1. (6 points)

Assume that in a particular program, 10% of the instructions are branches. The remaining 90% of the instructions has a CPI of 1.2. Consider a microarchitecture design A that does not have a branch predictor and has to expend 2 cycles for every branch instruction. Determine the CPI of design A. Consider another design B that has a branch predictor with an accuracy of 90%. While a correct prediction (in B) requires only 1 cycle, a misprediction requires 3 cycles for the branch to execute. Determine the CPI of design B. Consider yet another design C that has a branch predictor with an accuracy of 70%, and a branch execution time of 1 cycle for correct prediction and 5 cycles for misprediction. Determine the CPI of design C. Which of these designs are the best and the worst?

2. (12 points)

a. (4 points) H&P Problem 2.1 (page 142)

b. (4 points) H&P Problem 2.3

c. (4 points) H&P Problem 2.4

3. (6 points) Introduction to SimpleScalar

(This is an important problem. It is worth more than the points assigned to it as it is a gentle introduction to SimpleScalar, which could be the simulator that you use in your project.)

Using a CS Unix/Linux machine, download and install the Simplescalar v 3.0 Source code from http://www.simplescalar.com/. Descriptions for the Integer Benchmarks and Floating Point Benchmarks can be found at the Standard Performance Evaluation Corporation's website. To run the benchmarks, the command line arguments for each benchmark must be specified after the executables name. The command line options necessary for each benchmark are at /unsup/spec2000/benchspec/benchmark.cmdlines. From there, the associated Alpha executables and data files for the integer benchmarks and floating point benchmarks are in the CINT2000 and CFP2000 directories respectively. You will perform a simple characterization of one of

a. gcc with integrate.i as the input [if your last name begins with A-I]

b. mcf with inp.in as the input [if your last name begins with J-Q]

c. crafty with crafty.in as the input [if your last name begins with R-Z]

using sim-outorder. Use the benchmark-name_base executable for all the benchmarks.

Use the following basic configuration: Two Level Cache:
L1 instruction cache: 64k 2-way set associative with 64 byte lines.
L1 data cache: 64k 4-way set associative with 32 byte lines.
L2 unified cache: 256k 8-way set associative unified L2 cache with 64 byte lines.
Branch prediction: use the default branch prediction settings.

Submit the following. Report the IPC and the cache hierarchy miss rates if:

a. the simulator is run in in-order mode using LRU replacement policy for the cache hierarchy
b. the simulator is run in out-of-order mode using LRU replacement policy for the cache hierarchy
c. the simulator is run in out-of-order mode, changing the cache hierarchy to the following setting:

Two Level Cache using LRU replacement policy:
L1 instruction cache: 64k 4-way set associative with 64 byte lines.
L1 data cache: 64k 4-way set associative with 32 byte lines.
L2 unified cache: 256k 4-way set associative unified L2 cache with 64 byte lines.

For all of the above tests, fast-forward 100M instructions, and simulate for next 50M instructions.