Names: Tim Newman, Josh Conrad

Github link: https://github.com/imanewman/cpe315finalproj

CPE 315 Final Project Write up

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.

For backward branches, our data supports that it would almost always be better to predict the branch is taken, as many of the backward branches are due to loops (depending on implementation). Forward branches on the other hand are much less predictable, but based on the data from the 3 levels of shang optimization, it would also be better to predict these branches are taken.

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

With our data, Shang O0 had the best hit rate with an 8 byte block size, Shang O1 with a 32 byte block size, and Shang O2 with a 32 byte block size. The hit rate seems to typically decrease as you move away from the 32 byte block size in either direction, so that would be the best choice in most situations.

3. What conclusions can you draw about the differences between compiling with no optimization and -O2 optimization?

One significant difference is that the average cache hit rate is higher regardless of block size for no optimization than with O2 optimization. The O2 optimization also shifts the distribution of forward and backward branches to having more forward branches overall, with a much higher amount (nearly a third) not being taken compared to no optimization. There is also a shift in the ratio of backward branches taken to not taken, with the O2 optimization having a higher percentage of the backward branches being taken then before. In the O2 optimization, there is nearly 1/5th the amount of dynamic instructions as there are with no optimization. The same large reduction is seen in memory read and writes, as well as register read and writes. Below is a

chart detailing the differences between the 3 Shang levels. Thus, overall, the optimization vastly increases performance by greatly decreasing both the number of instructions and number of memory accesses.

Shang Data	O0	O1	O2
Dynamic Instructions	1326980	305135	261935
Memory Reads	354001	45582	44342
Memory Writes	73058	28950	27710
Register Reads	2967987	727978	613428
Register Writes	2512287	521319	439949
Forward Branches Taken	9010	8651	8001
Forward Branches Not Taken	461	11220	5210
Backward Branches Taken	41650	37569	39109
Backward Branches Not Taken	9741	1681	1801
256B Cache 4B blocksize	377632 hits, 30243 misses (hit rate: 92.5852%)	52096 hits, 22354 misses (hit rate: 69.9745%)	49918 hits, 22052 misses (hit rate: 69.3595%)
256B Cache 8B blocksize	383482 hits, 24393 misses (hit rate: 94.0195%)	59712 hits, 14738 misses (hit rate: 80.2042%)	57560 hits, 14410 misses (hit rate: 79.9778%)
256B Cache 16B blocksize	382199 hits, 25676 misses (hit rate: 93.7049%)	62343 hits, 12107 misses (hit rate: 83.7381%)	60231 hits, 11739 misses (hit rate: 83.689%)
256B Cache 32B blocksize	371018 hits, 36857 misses (hit rate: 90.9637%)	63907 hits, 10543 misses (hit rate: 85.8388%)	61685 hits, 10285 misses (hit rate: 85.7093%)
256B Cache 64B blocksize	347467 hits, 60408 misses (hit rate:	63320 hits, 11130 misses (hit rate:	60971 hits, 10999 misses (hit rate:

	85.1896%)	85.0504%)	84.7172%)
256B Cache 128B blocksize	296780 hits, 111095 misses (hit rate: 72.7625%)	47661 hits, 26789 misses (hit rate: 64.0175%)	45232 hits, 26738 misses (hit rate: 62.8484%)
256B Cache 256B blocksize	242527 hits, 165348 misses (hit rate: 59.4611%)	11798 hits, 62652 misses (hit rate: 15.8469%)	9715 hits, 62255 misses (hit rate: 13.4987%)