MICROPROCESSORS

AND

MICROCONTROLLERS

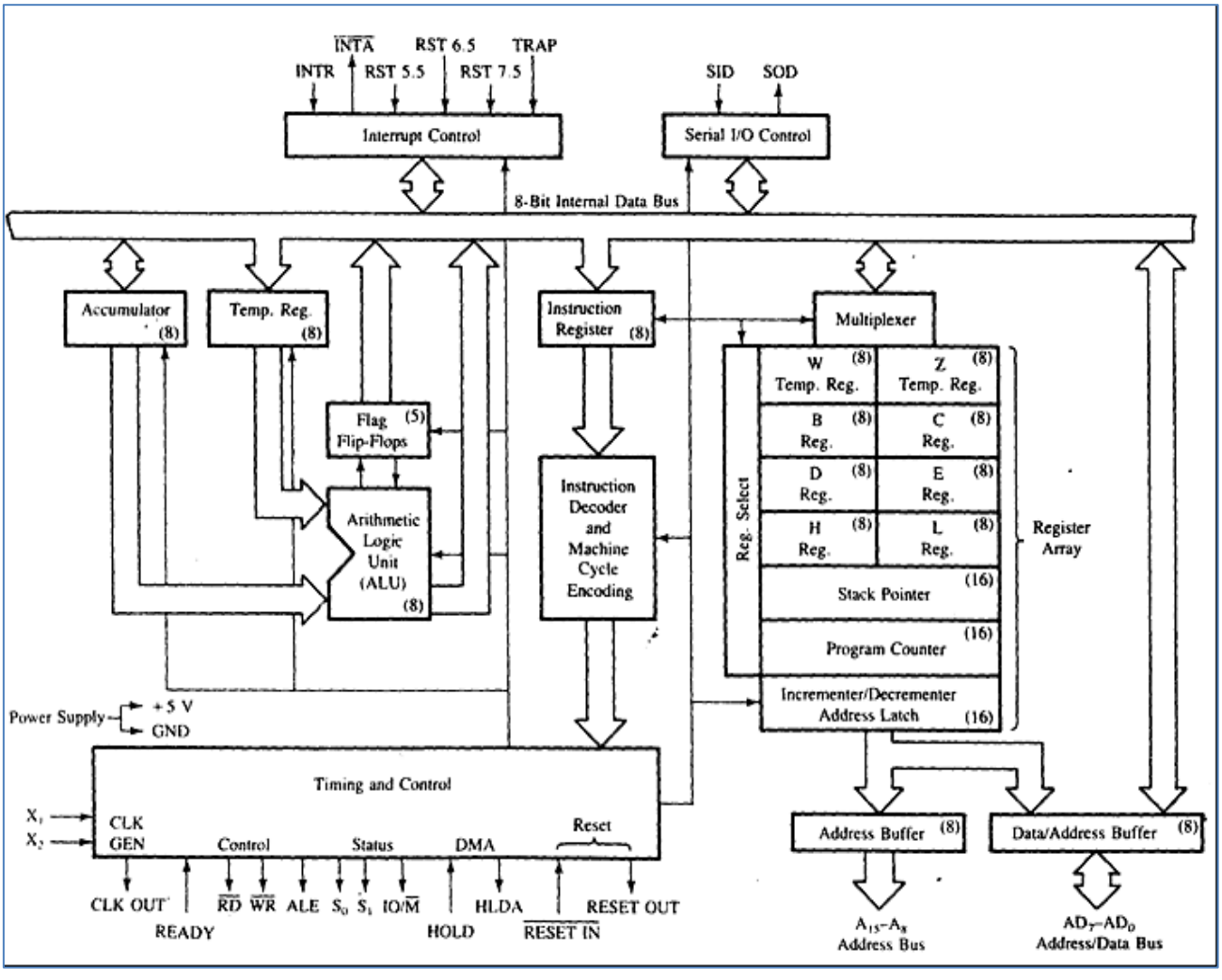
(EE6502)

FREQUENTLY ASKED QUESTIONS WITH ANSWERS

S.V.MAHESH KUMAR

UNIT 1
8085 PROCESSOR

1. Describe in detail about the architecture of 8085. (May/Jun 2016, Nov/Dec 2015, Dec 2016)



The 8085 is a 8-bit general-purpose microprocessor (μp)

- Capable of addressing **64 k** of **memory**
- Has **40 pins**
- Requires **+5 v** power supply
- Can operate with **3 MHz** clock

Arithmetic Logic Unit

- The ALU performs the actual numerical and logic operation such as 'add', 'subtract', 'AND', 'OR', etc.
- Uses data from memory and from Accumulator to perform arithmetic. Always **stores result of operation in Accumulator.**

Accumulator

- The accumulator is an **8-bit register** that is a part of ALU.
- This register is used to store 8-bit data and to perform arithmetic and logical operations.
- The **result of an operation is temporarily stored in the accumulator**. The accumulator is also **known as as register A**.
- It is a **Special Purpose Register**.

Flag Register

- The ALU includes **five flip-flops**, which are set or reset after an operation according to data conditions of the result in the accumulator and other registers.
- They are called **Zero (Z), Carry (CY), Sign (S), Parity (P), and Auxiliary Carry (AC)** flags.
- The microprocessor uses these flags to **test data conditions**. They are shown in the Figure below.

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

- The above Figure shows an 8-bit register, called the flag register, adjacent to the accumulator. However, it is not used as a register; five bit positions out of eight are used to store the outputs of the five flip-flops.
- The flags are stored in the 8-bit register so that the programmer can examine these flags (data conditions) by accessing the register through an instruction.
- Sign Flag is used for indicating the sign of the data in the accumulator. The sign flag is set if negative (1 –negative).The sign flag is reset if positive (0 –positive).
- For example, after an addition of two numbers, if the sum in the accumulator is larger

than eight bits, the flip-flop uses to indicate a carry -- called the Carry flag (CY) is set to one.

- When an arithmetic operation results in zero, the flip-flop called the Zero (Z) flag is set to one.

- Auxiliary Carry flag **i**s set if there is a carry out of bit 3.

- Parity flag is set if parity is even.

Registers (General-Purpose)

- The 8085/8080A has six general-purpose registers to store 8-bit data; these are identified as B,C,D,E,H, and L.
- They can be combined as register pairs - BC, DE, and HL - to perform some 16-bit operations.

Program Counter (PC)

- This 16-bit register deals with sequencing the execution of instructions. This register is a memory pointer. Memory locations have 16-bit addresses, and that is why this is a 16-bit register.

- The function of the program counter is to point to the memory address from which the **Next byte** is to be fetched.

- When a byte (machine code) is being fetched, the program counter is incremented by one to point to the next memory location.

Stack Pointer (SP)

- The stack pointer is also a 16-bit register used as a memory pointer. It points to a memory location in R/W memory, called the stack.

- **LIFO**

Instruction Register/Decoder

- Temporary store for the current instruction of a program. Latest instruction sent here from memory prior to execution. Decoder then takes instruction and ‘decodes’ or interprets the instruction. Decoded instruction then passed to next stage.

8085 System Bus

Address Bus:

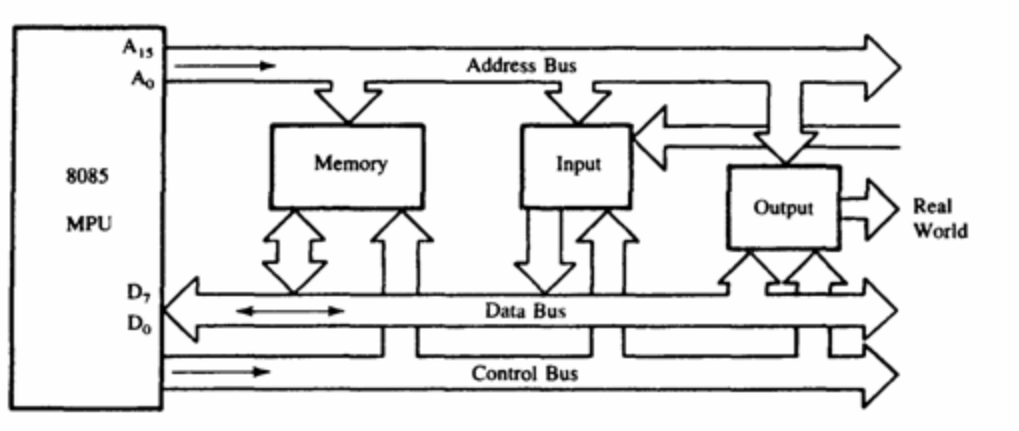
- The address bus has 8 signal lines A8 –A15, which are unidirectional
- (ie) numbers only sent from microprocessor to memory, not other way. The other 8 address bits are multiplexed (time shared) with the 8 data bits.

Data Bus:

- It carries 'data', in binary form, between μP and other external units, such as memory.
- It is Bi-directional. the bits AD0 –AD7are bi-directional and serve as A0 –A7and D0 –D7at the same time.
- During the execution of the instruction, these lines carry the address bits during the early part, and then during the late parts of the execution, they carry the 8 data bits.

Control Bus:

- Control Bus is various lines which have specific functions for coordinating and controlling uP operations



Interrupt Control

- The 8085A has5 interrupt inputs: INTR, RST5.5, RST6.5, RST 7.5, and TRAP.

- The interrupts are arranged in a fixed priority that determines which interrupt is to be
  recognized if more than one is pending as follows: TRAP highest priority, RST 7.5, RST 6.5, RST 5.5, INTR has lowest priority.

- The TRAP interrupt is useful for **critical errors** such as **power failure** or **bus error**.

  [**Note:** Refer **Section 1.5** for more details]

Serial I/O Control

- SID (Serial input data) line -The data on this line is loaded into accumulator bit 7 whenever a RIM instruction is executed.

- SOD (Serial output data) line. The output SOD is set or reset as specified by the SIM
  instruction.

Timing & Control

X1 and X2

- Crystal oscillator or R/C network connections are used to set the internal clock generator.

- The **input frequency is divided by 2** to give the internal **operating frequency**.

- Clock generator generates

  3.125 MHz internally

  6.25 MHz externally

CLK (Output)

- Clock Output for use as a system clock when a crystal or R/ C network is used as an input to the CPU.

- The period of CLK is twice the X1, X2 input period

ALE (Output)

- Address Latch Enable: It occurs during the first clock cycle of a machine state and enables the address to get latched into the on chip latch of peripherals.
- The falling edge of ALE is set to guarantee setup and hold times for the address information
- ALE can also be used to strobe the status information. ALE is never 3stated.

SO, S1 (Output)

Data Bus Status. Encoded status of the bus cycle:

S1 S0
0 0 HALT

0 1 WRITE
1 0 READ
1 1 FETCH

S1 can be used as an advanced R/W status.

READ ($\overline{RD}$) - (Output 3state)

READ; indicates the selected memory or 1/0 device is to be read and that the Data Bus is available for the data transfer.

WRITE ($\overline{WR}$)-(Output 3state)

WRITE; indicates the data on the Data Bus is to be written into the selected memory or 1/0 location. Data is set up at the trailing edge of WR. 3stated during Hold and Halt modes.

READY (Input)

If Ready is high during a read or write cycle, it indicates that the memory or peripheral is ready to send or receive data. If Ready is low, the CPU will wait for Ready to go high before completing the read or write cycle.

HOLD (Input)

HOLD; indicates that another Master is requesting the use of the Address and Data Buses. When the Hold is acknowledged, the Address, Data, RD, WR, and IO/M lines are 3stated.

HLDA (Output)

- HOLD ACKNOWLEDGE; indicates that the CPU has received the Hold request and

that it will relinquish the buses in the next clock cycle.

- HLDA goes low after the Hold request is removed. The CPU takes the buses one half clock cycle after HLDA goes low.

RESET IN ($\overline{RESET\ IN}$)

- Reset sets the Program Counter to zero and resets the Interrupt Enable and HLDA flip-flops. None of the other flags or registers (except the instruction register) are affected .The CPU is held in the reset condition as long as Reset is applied.

RESET OUT

- Indicates CPU is being reset. Can be used as a system RESET. The signal is synchronized to the processor clock.

IO/$\overline{M}$ *(IO/M)- (Output)*

- IO/M indicates whether the Read/Write is to memory or l/O Tristated during Hold and

Halt modes.

Vcc

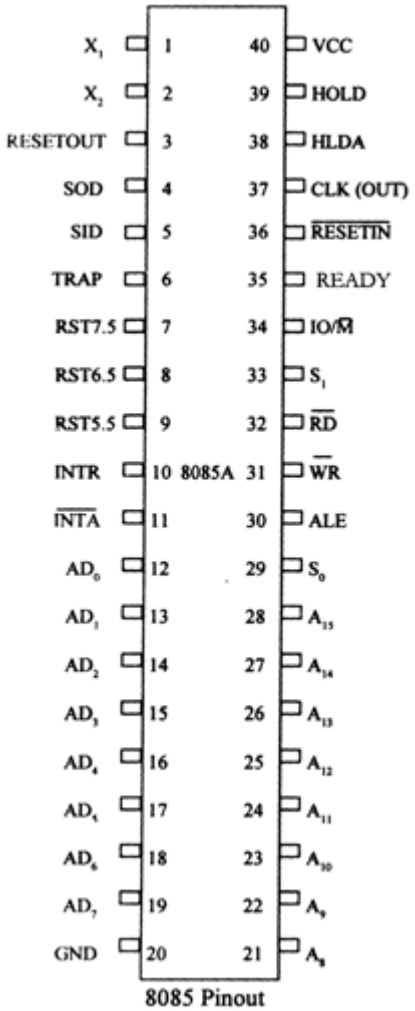
- +5 volt supply.

Vss

- Ground Reference.

---

2. Describe the functional pin diagram of 8085. (May/June 2017)



8085 Pinout

The description is given below

X1 and X2 (Pin 1 & Pin 2)

- Crystal oscillator or R/C network connections are used to set the internal clock generator.
- The **input frequency is divided by 2** to give the internal **operating frequency**.

- Clock generator generates

  3.125 MHz internally

  6.25 MHz externally

CLK (Output)- (Pin 37)

- Clock Output for use as a system clock when a crystal or R/ C network is used as an input to the CPU.

- The period of CLK is twice the X1, X2 input period

ALE (Pin 30)

- Address Latch Enable: It occurs during the first clock cycle of a machine state and enables the address to get latched into the on chip latch of peripherals.

SO, S1 (Output)- (Pin 29 & Pin 33)

These are status signals .The functions are given below

S1	S0	
0	0	HALT
0	1	WRITE
1	0	READ
1	1	FETCH

S1 can be used as an advanced R/W status.

READ ($\overline{RD}$) (Pin 32)

READ; indicates the selected memory or 1/0 device is to be read and that the Data Bus is available for the data transfer.

WRITE ($\overline{WR}$)(Pin 31)

WRITE; indicates the data on the Data Bus is to be written into the selected memory or 1/0 location. Data is set up at the trailing edge of WR. 3stated during Hold and Halt modes.

READY (Input)-(Pin 35)

If Ready is high during a read or write cycle, it indicates that the memory or peripheral is ready to send or receive data. If Ready is low, the CPU will wait for Ready to go high before completing the read or write cycle.

HOLD (Input)-(Pin 39)

HOLD; indicates that another Master is requesting the use of the Address and Data Buses. When the Hold is acknowledged, the Address, Data, RD, WR, and IO/M lines are 3stated.

HLDA (Output)- (Pin 38)

- HOLD ACKNOWLEDGE; indicates that the CPU has received the Hold request and

that it will relinquish the buses in the next clock cycle.

- HLDA goes low after the Hold request is removed. The CPU takes the buses one half clock cycle after HLDA goes low.

RESET IN ($\overline{\text{RESET IN}}$)- (Pin 36)

- Reset sets the Program Counter to zero and resets the Interrupt Enable and HLDA flip-flops. None of the other flags or registers (except the instruction register) are affected .The CPU is held in the reset condition as long as Reset is applied.

RESET OUT (Pin 3)

- Indicates CPU is being reset. Can be used as a system RESET. The signal is synchronized to the processor clock.

IO/$\overline{\text{M}}$ *(Output)- (Pin 34)*

- IO/M indicates whether the Read/Write is to memory or l/O Tristated during Hold and

Halt modes.

INTR (Input) - (Pin 10)

INTERRUPT REQUEST; is used as a general-purpose interrupt. It is sampled only during the next to the last clock cycle of the instruction. If it is active, the Program Counter (PC) will be inhibited from incrementing and an INTA will be issued. During this cycle a RESTART or CALL instruction can be inserted to jump to the interrupt service routine. The INTR is enabled and disabled by software. It is disabled by Reset and immediately after an interrupt is accepted.

$\overline{\text{INTA}}$ (Output) – (Pin 11)

- INTERRUPT ACKNOWLEDGE; is used instead of (and has the same timing as) RD during the Instruction cycle after an INTR is accepted. It can be used to activate the 8259 Interrupt chip or some other interrupt port.

RESTART INTERRUPTS

- These three inputs have the same timing as I NTR except they cause an internal RESTART to be automatically inserted.

RST 7.5 – (Pin 7)
RST 6.5 – (Pin 8)

RST 5.5 - (Pin 9)

TRAP (Input)- (Pin 6)

- Trap interrupt is a nonmaskable restart interrupt. It is recognized at the same time as INTR.it is unaffected by any mask or Interrupt Enable. It has the highest priority of any interrupt.

Serial I/O Control (Pin 4 & Pin 5)

- SID (Serial input data) line -The data on this line is loaded into accumulator bit 7 whenever a RIM instruction is executed.

- SOD (Serial output data) line. The output SOD is set or reset as specified by the SIM
  instruction.

Address bus (A8 –A15) - (Pin 21- Pin 28)

- The address bus has 8 signal lines A8 –A15, which are unidirectional (ie) numbers only sent from microprocessor to memory, not other way.

Address/ Data bus (AD0 –AD7)- (Pin 12- Pin 19)

- It is Bi-directional. The bits AD0 –AD7are bi-directional and serve as A0 –A7and D0 –D7at the same time.

Vcc (Pin 40)

- +5 volt supply.

Vss (Pin 20)

- Ground Reference.

---

3. Explain the 8085 interrupt system in detail. (May/June 2016) (May/June 2017)

An interrupt is considered to be an emergency signal. When the Microprocessor receives an interrupt signal, it **suspends the currently executing program** and **jumps to an interrupt Service Routine (ISR) to respond** to the incoming interrupt.

4.1 HARDWARE INTERRUPTS

The Hardware Interrupts of 8085 are given below:

TRAP

RST7.5

RST6.5,

RST5.5

INTR

- **TRAP: - TRAP has highest priority** .It is **non maskable edge and level triggered interrupt**. TRAP has the **highest priority** and **vectors interrupt**. Edge and level triggered means that the TRAP must go high and remain high until it is acknowledged. In case of sudden power failure, it executes a ISR and send the data from main memory to backup memory. As we know that TRAP cannot be masked but it can be delayed using HOLD signal. This interrupt transfers the microprocessor's control to location **0024H**.TRAP interrupts can only be masked by resetting the microprocessor. There is no other way to mask it.

- **RST7.5:-**It has the second highest priority. It is **maskable** and **edge triggered** interrupt. The **vector address** of this interrupt is **003CH**.Edge sensitive means input goes high and no need to maintain high state until it is recognized. It can also be reset or masked by resetting microprocessor. It can also be reset by DI instruction.

- **RST6.5 and RST5.5:-**These are **level triggered** and **maskable** interrupts. When RST6.5 pin is at logic 1, INTE flip-flop is set. RST 6.5 has third highest priority and RST 5.5 has fourth highest priority. It can be masked by giving **DI** and SIM instructions or by resetting microprocessor.

- **INTR:-**It is **level triggered** and **maskable** interrupt. It has the **lowest priority**. It can be disabled by resetting the microprocessor or by DI and SIM instruction.

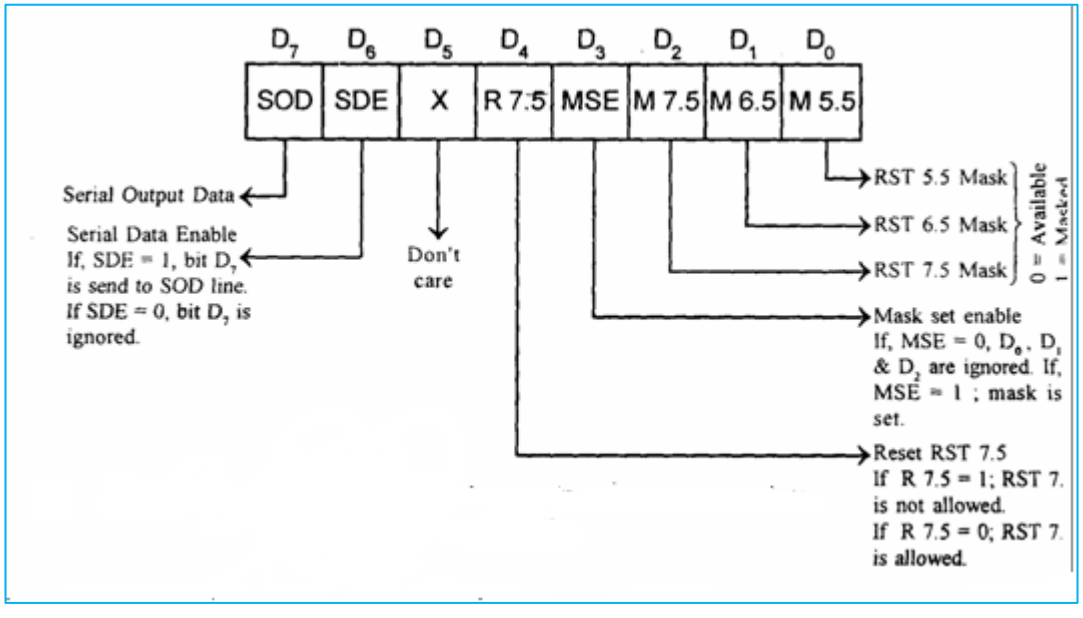
- **INTA** is not an interrupt. INTA is used by the Microprocessor for sending the acknowledgement**.**

- The vectored address of the interrupts are given in the below table. INTR is a non-vectored interrupt

INTERRUPT	VECTORED ADDRESS
RST 7.5	**003C H (7.5 x 0008 H)**
RST 6.5	**0034 H (6.5 x 0008 H)**
RST 5.5	**002C H (5.5 x 0008 H)**
TRAP	**0024 H (4.5 x 0008 H)**

SIM and RIM instructions 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.



Summary of Hardware interrupts

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	-

4.2 SOFTWARE INTERRUPTS

A software interrupts is a particular instructions that can be inserted into the desired location in the program. There are eight software interrupts in 8085 Microprocessor. From RST0 to RST7.

RST0
RST1
RST2
RST3
RST4
RST5
RST6
RST7

they allow the microprocessor to transfer program control from the main program to the subroutine program. After completing the subroutine program, the program control returns back to the main program.

We can calculate the vector address of these interrupts using the formula given below:

Vector Address = Interrupt Number * 8

So we can find simply vector address. For Example:

RST2: vector address=2*8 = 16
RST1: vector address=1*8 = 08
RST3: vector address=3*8 = 24

---

4. Explain the memory organization & data transfer concepts of 8085 (May /June 2017)

5.1. MEMORY

- Memory in a microprocessor system is where information (data and instructions) is kept. It can be classified into two main types:

  Main memory (RAM and ROM)

  Storage memory (Disks, CD ROMs, etc.)

- The simple view of RAM is that it is made up of registers that are made up of flip-flops (or memory elements). The number of flip-flops in a "memory register" determines the size of the memory word.

- ROM on the other hand uses diodes, instead of the flip-flops to permanently hold the information.

5.2. ACCESSING INFORMATION IN MEMORY

For the microprocessor to access (read or write) information in memory (RAM or ROM), it needs to do the following:

1. Select the right memory chip (using part of the address bus).
2. Identify the memory location (using the rest of the address bus).
3. Access the data (using the data bus).

5.3 MEMORY ORGANIZATION OF 8085

- The 8085 has 16 address lines. The memory capacity of 8085 is given as

  $$\mathbf{2^{16} = 64K}$$

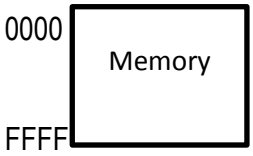
- Then it will need 1 memory chip with 64 k locations, or 2 chips with 32 K in each, or 4 with 16 K each or 16 of the 4 K chips, etc.

Memory Map

- The memory map is a picture representation of the address range and shows where the different memory chips are located within the address range.

- The 8085 has16 address lines. So, it can address a total of 64K memory locations. **($2^{16} = 64K$).**

- The address range of 8085 is 0000 H- FFFF H. The memory map of 8085 is shown below



Chip Select: Usually, each memory chip has a CS (Chip Select) input. The chip will only work if an active signal is applied on that input.

- To allow the use of multiple chips in the make up of memory, we need to use a number of the address lines for the purpose of "chip selection".

- These address lines are decoded to generate the 2^nnecessary CS inputs for the memory chips to be used.

- The 8085 has16 address lines. So, it can address a total of 64K memory locations.

- If we use memory chips with 1K locations each, then we will need 64 such chips.

- The 1K memory chip needs 10 address lines to uniquely identify the 1K locations.

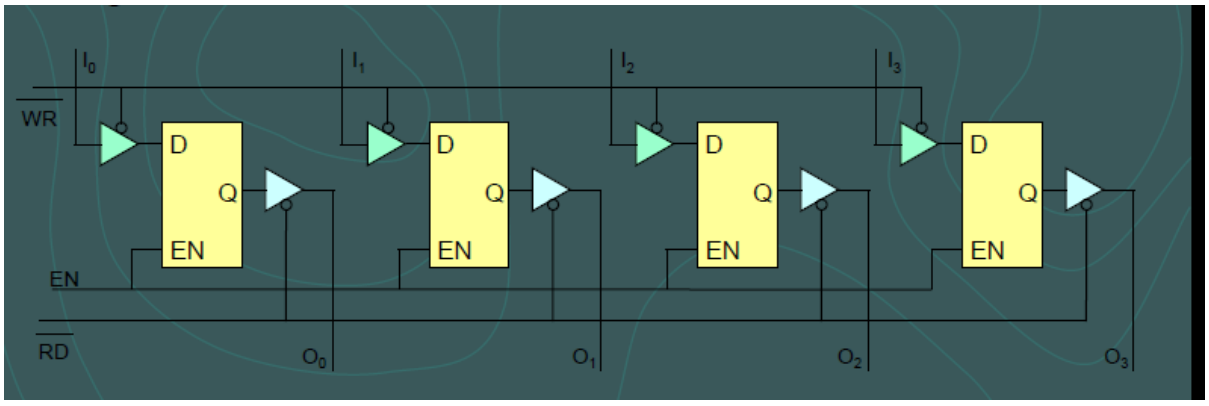
  ($\log_2 1024=10$).

- That leaves 6 address lines which is the exact number needed for selecting between the 64 different chips ($\log_2 64=6$).

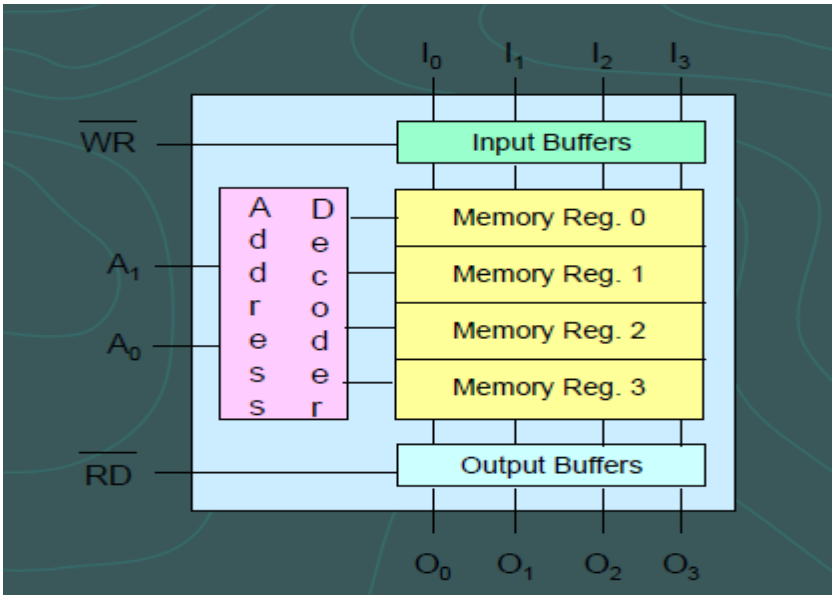
Example

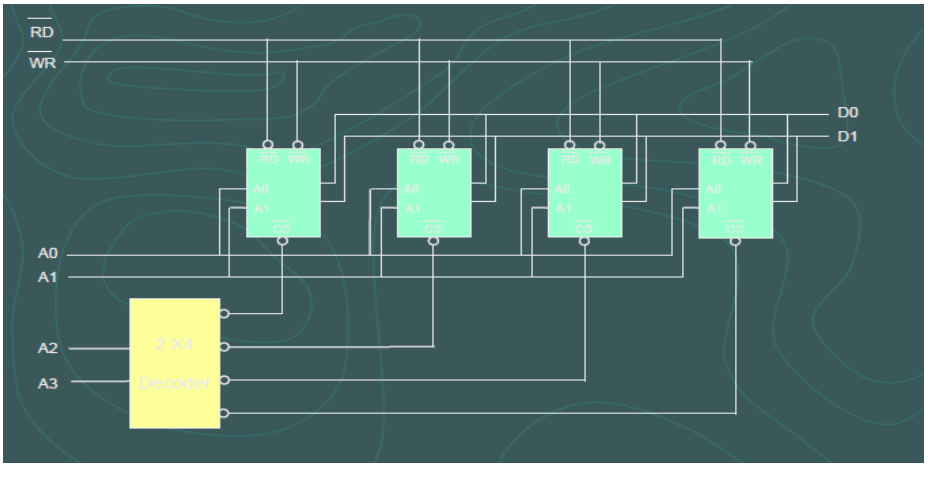
Build a memory system made up of 4 of the 4 X 4 memory chips.

If we take **four latches** and connect them together, we would have a 4-bit memory register. The 4 bit register is shown below



We will need to use 2 inputs and a decoder to identify which chip will be used at what time. This is shown in figure.





5. Draw the timing diagrams for the following operations

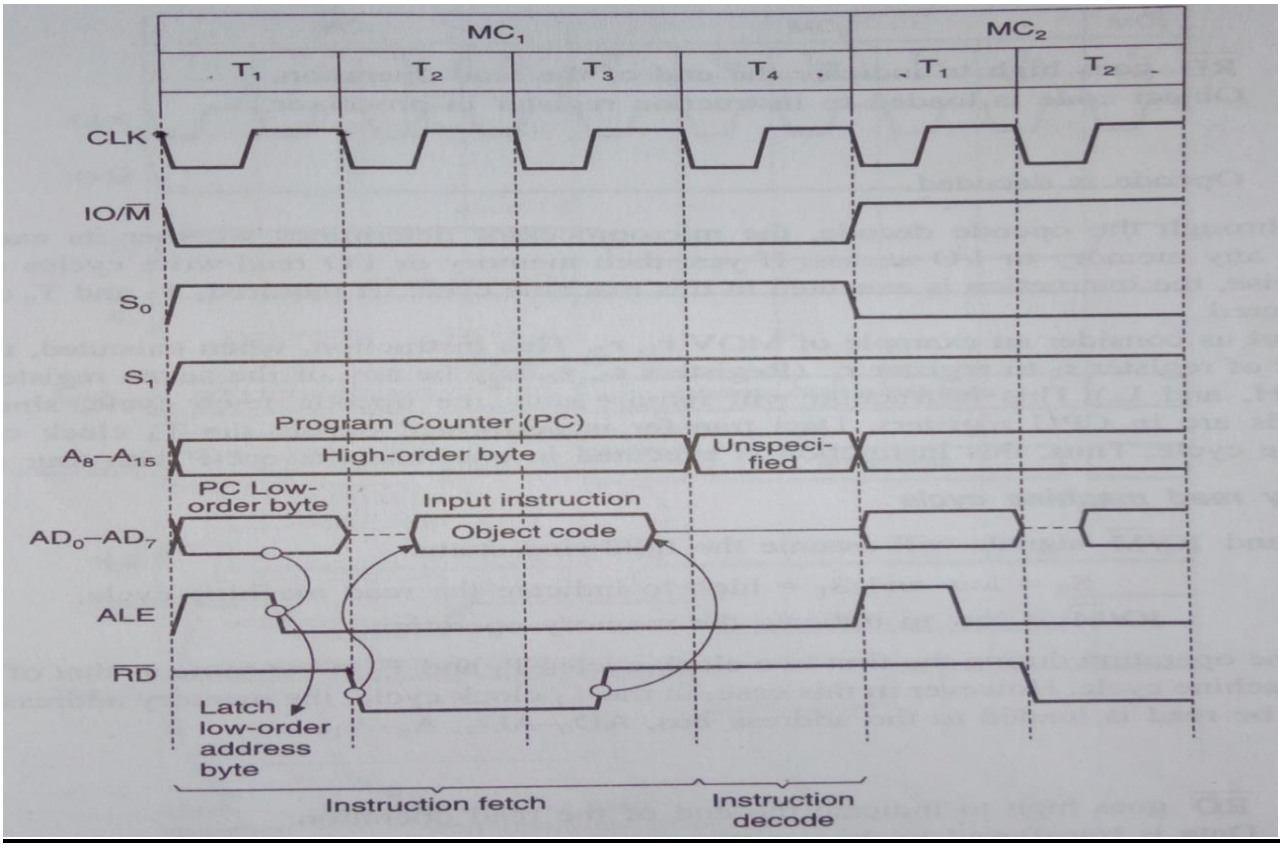
(i) Instruction Fetch (ii) Memory Read (iii) Memory Write (iv) MVI A, 32 H (v) STA 8000 (vi) IN

Representation of Various Control signals generated during Execution of an Instruction. Following Buses and Control Signals must be shown in a Timing Diagram:

Higher Order Address Bus.
Lower Address/Data bus
ALE
READ
WRITE
IO/$\overline{M}$

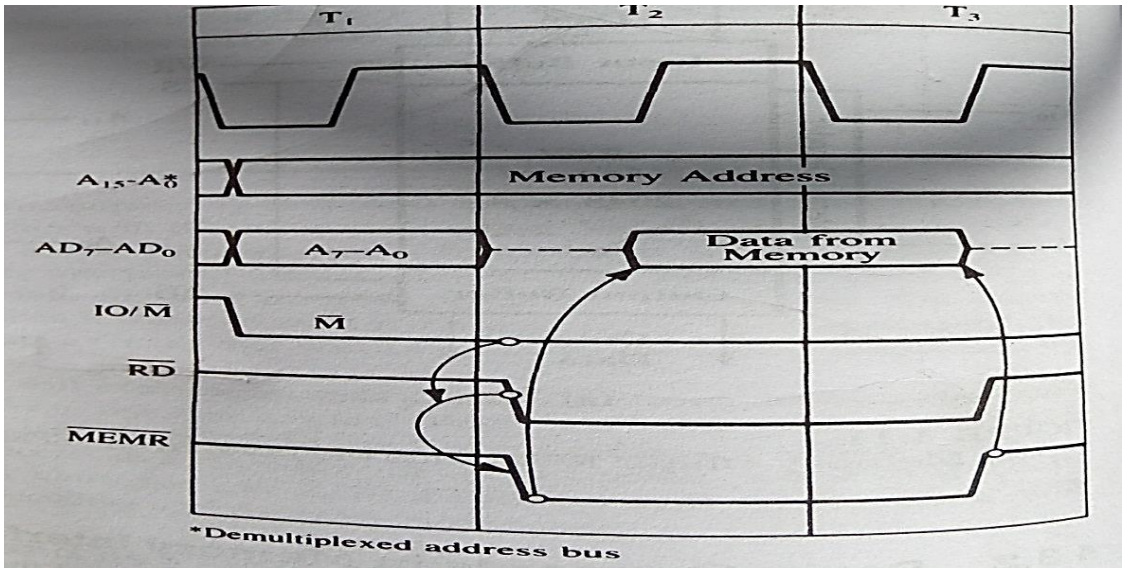
Timing Diagram 1

Instruction Fetch machine cycle



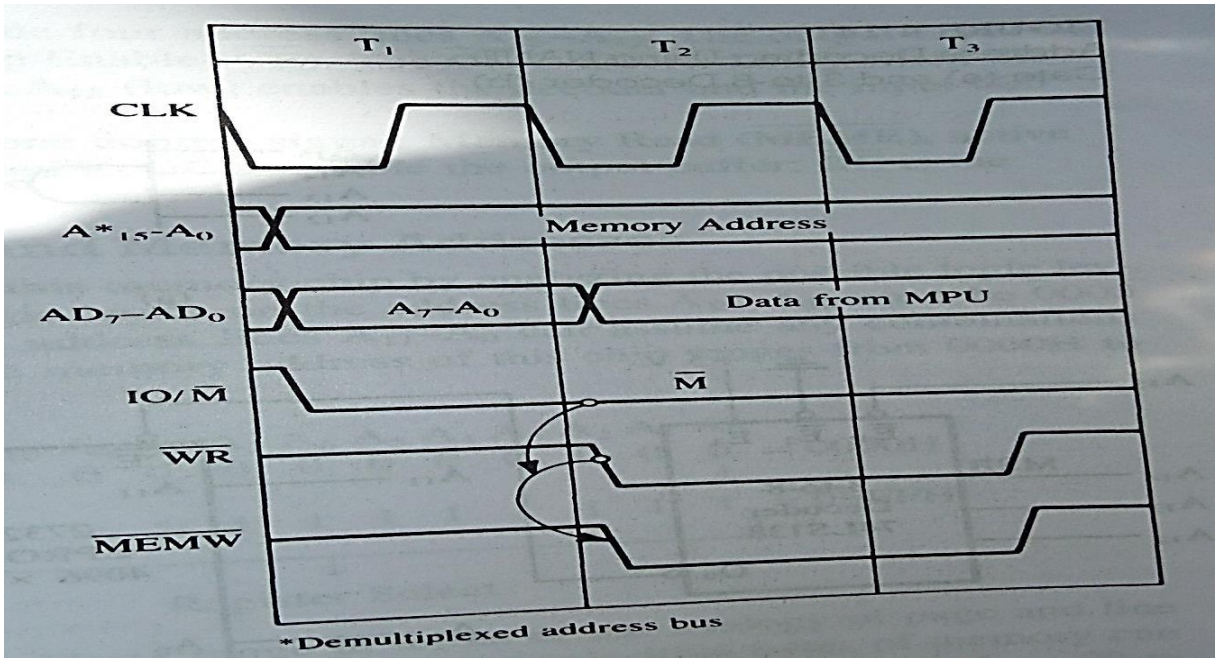
Timind Diagram 2

Memory Read



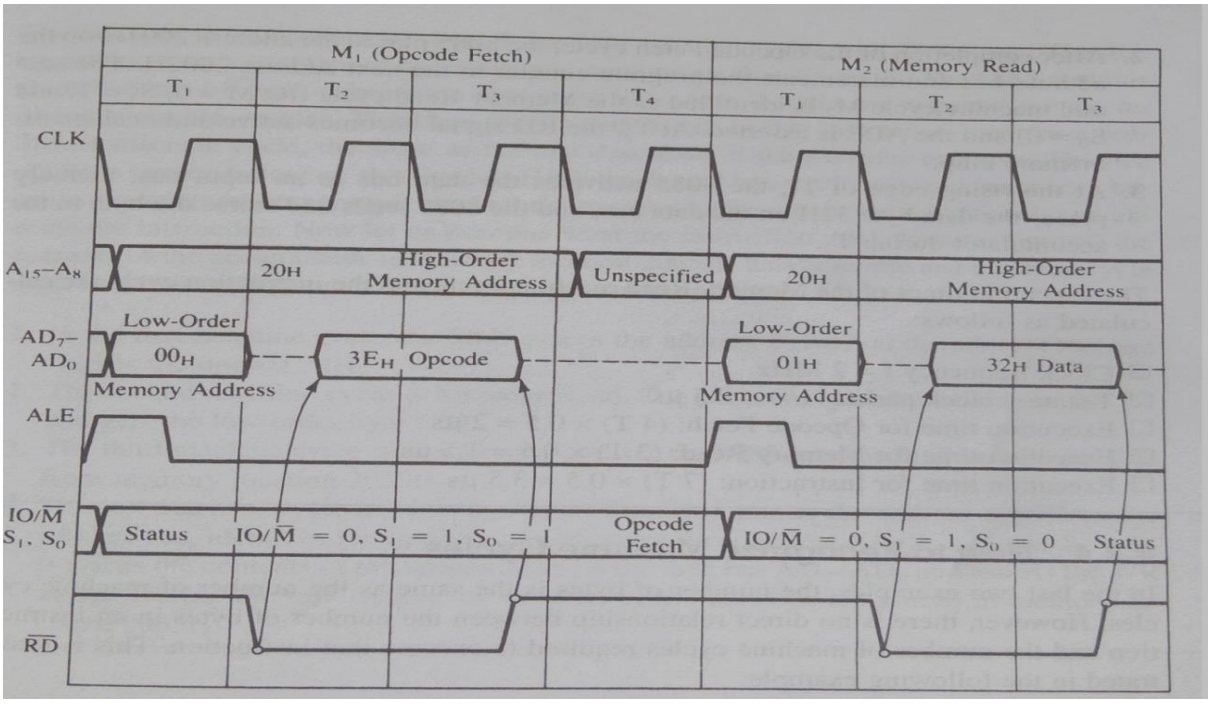
Timing Diagram 3

Memory Write



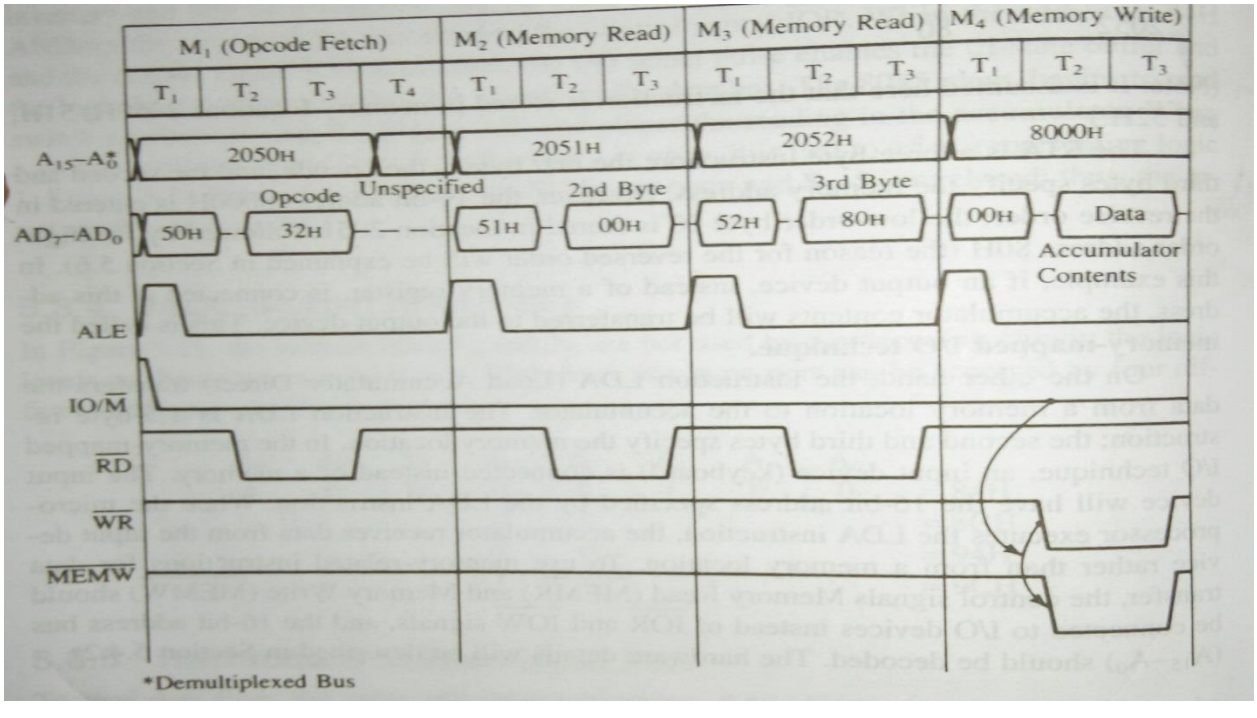
Timing Diagram 4

MVI A, 32 H



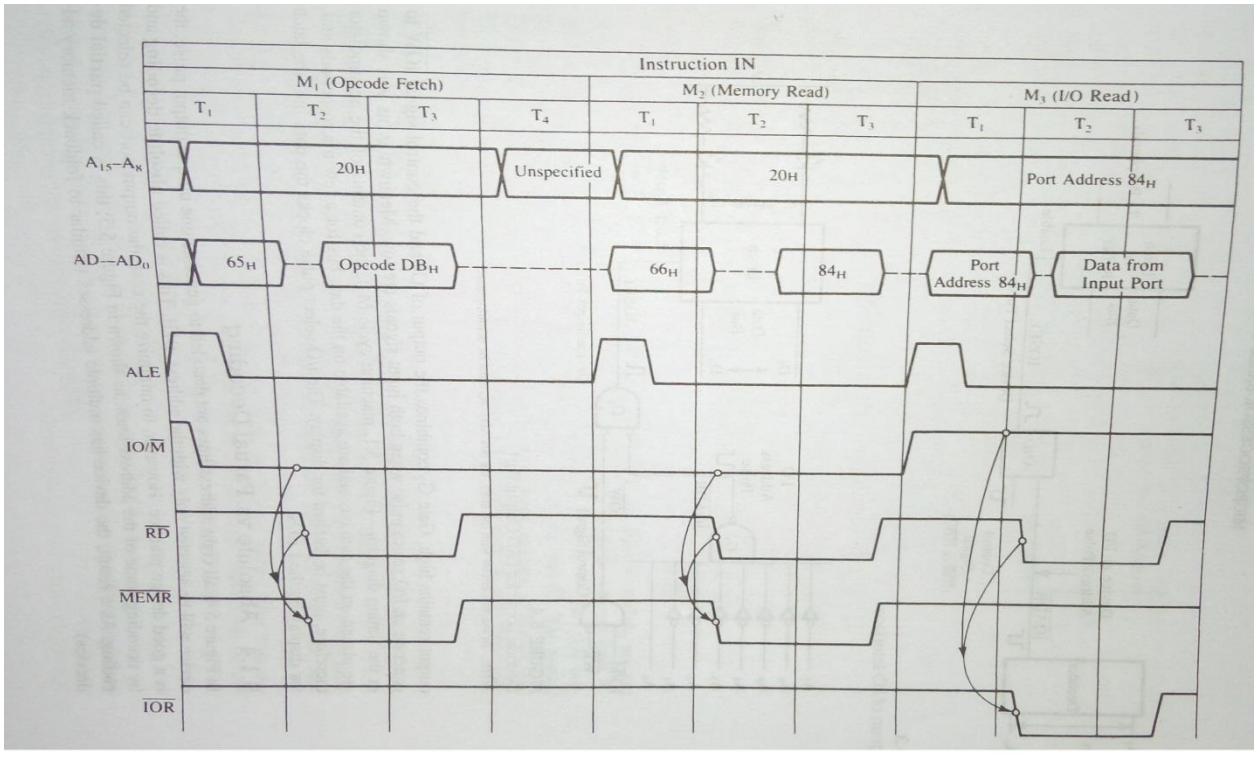
Timing Diagram 5

STA 8000

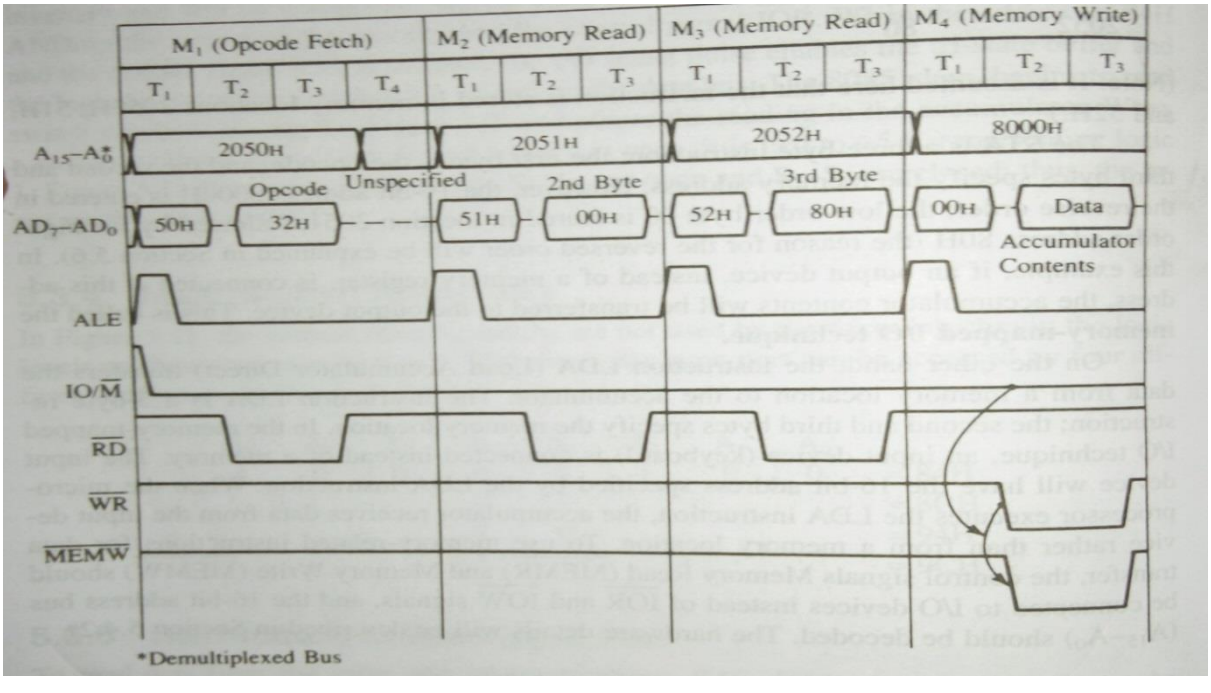


Timing Diagram 6

IN Instruction (Three Machine cycles : Opcode fetch +Memory Read + I/O Read)



7. Explain the timing diagram for STA 8000 (**May/Jun 2016)**



UNIT 2

PROGRAMMING OF 8085 PROCESSOR

PART B

1. Explain the various instruction formats of 8085.

- An **instruction** is a command to the microprocessor to perform a given task on a specified data. Each instruction has two parts: one is task to be performed, called the **operation code** (opcode), and the second is the data to be operated on, called the **operand.**

- The operand (or data) can be specified in various ways. It may include 8-bit (or 16-bit) data, an internal register, a memory location, or 8-bit (or 16-bit) address. In some instructions, the operand is implicit.

Instruction Word Size

The 8085 instruction set is classified into the following three groups according to word size:

1. 1-byte instructions
2. 2-byte instructions
3. 3-byte instructions

Note: In the 8085, "byte" and "word" are synonymous because it is an 8-bit microprocessor. However, instructions are commonly referred to in terms of bytes rather than words.

One-Byte Instructions

A 1-byte instruction includes the opcode and operand in the same byte. Operand(s) are internal register and are coded into the instruction.

Example: MOV A,B

Two-Byte Instructions

In a two-byte instruction, the first byte specifies the operation code and the second byte specifies the operand. Source operand is a data byte immediately following the opcode.

Example: MVI A, 32H

Three-Byte Instructions

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

(opcode + data byte + data byte)

Example: JMP 2085 H

---

2. **Explain the addressing modes of 8085 with example. (Nov/Dec 2015)(May/June 2017)**

The various formats for specifying operands are called the ADDRESSING MODES. For 8085, they are:

1. Immediate addressing mode.
2. Register addressing mode.
3. Direct addressing mode.
4. Register Indirect (or) indirect addressing mode.

Immediate addressing mode

Data is present in the instruction. Load the immediate data to the destination provided.

Format: MVI R,data

Example:

1. **MVI A, OF H**

   Load OF H to register A

After execution

(A)= OF H

2. **LXI D FFF0 H**

Load FFF0 H to D-E

After execution

(D) (E) = FFF0 H

Register addressing mode

Data is provided through the registers.

Format: MOV Rd, Rs

Example:

MOV B, D

INX H

Direct addressing mode

Used to accept data from outside devices to store in the accumulator or send the data stored in the accumulator to the outside device.

Example:

LDA 2000 H
STA 8000 H

Register Indirect (or) Indirect Addressing mode

This means that the Effective Address is calculated by the processor. And the contents of the address (and the one following) is used to form a second address. The second address is where the data is stored. Note that this requires several memory accesses; two accesses to retrieve the 16-bit address and a further access (or accesses) to retrieve the data which is to be loaded into the register.

Example: MOV B,M

Move the contents of memory location addressed by register pair H-L to B

---

3. Explain the Different types of instruction in 8085. (May/Jun 2016), Explain the logical operations of 8085. (Dec 2016)

An **instruction** is a binary pattern designed inside a microprocessor to perform a specific function. The entire group of instructions, called the **instruction set**,

determines what type of functions the microprocessor can perform. These instructions can be classified into following five functional categories:

1. Data transfer (copy) operations
2. Arithmetic operations
3. Logical operations
4. Branching operations
5. Machine-control operations

3.1. Data Transfer (Copy) Operations

This group of instructions copy data from a location called a source to another location called a destination, without modifying the contents of the source. In technical manuals, the term *data transfer* is used for this copying function. However, the term *transfer* is misleading; it creates the impression that the contents of the source are destroyed when, in fact, the contents are retained without any modification.

These operations transfer: Data between registers.

Data Byte to a register or memory location.
Data between a memory location and a register.
Data between an I\O Device and the accumulator.

The examples are listed below

MOV, MVI, LDA, and STA

3.2 .Arithmetic Operations

These instructions perform arithmetic operations such as addition, subtraction, increment, and decrement.

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 sum is stored in the accumulator. No two other 8-bit registers can be added directly (e.g., the contents of register B cannot be added directly to the contents of the register C). The instruction DAD is an exception; it adds 16-bit data directly in register pairs.

Example:

ADD, ADI

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 results stored in the accumulator. The subtraction is performed in 2's compliment, and the results if negative, are expressed in 2's complement. No two other registers can be subtracted directly.

Example:

SUB, SUI

Increment/Decrement - The 8-bit contents of a register or a memory location can be incremented or decrement by 1. Similarly, the 16-bit contents of a register pair (such as BC) can be incremented or decrement by 1. These increment and decrement operations differ from addition and subtraction in an important way; i.e., they can be performed in any one of the registers or in a memory location.

INX Rp (Incrementthe16-bitnumberintheregisterpair)

DCX Rp (Decrementthe16-bitnumberintheregisterpair)

3.3. Logical Operations

These instructions perform various logical operations with the contents of the accumulator.

AND, OR Exclusive-OR - Any 8-bit number, or the contents of a register, or of a memory location can be logically AND, OR, or Exclusive-OR with the contents of the accumulator. The results are stored in the accumulator.

Example:

ANA R/M	**AND Accumulator with Reg/Mem**
ANI #	**AND Accumulator With an 8-bit number**
ORA R/M	**OR Accumulator with Reg/Mem**
ORI #	**OR Accumulator With an 8-bit number**
XRA R/M	**XOR Accumulator with Reg/Mem**
XRI #	**XOR Accumulator With an 8-bit number**

Rotate- Each bit in the accumulator can be shifted either left or right to the next position.

Example:

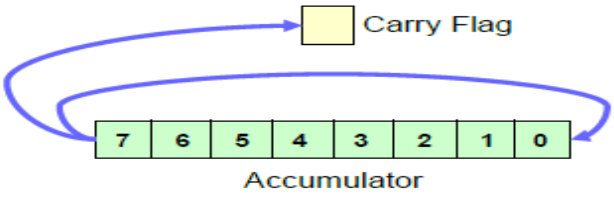
RLC **Rotate the accumulator left. Bit 7 goes to bit 0 and the carry flag.**

RAL **Rotate the accumulator left through the carry. Bit 7 goes to the carry and carry goes to bit 0.**

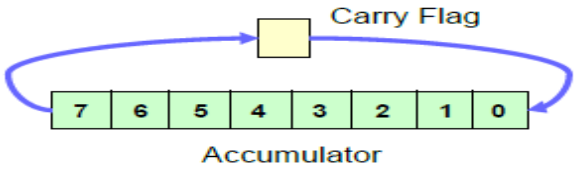
RRC **Rotate the accumulator right. Bit 0 goes to bit 7 AND the Carry flag .**

RAR **Rotate the accumulator right through the carry. Bit 0 goes to the carry and carry goes to bit 7.**

- RLC



- RAL



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

Example:

CMP R/M **Compare the contents of the register or memory location to the Contents of the accumulator.**

CPI number **Compare the 8-bit number to the contents of the accumulator.**

• The compare instruction sets the flags (Z, Cy, and S).

Complement - The contents of the accumulator can be complemented. All 0s are replaced by 1s and all 1s are replaced by 0s.

Example: CMA (ones complement of Accumulator (ACC) content)

3.4. Branching Operations

This group of instructions alters the sequence of program execution either conditionally or unconditionally.

Jump - Conditional jumps are an important aspect of the decision-making process in the programming. These instructions test for a certain conditions (e.g., Zero or Carry flag) and alter the program sequence when the condition is met. In addition, the instruction set includes an instruction called ***unconditional jump***.

CALL, Return, and Restart - These instructions change the sequence of a program either by calling a subroutine or returning from a subroutine. The conditional CALL and Return instructions also can test condition flags.

Unconditional Branch

JMP Address	Jump to the address specified (Go to).
CALL Address	Jump to the address specified but treat it as a subroutine.
RET	Return from a subroutine.

The addresses supplied to all branch operations must be 16-bits.

Conditional Branch

Go to new location if a specified condition is met.

JZ Address	(Jump on Zero) Go to address specified if the **Zero flag is set**.
JNZ Address	(Jump on NOT Zero) Go to address specified if the **Zero flag is Not set**.
JC Address	(Jump on Carry) Go to the address specified if the **Carry flag is Set**.
JNC Address	(Jump on No Carry)Go to the address specified if the **Carry flag is not set**.
JP Address	(Jump on Plus)Go to the address specified if the **Sign flag is not set**
JM Address	(Jump on Minus)Go to the address specified if the **Sign flag is set**.

3.5. Machine Control Operation

These instructions control machine functions such as Halt, Interrupt, or do nothing.

HLT
NOP

---

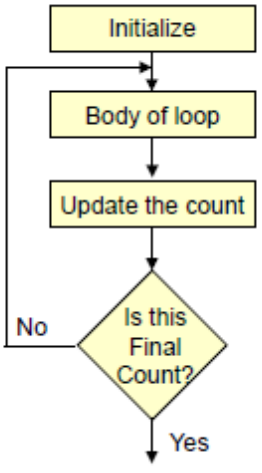
4. Explain the concept of loop indexing & delay loops with examples.

- A loop counter is set up by loading a register with a certain value ,then using the DCR (to decrement) and INR (to increment) the contents of the register are updated.

➢ A loop is set up with a conditional jump instruction that loops back or not depending on whether the count has reached the termination count.

Sample ALP and Flowchart

```
        MVI C, 15H
LOOP    DCR C
        JNZ LOOP
```



DELAY LOOP

We can use a loop to produce a certain amount of time delay in a program. The example ALP is given below

```
        MVI C, FFH
LOOP    DCR C
        JNZ LOOP
```

The T-states for the above ALP is

	MVI C, FFH	7 T-States
LOOP	DCR C	4 T-States
	JNZ LOOP	10 T-States

➢ The **first instruction** initializes the loop counter and is executed only once requiring only **7 T-States**.(Refer the Timing Diagram of MVI A, 32 H)

➢ The following **two instructions** form a loop that requires **14 T-States** to execute and is repeated **255**(Decimal value of FF H) times until C becomes 0.

➢ We need to keep in mind though that in the last iteration of the loop, the JNZ instruction will fail and require only 7 T-States rather than the 10.

➢ Therefore, we must **deduct 3 T-States** from the total delay to get an accurate delay calculation.

➢ To calculate the delay, we use the following formula:

$$T_{delay} = T_O + T_L$$

- T_{delay} = total delay
- T_O = delay outside the loop
- T_L = delay of the loop

Here, TO= 7 T-States

TL= (14 X 255) -3 = 3567 T-States14 T-States for the 2 instructions repeated 255 times (**$FF_H = 255_{10}$**) reduced by the 3 T-States for the final JNZ.

NESTED DELAY LOOP

➢ A nested loop structure can be used to increase the total delay produced. Here, more than one loop is used

Sample Program –Nested delay loop

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

5. What is meant by subroutine? Explain how the stack is affected while calling subroutine program. (May/ June 2017)

➢ A subroutine is a group of instructions

- That is used repeatedly in different places of the program.
- Rather than repeat the same instructions several times.
- It can be grouped into a subroutine and call from the different locations.

➢ Instructions for dealing with subroutines.

- The **CALL** instruction is used to redirect program execution to the subroutine.
- The **RET** instruction is used to return the execution to the calling routine.

Set the SP correctly before using CALL

FOR CALL INSTRUCTION

Example

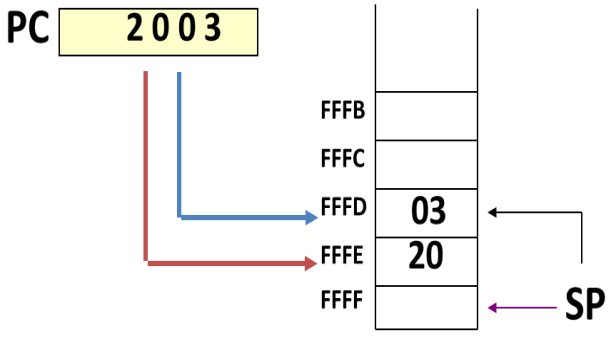
Instruction: CALL 5000 H

Assume that the CALL instruction is in the address 2000

2000 CALL 5000
2003

The Process can explained as

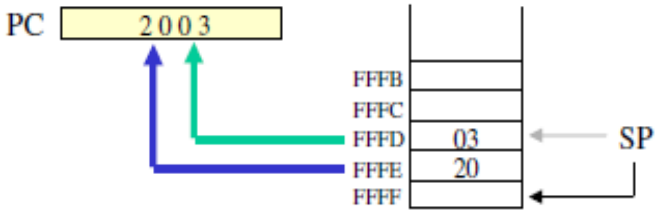
- Push the PC value onto the stack
- Load PC with 16-bit address supplied CALL instruction



FOR RET INSTRUCTION

The Process Is

- Load PC with stack top
- POP PC



Conditional CALL and RETURN Instructions

The 8085 supports conditional CALL and conditional RETURN 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

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.

A PROPER SUBROUTINE

- According to Software Engineering practices, a proper subroutine:

– Is only entered with a CALL and exited with an RTE
– Has a single entry point

- Do not use a CALL statement to jump into different points of the same subroutine.

– Has a single exit point

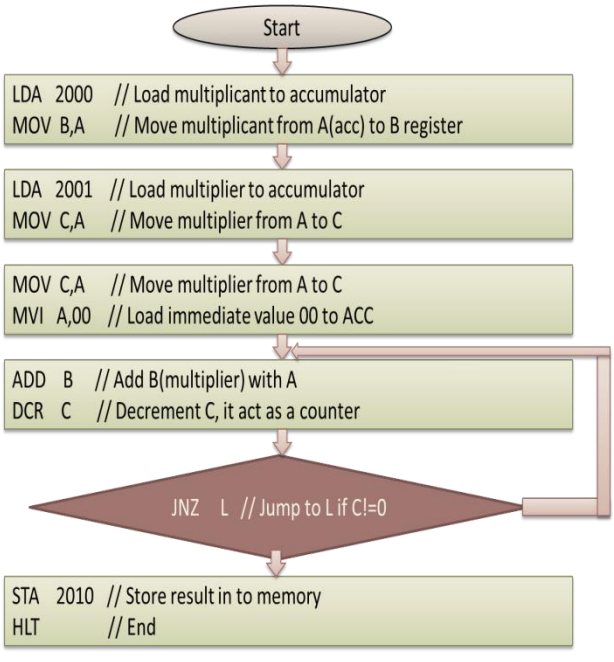
- There should be one return statement from any subroutine.

---

6. 8085 programs

(i) Write a ALP to multiply two numbers (Dec 2016)

Flowchart to multiply two number



PROGRAM

```
LDA   2000          // Load multiplicant to accumulator
      MOV B, A      // Move multiplicant from A (acc) to B register
      LDA   2001    // Load multiplier to accumulator
      MOV C, A      // Move multiplier from A to C
      MVI   A, 00   // Load immediate value 00 to a
Loop: ADD   B       // Add B(multiplier) with A
      DCR   C       // Decrement C, it act as a counter
      JNZ   Loop    // Jump to L if C reaches 0
```

STA 2010 // Store result in to memory
HLT // End

(ii) . Write a 8085 program to convert a Hexadecimal number to ASCII code (Dec 2016).

```
LDA 5000      //Get Hexa Data
 MOV B,A
 ANI 0F       //  Mask Upper Nibble
 CALL SUB1    //Get ASCII code for upper nibble
 STA 5001
 MOV A,B
 ANI F0       // Mask Lower Nibble
 RLC
 RLC
 RLC
 RLC
 CALL SUB1    // Get ASCII code for lower nibble
 STA 5002
 HLT          // Halt the program.

SUB1:  CPI 0A
 JC SKIP
 ADI 07
SKIP:  ADI 30
 RET          // Return Subroutine
```

RESULT

Input:

Data 1: HEX data - E4H in memory location 5000

Output:

Data 1: 34H in memory location 5001(ASCII Code for 4)

Data 2: 45H in memory location 5002(ASCII Code for E)

(iii) **Write a ALP to arrange n numbers in ascending order** (**May/Jun 2016)**

```
LXI H,5000    // Set pointer for array
      MOV C,M     // Load the Count
      DCR C       // Decrement Count
 REPEAT: MOV D,C
      LXI H,5001
  LOOP: MOV A,M      // copy content of memory location to Accumulator
      INX H
      CMP M
      JC SKIP      // jump to skip if carry generated
      MOV B,M      // copy content of memory location to B - Register
      MOV M,A      // copy content of Accumulator to memory location
      DCX H        // Decrement content of HL pair of registers
      MOV M,B      // copy content of B - Register to memory location
      INX H       // Increment content of HL pair of registers
  SKIP: DCR D      // Decrement content of Register - D
      JNZ LOOP     // jump to loop if not equal to zero
      DCR C       // Decrement count
      JNZ REPEAT   // jump to repeat if not equal to zero
      HLT         // Terminate Program
```

(iv) **Write a program to output square wave 1 kHz frequency on the SOD pin of 8085 for 5 seconds.** (**May/Jun 2016).**

```
 LXI SP, 27FFH          // Initialize stack pointer
    LXI B, 1388H        // Initialize counter with count 5000.
BACK: MVI A, COH
    SIM                 // Send high on SOD pin
    CALL DELAY          // Wait for 0.5 msec
    MVI A, 40H          // Send low on SOD pin
    CALL DELAY          // wait for. 5 msec
    DCX B               // Decrement count by 1
    MOV A, C
    ORA B               // Check if count = 0
    JNZ BACK            // If not, repeat
    HLT                 // Stop program execution
```

Delay subroutine:

```
Delay: LXI D, Count
Back: DCX D
      MOV A, D
      ORA E
      JNZ Back
      RET
```

(V) Write an 8085 assembly language program to divide an 8 bit number by another 8 bit number (May/June 2017)

ALGORITHM:

1) Start the program by loading HL register pair with address of memory location.
2) Move the data to a register (B register).
3) Get the second data and load into Accumulator.
4) Compare the two numbers to check for carry.
5) Subtract the two numbers.
6) Increment the value of carry .
7) Check whether repeated subtraction is over and store the value of product and carry in memory location.
8) Terminate the program.

PROGRAM:

```
        LXI  H, 4150
        MOV  B,M     Get the dividend in B – reg.
        MVI  C,00    Clear C – reg for qoutient
        INX  H
        MOV A,M      Get the divisor in A – reg.
NEXT:   CMP B        Compare A - reg with register B.
        JC  LOOP     Jump on carry to LOOP
        SUB  B       Subtract A – reg from B- reg.
        INR C        Increment content of register C.
        JMP NEXT     Jump to NEXT
LOOP:   STA 4152     Store the remainder in Memory
        MOV A,C
        STA 4153     Store the quotient in memory
        HLT          Terminate the program.
```

RESULT:

Input:

F (4150)
FF (4251)

Output:

01 (4152) ---- Remainder
FE (4153) ---- Quotient

(VI) Write an 8085 ALP to find the largest among N numbers [May/June 2017].

ALGORITHM:

1) Load the address of the first element of the array in HL pair
2) Move the count to B – reg.
3) Increment the pointer
4) Get the first data in A – reg.
5) Decrement the count.
6) Increment the pointer
7) Compare the content of memory addressed by HL pair with that of A - reg.
8) If Carry = 0, go to step 10 or if Carry = 1 go to step 9
9) Move the content of memory addressed by HL to A – reg.
10) Decrement the count
11) Check for Zero of the count. If ZF = 0, go to step 6, or if ZF = 1 go to next step.
12) Store the largest data in memory.
13) Terminate the program.

PROGRAM:

```
        LXI H, 4200 Set pointer for array
        MOV B, M    Load the Count
        INX H       Set 1st element as largest data
        MOV A, M
        DCR B       Decrements the count
LOOP:   INX H
        CMP M       f A- reg > M go to AHEAD
        JNC AHEAD
        MOVA, M     Set the new value as largest
AHEAD:  DCR B
        JNZ LOOP    Repeat comparisons till count = 0
        STA 4300    Store the largest value at 4300
        HLT
```

RESULTS:

Input: 05 (4200) Array Size

Output: 0A (4201)
F1 (4202)
1F (4203)
26 (4204)

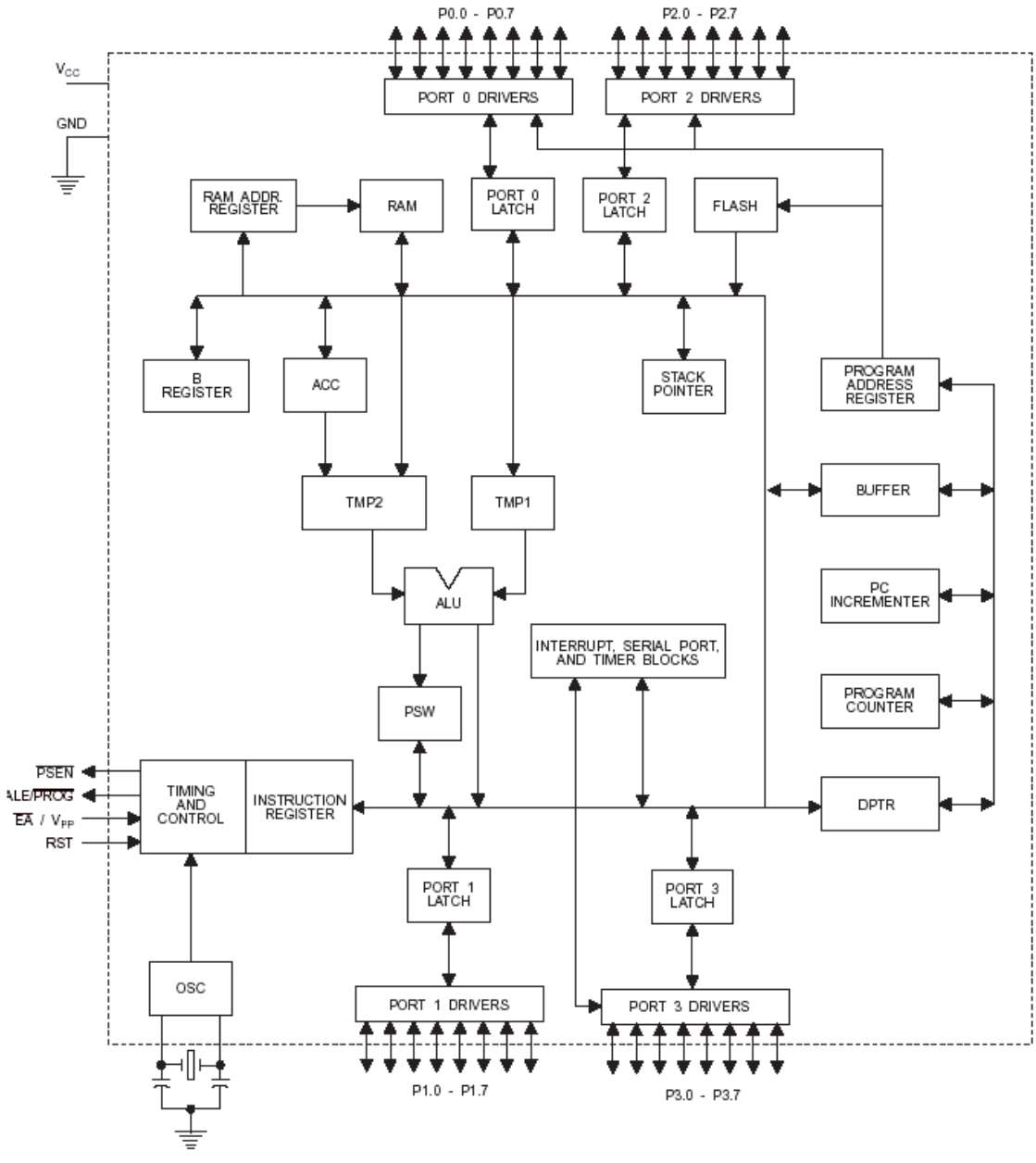
UNIT 3 - 8051 MICRO CONTROLLER

PART B

1. **Describe the architecture of 8051 with neat diagram.** (**May/Jun 2016, Nov/Dec 2015)(May/June 2017)**

The 8051 is a Harvard architecture (separate instruction/data memories) MCS-51 family of single chip microcontroller developed by Intel. This is mostly used microcontroller in the robotics, home appliances like mp3 player, washing machines, electronic iron and industries. The block diagram is given below

Block Diagram



ALU

Arithmetic Logical Unit .This unit is used for the arithmetic calculations.

A-ACCUMULATOR

This register is used for arithmetic operations. This is also bit addressable and 8 bit register.

B-REGISTER

This register is used in only two instructions MUL AB and DIV AB. This is also bit addressable and 8 bit register.

PC-PROGRAM COUNTER

PC Points to the address of next instruction to be executed from ROM .It is 16 bit register means the 8051 can access program address from 0000H to FFFFH.

8051 FLAG BITS AND PSW REGISTER

Flag register in 8051 is called as program status word (PSW). This special function register PSW is also bit addressable and 8 bit wide means each bit can be set or reset independently.

PSW0.7	PSW0.6	PSW0.5	PSW0.4	PSW0.3	PSW0.2	PSW0.1	PSW0.0
CY	AC	F0	RS1	RS0	OV	—	P

There are four flags in 8051

P → Parity flag → PSW 0.0
1 – odd number of 1 in ACC
0 – even number of 1 in ACC

OV (PSW 0.2) → overflow flag → this is used to detect error in signed arithmetic operation. This is similar to carry flag but difference is only that carry flag is used for unsigned operation.

RS1 (PSW0.4) & RS0 (PSW0.3) Register Bank Select

0 0 Bank 0
0 1 Bank 1
1 0 Bank 2
1 1 Bank 3

for selecting Bank 1, we use following commands

SETB PSW0.3 (means RS0=1)
CLR PSW0.4 (means RS1=0)
Initially by default always Bank 0 is selected.

F0 → user definable bit

AC → Auxiliary carry flag → when carry is generated from D3 to D4, it is set to 1.

CY → carry flag → Affected after 8 bit addition and subtraction. It is used to detect error in unsigned arithmetic operation. We can also use it as single bit storage.

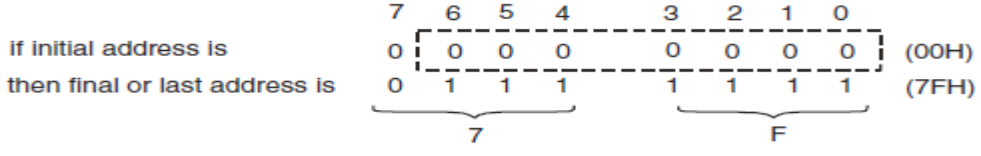
SETB C → for cy = 1

CLR C → for cy = 0

RAM /Data memory (128 Byte)

The 8051 has 128 byte Random Access memory for data storage RAM consists of the register banks, stack for temporary data storage. It also consists of some special function register (SFR) which are used for some specific purpose like timer, input output ports etc. Normally microcontroller has 256 byte RAM in which 128 byte is used for user space which is normally Register banks and stack. But other 128 byte RAM which consists of SFRs.

- We know that 128 byte = 2^7 byte



- Since 27 bytes so last 7 bits can be changed so total locations are from 00H to 7F H. This procedure of calculating the memory address is called as "memory mapping". We can save data on memory locations from 00H to 7FH.

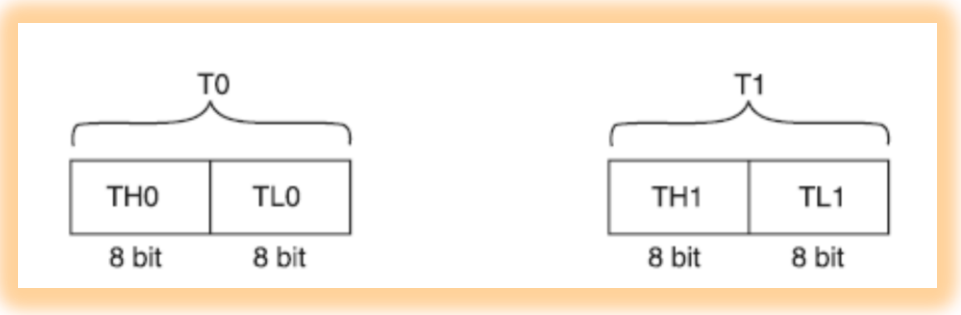
ROM (4KB)

- In 8051, 4KB read only memory (ROM) is available for program storage.
- The total 4KB locations are available from 0000H to 0FFFH. At which we can save the program.

TIMERS AND COUNTERS

- Timer means which can give the delay of particular time between some events. This delay can be provided through some assembly program but in microcontroller two hardware pins are available for delay generation. These hardware pins can be also used for counting some external events.

- In8051, two timer pins are available T0 and T1, by these timers we can give the delay of particular time if we use these in timer mode. Two special function registers are available.



STACK POINTER (SP)

SP is a 8bit register. It indicates current RAM address available for stack or it points the top of stack. The SP is Initially by default at 07H because first location of stack is 08H.

- After each PUSH instruction the SP is incremented by one after PUSH instruction SP is decremented.
- After each POP instruction the SP is decremented.

DPTR (DATA POINTER)

DPTR is a 16 bit register; it is divided into two parts DPH and DPL.

→ DPH for Higher order 8 bits, DPL for lower order 8 bits.

→ DPTR, DPH, DPL these all are SFR) s in 8051.

SERIAL PORT

There are two pins available for serial communication TXD and RXD.

- Normally TXD is used for transmitting serial data which is in SBUF register.
- RXD is used for receiving the serial data.
- SCON register is used for controlling the operation.

INPUT OUTPUT PORTS (I/O PORTS)

- There are four input output ports available P0, P1, P2, P3.

- Each port is 8 bit wide and has special function register P0, P1, P2, P3.which are bit addressable means each bit can be set or reset by the Bit instructions (SETB for high, CLR for low) independently.

- The port 0 can perform dual works. It is also used as Lower order address bus (A0 to A7) multiplexed with 8 bit data bus P0.0 to P0.7 is AD0 to AD7 respectively the address bus and data bus is demultiplex by the ALE signal and latch which is further discussed in details.

- Port 2 can be used as I/O port as well as higher order address bus A8 toA15.

- Port 3 also have dual functions it can be worked as I/O as well as each pin of P3 has specific function.

P3.0 – RXD – Serial I / P for Asynchronous communication Serial O / P for Synchronous communication.

P3.1 – TXD – Serial data transmit.

P3.2 – INT0 – External Interrupt 0.

P3.3 – INT1 – External Interrupt 1.

P3.4 – T0 – Clock input for counter 0.

P3.5 – T1 – Clock input for counter 1.

P3.6 – WR – Signal for writing to external memory.

P3.7 – RD – Signal for reading from external memory.

When external memory is interfaced with 8051 then P0 and P2 can't be worked as I/O port they works as address bus and data bus, otherwise they can be accessed as I/O ports.

OSCILLATOR

It is used for providing the clock to MC8051 which decides the speed or baud rate of MC. We use crystal which frequency varies from 4MHz to 30 MHz; normally we use 11.0592 MHz frequency.

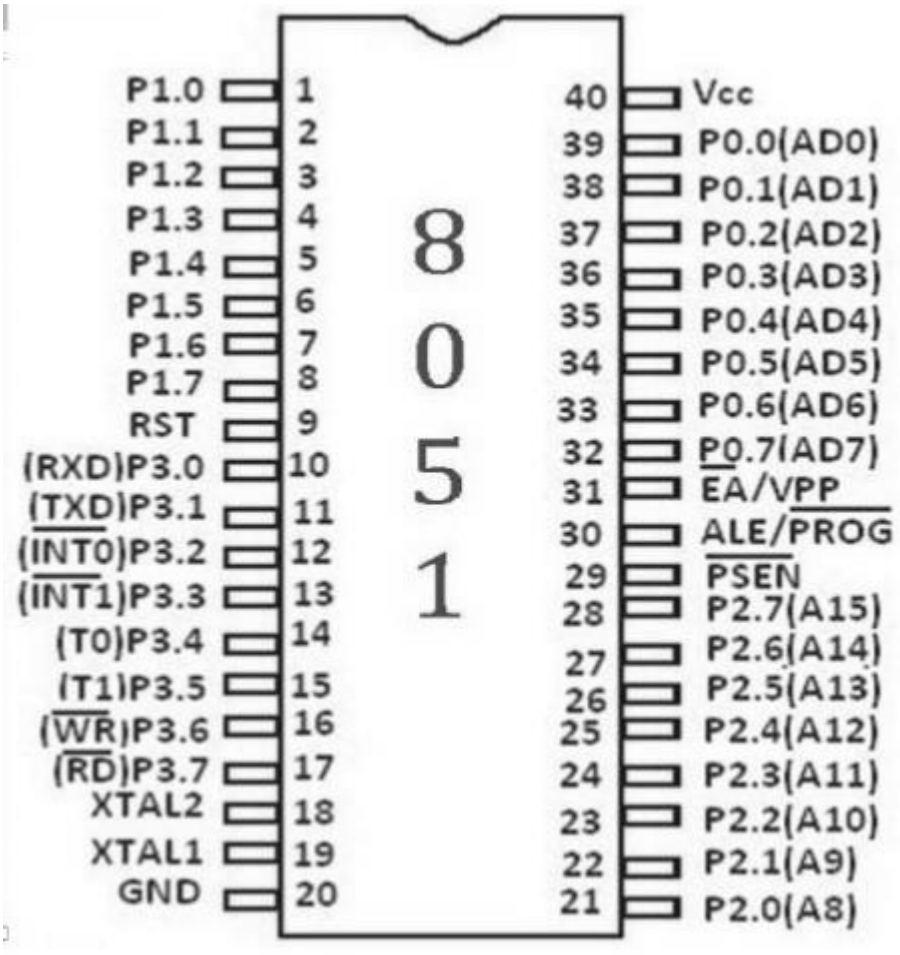
INTERRUPTS

Interrupts are defined as requests because they can be refused (masked) if they are not used, that is when an interrupt is acknowledged. A special set of events or routines are followed to handle the interrupts. These special routines are known as interrupt handler or interrupt service routines (ISR).These are located at a special location in memory.

- INT0 and INT1 are the pins for external interrupts.

---

2. Explain the functional pin diagram of 8051 Microcontroller.



- **Vcc（pin 40）**：
  - Vcc provides supply voltage to the chip.
  - The voltage source is +5V.
- **GND（pin 20）**：ground
- **XTAL1 and XTAL2（pins 19,18）：**
  - These 2 pins provide external clock.
  - Way 1：using a quartz crystal oscillator
  - Way 2：using a TTL oscillator

- **RST（PIN 9）：RESET**

  - It is an input pin and is active high（normally low）.
    - The high pulse must be high at least 2 machine cycles.

  - It is a power-on reset.
    - Upon applying a high pulse to RST, the microcontroller will reset and all values in registers will be lost.
    - Reset values of some 8051 registers.

- **/EA（PIN 31）：EXTERNAL ACCESS**

  - The /EA pin is connected to GND to indicate the code is stored externally.
  - /PSEN & ALE are used for external ROM.
  - For 8051, /EA pin is connected to Vcc (5 V).
  - "/" means active low.

- **/PSEN（PIN 29）：PROGRAM STORE ENABLE**

  - This is an output pin and is connected to the OE pin of the ROM.

- **ALE（PIN 30）：ADDRESS LATCH ENABLE**

  - It is an output pin and is active high.
  - 8051 port 0 provides both address and data.
  - The ALE pin is used for de-multiplexing the address and data .

- **I/O PORT PINS**

  - The four ports P0, P1, P2, and P3.
  - Each port uses 8 pins.
  - All I/O pins are bi-directional.

---

3. Explain the ROM and RAM structure of 8051 (Dec 2016).

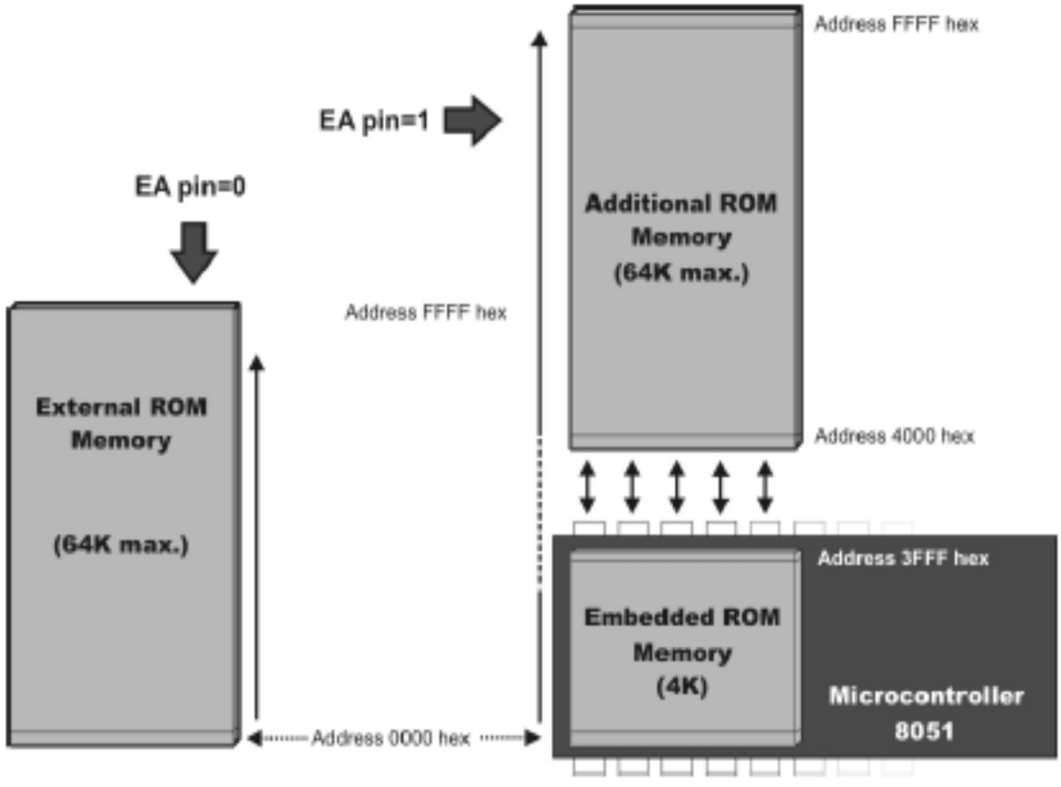
The 8051 microcontroller's memory is divided into Program Memory and Data Memory. Program Memory (ROM) is used for permanent saving program being executed, while Data Memory (RAM) is used for temporarily storing and keeping intermediate results and variables.

3.1PROGRAM MEMORY (ROM)

Program Memory (ROM) is used for permanent saving program (CODE) being executed. The memory is read only. Depending on the settings made in compiler, program

memory may also use to store a constant data. The 8051 executes programs stored in program memory only. Code memory type specified is used to refer to program memory.

8051 memory organization alows external program memory to be added. How does the microcontroller handle external memory depends on the pin EA logical state. This is shown in the following figure.



3.2. INTERNAL DATA MEMORY (RAM)

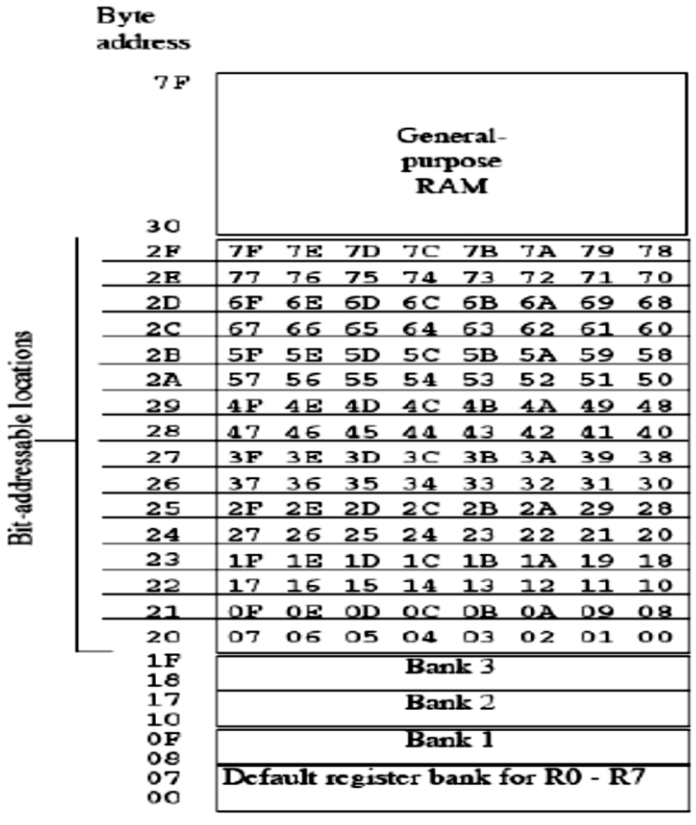
Up to 256 bytes of internal data memory are available depending on the 8051 derivative. Locations available to the user occupy addressing space from 0 to 7Fh, i.e.

First 128 registers and this part of RAM is divided in several blocks. The first 128 bytes of internal data memory are both directly and indirectly addressable. The upper 128 bytes of data memory (from 0x80 to 0xFF) can be addressed only indirectly.

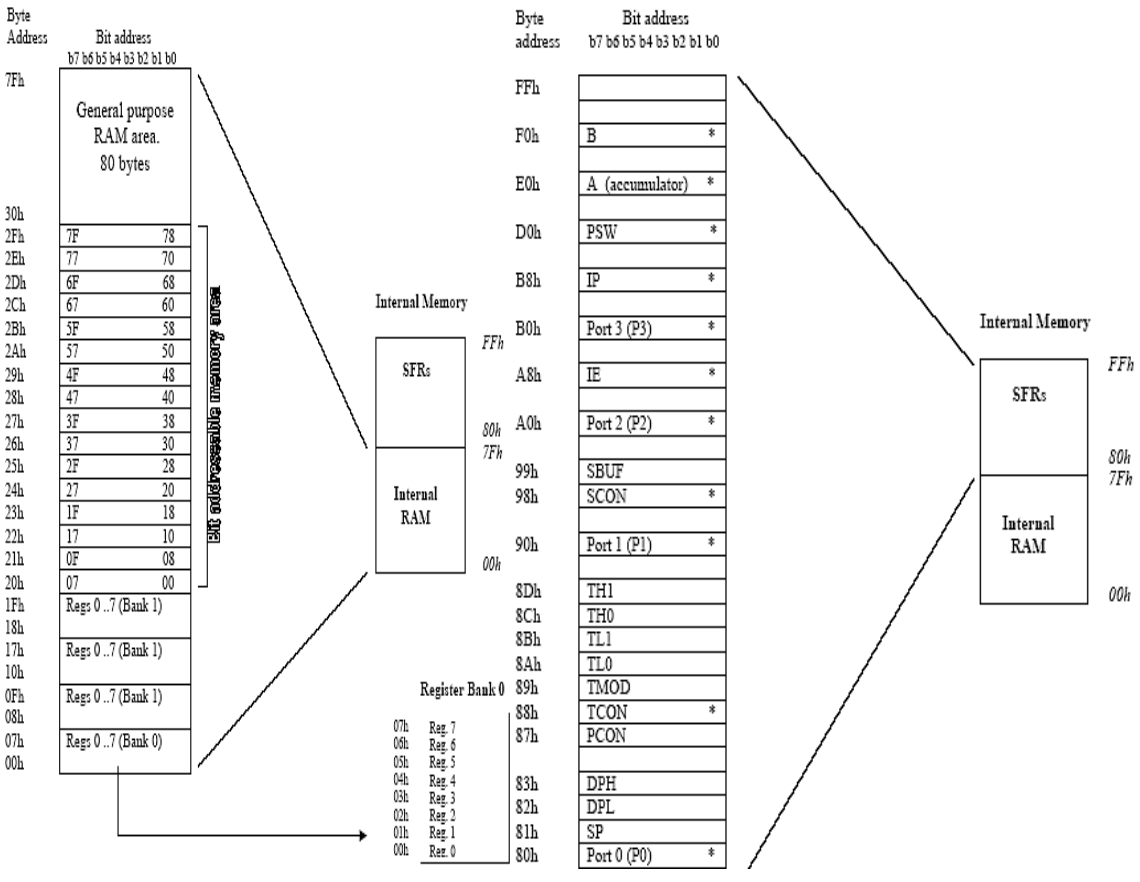
Since internal data memory is used for CALL stack also and there is only 256 bytes splited over few different memory areas fine utilizing of this memory is crucial for fast and compact code.

Memory block in the range of 20h to 2Fh is bit addressable, which means that each bit being there has its own address from 0 to 7Fh. Since there are 16 such registers, this block contains in total of 128 bits with separate addresses (Bit 0 of byte 20h has the bit address 0, and bit 7 of byte 2Fh has the bit address 7Fh).

RAM MEMORY MAP



8051 INTERNAL RAM PARTITIONING & SFR



EXTERNAL DATA MEMORY

Access to external memory is slower than access to internal data memory. There may be up to 64K Bytes of external data memory. Several 8051 devices provide on chip XRAM space that is accessed with the same instructions as the traditional external data space. This XRAM space is typically enabled via proper setting of SFR register and overlaps the external memory space. Setting of that register must be manually done in code, before any access to external memory or XRAM space is made.

SPECIAL FUNCTION REGISTERS (SFRS)

The 8051 provides 128 bytes of memory for SFRs. There **are 21 Special function registers (SFR)** in 8051 micro controller and this includes Register A, Register B, PSW, PCON etc. There are 21 unique locations for these 21 special function registers and each of these register is of 1 byte size. Some of these special function registers ***are bit addressable*** (which means you can access 8 individual bits inside a single byte), while some others are ***only byte addressable***. The SFR are listed in the below table.

Symbol	Name	Address
ACC*	Accumulator	0E0H
B*	B register	0F0H
PSW*	Program status word	0D0H
SP	Stack pointer	81H
DPTR	Data pointer 2 bytes	
DPL	Low byte	82H
DPH	High byte	83H
P0*	Port 0	80H
P1*	Port 1	90H
P2*	Port 2	0A0H
P3*	Port 3	0B0H
IP*	Interrupt priority control	0B8H
IE*	Interrupt enable control	0A8H
TMOD	Timer/counter mode control	89H
TCON*	Timer/counter control	88H
T2CON*	Timer/counter 2 control	0C8H
T2MOD	Timer/counter mode control	0C9H
TH0	Timer/counter 0 high byte	8CH
TL0	Timer/counter 0 low byte	8AH
TH1	Timer/counter 1 high byte	8DH
TL1	Timer/counter 1 low byte	8BH
TH2	Timer/counter 2 high byte	0CDH
TL2	Timer/counter 2 low byte	0CCH
RCAP2H	T/C 2 capture register high byte	0CBH
RCAP2L	T/C 2 capture register low byte	0CAH
SCON*	Serial control	98H
SBUF	Serial data buffer	99H
PCON	Power control	87H

* Bit-addressable

4. Explain the I/O ports and data transfer concepts of 8051. (Dec 2016)

- There are four input output ports available **P0, P1, P2, P3.**
- Each port is 8 bit wide and has special function register P0, P1, P2, P3.which are bit addressable means each bit can be set or reset by the Bit instructions (SETB for high, CLR for low) independently.
- The port 0 can perform dual works. It is also used as Lower order address bus (A0 to A7) multiplexed with 8 bit data bus P0.0 to P0.7 is AD0 to AD7 respectively the address bus and data bus is de-multiplex by the ALE signal and latch which is further discussed in details.
- Port 2 can be used as I/O port as well as higher order address bus A8 toA15.
- Port 3 also have dual functions it can be worked as I/O as well as each pin of P3 has specific function.

P3.0 – RXD – Serial I / P for Asynchronous communication Serial O / P for Synchronous communication.

P3.1 – TXD – Serial data transmit.

P3.2 – INT0 – External Interrupt 0.

P3.3 – INT1 – External Interrupt 1.

P3.4 – T0 – Clock input for counter 0.

P3.5 – T1 – Clock input for counter 1.

P3.6 – WR – Signal for writing to external memory.

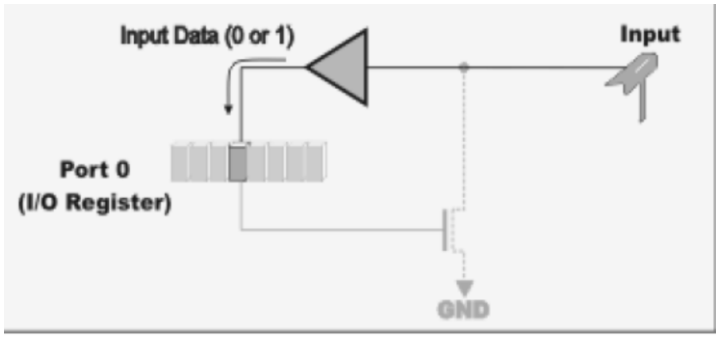
P3.7 – RD – Signal for reading from external memory.

When external memory is interfaced with 8051 then P0 and P2 can't be worked as I/O port they works as address bus and data bus, otherwise they can be accessed as I/O ports. The port addresses are listed in the below table.

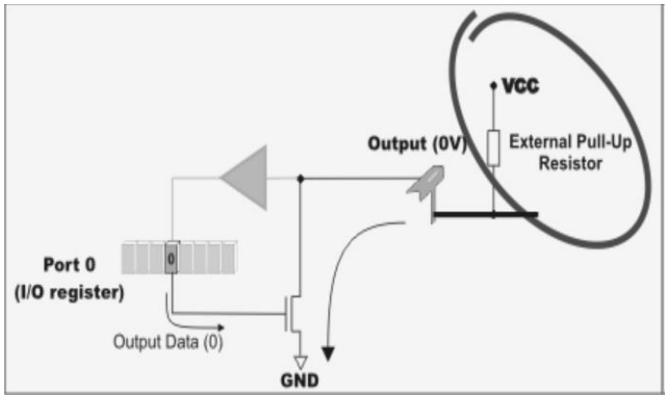
P0	Addr	P1	Addr	P2	Addr	P3	Addr	Port's Bit
P0.0	80	P1.0	90	P2.0	A0	P3.0	B0	D0
P0.1	81	P1.1	91	P2.1	A1	P3.1	B1	D1
P0.2	82	P1.2	92	P2.2	A2	P3.2	B2	D2
P0.3	83	P1.3	93	P2.3	A3	P3.3	B3	D3
P0.4	84	P1.4	94	P2.4	A4	P3.4	B4	D4
P0.5	85	P1.5	95	P2.5	A5	P3.5	B5	D5
P0.6	86	P1.6	96	P2.6	A6	P3.6	B6	D6
P0.7	87	P1.7	97	P2.7	A7	P3.7	B7	D7

PORT 0

The P0 port is characterized by two functions. If external memory is used then the lower address byte (addresses A0-A7) is applied on it. Otherwise, all bits of this port are configured as inputs/outputs. The other function is expressed when it is configured as an output. Unlike other ports consisting of pins with built-in pull-up resistor connected by its end to 5 V power supply, pins of this port have this resistor left out. This apparently small difference has its consequences:



If any pin of this port is configured as an input then it acts as if it "floats". Such an input has unlimited input resistance and in determined potential.



When the pin is configured as an output, it acts as an "open drain". By applying logic 0 to a port bit, the appropriate pin will be connected to ground (0V). By applying logic 1, the external output will keep on "floating". In order to apply logic 1 (5V) on this output pin, it is necessary to built in an external pull-up resistor.

PORT 1

P1 is a true I/O port, because it doesn't have any alternative functions as is the case with P0, but can be cofigured as general I/O only. It has a pull-up resistor built-in and is completely compatible with TTL circuits.

PORT 2

P2 acts similarly to P0 when external memory is used. Pins of this port occupy addresses intended for external memory chip. This time it is about the higher address byte with addresses A8-A15. When no memory is added, this port can be used as a general input/output port showing features similar to P1.

PORT 3

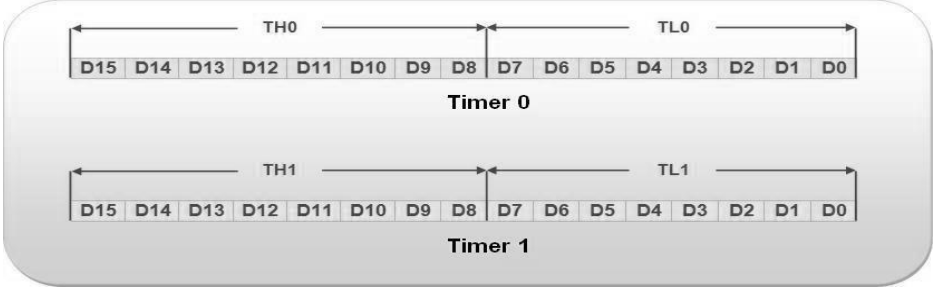
All port pins can be used as general I/O, but they also have an alternative function. In order to use these alternative functions, a logic one (1) must be applied to appropriate bit of the

P3 register. In terms of hardware, this port is similar to P0, with the difference that its pins have a pull-up resistor built-in.

---

5. Explain the timers modes of 8051 (Nov/Dec 2015) (May/June 2017)

The 8051 has two 16-bit on chip timers that can be used for timing durations or for counting external events. The two timers are shown in the below figure .



8051 Timer Mode Control (TMOD) Special Function Register

This is an 8-bit register which is used by both timers 0 and 1 to set the various timer modes. In this TMOD register, lower 4 bits are set aside for timer0 and the upper 4 bits are set aside for timer1. In each case, the lower 2 bits are used to set the timer mode and upper 2 bits to specify the operation.

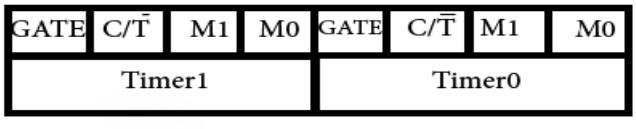


Fig. TMOD Register

- Bit 7: Gate bit; when set, timer only runs while \INT high. (T0)
- Bit 6: Counter/timer select bit; when set timer is an event **c**ounter when cleared timer is an interval **t**imer (T0)
- Bit 5: Mode bit 1 (T0)
- Bit 4: Mode bit 0 (T0)
- Bit 3: Gate bit; when set, timer only runs while \INT high. (T1)
- Bit 2: Counter/timer select bit; when set timer is an event **c**ounter when cleared timer is an interval **t**imer (T1)
- Bit 1: Mode bit 1 (T1)
- Bit 0: Mode bit 0 (T1)

Timer modes

M1	M0	**Mode**	**Description**
0	**0**	**Mode 0**	13-bit mode (not commonly used)
0	**1**	**Mode 1**	16-bit timer mode
1	**0**	**Mode 2**	8-bit auto-reload mode
1	**1**	**Mode 3**	Split timer mode

Mode 1- It is a 16-bit timer; therefore it allows values from 0000 to FFFFH to be loaded into the timer's registers TL and TH. After TH and TL are loaded with a 16-bit initial value, the timer must be started. We can do it by "SETB TR0" for timer 0 and "SETB TR1" for timer 1. After the timer is started. It starts count up until it reaches its limit of FFFFH. When it rolls over from FFFF to 0000H, it sets high a flag bit called TF (timer flag). This timer flag can be monitored. When this timer flag is raised, one option would be stop the timer with the instructions "CLR TR0" or CLR TR1 for timer 0 and timer 1 respectively. Again, it must be noted that each timer flag TF0 for timer 0 and TF1 for timer1. After the timer reaches its limit and rolls over, in order to repeat the process the registers TH and TL must be reloaded with the original value and TF must be reset to 0.

Mode0- Mode 0 is exactly same like mode 1 except that it is a 13-bit timer instead of 16-bit. The 13-bit counter can hold values between 0000 to 1FFFH in TH-TL. Therefore, when the timer reaches its maximum of 1FFH, it rolls over to 0000, and TF is raised.

Mode 2- It is an 8 bit timer that allows only values of 00 to FFH to be loaded into the timer's register TH. After TH is loaded with 8 bit value, the 8051 gives a copy of it to TL. Then the timer must be started. It is done by the instruction "SETB TR0" for timer 0 and "SETB TR1" for timer1. This is like mode 1. After timer is started, it starts to count up by incrementing the TL register. It counts up until it reaches its limit of FFH. When it rolls over from FFH to 00. It sets high the TF (timer flag). If we are using timer 0, TF0 goes high; if using TF1 then TF1 is raised. When Tl register rolls from FFH to 00 and TF is set to 1, TL is reloaded automatically with the original value kept by the TH register. To repeat the process, we must simply clear TF and let it go without any need by the programmer to reload the original value. This makes mode 2 auto reload, in contrast in mode 1 in which programmer has to reload TH and TL.

Mode3- Mode 3 is also known as a split timer mode. Timer 0 and 1 may be programmed to be in mode 0, 1 and 2 independently of similar mode for other timer. This is not true for mode 3; timers do not operate independently if mode 3 is chosen for timer 0. Placing timer 1 in mode 3 causes it to stop counting; the control bit TR1 and the timer 1 flag TF1 are then used by timer0.

TCON register-

Bits and symbol and functions of every bits of TCON are as follows:

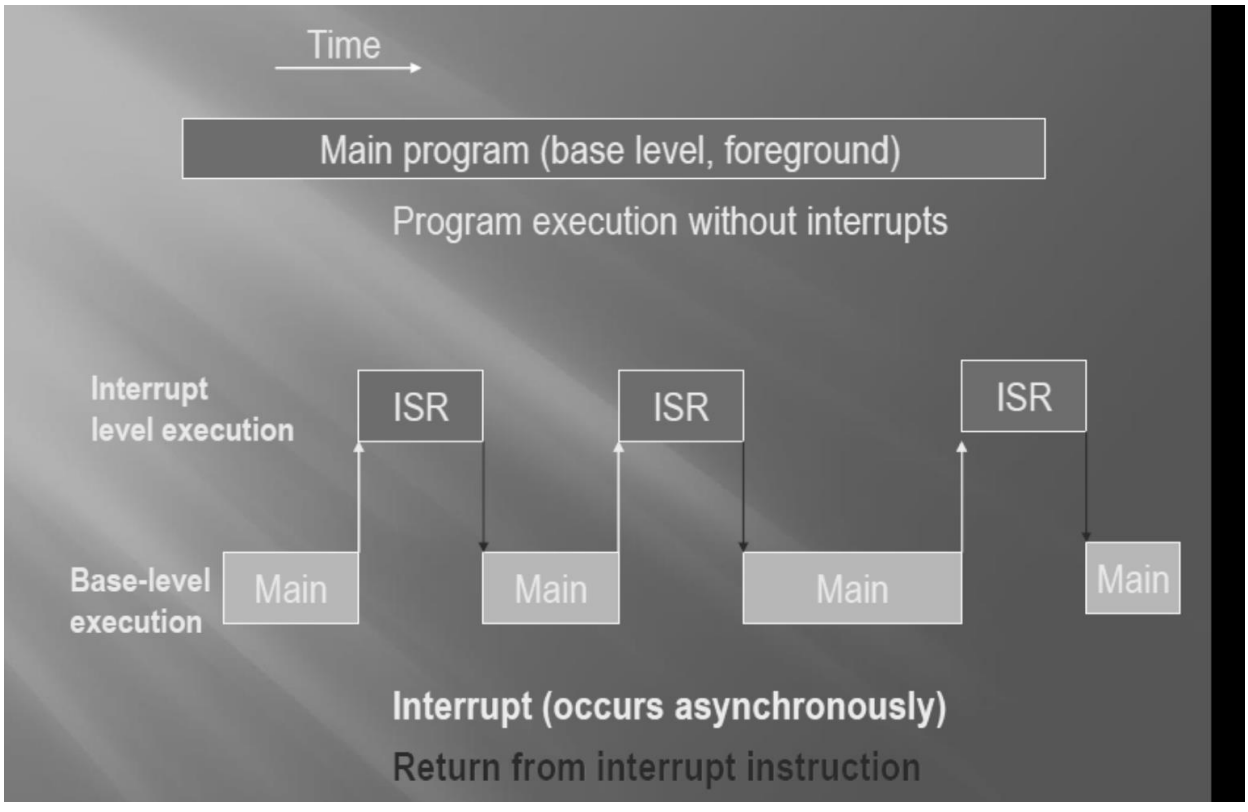
BIT Symbol Functions

7	TF1	Timer1 over flow flag. Set when timer rolls from all 1s to 0. Cleared When the processor vectors to execute interrupt service routine Located at program address 001Bh.
6	TR1	Timer 1 run control bit. Set to 1 by programmer to enable timer to count; Cleared to 0 by program to halt timer.
5	TF0	Timer 0 over flow flag. Same as TF1.
4	TR0	Timer 0 run control bit. Same as TR1.
3	IE1	External interrupt 1 Edge flag. Not related to timer operations.
2	IT1	External interrupt1 signal type control bit. Set to 1 by program to Enable external interrupt 1 to be triggered by a falling edge signal.
1	IE0	External interrupt 0 Edge flag. Not related to timer operations.
0	IT0	External interrupt 0 signal type control bit. Same as IT0.

---

6. Explain the interrupt structure of 8051 microcontroller Explain how interrupts are Prioritized (May/june 2016, Dec 2016).

- An interrupt is the occurrence of a condition--an event -- that cause a temporary suspension of a program while the event is serviced by another program (Interrupt Service Routine ISR or Interrupt Handler).
- Interrupt-Driven System-- gives the illusion of doing many things simultaneously, quick response to events, suitable for real-time control application.
- The entire interrupt handling process is shown in the below figure.



8051 INTERRUPT ORGANIZATION

- 5 interrupt sources: 2 external, 2 timer, a serial port. The interrupts and their **vector addresses** are listed in the below table

Interrupt Number	Interrupt Description	Address
0	EXTERNAL INT 0	0003h
1	TIMER/COUNTER 0	000Bh
2	EXTERNAL INT 1	0013h
3	TIMER/COUNTER 1	001Bh
4	SERIAL PORT	0023h

ENABLING AND DISABLING INTERRUPTS – Using IE Register (INTERRUPT ENABLE REGISTER -A8H)

This register is responsible for enabling and disabling the interrupt. It is a bit addressable register in which EA must be set to one for enabling interrupts. The corresponding bit in this register enables particular interrupt like timer, external and serial inputs. In the below IE register, bit corresponding to 1 activates the interrupt and 0 disables the interrupt.

EA	-	-	ES	ET1	EX1	ET0	EX0

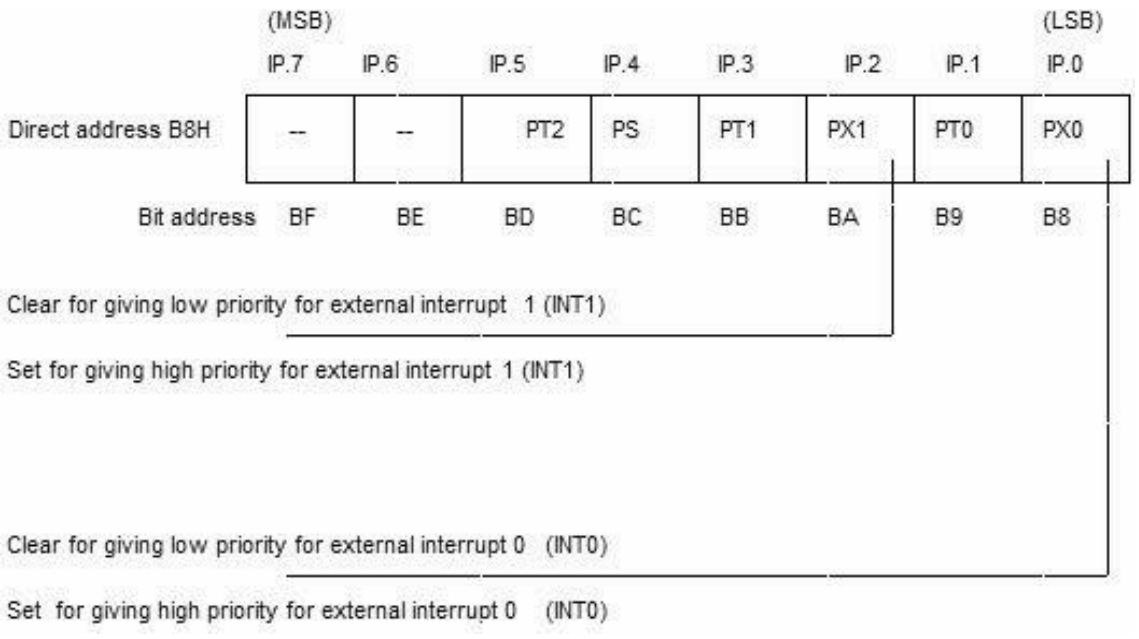
EA	IE.7	Disables all interrupts. If EA = 0, no interrupt will be acknowledged. If EA = 1, interrupt source is individually enable or disabled by setting or clearing its enable bit.
-	IE.6	Not implemented, reserved for future use*.
-	IE.5	Not implemented, reserved for future use*.
ES	IE.4	Enable or disable the Serial port interrupt.
ET1	IE.3	Enable or disable the Timer 1 overflow interrupt.
EX1	IE.2	Enable or disable External interrupt 1.
ET0	IE.1	Enable or disable the Timer 0 overflow interrupt.
EX0	IE.0	Enable or disable External Interrupt 0.

PRIORITY OF INTERRUPTS

The priority levels of the interrupts is changed by setting or clearing the corresponding bit in the Interrupt priority (IP) register as shown in the figure. This allows the low priority interrupt to interrupt the high-priority interrupt, but prohibits the interruption by another low priority interrupt. Similarly, the high-priority interrupt cannot be interrupted. If these interrupt priorities are not programmed, the microcontroller executes in predefined manner and its order is

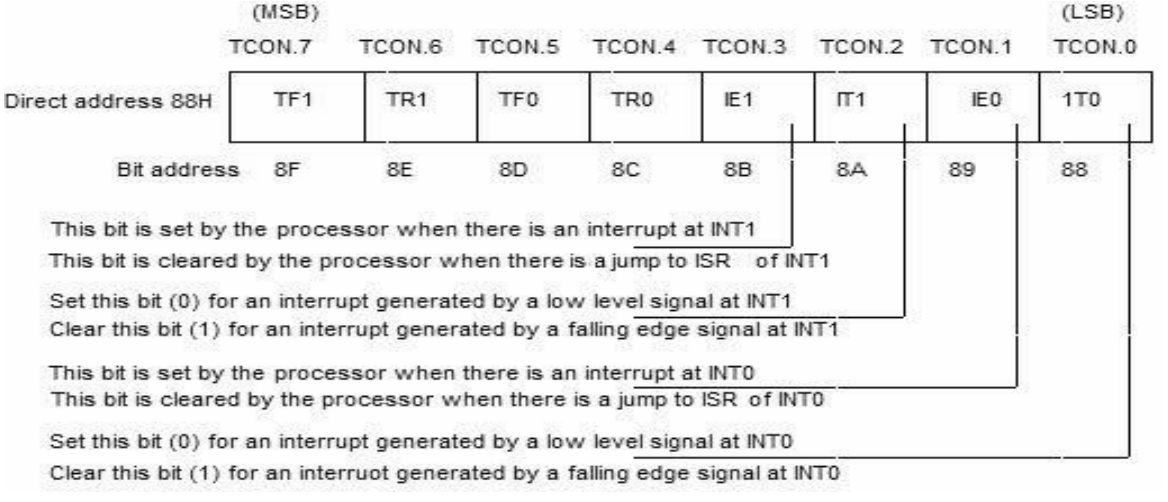
INT0 (Highest)
TF0
INT1
TF1
SERIAL PORT (lowest)

INTERRUPT PRIORITY (IP) REGISTER



TCON REGISTER

In addition to the above two registers, the TCON register specifies the type of external interrupt to the 8051 microcontroller, as shown in the figure. The two external interrupts, whether edge or level triggered, specify by this register by a set, or cleared by appropriate bits in it. And, it is also a bit addressable register. The TCON register is shown in the figure.



---

7. Compare the programming concepts of 8085 with 8051.

S.NO	Assembly language instructions of 8085	Assembly language instructions of 8051
1	**Data Transfer operations:** The MVI instruction is used to load the data in to a register. Ex: MVI A, 05; (A) = 05. MOV instruction is used to transfer either from memory to register and vice versa or from register to register. IN & OUT instructions are used to transfer the data through ports. LXI is used to load a register pair, STAX & STA are the instructions used to store the Accumulator contents in a memory. No flags are affected with these operations.	MOV Destination, Source; Destination = Source, Source contents does not change. Ex 1: MOV A,#55H ; (A) = 55H. The ‘ # ’ signifies that it is a value otherwise it is a memory address. Ex 2: MOV R5,#0F9H ; (R5) = F9H. The 0 between # and F to indicate that F is a hex number and not a letter. The size of destination & Source should be equal. Ex 3: MOV DPTR, #2040H ; (DPTR) = 2040H, which can also be entered as MOV DPH,#20H & MOV DPL,#40H.
2	**Arithmetic operations:** ADD, ADI, ADC, ACI, DAD, SUB, SUI, SBB are for addition and subtraction, in which one of the data and destination is Accumulator. The INR, INX, DCR, DCX are for incrementing and decrement operations. Flags are affected. DA A is for decimal adjustment of A. Multiplication can be in repeated addition and division can be done in repeated subtraction process.	(i) ADD A, Source ; (A) = (A) + Source, (ii) ADDC A, Source ; (A) = (A) + (Source) + CY, (iii) DA A for decimal adjustment. (iv) SUBB A, Source ; (A) = (A) – (Source) – CY (v) MUL AB ; (A) = (A) x (B), Upper byte of product in (B). (vi) DIV AB ; (A) = (A) / (B), the Quotient is in A and Remainder is in B register. INC, DEC are for increment and decrement operations
3	**Logic Operations:** ANA, ANI, ORA, ORI, XRA, XRI and CMA instructions are used to perform bit wise logical operations. All these instructions are in relation to the contents of Accumulator and no flags are affected	ANL Dest,Source ; (Dest) = (Dest) AND (Source), ORL Dest,Source ; (Dest) = (Dest) OR (Source), XOR Dest,Source ; (Dest) = (Dest) XOR (Source), CPL A ; (A) =(ଌ)
4	**Branch Operations:** Various types of Jump (Un conditional and conditional), CALL and Restart instructions are used to change the execution sequence of the program. All types of Jump and CALL instructions are 3 byte instructions	Two types of jumps, LJMP (3 byte) & SJMP (2 byte). All conditional jumps are short jumps only. Two types of call LCALL (3 byte) to call subroutine anywhere in the memory and ACALL (2 byte) to call a routine within the page.
5	**Machine Control Operations:** HLT, NOP and Interrupt instructions are used to indicate the end of the program, no operation and	ORG for origin, END for last line of a program, EQU for counter constant. The Define byte (DB) is used to define data. It defines ASCII strings larger than two

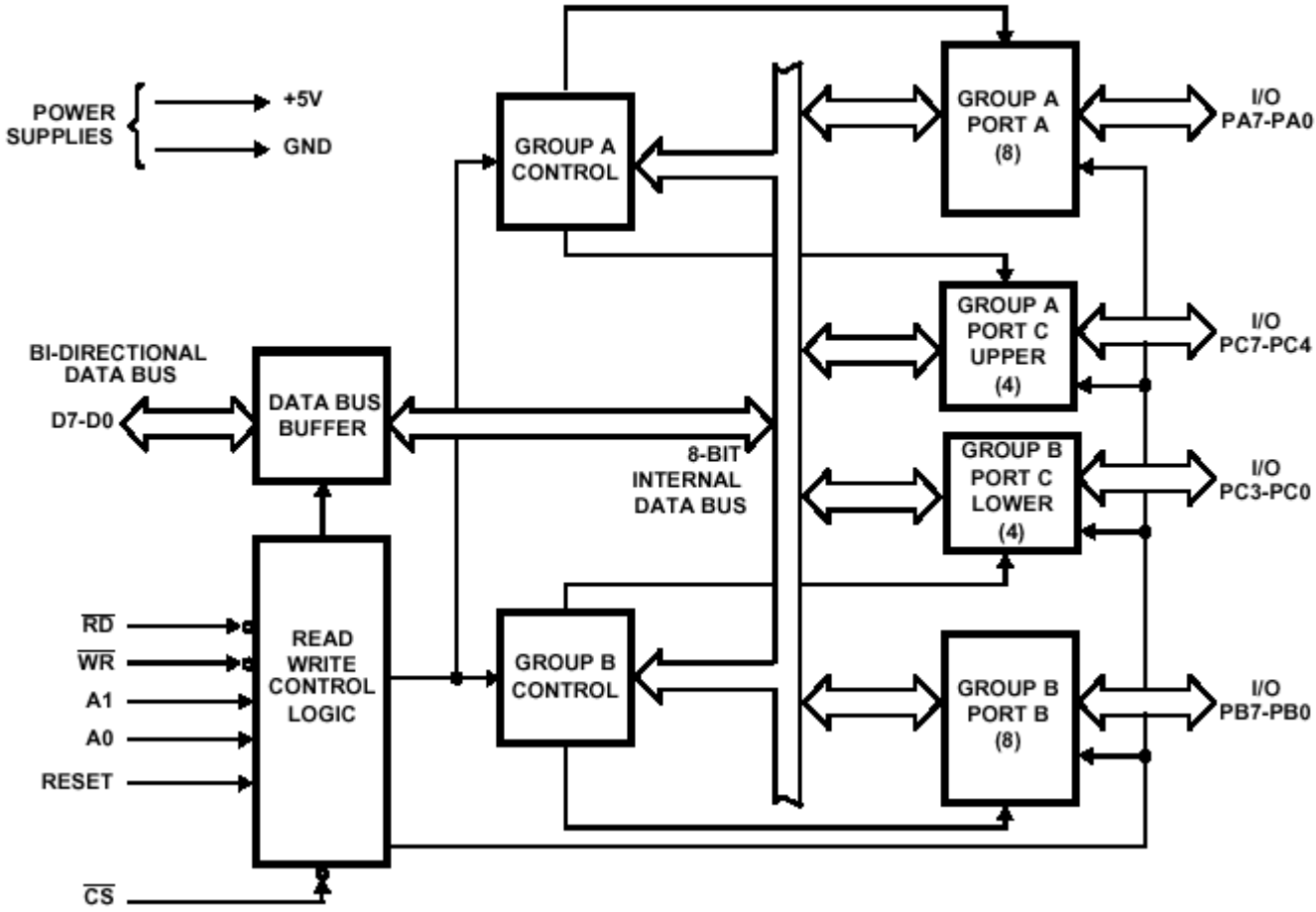
	interrupting the processor through external interrupts.	characters. Ex: DB 0011 0101B binary, DB 39H Hex, DB "2591" ASCII
6	**Logic Operation Rotate:** RLC &RAL are used to rotate Accumulator contents left without and with carry; RAC & RAR are used to rotate Accumulator contents right without and with carry. All are one-byte instructions only. XCHG exchanges HL pair contents with DE pair contents	RR A ; Rotate right the Accumulator contents. RL A ; Rotate left the Accumulator contents. RRC A ; Rotate right Accumulator contents through CY. RLC A ; Rotate left Accumulator contents through CY. SWAP A ; Swapping Accumulator bytes.
7	**Logic Operation Compare:** CMP & CPI ;Compare Accumulator contents with register / with immediate data by subtracting the data from A. If A < R / M; Cy = 1 & Z = 0; If A = R / M; Cy = 0 & Z = 1,If A > R / M; Cy = 0 & Z = 0	CJNE dest, Source, Relative address ; Compare destination contents with Source contents and jump if not equal. Ex: CJNE A,#55,Next ; If (A) ≠ 55H, then jump to Next. Even though it is a conditional jump, It is a 3 byte instruction. No need of involving A in compare instruction
8	**Word Size:** All instructions are classified in 1, 2 and 3 byte instructions based on whether it has only opcode or opcode followed by data or opcode followed by address.	All instructions classified in 1, 2 and 3 byte instructions based on whether it has only opcode or opcode followed by data or opcode followed relative address or full address.
9	CY flag bit related instructions: CMC ;Compliment CY, STC ; Set the Carry status, JC target ; jump to the target if CY = 1, JNC target ; jump to the target if CY≠ 1 CC target ; CALL subroutine labelled by target if CY = 1 CNC target ; CALL subroutine labelled by target if CY≠ 1	SETB C; CLR C; CPL C; MOV b,C; MOV c,b; JNC target; JC target; target is the relative address. ANL c,bit ; AND CY with bit and save it on CY ANL c,/bit ; AND CY with inverted bit and save it on CY ORL c,bit ; OR CY with bit and save it on CY ORL c,/bit ; OR CY with inverted bit and save it on CY
10	**Zero flag bit related instructions:** JZ target; jump to the target if Z = 1, target is 16 bit address. JNZ target ; jump to the target if Z≠ 1, CZ target ; CALL subroutine if Z = 1 CNZ target ; CALL subroutine labelled by target if Z≠ 1	Instead of zero flag bit Accumulator contents are used. JZ ; Jump if A = 0, JNZ ; Jump if A ≠ 0 DJNZ ; decrement and jump if A ≠ 0 CJNE A,byte ; jump if A = byte.

UNIT 4

PERIPHERAL INTERFACING

PART B

1 .Explain the operation of 8255 (PPI) in detail. (Nov/Dec 2015) (May/ June 2017)



DESCRIPTION

Data Bus Buffer

- This three-state bi-directional 8-bit buffer is used to interface the 8255 to the system data bus.
  - Data is transmitted or received by the buffer upon execution of input or output instructions by the CPU.
  - Control words and status words are also transferred through the data bus buffer.

Read/Write and Control Logic

The function of this block is to manage all of the internal and external transfers of both Data and Control or Status words.

$\overline{CS}$ - Chip Select. A "low" on this input pin enables the communication between the 8255 and the CPU.

$\overline{RD}$ - Read. A "low" on this input pin enables 8255 to send the data or status information to the CPU on the data bus. In essence, it allows the CPU to "read from" the 8255.

$\overline{WR}$ - Write. A "low" on this input pin enables the CPU to write data or control words into the 8255.

A0 and A1 - Port Select 0 and Port Select 1. These input signals, in conjunction with the RD and WR inputs, control the selection of one of the three ports or the control word register. They are normally connected to the least significant bits of the address bus (A0 and A1).

A1	A0	SELECTION
0	0	PORT A
0	1	PORT B
1	0	PORT C
1	1	CONTROL

RESET - Reset. A "high" on this input initializes the control register to 9Bh and all ports (A, B, C) are set to the input mode.

Group A and Group B Controls

The functional configuration of each port is programmed by the systems software. In essence, the CPU "outputs" a control word to the 8255.

The control word contains information such as "mode", "bit set", "bit reset", etc., that initializes the functional configuration of the 8255.

Each of the Control blocks (Group A and Group B) accepts "commands" from the Read/Write Control logic, receives "control words" from the internal data bus and issues the proper commands to its associated ports.

Ports A, B, and C

The 8255 contains three 8-bit ports (A, B, and C). All can be configured to a wide variety of functional characteristics by the system software but each has its own special features or "personality" to further enhance the power and flexibility of the 8255.

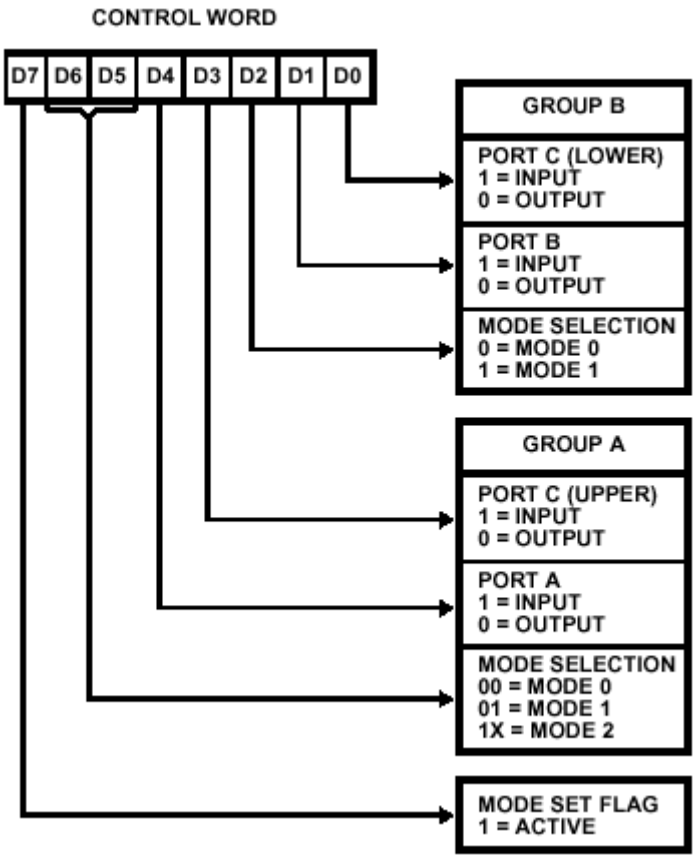
Port A - One 8-bit data output latch/buffer and one 8-bit data input latch. Both "pull-up" and "pull-down" bus-hold devices are present on Port A.

Port B - One 8-bit data input/output latch/buffer and one 8-bit data input buffer.

Port C - One 8-bit data output latch/buffer and one 8-bit data input buffer (no latch for input). This port can be divided into two 4-bit ports under the mode control. Each 4-bit port

contains a 4-bit latch and it can be used for the control signal output and status signal inputs in conjunction with ports A and B.

The control word for PORTS are shown below.

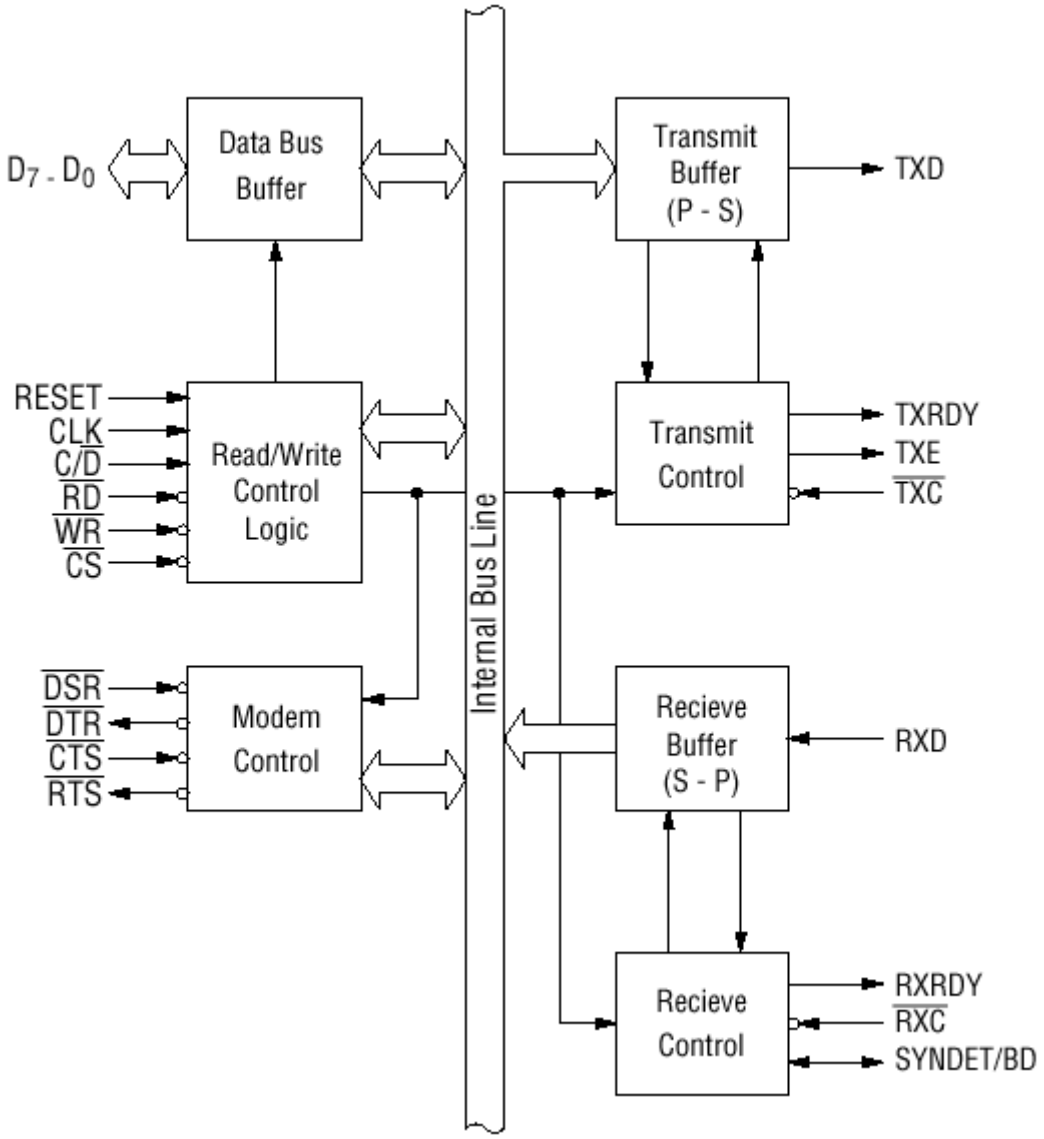


---

2. Discuss how 8251 is used for serial communication of data, Bring about the features of 8251, (May/Jun 2016).

- The 8251 is a USART (Universal Synchronous Asynchronous Receiver Transmitter) for serial data communication.
- This is a peripheral device of a microcomputer system, the 8251 receives parallel data from the CPU and transmits serial data after conversion.
- This device also receives serial data from the outside and transmits parallel data to the CPU after conversion.

Block diagram of the 8251 USART



The 8251 functional configuration is programed by software. The data transfer operation between the 8251 and a CPU is executed by program control. (Table shows the operation)

$\overline{CS}$	C/$\overline{D}$	$\overline{RD}$	$\overline{WR}$	
1	×	×	×	Data Bus 3-State
0	×	1	1	Data Bus 3-State
0	1	0	1	Status → CPU
0	1	1	0	Control Word ← CPU
0	0	0	1	Data → CPU
0	0	1	0	Data ← CPU

CONTROL WORDS

There are two types of control word.

1. Mode instruction (setting of function)

2. Command (setting of operation)

1) MODE INSTRUCTION

Mode instruction is used for setting the function of the 8251.

Mode instruction will be in "wait for write" at either internal reset or external reset. The following Items set by mode instruction.

- Synchronous/asynchronous mode
- Stop bit length (asynchronous mode)
- Character length
- Parity bit
- Baud rate factor (asynchronous mode)
- Internal/external synchronization (synchronous mode)
- Number of synchronous characters (Synchronous mode)

The bit configuration of mode instruction is shown in Figures 2 and 3. In the case of synchronous mode, it is necessary to write one-or two byte sync characters. If sync characters were written, a function will be set because the writing of sync characters constitutes part of mode instruction.

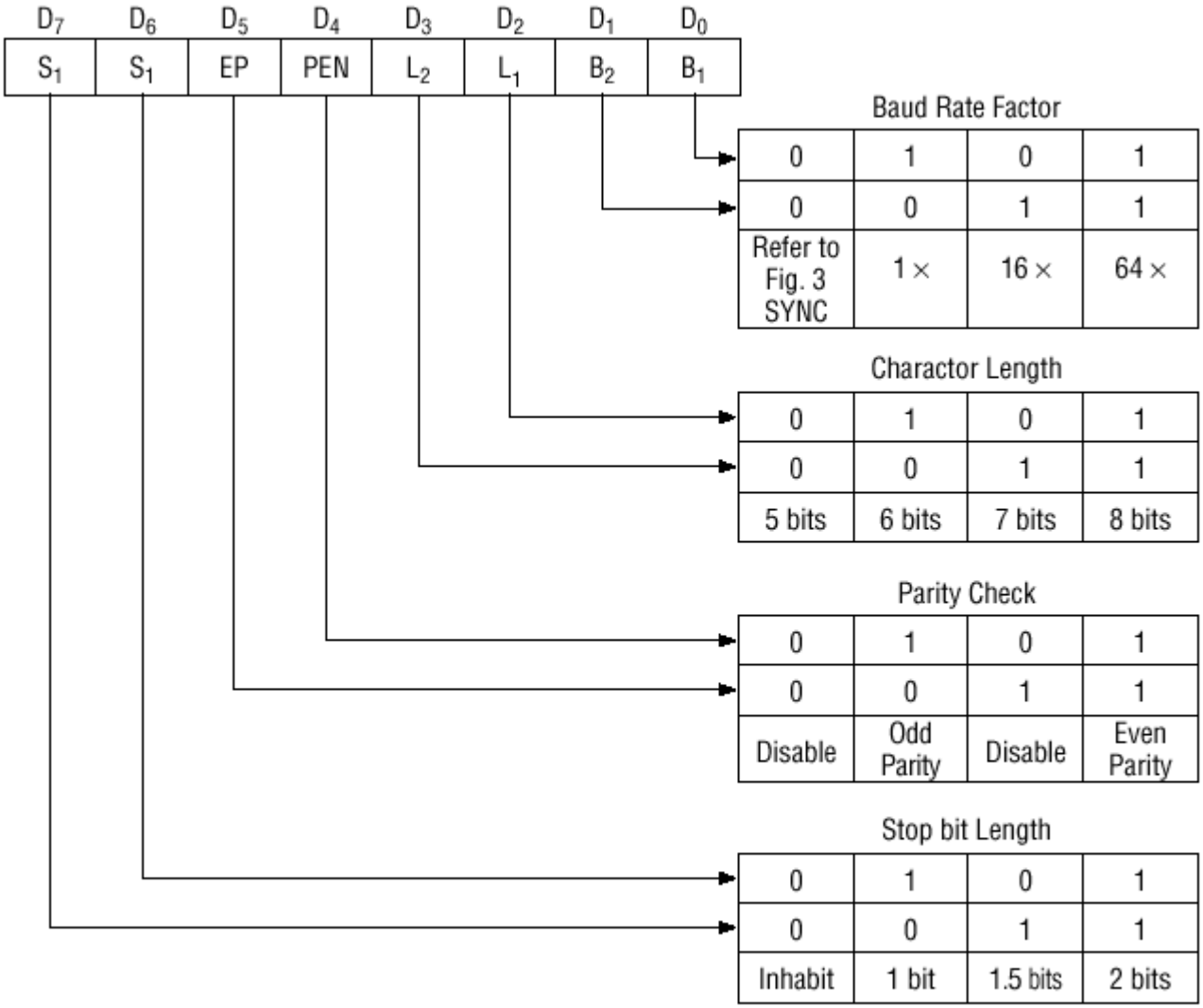


Fig. 2 Bit Configuration of Mode Instruction (Asynchronous)

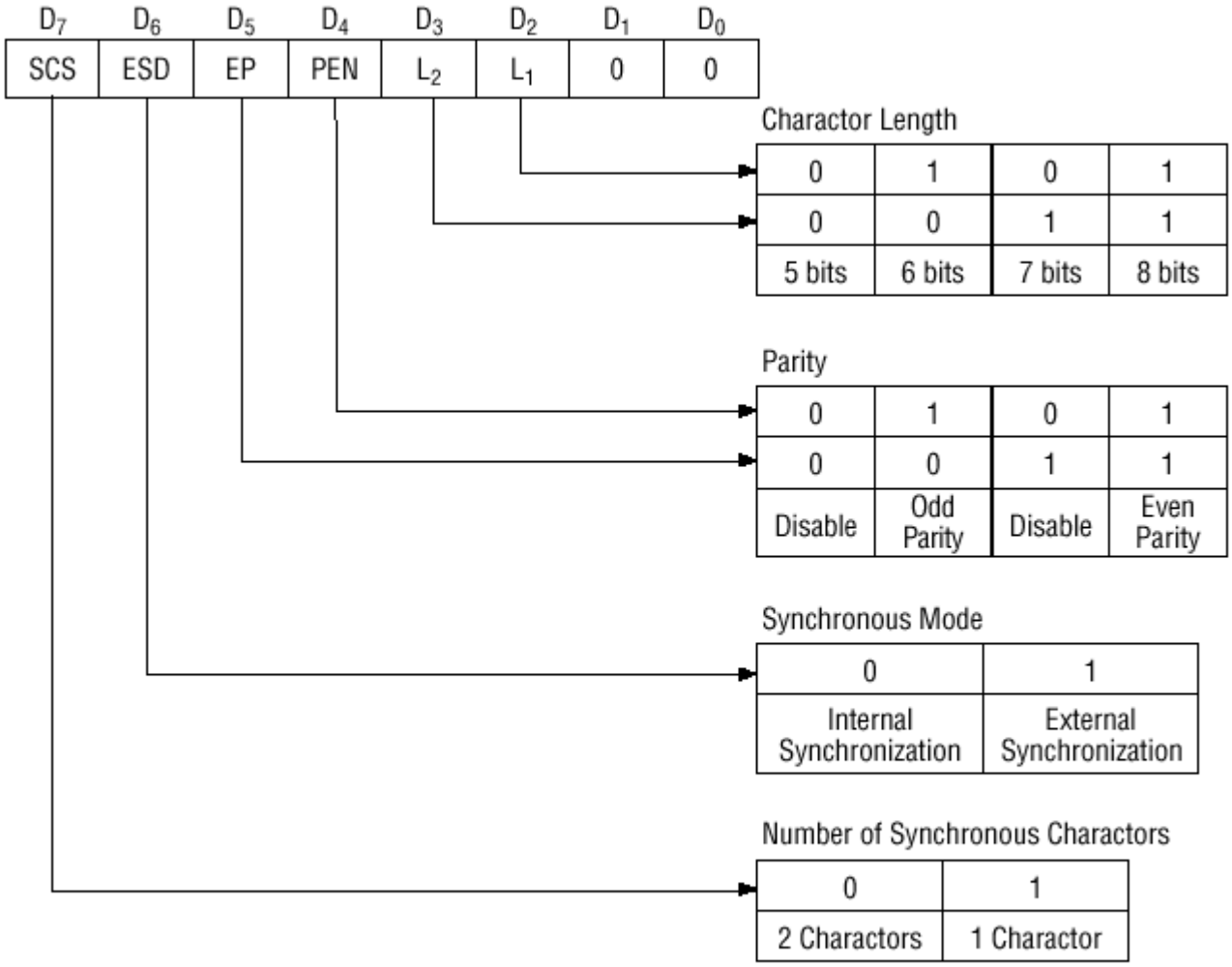


Fig. 3 Bit Configuration of Mode Instruction (Synchronous)

2) COMMAND

Command is used for setting the operation of the 8251. It is possible to write a command whenever necessary after writing a mode instruction and sync characters.

Items to be set by command are

• Transmit Enable/Disable

• Receive Enable/Disable

• DTR, RTS Output of data.

• Resetting of error flag.

• Sending to break characters

• Internal resetting

• Hunt mode (synchronous mode)

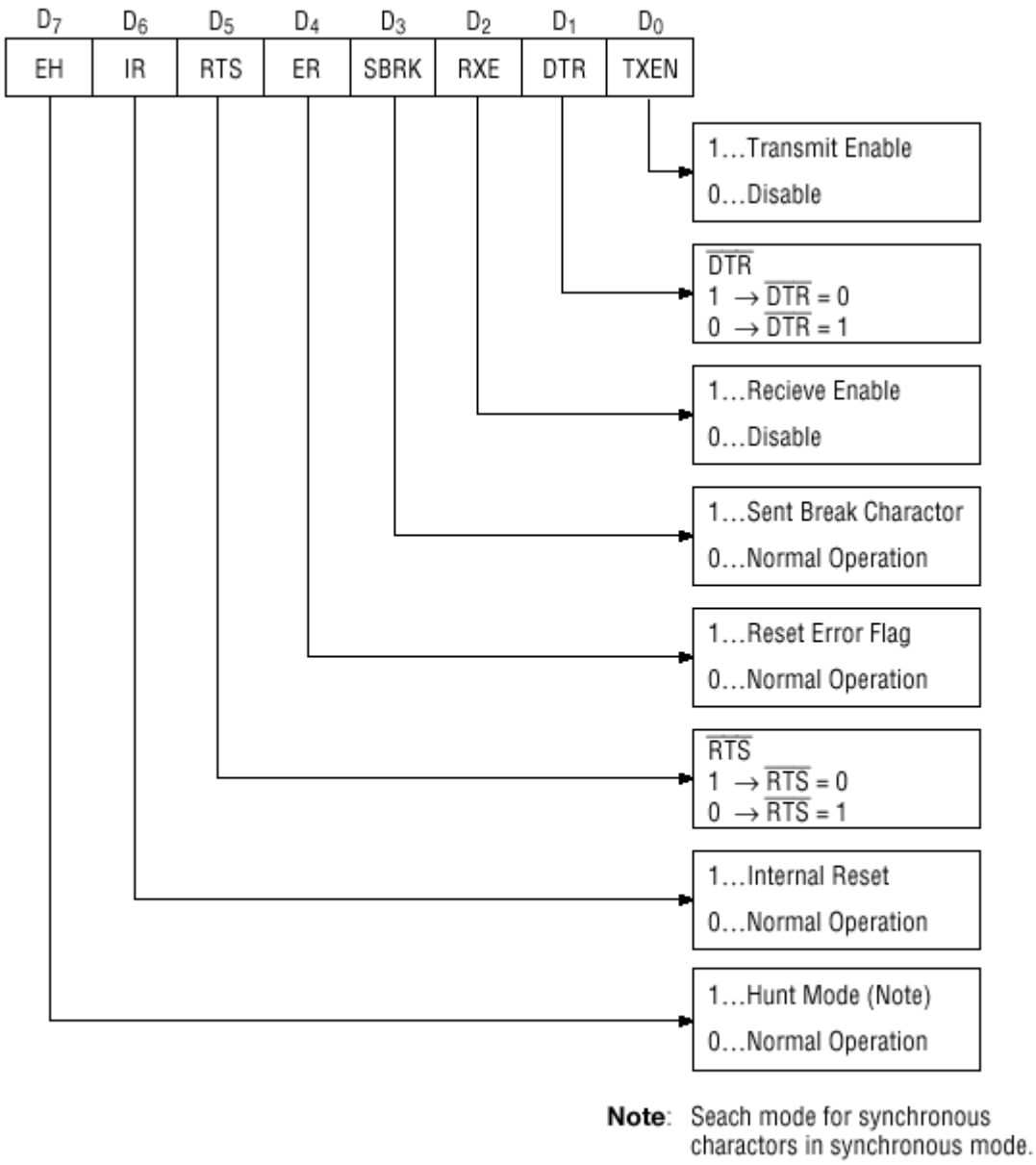


Fig. 4 Bit Configuration of Command

Status Word

It is possible to see the internal status of the 8251 by reading a status word. The bit configuration of status word is shown in Fig. 5.

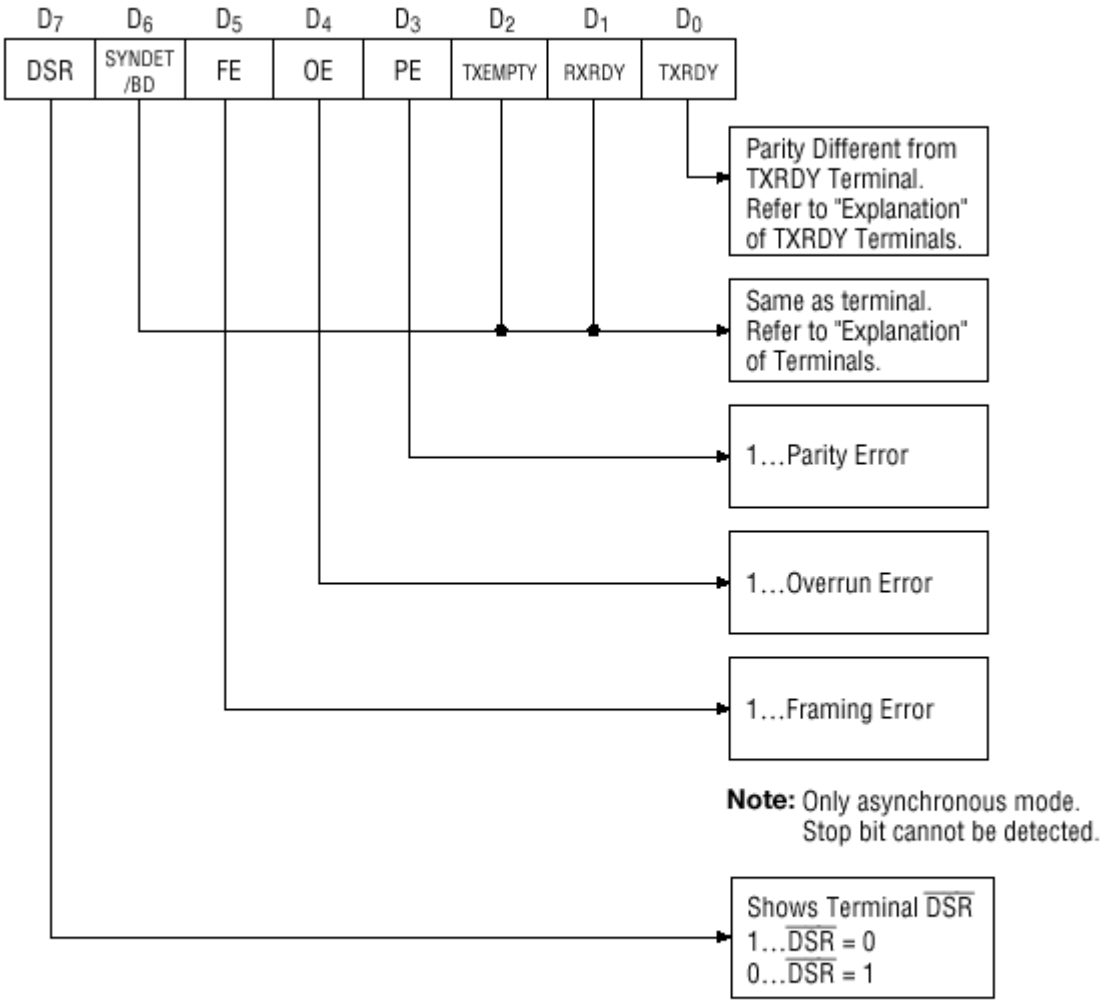
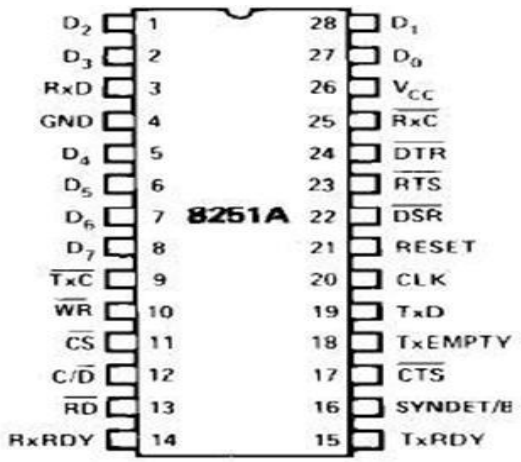


Fig. 5 Bit Configuration of Status Word

PIN & SIGNAL DESCRIPTION



D 0 to D 7 (l/O terminal)

This is bidirectional data bus which receive control words and transmits data from the CPU and sends status words and received data to CPU.

RESET (Input terminal)

A "High" on this input forces the 8251 into "reset status." The device waits for the writing of "mode instruction." The min. reset width is six clock inputs during the operating status of CLK.

CLK (Input terminal)

CLK signal is used to generate internal device timing. CLK signal is independent of RXC or TXC. However, the frequency of CLK must be greater than 30 times the RXC and TXC at Synchronous mode and Asynchronous "x1" mode, and must be greater than 5 times at Asynchronous "x16" and "x64" mode.

WR (Input terminal)

This is the "active low" input terminal which receives a signal for writing transmit data and control words from the CPU into the 8251.

RD (Input terminal)

This is the "active low" input terminal which receives a signal for reading receive data and status words from the 8251.

C/D (Input terminal)

This is an input terminal which receives a signal for selecting data or command words and status words when the 8251 is accessed by the CPU. If C/D = low, data will be accessed. If C/D = high, command word or status word will be accessed.

CS (Input terminal)

This is the "active low" input terminal which selects the 8251 at low level when the CPU accesses. Note: The device won't be in "standby status"; only setting CS = High.

TXD (output terminal)

This is an output terminal for transmitting data from which serial-converted data is sent out. The device is in "mark status" (high level) after resetting or during a status when transmit is disabled. It is also possible to set the device in "break status" (low level) by a command.

TXRDY (output terminal)

This is an output terminal which indicates that the 8251is ready to accept a transmitted data character. But the terminal is always at low level if CTS = high or the device was set in "TX disable status" by a command. Note: TXRDY status word indicates that transmit data

character is receivable, regardless of CTS or command. If the CPU writes a data character, TXRDY will be reset by the leading edge or WR signal.

TXEMPTY (Output terminal)

This is an output terminal which indicates that the 8251 has transmitted all the characters and had no data character. In "synchronous mode," the terminal is at high level, if transmit data characters are no longer remaining and sync characters are automatically transmitted. If the CPU writes a data character, TXEMPTY will be reset by the leading edge of WR signal. Note : As the transmitter is disabled by setting CTS "High" or command, data written before disable will be sent out. Then TXD and TXEMPTY will be "High". Even if a data is written after disable, that data is not sent out and TXE will be "High". After the transmitter is enabled, it sent out. (Refer to Timing Chart of Transmitter Control and Flag Timing)

TXC (Input terminal)

This is a clock input signal which determines the transfer speed of transmitted data. In "synchronous mode," the baud rate will be the same as the frequency of TXC. In "asynchronous mode", it is possible to select the baud rate factor by mode instruction. It can be 1, 1/16 or 1/64 the TXC. The falling edge of TXC sifts the serial data out of the 8251.

RXD (input terminal)

This is a terminal which receives serial data.

RXRDY (Output terminal)

This is a terminal which indicates that the 8251 contains a character that is ready to READ. If the CPU reads a data character, RXRDY will be reset by the leading edge of RD signal. Unless the CPU reads a data character before the next one is received completely, the preceding data will be lost. In such a case, an overrun error flag status word will be set.

RXC (Input terminal)

This is a clock input signal which determines the transfer speed of received data. In "synchronous mode," the baud rate is the same as the frequency of RXC. In "asynchronous mode," it is possible to select the baud rate factor by mode instruction. It can be 1, 1/16, 1/64 the RXC.

SYNDET/BD (Input or output terminal)

This is a terminal whose function changes according to mode. In "internal synchronous mode." this terminal is at high level, if sync characters are received and synchronized. If a status word is read, the terminal will be reset. In "external synchronous mode, "this is an input terminal. A "High" on this input forces the 8251 to start receiving data characters.

In "asynchronous mode," this is an output terminal which generates "high level "output upon the detection of a "break" character if receiver data contains a "low-level" space between the stop bits of two continuous characters. The terminal will be reset, if RXD is at high level. After Reset is active, the terminal will be output at low level.

DSR (Input terminal)

This is an input port for MODEM interface. The input status of the terminal can be recognized by the CPU reading status words.

DTR (Output terminal)

This is an output port for MODEM interface. It is possible to set the status of DTR by a command.

CTS (Input terminal)

This is an input terminal for MODEM interface which is used for controlling a transmit circuit. The terminal controls data transmission if the device is set in "TX Enable" status by a command. Data is transmittable if the terminal is at low level.

RTS (Output terminal)

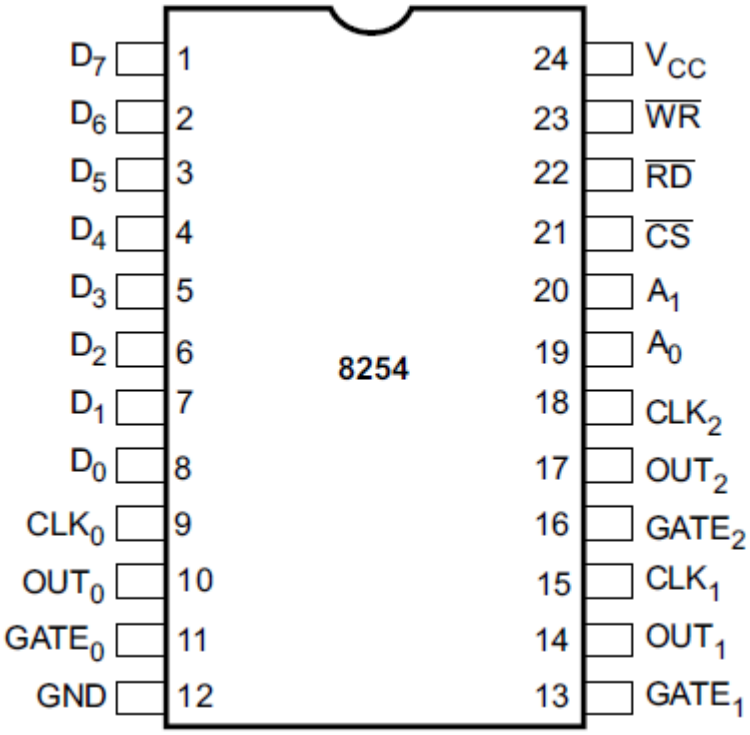
This is an output port for MODEM interface. It is possible to set the status RTS by a command.

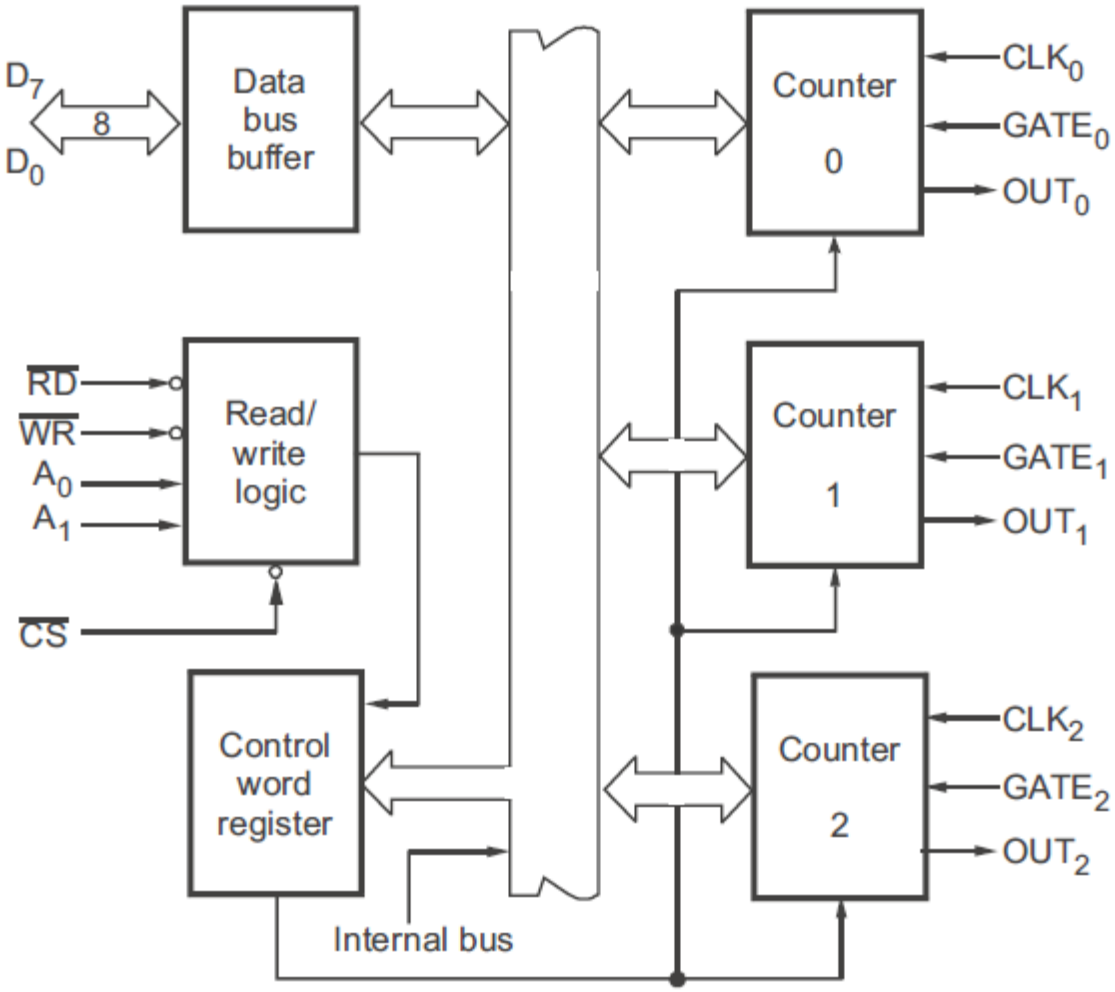
---

3. With neat block diagram explain the functions of 8254. (May/Jun 2016, Dec 2016)

PROGRAMMABLE INTERVAL TIMER 8253/54

- The 8253/54 includes three identical 16 bit counters that can operate independently.
- To operate a counter, a 16-bit count is loaded in its register and, on command; it begins to decrement the count until it reaches 0. At the end of the count, it generates a pulse that can be used to interrupt the CPU.
- In addition, a count can be read by the CPU while the counter is decrementing. The functioning of 8253 and 54 almost similar along with the pin configuration.


D7 1
D6 2
D5 3
D4 4
D3 5
D2 6
D1 7
D0 8
CLK0 9
OUT0 10
GATE0 11
GND 12
8254
24 VCC
23 WR
22 RD
21 CS
20 A1
19 A0
18 CLK2
17 OUT2
16 GATE2
15 CLK1
14 OUT1
13 GATE1


D7
8
D0
Data bus buffer
RD
WR
A0
A1
CS
Read/ write logic
Control word register
Internal bus
Counter 0
Counter 1
Counter 2
CLK0
GATE0
OUT0
CLK1
GATE1
OUT1
CLK2
GATE2
OUT2

The 8253/54 includes three counters, a data bus buffer, Read/Write control logic, and a control register. Each counter has two input signals CLOCK and GATE and one output signal OUT.

Data Bus Buffer:

This tri-state, bi-directional, 8-bit buffer is used to interface the 8253/54 to the system data bus. The Data bus buffer has three basic functions.

1. Programming the modes of 8253/54.
2. Loading the count registers.
3. Reading the count values.

Read/Write Logic: The Read/Write logic has five signals : RD, WR, CS and the address lines A0 and A1. In the peripheral I/O mode, the RD, and WR signals are connected to IOR and IOW, respectively. In memory-mapped I/O, these are connected to MEMR and MEMW. Address lines A0 and A1 of the CPU are usually connected to lines A0 and A1 of the 8253/54, and CS is tied to a decoded address. The control word register and counters are selected according to the signals on lines A0 and A1.

A1	A0	Selection
0	0	Counter 0
0	1	Counter 1
1	0	Counter 2
1	1	Control word Register

Control Word Register : This register is accessed when lines A0 and A1 are at logic 1. It is used to write a command word which specifies the counter to be used its mode, and either a read or write operation.
Counters: Three counters are there. The counters are fully independent. The programmer can read the contents of any of the three counters without disturbing the actual count in process. Each of the times has three pins associated with it. These are CLK (CLK) the gate (GATE) and the output (OUT).

CLK: This clock input pin provides 16-bit times with the signal to causes the times to decrement maximum clock input is 2.6MHz. The counters operate at the negative edge (H1 to L0) of this clock input. If the signal on this pin is generated by a fixed oscillator then the user has implemented a standard timer. If the input signal is a string of randomly occurring pulses, then it is called implementation of a counter.

GATE: The gate input pin is used to initiate or enable counting. The exact effect of the gate signal depends on which of the six modes of operation is chosen.

OUTPUT: The output pin provides an output from the timer. It actual use depends on the mode of operation of the timer. The counter can be read "in the fly" without inhibiting gate pulse or clock input.

4.3.2 MODES OF THE TIMER

Mode 0 : Interrupt on terminal count

a) Normal Operation :

1) The output will be initially low after the mode set operation.

2) After the count is loaded into the selected count Register the output will remain low and the counter will count.

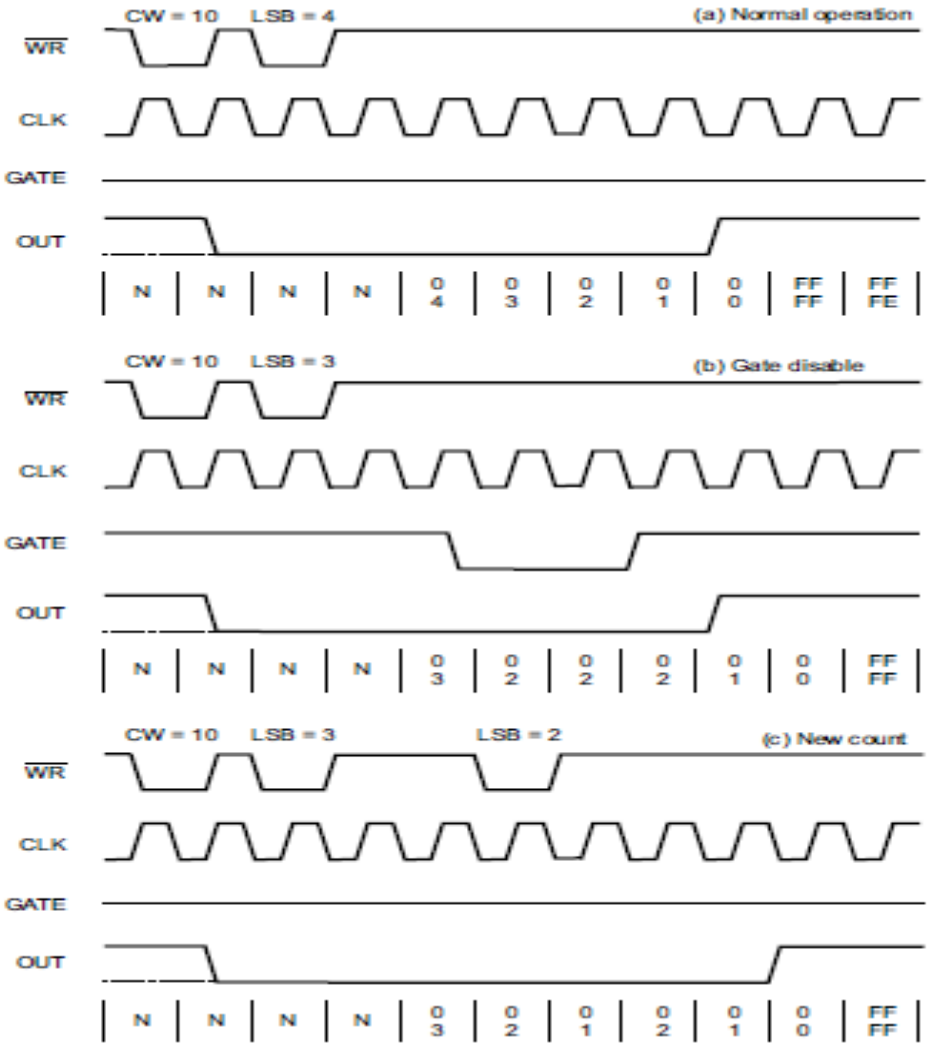
3) When the terminal count is reached the output will go high and remain high until the selected count is reloaded.

b) Gate Disable

1) Gate = 1 enables counting.

2) Gate = 0 disables counting.

Note : Gate has no effect on OUT.



MODE 1 : HARDWARE RETRIGGERABLE ONE-SHOT

a) Normal operation

1) The output will be initially high

2) The output will go low on the CLK pulse following the rising edge at the gate input.

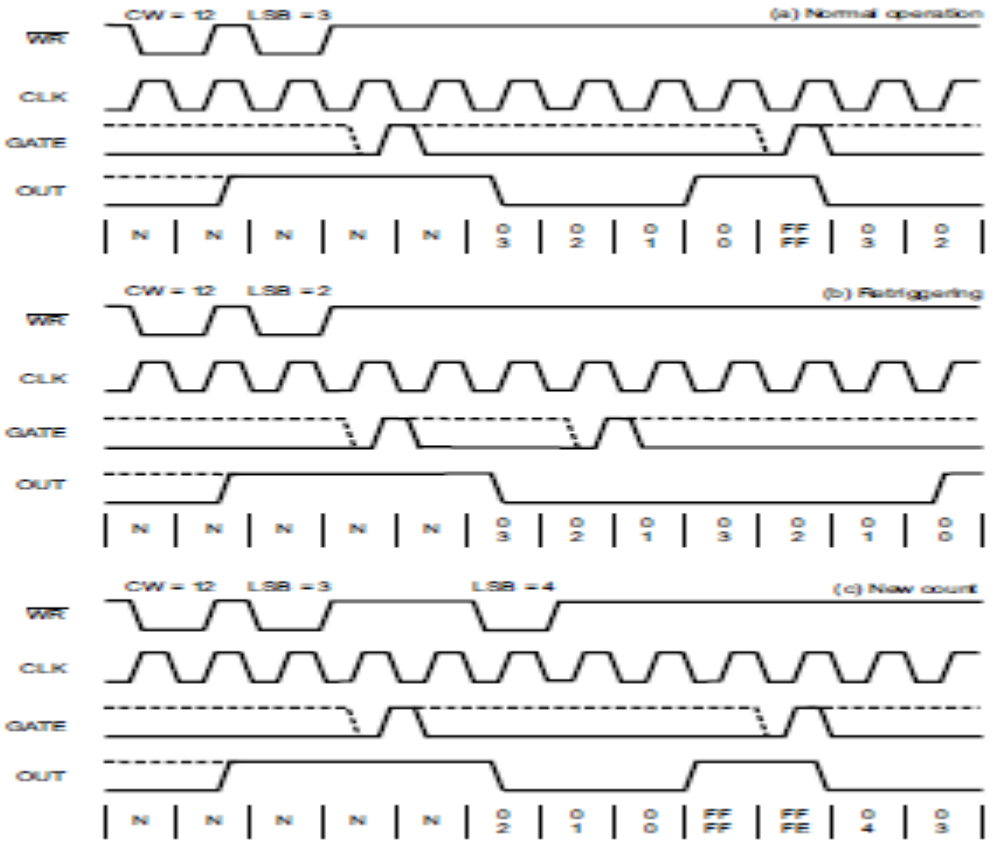
3) The output will go high on the terminal count and remain high until the next rising edge at the gate input.

b) Retriggering

The one shot is retriggerable, hence the output will remain low for the full count after any rising edge of the gate input.

c) New count

If the counter is loaded during one shot pulse, the current one shot is not affected unless the counter is retriggered. If retriggered, the counter is loaded with the new count and the one-shot pulse continues until the new count expires.



MODE 2 : RATE GENERATOR

This mode functions like a divide by-N counter.

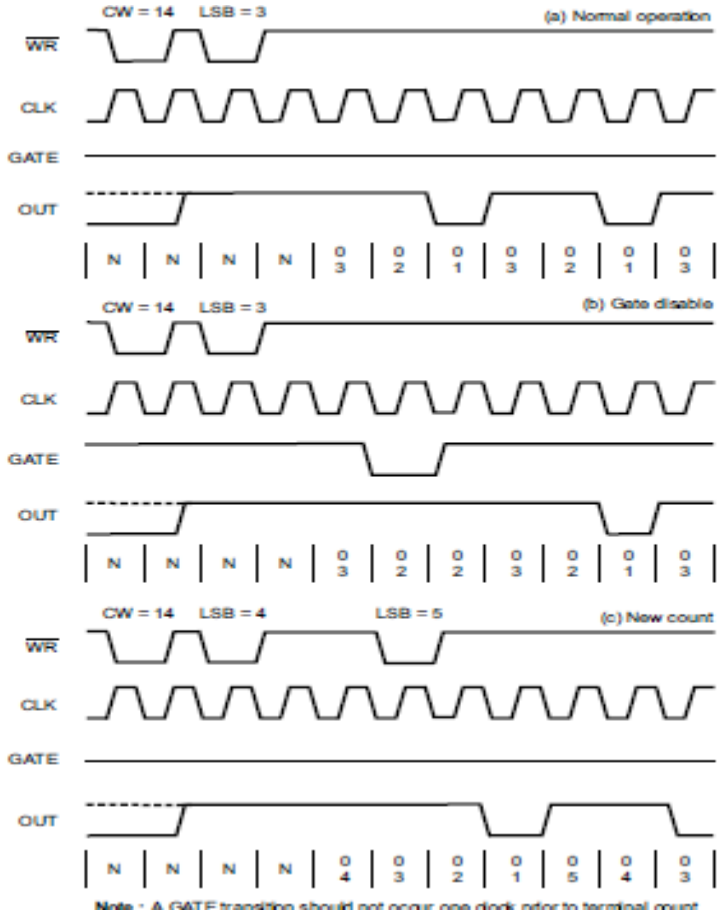
a) Normal Operation
1) The output will be initially high.
2) The output will go low for one clock pulse before the terminal count.
3) The output then goes high, the counter reloads the initial count and the process is repeated.
4) The period from one output pulse to the next equals the number of input counts in the count register.

b) Gate Disable
1) If Gate = 1 it enables a counting otherwise it disables counting (Gate = 0).
2) If Gate goes low during an low output pulse, output is set immediately high.
A trigger reloads the count and the normal sequence is repeated.

c) New count

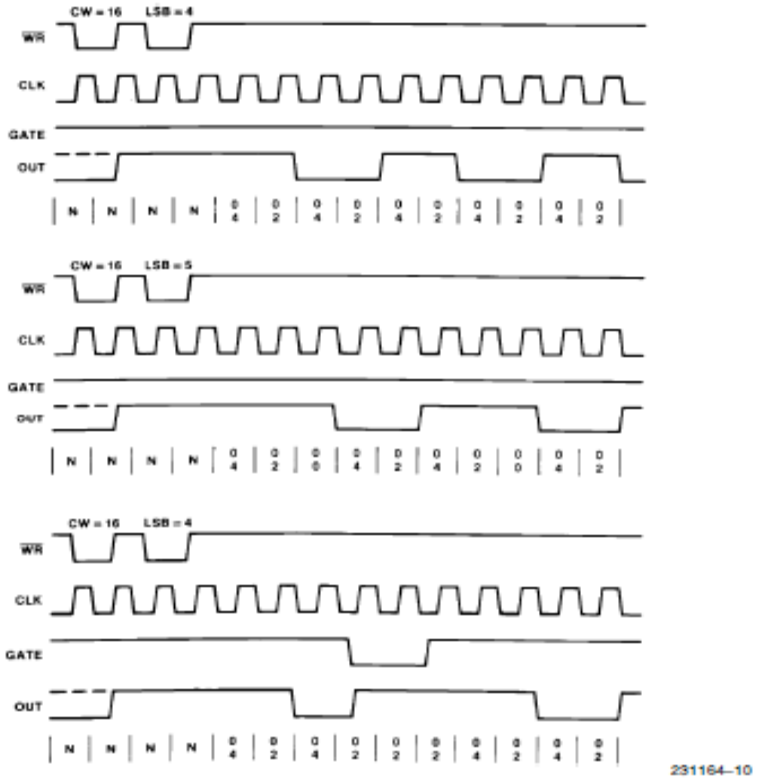
The current counting sequence does not affect when the new count is written. If a trigger is received after writing a new count but before the end of the current period, the new count will be loaded with the new count on the next CLK pulse and counting will continue from the new count. Otherwise, the new count will be loaded at the end of the current counting cycle.

Note : In mode 2, a count of 1 is illegal.



MODE 3: SQUARE WAVE RATE GENERATOR

Mode 3 is typically used for Baud rate generation. Mode 3 is similar to Mode 2 except for the duty cycle of OUT.



MODE 4: SOFTWARE TRIGGERED STROBE

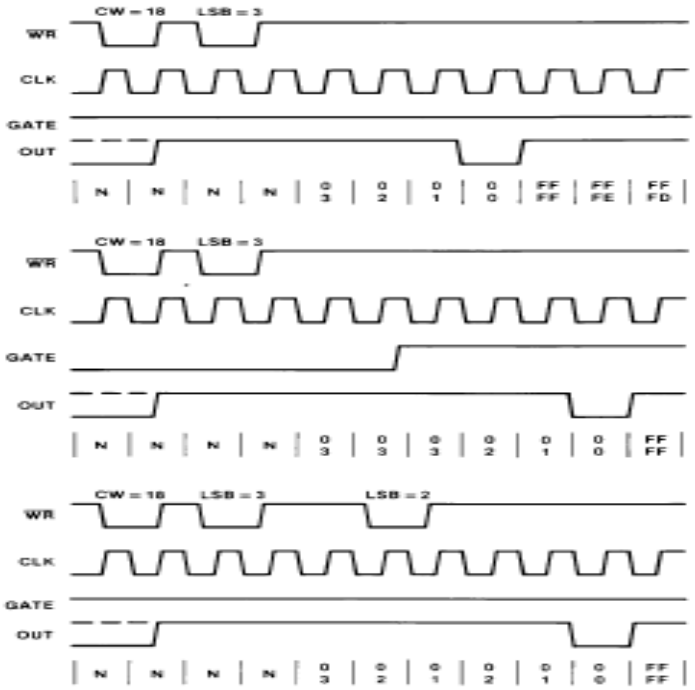
OUT will be initially high. When the initial count expires, OUT will go low for one CLK pulse and then go high again.

The counting sequence is ``triggered" by writing the initial count.

GATE has no effect on OUT.
After writing a Control Word and initial count, the Counter will be loaded on the next CLK pulse. This CLK pulse does not decrement the count, so for an initial count of N, OUT does not strobe low until a CLK pulses after the initial count is written.

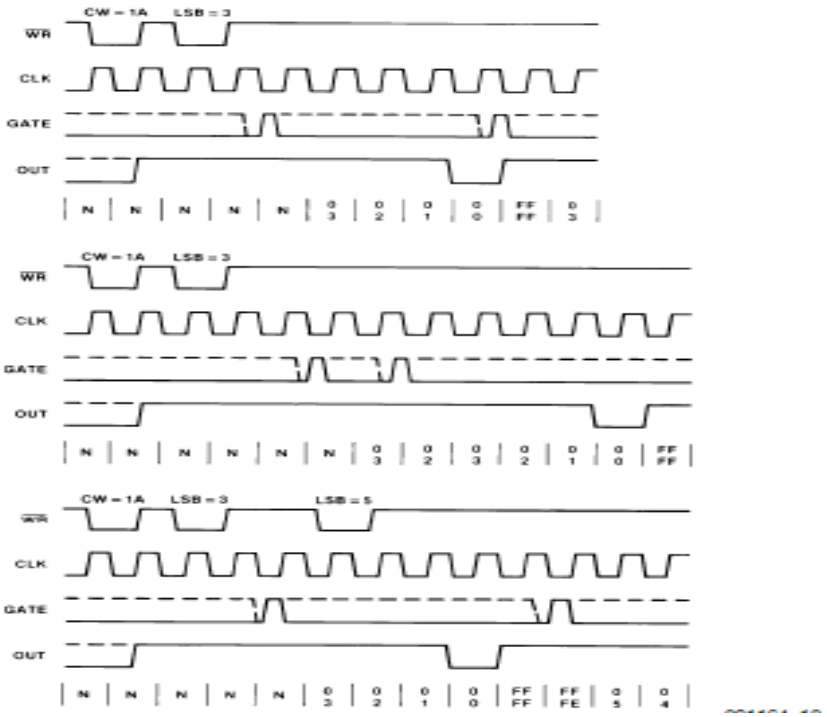
If a new count is written during counting, it will be loaded on the next CLK pulse and counting will continue from the new count. If a two-byte count is written, the following happens:
1) Writing the first byte has no effect on counting.
2) Writing the second byte allows the new count to be loaded on the next CLK pulse.

This allows the sequence to be ``retriggered" by software. OUT strobes low a CLK pulses after the new count of N is written.



MODE 5: HARDWARE TRIGGERED STROBE (RETRIGGERABLE)

OUT will initially be high. Counting is triggered by a rising edge of GATE. When the initial count has expired, OUT will go low for one CLK pulse and then go high again.



---

4. Explain the working of 8259 / 8259 A – (programmable interrupt controller) with neat block diagram (Dec 2016).

- The 8259A is a programmable interrupt controller designed to work with Intel microprocessor 8080 A, 8085, 8086 processors.
- The 8259 A interrupt controller can

1) Handle eight interrupt inputs. This is equivalent to providing eight interrupt pins on the processor in place of one INTR/INT pin.

2) Vector an interrupt request anywhere in the memory map. However, all the eight interrupt are spaced at the interval of either four or eight location. This eliminates the major drawback, 8085 interrupt, in which all interrupts are vectored to memory location on page 00H.

3) Resolve eight levels of interrupt priorities in a variety of modes.

4) Mask each interrupt request individually.

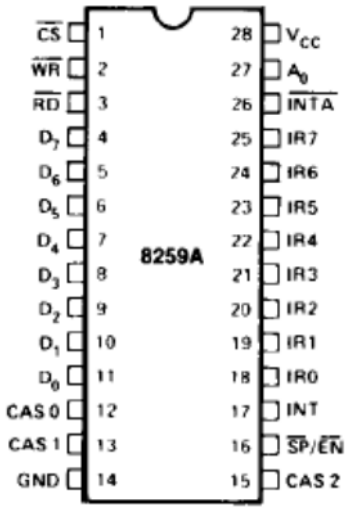
5) Read the status of pending interrupts, in service interrupts, and masked interrupts.

6) Be set up to accept either the level triggered or edge triggered interrupt request.

7) Mine 8259 as can be cascade in a master slave configuration to handle 64 interrupt inputs.

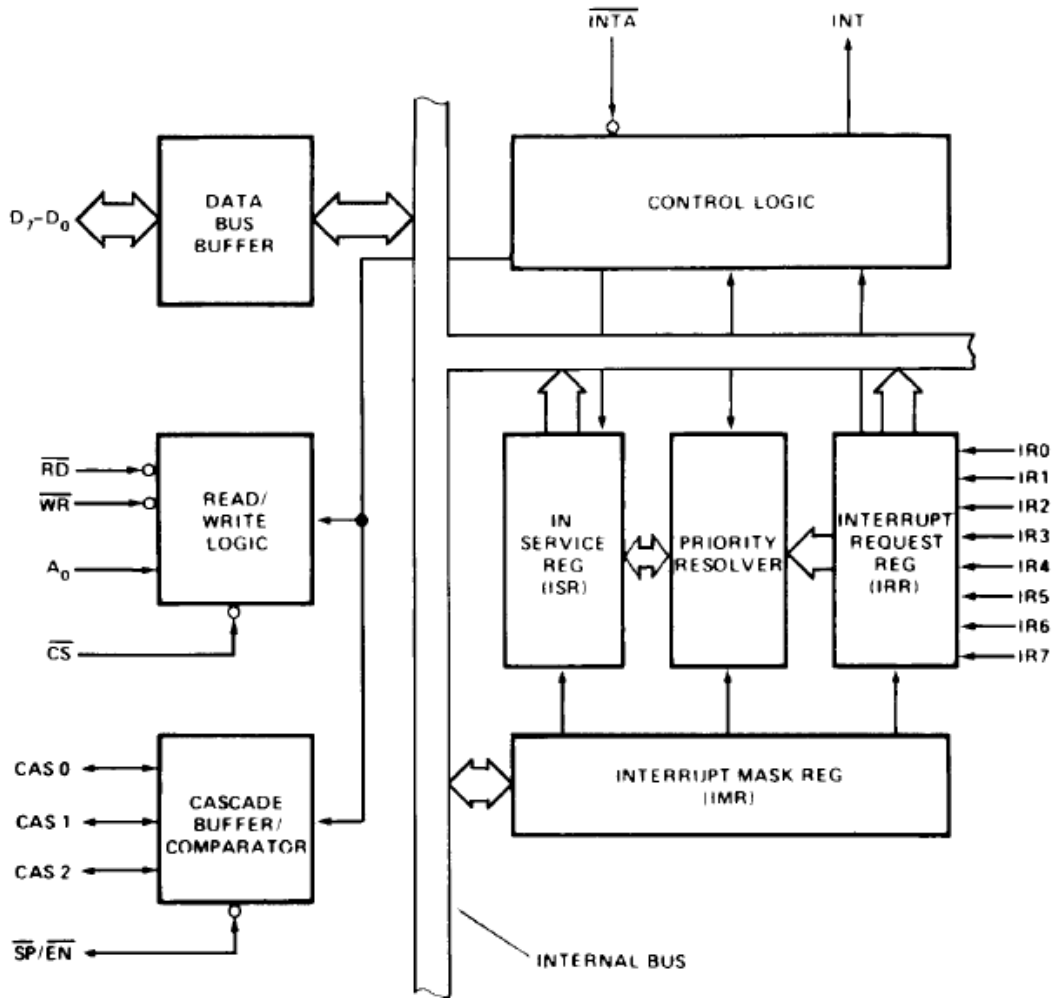
The 8259 A is contained in a 28-element in line package that requires only a compatible with 8259. The main difference between the two is that the 8259 A can be used with Intel 8086/8088 processor.

It also induces additional features such as level triggered mode, buffered mode and automatic end of interrupt mode. The pin diagram and block diagram is shown below:



PIN DESCRIPTION

Symbol	Pin No.	Type	Name and Function
V_{CC}	28	I	**SUPPLY:** +5V Supply.
GND	14	I	**GROUND**
$\overline{CS}$	1	I	**CHIP SELECT:** A low on this pin enables $\overline{RD}$ and $\overline{WR}$ communication between the CPU and the 8259A. INTA functions are independent of CS.
$\overline{WR}$	2	I	**WRITE:** A low on this pin when CS is low enables the 8259A to accept command words from the CPU.
$\overline{RD}$	3	I	**READ:** A low on this pin when CS is low enables the 8259A to release status onto the data bus for the CPU.
D_7-D_0	4–11	I/O	**BIDIRECTIONAL DATA BUS:** Control, status and interrupt-vector information is transferred via this bus.
CAS_0-CAS_2	12, 13, 15	I/O	**CASCADE LINES:** The CAS lines form a private 8259A bus to control a multiple 8259A structure. These pins are outputs for a master 8259A and inputs for a slave 8259A.
$\overline{SP}/\overline{EN}$	16	I/O	**SLAVE PROGRAM/ENABLE BUFFER:** This is a dual function pin. When in the Buffered Mode it can be used as an output to control buffer transceivers (EN). When not in the buffered mode it is used as an input to designate a master (SP = 1) or slave (SP = 0).
INT	17	O	**INTERRUPT:** This pin goes high whenever a valid interrupt request is asserted. It is used to interrupt the CPU, thus it is connected to the CPU's interrupt pin.
IR_0-IR_7	18–25	I	**INTERRUPT REQUESTS:** Asynchronous inputs. An interrupt request is executed by raising an IR input (low to high), and holding it high until it is acknowledged (Edge Triggered Mode), or just by a high level on an IR input (Level Triggered Mode).
$\overline{INTA}$	26	I	**INTERRUPT ACKNOWLEDGE:** This pin is used to enable 8259A interrupt-vector data onto the data bus by a sequence of interrupt acknowledge pulses issued by the CPU.
A_0	27	I	**AO ADDRESS LINE:** This pin acts in conjunction with the $\overline{CS}$, $\overline{WR}$, and $\overline{RD}$ pins. It is used by the 8259A to decipher various Command Words the CPU writes and status the CPU wishes to read. It is typically connected to the CPU A0 address line (A1 for 8086, 8088).



The descriptions of various blocks are,

Data bus buffer:
This 3- state, bidirectional 8-bit buffer is used to interface the 8259A to the system data bus. Control words and status information are transferred through the data bus buffer.

Read/Write & control logic:

The function of this block is to accept OUTPUT commands from the CPU. It contains the initialization command word (ICW) register and operation command word (OCW) register which store the various control formats for device operation. This function block also allows the status of 8159A to be transferred to the data bus.

Interrupt request register (IRR):
IRR stores all the interrupt inputs that are requesting service. Basically, it keeps track of which interrupt inputs are asking for service. If an interrupt input is unmasked, and has an interrupt signal on it, then the corresponding bit in the IRR will be set.

Interrupt mask register (IMR):

The IMR is used to disable (Mask) or enable (Unmask) individual interrupt inputs. Each bit in this register corresponds to the interrupt input with the same number. The IMR operation on the IRR. Masking of higher priority input will not affect the interrupt request lines of lower priority. To unmask any interrupt the corresponding bit is set '0'.

In service register (ISR):
The in service registers keeps tracks of which interrupt inputs are currently being serviced. For each input that is currently being serviced the corresponding bit will be set in the in service register. Each of these 3-reg can be read as status reg.

Priority Resolver:
This logic block determines the priorities of the set in the IRR. The highest priority is selected and strobe into the corresponding bit of the ISR during pulse.

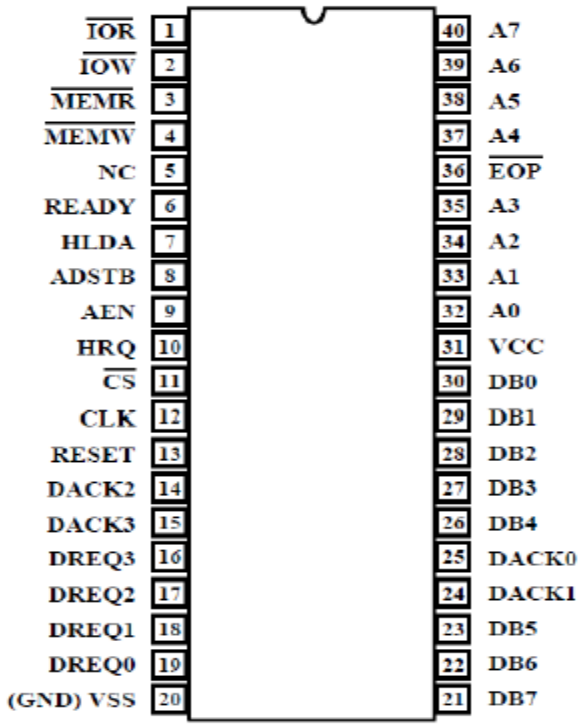
Cascade buffer/comparator:
This function blocks stores and compare the IDS of all 8259A's in the reg. The associated 3-I/O pins (CAS0-CAS2) are outputs when 8259A is used a master. Master and are inputs when 8259A is used as a slave. As a master, the 8259A sends the ID of the interrupting slave device onto the cas2-cas0. The slave thus selected will send its pre-programmed subroutine address on to the data bus during the next one or two successive pulses.

---

1. **Explain the operation of 8237 with neat block diagram. (May/June 2017)**

- The 8237 is a Direct Memory Access (DMA) Controller is a peripheral interface circuit for microprocessor systems. It is designed to improve system performance by allowing external devices to directly transfer information from the system memory.

PIN DIAGRAM

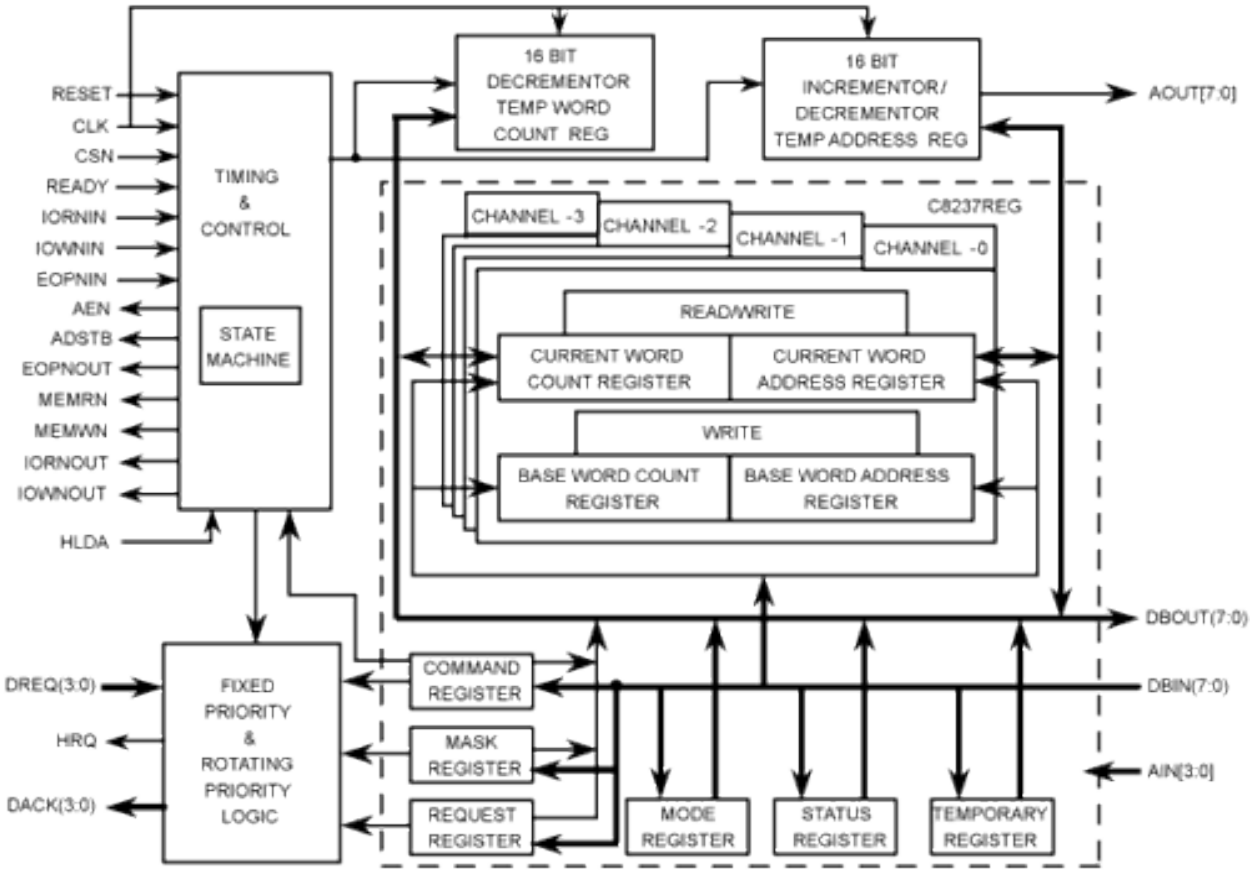


DETAILS

Symbol	Type	Name and Function
V_{CC}		**POWER:** +5V supply.
V_{SS}		**GROUND:** Ground.
CLK	I	**CLOCK INPUT:** Clock Input controls the internal operations of the 8237A and its rate of data transfers. The input may be driven at up to 5 MHz for the 8237A-5.
$\overline{CS}$	I	**CHIP SELECT:** Chip Select is an active low input used to select the 8237A as an I/O device during the Idle cycle. This allows CPU communication on the data bus.
RESET	I	**RESET:** Reset is an active high input which clears the Command, Status, Request and Temporary registers. It also clears the first/last flip/flop and sets the Mask register. Following a Reset the device is in the Idle cycle.
READY	I	**READY:** Ready is an input used to extend the memory read and write pulses from the 8237A to accommodate slow memories or I/O peripheral devices. Ready must not make transitions during its specified setup/hold time.
HLDA	I	**HOLD ACKNOWLEDGE:** The active high Hold Acknowledge from the CPU indicates that it has relinquished control of the system busses.
DREQ0–DREQ3	I	**DMA REQUEST:** The DMA Request lines are individual asynchronous channel request inputs used by peripheral circuits to obtain DMA service. In fixed Priority, DREQ0 has the highest priority and DREQ3 has the lowest priority. A request is generated by activating the DREQ line of a channel. DACK will acknowledge the recognition of DREQ signal. Polarity of DREQ is programmable. Reset initializes these lines to active high. DREQ must be maintained until the corresponding DACK goes active.
DB0–DB7	I/O	**DATA BUS:** The Data Bus lines are bidirectional three-state signals connected to the system data bus. The outputs are enabled in the Program condition during the I/O Read to output the contents of an Address register, a Status register, the Temporary register or a Word Count register to the CPU. The outputs are disabled and the inputs are read during an I/O Write cycle when the CPU is programming the 8237A control registers. During DMA cycles the most significant 8 bits of the address are output onto the data bus to be strobed into an external latch by ADSTB. In memory-to-memory operations, data from the memory comes into the 8237A on the data bus during the read-from-memory transfer. In the write-to-memory transfer, the data bus outputs place the data into the new memory location.
$\overline{IOR}$	I/O	**I/O READ:** I/O Read is a bidirectional active low three-state line. In the Idle cycle, it is an input control signal used by the CPU to read the control registers. In the Active cycle, it is an output control signal used by the 8237A to access data from a peripheral during a DMA Write transfer.
$\overline{IOW}$	I/O	**I/O WRITE:** I/O Write is a bidirectional active low three-state line. In the Idle cycle, it is an input control signal used by the CPU to load information into the 8237A. In the Active cycle, it is an output control signal used by the 8237A to load data to the peripheral during a DMA Read transfer.

FUNCTIONAL DESCRIPTION

The 8237 block diagram includes the major logic blocks and all of the internal registers. The data interconnection paths are also shown. Not shown are the various control signals between the blocks. The 8237A contains 344 bits of internal memory in the form of registers.



DMA CYCLES :

- The 8237 is designed to operate in two major cycles. These are called **Idle and Active cycles**. Each device cycle is made up of a number of states.
- The 8237A can assume seven separate states, each composed of one full clock period. The 8237A can assume seven separate states, each composed of one full clock period. State I (SI) is the inactive state.
- It is entered when the 8237A has no valid DMA requests pending. While in SI, the DMA controller is inactive but may be in the Program Condition, being programmed by the processor. StateS0 (S0) is the first state of a DMA service.
- The 8237A has requested a hold but the processor has not yet returned an acknowledge.
- The 8237A may still be programmed until it receives HLDA from the CPU. An acknowledge from the CPU will signal that DMA transfers may begin. S1, S2, S3 and S4 are the working states of the DMA service.
- If more time is needed to complete a transfer than is available with normal timing, wait states (SW) can be inserted between S2 or S3 and S4 by the use of the Ready line on the 8237A.
- Note that the data is transferred directly from the I/O device to memory (or vice versa) with IOR and MEMW (or MEMR and IOW) being active at the same time. The data is not read into or driven out of the 8237A in I/O-to-memory or memory-to-I/O DMA transfers. Memory-to-memory transfers require a read-from and a write-to-memory to complete each transfer.

- The states, which resemble the normal working states, use two digit numbers for identification. Eight states are required for a single transfer. The first four states (S11, S12, S13, S14) are used for the read from-memory half and the last four states (S21, S22,S23, S24) for the write-to-memory half of the transfer.

DMA CONTROLLER OPERATION STEPS IN A TYPICAL DMA CYCLE

Device wishing to perform DMA asserts the processors bus request signal.

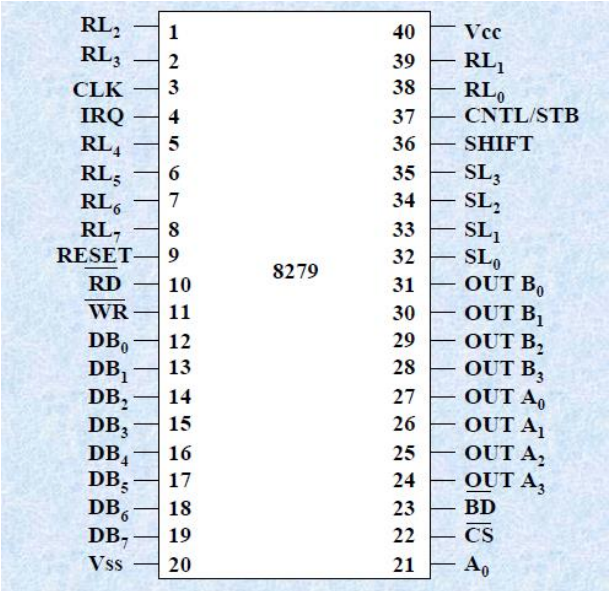
1. Processor completes the current bus cycle and then asserts the bus grant signal to the device.
2. The device then asserts the bus grant ack signal.
3. The processor senses in the change in the state of bus grant ack signal and starts listening to the data and address bus for DMA activity.
4. The DMA device performs the transfer from the source to destination address.
5. During these transfers, the processor monitors the addresses on the bus and checks if any location modified during DMA operations is cached in the processor. If the processor detects a cached address on the bus, it can take one of the two actions:

(i) Processor invalidates the internal cache entry for the address involved in DMA write operation
(ii) Processor updates the internal cache when a DMA write is detected.

6. Once the DMA operations have been completed, the device releases the bus by asserting the bus release signal.
7. Processor acknowledges the bus release and resumes its bus cycles from the point it left off.

---

6. Explain in detail about the 8279 keyboard and display controller. (May/Jun 2016, Dec 2016).



8279 Pin Configuration

PIN DESCRIPTION

Data Bus Lines, DB_0 - DB_7

These are 8 bidirectional data bus lines used to transfer the data to/from the CPU.

CLK

The clock input is used to generate internal timings required by the microprocessor.

RESET

As the name suggests this pin is used to reset the microprocessor.

CS Chip Select

When this pin is set to low, it allows read/write operations, else this pin should be set to high.

A_0

This pin indicates the transfer of command/status information. When it is low, it indicates the transfer of data.

RD, WR

This Read/Write pin enables the data buffer to send/receive data over the data bus.

IRQ

This interrupt output line goes high when there is data in the FIFO sensor RAM. The interrupt line goes low with each FIFO RAM read operation. However, if the FIFO RAM further contains any key-code entry to be read by the CPU, this pin again goes high to generate an interrupt to the CPU.

V_{ss}, V_{cc}

These are the ground and power supply lines of the microprocessor.

$SL_0 - SL_3$

These are the scan lines used to scan the keyboard matrix and display the digits. These lines can be programmed as encoded or decoded, using the mode control register.

$RL_0 - RL_7$

These are the Return Lines which are connected to one terminal of keys, while the other terminal of the keys is connected to the decoded scan lines. These lines are set to 0 when any key is pressed.

SHIFT

The Shift input line status is stored along with every key code in FIFO in the scanned keyboard mode. Till it is pulled low with a key closure, it is pulled up internally to keep it high

CNTL/STB - CONTROL/STROBED I/P Mode

In the keyboard mode, this line is used as a control input and stored in FIFO on a key closure. The line is a strobe line that enters the data into FIFO RAM, in the strobed input mode. It has an internal pull up. The line is pulled down with a key closure.

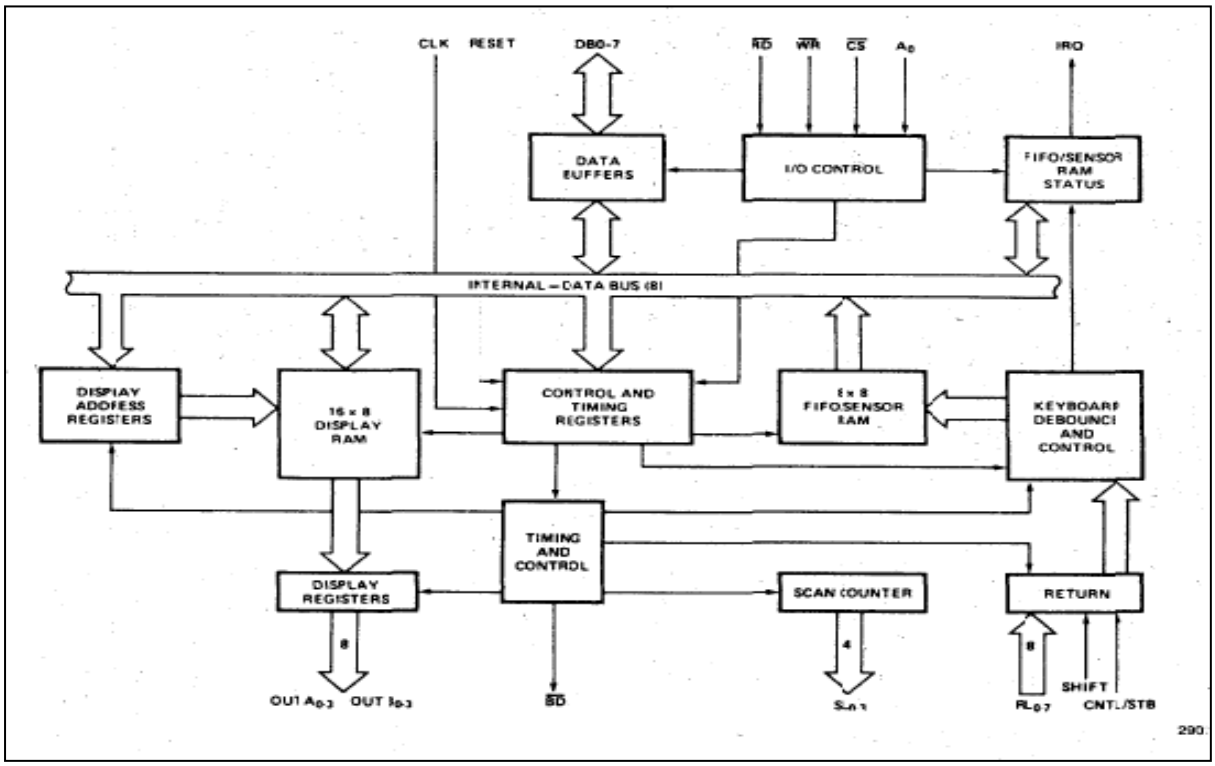
BD

It stands for blank display. It is used to blank the display during digit switching.

$OUTA_0$ – $OUTA_3$ and $OUTB_0$ – $OUTB_3$

These are the output ports for two 16x4 or one 16x8 internal display refresh registers. The data from these lines is synchronized with the scan lines to scan the display and the keyboard.

BLOCK DIAGRAM



- It provides 3 operating modes

1.Scanned keyboard mode
2.Scanned sensor matrix mode
3.Strobed Input mode.

- It has inbuilt debounce key .

- It consists 4 main section.

1. **CPU interface and control section.**
2. **Scan section**
3. **Keyboard Section**
4. **Display section**

CPU INTERFACE AND CONTROL SECTION:

It consists of

1. Data buffers
2. I/O control
3. Control and timing registers.
4. Timing and control logic.

Data Buffers:

- 8-bit bidirectional buffer.
- Used to connect the internal data bus and external data bus.

I/O control:

- I/O control section uses the A0,CS,RD and WR signals to controls the data flow.
- The data flow is enabled by CS=0otherwise it is the high impedance state.
- A0=0 means the data is transferred.
- A0=1 means status or command word is transferred.

I/O control signals

The I/O control signals are listed below

A0	RD	WR	Interpretation
0	1	0	Data from CPU to 8279
0	0	1	Data from 8279 to CPU
1	1	0	Command word from CPU to 8279

1	0	1	Status word from 8279 to CPU

- **DB_0-DB_7 :** These are bidirectional data bus lines. The data and command words to and from the CPU are transferred on these lines.

- **$\overline{RD}$, $\overline{WR}$ (Input / Output) READ/WRITE**
  These input pins enable the data buffers to receive or send data over the data bus.

- **A0(Address lines) :** A high on this line indicates the transfer of a command or status information. A low on this line indicates the transfer of data. This is used to select one of the internal registers of 8279.

- **$\overline{CS}$:** Chip Select – A low on this line enables 8279 for normal read or write operations. Other wise, this pin should remain high

- **RESET :** This pin is used to reset 8279. A high on this line reset 8279. After resetting 8279, its in sixteen 8-bit display, left entry encoded scan, 2-key lock out mode. The clock prescaler is set to 31.

- **CLK :** This is a clock input used to generate internal timing required by 8279.

- **IRQ :** This interrupt output lines goes high when there is a data in the FIFO sensor RAM. The interrupt lines goes low with each FIFO RAM read operation but if the FIFO RAM further contains any key-code entry to be read by the CPU, this pin again goes high to generate an interrupt to the CPU.

- **Vss, Vcc :** These are the ground and power supply lines for the circuit.

SCAN

- **SL0-SL3-Scan Lines :** These lines are used to scan the key board matrix and display digits. These lines can be programmed as encoded or decoded, using the mode control register.
- **RL0 - RL7 - Return Lines :** These are the input lines which are connected to one terminal of keys, while the other terminal of the keys are connected to the decoded scan lines. These are normally high, but pulled low when a key is pressed.

- **SHIFT :** The status of the shift input lines is stored along with each key code in FIFO, in scanned keyboard mode. It is pulled up internally to keep it high, till it is pulled low with a key closure.

KEYBOARD MODE CONTROL

- **CNTL/STB- CONTROL/STROBED I/P Mode :** In keyboard mode, this lines is used as a control input and stored in FIFO on a key closure. The line is a strobed lines that enters the data into FIFO RAM, in strobed input mode. It has an interrupt pull up. The lines is pulled down with a key closer.

DISPLAY REGISTERS & BLANK THE DISPLAY

- **OUT A0 – OUT A3 and OUT B0 – OUT B3 :** These are the output ports for two 16*4 or 16*8 internal display refresh registers. The data from these lines is synchronized with the scan lines to scan the display and keyboard. The two 4-bit ports may also as one 8-bit port.

- **BD – Blank Display:** This output pin is used to blank the display during digit switching or by a blanking closure.

UNIT 5

MICRO CONTROLLER PROGRAMMING & APPLICATIONS

PART B

1. Explain the various instruction set used in 8051 with examples.

The 8051 have in total of **255 instructions**. These instructions perform only 53 operations (addition, subtraction, copy etc.)

Depending on operation they perform, all instructions are divided in several groups:

- **Data Transfer Instructions**
- **Arithmetic Instructions**
- **Logical Instructions & Bit-oriented Instructions**
- **Branch Instructions**

5.1.1 DATA TRANSFER INSTRUCTIONS

Data transfer instructions move the content of one register to another. The register the content of which is moved remains unchanged. If they have the suffix "X" (MOVX), the data is exchanged with external memory.

- MOV dest, source dest ← source
- **Stack instructions**

PUSH byte ; increment stack pointer, ;move byte on stack
POP byte ; move from stack to byte, ; decrement stack pointer

- **Exchange instructions**

XCH a, byte ;exchange accumulator and byte
XCHD a, byte ;exchange low nibbles of
;accumulator and byte

5.1.2 ARITHMETIC INSTRUCTIONS

Add
Subtract
Increment
Decrement
Multiply
Divide
Decimal adjust

Mnemonic	Description
ADD A, byte	add A to byte, put result in A
ADDC A, byte	add with carry
SUBB A, byte	subtract with borrow
INC A	increment A
INC byte	increment byte in memory
INC DPTR	increment data pointer
DEC A	decrement accumulator
DEC byte	decrement byte
MUL AB	multiply accumulator by b register
DIV AB	divide accumulator by b register
DA A	decimal adjust the accumulator

INC A	increment A
INC byte	increment byte in memory
INC DPTR	increment data pointer
DEC A	decrement accumulator
DEC byte	decrement byte

- The increment and decrement instructions do NOT affect the C flag.

- We can only INCREMENT the data pointer, not decrement.

Multiply

When multiplying two 8-bit numbers, the size of the maximum product is 16-bits

MUL AB ; A * B
Note : The result is stored in B & A reg. B gets the High byte A gets the Low byte

Division

DIV AB ; divide A by B

A = Quotient(A/B)
B = Remainder(A/B)

OV - used to indicate a divide by zero condition.
C – set to zero

5.1.3. LOGICAL INSTRUCTIONS

Bitwise logic operations (AND, OR, XOR, NOT)
Clear

Rotate

Swap

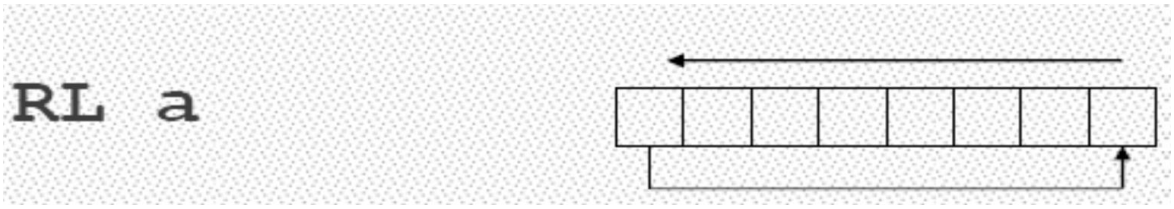
Bitwise logical operations

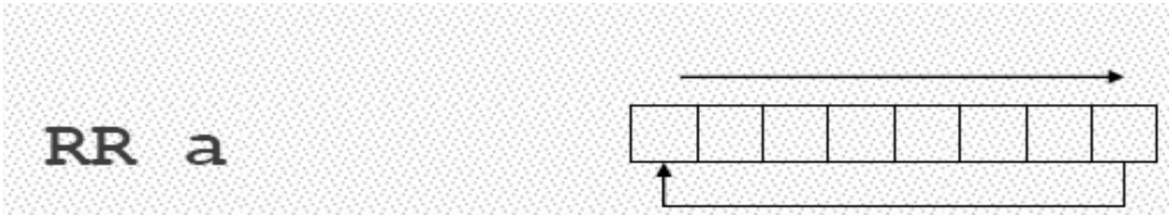
ANL (AND)
ORL (OR)
XRL (XOR)
CPL (Complement)

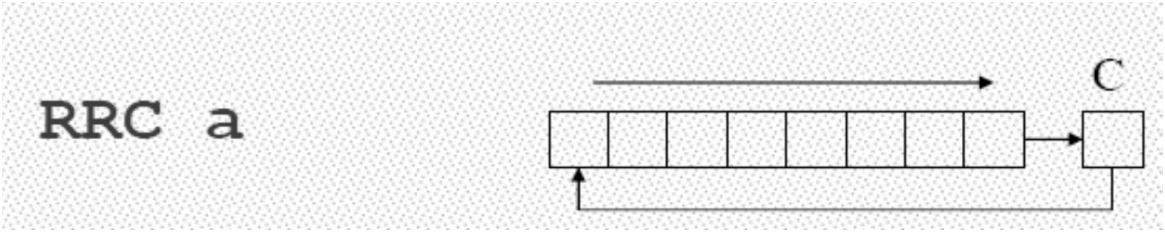
OTHER LOGICAL INSTRUCTIONS

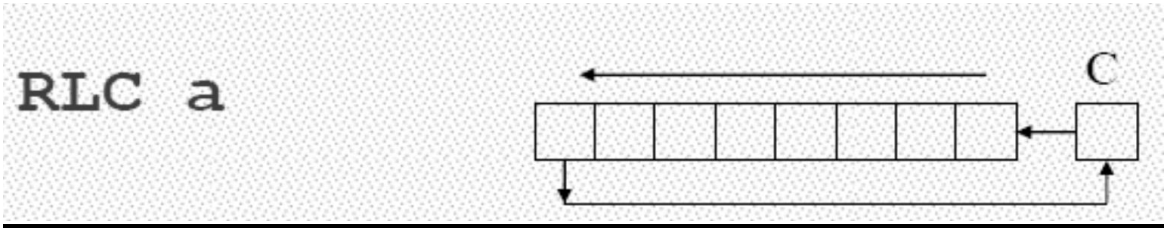
CLR - clear
RL – rotate left
RLC – rotate left through Carry
RR – rotate right
RRC – rotate right through Carry
SWAP – swap accumulator nibbles

ROTATE









5.1.4 PROGRAM FLOW CONTROL (BRANCH INSTRUCTIONS)

- **Unconditional jumps ("go to")**
- **Conditional jumps**
- **Call and return**

Unconditional Jump

SJMP <rel addr> ; Short jump

LJMP <address> ; Long jump

AJMP <address> ; Absolute jump to anywhere within 2K block of program memory

`JMP @A + DPTR ;` `Long indexed jump`

Conditional Jump

These instructions cause a jump to occur only if a condition is true. Otherwise, program execution continues with the next instruction.

Mnemonic
JZ <rel addr>
JNZ <rel addr>
JC <rel addr>
JNC <rel addr>
JB <bit>, <rel addr>
JNB <bit>,<rel addr>
JBC <bir>, <rel addr>
CJNE A, direct, <rel addr>

Call and Return

Call is similar to a jump, but Call pushes PC on stack before branching

acall <address > ; PC to stack

lcall <address > ; PC to stack

Return

Return is also similar to a jump, but

Return instruction pops PC from stack to get address to jump to

ret ; stack to PC

---

2. Explain the various addressing modes of 8051 with examples.

5.2.1 IMMEDIATE MODE

This mode specify data by its value

MOV A, #0

MOV R4, #10 H

MOV B, # 07

MOV DPTR,# 2500 H

5.2.2 REGISTER ADDRESSING

Either source or destination is one of register

```
 MOV R0,A
MOV A,R7
ADD A,R7
MOV DPTR,#25F5H
MOV R5,DPL
```

5.2.3DIRECT ADDRESSING MODE

Specify data by its 8-bit address

```
 MOV A, 70H          ; copy contents of RAM at 70h to a
MOV R0,40H       ; copy contents of RAM at 70h to a
```

5.2.4.REGISTER INDIRECT

The address of the source or destination is specified in registers. Uses registers R0 or R1 for 8-bit address:

```
mov psw, #0          ; use register bank 0
mov r0, #0x3C
mov @r0, #3
```

Uses DPTR register for 16-bit addresses:

```
MOV DPTR, #0X9000        ; DPTR ← 9000HS
MOVX A, @DPTR            ; A ← M[9000]
```

9000 is an address in external memory

5.2.5 REGISTER INDEXED MODE

Source or destination address is the sum of the base address and the accumulator (Index).**Base address can be DPTR or PC**

MOV DPTR, #4000H

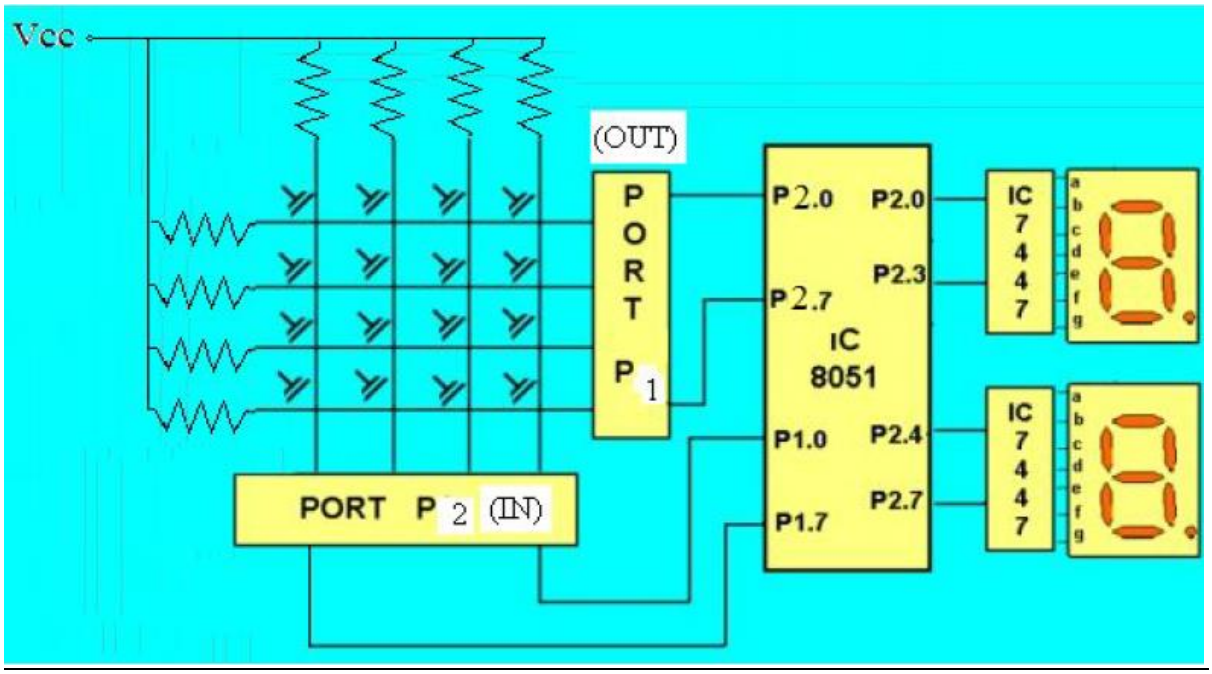
---

3. (a) Write a program to interface a LCD display to 8051 microcontroller (Dec 2016,May 2016)

(b) **Write a program to interface a Keyboard to 8051 microcontroller.**

3.1 KEY BOARD INTERFACING

The key board is designed with a particular rows and columns (matrix). These rows and columns are connected to the microcontroller through its ports of the micro controller 8051.



- First make sure that the preceding key has been released, 0s are output to all rows at once, and the columns are read and checked repeatedly until all the columns are high.
- When all columns are found to be high, the program waits for a short amount of time before it goes to the next stage of waiting for a key to be pressed.
- Columns re scanned over and over in an infinite loop until one of them has a 0 on it. Rows are 0. After a key is pressed mp wait for 20ms and then scans the column again

- To detect which rows the key press belongs to , the mp grounds one row at a time, if it finds that all columns are high, this means that the key press cannot belong to that row, therefore, it grounds the next row and continues.

- After finding the key, it sets up the starting address for the lookup table holding the scan codes To identify the key pressed the mp controller rotates the column bits one bit at time, in to carry flag and checks to see if its low.

- If zero, it pull the ASCII code for that key from the look-up table. Otherwise it increment the pointer to point the next element of the look-up table.

- The program for key board interfacing is given below.

PROGRAM

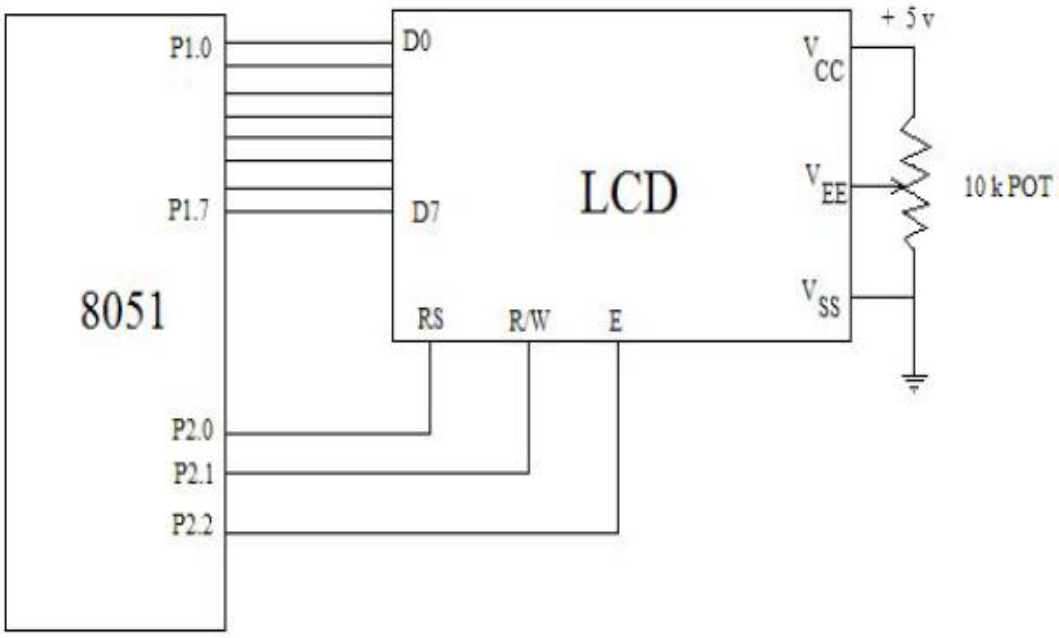
(P1.0 – P1.3 are connected to rows P2.0 –P2.3 connected to columns)

```
      MOV P2, #0FFH
K1: MOV P1, #0
      MOV A, P2
      ANI A, #00001111
      CJNE A, #00001111, K1
K2:  ACALL DELAY
       MOV A, P2
       ANI A, #00001111
      CJNE A,#00001111, OVER
      SJMP K2

OVER1: MOV P1, #11111110
           MOV A,P2
           ANI A, #00001111
           CJNE A,#00001111, ROW0
           MOV P1, #11111101
            MOV A,P2
          ANI A, #00001111
          CJNE A,#00001111, ROW1
         MOV P1, #11111011
          MOV A,P2
         ANI A, #00001111
        MOV P1, #11110111
        MOV A,P2
       ANI A, #00001111
       CJNE A,#00001111, ROW3
        LJMP K2
ROW0: MOV DPTR,#KCODE0
SJMP FIND
ROW1: MOV DPTR,#KCODE1
SJMP FIND
ROW2: MOV DPTR,#KCODE2
SJMP FIND
ROW3: MOV DPTR,#KCODE3
FIND: RRC A
JNC MATCH
```

```
INC DPTR
SJMP FIND
MATCH: CLR A
MOVC A,@A+DPTR
MOV P0,A
LJMP K1
```

```
Look-up table
Org 300h
Kcode0: DB '0' , '1', '2', '3'
Kcode1: DB '4' , '5', '6', '7'
Kcode2: DB '8' , '9', 'A', 'B'
Kcode3: DB 'C' , 'D', 'E', 'F'
End
```

3.2 LCD (DISPLAY) INTERFACING



The following setup is maintained to interface a LCD display with 8051 MC.

- To send command to LCD RS= 0, for data RS=1

- P1.0 to P1.7 – are connected to LCD data/command
- P2.0 is connected to RS pin of LCD
- P2.1 is connected to R/W pin of LCD
- P2.2 is connected to E pin of LCD

PROGRAM

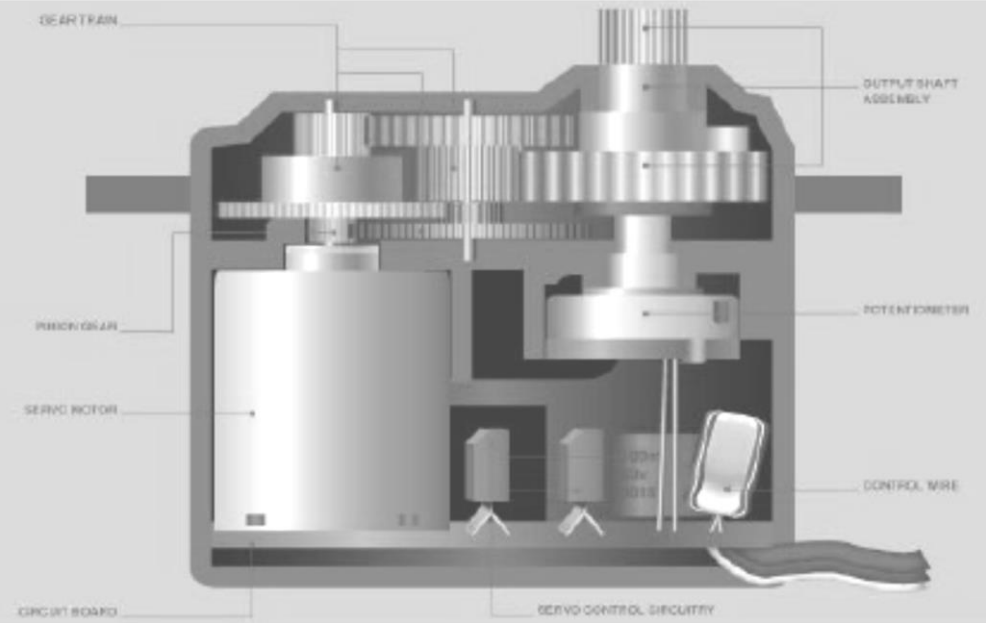
```
MOV A, # 38H ;                  init. Lcd 2 line, 5*7 matrix
ACALL COMMAND
MOV A,# 0EH ;                   display on cursor on
ACALL COMMAND
MOV A,#01H ;                    clear LCD
ACALL COMMAND
MOV A,#06H ;                    shift cursor right
ACALL COMMAND
MOV A,#84H ;                    cursor at line 1, position 4
ACALL COMMAND
MOV A, # 'N' ;                  display letter N
ACALL DATAWRT
MOV A,# 'O' ;                   display letter O
ACALL DATAWRT
HERE: SJMP HERE

COMMAND: ACALL READY
MOV P1,A
CLR P2.0 ;                      RS =0 for command
CLR P2.1 ;                      R/W =0 for write
SETB P2.2 ;                     E=1 for high pluse
CLR P2.2 ;                      E=0
RET

DATAWRT:

ACALL READY
MOV P1,A
SETB P2.0 ;                     RS =1 for data
CLR P2.1 ;                      R/W =0 for write
SETB P2.2 ;                     E=1 for high pluse
ACALL DELAY
CLR P2.2 ;                      E=0
RET

READY: SETB p1.7   ;            Make p1.7 as input port
CLR p2.0           ;            Access command register
SETB p2.1          ;            Read command register (R/W =1)

BACK : CLR p2.2
      ACALL DELAY
      SETB p2.2
      JB p1.7, BACK
       RET
        END
```

---

4. Draw the schematic for interfacing a servo motor with 8051 microcontroller and write 8051 ALP for servo motor control. (May/Jun 2016) (May/Jun 2017)

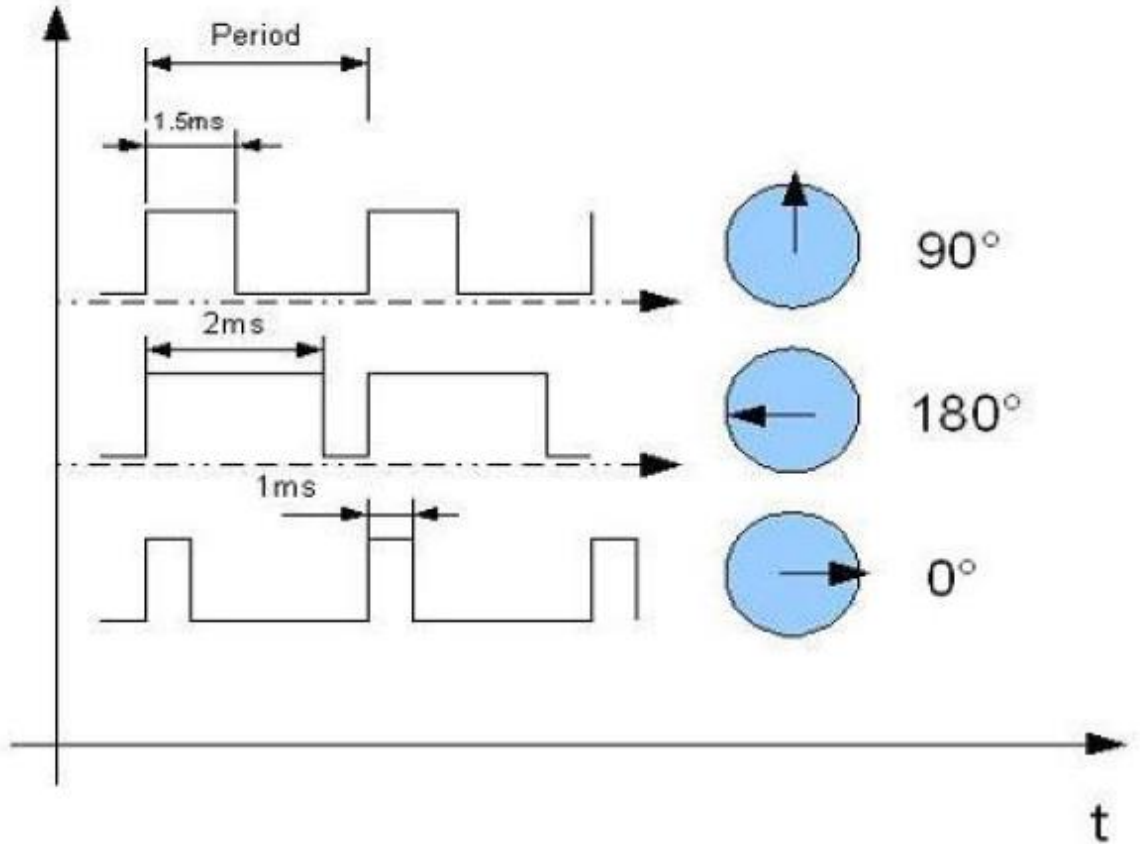
- Servo motors are so called "closed feedback" systems.

- This means that motor comes with control circuit, which senses if motor mechanism is in desired location and if not it continuously corrects an error until motor reaches proper point.
- Servo motors are widely used in robotics, remote controlled planes, and vehicles.
- The various parts of a servo motor is shown in the following Figure.



SERVO CONTROL SIGNALS

- Servo motor shaft is positioned with pulse width modulated signals. So all servos comes with three wires (Power, Ground and Control). So pulses are sent via control wire.
- Usually in servos with rotation angle 90° signal width vary between 1 and 2ms. If pulse is more wide rotation continues until reaches mechanical limits. This is shown in the below Figure.



MAIN PROGRAM

```
ORG 0000H
MAIN: MOV A,P0
MOV R1,A
ANL A,#01H
XRL A,#01H
JZ M1
MOV A,R1
ANL A,#02H
XRL A,#02H
JZ M1.5
MOV A,R1
ANL A,#04H
XRL A,#04H
JZ M2
AJMP  MAIN
```

SUB

1 ms DELAY

```
M1: MOV R6, #01H ; LOAD 10D IN R6
ACALL DELAY ; CALL 1 MS DELAY
CLR P2.0 ; SEND 0 TO PORT PIN
```

```
MOV R6, #01H ; LOAD 1D IN R6
ACALL DELAY ; CALL 1 MS DELAY
DELAY:
LP2: MOV R7, #0FAH
LP1: NOP ; 1 CYCLE
NOP ; 1+1=2 CYCLES
DJNZ R7, LP1 ; 1+1+2 = 4 CYCLES
DJNZ R6, LP2 ; 4×250 = 1000 CYCLES = 1000 MS = 1 MS
 RET
```

1.5 ms Delay

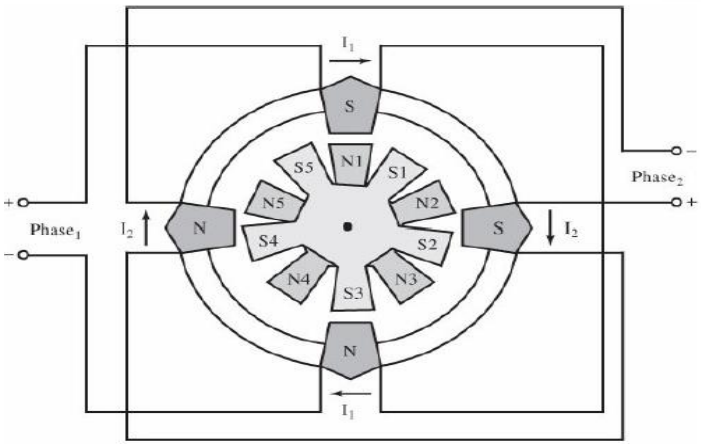
```
M1.5: MOV R6, #02H ; LOAD 10D IN R6
ACALL DELAY ; CALL 1 MS DELAY
CLR P2.0 ; SEND 0 TO PORT PIN
MOV R6, #02H ; LOAD 1D IN R6
ACALL DELAY ; CALL 1 MS DELAY
DELAY:
LP2: MOV R7, #0FAH
LP1: NOP ; 1 CYCLE
NOP ; 1+1=2 CYCLES
DJNZ R7, LP1 ; 1+1+2 = 4 CYCLES
DJNZ R6, LP2 ; 4×375 = 1000 CYCLES = = 1.5 MS
LP3: MOV R7, #07DH
LP4: NOP ; 1 CYCLE
NOP ; 1+1=2 CYCLES
DJNZ R7, LP4 ; 1+1+2 = 4 CYCLES
DJNZ R6, LP3 ; 4×125 = 500 CYCLES = 500 MS = .5 MS
RET
```

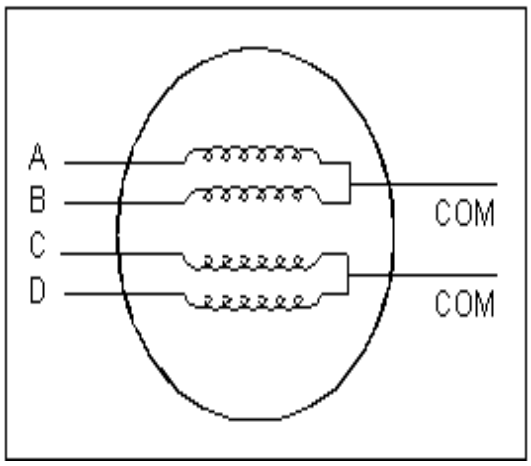
2 ms delay

```
M2: MOV R6, #02H ; LOAD 10D IN R6
ACALL DELAY ; CALL 1 MS DELAY
CLR P2.0 ; SEND 0 TO PORT PIN
MOV R6, #02H ; LOAD 1D IN R6
ACALL DELAY ; CALL 1 MS DELAY
DELAY:
LP2: MOV R7, #0FAH
LP1: NOP ; 1 CYCLE
NOP ; 1+1=2 CYCLES
DJNZ R7, LP1 ; 1+1+2 = 4 CYCLES
DJNZ R6, LP2 ; 4×250 = 1000 CYCLES = 1000 MS = 1 MS
RET
```

5. Draw the schematic for interfacing a stepper motor with 8051 microcontroller and write 8051 ALP for changing speed and direction of motor. (Nov/Dec 2015, May/Jun 2016) (May/Jun 2017)

- **A stepper motor is a special type of electric motor that moves in increments, or steps, rather than turning smoothly as a conventional motor does.**
- This Motor Moves Each Time a Pulse is Received Can Control Movement (Direction and Amount) Easily. It Can Force Motor to Hold Position Against an Opposing Force. The conceptual diagram of stepper motor is shown below.





Step Angle

The step angle is defined as

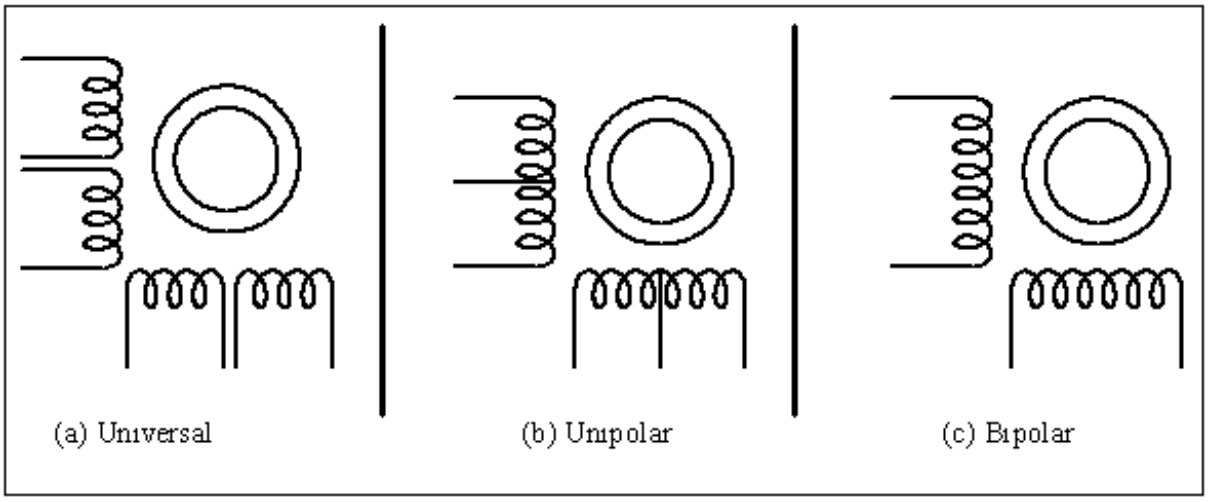
Step angle = 360/No. of Steps per Revolution

- Note : Commonly available no. of steps per revolution are 500, 200, 180, 144, 72, 48, 24

Revolutions per Minute (RPM)
RPM is defined as

$$rpm = \frac{60 \times Steps\ per\ Second}{Steps\ per\ \mathrm{Re}volution}$$

Common Stepper Motor Types



Drivers

May Need a Driver Circuit. The Reason for driver circuit is, The power consumption range is different for MP and stepper motor. The transition drivers are normally used (Usually a Darlington Pair).

PROGRAM (ANTI CLOCKWISE ROTATION)

```
MOV A, # 66
BACK : MOV P1, A
RL A
ACALL DELAY
SJMP BACK
DELAY
MOV R2, #100
L1: MOV R3, #255
L2: DJNZ R3, L2
DJNZ R2, L1
```

PROGRAM (CLOCKWISE ROTATION)

```
MOV A, # 66
BACK : MOV P1, A
```

```
RR A
ACALL DELAY
SJMP BACK
DELAY
MOV R2, #100
L1: MOV R3, #255
L2: DJNZ R3, L2
DJNZ R2, L1
```

---

6. Explain in detail about washing machine control using 8051.(Nov/Dec 2015, Dec 2016)

OPERATIONS OF A WASHING MACHINE

Fill:- water will be filled by the pump as per the load knob selected.

Agitate:- The wash basket will rotate in a clockwise direction for 10 revolutions, After that basket will stop for 2 seconds, then rotate 10 revolutions in anticlockwise direction. The process will be continued for specified minutes in cycle table.

Drain:- After agitation, the water and detergent are drained.

Spin:- During spin, agitator will be stationary, only the basket will rotate at high speed. Then the moisture of clothes are removed through holes in the inner metallic basket.

Indicator:- Machine ON : LED ON

After completion of washing cycle, buzzer sound will be generated.

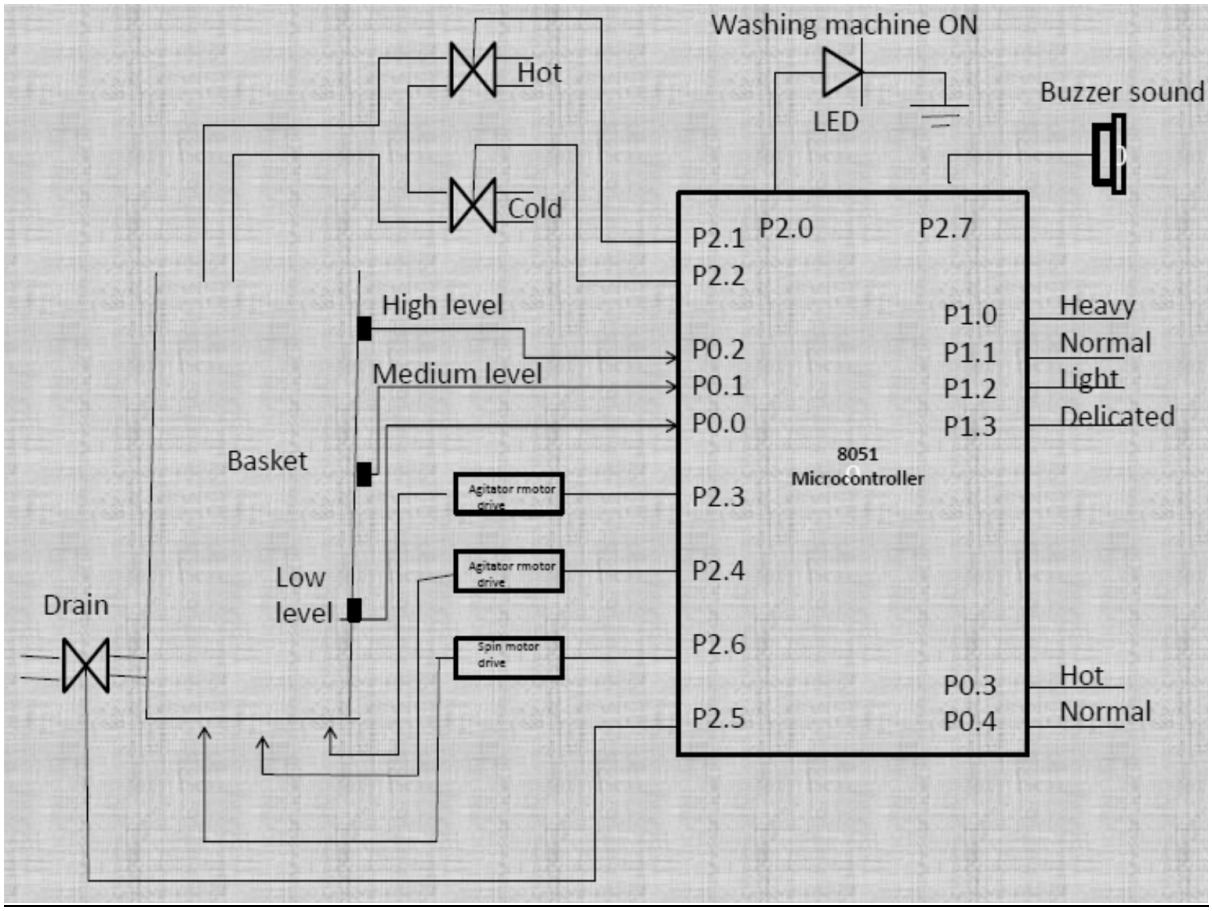
WASHING CYCLE FOR HEAVY, NORMAL, LIGHT AND DELICATE SETTING

Operation	Heavy	Normal	Light	Delicate
Fill water	Set by load Select knob	Set by load Select knob	Set by load Select knob	Set by load Select knob
Agitate	20 minutes	15 minutes	10 minutes	5 minutes
Drain	5 minutes	5 minutes	5 minutes	5 minutes
Fill water	Set by load Select knob	Set by load Select knob	Set by load Select knob	Set by load Select knob
Agitate	10 minutes	10 minutes	5 minutes	5 minutes
Drain	5 minutes	5 minutes	5 minutes	5 minutes
Spin	10 minutes	10 minutes	5 minutes	5 minutes

CONTROL KNOBS IN WASHING MACHINE

- Load select knob
- Water inlet select knob
- Mode select knob
- Program select knob

WASHING MACHINE CONTROL CIRCUIT DIAGRAM



PORTS USED

Operation	Signal	Input/output	Port pin no.
Load / water level select	Water level low Water level med Water level high	Input Input Input	P0.0 P0.1 P0.2
Water inlet	Hot water knob Normal water knob	Input Input	P0.3 P0.4
Program select	Heavy Normal Light Dedicate	Input Input Input Input	P1.0 P1.1 P1.2 P1.3
Machine ON	Machine on indic	Output	P2.0
Fill water	Hot water inlet Normal water inlet	Output Output	P2.1 P2.2
Agitation control	Motor rotate in cloc direction Motor rotate in anticlock direc	Output Output	P2.3 P2.4
Drain	Drain valve open	Output	P2.5
Spin	Spin motor ON/OFF	Output	P2.6
Washing ccomplete	Washing comp indic	Output	P2.7

WASHING MACHINE CONTROL PROGRAM

Labels	Mnemonics	Operands	Comments
	SETB LCALL JNB SJMP	P2.0 FILL_1 P1.0,LOOP_1 HEAVY	Machine ON indication Machine fill with water 1st time Chk prog setng knob for heavy. if P1.0 is not set,jump to LOOP_1 If P1.0 is set,jump to HEAVY
LOOP_1	JNB SJMP	P1.1,LOOP_2 NORMAL	Check prog setng knob for normal.if P1.1 is not set.jump to LOOP_2 If P1.1 is set, jump to NORM
LOOP_2	JNB SJMP	P1.2,LOOP_3	Chck prog setng knob for normal.if P1.2 is not set,jump to LOOP_3 If P1.2 is set,jump to LIGHT
LOOP_3	JNB SJMP	P1.3,LOOP_4 DELICATE	Check prog set knob for delicate. If P1.3is not set,jump to LOOP_4 If P1.2 is set,jump to delicate
DISPLAY	SETB	P2.7	Indicate the completion of wash cycle.
LOOP_4	NOP LJMP	 0000	 End of program
