Alexis Englund, Zackery Holloway
CENG 347 Lab 3
02/13/25

CENG 347 Lab 3

Introduction:

In this lab we were to get a similar setup to lab 1 working but this time instead of using C code we were to use assembly. For part one we were to use the given assembly code to blink the on board LED. We were then to complete part two which was recreating the binary counter with the leds also in assembly.

Procedure:

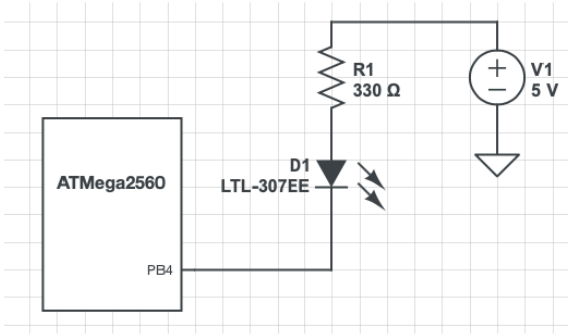
The lab is divided into two key parts:

Part 1: Load and run Blink.S

- Set up the environment for the Assembly
- Implement the Blink.S file given in the zip file
- Review and understand the delay functions
- Blink at 2Hz

Part 2: Binary Counter Implementation

- Recreate the binary counter from Lab 1
- Extend the delay to 500ms



Alexis Englund, Zackery Holloway
CENG 347 Lab 3
02/13/25

Application:

Part 1: Blink.S

Blink.cpp was downloaded and ran on the board with the needed include files.
Results were successful.

Code: (Was given to us)

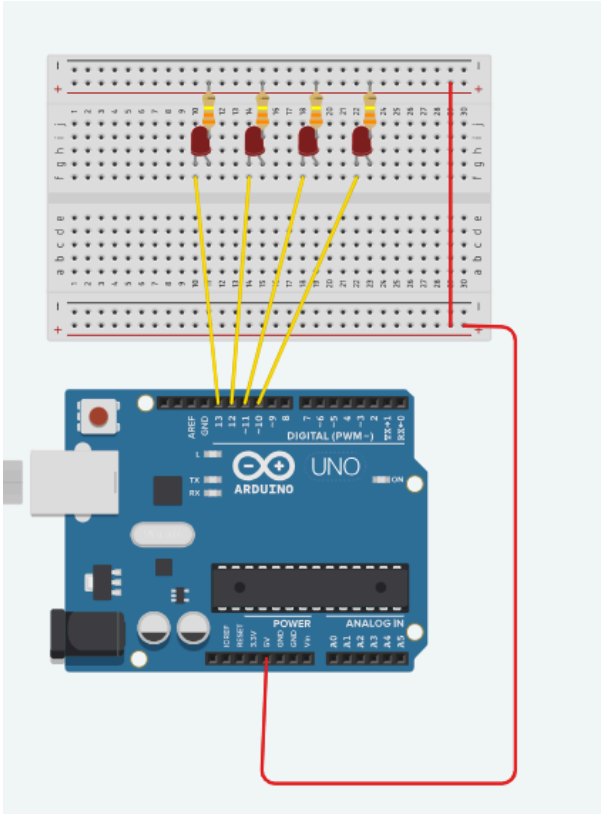
```
; Blink LED on PB7 (ATMEGA 2560 pin 13)
; http://forum.arduino.cc/index.php?topic=159572#msg1194604

#define __SFR_OFFSET 0

#include "avr/io.h"

; let the C compiler know these are called elsewhere via external
definition
.global start
.global blink

start:
  sbi   DDRB,7  ; Set PB7 as output (set bit immediate)
  ret

blink:
  ldi   r20,250 ; Set the delay duration in ms. Maximum value is 255.
  call  delay_n_ms
  sbi   PORTB,7 ; Set PB7 HIGH
  ldi   r20,250
  call  delay_n_ms
  cbi   PORTB,7 ; Set PB7 LOW ; (clear bit immediate)
  ret

delay_n_ms:
  ; Delay about r20*1ms. Destroys r20, r30, and r31.
  ; One millisecond is about 16000 cycles at 16MHz.
  ; The basic loop takes about 5 cycles, so we need about 3000 loops.
  ldi   r31, 3000>>8 ; high(3000)
  ldi   r30, 3000&255  ; low(3000)
```

Alexis Englund, Zackery Holloway
CENG 347 Lab 3
02/13/25

```
delaylp:
  sbiw    r30, 1 ; subtract 1 word
  brne    delaylp ; branch if noto equal
  subi    r20, 1 ; sub imident
  brne    delay_n_ms ; branch not equal
  ret
```

Part 2: Binary Counter Implementation in Assembly

Using the diagram provide in the lab we wired the lab to have a low signal act as a high LED. Shown below;



Blink.S was modified to allow for the counter.

Code(Zack):

```
; Count LED on PB7-4 (ATMEGA 2560 pin 13-10)
; http://forum.arduino.cc/index.php?topic=159572#msg1194604

#define __SFR_OFFSET 0

#include "avr/io.h"
```

Alexis Englund, Zackery Holloway
CENG 347 Lab 3
02/13/25

```
; let the C compiler know these are called elsewhere via external
definition
.global start
.global counter
.global delay_n_ms

start:
    ; Setup Pin 7-4 as output
    sbi     DDRB,7
    sbi     DDRB,6
    sbi     DDRB,5
    sbi     DDRB,4

    ; Sets to known state
    clr     r16
    out     PORTB, r16

    ; Sets count to zero
    clr     r17
    ret

counter:
    ldi     r20, 255            ; Set delay to 255ms
    call    delay_n_ms
    ldi     r20, 255            ; Set delay to 255ms
    call    delay_n_ms

    ; Preserve lower 4 bits of PORTB and update upper 4 bits with inverted
count
    in      r18, PORTB          ; Load PORTB value
    ldi     r21, 0x0F           ; Load 0x0F
    and     r18, r21            ; And R18 and 0x0F
    mov     r19, r17            ; Copy count value

    ; Swap nibbles to shift left 4 bits
    mov     r22, r19            ;Creates copy
    and     r19, r21            ;Keep lower nibble
    lsl     r19                 ;Shift Left by 4
```

Alexis Englund, Zackery Holloway
CENG 347 Lab 3
02/13/25

```
    lsl     r19
    lsl     r19
    lsl     r19

    and     r22, r21            ; Keeps only upper nibble
    lsl     r20                 ; shfits right by 4
    lsl     r20
    lsl     r20
    lsl     r20

    or      r19, r20            ; combines the nibbles

    ; Invers bits
    ldi     r23, 0xFF           ; loads all ones
    EOR     r19, r23            ; Xor to flip bits

    and     r18, r21            ; Ensures only upper 4 are affected
    or      r18, r19            ; Merge with PORTB lower 4 bits
    out     PORTB, r18          ; Output to PORTB

    ldi     r24, 1              ; Loads incerment value into regster
    add     r17, r24            ; increments by value

    and     r17, r21            ; Keep only lower 4 bits
    ret

delay_n_ms:
  ; Delay about r20*1ms. Destroys r20, r30, and r31.
  ; One millisecond is about 16000 cycles at 16MHz.
  ; The basic loop takes about 5 cycles, so we need about 3000 loops.
  ldi   r31, 3000>>8 ; high(3000)
  ldi   r30, 3000&255  ; low(3000)
  ldi   r25, 255
delaylp:
  sbiw    r30, 1
  brne    delaylp
  subi    r20, 1
  brne    delay_n_ms
  ret
```

Alexis Englund, Zackery Holloway
CENG 347 Lab 3
02/13/25

Code (Alexis):

```
#define __SFR_OFFSET 0

#include "avr/io.h"

; let the C compiler know these are called elsewhere via external definition
.global start
.global blink
.global delay_n_ms

start:
   ; PORTB to output
   ldi r16, 0b11110000
   in r17, DDRB
   or r16, r17
   out DDRB, r16

   ; PORTB to a known state
   in r17, PORTB
   andi r17, 0x0F
   out PORTB, r17

   ret

blink:
   ; counter to 15
   ldi r18, 15

loop:
   ; load PORTB value
   in r19, PORTB
   andi r19, 0x0F

   ; shift counter to upper nybble and merge with lower nybble
   swap r18
   andi r18, 0xF0
   or r19, r18
   out PORTB, r19
   swap r18

   ; delay 500ms
   ldi r20, 250
   call delay_n_ms
```

Alexis Englund, Zackery Holloway
CENG 347 Lab 3
02/13/25

```
   ldi r20, 250
   call delay_n_ms

   ; decrement counter
   dec r18
   andi r18, 0x0F

   rjmp loop

delay_n_ms:
 ; Delay about r20*1ms. Destroys r20, r30, and r31.
 ; One millisecond is about 16000 cycles at 16MHz.
 ; The basic loop takes about 5 cycles, so we need about 3000 loops.
 ldi   r31, 3000>>8 ; high(3000)
 ldi   r30, 3000&255  ; low(3000)
delaylp:
   sbiw r30, 1
   brne delaylp
   subi r20, 1
   brne delay_n_ms
   ret
```

Results:

Part 1: Blink.S

The environment was set up and the LED was blinking. TA visually confirmed

Part 2: Binary Counter Implementation in Assembly

The code ran successfully and counted as expected. This was verified by the TA visually.

Conclusion

In conclusion, in this lab we were to take the setup from Lab 1 and recreate it using assembly instead of C. Part 1 was straightforward, as we just had to load and run the Blink.S file, which worked as expected. Part 2 required us to build on the part 1 code that was given to us to build the binary counter running with the correct delays and LED output. Overall, the lab helped show how to control hardware at a lower level and understand AVR assembly instructions. The successful results were confirmed by the TA in the lab, and the experience gave us a better understanding of how delays and bit manipulation work in embedded systems.