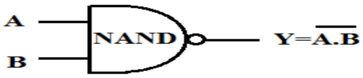
WINTER – 19EXAMINATIONS

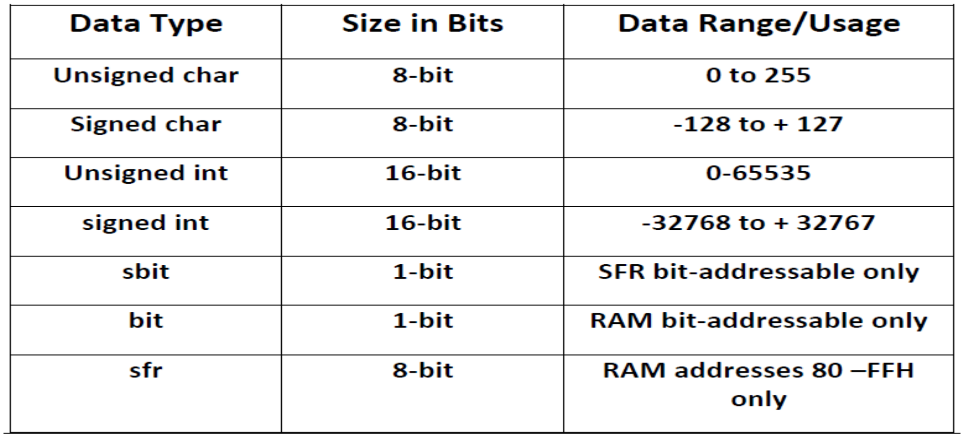
Subject Name:Microcontroller and ApplicationsModel Answer Subject Code: 17509

Important Instructions to examiners:

1) The answers should be examined by key words and not as word-to-word as given in themodel 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 moreImportance (Not applicable for subject English and Communication Skills.
4) While assessing figures, examiner may give credit for principal components indicated in thefigure. The figures drawn by candidate and model answer may vary. The examiner may give credit for anyequivalent 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 judgment 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>
<tr><th>Q. No.</th><th>Sub Q. N.</th><th>Answer</th><th>Marking Scheme</th></tr>
<tr><td><b>Q.1</b></td><td><b>(A)</b></td><td><b>Attempt any <u>THREE</u> of the following:</b></td><td><b>12- M</b></td></tr>
<tr><td></td><td><b>a)</b></td><td><b>Draw symbol of NAND gate and write its truth table.</b></td><td><b>4M</b></td></tr>
<tr><td></td><td><b>Ans:</b></td><td><b>Symbol:</b>


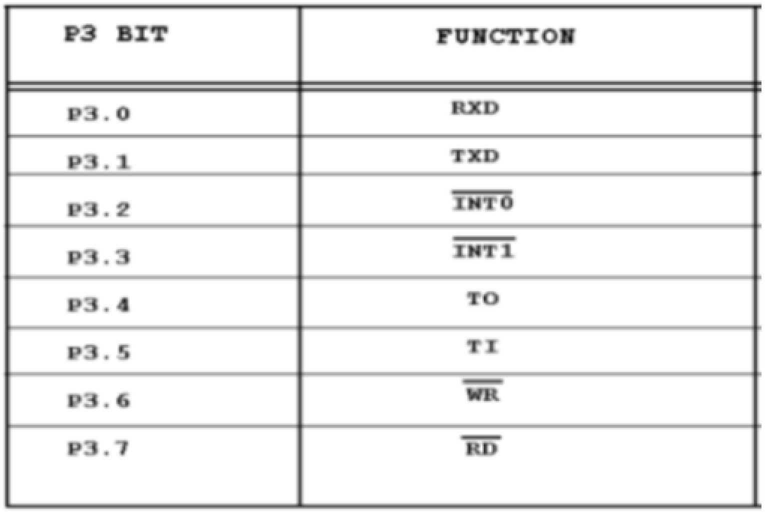
<b>Truth Table:</b>
<table>
<tr><th colspan="2">Inputs</th><th>Output</th></tr>
<tr><td>A</td><td>B</td><td>Y=$\overline{A.B}$</td></tr>
<tr><td>0</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>0</td></tr>
</table></td><td><b>Symbol :2M Truth Table: 2M</b></td></tr>
<tr><td></td><td><b>b)</b></td><td><b>State function of following pins of 16*2 LCD.</b>
<b>(i) RS</b>
<b>(ii) R/$\overline{W}$</b>
<b>(iii)EN</b>
<b>(iv)LED+</b></td><td><b>4M</b></td></tr>
<tr><td></td><td><b>Ans:</b></td><td>RS: RS is the register select pin used to write display data to the LCD (characters), this pin hasto be high when writing the data to the LCD. During the initializing sequence and other commands this pin should be low.
R/W: Reading and writing data to the LCD, for reading the data R/W pin should be high (R/W=1) to write the data to LCD R/W pin should be low (R/W=0).
EN: Enable pin is for starting or enabling the module. A high to low pulse of about 450ns Pulse is given to this pin. Sends data to data pins when a high to low pulse is given at this pin.</td><td><b>1M each</b></td></tr>
</table>

		iv)LED+ It is pin no 15, inputpin. Backlight LED pin positive terminal.	
	c)	**List any four C data types with its size and ranges.**	**4M**
	Ans:		**1M each**

Data Type	Size in Bits	Data Range/Usage
Unsigned char	8-bit	0 to 255
Signed char	8-bit	-128 to + 127
Unsigned int	16-bit	0-65535
signed int	16-bit	-32768 to + 32767
sbit	1-bit	SFR bit-addressable only
bit	1-bit	RAM bit-addressable only
sfr	8-bit	RAM addresses 80 –FFH only

	d)	**Write function of following pins of 8051µc.** **(i) RST** **(ii) $\overline{PSEN}$** **(iii)RXD** **(iv)$\overline{EA}$**	**4M**
	Ans:	**(i) RST** It is a RESET pin, which is used to reset the microcontroller to its initial values. **(ii) $\overline{PSEN}$** It is active low output control signal used to activate enable signal of external ROM/ EPRM .it is activated every six oscillator periods while reading the external memory. **(iii)RXD** Serial input line (Receive).RXD pin is pin no 10 and input pin to the microcontroller. It is used to input serial data to the microcontroller. **iv) $\overline{EA}$** It is active low output control signal. When EA = 1, µc accesses internal and external program memory when EA =0, µc accesses only external program memory.	**1M each pin**
	(B)	**Attempt any <u>ONE</u> of the following:**	**6M**
	a)	**State alternate functions of Port 3.**	**2M**

Ans:



P3 BIT	FUNCTION
P3.0	RXD
P3.1	TXD
P3.2	$\overline{\text{INT0}}$
P3.3	$\overline{\text{INT1}}$
P3.4	TO
P3.5	TI
P3.6	$\overline{\text{WR}}$
P3.7	$\overline{\text{RD}}$

b) **Describe addressing modes of 8051 with examples.** **4M**

Ans: **Addressing modes of 8051:**

1.Immediate Addressing mode
2. Register Addressing mode
3. Direct Addressing mode
4 Register Indirect addressing mode
5.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.

For 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 instruction.

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:

In Indirect addressing mode, the data is obtained from a memory location which is indirectly specified in the instruction.

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

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) 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 us referred to as the index address.

Q.2 **Attempt any TWO of the following:** **16- M**

a) **Write an ALP to find largest number from given array of 10 bytes in external RAM location 2000h onwards. Store largest number in internal RAM location 40h.** **8M**

Ans :

```
CLR PSW.3                      ; Select Bank 0 PSW.3
MOV R1, 0AH                    ; Initialize byte counter
MOV DPTR, # 2000H              ; Initialize memory pointer
DEC R1                         ; Decrement byte counter by 1
MOV X A, @DPTR                 ; Load number in accumulator
MOV 40 H, A                    ; Store number in memory location
UP: INC DPTR                   ; Increment memory pointer by 1
MOVXA, @DTPR                   ; Read next number
CJNE A, 40 H, DN               ; if number≠ next number, and then go to NEXT
DN: JC 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
LOOP: AJMP LOOP                ; Stop
```

Correct program: 8M

b) **Draw interfacing diagram of DAC 0808 with 8051µC and write C program to generate triangular wave.** **8M**

Ans :

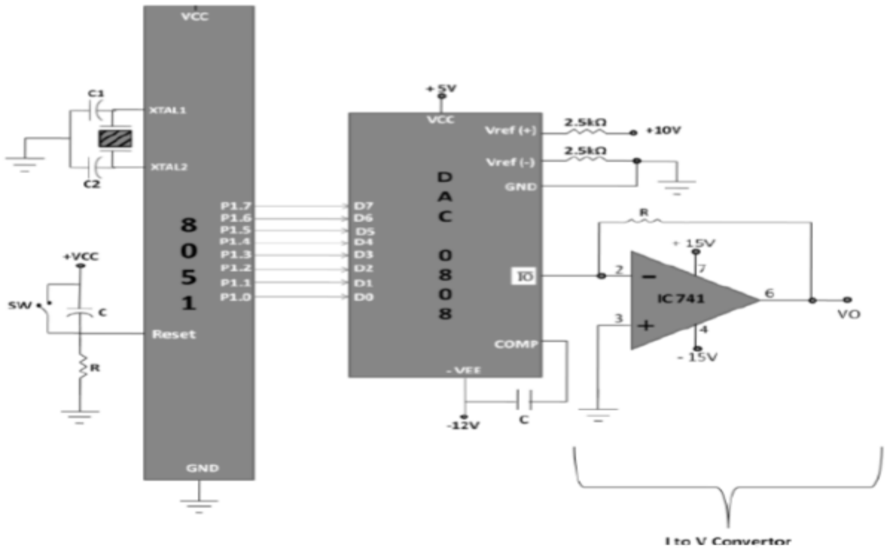


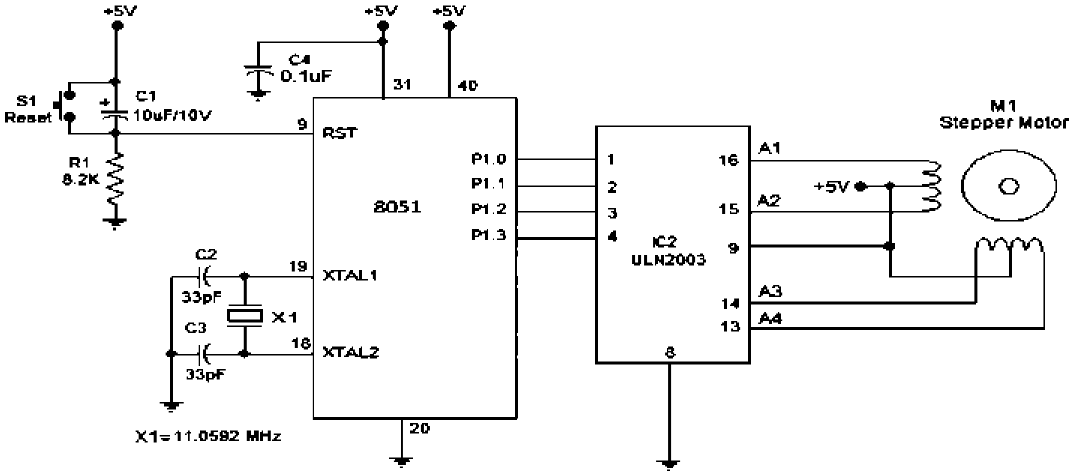
Diagram: 4M

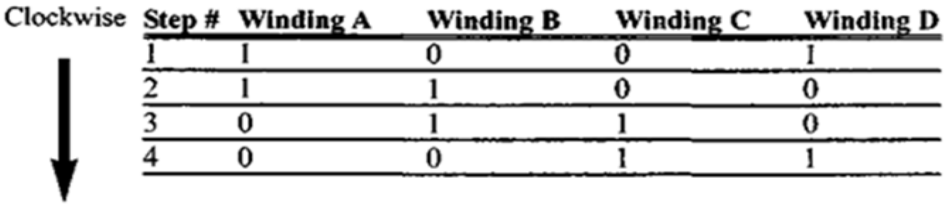
```
#include<reg51.h>
void main(void)
{
unsigned char d;
while(1)
{
for(d=0; d<255; d++)
{
P1 = d;
}
for(d=255; d>0; d--)
{
P1 = d;
}
}
}
```

Program :4M

c) **Draw the interfacing diagram of stepper motor with 8051. Write excitation code to rotate it in clockwise direction.** **8M**

Ans :





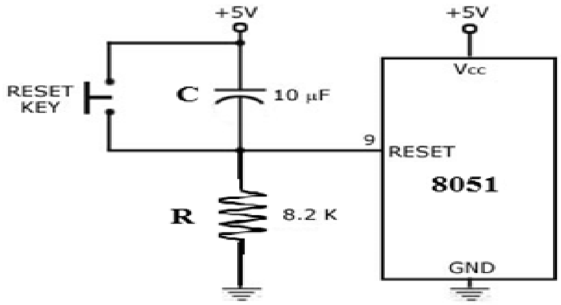
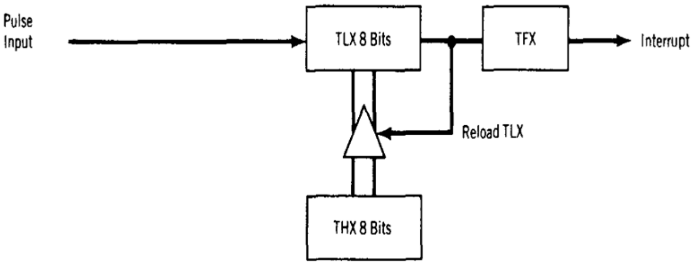
Clockwise

Step #	Winding A	Winding B	Winding C	Winding D
1	1	0	0	1
2	1	1	0	0
3	0	1	1	0
4	0	0	1	1

Diagram: 6M

Code:2M

Q.3		**Attempt any <u>FOUR</u> of the following:**	**12- M**
	a)	**Draw and explain Reset circuit of 8051µC.**	**4M**

	Ans :	 +5V +5V Vcc RESET KEY C 10 µF 9 RESET 8051 R 8.2 K GND **Power on & manual reset circuit** The power on reset circuit consists of 8.2 KΩ resistor and 10 µF capacitor. The values of these components are sufficient to provide a delay to make RST pin high for two machine cycles. For manual reset function switch is provided. Upon power ON or Key Press the RST pin goes HIGH and as capacitor charges through resistor R, RST signal goes LOW. This generates active high reset signal for specific time decided by values of R & C.	**2 marks: circuit diagram** **2marks : explanation**
	b)	**Describe mode 2 of timer. State application of it.**	**4M**
	Ans :	**Mode 2 – 8 bit Auto Reload** TL operates as an 8-bit Timer / counter.TH holds a reload value. When TL overflows (Reached FFH), the TFx flag is set, TL is reloaded from the value in TH and counting continues.  Pulse Input TLX 8 Bits TFX Interrupt Reload TLX THX 8 Bits **Application:** To generate baud rate in serial communication	**3Marks: mode 2 description** **1mark : application**
	c)	**Write C program to toggle bits of P2. Use software delay.**	**4M**

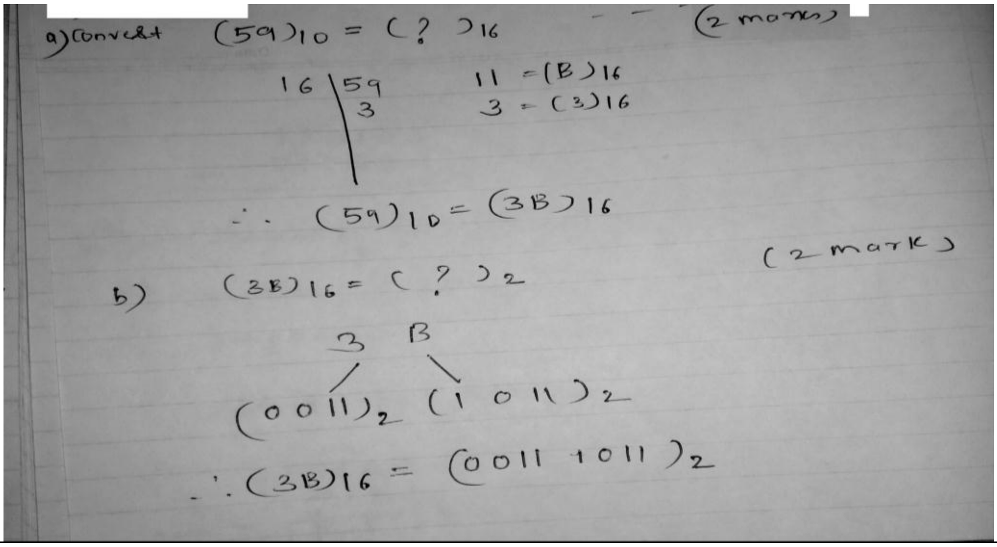
	Ans :	#include <reg51.h>
		void delay(unsigned int);
		void main(void)
		{
		P2=0X00; // PORT 2 as output port
		while(1)
		{
		P2=0X00;
		delay(200);
		P2=0XFF;
		delay (200);
		}
		}
		void delay(unsigned int t)
		{
			4M for correct program) Any amount of delay can be considered

```
unsigned inti,j;
for(i=0;i<=t;i++)
for(j=0;j<=1275;j++);
}
```

OR

```
#include <reg51.h>
void delay(unsigned int);
void main(void)
{
P2=0X00; // PORT 2 as output port
while(1)
{
P2= ~ P2;
delay(200);
}
}
void delay(unsigned int t)
{
unsigned inti,j;
for(i=0;i<=t;i++)
for(j=0;j<=1275;j++);
}
```

ANY OTHER CORRECT PROGRAM LOGIC SHOULD BE GIVEN MARKS

	d)	**Convert $(59)_{10}$ = $(?)_{16}$ = $(?)_2$.**	**4M**
	Ans :		

a) Convert $(59)_{10} = (?)_{16}$ — (2 marks)

$16 \,|\, 59$ 3 $\quad 11 = (B)_{16}$ $\quad 3 = (3)_{16}$

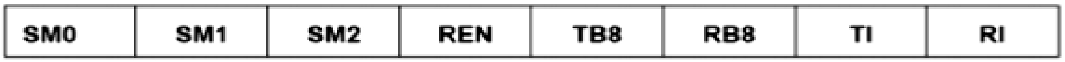
$\therefore (59)_{10} = (3B)_{16}$

b) $(3B)_{16} = (?)_2$ (2 mark)

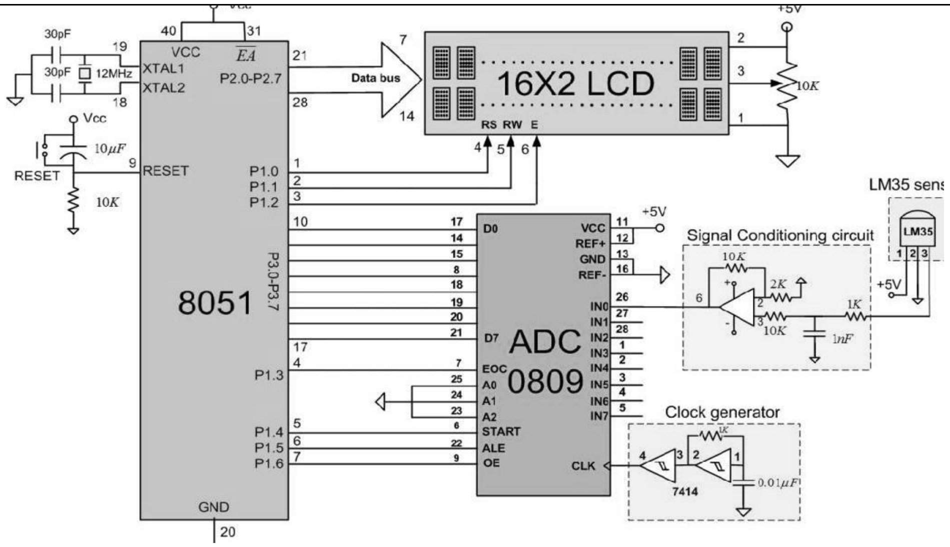
3 B

$(0011)_2 \quad (1011)_2$

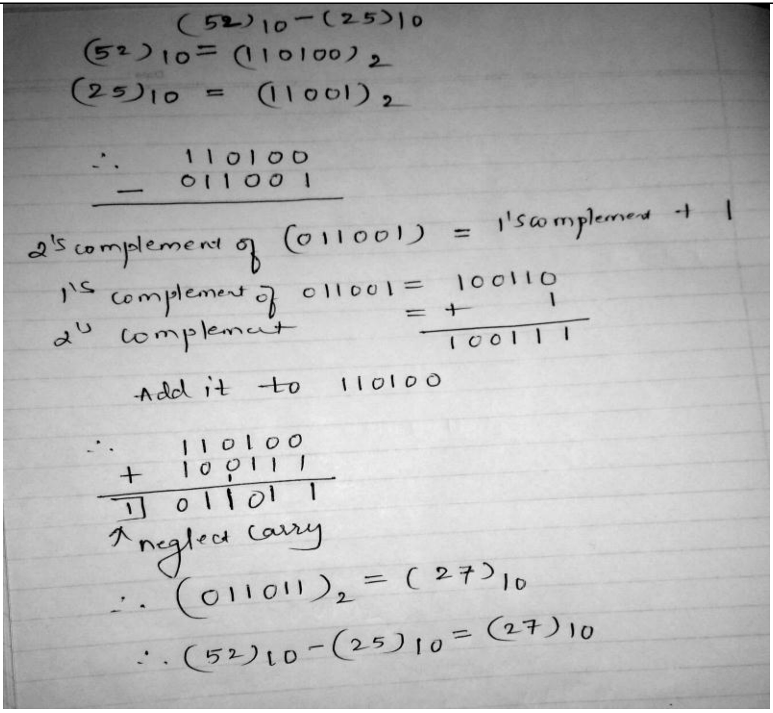
$\therefore (3B)_{16} = (0011\ 1011)_2$

	e)	**Draw the format of SCON SFR.**	**4M**
	Ans :	SCON Register format- 	**Correct format 4M**

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

Q.4	**(A)**	**Attempt any THREE of the following :**	**12- M**
	a)	**Draw interfacing diagram for temperature measurement using LM 35, ADC 0808 with 8051 microcontroller.**	**4M**
	Ans :	 Vcc 30pF 19 12MHz XTAL1 XTAL2 18 10μF RESET 9 10K 40 31 VCC EA P2.0-P2.7 21 28 Data bus 7 14 16X2 LCD +5V 2 3 1 RS RW E 4 5 6 P1.0 P1.1 P1.2 8051 P3.0-P3.7 P1.3 P1.4 P1.5 P1.6 GND 20 D0 D7 ADC 0809 EOC A0 A1 A2 START ALE OE CLK VCC REF+ GND REF- IN0 IN1 IN2 IN3 IN4 IN5 IN6 IN7 Signal Conditioning circuit LM35 sens LM35 2K 1K 1nF Clock generator 7414 0.01μF	**Correct diagram 4M**
	b)	**Explain bitwise shift operator with example.**	**4M**
	Ans :	**Bitwise Left Shift Operator in C : <<** [variable]<<[Number of Places] P0=0x3C<< 2 After execution of this instruction Shift number 2 bits to left: 3C = 0011 1100 1st left shift = 0111 1000 2nd left shift =1111 0000 So, P0=0xF0 **Bitwise Right Shift Operator in C: >>** [variable]>>[number of places] P0=0x3C >> 2 After execution of this instruction Shift number 2 bits to Right: 3C=0011 1100 1st right shift = 0001 1110 2nd right shift = 0000 1111 So, P0=0x0F	**2marks: left shift operator explanation** **2marks: Right shift operator explanation**
	c)	**Subtract $(25)_{10}$ from $(52)_{10}$ using 2's compliment method.**	**4M**

Ans :



$(52)_{10} - (25)_{10}$

$(52)_{10} = (110100)_2$

$(25)_{10} = (11001)_2$

$$\therefore \begin{array}{r} 110100 \\ -\ 011001 \\ \hline \end{array}$$

2's complement of (011001) = 1's complement + 1

1's complement of 011001 = 100110

2's complement $= \begin{array}{r} +\quad 1 \\ \hline 100111 \end{array}$

Add it to 110100

$$\therefore \begin{array}{r} 110100 \\ +\ 100111 \\ \hline \boxed{1}\ 011011 \end{array}$$

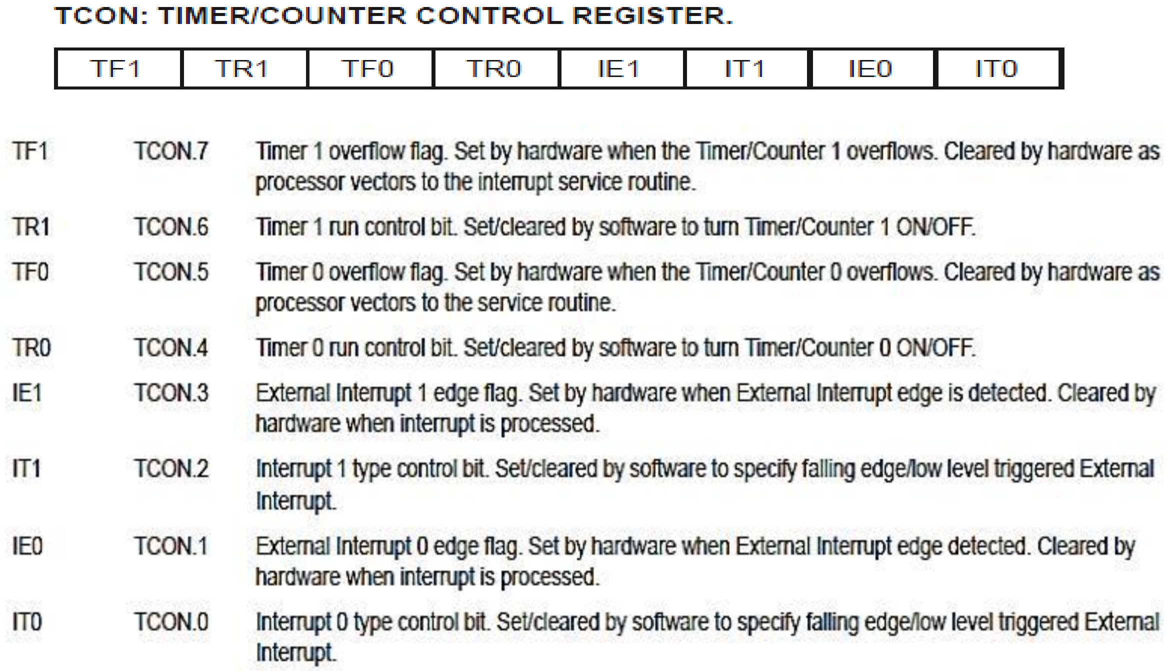
↑ neglect carry

$\therefore (011011)_2 = (27)_{10}$

$\therefore (52)_{10} - (25)_{10} = (27)_{10}$

d) **Draw the format of TCON sfr and explain each bit.** **4M**

Ans :



TCON: TIMER/COUNTER CONTROL REGISTER.

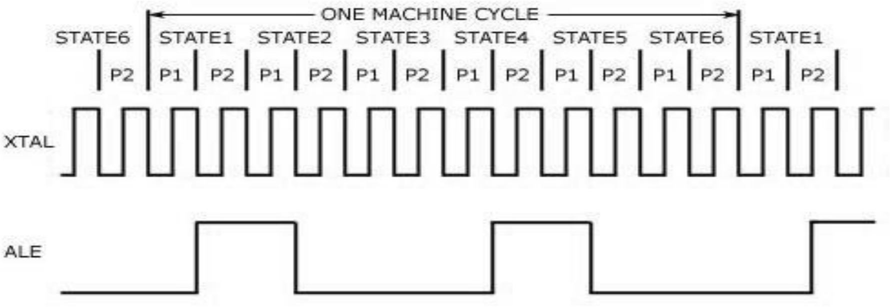
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

TF1	TCON.7	Timer 1 overflow flag. Set by hardware when the Timer/Counter 1 overflows. Cleared by hardware as processor vectors to the interrupt service routine.
TR1	TCON.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.

Format-2 marks
Function - 2marks

(B) **Attempt any ONE of the following:** **6M**

a) **Explain T-state, Machine cycle and instruction cycle.** **6M**

	Ans :	 ONE MACHINE CYCLE STATE6 STATE1 STATE2 STATE3 STATE4 STATE5 STATE6 STATE1 P1 P2 XTAL ALE One T-state is the time period of one clock signal. It is the reciprocal of system clock frequency. Machine cycle is the minimum time taken by microcontroller to perform an operation. 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, called instruction cycle is found by multiplying C by 12 and dividing product by Crystal frequency. T=(C*12)/crystal frequency **Where C is number of machine cycles**	**Explanation: 2 marks each.**
	b)	**Explain stack memory. Write any two stack related instruction.**	**6M**
	Ans :	1. The stack memory is part of RAM used by the CPU to store information temporarily. 2. This information may be either data or address. 3. The CPU needs this storage area as there are only a limited amount of registers. 4. The register used to access stack memory is called stack pointer. 5. Upon reset SP contains 07H; this causes the stack to begin to location 08H. So, Register banks 2, 3, 4 (08H to 1FH) form the default stack area. 6. The stack is generally placed in the general-purpose area (30H to 7FH) of the internal RAM. **Stack Related Instructions: (any two)** a) PUSH b) POP c) CALL (ACALL, LCALL) d) RET	**4 marks: Stack memory explanation Writing Any two instructions: 2 marks: (1 mark each instruction)**
Q.5		**Attempt any <u>TWO</u> of the following :**	**16- M**
	a)	**Write C program to transmit 'MSBTE' on TXD line.**	**8M**
	Ans :	Baud Rate Calculation: Timer Value = $\frac{2^{SMOD} \times Oscfreq}{12 \times 32 \times Re\,quired\ Baud\ rate}$ Considering SMOD = 1	**2Marks for Calculation**

Timer Value = $\dfrac{2^{0} \times Oscfreq}{12 \times 32 \times \text{Re}\,quired\ Baud\ rate}$

Oscfreq = 11.0592Mhz

Timer Value = $\dfrac{11.059Mhz}{12 \times 32 \times \text{Re}\,quired\ baud\ rate}$

Case – 1) Required Baud rate = 4800

Time value = $\dfrac{28{,}800}{4800}$

Timer value = 6

- 6 must be loaded in Timer for Required Delay.

Case – 2) Required Baud rate =9600

Timer value = 28,800 / 9600

Timer value = 3

-3 must be loaded in Timer for Required Delay

C Program:

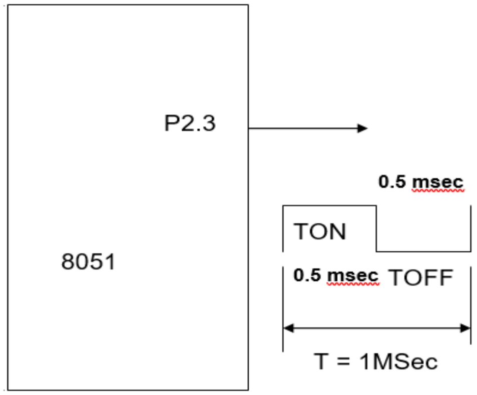
```
#include <REG51.h>
void Trans(unsigned char x);
void main(void)
{
TMOD = 0X20;
TH1 = -3;                                   // for 9600 Baud rate
SCON = 0X50;
TR1 = 1;
while(1)
     {
              Trans(‘  ‘);
              Trans(‘M');
              Trans(‘S');
              Trans(‘B');
              Trans(‘T');
              Trans(‘E’);
       }
void Trans(unsigned char x)
  {
      SBUF = x;
      while(TI = = 0)
        {      }
       TI=0;
  }
```

1 Mark for calculation and 5M For Program

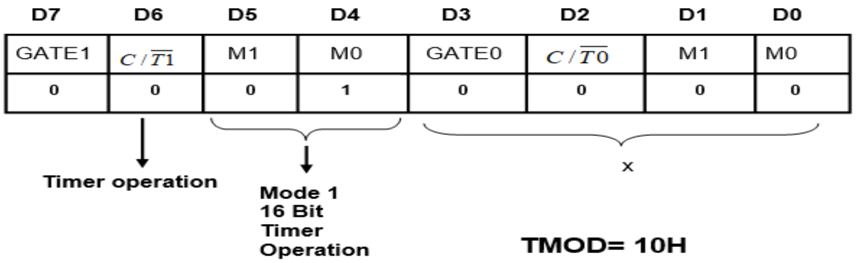
	b)	**Write an ALP to generate square wave of 1kH$_z$ frequency on p2.3, Use timer 1 in model1. f$_{OSC}$= 12 MH$_z$**	**8M**

Ans :

2Marks for TMOD



TMOD:



D7	D6	D5	D4	D3	D2	D1	D0
GATE1	$C/\overline{T1}$	M1	M0	GATE0	$C/\overline{T0}$	M1	M0
0	0	0	1	0	0	0	0

Timer Calculation for 1Khz :

Required Frequency f= 1khz

Time Period T = $\frac{1}{f} = \frac{1}{1\times10^3} = 1msec$

T_{0N} = T_{0FF} = 0.5msec

2Marks for Calculations

To calculate Timer reload value for 0.5msec following formula is used

Timer Value = $(65536 - \frac{\text{Re}\,quiredTimedelay \times OscFreq}{12})$

Timer Value = $(65536 - \frac{0.5m\,sec \times 12 \times 10^6}{12})$

Timer Value = $(65036)_{10}$

Timer Value = $(FE0C)_{16}$

4Marks for Program

Assembly Program:

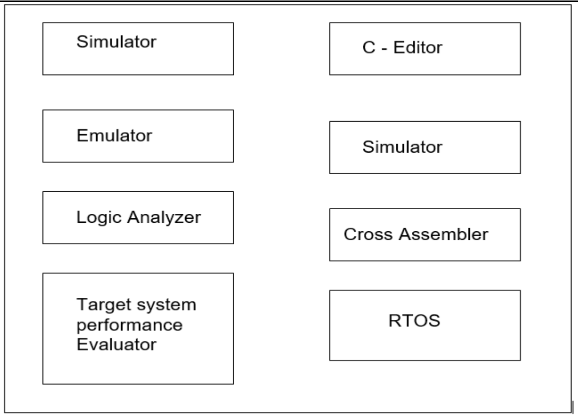
```
                  ORG 30H                        ; Main Program starting
location
               MOV  TMOD,#10H                   ; Timer 1, mode 1
     AGAIN:    MOV  TL1, #0CH                  ; Timer value
               MOV  TH1, #0FEH
               SETB TR1                         ; Start Timer
     BACK:     JNB  TF1, BACK              ; Wait till timer overflows
               CLR  TR1                       ; Stop Timer
               CPL  P2.3                     ; Get Next State of Square wave
               CLR  TF1                       ; Clear timer flag 1
               SJMP AGAIN                          ;  Reload timer & Continue
```

c)	Describe IDE with its components and state their functions.	8M

Ans:

2Marks

Diagram



6Marks

Explanation

- In Embedded systems Code generation tools are used for creating and compiling. Then codes are tested using simulators and a number of latest software tolls like simulators , Logic Analyzers, profiler, Emulators etc.
- When all of these programs are integrated in one software package then it is called as Integrated Development environment (IDE)
- Integrated Development Environment (IDE) consists of simulators with editors , compilers, assemblers, emulators, logic analyzers .

<u>IDE Components:</u>

<u>Editor:</u>

- You can type your assembly program-using editor.
- An editor is a program which helps you to construct your assembly language program in right format so that the assembler will translate it correctly to machine language. This form of your program is called as **source program**.
- The assembly program written using DOS Editor is stored as **.asm extension &**The C Program written using DOS Editor is stored as **.C extension.**

<u>Cross Assembler:</u>

- An Cross Assembler is program that allows an Assembly program written on one type of microcontroller to be used on another type.

<u>Simulator:</u>

- Simulator Simulates (Duplicates) the behavior of Target Hardware (Microcontroller) in Software.
- Provides the detailed information of the status of RAM and ports (simulated) of the defined target system and can execute each instruction in Single step mode.

<u>Emulation:</u>

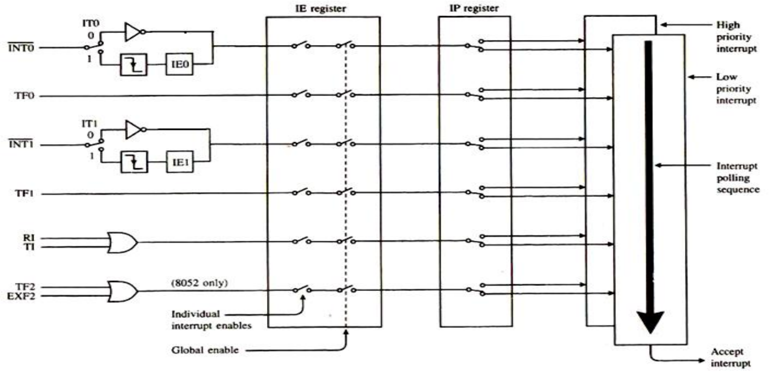
- An **emulator** in computer sciences duplicates (provides an **emulation** of) the functions of one system using a different system, so that the second system behaves like (and appears to be) the first system.

<u>Logic Analyzer:</u>

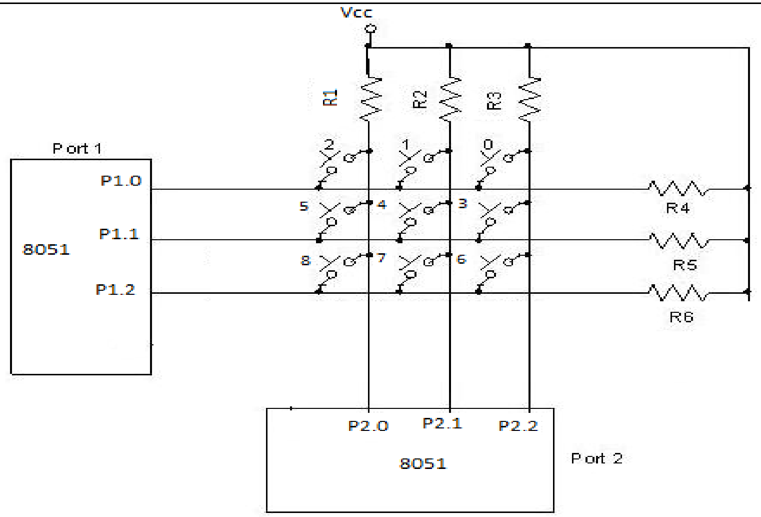
- A **logic analyzer** is an electronic instrument that captures and displays multiple signals from a digital system or digital circuit. A logic analyzer may convert the captured data into timing diagrams.

<u>RTOS:</u>

- RTOS are used in system to execute any task in defined time limits. It has following functions.
  1. Memory Management
  2. File Management
  3. Port Management

		4. Process Management 5. I/O Management **Target Process Evaluator:** - Target Evaluation is the systematic process of gathering and analyzing data and other objective information on processes and outcomes to determine the quality, value, and effectiveness of coding & performance improvement.	
Q.6		**Attempt any FOUR of the following:**	**16-M**
	a)	**Draw structure of Interrupt and explain it.**	**4M**
	Ans :	**Interrupt Structure:**  IE register IP register IT0 0 1 $\overline{\text{INT0}}$ IE0 TF0 IT1 0 1 $\overline{\text{INT1}}$ IE1 TF1 RI TI TF2 EXF2 (8052 only) Individual interrupt enables Global enable High priority interrupt Low priority interrupt Interrupt polling sequence Accept interrupt There are five interrupt sources on the 8051: 1. External 0 Interrupt 2. Timer 0 Interrupt 3. External 1 Interrupt 4. Timer 1 Interrupt 5. Serial Interrupt All Interrupt are disabled after a system reset and are enabled individually by software. In the event of two or more simultaneous interrupts or an interrupt occurring while another interrupt is being serviced, there is both a **polling sequence** and a **two level priority scheme** to schedule the interrupts. The polling sequence is fixed but the interrupt priority is programmable. As shown in the interrupt structure External 0 / External 1 interrupts can be **level triggered or Edge triggered.** IT0 / IT1 i.e. (ITx) in TCON are used to decide level triggering or edge triggering. If ITx = 0 then low level interrupt is used to trigger 8051 & if ITx =1 then Falling edge will set IEx flag and interrupt is generated. IT0 & IT1 bits are available in TCON SFR.	**2Marks for Diagram** **2Marks for Explanation**
	b)	**Draw the interfacing diagram of 3*3 keyboard matrix with 8051. Also explain logic to read key.**	**4M**

Ans:



2Marks for Diagram

Keyboard Logic to read keyboard:

1. Port P1 is used as an O/P port for microcontroller 8051 & Port 2 as an I/P port of microcontroller 8051
2. Make all rows of port P1 low so that it gives low voltage when key is pressed.
3. See if any key is pressed by scanning the port P2 by checking all columns for zero condition.
4. If any key is pressed, to identify which key is pressed make one row low at a time.
5. Initiate a counter to hold the count so that each key is counted.
6. Check port P2 for zero condition. If any zero number is there then start column scanning by following step 8.
7. Otherwise make next row low in port P1 and repeat from step 6
8. If any key pressed is found, then content in accumulator is rotated right through the carry until carry bit sets, while doing this increment the count in the counter till carry is found.
9. Move the content in the counter to display in data field or to memory location
10. To repeat the procedures go to step 2.

2Marks for Explanation

c) **List any four assembler directive and explain it.** **4M**

Ans:

Following are Assembler directives

1. **ORG**
2. **EQU**
3. **DB**
4. **DW**
5. **END**

1Marks for Each Assembler Directive

(1) ORG (Originate):

Org xxxx Originate the following code starting at address xxxx.

Example Program

		Address	**Hex**
Org 0400h	becomes	0400	79
Mov r2, #00h		0401	00

The ORG pseudo lets you put code and data anywhere in program memory you wish. Normally the program starts at 0000h using an org 0000h.

(2) EQU (Equate):

Label equxxxx Equate the label name to the number xxxx

Example program

		Address	**Hex**
Org 0000h	becomes	0000	74
Fredequ 12h		0001	12
Mov a, #fred			

EQU turns numbers into names; it makes the program much more readable because the name chosen for the label can have some meaning in the program, whereas the number will not.

(3) <u>DB(Define Byte)</u>

db xx Define a byte: Place the 8 bit number xx next in memory.

Example program

		Address	**Hex**
Org 0100h	becomes	0100	34
db 34h		0101	56
db 56h			

DB xx takes the number xx (from 0 to 255) and converts it to hex in the next memory location. DB permits the programmer to place any hex byte anywhere in memory.

(4) <u>DW (Define word):</u>

dwxxxx Define aword: place the 16bit number xxxx in memory.

Example program

		Address	**Hex**
org 0abcdh	becomes	abcd	12
dw 1234h		abce	34

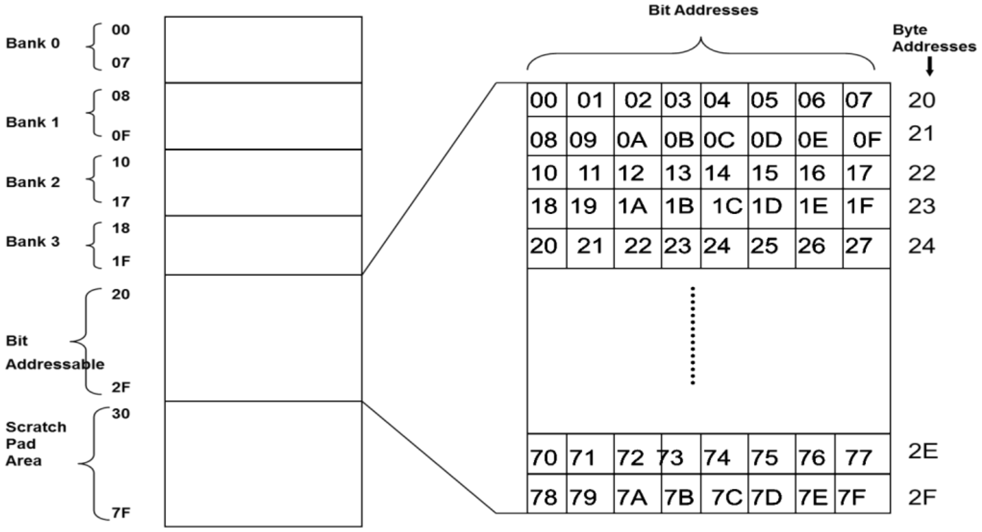
DW is a 16 bit version of db.

(5) <u>End:</u>

The End: Tells the assembler to stop assembling

d) **Draw the structure of internal RAM of 8051.** **4M**

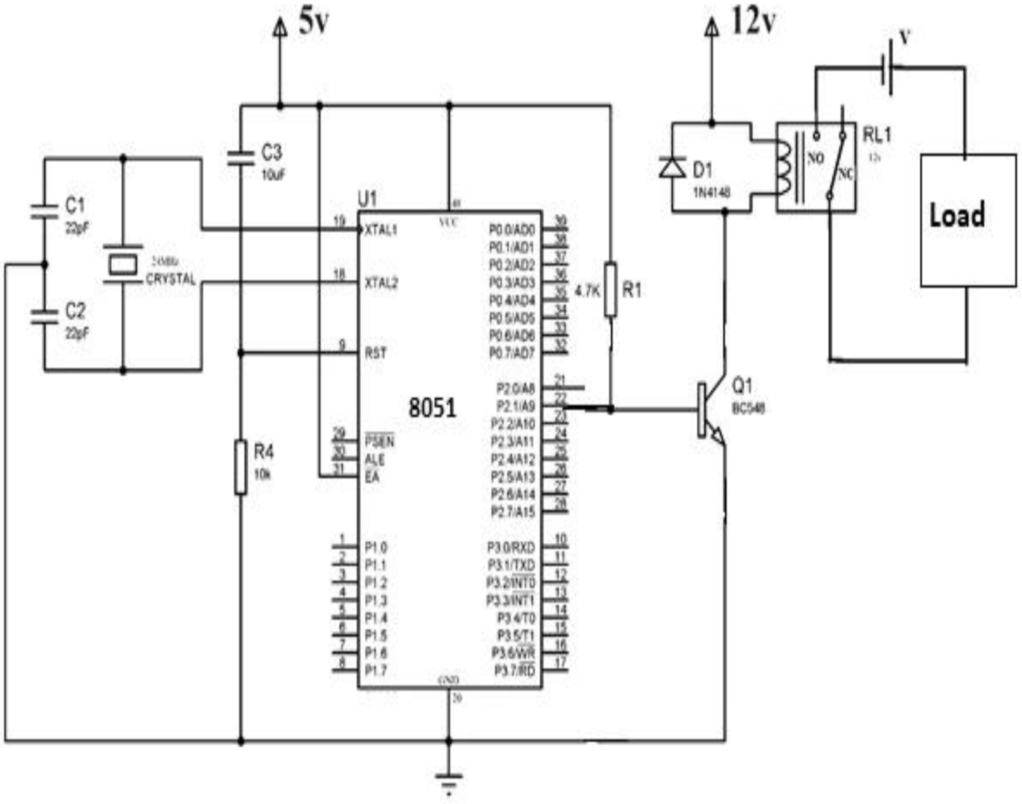
Ans:



Bit Addresses								Byte Addresses
00	01	02	03	04	05	06	07	20
08	09	0A	0B	0C	0D	0E	0F	21
10	11	12	13	14	15	16	17	22
18	19	1A	1B	1C	1D	1E	1F	23
20	21	22	23	24	25	26	27	24
				⋮				
70	71	72	73	74	75	76	77	2E
78	79	7A	7B	7C	7D	7E	7F	2F

4 Marks for Correct Diagram

e) **Draw the interfacing diagram of relay connected at P2.1 with 8051 microcontroller.** **4M**

	Ans :		4Marks for Correct Diagram