Microprocessors Module 5 Questions

Part A

1. Differentiate between Microprocessors and Microcontrollers.

Microprocessor	Microcontroller
CPU is standalone while RAM, ROM, I/O, Timer are separate	CPU, RAM, ROM, I/O and Timer are all on a single chip
Designer can decide on the amount of ROM, RAM, and I/O Ports	Fixed amount of on-chip ROM, RAM, I/O Ports

Expansive	Not Expansive
Versatile	Single Purpose
General Purpose (generally used in personal computers.)	Special Purpose (used in washing machines, and air conditioners.)
Expensive	Cheaper

2. List the IO ports available in 8051

- There are total of 4 ports for I/O operation in 8051 microcontroller
- The four ports are P0,P1,P2, and P3. Each use 8 pins, to make the port 8 bit ports.

  Port 0: Used as i/o pins and serves as lower order addresses in external memory interfacing
  Port 1: Used as i/o pins
  Port 2: Used as i/o pins and serves as higher order addresses in conjunction with port 0.
- Port 3: Used as i/o pins + performing dual functions like TXD, INT0, INT1, WR, RD etc.
- When ‘0’ is written to a port, it becomes an o/p port
- To use as an i/p port ‘1’ must sent to the port

3. Draw and explain the format of program status word in 8051.

- The Program Status Word (PSW) is a key register in the 8051 microcontroller that holds various status bits indicating the current state of the processor.
- The PSW is an 8-bit register, and its format is as follows:

CY	AC	F0	RS1	RS0	OV	–	P

CY	PSW.7	Carry flag.
AC	PSW.6	Auxiliary carry flag.
F0	PSW.5	Ayailable 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 bit.
P	PSW.0	Parity flag. Set/cleared by hardware each instuction cycle to indicate an odd/even number of 1 bits in the accumulator.

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

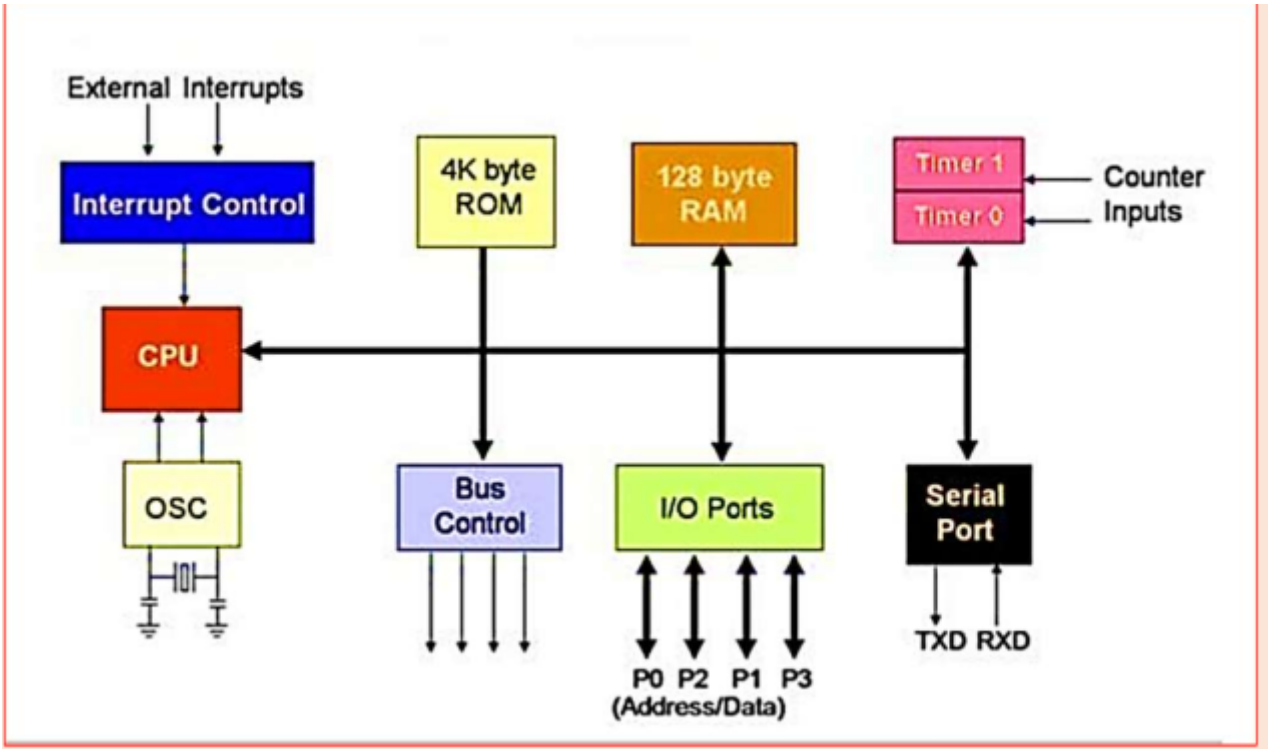
4. Write an assembly language program to compute x to the power n where both x and n are 8 bit numbers given by user and result should not be more than 16 bits

```
LOOP1:
ORG 0000H           // Origin of the program at address 0000H
MOV A, #02H         // Load accumulator A with the base value (x)
MOV B, #03H         // Load register B with the exponent value (n)
MOV R0, B           // Copy the exponent value to register R0 (RO)
MOV R1, A           // Copy the base value to register R1 (R1)
MOV R2, #01H        // Initialize register R2 with the constant value 01H for
multiplication

// Loop for exponentiation using multiplication
MOV A, R2           // Move the value in R2 to accumulator A for
multiplication
MOV B, R1           // Move the value in R1 to register B for multiplication
MUL AB              // Multiply A and B, result in A:B (16-bit product)
DEC R0              // Decrement the exponent counter (RO)
MOV R2, A           // Move the lower byte of the product to R2 for next
multiplication

CJNE R0, #00H, LOOP1  // Compare R0 with 00H, if not equal, jump to LOOP1
for the next multiplication
```

```
MOV A, R2          // Move the final result from R2 to accumulator A
END
```

Part B

1. Explain the internal architecture of 8051 with a neat diagram.



The internal architecture of 8051 contains the following blocks

- Accumulator (A)
  - Saves operand for operations by ALU
- B Register(B)
  - Saves second Operand for ALU
- PSW (Process Status Word)
  - 8 bit register to save the status information
- Stack Pointer (SP)
  - 8 bit register is incremented before data is stored onto the stack using PUSH Instruction
- Data Pointer (DPTR)
  - Holds external data memory address of data being fetched or to be fetched
- Port P0

- Port P1
- Port P3
- Serial Data Buffer (SBUF)
    - Contains 2 independent registers
    - Transmit Buffer
        - Parallel in Serial out register
    - Receive buffer
        - Serial in Parallel out register
- Timer register
    - Two 16 bit register, can be accessed as their lower and upper bytes
        - TL0 -> Lower byte of timing register 0
        - TH0 -> Higher byte of timing register 0
        - TL1 ->Lower byte of timing register 1
        - TH1 -> Higher byte of timing register 1
- Control Registers
    - IP - interrupt Priority Register
    - IE - Interrupt Enable register
    - TMOD - Timer Mode Register
    - TCON - Timer Control Register
    - SCON - Serial Control Register
    - PCON - Power Control Register
- Timing and control unit
    - This unit derives all the necessary timing and control signals required for the internal operation of the circuit.
- Oscillator
    - This circuit generate the basic timing clock signals for the operation of the circuit using crystal oscillator
- Instruction Register:
    - This register decodes the opcode of an instruction to be executed
- RAM
    - RAM is 128 byte memory for the read and write and is indirectly and directly addressable.
- ROM
- ALU
- SFR Register Bank
    - 4 Register Banks each of 8 registers

2. List any four addressing modes supported by 8051 microcontrollers, with one example each

8051 supports 6 addressing modes:

1. **Immediate Addressing:** Data is immediately available in the instruction.
   For example -

ADD A, #77	Adds 77 (decimal) to A and stores in A
MOV DPTR, #1000H	Moves 1000 (hexadecimal) to the data pointer

2. **Register Addressing:** This way of addressing accesses the bytes in the current register bank. Data is available in the register specified in the instruction. The register bank is decided by 2 bits of Processor Status Word (PSW).
   For example -

ADD A, R0	Adds content of R0 to A and stores in A

3. **Direct Addressing:** The address of the data is available in the instruction.
   For example -

MOV A, 088H	Moves the content of SFR TCON to A (088H is the address of special function register TCON in 8051)

4. **Register Indirect Addressing:** The address of data is available in the R0 or R1 registers as specified in the instruction.
   For example -

MOV A, @R0	Moves the content of the address pointed by R0 to A

5. **Register-specific Addressing:** The operand is implicitly specified using one of the registers ie, such instructions operate only on a specific register.
   For example -

RLA	This instruction rotates accumulator left

6. **Indexed Addressing:** Only program memory can be accessed using this mode. It is mainly used for look-up table manipulations.
   For example -

MOVC A, @A+DPTR	DPTR has the base address of LUT and A has the relative (offset) address. The LUT content at the effective address obtained by adding the base address and relative address is moved to ACC (accumulator)

3. State the name and purpose of any 6 special function registers (SFRs) of 8051 microcontroller

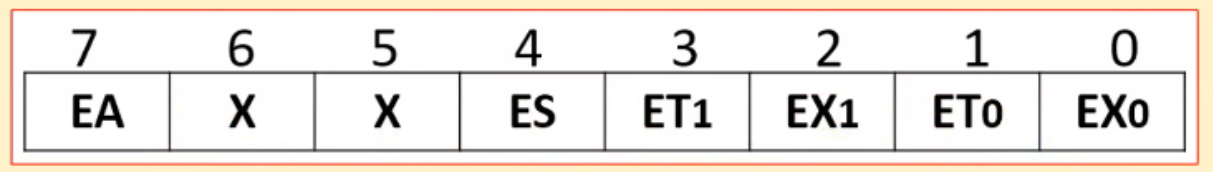
1. P0 (Port 0):
   Purpose: P0 is an 8-bit bidirectional I/O port. It can be used to interface with external devices or as general-purpose I/O pins.
2. P1 (Port 1):
   Purpose: Similar to P0, P1 is an 8-bit bidirectional I/O port. It also has additional functions such as external interrupt and timer input.
3. P2 (Port 2):
   Purpose: P2 is an 8-bit bidirectional I/O port with additional control signals. It can be used for interfacing and controlling external devices.
4. P3 (Port 3):
   Purpose: P3 is an 8-bit bidirectional I/O port, and like P1 and P2, it can be used to interface with external devices. Additionally, P3 has specific control signals for external interrupts.
5. TCON (Timer Control):
   Purpose: TCON is the Timer Control register. It is used to control and monitor the status of Timer 0 and Timer 1, including their modes and interrupt flags.
6. TMOD (Timer Mode):
   Purpose: TMOD is the Timer Mode register. It is used to set the operating modes of Timer 0 and Timer 1. It allows configuration of timers for different counting and timing modes.

5. Explain the interrupt and stack structure of 8051.

Interrupt

- Interrupt is a signal sent by external device to the processor, to request the processor to perform a particular task or work.
- When a processor recognizes an interrupt, it saves processor status in stack
  - Then it calls and executes interrupt Service Routine (ISR)
- At the end of ISR, it restores processor status and program control is transferred to main program
- Types of interrupts are
  - Software and Hardware interrupt
  - Vectored and non vectored interrupt
  - Maskable and non maskable interrupt
- Types of interrupt in 8051
  - Timer 0 overflow interrupt TF0
  - Timer 1 overflow Interrupt TF1
  - External hardware interrupt INT0
  - Internal hardware interrupt INT1
  - Serial communication interrupt RI/TI
- There are 2 Special function registers (SFRs) for interrupt handling
  - Interrupt Enable Register
  - Interrupt Priority Register

Interrupt Enable Register

- Responsible for enabling and disabling interrupt
- Bit addressable register
  - 

7	6	5	4	3	2	1	0
EA	X	X	ES	ET1	EX1	ET0	EX0

  - Each of these determines Enable/Disable of the below registers
  - EX0 -> External Interrupt 0
  - ET0 -> Timer 0 overflow interrupt
  - EX1 -> External interrupt 1
  - ET1 -> Timer 1 overflow interrupt
  - ES -> Serial port interrupt
  - EA -> All interrupts
  - Here positions 6 and 5 are not used

Interrupt Priority Register

- Its possible of changing priority levels of interrupts by setting or clearing corresponding bits in the IP register

TCON Register

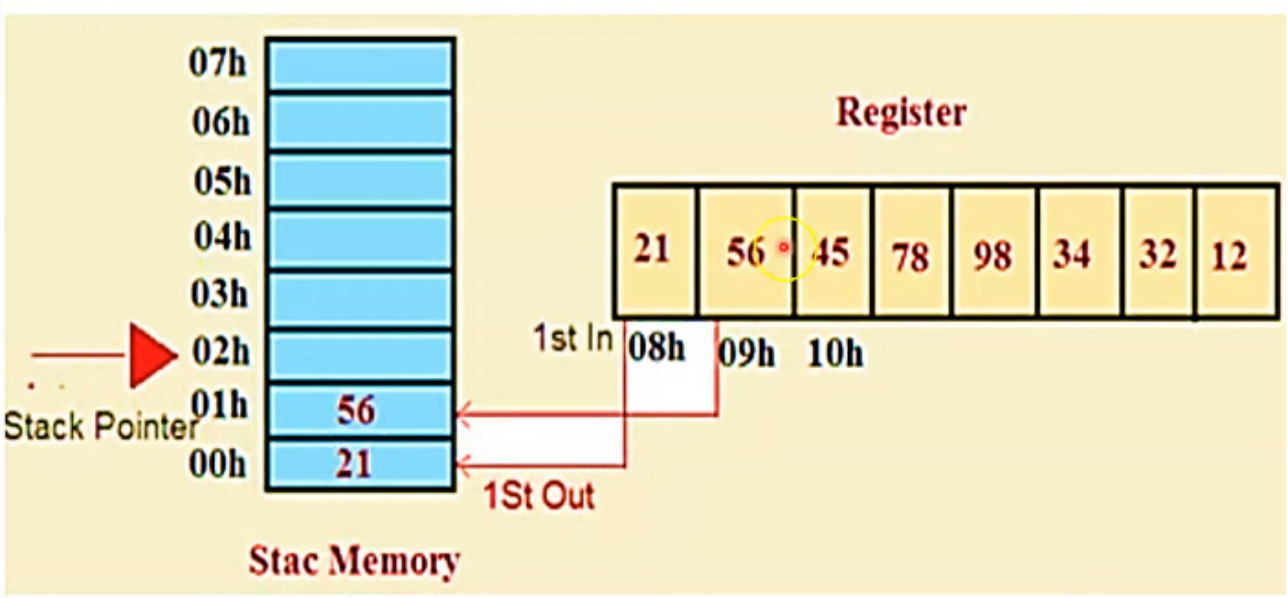
- Specifies the type of external interrupt to 8051 microcontroller

Stack

- Stack is a area of RAM allocated to hold temporarily all parameters of the variable
- Whenever the function is called parameters and local variables are added to stack
- When Function returns, Parameters and variables are removed from stack
- Register used to access the stack is called stack pointer register
- When we push something into stack memory, the stack pointer increases

PUSH

- Used to take values from any register and storing in starting address of stack pointer



PUSH operation of Stack

- Consider this code

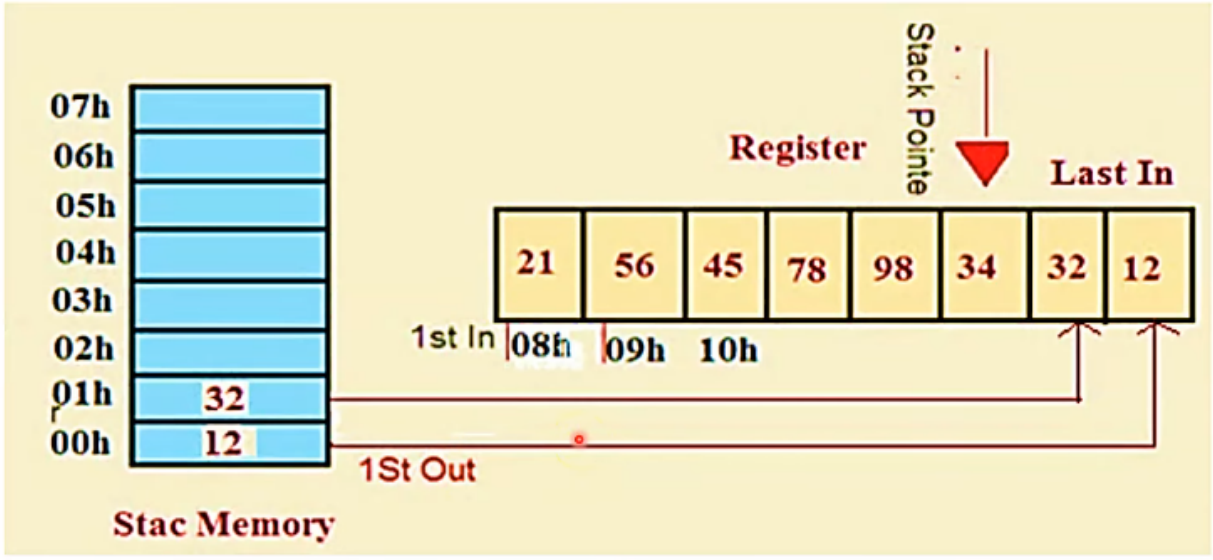
```
Eg:  0000H
     MOV  08H,  #21H
     MOV  09H,  #56H
     PUSH 00H
     PUSH 01H
     END
```

- 21 and 56 and moved to located 08 and 09 in the register
- PUSH 00H
  - Stack pointer is initially at 08H
  - The value at stack pointer is pushed to Stack
    - Here the value at stackpointer (08H) is 21
    - 21 is pushed to stack
  - After pushing, stack pointer is incremented by 1
    - Now its value is 09H
- PUSH 01H
  - Stack pointer is now at 09H
  - The value at stack pointer is pushed to Stack

- Here the value at stackpointer (09H) is 56
- 56 is pushed to stack

POP

- Used for placing values from stack pointers maximum address to other register address
- It decrements the stack pointer by 1



POP Operation in Stack

-
```
Eg:   0000H
      MOV  00H,  #12H
      MOV  01H,  #32H
      POP 1FH
      POP 0EH
      END
```

- In the above code
- First we move 12 and 32 to the stack memory locations 00H and 01H
- POP 1FH
  - Value at top of the stack is popped
    - Here its 12, located at 00H
  - This popped value is stored at 1FH
  - Stack pointer is decremented

- POP 0EH
  - Value at top of the stack is popped
    - Here its 32, located at 01H
  - This popped value is stored at 0EH

6. Write an assembly language program for 8051 to perform addition of two 2x2 matrices

```
ORG 0000H          ; Program origin at address 0000H

MOV R0, #20H       ; Initialize R0 with the address of the first element of
matrix 1
MOV R1, #30H       ; Initialize R1 with the address of the first element of
matrix 2
MOV R3, #00H       ; Initialize R3 to store the temporary sum
MOV R4, #04H       ; Initialize R4 as a counter for the number of elements
(2x2 matrix)

AGAIN:
    ; Load the value from the current position in matrix 1 into register A
    MOV A, @R0

    ; Move the value in register A to R3 for temporary storage
    MOV R3, A

    ; Load the value from the current position in matrix 2 into register A
    MOV A, @R1

    ; Add the value in R3 (from matrix 1) to the value in register A (from
matrix 2)
    ADD A, R3

    ; Store the result back in the current position in matrix 1
    MOV @R0, A

    ; Decrement the counter
    DEC R4

    ; Increment the pointers to move to the next element in both matrices
    INC R0
    INC R1

    ; Check if all elements have been processed
    CJNE R4, #00H, AGAIN
```

```
END                 ; End of the program
```

7. Write an assembly language program for 8051 to find the transpose of a 2x2 matrix

```
ORG 0000H           ; Program origin at address 0000H

MOV R0, #20H        ; Initialize R0 with the address of the first element of
the matrix
MOV R1, #30H        ; Initialize R1 with the address of the transposed matrix

; Copy the first element of the original matrix to the transposed matrix
MOV A, @R0          ; Load the value from the current position in the original
matrix into register A
MOV @R1, A          ; Store the value in the transposed matrix at the current
position

INC R0              ; Move to the next element in the original matrix
INC R1              ; Move to the next column in the transposed matrix

INC R1              ; Move to the next row in the transposed matrix
MOV A, @R0          ; Load the value from the current position in the original
matrix into register A
MOV @R1, A          ; Store the value in the transposed matrix at the current
position

INC R0              ; Move to the next element in the original matrix
DEC R1              ; Move back to the first column in the transposed matrix
INC R1              ; Move to the next row in the transposed matrix

MOV A, @R0          ; Load the value from the current position in the original
matrix into register A
MOV @R1, A          ; Store the value in the transposed matrix at the current
position

INC R0              ; Move to the next element in the original matrix
INC R1              ; Move to the next column in the transposed matrix

INC R1              ; Move to the next row in the transposed matrix
MOV A, @R0          ; Load the value from the current position in the original
matrix into register A
MOV @R1, A          ; Store the value in the transposed matrix at the current
position
```

```
END             ; End of the program
```