CISC 810: Fundamentals of Computational Science, Assignment 1

Set September 20th due Oct 4th

Notes: Questions that begin with "Research" may require you to look up auxiliary information outside of the lectures notes in class. To make this process easier, I have ensured that all the information you need can be found rapidly by Internet searches.

Q1. (Research) (a) The Cray supercomputers of the 1970s and 80s were "vector-pipeline" computers. These machines have sophisticated, and expensive, memory subsystems that allow very high bandwidths (for the technically interested, they are built using SRAM rather than DRAM). Each CPU has *vector registers* which perform the same function as general registers on scalar CPU. The vector CPU then operates on a pair of registers by *pipelining* the operations through the floating point calculation unit. For example, suppose you want to add two vectors **A** and **B**. The elements of the vectors (a_i and b_i) will be added together with all the addition operations being pipelined through the floating point unit. Classify the nature of these computers (there are many websites detailing their operation if you wish to know more) under Flynn's taxonomy and explain your answer in detail (HINT: pipelining is the key here and wikipedia is misleading).
(b) Modern Intel and AMD CPUs are equipped with "streaming SIMD extensions" known by the acronym SSE. Are these true SIMD operations under the Flynn's taxonomy definition? Explain your answer in the context of how the SSE instructions work.

Q2. (Research) Find the approximate clock speed in GHz of the Intel Pentium 4 for each year of its release since 2000 (Intel has a processor "quick reference guide" on their website). Don't worry about all the different models, focus on the clock speed and when it first appeared. Perform a similar analysis for the AMD Opteron (which was launched in 2003). In both cases compare only single core versions of the processor. www.spec.org provides the relative performance levels of these processors as measured by the "SPEC" benchmarks. The CFP2000 result (fp="floating point" on the year 2000 benchmark) gives a measure of the relative floating point speed of each processor on this benchmark. You will need to collate the CFP2000 results for each of the processor models to answer the following questions.
(a) Plot a graph of the clock speed versus year for both processors. How would you classify the growth of the clock speed versus year? How does this compare with the growth predicted by Moore's Law?
(b) Plot up the clock speed in GHz versus the CFP2000 base value. Of the two processors, which gives more performance per clock cycle? Estimate the approximate improvement in cycles per instruction for the superior processor at the highest clock speeds available.
(c) Given that it is exceptionally difficult to push clock speeds above 4 GHz with current technology, and that we are not expected to see significant improvements in the immediate future, as an Intel designer what would your next generation of processors need to do to improve performance? Contrast the Pentium 4 clock speeds to those of the new Intel Core 2 processors.

Q3. (a) Describe how caches help improve the speed of processors. If caches are so beneficial why aren't systems built with huge caches? (HINT: one issue relates to the nature of algorithms, another to economics)

Let's assume that we have a *direct mapped* cache system where the cache lines are a single memory

words, and there are a total of $2^4 = 16$ lines within the cache (from 1 to 16). Initially there are no memory words stored in the cache, and an algorithm requests the following memory addresses: 1,2,17,13,41,52,2,9,17,33,34,2,1,51,61.
(b) For each of these memory addresses work out the binary address in memory and, using the cache mapping formula given in the lectures, the binary number of the cache line the memory word is assigned to. Also state whether the memory request is held in the cache (a *cache hit*) or in main memory (a *cache miss*). It is probably easiest to give your answer in table with headings: Address, Binary address, Binary cache block, hit or miss.

Q4. We talked about how good cache reuse does not have to ensure high performance. Suggest an algorithm that accesses memory values and then performs calculations, that while having good cache reuse might not be expected to show good performance. (HINT: accesses of memory words in random locations are always slow for a piece of memory that is significantly larger than the cache.)