CS270 Recitation 8
Subroutines in Assembly language

Goals

To understand how subroutines work. In the first part you will observe how the argument values and the return value are passed and how the return address is saved and retrieved. In the second part you will link the program to a pre-written subroutine to convert a binary number into ASCII, and thus print the result.

Tools needed:

- LC3 Tools assembler to generate the object code and Simulator for simulation.

The Assignment

Make a subdirectory called R7 for the recitation assignment, all files should reside in this subdirectory.

1. Download or create the file subtractSubroutine.asm (below)

```
;Main program load numbers and performs subtraction
; Num3<- Num1- Num2
;using subroutine SUB
      .ORIG x3000
      LD R1, Num1 ;subroutine argument
      LD R2, Num2 ;subroutine argument
      JSR SUBT
      ST R3, Num3 ;value returned by SUBT
      HALT
;    Data
Num1  .fill 23
Num2  .fill 38
Num3  .blkw 1

;Subtract subroutine: Performs R3 <- R1-R2
;Arguments: R1, R2, Returns value in R3
;
SUBT  NOT R3, R2
      ADD R3, R3, #1
      Add R3, R1, R3
      RET
      .END
```

2. Read and try to mentally comprehend the program. Then assemble the file to generate subtractSubroutine.obj.

3. Launch the simulator and load subtractSubroutine. Obj. Set the breakpoints at instructions shown bold above (note execution stops before the instruction selected). Fill the table below with the values of the specific registers.

Instruction	PC	R1	R2	R3	R7	Comment
JSR SUBT	x3001					
NOT R3, R2	x3002					
RET	x3003					
ST R3, Num3	x3004					

Are they what you expect? If not, ask the TA. Show the TA the filled table.

4. Insert the subroutine BinarytoASCII.asm (below, from p. 277 in the book) to your code just before `.END`. You don't need to know how it works, just need to know that it takes the number in R0 and places a string of 3 ASCII characters (numerals)

    a. Insert some code right below `ST R3, Num3` so that you will place the result in R3 into R0 and then call BinarytoASCII. You will then load R0 with address of ASCIIBUFF and then call PUTS to print out the ASCII string on the console.
    b. Add this to the data at the bottom of BinarytoASCII (just below .END):
    ```
    ASCIIBUFF .BLKW 5
    ```
    c. Assemble and load the program. Run and verify the operation.
    d. Change the contents of location Num2 so that the result of subtraction will be positive and test the program. Finally change Num2 so that the result to subtraction will be 0 and test the program. Demonstrate the program to the TA.

```
; Figure 10.20, page 277
;
;  This algorithm takes the 2's complement representation of a signed
;  integer, within the range -999 to +999, and converts it into an ASCII
;  string consisting of a sign digit, followed by three decimal digits.
;  R0 contains the initial value being converted.
;
BinarytoASCII  LEA   R1,ASCIIBUFF  ; R1 points to string being generated
               ADD   R0,R0,#0      ; R0 contains the binary value
               BRn   NegSign       ;
               LD    R2,ASCIIplus  ; First store the ASCII plus sign
               STR   R2,R1,#0
               BRnzp Begin100
NegSign        LD    R2,ASCIIminus ; First store ASCII minus sign
               STR   R2,R1,#0
               NOT   R0,R0         ; Convert the number to absolute
               ADD   R0,R0,#1      ; value; it is easier to work with.
;
Begin100       LD    R2,ASCIIoffset ; Prepare for "hundreds" digit
;
               LD    R3,Neg100     ; Determine the hundreds digit
Loop100        ADD   R0,R0,R3
               BRn   End100
```

```
                ADD   R2,R2,#1
                BRnzp Loop100
;
End100          STR    R2,R1,#1   ; Store ASCII code for hundreds digit
                LD     R3,Pos100
                ADD    R0,R0,R3   ; Correct R0 for one-too-many subtracts
;
                LD     R2,ASCIIoffset ; Prepare for "tens" digit
;
Begin10         LD     R3,Neg10   ; Determine the tens digit
Loop10          ADD    R0,R0,R3
                BRn    End10
                ADD    R2,R2,#1
                BRnzp  Loop10
;
End10           STR    R2,R1,#2   ; Store ASCII code for tens digit
                ADD    R0,R0,#10  ; Correct R0 for one-too-many subtracts
Begin1          LD     R2,ASCIIoffset ; Prepare for "ones" digit
                ADD    R2,R2,R0
                STR    R2,R1,#3
                RET
;
ASCIIplus       .FILL  x002B
ASCIIminus      .FILL  x002D
ASCIIoffset     .FILL  x0030
Neg100          .FILL  xFF9C
Pos100          .FILL  x0064
Neg10           .FILL  xFFF6
```