

CSE 141 – Computer Architecture Spring 2005

Lecture 2 Performance

Pramod V. Argade
March 30, 2005

CSE141: Introduction to Computer Architecture

Instructor: Pramod V. Argade (p2argade@cs.ucsd.edu)
Office Hour:
Mon. 5:00 - 6:00 PM (AP&M 5218)
Wed. 5:00 - 6:00 PM (AP&M 2444)

TAs:

Baris Arslan: barslan@cs.ucsd.edu
Chris Roedel: croedel@cs.ucsd.edu
Raid Ayoub: rayoub@cs.ucsd.edu
Leo Porter: leporter@cs.ucsd.edu

Textbook: Computer Organization & Design
The Hardware Software Interface, 3rd Edition.
Authors: Patterson and Hennessy

Web-page: <http://www.cse.ucsd.edu/classes/sp05/cse141>

Announcements

- **Discussions Sections for 141:**
Fridays, 10:00 - 10:50 pm, Peterson Hall 104 (Leo)
Fridays, 2:00 - 2:50 pm, Peterson Hall 104 (Leo)
- **Reading Assignment**
 - Chapter 2: Instructions. Sec. 2.1 - 2.15
 - Chapter 3: Arithmetic for Computers Sec. 3.1 - 3.9
- **Homework 1: Due Mon., April 4 in class**
- **Quiz**
When: Mon., April 4th, First 10 minutes of the class
Topic: Performance, Chapter 4
Need: Paper, pen, calculator

Course Schedule

Lecture #	Date	Day	Lecture Topic	Quiz Topic	Homework Due
1	3/28	Monday	Introduction, Ch. 1	-	-
2	3/30	Wednesday	Performance, Ch. 2	-	-
3	4/4	Monday	ISA, Ch. 3	Performance	#1
4	4/6	Wednesday	Arithmetic, Ch. 4	-	-
5	4/11	Monday	Arithmetic, Ch. 4	ISA	#2
6	4/13	Wednesday	Single cycle CPU, Ch. 5	-	-
7	4/18	Monday	Single cycle CPU, Ch. 5	Arithmetic	#3
8	4/20	Wednesday	Multi-cycle CPU, Ch. 5	-	-
9	4/25	Monday	Multi-cycle CPU, Ch. 5	Single Cycle CPU	#4
10	4/27	Wednesday	Review for the Midterm	-	-
	5/2	Monday	Mid-term Exam	-	-
11	5/4	Wednesday	Exceptions, Ch. 5 and Pipelining, Ch. 6	-	-
12	5/9	Monday	Pipelining, Ch. 6	-	-
13	5/11	Wednesday	Data and control hazards, Ch. 6	-	-
14	5/16	Monday	Data and control hazards, Ch. 6	Pipeline Hazards	#5
15	5/18	Wednesday	Memory & cache design, Ch. 7	-	-
16	5/23	Monday	Memory & cache design, Ch. 7	Cache	#6
17	5/25	Wednesday	Virtual Memory & cache design, Ch. 7	-	-
No Class	5/30	Monday	Memorial Day Holiday	-	-
18	6/1	Wednesday	Course Review	-	-
	6/10	Friday	Final Exam	-	-

Performance Assessment

Plane	DC to Paris time	Speed	Passengers	Throughput (p x mph)
Boeing 747	6.5 hours	610 mph	470	286,700
Concorde	3 hours	1350 mph	132	178,200

- Example definitions of performance
 - Passengers * MPH
 - Cruising speed
 - Passenger capacity
 - (Passenger * MPH)/\$ ⇐ need more data!
- Clearly formulate your definition of performance

Why worry about performance?

- Learn to measure, report, and summarize
- Make intelligent choices
- See through the marketing hype
- As a designer or purchaser, assess:
 - which system has best performance?
 - which system has lowest cost?
 - which system has highest performance/cost?
- Formulate metrics for performance measurement
 - How to report relative performance?
- Understand impact of architectural choices on performance

Computer Performance: TIME, TIME, TIME

- Response Time (latency)
 - How long does it take for my job to run?
 - How long does it take to execute a job?
 - How long must I wait for the database query?
- Throughput
 - How many jobs can the machine run at once?
 - What is the average execution rate?
 - How much work is getting done?
- *If we upgrade a machine with a new processor what do we decrease?*
- *If we add a new machine to the lab what do we increase?*

Execution Time

```
% time program
... program results ...
90.7u 12.9s 2:39 65%
%
```

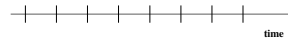
- Elapsed Time (2:39)
 - Counts everything (*disk and memory accesses, I/O, etc.*)
 - A useful number, but often not good for comparison purposes
- Total CPU time (103.6 s)
 - Doesn't count I/O or time spent running other programs
 - Comprised of system time (12.9 s), and user time (90.7 s)
- Our focus: user CPU time
 - Time spent executing the lines of code that are "in" our program

A Definition of Performance

- For some program running on machine X,
 $\text{Performance}_X = 1 / \text{Execution time}_X$
- “Machine X is n times faster than Y”
 $n = \text{Performance}_X / \text{Performance}_Y$
 $= (\text{Execution time}_Y) / (\text{Execution time}_X)$
- Problem:
 Execution time: Machine A: 12 seconds, B: 20 seconds
 - $A/B = .6$, so A is 40% faster, or 1.4X faster, or B is 40% slower
 - $B/A = 1.67$, so A is 67% faster, or 1.67X faster, or B is 67% slower
- Need a precise definition
 \Rightarrow “X is n times faster than Y”, $n > 1$
 - A is 1.67 times faster than B

Clock Cycles

- Operation of conventional computer is controlled by Clock “ticks”

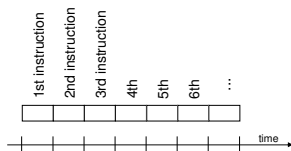


- Clock rate (frequency) = cycles per second (Hertz)
 MHz = Million cycles per second
 GHz = Giga (Billion) cycles per second
- Cycle time = time between ticks = seconds per cycle
 - A 2 GHz clock \Rightarrow Cycle time = $1/(2 \times 10^9)$ s = 0.5 nanoseconds
- Instead of reporting execution time in seconds, we often use clock cycles

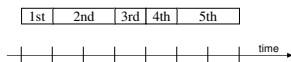
$$\frac{\text{seconds}}{\text{program}} = \frac{\text{cycles}}{\text{program}} \times \frac{\text{seconds}}{\text{cycle}}$$

How many cycles are required for a program?

- Assumption: # of cycles = # of instructions



- Assumption is incorrect
 - Typically, different instructions take different number of clock cycles
 - Integer Multiply instructions are multi-cycle
 - Floating point instructions are multi-cycle



Now that we understand cycles

- A given program will require
 - some number of instructions (machine instructions)
 - some number of cycles
 - some number of seconds
- We have a vocabulary that relates these quantities:
 - cycle time (seconds per cycle)
 - clock rate (cycles per second)
 - CPI (cycles per instruction)

a floating point intensive application might have a higher CPI

CPI Example 2.1

- Suppose we have two implementations of the same instruction set architecture (ISA).

For some program,

- Machine A has a clock cycle time of 10 ns. and a CPI of 2.0
- Machine B has a clock cycle time of 20 ns. and a CPI of 1.2

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

Compiler Optimization: Example 2.2

- A compiler designer is trying to decide between two code sequences for a particular machine with 3 instruction classes.
 - CPI: Class A = 1, Class B = 2, and Class C = 3
 - Code sequence X has 5 instructions:
 - > 2 of A, 1 of B, and 2 of C
 - Code sequence Y has 6 instructions:
 - > 4 of A, 1 of B, and 1 of C.
- Which sequence will be faster? How much?

Execution time: Contributing Factors

$$\text{CPU time} = \frac{\text{Seconds}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Cycle}}$$

	Instruction Count	CPI	Clock Rate
Program	X		
Compiler	X	(X)	
ISA	X	X	
Organization		X	X
Technology			X

- Improve performance => reduce execution time
 - Reduce instruction count (Programmer, Compiler)
 - Reduce cycles per instruction (ISA, Machine designer)
 - Reduce clock cycle time (Hardware designer, Process engineer)

Performance

- Performance is determined by program execution time
- Do any of the other variables equal performance?
 - # of cycles to execute program?
 - # of instructions in program?
 - # of cycles per second?
 - average # of cycles per instruction?
 - average # of instructions per second?
- Common pitfall: thinking one of the variables is indicative of performance when it really isn't.

Performance Variation

$$\text{CPU Execution Time} = \text{Instruction Count} \times \text{CPI} \times \text{Clock Cycle Time}$$

	Number of instructions	CPI	Clock Cycle Time
Same machine different programs	different	similar	same
same programs, different machines, same ISA	same	different	different
Same programs, different machines	somewhat different	different	different

Amdahl's Law

- The impact of a performance improvement is limited by the percent of execution time affected by the improvement

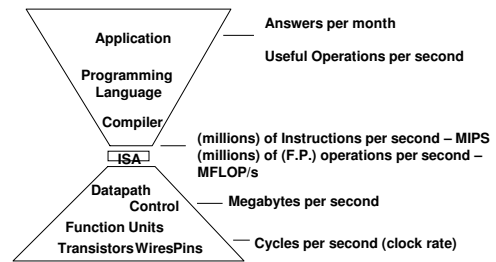
$$\text{Execution time after improvement} = \frac{\text{Execution Time Affected}}{\text{Amount of Improvement}} + \text{Execution Time Unaffected}$$

- Make the common case fast!
- Amdahl's law sets limit on how much improvement can be made

Amdahl's Law: Example 2.3

- Example: A program runs in 100 seconds on a machine, with multiply responsible for 80 seconds of this time. How much do we have to improve the speed of multiplication if we want the program to run 4 times faster?
- Is it possible to get the program to run 5 times faster?

Metrics of Performance



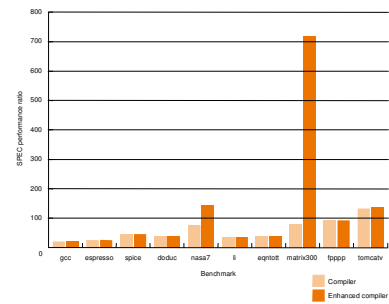
Each metric has a place and a purpose, and each can be misused

Benchmarks

- 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
 - can still be abused (Intel's "other" bug)
 - valuable indicator of performance (and compiler technology)

SPEC '89

- Compiler "enhancements" and performance

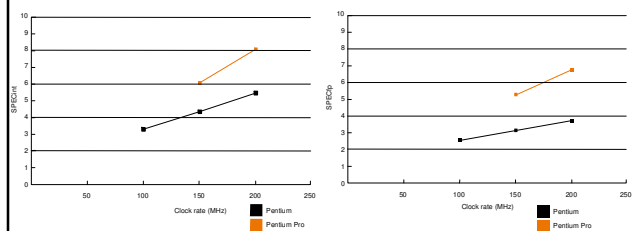


SPEC '95

Benchmark	Description
go	Artificial intelligence: plays the game of Go
m8ksim	Motorola 88k chip simulator; runs test program
gcc	The Gnu C compiler generating SPARC code
compress	Compresses and decompresses file in memory
li	Lisp interpreter
ljpeg	Graphic compression and decompression
perl	Manipulates strings and prime numbers in the special-purpose programming language Perl
vortex	A database program
tomcatv	A mesh generation program
swim	Shallow water model with 513 x 513 grid
sl2cor	quantum physics: Monte Carlo simulation
hydro2d	Astrophysics: Hydrodynamic Navier Stokes equations
mgrid	Multigrid solver in 3-D potential field
apdu	Parabolic/elliptic partial differential equations
trub3d	Simulates isotropic, homogeneous turbulence in a cube
apsi	Solves problems regarding temperature, wind velocity, and distribution of pollutant
lpppp	Quantum chemistry
wave5	Plasma physics; electromagnetic particle simulation

SPEC '95

- Does doubling the clock rate double the performance?*
Can a machine with a slower clock rate have better performance?



MIPS, MFLOPS etc.

- MIPS - million instructions per second

$$= \frac{\text{number of instructions executed in program}}{\text{execution time in seconds} \times 10^6} = \frac{\text{Clock rate}}{\text{CPI} \times 10^6}$$

- MFLOPS - million floating point operations per second

$$= \frac{\text{number of floating point operations executed in program}}{\text{execution time in seconds} \times 10^6}$$

- Hard to relate MIPS/MFLOPS with execution time

$$\text{CPU Execution Time} = \text{Instruction Count} \times \text{CPI} \times \text{Clock Cycle Time}$$

- Highly program-dependent metrics
- Deceptive (See pages 78 - 79 in the text)
 - MIPS would be higher for a program using simple instructions (e.g. a loop of NOPs)

RISC Processor: Example 2.4

Base Machine (Reg / Reg)

Op	Freq	Cycles	CPI(i)	% Time
ALU	50%	1		
Load	20%	5		
Store	10%	3		
Branch	20%	2		

- What is average CPI?
- What percentage of time is spent in each instruction class?
- How much faster would the machine be if a better data cache reduced the average load time to 2 cycles?

- How does this compare with using branch prediction to shave a cycle off the branch time?

- What if two ALU instructions could be executed at once?

Performance assessment is tricky

- Performance is specific to particular program(s)
 - Total execution time is a consistent summary of performance
- For a given architecture performance increases come from:
 - increases in clock rate (without adverse CPI affects)
 - improvements in processor organization that lower CPI
 - compiler enhancements that lower CPI and/or instruction count
- Pitfall: expecting improvement in one aspect of a machine's performance to affect the total performance
- You should not always believe everything you read! Read carefully!

Announcements

- **Discussions Sections for 141:**
 - Fridays, 10:00 - 10:50 pm, Peterson Hall 104 (Leo)
 - Fridays, 2:00 - 2:50 pm, Peterson Hall 104 (Leo)
- **Reading Assignment**
 - Chapter 2: Instructions. Sec. 2.1 - 2.15
 - Chapter 3: Arithmetic for Computers Sec. 3.1 - 3.9
- **Homework 1: Due Mon., April 4 in class**
- **Quiz**
 - When:** Mon., April 4th, First 10 minutes of the class
 - Topic:** Performance, Chapter 4 **Need:** Paper, pen, calculator