Experiment 2: Interrupts and Timers in Atmel AVR Atmega

Andapally Snehitha
EE20B008

1. Aim

Using Atmel AVR assembly language programming, implement interrupts and timers in Atmel Atmega microprocessor. The main constraint is that, it should be emulation only (due to ongoing pandemic). Aims of this experiment are:

- Generate an external (logical) hardware interrupt using an emulation of a push button switch.
- Write an ISR to switch ON an LED for a few seconds (10 secs) and then switch OFF.(The lighting of the LED could be verified by monitoring the signal to switch it ON).
- Use the 16 bit timer to make an LED blink with a duration of 1 sec. Also, one

needs to implement all of the above using C-interface.

2. Experiment Required

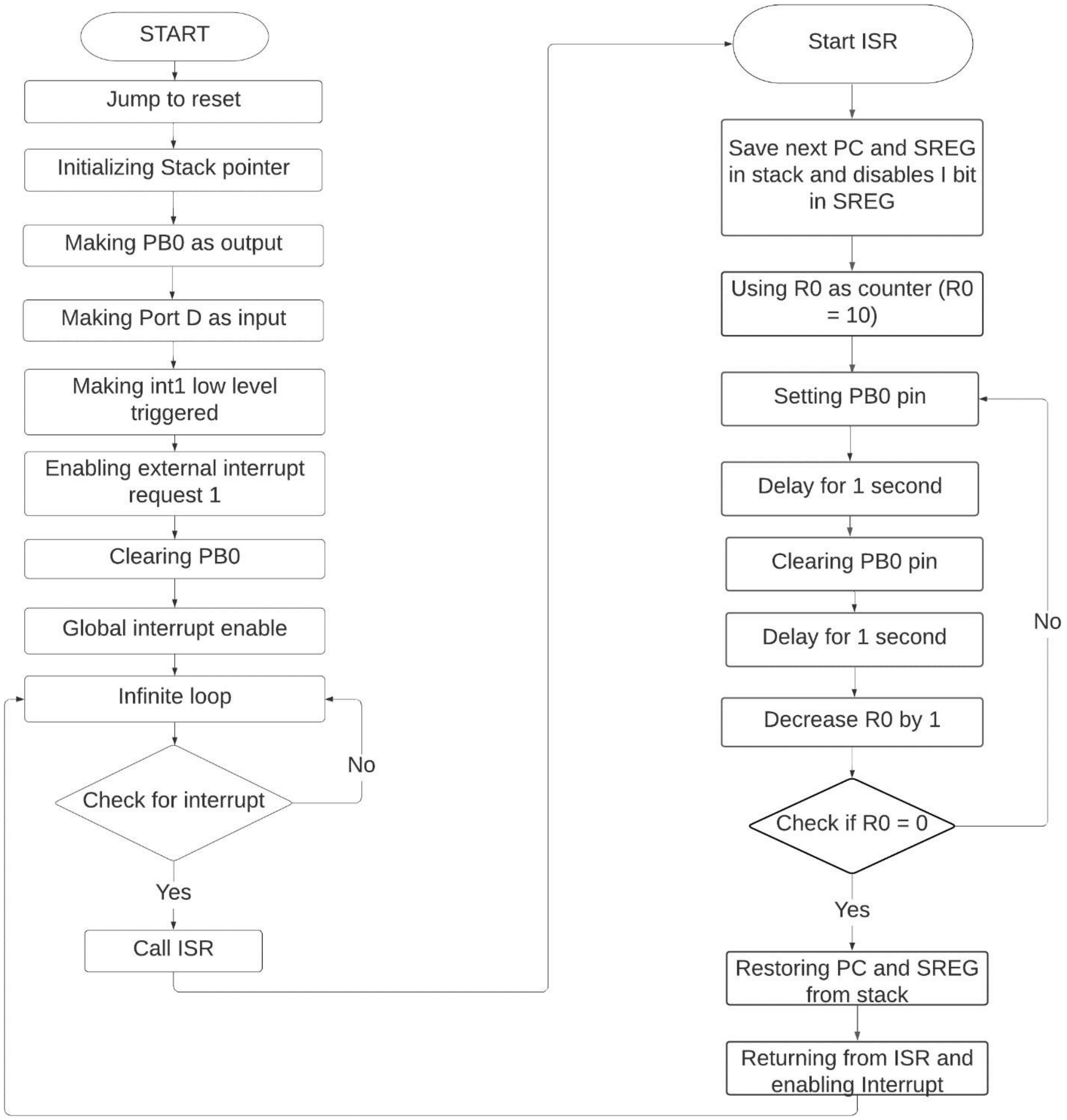
Since this is an emulation based experiment, we need only a PC with the following software : Microchip studio simulation software.

3. Tasks to be done

1. Fill in the blanks in the assembly code.
2. Use int0 to redo the same in the demo program (duely filled in). Once the switch is pressed the LEDshould blink 10 times (ON (or OFF) - 1 sec, duty cycle could be 50 %).Demonstrate both the cases
3. Rewrite the program in ’C’ (int1). Rewrite the C program for int0.
4. Demonstrate both the cases (of assembly and C). (Taking CPU frequency to be 1MHZ)

1.

(a) Flow chart:


START
Jump to reset
Initializing Stack pointer
Making PB0 as output
Making Port D as input
Making int1 low level triggered
Enabling external interrupt request 1
Clearing PB0
Global interrupt enable
Infinite loop
Check for interrupt
No
Yes
Call ISR
Start ISR
Save next PC and SREG in stack and disables I bit in SREG
Using R0 as counter (R0 = 10)
Setting PB0 pin
Delay for 1 second
Clearing PB0 pin
Delay for 1 second
Decrease R0 by 1
Check if R0 = 0
No
Yes
Restoring PC and SREG from stack
Returning from ISR and enabling Interrupt

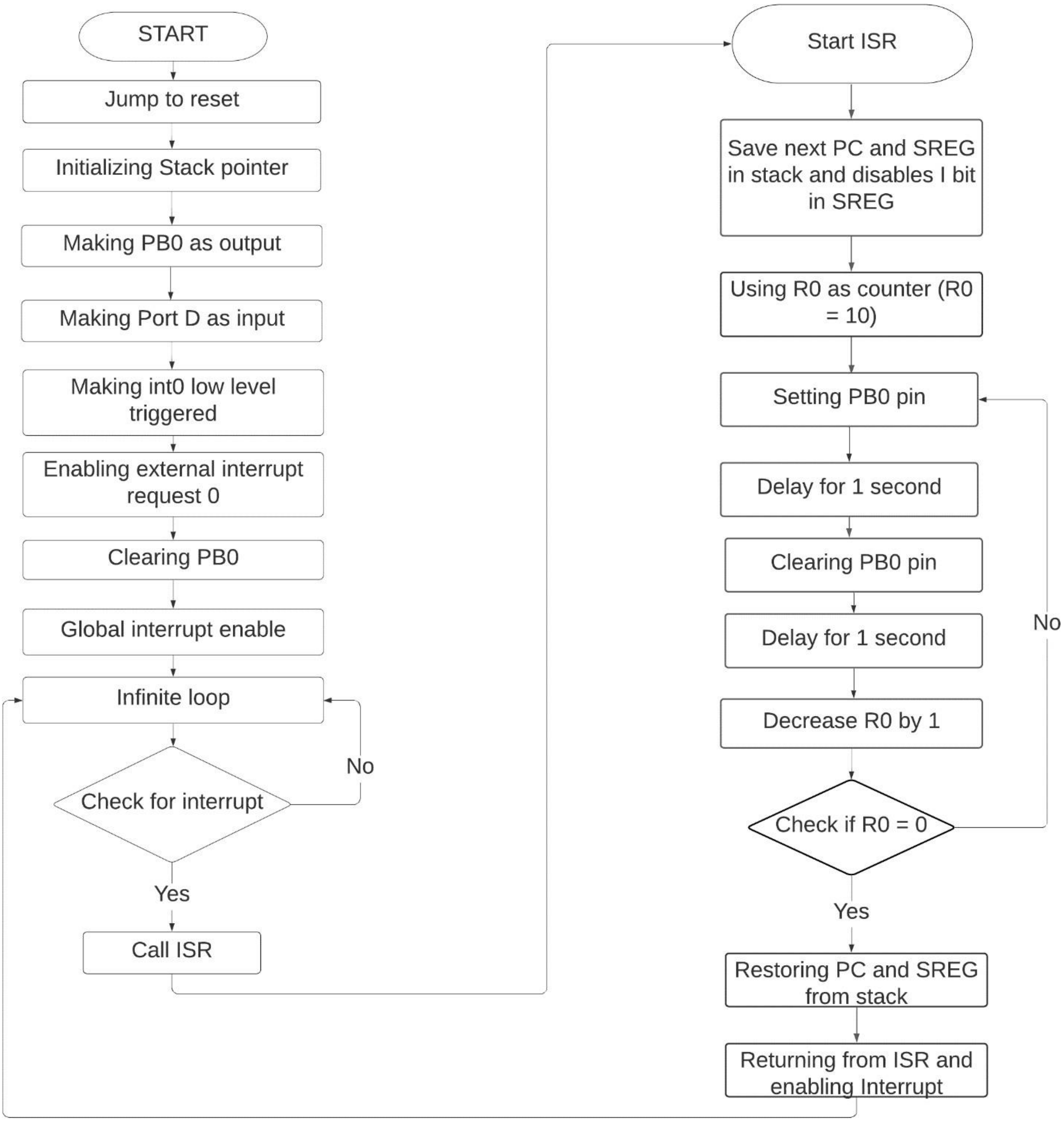
(b) Code:

```
.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.

(a)   Flow chart:


START
Jump to reset
Initializing Stack pointer
Making PB0 as output
Making Port D as input
Making int0 low level triggered
Enabling external interrupt request 0
Clearing PB0
Global interrupt enable
Infinite loop
Check for interrupt
No
Yes
Call ISR
Start ISR
Save next PC and SREG in stack and disables I bit in SREG
Using R0 as counter (R0 = 10)
Setting PB0 pin
Delay for 1 second
Clearing PB0 pin
Delay for 1 second
Decrease R0 by 1
Check if R0 = 0
No
Yes
Restoring PC and SREG from stack
Returning from ISR and enabling Interrupt

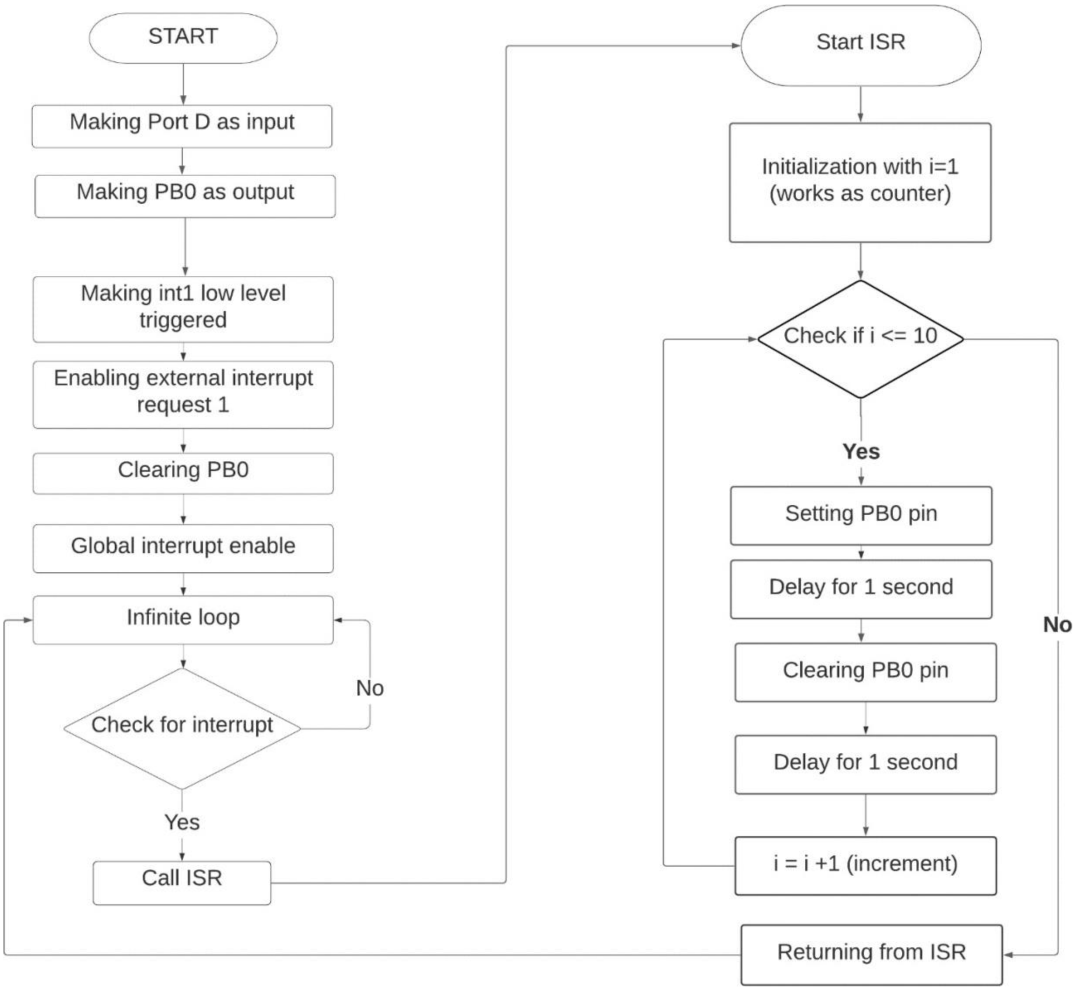
(b) Code:

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

.org 0x0001;location for external interrupt Int0
rjmp 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
```

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

For int1:

(a) Flow chart:



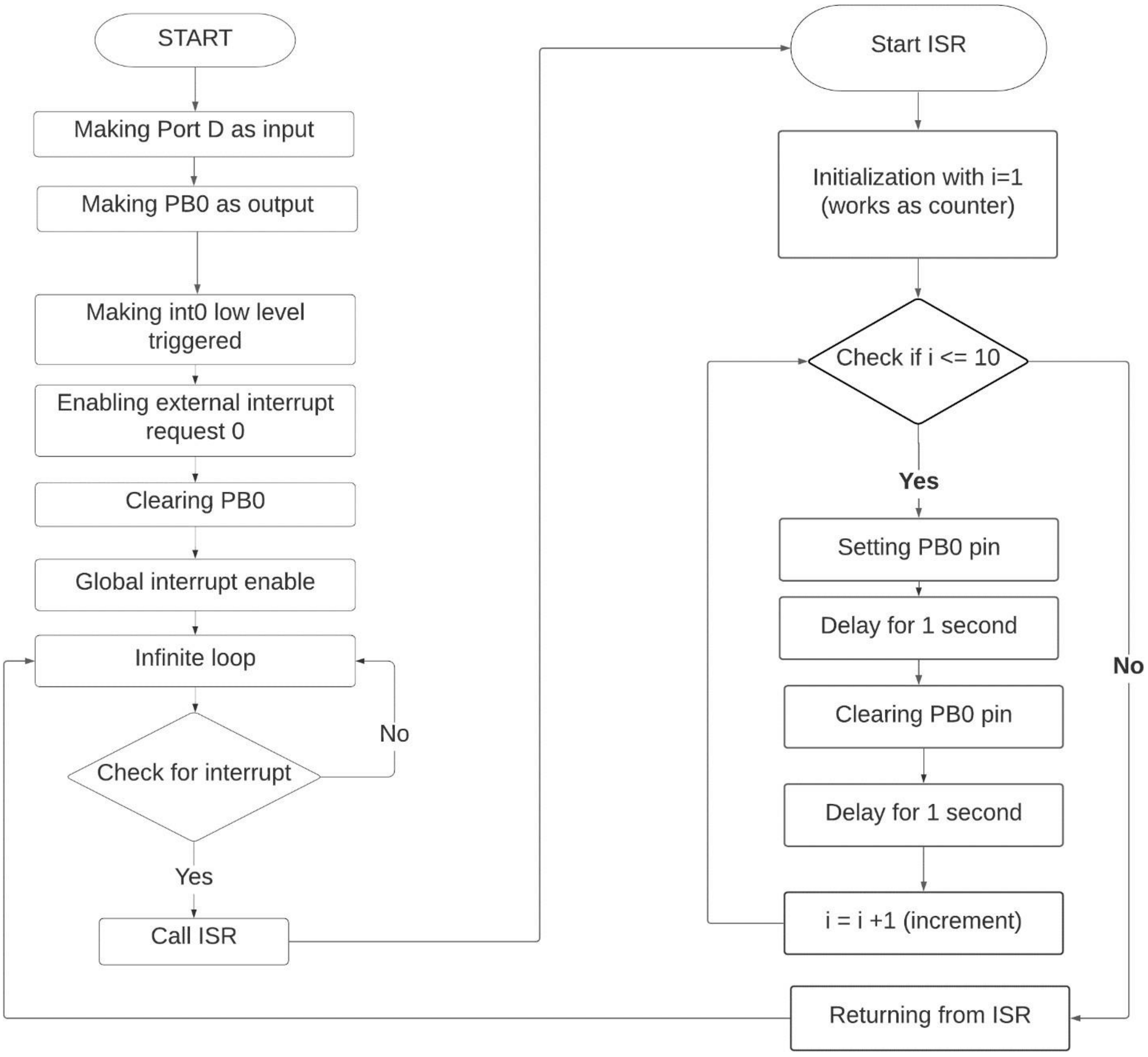
(b) Code:

```
#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 sec
                PORTB=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 D
        DDRB=0x01;    //Make PB0 as output
        MCUCR=0x00; //Set MCUCR to level triggered
        GICR=0x80;    //Enable interrupt 1
        PORTB=0x00;
        sei();        // global interrupt flag

        while (1) //wait
        {

        }
}
```

For int0:

(a) Flow chart:



(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 sec
              PORTB=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 D
       DDRB=0x01;   //Make PB0 as output
       MCUCR=0x00; //Set MCUCR to level triggered
       GICR=0x40;   //Enable interrupt 0
       PORTB=0x00;
       sei();       // global interrupt flag

       while (1) //wait
       {

       }
}
```

Inferences:

I learnt how an AVR handles external interrupts using C and assembly language. I learnt the various ways to add delay in assembly programming, and how to use delay ms function in C programming. We also learnt about duration of various instruction in terms of instrution cycle. Finally we also got to learn about the organization of ATmega8.