USN										

INTERNAL ASSESSMENT TEST – II

Sub:	Microcontroller							Code:	BEC405A
Date:	26/05/25	Duration:	90 mins	Max Marks:	50	Sem:	IV	Branch:	ECE

Answer any 5 full questions

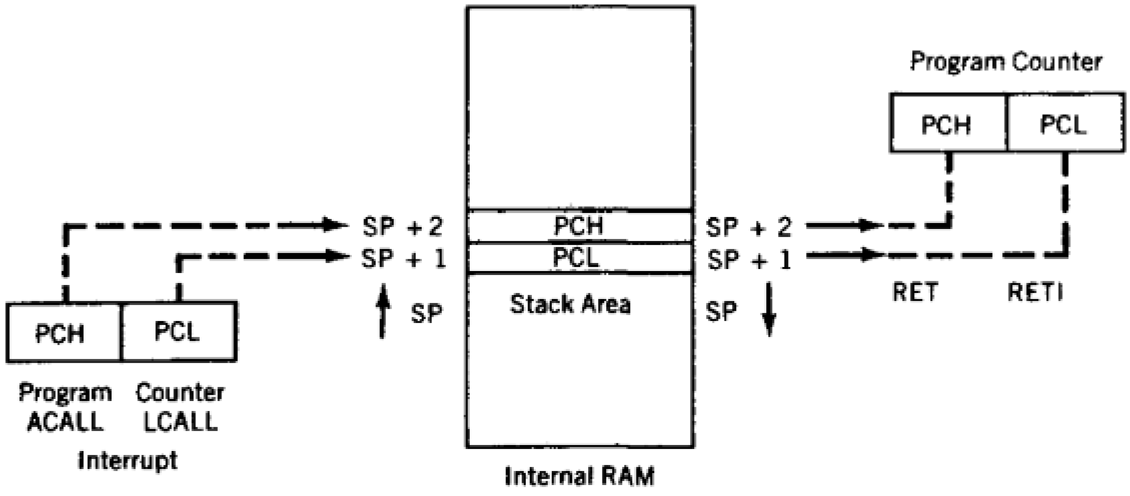
		Marks	**CO**	**RBT**
1	With diagrammatical representation and example explain how stacks play its role in subroutine operations.	[10]	CO3	L2
2	Write an 8051 ALP program to find the factorial of a given number.	[10]	CO3	L3
3	Write 8051 ALP program to interfacing simple switch and LED to I/O ports to switch on/off LED with respect to switch status.	[10]	CO3	L3
4	Explain the bit contents of TCON and TMOD registers.	[10]	CO4	L2

		Marks	**CO**	**RBT**
5	Write an assembly language program to generate a square wave on port pin P1.2 of frequency 5k Hz.	[10]	CO4	L3
6	Write an 8051 C program to transfer the message “YES” serially at 9600 baud rate, 8 bit data, 1 stop bit, do this continuously.	[10]	CO4	L3
7	Show how LCD can be interfaced to the microcontroller and using that display “HELLO” on LCD.	[10]	CO5	L3
8	Write an ALP program to rotate stepper motor in clockwise and counter clock wise direction using the status of the switch connected to the microcontroller. Draw the diagram to show the connection of the switch and stepper motor to 8051 microcontroller.	[10]	CO5	L3

CI　　　　CCI　　　　HoD/ECE

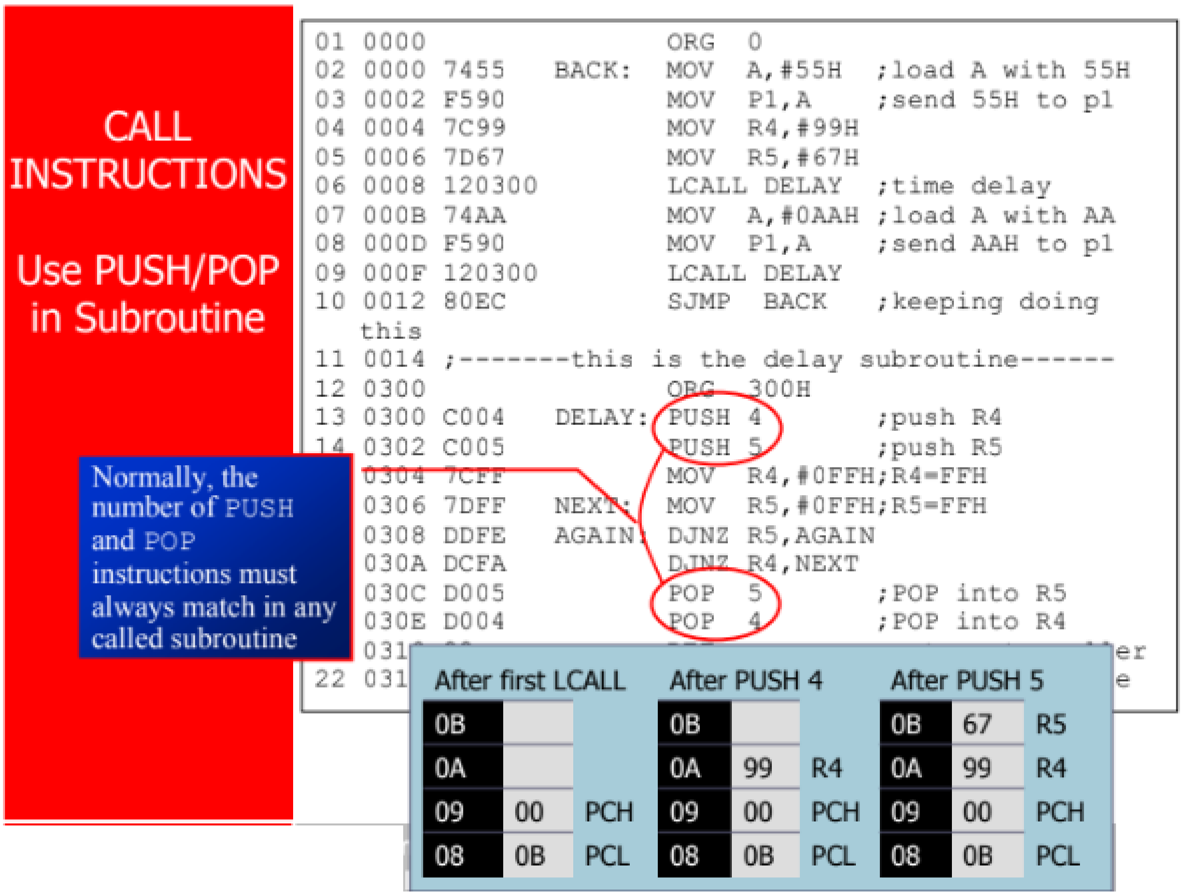
1. With diagrammatical representation and example explain how stacks play its role in subroutine operations.

Storing and Retrieving the Return Address



The following program example use a call to a subroutine.

```
ADDRESS    MNEMONIC        COMMENT
MAIN:      MOV 81h,#30h    ;set the stack pointer to 30h in RAM
           LCALL SUB       ;push address of NOP; PC = #SUB; SP = 32h
           NOP             ;return from SUB to this opcode
           . . .
           . . .
SUB:       MOV A,#45h      ;SUB loads A with 45h and returns
           RET             ;pop return address to PC; SP = 30h
```



CALL INSTRUCTIONS

Use PUSH/POP in Subroutine

```
0000                 ORG   0
0000 7455   BACK:    MOV   A,#55H   ;load A with 55H
0002 F590            MOV   P1,A     ;send 55H to pl
0004 7C99            MOV   R4,#99H
0006 7D67            MOV   R5,#67H
0008 120300          LCALL DELAY    ;time delay
000B 74AA            MOV   A,#0AAH  ;load A with AA
000D F590            MOV   P1,A     ;send AAH to pl
000F 120300          LCALL DELAY
0012 80EC            SJMP  BACK     ;keeping doing
   this
0014 ;-------this is the delay subroutine------
0300                 ORG   300H
0300 C004   DELAY:   PUSH  4        ;push R4
0302 C005            PUSH  5        ;push R5
0304 7CFF            MOV   R4,#0FFH;R4=FFH
0306 7DFF   NEXT:    MOV   R5,#0FFH;R5=FFH
0308 DDFE   AGAIN:   DJNZ  R5,AGAIN
030A DCFA            DJNZ  R4,NEXT
030C D005            POP   5        ;POP into R5
030E D004            POP   4        ;POP into R4
031                                              er
031                                              e
```

Normally, the number of PUSH and POP instructions must always match in any called subroutine

After first LCALL

0B		
0A		
09	00	PCH
08	0B	PCL

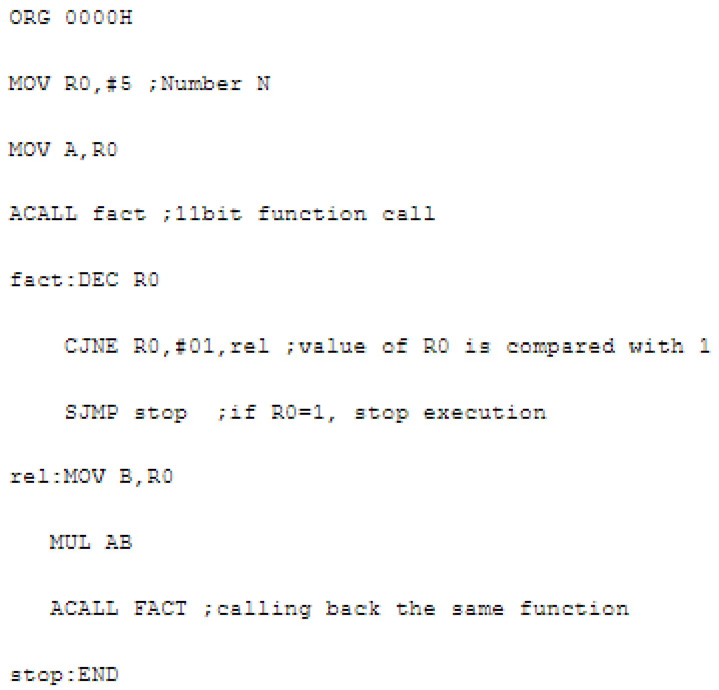
After PUSH 4

0B		
0A	99	R4
09	00	PCH
08	0B	PCL

After PUSH 5

0B	67	R5
0A	99	R4
09	00	PCH
08	0B	PCL

2. Write an 8051 ALP program to find the factorial of a given number.



```
ORG 0000H

MOV R0,#5 ;Number N

MOV A,R0

ACALL fact ;11bit function call

fact:DEC R0

    CJNE R0,#01,rel ;value of R0 is compared with 1

    SJMP stop  ;if R0=1, stop execution

rel:MOV B,R0

   MUL AB

   ACALL FACT ;calling back the same function

stop:END
```

3. Write 8051 ALP program to interfacing simple switch and LED to I/O ports to switch on/off LED with respect to switch status.

```
ORG 0000H        ; Origin, start of the program

; Initialize Ports

MOV P1, #00H     ; Clear Port 1 (LEDs off)

MOV P2, #0FFH    ; Set Port 2 as input (Switches)

MAIN:

    MOV A, P2    ; Read the status of the switch

    JNB ACC.0, LED_ON   ; If switch (P2.0) is pressed, jump to LED_ON

    SJMP MAIN   ; Otherwise, keep checking the switch

LED_ON:

    SETB P1.0    ; Turn on LED connected to P1.0

    SJMP MAIN   ; Go back to check the switch status
```

Explanation:

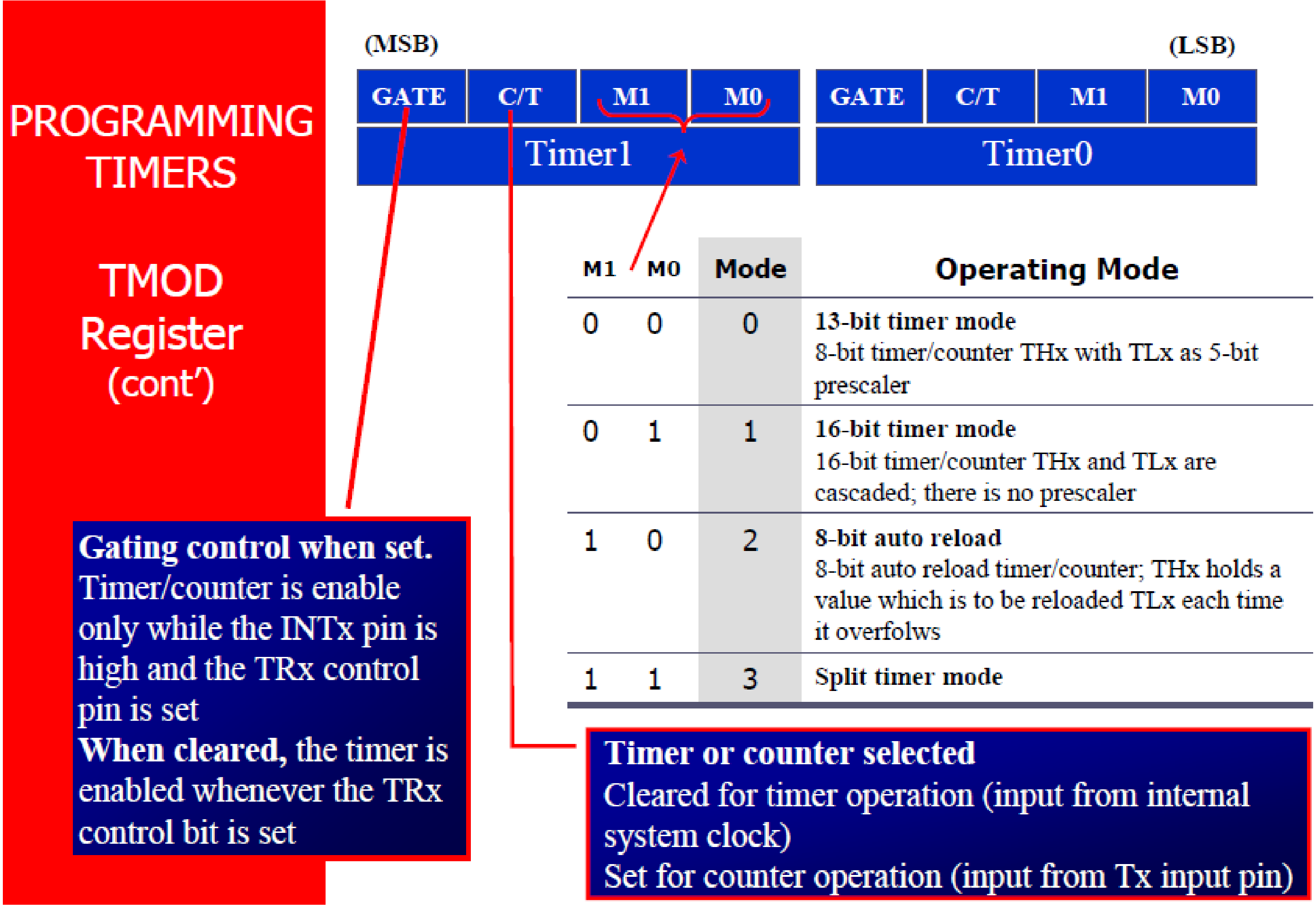
- **Initialization**:
  - `MOV P1, #00H`: Clears Port 1, turning off all LEDs.
  - `MOV P2, #0FFH`: Sets Port 2 as input, assuming switches are connected here.

  Main Loop:
  - `MOV A, P2`: Reads the status of the switches into the accumulator.
  - `JNB ACC.0, LED_ON`: Checks if the switch connected to P2.0 is pressed (logic low). If pressed, it jumps to the `LED_ON` label.
  - `SJMP MAIN`: If the switch is not pressed, it keeps looping back to check the switch status.

  LED_ON:
  - `SETB P1.0`: Turns on the LED connected to P1.0.
  - `SJMP MAIN`: After turning on the LED, it goes back to the main loop to keep checking the switch status.

---

4. Explain the bit contents of TCON and TMOD registers.



PROGRAMMING TIMERS

TMOD Register (cont')

(MSB) (LSB)

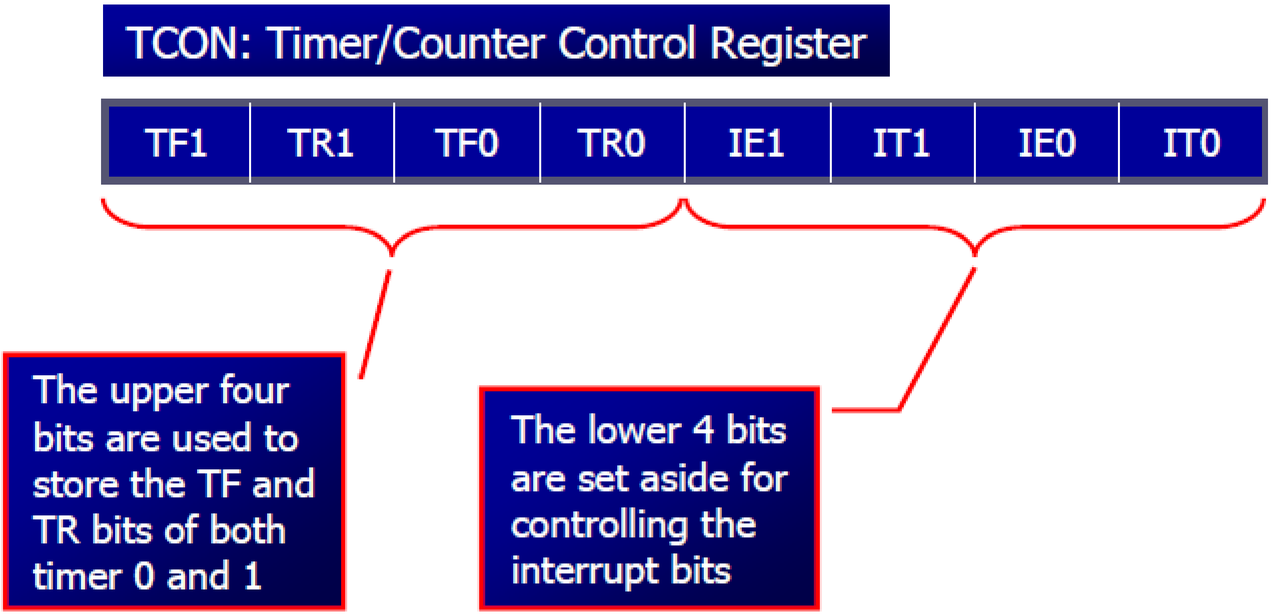
GATE	C/T	M1	M0	GATE	C/T	M1	M0
Timer1				Timer0			

M1	M0	Mode	Operating Mode
0	0	0	**13-bit timer mode** 8-bit timer/counter THx with TLx as 5-bit prescaler
0	1	1	**16-bit timer mode** 16-bit timer/counter THx and TLx are cascaded; there is no prescaler
1	0	2	**8-bit auto reload** 8-bit auto reload timer/counter; THx holds a value which is to be reloaded TLx each time it overfolws
1	1	3	**Split timer mode**

Gating control when set. Timer/counter is enable only while the INTx pin is high and the TRx control pin is set
When cleared, the timer is enabled whenever the TRx control bit is set

Timer or counter selected
Cleared for timer operation (input from internal system clock)
Set for counter operation (input from Tx input pin)

- TCON (timer control) register is an 8-bit register



TCON: Timer/Counter Control Register

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

The upper four bits are used to store the TF and TR bits of both timer 0 and 1

The lower 4 bits are set aside for controlling the interrupt bits

TFx - (timer flag bit , control flag)

- **TF0 flag bit for timer/counter 0**
- **TF1 flag bit for timer/counter 1.**

TF=1 When TH-TL roll over to 0000 from FFFFHo 1. If we enable interrupt, TF=1 will trigger ISR.

IT0 and IT1: determines whether interrupt is edge triggered or level triggered.

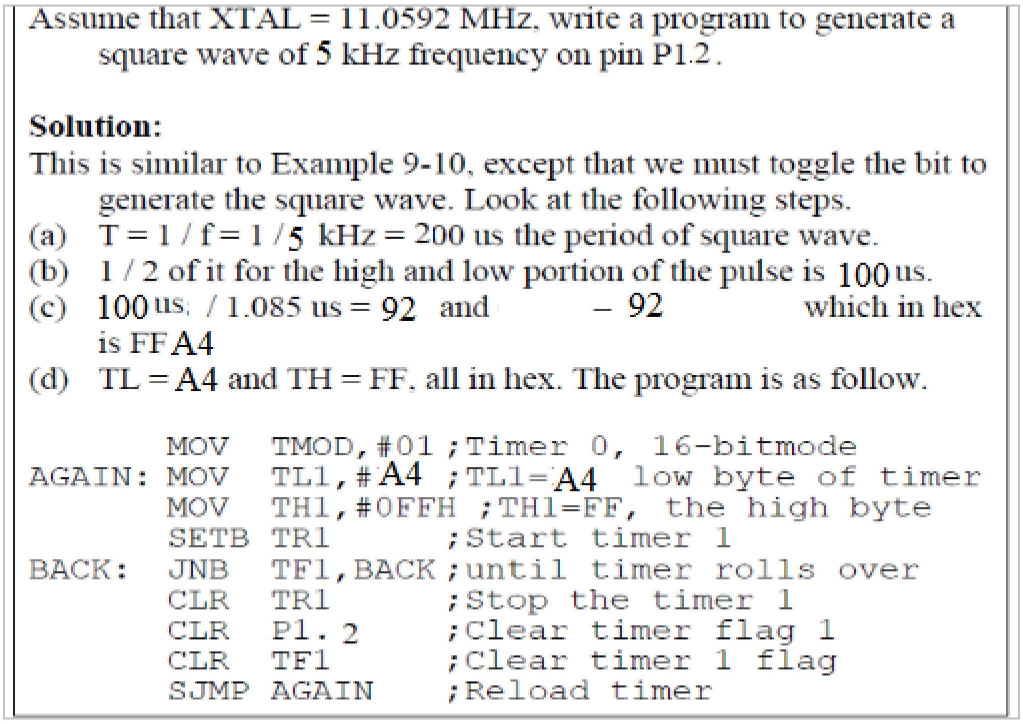
When = 1 ; it is edge triggered

= 0 ; level triggered

IE0 and IE1: when =1; interrupt is executed

when = 0 interrupt is pending

5. Write an assembly language program to generate a square wave on port pin P1.2 of frequency 5k Hz.



Assume that XTAL = 11.0592 MHz, write a program to generate a square wave of 5 kHz frequency on pin P1.2.

Solution:

This is similar to Example 9-10, except that we must toggle the bit to generate the square wave. Look at the following steps.

(a) T = 1 / f = 1 / 5 kHz = 200 us the period of square wave.

(b) 1 / 2 of it for the high and low portion of the pulse is 100 us.

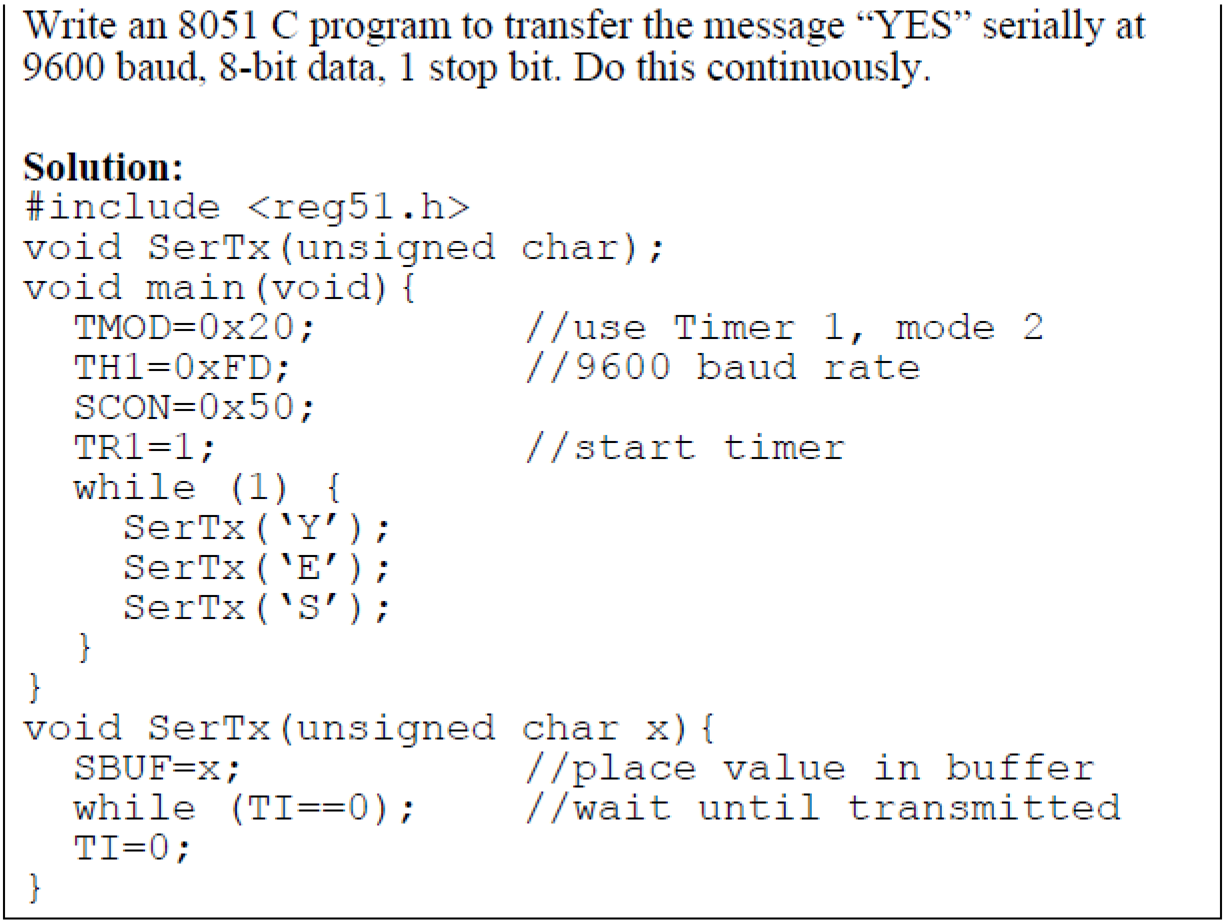
(c) 100 us / 1.085 us = 92 and – 92 which in hex is FF A4

(d) TL = A4 and TH = FF, all in hex. The program is as follow.

```
        MOV  TMOD,#01 ;Timer 0, 16-bitmode
AGAIN: MOV  TL1,#A4  ;TL1=A4  low byte of timer
        MOV  TH1,#0FFH ;TH1=FF, the high byte
        SETB TR1       ;Start timer 1
BACK:   JNB  TF1,BACK ;until timer rolls over
        CLR  TR1       ;Stop the timer 1
        CLR  P1.2      ;Clear timer flag 1
        CLR  TF1       ;Clear timer 1 flag
        SJMP AGAIN     ;Reload timer
```

---

6. Write an 8051 C program to transfer the message "YES" serially at 9600 baud rate, 8 bit data, 1 stop bit, do this continuously.



Write an 8051 C program to transfer the message "YES" serially at 9600 baud, 8-bit data, 1 stop bit. Do this continuously.

Solution:

```
#include <reg51.h>
void SerTx(unsigned char);
void main(void){
  TMOD=0x20;          //use Timer 1, mode 2
  TH1=0xFD;           //9600 baud rate
  SCON=0x50;
  TR1=1;              //start timer
  while (1) {
    SerTx('Y');
    SerTx('E');
    SerTx('S');
  }
}
void SerTx(unsigned char x){
  SBUF=x;             //place value in buffer
  while (TI==0);      //wait until transmitted
  TI=0;
}
```

7. Show how LCD can be interfaced to the microcontroller and using that display "HELLO" on LCD.

```
MOV A,#38H ;INIT. LCD 2 LINES, 5X7 MATRIX
ACALL COMNWRT ;call command subroutine
ACALL DELAY ;give LCD some time
MOV A,#0EH ;display on, cursor on
ACALL COMNWRT ;call command subroutine
ACALL DELAY ;give LCD some time
MOV A,#01 ;clear LCD
ACALL COMNWRT ;call command subroutine
ACALL DELAY ;give LCD some time
MOV A,#06H ;shift cursor right
ACALL COMNWRT ;call command subroutine
ACALL DELAY ;give LCD some time
MOV A,#86H ;cursor at line 1, pos. 6
ACALL COMNWRT ;call command subroutine
ACALL DELAY ;give LCD some time
MOV A, #H ;display letter H
ACALL DATAWRT ;call display subroutine
ACALL DELAY ;give LCD some time
MOV A, #E ;display letter E
ACALL DATAWRT ;call display subroutine
ACALL DELAY ;give LCD some time
MOV A, #L ;display letter L
ACALL DATAWRT ;call display subroutine
ACALL DELAY ;give LCD some time
MOV A, #L ;display letter L
ACALL DATAWRT ;call display subroutine
ACALL DELAY ;give LCD some time
MOV A, #O ;display letter O
ACALL DATAWRT ;call display subroutine
ACALL DELAY ;give LCD some time
AGAIN: SJMP AGAIN ;stay here
COMNWRT: ;send command to LCD
MOV P1,A ;copy reg A to port 1
CLR P2.0 ;RS=0 for command
CLR P2.1 ;R/W=0 for write
```

```
SETB P2.2 ;E=1 for high pulse
ACALL DELAY ;give LCD some time
CLR P2.2 ;E=0 for H-to-L pulse
RET
DATAWRT: ;write data to LCD
MOV P1,A ;copy reg A to port 1
SETB P2.0 ;RS=1 for data
CLR P2.1 ;R/W=0 for write
SETB P2.2 ;E=1 for high pulse
ACALL DELAY ;give LCD some time
CLR P2.2 ;E=0 for H-to-L pulse
RET
DELAY:
MOV R3,#50 ;50 or higher for fast CPUs
HERE2: MOV R4,#255 ;R4 = 255
HERE: DJNZ R4,HERE ;stay until R4 becomes 0
DJNZ R3,HERE2
RET
END
```

8.Write an ALP program to rotate stepper motor in clockwise and counter clock wise direction using the status of the switch connected to the microcontroller. Draw the diagram to show the connection of the switch and stepper motor to 8051 microcontroller.

```
ORG 0000H
SETB P2.7
MOV A, #66H
MOV P1,A
TURN: JNB P2.7, CW
RL A
ACALL DELAY
MOV P1,A
SJMP TURN
CW: RR A
ACALL DELAY
MOV P1,A
SJMP TURN
DELAY: MOV R1,#100
UP1: MOV R2,#50
UP: DJNZ R2,UP
```

```
DJNZ R1,UP1
RET
```

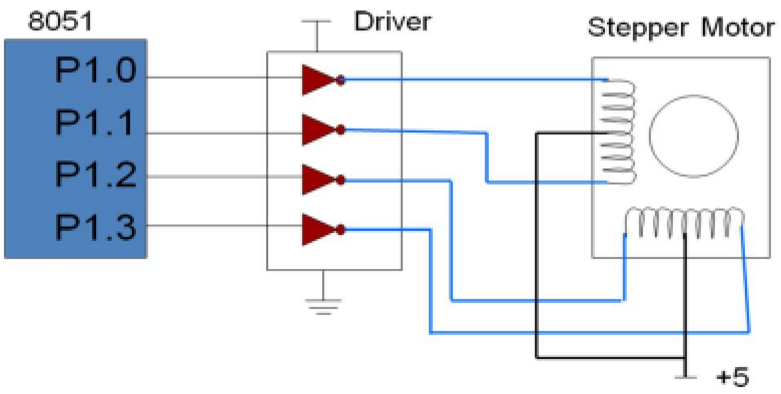


Figure : 8051 interfaces to stepper motor