Coursework 4

Due: May 16, 2011

Write an essay of between 2000 and 3000 words on one of the topics from the list below. You may select a topic that has the same number in in base-5 modulo arithmetic as your student number. For example, if your student number is 12345, you may pick from topics 5, 10, 15, and 20. If your student number is 12346, you may pick from 1, 6, 11, and 16.

1. Discuss the increasing use of commodity hardware for high-performance computing.
2. Give five examples of design patterns for scalable programs and discuss their advantages, disadvantages, and some example problems where they are applicable.
3. Compare the architecture of a recent nVidia GPU to the Cray 1.
4. What design trades are made when designing a microprocessor, and what are the characteristics that a 'general-purpose' CPU is optimised for?
5. Give a detailed description of the SETI@HOME project, including the algorithms used for distributing the work and why they are not applicable in general to high-performance computing.
6. Give some examples of optimisations that can make code faster on a typical CPU but slower on a GPU (or vice versa), and why they have this characteristic.
7. Look at the transistor counts of x86 chips since the 8086 and work out how many of each generation could be squeezed onto the same die size as a recent Intel chip. Discuss the kind of algorithm that might benefit from this approach for several versions.
8. Dedicated hardware for a specific algorithm is usually faster than a pure software implementation. Give some examples of hardware acceleration found in modern chips and its performance characteristics. Discuss the benefits and costs associated with this approach.
9. What is meant by the term 'latency hiding'? What programming techniques are good at hiding latency, and why is it important?

10. Compare OpenMP and MPI. What are the advantages and disadvantages of each approach?

11. Look at the libdispatch library from Apple. What costs that are commonly associated with multithreaded programming does it avoid, and how?

12. What effect does the existence of a cache have on algorithm design?

13. In some cases, running an algorithm on multiple processors is slower than running it on just one. What kind of algorithm exhibits this property, and why? How would you avoid it?

14. Compare Erlang and Go. How do their approaches to concurrency differ, and what kinds of problems are each good at?

15. Languages like C are low-level languages because they expose an abstract model similar to the concrete model of the computers for which they were designed (e.g. PDP-11). What features would you expect to find in a modern low-level language?

16. Discuss the difficulties of implementing transactional memory in hardware and in software, and the advantages of each approach.

17. What are 'wait-free' algorithms? Give some examples and the kind of problems that they solve.

18. If you had a large legacy codebase, what steps would you take to allow it to advantage of multicore processors, without requiring a complete rewrite?

19. Give some examples of common bugs in multithreaded code. What techniques would you use to avoid them and to identify them if they are present?

20. Modern computers are increasingly focussed on power efficiency, rather than raw performance. Discuss how the focus on power usage has impacted modern processor and software design.