ECE 372: Microproc & Embedded Sys Lab

Lab 3: Subroutines and Input/Output

June 21st, 2018

Summer 2018

Honor Code: I have neither given nor received unauthorized assistance on this graded report.

x Brian Vera-Burgos x Srinivas Simhan

Table of Contents

Objective

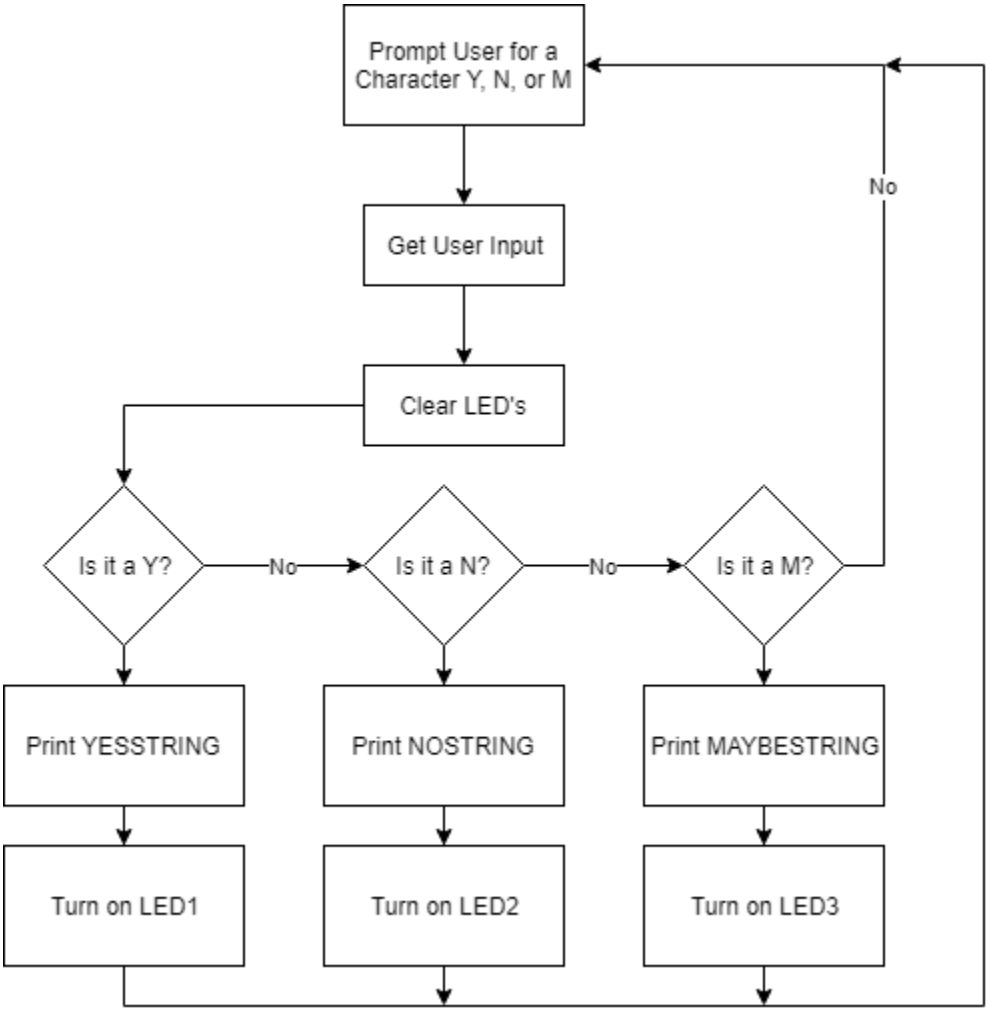
- To become familiar with CodeWarrior
- To become familiar with subroutines and input/output
- To understand and create flowcharts to be used in explanations in terms of documentation for our code

Equipment Used

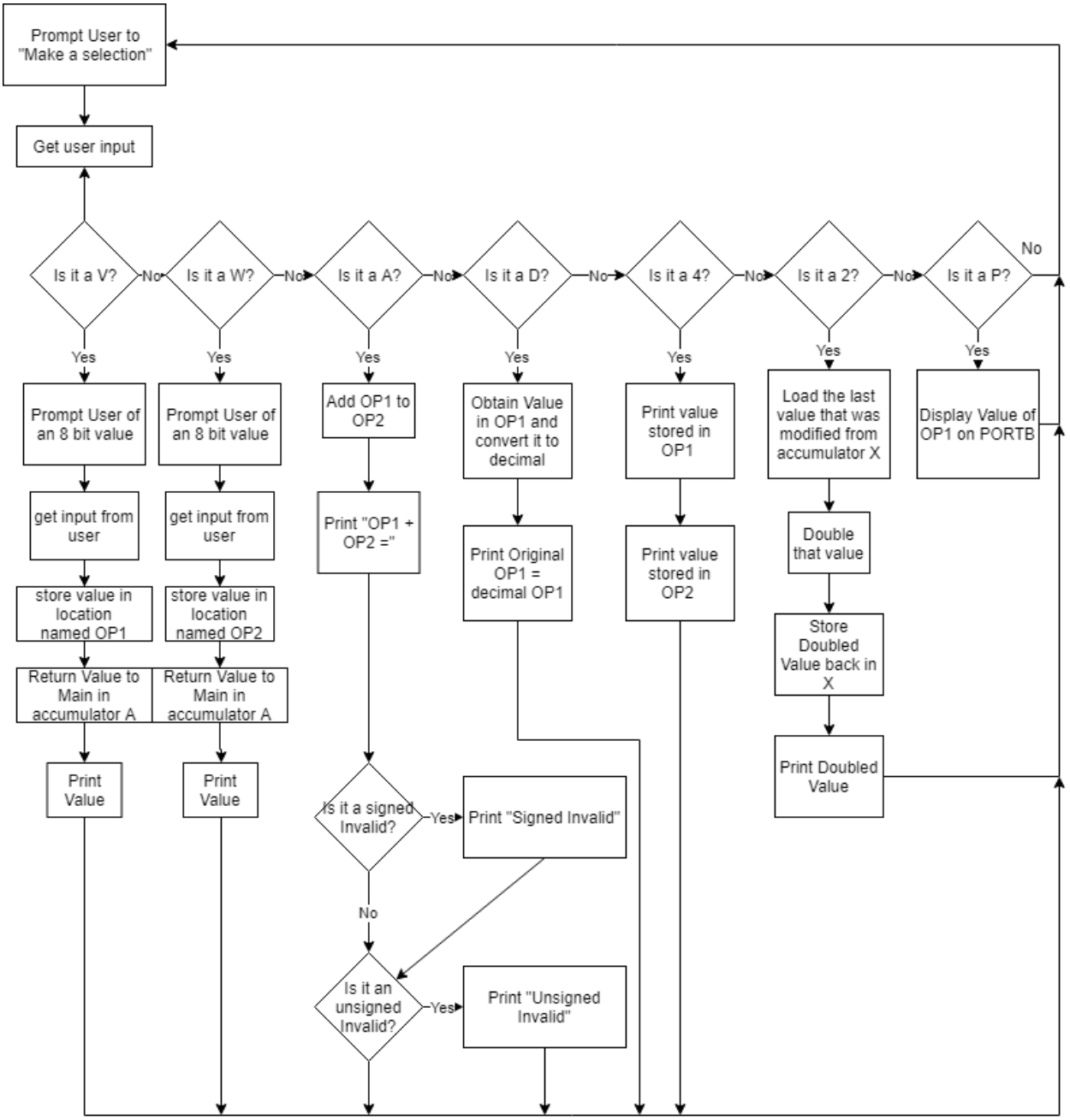
- CodeWarrior
- HCS Dragon12-Light

Pre-Lab

- The purpose of the prelab was to get us familiar with creating flowcharts based on the 7 cases described in the lab assignment, as well as the flowchart from the Parsing Loop ASM sample project. The prelab had us accomplish this by having us draw it out in a manner that would be able to be used to explain the program to another person.
    - Parsing Loop ASM Flowchart



Lab Flowchart



Procedure/ Full Code

- Full Code

```
;*****************************************************************
;* Lab3.ASM
;*
;*****************************************************************
; export symbols
        XDEF Entry, _Startup            ; export 'Entry' symbol
        ABSENTRY Entry        ; for absolute assembly: mark this as application entry point

; Include derivative-specific definitions
                        INCLUDE 'derivative.inc'

;-------------------------------------------------
; Equates Section
;-------------------------------------------------
ROMStart    EQU  $2000  ; absolute address to place my code

;-------------------------------------------------
; Variable/Data Section
;-------------------------------------------------
        ORG RAMStart   ; loc $1000  (RAMEnd = $3FFF)
; Insert here your data definitions here

PROMPT      dc.b    $0A, $0D   ; CR LF
        dc.b    "Type a character:  V W A D 4 2 P:  "
        dc.b    0    ; using zero terminated strings

VPROMPT     dc.b   "   Enter an 8 bit integer:   "
        dc.b    0

VENDPROMPT  dc.b   "   OP1 = "
        dc.b    0

WPROMPT      dc.b   "   Enter an 8 bit integer:   "
        dc.b    0

WENDPROMPT  dc.b   "   OP2 = "
        dc.b    0

UNSINVALID   dc.b  "   UNSIGNED INVALID  "
         dc.b   0

SIGINVALID   dc.b  "   SIGNED INVALID  "
         dc.b   0


         ORG $1500
OP1        dc.b  0
OP2        dc.b  0

TENS_VAL    dc.b    $0   ;Temperary storage for the tens place decimal value in DEC_PRINT
ONES_VAL    dc.b    $0   ;Temperary storage for the once place decimal value in DEC_PRINT


    INCLUDE 'utilities.inc'
```

```
      INCLUDE 'LCD.inc'

;-------------------------------------------------------
; Code Section
;-------------------------------------------------------
          ORG   ROMStart  ; loc $2000
Entry:
_Startup:
          ; remap the RAM &amp;amp; EEPROM here. See EB386.pdf
 ifdef _HCS12_SERIALMON
          ; set registers at $0000
          CLR   $11               ; INITRG= $0
          ; set ram to end at $3FFF
          LDAB  #$39
          STAB  $10               ; INITRM= $39

          ; set eeprom to end at $0FFF
          LDAA  #$9
          STAA  $12               ; INITEE= $9
          JSR   PLL_init      ; initialize PLL
  endif

;-------------------------------------------------------
; Insert your code here
;-------------------------------------------------------
        LDS   #ROMStart ; load stack pointer
        JSR   TermInit  ; needed for Simulator only
        JSR   led_enable   ; enable PORTB for LED's
LOOP     LDD   #PROMPT; pass the adr of the string
        JSR   printf     ; print the string
        JSR   getchar  ; call getchar function -result is: character in B
        JSR   ClearLeds  ; clear LED's when starting over
VCASE    CMPB  #'V'   ; is it a V ?
                BNE   WCASE  ; jump ahead if not
                JSR   VFCN   ; else do corresponding function for V  -----
WCASE    CMPB  #'W'   ; is it a W ?
        BNE   ACASE  ; jump ahead if not
        JSR   WFCN   ;  else do corresponding function for W
ACASE    CMPB  #'A'   ; is it a A ?
        BNE   DCASE  ; jump ahead if not
                JSR   AFCN   ; else do corresponding function for A
DCASE    CMPB  #'D'   ; is it a D ?
        BNE   FOURCASE; jump ahead if not
                JSR   DFCN   ; else do corresponding function for D
FOURCASE CMPB  #'4'   ; is it a 4 ?
        BNE   TWOCASE; jump ahead if not
                JSR   FOURFCN; else do corresponding function for 4
TWOCASE CMPB  #'2'    ; is it a 2 ?
        BNE   PCASE   ; jump ahead if not
                JSR   TWOFCN ; else do corresponding function for 2
PCASE CMPB  #'P'      ; is it a 2 ?
        BNE   NEXT   ; jump ahead if not
                JSR   PFCN ; else do corresponding function for 2
NEXT     JMP   LOOP  ; loop for more input

; Note: main program is an endless loop and subroutines follow
; (Must press reset to quit.)

;===============================================================
; FUNCTIONS CALLED BY MAIN LOOP
;---------------------------------------------------------------
```

```
VFCN    PSHD
            LDD         #VPROMPT   ; pass the address of the string
        JSR     printf        ; print the string
        BSET    PORTB,$01   ; light LED0
        PULD
        JSR getchar         ;get first digit
        SUBB    #$30        ;Subtract 30
        LDAA    #0          ;Load A as a temporary count

        CMPB #$09           ;Check if Value is above $09, hex value of 9
        BLS NOTOVR            ;brach if value is Below $09, go to NOTOVR
        SUBB #$10           ;If its a letter subtract 5 and add 9 to make it appear so in hex
        ADDB #$09
NOTOVR
        STAB OP1            ;Store B for the upcoming mulitiplication
        LDAB #$0            ;Load 0 as B
MULT_16
        ADDB OP1            ;Add OP1 to B until the value becomes the first digit
        CMPA #$F
        INCA
        BLO MULT_16


        STAB OP1            ;Store First Digit
        JSR getchar         ;Get second Digit
        SUBB    #$30        ;Subtract 30
UNDER9_2
        CMPB #$09           ;Check if Value is above $09, hex value of 9
        BLS NOTOVR_2          ;brach if value is Below $09, go to NOTOVR_2
        SUBB #$10           ;If its a letter subtract 5 and add 9 to make it appear so in hex
        ADDB #$09
NOTOVR_2

        ADDB OP1            ;Add first digit to second digit
        STAB OP1            ;Store Value in OP1
        LDAA OP1            ;Load A as the value of OP1 so it can be returned to the main
        PSHD
        LDD  #VENDPROMPT    ;Print out end prompt
        JSR     printf
        PULD
        LDAB OP1            ;Load and print OP1
        LDX #OP1            ;Load location of OP1 into x
        JSR out2hex         ;Final output of OP1
        LDAB #$0
        RTS
;---------------------------------------------------------------------

WFCN
        PSHD
            LDD         #WPROMPT   ; pass the address of the string
        JSR     printf        ; print the string
        BSET    PORTB,$02   ; light LED1
        PULD
        JSR getchar         ;get first digit
        SUBB    #$30        ;Subtract 30
        LDAA    #0          ;Load A as a temporary count

        CMPB #$09           ;Check if Value is above $09, hex value of 9
        BLS NOTOVRW           ;brach if value is Below $09, go to NOTOVR
        SUBB #$10           ;If its a letter subtract 5 and add 9 to make it appear so in hex
        ADDB #$09
```

```
NOTOVRW
      STAB OP2          ;Store B for the upcoming mulitiplication
      LDAB #$0          ;Load 0 as B

MULT_16W
      ADDB OP2          ;Add OP1 to B until the value becomes the first digit
      CMPA #$F
      INCA
      BLO MULT_16W


      STAB OP2          ;Store First Digit
      JSR getchar       ;Get second Digit
      SUBB    #$30      ;Subtract 30
UNDER9W_2
      CMPB #$09         ;Check if Value is above $09, hex value of 9
      BLS NOTOVRW_2         ;brach if value is Below $09, go to NOTOVR_2
      SUBB #$10         ;If its a letter subtract 5 and add 9 to make it appear so in hex
      ADDB #$09
NOTOVRW_2

      ADDB OP2          ;Add first digit to second digit
      STAB OP2          ;Store Value in OP2
      LDAA OP2          ;Load A as the value of OP2 so it can be returned to the main
      PSHD
      LDD  #WENDPROMPT    ;Print out end prompt
      JSR     printf
      PULD
      LDAB OP2          ;Load and print OP2
      LDX #OP2          ;Load Location of OP2 into X
      JSR out2hex       ;Print Final OP2
      LDAB #$0
      RTS
;---------------------------------------------------------------------
AFCN
      BSET   PORTB,$04   ; light LED2
      LDAB  #$24      ;Load and print a $
      JSR putchar
      LDAB OP1        ;Load and print OP1
      JSR out2hex
      LDAB #$2B       ;Load and print a +
      JSR putchar
      LDAB  #$24      ;Load and print a $
      JSR putchar
      LDAB OP2        ;Load and print OP2
      JSR out2hex
      LDAB #$3D       ;Load and Print =
      JSR putchar
      LDAB  #$24      ;Load and print a $
      JSR putchar
      LDAB OP1        ;add OP1 to OP2 and print
      ADDB OP2
      JSR out2hex     ;Output value

      LDAA OP1
      ADDA OP2
      BCC L1
      LDD               #UNSINVALID   ; pass the address of the string
      JSR     printf        ; print the string
L1
      LDAA OP1
```

```
        ADDA OP2
        BVC  L2
        LDD             #SIGINVALID   ; pass the address of the string
        JSR     printf        ; print the string
L2
        RTS
;---------------------------------------------------------------------
DFCN
        BSET   PORTB,$08   ; light LED3
        LDAB  #$24       ;Load and print a $
        JSR putchar
        LDAB OP1        ;Load and print OP1
        JSR out2hex
        LDAB #$3D       ;Load and Print =
        JSR putchar
        LDAB OP1         ;Load OP1

        ;Start of Hex to decimal algorithm (uses ONES_VAL and TENS_VAL)
        LDAA #0        ;load zero in for the tens place
HEXLOOP
        CMPB #$0A      ;check if value is already less than 10
        BLO DEC_PRINT  ;if so go to dec_print
        SUBB #$A      ;subtract 10
        INCA          ;increase tens place
        CMPB #$0A      ;check if its below 10 now
        BHS HEXLOOP    ;if not go back to loop


DEC_PRINT
        STAB ONES_VAL  ;store ones value
        STAA TENS_VAL  ;store tens value
        LDAB TENS_VAL  ;obtain tens value
        ADDB #$30      ;add $30 so the ascii value lines up
        JSR putchar    ;print tens value
        LDAB ONES_VAL  ;load ones value
        ADDB #$30      ;add $30 so the ascii value lines up
        JSR putchar    ;print ones value
        ;End of Hex to decimal algorithm

        LDAB #$0

        RTS
;---------------------------------------------------------------------
FOURFCN
        BSET   PORTB,$20; light LED4
        LDAB  #$24      ;Load and print a $
        JSR putchar
        LDAB  OP1
        JSR out2hex     ;Load and print OP1
        LDAB  OP2
        JSR out2hex     ;Load and print OP2
        RTS
;---------------------------------------------------------------------
TWOFCN
        BSET   PORTB,$10; light LED5
        LDAA 0,X
        ADDA 0,X
        STAA 0,X
        LDAB 0,X
        JSR out2hex
```

```
      RTS
;--------------------------------------------------------------------
PFCN    LDAA OP1
      STAA PORTB      ;Set value as portb
      RTS

ClearLeds
      CLR   PORTB     ; clear all LED's
      RTS
;--------------------------------------------------------------------

;*****************************************************************
;*              Interrupt Vectors                        *
;*****************************************************************
          ORG   Vreset
          DC.W  Entry        ; Reset Vector
```

- Part A: User presses V
    - *Explanation of Code:*
        - *We created a subroutine to prompt the user to enter an 8-bit integer value from the keyboard. This uses a lengthy process to input two keystrokes by transforming the first keystroke into the first four bits and then the second keystroke into the second four bits. We then took the memory value entered and stored it into a memory location named OP1. The entered value was finally returned it to the main program in register A, and then the value was printed like so: "OP1 = $2F" (if $2F was the input value).This also stores the value in register X for use in TWOFCN before returning to the main program.*

- Part B: User presses W
    - *Explanation of Code:*
        - *This function is exactly the same as the V function. This uses a lengthy process to input two keystrokes by transforming the first keystroke into the first four bits and then the second keystroke into the second four bits. We then took the memory value entered and stored it into a memory location named OP2. The entered value was finally returned it to the main program in register A, and then the value was printed like so: "OP2 = $1C" (if $1C was the input value). This also stores the value in register X for use in TWOFCN before returning to the main program.*

- Part C: User presses A
    - *Explanation of Code:*

        - *When the user presses A, the program will call AFCN. This function will call the values of OP1 and OP2 and add them together, printing out the following format: $2F + $1C = 4B (given that the two variables stored were $2F and $1C). It will then do the addition of the two variables again and check for a C flag, if found it will print "Signed overflow". Similarly it will go on to do the addition again and check for a V flag, if found it will print "Unsigned overflow" before returning to the main program.*

- Part D: User presses D
    - *Explanation of Code:*
        - *When the user presses D, the DFCN will be called. This function will ultimately print out the OP1 = the decimal version of OP1, Ex $1F = 31. The decimal conversion is done by subtracting $A from OP1 until it is below $A, incrementing a count each time it does the subtraction. That count is then used as the tens place digit for the resulting decimal number. This is simply added to the ones place digit which is left over from the subtractions and then what remains is the decimal number.*

- Part E: User presses 4
    - *Explanation of Code:*
        - *When the user presses 4, the program calls FOURFCN. This will simply print out OP1 and OP2 next to each other, Ex $1F2C (given that OP1 was $1F and OP2 was $2C).*

- Part F: User presses 2
    - *Explanation of Code:*
        - *When the user presses 2, the subroutine assumes that the address of the memory location that it is looking to modify is in the X register (this is the last value that was stored for either OP1 or OP2). It then takes the value and doubles it. If OP1 is in the X register and is called, then the contents of OP1 will be replaced by double the value that was originally in OP1.*

- Part G: User presses P
    - *Explanation of Code:*
        - *When the user presses P, the PFCN is called. This will take the current value of OP1 and store it on PORTB to be displayed on the prototyping board.*

Post-Lab

1.

```
;FUNCTIONS
PLQ1
        LDAA POSTLABNUM   ;Load $3D
        EORA #$6F         ;XOR #3D with $6F
        STAA TEMPVAR      ;store in a temporary variable
        LDAB #$20         ;load and print a space
        JSR putchar       ;for readabillity
        LDAB TEMPVAR      ;load the XOR value
        JSR out2hex       ;print the value

        RTS
```

```
Enter a letter: W 52

Enter a letter:
```

2.

```
PLQ2
        LDAA POSTLABNUM   ;Load in $3D
        LSLA              ;Left shift $3D
        STAA TEMPVAR      ;store in a temporary variable
        LDAB #$20         ;load and print a space
        JSR putchar       ;for readabillity
        LDAB TEMPVAR      ;Load in the shifted variable
        JSR out2hex       ;Print the variable
        RTS
```

```
Enter a letter: X 7A

Enter a letter:
```

3.

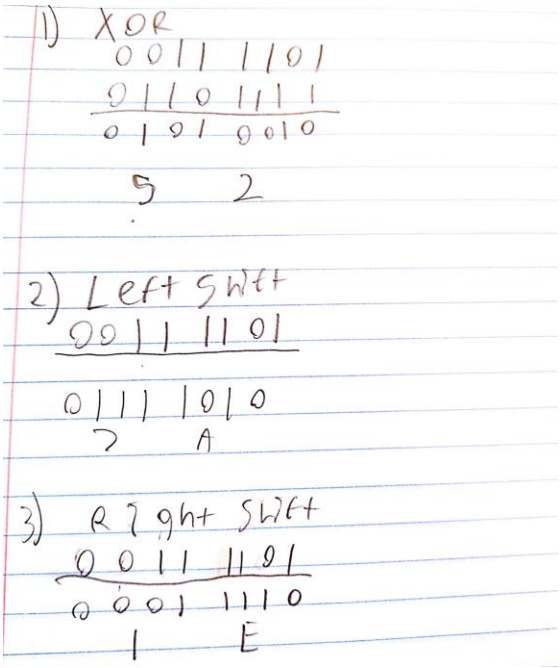
```
PLQ3
        LDAA POSTLABNUM   ;Load in $3D
        ASRA              ;Aritmetic right shift
        STAA TEMPVAR      ;store in a temporary variable
        LDAB #$20         ;load and print a space
        JSR putchar       ;for readabillity
        LDAB TEMPVAR      ;Load in the shifted variable
        JSR out2hex       ;Print the variable

        RTS
```

```
Enter a letter: Y 1E

Enter a letter:
```

4. The programs worked as expected. See below



5. Except for one difference, the arithmetic shift and the logic shift do the exact same thing. The difference between them is that the arithmetic shift replicates the sign bit while a logic shift will bring in a zero to shift the number. This makes the arithmetic shift better suited for signed numbers as a negative number will remain negative after the shift.

6.

```
PLQ4
            LDD #PROMPT2     ;Prompt the user for the voting number
            JSR printf
            LDAA #0          ;Use A as a vote counter
            JSR getchar      ;Get input from user
            JSR out2hex      ;Print entered number
            STAB TEMPVAR
            BRCLR TEMPVAR,%1000000,L1 ;Check first bit
            INCA
L1
            BRCLR TEMPVAR,%0100000,L2 ;Check second bit
            INCA
L2
            BRCLR TEMPVAR,%0010000,L3 ;Check third bit
            INCA
L3
            BRCLR TEMPVAR,%0001000,L4 ;Check fourth bit
            INCA
L4

            CMPA #$3         ;Check if vote counter is 3 or higher
            BLO RESULTS      ;if not jump to results
            LDD #ONES_WON    ;if so print ONES_WON
            JSR printf
            RTS
RESULTS
            LDD #ZEROS_WON   ;Print ZEROES_WON
            JSR printf
            RTS
```

```
Enter a letter: Z

Enter a letter to vote on: 5A

The number of 1's in the 5 left most bits is greater than the number of 0's

Enter a letter: Z

Enter a letter to vote on: 41

The number of 0's in the 5 left most bits is greater than the number of 1's

Enter a letter:
```

Conclusion

- In the prelab, we focused a lot on how to create flowcharts for the functions asked for in the Lab Assignment. Those flowcharts created will be used when explaining to someone what each function does and how it works.
- In the Lab Assignment, the focus was creating subroutines that could be called by the terminal for implementation. We learned how to input a letter/number to call a function, and how to program each of the chosen letters/numbers to perform a subroutine, each with their own separate output.
- In the postlab we looked at some of the new functions we learned in class. Specifically we looked at the use of the new XOR function (EORA) and the arithmetic shift. We also looked into basic manipulation of the memory on the port of the dragon_12 board using the BCLR command.