Students: Kevin (Kaiwen) Xue - 20814292, Anni Wang - 20849760

BME 393L: Lab 5

4.1 LED Blink

```
#define __SFR_OFFSET 0

#include "avr/io.h"

.global start
.global blink
;------------- Do NOT change anything above this line
start:
  ret

blink:
  ldi r16, 0xFF  ; Load register 16 with 0xFF (all bits 1)
  out DDRB, r16 ; Write 0xFF to Data Direction Register for port B. This defines
all pins on port B as output.
  ldi r16, 0x0  ; Load register 16 with 0x00 (all bits 0)
  out PORTB, r16 ; Write 0x00 to port B. This sets all pins to 0.
  sbi PORTB, 0  ; Sets bit 0 on port B to 1.
  call delay
  cbi PORTB, 0
  call delay
jmp blink

delay:
    ldi r20, 0xE9
    outer_loop:
      ldi r19, 0x0 ; Load r17 with zero
      middle_loop:
        ldi r18, 0x0 ; Load r18 with zero
        inner_loop:
          mul r17, r17 ; Takes 2 clock cycles to complete
          inc r18 ; Increment r18
          brne inner_loop ; If not zero, jump to loop
        inc r19
        brne middle_loop
      inc r20
      brne outer_loop
ret
```

Figure 1: ‘LED blink’ assembly code

Questions:

The 1 Hz frequency of the blinking LED was created by defining a subroutine that delays for 0.5 seconds. The subroutine has 3 levels of nested looping. The 2 inner loop indices are fixed, and the delay for one iteration of the outer loop is calculated by the following: (256 iterations of innermost loop* 4 clock cycles per iteration / 12 MHz) * 256 iterations of the middle loop = 0.0218453 seconds. Number of iterations needed for the outer loop to have a total delay of 0.5 seconds is 0.5 / 0.0218453 ≈ 23. So, 256 - 23 = 233 = 0xE9, which is the starting index of the outer loop.

4.2 Seven-segment counting

```
#define __SFR_OFFSET 0

#include "avr/io.h"

.global start
.global seven_segment_counting
;------------- Do NOT change anything above this line
start:
  ret

seven_segment_counting:
  ldi r16, 0xFF
  out DDRB, r16
  ldi r16, 0x00
  out PORTB, r16

  ldi r16, 0xFF
  out DDRD, r16
  ldi r16, 0x00
  out PORTD, r16

  ldi r17, 0x00
  loop:

    call delay

    cbi PORTD, 3
    cbi PORTD, 2
    cbi PORTD, 4
    cbi PORTD, 5
    cbi PORTB, 0
    cbi PORTB, 1
    cbi PORTB, 2

    andi r17, 0x0F

    send_F:
        cpi r17, 0x0F
        brne send_E
        sbi PORTD, 3
        sbi PORTD, 4
        sbi PORTD, 5
        sbi PORTB, 2
        inc r17
        jmp loop
```

```
send_E:
    cpi r17, 0x0E
    brne send_D
    sbi PORTD, 3
    sbi PORTD, 4
    sbi PORTD, 5
    sbi PORTB, 1
    sbi PORTB, 2
    inc r17
    jmp loop

send_D:
    cpi r17, 0x0D
    brne send_C
    sbi PORTD, 2
    sbi PORTD, 5
    sbi PORTB, 0
    sbi PORTB, 1
    sbi PORTB, 2
    inc r17
    jmp loop

send_C:
    cpi r17, 0x0C
    brne send_B
    sbi PORTD, 3
    sbi PORTD, 4
    sbi PORTB, 1
    sbi PORTB, 2
    inc r17
    jmp loop

send_B:
    cpi r17, 0x0B
    brne send_A
    sbi PORTD, 4
    sbi PORTD, 5
    sbi PORTB, 0
    sbi PORTB, 1
    sbi PORTB, 2
    inc r17
    jmp loop

send_A:
    cpi r17, 0x0A
    brne send_9
```

```
        sbi PORTD, 3
        sbi PORTD, 2
        sbi PORTD, 4
        sbi PORTD, 5
        sbi PORTB, 0
        sbi PORTB, 2
        inc r17
        jmp loop

    send_9:
        cpi r17, 0x09
        brne send_8
        sbi PORTD, 3
        sbi PORTD, 2
        sbi PORTD, 4
        sbi PORTD, 5
        sbi PORTB, 0
        sbi PORTB, 1
        inc r17
        jmp loop

    send_8:
        cpi r17, 0x08
        brne send_7
        sbi PORTD, 3
        sbi PORTD, 2
        sbi PORTD, 4
        sbi PORTD, 5
        sbi PORTB, 0
        sbi PORTB, 1
        sbi PORTB, 2
        inc r17
        jmp loop

    send_7:
        cpi r17, 0x07
        brne send_6
        sbi PORTD, 3
        sbi PORTD, 2
        sbi PORTB, 0
        inc r17
        jmp loop

    send_6:
        cpi r17, 0x06
        brne send_5
        sbi PORTD, 3
```

```
        sbi PORTD, 4
        sbi PORTD, 5
        sbi PORTB, 0
        sbi PORTB, 1
        sbi PORTB, 2
        inc r17
        jmp loop

    send_5:
        cpi r17, 0x05
        brne send_4
        sbi PORTD, 3
        sbi PORTD, 4
        sbi PORTD, 5
        sbi PORTB, 0
        sbi PORTB, 1
        inc r17
        jmp loop

    send_4:
        cpi r17, 0x04
        brne send_3
        sbi PORTD, 4
        sbi PORTD, 2
        sbi PORTD, 5
        sbi PORTB, 0
        inc r17
        jmp loop

    send_3:
        cpi r17, 0x03
        brne send_2
        sbi PORTD, 3
        sbi PORTD, 2
        sbi PORTD, 5
        sbi PORTB, 0
        sbi PORTB, 1
        inc r17
        jmp loop

    send_2:
        cpi r17, 0x02
        brne send_1
        sbi PORTD, 2
        sbi PORTD, 3
        sbi PORTD, 5
        sbi PORTB, 2
```

```
        sbi PORTB, 1
        inc r17
        jmp loop

    send_1:
        cpi r17, 0x01
        brne send_0
        sbi PORTB, 0
        sbi PORTD, 2
        inc r17
        jmp loop

    send_0:
        cpi r17, 0x00
        brne send_error
        sbi PORTB, 0
        sbi PORTB, 1
        sbi PORTB, 2
        sbi PORTD, 4
        sbi PORTD, 3
        sbi PORTD, 2
        inc r17
        jmp loop

    send_error:
        sbi PORTD, 3
        sbi PORTD, 5
        sbi PORTB, 1
        jmp loop

delay:
    ldi r20, 0xD2
    outer_loop:
      ldi r19, 0x0 ; Load r17 with zero
      middle_loop:
        ldi r18, 0x0 ; Load r18 with zero
        inner_loop:
          mul r17, r17 ; Takes 2 clock cycles to complete
          inc r18 ; Increment r18
          brne inner_loop ; If not zero, jump to loop
        inc r19
        brne middle_loop
      inc r20
      brne outer_loop
ret
```

Figure 2: ‘Seven-segment counting’ assembly code

Video of the ‘Seven-segment counting’ circuit:
https://drive.google.com/file/d/1ezPufk5z6QS5W4QHZleHNpcWRhtgm5z-/view?usp=sharing

Questions:
What I learned in this task which was not clear to me before was how to convert from relative addressing to absolute addressing. It was briefly mentioned in the slides, but a more concrete example could have helped.

4.3 Adding a GO command

```
#define __SFR_OFFSET 0

#include "avr/io.h"

.global start
.global seven_segment_counting_with_go
;------------- Do NOT change anything above this line
start:
  ret

seven_segment_counting_with_go:
  ldi r16, 0xFF
  out DDRB, r16
  ldi r16, 0x00
  out PORTB, r16

  ldi r16, 0xBF
  out DDRD, r16
  ldi r16, 0x40
  out PORTD, r16

  wait_for_button_press:
    cbi PORTD, 3
    cbi PORTD, 2
    cbi PORTD, 4
    cbi PORTD, 5
    cbi PORTB, 0
    cbi PORTB, 1
    cbi PORTB, 2
    in r16, PIND
    andi r16, 0x40

    brne wait_for_button_press
    loop:
      call delay

      cbi PORTD, 3
      cbi PORTD, 2
      cbi PORTD, 4
      cbi PORTD, 5
      cbi PORTB, 0
      cbi PORTB, 1
      cbi PORTB, 2
```

```
    andi r17, 0x0F

    send_F:
        cpi r17, 0x0F
        brne send_E
        sbi PORTD, 3
        sbi PORTD, 4
        sbi PORTD, 5
        sbi PORTB, 2
        inc r17
        call delay
        jmp wait_for_button_press

    send_E:
        cpi r17, 0x0E
        brne send_D
        sbi PORTD, 3
        sbi PORTD, 4
        sbi PORTD, 5
        sbi PORTB, 1
        sbi PORTB, 2
        inc r17
        jmp loop

    send_D:
        cpi r17, 0x0D
        brne send_C
        sbi PORTD, 2
        sbi PORTD, 5
        sbi PORTB, 0
        sbi PORTB, 1
        sbi PORTB, 2
        inc r17
        jmp loop

    send_C:
        cpi r17, 0x0C
        brne send_B
        sbi PORTD, 3
        sbi PORTD, 4
        sbi PORTB, 1
        sbi PORTB, 2
        inc r17
        jmp loop

    send_B:
        cpi r17, 0x0B
```

```
        brne send_A
        sbi PORTD, 4
        sbi PORTD, 5
        sbi PORTB, 0
        sbi PORTB, 1
        sbi PORTB, 2
        inc r17
        jmp loop

    send_A:
        cpi r17, 0x0A
        brne send_9
        sbi PORTD, 3
        sbi PORTD, 2
        sbi PORTD, 4
        sbi PORTD, 5
        sbi PORTB, 0
        sbi PORTB, 2
        inc r17
        jmp loop

    send_9:
        cpi r17, 0x09
        brne send_8
        sbi PORTD, 3
        sbi PORTD, 2
        sbi PORTD, 4
        sbi PORTD, 5
        sbi PORTB, 0
        sbi PORTB, 1
        inc r17
        jmp loop

    send_8:
        cpi r17, 0x08
        brne send_7
        sbi PORTD, 3
        sbi PORTD, 2
        sbi PORTD, 4
        sbi PORTD, 5
        sbi PORTB, 0
        sbi PORTB, 1
        sbi PORTB, 2
        inc r17
        jmp loop

    send_7:
```

```
        cpi r17, 0x07
        brne send_6
        sbi PORTD, 3
        sbi PORTD, 2
        sbi PORTB, 0
        inc r17
        jmp loop

    send_6:
        cpi r17, 0x06
        brne send_5
        sbi PORTD, 3
        sbi PORTD, 4
        sbi PORTD, 5
        sbi PORTB, 0
        sbi PORTB, 1
        sbi PORTB, 2
        inc r17
        jmp loop

    send_5:
        cpi r17, 0x05
        brne send_4
        sbi PORTD, 3
        sbi PORTD, 4
        sbi PORTD, 5
        sbi PORTB, 0
        sbi PORTB, 1
        inc r17
        jmp loop

    send_4:
        cpi r17, 0x04
        brne send_3
        sbi PORTD, 4
        sbi PORTD, 2
        sbi PORTD, 5
        sbi PORTB, 0
        inc r17
        jmp loop

    send_3:
        cpi r17, 0x03
        brne send_2
        sbi PORTD, 3
        sbi PORTD, 2
        sbi PORTD, 5
```

```
            sbi PORTB, 0
            sbi PORTB, 1
            inc r17
            jmp loop

        send_2:
            cpi r17, 0x02
            brne send_1
            sbi PORTD, 2
            sbi PORTD, 3
            sbi PORTD, 5
            sbi PORTB, 2
            sbi PORTB, 1
            inc r17
            jmp loop

        send_1:
            cpi r17, 0x01
            brne send_0
            sbi PORTB, 0
            sbi PORTD, 2
            inc r17
            jmp loop

        send_0:
            cpi r17, 0x00
            brne send_error
            sbi PORTB, 0
            sbi PORTB, 1
            sbi PORTB, 2
            sbi PORTD, 4
            sbi PORTD, 3
            sbi PORTD, 2
            inc r17
            jmp loop

        send_error:
            sbi PORTD, 3
            sbi PORTD, 5
            sbi PORTB, 1
            jmp loop


delay:
    ldi r20, 0xD2
    outer_loop:
      ldi r19, 0x0 ; Load r17 with zero
```

```
      middle_loop:
        ldi r18, 0x0 ; Load r18 with zero
        inner_loop:
          mul r17, r17 ; Takes 2 clock cycles to complete
          inc r18 ; Increment r18
          brne inner_loop ; If not zero, jump to loop
        inc r19
        brne middle_loop
      inc r20
      brne outer_loop
ret
```

Figure 3: 'Adding a Go command' assembly code

Video of the 'Adding a GO command' circuit:
https://drive.google.com/file/d/1G6XGeTp5wUZSdUc1z-uPoOZvsCbC9st2/view?usp=sharing

5. Post Lab task

```
#define __SFR_OFFSET 0

#include "avr/io.h"

.global start
.global eight_bit_multiplier
;------------- Do NOT change anything above this line
start:
  ret

eight_bit_multiplier:
  ldi r16, 0xFF
  out DDRB, r16
  ldi r16, 0x00
  out PORTB, r16

  ldi r16, 0x3F
  out DDRD, r16
  ldi r16, 0xC0
  out PORTD, r16


  wait_for_initialize_first_value_LSB:
    call display_q_symbol
    in r16, PIND
    andi r16, 0x80
    brne wait_for_initialize_first_value_LSB
    call delay_02
    call display_blank
    ldi r18, 0x00

    first_value_LSB_check_if_increment_pressed:
        call display_r18
        in r16, PIND
        andi r16, 0x80
        brne first_value_LSB_check_if_save_pressed
        call delay_02
        call display_blank
        inc r18
        jmp first_value_LSB_check_if_increment_pressed

    first_value_LSB_check_if_save_pressed:
        in r16, PIND
        andi r16, 0x40
        brne first_value_LSB_check_if_increment_pressed
```

```
        call flash_r18
        mov r19, r18
        call display_blank
        call clear_r18
        jmp wait_for_initialize_first_value_MSB



wait_for_initialize_first_value_MSB:
  call display_q_symbol
  in r16, PIND
  andi r16, 0x80
  brne wait_for_initialize_first_value_MSB
  call delay_02
  call display_blank
  ldi r18, 0x00

  first_value_MSB_check_if_increment_pressed:
      call display_r18
      in r16, PIND
      andi r16, 0x80
      brne first_value_MSB_check_if_save_pressed
      call delay_02
      call display_blank
      inc r18
      jmp first_value_MSB_check_if_increment_pressed

  first_value_MSB_check_if_save_pressed:
      in r16, PIND
      andi r16, 0x40
      brne first_value_MSB_check_if_increment_pressed
      call flash_r18
      mov r20, r18
      call display_blank
      call clear_r18
      jmp wait_for_initialize_second_value_LSB

wait_for_initialize_second_value_LSB:
  call display_q_symbol
  in r16, PIND
  andi r16, 0x80
  brne wait_for_initialize_second_value_LSB
  call delay_02
  call display_blank
  ldi r18, 0x00
```

```
    second_value_LSB_check_if_increment_pressed:
        call display_r18
        in r16, PIND
        andi r16, 0x80
        brne second_value_LSB_check_if_save_pressed
        call delay_02
        call display_blank
        inc r18
        jmp second_value_LSB_check_if_increment_pressed

    second_value_LSB_check_if_save_pressed:
        in r16, PIND
        andi r16, 0x40
        brne second_value_LSB_check_if_increment_pressed
        call flash_r18
        mov r21, r18
        call display_blank
        call clear_r18
        jmp wait_for_initialize_second_value_MSB

wait_for_initialize_second_value_MSB:
    call display_q_symbol
    in r16, PIND
    andi r16, 0x80
    brne wait_for_initialize_second_value_MSB
    call delay_02
    call display_blank
    ldi r18, 0x00

    second_value_MSB_check_if_increment_pressed:
        call display_r18
        in r16, PIND
        andi r16, 0x80
        brne second_value_MSB_check_if_save_pressed
        call delay_02
        call display_blank
        inc r18
        jmp second_value_MSB_check_if_increment_pressed

    second_value_MSB_check_if_save_pressed:
        in r16, PIND
        andi r16, 0x40
        brne second_value_MSB_check_if_increment_pressed
        call flash_r18
        mov r22, r18
        call display_blank
        call clear_r18
```

```
swap r20
swap r22
or r19, r20
or r21, r22
mul r19, r21

mov r19, r0
mov r20, r0
mov r21, r1
mov r22, r1

andi r19, 0x0F
swap r20

andi r21, 0x0F
swap r22

display_output:
  mov r18, r19
  call display_r18
  call delay_05
  call delay_05
  call display_blank
  call delay_05

  mov r18, r20
  call display_r18
  call delay_05
  call delay_05
  call display_blank
  call delay_05

  mov r18, r21
  call display_r18
  call delay_05
  call delay_05
  call display_blank
  call delay_05

  mov r18, r22
  call display_r18
  call delay_05
  call delay_05
  call display_blank
  call delay_05
```

```
    call display_dash
    call delay_05
    call delay_05

    jmp display_output

display_dash:
    sbi PORTD, 5
    ret

flash_r18:
    call display_blank
    call delay_05
    call display_r18
    call delay_05
    call display_blank
    call delay_05
    call display_r18
    call delay_05
    ret

delay_05:
    push r20
    push r19
    push r18

    ldi r20, 0xE9
    delay_05_outer_loop:
      ldi r19, 0x0 ; Load r17 with zero
      delay_05_middle_loop:
        ldi r18, 0x0 ; Load r18 with zero
        delay_05_inner_loop:
          mul r17, r17 ; Takes 2 clock cycles to complete
          inc r18 ; Increment r18
          brne delay_05_inner_loop ; If not zero, jump to loop
        inc r19
        brne delay_05_middle_loop
      inc r20
      brne delay_05_outer_loop

    pop r18
    pop r19
    pop r20
    ret

delay_02:
    push r20
```

```
    push r19
    push r18

    ldi r20, 0xF6
    delay_02_outer_loop:
      ldi r19, 0x0 ; Load r17 with zero
      delay_02_middle_loop:
        ldi r18, 0x0 ; Load r18 with zero
        delay_02_inner_loop:
          mul r17, r17 ; Takes 2 clock cycles to complete
          inc r18 ; Increment r18
          brne delay_02_inner_loop ; If not zero, jump to loop
        inc r19
        brne delay_02_middle_loop
      inc r20
      brne delay_02_outer_loop

    pop r18
    pop r19
    pop r20
    ret

clear_r18:
    sbr r18, 0x00
    ret

display_q_symbol:
    sbi PORTD, 3
    sbi PORTD, 2
    sbi PORTD, 5
    sbi PORTB, 2
ret

display_blank:
    cbi PORTD, 3
    cbi PORTD, 2
    cbi PORTD, 4
    cbi PORTD, 5
    cbi PORTB, 0
    cbi PORTB, 1
    cbi PORTB, 2
ret

display_r18:

      andi r18, 0x0F
```

```
send_F:
    cpi r18, 0x0F
    brne send_E
    sbi PORTD, 3
    sbi PORTD, 4
    sbi PORTD, 5
    sbi PORTB, 2
    ret

send_E:
    cpi r18, 0x0E
    brne send_D
    sbi PORTD, 3
    sbi PORTD, 4
    sbi PORTD, 5
    sbi PORTB, 1
    sbi PORTB, 2
    ret

send_D:
    cpi r18, 0x0D
    brne send_C
    sbi PORTD, 2
    sbi PORTD, 5
    sbi PORTB, 0
    sbi PORTB, 1
    sbi PORTB, 2
    ret

send_C:
    cpi r18, 0x0C
    brne send_B
    sbi PORTD, 3
    sbi PORTD, 4
    sbi PORTB, 1
    sbi PORTB, 2
    ret

send_B:
    cpi r18, 0x0B
    brne send_A
    sbi PORTD, 4
    sbi PORTD, 5
    sbi PORTB, 0
    sbi PORTB, 1
    sbi PORTB, 2
    ret
```

```
send_A:
    cpi r18, 0x0A
    brne send_9
    sbi PORTD, 3
    sbi PORTD, 2
    sbi PORTD, 4
    sbi PORTD, 5
    sbi PORTB, 0
    sbi PORTB, 2
    ret

send_9:
    cpi r18, 0x09
    brne send_8
    sbi PORTD, 3
    sbi PORTD, 2
    sbi PORTD, 4
    sbi PORTD, 5
    sbi PORTB, 0
    sbi PORTB, 1
    ret

send_8:
    cpi r18, 0x08
    brne send_7
    sbi PORTD, 3
    sbi PORTD, 2
    sbi PORTD, 4
    sbi PORTD, 5
    sbi PORTB, 0
    sbi PORTB, 1
    sbi PORTB, 2
    ret

send_7:
    cpi r18, 0x07
    brne send_6
    sbi PORTD, 3
    sbi PORTD, 2
    sbi PORTB, 0
    ret

send_6:
    cpi r18, 0x06
    brne send_5
    sbi PORTD, 3
```

```
        sbi PORTD, 4
        sbi PORTD, 5
        sbi PORTB, 0
        sbi PORTB, 1
        sbi PORTB, 2
        ret

    send_5:
        cpi r18, 0x05
        brne send_4
        sbi PORTD, 3
        sbi PORTD, 4
        sbi PORTD, 5
        sbi PORTB, 0
        sbi PORTB, 1
        ret

    send_4:
        cpi r18, 0x04
        brne send_3
        sbi PORTD, 4
        sbi PORTD, 2
        sbi PORTD, 5
        sbi PORTB, 0
        ret

    send_3:
        cpi r18, 0x03
        brne send_2
        sbi PORTD, 3
        sbi PORTD, 2
        sbi PORTD, 5
        sbi PORTB, 0
        sbi PORTB, 1
        ret

    send_2:
        cpi r18, 0x02
        brne send_1
        sbi PORTD, 2
        sbi PORTD, 3
        sbi PORTD, 5
        sbi PORTB, 2
        sbi PORTB, 1
        ret

    send_1:
```

```
        cpi r18, 0x01
        brne send_0
        sbi PORTB, 0
        sbi PORTD, 2
        ret

    send_0:
        cpi r18, 0x00
        brne send_error
        sbi PORTB, 0
        sbi PORTB, 1
        sbi PORTB, 2
        sbi PORTD, 4
        sbi PORTD, 3
        sbi PORTD, 2
        ret

    send_error:
        sbi PORTD, 3
        sbi PORTD, 5
        sbi PORTB, 1
        ret
```

Figure 4: Post lab assembly code

Table 5.1 - Wire connections for post-lab

Segment/Button	A	B	C	D	E	F	G	UP	ENTER
Arduino Uno Pin #	3	2	8	9	10	4	5	7	6

Video of the post lab circuit:
https://drive.google.com/file/d/1uo0D4xdwWohqmUbHUkdYvFXFgv_a09eX/view?usp=sharing