LC3 Programming

CS 350: Computer Organization & Assembly Language Programming

Lab 9, due Fri Apr 11

A. Why?

- The LC-3 and similar architectures are representative of simple instruction set architectures.
- Writing low-level programs illustrates what low-level hardware does and how compilers and interpreters work.

B. Outcomes

After this lecture lab, you should be able to:

- Translate pseudocode into programs in LC-3 assembler.
- Write short assembler programs to do calculations and searches.

C. Programming Problems [50 points total]

Note: Take your `*.asm` files, zip them together, and submit the zipped file through Blackboard. (Don't include the `*.obj`, etc. files or runs of the programs.) Be sure to put your name inside each `asm` file.

In general for all the programs, up to 30% of the credit is for good commenting. At the top of each program file, say what the program does. Also, say what registers are holding which values and what the relationships are between those values. Also include line comments to say what you're doing (in higher-level terms if you can: "`; count++`" is better than just "`; R0++`").

1. [15 pts] Translate the following pseudocode for calculating 2^N into LC-3 assembler. (Note you'll have to implement `p` and `x` with registers.) There's a skeleton file `Lab09_q1_skel.asm` to help getyou started.

```
; Calculate 2^N where N >= 0 ( ^ means exponentiation.)
; If N is too large, this calculation overflows.
;
```

```
; Properties of values:
;   0 ≤ x ≤ N and p * 2^x = 2^N  (These hold every time
;   we hit the loop test.)
;
; Register usage:
;   R... = p; R... = x  (specify whichever registers you decide to use)
;
   p ← 1          ; Establish x and p initially
   x ← N
   while x ≠ 0 do
      p ← p + p
      x ← x - 1
   end loop
   result ← p        ; result = p = 2^N
   halt

N      .FILL  (a value ≥ 0)
result .BLKW  1
```

2. [15 pts] Write an LC-3 program that left-shifts a value from one word to another; that is, we left-shift but save the bits that are shifted out and store them into a second result. For example, if we shift `1111 1111 1111 1111` left five places, then we would have two words of results: `0000 0000 0001 1111` and `1111 1111 1110 0000`.

   Your program should have labels `X`, `Y1`, `Y2`, and `N` and left-shift the value in `X` by `N` positions, saving the results into locations `Y1` and `Y2`. Example: for

   ```
   X  .FILL  xFFFF
   Y1 .BLKW  1
   Y2 .BLKW  1
   N  .FILL  5
   ```

   then on completion, `Y1` and `Y2` should contain `x001F` and `xFFE0` respectively.

   The labels don't have to be declared in the order shown above, and you can declare extra labels and variables as you like. Also, you don't have to check `N` to see that it's nonnegative or ≤ 16. Here's some pseudocode:

```
Y1 ← 0
Y2 ← X
counter ← N
while counter > 0
   Y1 ← 2*Y1
   if leftmost bit of Y2 is 1
       Y1 ← Y1 + 1
   Y2 ← 2*Y2
   counter--
```

(Hint: A word contains a negative value iff its leftmost bit is a 1.)

3. [20 pts] Write an LC-3 program that read a character and a string, counts the number of occurrences of the character in the string and prints the count (if it's ≤ 9, else print a message saying the count was > 9). For example, the LC-3 console might look like this: (user input is underlined; "↵" means carriage return). Note the string can contain spaces.

```
Char to search for: e
String to search through: a*cdee exe? !xy& ↵
The count was 5
```

Here's some pseudocode for the program:

```
Print a prompt for a character s to search for
Read it in and echo it (print it back out)
Initialize count = 0
Print a prompt for a string
Read a character c and echo it

while c ≠ line feed
   if c = s then ++count
   read next value of c and echo it
end loop

if count ≤ 9
   Convert the count to an ASCII character
   Print the count to the monitor
else
   Print "There were more than 9 occurrences"
halt
```