Lab 3: Arithmetic and Logical Instructions

CS/CoE 0447(B) Spring 2015 — Dr Kosiyatrakul (Tan)

Released: Friday 30 January 2015, 06:00 EST
Due: Thursday 5 February 2015, 23:59 EST

Submission timestamps will be checked and enforced strictly by CourseWeb; **late submissions will not be accepted.** Remember that, per the course syllabus, if you are not marked by your recitation instructor as having attended a recitation, your score will be cut in half. **Follow all instructions.**

A. Bit Manipulation

Read an integer *A* from a user, and place it in register $t0. Recall that the 32 bits in a word are counted from bit 0, which is the right-most or least significant bit (LSB), to bit 31, which is the left-most or most significant bit (MSB).

Now, set register $t1 to contain **only** bits 15, 16, and 17 of *A*. That is, the LSB of $t1 should contain bit 15 from *A*, the second least significant bit of $t1 should contain bit 16 from *A*, etc. Then return the resulting number as output to the user.

For example, say your input integer is 985980. This corresponds to 0x000f0b7c in hexadecimal, which is 0000 0000 0000 1111 0000 1011 0111 1100 in binary. If you isolate bits 15, 16, and 17 (which are underlined above) and move them to the least significant bits of $t1, you will get 0x00000006.

Here are some sample executions for your program:

```
Please enter your integer: 985980
Here is the output: 6
```

```
Please enter your integer: 954367
Here is the output: 5
```

You should be able to accomplish the bit manipulation using at most one shift instruction (e.g., `sll`, `srl`, `sra`) and one logical instruction (e.g., `andi`, `ori`, `xori`, etc.) There are other ways of accomplishing the same manipulation that may use more instructions, or others altogether, but you should make your program as minimal as you can.

- ▶ **Save your program as `lab03part1.asm`. Please match the provided output format exactly.**

B. Exponent Calculator

Write a program which, given two nonnegative integers *x* and *y*, computes the value of *x^y*.

You must accomplish this task using the branching, comparison, and control flow instructions covered in Lab 2; in particular, you may not employ the use of functions. You are also not permitted to use `mult`, `mul`, or similar instructions, so **all of your math must be done using the other arithmetic and logical instructions you have seen.**

Prompt the user for the two integers, each in turn. If the user enters a negative number, remind them that they must provide a nonnegative integer, and ask again.

Your program should handle both 0^*y* = 0 and *x*^0 = 1. **For the purposes of this assignment, 0^0 = 1.** Here are some sample executions:

```
x^y calculator
Please enter x: 5
Please enter y: 7
5^7 = 78125

x^y calculator
Please enter x: 0
Please enter y: 4
0^4 = 0

x^y calculator
Please enter x: 6
Please enter y: 0
6^0 = 1
```

```
x^y calculator
Please enter x: -3
Integer must be nonnegative.
Please enter x: 2
Please enter y: -4
Integer must be nonnegative.
Please enter y: 20
2^20 = 1048576
```

Of course, your program won't work for particularly large *x* values, and even in most cases when *y* is more than a few dozen. This is because we're doing our calculations in 32-bit registers and so we can only represent 2^{32} different values at most. It should be straightforward to see, then, that requests like 3^{32} or 2^{36} will overflow the register. **Assume that any valid nonnegative integers provided as input will result in a value that will fit in a 32-bit register.** In other words, don't worry about overflow.

- **Save your program as `lab03part2.asm`. Please match the provided output format exactly.**
- **Zip your two programs for this assignment into a single zip file called `lab03.zip` and submit it via CourseWeb.**