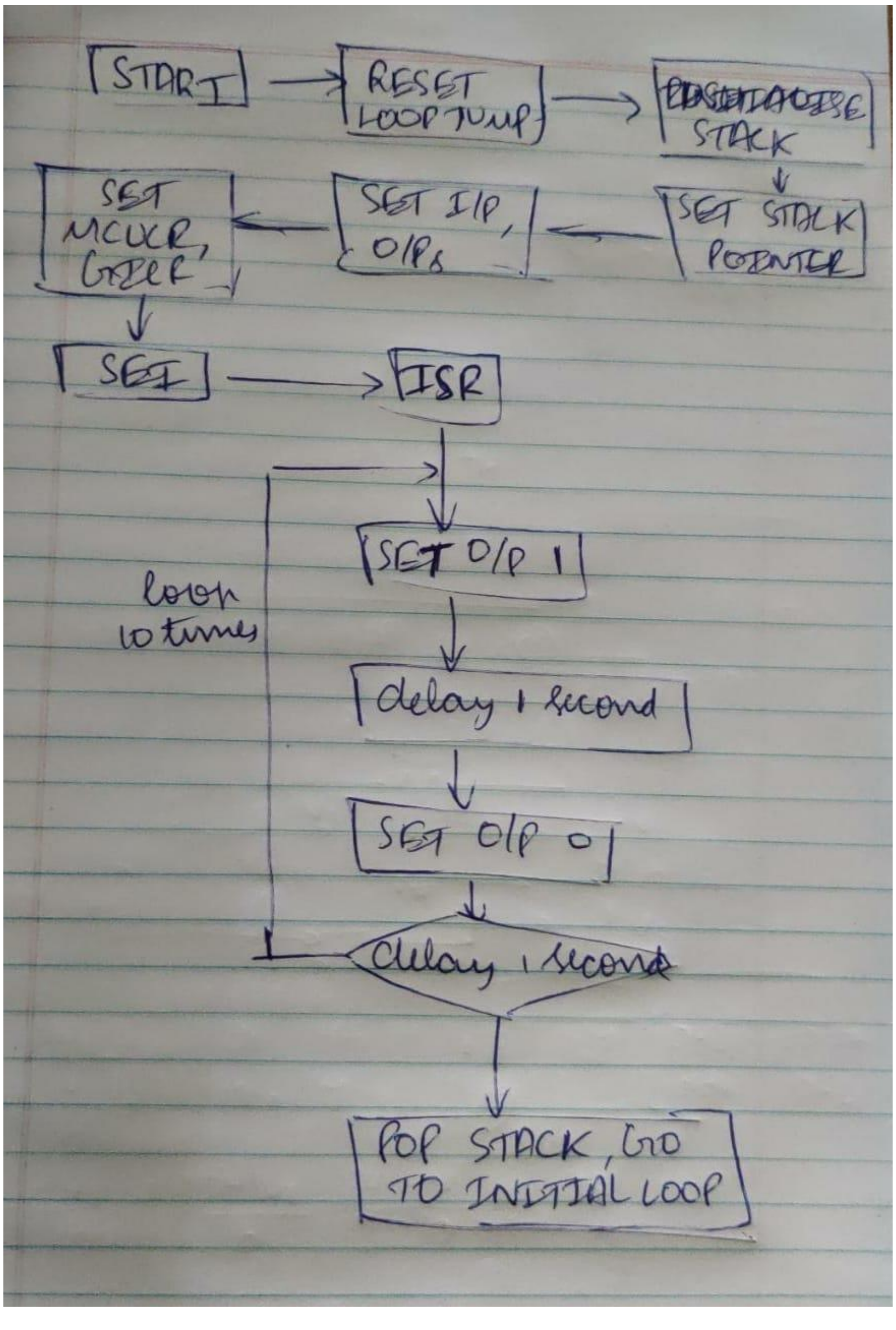
EXPERIMENT 2: INTERRUPTS AND TIMERS IN ATMEL AVR ATMEGA8

EE20B056 – KATHIR PAGALAVAN

TARGET: Implement interrupts to flash an LED with 50% duty cycle for 10seconds when a push button emulator is pressed

FLOWCHART:



CODES:

ASM PROGRAM WITH INT1 INTERRUPT

```
.org 0x0000
rjmp reset
.org 0x0004
rjmp int1_ISR
.org 0x0100
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
OUT DDRB,R16 ;so to view LED blinking
LDI R16,0x00
OUT DDRD,R16
LDI R16,0X00 ;Set MCUCR register to enable low level interrupt
OUT MCUCR,R16
LDI R16,1<<INT1 ;Set GICR register to enable interrupt 1
OUT GICR,R16
LDI R16,0x00
OUT PORTB,R16
SEI
ind_loop:rjmp ind_loop
```

```
int1_ISR:
IN R16,SREG
PUSH R16
LDI R16,0x0A
MOV R0,R16
;Modify below loops to make LED blink for 1 sec
c1: LDI R16,0x01
OUT PORTB,R16
LDI R16,4
a1: LDI R17,200
a2: LDI R18,250
a3:
NOP
NOP
DEC R18
BRNE a3
DEC R17
BRNE a2
DEC R16
BRNE a1
LDI R16,0x00
OUT PORTB,R16
LDI R16,4
b1: LDI R17,200
b2: LDI R18,250
b3:
```

```
NOP
NOP
DEC R18
BRNE b3
DEC R17
BRNE b2
DEC R16
BRNE b1
DEC R0
BRNE c1
POP R16
OUT SREG, R16
RETI
```

C PROGRAM WITH INT1 INTERRUPT

```
#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)
{
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
{
}
}
```

ASM PROGRAM WITH INT0 INTERRUPT

```
.org 0x0000
rjmp reset
.org 0x0002
rjmp int0_ISR
.org 0x0100
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
OUT DDRB,R16 ;so to view LED blinking
LDI R16,0x00
OUT DDRD,R16
LDI R16,0X00 ;Set MCUCR register to enable low level interrupt
OUT MCUCR,R16
LDI R16,1<<INT0 ;Set GICR register to enable interrupt 0
OUT GICR,R16
LDI R16,0x00
OUT PORTB,R16
SEI
ind_loop:rjmp ind_loop
int0_ISR:
IN R16,SREG
PUSH R16
LDI R16,0x0A
MOV R0,R16
; below loops to make LED blink for 1 sec
c1: LDI R16,0x01
OUT PORTB,R16
LDI R16,4
```

```
a1: LDI R17,200
a2: LDI R18,250
a3:
NOP
NOP
DEC R18
BRNE a3
DEC R17
BRNE a2
DEC R16
BRNE a1
LDI R16,0x00
OUT PORTB,R16
LDI R16,4
b1: LDI R17,200
b2: LDI R18,250
b3:
NOP
NOP
DEC R18
BRNE b3
DEC R17
BRNE b2
DEC R16
BRNE b1
DEC R0
```

```
BRNE c1
POP R16
OUT SREG, R16
RETI
```

C PROGRAM FOR INT0 INTERRUPT

```
#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)
{
DDRD=0x00; //Set appropriate data direction for D
DDRB=0x01; //Make PB0 as output

MCUCR=0x00; //Set MCUCR to low level triggered

GICR=0x40; //Enable interrupt 0

PORTB=0x00;

sei(); // global interrupt flag

while (1) //wait

{

}

}
```

INFERENCES:

NOP instructions can be used to make deliberate 1 cycle delays to time the loops to 1 second exactly

Interrupt execution response lasts 4 clock cycles