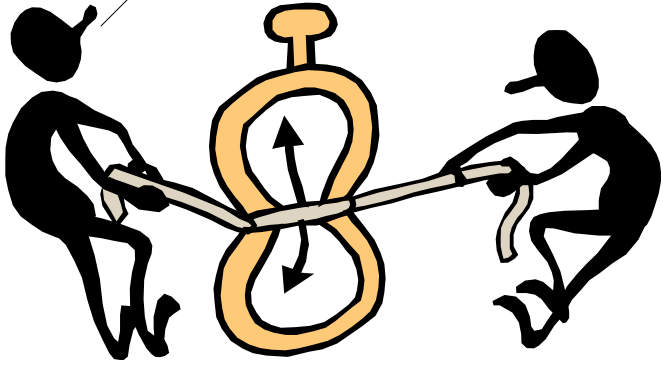


## Computer Performance

He said, to speed things up we need to squeeze the clock



Read Chapter 4  
(Chapter 2 in old book)

Comp 120 – Spring 2005 3/10/05 L15 – Computer Performance 1

## Why Study Performance?

Make intelligent design choices

See through the marketing hype

Key to understanding underlying computer organization

*Why is some hardware faster 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 a machine's instruction set affect its performance?*

Comp 120 – Spring 2005 3/10/05 L15 – Computer Performance 2

## Which Airplane 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 than the 747? **2.213 X**

How much larger is the 747's capacity than the Concorde? **4.65 X**

It is roughly 4000 miles from Raleigh to Paris. What is the throughput of the 747 in passengers/hr? The Concorde?

$$470 \times \frac{610}{4000} = 71.675$$

$$101 \times \frac{1350}{4000} = 34.0875$$

What is the latency of the 747? The Concorde? **6.56 hours, 2.96 hours**

## Computer vs H/W Performance

Latency/Response Time (clocks from input to corresponding output)

- How long does it take for my program to run?
- How long must I wait after typing return for the result?

Throughput (How many results per clock)

- How many results can be processed per second?
- What is the average execution rate of my program?
- How much work is getting done?


If we upgrade a machine with a new processor what do we improve?

**Response Time/Latency**


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

**Throughput**

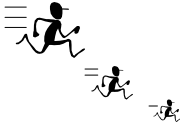
## Design Tradeoffs




**Maximum Performance:** measured the numbers of instructions executed | second



**Minimum Cost:** measured by the size of the circuit.




**Best Performance/Price:** measured by the ratio of MIPS to size. In power-sensitive applications MIPS/Watt is important too.



**1**  
**EARTH SIMULATOR**  
Earth Simulator Center  
Yokohama  
NEC  
Rmax: 35.86 TFlops

XILINX DELIVERS WORLD'S LOWEST COST FPGA-BASED EMBEDDED PROCESSOR FOR LESS THAN 75 CENTS  
Industry's only low-cost FPGA with dedicated Multipliers & Block RAM for embedded system solutions



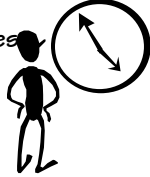
Comp 120 – Spring 2005
3/10/05
L15 – Computer Performance 5

## Execution Time

**Elapsed Time/Wall Clock Time**  
counts everything (disk and memory accesses, I/O, etc.)  
a useful number, but often not good for comparison purposes

**CPU time**  
Doesn't include 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 actual instructions of "our" program



Comp 120 – Spring 2005
3/10/05
L15 – Computer Performance 6

## Book's Definition of Performance

For some program running on machine X,

$$\text{Performance}_x = \text{Program Executions} / \text{Time}_x \text{ (executions/sec)}$$

"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

$$\text{Performance}_A = 1/20 \quad \text{Performance}_B = 1/25$$

$$\text{Machine A is } (1/20)/(1/25) = 1.25 \text{ times faster than Machine B}$$

## Clock Cycles

Instead of reporting execution time in seconds, we often use cycle counts

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

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



cycle time = time between ticks =  $\frac{\text{seconds}}{\text{cycle}}$

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

A 200 MHz clock has a

$$\frac{1}{200 \times 10^6} \times 10^9 = 5 \text{ ns} \quad \text{cycle time}$$

A 3 GHz clock has a

$$\frac{1}{3.10^9} \times 10^9 = 0.33 \text{ ns} \quad \text{cycle time}$$

(note:  $c \times 0.33\text{ns} = 0.1\text{m}$ )

## Computer Performance Measure

$$\text{MIPS} = \frac{\text{clocks/sec}}{\text{AVE(clocks/instruction)}}$$

Millions of Instructions per Second

Frequency in MHz

CPI (Average Clocks Per Instruction)

Which of these terms are program dependent?



Historically:

PDP-11, VAX, Intel 8086 CPI > 1

Load/Store RISC machines

MIPS, SPARC, PowerPC, miniMIPS: CPI = 1

Modern CPUs, Pentium, Athlon CPI < 1

## How to Improve Performance?

$$\frac{\text{seconds}}{\text{program}} = \frac{\text{cycles}}{\text{program}} \times \frac{\text{seconds}}{\text{cycle}} \quad \text{MIPS} = \frac{\text{Freq}}{\text{CPI}}$$

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

Decrease the # of required cycles for a program, or (improve ISA/Compiler)

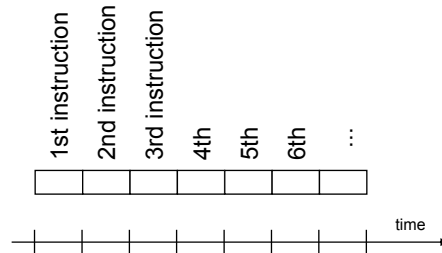
Decrease the clock cycle time or, said another way,

Increase the clock rate. (reduce propagation delays or use pipelining)

Decrease the CPI (average clocks per instruction) (new H/W)

## How Many Cycles in a Program?

Could assume that # of cycles = # of instructions



*This assumption can be incorrect,*

*Different instructions take different amounts of time on different machines.*

*Memory accesses might require more cycles than other instructions.*

*Floating-Point instructions might require multiple instructions to execute.*

*Branches might stall execution rate*

## Example

Our favorite program runs in 10 seconds on computer A, which has a 400 Mhz clock. We are trying to help a computer designer build a new machine B, to 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?

$$\frac{\text{cycles}}{\text{program}} = \left( \frac{\text{sec onds}}{\text{program}} \right)_A \times \frac{\text{cycles}}{\text{sec ond}} = 10 \times 400 \times 10^6 = 4 \times 10^9$$

$$\frac{\text{cycles}}{\text{sec ond}} = \frac{\text{cycles/program}}{\left( \text{sec onds/program} \right)_B} = \frac{1.2 \times 4 \times 10^9}{6} = 800 \times 10^6$$

Don't panic, can easily work this 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 (average clocks per instruction)*

*a floating point intensive application might have a higher CPI*

*MIPS (millions of instructions per second)*

*this would be higher for a program using simple instructions*

## Performance Traps

Performance is determined by the execution time of a program that you care about

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 only one of the variables is indicative of performance when it really isn't.

## 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?

$$\text{MIPS}_A = \frac{\text{freq}}{\text{CPI}} = \frac{10^{-6}/(10 \times 10^{-9})}{2} = 50 \quad \text{MIPS}_B = \frac{\text{freq}}{\text{CPI}} = \frac{10^{-6}/(20 \times 10^{-9})}{1.2} = 41.66$$

$$\text{Performance} = \frac{\text{MIPS}_A}{\text{MIPS}_B} = \frac{50}{41.66} = 1.2$$

If two machines have the same ISA which quantity (e.g., clock rate, CPI, execution time, # of instructions, MIPS) will always be identical?

## Compiler's Performance Example

Two different compilers are being tested for a 500 MHz 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 million Class A instructions, 1 million Class B instructions, and 2 million Class C instructions. The second compiler's code uses 7 million Class A instructions, 1 million Class B instructions, and 1 million Class C instructions.

Which sequence uses the fewest number of instructions?

$$\text{Instructions}_1 = (5+1+2) \times 10^6 = 8 \times 10^6$$

$$\text{Instructions}_2 = (7+1+1) \times 10^6 = 9 \times 10^6$$

Which sequence uses the fewest clock cycles?

$$\text{Cycles}_1 = (5(1)+1(2)+2(3)) \times 10^6 = 13 \times 10^6$$

$$\text{Cycles}_2 = (7(1)+1(2)+1(3)) \times 10^6 = 12 \times 10^6$$

# 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)

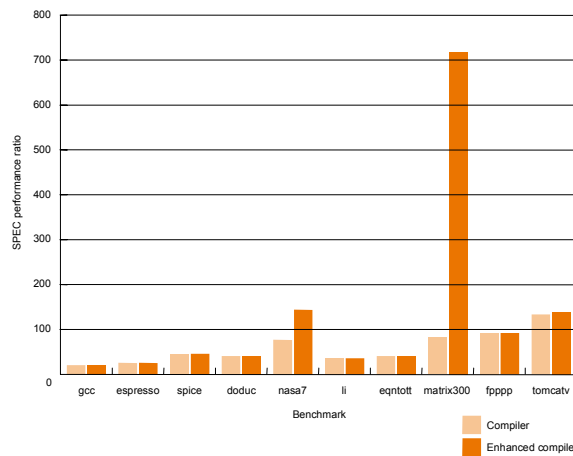
companies have agreed on a set of real program and inputs

can still be abused (Intel's "other" bug)

valuable indicator of performance (and compiler technology)

# SPEC '89

Compiler "enhancements" and performance



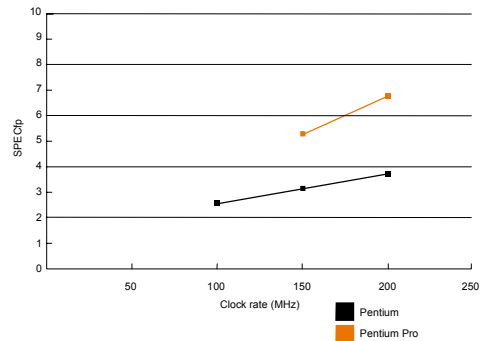
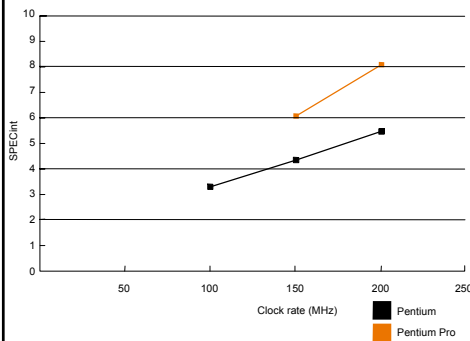
# SPEC '95

Benchmark	Description
go	Artificial intelligence; plays the game of Go
m88ksim	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
su2cor	quantum physics; Monte Carlo simulation
hydro2d	Astrophysics; Hydrodynamic Navier Stokes equations
mgrid	Multigrid solver in 3-D potential field
applu	Parabolic/elliptic partial differential equations
trub3d	Simulates isotropic, homogeneous turbulence in a cube
apsi	Solves problems regarding temperature, wind velocity, and distribution of pollutant
fpccc	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?*



## Amdahl's Law

$$t_{\text{improved}} = \frac{t_{\text{affected}}}{r_{\text{speedup}}} + t_{\text{unaffected}}$$

Example:

"Suppose a program runs in 100 seconds on a machine, where multiplies are 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?"

$$25 = 80/r + 20 \quad r = 16x$$

How about making it 5 times faster?

$$20 = 80/r + 20 \quad r = ?$$

*Principle: Make the common case fast*

## Example

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 only half of the 10 seconds is spent executing floating-point instructions?

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?

## Remember

Performance is specific to a particular programs

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 improvements in one aspect of a machine's performance to affect the total performance

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