Engineering 9859
CoE Fundamentals — Computer Architecture
Introduction

Dennis Peters[1]

Fall 2007

[1] Based on notes from Dr. R. Venkatesan

Course Details

Classes Monday, Wednesday, Friday 9 – 10 EN-4033

Course evaluation final exam (60%), assignments (40%)

Problem sets 3, selected problems will be graded.

Plagiarism consult course information sheet

Course notes available at http://www.engr.mun.ca/˜dpeters/9859

Outline

Objective Review basic computer architecture topics and thus prepare students for ENGR9861 (High Performance Comp. Arch.)

Textbook Hennessy & Patterson: *Computer Architecture — A Quantitative Approach*

Syllabus Chapters 1, 2 & 5 in the textbook

- Computer organization
- Measuring performance, speed up
- Instruction set principles. Example: MIPS
- Memory organization: cache, main memory, virtual memory

Computer Organization

Input & Output keyboard, mouse, monitor, speaker, microphone

CPU (micro)processor that includes ALU, registers, internal buses, cache, controller

- **Program counter** contains address of the instruction to be executed
- **General purpose registers** temporary storage (not memory)

Memory unit kernel of the operating system on ROM, higher levels of cache, DRAM (main memory), disks

Network LAN, WLAN, WAN: ethernet, Internet, ATM

Ways to improve performance

- Use faster material: silicon, GaA, InP
- Use faster technology: photochemical lithography
- Employ better architecture within one processor
  - Selection of instruction set: RISC/CISC, VLIW
  - Cache (levels of cache): higher throughput
  - Virtual memory: relocatability, security
  - Pipelining: k stages gives a maximum speedup of k
    - Superpipelining,
    - Superscalar (multiple pipelines) with dynamic scheduling
  - Branch prediction
- Use multiple processors: emphasis of ENGR9861
  - Scalability, level of parallelism
  - Shared memory, array processing, multicomputers, MPP
- Employ better software: compilers, etc.

Speedup

- Any (architectural) enhancement will hopefully lead to better performance. *speedup* is a measure of this improvement.
- Performance improvement should be based on the total CPU time taken to execute the application, and not just any of the component times like memory access time or clock period.
- If the whole processor is replicated, then the fraction enhanced is 100%, as the whole computation will be impacted.
- If an enhancement affects only a part of the computation, then we need to determine the fraction.

Amdahl's Law

Consider a task, T, and proposed enhancement, E:

$$\text{Speedup} = \frac{\text{Total time for T without using E}}{\text{Total time for T using E}}$$
$$= \frac{1}{(1 - \text{fraction enhanced}) + \frac{\text{fraction enhanced}}{\text{Speedup of E}}}$$

$\therefore$ We need to always aim at making enhancements that will affect a large fraction of the computation, if not the whole computation.

Speedup Example

Consider three enhancements, as follows

	Speedup	% enhanced	total speedup
E1	40	20	
E2	20	30	
E3	5	70	

Assuming all cost the same, which is a better choice?

- Higher fraction enhanced is more beneficial than huge speedup for a small fraction.

$\therefore$ frequency of execution of different instructions becomes important — statistics are needed.

CPU (computation) time

CPU time is the product of three quantities:

1. Number of instructions executed or *Instruction Count* (IC): remember this is not the code (program) size.
2. Average number of clock *cycles per instruction* (CPI): if CPI varies for different instruction, a weighted average is needed
3. Clock period (τ)

$$\mathrm{CPUtime} = \mathrm{IC} * \mathrm{CPI} * \tau$$

- An architectural (or compiler-based) enhancement aimed to decrease one of the above two factors might increase one or both of the other two.
- The product of the three quantities after applying the enhancement that gives us the new CPU time.

Measuring Performance

CPU clock speed an indicator, but not sufficient

CPU speed, memory speed, other devices, system configuration
gives better idea, but not sufficient

Metrics

- millions of instructions per second (MIPS),
- billions of floating point instructions per second (GFLOPS),
- trillions of operations per second (TOPS),
- thousands of polygons per second (kpolys),
- millions of network transactions per second.
- Give only a partial picture — they leave out IC

Measuring Performance (cont'd)

Benchmark programs

- real applications: gcc, Tex
- toy benchmarks such as sieve of Eratosthenes, Puzzle
- kernels such as Livermore loops, Linpack
- synthetic benchmarks such as Whetstone, Dhrystone, Dhampstone

Benchmark suites offer collections of the above

- For example: SPECint95 and SPEC CPU2000.
- Performance is compared with a selected standard (using geometric mean), and given as a dimensionless number.