USN | | | | | | | | | | |

INTERNAL ASSESSMENT TEST – III

Sub:	Microcontroller							Code:	BEC405A
Date:	02/08/24	Duration:	90 mins	Max Marks:	50	Sem:	IV	Branch:	ECE

Answer any 5 full questions

		Marks	CO	RBT
1	Give notes on Interrupt and explain IE register.	[10]	CO4	L2
2	Assume that the INT1 pin is connected to a switch that is normally high. Whenever it goes low it should turn ON an LED. The LED is connected to P1.3 and is normally off. When it is turned ON it should stay ON for a fraction of a second. As long as the switched is pressed low, the LED should stay ON.	[10]	CO4	L3
3	A switch is connected to pin P2.7. Write a C program to monitor the status of the SW. If SW = 0, stepper motor moves clockwise and if SW = 1, stepper motor moves anticlockwise.	[10]	CO5	L3
4	A switch is connected to pin P2.7. Write a C to monitor the status of the SW. If SW = 0, DC motor moves 50% duty cycle pulse and if SW = 1, DC motor moves with 25% duty cycle pulse.	[10]	CO5	L3
5	Write an 8051 C program to generate a sine waveform.	[10]	CO5	L3
6	Write an 8051 C program to send letters ‘P’, ‘I’ and ‘C’ to the LCD using the busy flag method.	[10]	CO3	L3

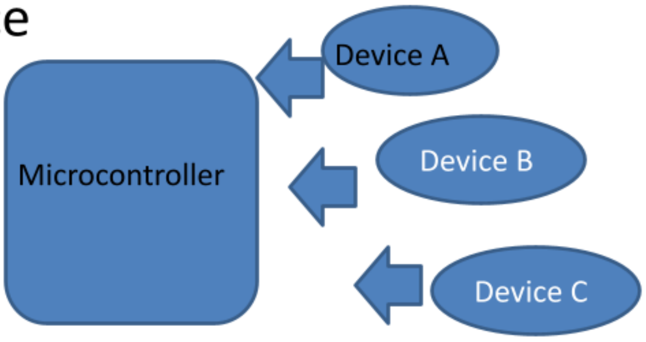
CI CCI HoD/ECE

Solution- INTERNAL ASSESSMENT TEST – III

1.
2. Give notes on Interrupt and explain IE register.

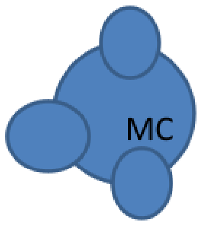
Interrupts

- In Interrupts, Whenever any device needs its service, the device notifies the microcontroller by sending it an interrupt signal
- Upon receiving an interrupt signal, the microcontroller interrupts whatever it is doing and serves the device



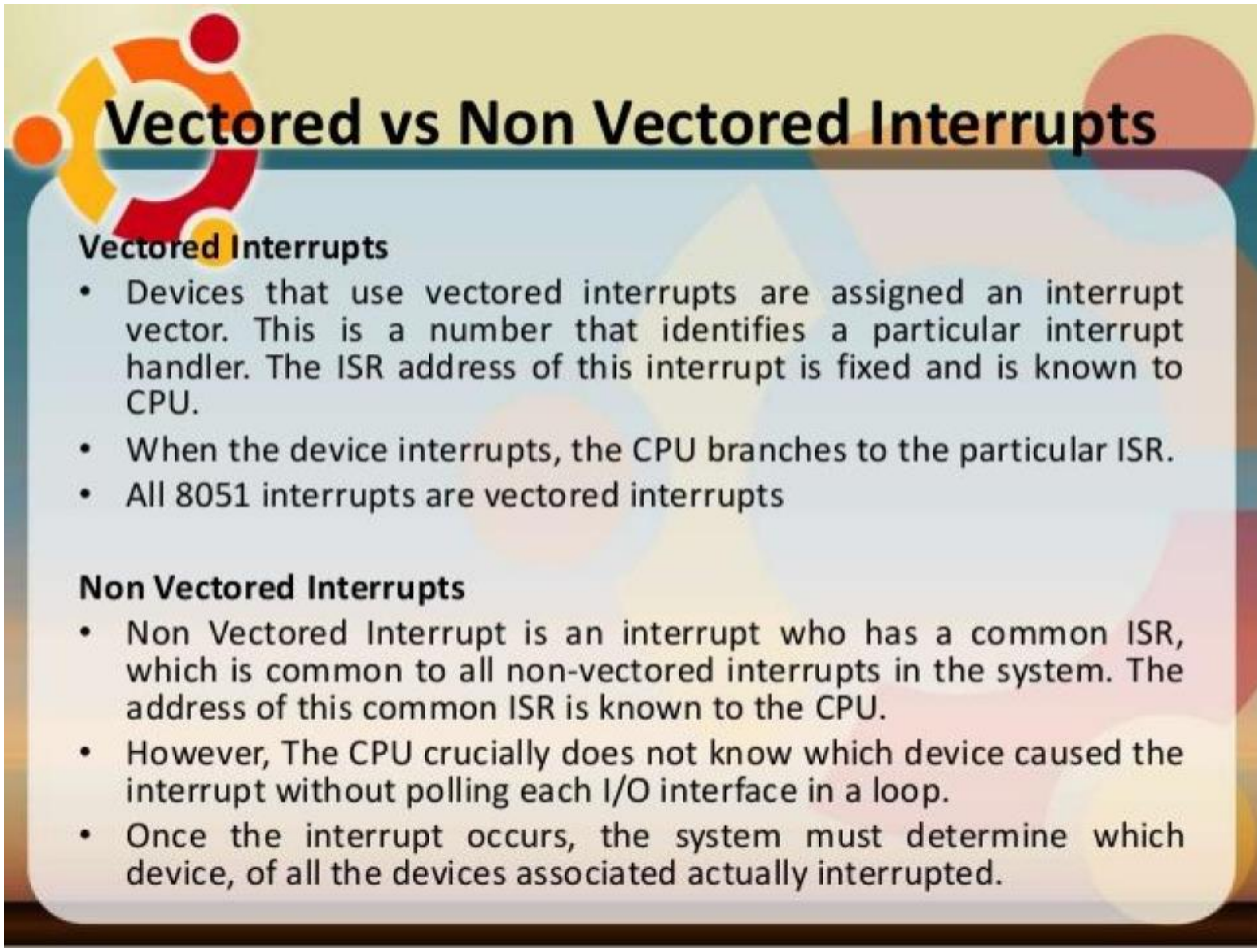
Polling

- The microcontroller continuously monitors the status of all devices in Round-robin manner.
- When the conditions for a given device are met, it performs the service.
- After that, it moves on to monitor the next device until every one is serviced



Interrupt

- An interrupt is an external or internal event that interrupts the microcontroller to inform it that a device needs its service
- The program which is associated with the interrupt is called the Interrupt Service Routine (ISR) or interrupt handler.



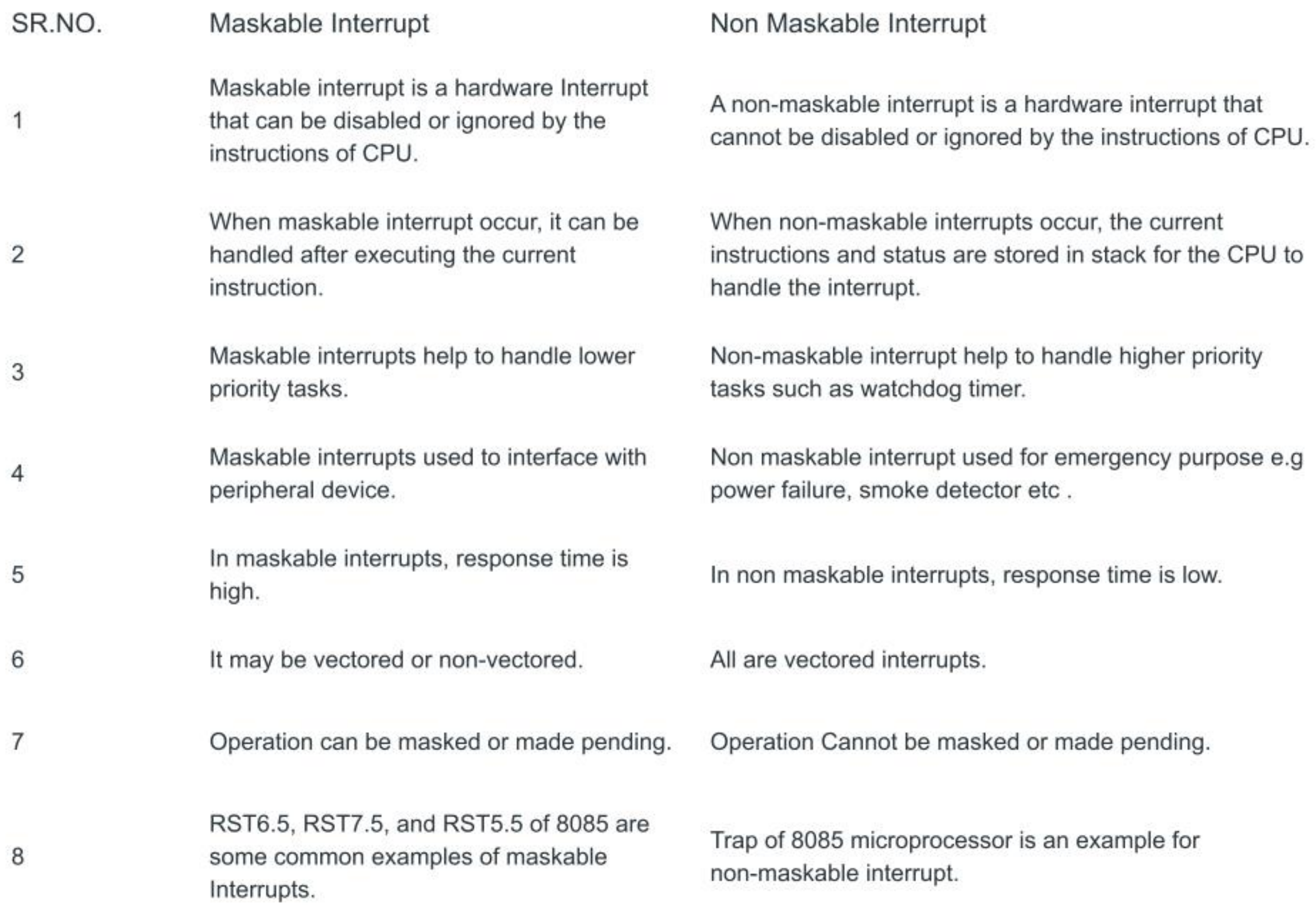
Vectored vs Non Vectored Interrupts

Vectored Interrupts

- Devices that use vectored interrupts are assigned an interrupt vector. This is a number that identifies a particular interrupt handler. The ISR address of this interrupt is fixed and is known to CPU.
- When the device interrupts, the CPU branches to the particular ISR.
- All 8051 interrupts are vectored interrupts

Non Vectored Interrupts

- Non Vectored Interrupt is an interrupt who has a common ISR, which is common to all non-vectored interrupts in the system. The address of this common ISR is known to the CPU.
- However, The CPU crucially does not know which device caused the interrupt without polling each I/O interface in a loop.
- Once the interrupt occurs, the system must determine which device, of all the devices associated actually interrupted.



SR.NO.	Maskable Interrupt	Non Maskable Interrupt
1	Maskable interrupt is a hardware Interrupt that can be disabled or ignored by the instructions of CPU.	A non-maskable interrupt is a hardware interrupt that cannot be disabled or ignored by the instructions of CPU.
2	When maskable interrupt occur, it can be handled after executing the current instruction.	When non-maskable interrupts occur, the current instructions and status are stored in stack for the CPU to handle the interrupt.
3	Maskable interrupts help to handle lower priority tasks.	Non-maskable interrupt help to handle higher priority tasks such as watchdog timer.
4	Maskable interrupts used to interface with peripheral device.	Non maskable interrupt used for emergency purpose e.g power failure, smoke detector etc .
5	In maskable interrupts, response time is high.	In non maskable interrupts, response time is low.
6	It may be vectored or non-vectored.	All are vectored interrupts.
7	Operation can be masked or made pending.	Operation Cannot be masked or made pending.
8	RST6.5, RST7.5, and RST5.5 of 8085 are some common examples of maskable Interrupts.	Trap of 8085 microprocessor is an example for non-maskable interrupt.

Interrupt Service Routine

- For every interrupt, there must be an interrupt service routine (ISR), or interrupt handler.
- When an interrupt is invoked, the microcontroller runs the interrupt service routine.
- For every interrupt, there is a fixed location in memory that holds the address of its ISR .
- The group of memory locations set aside to hold the addresses of ISRs is called interrupt vector table

Steps in executing an interrupt Upon activation of an interrupt

The microcontroller goes through the following steps

1. It finishes the instruction it is executing and saves the address of the next instruction (PC) on the stack
2. It also saves the current status of all the interrupts internally (i.e: not on the stack)
3. It jumps to a fixed location in memory, called the interrupt vector table, that holds the address of the ISR

Steps in executing an interrupt

4. The microcontroller gets the address of the ISR from the interrupt vector table and jumps to it
5. It starts to execute the interrupt service subroutine until it reaches the last instruction of the subroutine which is RETI (return from interrupt)
6. Upon executing the RETI instruction, the microcontroller returns to the place where it was interrupted

- ❖ First, it gets the program counter (PC) address from the stack by popping the top two bytes of the stack into the PC
- ❖ Then it starts to execute from that address

Interrupts in 8051 There are SIX interrupts in 8051

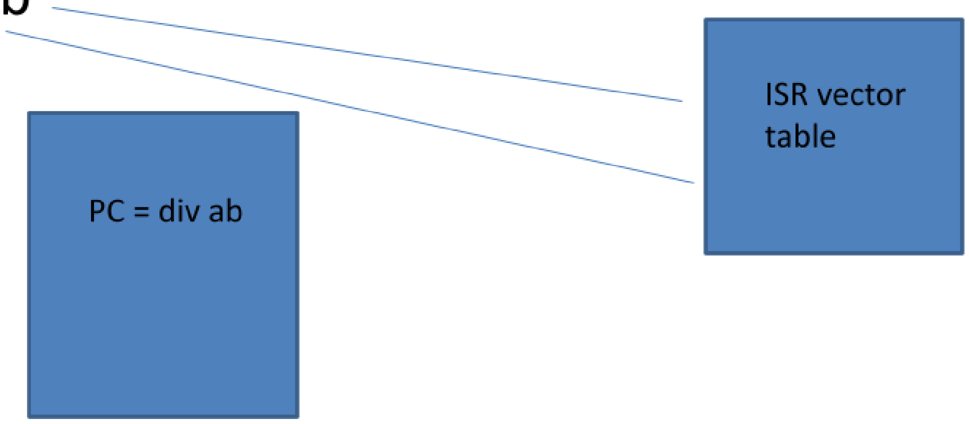
- Reset – power-up reset
- Two interrupts are set aside for the timers:

one for timer 0 and one for timer 1

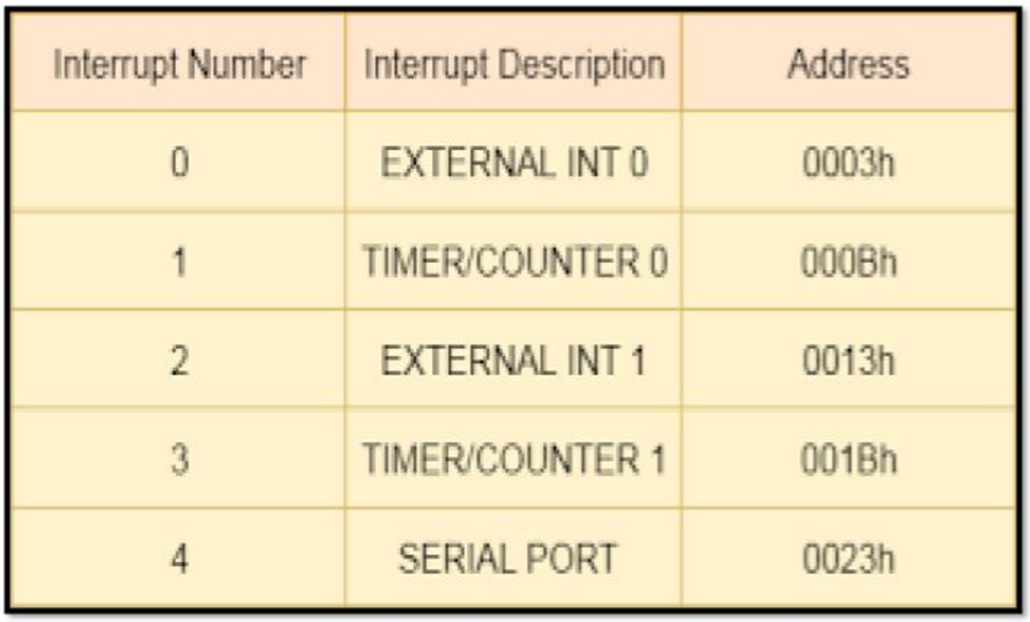
- Two interrupts are set aside for hardware external interrupts

  P3.2 and P3.3 are for the external hardware interrupts INT0 (or EX1), and INT1 (or EX2)
- Serial communication has a single interrupt that belongs to both receive and transfer

- Mov a,b
- div ab



Interrupt Vector Table of 8051



Interrupt Number	Interrupt Description	Address
0	EXTERNAL INT 0	0003h
1	TIMER/COUNTER 0	000Bh
2	EXTERNAL INT 1	0013h
3	TIMER/COUNTER 1	001Bh
4	SERIAL PORT	0023h

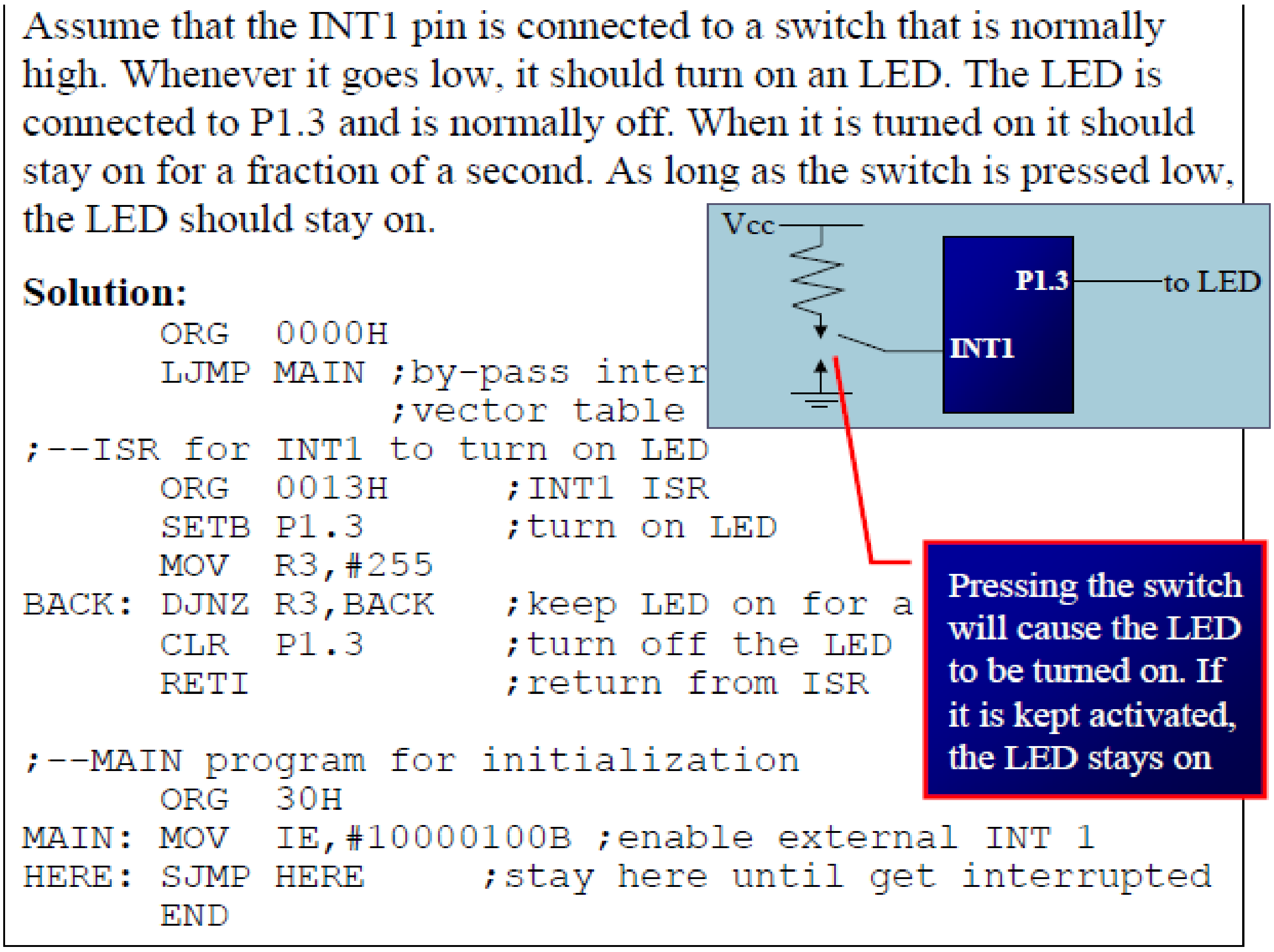
Enabling and Disabling of Interrupts

- Upon reset, all interrupts are disabled (masked), meaning that none will be responded to the microcontroller, even if they are activated.
- The interrupts must be enabled by software in order for the microcontroller to respond to them.
- A register called IE (interrupt enable) that is responsible for enabling (unmasking) and disabling (masking) the interrupts

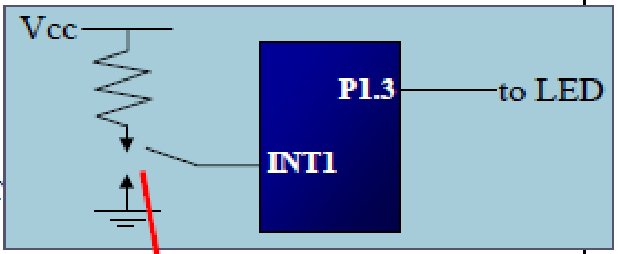
IE Register (Interrupt Enable Register)

- Bit D7 of the IE register (EA) must be set to high to allow the rest of register to take effect.
- The value of EA , If EA = 1, interrupts are enabled and will be responded to if their corresponding bits in IE are high
- If EA = 0, no interrupt will be responded to, even if the associated bit in the IE register is high

2. Assume that the INT1 pin is connected to a switch that is normally high. Whenever it goes low it should turn ON an LED. The LED is connected to P1.3 and is normally off. When it is turned ON it should stay ON for a fraction of a second. As long as the switch is pressed low, the LED should stay ON.



Assume that the INT1 pin is connected to a switch that is normally high. Whenever it goes low, it should turn on an LED. The LED is connected to P1.3 and is normally off. When it is turned on it should stay on for a fraction of a second. As long as the switch is pressed low, the LED should stay on.



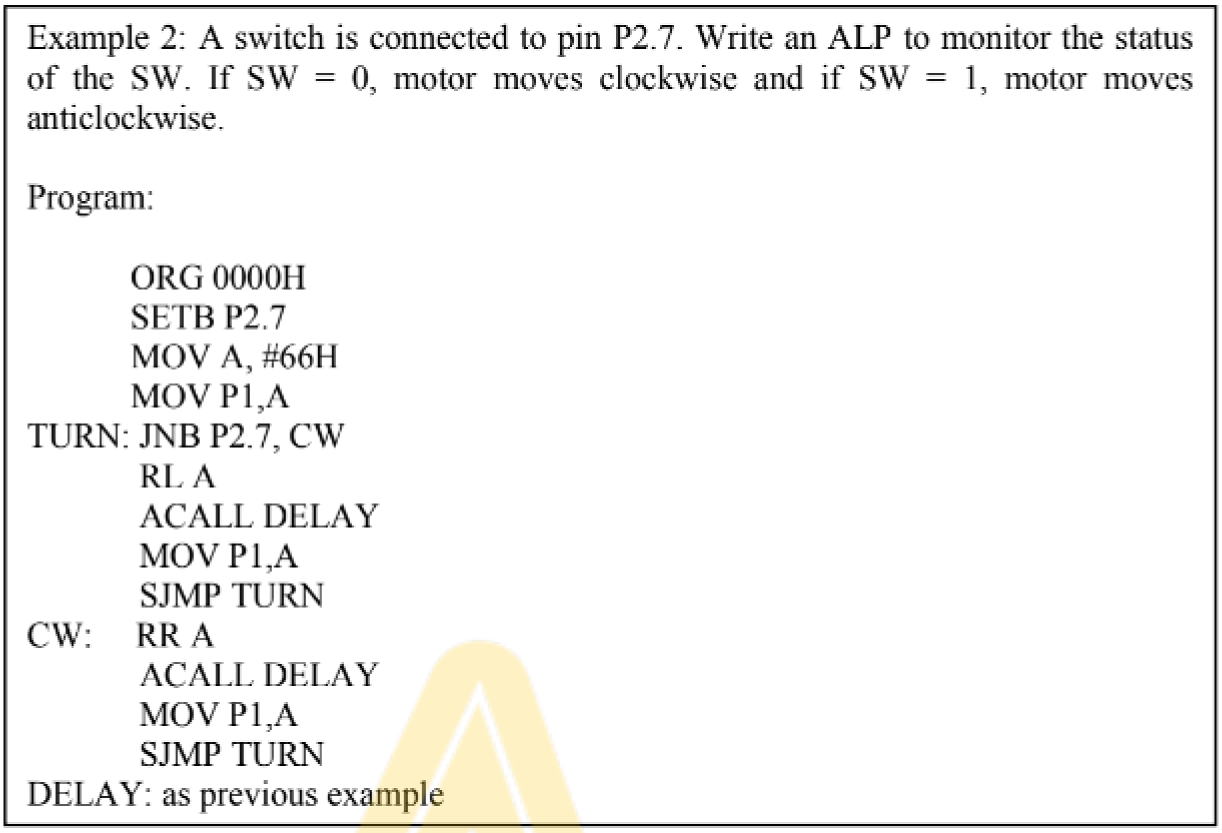
Solution:

```
       ORG   0000H
       LJMP MAIN ;by-pass inter
                 ;vector table
;--ISR for INT1 to turn on LED
       ORG   0013H      ;INT1 ISR
       SETB P1.3        ;turn on LED
       MOV   R3,#255
BACK:  DJNZ R3,BACK     ;keep LED on for a
       CLR   P1.3       ;turn off the LED
       RETI             ;return from ISR

;--MAIN program for initialization
       ORG   30H
MAIN:  MOV   IE,#10000100B ;enable external INT 1
HERE:  SJMP HERE       ;stay here until get interrupted
       END
```

Pressing the switch will cause the LED to be turned on. If it is kept activated, the LED stays on

3. A switch is connected to pin P2.7. Write a C program to monitor the status of the SW. If SW = 0, stepper motor moves clockwise and if SW = 1, stepper motor moves anticlockwise.

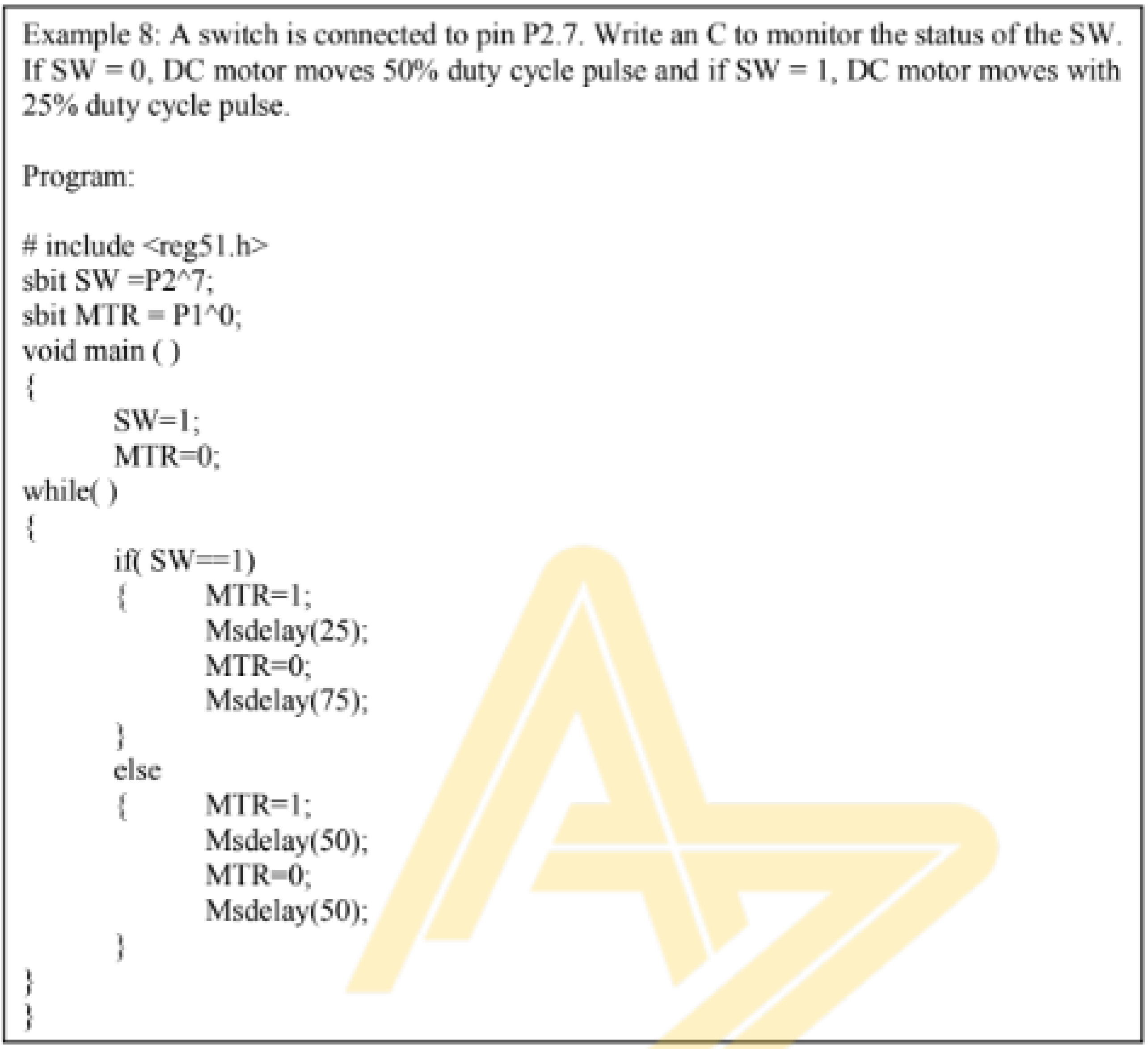


Example 2: A switch is connected to pin P2.7. Write an ALP to monitor the status of the SW. If SW = 0, motor moves clockwise and if SW = 1, motor moves anticlockwise.

Program:

```
            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: as previous example
```

4. **A switch is connected to pin P2.7. Write a C to monitor the status of the SW. If SW = 0, DC motor moves 50% duty cycle pulse and if SW = 1, DC motor moves with 25% duty cycle pulse.**

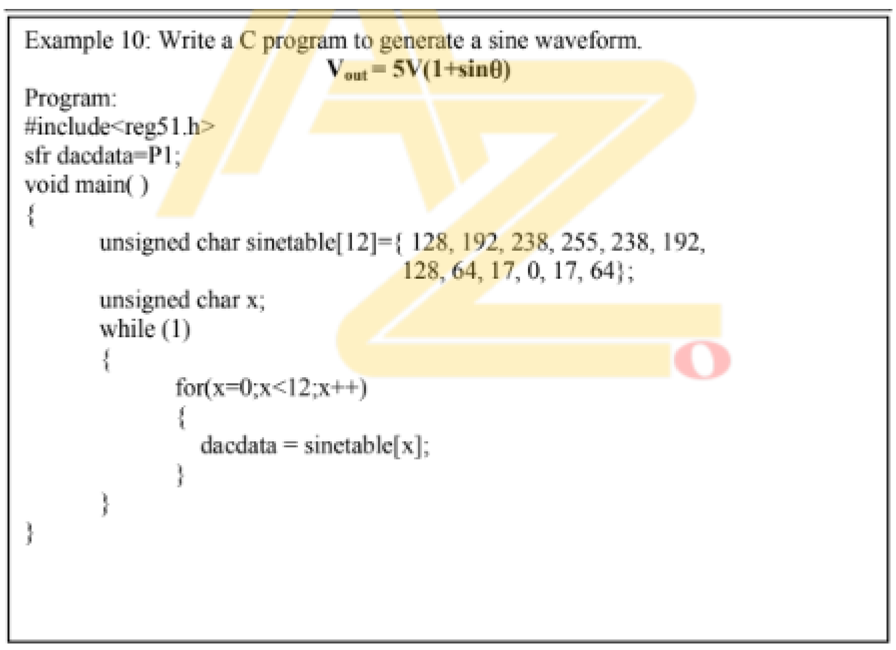


Example 8: A switch is connected to pin P2.7. Write an C to monitor the status of the SW. If SW = 0, DC motor moves 50% duty cycle pulse and if SW = 1, DC motor moves with 25% duty cycle pulse.

Program:

```
# include <reg51.h>
sbit SW =P2^7;
sbit MTR = P1^0;
void main ( )
{
        SW=1;
        MTR=0;
while( )
{
        if( SW==1)
        {       MTR=1;
                Msdelay(25);
                MTR=0;
                Msdelay(75);
        }
        else
        {       MTR=1;
                Msdelay(50);
                MTR=0;
                Msdelay(50);
        }
}
}
```

5. Write an 8051 C program to generate a sine waveform.



Example 10: Write a C program to generate a sine waveform.

$V_{out} = 5V(1+\sin\theta)$

```
Program:
#include<reg51.h>
sfr dacdata=P1;
void main( )
{
        unsigned char sinetable[12]={ 128, 192, 238, 255, 238, 192,
                                      128, 64, 17, 0, 17, 64};
        unsigned char x;
        while (1)
        {
                for(x=0;x<12;x++)
                {
                  dacdata = sinetable[x];
                }
        }
}
```

6. Write an 8051 C program to send letters 'P', 'I' and 'C' to the LCD using the busy flag method.

Example 13: Write an 8051 C program to send letters ‘P’, ‘I’, and ‘C’ to the LCD using the busy flag method.

Solution:

```
#include <reg51.h>

sfr ldata = 0x90;                       //P1=LCD data pins

sbit rs = P2^0;

sbit rw = P2^1;

sbit en = P2^2;

sbit busy = P1^7;

void main(){

        lcdcmd(0x38);

        lcdcmd(0x0E);

        lcdcmd(0x01);

        lcdcmd(0x06);

        lcdcmd(0x86);                   //line 1, position 6

        lcddata(‘P’);
        lcddata(‘I’);

        lcddata(‘C’);

}

void lcdcmd(unsigned char value){

        lcdready();                     //check the LCD busy flag

        ldata = value;                  //put the value on the pins
```