COMPUTER ARCHITECTURE

PERFORMANCE

Dr. Bill Yi
Santa Clara University

(Based on text: David A. Patterson & John L. Hennessy, Computer Organization and Design: The Hardware/Software Interface, 3rd Ed., Morgan Kaufmann, 2007)
(Also based on presentation: Dr. Nam Ling, COEN210 Lecture Notes)
COURSE CONTENTS

- Introduction
- Instructions
- Computer Arithmetic
- **Performance**
  - Processor: Datapath
  - Processor: Control
  - Pipelining Techniques
- Memory
- Input/Output Devices
PERFORMANCE

- Measuring Performance
- Time versus throughput
- Key to understanding underlying organizational motivation
- Improving Performance
Performance

- Measure, Report, and Summarize
- Make intelligent choices
- See through the marketing hype

*Why is some hardware better than others for different programs?*

*What factors of system performance are hardware related? (e.g., Do we need a new machine, or a new operating system?)*

*How does the machine’s instruction set affect performance?*
Which has the best performance?

<table>
<thead>
<tr>
<th>Airplane</th>
<th>Passengers</th>
<th>Range (mi)</th>
<th>Speed (mph)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boeing 737-100</td>
<td>101</td>
<td>630</td>
<td>598</td>
</tr>
<tr>
<td>Boeing 747</td>
<td>470</td>
<td>4150</td>
<td>610</td>
</tr>
<tr>
<td>BAC/Sud Concorde</td>
<td>132</td>
<td>4000</td>
<td>1350</td>
</tr>
<tr>
<td>Douglas DC-8-50</td>
<td>146</td>
<td>8720</td>
<td>544</td>
</tr>
</tbody>
</table>

- How much faster is the Concorde compared to the 747?
- How much bigger is the 747 than the Douglas DC-8?
Computer Performance: 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 increase?*
- *If we add a new machine to the lab what do we increase?*
Execution Time

- **Elapsed Time**
  - Counts everything (*disk and memory accesses, I/O, etc.*)
  - A useful number, but often not good for comparison purposes

- **CPU time**
  - Doesn’t count I/O or time spent running other programs
  - Can be broken up into system time, and user time

- **Our focus: user CPU time**
  - Time spent executing the lines of code that are “in” our program
Book’s Definition of Performance

- From some program running on machine $X$, 
  $\text{Performance}_X = 1 / \text{Execution time}_X$
- "$X$ is $n$ times faster than $Y$" 
  $\text{Performance}_X / \text{Performance}_Y = n$

Problem:
- Machine A runs a program in 20 seconds
- Machine B runs the same program in 25 seconds
Clock Cycles

- Instead of reporting execution time in seconds, we use cycles

\[
\frac{\text{seconds}}{\text{program}} = \frac{\text{cycles}}{\text{program}} \times \frac{\text{seconds}}{\text{cycle}}
\]

- Clock “ticks” indicate when to start activities (one abstraction):

- cycle time = time between ticks = seconds per cycle
- clock rate (frequency) = cycles per second (1 Hz. = 1 cycle/sec)

A 4 Ghz. clock has a cycle time of \( \frac{1}{4 \times 10^9} \times 10^{12} = 250 \) picoseconds (ps)

9
How to Improve Performance

\[ \frac{\text{seconds}}{\text{program}} = \frac{\text{cycles}}{\text{program}} \times \frac{\text{seconds}}{\text{cycle}} \]

- So, to improve performance (everything else being equal) you can either (increase or decrease?)

  - _________ the # of required cycles for a program, or
  - _________ the clock cycle time or, said another way,
  - _________ the clock rate.
Cycles Are Required for a Program

- Could assume that number of cycles equals number of instructions

This assumption is incorrect,

different instructions take different amounts of time on different machines. Why? hint: remember that these are machine instructions, not lines of C code
Different Numbers of Cycles for Different Instructions

- Multiplication takes more time than addition
- Floating point operations take longer than integer ones
- Accessing memory takes more than accessing registers

**Important point:** changing the cycle time often changes the number of cycles required for various instructions (more later)
Now 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)
  - MIPS (million instructions per second)
Performance

- Performance is determined by 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.
CPU Time Example

- CPU time = Instruction count x CPI x Clock cycle time
  - (CPI = Clock cycles per instruction, clock cycle time = 1/clock_rate)
  - CPU clock cycles = \( \sum_{i=1}^{n} (CPI_i \times C_i) \)
    - \( C_i \) = count of number of instructions of class \( i \)
    - Execution time = CPU time + System time (e.g. CPU time in OS) + Waiting time (e.g. CPU waits for I/O or memory)

- Example: CPU is 500 MHz, executes 2000 instructions with CPI = 1, 1000 instructions with CPI = 2, and 2000 instructions with CPI = 3
- CPU time = \( (2000 \times 1 + 1000 \times 2 + 2000 \times 3) \times 2 \text{ nsec} \)
  = 20000 nsec or 20 usec

- Speedup = performance after improvement / performance before improvement
CPI Example

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

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?

Solution:
A compiler designer is trying to decide between two code sequences for a particular machine. Based on the hardware implementation, there are three different classes of instructions: Class A, Class B, and Class C, and they require one, two, and three cycles respectively.

The first sequence has 5 instructions: 2 of A, 1 of B, and 2 of C.

The second sequence has 6 instructions: 4 of A, 1 of B, and 1 of C.

Which sequence will be faster? How much?

What is the CPI for each sequence?
Two different compilers are being tested for a 4 GHz. Machine with three different classes of instructions: Class A, Class B, and Class C, which require one, two, and three cycles respectively. Both compilers are used to produce code for a large piece of software.

The first compiler’s code uses 5 billion Class A instructions, 1 billion Class B instructions, and 1 billion Class C instructions.

The second compiler’s code uses 10 billion Class A instructions, 1 billion Class B instructions, and 1 billion Class C instructions.

Which sequence will be faster according to MIPS?

Which sequence will be faster according to execution time?
Benchmarks

- Performance best determined by running a real application
  - Use programs typical of expected workload
  - Or, typical of expected class of applications
    e.g., compiler/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
SPEC 2000

- Does doubling the clock rate double the performance?
- Can a machine with a slower clock rate have better performance?
Amdahl’s Law

- **Execution time after improvement** = 
  \[(\text{Exec\_time\_affected\_by\_improvement} / \text{Amount\_of\_improvement}) + \text{Exec\_time\_unaffected}\]

- **Example:**
  “Suppose 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?”

  How about making it 5 times faster?

- **Principle:** Make the common case fast
Remember

- Performance is specific to a particular program
  - 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
  - Algorithm/Language choices that affect instruction count

- Pitfall: expecting improvement in one aspect of a machine’s performance to affect the total performance
Chapter Summary

- Measuring performance
- CPU time, CPI, MIPS, Amadahl’s Law
- Improving performance