EE23B022RprtLab4

Deepak Charan S ee23b022

August 2024

1 Introduction:

I am Deepak Charan S (Roll No: EE23B022) and this is my report for the fourth assignment of Microprocessor Lab, which I had done on 3/9/24.

2 Objective:

1. Generate an external hardware interrupt using an emulation of a push button switch.
2. Write an ISR (Interrupt Service Routine) to switch ON an LED for a few seconds (10secs) and then switch OFF.

3 Equipments/Software Required:

1. Atmel AVR (Atmel8L) Chip
2. A breadboard with microprocessor socket
3. push button and an LED
4. Resistor and wires
5. AVR Programmer (USB-ASP)
6. A windows PC loaded with Microchip Studio 7 and AVR Burn-O-MAT (for burning asm)

4 General Procedure:

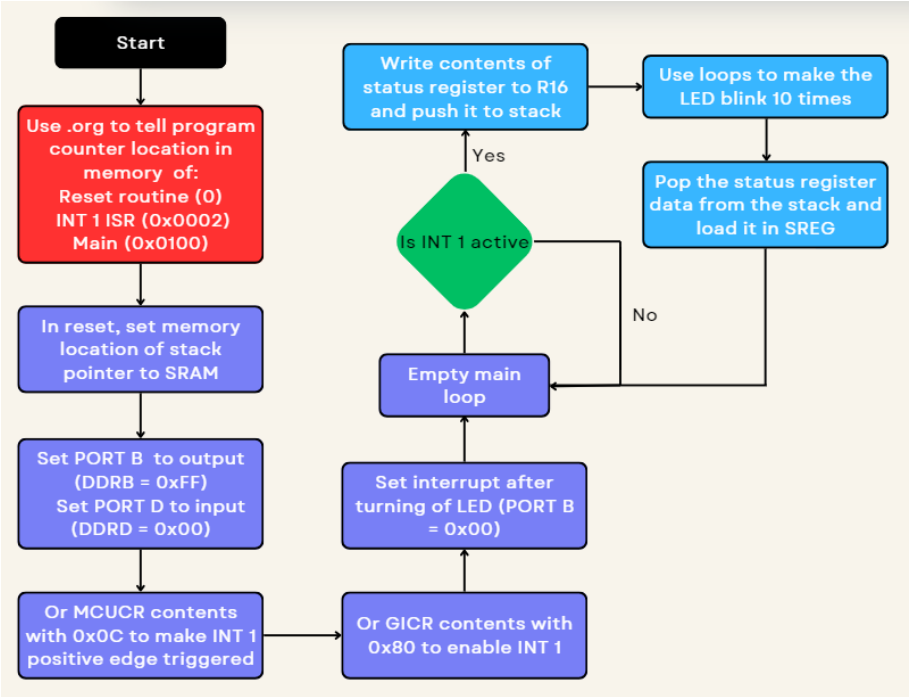
1. Write the AVR assembly code on microchip studio IDE to accomplish the objective (Using the sample code)
2. Clean existing solutions and build new solution at the start of debugging
3. Debug and test them out line by line on microchip studio and analyse the values in registers, SRAM and carry flags
4. Wire the required components properly on the breadboard
5. Use the AVR Programmer to connect the breadboard and the PC
6. generate the .hex file from Microchip Studio
7. Use the .hex to burn the file into the AVR chip using AVR Burn-O-MAT
8. Test out the hardware to verify the result (ensure LED blinks exactly 10 times with a 1 sec frequency)

5 Problem 1: Interrupt using INT 1

5.1 Implemention:

1. Direct the program counter to correct location in program memory when reset or INT 1 occurs (ISR), and a main code loop, using .org
2. In reset, load memory location of stack pointer, which we will used to store status register while context switching
3. Make PORT B output by loading 1s in DDRB
4. Make PORT D input by loading 0s in DDRD
5. Or MCUCR contents with 0x0C to make INT 1 positive edge triggered
6. Or GICR contents with 0x80 to enable INT 1
7. Write a main code loop that does nothing
8. In the ISR, write contents of status register to the stack
9. Create a loop that runs 10 times

10. Then create a nested for loop which does noting but iterate over 10,00,000 times. This is done to create a delay for the LED to toggle to 0
11. Set PORTB to 0 so that LED stops glowing
12. Repeat the delay process and switch on the LED again
13. Loop back to step 2 so that we observe a continuous pulse for 10 cycles
14. Retrieve the Status register data from the stack and load it



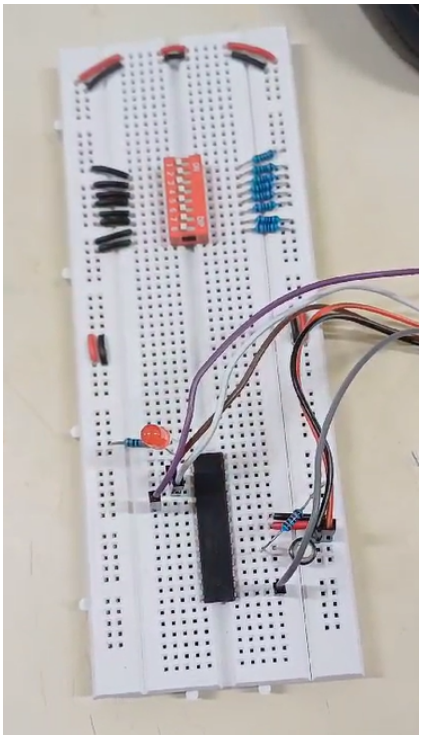
5.2 Code:

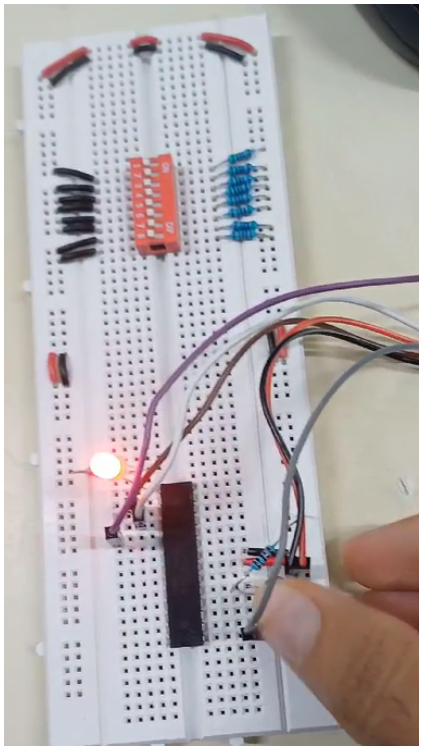
```
; int1.asm

.org 0
rjmp reset

.org 0x0004      ;address of interrupt 1
rjmp int1_ISR

.org 0x0100

```

```
11 reset:
12     LDI R16,0x70
13     OUT SPL,R16
14     LDI R16,0x00
15     OUT SPH,R16
16
17     LDI R16,0xFF
18     OUT DDRB,R16
19
20     LDI R16,0x00
21     OUT DDRD,R16
22
23     IN R16, MCUCR;Load MCUCR register
24     ORI R16,0x0C
25     OUT MCUCR,R16
26
27     IN R16, GICR; Load GICR register
28     ORI R16,0x80
29     OUT GICR,R16
30
31     LDI R16,0x00
32     OUT PORTB,R16
33
34     SEI
35
36 ind_loop:rjmp ind_loop
37
38 int1_ISR:        ; Interrupt 1 sequence
39     IN R16,SREG
40     PUSH R16
41
42     LDI R16,0x0A
43     MOV R0,R16
44
45     c1:
46         LDI R16,0xFF
47         OUT PORTB,R16
48
49         LDI R17, 0x21
50         LOOP1:LDI R18, 0x32
51             LOOP2:LDI R19, 0x64
52                 LOOP3:DEC R19
53                     BRNE LOOP3
```

```
54                  DEC R18
55                  BRNE LOOP2
56              DEC R17
57              BRNE LOOP1
58
59          LDI R16,0x00
60          OUT PORTB,R16
61
62          LDI R17, 0x21
63          LOOP4:LDI R18, 0x32
64              LOOP5:LDI R19, 0x64
65                  LOOP6:DEC R19
66                      BRNE LOOP6
67                  DEC R18
68                  BRNE LOOP5
69              DEC R17
70              BRNE LOOP4
71
72          DEC R0
73          BRNE c1
74      POP R16
75      OUT SREG,R16
76
77      RETI
```

5.3 Photos:





6 Problem 2: Interrupt using INT 0

6.1 Implemention:

1. Direct the program counter to correct location in program memory when reset or INT 0 occurs (ISR), and a main code loop, using .org
2. In reset, load memory location of stack pointer, which we will used to store status register while context switching
3. Make PORT B output by loading 1s in DDRB
4. Make PORT D input by loading 0s in DDRD
5. Or MCUCR contents with 0x03 to make INT 0 positive edge triggered
6. Or GICR contents with 0x40 to enable INT 0
7. Write a main code loop that does nothing

8. In the ISR, write contents of status register to the stack
9. Create a loop that runs 10 times
10. Then create a nested for loop which does noting but iterate over 10,00,000 times. This is done to create a delay for the LED to toggle to 0
11. Set PORTB to 0 so that LED stops glowing
12. Repeat the delay process and switch on the LED again
13. Loop back to step 2 so that we observe a continuous pulse for 10 cycles
14. Retrieve the Status register data from the stack and load it

6.2 Code:

```
; int0.asm

.org 0
rjmp reset

.org 0x0002      ; address of interrupt 0
rjmp int0_ISR

.org 0x0100

reset:
    LDI R16,0x70
    OUT SPL,R16
    LDI R16,0x00
    OUT SPH,R16

    LDI R16,0xFF
    OUT DDRB,R16

    LDI R16,0x00
    OUT DDRD,R16

    IN R16, MCUCR;Load MCUCR register
    LDI R16,0x03
    OUT MCUCR,R16

    IN R16, GICR; Load GICR register
```

```
28      LDI R16,0x40
29      OUT GICR,R16
30
31      LDI R16,0x00
32      OUT PORTB,R16
33
34      SEI
35
36  ind_loop:rjmp ind_loop
37
38  int0_ISR:    ; interrupt 0 sequence
39      IN R16,SREG
40      PUSH R16
41
42      LDI R16,0x0A
43      MOV R0,R16
44
45      c1:
46          LDI R16,0xFF
47          OUT PORTB,R16
48
49          LDI R17, 0x21
50          LOOP1:LDI R18, 0x32
51              LOOP2:LDI R19, 0x64
52                  LOOP3:DEC R19
53                      BRNE LOOP3
54                  DEC R18
55                  BRNE LOOP2
56              DEC R17
57              BRNE LOOP1
58
59          LDI R16,0x00
60          OUT PORTB,R16
61
62          LDI R17, 0x21
63          LOOP4:LDI R18, 0x32
64              LOOP5:LDI R19, 0x64
65                  LOOP6:DEC R19
66                      BRNE LOOP6
67                  DEC R18
68                  BRNE LOOP5
69              DEC R17
70              BRNE LOOP4
```

```
71
72          DEC R0
73          BRNE c1
74      POP R16
75      OUT SREG,R16
76
77      RETI
```

7 Interpretations of result

Controlling the atmgea8 through an external interrupt was demonstrated through a breadboard circuit.

Note: The interrupt part of the assembly code was handled by me.

8 References:

Handouts, Sample Code and Instruction Manual of AVR provided in moodle. Slight reference from Mazidi's "The AVR Microcontroller and embedded systems" was also used.