

15-740/18-740 Computer Architecture

Homework 2

Due Wednesday, September 21, at 12:00 PM

1. Amdahl's Law I

A particular program P running on a single-processor system takes time T to complete. Let us assume that 40% of the program's code is associated with "data management housekeeping" (according to Amdahl) and, therefore, can only execute sequentially on a single processor. Let us further assume that the rest of the program (60%) is "embarrassingly parallel" in that it can easily be divided into smaller tasks executing concurrently across multiple processors (without any interdependencies or communications among the tasks).

- Calculate T_2 , T_4 , T_8 , which are the times to execute program P on a two-, four-, eight-processor system, respectively.
- Calculate T_∞ on a system with an infinite number of processors. Calculate the speedup of the program on this system, where *speedup* is defined as $\frac{T}{T_\infty}$. What does this correspond to?

2. Amdahl's Law II

Amdahl compares and contrasts the performance of three different machines in his paper (Machines A, B, C). For this problem, consider Machines X, Y, Z configured as follows.

- **Machine X** A scalar processor with 1 arithmetic unit, running at frequency $2f$.
- **Machine Y** An array processor with 4 arithmetic units, running at frequency f .
- **Machine Z** A VLIW processor with 4 arithmetic units, running at frequency f .

Additionally, we define instruction-level parallelism as the degree in which an application's instructions are independent of each other and, therefore, can be executed concurrently.

- If a large portion of an application's code has very low instruction-level parallelism, on which machine would it run the fastest, if any, and why?
- Describe the characteristics of an application which would perform better on Machine Y than on Machine Z.
- Describe the characteristics of an application which would perform better on Machine Z than on Machine Y.

3. Moore's Law

- At each generation, silicon fabrication technology is commonly referred to by its feature size (or technology node), which can be loosely thought of as the smallest resolution at which transistors can be "drawn" on a silicon die (called "dimensional tolerances" by Moore.) Cutting-edge silicon chips of today are fabricated using a feature size of 32nm. Two years ago, it was 45nm. Four years ago, it was 65nm. According to these three numbers, is the Moore's Law still in effect? Explain why or why not.

- (b) Consider two approaches of doubling the number of transistors: halving the size of a single transistor while maintaining constant die area (Moore's Law) versus maintaining the size of a single transistor while doubling the die area. List some reasons why the first approach is superior to the second approach.

4. Compiler Potpourri

- (a) Wulf motivates his principles of architecture design from a cost perspective. Yet in most implemented architectures, where cost is a key design constraint, it is possible to find many violations of Wulf's principles. Why do you think this is so? Pick three principles and list the trade-offs of abiding by each of them in terms of, for example, performance, power, reliability, and design and verification cost.
- (b) At the time, Wulf claimed that "designing irregular structures at the chip level is *very* expensive." With an abundance of transistors available on-chip today, do you think this statement still applies? In what cases would it make sense and what cases would it not?
- (c) Colwell et al. discuss some of the merits and definitions of RISC versus CISC architectures. In your opinion, what are the key strengths and weaknesses of RISC architectures compared to CISC architectures, as defined in the paper?
- (d) Colwell et al. also describe the concept of the "semantic gap"—a notion of how easily higher level language concepts can be expressed in the instruction set. Consider an ISA which tries to shorten the semantic gap with a `FOREACH x y` instruction which takes the location of a null-terminated array, `x` and the location of some code to execute on each element of the array, `y`. What are the trade-offs for the programmer, compiler designer, and hardware designer, of implementing such an instruction? What do you think Wulf would have to say about such an instruction (specifically, which of his principles might it violate)?

5. SIMD and Multi-Threading

List some examples of when a SIMD engine would be more beneficial than multi-threading. When would multi-threading be more beneficial to use than a SIMD engine?