EE2016 : Microprocessor Theory and Lab

Lab Experiment # 3

Hardware Wiring and Programming for interrupts by ASM and C-programming using Atmel Atmega8 AVR

Batch 35

Sarathchandra Koppera EE21B073

Vikas Gollapalli EE21B051

Aim:

This experiment introduces assembly programming and interaction with peripherals in Atmel At-mega8 microcontroller

• Wire the microcontroller along with the given peripherals in a breadboard.

• Program the microcontroller to read the DIP switch values and display it in an LED using assembly programming.

• Program the microcontroller to perform the addition and multiplication of two four bit numbers which are read from the DIP switches connected to a port and display the result using LEDs connected to another port.

Code for blink (without switch)

```
.CSEG
 LDI R16, 0x01
 OUT DDRB, R16

 again: LDI R16, 0x01
OUT PORTB, R16

LDI R16, 0xFF
loop1:  LDI R17, 0xFF
loop2:  DEC R17
BRNE loop2
DEC R16
BRNE loop1


LDI R16, 0x00
OUT PORTB, R16

LDI R16, 0xFF
back3:  LDI R17, 0xFF
back4:  DEC R17
BRNE back4
DEC R16
BRNE back3
rjmp again
```

If current passes through the LED, it lights up. When Port is LOW, P0 is Grounded and the current passes through the LED and it starts to glow. When P1 is HIGH, current doesn't pass through the LED and it will not glow. By repeating these two steps continuously we can create the blinking effect with the LED.

The LED blinks continuously immediately after we write the code in Burn-O-Mat

Code for blink using interrupts(Switch)

```
.CSEG
LDI R16, 0x01
OUT DDRB,R16
LDI R16,0x00
OUT DDRD, R16
again: LDI R16,0x00
OUT PORTB, R16
IN R16, PIND
COM R16
ANDI R16, 0x01
OUT PORTB,R16
rjmp again
```

When the Switch is pushed the LED is turned on and the LED is turned off when the switch is released. The switch here is the interrupt.

The hexa-decimal number in the binary is a 4 bit number. This makes pin 0 of port 1 as input pin and rest of the 7 pins are acting as output. Our button is connected to port 1 pin 0 which is declared as input by the statement P1=0x01 which can be used as input from the button. A continuous loop in code is used to continuously run the logic declared in it. According to the logic, whenever the button is high, i.e. the button is pushed , the LED glows and is switched off when released.

Code for Addition

```
#include "m8def.inc"

START:
    LDI R16, 0x00;
    OUT DDRD, R16;  Setting PORTD to INPUT

    LDI R16, 0xFF;
    OUT DDRC, R16;  Setting PORTC to OUTPUT

ADDITION:
    IN R21, PIND;   R21 <-- (<NUM2><NUM1>)
    MOV R20, R21;   Making copy of R21 in R20 for having the 2 numbers in
separate registers
    ANDI R20, 0xF0; Assigning R20 as "<NUM2>0000"
    SWAP R20;  Swapping higher and lower nibbles of R20. R20 <--
"0000<NUM2>"
    ANDI R21, 0x0F; Assigning R21 as "0000<NUM1>"
    ADD R20, R21;   R20 <-- R20 + R21

END:
    OUT PORTC, R20; PORTC <-- R20
    NOP; End of program
```

- Addition of two unsigned nibbles taken from a DIP switch
- INPUT - from DIP switch connected to PORTD
- OUTPUT - To the LEDs connected to PORTC

PORTD has 8 pins 4 for first number and 4 for second number. We assign them to their respective registers to perform addition. The final output is obtained at register R20 and we get the practical result at PORTC, which is connected to LEDs that display the result.

Code for Multiplication

```
#include "m8def.inc"

START:
    LDI R16, 0x00;
    OUT DDRD, R16;  Setting PORTD to INPUT so in the
code we take no inputs only the hardware inputs
     LDI R16, 0xFF;
    OUT DDRC, R16;  Setting PORTC to OUTPUT
ADDITION:
    IN R21, PIND;  Storing inputs from pins of port D
i.e. R21 <-- (<NUM2><NUM1>)
    MOV R20, R21;   Making copy of R21 in R20 for having
the 2 numbers in separate registers
    ANDI R20, 0xF0; Assigning R20 as "<NUM2>0000"
    SWAP R20;  Swapping higher and lower nibbles of R20.
R20 <-- "0000<NUM2>"
    ANDI R21, 0x0F; Assigning R21 as "0000<NUM1>"
    MUL R20, R21; Product stored in R0,R1, but here
effectively it will be product of two 4 bit numbers
which will be less than 8 bits , just R0 will give the
product as R1 will have 0x00
     MOV R24,R0;
END:
    OUT PINC, R24;  PORTC <-- R24
    NOP; End of program
```

PORTD has 8 pins 4 for first number and 4 for second number. Then we assign them to their respective registers to perform multiplication. The final output is obtained at register R0, copied to R24 and we get the practical result at PORTC, which is connected to LEDs that display the result.

Results

1.

main.asm

```
.CSEG
 LDI R16, 0x01
 OUT DDRB, R16

 again: LDI R16, 0x01
OUT PORTB, R16

LDI R16, 0xFF
loop1:  LDI R17, 0xFF
loop2:  DEC R17
BRNE loop2
DEC R16
BRNE loop1


LDI R16, 0x00
OUT PORTB, R16

LDI R16, 0xFF
back3:  LDI R17, 0xFF
back4:  DEC R17
BRNE back4
DEC R16
BRNE back3
rjmp again
```

Output

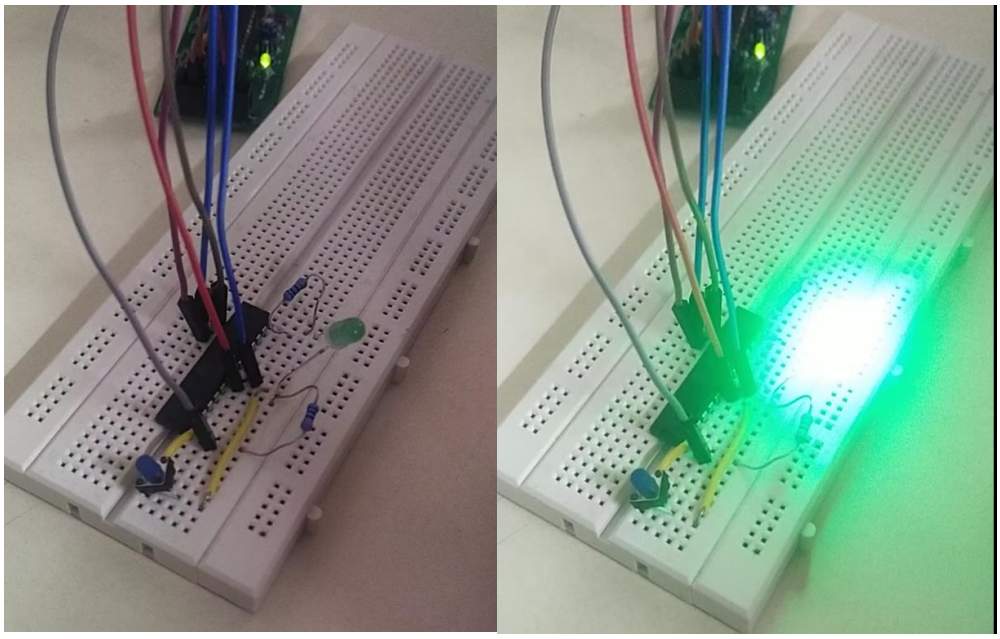
Show output from: Build

```
        [.cseg] 0x000000 0x000026     38      0     38    8192   0.5%
        [.dseg] 0x000060 0x000060      0      0      0    1024   0.0%
        [.eseg] 0x000000 0x000000      0      0      0     512   0.0%
        Assembly complete, 0 errors. 0 warnings
    Done executing task "RunAssemblerTask".
Done building target "CoreBuild" in project "AssemblerApplication5.asmproj".
Target "PostBuildEvent" skipped, due to false condition; ('$(PostBuildEvent)' != '') was evaluated as ('' != '').
Target "Build" in file "C:\Program Files (x86)\Atmel\Studio\7.0\Vs\Avr.common.targets" from project "c:\users\sarathchandra k\Docum
Done building target "Build" in project "AssemblerApplication5.asmproj".
Done building project "AssemblerApplication5.asmproj".

Build succeeded.
========== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped ==========
```

Output

Ready



LED while blining

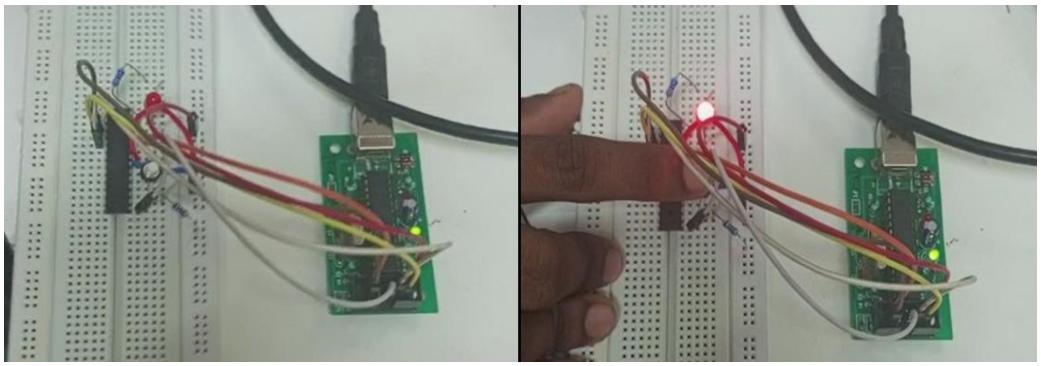
2.

```
;
; AssemblerApplication5.asm
;
; Created: 08-10-2022 22:42:19
; Author : Sarathchandra K
;


; Replace with your application code
.CSEG
LDI R16, 0x01
OUT DDRB,R16
LDI R16,0x00
OUT DDRD, R16
again: LDI R16,0x00
OUT PORTB, R16
IN R16, PIND
COM R16
ANDI R16, 0x01
OUT PORTB,R16
rjmp again
```

```
Output
Show output from: Build
        [.cseg] 0x000000 0x000016     22      0     22    8192   0.3%
        [.dseg] 0x000060 0x000060      0      0      0    1024   0.0%
        [.eseg] 0x000000 0x000000      0      0      0     512   0.0%
        Assembly complete, 0 errors. 0 warnings
    Done executing task "RunAssemblerTask".
Done building target "CoreBuild" in project "AssemblerApplication5.asmproj".
Target "PostBuildEvent" skipped, due to false condition; ('$(PostBuildEvent)' != '') was evaluated as ('' != '').
Target "Build" in file "C:\Program Files (x86)\Atmel\Studio\7.0\Vs\Avr.common.targets" from project "c:\users\sarathchandra k\Documents\Atmel St
Done building target "Build" in project "AssemblerApplication5.asmproj".
Done building project "AssemblerApplication5.asmproj".

Build succeeded.
========== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped ==========
Output
Build succeeded
```



LED glows only when the dip switch is pushed

3.

main.asm

```
#include "m8def.inc"

START:
    LDI R16, 0x00;
    OUT DDRD, R16;  Setting PORTD to INPUT

    LDI R16, 0xFF;
    OUT DDRC, R16;  Setting PORTC to OUTPUT

ADDITION:
    IN R21, PIND;   R21 <-- (<NUM2><NUM1>)
    MOV R20, R21;   Making copy of R21 in R20 for having the 2 numbers in separate registers
    ANDI R20, 0xF0; Assigning R20 as "<NUM2>0000"
    SWAP R20;  Swapping higher and lower nibbles of R20. R20 <-- "0000<NUM2>"
    ANDI R21, 0x0F; Assigning R21 as "0000<NUM1>"
    ADD R20, R21;   R20 <-- R20 + R21

END:
    OUT PORTC, R20; PORTC <-- R20
    NOP; End of program
```

Output

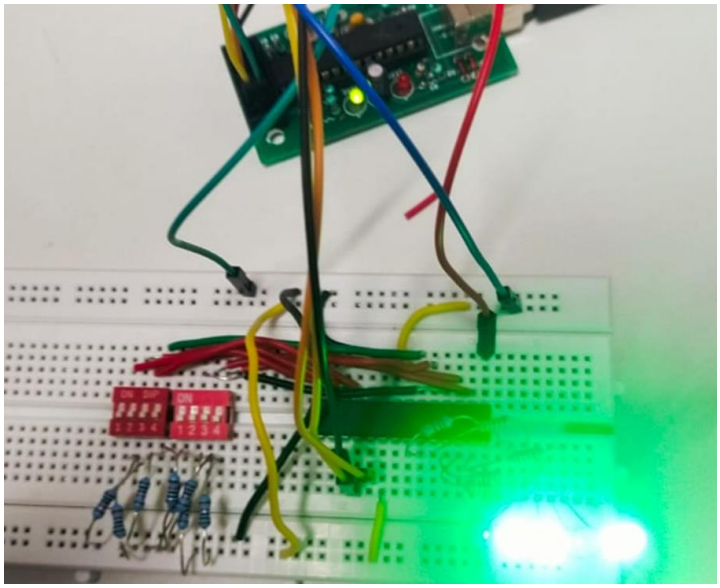
Show output from: Build

```
        [.cseg] 0x000000 0x000018     24      0     24    8192   0.3%
        [.dseg] 0x000060 0x000060      0      0      0    1024   0.0%
        [.eseg] 0x000000 0x000000      0      0      0     512   0.0%
        Assembly complete, 0 errors. 0 warnings
    Done executing task "RunAssemblerTask".
Done building target "CoreBuild" in project "AssemblerApplication5.asmproj".
Target "PostBuildEvent" skipped, due to false condition; ('$(PostBuildEvent)' != '') was evaluated as ('' != '').
Target "Build" in file "C:\Program Files (x86)\Atmel\Studio\7.0\Vs\Avr.common.targets" from project "c:\users\sarathc
Done building target "Build" in project "AssemblerApplication5.asmproj".
Done building project "AssemblerApplication5.asmproj".

Build succeeded.
========== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped ==========
```

Output

Build succeeded



LEDs glow and tell the final output(Addition)

4.

main.asm

```
#include "m8def.inc"

START:
    LDI R16, 0x00;
    OUT DDRD, R16;  Setting PORTD to INPUT so in the code we take no inputs only the hardware inputs
     LDI R16, 0xFF;
    OUT DDRC, R16;  Setting PORTC to OUTPUT
ADDITION:
    IN R21, PIND;  Storing inputs from pins of port D i.e. R21 <-- (<NUM2><NUM1>)
    MOV R20, R21;   Making copy of R21 in R20 for having the 2 numbers in separate registers
    ANDI R20, 0xF0; Assigning R20 as "<NUM2>0000"
    SWAP R20;  Swapping higher and lower nibbles of R20. R20 <-- "0000<NUM2>"
    ANDI R21, 0x0F; Assigning R21 as "0000<NUM1>"
    MUL R20, R21; Product stored in R0,R1, but here effectively it will be product of two 4 bit numbers wh
     MOV R24,R0;
END:
    OUT PINC, R24;  PORTC <-- R24
    NOP; End of program
```

Output

Show output from: Build

```
        [.cseg] 0x000000 0x00001a     26      0     26    8192   0.3%
        [.dseg] 0x000060 0x000060      0      0      0    1024   0.0%
        [.eseg] 0x000000 0x000000      0      0      0     512   0.0%
        Assembly complete, 0 errors. 0 warnings
    Done executing task "RunAssemblerTask".
Done building target "CoreBuild" in project "AssemblerApplication5.asmproj".
Target "PostBuildEvent" skipped, due to false condition; ('$(PostBuildEvent)' != '') was evaluated as ('' != '').
Target "Build" in file "C:\Program Files (x86)\Atmel\Studio\7.0\Vs\Avr.common.targets" from project "c:\users\sarathchandra k\Docume
Done building target "Build" in project "AssemblerApplication5.asmproj".
Done building project "AssemblerApplication5.asmproj".

Build succeeded.
========== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped ==========
```

Output

Build succeeded

Conclusion

1. We learnt to write the code in assembly level language and how to implement AVR Burn-O-MAT software to burn the code.

2. We learnt to make an LED blink with and without a switch button.

3. We learnt to implement the logic of addition , multiplication and display the result using LEDs.