CIS501 HW1b

Zi Yan
PennID: 14137362

1 Total speedup

2 Relative speedup

- -O3 is 3.85045 times faster than -O0.
- unrolled is 1.07246 times faster than -O3.
- unrolled & vectorized is 1.77025 times faster than just unrolled.
- "-fopenmp" is 2.14650 times faster than unrolled & vectorized.

3 Static instruction counts

- -O0: 15 instructions.
- -O3: 7 instructions.
- unrolled: 42 instructions.
- unrolled & vectorized: 42 instructions.

4 Dynamic instruction counts

- -O0: $65536 \times 15 = 983040$ instructions.
- -O3: $65536 \times 7 = 458752$ instructions.
- unrolled: $8192 \times 42 = 344064$ instructions.
- unrolled & vectorized: $2048 \times 42 = 86016$ instructions.

5 Static vs. dynamic instruction counts

The most optimized .s file (unrolled & vectorized) is 3 times as many static instructions as the less optimized one (-O0). However, the most optimized file has only 0.0875 times dynamic instructions of the less optimized one.

Apparently optimization increases the static instructions, but it decreases the iteration times of the whole loop, therefore, the dynamic instructions are decreased.

6 Performance estimate based on dynamic instruction count

Because CPI is 1, the number of clock cycles a program runs is the very dynamic instruction number.

- -O3 is 2.14286 times faster than -O0.
- unrolled is 1.33333 times faster than -O3.
- vectorized is 4 times faster than -O3 + unrolling.

7 Loop unrolling pro/con

One advantage: The program runs faster.
One disadvantage: The size of the program doubles.

8 Estimated vs. actual performance

Except the performance improvement of -O3 over -O0, all the others are high than the actual performances.

9 CPI calculation

- -O0: $\text{CPI} = 3,000,000,000 \times 45.76 \div (983040 \times 200000) = 0.6982$.
- -O3: $\text{CPI} = 3,000,000,000 \times 11.88 \div (458752 \times 200000) = 0.3884$.
- unrolled: $\text{CPI} = 3,000,000,000 \times 11.08 \div (344064 \times 200000) = 0.4830$.
- unrolled+vectorized: $\text{CPI} = 3,000,000,000 \times 6.26 \div (86016 \times 200000) = 1.0917$.

10 CPI and compiler optimizations

As more optimizations are enabled, the CPI goes down then climbs up.

From -O0 to -O3, the compiler uses more efficient instructions for array multiplication and addition, like movss, mulss, addss, and removes the 32-bit instructions, like movl, cltq, so that CPI is decreased.

From -O3 to unrolled, to unrolled+vectorized, the compiler continuously adds more complex instructions, such that the CPI is decreased by those instructions, because those may take more clocks to execute.