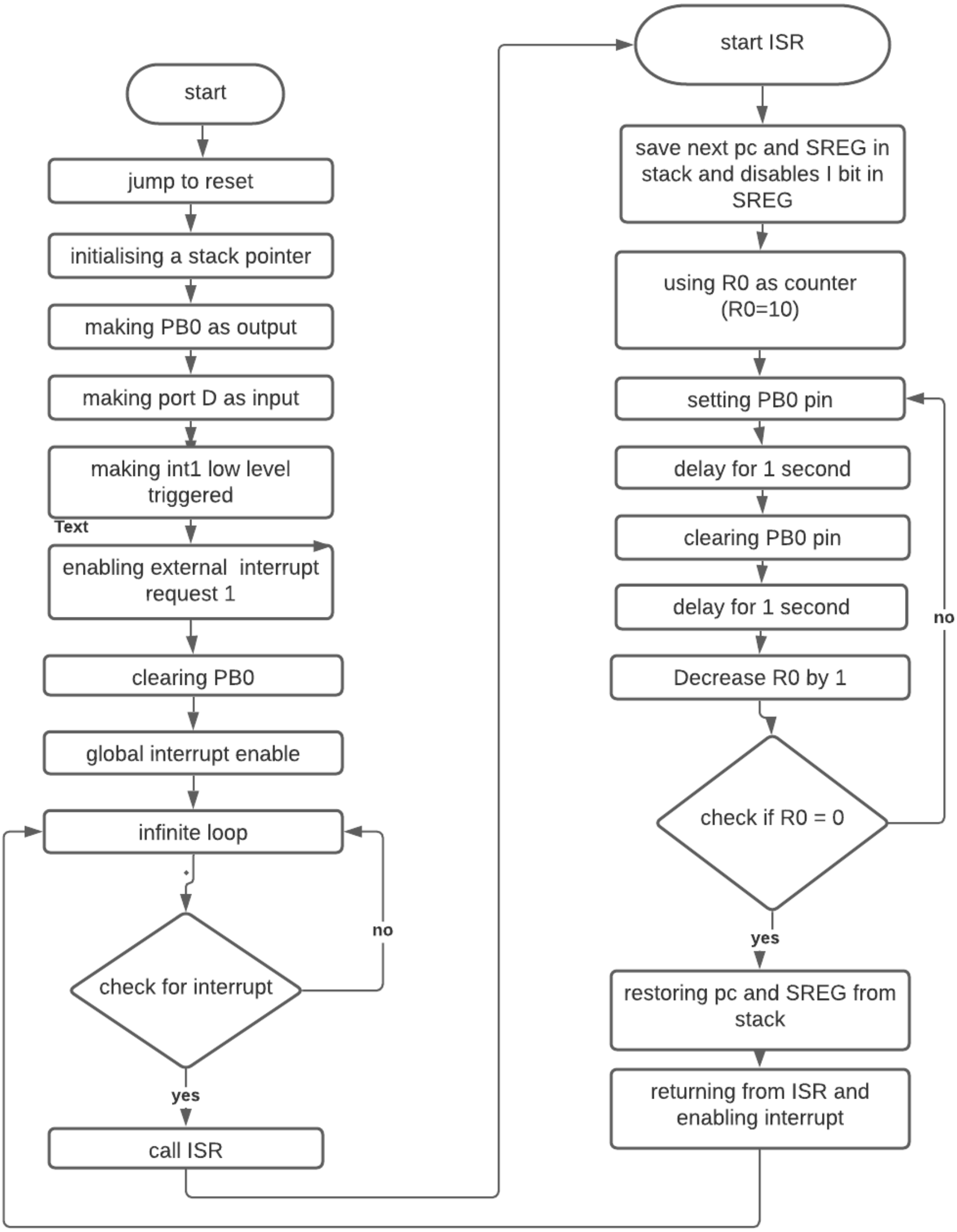
NAME: KEERTHANA RACHURI

ROLL-NO: EE20B102

INTERRUPTS and TIMERS in Atmega

Questions in handouts and their corresponding answers:

1. **Fill in the blanks in the assembly code:**
   (a) **Flowchart:**



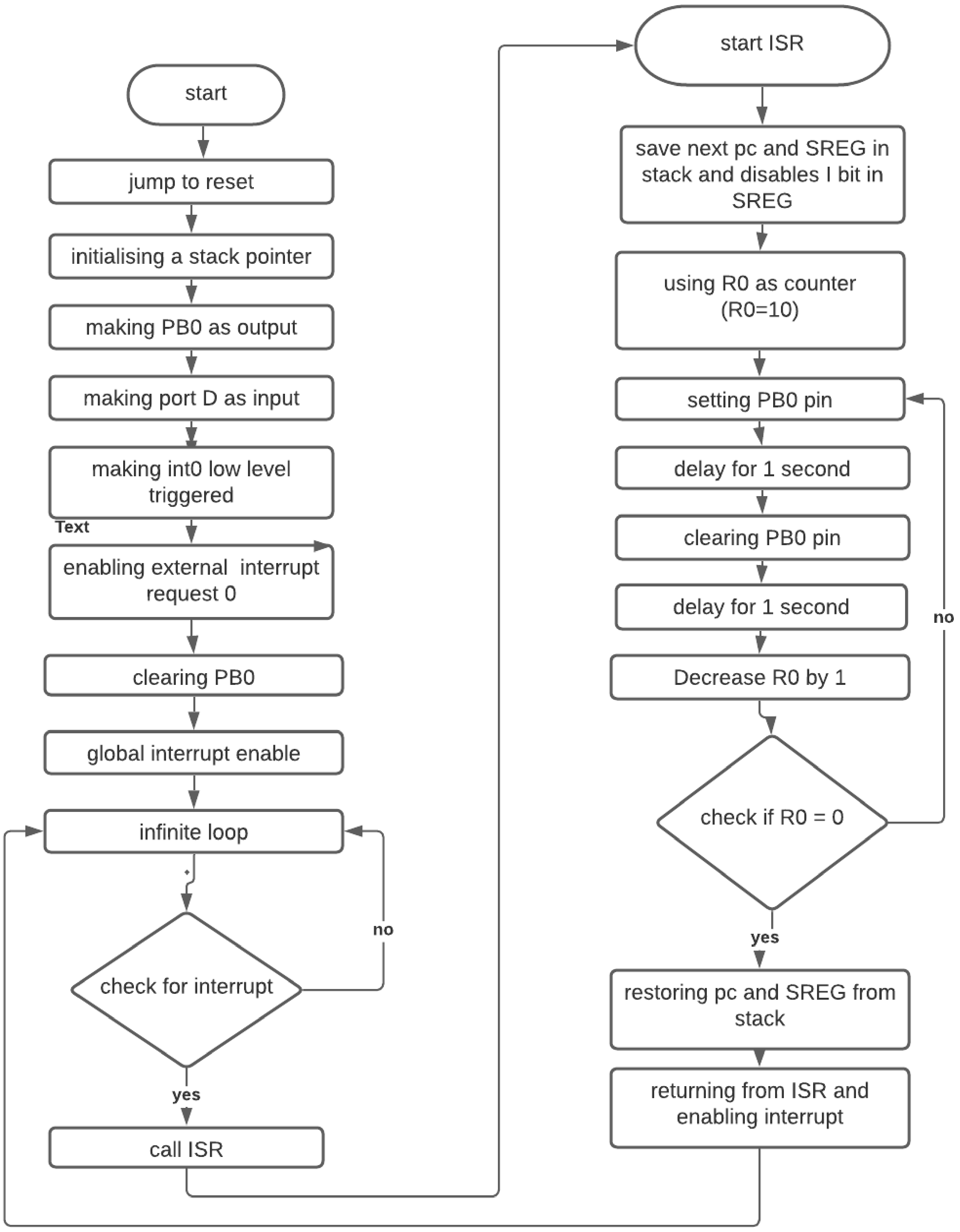
(b) Code:

```
.org 0x0000;location for reset
rjmp reset

.org 0x0002;location for external interrupt Int1
rjmp int1_ISR

.org 0x0100;main program for initialization and keeping CPU busy

reset:
         ;Loading stack pointer address
      LDI R16,0x70
         OUT SPL,R16
         LDI R16,0x00
         OUT SPH,R16

         LDI R16,0x01;Interface port B pin0 to be output to view LED blinking
         OUT DDRB,R16

         LDI R16,0x00;Interface port D as input
         OUT DDRD,R16

         IN R16,MCUCR;Set MCUCR register to enable low level interrupt
         ORI R16,0x00
         OUT MCUCR,R16

         IN R16,GICR;Set GICR register to enable interrupt 1
         ORI R16,0x80
         OUT GICR,R16

         LDI R16,0x00;clearing port B
         OUT PORTB,R16

         SEI;setting interrupt bit in SREG to 1 (enables interrupt globally)
ind_loop:rjmp ind_loop;infinite loop

int1_ISR:IN R16,SREG
               PUSH R16

               LDI R16,0x0A
               MOV R0,R16;Loading 10 value and counting it in R0
               ;to make LED toggle for 20 seconds
       c1:     LDI R16,0x01;LED on
               OUT PORTB,R16

               LDI R16,0x04
       a1:     LDI R17,0xFA
       a2:     LDI R18,0xFA
       a3:     DEC R18
               NOP; wasting clock cycle for delay
               BRNE a3;Branch if Z flag = 0 (R18 not equals 0)
               DEC R17
               BRNE a2;Branch if Z flag = 0 (R17 not equals 0)
               DEC R16
               BRNE a1;Branch if Z flag = 0 (R16 not equals 0)

               LDI R16,0x00;LED off
               OUT PORTB,R16

               LDI R16,0x04
       b1:     LDI R17,0xFA
       b2:     LDI R18,0xFA
       b3:     DEC R18
               NOP; wasting clock cycle for delay
               BRNE b3;Branch if Z flag = 0 (R18 not equals 0)
               DEC R17
               BRNE b2;Branch if Z flag = 0 (R17 not equals 0)
               DEC R16
               BRNE b1;Branch if Z flag = 0 (R16 not equals 0)
```

```
          DEC R0
          BRNE c1;Branch if Z flag = 0 (R0 not equals 0)POP R16
          OUT SREG, R16
      RETI;return from interrupt
```

2. Use int0 to redo the same in the demo program (duly filled in). Once the switch is pressed the LED should blink 10 times (ON (or OFF) - 1 sec, duty cycle could be 50 %). Demonstrate both the cases.

(a) Flow chart:



(b)<u>**Code:**</u>

```
Ad .org 0x0000;location for resetrjmp reset

.org 0x0001;location for external interrupt Int0rjmp int0_ISR

.org 0x0100;main program for initialization and keeping CPU busy
```

```
reset:
         ;Loading stack pointer address
      LDI R16,0x70
         OUT SPL,R16
         LDI R16,0x00
         OUT SPH,R16

         LDI R16,0x01;Interface port B pin0 to be output to view LED blinking
         OUT DDRB,R16

         LDI R16,0x00;Interface port D as input
         OUT DDRD,R16

         IN R16,MCUCR;Set MCUCR register to enable low level interrupt
         ORI R16,0x00
         OUT MCUCR,R16

         IN R16,GICR;Set GICR register to enable interrupt 0
         ORI R16,0x40
         OUT GICR,R16

         LDI R16,0x00;clearing port B
         OUT PORTB,R16

         SEI;setting interrupt bit in SREG to 1 (enables interrupt globally)
ind_loop:rjmp ind_loop;infinite loop

int0_ISR:IN R16,SREG
               PUSH R16

               LDI R16,0x0A
               MOV R0,R16;Loading 10 value and counting it in R0
               ;to make LED toggle for 20 seconds
       c1:     LDI R16,0x01;LED on
               OUT PORTB,R16

               LDI R16,0x04
       a1:     LDI R17,0xFA
       a2:     LDI R18,0xFA
       a3:     DEC R18
               NOP; wasting clock cycle for delay
               BRNE a3;Branch if Z flag = 0 (R18 not equals 0)
               DEC R17
               BRNE a2;Branch if Z flag = 0 (R17 not equals 0)
               DEC R16
               BRNE a1;Branch if Z flag = 0 (R16 not equals 0)

               LDI R16,0x00;LED off
               OUT PORTB,R16

               LDI R16,0x04
       b1:     LDI R17,0xFA
       b2:     LDI R18,0xFA
       b3:     DEC R18
               NOP; wasting clock cycle for delay
               BRNE b3;Branch if Z flag = 0 (R18 not equals 0)
               DEC R17
               BRNE b2;Branch if Z flag = 0 (R17 not equals 0)
               DEC R16
               BRNE b1;Branch if Z flag = 0 (R16 not equals 0)
```

```
        DEC R0
        BRNE c1;Branch if Z flag = 0 (R0 not equals 0)POP R16
        OUT SREG, R16
RETI;return from interrupt
```

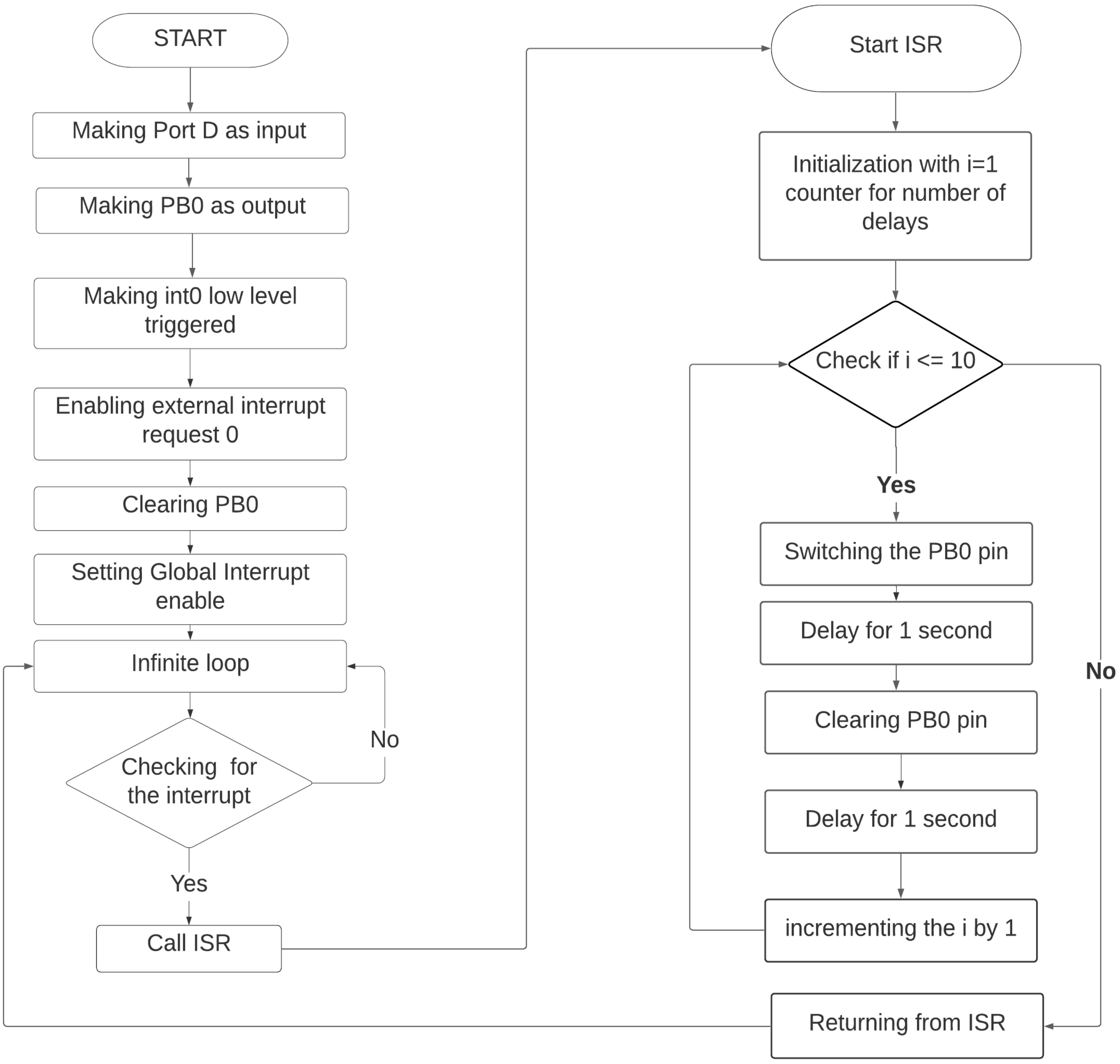
Logical explanation for 1 second delay in assembly language:

Using simple loops and counting clock cycles, we can create desired delay in assembly. Here for 1 second, we need 106 clock cycles as clock frequency of Atmega8 is 1MHz. DEC takes 1 instruction cycle and BRNE takes 2 instruction cycles if it jumps back else it takes 1 instruction cycle and NOP wastes 1 cycle. Together they make 4 cycles which has to be repeated 2,50,000 times. So, we should take 3 appropriates values which when multiplied together gives 2,50,000 and load them in 3 different GPRs. After execution of all instructions, we get a tiny delay which is due to some instructions executing when registers get empty or if BRNE doesn't jump back

3.Rewrite the program in 'C' (int1). Rewrite the C program for int0.

For int1:

(a) **Flow chart:**

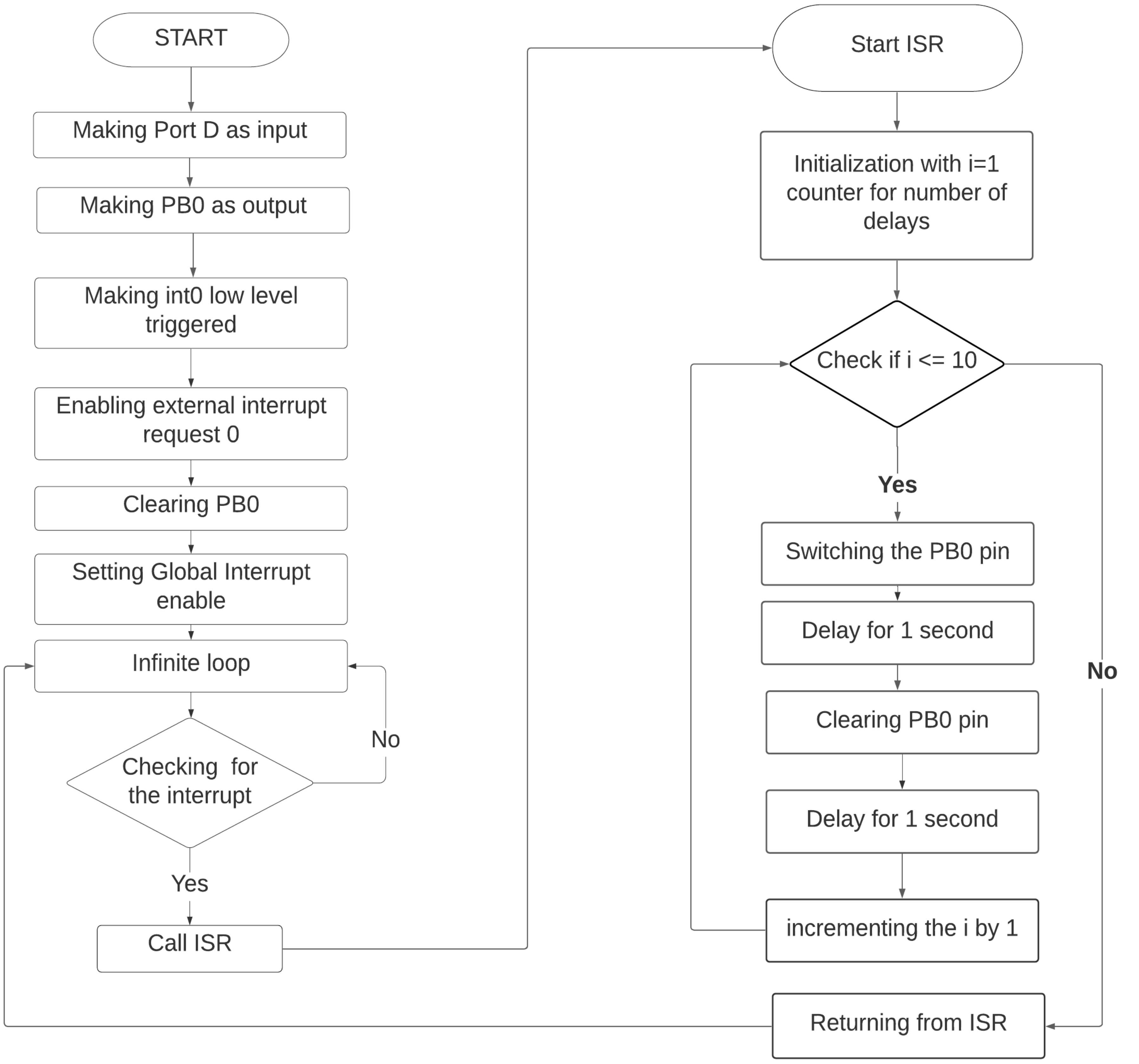


(B)code:

```
#define F_CPU 1000000 // clock frequency#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>

ISR (INT1_vect)
{
        int i;
        for (i=1;i<=10;i++) // for 10 times LED blink

        {
                PORTB=0x01;
                _delay_ms(1000);   // delay of 1 secPORTB=0x00;
                _delay_ms(1000);


        }
                   int main(void)
    {
        //Set the input/output pins appropriately
        //To enable interrupt and port interfacing
        //For LED to blink
        DDRD=0x00;   //Set appropriate data direction for DDRB=0x01;  //Make PB0 as output
        MCUCR=0x00; //Set MCUCR to level triggeredGICR=0x80;    //Enable interrupt 1
        PORTB=0x00;
        sei();       // global interrupt flag

        while (1) //wait


        }


        }
```

FOR INT 0:

(A) FLOWCHART:



(b) Code:

```
#define F_CPU 1000000 // clock frequency#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>

ISR (INT0_vect)
{
       int i;
       for (i=1;i<=10;i++) // for 10 times LED blink

       {
              PORTB=0x01;
              _delay_ms(1000);   // delay of 1 secPORTB=0x00;
              _delay_ms(1000);


       }


}
int main(void)
{
       //Set the input/output pins appropriately
       //To enable interrupt and port interfacing
       //For LED to blink
       DDRD=0x00;   //Set appropriate data direction for DDDRB=0x01; //Make PB0
       as output
       MCUCR=0x00; //Set MCUCR to level triggeredGICR=0x40;   //Enable
       interrupt 0 PORTB=0x00;
       sei();       // global interrupt flag

       while (1) //wait
       {

       }
}
```

My learnings from the experiment:

I learnt how to make appropriate delays using loops by required no of clock cycles. I learnt about the interrupt vector table and understood how an interruptworks and how critical the role of stack in it is. I also got and idea about how to enable/disable an interrupt using General Interrupt Control Register (GICR) and how to configure external hardware interrupts using MCU Control Register (MCUCR).