Experiment 3: Hardware Wiring and Assembly Programming using Atmel Atmega(8) AVR

Abhinav I S EE23B002

September 3, 2024

1 Objectives

1. Wire the micrcontroller with the peripherals in a breadboard.
2. Program the microcontroller to read the DIP switch values and display it in an LED using assembly programming.
3. 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 LED's connected to another port.

2 Procedure

2.1 Blinking LED Light

2.1.1 Code

```
        .include "m8def.inc"

    .def        mask         = r16                ; mask register
    .def        ledR         = r17                ; led register
    .def        oLoopR         = r18                ; outer loop register
    .def        iLoopRl = r24                ; inner loop register low
    .def        iLoopRh = r25                ; inner loop register high

    .equ        oVal         = 71                ; outer loop value
    .equ        iVal         = 1760                ; inner loop value

    .cseg
    .org        0x00
    clr        ledR                        ; clear led register
    ldi        mask,(1<<PINB0)                ; load 00000001 into mask register
    out        DDRB,mask                ; set PINB0 to output

start:
    eor        ledR,mask                ; toggle PINB0 in led register
    out        PORTB,ledR                ; write led register to PORTB

    ldi        oLoopR,oVal                ; initialize outer loop count

oLoop:
```

```
    ldi         iLoopRl,LOW(iVal)        ; intialize inner loop count in inner
    ldi         iLoopRh,HIGH(iVal)        ; loop high and low registers

iLoop:

    sbiw  iLoopRl,1                 ; decrement inner loop registers
    brne        iLoop                         ; branch to iLoop if iLoop registers != 0

    dec         oLoopR                        ; decrement outer loop register
    brne        oLoop                         ; branch to oLoop if outer loop register !=
    ↪  0

    rjmp        start                         ; jump back to start
```

1. out DDRB, mask:
   DDRB is a data direction register for port B. This line sets PINB0 to output. since mask contains the bit for pin 0 set.
2. The above code XORs the mask register with the LED register continously after a delay element, created by two nested loops.
3. the outer loop runs for 71 iterations and the inner loop runs for 1760 loops, which is implemented using a word, and we use the sbiw instruction (Subtract Immediate from Word)to decrement the inner loop variable
4. Calculating delay time, with instructions
   (a) Starting from oLoop, the first two ldi instruction takes 1 clock cycle each
   (b) The time for this loop is actually easy to calculate. We will go through the loop iVal times. Each sbiw instruction will take 2 clock cycles and the brne instruction will take 2 cycles eveytime except for the last iteration, which will only take 1.
   (c) dec oLoopR will take 1 cycle
   (d) brne oLoop will take 1 cycle normally, and 2 for the last iteration
   (e) rjmp start will take 2 cycles
   (f) eor, out, ldi takes one cycle each
   (g) so totally, delay gets estimated to

   $$outerloopcount = oVal * (1 + 1 + innerLoopCount + 1 + 2) - 1$$

   $$\text{no of clock cycles} = 1 + 1 + 1 + outerLoopCount + 2$$

   (h) The frequency of our chip is 1Mhz,

   $$total = 4 + 4 * oVal * (1 + iVal)$$

   $$500000 = total$$

   $$letoLoop = 71$$

   $$\implies iLoop \approx 1760$$

5. After writing the code, compile it, copy the hex file, use Burn O Mat to burn the program onto the microcontroller.
6. Wire the AVR programmer accordingly, and connect the led throw a pulldown resistor to pin B0.

2.1.2 output

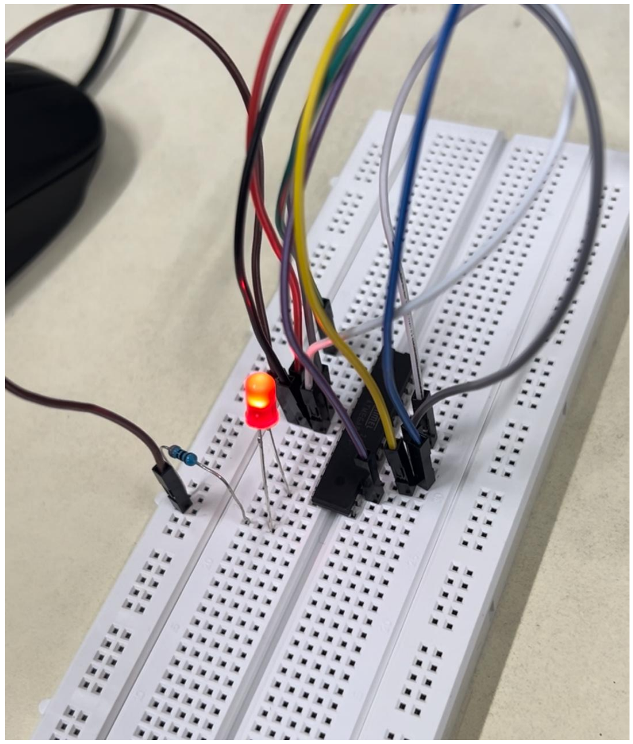

Figure 1: LED Pulse

2.2 Controlling LED using push button switch

2.2.1 Code

```
  start:
        .cseg
        .org        0x00

        ldi r16, 0x00; load pinb0 to r16
        out DDRB,r16 ;setting it to input

check_input:



        SBIS PINB, 0;0-> switch on
        rjmp light_led

        ldi r17, 0x0
        OUT PORTD, r17
        rjmp check_input

light_led:
        ldi r16, 0xFF
        OUT DDRD, r16
        ldi r17, 0x1
        OUT PORTD, r17
```

```
        rjmp check_input
```

1. First we set pinb0 to input.
2. then we check if switch is on, since is switch is pulled up to 5V when off, we have to check if it is 0 when on.
3. If button is pressed, go to light_led, which sets PORTD pin 0 to 1.
4. If button is not pressed, turn off led.
5. Similar to last assignment, compile the program, burn to microcontroller using Burn O Mat.
6. Wire the switch, pulled up to 5V to pin D0 and, ground the other side.

2.2.2 output

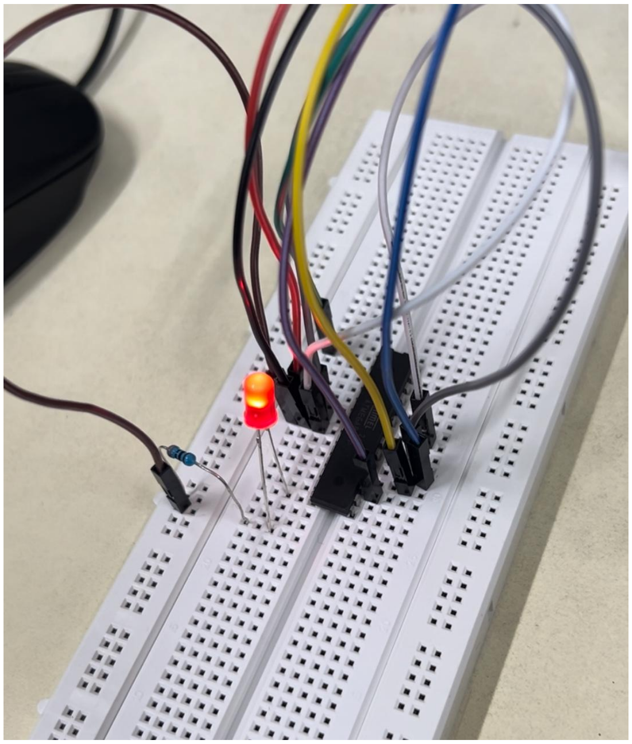

Figure 2: Push button to light LED

2.3 Adding two 4 bit numbers from DIP switch inputs

```
  .include "m8def.inc"

start:
        .cseg
        .org        0x00

        ldi r16, 0x00
        out DDRB,r16 ;setting it to input
```

```
check_input:


        IN r16, PINB
        COM r16
        mov r17, r16
        mov r18, r16

        ANDI R17, 0x0F
        ANDI R18, 0XF0

        LSR R18
        LSR R18
        LSR R18
        LSR R18

        add r17, r18

        ldi r16, 0xFF
        OUT DDRD, r16
        OUT PORTD, r17
        rjmp check_input
```

1. First we set every pin in PORTB to input
2. In the subroutine check_input
   (a) First we take PINB Input into register r16.
   (b) We take complement because the switches are pulled up in OFF condition
   (c) Copy r16 into r17 and r18
   (d) AND R17 with 0x0F to take the 4 least significant bits
   (e) AND R18 with 0xF0 to take the 4 most significant bits
   (f) then we right shift R18 four times.
3. Then we add r17 and r18
4. Set portD to output
5. Then we copy r17 into PORTD

2.3.1 Output

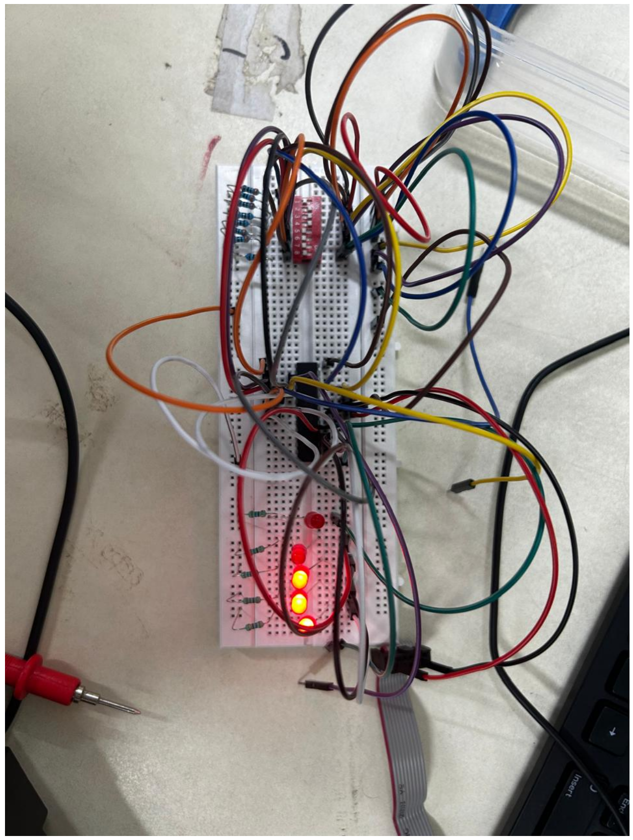

Figure 3: Adder circuit

3 My Contribution

I have written the code for all three sections, while my teammates have done majority of the wiring.