @Ri (Exchange digit)

Exchanges the low-order nibble of the byte source operand with the low-order nibble of register A.

XCHD A, @ R0

MOVX <dest-byte>,<*src-byte*> *(Move external)*

The MOVX instructions transfer data between the accumulator and a byte of the external data memory.

MOVX A, @Ri

(A) ← ((Ri))

MOVX @Ri, A

MOVX A, @DPTR

MOVX @DPTR, A

MOVC A, @A + <base-reg> (Move code byte)

The MOVC instructions load the accumulator with a code byte, or a constant from the program memory. No flags are affected.

MOVC A, @A^DP

MOVC A, @A + PC

Address-object Transfer

MOV DPTR, #data loads 16 bits of immediate data into a pair of destination registers, DPH and DPL.

2. Explain in detail about the manipulation/arithmetic & logical instructions of 8051 micro controller? [April/May-2011]

1. ARITHMETIC INSTRUCTIONS

The 8051 provides the basic mathematical operations like addition, subtraction, multiplication, division, increment, decrement, etc.

Addition

There are four addition operations.

- ADD performs an addition between the register A and the second source operand.
- ADDC (add with carry) performs an addition between the register A and the second source operand; adds 1 if the C flag is found previously set.
- DA (decimal-add-adjust for BCD addition) performs a correction to the sum which resulted from the binary addition of two two-digit decimal operands.
- Performs an addition of the source operand and 1.

ADD A, <src.byte> (Add)

ADD adds the byte variables indicated to the accumulator, leaving the result in the accumulate! Four source operand addressing modes are allowed—register, direct, register indirect, all immediate.

ADD A, Rn

ADD A, direct

ADD A, @Ri

ADD A, #data

ADDC A, <src.byte> (Add with carry)

The ADDC simultaneously adds the byte variables indicated, the carry flag and the accumulator contents, leaving the result in the accumulator. Four source operand addressing modes allowed—register, direct, register indirect, and immediate.

ADDC A, Rn
ADDC A, direct
ADDC A, @Ri

DAA (Decimal-adjust accumulator for addition)

The DAA adjusts the 8-bit value in the accumulator, resulting from the earlier addition o two variables (each in packed-BCD format), producing two 4-bit digits. Any ADD ADDC instruction may have been used to perform the addition. The algorithm followed is as follows:

1. If the value of the lower nibble in ACC is greater than 9, or if AC flag is set, then 6 is added to ACC.
2. If the value of the higher nibble is now greater than 9, or if CY flag is set, then 6 is added to
   the higher nibble of ACC.

All flags are affected.

INC <byte> (Increment)

INC increments the indicated variable by 1. No flags are affected.

Four addressing modes are allowed—accumulator, register, direct, or register indirect.

INC A
INC Rn
INC direct
INC @Ri

INC DPTR (Increment Data Pointer)

Increments the 16-bit data pointer by 1. No flags are affected. (DPTR)

Subtraction

There are two subtraction operations.

- SUBB, (subtract with borrow) performs a subtraction of the second source operand from the first operand (the accumulator), subtracts 1 if the C flag is found previously set.
- DEC (decrement) performs a subtraction of 1 from the source operand.

SUBB A, <src-byte> (Subtract with borrow)

SUBB subtracts the indicated variable and the carry flag together from the accumulator, leaving the result in the accumulator. The source operand allows four addressing modes—register, direct, register indirect, and immediate.

SUBB A, Rn

SUBB A, direct

SUBB A, @Ri

SUBB A, #data

DEC byte (Decrement)

The variable indicated is decremented by 1. No flags are affected. Four operand address modes are allowed—accumulator, register, direct, and register indirect.

DEC A

DEC Rn

DEC direct

Multiplication

MUL performs an unsigned multiplication of register A by register B, returning a double-result.

MUL AB (Multiply)

MUL A B multiplies the unsigned 8-bit integers in the accumulator and register B. The l order byte of the 16-bit product is left in the accumulator, and the high-order byte in register If the product is greater than 255 (OFFH), the overflow flag is set; otherwise it is cleared, carry flag is always cleared.

Division

The DIV performs an unsigned division of register A by register B.

DIV AB (Divide)

The DIV AB divides the unsigned 8-bit integer in the accumulator by the unsigned 8-bit integer in register B. The accumulator receives the integer part of the quotient; register B receives the integer remainder. The carry and the overflow flags will be cleared. An exception is: if B had originally contained 00H, the values returned to the accumulator and B register will be undefined and the overflow flag will be set. The carry flag is cleared in any case.

2. LOGIC INSTRUCTIONS

The 8051 performs the basic logic operations on both bit and byte operands

Single-operand Operations

There are seven single-operand logical operations as follows:

- CLR is used to set either the register A, the carry, or any direct addressed bit to 0.
- SETB sets either the carry or any direct addressed bit to 1.

- CPL either forms the 1's complement of the operand in register A or the 1's complement of the carry or any direct addressed bit.
- RL, RLC, RR, RRC, SWAP. Five rotate operations can be performed on register A—RL (rotate left), RR (rotate right), RLC (rotate left through carry), RRC (rotate right

through carry), and SWAP (swap nibbles).-For RLC and RRC, the C flag becomes equal to the last bit rotated out. SWAP rotates the register A four places left to exchange bits 3 through 0 with bits 7 through 4.

CLR A (Clear accumulator)

The accumulator is cleared (all bits set to 0). No flags are affected.

CLR bit (Clear bit)

The indicated bit is cleared (reset to 0). No other flags are affected. CLR can operate on the carry flag or any directly addressable bit.

CLR C

CLR bit

SETB <bit> (Set bit)

The SETB sets the indicated bit to 1. The SETB can operate on the carry flag or any directly addressable bit. No other flags are affected.

SETB C

SETB bit

CPL A(Complement accumulator)

Each bit of the accumulator is logically complemented (1 's complement). No flags are affected.

CPL bit (Complement bit)

The bit variable specified is complemented. A bit which had been 1 is changed to 0 and vice-versa. No other flags are affected. The CPL can operate on the carry or any directly addressable bit.

CPL C

CPL bit

RL A (Rotate accumulator left)

The eight bits in the accumulator are rotated 1 bit to the left. Bit 7 is rotated into the bit 0 position. No flags are affected.

RLC A (Rotate accumulator left through the carry flag)

The eight bits in the accumulator and the carry flag are together rotated 1 bit to the left. Bit 7 moves into the carry flag; the original state of the carry flag moves into the bit 0 position. No other flags are affected.

RR A (Rotate accumulator right)

The eight bits in the accumulator are rotated 1 bit to the right. Bit 0 is rotated into the bit 7.

RRC A (Rotate accumulator right through carry flag)

The eight bits in the accumulator and the carry flag are together rotated 1 bit to the right. Bit 0 moves into the carry flag, the original value of the carry flag moves into the bit 7 positions. No other flags are affected.

SWAP A (Swap nibbles within the accumulator)

SWAP A interchanges the low- and high-order nibbles (4-bit fields) of the accumulator (bits 3-0 and bits 7-4). The operation can also be thought of as a 4-bit rotate instruction. No flags are affected.

Two-operand Operations

Three two-operand logical instructions ANL, ORL and XRL are provided to perform AND, OR and XOR operations respectively on bit as well as byte operands.

ANL <dest.byte>, <src.byte> (Logical AND for byte variables)

ANL performs the bit-wise logical AND operation between the variables indicated and stores the results in the destination variable. No flags are affected.

When the destination is the accumulator, the source can use register, direct, register indirect, or immediate addressing; when the destination is a direct address, the source can be the accumulator or the immediate data.

ANL A, Rn ANL
A, direct ANL A,
@Ri ANL A,
#data ANL
direct, A ANL
direct, #data

ANL C, <src.bit> (Logical AND for bit variables)

The carry flag is modified by ANDing with the source bit or its logical complement. No flags are affected. Only the direct bit addressing is allowed for the source operand.

ANL C, bit
ANL C, /bit
ANL C, bit
ANL C, /bit

ORL <dest-byte>,<src-byte>(Logical OR for byte variables)

The ORL performs the bit-wise logical OR operation between the indicated variables, store the results in the destination byte. No flags are affected. The two operands allow six addressing mode combinations. When the destination is the accumulator, the source can use register, direct, register indirect, or immediate addressing; when the destination is a direct address, the source can be the accumulator or immediate data.

ORL A, Rn ORL
A, direct ORL A,

@Ri ORL A,
#data ORL direct,
A ORL direct,
#data

ORL C, <src-bit> (Logical OR for bit variables)

The carry flag is modified by ORing with the source bit or its iogicaV complement.

Noother flags are affected.

ANL C, <src.bit> (Logical AND for bit variables)

The carry flag is modified by AND ing with the source bit or its logical complement. No other flags are affected. Only the direct bit addressing is allowed for the source operand.

ANL C, bit

ORL <dest-byte>, <src-byte> (Logical OR for byte variables)

The ORL performs the bit-wise logical OR operation between the indicated variables, storing the results in the destination byte. No flags are affected. The two operands allow six addressing mode combinations. When the destination is the accumulator, the source can use register, direct, register indirect, or immediate addressing; when the destination is a direct address, the source can be the accumulator or immediate data.

ORL A, Rn
ORL A, direct
ORL A, @Ri
ORL A, #data
ORL direct, A
ORL direct, #data

ORL C, <src-bit> (Logical OR for bit variables)

The carry flag is modified by ORing with the source bit or its logical complement. No other flags are affected.

ORL C, bit
ORL C, /bit

XRL <dest-byte>, <scr-byte> (Logical XOR for byte variables)

The XRL performs the bit-wise logical XOR operation between the indicated variables, storing the results in the destination. No flags are affected. The two operands allow six addressing mode combinations. When the destination is the accumulator, the source can use register, direct, register indirect, or immediate addressing. When the destination is a direct address, the source can be the accumulator or immediate data.

XRL A, Rn XRL

A, direct XRL A,
#data XRL direct,
A XRL direct,
#data

3. Explain in detail about control transfer instructions? [April/May-2011]

There are three classes of control transfer operations.

1. Unconditional calls, returns and jumps
2. Conditional jumps
3. Interrupts

All control transfer operations cause, some upon a specific condition, the program execution to continue at a non-sequential location in the program memory.

Unconditional Calls, Returns and Jumps

Unconditional calls, returns and jumps transfer the control from the current value of the Program Counter to the target address. Both direct and indirect transfers are supported. The three transfer operations are described below.

- ACALL and LCALL push the address of the next instruction onto the stack (PCL to low-order address, PCH to high-order address) and then transfer the control to the target address. Absolute Call is a 2-byte instruction and used when the target address is in the current 2K page. Long Call is a 3-byte instruction that addresses the full 64K program space. RET

  (Return from subroutine) and RETI (Return from Interrupt) transfer control to return address saved on the stack and decrement SP register by 2.
- AJMP, LJMP and SJMP are unconditional branch instructions used to transfer control to the target operand. The operation of AJMP and LJMP are analogous to ACALL and LCALL. The SJMP (short jump) instruction provides for transfers within a 256 byte range centred about the starting address of the next instruction (-128 to +127). The PC-relative short jump facilitates the relocatable code.
- JMP @ A + DPTR performs a jump relative to the DPTR register. The operand in the register A is used as the offset (0-255) to the address in the DPTR register. Thus, the effective destination for a jump can be anywhere in the program memory space. This indirect jump is also useful for implementing N-way branches.

ACALL addr11 (Absolute call)

ACALL unconditionally calls a subroutine located at the indicated address. Since ACALL is a 2-byte instruction, PC is incremented by 2 to point to the next instruction. The destination address is obtained by successively concatenating the five high-order bits of the

incremented PC, op-code bits 7-5, and the second byte of the instruction. The subroutine called must, therefore, start within the same 2K block of the program memory as the first byte of the instruction following ACALL. No flags are affected.

LCALL addrW (Long call)

The LCALL calls a subroutine located at the indicated address. Since LCALL is a 3-byte instruction, PC is incremented by 3 to point to the next instruction. The destination address is mentioned in absolute term in the instruction as addr 16. No flags are affected.

RET (Return from subroutine)

The RET pops the return address from the stack and loads into the PC. Program execution continues at the resulting address. No flags are affected.

RETI (Return from interrupt)

The RETI pops the return address from the stack and loads into the PC and restores the interrupt logic to accept additional interrupts at the same priority level as the one just processed. Program execution continues at the resulting address.

AJMP addr11 (Absolute jump)

The AJMP transfers the program execution to the indicated address, which is formed at runtime by concatenating the high-order five bits of the PC (after incrementing the PC twice), opcode bits 7-5, and the second byte of the instruction. The destination must, therefore, be within the same 2K block of the program memory as the first byte of the instruction following AJMP.

LJMP addr16 (Long jump)

The LJMP causes an unconditional branch to the indicated address. No flags are affected.

SJMP rel (Short jump)

Program control branches unconditionally to the address indicated. The branch destination is computed by adding the signed displacement in the second instruction byte to the PC, after incrementing the PC twice. Therefore, the range of destinations allowed is from 128 bytes preceding this instruction to 127 bytes following it.

NOP (No operation)

Execution continues at the following instruction. Other than the PC, no registers or flags are affected.

Conditional Jumps

In the control transfer group, the conditional jumps perform a jump contingent upon a specific condition. The destination will be within a 256 byte range (-128 to +127) centered about the starting address of the next instruction.

The General Format is "Jcond .rel" where rel is the relative address specified. In all the cases, the branch destination is computed by adding the signed relative displacement to the PC, after incrementing the PC to the first byte of the next instruction.

JB bit, rel (Jump if bit set)

If the indicated bit is a 1, jump to the address indicated; otherwise, proceed with the next instruction. The bit tested is not modified. No flags are affected.

JBC bit, rel (Jump if bit is set and clear bit)

If the indicated bit is a 1, branch to the address indicated; otherwise, proceed with the next instruction. In either case, clear the designated bit. No flags are affected.

JC rel {Jump if carry is set)

I f the carry flag is set, branch to the address indicated; otherwise proceed with the next instruction. No flags are affected.

JNB bit, rel (Jump if bit not set)

If the indicated bit is a 0, branch to the indicated address; otherwise proceed with the next instruction. The bit tested is not modified. No flags are affected

JNC rel (Jump if carry not set)

I f the carry flag is a 0, branch to the address indicated; otherwise, proceed with the next instruction. The carry flag is not modified

JNZ rel (Jump if accumulator not zero)

If any bit of the accumulator is a 1, branch to the indicated address; otherwise, proceed with the next instruction. The accumulator is not modified. No flags are affected.

CJNE <destbyte>, <src.byte>, rel (Compare and jump if not equal)

The CJNE compares the magnitudes of the first two operands, and branches if their values are not equal. The carry flag is set, if the unsigned integer value of <dest-byte> is less than the unsigned integer value of <src-byte>; otherwise, the carry is cleared. Neither operand is affected. The first two operands allow four addressing mode combinations—the accumulator may be compared with any directly addressed byte or immediate data and any indirect RAM location, or the working register can be compared with an immediate data.

CJNE A, direct, rel
CJNE A, #data, rel
CJNE Rn, #data, rel
CJNE @Ri, #data, rel (where Ri = RO or R1)

DJNZ <byte>, <rel-addr> (Decrement and jump if not zero)

The DJNZ decrements by 1 the contents of the location indicated and branches to the address indicated by the second operand if the resulting value is not a zero. No flags are affected. The location (whose contents are decremented) may be a register or a directly addressed byte.

DJNZ Rn, rel

4. Explain the addressing modes used in 8051?(May/June 14)

The addressing modes are the ways of accessing data in register or in memory or be provided as an immediate value. The 8051 mnemonics are written with the destination address named first, followed by the source address.

The following addressing modes are used to access data:

1. Immediate addressing mode
2. Register addressing mode
3. Direct addressing mode
4. Register indirect addressing mode
5. Base register plus Index register indirect addressing mode.

Immediate Addressing Mode

When a source operand is a constant rather than a variable, then the constant can be embedded into the instruction itself. These kinds of instructions take two bytes and first one specifies the opcode and second byte gives the required constant. The operand comes immediately after the opcode. The mnemonic for immediate data is the pound sign (#). This addressing mode can be used to load information into any of the registers including DPTR register.

Examples:

MOV A, # 18H

MOV R6, # 25H

Register Addressing Mode

Register addressing accesses the eight working registers (R0 – R7) of the selected register bank. The least significant three bits of the instruction opcode indicate which register is to be used for the operation. One of the four banks of registers is to be predefined in the PSW before using register addressing instruction. ACC, B and DPTR can also be addressed in this mode.

Examples:

ADD A, R6

MOV A, Rn

Direct Addressing Mode

In the direct addressing mode, all 128 bytes of internal RAM and the SFRs may be addressed directly using the single - byte address assigned to each RAM location and each SFR. Internal

RAM uses address from 00H to 7FH to address each byte. The SFR addresses exist from 80H to FFH. (Refer Table 4.3).

Examples

MOV R2,61H

MOV 6FH, A

Register Indirect Addressing Mode

In this mode a register is used as a pointer to the data. If the data is inside the CPU, only registers RO and Rl are used for this purpose. When RO and Rl hold the addresses of RAM locations, they must be prect led by the "@" sign.

Examples

M0V @R1, A

Base register plus Index register indirect addressing mode

Only the program memory can **be** accessed **by** this mode. This mode is intended for reading lookup tables in the program memory. A 16 bit base register (DPTR or PC) points to the base of the lookup tables and accumulator carries the constant indicating table entry number. The address of the exact location of the table is formed by adding the accumulator data to the base pointer.

Example

MOVC A, @A + DPTR

5.**List some programs using 8051 microcontroller?**

1. Addition of two 16 bit using 8051 microcontroller: [April/May-2011]

```
CLR C

MOV A ,#DATA1

ADD A,#DATA2

MOV DPTR #8150

MOV X,@DPTR A

INC DPTR

 MOV #DATA M1

 ADD C #DATA M2

 MOV X @DPTR A

 SJMP HERE
```

2. Subtraction of two 16 bit using 8051 microcontroller:

```
CLR C
MOV A, #DATA1
SUB A, #DATA2
MOV DPTR #8150
MOV X,@DPTR A
SJMP 8109
```

3. Multiplication of two 16 bit using 8051 microcontroller:

```
MOV A, #DATA1
MOV B, #DATA2
MUL A, B
MOV DPTR #8500
MOV X,@DPTR A
INC DPTR
MOV A, B
MOV X,@DPTR A
HERE SJMP HERE
```

4. Division of two 16 bit using 8051 microcontroller:

```
MOV A ,#DATA1
MOV B ,#DATA2
DIV A,B
MOV DPTR #8500
MOV X,@DPTR
A INC DPTR
MOV A,B
MOV X,@DPTR A
HERE SJMP HERE
```

6. <u>An 8051 program for keyboard and display interface: [April/May-11][Nov/Dec-11][April/May-15](Nov/Dec 16)[Apr/May 2018]</u>

The key board here we are interfacing is a matrix keyboard. This key board is designed with a particular rows and columns. These rows and columns are connected to the microcontroller through its ports of the micro controller 8051.

We normally use **8*8 matrix key board**. So only **two ports of 8051** can be easily connected to the rows and columns of the key board.

- When ever a key is pressed, a row and a column gets shorted through that pressed key and all the other keys are left open.
- When a key is pressed only a bit in the port goes high. Which indicates microcontroller that the key is pressed. By this high on the bit key in the corresponding column is identified.
- Once we are sure that one of key in the key board is pressed next our aim is to identify that key. To do this we firstly check for particular row and then we check the corresponding column the key board.

To check the row of the pressed key in the keyboard, one of the row is made high by making one of bit in the output port of 8051 high . This is done until the row is found out. Once we get the row next out job is to find out the column of the pressed key. The column is detected by contents in the input ports with the help of a counter. The content of the input port is rotated with carry until the carry bit is set.

The contents of the counter is then compared and displayed in the display. This display is designed using a seven segment display and a BCD to seven segment decoder IC 7447.

The BCD equivalent number of counter is sent through output part of 8051 displays the number of pressed key.

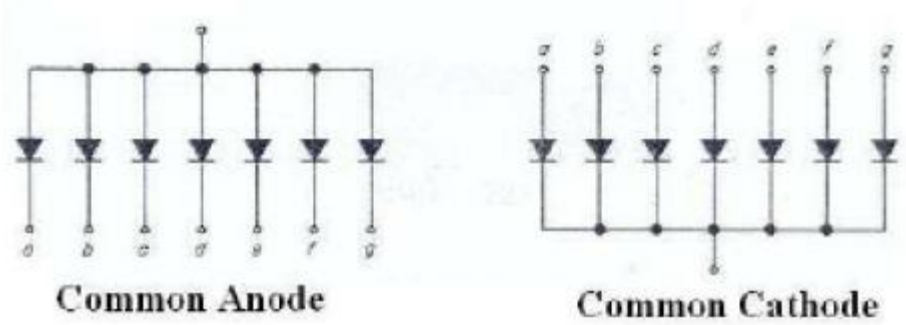
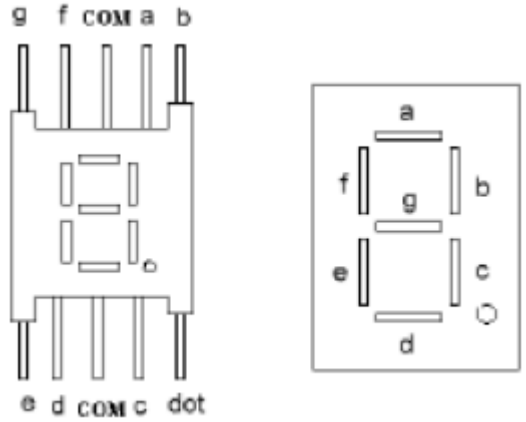
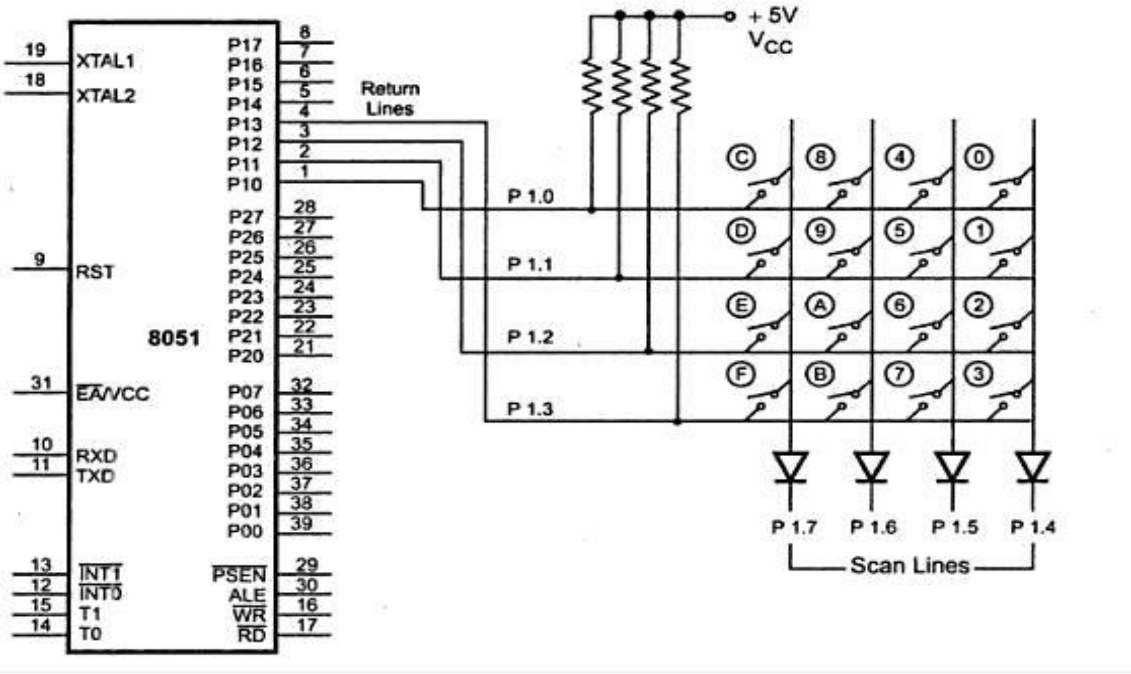


Fig 5.3 Interfacing LEDS to 8051 Microcontroller





The steps in algorithm are as follows:

1. The 8051 has 4 I/O ports P0 to P3 each with 8 I/O pins, P0.0 to P0.7,P1.0 to P1.7, P2.0 to P2.7, P3.0 to P3.7. The one of the port P1 (it understood that P1 means P1.0 to P1.7) as an I/P port for microcontroller 8051, port P0 as an O/P port of microcontroller 8051 and port P2 is used for displaying the number of pressed key.
2. Make all rows of port P0 high so that it gives high signal when key is pressed.
3. See if any key is pressed by scanning the port P1 by checking all columns for non zero condition.
4. If any key is pressed, to identify which key is pressed make one row high at a time.
5. Initiate a counter to hold the count so that each key is counted.
6. Check port P1 for nonzero condition. If any nonzero number is there in [accumulator], start column scanning by following step 9.
7. Otherwise make next row high in port P1.
8. Add a count of 08h to the counter to move to the next row by repeating steps from step 6.
9. If any key pressed is found, the [accumulator] content is rotated right through the carry until carry bit sets, while doing this increment the count in the counter till carry is found. 10. Move the content in the counter to display in data field or to memory location
11. To repeat the procedures go to step 2.

Program to interface matrix keyboard to microcontroller 8051

Start of main program:

to check that whether any key is pressed

```
start:  mov a,#00h
                        ;making all rows of port p1
        mov p1,a        zero
        mov a,#0fh
                        ;making all rows of port p1
        mov p1,a        high
press:  mov a,p2
        jz press        ;check until any key is pressed
```

after making sure that any key is pressed

```
        mov a,#01h      ;make one row high at a time
        mov r4,a
        mov r3,#00h     ;initiating counter
next:   mov a,r4
        mov p1,a        ;making one row high at a time
        mov a,p2        ;taking input from port A
        jnz colscan     ;after getting the row jump to check
                          column
        mov a,r4
        rl a            ;rotate left to check next row
        mov r4,a
        mov a,r3
        add             ;increment counter by 08 count
        a,#08h
```

```
                mov r3,a
                sjmp next
```

;jump to check next row

after identifying the row to check the colomn following steps are followed

```
col scan:  mov r5,#00h
                        ;rotate right with carry until get the
      in:  rrc a        carry
           jc out       ;jump on getting carry
           inc r3       ;increment one count
           jmp in
     out:  mov a,r3
           da a         ;decimal adjust the contents of counter
                            before display
           mov p2,a
           jmp start    ;repeat for check next key.
```

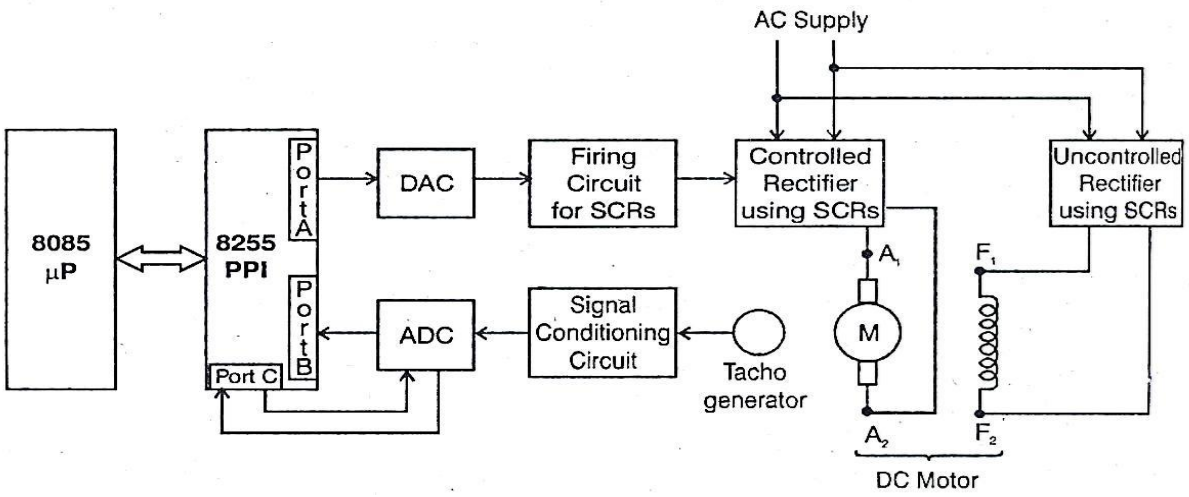
7. Explain in detail about the closed loop control of servo motor using 8051?[Nov/Dec-14] [April/May-2011](April/May 2017)[Nov/Dec 2017]

DC MOTOR INTERFACING WITH MICROPROCESSOR

The microprocessor based speed control system can be used to automatically control the speed of the motor. The speed of the DC motor is varied by varying the armature voltage and the field voltage is kept constant. A controlled rectifier using SCR develops the required armature voltage and the uncontrolled rectifier generates the required field voltage from the AC supply.

The microprocessor controls the speed of the motor by varying the firing angle of the SCRs in the controlled rectifier. A keyboard and displays in the system allow the operator to enter desired speed and display the actual speed.

Dc motor based speed control system



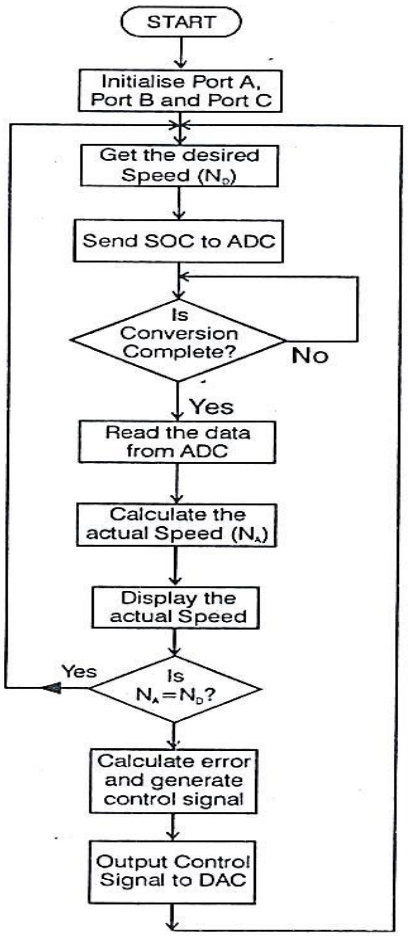
The speed of the dc motor is measured using tacho generator. It produces an analog voltage proportional to the speed of the motor. Then the analog signal is scaled to desired level by the signal conditioning circuit and converted into digital signal using ADC.

The ADC is interfaced to 8085 microprocessor through the port B and port C of 8255, programmable peripheral interface (PPI). The microprocessor can send a start of conversion (SOC) signal to ADC through port C. it can read digital data from port B of 8255. The digital data is proportional to the speed of DC motor.

Thus the microprocessor calculates the actual speed (NA) and displays it. Also the microprocessor compares the actual speed with the desired speed (ND) entered by the operator. If there is a difference between NA and ND then the error is calculated.

The error will be modified by a digital control algorithm (PI or PID or Fuzzy logic) to produce a digital control signal. This digital signal is converted into analog signal by DAC. The analog control signal is used to vary the firing angle of SCRs in the controlled rectifiers.

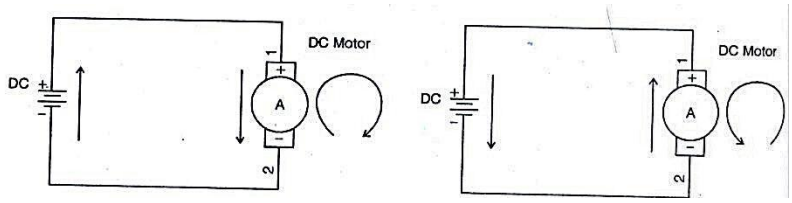
Flow chart for DC motor speed control



DC MOTOR INTERFACING WITH MICROCONTROLLER

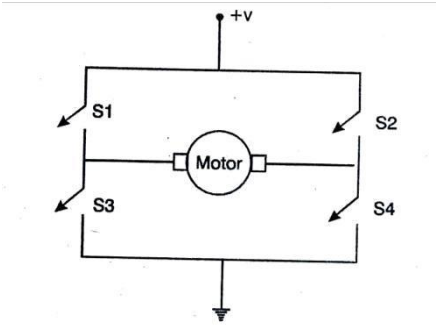
Controlling of DC motor is nothing but controlling the speed and the direction of a motor. When DC voltage is applied with proper current to a motor, it rotates in a particular direction but when the connection of voltage between two terminals is reversed, motor rotates in another direction.

DC motor direction

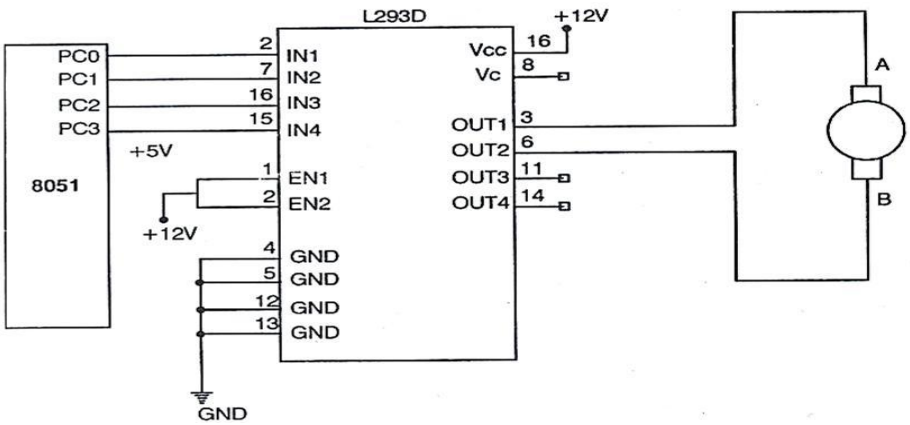


H-bridge

L293D is dual H-bridge motor driver, so with one IC two motors can be interfaced and controlled in both clockwise and counterclockwise direction. L293D has output current of 600mA and peak current of 1.2A per channel. Moreover for protection of circuit from back EMF output, diodes are included within IC. The output supply has a wide range from 4.5V to 36V.



DC motor interfacing with 8051



The decision of motor is considered as,

IN1	IN2	MOTOR1
0	1	rotates in one direction
1	0	rotates in one direction

Similarly another motor connected to out3 and out4 of L293D can be controlled through IN3 and IN4.

Label	Mnemonics	Comments
	POSITIVE EQU P2.0	;L293D A- Positive of motor
	NEGATIVE EQU P2.1	;L293D B-Negative of motor
	ENABLE EQU P2.2	;L293D E- Enable pin of IC
	ORG 00H	
MAIN	ACALL ROTATE_F	;Rotate motor forward
	ACALL DELAY	;Let the motor rotate
	ACALL BREAK	;Stop the motor
	ACALL DELAY	;Wait for sometime
	ACALL ROTATE_B	;Rotate motor backward
	ACALL DELAY	;Let the motor rotate
	ACALL BREAK	;Stop the motor
	ACALL DELAY	;Wait for sometime
	SJMP MAIN	;Do this in loop
ROTATE_F	SETB POSITIVE	;Make positive of motor 1
	CLR NEGATIVE	;Make negative of motor 0
	SETB ENABLE	;Enable to run the motor
	RET	;Return from routine

BREAK	CLR POSITIVE	;Make positive of motor 0
	CLR NEGATIVE	;Make negative of motor 0
	CLR ENABLE	;Disable the output
	RET	;Return from routine
DELAY	MOV R7, #20H	
BACK	MOV R6, #FFH	
BACK1	MOV R5, #FFH	
HERE	DJNZ R5, HERE	
	DJNZ R6, BACK1	
	DJNZ R7, BACK	
	HLT	

8. Explain in detail about stepper motor of 8051? [Nov/Dec-11[May/June-11][April/May-15] [Nov/Dec-15](April/May 2017)](April/May 2018)

STEPPER MOTOR

A stepper motor is a specially constructed DC motor. It has 4 windings arranged such that instead of running in the usual continuous fashion the motor rotates in precise steps, from one fixed position to another. The common step sizes ranges from 0.9 to 30 degrees.

The stepping action is achieved by proper combination of fields. From the pulsing sequence of winding when bit D0 is in 0 levels the coil A1 is energized, and so on. Changing the excitation from A1 & A2 to A2 & B1 causes the shaft to rotate by one step clockwise.

Each successive line of the table causes one step of rotation. Reversing the order in the sequence table causes the shaft to rotate counter clockwise. Thus the shaft can be given precise angular rotation.

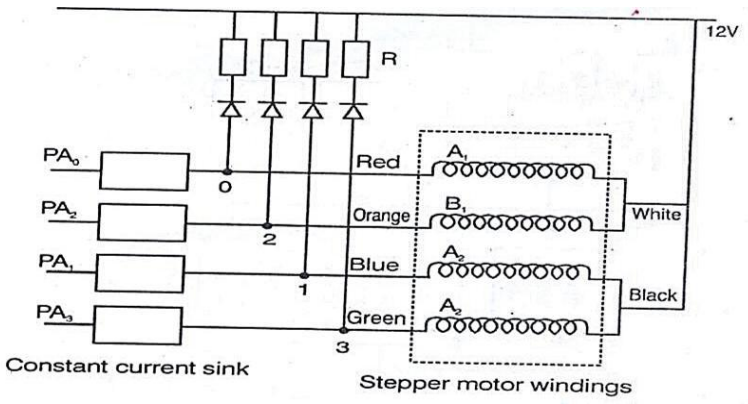
Uses

Stepper motors can be used to provide precise movements as in robot arms, floppy drives, print head movement and paper rotation in computer printers, machine tool control etc. Including these stepper motors are used in many other applications.

Coil pulse sequence for full step rotation

Clockwise	B2	B1	A2	A1	Anticlockwise
	D3	D2	D1	D0	
⇩	1	1	0	0	⇧
	1	0	0	1	
	0	0	1	1	
	0	1	1	0	
	1	1	0	0	

Stepper motor windings



Working

The coils require 1 amp current for rotation, at 12V. The 12V power supply should be capable of supplying more current. Transistor circuits are used as driver stages.

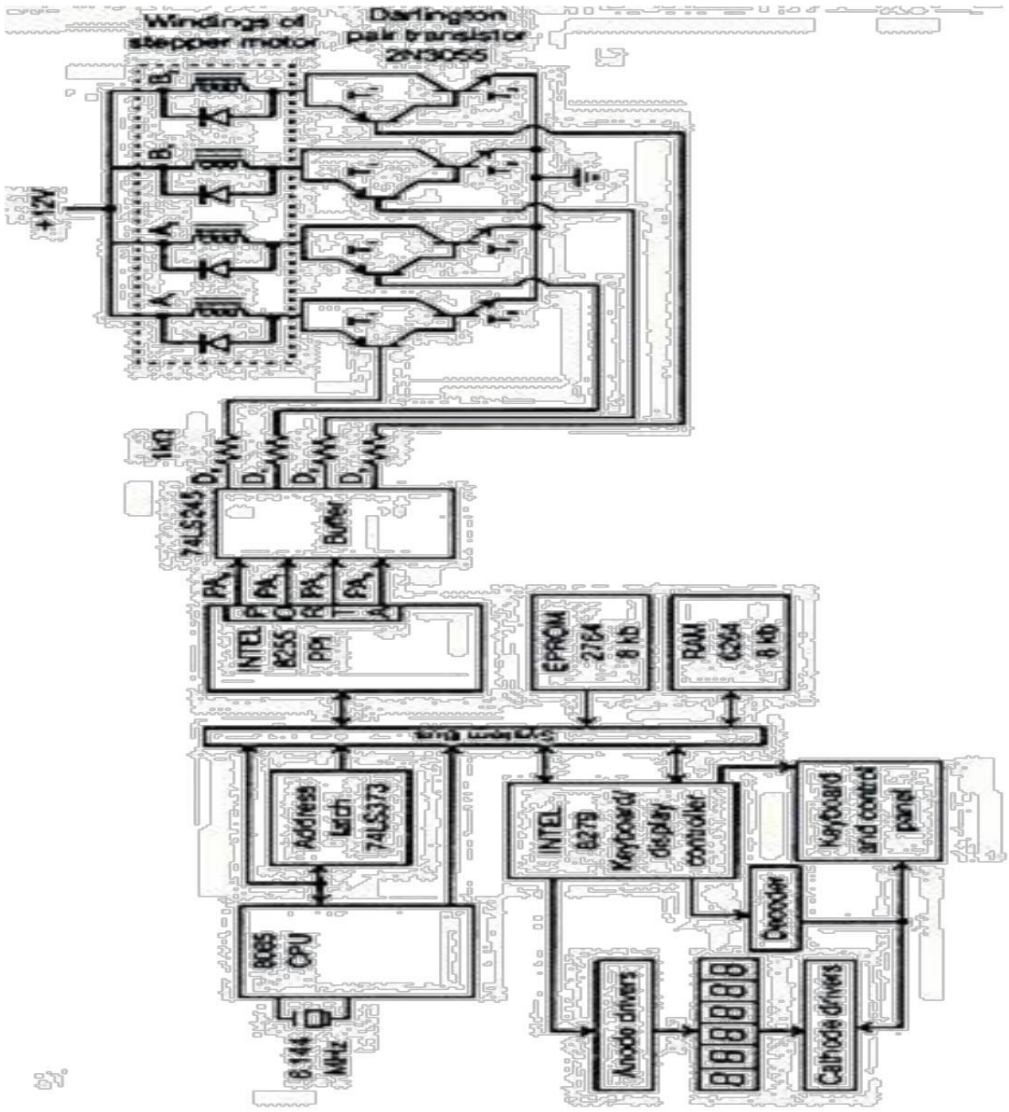
1. The PNP transistor T1 is ON when D3 bit is 0, it is OFF when D3 bit is 1.
2. T2 is on when T1 is ON, it is off when T1 is OFF.
3. When T2 conducts, there is a current through the coil B2. From this it is clear that when D3 bit is 0(zero), coil B2 is energized.

Similarly there are three pairs of transistor circuits to energize the coils A1, A2 and B1.

1. Zero level in D0 bit energizes coil A1.
2. Zero level in D1 bit energizes coil A2.
3. Zero level in D2 bit energizes coil B1.

The pulsing of these stages can be done with the program given. The starting word 1100 is essentially shifted left one bit in each step of sequence. The PPI chip 8255 on the kit can be used for the purpose of interfacing.

Microprocessor based stepper motor control system



ANTICLOCKWISE						CLOCKWISE					
STEP	A1	A2	B1	B2	DATA	STEP	A1		B1	B2	DATA
1	1	0	0	1	9h	1	1		1	0	Ah
2	0	1	0	1	5h	2	0		1	0	6h
3	0	1	1	0	6h	3	0		0	1	5h
4	1	0	1	0	Ah	4	1		0	1	9h

The program starts with the word 1100 and outputs it through one of the ports. It shifts the word left and output its through the same port for each step of rotation of shaft. For anti-clockwise rotation, the word is shifted right. The number of steps required and the direction of rotation (D0=0 for clockwise, D0=1 for anti-clockwise) are loaded in registers.

Program

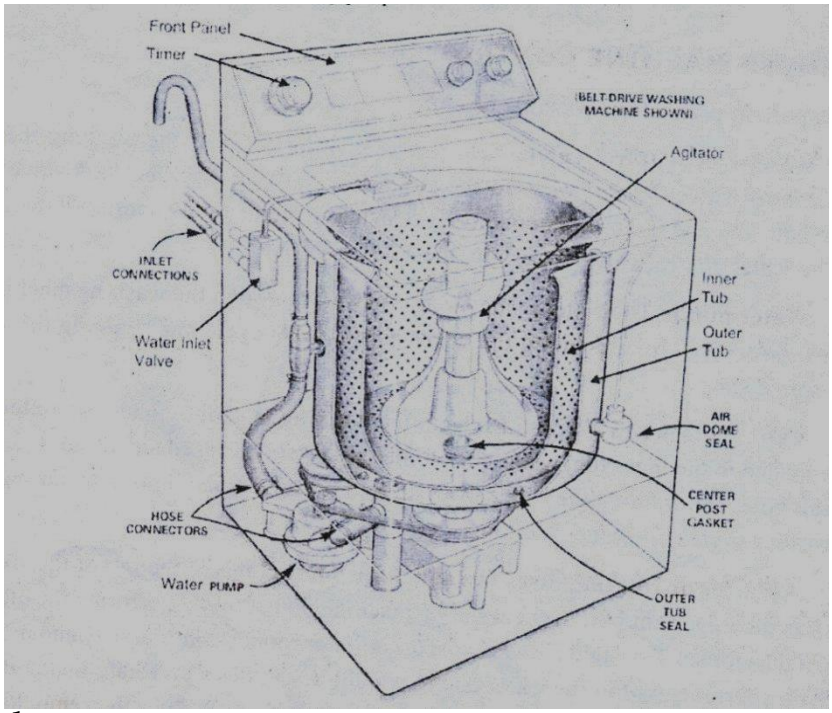
Stepper motor rotating in forward and reverse direction:

	ORG		4100H
START:	MOV		R4, # 33H
L2:	MOV		DPTR, # FORWARD
	LCALL		L1
	DJNZ		R4, # 33H
L3	MOV		DPTR, # REVERSE
	LCALL		L1
	DJNZ		R4, L3
	LCALL		DELAY
	SJMP		START
L1:	MOV		R0, #04H
LOOP:	MOVX		A,@DPTR
	PUSH		83H
	PUSH		82H
	MOV		DPTR, # 0FFC0H
	MOV		R2, # 04H
L7:	MOV		R1, # 05H
L6:	MOV		R3, # 0FFH
L4:	DJNZ		R3, L4
	DJNZ		R1, L6
	DJNZ		R2, L7
	MOVX		@DPTR, A
	POP		82H
	POP		83H
	INC		DPTR
	DJNZ		R0,LOOP
	RET		
DELAY:	MOV		R5, # 01H
L9:	MOV		R2, #05H
L8:	DJNZ		R2, L8
	DJNZ		R5, L9
	RET		
FORWARD:	DB	09H, 05H, 06H, 0AH	
REVERSE:	DB	0AH, 06H, 05H, 09H	

9. Explain in detail about washing machine control? [Nov/Dec-2011] [May/June-2011][Nov/Dec-14][April/May-15][Nov/Dec-15]. Explain the washing machine control using 8051 and write a program for the same.(Nov/Dec 2016).

Washing machine- architecture

Parts of washing machine



1. water inlet control

Near the water inlet point of washing there is water inlet control valve. When clothes are loaded in washing machine, this valve gets opened automatically and it closes automatically depending on the total quantity of water required. The water control valve is actually a solenoid valve.

2. water pump

The water pump circulates the water through the washing machine. It works in two directions, re-circulating the water during washing cycle and draining the water during the spin cycle.

3. Tub

There are two types of tubs in washing machine; inner and outer. The clothes are loaded in the inner tub, where the clothes are washed rinsed and dried. The inner tub has small holes for draining the water. The external covers the inner tub and supports it during various cycles of cloth washing.

4. Agitator or rotating disc

The agitator is located inside the tub of the washing machine. It is the important part of the washing machine that actually performs the cleaning operation of the clothes. During wash cycle the agitator rotates continuously and produces strong rotating currents within the water due to which the clothes also rotate inside the tub.

The rotation of clothes inside the water containing detergent enables the removal of the dirt particles from the fabrics of the clothes. Thus the agitator produces the most important function of rubbing the clothes with each other as well as with water.

5.Motor

The motor is coupled to the agitator or disc and produces its rotation motion. These are multi-speed motors, whose speed can be changed as per the requirement. In fully automatic washing machine the speed of the motor. i.e., agitator changes automatically as per the load on the washing machine.

6. Timer

The timer helps setting the wash time for the clothes manually. In the automatic mode the time is set automatically depending on number of clothes inside the washing machine.

7. Printed circuit board (PCB)

The PCB comprises of various electronic components and circuits which are programmed to perform in unique ways depending on the load conditions. Microcontroller or fuzzy logic systems based PCB will calculate the total weight of the clothes, and find out the quantity of water and detergent required and the total time required for washing the clothes. Then they will decide the time required for washing and rinsing.

8.Drain pipe

The drain pipe enables removing the dirty water from the washing that has been used for washing purpose.

Washing machine control using 8051

Washing machine usually employs a single phase motor. In semi-automatic washing machines, a purely mechanical switch controls the timing and direction of the motor. These switches are costly and wear out easily.

Basically, a single phase motor requires a master timer, which decides the time for which the motor should keep rotating and spin direction controller which stops the motor for three seconds after every ten seconds and then resumes rotation in opposite direction.

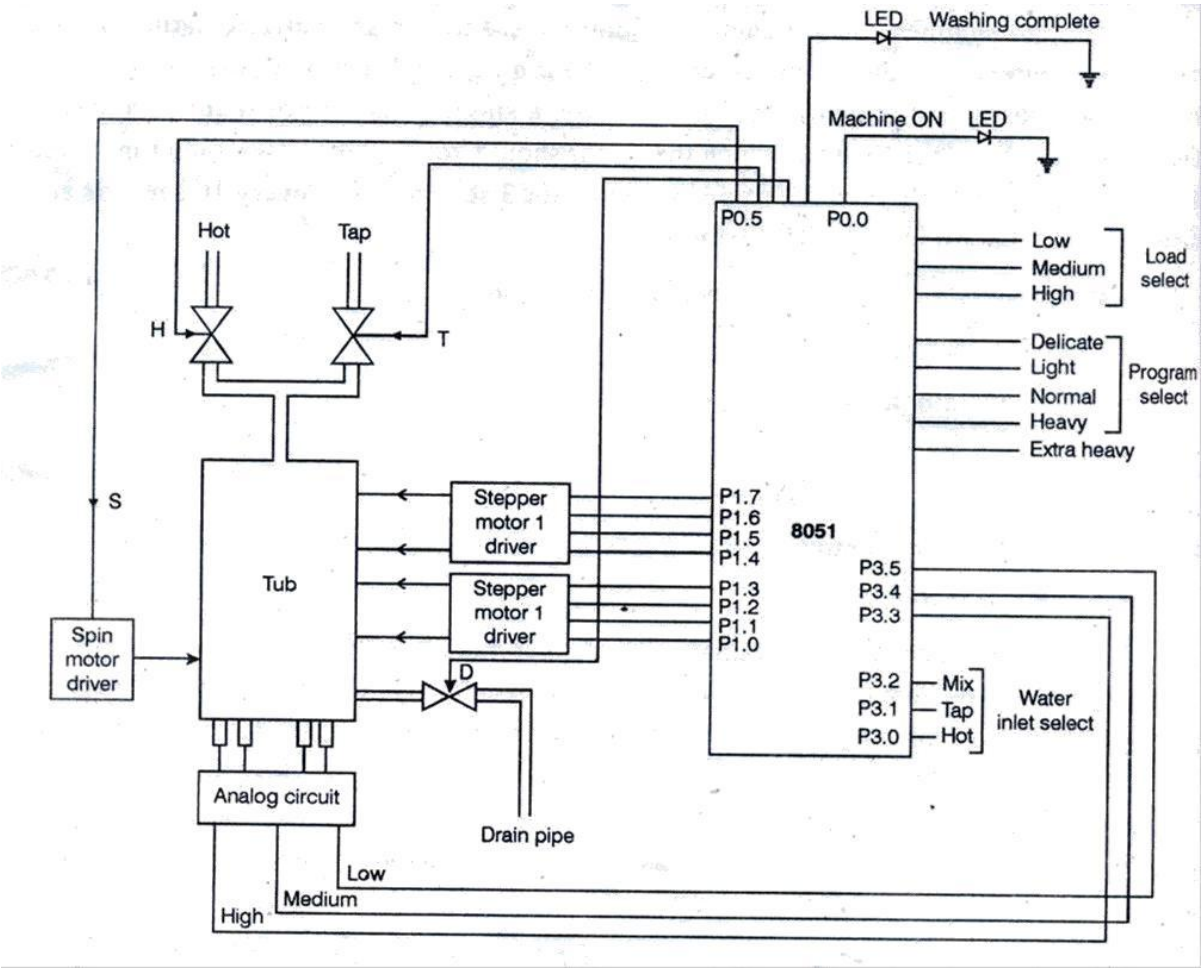
Microprocessor meets most of the complex requirements;

- Washing time,
- Temperature
- Freely set water levels
- Number of rinses and spins
- Liquid soap pump control
- Water recovery etc.

Most of the washing machines have two stepper motors for water quantity control and agitator control. The quantity of water to be filled is decided by the load selected by the user (high-medium-low or large-medium-small-extra small). As soon as the water quantity level reaches the desired level, the signal will be received and water flow is stopped.

During the agitate operation, the agitator moves one rotation in clockwise direction, followed by a rotation in anti-clockwise rotation. This cycle is continuously repeated for specified time.

Washing machine control using 8051



8051 port assignments are required for the following washing machine operations.

Port assignments

Port line	Input / Output	Assignments
P0.0	Output	Machine ON indication
P0.1	Output	Washing complete indications
P0.2	Output	D- Drain control signal
P0.3	Output	H- Hot water inlet control system
P0.4	Output	T- Tap water inlet control system
P0.5	Output	S- spin motor ON-OFF water inlet control system
P1.0 - P1.3	Output	Stepper motor 1 control signals
P1.4 – P1.7	Output	Stepper motor 2 control signals
P2.0	Input	Program select – Extra-heavy
P2.1	Input	Program select - Heavy
P2.2	Input	Program select - Normal
P2.3	Input	Program select - Light
P2.4	Input	Program select - Delicate
P2.5	Input	Load select – High
P2.6	Input	Load select – Medium
P2.7	Input	Load select – Low
P3.0	Input	Water inlet select – Hot
P3.1	Input	Water inlet select – Tap
P3.2	Input	Water inlet select – Mix
P3.3	Input	Water level – High
P3.4	Input	Water level – Medium

P3.5	Input	Water level – Low

WASHING MACHINE PROGRAM

(a) Delay Subroutine

```
PSW   :EQU DOH
MIN   :EQU R0
SEC   :EQU R0
      MOV PSW, # 00H
LOOP 3  : MOV MIN, # 3CH      ;       1 minute =60seconds.
LOOP 2  : MOV SEC; # 83H      ;131 TIMES REPETITION OF LOOP WILL
                  ; yield 1 second delay
LOOP 1 : MOV RH,#FFH
LOOP 0 :DEC RH
          JNZ LOOP0
          DEC SEC
          JNZ LOOP1
           DEC MIN
           JNZ LOOP2
          DEC R3
           JNZ LOOP3
```

Agitator control requires controlling of both stepper motors 1 and 2. Hence eight lines will be required for this purpose.

(b) Main program

; Initialise variables

```
MIN  : EQU R0
SEC  : EQU R1
PSW          : EQU DOH
P0           : EQU 80H
PI           : EQU A0H
P2           : EQU B0H
P3           : EQU 89H
TMOD         : EQU 89H
TCON         : EQU 88H
IE           : EQU A8H
TL1          : EQU 8BH
TH1          : EQU 8DH
```

```
;Put machine on indication
        SET B P0.0
; Fill the machine with water
        LCALL FILL 1
;Check if extra heavy setting?
;Extra heavy setting
                LCALL AGITATE(4)
                LCALL SOAK (6)
Heavy:          LCALL AGITATE(4)
                LCALL SOAK(6)
Normal:         LCALL AGITATE(14)
                LCALL DRAIN(4)
                LCALL SPIN(4)
                LCALL FILL(2)
                LCALL AGITATE(4)
                LCALL DRAIN(4)
                LCALL SPIN(10)
                SJMP DISP-END
;Check for heavy setting.
NEXT:           JNB P2.1, NEXT 1
                SJMP Heavy
; Check for normal setting
NEXT 1:JNB P2.2, NEXT 2 SJMP
                NORMAL
;Check for light setting
NEXT 2: JNB P2.3, NEXT 3
;Execute program for light
setting
                LCALL AGITATE(8)
                LCALL SOAK(4)
```

```
Delicate:LCALL AGITATE(2)
         LCALL DRAIN(4)
         LCALL SPIN(4)
         LCALL FILL(2)

         LCALL AGITATE(2)
         LCALL DRAIN(4)
         LCALL SPIN(4)
         SJMP DISP-END
;Check for delicate setting.
NEXT 3:  JNB P2.4, STP
         SJMP Delicate
; Display Washing
Complete DISP- End : SET
B, P0.1
STP: NOP : END
```

(c) FILL 1 Subroutine

```
Subroutine FILL 1
         P0   : EQU 80H
         PI   : EQU A0H
         P2   : EQU B0H
  ; Check water inlet setting – Hot, Tap or
              Mix JB P3.0, HOT-W
         JB P3.2, MIX-W
;Tap water setting, send control signal to start the tap water
;Hot water setting
;Mix water setting
      MIX-W: SETB   P0.4
         SETB P0.3
;Check load setting – Low, Medium or High
CHK-Level:  JB P2.7, Low-level
         JB P2.6, Med-level
;High level selected
```

```
High level    JB   P3.3, Level
              SJMP High-level
Med-level     JB    P3.4, Level
              SJMP Med-level
Low-level     JB     P3.5, Level
              SJMP Low-level
;Filling complete, stop water inlet tap
Level         : CLR    P0.3
              CLR    P0.4
              RET
```

(d) Agitate Subroutine

```
Subroutine AGITATE(1)
PSW           : EQU DOH
TMOD          : EQU 89H
TCON          : EQU 88H
TL1           : EQU 8BH
TH1           : EQU 8DH
IE            : EQU A8H
P1            :EQU 90H
OUT LIMIT : DSB 1
IN LIMIT      : DSB 1
LIMIT 1       : DSB 1
LIMIT 2        : DSB 1
; Set register bank = 0
        MOV          PSW, 00H
;Set timer 1 mode = 2(TMOD = 20H)
        MOV          A, # E2H
        MOV          TL1,A
        MOV          TH1,A
        MOV          TMOD, #20H
        MOV          OUT LIMIT, #08H
```

Mnemonic	Operation	Addressing modes	Execution time
ADD A, <byte>	A = A + <byte>	Dir, Ind, Reg, Imm	1
ADDC A, <byte>	A = A + <byte> + C	Dir, Ind, Reg, Imm	1
SUBB A, <byte>	A = A - <byte> - C	Dir, Ind, Reg, Imm	1
INC A	A = A + 1	Accumulator only	1
INC <byte>	<byte> = <byte> + 1	DPTR only	1
INC DPTR	DPTR = DPTR + 1	Dir, Ind, Reg	2
DEC A	A = A – 1	Accumulator only	1
DEC <byte>	<byte> = <byte> - 1	Dir, Ind, Reg	1
MUL AB	B:A=B x A	ACC and B only	4
DIV AB	A = Int [A/B] B = Mod [A/Bl	ACC and B only	4
DA A	Decimal Adjust	Accumulator only	1

10) Explain the different types of instructions set used in 8051 microcontroller.[Nov/Dec 2017]

1)ARITHMETIC INSTRUCTIONS

Mnemonic	Operation	Addressing modes	Execution time
ADD A, <byte>	A = A + <byte>	Dir, Ind, Reg, Imm	1
ADDC A, <byte>	A = A + <byte> + C	Dir, Ind, Reg, Imm	1
SUBB A, <byte>	A = A - <byte> - C	Dir, Ind, Reg, Imm	1
INC A	A = A + 1	Accumulator only	1
INC <byte>	<byte> = <byte> + 1	DPTR only	1
INC DPTR	DPTR = DPTR + 1	Dir, Ind, Reg	2
DEC A	A = A – 1	Accumulator only	1
DEC <byte>	<byte> = <byte> - 1	Dir, Ind, Reg	1
MUL AB	B:A=B x A	ACC and B only	4
DIV AB	A = Int [A/B] B = Mod [A/Bl	ACC and B only	4
DA A	Decimal Adjust	Accumulator only	1

2)LOGICAL INSTRCUTIONS

Mnemonic	Operation	Addressing modes	Execution time
ANL A, <byte>	A = A .AND. <byte>	Dir, Ind, Reg, Imm	1
ANL A, <byte>	<byte> = <byte> .AND. A	Dir	1
ANL <byte>, #data	<byte> = <byte> .AND. #data	Dir	2
ORL A, <byte>	A = A .OR. <byte>	Dir, Ind, Reg, Imm	1
ORL A, <byte>	<byte> = <byte> .OR. A	Dir	1
ORL <byte>, #data	<byte> = <byte> .OR. #data	Dir	2
XRL A, <byte>	A = A .XOR. <byte>	Dir, Ind, Reg, Imm	1
XRL A, <byte>	<byte> = <byte> .XOR. A	Dir	1
XRL <byte>, #data	<byte> = <byte> .XOR. #data	Dir	2
CRL A	A = 00H	Accumulator only	1
CPL A	A = .NOT. A	Accumulator only	1
RL A	Rotate ACC Left 1 bit	Accumulator only	1
RLC A	Rotate Left through Carry	Accumulator only	1
RR A	Rotate ACC Right 1 bit	Accumulator only	1
RRC A	Rotate Right through Carry	Accumulator only	1
SWAP A	Swap Nibbles in A	Accumulator only	1

3)DATA TRANSFER INSTRUCTION THAT ACCESS THE INTERNAL MEMORY

Mnemonic	Operation	Addressing modes	Execution time
MOV A, <src>	A = <src>	Dir, Ind, Reg, Imm	1
MOV <dest>, A	<dest> = A	Dir, Ind, Reg	1
MOV <dest>, <src>	<dest> = <src>	Dir, Ind, Reg, Imm	2
MOV DPTR, #data 16	DPTR = 16-bit immediate constant	Imm	2
PUSH <src>	INCSP: MOV "@"SP", <src>	Dir	2
POP <dest>	MOV <dest>, "@SP": DECSP	Dir	2
XCH A, <byte>	ACC and <byte> exchange data	Dir, Ind, Reg	1
XCHD A, @Ri	ACC and @Ri exchange low nibbles	Ind	1

4))DATA TRANSFER INSTRUCTION THAT ACCESS THE EXTERNAL MEMORY

Address width	Mnemonic	Operation	Execution time
8 bits	MOVX A, @Ri	Read external RAM @Ri	2
8 bits	MOVX @Ri, A	Write external RAM @Ri	2
16 bits	MOVX A, @DPTR	Read external RAM @DPTR	2
16 bits	MOVX @DPTR, A	Write external RAM @DPTR	2

5)LOOK UP TABLES

Mnemonic	Operation	Execution time
MOVC A, @A+DPTR	Read Program Memory at (A + DPTR)	2
MOVC A, @A+PC	Read Program Memory at (A + PC)	2

6)BOOLEAN INSTRUCTION

Mnemonic	Operation	Execution time
ANL C, bit	C = C .AND. bit	2
ANL C, /bit	C = C .AND. .NOT. bit	2
ORL C, bit	C = C .OR. bit	2
ORL C, /bit	C = C .OR. .NOT. bit	2
MOV C, bit	C = bit	1
MOV bit, C	bit = C	2
CRL C	C = 1	1
CRL bit	bit = 0	1
SETB C	C = 1	1
SETB bit	bit = 1	1
CPL C	C = .NOT. C	1
CPL bit	bit = .NOT. bit	1
JC rel	Jump if C = 1	2
JNC rel	Jump if C = 0	2
JB bit, rel	Jump if bit = 1	2
JNB bit, rel	Jump if bit = 0	2
JBC bit, rel	Jump if bit = 1; CLR bit	2

7)JUMP INSTRUCTIONS

Mnemonic	Operation	Execution time
ANL C, bit	C = C .AND. bit	2
ANL C, /bit	C = C .AND. .NOT. bit	2
ORL C, bit	C = C .OR. bit	2
ORL C, /bit	C = C .OR. .NOT. bit	2
MOV C, bit	C = bit	1
MOV bit, C	bit = C	2
CRL C	C = 1	1
CRL bit	bit = 0	1
SETB C	C = 1	1
SETB bit	bit = 1	1
CPL C	C = .NOT. C	1
CPL bit	bit = .NOT. bit	1
JC rel	Jump if C = 1	2
JNC rel	Jump if C = 0	2
JB bit, rel	Jump if bit = 1	2
JNB bit, rel	Jump if bit = 0	2
JBC bit, rel	Jump if bit = 1; CLR bit	2

ANNA UNIVERSITY QUESTIONS

PART-A

1. Write a program to perform multiplication of 2 nos using 8051? **[Nov/Dec-2013]**
2. Write a program to load accumulator DPH & DPL using 8051? **[April/May-2011]**
3. Explain the operating mode0 of 8051 serial ports? **[April/May-2011]**
4. What are the tasks involved in keyboard interfacing? **[May/June-2012]**
5. Give some example of input devices to microprocessor-based systems**? [April/May-2011]**
6. Write about CALL statement in 8051?**[Nov/Dec-14]**
7. Write about program status word**.[May/June 14][April/May-15]**
8. State the functions performed by JBC and CJNE instructions in 8051 microcontroller. **[May/June 14][Nov/Dec-14]**
9. Give some example of input devices to microprocessor-based systems? **[April/May-2011][April/May-15]**
10. What is meant py PSW**(Nov/Dec-15)**
11. List out the difference between MOV and MOVX instruction**(Nov/Dec-15)**
12. What is the use of PSW? **(Nov/Dec-16)**
13. Mention any four data transfer instructions of 8051 microcontroller. **(Nov/Dec-16)**
14. What is meant by bit oriented instructions **(April/May- 2017)**

PART-B

1. Explain in detail about the data transfer instructions? **[April/May-2011]**

2. Explain in detail about the manipulation/arithmetic & logical instructions of 8051 microcontroller? **[April/May-2011]**

3. Explain in detail about control transfer instructions? **[April/May-2011] 4**. Addition of two 16 bit using 8051 microcontroller: **[April/May-2011]**

5. An 8051 program for keyboard and display interface: **[April/May-2011][Nov/Dec-2011][April/May-15][Apr/May 2018]**

6. Explain in detail about the closed loop control of servo motor using 8051? **[April/May-2011][Nov/Dec-14]**

7. Explain in detail about stepper motor of 8051? **[Nov/Dec-2011] [May/June-2011] [April/May-15] (Nov/Dec-15)[Apr/May 2018]**

8. Explain In Detail About Washing Machine Control**? [Nov/Dec-2011] [May/June-2011][Nov/Dec-14][April/May-15] (Nov/Dec-15)**

Question Paper Code : 71777

B.E./B.Tech. DEGREE EXAMINATION, APRIL/MAY 2017.

Sixth/Fifth Semester

Electrical and Electronics Engineering

EE 6502 — MICROPROCESSORS AND MICROCONTROLLERS

(Common to Robotics and Automation Engineering, Electronics and Instrumentation Engineering, Instrumentation and Control Engineering, Manufacturing Engineering)

(Regulations 2013)

Time : Three hours Maximum : 100 marks

Answer ALL questions.

PART A — (10 × 2 = 20 marks)

1. Why data bus is bi-directional?
2. List out the machine cycles of 8085 microprocessor.
3. Write an 8085 program to swap lower and higher nibble of the contents of accumulator.
4. List different instruction formats.
5. Classify the addressing modes of 8051 microcontroller.
6. List any four Special Function registers.
7. What are the modes of 8254 timer?
8. What is meant by cascading in 8259?
9. Explain the function of DJNZ instruction.
10. What is meant by bit oriented instructions?

PART B — (5 × 16 = 80 marks)

11. (a) (i) Explain the interrupt structure of 8085 microprocessor. (8)

(ii) With pin diagram explain 8085 microprocessor. (8)

Or

(b) (i) Explain the registers of 8085 microprocessor? (8)

(ii) What is meant by memory interfacing? Explain with an example. (8)

12. (a) (i) Explain the addressing modes of 8085 microprocessor. (8)

(ii) Write an 8085 assembly language program to divide an 8 bit number by another 8 bit number? (8)

Or

(b) (i) Write an 8085 assembly language program to find the largest among 'N' number where the value of N should be stored in 4200 and the array of elements from 4201. Store the result in 4300? (8)

(ii) What is meant by subroutine? Explain how the stack is affected while calling a subroutine program. (8)

13. (a) Explain Timer modes of 8051 microcontroller. (16)

Or

(b) Explain the architecture of 8051 microcontroller with a block diagram. (16)

14. (a) Explain the functioning of 8255 programmable peripheral interface and its modes. (16)

Or

(b) Explain the working of 8237 as a DMA controller and its command registers and their functions. (16)

15. (a) Explain the stepper motor control using 8051 and write an assembly language program for running the stepper motor in clockwise direction. (16)

Or

(b) Explain the Closed loop control of a servo motor using 8051 with a neat diagram. (16)

———————

Reg. No. :

Question Paper Code : 80378

B.E./B.Tech. DEGREE EXAMINATION, NOVEMBER/DECEMBER 2016.

Fifth Semester

Electrical and Electronics Engineering

EE 6502 — MICROPROCESSOR AND MICROCONTROLLER

(Common to Electronics and Instrumentation Engineering/Instrumentation and Control Engineering and Robotics and Automation Engineering and Sixth Semester Manufacturing Engineering)

(Regulations 2013)

Time : Three hours

Maximum : 100 marks

Answer ALL questions.

PART A — (10 × 2 = 20 marks)

1. Write an 8085 assembly program to add two digit BCD numbers in memory locations 5000H and 5001H and store the result in memory location 5002H.

2. List out the machine cycles for executing the instruction MVI A, 34 H.

3. Classify the addressing modes of 8085 microprocessor.

4. What is the function of the CALL instruction?

5. Explain the interrupts of 8051 microcontroller.

6. What is the significance of PSEN and EA pin in 8051 microcontroller?

7. Draw the command word format of 8255 in I/O mode.

8. List some of the features of 8259 Programmable Interrupt controller.

9. What is the use of PSW?

10. Mention any four data transfer instructions of 8051 microcontroller.

PART B — (5 × 16 =

11. (a) (i) Draw the timing diagram for I/O read and Write Machine cycles. (8)

(ii) Draw the interfacing diagram to interface 8085 with 2KB RAM and 4KB EPROM. (8)

4

Or

(b) Explain the Architecture of 8085 microprocessor with a neat block diagram. (16)

12. (a) (i) Explain the logical instructions with examples. (8)

(ii) Write an 8085 Assembly program to convert a Hexadecimal Number to ASCII code. (8)

Or

(b) Write an 8085 Assembly language program to multiply two numbers in memory locations 4200 and 4201 and store the product in memory locations 4202 and 4203. (16)

13. (a) (i) Explain the interrupt structure of 8051 microcontroller. (8)

(ii) Explain the RAM structure of 8051 microcrontroller. (8)

Or

(b) Explain the I/O ports of 8051 microcrontroller in detail. (16)

14. (a) (i) Explain the working of 8254 timer with a neat block diagram and its command word format. (8)

(ii) Explain the working of 8259 with a neat block diagram. (8)

Or

(b) Explain the working of 8279 as a keyboard/display controller and explain its command registers and their functions. (16)

15. (a) Explain the washing machine control using 8051 and write a program for the same. (16)

Or

(b) Explain the interfacing of four digit 7 segment display to 8051 and its program. (16)

Reg. No. : | | | | | | | | | | | | |

Question Paper Code : 27219

B.E./B.Tech. DEGREE EXAMINATION, NOVEMBER/DECEMBER 2015.

Fifth Semester

Electrical and Electronics Engineering

EE 6502 — MICROPROCESSOR AND MICROCONTROLLER

(Common to Electronics and Instrumentation Engineering/ Instrumentation and Control Engineering and Robotics and Automation Engineering)
(Regulations 2013)

Time : Three hours Maximum : 100 marks

Answer ALL questions.

PART A — (10 × 2 = 20 marks)

1. What is the use of stack pointer?
2. Mention the use of ALE.
3. How is time delay generated using subroutines?
4. Explain the functioning of CMP instruction.
5. List the interrupts of 8051 microcontroller.
6. Write the function of TMOD register in 8051 microcontroller.
7. Write the control word value for 8255 PPI when PORT A and PORT B are inputs in simple I/O mode.
8. What are the working modes of 8254 timer?
9. What is meant by PSW?
10. List out the difference between MOV and MOVX instructions.

B.E./B.Tech. Degree Examination,

November/December 2014

Sixth Semester

Electrical and Electronics Engineering

EE 2354—MICROPROCESSORS AND MICROCONTROLLERS

(Regulation 2008/2010)

PART-A

1. State the function of keyboard interrupts.
2. List the 8085 flags.
3. List any two data manipulation instructions.
4. What is meant by lookup table?
5. What are the function of USART?
6. List out the operating modes in 8253 Timer/Counter.
7. Mention the registers used for serial communication in 8051 microcontroller.
8. What are the addressing modes followed in 8051 microcontroller?
9. What are the I/O instruction used in 8051?
10. State the principle of microcontroller based stepper motor control system.

PART-B

11.(a) Explain the architecture of 8086 microprocessor.

(or)

11.(b)(i) Explain the function of 8085 signals.

11.(b)(ii) Draw and explain the timing diagram of memory write operation.

12. (a) Write an assembly language program for:

(i) Adding a set of *n* numbers.

(ii) To generate fibonaci series using subroutines.

(or)

12.(b)	Explain the types of addressing modes in 8085 processor. with suitable examples.
13.(a)	Explain about 8279 keyboard display controller in detail.
	(or)
13.(b)	Explain A/D converter interfacing in detail.
14.(a)	Explain the functional block diagram of 8051 microcontroller.
	(or)
14.(b)	Explain Timing Diagram interrupt structure of 8051 in detail.
15.(a)	Explain the closed loop control of servo motor in detail.
	(or)
15.(b)	Explain about Washing machine control using microcontroller programming.

Reg. No. : | | | | | | | | | | | | |

Question Paper Code : 72292

B.E./B.Tech. DEGREE EXAMINATION, APRIL/MAY 2015.

Sixth Semester

Electrical and Electronics Engineering

EE 2354/10133 EC 506/10133 EE 503 — MICRO PROCESSORS AND MICROCONTROLLERS

(Regulation 2008/2010)

(Common to PTEE 2354 — Microprocessors and Microcontrollers for B.E. (Part – Time) Electrical and Electronics Engineering — Regulation 2009)

Time : Three hours Maximum : 100 marks

Answer ALL questions.

PART A — (10 × 2 = 20 marks)

1. What is the function of ALE in 8085 microprocessor?
2. What is the maximum number of byte of memory addressable by the 8086 microprocessor?
3. What is the function of Rotate instructions? Give an example.
4. How is time delay generated using subroutines?
5. What are the internal Registers available in 8259 PIC?
6. Distinguish between synchronous and asynchronous transmission.
7. Write down the instruction format for 8051 microcontroller.
8. What is the purpose of timing diagram in 8051 microcontroller?
9. How pulse is generated using 8051 Microcontroller?
10. What are the control signals from 8051 microcontroller required for Washing machine control?

PART B — (5 × 16 = 80 marks)

11. (a) (i) Explain with a neat block diagram, the hardware architecture of 8085 microprocessor. (10)

(ii) Describe the interrupt structure of 8085 Microprocessor from the order their priority. (6)

Or

(b) (i) Describe the data transfer concepts in 8086 microprocessor. (8)

(ii) Draw the timing diagram of memory READ and WRITE operations in 8086 Microprocessor. (8)

12. (a) (i) Describe the 8085 Assembly Language Program for the Loop structure with counting of 10 numbers. (10)

(ii) Describe the different addressing modes of 8085 microprocessor. (6)

Or

(b) (i) Write an assembly language program using 8085 instructions to find the biggest number in a block of data stored in the memory locations from 70H-7FH. (10)

(ii) Write short notes on Look up table and its usage. (6)

13. (a) (i) With a neat functional block diagram, explain the functions of 8255 PPI. (8)

(ii) With a neat functional block diagram, explain the functions of 8279 keyboard controller. (8)

Or

(b) (i) With a neat functional block diagram, explain the function of 8259 PIC. (8)

(ii) Explain with a neat sketch, the A/D converter interfacing with 8085 microprocessor. (8)

14. (a) (i) Explain with a neat functional block diagram, the 8051 Microcontroller hardware. (10)

(ii) Describe the interrupt structure of 8051 Microcontroller. (6)

Or

(b) (i) Explain various I/O ports and its functions of 8051 Microcontroller.(8)

(ii) Explain how the internal timers are used to generate time delay by using 8051 Microcontroller. (8)

Reg. No. : | | | | | | | | | | | | |

Question Paper Code : 51446

B.E./B.Tech. DEGREE EXAMINATION, MAY/JUNE 2014.

Sixth Semester

Electrical and Electronics Engineering

EE 2354/EC 2312/EE 64/10133 EC 506/10133 EE 503 — MICROPROCESSORS AND MICROCONTROLLER

(Common to Fifth Semester Electronics and Instrumentation Engineering and Instrumentation and Control Engineering)

(Regulation 2008/2010)

(Common to PTEE 2354/PTEC 2312 – Microprocessors and Microcontroller for B.E. (Part-Time) Fourth Semester – Electrical and Electronics Engineering and Electronics and Instrumentation Engineering – Regulation 2009)

Time : Three hours Maximum : 100 marks

Answer ALL questions.

PART A — (10 × 2 = 20 marks)

1. To obtain a 320 ns clock, what should be the input clock frequency? What is the frequency of clock signal at CLK OUT?
2. What is meant by level-triggered interrupt? Which of the interrupts in 8085 are level triggered?
3. Mention the similarity and difference between compare and subtract instructions.
4. State the purpose and importance of NOP instruction.
5. What are the salient features of INTEL 8259 programmable interrupt controller?
6. How data is transmitted in asynchronous serial communication?
7. Mention the purpose of $\overline{\text{PSEN}}$ and $\overline{\text{EA}}$ in 8051 microcontroller.
8. List the interrupt sources in 8051 microcontroller.
9. State the functions performed by JBC and CJNE instructions in 8051 microcontroller.
10. What is Program Status Word?

PART B — (5 × 16 = 80 marks)

11. (a) Explain how pipelined architecture is implemented in 8086.

Or

(b) Draw the signal configuration of 8085 and explain the purpose of each signals.

12. (a) (i) Describe the interrupt structure of 8085 microprocessor and compare the same with 8086 microprocessor. (10)

(ii) Write an 8085 Assembly Language Program to generate a time delay of 1ms. Show the calculations. (6)

Or

(b) Write a program to calculate and store in the results as mentioned. Five memory locations 2401H, 2402H, 2403H, 2404H and 2405H have data called X1, X2, X3, X4 and X5.

(2405H) = X1 + X2 + X3 + X4

(2403H) = X5 – X3 – X2 – X1.

13. (a) Draw the block diagram of 8255A programmable peripheral interface and explain each block.

Or

(b) Discuss the internal architecture of 8253 programmable interval timer.

14. (a) Explain the port operation in 8051 microcontroller.

Or

(b) Explain the different modes with which the timer/counter in 8051 microcontroller can be programmed.

15. (a) Explain the different operand addressing modes in 8051 microcontroller with examples.

Or

(b) Describe the control system design of washing machine.