CS520, Spring 2018: Homework 3

Due: ??
Cutoff: ??

Objectives

- Fork the CpuSimulator repo: https://github.com/theosib/CpuSimulator
- Add the following parallel functional units that are directly fed instructions from the Decode stage:
    - Execute (single cycle ALU operations; already there)
        - Add FOUT instruction that prints a float value
    - Memory (already there, but extend to 3 pipeline stages)
    - Integer multiply (4 cycle pipeline; see examples/MultiStageFunctionalUnit.java)
        - MUL instruction (produces lower 32 bits of product; already in example)
        - MULU instruction (upper 32 bits of product)
    - Integer divide (16 cycle NON-pipelined; use counter and setResourceWait())
        - DIV instruction (quotient)
        - MOD instruction (remainder)
    - Floating point add/sub (6 cycles pipelined)
        - FADD instruction (see float/int conversion notes below)
        - FSUB instruction
    - Floating point multiply (6 cycles pipelined)
        - FMUL instruction
    - Floating point divide (16 cycles NON-pipelined)
        - FDIV
- Note that you now have 7 units being fed by Decode, 7 units feeding results to Writeback, and 7 forwarding sources.

Rules about your submission

- You should use the code from https://github.com/theosib/CpuSimulator as your starting point. You shouldn't need to copy anything from HW2.
- Keep all of your changes confined to the implementation directory
- Make sure your bitbucket repository is not public and shared with only the instructor and graders.

Rules about your processor architecture

- Registers may only be read in Decode.
- Registers may only be written in Writeback.
- Main memory may only be read and written in Memory.
- Instructions may only be retrieved from the program file in Fetch. Any other places where you need the instruction (including forwarding) should come from pipeline registers.

Tips and suggesions

Substantial change have been made to the infrastructure. There are copious comments in the utilitytypes directory, particularly for the interface classes. The code checked in is a complete working implementation of HW2. There are also examples and some pre-made modules like a pass-through pipeline stage, a multi-stage pass-through, one-in-one-out and many-in-many-out examples, multistage functional unit example, and more.

Converting between integer and floating point

Your simulator will support single-precision (32-bit) floating point. However, the register file and instructions all represent numbers as ints. Casting between int and float will perform a numerical conversion. However, what we need to do instead is use ints to store the binary representation of floats. Java's Float class has methods for this.

To store a float value's binary representation in an int, use this method:

```
public static int Float.floatToRawIntBits(float value)
```

To extract a float from an int, use this method:

```
public static float Float.intBitsToFloat(int bits)
```

For example, to extract two floats stored in ints, multiply them, and then store the result in an int, do the following:

```
float a_float = Float.intBitsToFloat(a_int);
float b_float = Float.intBitsToFloat(b_int);
float c_float = a_float * b_float;
int c_int = Float.floatToRawIntBits(c_float);
```

Test Code

To be provided (Newton-Raphson approximations of square root and sine).