CS270 Recitation 10
"Help Session for LC-3 Assignment"

Goals

To help students with the LC-3 programming assignment on floating-point addition and subtraction, or more specifically:

1. To resolve any problems you might have with the LC-3 assembler and simulator.
2. To explain the LC-3 assignment in more detail and answer any questions you might have.
3. To provide the solution to right shift for students that did not get it done themselves.

We will try to help you with your LC-3 code, but of course we only have 50 minutes, so you might want to be prepared when you come in!

The Assignment

1) The teaching assistant will show how to call one LC-3 function from another, by setting parameters and saving and restoring the return address stored in the R7 register.

2) The teaching assistant will present the code for implementing right shift.

RIGHT SHIFT

```
right_shift     ; Result is Param1 shifted right by Param2 bits
                ; Algorithm: walk source and destination bit

                 LD R0,Param1          ; load parameter
                 LD R2,Param1          ; load parameter
                 LD R1,Param2          ; load count
                 BRnz return_rs        ; count must be positive
                 AND R2,R2,0           ; clear result
                 LD R3,ONE             ; source mask = 1
                 LD R4,ONE             ; destination mask = 1

rshift_loop1     ADD R3,R3,R3          ; left shift source mask
                 ADD R1,R1,#-1         ; decrement count
                 BRp rshift_loop1      ; continue looping

rshift_loop2     AND R5,R0,R3          ; source bit set?
                 BRz rshift_next       ; not set, do nothing
                 ADD R2,R2,R4          ; set, update result
rshift_next      ADD R4,R4,R4          ; shift destination mask
                 ADD R3,R3,R3          ; shift source mask
                 BRnp rshift_loop2     ; continue looping

return_rs        ST R2,Result          ; store result
                 RET
```

3) The teaching assistant will talk about how to do incremental development for floating point addition, as follows:

Step A) Make sure you can extract the sign, exponent, and mantisaa fields from both floating point operands.
Step B) Make sure you can construct the floating point result from the sign, exponent, and mantissa fields.
Step C) Start with positive operands to avoid needing any 2's complement conversion code.
Step D) Start with identical exponents to avoid needing any normalization code for operands.
Step E) Write the code to add the mantissas together to get the mantissa for the sum.
Step F) Write the code to normalize the resulting sum, at least in the case where right shift is required.
Step G) Test out 5.5 + 6.25 = 11.75. The associated hexadecimal values are 0x4580, 0x4640, and 0x49E0.
Step H) Implement 2's complement conversion for the operands and result, including setting the result sign.
Step I) Test out 5.5 + -2.5 = 3.0, you must figure out the hexadecimal values for this test case.
Step J) Implement normalization of operands, you can assume the first operand exponent >= second operand exponent.
Step K) Test out 11.25 + 6.5 = 17.75, you must figure out the hexadecimal values for this test case.
Step L) Implement floating point subtraction by negating the second operand and calling the addition function.

Here are some comments you might want to copy into your flt32_add code:

```
; STEP ONE) Extract fields from operands

; STEP TWO) Equalize operand exponents

; STEP THREE) Convert operands to 2's complement

; STEP FOUR) Add mantissas

; STEP FIVE) Convert sum from 2's complement

; STEP SIX) Normalize sum

; STEP SEVEN) Compose sum from fields
```