Stephen Corya
ECE 362
Post-Lab #3

Introduction:
The purpose of this lab was utilizing the HC(S)12 board to perform arithmetic operations, output to hardware devices, and explore the branching instructions available in assembly code.

Lab 3.1.1:
Objective/Purpose:
The objective of this first experiment is solving a linear equation in slope-intercept form.

Explanation:
The code defines a byte which we will change in the debugger as ‘Val.’ This byte represents ‘x’ in the slope-intercept equation, y=mx+b. The values of ‘m’ and ‘b’ are defined as contsants. The factor by which to scale the product of ‘x’ and ‘m’ is also defined as a constant. The code finds the product of ‘x’ and ‘m’, divides by the scale, and adds the constant y-intercept, ‘b’.

Results:
The code produced the expected result, rounded down to the nearest integer. This is to be expected, as decimal values were not taken into account.

Code:
```
      XDEF  Entry
      XREF  __SEG_END_SSTACK

MY_CONSTANT_ROM: SECTION ; constant/definition section
Val:        ds.b  1
Slope:      dc.b  $44
Intercept:  dc.b  $0C
Scale:      dc.b  $64
Out:        ds.w  1

; code section
MyCode:     SECTION
main:
_Startup:
Entry:
  LDAA      Val         ;load the value ‘x’ to be defined in the debugger into acc. A
  LDAB      Slope       ;load the provided constant slope ‘m’ into acc. B
  MUL                   ;multiply the value ‘x’ times the constant ‘m’
  LDX       Scale       ;load the value to scale the slop ‘m’ into reg. x
  DIV                   ;divide the product of ‘m’ and ‘b’ by the scale
  ADDD      Intercept   ;add the constant y-intercept to the dividend
  STD       Out         ;store the output of accumulator D
```

Lab 3.1.2:
Objective/Purpose:
The purpose of this experiment was similar to experiment 3.1.1., except with a parabolic equation, opposed to the previous linear equation. That is, the value of ‘x’ will be squared prior to performing other requisite arithmetic.

Expected Results:
As in the previous experiment, the code should solve the provided equation (y=x^2*m+b) using various values of ‘x’.

Code:
```
      XDEF  Entry
```

```
      XREF  __SEG_END_SSTACK

; variable/data w
MY_EXTENDED_RAM: SECTION
Val:  ds.b  1
Out:  ds.w  1

MY_CONSTANT_ROM: SECTION ; constant/definition section
Slope:      dc.b  $44
Intercept:  dc.b  $0C
Scale:      dc.b  $64

; code section
MyCode:     SECTION
main:
_Startup:
Entry:
  LDAB      Val         ;load the value ‘x’ into accumulator B
  LDAA      Val         ;load the same value ‘x’ into accumulator A
  MUL                   ;multiple acc. B and acc. A, squaring ‘x’
  LDY       Slope       ;load the provided slope ‘m’ into register Y
  EMUL                  ;multiply the product of ‘x’ times ‘x’ with the slope ‘m’
  LDX       Scale       ;load the scale factor into register X
  EDIV                  ;divide the product of x^2*m by the scale factor
  tfr       y,d         ;transfer the dividend in register Y to accumulator D
  ADDD      Intercept   ;add the y-intercept ‘b’ to the dividend in acc. D
  STD       Out         ;store the result in the word Out
```

Lab 3.2:

Objective/Purpose:

The objective of this experiment was exploration of how various arithmetic operations performed within accumulator A affected branch conditions and ultimately the value in accumulator A at the end of the code modules.

Expected Results:

Depending on the code provided, the value of A will change, depending on the condition code register values and branch instructions.

Code:

```
      XDEF  Entry
      XREF  __SEG_END_SSTACK
      MyConstant:       section
      num_1:      dc.b  $40
      num_2:      db.b  $50
      MyCode:           section
      Entry:      ldaa  num_1
                  adda  num_2
                  nop
;---------------------------------------
      XDEF  Entry
      XREF  __SEG_END_SSTACK
                  ldaa  #$D3
                  adda  #$F2
                  bvs   done
                  ldaa  #0
            done: nop
;---------------------------------------
      XDEF  Entry
      XREF  __SEG_END_SSTACK
                  ldaa  #$D3
                  adda  #$F2
                  bcs   done
```

```
                ldaa  #0
          done: nop
;---------------------------------------
      XDEF  Entry
      XREF  __SEG_END_SSTACK
                ldaa  #$D3
                adda  #$F2
                bvs   done
                ldaa  #0
          done: nop
;---------------------------------------
      XDEF  Entry
      XREF  __SEG_END_SSTACK
                ldaa  #$41
                adda  #$5A
                bvs   done
                ldaa  #0
          done: nop
```

Lab 3.3:

Objective/Purpose:

The purpose of this experiment was outputting a sequence of values to a port, connected to a motor, and cause the motor to spin. The data direction register for port P was to be set appropriately.

Expected Results:

The motor, upon being provided different values on its port, should spin clockwise until the code is manually terminated.

Code:

```
      XDEF  Entry
      XREF  __SEG_END_SSTACK

; variable/data w
MY_EXTENDED_RAM: SECTION
ARR         dc.b  $0A,$12,$14,$0C,$0
port_p      equ   $258
ddrp        equ   $25A

MY_CONSTANT_ROM: SECTION ; constant/definition section

; code section
MyCode:     SECTION
main:
_Startup:
Entry:
        ldaa      #$1e        ;load the proper direction modes to acc. A
        staa      ddrp        ;push the direction modes into port P DDR
loop:   ldx       #ARR        ;load the start of sequence of bytes to spin the motor into
                              ;register X
loop1:
        ldaa      1,x+        ;load into A the first value of the array
        cmpa      #0          ;check that we are not at the end of the array
        beq       loop        ;if we are at the end of the array, reload the first array
                              ;value
        staa      port_p      ;push the value in accumulator A to port P, spinning the
                              ;motor one quarter turn
        ldy       #30000      ;load into register Y the amount by which we will delay
                              ;loop execution
delay1:
        dey                   ;decrement our delay value by 1
        bne       delay1      ;if we have not decreased our delay value to zero, continue
```

```
                        ;decrementing our array value
        bra     loop1   ;repeat the code which turns the motor
```

Conclusion:

Using the concepts learned in this lab, we can perform arithmetic operations involving both variable values and constant values. We can also traverse arrays, and implement loops which will perform useful functions, including delaying certain other instructions.

I struggled traversing the array in Experiment 3.3. I found it useful to add a value equal to ‘0’ at the end of the array.