Assembler and Simulator

We are providing an assembler and a simulator for you to generate test programs and to verify your program's behavior. We will not be providing true test files for you. The assembler and simulator are available on Sakai (within assembly.zip on the HW assignment page). These are very limited tools (e.g., no hex values for constants - only integers). We have tested the assembler on Linux machines (i.e., teerXX.oit.duke.edu). You will have to copy the generated memory image files to your own machine, or you can port the assembler to whatever machine type you have.

The simulator is useful for debugging your design. Note that using the verbose flag of the simulator will spit out every instruction executed as well as the correct contents of every register—this is very helpful during debugging.

There are two pseudo-instructions available for use in your programs:

1) ldia $rs, label   # load address
2) halt

The ldia pseudo-instruction is converted into several actual machine instructions that have the effect of loading a 16-bit address into the specified register. The halt instruction is actually a branch that simply branches back to itself, creating an infinite loop.

Setting Up Assembler and Simulator

1. Copy asm.cpp, sim.cpp, format.h, and Makefile onto a Teer machine.
2. Run make (i.e. type 'make'). This will compile the source code for the assembler and simulator. You should now see asm and sim show up in the folder.

Using the Assembler

1. Write the assembly program. See simple.s for an example on how to write one. Remember that you can only use the instructions defined in the instruction set, not other instructions. Also, you can use the pseudo-instructions 'ldia' and 'halt'.
2. Run it through the assembler (./asm filename). The assembler should generate three files: one with .sim extension for the simulator, one with .dmem.lgsim extension for data memory, and one with .imem.lgsim extension for instruction memory.
3. Open your processor in Logisim.
4. Load the instruction memory. To do this, right click on your ROM and click Load Image. Find the .imem.lgsim file.
5. Load the data memory (right lick on RAM and load the .dmem.lgsim file).
6. Start the clock (i.e. run the program)

Using the Simulator

When you run your code through the assembler, the assembler will generate a .sim file to use for the simulator. To run it, type './sim file.sim'. Use the -n option (./sim -n file.sim) to display the number of dynamic instructions executed. Use the -v option (./sim -v file.sim) to turn on verbose output which will spit out every instruction executed and the values of all the registers after each instruction. You may consider dumping the output into another file to analyze (i.e. ./sim -v file.sim > outputfile) or using less (./sim -v file.sim | less).

Sample verbose output:

```
000a 1fc1 addi $r7,$r7,1
0000 00fd 0000 0000 0000 0000 0000 0001
```

000a is the PC. 1fc1 is the instruction. The next line contains 8 values representing the values of the registers.

Note that the output instruction will actually output a character and the input instruction will wait until a character is inputted.

Helpful hints

If you end your program with the halt pseudoinstruction (halt takes no arguments), it will cause an infinite loop when you run it in the processor. This prevents your processor from trying to execute instructions after your program has ended. Additionally, the simulator will stop when it reaches the halt pseudoinstruction.