# 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.

<table>
<thead>
<tr>
<th>Q. No.</th>
<th>Sub Q. N.</th>
<th>Answers</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Attempt any THREE:</td>
<td></td>
</tr>
<tr>
<td>i</td>
<td>Draw the format of TCON and state the function of each bit in it.</td>
<td></td>
</tr>
</tbody>
</table>

**Ans:**

**TCON: TIMER/COUNTER CONTROL REGISTER. BIT ADDRESSABLE.**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Symbol</th>
<th>TCON Bit Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>TF1</td>
<td>Timer 1 Overflow flag. Set when timer rolls from all 1's to 0. Cleared when processor vectors to execute interrupt service routine located at program address 001Bh.</td>
</tr>
<tr>
<td>6</td>
<td>TR1</td>
<td>Timer 1 run control bit. Set to 1 by program to enable timer to count; cleared to 0 by program to halt timer.</td>
</tr>
<tr>
<td>5</td>
<td>TF0</td>
<td>Timer 0 Overflow flag. Set when timer rolls from all 1's to 0. Cleared when processor vectors to execute interrupt service routine located at program address 001Bh.</td>
</tr>
</tbody>
</table>
program address 000Bh.

4  TR0  Timer 0 run control bit. Set to 1 by program to enable timer to count; cleared to 0 by program to halt timer.

3  IE1  External interrupt 1 Edge flag. Set to 1 when a high-to-low edge signal is received on port 3.3 (INT1). Cleared when processor vectors to interrupt service routine at program address 0013h. Not related to timer operations.

2  IT1  External interrupt 1 signal type control bit. Set to 1 by program to enable external interrupt 1 to be triggered by a falling edge signal. Set to 0 by program to enable a low-level signal on external interrupt 1 to generate an interrupt.

1  IE0  External interrupt 0 Edge flag. Set to 1 when a high-to-low edge signal is received on port 3.2 (INT0). Cleared when processor vectors to interrupt service routine at program address 0003h. Not related to timer operations.

0  IT0  External interrupt 0 signal type control bit. Set to 1 by program to enable external interrupt 0 to be triggered by a falling edge signal. Set to 0 by program to enable a low-level signal on external interrupt 0 to generate an interrupt.

<table>
<thead>
<tr>
<th>ii</th>
<th>Compare between RISC and CISC machines (any four points).</th>
<th>4M</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ans</td>
<td>Any 4 points – 1mark each</td>
<td></td>
</tr>
</tbody>
</table>
iii Write a program in 'C' to read a byte of data from Port 1 if it is greater than 99 send it to Port 0 otherwise send it to Port 2.

Ans : Program:

```c
#include <reg51.h>
void main(void)
{
    unsigned char x;
P1=0xFF;            //make P1 input port
while (1)
{
    x=P1;            //get a byte from P1
    if (x>99)        //check if a byte is greater than 99h
        {          
P0=x;         
}
else
    {                      
}
}
```

Correct program-4 marks
iv. Define Baud rate. State the Baud rate of each mode in UART.

Ans:

**Baud rate**: Baud rate is defined as the number of bits transmitted per sec.
OR
Baud rate is defined as the number of signal changes per second.

<table>
<thead>
<tr>
<th>Mode</th>
<th>Baud Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mode 0</td>
<td>(f/12)</td>
</tr>
<tr>
<td>Mode 1</td>
<td>Variable determined by Timer 1</td>
</tr>
<tr>
<td>Mode 2</td>
<td>(f/32) or (f/64)</td>
</tr>
<tr>
<td>Mode 3</td>
<td>Variable determined by Timer 1</td>
</tr>
</tbody>
</table>

Where \(f\) is oscillator frequency

b. Attempt any ONE:

i. Describe the function of following instructions in terms of length of bytes and operation.

   a) RRC A

   b) DIV AB

   c) JNB P1.3, DOWN

Ans:

a) **RRCA**

   The **RRC A** instruction rotates the eight bits in the accumulator and the one bit in the carry flag right one bit position. Bit 0 of the accumulator is rotated into the carry flag while the original value of the carry flag is rotated in to bit 7 of the accumulator. Bit 7 of the accumulator is rotated into bit 6, bit 6 into bit 5, and so on.

RRC

\[ A_n = A_{n+1} \text{ where } n = 0 \text{ to } 6 \]
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>A7 = C</strong></td>
<td></td>
</tr>
<tr>
<td><strong>C = A0</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Bytes: 1</strong></td>
<td></td>
</tr>
<tr>
<td><strong>b) DIV AB</strong></td>
<td></td>
</tr>
<tr>
<td>The <strong>DIV</strong> instruction divides the unsigned 8-bit integer in the accumulator by the unsigned 8-bit integer in register B. After the division, the quotient is stored in the accumulator and the remainder is stored in the B register. The carry and OV flags are cleared.</td>
<td></td>
</tr>
<tr>
<td><strong>AB = A / B</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Bytes: 1</strong></td>
<td></td>
</tr>
<tr>
<td><strong>c) JNB P1.3, DOWN</strong></td>
<td></td>
</tr>
<tr>
<td>The <strong>JNB</strong> instruction branches to the specified address (DOWN) if the bit P1.3 has a value of 0. Otherwise, execution continues with the next instruction. No flags are affected by this instruction.</td>
<td></td>
</tr>
<tr>
<td><strong>JNB</strong></td>
<td></td>
</tr>
<tr>
<td><strong>PC = PC + 3</strong></td>
<td></td>
</tr>
<tr>
<td><strong>IF (bit) = 0</strong></td>
<td></td>
</tr>
<tr>
<td><strong>PC = PC + offset</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Bytes: 3</strong></td>
<td></td>
</tr>
<tr>
<td><strong>ii</strong></td>
<td><strong>Draw the architecture of 8051 microcontroller.</strong></td>
</tr>
<tr>
<td><strong>Ans:</strong></td>
<td><strong>6M</strong></td>
</tr>
<tr>
<td><strong>Correct labelled diagram-4 marks</strong></td>
<td></td>
</tr>
</tbody>
</table>
### Question 2

**Attempt any TWO:**

<table>
<thead>
<tr>
<th>Q. No.</th>
<th>Sub Q. N.</th>
<th>Answers</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td></td>
<td>Write assembly language program to find largest number from the array of ten numbers stored in external memory RAM. Assume suitable data.</td>
</tr>
<tr>
<td>Ans:</td>
<td></td>
<td>Assumptions: Array of ten numbers are stored from memory location 3000H onwards and result is stored in memory location 6000H</td>
</tr>
</tbody>
</table>

**Program:**

```
CLR PSW.3                      ; Select Bank 0 PSW.3
MOV R1, 0AH                   ; Initialize byte counter
MOV DPTR, #3000H              ; Initialize memory pointer
DEC R1                          ; Decrement byte counter by 1
MOV X A, @DPTR                ; Load number in accumulator
MOV 40H, A                     ; Store number in memory location
UP: INC DPTR                  ; Increment memory pointer by 1
MOVXA, @DPTR                  ; Read next number
CJNE A, 40H, DN               ; if number ≠ next number, and then go to NEXT
DN: JNC NEXT                 ; If next number < number then go to NEXT
MOV 40H, A                     ; Else replace NEXT number with number
NEXT: DJNZ R1, UP             ; Decrement byte counter by 1, if byte counter ≠ 0 then go to UP
MOV DPTR,#6000H                ; Increment memory pointer by 1
MOV A, 40H                     ; Store result in external memory location
MOVX@ DPTR, A                 ; Store result in external memory location
LOOP: AJMP LOOP               ; Stop
```

**b** Draw interface diagram of ADC 0809 with 8051. Write 'C' language program to generate 8M
50 Hz sq. wave with crystal freq = 12 MHz.

Ans:

'C' language program to generate 50 Hz sq. wave with crystal freq = 12 MHz.:

Square wave Frequency = 50Hz
Therefore Time period T = 1 / 50HZ = 0.02s
Therefore TON = TOFF = 0.02s / 2 = 0.01s
(65535-count+1)1MHZ=0.01s
65536-count=0.01s/1µs
Count=65536-10000
Count=55536
Count=D8F0H
Program:
#include<reg51.h>
void delay(void);
sbit sq=P1^5;
void main (void)
{
    while (1)
    {
        sq=~sq;
        delay();
    }
}
void delay()
{
    TMOD=0X01;  //set timer 0 in mode 1 i.e. 16 bit number
    TL0=0X0F0;  //load TL register with LSB of count
    TH0=0X0D8;  //Load TH register with MSB of count
    TR0 =1;    //Start timer 0
    While(TF0==0); //wait until timer rolls over
    TR0=0;    //Stop timer 0
    TF0=0;    //Clear timer flag 0
}

c Draw interfacing diagram for stepper motor with 8051. Draw flow chart for rotating stepper motor in clockwise direction. 8M

Ans: Diagram 4 marks
Flowchart- 4 marks
START

Transfer first code to rotate motor by one step

call delay

Transfer second code to rotate motor by second step

call delay.

Transfer third code to rotate motor by third step.

call delay.

Transfer fourth code to rotate motor by forth step.

call delay.
Subject Name: Microprocessor & Applications

Q. No. | Sub Q. N. | Answers | Marking Scheme |
--- | --- | --- | --- |
3 | Attempt any FOUR: | 16- Total Marks |
3a | i) Convert $(1011101)_2$ to $( )_{10}$. |
| | ii) Subtract $(1001)_2$ from $(1100)_2$ by using 2's complement method. |
| Ans: | |

\[
(1011101)_2 = (1 \times 2^6) + (0 \times 2^5) + (1 \times 2^4) + (1 \times 2^3) + (0 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) \\
= 64 + 0 + 16 + 8 + 4 + 0 + 1 \\
= (93)_{10}
\]
(ii) \((1100)_2 - (1001)_2\)

1's complement of \((1001)_2 = 0110\)

\[\begin{array}{c}
2's \text{ comp} = +1 \\
\hline
0111
\end{array}\]

Now add, \((1100) + 0111\)

\[\begin{array}{c}
1100 \\
+ \quad 0111 \\
\hline
\quad 0011
\end{array}\]

Neglect the carry,

So the result is \((0011)_2 \rightarrow (3)_{10}\)

b) Draw program memory organization for i) \(E\overline{A} = 0\) ii) \(E\overline{A} = 1\).

Ans:

\[\begin{array}{c}
\text{XX51} \\
\text{EA} \\
\text{Vcc}
\end{array}\]

\[\begin{array}{c}
\text{FFFFF} \\
\text{FFFFF}
\end{array}\]

External Program Memory (60K)

External Program Memory (54K)

Program memory organization

4M

2M each
c) Compare between 8051 and 8052 microcontroller.

<table>
<thead>
<tr>
<th>Feature</th>
<th>8051</th>
<th>8052</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROM (bytes)</td>
<td>4K</td>
<td>8K</td>
</tr>
<tr>
<td>RAM (bytes)</td>
<td>128</td>
<td>256</td>
</tr>
<tr>
<td>Timers</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>I/O pins</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>Serial Port</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Interrupts</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>Watchdog timer</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

Ans:

1M each

---

d) Describe the standard data types in 'C' for 8051 with suitable example.

<table>
<thead>
<tr>
<th>DATA TYPES</th>
<th>SIZE IN BITS</th>
<th>DATA RANGE</th>
<th>EXAMPLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unsigned char</td>
<td>8-bit</td>
<td>0 to 255</td>
<td>Unsigned char x;</td>
</tr>
<tr>
<td>(signed) char</td>
<td>8-bit</td>
<td>-128 to +127</td>
<td>Signed char x;</td>
</tr>
<tr>
<td>Unsigned int</td>
<td>16-bit</td>
<td>0 to 65535</td>
<td>Unsigned int x;</td>
</tr>
<tr>
<td>(signed) int</td>
<td>16-bit</td>
<td>-32768 to +32767</td>
<td>Signed int x;</td>
</tr>
<tr>
<td>Sbit</td>
<td>1-bit</td>
<td>SFR bit addressable</td>
<td>Sbit LED=P1^2;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>only</td>
<td></td>
</tr>
<tr>
<td>Bit</td>
<td>1-bit</td>
<td>RAM bit addressable</td>
<td>Bit x;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>only</td>
<td></td>
</tr>
<tr>
<td>sfr</td>
<td>8-bit</td>
<td>RAM addresses</td>
<td>Sfr P1= 0x90;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>80-FFH only</td>
<td></td>
</tr>
</tbody>
</table>
**SUMMER–18 EXAMINATION**  
**Subject Name: Microprocessor & Applications**  
**Model Answer**

<table>
<thead>
<tr>
<th>Q. No.</th>
<th>Sub Q. N.</th>
<th>Answers</th>
<th>Marking Scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>a</td>
<td>Attempt any THREE:</td>
<td>12- Total Marks</td>
</tr>
<tr>
<td></td>
<td>i</td>
<td>State the alternate function of Port 3 pins of microcontroller 8051.</td>
<td>4M</td>
</tr>
</tbody>
</table>

**Ans:**

![Interlacing diagram of DAC 0808 with 8051 microcontroller.](image)
ii) Compare EPROM and EEPROM (any four points).

Ans:

1M each
### BASIS FOR COMPARISON

<table>
<thead>
<tr>
<th>Basic</th>
<th>EPROM</th>
<th>EEPROM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ultraviolet Light is used to erase the content ofEPROM.</td>
<td>EEPROM contents are erased using electronic signal.</td>
</tr>
<tr>
<td>Appearance</td>
<td>PROM has a transparent quartz crystal window at the top.</td>
<td>EEPROM are totally encased in an opaque plastic case.</td>
</tr>
<tr>
<td>Erased and Reprogrammed</td>
<td>PROM chip has to be removed from the computer circuit to erase and reprogram the computer BIOS.</td>
<td>EEPROM chip can be erased and reprogrammed in the computer circuit to erase and reprogram the content of computer BIOS.</td>
</tr>
<tr>
<td>Technology</td>
<td>PROM is an older technology.</td>
<td>PROM is a modern version over EEPROM.</td>
</tr>
</tbody>
</table>

### iii Write 'C' program to toggle only bit P1.2 continuously with 200 ms delay.

**Ans:**

(4M for correct program)

```c
#include <reg51.h>

sbit LED = P1^2;

void delay(unsigned int);

void main()
{
    while(1)
    {
        LED = 0;
        delay(200);
    }
}
```

4M
iv. Draw interfacing of LM 35 temperature sensor with 8051 microcontroller.

Ans:

Note: (Diagrams drawn using any other ADC (0808, 0848 etc.) also to be considered)
### Attempt any ONE:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>Write assembly language program to count time period of square wave using 8051 counter. Assume suitable data.</td>
<td>6M</td>
</tr>
</tbody>
</table>

**Ans**

Assumptions :

This program will display the time period in ms

Oscillator frequency = 12 MHz, Timer0 as counter in Mode1, Timer1 for delay generation in Mode1, Square wave whose time period to be found is applied to P3.4 (T0 input for Timer0), Higher byte of result is displayed in Port2 and lower byte in Port1.

Explanation: If we count the number of pulses applied to P3.4 in 1 ms, we will directly get the time period in milliseconds. Timer0 is used to count the pulses and Timer1 is used to generate a delay of 1ms.

The value to be given to TMOD register is 15H

The count value to generate time delay of 1ms is FC18H

Program:

- MOV TMOD,#15H  //LOAD 15H TO TOMOD REGISTER
- SETB P3.4  //P3.4 AS INPUT
- MOV TL0,#00H
- MOV TH0,#00H  //LOAD INITIAL VALUE FOR COUNTING
- SETB TR0  //START COUNTER
- MOV TL1,#18H
- MOV TH1,#0FCH  //LOAD COUNT VALUE FOR 1ms DELAY
- SAME: JB P3.4,SAME  //WAIT FOR FIRST FALLING EDGE OF SQUARE WAVE
- SETB TR1  //START TIMER1
- BACK: JNB TF1,BACK  //CHECK FOR 1ms DELAY
- CLR TF1  //CLEAR TF1
- CLR TR1  //STOP TIMER1
- CLR TR0  //STOP COUNTER0
- MOV A,TL0

Correct program: 6M
### SUMMER–18 EXAMINATION

**Subject Name:** Microprocessor & Applications  
**Model Answer**

<table>
<thead>
<tr>
<th>Q. No.</th>
<th>Sub Q. N.</th>
<th>Answers</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Attempt any TWO:</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>Describe the addressing modes of 8051 with suitable example.</td>
<td>8M</td>
</tr>
</tbody>
</table>

#### MOV P1,A  
// LOWER BYTE OF COUNT IN P1  
MOV A,TH0  
// HIGHER BYTE OF COUNT IN P2  
MOV P2,A  
SJMP $  

ii) Draw the diagram to interface external RAM and ROM with 8051 microcontroller and explain the function of ALE and PSEN pins of 8051.

**Ans:**

![Diagram](image)

PSEN pin stands for Program Store Enable. It is used to read a signal from the external program memory.

ALE/PROG: Address Latch Enable output pulse for latching the low byte of the address during accesses to external memory. This pin is also the program pulse input (PROG) during EPROM programming.

**Remarks:**

4 Marks (drawing)  
1 Mark each (function)
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 the R0. 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-bit signed 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 \((2^{16} = 64K)\). An example instruction is: LJMP 5000h; full 16 bit address is specified in operand

8) Indexed 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 is referred to as the index address.

<table>
<thead>
<tr>
<th>b</th>
<th>Write 'C' language program to transfer the message &quot;AICTE&quot; serially at baud rate 9600. Assume crystal frequency 11.0592 MHz. 8M</th>
</tr>
</thead>
</table>
| Ans | Normally Serial communication MODE 1 is used (8 bit UART with variable baud rate). For setting baud rate, TIMER 1 to be programmed in Mode 2 – Auto reload mode. Following formula is used for calculation of Count to be given to Timer 1 register, TH1 to set standard baud rate.

\[
f_{\text{baud}} = \frac{2^{\text{smod}} \times f_{\text{osc}}}{12 \times (256 - TH1)}
\]

Calculatio

<table>
<thead>
<tr>
<th>n of count</th>
<th>– 1M</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program :</td>
<td>7M</td>
</tr>
</tbody>
</table>
Hence for 9600 baud, \[ 9600 = \frac{2^0}{32} \times \frac{11059200}{12 \times (256-TH1)} \]
From this formula we get value to be loaded to TH1 as FD H
(Any other logically similar method may also be used for calculation)

Program:
```c
#include <REG51.H>
void main(void)
{
  unsigned char text[] = "AICTE";
  unsigned char i;
  TMOD = 0x20;
  TH1 = 0xFD;
  SCON = 0x50;
  TR1 = 1;
  while(1)
  {
    for(i =0;i<5;i++)
    {
      SBUF= text[i];
      while(TI ==0);
      TI = 0;
    }
  }
}
```

**c** Draw and explain the interfacing of seven segment LED display in common cathode with 8051 microcontroller. Write 'C' language program to display digit 0 to 9.

**Ans:**

A = P1.0  B = P1.1  C = P1.2  D = P1.3

E = P1.4  F = P1.5  G = P1.6  DP = P1.7

DB 3FH  // digit drive pattern for 0  
DB 06H  // digit drive pattern for 1  
DB 5BH  // digit drive pattern for 2  
DB 4FH  // digit drive pattern for 3  
DB 66H  // digit drive pattern for 4  
DB 6DH  // digit drive pattern for 5  
DB 7DH  // digit drive pattern for 6  
DB 07H  // digit drive pattern for 7  
DB 7FH  // digit drive pattern for 8  
DB 6FH  // digit drive pattern for 9

void main ()
```c
{ 
    P1 = 0xFF;  //DISPLAY OFF
    while(1)
    {
        P1 = 0x3F;  //DISPLAY 0
        delay_ms(1000);
        P1 = 0x06;  //DISPLAY 1
        delay_ms(1000);
        P1 = 0x5B;  //DISPLAY 2
        delay_ms(1000);
        P1 = 0x4F;  //DISPLAY 3
        delay_ms(1000);
        P1 = 0x66;  //DISPLAY 4
        delay_ms(1000);
        P1 = 0x6D;  //DISPLAY 5
        delay_ms(1000);
        P1 = 0x7D;  //DISPLAY 6
        delay_ms(1000);
        P1 = 0x07;  //DISPLAY 7
        delay_ms(1000);
        P1 = 0x7F;  //DISPLAY 8
        delay_ms(1000);
        P1 = 0x6F;  //DISPLAY 9
        delay_ms(1000);
    }
```
NOTE: Program may change. Student can also use the other logic. Please check the logic and understanding of students.

**Q. No.** Attempt any FOUR: **16- Total Marks**

<table>
<thead>
<tr>
<th>Sub Q. N.</th>
<th>Answers</th>
<th>Marking Scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>a) Draw the format of IE register of 8051 and state the function of each bit.</td>
<td>4M</td>
</tr>
</tbody>
</table>

**Ans:**

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.

<table>
<thead>
<tr>
<th>EA</th>
<th>—</th>
<th>—</th>
<th>ES</th>
<th>ET1</th>
<th>EX1</th>
<th>ET0</th>
<th>EX0</th>
</tr>
</thead>
</table>

- **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.*
- — IE.5 Not implemented, reserved for future use.*
- **ES** IE.4 Enable or disable the serial port interrupt.
- **ET1** IE.3 Enable or disable the Timer 1 overflow interrupt.
- **ET0** IE.2 Enable or disable External Interrupt 1.
- **EX1** IE.1 Enable or disable the Timer 0 overflow interrupt.
- **EX0** IE.0 Enable or disable External Interrupt 0.

<table>
<thead>
<tr>
<th>b) Describe the following assembler directives with one example.</th>
<th>4M</th>
</tr>
</thead>
<tbody>
<tr>
<td>i) ORG</td>
<td>1 mark each</td>
</tr>
<tr>
<td>ii) DB</td>
<td></td>
</tr>
<tr>
<td>iii) EQU</td>
<td></td>
</tr>
<tr>
<td>iv) END</td>
<td></td>
</tr>
</tbody>
</table>

**Ans:**

(Each directive explanation with example: 1 mark each)

1. **ORG**: ORIGIN The ORG directive is used to indicate the beginning of the address. The
number that comes after ORG is the address from where program will begin. The number can be either in hex and decimal

e.g. ORG 1000H

It indicates that program shall start from memory address 1000h

2. **DB**: Define Byte It is used to define the 8-bit data. It is used to write the value after DB, into the program memory. When DB is used to define data, the numbers can be in decimal, binary, hex, ASCII formats

   e.g. ORG 1000H

   MYDATA: DB 20,21

   After execution of this, location 1000h=20 & 1001h = 21

3. **EQU**: EQUATE It is used to define a constant without occupying a memory location  
   The EQU directive assigns a constant value to a label. When the label appears in the program, its constant value will be substituted for the label.

   e.g. COUNT EQU 10

   MOV R2, #COUNT

   When the instruction is executed, register R2 is loaded with value 10.

4. **END**: This indicates to the assembler the end of the source (asm) file. The END directive is the last line of an 8051 program. Means that in the program anything after the END directive is ignored by the assembler

   e.g. MOV A, #20

   ADD A, #10

   END

c) **Write down the function of 16 pin connector of LCD module.**

<table>
<thead>
<tr>
<th>Pin</th>
<th>Pin Name:</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No.</td>
<td>Pin Name</td>
<td>Description</td>
</tr>
<tr>
<td>-----</td>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>1</td>
<td>Vss (Ground)</td>
<td>Ground pin connected to system ground</td>
</tr>
<tr>
<td>2</td>
<td>Vdd (+5 Volt)</td>
<td>Powers the LCD with +5V (4.7V – 5.3V)</td>
</tr>
<tr>
<td>3</td>
<td>VE (Contrast V)</td>
<td>Decides the contrast level of display. Grounded to get maximum contrast.</td>
</tr>
<tr>
<td>4</td>
<td>Register Select</td>
<td>Connected to Microcontroller to shift between command/data register</td>
</tr>
<tr>
<td>5</td>
<td>Read/Write</td>
<td>Used to read or write data. Normally grounded to write data to LCD</td>
</tr>
<tr>
<td>6</td>
<td>Enable</td>
<td>Connected to Microcontroller Pin and toggled between 1 and 0 for data acknowledgement</td>
</tr>
<tr>
<td>7</td>
<td>Data Pin 0</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Data Pin 1</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Data Pin 2</td>
<td>Data pins 0 to 7 forms a 8-bit data line. They can be connected to Microcontroller to send 8-bit data.</td>
</tr>
<tr>
<td>10</td>
<td>Data Pin 3</td>
<td>These LCD’s can also operate on 4-bit mode in such case Data pin 4,5,6 and 7 will be left free.</td>
</tr>
<tr>
<td>11</td>
<td>Data Pin 4</td>
<td></td>
</tr>
</tbody>
</table>
d) Draw and describe the interfacing of opto-isolator with 8051 microcontroller.

Ans:

Optocoupler is also called optoisolator, photocoupler and optical isolator. It is used to provide isolation between two electrical circuits. It is a electrical component which is used to transmit input signal by using light energy signals. It provides electrical coupling between input and output through light waves. Its main purpose is to avoid changing in voltages at output side to appearing at input side. Higher voltage fluctuations may
damage input side electrical components.
The output of opto coupler is given to relay which further operates the output device.

**e** Explain system clock and machine cycle of 8051 microcontroller.  

**Ans:**

An 8051 clock circuit is shown above. In general cases, a quartz crystal is used to make the clock circuit. The connection is shown in figure (a) and note the connections to XTAL 1 and XTAL 2. In some cases external clock sources are used and you can see the various connections above. Clock frequency limits (maximum and minimum) may change from device to device. Standard practice is to use 12MHz frequency. If serial communications are involved then its best to use 11.0592 MHz frequency.
One machine cycle has 6 states. One state is 2 T-states. Therefore one machine cycle is 12 T-states. Time to execute an instruction is found by multiplying C by 12 and dividing product by Crystal frequency.

\[ T = \frac{(C \times 12d)}{\text{crystal frequency}} \]