Q1. Give difference between microprocessor 8086 and micro controller 8051.

Answer:-

Microcontroller	Microprocessor
Microcontrollers are used to execute a single task within an application.	**Microprocessors are used for big applications.**
Its designing and hardware cost is low.	**Its designing and hardware cost is high.**
Easy to replace.	**Not so easy to replace.**
It is built with CMOS technology, which requires less power to operate.	**Its power consumption is high because it has to control the entire system.**
It consists of CPU, RAM, ROM, I/O ports.	**It doesn't consist of RAM, ROM, I/O ports. It uses its pins to interface to peripheral devices.**

Q2. Explain feature of 8051.

Answer:- A **microcontroller** (**MCU** for *microcontroller unit*) is a small computer on a single metal-oxide-semiconductor (MOS) integrated circuit chip.

A **microcontroller** contains **CPU** along with **memory** and **programmable input/output peripherals**.

The 8051 Microcontroller was designed in 1980s by Intel. Intel re-intended Microcontroller 8051 employing CMOS technology and a new edition came into existence with a letter C in the title name, for illustration 80C51 rather than NMOS technology.

There are two buses in 8051 Microcontroller one for program and other for data. The microcontroller comprise of 8 bit accumulator 8 bit processing unit.

- It having 8 bit CPU.
- It is built with 40 pins DIP (dual inline package) IC.
- On-chip program memory (ROM) 4KB bytes

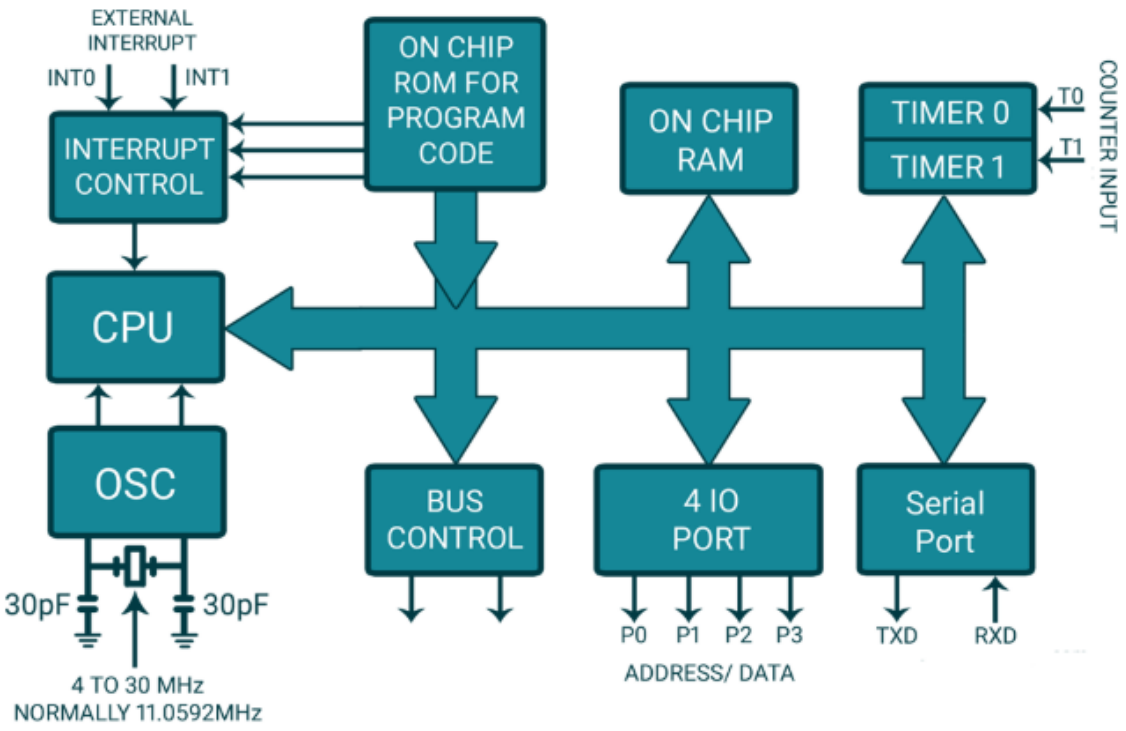
- On-chip data memory (RAM) 128 bytes
- Four register banks (Bank 0- Bank 3)
- 8-bit bidirectional data bus
- 16-bit unidirectional address bus
- 32 general purpose registers each of 8-bit
- 2- sixteen bit Timer/Counter (Timer 0 and Timer 1).
- Three internal and two external Interrupts.
- 32 bidirectional I/O lines arranged as four 8-bit port (port 0 – port 3).
- 16-bit program counter and data pointer.
- An on-chip crystal oscillator is integrated in the microcontroller having crystal frequency of 12 MHz.(operating frequency 11.029MHZ)

Q3. Explain architecture of 8051 micro-controller.

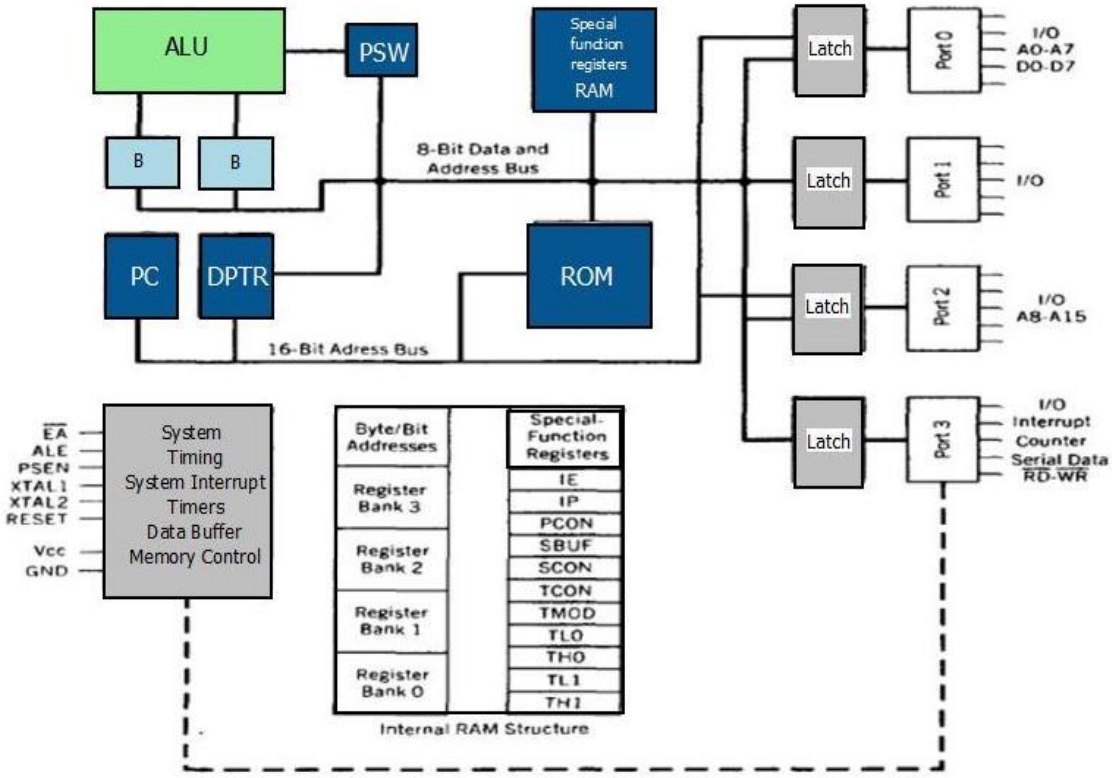
Answer:-

Intel's 8051 Microcontroller Architecture

You can draw either diagram:



Or



- ALU performs arithmetic like addition, subtraction, multiplication and Logical Operations like NAND, NOR etc.
- Since 8051 is an 8 bit microcontroller, it takes input from two 8 bit registers namely A and B and processes them.

Registers of 8051

8 bit Registers

- The commonly used registers are R0 to R7, A, B, PSW.
- These are all 8 bit registers.
- The registers associated with the ALU are A,B and PSW.
- In most operations, ALU performs specified operation on registers 'A' and 'B' and the result is stored back in 'A'. Hence Register 'A' is often called accumulator.

8051 contains:

- **4K** Bytes of on-chip ROM instruction memory
- 256 Bytes of On-chip RAM 128 Bytes of on-chip RAM for temporary data storage and the stack + 128 SFR space

- 2 timers,
- **1** serial port, and
- **4** 8-bit parallel I/O ports.

The 8051 memory is divided into Program Memory and Data Memory.

- The Program Memory of the 8051 Microcontroller is used for storing the program to be executed i.e. instructions.
- The Data Memory on the other hand, is used for storing temporary variable data and intermediate results.

CPU :

It contains

- **A (Accumulator):** which stores the result for all arithmetic and logic instructions
- **B (register):**
  - Used to store 8-bit data
  - Store MSB part of the result in MUL instruction
  - Stores the remainder part of the result of DIV instruction
- **DPTR (data pointer):** is a **16**-bit register which holds the address of data in the memory
- **PC (program counter):** is a **16**-bit register which holds the address of next executable instruction in the program memory.
- **SP (stack pointer):** is a **8**-bit register which holds the address of stack memory. So the stack address range 00H to FFH. On reset, the SP is initialized to 07H.
- During PUSH the SP is incremented, while during POP the SP is decremented.

FLAG OR PSW (Program status word) OF 8051 :

- The PSW contain the status flags, which reflects the current status of the CPU results
- It also two register select bits that identify which of the four General-purpose register banks is currently in use by the program.
- The PSW is bit addressable

RS1	RS0	Register Bank	Address
0	0	0	00H – 07H
0	1	1	08H – 0FH
1	0	2	10H – 17H
1	1	3	18H – 1FH

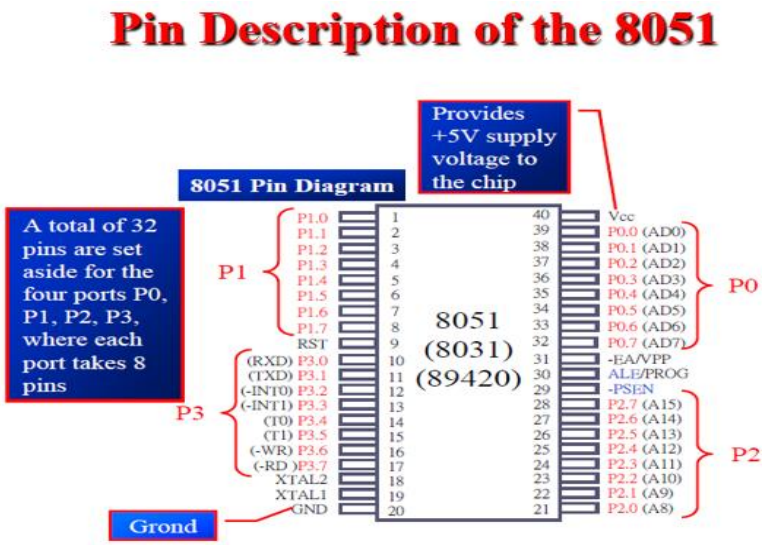
CY	AC	F0	RS1	RS0	OV	—	P

CY	PSW.7	Carry Flag
AC	PSW.6	Auxiliary Carry Flag
FO	PSW.5	Flag 0 available to the user for general purpose
RS1	PSW.4	Register Bank selector bit 1
RS0	PSW.3	Register Bank selector bit 0
OV	PSW.2	Overflow Flag
—	PSW.1	User definable flag
P	PSW.0	Parity flag. Set/cleared by hardware each instruction cycle to indicate an odd/even umber of '1' bits in the accumulator.

- **System Timing**, refers to the entire timing unit of the microcontroller. The primary source of timing for 8051 is external crystal. The system timing unit is mainly responsible for providing timing to critical events like instruction fetch, decode and execute operations.
- **Interrupts**: 8051 provides a method to stop current execution and switch over to a priority task with the use of Interrupts. They are associated with external pins P3.2(INT0) and P3.3(INT1). As well as internal units like timers and serial communication. Thus an interrupt to 8051 can be external generated by peripheral device or internal from one of its inbuilt units sometimes referred as software interrupt.
- **Timers/Counters**: Again one of the most widely used features, timers are used to count time internally or can be made to count external event as counters.
- **Memory Control**: It is an internal unit of the microcontroller, responsible for accessing data from RAM and ROM.

Q4. Give PINS Detail of 8051.

Answer:- Total 40 pins



- **Pins 1 to 8** − These pins are known as Port 1. This port doesn't serve any other functions. It is internally pulled up, bi-directional I/O port.
- **Pin 9** − It is a RESET pin, which is used to reset the microcontroller to its initial values.
- **Pins 10 to 17** − These pins are known as Port 3. This port serves some functions like interrupts, timer input, control signals, serial communication signals RxD and TxD, etc.
- **Pins 18 & 19** − These pins are used for interfacing an external crystal to get the system clock.
- **Pin 20** − This pin provides the power supply to the circuit.
- **Pins 21 to 28** − These pins are known as Port 2. It serves as I/O port. Higher order address bus signals are also multiplexed using this port.
-
- **Pin 29 – This is PSEN pin which stands for Program Store Enable. It is used to read a signal from the external program memory.**
- **Pin 30 – This is EA pin which stands for External Access input. It is used to enable/disable the external memory interfacing.**
- **Pin 31 – This is ALE pin which stands for Address Latch Enable. It is used to demultiplex the address-data signal of port.**

➢ RESET Pin

It is a reset input Pin which used to reset the 8051 microcontrollers. It is an active HIGH Pin i.e. if we apply a high pulse to this pin, the microcontroller will reset and terminate all activities.

➢ Oscillator Pins

Pin 18 and 19 are used for interfacing an **external crystal oscillator** to get the **system clock**. We should use the ceramic capacitor with an external crystal oscillator.

➢ Power Pins (+5V and GND)

➢ PSEN Pin

- **Program Store Enable Pin** (PSEN).
- It is an **output** pin and
- Using this pin, **external Program Memory** can be read.

➢ ALE/PROG Pin

- It is an **output** pin and
- Using this Pin, **address can be separated from data** (as they are multiplexed by 8051).

➢ EA/VPP Pin

- **External Access Enable Pin** i.e. allows external Program Memory.
- **It is an input pin** and must be connected from VCC or GND.
- If you want to access the code from external program memory, it must be connected with GND. If you want to use on-chip memory, it must be high (connected with VCC)

Q5. Discuss Input output ports of 8051

Answer:- Four port are available, in each port there is 8 pins.

I/O PORTS:

Port 0:

- It is used for dual purpose port (for I/O and multiplexed address/data bus).
- As I/O port , it can be used either as input or as output port.
- When an external memory is interfaced, this port carry the low byte of address (A0-A7) strobed by ALE signal.
- Then it carries the data (D0-D7) for external memory stroabed by RD' and WR' control signal
- It is both byte accessible (P0) and bit accessible (P0.0 , P0.1, ..., P0.7)
- To use this port as both input/output ports each pin must be connected externally to pull-up resistor.

Port 1:

- It is a dedicated port, and can be used either as input or as output port.
- Upon reset it is configured as output port.
- It is both byte accessible (**P1**) and bit accessible (**P1.0** , **P1.1**, ..., **P1.7**)
- It does not require external pull-up resistor like P0.

Port 2:

- It is used for dual purpose port (for I/O and high byte address).
- As I/O port , it can be used either as input or as output port.
- When an external memory is interfaced, this port carry the high byte of address (A8-A15)
- It is both byte accessible (**P2**) and bit accessible (**P2.0** , **P2.1**, ..., **P2.7**)
- It does not require any external pull up resistors.

Port 3:

- It is used for dual purpose port (for I/O and other functions).
- As I/O port , it can be used either as input or as output port.
- It is both byte accessible (**P3**) and bit accessible (**P3.0** , **P3.1**, ..., **P3.7**)
- Designers generally avoid using this port unnecessarily for I/O because the pins have alternate functions which are related to special features of the 8051 :
  - ✓ Serial I/O - **TXD, RXD**
  - ✓ Timer clocks - **T0, T1**
  - ✓ Interrupts - **INT0, INT1**
  - ✓ Data memory- **RD', WR'**
- ✓ It does not require any external pull up resistors.

Q6. Discuss addressing modes of 8051

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

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

1. **Immediate Addressing**

In this, the data **itself** is specified in the instruction as source operand .

Example: ADD A, **#20H**

MOV A, **#255**

MOV R0, **10101110B**

MOV R7, **#0A0H**

MOV DPTR, **#2001H**

In all the above examples, the data **itself** is specified as source operand in the instruction, and hence it is an immediate addressing instruction. EITHER A OR ANY R REGISTER.LENGTH OF DATA SHOULD BE SAME.REMEMBER PSW TO SELECT RESISTER BANK.

2. Register Addressing

In this, **both** the source and destination operand location is specified via name of the registers.

Example: ADD **A**, **B**

MOV **A**, **R0**

MOV **P0**, **A**

In all the above examples, **both** the source and destination operand location is specified via name of the registers, and hence it is an register addressing instruction.ONE OPERAND MUST BE ACUUMULATOR(A) REGISTER.

3. Direct Addressing

In this, one of the operand is in memory, and its location address is **directly** specified in the instruction .

Example: ADD A, 70H

MOV 30H, R0

MOV R1, 55H

MOV 55H,70H

MOV 55H,#12H

In all the above examples, the memory address is **directly** specified in the instruction , and hence it is a direct addressing instruction.

4. **Indirect Addressing**

In this, one of the operand is in memory, and its location address is **indirectly** specified using register contents.

Example: ADD A, @R0
MOV @R1, R5
MOV R7, @R1

In all the above examples, the memory address is **NOT** directly specified in the instruction , where as they are **indirectly** available in some registers, and hence it is a indirect addressing instruction.

5. **Indexed Addressing**

In this, one of the operand is in memory, and its effective address is the **sum** of a base register and an offset register. The base register can be either Data Pointer (**DPTR**) or Program Counter (**PC**) while the offset register is the Accumulator (**A**)

Example: MOVC A, @A+DPTR
MOVC A, @A+PC

- In both the above examples, the memory address is the sum of contents of DPTR/PC and Accumulator.
- Indexed Addressing mode is useful when retrieving data from look-up tables

Q7. Discuss on interrupts of 8051

Answer:-

An interrupt is an external or internal event that interrupts the microcontroller to inform it that a device needs its service

Upon activation of an interrupt, the microcontroller goes through the following steps

1. It finishes the instruction it is executing and saves the address of the next instruction (PC) on the stack
2. It also saves the current status of all the interrupts internally (i.e.: not on the stack)
3. It jumps to a fixed location in memory, called the interrupt vector table, that holds the address of the ISR
4. The microcontroller gets the address of the ISR from the interrupt vector table and jumps to it

5. Upon executing the RETI instruction, the microcontroller returns to the place where it was interrupted.

INTERRUPTS

- **There are 6 Interrupt Sources (in order of priority):**

1. **External Interrupt 0 (IE0) (from INT0 pin)**
2. **Timer 0 (TF0)**
3. **External Interrupt 1 (IE1) (from INT1 pin)**
4. **Timer 1 (TF1)**
5. **Serial Port (RI/TI)**
6. **RESET**

- **Each interrupt type can be programmed to one of two priority levels.**
- **External interrupts can be programmed for edge or level sensitivity.**
- **Each interrupt type has a separate vector address. When any of the interrupt arrives, then the program execution will be transferred to the vector address shown below.**

Interrupt	ROM Location (hex)	Pin
Reset	0000	9
External HW (INT0)	0003	P3.2 (12)
Timer 0 (TF0)	000B	
External HW (INT1)	0013	P3.3 (13)
Timer 1 (TF1)	001B	
Serial COM (RI and TI)	0023	

```
        ORG   0       ;wake-up ROM reset location
        LJMP  MAIN    ;by-pass int. vector table
;----- the wake-up program
        ORG   30H
MAIN:
        ....
        END
```

Only three bytes of ROM space assigned to the reset pin. We put the LJMP as the first instruction and redirect the processor away from the interrupt vector table.

-

Role of interrupts enable register: 8-bit register

It is Used to enable of disable the available interrupt in 8051

INTERRUPT ENABLE REGISTER

IE (Interrupt Enable) Register

D7							D0
EA	--	ET2	ES	ET1	EX1	ET0	EX0

EA (enable all) must be set to 1 in order for rest of the register to take effect

EA	IE.7	Disables all interrupts
--	IE.6	Not implemented, reserved for future use
ET2	IE.5	Enables or disables timer 2 overflow or capture interrupt (8952)
ES	IE.4	Enables or disables the serial port interrupt
ET1	IE.3	Enables or disables timer 1 overflow interrupt
EX1	IE.2	Enables or disables external interrupt 1
ET0	IE.1	Enables or disables timer 0 overflow interrupt
EX0	IE.0	Enables or disables external interrupt 0

Role of Interrupt Priority register:-8 bit register

It is used to set the priority level in interrupt process

Interrupt Priority Register (Bit-addressable)

D7							D0
--	--	PT2	PS	PT1	PX1	PT0	PX0

--	IP.7	Reserved
--	IP.6	Reserved
PT2	IP.5	Timer 2 interrupt priority bit (8052 only)
PS	IP.4	Serial port interrupt priority bit
PT1	IP.3	Timer 1 interrupt priority bit
PX1	IP.2	External interrupt 1 priority bit
PT0	IP.1	Timer 0 interrupt priority bit
PX0	IP.0	External interrupt 0 priority bit

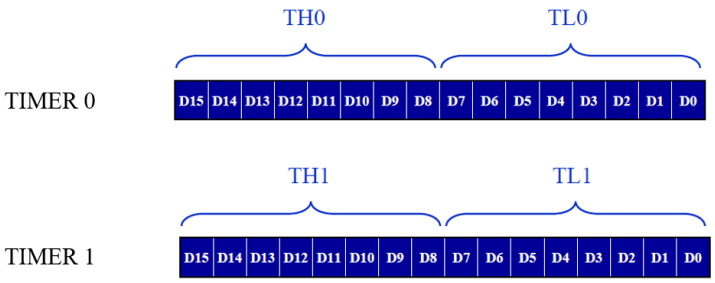
Priority bit=1 assigns high priority
Priority bit=0 assigns low priority

Q8. (i) Discuss on Timers of 8051 (ii) Discuss the role of TMOD register in time delay generation (iii) Discuss the role of TCON register in time delay generation.

Answer:-

The 8051 has two timers:

1. TIMER 0
2. TIMER 1



The timers have three general functions :

1.Keeping time and /or calculating the amount of time between events.

2.Counting the events themselves.

3.Generating band rates for the serial port.

- Timer 0 Registers

- The 16-bit register of timer 0 is accessed as low byte and high byte. The low byte register is called TL0 (Timer 0 low byte) and high byte register is called TH0 (Timer 0 High byte).
- These registers can be accessed like any other register such as A,B,R0,R1,R2, etc. For example :MOV TL0,#2FH moves the value 2FH into TL0, the low byte of timer 0.
- These registers can be also read like any other registers. For example: MOV R5, TH0 saves TH0 (high byte of Timer 0) in R5.

- Timer 1 Registers

- Timer 1 is also 16-bits, and its 16-bit register is split into two bytes, referred to as TL1 (Timer 1 low byte) and TH1(Timer 1 high byte).
- These registers are accessible in the same way as the registers of Timer 0.

(ii)Modes in Timers:

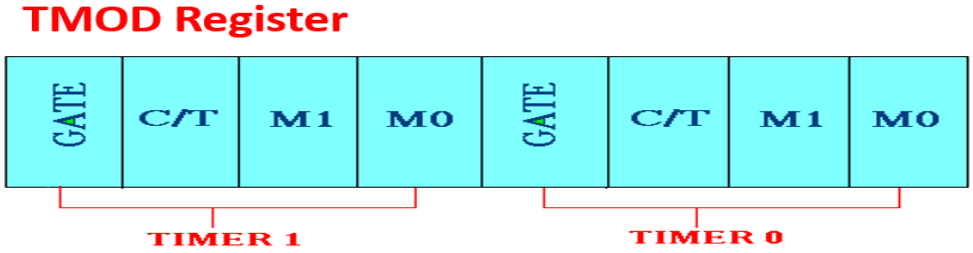
The 4- modes available for the timers in the 8051 microcontroller include:

1. Mode 0 (13-bit Timer):
    - Timer 0 and Timer 1 can operate as 13-bit timers.
    - In this mode, the timers count from 0 to 8191 (2^13 - 1).
2. Mode 1 (16-bit Timer):
    - Timer 0 and Timer 1 can operate as 16-bit timers.
    - In this mode, the timers count from 0 to 65535 (2^16 - 1).
3. Mode 2 (8-bit Auto-Reload):
    - Timer 0 and Timer 1 can operate in 8-bit auto-reload mode.
    - In this mode, the timers are used as timers/counters with automatic reload.
    - Timer 0 automatically reloads from TH0 and TL0 when it overflows.
    - Timer 1 automatically reloads from TH1 and TL1 when it overflows.
4. Mode 3 (Split Timer):
    - Timer 1 can operate as two separate 8-bit timers (TL1 and TH1).
    - TL1 and TH1 can be used independently, providing more flexibility for certain applications.

The mode of operation is determined by the control bits in the Timer Mode Control Register ..**TMOD REGISTER and TCON REGISTER**

(iii)Role of TMOD REGISTER(Timer Mode Control Register (TMOD)

The mode of operation is determined by the control bits in the Timer Mode Control Register (TMOD) of the 8051 microcontroller. The TMOD register is an 8-bit register, and each 4 bits correspond to a timer. For example, the lower 4 bits (TMOD.3 - TMOD.0) control Timer 0, and the upper 4 bits (TMOD.7 - TMOD.4) control Timer 1.



- Gate : When set, timer only runs while INT(0,1) is high.
- C/T : Counter/Timer select bit.
- M1 : Mode bit 1.
- M0 : Mode bit 0.

M1	M0	MODE
0	0	13-bit timer mode
0	1	16-bit timer mode
1	0	8-bit auto-reload mode
1	1	split mode

Symbol and pins	Functions
GATE	This bit controls whether to use an internal or external source for the timer or counter.
GATE = 1	We can start and stop the timer or counter from an external source.
GATE = 0	In this case, we don't need any external hardware to start and stop the timer or counter.
C/T	This bit decides whether the timer is used as a time delay generator or as an event counter.
C/T = 0	The timer (Timer0 or Timer1) is used as a timer.
C/T = 1	The timer (Timer0 or Timer1) is used as a counter.

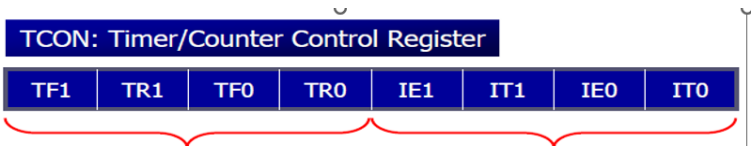
These instructions set the Timer Mode Control Register (TMOD) to specify the desired modes for Timer 0 and Timer 1.

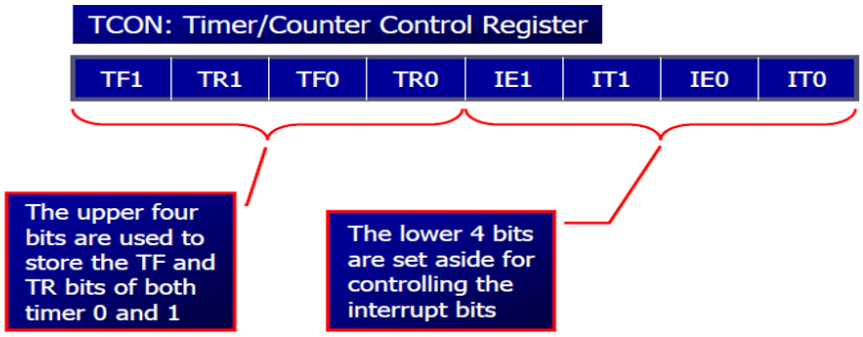
MOV TMOD, #01H ; Set Timer 0 in 16-bit mode (Mode 1)

MOV TMOD, #10H ; Set Timer 1 in 8-bit auto-reload mode (Mode 2)

(iv) Role of TCON REGISTER

Timer Control Register (TCON)

- TCON is an 8-bit register.
- The upper four bits are used to store the TF and TR bits of both Timer 0 and Timer 1.
- The lower four bits are set aside for controlling the interrupt bits.
- TR0 and TR1 flags are used to turn on or turn off the timers
- 



TCON.7 - TF1: Timer 1 overflow flag.
TCON.6 - TR1: Timer 1 run control bit.
TCON.5 - TF0: Timer 0 overflag.
TCON.4 - TR0: Timer 0 run control bit.
TCON.3 - IE1: External interrupt 1 edge flag.
TCON.2 - IT1: External interrupt 1 type flag.
TCON.1 - IE0: External interrupt 0 edge flag.
TCON.0 - IT0: External interrupt 0 type flag.

Symbol and Bits	Functions
TF1	Overflow flag for Timer1.
TF1 = 1	Set when timer rolls from all 1s to 0.
TF1 = 0	Cleared to execute the interrupt service routine.
TR1	Run the control bit for Timer1.
TR1 = 1	Turns on Timer1.
TR1 = 0	Turns off Timer1.
TF0	Overflow flag for timer0, same as TF1.
TR0	Run the control bit for Timer0, same as TR1.
IE1	External interrupt 1 Edge flag It is not related to timer operations.
IT1	External interrupt 1 signal type control bit.
IT1 = 1	Enables external interrupt 1 to be triggered by a falling edge signal.
IT1 = 0	Enables a low-level signal on external interrupt 1 to generate an interrupt.

Q9.B Write a program to Blink the led at delay interval of 10ms with the help of TIMER1 using MODE1 . The crystal frequency is 11.059 MHz.

Solution : From the question : Note that , you have to use TIMER 1 and mode 1. In mode 1, the maximum value upto which the timer1 will roll is FFFFH(OR 2^{16}-1=65536)

1.**Timer Clock Source**

The timer needs a clock source. If C/T = 0, the **crystal frequency** attached to the 8051 is the source of the clock for the timer. The value of the crystal frequency attached to the microcontroller determines the speed at which the timer ticks. The crystal frequency is 11.059 MHz.

2. Timer Clock Frequency

The frequency for the timer is always 1/12th of the frequency of the crystal attached to the 8051.

TF = 1/12 x 11.059 MHz = 921583 Hz

3.Timer Clock Period

The time delay of one machine cycle is given below. We use this to generate the delay.

TP = 1/921583 = 1.085 µ sec.

4. CALCULATE the number of steps the timer clock period will take to generate the desire delay:

N=Number of steps= 10ms (given in question)/ 1.085 µ sec.=-9216

5. calculate the initial value of the timer:

M=Initial value = 2^{16}-9216=56320……

Optional:-you can also use formula

(Desired delay= (max value –initial value)x TP(timer clock period)

10ms=(2^{16}-initial value)x1.085 µ sec.

INITIAL VALUE=56320

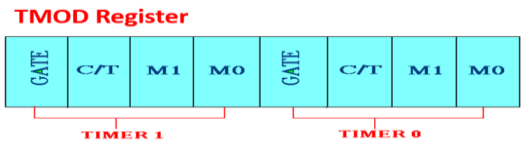
###NOTE THE MAXIMUM VALUE WILL CHANGE WITH RESPECT TO MODE USED

6. Convert M(initial value into hexa decimal

M=56320=DC00H

LOAD LOWER BYTE M IN TL1 and upper byte in TH1 of timer1

7. CONFIGURE TMOD REGISTER



SINCE YOU ARE USING TIMER1 IN MODE 1, SEE IN QUESTION SO USE UPPER 4 BIT OF TMOD REGISTER , LOWER 4 BIT IS KEPT ZERO. MO=1 and M1=0

0	0	0	1	0	0	0	0

THE VALUE IN TMOD WILL BE 00010000B 0R 10 H

WRITE PROGRAM USING INITIAL VALUE AND TMOD VALUE

```
ORG 0H
        MOV TMOD,#00010000B
BACK:   MOV TL0,#00H
        MOV TH0,#DCH
        CPL P1.0
        ACALL DELAY
        SJMP BACK
DELAY:SETB TR1----------tis is to start the timer1, see TCON register
HERE:   JNB TF1, HERE-……TF1 WILL become 1 at delay of 10ms or reach at max value
        CLR TR1
        CLR TF1
        RET
END
```

Q9C. . Example: Blinking led with the help of timer using MODE2-Auto reload mode

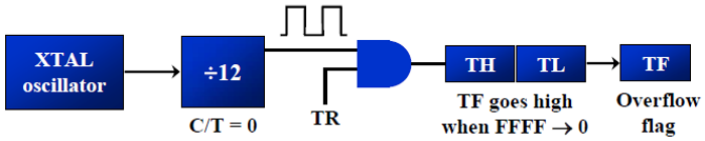
```
ORG 0H
        MOV TMOD,#00100000B
        MOV TH1, #05H
        SETB TR1
HERE:   JNB TF1, HERE
        CPL P2.0
        CLR TF1
        SJMP HERE
END
```

Q9.D Write the algorithm to use the timer to generate the delay

Solution:-

1. Load the TMOD value register indicating which timer (Timer0 or Timer1) is to be used and which timer mode is selected.
2. Load registers TL & TH with initial count values.
3. Start the timer
4. Keep monitoring the timer flag(TF)
5. Stop the timer.
6. Clear TF for next round .
7. Go back to step 2 to load TH & TL again

NOTE: MODE 0 is exactly like mode-1 except that it is 13-bit timer instead 16-bit. The 13-bit timer/counter can hold values between 0000H to 1FFFH in TH-TL.



Q10. Discuss Serial Ports

SERIAL PORT

- The 8051 has two pins for transferring (**TxD**) and receiving (**RxD**)data by serial communication. These two pins are part of the Port3 (P3.0 &P3.1)
- These pins are TTL compatible and hence they require a line driver to make them **RS232** compatible
- Serial communication is controlled by an 8-bit register called **SCON** register, it is a bit addressable register.
- **SBUF** register will be used to hold the data to be transmitted or the received data

Q11. Discuss Asynchronous , synchronous communication ,UART AND USART

ANSWER:-

Serial data communication uses two methods

1. **Synchronous**
2. **Asynchronous**

Synchronous method transfers a block of data at a time

Asynchronous method transfers a single byte at a time

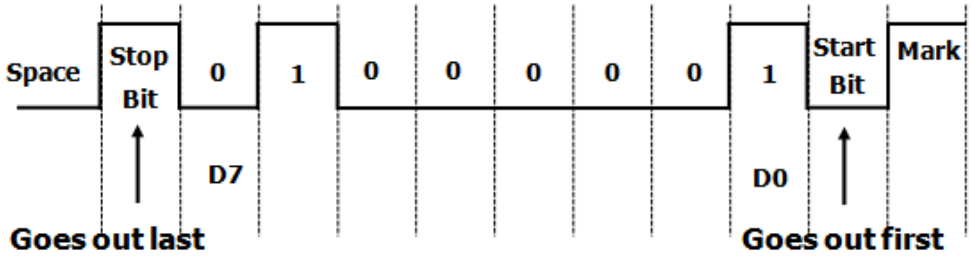
OTHER differences given below

Sr. No.	Factor	Asynchronous	Synchronous
1.	Data sent at one time	Usually 1 byte	Multiple bytes
2.	Start and Stop bit	Used	Not used
3.	Gap between data units	Present	Not present
4.	Data transmission speed	Slow	Fast
5.	Cost	Low	High

- **Accordingly, there are special IC chips made by many manufacturers for serial communications**
  - **UART (universal asynchronous Receiver- transmitter)**
  - **USART (universal synchronous-asynchronous Receiver-transmitter)**

UART (universal asynchronous Receiver- transmitter)

- Asynchronous serial data communication is widely used for character-oriented transmissions
  - Each character is placed in between start and stop bits, this is called framing
  - Block-oriented data transfers use the synchronous method
- The start bit is always one bit, but the stop bit can be one or two bits



- The rate of data transfer in serial data communication is stated in *bps* (bits per second)
- Another widely used terminology for bps is ***baud rate***

- ❑ It is modem terminology and is defined as the number of signal changes per second
- ❑ In modems, there are occasions when a single change of signal transfers several bits of data
- ❑ To allow data transfer between the PC and an 8051 system without any error, we must make sure that the baud rate of 8051 system matches the baud rate of the PC's COM port. The following are the standard baud rate used in serial communication.

USART (universal synchronous-asynchronous Receiver-transmitter)

USART, which stands for Universal Synchronous-Asynchronous Receiver-Transmitter, is a widely used communication protocol in microcontroller-based systems. It provides a versatile means of serial communication between devices, allowing them to exchange data. USART supports both synchronous and asynchronous communication modes, providing flexibility for different applications.

Here are some key aspects of USART:

1. **Synchronous and Asynchronous Modes:**
    - **Synchronous Mode:** In this mode, communication is synchronized by a common clock signal shared between the transmitter and receiver. Both devices must be configured to use the same clock rate.
    - **Asynchronous Mode:** In this mode, there is no common clock signal between the transmitter and receiver. Instead, the devices use agreed-upon baud rates to set the speed of data transfer. Start and stop bits are used to frame each data byte.
2. **Baud Rate:**
    - Baud rate refers to the speed at which data is transmitted over the communication channel. It is expressed in bits per second (bps). In asynchronous mode, both the transmitter and receiver must be configured to use the same baud rate for successful communication.
3. **Data Frame Format:**
    - The data frame in USART typically consists of a start bit, a specific number of data bits (usually 8 bits), an optional parity bit (for error checking), and one or more stop bits to signal the end of the data byte.
    - The format of the data frame (number of data bits, parity, and number of stop bits) is configurable based on the application requirements.

Q12. Discuss Resisters used for serial communication in 8051

Answer:-

(I) SBUF (SERIAL BUFFER)

(II) SCON(SERIAL CONTROL REGISTER)

(III) PCON(POWER CONTROL REGISTER)

(IV) TH1 TIMER FOR BAUD RATE CONTROLLER (TIMER MODE 2)

1.SBUF (SERIAL BUFFER)

- **SBUF** is an 8-bit register used solely for serial communication.
- For a byte data to be transferred via the TxD line, it must be placed in the SBUF register.
  - The moment a byte is written into SBUF, it is framed with the start and stop bits and transferred serially via the TxD line.
- **SBUF** holds the byte of data when it is received by 8051 RxD line.
  - When the bits are received serially via RxD, the 8051 deframes it by eliminating the stop and start bits, making a byte out of the data received, and then placing it in SBUF
- **SBUF** register can be used as any other register in the program as shown below.

```
MOV SBUF,#'D'   ;load SBUF=44h, ASCII for 'D'
MOV SBUF,A      ;copy accumulator into SBUF
MOV A,SBUF      ;copy SBUF into accumulator
```

(II) SCON(SERIAL CONTROL REGISTER)

❑ **SCON** is an 8-bit register used to program the start bit, stop bit, and data bits of data framing, among other things.

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

SM0	SCON.7	Serial port mode specifier
SM1	SCON.6	Serial port mode specifier
SM2	SCON.5	Used for multiprocessor communication
REN	SCON.4	Set/cleared by software to enable/disable reception
TB8	SCON.3	Not widely used
RB8	SCON.2	Not widely used
TI	SCON.1	**Transmit interrupt** flag. Set by HW at the begin of the stop bit mode 1. And cleared by SW
RI	SCON.0	**Receive interrupt** flag. Set by HW at the begin of the stop bit mode 1. And cleared by SW

Note 1: When 8051 finishes the transfer of 8-bit character, it raises **TI** flag to indicate that it is ready to transfer another byte

SM0	SM1	Mode
0	0	Serial mode 0
0	1	**Serial mode 1 (8-bit data, 1 start bit, 1 stop bit)**
1	0	Serial mode 2
1	1	Serial mode 3

Note 2:

- When 8051 receives data serially via RxD, it gets rid of the start and stop bits and places the byte in **SBUF** register.
- It raises the **RI** flag bit to indicate that a byte has been received and should be picked up before it is lost

Role of T1 and R1 in SCON register:

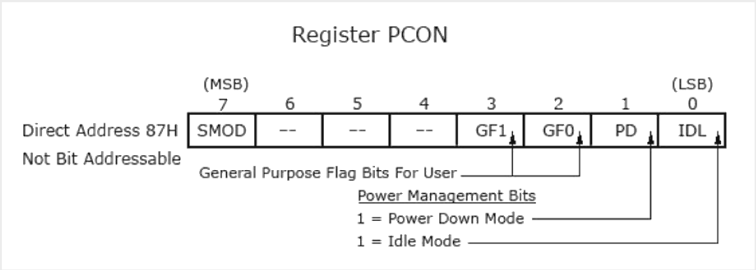
RI & TI in SCON play an important role in serial communications. Both bits are set by hardware but must be cleared by software.

- RI is set at the end of character reception and indicates "receive buffer full".
- This condition is tested in software or programmed to cause an interrupt.
- If software wishes to input a character from the device connected to the serial port, it must wait until RI is set, then clear RI and read the character from SBUF.

(III) PCON(POWER CONTROL REGISTER)

PCON

PCON (Power control) register is used to force the 8051 microcontroller into power saving mode. Power control register of 8051 contains two power saving mode bits and one serial baud rate control bit.



SMOD **1**= Baud rate is doubled in UART mode 1, 2 and 3. 0 = No effect on Baud rate {BR-(1/64)*Fcrystal}.

GF1 & GF0 : These are general purpose bit for user.

IDl- CLK stop

PD-inter nal oscillator stop and power reduced from 5v to 2v

SMOD (Serial Mode Bit):
•SMOD is a bit in the SCON register that is used to select between two different modes of serial communication, known as Mode 0 and Mode 1.
•When SMOD is set (SMOD = 1), it enables Mode 1, which is a variable baud rate mode. When cleared (SMOD = 0), it enables Mode 0, which is a fixed baud rate mode.

The PCON (Power Control) register in the 8051 microcontroller is used to control various power-related features. The specific bits in the PCON register can vary between different variants of the 8051 microcontroller, so it's important to refer to the datasheet of the particular microcontroller you are working with for precise details. However, I'll provide a general description based on common features found in many 8051 microcontrollers:

IDL (Idle Mode Control):
The IDL bit (Bit 0) in PCON controls the entry into the Idle mode.
When IDL is set (PCON.0 = 1), the 8051 enters the Idle mode when the software executes the "IDLE" instruction.
In Idle mode, the CPU is halted, but the peripherals (such as timers and serial ports) continue to operate.

PD (Power-Down Mode Control):
The PD bit (Bit 1) in PCON controls the entry into the Power-Down mode.
When PD is set (PCON.1 = 1), the 8051 enters the Power-Down mode when the software executes the "PD" instruction or when external hardware triggers a wake-up event.
In Power-Down mode, both the CPU and the peripherals are in a low-power state.

Q12 A. Write a program for 8051 to transfer a letter 'A' at the rate of 9600 baud per second: Crystal frequency 11.0593MHz

Solution:- XTAL = *11.0592 MHz*

STEP-1 $Machine\ cycle\ frequency\ of\ 8051 = \frac{11.0592\text{MHz}}{12} = 921.6KHz$

STEP-2 $Frequency\ provided\ by\ UART\ to\ Timer\ 1 = \frac{921.6KHz}{32} = 28{,}800Hz$

(A) To have the baud rate of 9600, Timer T1 value can be calculated as

STEP-3 $\frac{28{,}800}{9600} = 3$

STEP-4 **Decimal value of TH1** = negative of answer in **STEP-3** = -3

To find HEX value, start from the highest value in HEX that can be loaded into the timer and count in decrementing order thrice to get the HEX value as

STEP-5 FFH -> FEH -> FDH

STEP-6 LOAD the TMOD REGISTER in mode 2 for timer 1

MOV TMOD,#20H	timer 1,mode 2(auto reload)

Timer 1				Timer 0			
GATE	C/T*	M1	M0	GATE	C/T*	M1	M0
0	0	1	0	0	0	0	0

STEP-7 LOAD TIMER1 WITH THE INITIAL VALUE, which is calculated at step4

MOV TH1,#-6	9600 baud rate (28800/9600)=3 (USE -3 and 2's complement of (-3) is FD (SO, it will from initial value FD to FF). Max value in auto reload mode is FF

STEP-8 LOAD SCON register

MOV SCON,#50H ; Note 1: When 8051 finishes the transfer of 8-bit character, it raises TI flag to indicate that it is ready to transfer another byte	8-bit, 1 stop, REN enabled

SM0	SM1	SM2	REN	TB8	RB8	TI	RI
0	1	0	1	0	0	0	0

0 1 Serial mode 1 (8-bit data, 1 start bit, 1 stop bit)

- **When 8051 receives data serially via RxD, it gets rid of the start and stop bits and places the byte in SBUF register.**
- **It raises the RI flag bit to indicate that a byte has been received and should be picked up before it is lost**

STEP-9: THE PROGRAM:-

```
        ORG 00H
        MOV  TMOD,#20H ------FROM STEP 6
        MOV TH1, #0FDH-------FROM STEP7
        MOV SCON,#50 H--------FROM STEP8
        SETB TR1----------------START THE TIMER
AGAIN: MOV SBUF,#'A'--------LOAD SBUF WITH LETTER R
HERE:       JNB T1, HERE……WAIT FOR LAST BIT TO TRNSFER
            CLR TR1
            CLR T1
            SJMP AGAIN…….THE PROGRAM CONTINUE FOR INFINTE NUM
OF LOOPS
```

Q12-B

Write a program for the 8051 to transfer "YES" serially at 9600 baud, 8-bit data, 1 stop bit, do this continuously

Solution:

```
        MOV   TMOD,#20H   ;timer 1,mode 2(auto reload)
        MOV   TH1,#-3     ;9600 baud rate
        MOV   SCON,#50H   ;8-bit, 1 stop, REN enabled
        SETB  TR1         ;start timer 1
AGAIN:  MOV   A,#"Y"      ;transfer "Y"
        ACALL TRANS
        MOV   A,#"E"      ;transfer "E"
        ACALL TRANS
        MOV   A,#"S"      ;transfer "S"
        ACALL TRANS
        SJMP  AGAIN       ;keep doing it
;serial data transfer subroutine
TRANS:  MOV   SBUF,A      ;load SBUF
HERE:   JNB   TI,HERE     ;wait for the last bit
        CLR   TI          ;get ready for next byte
         RET
```

Q12-C WRITE AN ALGORITHM to transfer data serially

SOLUTION: In programming 8051 to transfer character bytes serially, the following steps must be taken

1. **The TMOD register is loaded with the value 20H , indicating the use of Timer 1 in mode 2(8-bit auto reload) to set the baud rate**
2. **The TH1 is loaded with one of the values to set the baud rate for serial data transfer(assuming XTAL= 11.0592 MHz).**
3. **The SCON register is loaded with the value 50H indicating serial mode 1, where an 8-bit data is framed with start & stop bits.**
4. **TR1 is SET to 1 to start Timer 1.**
5. **TI is cleared when transmission is complete.**
6. **The character byte to be transferred serially is written into SBUF register.**
7. **TI is monitored for transmission .**
8. **To transfer the next character go to step 5**

Q13. EXPLAIN 8051 Microcontroller Memory Organization, Program Memory (ROM), Data Memory (RAM), External Memory.

Answer:- The main difference can be stated as on-chip memory i.e., a Microcontroller has both Program Memory (ROM) and Data Memory (RAM) on the same chip (IC), whereas a Microprocessor has to be externally interface with the memory modules**.**

The 8051 Microcontroller Memory is separated in Program Memory (ROM) and Data Memory (RAM).

- The Program Memory of the 8051 Microcontroller is used for storing the program to be executed i.e., instructions.
- The Data Memory on the other hand, is used for storing temporary variable data and intermediate results.

Program Memory (ROM) of 8051 Microcontroller

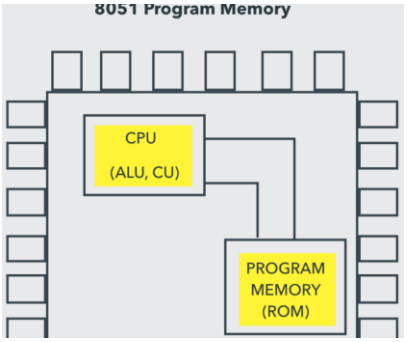
Internal Or External ROM Is The Program Memory

In 8051 Microcontroller, the code or instructions to be executed are stored in the Program Memory, which is also called as the ROM of the Microcontroller. The original 8051 Microcontroller by Intel has 4KB of internal ROM.

- In case of 4KB of Internal ROM, the address space is 0000H to 0FFFH. If the address space i.e., the program addresses exceed this value, then the CPU will automatically fetch the code from the external Program Memory.

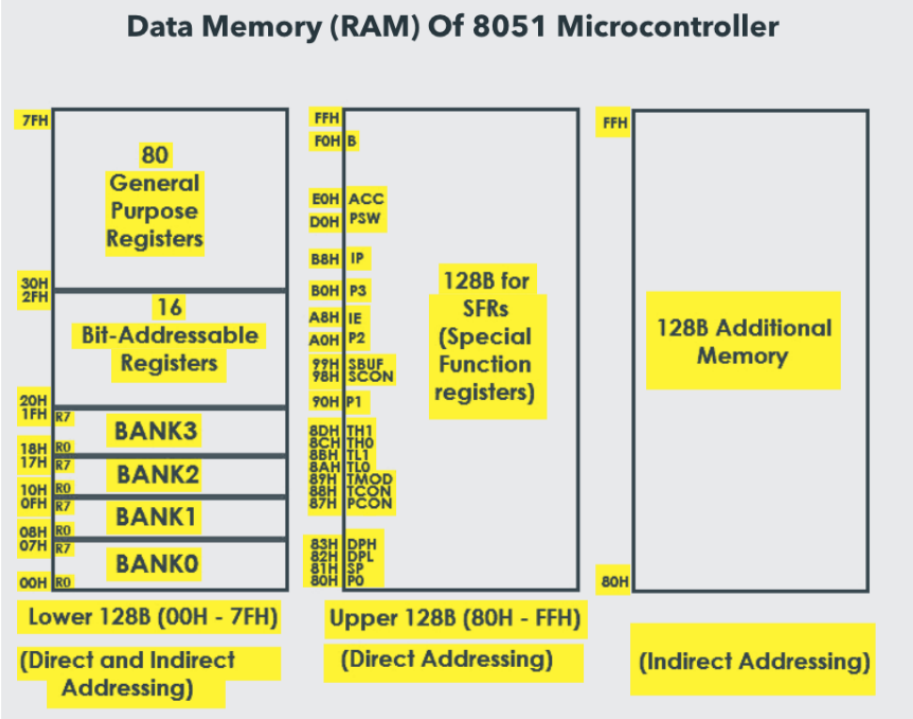
- For this, the External Access Pin (EA Pin) must be pulled HIGH i.e., when the EA Pin is high, the CPU first fetches instructions from the Internal Program Memory in the address range of 0000H to 0FFFFH

- and if the memory addresses exceed the limit, then the instructions are fetched from the external ROM in the address range of 1000H to FFFFH.



Data Memory (RAM),

- The Data Memory or RAM of the 8051 Microcontroller stores temporary data and intermediate results that are generated and used during the normal operation of the microcontroller. Original Intel's 8051 Microcontroller had 128B of internal RAM.
- But almost all modern variants of 8051 Microcontroller have 256B of RAM. In this 256B, the first 128B i.e., memory addresses from 00H to 7FH is divided in to Working Registers (organized as Register Banks), Bit – Addressable Area and General Purpose RAM (also known as Scratchpad area).
- In the first 128B of RAM (from 00H to 7FH), the first 32B i.e., memory from addresses 00H to 1FH consists of 32 Working Registers that are organized as four banks with 8 Registers in each Bank.



- The 4 banks are named as Bank0, Bank1, Bank2 and Bank3. Each Bank consists of 8 registers named as R0 – R7. Each Register can be addressed in two ways: either by name or by address.
- To address the register by name, first the corresponding Bank must be selected. In order to select the bank, we have to use the RS0 and RS1 bits of the Program Status Word (PSW) Register (RS0 and RS1 are 3rd and 4th bits in the PSW Register).
- When addressing the Register using its address i.e., 12H for example, the corresponding Bank may or may not be selected. (12H corresponds to R2 in Bank2).
- The next 16B of the RAM i.e., from 20H to 2FH are Bit – Addressable memory locations. There are totally 128 bits that can be addressed individually using 00H to 7FH or the entire byte can be addressed as 20H to 2FH.
- For example 32H is the bit 2 of the internal RAM location 26H.
- The final 80B of the internal RAM i.e., addresses from 30H to 7FH, is the general purpose RAM area which are byte addressable.
- These lower 128B of RAM can be addressed directly or indirectly.

- The upper 128B of the RAM i.e., memory addresses from 80H to FFH is allocated for Special Function Registers (SFRs). SFRs control specific functions of the 8051 Microcontroller. Some of the SFRs are I/O Port Registers (P0, P1, P2 and P3), PSW (Program Status Word), A (Accumulator), IE (Interrupt Enable), PCON (Power Control), etc.

Q14. Explain SFR REGISTER OF 8051

ANSWER:

The upper 128B of the RAM i.e., memory addresses from 80H to FFH is allocated for Special Function Registers (SFRs). SFRs control specific functions of the 8051 Microcontroller. Some of the SFRs are I/O Port Registers (P0, P1, P2 and P3), PSW (Program Status Word), A (Accumulator), IE (Interrupt Enable), PCON (Power Control), etc.

Name of the Register	*Function*	*Internal RAM Address (HEX)*
ACC	Accumulator	E0H
B	B Register (for Arithmetic)	F0H
DPH	Addressing External Memory	83H
DPL	Addressing External Memory	82H
IE	Interrupt Enable Control	A8H
IP	Interrupt Priority	B8H
P0	PORT 0 Latch	80H
P1	PORT 1 Latch	90H
P2	PORT 2 Latch	A0H
P3	PORT 3 Latch	B0H
PCON	Power Control	87H
PSW	Program Status Word	D0H
SCON	Serial Port Control	98H
SBUF	Serial Port Data Buffer	99H
SP	Stack Pointer	81H
TMOD	Timer / Counter Mode Control	89H
TCON	Timer / Counter Control	88H
TL0	Timer 0 LOW Byte	8AH
TH0	Timer 0 HIGH Byte	8CH
TL1	Timer 1 LOW Byte	8BH
TH1	Timer 1 HIGH Byte	8DH

SRFs Memory addresses are only direct addressable. Even though some of the addresses between 80H and FFH are not assigned to any SFR, they cannot be used as additional RAM area.

Q16.DISCUSS instruction sets of 8051

Answer:-

Arithmetic Instructions of 8051 μC

✓ Addition instructions

- ADD – It will Add Accumulator data with 8 bits & stores result into A.

```
ADD A, #50H     ; A ← A + 50H
ADD A, R1       ; A ← A + R1
ADD A, 17H      ; A ← A + [17H]
ADD A, @R1      ; A ← A + [R1]
```

- ADDC – It will Add Accumulator data with 8 bits along with Carry & stores result into A.

```
ADDC A, #50H    ; A ← A + 50H + Carry
ADDC A, R1      ; A ← A + R1 + Carry
ADDC A, 17H     ; A ← A + [17H] + Carry
ADDC A, @R1     ; A ← A + [R1] + Carry
```

✓ Subtraction instructions

- SUBB – It will Sub Accumulator data with 8 bits along with Carry & stores result into A.

```
SUBB A, #50H    ; A ← A - 50H - Carry
SUBB A, R1      ; A ← A - R1 - Carry
SUBB A, 17H     ; A ← A - [17H] - Carry
SUBB A, @R1     ; A ← A - [R1] - Carry
```

✓ Increment instructions

- INC – It will increment Register, Pointer or data of memory locations.

```
INC A     ; A ← A + 1
INC R1    ; R1 ← R1 + 1
INC 25H   ; [25H] ← [25H] + 1
INC @R1   ; [R1] ← [R1] + 1
INC DPTR; DPTR ← DPTR + 1
```

✓ Decrement instructions

- DEC – It will decrement Register, Pointer or data of memory locations.

```
DEC A     ; A ← A - 1
DEC R1    ; R1 ← R1 - 1
DEC 25H   ; [25H] ← [25H] - 1
DEC @R1 ; [R1] ← [R1] - 1
DEC DPTR          ; Does not exists
```

Arithmetic Instructions of 8051 μC

✓ Multiplication instruction

- MUL AB – It will Multiply A and B and Answer will be stored in BA where B hold higher Byte and A hold Lower Byte.

```
MUL AB   ; (B_Higher Byte A_Lower Byte) ← A X B
```

✓ Division instruction

- DIV AB – It will Divide A by B and Answer remainder will be stored in B & Quotient will be stored in A.

```
DIV AB   ; (B_Remainder A_Quotient) ← A / B
```

✓ Decimal Adjustment instruction

- DA A – It will be used after ADD instruction.
- It is used to convert that given Hex addition in BCD addition.
- Normal Addition is done by ADD that is Binary or Hex Addition.
- After when you use DA A, it adjust that addition in BCD form.
- DA A performs following adjustments to show given addition in BCD addition.
- If Lower Nibble > 9 or Auxiliary Carry is 1 then Add 06H with A
- If Higher Nibble > 9 or Carry is 1 then Add 60H with A

Example:

```
ADD A, R1       ; A ← A + R1
DA A            ; BCD Addition Adjustment
```

		AC = 1
A = 56H	A = 36H	A = 38H
R1 = 23H	R1 = 29H	R1 = 29H
A = 79H	A = 5FH	A = 61H
	06H	06H
	A = 65H	A = 67H

		1 1
A = 60H	A = 99H	
R1 = 70H	R1 = 99H	
A = D0H	A = 32H	
60H	06H	
A = 30H	60H	
Carry = 1	AL = 98H	
	Carry = 1	

Branch Instructions of 8051 μC

✓ Unconditional Jump instructions

- SJMP Label – It jump to location with respect to Label {8 bits}.
- Range of SJMP is -128 to +127 locations.
- Final Address will be PC = PC + Label
- AJMP Label – It jump to location with respect to Label {8 bits}.
- Range of AJMP is 2KB.
- Final Address will be PC = 1st 5 bits of PC + 3 bits of AJMP + Label.
- LJMP Label – It jump to location at Label {16 bits}.
- It can jump anywhere in 64KB memory of 8051.
- Final Address will be PC = Label
- JMP @A+DPTR – It will jump to the location A + DPTR.

✓ Boolean Conditional Jump instructions {All are SJMP}

- JB P0.0, Label – Jump to Label Only if P0.0 = 1
- JNB P0.0, Label – Jump to Label Only if P0.0 = 0
- JBC P0.0, Label – Jump to Label Only if P0.0 = 1 and also make P0.0 = 0

✓ Conditional Jump instructions {All are SJMP}

- DJNZ R3, Label – It will decrement R3, and jump to the Label only if R3 is not Zero.
- DJNZ 25H, Label – It will decrement [25H], and jump to the Label only if [25H] is not Zero.
- CJNE A, #25H, Label – It will compare A with #25H and jump to the Label only if A and #25H are not equal.
- CJNE A, 25H, Label – It will compare A with [25H] and jump to the Label only if A and [25H] are not equal.
- CJNE R2, #25H, Label – It will compare R2 with #25H and jump to the Label only if R2 and #25H are not equal.
- CJNE @R2, #25H, Label – It will compare [R2] with #25H and jump to the Label only if [R2] and #25H are not equal.
- JC Label – It will jump to Label if with previous instruction, carry flag is 1.
- JNC Label – It will jump to Label if with previous instruction, carry flag is 0.
- JZ Label – It will jump to Label if with previous instruction, Zero flag is 1.
- JNZ Label – It will jump to Label if with previous instruction, Zero flag is 0.

Example:-Steps to Program in timer:

- To generate a time delay, using the timers mode 1, the following steps are taken.
- 1)Load the TMOD value register indicating which timer is to be used and which timer mode (0 or 1) is selected.
- 2)Load register TL and TH with initial count values. 3)Start the timer.
- 4)Keep monitoring the timer raised. Get out if the loop when TF becomes high.
- 5)Stop the timer.
- 6)Clear the TF flag for the next round.
- 7)Go back to step 2 to load TH and TL again.

Example 9.4 : *If the crystal frequency is 12 MHz, find the counts we need to load into the timer registers, if a required time delay is 5 ms. Also write a program to create a pulse having width of 5 ms on P1.5 using timer 0.*

Solution : Crystal frequency = 12 MHz

$$\therefore \quad T = \frac{12}{12 \times 10^{-6}} = 1\ \mu s.$$

i.e. the counter counts up every 1 μs. Out of many 1 μs intervals, we have to make a 5 ms pulse.

We need 5 ms/1 μs = 5000 clocks.

N = 65536 – 5000 = 60536 = EC78H

We have to load TH with ECH and TL with 78H.

Program :

```
        CLR P1.5          ; Clear P1.5
        MOV TMOD, #01     ; Timer 0, mode 1(16-bit mode)
START : MOV TL0, #78H     ; TL0 = 78H, Timer 0 lower byte
                            register
        MOV TH0, #0ECH    ; TH0 = 0ECH, Timer 0 higher byte
                            register
        SETB P1.5         ; SET P1.5 high
        SETB TR0          ; start timer 0
REPEAT : JNB TF0, REPEAT  ; Monitor timer flag 0 till it becomes 1
        CLR TR0           ; Stop timer 0.
        CLR TF0           ; Clear timer 0 flag.
```

4. Example: Blinking led with the help of timer using MODE1

```
ORG 0H
        MOV TMOD,#00000001B
BACK:   MOV TL0,#0F0H
        MOV TH0,#0FFH
        CPL P1.0
        ACALL DELAY
        SJMP BACK
DELAY:SETB TR0
HERE:   JNB TF0, HERE
        CLR TR0
        CLR TF0
        RET
END
```

- Write a program to generate a square wave if 50Hz frequency on pin P1.2. Assume that XTAL=11.0592MHz

```
ORG 0
        LJMP MAIN
ORG 000BH
        CPL P1.2
        MOV TL0,#00
        MOV TH0,#0DCH
        RETI
ORG 30H
MAIN:  MOV TM0D,#00000001B
        MOV TL0,#00
        MOV TH0,#0DCH
        MOV IE,#82H
        SETB TR0
HERE:  SJMP HERE
END
```

5. To Develop an ALP to accept the external interrupt and to toggle the LED

PROGRAM:

```
ORG 0000H
LJMP MAIN
ORG 0003H
CPL P1.0
acall delay
RETI
ORG 0013H
CPL P1.1
acall delay
RETI
MAIN:
MOV IE,#85H
again:
mov p2,#00h
mov a,#01h
mov p2,a
acall delay
mov a,#07h
mov p2,a
acall delay
acall delay
acall delay
mov a,#0fh
mov p2,a
acall delay
acall delay
acall delay
SJMP again
delay:
mov r0,#0ffh
here1: mov r1,#0ffh
here: djnz r1,here
djnz r0,here1
ret
END
```

Write a program for the 8051 to transfer "YES" serially at 9600 baud, 8-bit data, 1 stop bit, do this continuously

Solution:

```
        MOV  TMOD,#20H  ;timer 1,mode 2(auto reload)
        MOV  TH1,#-3    ;9600 baud rate
        MOV  SCON,#50H  ;8-bit, 1 stop, REN enabled
        SETB TR1        ;start timer 1
AGAIN:  MOV  A,#"Y"     ;transfer "Y"
        ACALL TRANS
        MOV  A,#"E"     ;transfer "E"
        ACALL TRANS
        MOV  A,#"S"     ;transfer "S"
        ACALL TRANS
        SJMP AGAIN      ;keep doing it
;serial data transfer subroutine
TRANS:  MOV  SBUF,A     ;load SBUF
HERE:   JNB  TI,HERE    ;wait for the last bit
        CLR  TI         ;get ready for next byte
         RET
```

5. Example: Blinking led with the help of timer using MODE2-Auto reload mode

```
ORG 0H
        MOV TMOD,#00100000B
        MOV TH1, #05H
        SETB TR1
HERE:   JNB TF1, HERE
        CPL P2.0
        CLR TF1
        SJMP HERE
        END
```

Example Programs

Write a program to copy the value 55H into RAM memory locations 40H to 41H using
(a) direct addressing mode, (b) register indirect addressing mode without a loop, and (c) with a loop

Solution:

(a)

```
    MOV A,#55H   ;load A with value 55H
    MOV 40H,A    ;copy A to RAM location 40H
    MOV 41H.A    ;copy A to RAM location 41H
```

(b)

```
    MOV A,#55H   ;load A with value 55H
    MOV R0,#40H ;load the pointer. R0=40H
    MOV @R0,A    ;copy A to RAM R0 points to
    INC R0       ;increment pointer. Now R0=41h
    MOV @R0,A    ;copy A to RAM R0 points to
```

(c)

```
        MOV A,#55H     ;A=55H
        MOV R0,#40H    ;load pointer.R0=40H,
        MOV R2,#02     ;load counter, R2=3
AGAIN:  MOV @R0,A      ;copy 55 to RAM R0 points to
        INC R0         ;increment R0 pointer
        DJNZ R2,AGAIN ;loop until counter = zero
```

Write a program to clear 16 RAM locations starting at RAM address 60H

Solution:

```
        CLR A          ;A=0
        MOV R1,#60H    ;load pointer. R1=60H
        MOV R7,#16     ;load counter, R7=16
AGAIN:  MOV @R1,A      ;clear RAM R1 points to
        INC R1         ;increment R1 pointer
        DJNZ R7,AGAIN ;loop until counter=zero
```

Write a program to copy a block of 10 bytes of data from 35H to 60H

Solution:

```
      MOV R0,#35H  ;source pointer
      MOV R1,#60H  ;destination pointer
      MOV R3,#10   ;counter
BACK: MOV A,@R0    ;get a byte from source
      MOV @R1,A    ;copy it to destination
      INC R0       ;increment source pointer
      INC R1     ;increment destination pointer
      DJNZ R3,BACK ;keep doing for ten bytes
```

Assume that register A has packed BCD, write a program to convert packed BCD to two ASCII numbers and place them in R2 and R6.

```
MOV    A,#29H   ;A=29H, packed BCD
MOV    R2,A     ;keep a copy of BCD data
ANL    A,#0FH   ;mask the upper nibble (A=09)
ORL    A,#30H   ;make it an ASCII, A=39H('9')
MOV    R6,A     ;save it
MOV    A,R2     ;A=29H, get the original
data
ANL    A,#0F0H ;mask the lower nibble
RR     A        ;rotate right
RR     A        ;rotate right
RR     A        ;rotate right
RR     A        ;rotate right
ORL    A,#30H   ;A=32H, ASCII char. '2'
MOV    R2,A     ;save ASCII char in R2
```

SWAP A

Write a program for the 8051 to transfer letter "A" serially at 4800 baud, continuously.

Solution:

```
        MOV  TMOD,#20H  ;timer 1,mode 2(auto reload)
        MOV  TH1,#-6    ;4800 baud rate
        MOV  SCON,#50H  ;8-bit, 1 stop, REN enabled
        SETB TR1        ;start timer 1
AGAIN:  MOV  SBUF,#"A"  ;letter "A" to transfer
HERE:   JNB  TI,HERE    ;wait for the last bit
        CLR  TI         ;clear TI for next char
        SJMP AGAIN      ;keep sending A
```

Write a program for the 8051 to transfer "YES" serially at 9600 baud, 8-bit data, 1 stop bit, do this continuously

Solution:

```
        MOV  TMOD,#20H  ;timer 1,mode 2(auto reload)
        MOV  TH1,#-3    ;9600 baud rate
        MOV  SCON,#50H  ;8-bit, 1 stop, REN enabled
        SETB TR1        ;start timer 1
AGAIN:  MOV  A,#"Y"     ;transfer "Y"
        ACALL TRANS
        MOV  A,#"E"     ;transfer "E"
        ACALL TRANS
        MOV  A,#"S"     ;transfer "S"
        ACALL TRANS
        SJMP AGAIN      ;keep doing it
;serial data transfer subroutine
TRANS:  MOV  SBUF,A     ;load SBUF
HERE:   JNB  TI,HERE    ;wait for the last bit
        CLR  TI         ;get ready for next byte
         RET
```

9.Example Program for Serial Communication to transfer word “KLU”

```
ORG 0H
        MOV TMOD,#00100000B------(20H)
        MOV TH1,#-3
        MOV SCON,#01010000B-------(50H)
        SETB TR1
AGAIN:MOV A,# ” “
        ACALL TRANSFER
       MOV A,# ”K“
        ACALL TRANSFER
        MOV A,# ”L“
        ACALL TRANSFER
        MOV A,# ”U“
        ACALL TRANSFER
        MOV A,# ” “
        ACALL TRANSFER
TRANSFER:  MOV SBUF,A
HERE: JNB TI, HERE
 CLR TI
 RET
```

```
SJMP AGAIN
END
ORG 00H
CLR A
MOV P1,A
REPEAT: CPL P1.0
ACALL DELAY1
SJMP REPEAT
DELAY1: MOV R0,#0FFH
BACK3: MOV R1,#0FFH
BACK2: MOV R2,#01H
BACK1 : DJNZ R2,BACK1
DJNZ R1,BACK2
DJNZ R0,BACK3
RET
END
```

Example.10 Program for BLINKIG LED

```
org 00h
clr A
```

```
        mov  p1.0 ,A
back: mov  p1.0 ,A
        acall delay
        cpl p1.0
        acall delay
        sjmp back
delay: mov r0,#5d
here1:mov r1,#2d
here:djnz r1,here
        djnz r0,here1
        ret
end
```