CSE 471 Autumn 2001

Computer Design and Organization
Assignment #4

Due: Wednesday November 21

In the previous assignments you were given strict guidelines on how to conduct experiments. This assignment is more open-ended on purpose. Its goal is to help you gain some further understanding on the effects of various architectural parameters of superscalar processors on the resulting CPI.

You can do this assignment in groups of two under the same constraints as previously, i.e., with a new partner. Turn in a report with a structure similar to the one you did for Assignment #2.

For this assignment you should use the simulator found in **HW4/simplesim-3.0**. This simulator has several new features that are described in the file **hw4_readme.txt**. You should use an integer and a floating-point SPEC95 benchmark, respectively:
Integer: **perl charcount all_gre_words**
Floating-point: **swim < swim.in**

In addition, the command line option **-max:inst** allows you to finish the simulation after a set number of instructions. **Be sure to set it up for swim** since otherwise it will take hours, if not days, to simulate it in its entirety – it has several billions of instructions.

For example, if we wanted to simulate 20 Million instructions of the integer benchmark, a typical command line would be:

```
HW4/simplesim-3.0/sim-outorder -config ~/myconfig -max:inst 20000000
spec95-little/perl.ss inputs/charcount inputs/all_gre_words >& ~/myout
```

As a basis, you should be using the configuration file that you used in Assignment #2 with a 2-level GAg correlated branch predictor with 10 bits of history (option *2lev*) and the "xor" option turned on (i.e. a *gshare* predictor).

1. Find what is the reasonable number of instructions that should be simulated for each of the two benchmarks. Be sure to explain how you came up with your reasonable number.

2. Simulate both benchmarks on an in-order machine increasing the issue width from 1 to 8 (by powers of 2) while keeping the number of functional units the same as in Assignment #2. Note that you may have to change more than one

parameter to change the issue width. In particular the commit bandwidth can be changed in the configuration file although it is not so documented (use "-commit:width number"). What do your results tell you about the ability of an in-order superscalar processor to use functional units?

3. Repeat the same experiment for an out-of-order processor. Answer the same questions. Comment on the importance of issue width for in-order and out-of order processors.

In the following, we will only look at out-of-order processors. For issue widths 2 and 4, we are going to increase the number of functional units so that they are a better match for each issue width.

4. Assume that you can add one functional unit, i.e., either an integer ALU, or a multiplier ALU, or a floating-point ALU, or a floating-point multiplier to your processor. Looking at the results of your experiments for answering Question 3, which one would you choose for issue width 2? Simulate and show the improvements in CPI (or IPC).

5. Repeat Question 4 for issue width 4.

6. Considering issue width 2, is there any additional functional unit that can be added and that would provide a *significant* improvement in CPI? Which experiments did you run to answer this question and why?

7. Considering issue width 4, what functional unit(s) would you add to the base configuration (that of Assignment #2) in order to have significant performance improvement. You are not allowed to have as many functional units as you wish: you are limited to an additional budget of cost 3 where an integer ALU costs 1, a floating-point ALU costs 2, and the multipliers (integer and floating-point) cost 3 each. Justify your answer, i.e., which experiments you run and why.