Josh Martin
ECE 362
Pre-Lab #5

Introduction:

In this lab, we were continuing to build our skill with assembly and applying what we have learned in the past to more applicable things with the board. The first thing we did was use logical operators. The second thing we did was use bset and bclr and understand how that was working, while also using a delay. And the last thing we did was use loops to make the stepper spin in different directions and apply changing conditions to it.

Lab 5.1.1:

Objective/Purpose:
The goal of this program was to learn how to use logical operators and incorporate that idea into our code.
Expected Results:
Change EC to CC.

Code:

```
; variable/data se  ction
MY_EXTENDED_RAM: SECTION
; Insert here your data definition.

Var_1       ds.b 1
MY_EXTENDED_ROM:
port_t    equ $240 ; initialize all the things I need
ddr_s     equ $24A
port_s    equ $248
; code section
MyCode:     SECTION
main:
_Startup:
Entry:
    movb #$FF,ddr_s ; start by clearing everything in the led
    movb #$EC,Var_1 ; load the first value wich is $EC
    LDAA Var_1; load value of Var_1 into accumulator a
    ANDA #%11011111 ; clears the sixth bit while keeping it in Accumulator A
    STAA Var_1; stores in var_a
    ANDA #%11110111 ; this will set the forth bit in and it is still stored in var_1
    STAA Var_1; stores in var_a
Redo:LDAB port_t ; this checks to see if bit 1 Is high and if its it will continue in the loop. IF it is Is low, it will
break and store the value in Port_s
    andb #$04
    BNE Redo
    staa port_s
    nop
```

Lab 5.1.2:

Objective/Purpose:

The goal of this part was to pretty much create the same program from 5.1.1 but using bset and bclr. We learned how that worked and why it worked.

Expected Results:

Using bset we created a program that waits for the second switch to go low and then display the lights.

Code:

```
        Var_1       ds.b 1
        MY_EXTENDED_ROM:
        port_t    equ $240
        ddr_s     equ $24A
        port_s    equ $248
        ; code section
        MyCode:     SECTION
        main:
        _Startup:
        Entry:
                bset  ddr_s, #$FF  ; used to set port s. Intializing the value
                bclr port_s, #$FF  ; used to clr all the values in port s ($248)

        loop1:      brclr port_t, #%00000010, loop1;this waits till the second bit goes high and then it exits the loop
        loop2:      brset port_t, #%00000010, loop2; waits till the bit goes low and then it moves
                bset  port_s, #%00001100  ; sets all the leds to empty.
                nop
```

Lab 5.2:

Objective/Purpose:

The goal of this lab was to have the lights display one value and if we pressed the switch again we would display the same value shifted over by 4 bits. Overall we practiced multiple different things like creating loops and using subroutines. We also used a flag to design the checking system.

Expected Results:

The lights should display values back and forth.

Code:

```
; variable/data section
MY_EXTENDED_RAM: SECTION
; Insert here your data definition.
; code section
MY_EXTENDED_RAM: SECTION
; Insert here your data definition.
var_a ds.b 1
var_b ds.b 1
var_c ds.b 1
```

```
MY_EXTENDED_ROM: SECTION
port_t    equ $240
ddr_s     equ $24A
port_s    equ $248
port_u    equ $268
ddr_port_u equ $26A
psr_port_u equ $26D
pde_port_u equ $26C
SEQ:   dc.b $70,$B0,$D0, $E0
var1: dc.b $EB, $77, $7B, $7D, $B7, $BB, $BD, $D7, $DB, $DD, $E7, $ED, $7E, $BE, $DE,$EE, $00
; code section
MyCode:     SECTION
main:
_Startup:
Entry:
;first bullet point
        ; this section is setting the value $F0 to the DDR, PSR, and PDE



begin:      lds  #__SEG_END_SSTACK ;used to intiliaze the stack
            bset ddr_s, #$FF  ; used to intiliaze the LED display
            bset ddr_port_u, #$F0; used to intiliaze the hex keys
            bset psr_port_u, #$F0
            bset pde_port_u, #$0F
            bclr var_a, #$FF  ; intiliaze the variables
            bclr var_b, #$FF
            bclr var_c, #$FF

looop2:     jsr         looop1   ; subroutine for my loop.
            brclr       var_a, #$FF, move;
            bclr        var_b, #$F0
            lslb
            lslb
            lslb
            lslb
            orab        var_b
            stab        port_s
            stab        var_b
            bclr        var_a,#$FF
            bra         looop2

move:       bclr        var_b,#$0F
            orab        var_b
            stab        port_s
            stab        var_b
            bset        var_a,#$FF
            bra        looop2;

Delay:     PSHX
            LDX #1000
DLoop:     DEX
            BNE DLoop
            PULX
            RTS

looop1:    ldx      #SEQ      ; load the sequence in to x
next:      ldaa     1,x+      ; load one, and incrament it by x
            beq      looop1    ; if equal then branch
            staa     port_u    ; display the value
            jsr      Delay     ; my delay counter
            ldaa     port_u    ; load value from port_u in to a
            staa     var_c     ; store that value in var_c
            anda     #$0F      ; checks if the button is pressed or not; uses logical anda to make check if
is on or not
            cmpa     #$0F      ; compares whats in a to this memory value
            beq      next      ; if they are not eq

up:        ldaa       port_u    ;load port_u
            anda       #$0F      ; use logical and operator to check if it is pushed or not
            cmpa       #$0F      ; compare to the same value.
            bne        up        ; branch up if equal
                                 ; look up table
            ldy        #var1;    ; load the look up table with the table
            ldab       #0        ; set up incrament
redo:      ldaa       1, y+     ; incrament after each time through
            beq        looop1;   ; if equal it loops up
            incb                 ; if not equal incrament b.
            cmpa       var_c;    ; compare if it is or isnt
            bne        redo      ; if not equal you redo
            decb                 ; decrement to set b back to orginial value
            rts
```

Lab 5.3:

Objective/Purpose:

Create a sort of menu for the stepper motor. This one was interesting. I used for loops to go through and check values and to output the required outcome.

Expected Results:

When switch 0 and 1 were on or off the stepper motor should not work. When switch 2 was high it would spin fast, when low it would spin slowly. When 1 is high it would go counter clock wise. When 0 was high it would spin clockwise.

Code:

```
; variable/data section
MY_EXTENDED_RAM: SECTION
; Insert here your data definition.
val ds.b 1
highorlow ds.w 2
DelayCount ds.w 1
MY_CONSTANT_RAM: SECTION
port_t: equ $240
port_p: equ $258
port_p_ddr: equ $25A
vals: dc.b $0A, $12, $14, $0C, $0
vals1: dc.b $0C, $14, $12, $0A, $0



; code section
MyCode:     SECTION
main:
_Startup:
Entry:
top:      movb    #%00011110, port_p_ddr  ; intialize the motor
          LDS     #__SEG_END_SSTACK   ; intiliaze the stack
          ldaa    port_t          ; load port-t into a
          anda    #%00000100      ; clear out all the other values except the 3 bit
          cmpa    #$4             ; compare to 4
          beq     higher;         ; if it is equal it branches to the higher loop
          bne     lowest;         ; if it is not equal it branches to the lowest loop

higher:   movw    #9000, highorlow ; since higher changes the delay counter to a lower value to
spin faster
          bra     skip    ; skips over the lower branch
lowest:   movw    #30000, highorlow ; since lower changes delay counter to higher value so it
moves slower
          bra     skip ; not necessary, i just like it there to make me feel better

skip:     ldaa    port_t ; load port_t into a again
          anda    #%00000011; clear everything except for the first 2 bits
          cmpa    #$3       ; compares to 3 if it is equal to 3 it skips
          beq     nope
          cmpa    #$0       ; compares to 0 if it is equal to 0 it will skip
          beq     nope
          cmpa    #$2       ; if equal to 2 it will send vals in to x
          beq     clock
          cmpa    #$1       ; if equal to 1 it will send vals1 into x
          beq     counter
```

```
;alternate from clock wise to counter clock wise

clock:     LDX     #vals       ;clock wise
           Bra     again1
counter:   LDX     #vals1      ; counter clock wise
           BRA     again1
again1:
again:    LDAA    1,x+
          STAA    port_p
          cmpa    $0
          beq     top          ; goes back to top to check if it has changed
          jsr     Delay
          bra     again

nope:     bra     top

Delay:    PSHX                   ; my delay
          LDX   highorlow         ; depends on if bit 3 is high or low
LOOP      DEX
          BNE LOOP
          PULX
          RTS
```

Conclusion:

The conclusion should consist of 2 parts:
We built our understanding of flags and conditions and how they are implemented and how to use them. We also learned on how to apply logical operators to isolate an LED which is more applicable than just being able to do it.

The hard part about this lab was setting up the switch. I had a lot of help from a lot of people setting that up and using operands that I do not know well,

Note:

- Pay attention to grammatical and spelling errors
- Use your own words (don't copy the slides)
- Single spaced
- Code should be commented (useful and meaningful comments)
- Fonts and sizes:
    - Use the "Times New Roman" font or any similar font
    - Use font size 14 bold for headings
    - Use font size 12 for subheadings
    - Use font size 12 for text
    - Use the "Courier New" font for the code and the size should be 10