Performances of Computer Systems

Andrea Janes

Content

- **Review of the evolution of the speed of computer systems**
- The role of performances
- Measuring performances
- How to combine different performance measures to make decisions
- Programs to determine comprehensive performance indexes

Evolution of the speed

Now we know…

- Rapidly changing field:
  - vacuum tube -> transistor -> IC -> VLSI (see section 1.4)
  - doubling every 1.5 years:
    memory capacity
    processor speed

Freie Universität Bozen
Libera Università di Bolzano
Free University of Bozen · Bolzano

Increase of technology enabled abstraction

- Delving into the depths reveals more information
- An abstraction omits unneeded detail, helps us cope with complexity

High-level language program (in C)

```
swap(int v[], int k)
{int temp;
   temp = v[k];
   v[k] = v[k+1];
   v[k+1] = temp;
}
```

C compiler

Assembly language program (for MIPS)

```
swap:
      muli $2, $5,4
      add  $2, $4,$2
      lw   $15, 0($2)
      lw   $16, 4($2)
      sw   $16, 0($2)
      sw   $15, 4($2)
      jr   $31
```

Assembler

Binary machine language program (for MIPS)

```
00000000101000010000000000011000
00000000100011100001100000100001
10001100011000100000000000000000
10001100111100100000000000000100
10101100111100100000000000000000
10101100011000100000000000000100
00000011111000000000000000001000
```

Instruction Set Architecture

- A very important abstraction
  - interface between hardware and low-level software
  - standardizes instructions, machine language bit patterns, etc.
  - advantage: *different implementations of the same architecture*
  - disadvantage: *sometimes prevents using new innovations*

Organization of computers

- All computers consist of five components
  - Processor/CPU: (1) datapath and (2) control
  - (3) Memory
  - (4) Input devices and (5) Output devices
- Not all “memory” are created equally
  - Cache: fast (expensive) memory are placed closer to the processor
  - Main memory: less expensive memory--we can have more
  - Secondary memory: even less expensive and we can have much more
- Input and output (I/O) devices have the messiest organization
  - Wide range of speed: graphics vs. keyboard
  - Wide range of requirements: speed, standard, cost
  - Least amount of research (so far)

Content

- *Review of the evolution of the speed of computer systems*
- **The role of performances**
- Measuring performances
- How to combine different performance measures to make decisions
- Programs to determine comprehensive performance indexes

Performance – why?

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

Typical questions

- “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?

What if we consider a new index…

- Passenger throughput:

$$\sum_i passenger_i \times speed(passenger_i)$$

Airplane	Passengers	Range (mi)	Speed (mph)	PT (passxmph)
Boeing 737-100	101	630	598	228,750
Boeing 747	**470**	4150	610	**286,700**
BAC/Sud Concorde	132	4000	**1350**	178,200
Douglas DC-8-50	146	**8720**	544	79,424

Content

- *Review of the evolution of the speed of computer systems*
- *The role of performances*
- **Measuring performances**
- How to combine different performance measures to make decisions
- Programs to determine comprehensive performance indexes

How do we define performances?

The GQM

- Goal – set the goal why you measure
- Question – set suitable questions you are interested in determining
- Metrics – set up a suitable measurement device
  - Your example #1: performances of a car
  - Your example #2: performances of a computer system

Performances of a computer system

- Goal:
  - To measure how fast is a computer system
- Question
  1. How long does it take for a job to run?
  2. How many jobs can the machine run at once?

Two kinds of indexes

- How long does it take for a job to run?
  - ***Response time***

- How many jobs can the machine run at once?
  - ***Throughput***

Response time

AKA Execution time

- 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?

Different Execution Times

- ***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

Throughput

- How many jobs can the machine run at once?
- What is the average execution rate?
- How much work is getting done?

Response time and throughput

- Response time and throughput are related … sometimes
  - Replacing a processor with a faster yields both improves of response time and throughput
  - Adding a second processor, improves the throughput but NOT the response time

Performances

- For some program running on a 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: A runs a program in 20 seconds; B runs the same program in 25 seconds; how faster is A than B?

$\text{Performance}_A / \text{Performance}_B =$

$= (1/ \text{Execution time}_A)/(1/ \text{Execution time}_B) =$

$= \text{Execution time}_B / \text{Execution time}_A = 25/20 = 5/4 = 1.25$

The notion of clock cycle

- Inside computers there is a device which measures time, the ***clock***
- The clock determines the sequencing of events …
- … via clock cycles of constant time, aka clock periods, clock ticks, or ***ticks***
- The duration is called ***cycle time***
- The ***clock rate*** is (clock cycles)$^{-1}$

Content

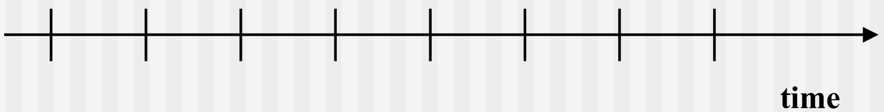
- *Review of the evolution of the speed of computer systems*
- *The role of performances*
- *Measuring performances*
- **How to combine different performance measures to make decisions**
- Programs to determine comprehensive performance indexes

Using Clock Cycles

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

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

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



The clock rate

We have often heard the concept of clock rate:

- clock rate (frequency) = cycles per second (1 Hz. = 1 cycle/sec)
- So, a 200 Mhz. clock has a cycle time

$$\frac{1}{200 \times 10^{6}} = 5 \times 10^{-9}\ sec = 5 nano\, sec$$

Relating the metrics

For a program PGM:

$$ExecutionTime(PGM) = ClockCycles(PGM) \times CycleTime$$

$$ExecutionTime(PGM) = \frac{ClockCycles(PGM)}{ClockRate}$$

How to Improve Performance

$$Seconds(PGM) = Cycles(PGM) \times Seconds(Cycle)$$

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

- reduce the # of required cycles for a program –i.e., make your program more efficient, or
- reduce the clock cycle time or, said another way,
- increase the clock rate –i.e., buy a faster processor

Exercise

- A program P runs in 25 seconds on computer A, a 800 MHz machine
- What should be the clock rate of another computer B for the program to run in 15 seconds? B should be otherwise identical to A, e.g., same instruction set, modulo the speed up of access times to support the higher rate

Solution

$$Seconds(A) = Cycles(A) \times Seconds(Cycle_A)$$

$$Cycles(A) = \frac{Seconds(A)}{Seconds(Cycle_A)} = \frac{25}{\frac{1}{800 \times 10^6}} =$$

$$= 25 \times 8 \times 10^8 = 2 \times 10^{10}$$

$$Seconds(Cycle_B) = \frac{Seconds(B)}{Cycles(B)}$$

$$= \frac{15}{2 \times 10^{10}} = 7.5 \times 10^{-10}$$

$$Rate_B = \frac{1}{Seconds(Cycle_B)} = \frac{10^{10}}{7.5} = 1.33GHz$$

Proposed Exercise #1

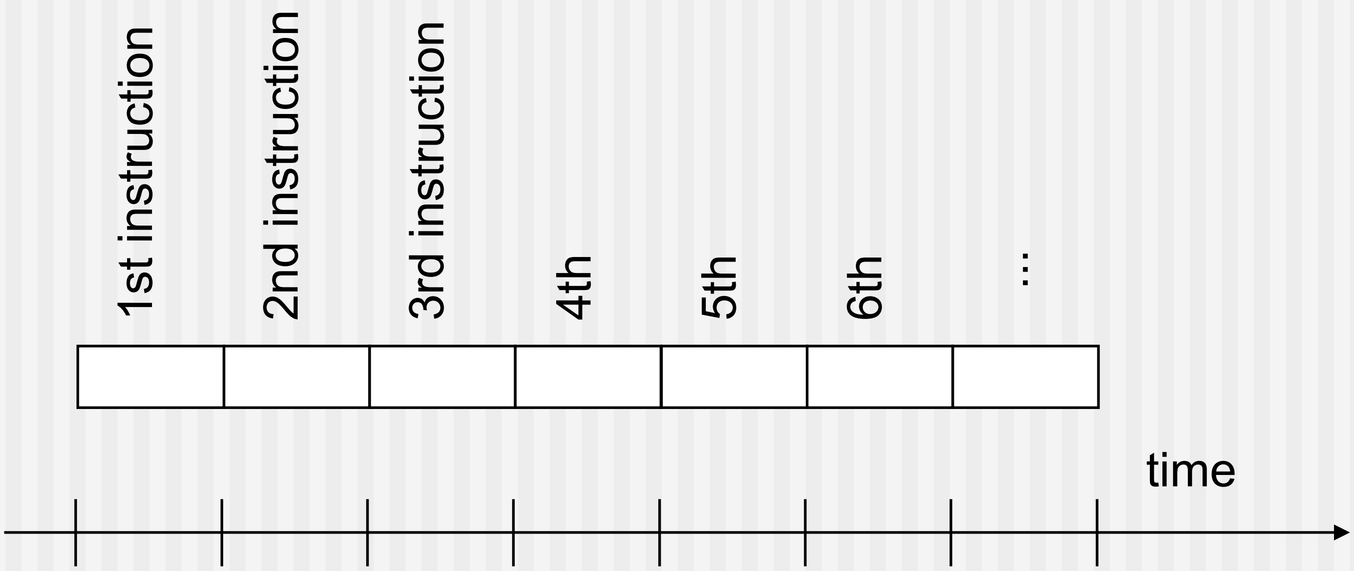
- Our favorite program runs in 1.5 seconds on computer A, which has a 700 Mhz. clock. We are trying to help a computer designer build a new machine B, that will run this program in 1 seconds. The designer can use new (or perhaps more expensive) technology to substantially increase the clock rate, but has informed us that this increase will affect the rest of the CPU design, causing machine B to require 2 times as many clock cycles as machine A for the same program. What clock rate should we tell the designer to target?

Proposed Exercise #2

- 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, 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?"

How many cycles are required for a program?

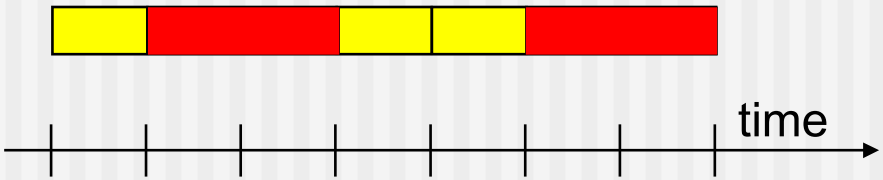
- Can we assume that # of cycles = # of instructions



Solution

- This assumption is incorrect
- Different instructions take different amounts of time on different machines.
- Why? Remember that these are machine instructions, not lines of Java 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 time than accessing registers
- *Important point: changing the cycle time often changes the number of cycles required for various instructions (more later)*

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 cycles per instruction) this is an ***average***!!
  - MIPS (millions of instructions per second), higher for programs using simple instruction sets

Use of CPI

- Using the notion of CPI and CPU time we can rewrite the equation:

$$Seconds(P) = Cycles(P) \times Seconds(Cycle)$$

- Into first:

$$CPUT(P) = Cycles(P) \times Seconds(Cycle)$$

- And then, with an approximation:

$$CPUT(P) = Instructions(P) \times CPI \times Seconds(Cycle)$$

$$CPUT(P) = \frac{Instructions(P) \times CPI}{ClockRate}$$

Example of CPI

Suppose that we have two implementations of the same instruction set:

- A with clock cycle time of .5ns and CPI for a program P of 2.5
- B with clock cycle time of .7ns and CPI for the same program P of 2

Which machine is faster for P?

Solution

$$CPUT_A(P) = Instructions(P) \times CPI_A \times Seconds(Cycle_A) =$$
$$= Instructions(P) \times 2.5 \times .5 \times 10^{-9}$$
$$= Instructions(P) \times 1.25 \times 10^{-9}$$

$$CPUT_B(P) = Instructions(P) \times CPI_B \times Seconds(Cycle_B) =$$
$$= Instructions(P) \times 2 \times .7 \times 10^{-9}$$
$$= Instructions(P) \times 1.4 \times 10^{-9}$$

Proposed exercise

Suppose that we have three implementations of the same instruction set:

- A with clock rate of 1GHz and CPI for a program P of 2
- B with clock rate of 800MHz and CPI for the same program P of 1.6
- C with clock rate of 900MHz and CPI for the same program P of 1.8

Which machine is faster for P?

of Instructions Example

- A compiler designer is trying to decide between three 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. The third has 5 instructions: 2 of A, 2 of B, and 1 of C.
- Which sequence will be faster? How much?
- What is the CPI for each sequence?

Content

- *Review of the evolution of the speed of computer systems*
- *The role of performances*
- *Measuring performances*
- *How to combine different performance measures to make decisions*
- **Programs to determine comprehensive performance indexes**

Performance

- Performance is determined by execution time
- Do any of the other variables equal performance?
  - # of cycles to execute program?
  - # of instructions in program? (MIPS)
  - # 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.

MIPS example

- 3 compilers are being tested for a 100 MHz. machine with 3 classes of instructions: A, B, and 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, 1Mio Bs, and 1Mio Cs.
- The second compiler's code uses 10Mio As, 1Mio Bs, and 1Mio Cs.
- The third compiler’s code uses 7Mio As, and 1 Mios Bs.
- Which sequence will be faster according to MIPS? Which according to execution time?

Simple way to compute the solution

Classes / Compiler	A (Mio)	B (Mio)	C (Mio)	TotInstr (Mio)	Time (sec)	MIPS
1	5	1	1	7	0.10	70.00
2	10	1	1	12	0.15	80.00
3	7	1	0	8	0.09	88.89

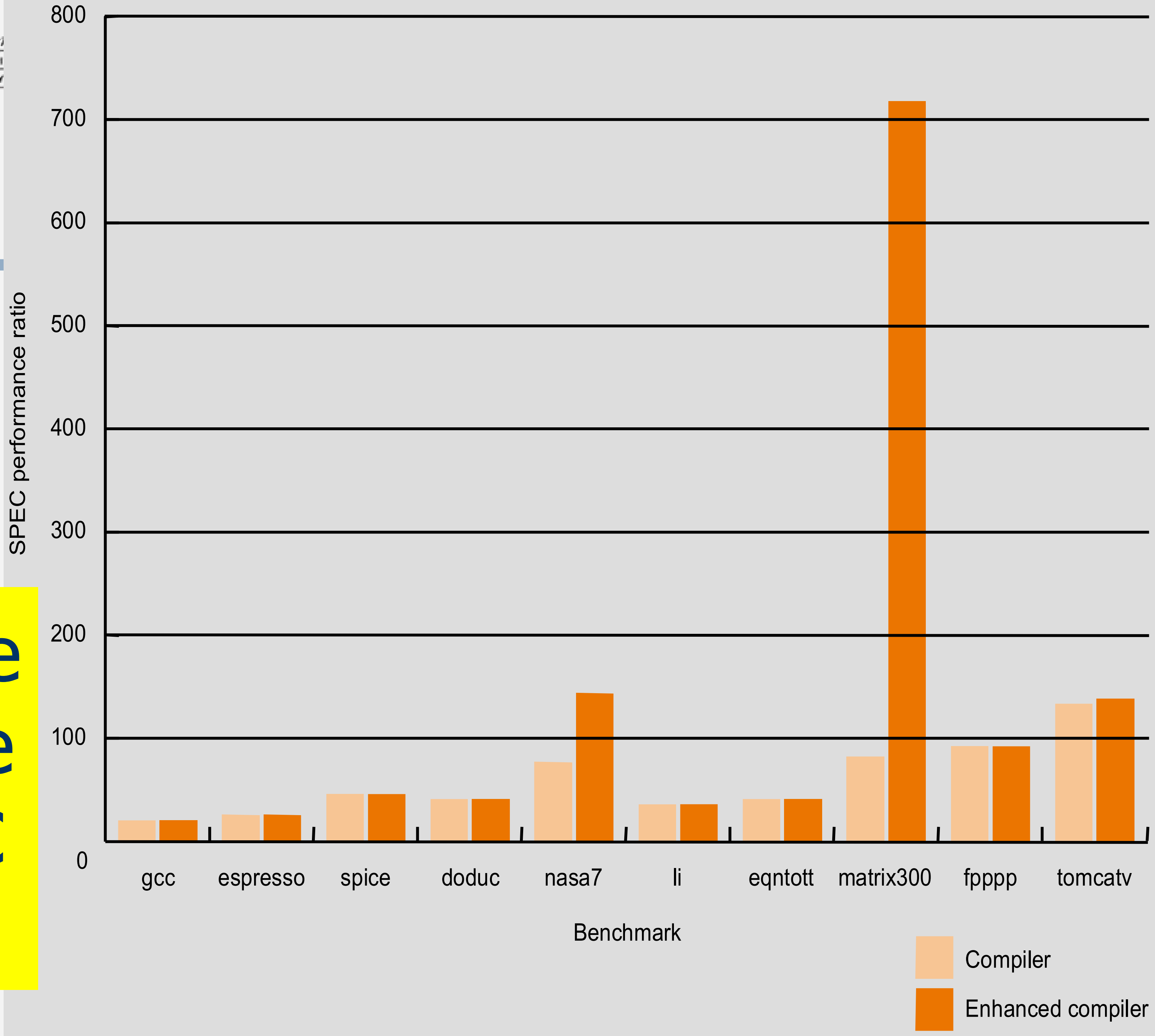
Benchmarks

- 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

- 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

Abuse in the SPEC '89

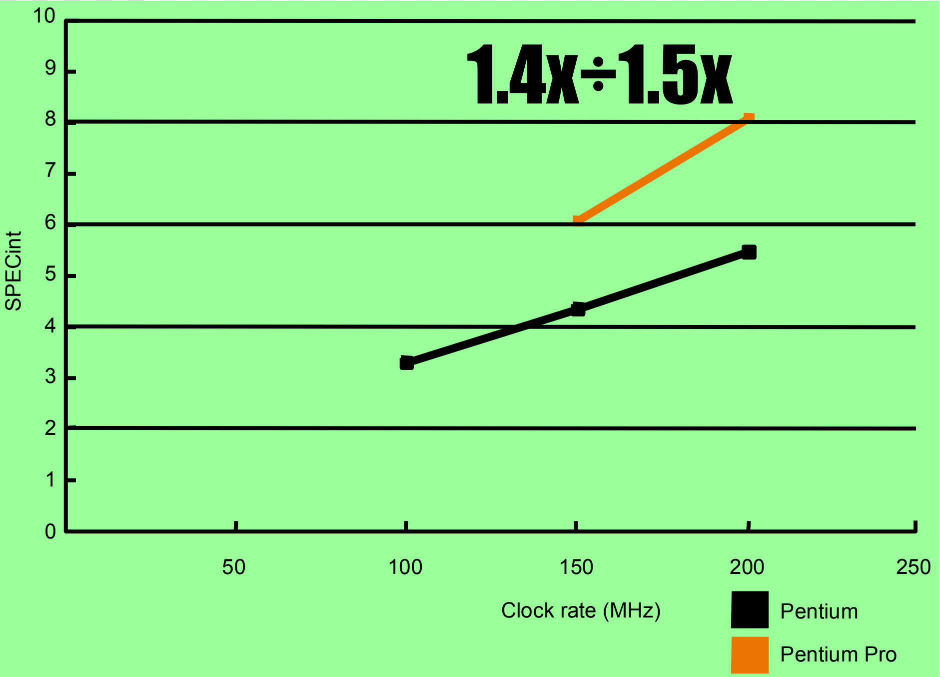


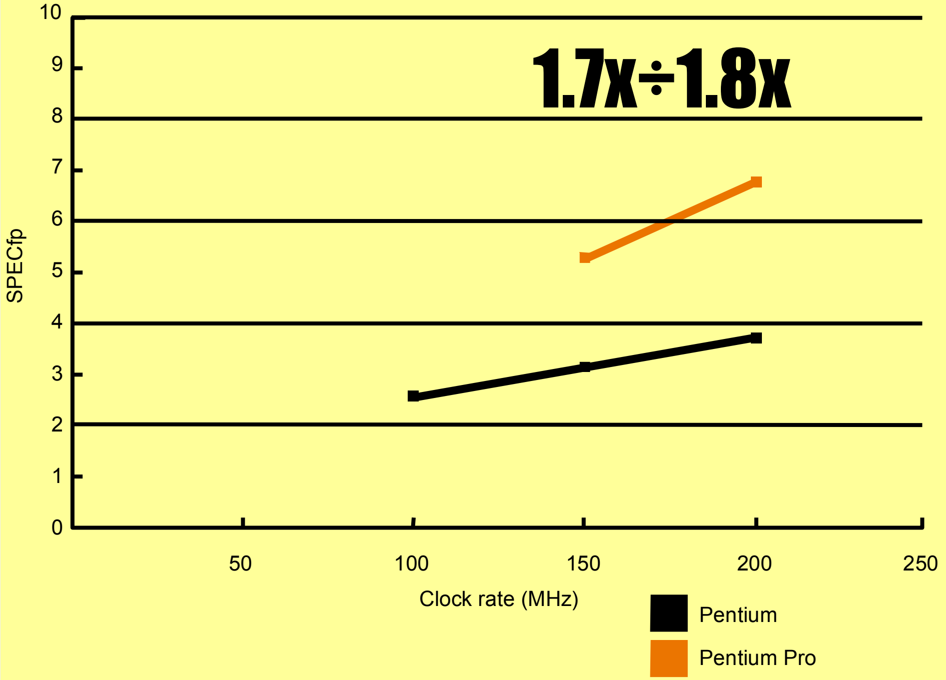
SPEC '95

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
ijpeg	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 Naiver 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
fpppp	Quantum chemistry
wave5	Plasma physics; electromagnetic particle simulation

Questions on SPEC '95

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





Amdahl's Law

Execution Time After Improvement =
Execution Time Unaffected +
(Execution Time Affected / Amount of Improvement)

Example of the Amdahl's law

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*

Proposed exercises

- 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 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?

A trivial system of equations…

$$\begin{cases} R + FPU = 100 \\ R + 0.2FPU = 33.33 \end{cases} \Rightarrow \begin{cases} -0.2R - 0.2FPU = -20 \\ R + 0.2FPU = 33.33 \end{cases}$$

$$\Rightarrow 0.8R = 13.33 \Rightarrow R = \frac{13.33}{0.8} \approx 18.8$$

Remember (1/2)

- Performance is specific to a 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

Remember (2/2)

- 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!
  - (see newspaper articles, e.g., Exercise 2.37)

For Next Week

Read the textbook

Review your notes

Do the exercises in the textbook

Read the textbook for next week:

www.unibz.it/informatic/courses/csa/schedule.htm