

Performance

Lecturer: Dr. Hui Annie Guo
huig@cse.unsw.edu.au
K17-501F (ext. 57136)

Overview

- **Definitions**
 - Latency and Throughput
- **Components of processor performance:**
 - Instruction count, CPI, clock period
- **Improving performance and limits on performance improvement**
 - Amdahl's Law
- **Benchmarks**

COMP3211/9211

2011S1 wk2_2 P2

Performance

- **Performance has been a critical design issue in computer system design**
- **When we deal with performance, the following questions should be answered:**
 - What do we mean by performance?
 - How do we evaluate performance?
 - What design factors that affect performance?
 - How can we improve performance?

COMP3211/9211

2011S1 wk2_2 P3

Motivation

Airplane	Passenger capacity	Cruising range (miles)	Cruising speed (mph)	Passenger throughput (passengers x mph)
Boeing 777	375	4630	610	228,750
Boeing 747	470	4150	610	286,700
Concorde	132	4000	1350	178,200
DC8	146	8720	544	79,424

- **This example illustrates that there can be different measures of performance: cruising speed, capacity, range**
- **If speed is the desired measure of performance, which airplane is fastest?**

COMP3211/9211

2011S1 wk2_2 P4

How do we quantify performance of computers?

- **Two principle measures:**
 - **Response time/execution time/elapsed time/latency**
 - How long does it take to execute a task? or
 - How long does it take to return a query?
 - **Throughput**
 - How many jobs can be run at once?
 - What is the average execution rate?
 - How much work can be done per unit time?

Unix `time` command

- `% time find . -name myfile`
`90.7u 12.9s 2:39 65%`
 - 90.7u ——— user CPU time in seconds
 - 12.9s ——— system CPU time in seconds
 - 2:39 ——— response time
 - $(90.7 + 12.9)/(2*60+39) = 65\%$
 - 65% of the elapsed time was used by the CPU executing the program – referred to as “utilization”

Response time /execution time

- *Elapsed time = finish time – start time*
 - Counts everything (disk and memory access, I/O etc)
 - Not so good for purpose of comparison \Rightarrow depends upon machine load, location of data
- *CPU time*
 - Doesn't count I/O time or time spent running other programs by the CPU
 - Can be broken up into system time, and user time
 - But system time includes waits & OS overheads
- *Our focus: user CPU time*
 - Time spent executing the lines of code that are “in” our program

What is “performance”?

- **Performance is in units of things executed per second**
 - Higher values intuitively better
- **If we are primarily concerned with response time, the performance of machine X is**
 $performance(X) = 1/execution_time(X)$
- **“X is n times faster than Y” means**
 - $performance(X)/performance(Y) = n$
 - $execution_time(Y)/execution_time(X) = n$

Relative performance example

- If computer A runs a program in 10 secs and computer B runs the same program in 15 secs, how much faster is A than B?

$$\begin{aligned} \text{Perf(A)} / \text{Perf(B)} &= \text{Exec.time(B)} / \text{Exec.time(A)} \\ &= 15 / 10 \\ &= 1.5 \end{aligned}$$

Clock cycles

- Execution time of a program can be expressed as

$$\frac{\text{seconds}}{\text{program}} = \frac{\text{cycles}}{\text{program}} \times \frac{\text{seconds}}{\text{cycle}}$$

- clock rate (frequency) = cycles per second (1 Hz = 1 cycle/sec)

- A 4 Ghz. clock has a $\frac{1}{4 \times 10^9} = 250$ picosecond (ps) cycle time

Exercise

- Our favorite program runs in 10 seconds on computer A, which has a 4 GHz clock.

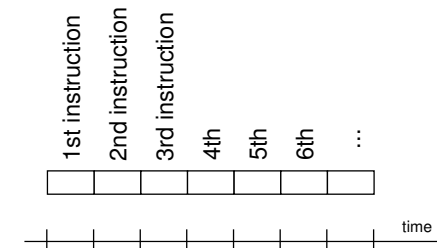
We are trying to help a computer designer build a new machine B, that will run this program in 6 seconds.

The designer can use new (or perhaps better) technology to substantially increase the clock rate, but has informed us that this increase will affect the rest of the CPU design, causing machine B to require 1.2 times as many clock cycles as machine A for the same program.

What clock rate should we tell the designer to target?

How many cycles are there in a program?

- Could assume that number of cycles equals number of instructions



- But this assumption is not necessarily true
 - different instructions take different amounts of time on different machines.

Example

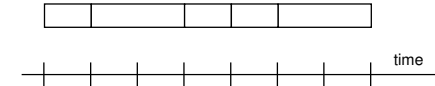
- **HC68000**

- *clr Rd*: 4 cc
- *jmp k*: (8~14) cc
- *mul*: (70~80) cc
- *move*: (9~17) cc

- **AVR machine**

- *clr Rd*: 1 cc
- *jmp k*: 3 cc
- *mul Rd, Rr*: 2 cc
- *Lpm Rd, Z*: 3 cc

Different instructions require different numbers of cycles



- **Multiplication takes more time than addition**
- **Floating point operations take longer than integer ones**
- **Accessing memory takes more time than accessing registers**
- **Important point: changing the cycle time often changes the number of cycles required for various instructions (more later)**

The CPU performance equation

- **Evaluated based on the CPU time of a certain program**

$$\text{cpu_time} = \frac{\text{seconds}}{\text{program}} = \frac{\text{instructions}}{\text{program}} \times \frac{\text{cycles}}{\text{instruction}} \times \frac{\text{seconds}}{\text{cycle}}$$

- **CPU time**

- **Time in seconds the CPU spends executing a program**
- **Comprised of three key components**
 - **instruction count = instructions/program**
 - **CPI = average cycles/instruction**
 - **clock period (or cycle time) = seconds/cycle**
 - **clock frequency = 1/(clock period)**

CPI – average cycles per instruction

$$\text{cpu_time} = \text{clock_period} * \sum_{i=1}^n \text{CPI}_i * I_i$$

Where n = the number of instructions in the ISA

I_i = instruction count for instruction i

CPI_i = number of clock cycles for instruction i

$$\begin{aligned} \text{CPI} &= (\text{cpu_time} * \text{clock_frequency}) / \text{instr_count} \\ &= (\text{cpu_time} / \text{clock_period}) / \text{instr_count} \\ &= \text{cycles} / \text{instr_count} \\ &= \sum_{i=1}^n \text{CPI}_i * \frac{I_i}{\text{instr_count}} \end{aligned}$$

CPI Exercises

(1) If two machines have a same Instruction Set Architecture (ISA), which of the following will always be identical for a given assembly program?

- clock rate
- CPI
- execution time
- # of instructions

(2) Suppose we have two implementations of a same ISA. Suppose also that for some program,

Machine A has a clock cycle time of 250 ps and a CPI of 2.0

Machine B has a clock cycle time of 500 ps and a CPI of 1.2

Which machine is faster for this program, and by how much?

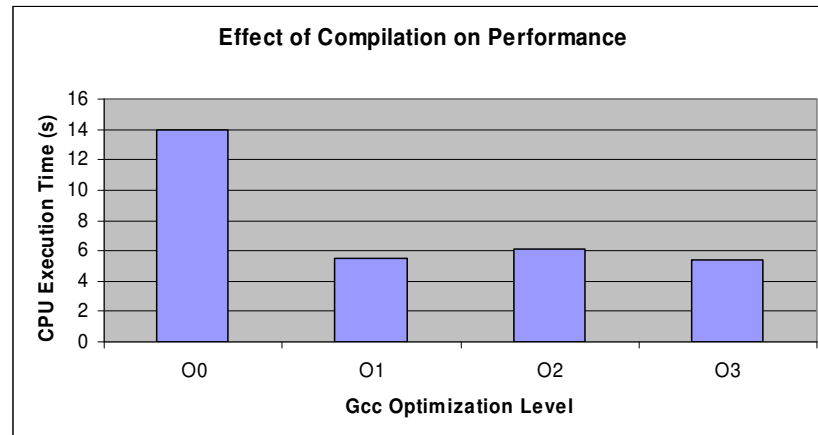
Aspects of CPU performance

- The various levels of computer design affect each aspect differently

Design level	Instr. count	CPI	Clock period
program	X	X	
compiler	X	X	
instr. set	X	X	X
organization		X	X
technology			X

Effect of compilation on performance

- Example



Exercise

A compiler writer must decide between two code sequences for a machine given the following:

Instruction class	CPI for this inst. class
A	1
B	2
C	3

The two code sequences require the following instruction counts:

Code sequence	A	B	C
1	2	1	2
2	4	1	1

(a) Which code sequence executes faster?

Exercise (cont.)

(b) What is the CPI for each sequence?

$CPI = (\text{CPU clock cycles}) / (\text{Instruction count})$

$CPI_1 = 10/5 = 2$

$CPI_2 = 9/6 = 1.5$

Remarks:

This example shows the danger of using only one factor (instruction count) to assess performance – all three components must be considered when comparing machines on the basis of execution time.

Improving performance

- Given an instruction set, CPU performance can be improved by
 - increase in the clock rate
 - Improvement in processor organization that lower the CPI
 - compiler enhancements that lower the instruction count or generate instructions with a lower CPI
- Improvement in one aspect may affect other aspects e.g. compiler that replaces sequence 2 with 1 in previous example reduces the instruction count only to increase CPI

Speedup

- Speedup due to enhancement E :

$$\text{speedup}(E) = \frac{\text{ex_time}(\quad)}{\text{ex_time}(E)} = \frac{\text{performance}(E)}{\text{performance}(\quad)}$$

Amdahl's Law

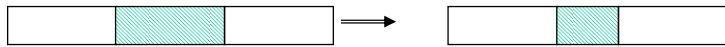
*“The performance enhancement possible with a given improvement is limited by the **amount that the improved feature is used**”*

Amdahl's Law (cont.)

- Suppose that enhancement E accelerates a fraction F of the task by a factor S and the remainder of the task is unaffected. Then

$$ex_time(E) = ((1-F) + F/S) \times ex_time()$$

$$ex\ time\ after\ improv = \frac{ex\ time\ affected\ by\ improv}{amount\ of\ improv} + ex\ time\ unaffected$$



- Speedup is limited by the amount that the improved feature is used

$$speedup(E) = \frac{1}{(1-F) + F/S}$$

Example:

- Suppose a program runs in 100 seconds on a machine, with multiply operations responsible for 80 seconds of this time. How much does the speed of the multiplication operation have to be improved for the program to run five times faster?

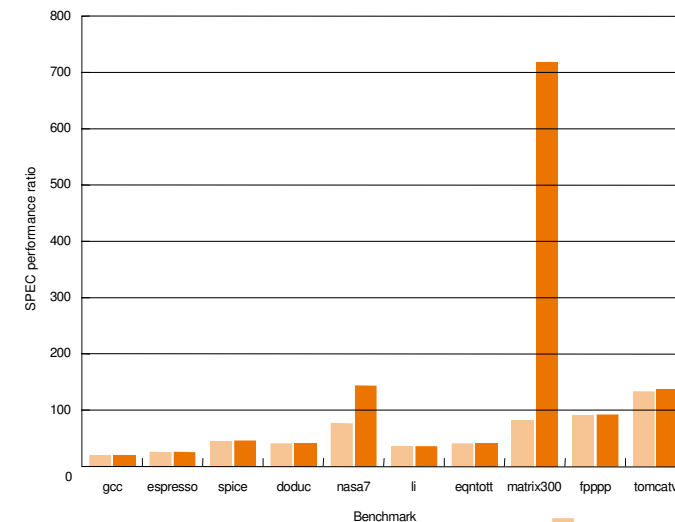
$$\frac{100}{5} = \frac{80}{x} + 20$$

$$x \rightarrow \infty$$

Benchmarks

- A computer performance best determined by running a real application
 - Use programs typical of expected workload
 - Or, typical of expected class of applications e.g., compilers/editors, scientific applications, graphics, etc.
- Small benchmarks
 - nice for architects and designers
 - easy to standardize
 - can be abused
- SPEC (System Performance Evaluation Cooperative)
 - companies have agreed on a set of real program and inputs
 - valuable indicator of performance (and compiler technology)
 - can still be abused

Example: SPEC '89



- Compiler "enhancements" and performance

SPEC CPU2000

Integer benchmarks		FP benchmarks	
Name	Description	Name	Type
gzip	Compression	wupwise	Quantum chromodynamics
vpr	FPGA circuit placement and routing	swim	Shallow water model
gcc	The Gnu C compiler	mgrid	Multigrid solver in 3-D potential field
mcf	Combinatorial optimization	applu	Parabolic/elliptic partial differential equation
crafty	Chess program	mesa	Three-dimensional graphics library
parser	Word processing program	galgel	Computational fluid dynamics
eon	Computer visualization	art	Image recognition using neural networks
perlbmk	perl application	equake	Seismic wave propagation simulation
gap	Group theory, interpreter	facerec	Image recognition of faces
vortex	Object-oriented database	ammp	Computational chemistry
bzip2	Compression	lucas	Primality testing
twolf	Place and rote simulator	fma3d	Crash simulation using finite-element method
		sixtrack	High-energy nuclear physics accelerator design
		apsi	Meteorology: pollutant distribution

FIGURE 4.5 The SPEC CPU2000 benchmarks. The 12 integer benchmarks in the left half of the table are written in C and C++, while the floating-point benchmarks in the right half are written in Fortran (77 or 90) and C. For more information on SPEC and on the SPEC benchmarks, see www.spec.org. The SPEC CPU benchmarks use wall clock time as the metric, but because there is little I/O, they measure CPU performance.

Exercises

- (a) Suppose we enhance a machine making all floating-point instructions run five times faster. If the execution time of some benchmark before the floating-point enhancement is 10 seconds, what will the speedup be if half of the 10 seconds is spent executing floating-point instructions?
- (b) We are looking for a benchmark to show off the new floating-point unit described above, and want the overall benchmark to show a speedup of 3. One benchmark we are considering runs for 100 seconds with the old floating-point hardware. How much of the execution time would floating-point instructions have to account for in this program in order to yield our desired speedup on this benchmark?