**WINTER– 18 EXAMINATION**

**Subject Name: Microcontroller** **Model Answer** Subject Code: 17534

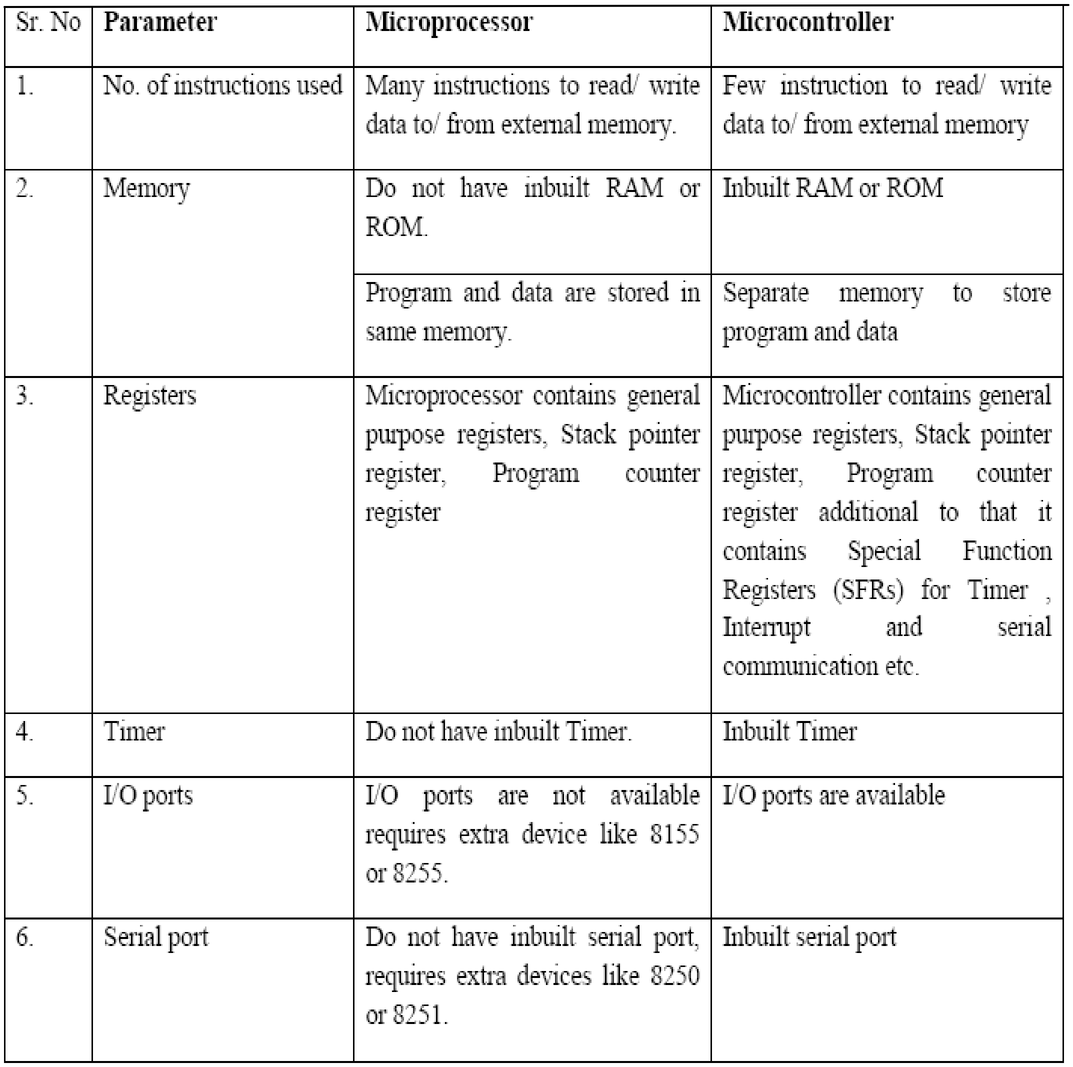
**Important Instructions to examiners:**

1)The answers should be examined by key words and not as word-to-word as given in the model answer scheme.
2)The model answer and the answer written by candidate may vary but the examiner may try to assess the understanding level of the candidate.
3)The language errors such as grammatical, spelling errors should not be given more Importance (Not applicable for subject English and Communication Skills.
4)While assessing figures, examiner may give credit for principal components indicated in the figure. The figures drawn by candidate and model answer may vary. The examiner may give credit for any equivalent figure drawn.
5)Credits may be given step wise for numerical problems. In some cases, the assumed constant values may vary and there may be some difference in the candidate’s answers and model answer.
6)In case of some questions credit may be given by judgement on part of examiner of relevant answer based on candidate’s understanding.
7)For programming language papers, credit may be given to any other program based on equivalent concept.

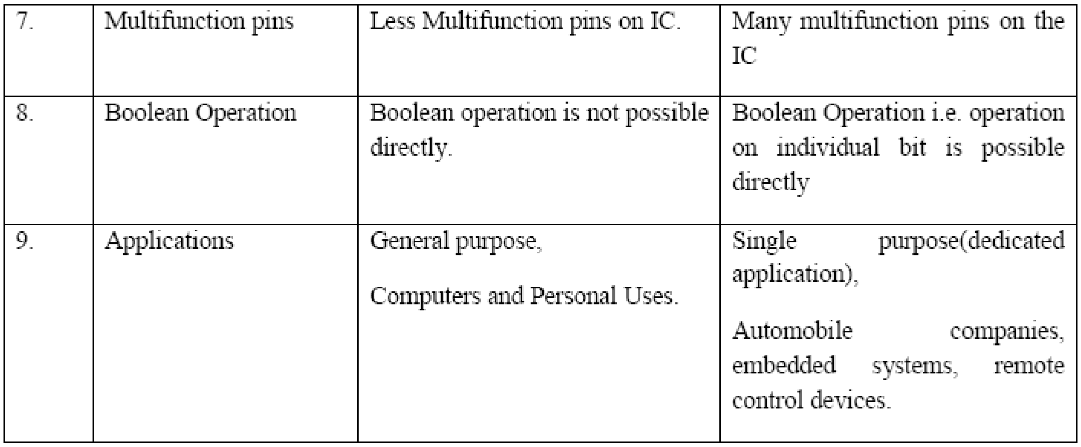
## 1 a) Attempt any THREE of the following: 12

**i) Compare microprocessor and microcontroller(any four points)**

**Ans: (any four points: 4M)**



| Sr. No | Parameter | Microprocessor | Microcontroller |
|---|---|---|---|
| 1. | No. of instructions used | Many instructions to read/ write data to/ from external memory. | Few instruction to read/ write data to/ from external memory |
| 2. | Memory | Do not have inbuilt RAM or ROM. | Inbuilt RAM or ROM |
| | | Program and data are stored in same memory. | Separate memory to store program and data |
| 3. | Registers | Microprocessor contains general purpose registers, Stack pointer register, Program counter register | Microcontroller contains general purpose registers, Stack pointer register, Program counter register additional to that it contains Special Function Registers (SFRs) for Timer , Interrupt and serial communication etc. |
| 4. | Timer | Do not have inbuilt Timer. | Inbuilt Timer |
| 5. | I/O ports | I/O ports are not available requires extra device like 8155 or 8255. | I/O ports are available |
| 6. | Serial port | Do not have inbuilt serial port, requires extra devices like 8250 or 8251. | Inbuilt serial port |



| | | | |
|---|---|---|---|
| 7. | Multifunction pins | Less Multifunction pins on IC. | Many multifunction pins on the IC |
| 8. | Boolean Operation | Boolean operation is not possible directly. | Boolean Operation i.e. operation on individual bit is possible directly |
| 9. | Applications | General purpose,<br>Computers and Personal Uses. | Single purpose(dedicated application),<br>Automobile companies, embedded systems, remote control devices. |

**ii) Explain RISC and CISC**

**Ans: 2M –each**

**RISC (*Reduced Instruction Set Computer)***

RISC stands for *Reduced Instruction Set Computer*. To execute each instruction, if there is separate electronic circuitry in the control unit, which produces all the necessary signals, this approach of the design of the control section of the processor is called RISC design. It is also called *hard-wired approach*.

**Examples of RISC processors:**

- IBM RS6000, MC88100
- DEC’s Alpha 21064, 21164 and 21264 processors

**Features of RISC Processors:**

The standard features of RISC processors are listed below:

- RISC processors use a small and limited number of instructions.
- RISC machines mostly uses hardwired control unit.
- RISC processors consume less power and are having high performance.
- Each instruction is very simple and consistent.
- RISC processors uses simple addressing modes.
- RISC instruction is of uniform fixed length.

**CISC (Complex Instruction Set Computer)**

CISC stands for *Complex Instruction Set Computer*. If the control unit contains a number of micro-electronic circuitry to generate a set of control signals and each micro-circuitry is activated by a micro-code, this design approach is called CISC design.

**Examples of CISC processors are:**

- Intel 386, 486, Pentium, Pentium Pro, Pentium II, Pentium III
- Motorola’s 68000, 68020, 68040, etc.

**Features of CISC Processors:**

The standard features of CISC processors are listed below:

- CISC chips have a large amount of different and complex instructions.
- CISC machines generally make use of complex addressing modes.
- Different machine programs can be executed on CISC machine.
- CISC machines uses micro-program control unit.
- CISC processors are having limited number of registers.

**iii) List alternate functions of port 3 of 8051 microcontroller.**

**Ans: (Each function ½ M)**

| Pin | Name | AlternateFunction |
|---|---|---|
| P3.0 | RXD | Serial input line |
| P3.1 | TXD | Serial output line |
| P3.2 | _<br>INT0 | External interrupt 0 |
| P3.3 | _<br>INT1 | External interrupt 1 |
| P3.4 | T0 | Timer0 external input |
| P3.5 | T1 | Timer0 external input |
| P3.6 | _<br>WR | External datamemorywritestrobe |
| P3.7 | _<br>RD | External datamemoryread strobe |

**iv) Explain software development cycle in microcontroller programming.**

**Ans: (2M–diagram, 2M- explanation)**

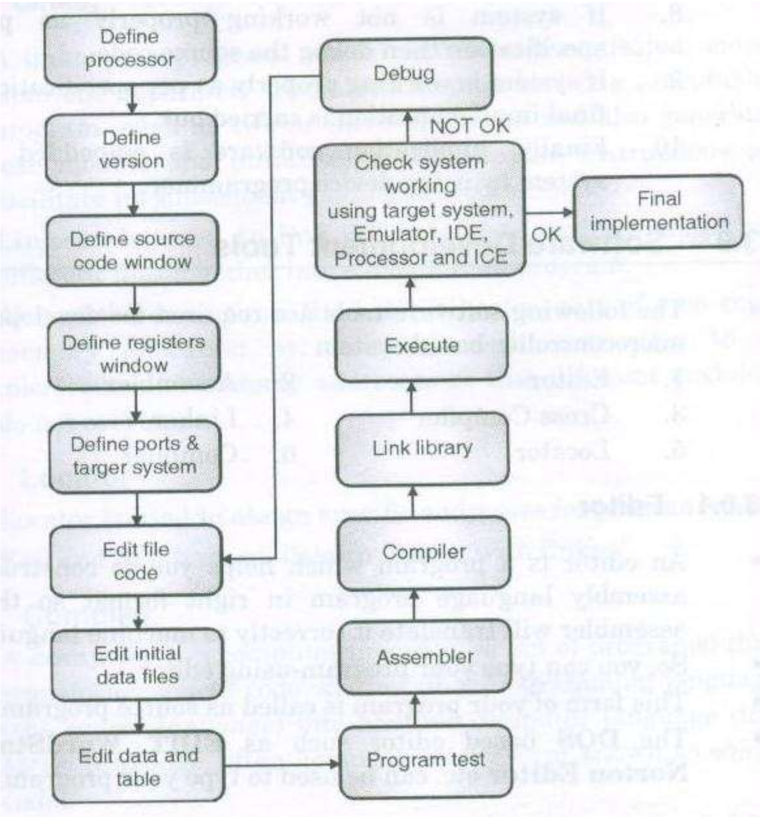


**Fig: software development cycle.**

**OR**

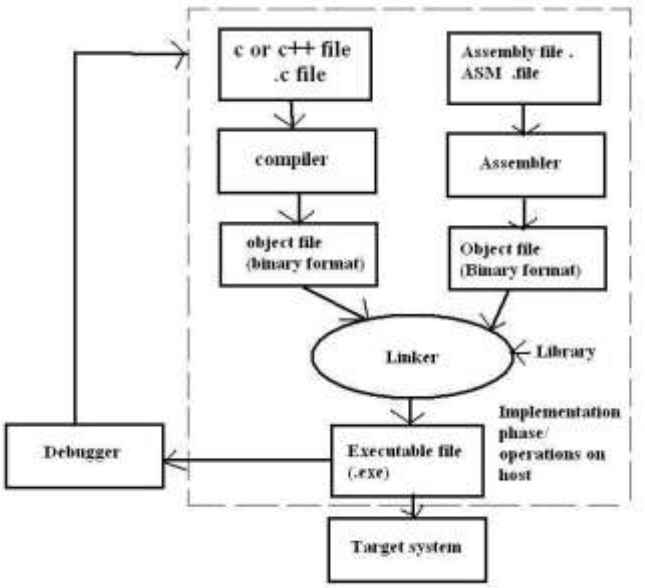
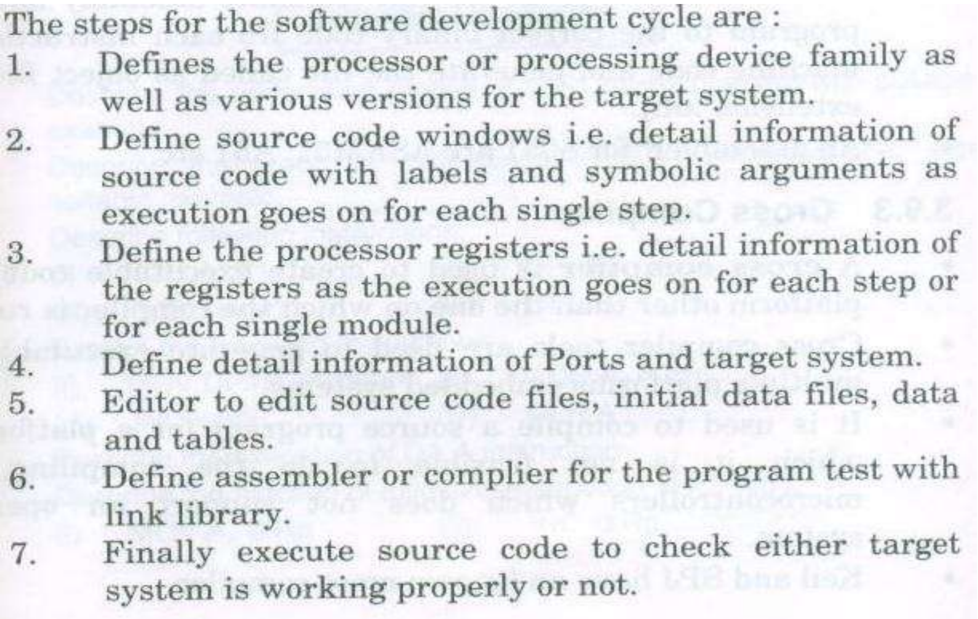

c or c++ file .c file
compiler
object file (binary format)
Assembly file . ASM .file
Assembler
Object file (Binary format)
Linker
Library
Executable file (.exe)
Implementation phase/ operations on host
Debugger
Target system

**Fig: software development cycle**



The steps for the software development cycle are :

1. Defines the processor or processing device family as well as various versions for the target system.
2. Define source code windows i.e. detail information of source code with labels and symbolic arguments as execution goes on for each single step.
3. Define the processor registers i.e. detail information of the registers as the execution goes on for each step or for each single module.
4. Define detail information of Ports and target system.
5. Editor to edit source code files, initial data files, data and tables.
6. Define assembler or complier for the program test with link library.
7. Finally execute source code to check either target system is working properly or not.

**v) Compare absolute decoding and linear decoding.**

**Ans: (1M- each point)**

| **Absolute Decoding** | **Linear Decoding** |
|---|---|
| 1. It is also called as full decoding as all the address lines are used for decoding. | It is also called as partial decoding as all address lines are not used for decoding |
| 2. It is used in large memory systems. | It is used in small systems |
| 3. Hardware required for decoding logic is more | Hardware used for decoding logic is eliminated. |
| 4. Multiple addresses are not generated | Multiple addresses are generated |

**b) Attempt any ONE of the following: 6**

**i)Write an assembly language program to find largest number from the array of five numbers stored in internal RAM memory.**

**Ans: (4M- correct prog, 2M- comments)**

**Program:** Select Bank 0

```
          MOV R1, #05H                 ; initialize the counter
          MOV R0, #40H                 ; initialize the memory pointer
          DEC R1                       ; decrement counter by one
          MOV A,@R0                    ; load number in accumulator
          MOV B, A                     ; move that number to register B
  UP:     INC R0                       ; increment the memory pointer
          MOV A,@R0                    ; read the next number in A
          CJNE A, B, DOWN    ; compare the first two numbers, if not equal go to DOWN
          AJMP NEXT                    ; else go to NEXT
          DOWN: JC NEXT                ; if number in A is greater then go to NEXT
          MOV B, A                     ; else move the number in register B
       NEXT:   DJNZ R1, UP   ; decrement the counter by one, if count ≠ 0, then go to UP
          INC R0                       ; increment the memory pointer
          MOV A,B
          MOV 50H, A                   ; store result at memory location 50H
HERE: SJMP HERE
```

**ii) Draw interfacing diagram of stepper motor with 8051 microcontroller. Write assembly language program to rotate stepper motor clockwise.**

**Ans:1M- Diagram,1M- step sequence, 3M- program, 1M- comments**

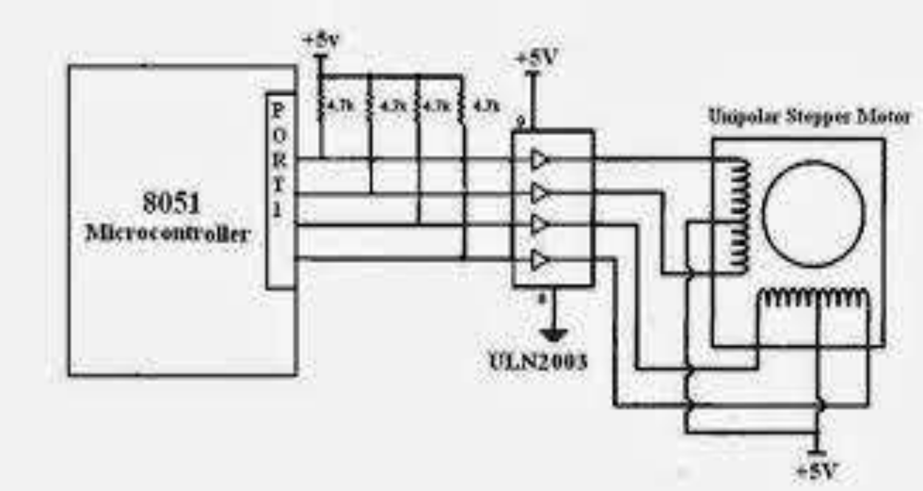
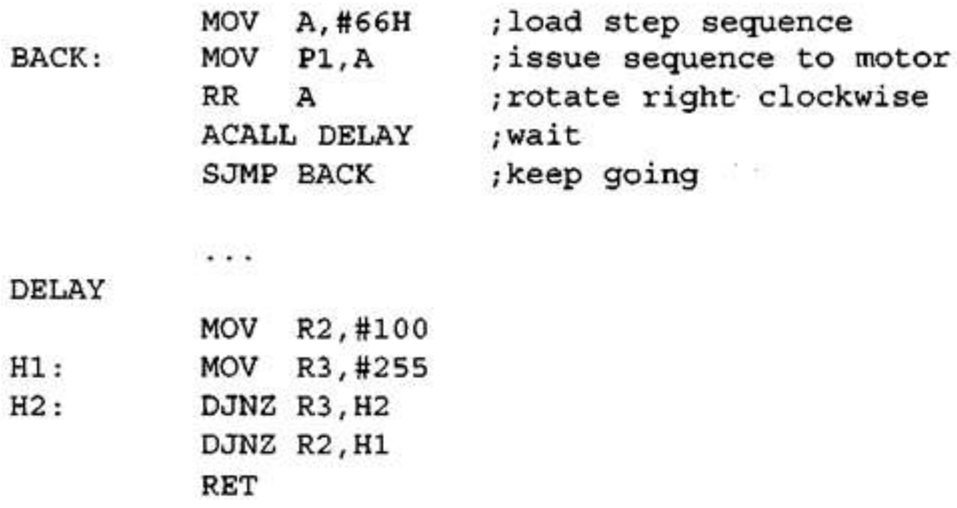


Fig: interfacing diagram of stepper motor with 8051 microcontroller

| Step no | Winding A | Winding B | Winding C | Winding D | Clockwise |
|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 1 | ↓ |
| 2 | 1 | 1 | 0 | 0 | |
| 3 | 0 | 1 | 1 | 0 | |
| 4 | 0 | 0 | 1 | 1 | |



```
          MOV   A,#66H     ;load step sequence
BACK:     MOV   P1,A       ;issue sequence to motor
          RR    A          ;rotate right clockwise
          ACALL DELAY      ;wait
          SJMP BACK        ;keep going

          ...
DELAY
          MOV   R2,#100
H1:       MOV   R3,#255
H2:       DJNZ R3,H2
          DJNZ R2,H1
          RET
```

## 2. Attempt any FOUR of the following: 16

**a) Draw the architecture of 8051 microcontroller.**

**Ans: (correct diagram –4 marks)**

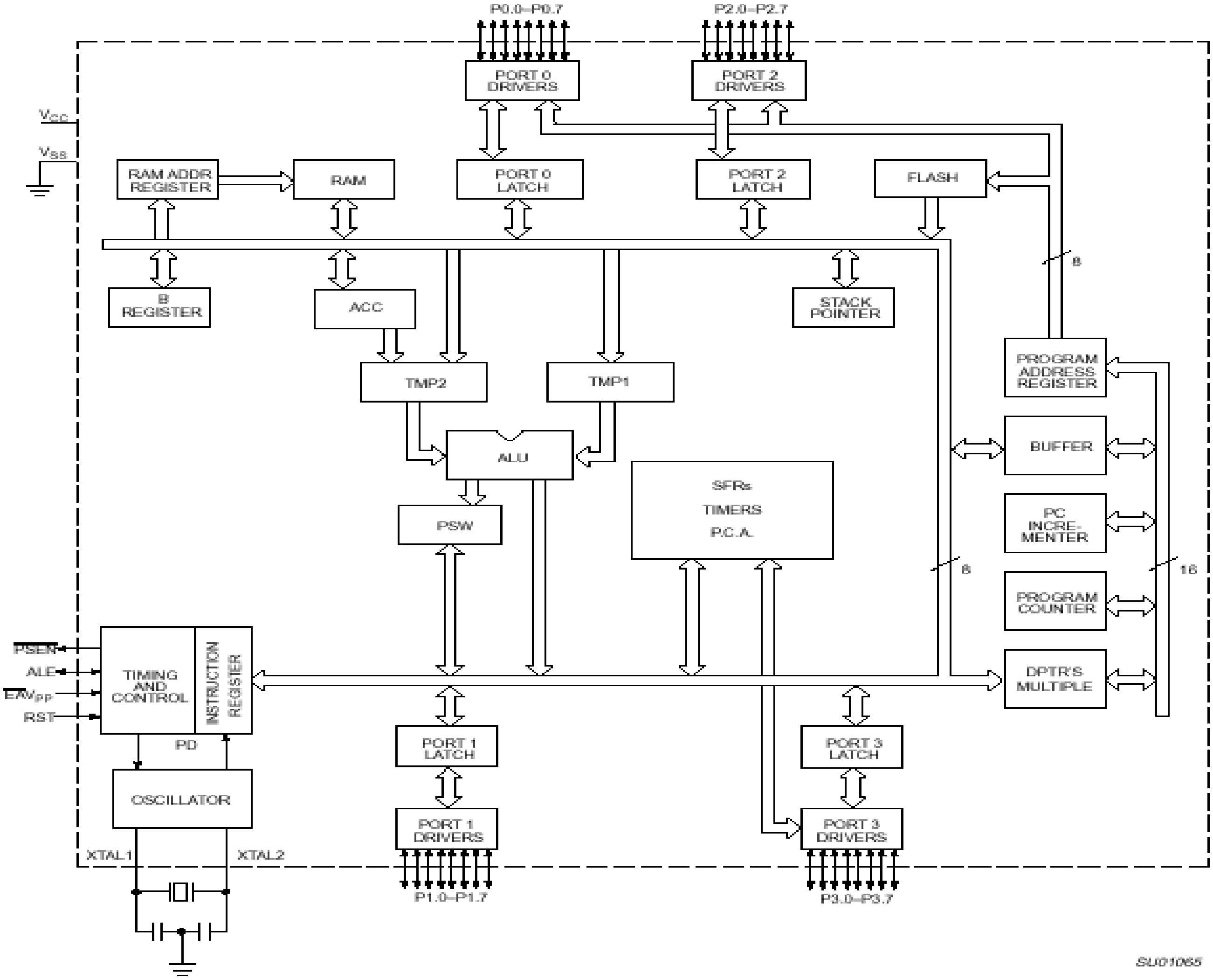


Fig: architecture of 8051 microcontroller.

**b) State the features of 8051 microcontroller.**

**Ans: (any correct four points – 1M each)**

Features of 8051 micro controller are as follows:-

1) 8- bit data bus and 8- bit ALU.

2) 16- bit address bus – can access maximum 64KB of RAM and ROM.

3) On- chip RAM -128 bytes (Data Memory‖)

4) On- chip ROM – 4 KB (Program Memory‖)

5) Four 8-bit bi- directional input/output ports Four 8-bit bi- directional input/ output ports.

6) Programmable serial ports i.e. One UART (serial port)

7) Two 16- bit timers- Timer 0& Timer 1

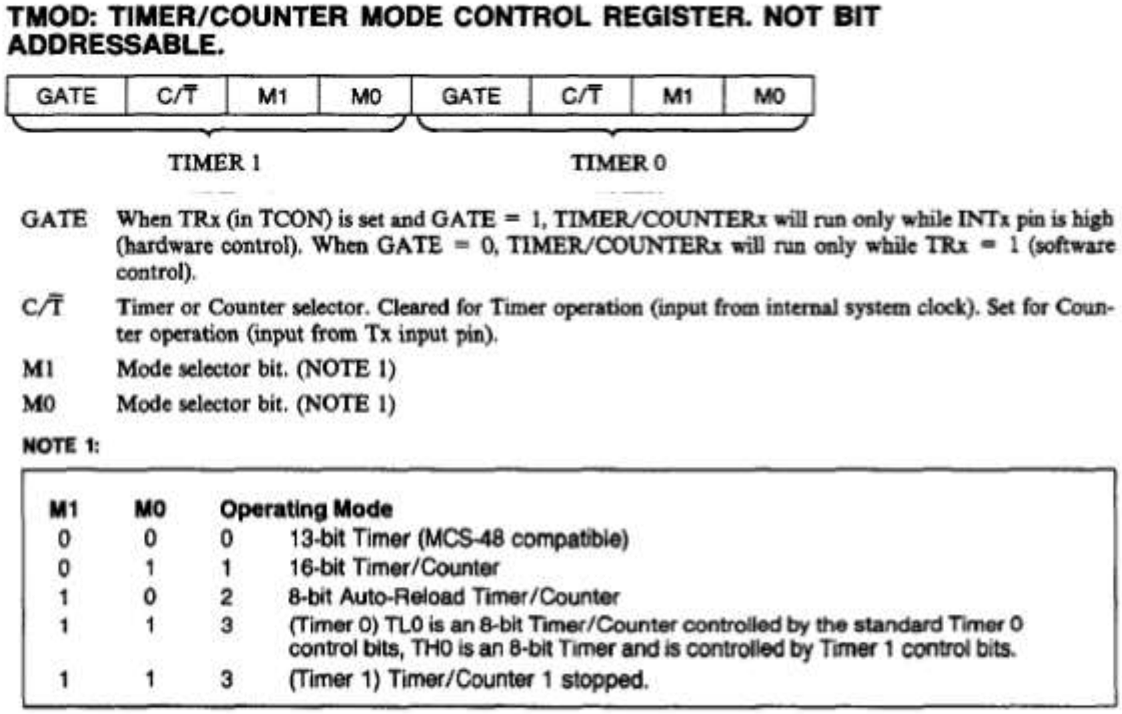
8) Works on crystal frequency of 11.0592 MHz.

9) Has power saving and idle mode in microcontroller when no operation is performed.

10) Six interrupts are available: Reset, Two interrupts Timers i.e. Timer 0 and Timer 1, two external hardware interrupts- INT0 and INT1, Serial communication interrupt for both receive and transmit.

**c) Draw the format of TMOD register and explain.**

**Ans:- ( Format- 2 mks, function of each bit-2 mks)**



**TMOD: TIMER/COUNTER MODE CONTROL REGISTER. NOT BIT ADDRESSABLE.**

| GATE | C/$\overline{T}$ | M1 | M0 | GATE | C/$\overline{T}$ | M1 | M0 |
|---|---|---|---|---|---|---|---|
| TIMER 1 | | | | TIMER 0 | | | |

GATE When TRx (in TCON) is set and GATE = 1, TIMER/COUNTERx will run only while INTx pin is high (hardware control). When GATE = 0, TIMER/COUNTERx will run only while TRx = 1 (software control).

C/$\overline{T}$ Timer or Counter selector. Cleared for Timer operation (input from internal system clock). Set for Counter operation (input from Tx input pin).

M1 Mode selector bit. (NOTE 1)

M0 Mode selector bit. (NOTE 1)

**NOTE 1:**

| M1 | M0 | Operating Mode | |
|---|---|---|---|
| 0 | 0 | 0 | 13-bit Timer (MCS-48 compatible) |
| 0 | 1 | 1 | 16-bit Timer/Counter |
| 1 | 0 | 2 | 8-bit Auto-Reload Timer/Counter |
| 1 | 1 | 3 | (Timer 0) TL0 is an 8-bit Timer/Counter controlled by the standard Timer 0 control bits, TH0 is an 8-bit Timer and is controlled by Timer 1 control bits. |
| 1 | 1 | 3 | (Timer 1) Timer/Counter 1 stopped. |

**d) Explain Boolean processor of 8051 microcontroller.**

**Ans: 3M- explanation 1M-example**

The 8051 instruction set is optimized for the one bit operations. So often desired in real world, real time control applications. The Boolean processor provides direct support for bit manipulation. This leads to more efficient programs that need to deal with binary input and output conditions inherent in digital control problems.

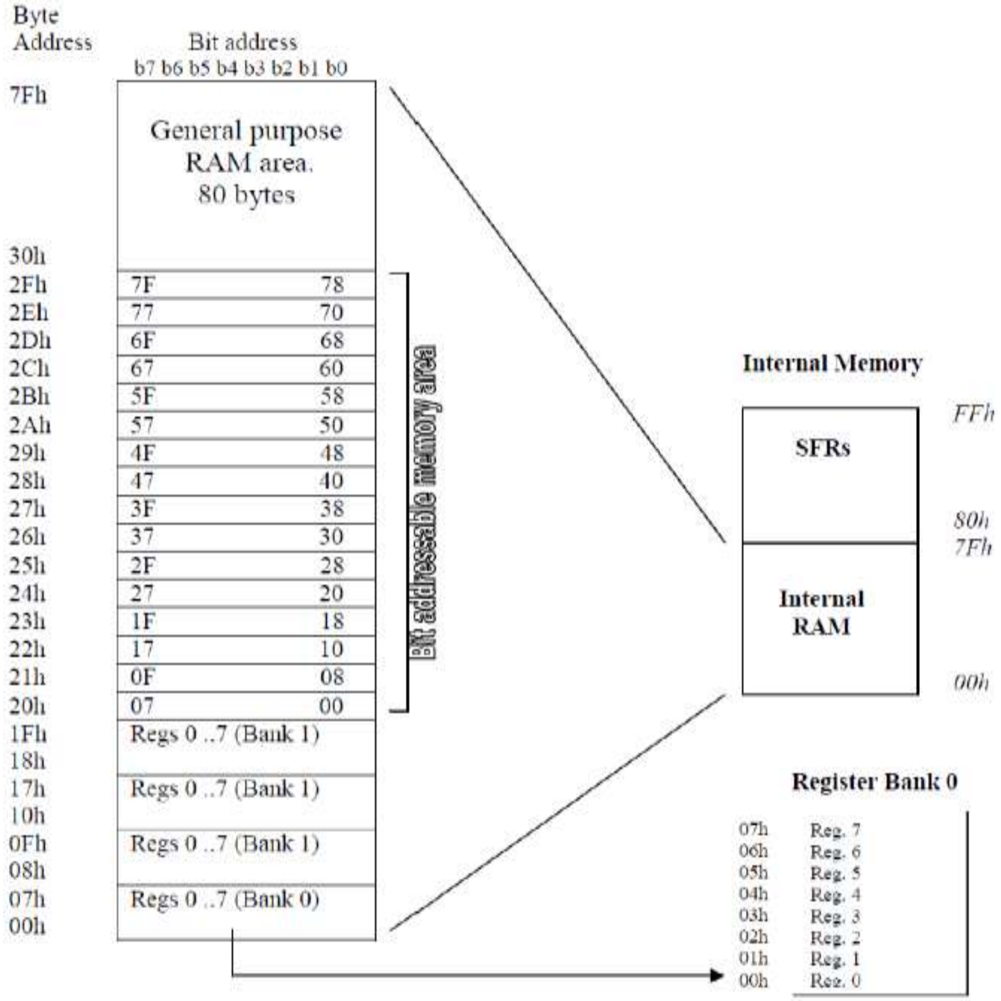
Bit addressing can be used to test pin monitoring or program control flags. For examples, instructions for Boolean function are as given below.

(a) ORL P0, # 1 ; Set P0.0

(b) XRL P0, # 1 ; Toggle P0.0

**e) Draw the internal memory organization of 8051 and explain.**

**Ans: (2M-diagram, 2M- explanation)**



| Byte Address | Bit address b7 b6 b5 b4 b3 b2 b1 b0 | |
|---|---|---|
| 7Fh | General purpose RAM area. 80 bytes | |
| 30h | | |
| 2Fh | 7F | 78 |
| 2Eh | 77 | 70 |
| 2Dh | 6F | 68 |
| 2Ch | 67 | 60 |
| 2Bh | 5F | 58 |
| 2Ah | 57 | 50 |
| 29h | 4F | 48 |
| 28h | 47 | 40 |
| 27h | 3F | 38 |
| 26h | 37 | 30 |
| 25h | 2F | 28 |
| 24h | 27 | 20 |
| 23h | 1F | 18 |
| 22h | 17 | 10 |
| 21h | 0F | 08 |
| 20h | 07 | 00 |
| 1Fh | Regs 0 ..7 (Bank 1) | |
| 18h | | |
| 17h | Regs 0 ..7 (Bank 1) | |
| 10h | | |
| 0Fh | Regs 0 ..7 (Bank 1) | |
| 08h | | |
| 07h | Regs 0 ..7 (Bank 0) | |
| 00h | | |

Register Bank 0

| | |
|---|---|
| 07h | Reg. 7 |
| 06h | Reg. 6 |
| 05h | Reg. 5 |
| 04h | Reg. 4 |
| 03h | Reg. 3 |
| 02h | Reg. 2 |
| 01h | Reg. 1 |
| 00h | Reg. 0 |

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

**Internal RAM**

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

**Register Banks**

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

### Bit-addressable Locations

The 8051 contains 210 bit-addressable locations of which 128 are at locations 20H to 2FH while the rest are in the SFRs. Each of the 128 bits from 20H to 2FH have a unique number (address) attached to them, as shown in the table above. The 8051 instruction set allows you to set or reset any single bit in this section of RAM. With the general purpose RAM from 30H to 7FH and the register banks from 00H to 1FH, you may only read or write a full byte (8 bits) at these locations. However, with bit-addressable RAM (20H to 2FH) you can read or write any single bit in this region by using the unique address for that bit.

### General Purpose RAM

These 80 bytes of Internal RAM memory are available for general-purpose data storage. The general purpose RAM can be accessed using direct or indirect addressing mode instructions.

### Special Function Registers (SFRs)

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

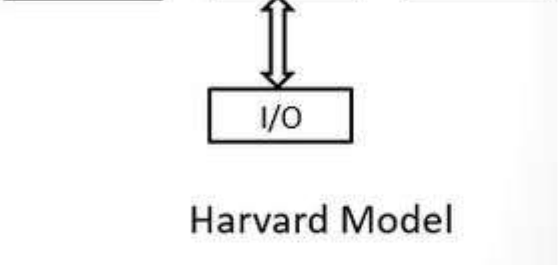
**f) Explain :**

**i)Harvard Architecture**

**ii) Von Neumann architecture**

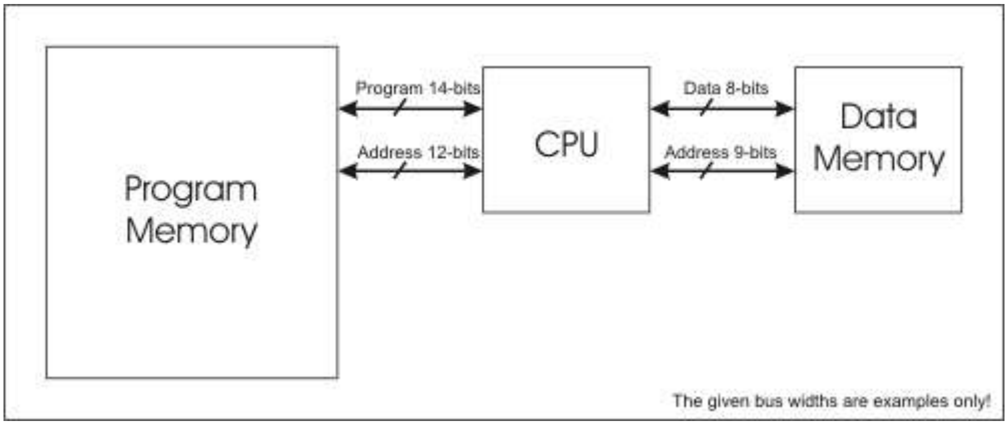
**Ans: 2M- explanation of each**

### i) Harvard Architecture



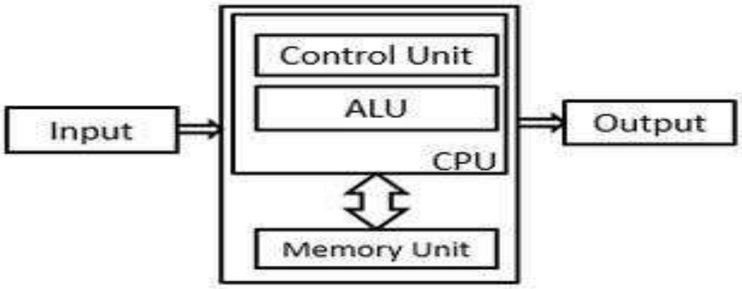
Harvard Model

OR



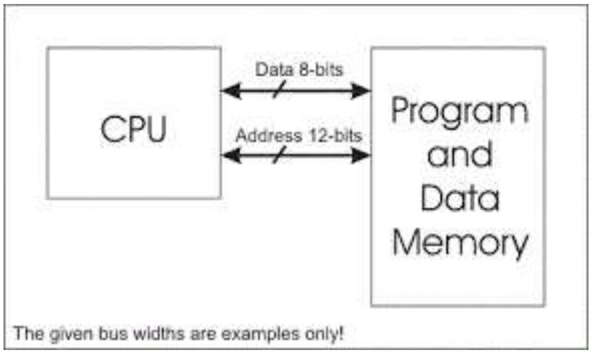
- The name is originated from “Harvard Mark I” a relay based old computer.
- It required two memories for their instruction and data. Design of Harvard architecture is complicated.
- Harvard architecture is required separate bus for instruction and data. Processor can complete an instruction in one cycle.
- Easier to pipeline, so high performance can be achieve.
- Comparatively high cost.

**ii) Von Neumann architecture:**



Von Neumann Model

**OR**



- It is named after the mathematician and early computer scientist John Von Neumann.
- It required only one memory for their instruction and data.
- Design of the von Neumann architecture is simple.
- Von Neumann architecture is required only one bus for instruction and data.
- Processor needs two clock cycles to complete an instruction.
- Low performance as compared to Harvard architecture.
- It is cheaper.

## Q.3 Attempt any four of the following 16

a) List and explain addressing modes of 8051 with example.

Ans: (List addressing modes: 1M, Explanation of any three addressing modes: 3M)

There are a number of addressing modes available to the 8051 instruction set, as follows:
1. Immediate Addressing mode
2. Register Addressing mode
3. Direct Addressing mode
4 Register Indirect addressing mode
5. Relative Addressing mode
6. Absolute addressing mode
7. Long Addressing mode
8. Indexed Addressing mode

1) **Immediate Addressing mode:**
Immediate addressing simply means that the operand (which immediately follows the Instruction op. code) is the data value to be used.
For example the instruction:
MOV A, # 25H ; Load 25H into A
Moves the value 25H into the accumulator. The # symbol tells the assembler that the immediate addressing mode is to be used.

**2) Register Addressing Mode:**

One of the eight general-registers, R0 to R7, can be specified as the instruction Operand. The
assembly language documentation refers to a register generically
as Rn. An example instruction using register addressing is :
ADD A, R5; Add the contents of register R5 to contents of A (accumulator)
Here the contents of R5 are added to the accumulator. One advantage of register addressing is that the instructions tend to be short, single byte instructions.

**3) Direct Addressing Mode:**

Direct addressing means that the data value is obtained directly from the memory location
specified in the operand.
For example, consider the instruction:
MOV R0, 40H; Save contents of RAM location 40H in R0.
The instruction reads the data from Internal RAM address 40H and stores this in theR0.
Direct addressing can be used to access Internal RAM, including the SFR registers.

**4) Register Indirect Addressing Mode:**

Indirect addressing provides a powerful addressing capability, which needs to be appreciated.
An example instruction, which uses indirect addressing, is as follows:
MOV A, @R0; move contents of RAM location whose address is held by R0 into A
Note the @ symbol indicated that the indirect addressing mode is used. If the data is inside the
CPU, only registers R0 & R1 are used for this purpose.

**5) Relative Addressing Mode:**

This is a special addressing mode used with certain jump instructions. The relative address, often
referred to as an offset, is an 8-bit signed number, which is automatically added to the PC to make the address of the next instruction. The 8-bitsigned offset value gives an address range of +
127 to –128 locations.
Consider the following
example: SJMP
LABEL_X
An advantage of relative addressing is that the program code is easy to relocate in memory in that the addressing is relative to the position in memory.

**6) Absolute addressing Mode:**

Absolute addressing within the 8051 is used only by the AJMP (Absolute Jump) and ACALL
(Absolute Call) instructions.

**7) Long Addressing Mode:**

The long addressing mode within the 8051 is used with the instructions LJMP and LCALL. The
address specifies a full 16 bit destination address so that a jump or a call can be made to a location within a 64KByte code memory space (216 = 64K).
An example instruction is:
LJMP 5000h; full 16 bit address is specified in operand

8) **Index Addressing Mode:**

With indexed addressing a separate register, either the program counter, PC, or the data pointer
DTPR, is used as a base address and the accumulator is used as an offset address. The effective address is formed by adding the value from the base address to the value from the offset address. Indexed addressing in the 8051 is used with the JMP or MOVC instructions. Look up tables are easy to implement with the help of index addressing.
Consider the example
instruction: MOVC A,
@A+DPTR
MOVC is a move instruction, which moves data from the external code memory space. The address operand in this example is formed by adding the content of the DPTR register to the accumulator value. Here the DPTR value is referred to as the base address and the accumulator value us referred to as the index address.

b) Explain the function of following instructions-

i. DAA
ii. ANL
iii. RET
iv. SWAP A

**Ans : (Each instruction :1M)**

**i)DAA**

**Description:** DA adjusts the contents of the Accumulator to correspond to a BCD (Binary Coded Decimal) number after two BCD numbers have been added by the ADD or ADDC instruction. If the carry bit is set or if the value of bits 0-3 exceed 9, 0x06 is added to the accumulator. If the carry bit was set when the instruction began, or if 0x06 was added to the accumulator in the first step, 0x60 is added to the accumulator

The **Carry bit (C)** is set if the resulting value is greater than 0x99, otherwise it is cleared.

Example: MOV A, # 47H ; A= 0100 0111

ADD A, # 38 H ; A= 47 H + 38H = 7FH, invalid BCD DA A

; A= 1000 0101 =85 H, valid BCD

**ii) ANL**

Description: This instruction performs bitwise logical AND operation between the variables indicated and stores the result in the destination variable.

No flags are affected.

Example :ANL A,$R_n$

**iii) RET**

Description: RET pops the high and low order bytes of the PC successively from the stack, decrementing the stack pointer by two program execution at the resulting address.

No flags are affected

Example: the stack pointer originally contains the value 0BH, internal RAM location contain the values 23H and 01H,respectively

RET will leave the stack pointer equal to value 09h.Program execution will continue at location 0123H.

**iv) SWAP A**

Description: This instruction exchanges bits 0-3 of the Accumulator with bits 4-7 of the Accumulator. This instruction is identical to executing "RR A" or "RL A" four times.

No of bytes: 1 byte
Addressing mode: register specific

```
Example:   MOV A, #59H        ; A= 59H
           SWAP A             ; A= 95H
```

c) Explain any four directives with examples.

**Ans :(Each directive :1M)**

**(i) ORG (ii) DB (iii) EQU (iv) END**

**i)** ORG:-ORG stands for Origin

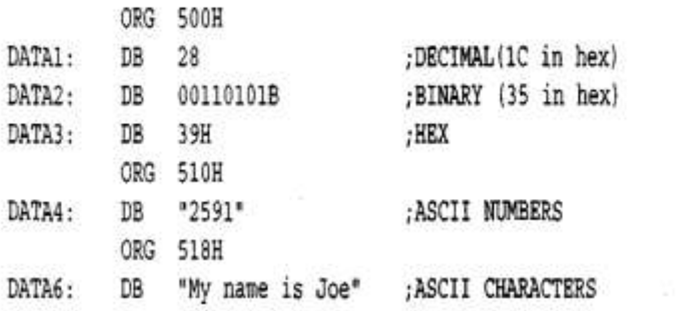
Syntax: ORG Address

The ORG directive is used to indicate the beginning of the address. The number that comes after ORG can be either in hex or in decimal. If the number is not followed by H, it is decimal and the assembler will convert it to hex. Some assemblers use –**.**ORG‖ (notice the dot) instead of –ORG‖ for the origin directive.

**ii)** DB: (Data Byte)

Syntax: Label: DB Byte

Where byte is an 8-bit number represented in either binary, Hex, decimal or ASCII form. There should be at least one space between label & DB. The colon (:) must present after label. This directive can be used at the beginning of program. The label will be used in program instead of actual byte. There should be at least one space between DB & a byte. Following are some DB examples:



```
        ORG  500H
DATA1:  DB   28                ;DECIMAL(1C in hex)
DATA2:  DB   00110101B         ;BINARY (35 in hex)
DATA3:  DB   39H               ;HEX
        ORG  510H
DATA4:  DB   "2591"            ;ASCII NUMBERS
        ORG  518H
DATA6:  DB   "My name is Joe"  ;ASCII CHARACTERS
```

**iii)** EQU: Equate
It is used to define constant without occupying a memory location.

Syntax:

| Name | EQU | Constant |
|---|---|---|

By means of this directive, a numeric value is replaced by a symbol.
For e.g. MAXIMUM EQU 99 After this directive every appearance of the label –MAXIMUM in the program, the assembler will interpret as number 99 (MAXIMUM=99).

iv) **END:**
This directive must be at the end of every program. meaning that in the source code anything after the END directive is ignored by the assembler.
This indicates to the assembler the end of the source file (asm).
Once it encounters this directive, the assembler will stop interpreting program into machine code.
e. g. END ; End of the program.

d) Write assembly language program to multiply two 8 bit numbers store result at 55 H.

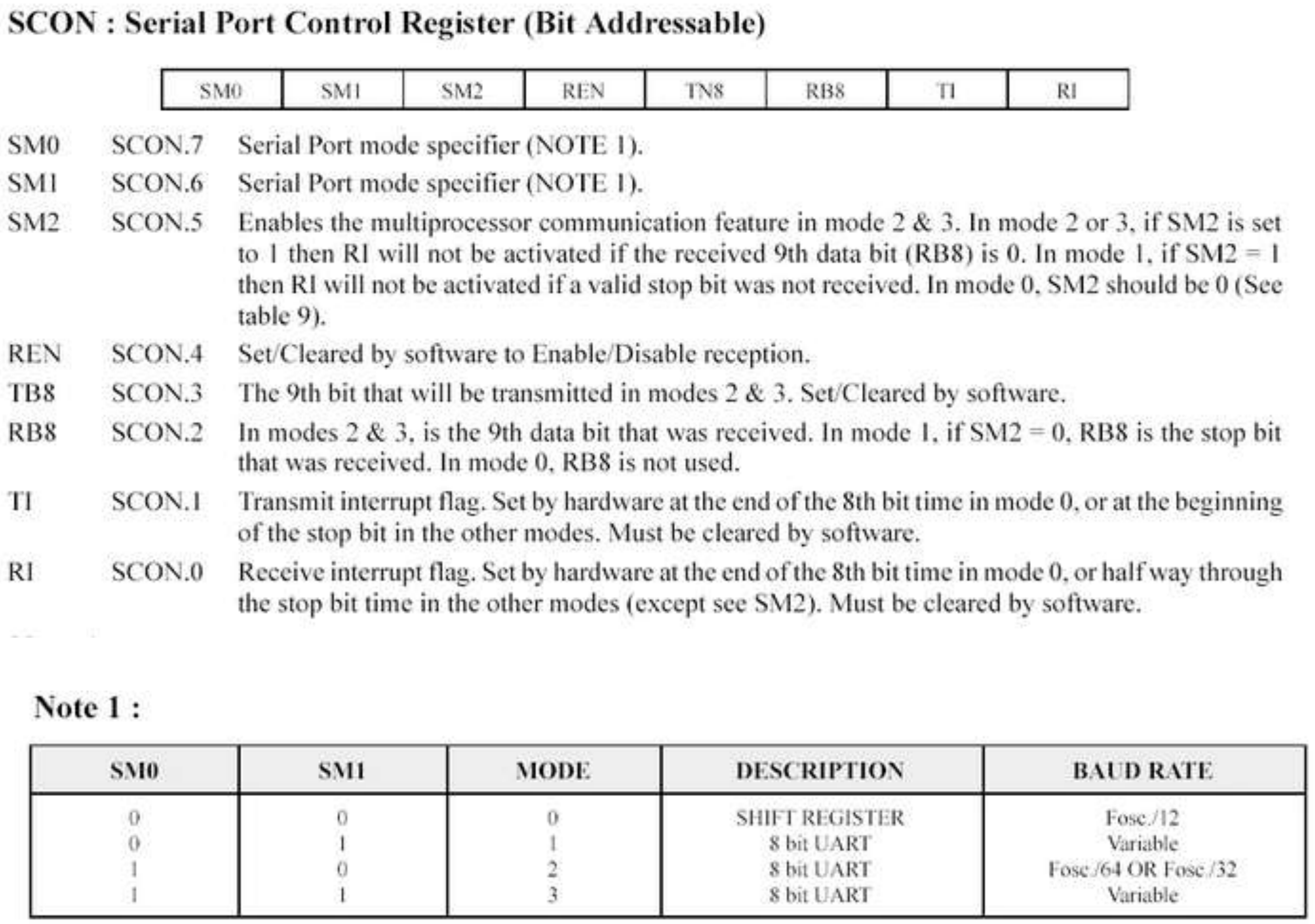
**Ans: (logically correct program:3M, comments :1M)**
**Program**

```
 MOV 10H, # 23H          ; store first 8-bit no. in 10H
MOV 11H, #15H            ; store second 8-bit no. in 11H
MOV A, 10H               ; move first number to A
MOV B, 11H               ; move second number to B
MUL AB                   ; multiply the numbers
MOV 55H, A               ; move result to 55H

HERE: SJMP HERE
```

e) Draw the format of SCON register and explain.

**Ans: (SCON register format - -2 Marks, Explanation of each bit --2 Marks)**



**SCON : Serial Port Control Register (Bit Addressable)**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| SM0 | SM1 | SM2 | REN | TN8 | RB8 | TI | RI |

SM0 SCON.7 Serial Port mode specifier (NOTE 1).

SM1 SCON.6 Serial Port mode specifier (NOTE 1).

SM2 SCON.5 Enables the multiprocessor communication feature in mode 2 & 3. In mode 2 or 3, if SM2 is set to 1 then RI will not be activated if the received 9th data bit (RB8) is 0. In mode 1, if SM2 = 1 then RI will not be activated if a valid stop bit was not received. In mode 0, SM2 should be 0 (See table 9).

REN SCON.4 Set/Cleared by software to Enable/Disable reception.

TB8 SCON.3 The 9th bit that will be transmitted in modes 2 & 3. Set/Cleared by software.

RB8 SCON.2 In modes 2 & 3, is the 9th data bit that was received. In mode 1, if SM2 = 0, RB8 is the stop bit that was received. In mode 0, RB8 is not used.

TI SCON.1 Transmit interrupt flag. Set by hardware at the end of the 8th bit time in mode 0, or at the beginning of the stop bit in the other modes. Must be cleared by software.

RI SCON.0 Receive interrupt flag. Set by hardware at the end of the 8th bit time in mode 0, or half way through the stop bit time in the other modes (except see SM2). Must be cleared by software.

**Note 1 :**

| SM0 | SM1 | MODE | DESCRIPTION | BAUD RATE |
|---|---|---|---|---|
| 0 | 0 | 0 | SHIFT REGISTER | Fosc./12 |
| 0 | 1 | 1 | 8 bit UART | Variable |
| 1 | 0 | 2 | 8 bit UART | Fosc./64 OR Fosc./32 |
| 1 | 1 | 3 | 8 bit UART | Variable |

**SM2**: SM2 is the D5 bit of the SCON register.

This bit enables the multiprocessing capability of the8051. MakeSM2=0 since we are not using the 8051 in a multiprocessor environment.

**REN**: The REN (receive enable)bit is D4 of the SCON register. The REN bit is also referred to as SCON.4 since SCON is a bit addressable register.When the REN =1, it allows the 8051 to receive data on the RxD pin of the 8051. As a result, if we want the8051 to both transfer and receive data, REN must be set to 1.

By making REN=0, the receiver is disabled. Making REN=1 or REN=0can be achieved by the instructions “SETBSCON.4” and “CLR SCON.4”, respectively.

This bit can be used to block any serial data reception and is an extremely important bit in the SCON register.

**TB8**: TB8 (transfer bit 8) is bit D3 of SCON. It is used for serial modes 2 and 3. WemakeTB8=0 since it is not used in our applications.

**RB8**: RB8 (receive bit 8) is bit D2 of the SCON register. In serial mode1, this bit gets copy of the stop bit when an 8-bitdata is received. This bit (as is the case for TB8) is rarely used any more. In all our applications wewillmakeRB8=0.Like TB8, the RB8 bit is also used in serial modes 2and 3.

**TI**: TI (transmit interrupt)is bit D1 of the SCON register. This is an extremely important flag bit in the SCON register.When the 8051 finishes the transfer of the 8-bit character, it raises the T1flagto indicate that it is ready to transfer another byte. The TI bit is raised at the beginning of the stop bit.

**RI**: RI(receive interrupt)is theD0 bit of the SCON register. This is another extremely important flag in the SCON register

## Q.4 a) Attempt any three of the following 12

i)Explain the function of following instructions:

1. MOV A, @R0
2. MOVX A, @DPTR
3. INC
4. MOV A, B

Ans:- ( **1Mark for each instruction)**

**1.MOVA, @R0**

This instruction moves the contents of memory location pointed by R0 to the Accumulator

No of bytes : 1

Addressing mode: register indirect

No flags affected

**2.MOVX A, @DPTR**

This instruction moves the contents of the external RAM memory pointed by (or stored in) DPTR to the Accumulator.

```
Example:    MOV DPTR, # 2000H   ; DPTR = 2000H(external RAM address)

            MOV A, @DPTR        ;2000h=0BH

                                ;A=0BH
```

**3. INC**

Description :Add a 1 to the register.

No. of bytes: 1

Addressing mode :Register specific

No flags affected

**4. MOV A, B**

This instruction moves the contents of memory of register B to the Accumulator

No of bytes : 1

Addressing mode: register

No flags affected

ii) Describe the serial communication modes of 8051.

Ans:- :**1mark for each mode**

8051 micro controller communicate with another peripheral device through RXD and TXD pin of port3.controller have four mode of serial communication. This four mode of serial communication are below.

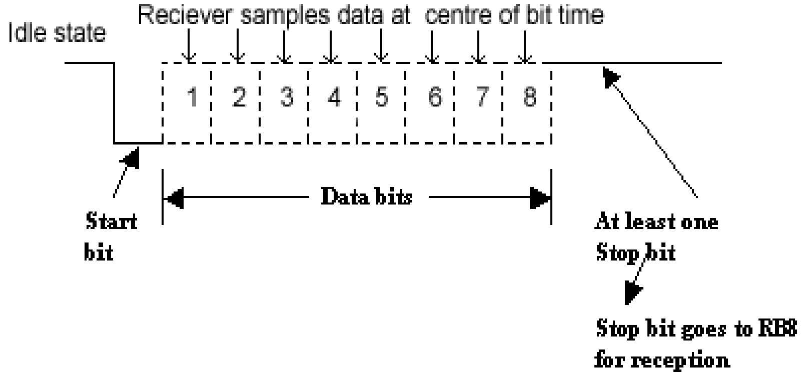
1. Serial data mode 0-fixed baud rate.
2. Serial data mode 1-variable baud rate.
3. Serial data mode 2 -fixed baud rate.
4. Serial Data mode 3 -variable baud rate.

**1. Serial Data Mode-0 (Baud Rate Fixed)**

In this mode, the serial port works like a shift register and the data transmission works synchronously with a clock frequency of $f_{osc}$ /12. Serial data is received and transmitted through RXD. 8 bits are transmitted/ received aty a time. Pin TXD outputs the shift clock pulses of frequency $f_{osc}$ /12, which is connected to the external circuitry for synchronization. The shift frequency or baud rate is always 1/12 of the oscillator frequency

**2. Serial Data Mode-1 (standard UART mode)(baud rate is variable)**

In mode-1, the serial port functions as a standard Universal Asynchronous Receiver Transmitter (UART) mode. 10 bits are transmitted through TXD or received through RXD. The 10 bits consist of one start bit (which is usually '0'), 8 data bits (LSB is sent first/received first), and a stop bit (which is usually '1'). Once received, the stop bit goes into RB8 in the special function register SCON. The **baud rate is variable**


Idle state
Reciever samples data at centre of bit time
1
2
3
4
5
6
7
8
Start bit
Data bits
At least one Stop bit
Stop bit goes to RB8 for reception

$$f_{baud} = \frac{2^{SMOD}}{32} X \frac{fosc}{12 X [256-(TH1)]}$$

**3. Serial Data Mode-2 Multiprocessor (baud rate is fixed)**

In this mode 11 bits are transmitted through TXD or received through RXD.The various bits are as follows: a start bit (usually '0'), 8 data bits (LSB first), a programmable 9 th (TB8 or RB8)bit and a stop bit (usually '1'). While transmitting, the 9 th data bit (TB8 in SCON) can be assigned the value '0' or '1'. For example, if the information of parity is to be transmitted, the parity bit (P) in PSW could be moved into TB8.On reception of the data, the 9 th bit goes into RB8 in 'SCON', while the stop bit is ignored. The baud rate is programmable to either 1/32 or 1/64 of the oscillator frequency

$$f_{baud} = (2^{SMOD}/64) f_{osc}$$

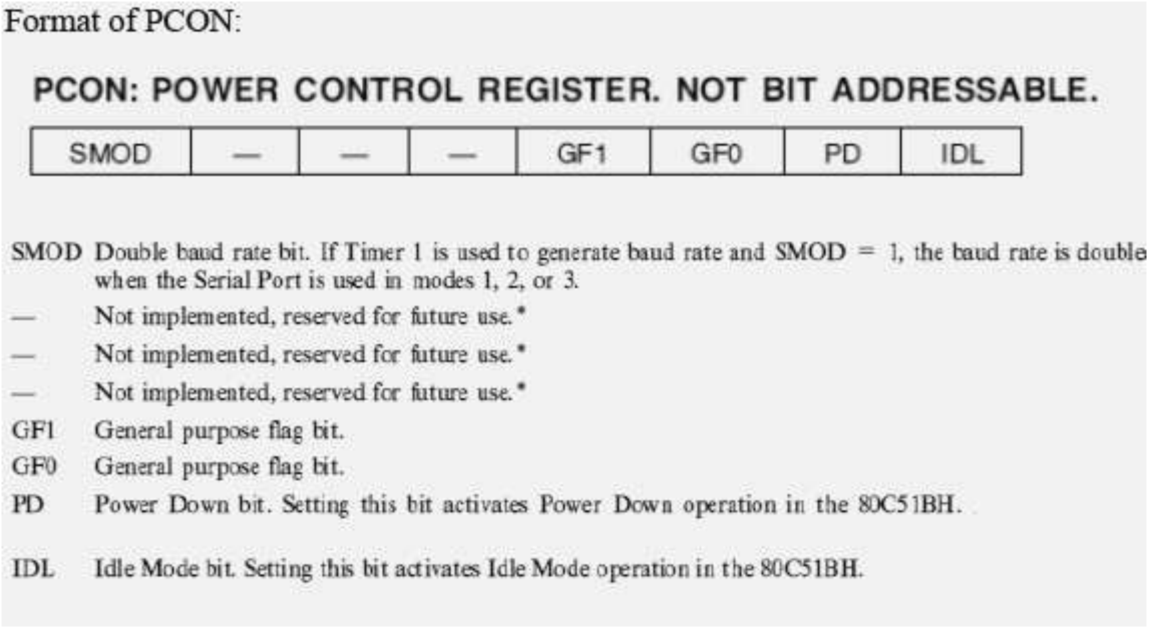
**4. Serial Data Mode-3 - Multi processor mode(Variable baud rate)**

In this mode 11 bits are transmitted through TXD or received through RXD.The various bits are: a start bit (usually '0'), 8 data bits (LSB first), a programmable 9 th bit and a stop bit (usually '1'). Mode-3 is same as mode-2, except the fact that the baud rate in mode-3 is variable (i.e., just as in mode-1).

$$f_{baud} = (2^{SMOD}/32) * (f_{osc} / 12 (256\text{-}TH1))$$

iii) Draw and explain format of PCON register.

**Ans** : (format 2M, explain each bit 2M)



Format of PCON:

PCON: POWER CONTROL REGISTER. NOT BIT ADDRESSABLE.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| SMOD | — | — | — | GF1 | GF0 | PD | IDL |

SMOD Double baud rate bit. If Timer 1 is used to generate baud rate and SMOD = 1, the baud rate is double when the Serial Port is used in modes 1, 2, or 3.

— Not implemented, reserved for future use.*

— Not implemented, reserved for future use.*

— Not implemented, reserved for future use.*

GF1 General purpose flag bit.

GF0 General purpose flag bit.

PD Power Down bit. Setting this bit activates Power Down operation in the 80C51BH.

IDL Idle Mode bit. Setting this bit activates Idle Mode operation in the 80C51BH.

iv) Write assembly language program to send message ‘WELCOME’ serially at 9600 baud rate continuously.

**Ans:(3M- logically correct Prog,1M-comments)**

```
       MOV TMOD, #20H                  ; timer 1, mode2
 MOV TH1,#-3                    ; 9600 baud rate
 MOV SCON, #50H                 ; 8-bit data,1 stop bit, REN enabled
 SETB TR1                       ; Start timer 1
      AGAIN: MOV A, #”W”               ; transfer “W”
 ACALL MESSAGE                  ; Some delay
        MOV A, #”E”                    ; transfer “E”
 ACALL MESSAGE
        MOV A, #”L”                    ; transfer “L”
 ACALL MESSAGE
        MOV A, #”C”                     ; transfer “C”
 ACALL MESSAGE
```

```
          MOV A, #”O”                        ; transfer “O”

          ACALL MESSAGE

          MOV A, #”M”                        ; transfer “M”

ACALL MESSAGE

         MOV A, #”E”                         ; transfer “E”

  ACALL MESSAGE

          SJMP AGAIN



  MESSAGE: MOV SBUF, A;

JNB T1, HERE;

CLR T1;

         RET
```

**b) Attempt any one of the following** **6**

i)Write a program to arrange 8 numbers in array stores in internal RAM in ascending order.

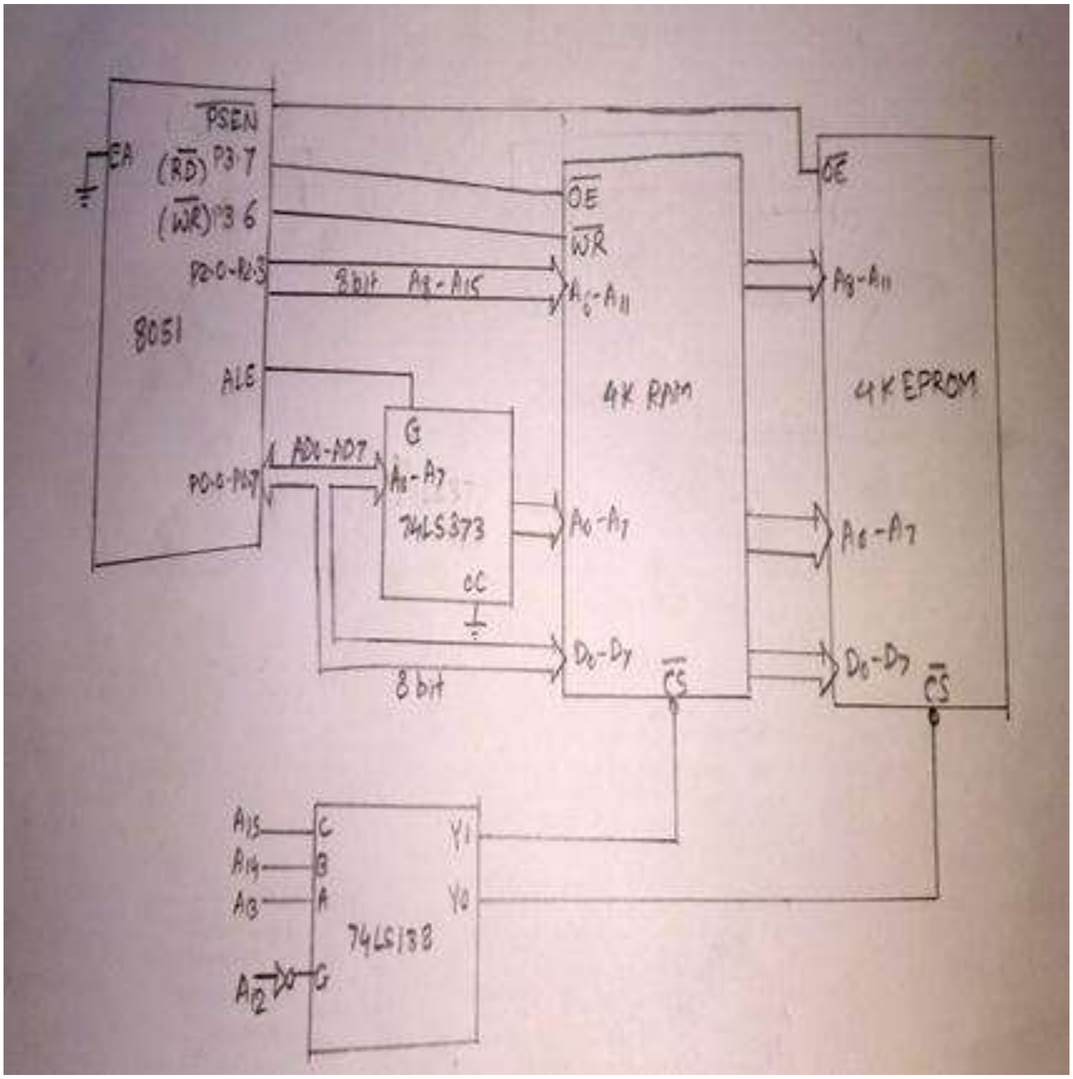
**Ans: (5M-prog,1M-comments)**

```
MOV R3, #08H            ; Initialize the pass counter
UP1:      MOV R0, #40H                 ; Initialize the memory pointer
MOV R7, #05H                    ; Initialize the byte counter
UP:    MOV A, @R0                     ; Read number from array
  MOV 0F0H, A                   ; Copy number to B register
INC R0                  ; Increment memory pointer
          MOVA,@R0                     ; Read next number from array
CJNE A, 0F0, DWN                ; Compare  number with next number
          AJMP SKIP
DWN: JNC SKIP                          ; If number> next number then go to SKIP
          MOV R0, R3                   ; Else exchange the number with next number
          MOV @R0, A
          INC R0
          MOV A, 0F0H
          MOV@R0, A
SKIP: DJNZ R3, UP                ; Decrement byte counter if not zero, go to UP
          DJNZ R7 UP1                  ; Decrement pass counter if not zero, go to UP1
```

```
LOOP:       AJMP LOOP                    ; Stop
```

ii) Draw interfacing diagram of 4 KB EPROM & 4 KB RAM of 8051. Draw memory map.

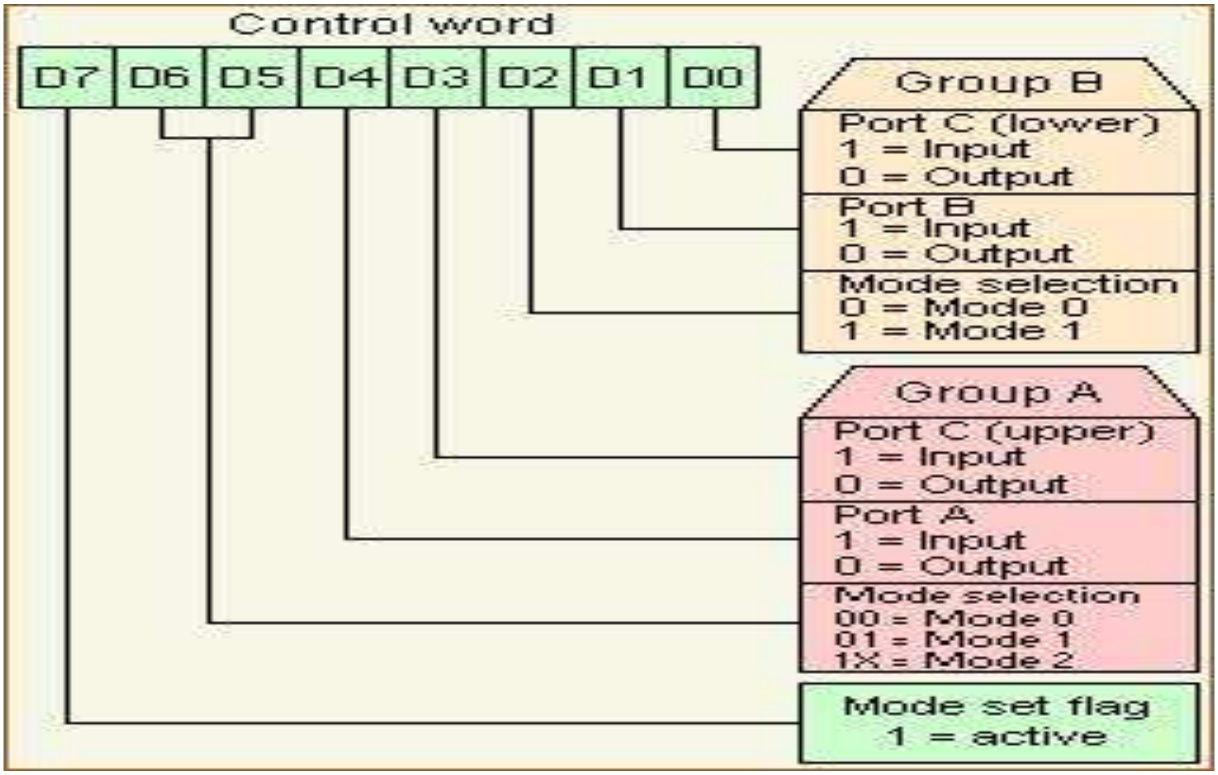
Ans: (correct interfacing : 3M,correct mapping;3M)



| | A15 | A14 | A13 | A12 | A11 | A10 | A9 | A8 | A7 | A6 | A5 | A4 | A3 | A2 | A1 | A0 | ADDR |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Start Addr of EPROM | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0000H** |
| End Addr of EPROM | **0** | **0** | **0** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **0FFFH** |
| Start Addr of RAM | **0** | **0** | **1** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **2000H** |
| End Addr of RAM | **0** | **0** | **1** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **2FFFH** |

iii) Draw and explain control word register of 8255.

**Ans: (4M for correct ans)**



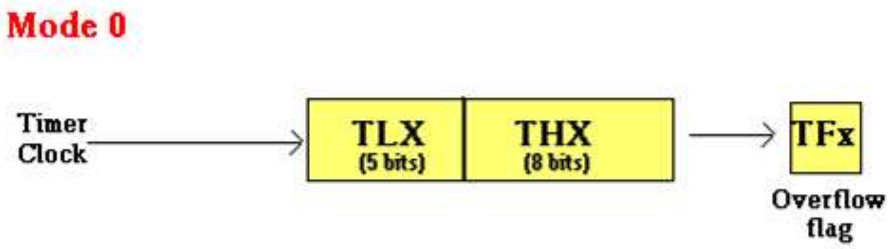
**Q.5 Attempt any FOUR of the following** **16**

a) Explain timer modes of 8051.

**Ans :(List-1M,Description of any two timers-3M)**

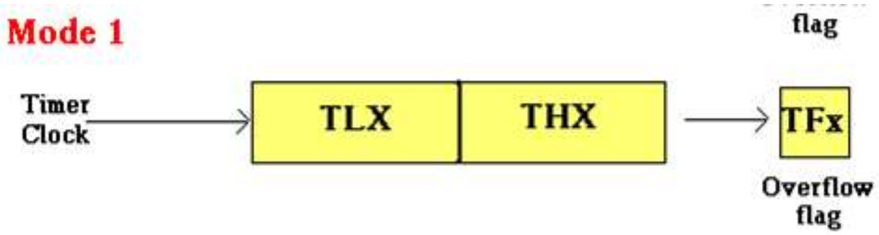
| M1 | M0 | MODE | DESCRIPTION |
|---|---|---|---|
| 0 | 0 | 0 | 13-bit timer |
| 0 | 1 | 1 | 16-bit timer |
| 1 | 0 | 2 | 8-bit auto-reload |
| 1 | 1 | 3 | Split mode |

Operating modes of Timer: The timer may operate in any of the four modes that are determined byM1 and M0 bit in TMOD register.

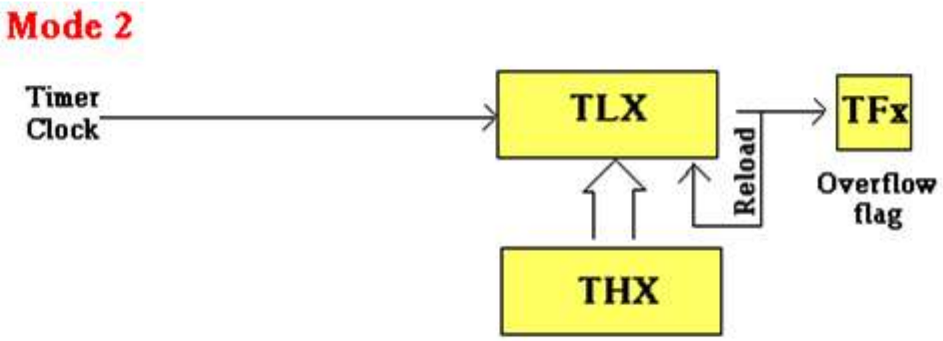


**Mode 0:**

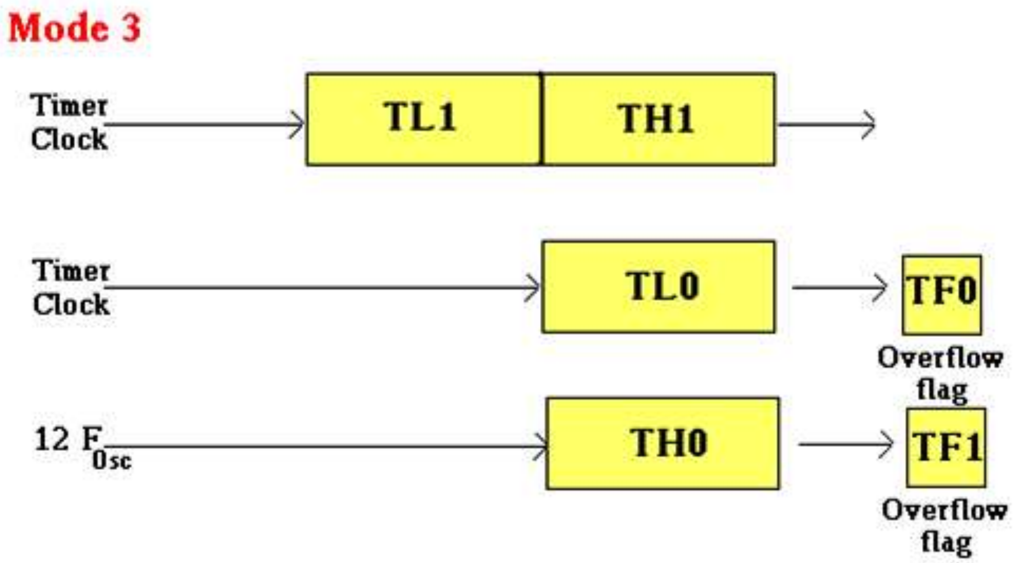
**I**n mode0 the register THX is used as 8 bit counter and TLX is used as 5 bit counter. The pulse i/p is divided by (32)10so that TH counts. Hence original oscillator frequency is divided by (384)10.The timer flag is set when THX rolls over from FF to 00H.



It is similar to Mode 0 except TLX is configured as a full 8-bit counter. Hence pulse input is divided by 25610 so that TH counts the timer flag is set when THX rolls over from FF to 00H.



In this mode only TLX is used as 8-bit counter. THX is used to hold the value which is loaded in TLX initially. Everytime TLX overflows from FFH to 00H the timer flag is set and the value from THX is automatically reloaded in TLX register.



In this mode, timer 0 becomes two completed separate 8-bit timers. TL0 is controlled by gate arrangement of timer 0 and sets timer 0 flag when it overflows. TH0 receives the timer clock under the control of TR1 bit and sets TF1 flag when it overflows. Timer 1 may be used in mode 0, 1 and 2 with one important exception that no interrupt will be generated by the timer when the timer 0 is using TF1 overflow flag

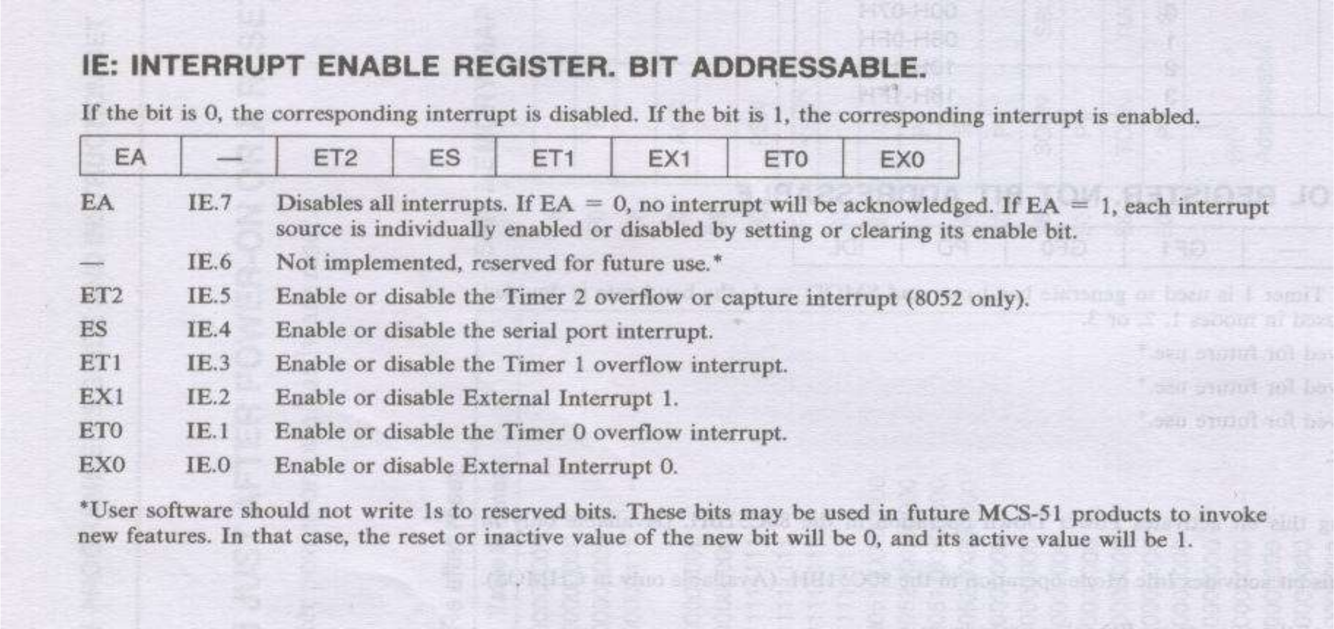
b) List the interrupts of 8051. Give their priorities and vector address.

**Ans:(vector locations-2M,Priority-2M)**

| **Interrupt Source** | **Vector address** | **Interrupt priority** |
|---|---|---|
| External Interrupt 0 – INT0 | 0003H | 1 |
| Timer 0 Interrupt | 000BH | 2 |
| External Interrupt 1 – INT1 | 0013H | 3 |
| Timer 1 Interrupt | 001BH | 4 |
| Serial Interrupt | 0023H | 5 |

c) Draw and explain IE register.

Ans: **(2M- format, 2M –explanation)**



IE: INTERRUPT ENABLE REGISTER. BIT ADDRESSABLE.

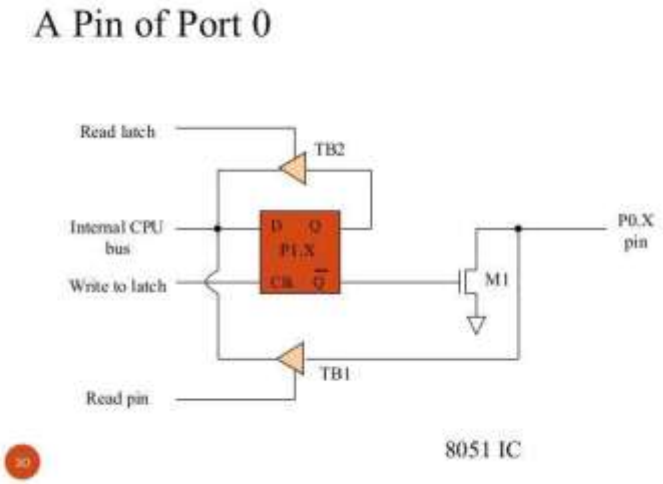
If the bit is 0, the corresponding interrupt is disabled. If the bit is 1, the corresponding interrupt is enabled.

| EA | — | ET2 | ES | ET1 | EX1 | ET0 | EX0 |
|---|---|---|---|---|---|---|---|

| | | |
|---|---|---|
| EA | IE.7 | Disables all interrupts. If EA = 0, no interrupt will be acknowledged. If EA = 1, each interrupt source is individually enabled or disabled by setting or clearing its enable bit. |
| — | IE.6 | Not implemented, reserved for future use.* |
| ET2 | IE.5 | Enable or disable the Timer 2 overflow or capture interrupt (8052 only). |
| 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. |

*User software should not write 1s to reserved bits. These bits may be used in future MCS-51 products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1.

d) Draw port 0 pin circuit & explain bit function.(2M diagram , 2M explanation)



A Pin of Port 0

Port 0: It may be used as input/output or bidirectional low order address and data bus for external memory. It does not have internal pull up resistors

When port 0 is used as address/data bus internal control logic switches to address lines to the gate of FET.

A logic 1 will turn off lower FET to provide high output at that pin. A logic 0 will turn on lower FET to provide low output at that pin.

After the address has been formed ALE signal is used to latch the address and then bus is turns around to become data bus. Port 0 now reads data from external memory; hence it must be configured as input. The internal control logic writes logic 1 to all the latches of port 0.

e) Write an assembly language program to check bit P2.3. If it is high send 55 H to P1 else AAH to P3.

**Ans: (3 M—any correct program, 1M comments)**

```
MOV A, #55H          ;Load the Accumulator with data 55H

JNB P2.3, HERE       ;if P2.3 is not set then go to specified address else go to next address

MOV P1, A            ;send 55H to port 1

SJMP $

HERE: MOV A, #0AAh

MOV P3, A            ;send AAH to port 3

SJMP $
```

## Q.6 Attempt any <u>FOUR</u> of the following. 16

a) Write a program to generate square wave of 5 KHz on P3.7 using timer1, model, XTAL = 11.0592 MHz.

**Ans: ( 1M- calculation, 2M- Program,1m- comments)**

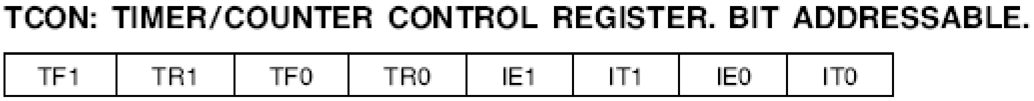
Crystal frequency= 12 MHz
I/P clock = $(11.059 \text{ X } 10^6)/12$= 1000000 = 921.58KHz
$T_{in}$ = 1.085µ sec
For 5 kHz square wave
$F_{out}$ = 5 KHz
$T_{out} = 1/\ 5\text{X } 10^3$
$T_{out}$ = 0.2msec =200us
Consider half of it = $T_{out}$ = 100µ sec
N = $T_{out}$ / $T_{in}$ = 100/1.085 = 92.165
65536-92.165= $(65444)_{10}$ = $(FFA4)_{16}$

**Program:-**

```
MOV TMOD, # 10H                 ; Set timer 1 in Mode 1, i.e., 16 bit timer
L2:    MOV TL1, # 0A4H          ; Load TL register with LSB of  count
       MOV TH1, # 0FFH          ; load TH register with MSB of count
       SETB TR1                 ; start timer 1
L1:      JNB TF1, L1            ; poll till timer roll over
CLR TR1                         ; stop timer 1
CPL P3.7                        ; complement port 1.4 line to get c high or low
CLR TF1                        ; clear timer flag 1
SJMP L2                         ; re-load timer with count as mode 1 is not auto reload
```

b) Draw and format of TCON register.

**Ans :( format : 2 Marks Explanation : 2 Marks)**



TCON: TIMER/COUNTER CONTROL REGISTER. BIT ADDRESSABLE.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| TF1 | TR1 | TF0 | TR0 | IE1 | IT1 | IE0 | IT0 |

TF1 TCON. 7 Timer 1 overflows flag. Set by hardware when the Timer/Counter 1 Overflows.Cleared by hardware as processor vectors to the interrupt service routine.

TR1TCON. 6 Timer 1 run control bit. Set/cleared by software to turn Timer/Counter 1 ON/OFF.

TF0 TCON. 5 Timer 0 overflow flag. Set by hardware when the Timer/Counter 0 overflows. Cleared by hardware as processor vectors to the service routine.

TR0 TCON. 4 Timer 0 run control bit. Set/cleared by software to turn Timer/Counter 0 ON/OFF.

IE1 TCON. 3 External Interrupt 1 edge flag. Set by hardware when External Interrupt edge is detected. Cleared by hardware when interrupt is processed.

IT1 TCON. 2 Interrupt 1 type control bit. Set/cleared by software to specify falling edge/low level triggered External Interrupt.

IE0 TCON. 1 External Interrupt 0 edge flag. Set by hardware when External Interrupt edge detected. Cleared by hardware when interrupt is processed.

IT0 TCON. 0 Interrupt 0 type control bit. Set/cleared by software to Specify falling edge/low level triggered External Interrupt

c) Draw and explain block diagram of 8255 PPI.

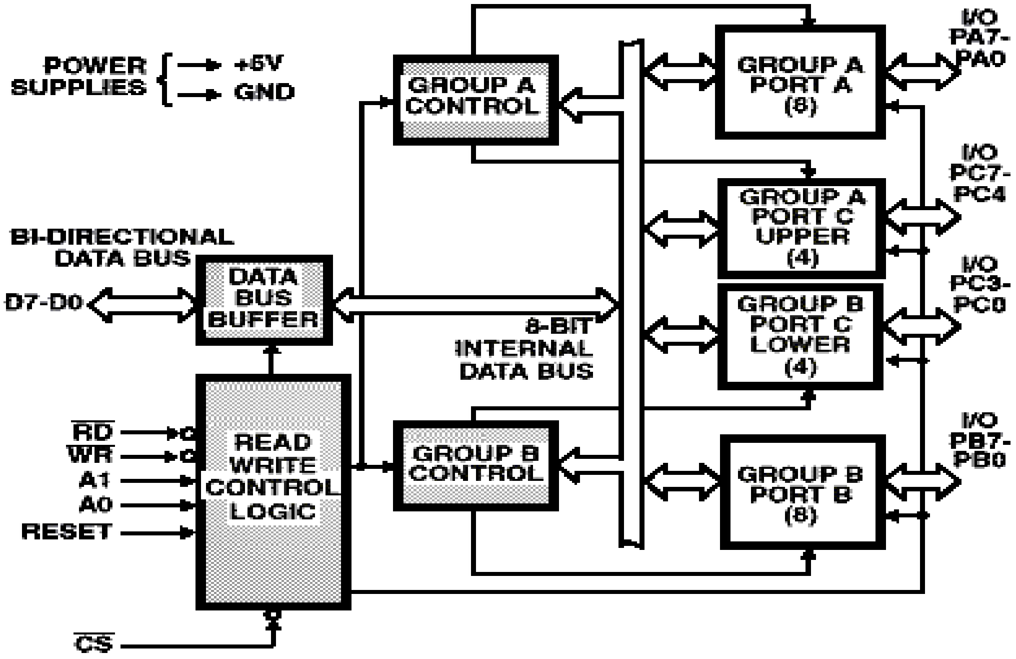


Fig shows the internal block diagram of 8255.

It consist of data bus buffer, control logic and Group A and Group B controls.

**Data Bus Buffer**: This tristate bidirectional buffer is used to interface the internal data bus of 8255 to the system data bus. Input or output instructions executed by the CPU either read data from, or Write data into the buffer. Output data from the CPU to the ports or control register, and input data to the CPU from the ports or status register are all passed through the buffer.

**Control Logic** The control logic block accepts control bus signals as well as inputs from the address bus, and issues commands to the individual group control blocks (Group A control and Group B control). It issues appropriate enabling signals to access the required data /control words or status word. The input pins for the control logic section are described here.

**Group A and Group B Controls**: Each of the Group A and Group B control receives control words from the CPU and issues appropriate command to the ports associated with it. The Group A control block controls Port A and PC7-PC4 while the Group B control block controls Port B and PC3-PC0.

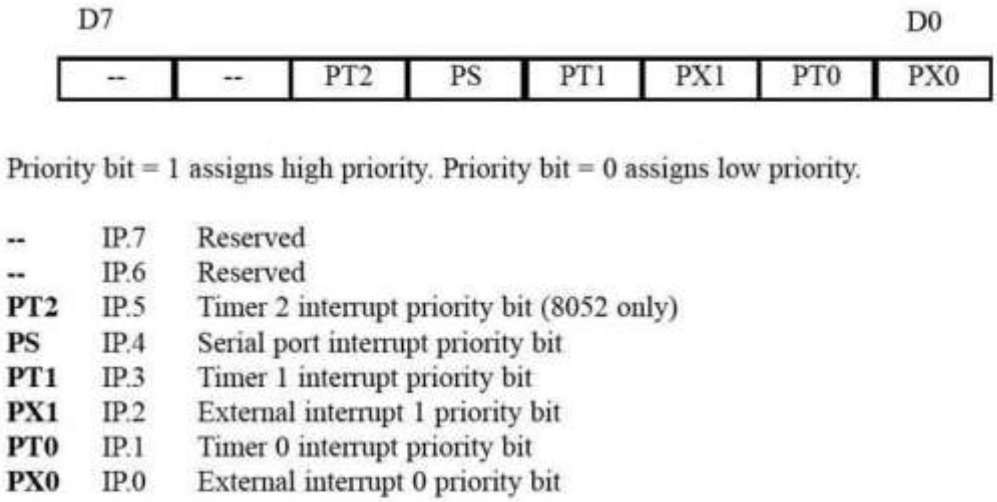
Port A: This has an 8 bit latched and buffered output and an 8 bit input latch. It can be programmed in three modes: mode 0, mode1 and mode2.

Port B: This has an 8 bit data I/O latch/buffer and an 8 bit data input buffer. It can be programmed in mode0 and mode1.

Port C: This has one 8-bit unlatched input buffer and an 8-bit output latch/buffer. Port C can be separated into two parts and each can be used as control signals for ports A and B in the handshake mode. It can be programmed for bit set/reset operation.

d) Draw the format of IP register and explain.

**Ans: (2M- format, 2M –explanation)**



D7 D0

| -- | -- | PT2 | PS | PT1 | PX1 | PT0 | PX0 |
|---|---|---|---|---|---|---|---|

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

| | | |
|---|---|---|
| -- | 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 |

e) Explain any four selection factors of microcontroller.

**Ans:(Any four factors ,each –1 mark)**

The selection of microcontroller depends upon the type of application. The following factors must be considered while selecting the microcontroller.

1. Word length: The word length of microcontroller is either 8, 16 or 32 bit. As the word length increases, the cost, power dissipation and speed of the microcontroller increases.
2. Power dissipation: It depends upon various factors like clock frequency, speed, supply voltage, VLSI technology etc. For battery operated embedded systems, we must use low power microcontrollers.
3. Clock frequency: The speed of an embedded system depends upon the clock frequency. The clock frequency depends upon the application.
4. Instruction Set: On the basis of instructions microcontrollers are classified into two categories 1. CISC 2. RISC.

CISC system improves software flexibility. Hence it is used in general purpose systems.

RISC improves speed of the system for the particular applications.

5. Internal resources: The internal resources are ROM, RAM, EEPROM, FLASH ROM, UART, TIMER, watch dog timer, PWM, ADC, DAC, network interface, wireless interface etc. It depends upon the application for which microcontroller is going to be used.
6. I/O capabilities: The number of I/O ports, size and characteristics of each I/O port, speed of operation of the I/O port, serial port or parallel ports. These are the considerations needed to ascertain correct selection of microcontroller.