Assignment 3

Problems 1-4 are due in the dropbox on mycourses by Friday 04/13/12 @ 10pm.

Problem 5 is due in the dropbox on mycourses by Tuesday 04/17/12 @ 10pm.

1) (5p) Problem B.2 – Solve for a two-way set associative cache. Below are the solution examples for direct and four-way set associative cache organizations. Create a similar table for your solution.

Cache Block	Set	Way	Possible Memory Blocks
0	0	0	M0, M8, M16, M24
1	1	0	M1, M9, M17, M25
2	2	0	M2, M10, M18, M26
3	3	0	….
4	4	0	….
5	5	0	….
6	6	0	….
7	7	0	M7, M15, M23, M31

Direct-mapped cache organization.

Cache Block	Set	Way	Possible Memory Blocks
0	0	0	M0, M2, …., M30
1	0	1	M0, M2, …., M30
2	0	2	M0, M2, …., M30
3	0	3	M0, M2, …., M30
4	1	0	M1, M3, ….., M31
5	1	1	M1, M3, ….., M31
6	1	2	M1, M3, ….., M31
7	1	3	M1, M3, ….., M31

Four-way set-associative cache organization.

2) (5p) Read the paper about CACTI posted on mycourses, and submit a ~500 words or less (~<1page) summary for it. Address the following:
   a. What is the paper talking about
   b. What did you find out new that you didn't know before
   c. What are the limitations of the model

3) (15p, 5p/each) Solve 2.8 a, b, and c. Print to pdf the results of your CACTI results, and append to your solution.

4) (10p, 5p/each) Solve 2.10 a and b. Print to pdf the results of your CACTI results, and append to your solution.

5) Project 3 (60p):

    a. Complete the implementation of the RAW-data and control dependencies/hazards detection hardware. **Save your current processor version for future use!**

    b. Augment your DP with a multiplier and divider units. These should have latencies greater than one pipeline cycle. As I showed in class, the multiplication unit could have a latency of ~4 cycles, and the divide unit of ~8 cycles. If you implement using:

        i. Schematic: use the wizard to create the units and select the above latencies.

        ii. VHDL or Verilog: use the wizard to create the units, and then generate the HDL code. Then, embed these in your design.

        iii. For either hardware implementations above, you can also choose to implement your own designs, using one of the sequential multiplication and division algorithms.

        iv. For the software based architecture simulator, you can perform the multiplication and division in straight code, but delay the availability of the result as required above.

        v. The multiplication will source its multiplicand and multiplier (both 8-bits values) from two registers of the register file, and write the 16-bits product to the same two registers.

        vi. The division will source its dividend and divisor (both 8-bits values) from two registers of the register file, and write the quotient and remainder (both 8-bits values) to the same two registers.

    c. Similar to the ROB in Tomasulo's algorithm with speculation add an ROB to your DP, which will allow you to execute instructions behind a MUL or a DIV.

        i. The ROB will store the status and results of these instructions.

        ii. The results will not be written back to their destinations until MUL or DIV have completed.

        iii. In case there is a BR in the instruction stream entering the ROB, the results of the instructions behind the BR are written back only if the predicted branch is confirmed to be correct. Otherwise, all those results are flushed.

iv. Although, you'll need random access to each location of the ROB, writing back from the ROB will occur in order, i.e. FIFO, at a rate of one write back per cycle.

v. Dependencies within the instruction sequence following MUL or DIV, or between the latter two and any follow-up instruction(s) will be resolved as designed in part a. This includes stalling, if necessary, and data forwarding, if possible.

vi. If a BR is entering the ROB, and the branch direction is not yet known, say it depends on the outcome of MUL or DIV, you can choose to predict and execute either branch directions.

vii. You can choose to use the ROB continuously, or only when a MUL or DIV are encountered. In the former case, the ROB would be appended at the end of the pipeline instead of the current MEM and WB stages (for a 5-stage pipeline). In the latter case, it would be in parallel with the current MEM and WB stages (for a 5-stage pipeline).

viii. The depth of the ROB is equal to the latency of your divide unit plus 0 or plus 1 depending on how you decide to implement the write back of the MUL or DIV

d. To test your processor with ROB:

i. Assume the following array of 16 elements (all values are in decimal: 1, 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47; these are the first 16 prime numbers.

ii. Convert the following code to assembly:

```
for (i=16; i>0; i=i–1)
        x[i] = x[i] * 53;
for (i=16; i>0; i=i–1)
        x[i] = 53 / x[i];
```

iii. Assemble the code, run and confirm that your processor is working properly. You don't need to collect any performance information at this time.

e. Along with the relevant design files, submit a report in which you describe your design choices and operation of your processor with ROB.