**Computer Science and Engineering 431**
**Computer Architecture**
**Fall 2015**

**Homework #2**

**Distributed**: September 6, 2015
**Due**: September 17, 2015 via Angel by midnight
**Points**: 20

1. (2 pts) The add and shift multiplication design we discussed in class is an iterative multiplier. Consider the design of an (unsigned) 16-bit iterative multiplier implemented with a ripple-carry adder (RCA) that takes 4ps per bit to complete the add. How long would it take to complete a 16-bit multiplication using a RCA worst case? Now assume the 16-bit iterative multiplier is implemented using a CLA that takes log n of the time that the RCA takes. How long would it take to complete a 16-bit multiplication using a CLA worst case?
2. (2.5 pts) Give the representation of the following numbers in IEEE 754 FP Standard single precision **normalized** format (include the hidden bit).

   $-2.25 \times 2^{+16}$

   $0.50 \times 2^{-10}$

   The smallest negative number (closest to zero, negative)

   The largest negative number

   $-\infty$
3. (3.5 pts) You are to modify the MIPS single cycle datapath discussed in class and in the book to accommodate the instruction `jalr` (jump and link register). All other instructions should remain unaffected (i.e., continue to operate as expected). A powerpoint file which contains the (unmodified) single cycle MIPS datapath is proved in Angel under the Homework Assignments tab for your use. You can add your new logic to the ppt picture (please use red so we can easily tell what has been added) and include that in your homework submission (in msword use Insert tab, Picture) or print off the ppt picture, draw your new logic on it (being sure to make it so that we can easily tell what has been added), and scan in the picture to include in your homework submission. What new or expanded (in terms of number of bits) control signals will need to be added, if any? Be sure to indicate what instruction format you are using and how the fields in the instruction are used.

4. (3 pts) For the following instruction sequence:

```
lw   $4, -100($3)
sw   $2, 8($4)
add  $2, $4, $4
```

a. Assuming there is no forwarding hardware support, indicate what data hazards exist and rewrite the code inserting `nop` instructions where needed to eliminate them.

b. Assuming there is full forwarding hardware support, indicate what data hazards exist (if any) and rewrite the code inserting `nop` instructions where needed to eliminate them.

5. (2 pts) For the following instruction sequence:

```
lw   $3, 50($2)
beq  $3, $0, target
```

a. Assuming there is no branch prediction support, that the branch decision hardware is in the EX stage, and that there *is no* forwarding hardware support, how many stalls if any are incurred?

b. Assuming there is no branch prediction support, that the branch decision hardware is in the EX stage, and that there *is* forwarding hardware support, how many stalls if any are incurred?

6. (7 pts) This is the first of a set of performance evaluation experiments you will run using SimpleScalar. First read *The SimpleScalar Tool Set, Version 2.0* document that is available on Angel and then attend one of the SimpleScalar evening tutorials scheduled for 218IST (see Angel email for schedule details).

   In this first set of experiments you will be investigating the impact on the datapath's CPI of instruction and data memories (in SimpleScalar those are the L1I$ (level one instruction cache), the L1D$ (level one data cache), and the unified UL2$ (unified level two cache that holds both instructions and data) with different performances.

   To setup SimpleScalar simulation environment, please follow the "SimpleScalar Lab Setup" which is posted on Angel. There are six SPEC benchmarks you will be experimenting with this semester:

   four integer SPEC benchmarks: `bzip2`, `hmmer`, `mcf`, `sjeng`

   and two floating point SPEC benchmarks: `milc`, `equake`

   The benchmarks can be downloaded from the /home/software/simplesim/ss-benchmark/ and each benchmark is located within a directory. To better organize your homework sets, you will probably want to create a subfolder within each benchmark folder to hold the simulation outputs from each

homework set. For example, for benchmark `bzip2`, you can have several subfolders such as `homework2`, `homework3` … and so on.

For all experiments you run this semester you should fast forward past the first 500,000 MIPS instructions before starting to collect simulation data and then simulate for 2,500,000 instructions. As you run the different experiments, you will probably want to change the default simulation output file name for each benchmark to a descriptive name that somehow captures the simulation experiment being run (for example, output_cachelat, output_branch).

Although the tool set contains several simulators, we will only be using sim-outorder this semester. Most of the settings can be left to the default values specified in the manual (e.g., the cache size settings on page 6). For this homework, the ones you should change from their default values are the fetch, decode and issue queue sizes, the number of integer and floating point ALUs, the main memory latency, and the branch prediction strategy. For example, the **nondefault** settings for the first set of simulation experiments (one simulation run per benchmark) are indicated in bold in the sample sim command line for running the `mcf` benchmark below

```
sim: command line: /home/software/simplesim/simplesim-3.0/sim-outorder
-fastfwd 500000 –max:inst 2500000 –fetch:ifqsize 1 –fetch:speed 1
–decode:width 1 –issue:width 1 –ruu:size 2 –lsq:size 2 –res:ialu 1
–res:imult 1 –res:memport 2 –res:fpalu 1 –res:fpmult 1
–issue:inorder true –issue:wrongpath false -cache:il1 il1:128:64:1:l
-cache:il2 dl2 -cache:dl1 dl1:256:32:1:l -cache:dl2 ul2:1024:64:2:l
–cache:dl1lat 1 –cache:il1lat 1 –cache:dl2lat 5 -cache:il2lat 5
–mem:lat 100 10 –mem:width 8 –tlb:lat 30 –bpred nottaken
mcf_base.i386-m32-gcc42-nn inp.in
```

This sets the simulator to fetch one instruction per cycle and execute them in-order, i.e., our simple MIPS single issue pipelined processor.

For your first set of experiments (the first machine), run all six benchmarks at this setting and collect the report data (specified below).

For the second set of experiments (the second machine), change the performance of the L2 cache latency to 10 (all other settings should remain the same as in your first set of experiments).

```
–cache:dl2lat 10 -cache:il2lat 10
```

For the third set of experiments (the third machine), change the performance of the L2 cache latency to 15 (all other settings should remain the same as in your second set of experiments).

For each set of simulations (a set are the three runs for a benchmark with the three different memory speed settings (three different machines)), report in tabulated form the total number of instructions committed, the CPI, the Il1

cache miss rate, the Dl1 cache miss rate, and the UL2 cache miss rate. What conclusions can you draw from your simulation output?

Now assume that the clock rate for the original machine is 1GHz, for the second machine is 1.5GHz and for the third machine is 2GHz. Compute the geometric mean of the performance for each of the three machines (one mean per machine for the integer benchmarks and one mean for the floating point benchmarks). Which machine has the best performance for the integer benchmarks? Which has the best performance for the floating point benchmarks?

Also turn in a hardcopy of the output from *one* of your simulation runs – the one with L2 cache latencies set to 15 for `bzip2`.