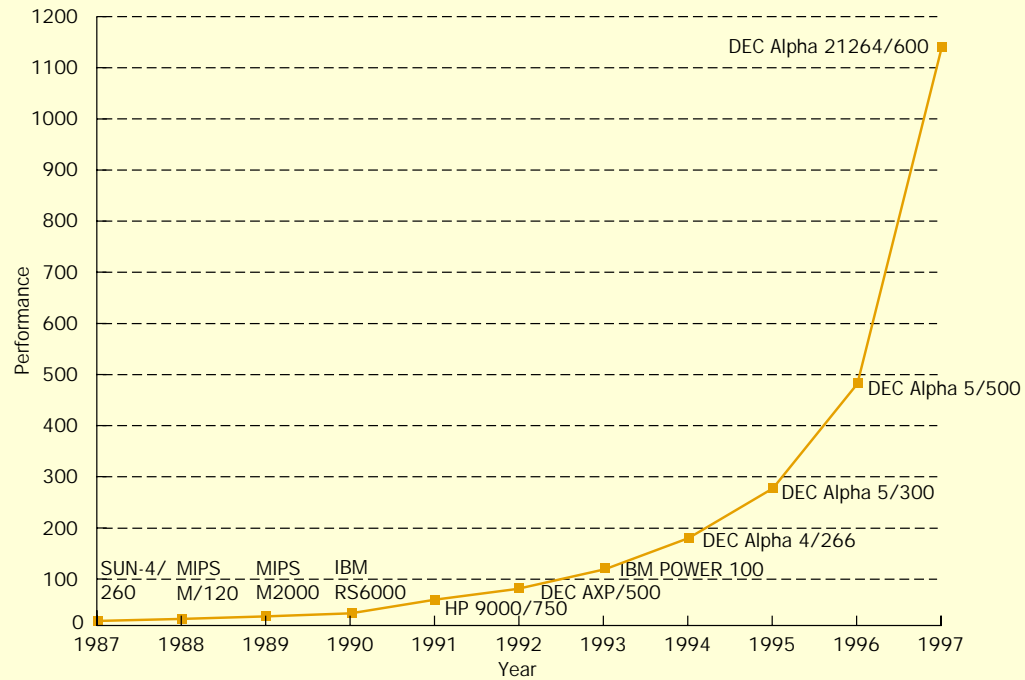
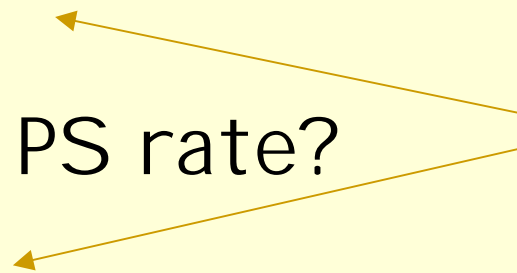


Performance Part 2



How do you judge computer performance?

- Clock speed?
 - No
 - Peak MIPS rate?
 - No
 - Relative MIPS, normalized MFLOPS?
 - Sometimes (if program tested is like yours)
 - How fast does it execute MY program
 - The best method!
- Unless ISA is same
- 

Benchmarks

- It's hard to convince manufacturers to run *your* program (unless you're a BIG customer)
- A benchmark is a set of programs that are representative of a class of problems.
 - **Microbenchmarks** - measure one feature of system
 - e.g. memory accesses or communication speed
 - **Kernel** - most compute-intensive part of applications
 - e.g. Linpack and NAS kernel b'marks (for supercomputers)
 - **Full application**:
 - SPEC (int and float) (for Unix workstations)
 - Other suites for databases, web servers, graphics,...

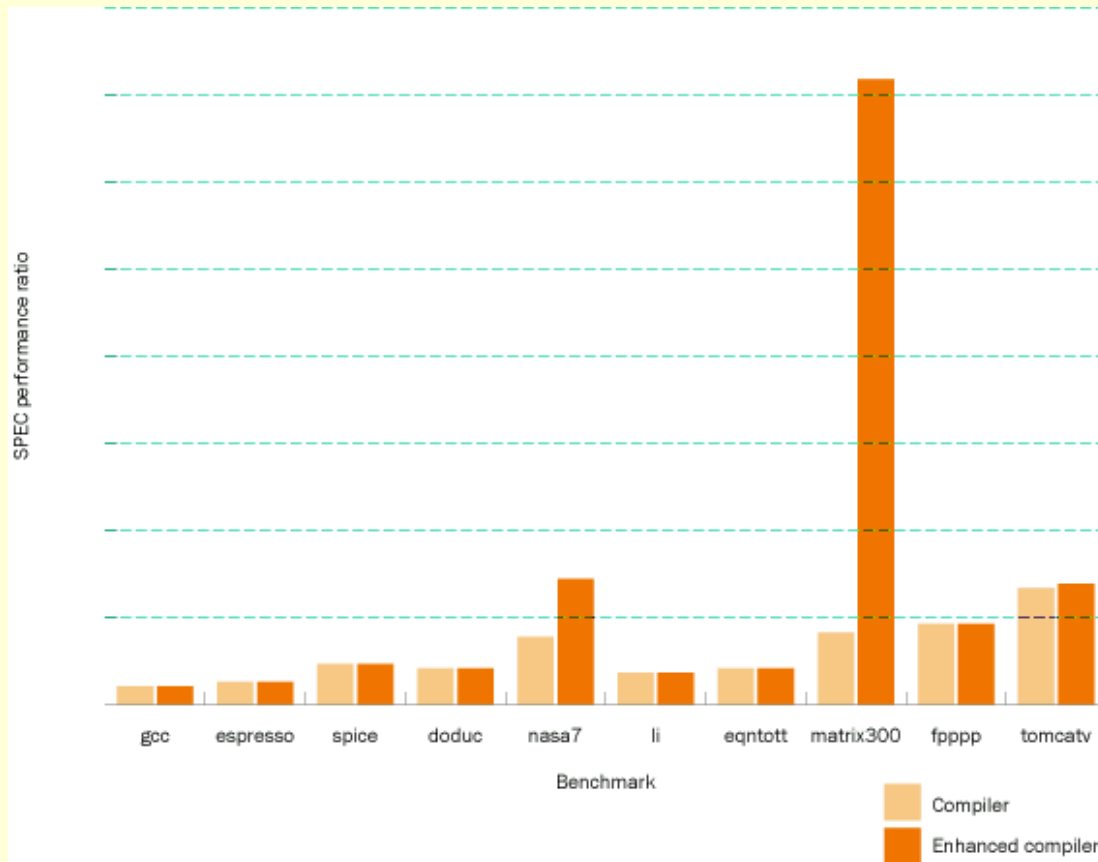
The SPEC benchmarks

SPEC = *System Performance Evaluation Cooperative*

(see www.specbench.org)

- A set of "real" applications along with strict guidelines for how to run them.
- Relatively unbiased means to compare machines.
 - Very often used to evaluate architectural ideas
- New versions in 89, 92, 95, 2000, 2004, ...
 - SPEC 95 didn't really use enough memory
- Results are speedup compared to reference machine
 - SPEC 95: Sun SPARCstation 10/40 performance = "1"
 - SPEC 2000, Sun Ultra 5 performance = "100"
- Geometric mean used to average results

SPEC89 and the compiler

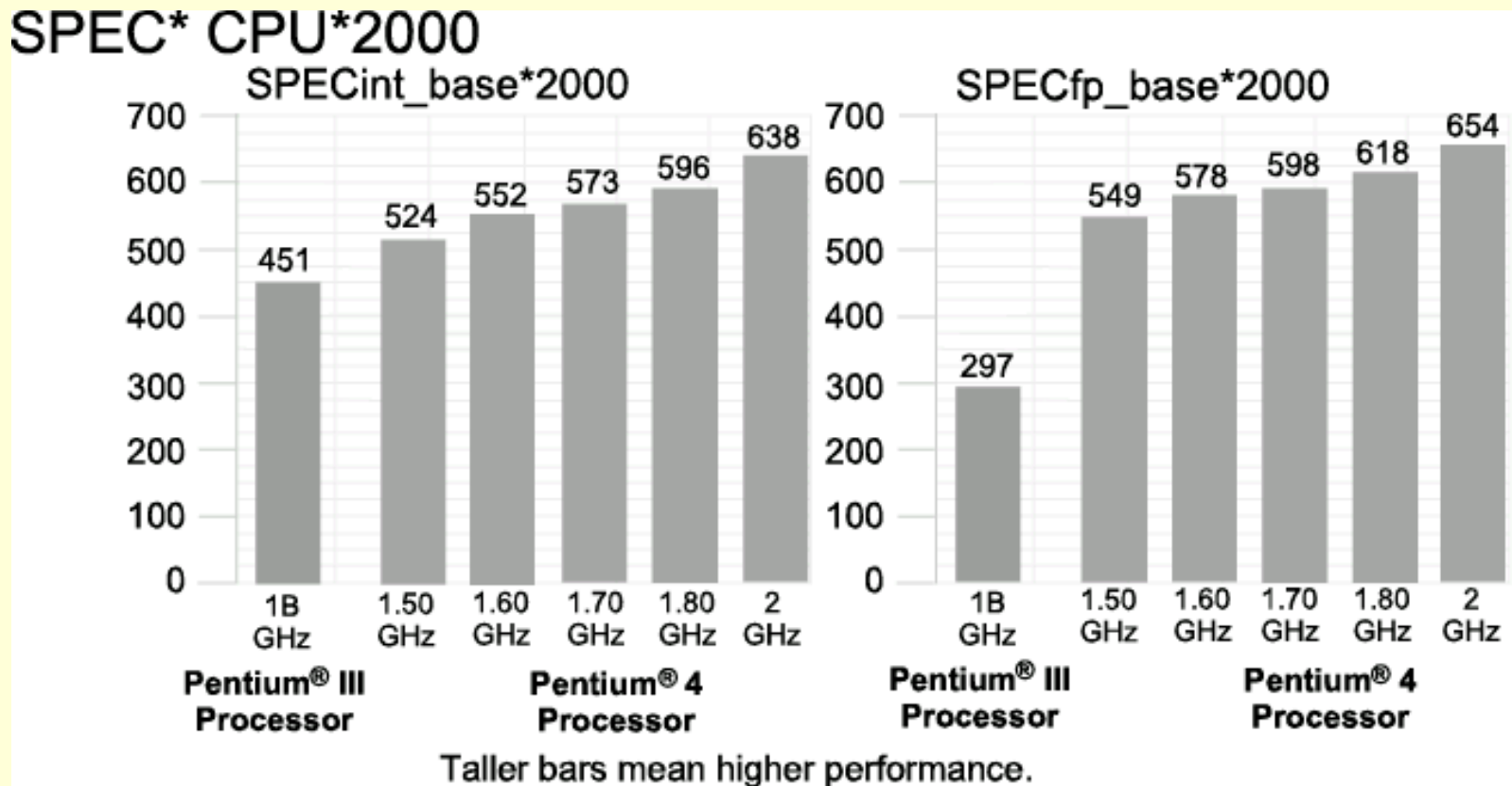


Darker bars show performance with compiler improvements (same machine as light bars)

The SPEC CPU2000 suite

- SPECint2000 – 12 C/Unix or NT programs
 - gzip and bzip2 - compression
 - gcc – compiler; 205K lines of messy code!
 - crafty – chess program
 - parser – word processing
 - vortex – object-oriented database
 - perlbnk – PERL interpreter
 - eon – computer visualization
 - vpr, twolf – CAD tools for VLSI
 - mcf, gap – “combinatorial” programs
- SPECfp2000 – 10 Fortran, 3 C programs
 - scientific application programs (physics, chemistry, image processing, number theory, ...)

SPEC on Pentium III and Pentium 4



- What do you notice?

Weighted Averages

Average of x_1, x_2, \dots, x_n is $(x_1 + x_2 + \dots + x_n) / n$

This is a special case of weighted average where all the “weights” are equal.

Suppose w_1, w_2, \dots, w_n are the relative frequency of the x_i 's.

Assume $w_1 + w_2 + \dots + w_n = 1$.

The w_i 's are called weights.

The weighted average of the x_i 's is:

$$w_1 x_1 + w_2 x_2 + \dots + w_n x_n$$

Weighted Average Example

Suppose for some store,

50% of the computers sold cost \$700

30% cost \$1000

20% cost \$1500

The fractions .5, .3 and .2 are weights.

The average cost of computers sold is

$$.5 \times \$700 + .3 \times \$1000 + .2 \times \$1500 = \$950$$

The average cost x number sold = total \$\$

Weighted averaging pitfall

The units of the numbers being averaged must correspond to what the weights represent.

Specifically, if the units are “A’s per B” (e.g. $\$/\text{computer}$) then the weights should be fractions of B’s (computers, in the example).

CPI as a weighted average

Earlier, I said CPI was derived from time, #instruction, and cycle time.

But if you know fraction of instructions that required k cycles (for all relevant k 's) you can calculate CPI using weighted average.

CPI as a weighted average

Suppose 1 GHz computer ran short program:

Load (4 cycles), Shift (1), Add (1), Store (4).

We have $\frac{1}{2}$ instructions are CPI = 4, $\frac{1}{2}$ are CPI = 1.

So weighted average CPI = $\frac{1}{2} 4 + \frac{1}{2} 1 = 2.5$

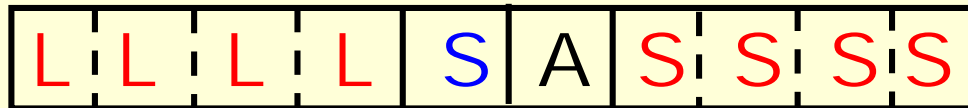
Time = 4 instructions x 2.5 CPI x 1 ns = 10 ns

But 8/10 of cycles have CPI = 4, 2/10 have CPI = 1.

Average CPI = $\frac{8}{10} \times 4 + \frac{2}{10} \times 1 = 3.4$

Time = 4 ins x 3.4 CPI x 1 ns = 13.6 ns

Which is right? Why ???



Improving Latency

- Latency is (ultimately) limited by physics.
 - e.g. speed of light
- Some improvements are incremental
 - Smaller transistors shorten distances.
 - To reduce disk access time, make disks rotate faster.
- Improvements often require new technology
 - Replace stagecoach by pony express or telegraph.
 - Replace DRAM by SRAM.
 - Once upon a time, bipolar or GaAs were much faster than CMOS.
 - But incremental improvements to CMOS have triumphed.

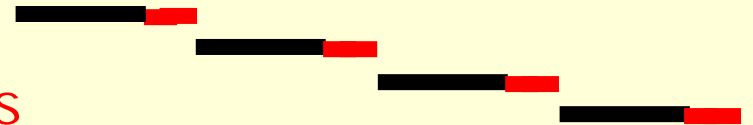
Improving Bandwidth

- You can improve bandwidth or throughput by throwing money at the problem.
 - Use wider buses, more disks, multiple processors, more functional units ...
- Two basic strategies:
 - Parallelism: duplicate resources.
 - Run multiple tasks on separate hardware
 - Pipelining: break process up into multiple stages
 - Reduces the time needed for a single stage
 - Build separate resources for each stage.
 - Start a new task down the pipe every (shorter) timestep

Pipelining

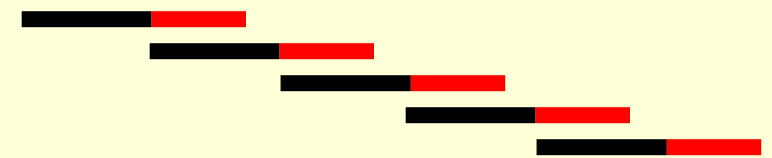
- Modern washing machine:

- Washing/rinsing and spinning done in same tub.
- Takes 15 (wash/rinse) + 5 (spin) minutes
- Time for 1 load: 20 minutes
- Time for 10 loads: 200 minutes



- Old fashioned washing machine:

- Tub for washing & rinsing (15 minutes)
- Separate spinner (10 minutes)
- Time for 1 load: 25 minutes
- Time for 10 loads: 160 minutes
(25 minutes for first load, 15 minutes for each thereafter)



Parallelism vs pipelining

Both improve throughput or bandwidth

- Automobiles: **More plants vs. assembly line**
- I/O bandwidth: **Wider buses (e.g. parallel port) vs. pushing bits onto bus faster (serial port).**
- Memory-to-processor: **wider buses vs. faster rate**
- CPU speed:
 - superscalar processor - having multiple "functional units" so you can execute more than one instructions per cycle.
 - superpipelining - using more steps than "classical" 5-stage pipeline
 - recent microprocessors use both techniques.

Latency vs Bandwidth of DRAM

- I claim, “DRAM is much slower than SRAM”
 - Perhaps 30 ns vs 1 ns access time
- But we also hear, “SDRAM is much faster than ordinary DRAM”
 - e.g. “RDRAM (from Rambus) is 5 times faster...”
- Are S(R)DRAM's almost as good as SRAM?

What are limits?

- Physics: speed of light, size of atoms, heat generated (speed requires energy loss), capacity of electromagnetic spectrum (for wireless), ...
- Limits with current technology: size of magnetic domains, chip size (due to defects), lithography, pin count.
- New technologies on the horizon: quantum computers, molecular computers, superconductors, optical computers, holographic storage, ...
- Fallacy – improvements will stop
- Pitfall – trying to predict > 5 years in future

Amdahl's Law

- Suppose:

total time = time on part A + time on part B,
you improve part A to go p times faster,

- then:

improved time = time on part A/p + time on part B.

- The impact of an improvement is limited by the fraction of time affected by the improvement.

new speed/old = $(t_A + t_B) / (t_A/p + t_B) < (t_A + t_B) / t_B$
< 1/fraction of unaffected time

Moral: Make the common case fast!!

A challenge for the future

- Latency of moving data to processor is hard to improve.
- Processors are getting faster.
- Processors must “tolerate latency”
 - Request data longer before its needed
 - Find something else to do while waiting.

Key Points

- Be careful how you specify performance
- Execution time = instructions *CPI *cycle time
- Use real applications to measure performance
- Throughput and latency are different
- Parallelism and pipelining improve throughput

Computer(s) of the day

- One-of-a-kind computers of the 1940's.
 - 1941: Z3 - Konrad Zuse (Germany)
 - programmable, special purpose, relays;
 - lost funding from Hitler
 - 1943: Colossus - Alan Turing et al.
 - special purpose electronic computer; won WWII
 - Early 1940's: ENIAC - Eckert & Mauchley at U. Penn
 - general purpose; conditional jumps;
 - programmed via plug cables
 - 80 feet long, 18,000 vacuum tubes, 1900 10-digit adds/sec
 - 1949: EDSAC - Cambridge England
 - first full-scale, operational, stored-program computer