Khari Ollivierre
Lab 2
Section 1490

Problems Encountered

My DAD board is at my pareant's house currently, so I was unable to complete some parts of the lab. I also was unable to program my board. According to the emulation demo, I am to use an "Atmel ICE" to program the board, but I do not have one.

Applications

This lab was a great experience in working with computer hardware at its most basic level. I learned how basic input and output work with this board, which is fundamental to all digital operations.

Pseudocode

```
BEGIN lab2b
   while(true)
      reset(switch_ports)
      switch_ports = read(switches)
      leds.set(switch_ports)
   end while
END lab2b


BEGIN delay10ms
   count = 0;
   delay = false;
   IF xmega clock asserted
   THEN
      IF count == 10000
      THEN
         count = 0;
         delay = true;
```

```
        ELSE
            count++;
            delay = false;
        ENDIF
    ENDIF
    RETURN delay
END delay10ms

BEGIN lab2c
    count = 0;
    output = false;
    WHILE true
        timer = delay10ms();
        IF timer == true
        THEN
            IF count == 19
            THEN
                count = 0;
            ELSE
                count++;
            ENDIF
            IF count < 10
            THEN
                output = true;
            ELSE
                output = false;
            ENDIF
        ENDIF
    END WHILE true
```

```
END lab2c


BEGIN lab2d
    LET state0 = "10000001";
    LET state1 = "01000010";
    LET state2 = "00100100";
    LET state3 = "00011000";
    LET state4 = "00000000";
    LET states = {state0, state1, state2, state3, state2, state1, state0, state4};
    LET c_state = state4;
    LET S2 = the port for S2;
    LET red = RGB(255, 0, 0);
    LET green = RGB(0, 255, 0);
    LET res = port D;
    LET clock = delayed board clock;
    LET count = 0;
    WHILE true
        IF clock && !S2;
        THEN
            IF count == 7;
            THEN
                count = 0;
            ELSE
                count++;
            ENDIF
        ENDIF
        c_state = states[count];
        IF S2
        THEN
```

```
        IF c_state = state3;
        THEN
            res = green;
        ELSE
            res = red;
        ENDIF
      ENDIF
   END WHILE true
END lab2d
```

Program Code

```
/****
 * lab2b
 * Name: Khari Ollivierre
 * Section: 1490
 * TA: Samantha Soto
 * Description: Switch control for LEDs
****/

.include "ATxmega128A1Udef.inc"

.equ ONES = 0xFF
.equ ZERO = ~ONES

.ORG 0x0000
        rjmp MAIN

.ORG 0x0100
```

```
MAIN:
        ldi R16, ONES
        sts PORTA_DIRCLR, R16    ; Configures Port A to receive switch input
        sts PORTC_DIRSET, R16    ; Configures Port C to output to LEDs
        sts PORTC_OUTSET, R16    ; Turns off LEDs (active-low)
        sts PORTA_IN, PORTC_OUT ; Sets the output to the switch state
        rjmp MAIN               ; Loops forever

/****
 * lab2c
 * Name: Khari Ollivierre
 * Section: 1490
 * TA: Samantha Soto
 * Description: Clock Delay subroutines
****/

.include "ATxmega128A1Udef.inc"

.equ LIMIT = 0xFF
.equ DELAY = 0x13
.equ ZERO = 0x00
.equ OFFSET = 0x10
.equ CUSTOM = 0xFF

.ORG 0x0000
        rjmp MAIN

.dseg
.ORG 0x0100
```

```
        TMP:  .byte 1


.cseg
MAIN:
        ldi     R18, ZERO           ; Zero registers for counting
        ldi     R19, ZERO
        ldi     R20, LIMIT
        ldi     R21, ZERO           ; Register for delay extension multiplier counter
        ldi     R22, OFFSET         ; Multiplier value stored in R20.
        ldi     R23, CUSTOM                 ; Custom delay value set to max
        sts     PORTC_DIRSET, R13           ; Allow Port D to output the delayed clock
        sts     PORTC_OUTCLR, R13


DELAY10MS:
DLOOP:
        inc     R18             ; Increment counter register
        cpi     R18, LIMIT      ; Counter check
        brne    DLOOP                   ; This loops 256 times, 3*256 = 768 instructions per loop
                                        ; It takes the CPU 10 ms to execute 20000 instructions,
                                        ; and by the start of the loop, it has executed 2 instructions, so
                                        ; 19998 instructions are left to be executed.
                                        ;
                                        ; 19998/3 = 6666, so the loop would have to execute 6666 times to
delay appropriately.
                                        ; Since the registers only hold 2 bytes each, multiple registers must
track the loop
                                        ; execution.
        inc     R19             ; increments the 2nd counter
        cpi     R19, DELAY      ; counter check
```

```
        breq    LOOP            ; Each increment will execute 771 instructions. 19998/771 ~= 26, so this entire loop
                                ; should execute 26 times. The end result will be marginally identical to 10 ms
        ldi     R18, ZERO
        rjmp    DLOOP

LOOP:
        inc     R21
        cp      R21, R22
        breq    END
        ldi     R18, ZERO
        ldi     R19, ZERO
        rjmp    DLOOP
END:
        sts     PORTC_OUTTGL, R18       ; Toggle the output like a clock
        ldi     R18, ZERO
        ldi     R19, ZERO
        ldi     R21, ZERO
        rjmp    DLOOP

DELAYX10MS:
        sts     TMP, R23
        lds     R22, TMP
        rjmp    DLOOP
/****
 * lab2d
 * Name: Khari Ollivierre
 * Section: 1490
```

```
 * TA: Samantha Soto
 * Description: LED game
****/

.include "ATxmega128A1Udef.inc"

.equ STATE0  = 0x81
.equ STATE1  = 0x42
.equ STATE2  = 0x24
.equ STATE3  = 0x18
.equ ZERO    = 0x00
.equ ONES    = ~ZERO
.equ TABLE_S         = 8
.equ RED     = 0xEF
.equ GREEN   = 0xDF

.equ LIMIT = 0xFF
.equ DELAY = 0x13
.equ REPS = 0x0A

.def C_STATE         = R16
.def TMP     = R17

.ORG 0x0000
        rjmp MAIN

.ORG 0x0100
        STATES: .db ~STATE0, ~STATE1, ~STATE2, ~STATE3, ~STATE2, ~STATE1, ~STATE0,
~ZERO
```

```
MAIN:
        ldi     ZL, low(STATES << 1)        ; Loads table location into Z
        ldi     ZH, high(STATES << 1)
        ldi     TMP, ONES           ; Set Port C and D to outputs
        sts     PORTC_DIRSET, TMP
        sts     PORTD_DIRSET, TMP
        sts     PORTF_DIRCLR, TMP       ; Set Port F to input
        clr     TMP

LOOP:
        lpm     C_STATE, Z+         ; Loads the current state and increments
        sts     PORTC_OUT, C_STATE      ; Sets output to state
        rjmp    DELAYX10MS

RE:
        cpi     C_STATE, ~ZERO          ; Checks if current state is zero, resets Z if so
        breq    RESET
        rjmp    LOOP

RESET:
        ldi     ZL, low(STATES << 1)
        ldi     ZH, high(STATES << 1)
        rjmp    LOOP

CHECK:
        cpi     C_STATE, ~STATE3
        breq    WIN
```

```
LOSS:
        ldi     TMP, RED      ; Set LED to red
        sts     PORTD_OUT, TMP
        lds     TMP, PORTF_IN
        sbrs    TMP, 2        ; If S1 is pressed, reset game
        rjmp    RESET
        clr     TMP
        rjmp    LOSS

WIN:
        ldi     TMP, GREEN    ; Set LED to green
        sts     PORTD_OUT, TMP
        lds     TMP, PORTF_IN
        sbrs    TMP, 2        ; If S1 is pressed, reset game
        rjmp    RESET
        clr     TMP
        rjmp    WIN

DELAYX10MS:
        ldi     R18, ZERO     ; Zero registers for counting
        ldi     R19, ZERO
        ldi     R20, LIMIT
        ldi     R21, ZERO     ; Register for delay extension multiplier counter
        ldi     R22, REPS     ; Multiplier value stored in R22

DLOOP:
        inc     R18           ; Increment counter register
        lds     TMP, PORTF_IN
        sbrs    TMP, 3        ; If button is pressed, check win conditions
```

```
        rjmp    CHECK
        cpi     R18, LIMIT    ; Counter check
        brne    DLOOP         ; This loops 256 times, 3*256 = 768 instructions per loop
                              ; It takes the CPU 10 ms to execute 20000 instructions,
                              ; and by the start of the loop, it has executed 2 instructions, so
                              ; 19998 instructions are left to be executed.
                              ;
                              ; 19998/3 = 6666, so the loop would have to execute 6666 times to delay appropriately.
                              ; Since the registers only hold 2 bytes each, multiple registers must track the loop
                              ; execution.
        inc     R19           ; increments the 2nd counter
        lds     TMP, PORTF_IN
        sbrs    TMP, 3        ; If button is pressed, check win conditions
        rjmp    CHECK
        cpi     R19, DELAY    ; counter check
        breq    XLOOP         ; Each increment will execute 771 instructions. 19998/771 ~= 26, so this entire loop
                              ; should execute 26 times. The end result will be marginally identical to 10 ms
        ldi     R18, ZERO
        rjmp    DLOOP

XLOOP:
        inc     R21
        lds     TMP, PORTF_IN
        sbrs    TMP, 3        ; If button is pressed, check win conditions
        rjmp    CHECK
        cp      R21, R22      ; Jump to end if delay is complete
```

```
        breq    END
        ldi     R18, ZERO
        ldi     R19, ZERO
        rjmp    DLOOP
END:
        ldi     R18, ZERO
        ldi     R19, ZERO
        ldi     R21, ZERO
        rjmp    RE              ; Return to game code
```

Appendix

LED design