Algorithm for Designing an ISA
Step 1 -- Identify constraints

- Instruction word size (17 bits)
- Target code you need to run (Fib() + SuperGarbage())
- Should be vaguely RISC-like
- Should maybe be innovative in some way.
Step 2 -- Construct a “straw man” architecture

• Make some preliminary decisions
  • How many registers?
  • How many operands?
  • How many opcodes?
• Write down a preliminary ISA
  • A list of instructions
Step 3 -- Evaluate it

- Try it out
  - Go through the simple code fragment we looked at in class (e.g., $a = b + c$).
  - Look at Fib() and SuperGarbage(), too.
  - Try to code them up in your ISA.
- What was hard? What was easy?
- How did the design decisions you made affect how you wrote the code?
- Did you have all the instructions you needed?
- Was any thing cumbersome or needlessly complex?
- Were all your instructions useful?

- Think about it critically
  - Is it RISC-like?
  - Is it orthogonal?
  - We’ve forced you to make hard choices by restricting you to 17 bits, so it won’t have every good property we’ve discussed in class.
  - Could it be more efficient?
  - More elegant?
Step 4 -- Revise

- Are there big things to fix?
  - Try to fix them. Go back to Step 3
- No?
  - Go to step 5
Step 5 -- Design a “straw man” calling convention

- Code up a simple function call.
- Code up fib().
- Refine and repeat as you did in steps 3 and 4.
Step 6 -- Details

• Assign the opcodes (i.e., add = 0x4, sub = 0x7, etc.)

• Write out the RTL for all the instructions.
Step 7 -- Build Your Simulator

- Instantiate the architectural state for your processor in the simulator class (the PC, register file, etc.).
- Translate the RTL for each instruction into Java so it will work in the simulator.
- See the project page, the simulator infrastructure page, and the slides from the tool overview for details. (They are posted on Thursday the 13th)
- Hand-assemble some code into a binary to test it.
- The same assembly will be your test case for the assembler + simulator.
Step 8 -- Build the Assembler

- Add support for each instruction format to the assembler framework.
- See the project page, the simulator infrastructure page, and the slides from the tool overview for details.
- Implement a pseudo instruction (Probably “load immediate”)
- Test the assembler on the code you hand assembled in Step 7.
- Write test cases for all your instructions. Assemble them, run them, and be sure they pass.
- Try some larger programs, building up to Fib() and SuperGarbage().
Step 9 -- Consider the Implementation

• If you are in 141L:

• Sketch out the data path for your ISA

• What were the consequences of the decisions you made in designing the ISA?

• Could you simplify the hardware by changing the ISA?

• How complex do you think the control is going to be?

• Make any changes that you think would improve your ISA and the hardware it requires. Repeat until satisfied.