Lab 7: LC-3 Instruction Set Architecture

Due Date: Thursday 3/30/2017 11:59PM

This lab covers material on the ISA of the LC-3 (lectures 13-16). There are 100 points total.

Written Problems (*20 points*)

Rewrite the table shown below, filling in any missing parts indicated by question marks. The comment for `x7FFF` shows the level of comment needed (you can write in more detail if you like). Note the values at `x800B`, `x800C`, and `x800D` will change; show this in their comment fields.

Addr	Assembler	Action/Comment
x7FFF	LD R0,9	R0 ← M[x8009] = 12
x8000	LD R1,9	?
x8001	LDR R2,R1,0	?
x8002	LEA R4,8	?
x8003	STR R2,R4,0	?
x8004	AND R3,R3, ?	R3 ← 0
x8005	STR R3,R4,1	?
x8006	ADD R5,R0,R0	?
x8007	STI R5,2	?
x8008	TRAP x25	HALT
x8009	.FILL 12	
x800A	.FILL x800D	
x800B	.FILL 0	?
x800C	.FILL -1	?
x800D	.FILL 18	?
x800E	.FILL 0	

Coding Assignment (*80 points*)

Evaluate a Quadratic Polynomial

- You are to write a multiplication subroutine and call it multiple times (using `JSR`) to evaluate a quadratic polynomial. You can use the LC-3 simulator I posted earlier to verify that your program does what you expect.

- The polynomial $poly(x)$ is represented as its three coefficients; the value X_1 is an integer; you evaluate and store $Y_1 = poly(X_1)$. For example, say we begin with

```
; poly(x) = -4*X^2 + 3*X - 5
;
POLY .FILL -4       ; -4*X^2
     .FILL 3        ;+3*X
     .FILL -5       ;-5
     ; To calculate: Y1 = poly(X1)
     ;
     X_1      .FILL     -2
     Y_1      .BLKW     1
```

then you would calculate $-4 \times (-2)^2 + 3 \times (-2) - 5 = -27$ and store that in `Y_1`. (Programming hint: You can't use `X1` as a variable; the LC-3 assembler treats it as the constant `x0001`.)

In the lecture on subroutines, we saw a routine that calculates `X * Y` if `Y` $\geq$ `0`; you'll need to extend it towork if `Y` $<$ `0`.

Program Structure Your main program can be written in various ways (e.g., $((A \times x_1) \times x_1)$ vs. $((x_1 \times x_1) \times A)$; in pseudocode you could have

```
main:
---------------------------
R0 = A          ; coefficient A from POLY
R1 = X_1
JSR MULT        ; R1 = A * X_1
R0 = X_1
JSR MULT        ; R1 = A * X_1^2
tmp = R1        ; tmp = A * X_1^2

R0 = B          ; coefficient B from POLY
R1 = X_1
JSR MULT        ; R1 = B * X_1
tmp = tmp + R1 ; tmp = A * X_1^2 + B * X
tmp = tmp + C  ; add coefficient C from POLY
Y_1 = tmp       ; save result
HALT
```

(Programming hint: If you keep the address of `POLY` in register *reg*, you can use `LDR R0,` *reg*, `0` to set `R0` = coefficient A; replacing `0` by `1` or `2` gives you B or C.)
Write your program as
main program
declarations of `POLY`, `X_1`, *and* `Y_1`
`MULT` *routine*
and turn in your `*.asm` files.

Hand-in Instructions

You will be handing in this assignment digitally on `fourier`. Make sure your solution to the written portion is in a `.pdf` file.

Make sure to put your name on your submission. Submissions without names will be given zero points! For code, this means put a comment at the top of your code file(s) with your name on it.

To handin your files, do the following (assuming you've called your written portion `lab7.pdf` and all your files, including the PDF and your `.asm` files are in your current directory).

```
[you@fourier] tar cvzf `whoami`-lab7.tgz lab7.pdf *.asm
[you@fourier] cp `whoami`-lab7.tgz /home/khale/HANDIN/lab7
```

Note that those are backticks, not single quotes.

Late handins If you're turning in your code late, you'll need to e-mail it to me after having created a `.tgz` file from it **on fourier**.

You'll then want to use `scp` or Filezilla or the equivalent to get that file off of fourier onto your local machine and send it to me as an e-mail attachment.