CPSC 213: Assignment 6

Due: Monday, March 5, 2012 at 6pm.

Late assignments are accepted until Wednesday, March 7 at 6pm with a 25% penalty per day (or fraction of a day) past the due date. This rule is strictly applied and there are no exceptions.

Goal

The goal of this assignment is to explore the use of double-indirect jumps to implement polymorphic dispatch and switch statements. You will implement two new instructions, observe the behaviour of two snippets, and crack another mystery program.

Extending the ISA

You will implement two additional instructions.

Instruction	Assembly	Format	Semantics
jump double ind, b+disp	j *o(rs)	D*spp*	pc ← m[r[s] + (o == pp*4)]
jump double ind, index	j *(rs,ri,4)	E*si*-	pc ← m[r[s] + r[i]*4]

Code Snippets Used this Week

As explained in detail below, you will use the following code snippets this week. There are C, Java, and SM213 Assembly versions for each snippet (except the `SB-switch`, for which there is no Java version).

- `SA-dynamic-call`
- `SB-switch`

Requirements

Here are the requirements for this week's assignment.

1. Implement the double-indirect jump instructions listed above and extend your test program to test them.
2. Execute snippets `SA-dynamic-call` and `SB-switch` in the simulator, step by step. Carefully examine their behaviour and document the key changes you see to the register-file and memory.
3. Execute the SM213 program `A6.s` to determine what it does. Explain its behaviour by both giving an equivalent C program and by explaining in plain English what simple computation it performs.

Material Provided

The snippets and the mystery program are provided in the file `code.zip`.

What to Hand In

Use the **handin** program. The assignment directory is **a6**. Please hand in exactly the following files with the specified names. Do not hand in class files, or your entire Eclipse project, or a README in formats like `.doc` or `.rtf`.

1. `CPU.java` with the two additional double-indirect jump instructions implemented.
2. `test-lab6.s` that tests all newly implemented instructions.
3. `A6.c` that gives equivalent C code for `A6.s`, as specified in Requirement 3.
4. `README.txt` that contains:
   - header with your name, student number, four-digit cs-department undergraduate id (e.g., the one that's something like a0b1)
   - statement that "I have read and complied with the collaboration policies" at http://www.ugrad.cs.ubc.ca/~cs213/winter11t2/policies.html
   - Description of your observation during the execution of `SA-dynamic-call`, as specified in Requirement 2.
   - Description of your observation during the execution of `SB-switch`, as specified in Requirement 2.
   - Description of `A6.s` (i.e. what it does) in plain English language, as specified in Requirement 3.