CCPS310 Lab 2 – ARC II

Preamble

In this lab you'll get some more experience writing ARC assembly. You'll use logical and arithmetic instructions in new ways, you'll practice branching, and jumping to/returning from subroutines.

Lab Description

1) **(3 marks)** SPARC supports floating point numbers and arithmetic. ARC, the subset of SPARC we are using, does not. For this question, you will write an assembly program that loads a 32-bit word assumed to be in encoded in the format of a single precision floating point number (IEEE-754)

   Your program will extract the exponent portion of the number and subtract the bias (127 in this case). Load the register from memory location `[flt]`, and store the exponent result at location `[exp]`. Use the code snippet below to get started.

```
      .begin
      .org 2048
prog: ! Your assembly code here

flt:  0xc14a0000 ! 32-bit float to load
exp:  0x0        ! store exponent as integer here
     .end
```

2) **(6 marks)** In class, we saw an example of iterative multiplication using branching. For this question, you will implement iterative *exponentiation*. Base and exponent values will be found in memory at labels **`base:`** and **`exp:`** respectively. Your program should calculate the exponentiation and store the result at memory location **`res`**.

   The base and exponent values are assumed to be positive integers. Other than this, make no assumptions. See the program snippet below to get you started.

```
       .begin
       .org 2048
prog:     ! Your assembly code here

base:  4  ! Here, we're calculating
exp:   7  ! 4 to the power of 7 (4^7)
res:   0  ! Should be 16384 when program is done
       .end
```

3) **(6 marks)** In similar fashion to iterative multiplication and exponentiation, you will implement integer division using branching (think of how achieving division would differ…). In the code below, operands are stored at labels **`x:`** and **`y:`**. The integer result is to be stored at label **`quot:`** and the remainder is to be stored at label **`rem:`**.

```
        .begin
        .org 2048
prog:     ! Your assembly code here

x:      8  ! Here, we're dividing 8/3
y:      3
quot:   0  ! Integer result would be 2 in this case
rem:    0  ! remainder would also be 2
        .end
```

ARC Specifics

Aside from the specifications indicated in each of the above questions, there are no restrictions on how you structure your code, or which registers you use. Use as many registers as you need, as much extra memory as you need, etc. As long as the input and results are found at the correct labels, you're all set.

Submission

Labs are to be submitted ***individually***!

Submit a single file containing all three of your programs. An assembly .asm file is fine, as is a plain text file (.txt) containing your code. No PDFs, no word documents, etc. This is to make it easy for me to copy/paste your code into ARCTools. Make sure each program is clearly separated in your submission. Upload this file, containing all three programs, on D2L under Lab #2.