Computer Systems

Performance

Outline

- Introduction
- Computer Performance
- CPU Time
- Benchmark

Performance

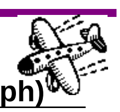
- **Measure, Report**
- **Make intelligent choices**
- **See through the marketing hype**
- **Key to understanding underlying organizational motivation**

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?

Which of these airplanes has the best performance?



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

- **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**
- **System CPU time**
  - **time spent in OS performing tasks on behalf of a user program**
- **Our focus: user CPU time**
  - **time spent executing the lines of code that are "in" our program**
- **Usually:**
  - **System performance: elapsed time**
  - **User performance: user CPU time**

Definition of Performance

- For 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

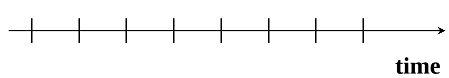
  How much faster is A than B?

Clock Cycles

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

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

- Cycle: clock "ticks" indicate when to start activities:



- cycle time = time between ticks = seconds per cycle
- clock rate (frequency) = cycles per second

  (1 Hz. = 1 cycle/sec)

A 200 MHz clock has a cycle time $\frac{1}{200 \times 10^6} \times 10^9 = 5 \text{ nanoseconds}$

How to Improve Performance

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

So, to improve performance (everything else being equal) you can either

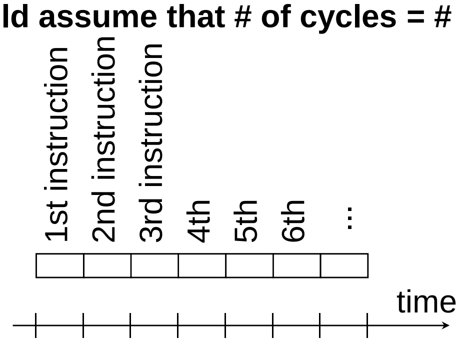
________ the # of required cycles for a program, or

________ the clock cycle time or, said another way,

________ the clock rate.

How many cycles are required for a program?

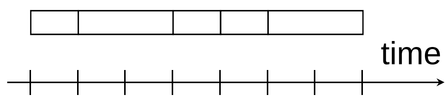
- **Could assume that # of cycles = # of instructions**



This assumption is incorrect, as

different instructions take different amounts of time on different machines.

Different numbers of cycles for different instructions



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

Example

- **Our favorite program runs in 10 seconds on computer A that has a 400 MHz 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 more expensive) 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?"**

- To answer it, let’s work out from basic principles

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

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

  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?

More CPI Example

- **Assume a program has the following instruction mix. What is its average CPI?**

Operations	Frequency	Cycles
Arithmetic/logical instructions	45%	1
Register load operations	20%	4
Register store operations	7%	2
Unconditional branch	9%	1
Conditional branch	12%	2
MIsc.	7%	1

CPU Time (IMPORTANT FORMULA!)

- **CPU time** $= \dfrac{\text{Seconds}}{\text{Program}}$

$$= \frac{\text{Cycles}}{\text{Program}} \times \frac{\text{Seconds}}{\text{Clock cycle}}$$

$$= \frac{\text{Instruction}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock cycle}}$$

= (Instruction count) x CPI x (Clock cycle time)
= (Instruction count) x CPI x (1/Clock rate)

- **Given the equation, to improve the performance, we want to**
  - **___ Instruction count: ISA, complier, algorithm**
  - **___ CPI: computer organization, ISA**
  - **___ Clock cycle time: chip technology, computer organization**
  - **___ Clock rate**

CPU Time

- **# of instructions example:**

  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?

More CPU Time Example

- **Assume a program has the following instruction mix of 300 instructions. What is the CPU time if it runs on a processor of 1.2 GHz?**

Operations	Frequency	Cycles
Arithmetic/lngical instructions	45%	1
Register load operations	20%	4
R%gister store operations	7%	2
Unconditional branch	9%	1
C/nditional branch	12%	2
Misc.	7%	1

CPU Time Formula

- The formula we use
- CPU time = $\frac{\text{Instruction}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock cycle}}$
- = (Instruction count) x CPI x (ClOck cycle time)

- Our textbook
- Taxecute = Ninst 8 !/Winst x CPI x Tcyc
- Difference: 1/Winst, where Winst (wgrk carried out per instruction) is a made-up
- We will use the first one.

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 (performance improved over small benchmark)
- SPEC (System Performance Evaluation Cooperative)
  - companies have agreed on a set of real program and inputs
  - can still be abused (1995, Intel's internal compiler)
  - valuable indicator of performance (and compiler technology)