IC220
Slide Set #5B: Performance
(Chapter 1: 1.6, 1.9-1.11)

Computer Performance:

- Execution / 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 improve?

- If we add a new machine to the lab what do we improve?

Performance

- Measure, Report, and Summarize
- Make intelligent choices
- See through the marketing hype
- Key to understanding underlying organizational motivation

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?

- Execution Time

  - Elapsed Time =
    - 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 is?
For some program running on machine X,

\[ \text{Performance}_X = \]

"X is n times faster than Y"

Example:
- Machine A runs a program in 20 seconds
- Machine B runs the same program in 25 seconds
- How much faster is A than B?
- (always use "times faster" NOT "10 sec faster")

**Book’s Definition of Performance**
**Clock Cycles**

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

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

\[ \text{CPUtime} = \text{CPUClockCycles} \times \text{ClockCycleTime} \]

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

\[ \text{Clock Cycle time} = \]

- Clock rate (frequency) =

What is the clock cycle time for a 200 Mhz. clock rate?

**Measuring Execution Time**

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

\[ \text{CPUtime} = \text{CPUClockCycles} \times \text{ClockCycleTime} \]

(or, equivalently)

\[ \text{CPUtime} = \frac{\text{CPUClockCycles}}{\text{ClockRate}} \]

Example: Some program requires 100 million cycles. CPU A runs at 2.0 GHz. CPU B runs at 3.0 GHz. Execution time on CPU A? CPU B?

**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

\[ \square \] the # of required cycles for a program, or
\[ \square \] the clock cycle time or, said another way, 
\[ \square \] the clock rate.
Performance / Clock Cycle Review

1. Performance = 1 / Execution Time = 1 / CPU time

2. How do we compute CPU Time?
   - CPU Time = CPU Clock Cycles * Clock Cycle Time
     
     \[
     \text{seconds} \quad \text{program} = \frac{\text{cycles}}{\text{program}} \times \frac{\text{seconds}}{\text{cycle}}
     \]

3. How do we get these?
   - Clock Cycle Time = time between ticks (seconds per cycle)
     - Usually a given
     - Or compute from Clock Rate
   - CPU Clock Cycles = # of cycles per program
     - Where does this come from?

Cycles Per Instruction (CPI)

CPU Clock Cycles
= Total # of clock cycles
= avg # of clock cycles per instruction * program instruction count
= CPI * IC

What is CPI?
- Average cycle count of all the instruction executed in the program
- CPI provides one way of comparing 2 different implementations of the same ISA, since the instruction count for a program will be the same

New performance equation:
Time = Instruction count * CPI * ClockCycleTime

How many cycles are required for a program?

- Could assume that # of cycles = # of instructions

CPI Example

- 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?
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 code 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?

Performance
- Performance is determined by ______!
- 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:

Evaluating Performance
- Best scenario is head-to-head
  - Two or more machines running the same programs (workload), over an extended time
  - Compare execution time
  - Choose your machine

- Fallback scenario: BENCHMARKS
  - Packaged in ‘sets’
  - Programs specifically chosen to measure performance
    - Programs typical of ___________
  - Composed of real applications
    - Specific to workplace environment
    - Minimizes ability to speed up execution

Types of Benchmarks used depend on position of development cycle
- Small benchmarks
  - Nice for architects and designers
  - Very small code segments
  - Easy to standardize
  - Can be abused

- SPEC (System Performance Evaluation Cooperative)
  - http://www.specbench.org/
  - Companies have agreed on a set of real program and inputs
  - Valuable indicator of performance (and compiler technology)
  - Latest: SPEC CPU2006
  - (still???) In development: SPEC CPUv6
### Amdahl's Law

**Execution Time After Improvement** = 

\[
\text{Execution Time After Improvement} = \frac{\text{Execution Time Unaffected} + \left( \frac{\text{Execution Time Affected}}{\text{Amount of Improvement}} \right)}{}
\]

- **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?**

- **Corollary:** Make the common case fast

### Remember

- Performance is specific to _____________________
  - Only total execution time is a consistent summary of performance

- For a given architecture, performance increases come from:

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

- You should not always believe everything you read! Read carefully!