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.**

You should assume forward branches are not taken and backwards branches aren't. In for loops such as for(i = 0; i < 100; ++i), the forward branch out of the loop will only be taken once out of 100 times. But the branch to go back to the start of the loop is the reverse.

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

We had the most success for a block size of 64 bytes. This caused the most hits because more data is being stored per block. Because of this, we don't have to access main memory as often, speeding up the program. It's better than having block sizes of 128 and 256 because there are more blocks.
More hits = less miss. WOW!!

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

Having an -O2 optimization will unroll loops by a level of 2. This will minimize the number of backwards branches. This helps with branch prediction and fewer branches are mispredicted so efficiency is improved.