CPE315 Lab 5 Report:

Written by: Tyler Holland and Ryan Coonan

1. **If you are building a processor and have to do static branch prediction (meaning you have to assume at compile time whether a branch is taken or not), how should you do it? You can make a different decision for branches that go forward or backward.**

A) If it is a backward branch, I would decide to predict that the branch will be taken. This is because a backwards branch is most likely used to make a loop, so it will almost always be taken. With a forward branch, I would decide to predict that the branch will not be taken, because a forward branch will be skipping instructions if it is taken.

2. **How good is the GCC MIPS compiler in filling the branch delay slot?**

   A) Using the -O1 optimization compared to no optimization, the GCC MIPS compiler does very well with filling the branch delay slot. With -O1 optimization, there were 52756579 useful instructions after branches, and only 5256482 useless ones. That means 90% of the instructions in the branch delay slot were useful, which is a good enough number for me.

3. **How good is the GCC MIPS compiler in avoiding load-use hazards?**

A) Using the -O1 optimization compared to no optimization, the GCC MIPS compiler does very well with avoiding load-use hazards. With no optimization, there were 1793266 load use hazards, but with O1 optimization there were 0. That is a 100% improvement, which is as good as it gets.

4. **If you are building a 256-byte direct-mapped cache, what should you choose as your block (line) size?**

A) I would use a block size of 64 bytes. With the O1 optimization, the 256 byte cache with block size of 64 bytes had the highest hit rate with 84.0797% hits. This is compared to block sizes ranging from 2^2 to 2^8.

5. **What conclusions can you draw about the differences between compiling with no optimization and -O3 optimization?**

A) With no optimization, the run time of shang took around 10 minutes. With O3 optimization, the run time was around 6 minutes. Therefore, you get about a 40% improvement in speed using O3 optimized code over non-optimized code.