Joseph Godinez

Professor Oscar Ho

Computer Science 240-20

December 4, 2021

1. Screenshot of code

```
.ORIG x3000
        TRAP x23        ; Get user input
        ADD R2, R0, #0  ; Putting the first value into R2
        LD R3, NEGASCII ; Load R3 with negative -30
        TRAP x23        ; get the second input
        ADD R4, R0, #0  ; Puts the second input into R4
        LD R5, negNine  ; negative ascii 9
        ADD R1, R4, R3  ; Adds and store the results of second input with -30
        ADD R1, R2, R1  ; Taking the value in R1 adding that to the first input
        ADD R5, R1, R5  ; hold remainder past 9
        BRp Greater     ; Branching if the value would be greater than 9
        ADD R0, R1, #0  ;
        OUT             ; Displaying the value
        BRnzp DONE      ; Branching to end
Greater LD R2, vA       ; ascii value of 'A'
        ADD R2, R2, #-1 ; Add -1 to R2
        ADD R0, R2, R5  ; Add the value of R2 and R5 into R2
        OUT             ; Display message
DONE TRAP x25

NEGASCII .FILL x-0030
ASCII .FILL x0030
vA .FILL #0065
negNine .FILL x-0039
.END
```

2. Screenshot of the subroutine

```
;
;  The Calculator, Main Algorithm
;
            LEA       R6,StackBase  ; Initialize the Stack Pointer.
            ADD       R6,R6,#1      ; R6 = StackBase + 1 --> empty stack

NewCommand  LEA       R0,PromptMsg
            PUTS
            GETC
            OUT
;
; Check the command
;
TestX       LD        R1,NegX       ; Check for X.
            ADD       R1,R1,R0
            BRnp      TestC
            HALT
;
TestC       LD        R1,NegC       ; Check for C.
            ADD       R1,R1,R0
            BRnp      TestAdd
            JSR       OpClear       ; See Figure 10.20
            BRnzp     NewCommand
;
TestAdd     LD        R1,NegPlus    ; Check for +
            ADD       R1,R1,R0
            BRnp      TestMult
            JSR       OpAdd         ; See Figure 10.8
            BRnzp     NewCommand
;
TestMult    LD        R1,NegMult    ; Check for *
            ADD       R1,R1,R0
            BRnp      TestMinus
            JSR       OpMult        ; See Figure 10.12
            BRnzp     NewCommand
;
TestMinus   LD        R1,NegMinus   ; Check for -
            ADD       R1,R1,R0
            BRnp      TestD
            JSR       OpNeg         ; See Figure 10.13
            BRnzp     NewCommand
;
TestD       LD        R1,NegD       ; Check for D
            ADD       R1,R1,R0
            BRnp      EnterNumber
            JSR       OpDisplay     ; See Figure 10.19
            BRnzp     NewCommand
;
; Then we must be entering an integer
;
EnterNumber JSR       PushValue     ; See Figure 10.16
            BRnzp     NewCommand
;
PromptMsg   .FILL     x000A
            .STRINGZ "Enter a command:"
NegX        .FILL     xFFAB
NegC        .FILL     xFFBD
NegPlus     .FILL     xFFD5
NegMinus    .FILL     xFFD3
NegMult     .FILL     xFFD6
NegD        .FILL     xFFBC

; Globals
StackMax    .BLKW     #9
StackBase   .BLKW     #1
ASCIIBUFF   .BLKW     #4
            .FILL     x0000 ; ASCIIBUFF sentinel
```

In order for the program to execute the BRnzp PushValue we would have to make a whole subroutine. Every time we make a new subroutine we must always load the data into the register, so LD R1, Neg0. Then we are changing the data from R1 with R0 to R1 of R0 data, ADD R1, R1, R0. Then check the calculator for BRn HaltCalc if the user enters a negative, then the same for positive ADD R1, R1, R0 and BRp HaltCalc. Then last is BRnz EnterNumber to enter a number. Last is to make the HaltCalc which is LD R0, XVal and BRnzp TestX for when the program goes to X.