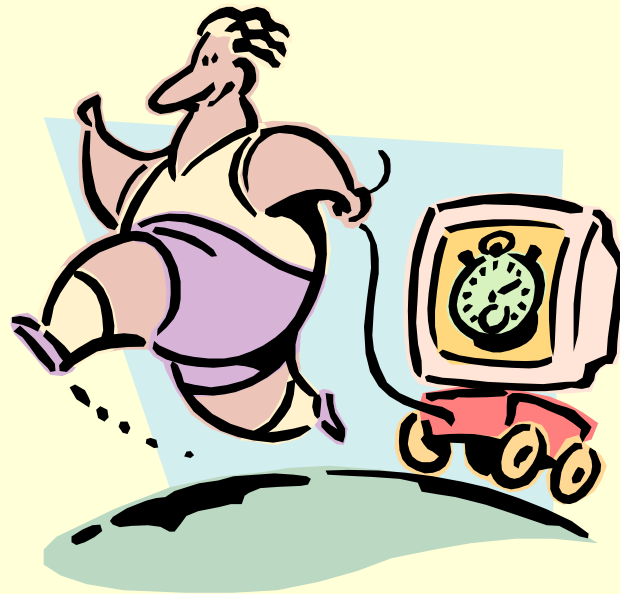
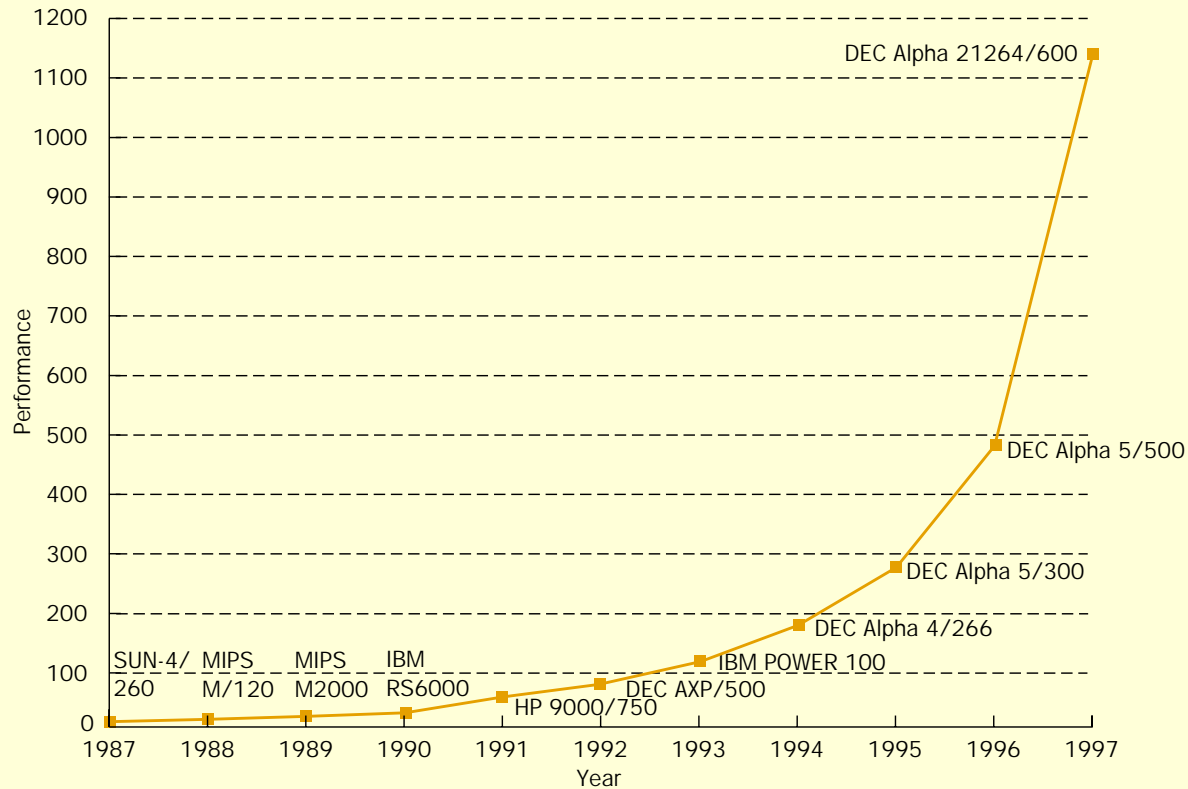


Measuring Performance Part I



Performance Marches On ...



But what *is* performance?

Time versus throughput

Vehicle	Time to Bay Area	Speed	Passengers	Throughput (pm/h)
Ferrari	3.1 hours	160 mph	2	320
Greyhound	7.7 hours	65 mph	60	3900

- Time to do the task from start to finish
 - execution time, response time, latency
- Tasks per unit time
 - throughput, bandwidth

mostly used for data movement

Time versus throughput

- Time is measured in time units/job.
- Throughput is measured in jobs/time unit.
- But “time = 1/throughput” may be false.
 - It takes 4 months to grow a tomato.
Can you only grow 3 tomatoes a year ??
 - If you run only one job at a time,
time = 1/throughput

How do you measure Execution Time?

```
> time foo
... foo's results ...
90.7u 12.9s 2:39 65%
>
```

A box containing the text $\frac{\text{user + kernel}}{\text{wallclock}}$ has an arrow pointing to the '2:39' value in the terminal output. Three arrows point from the '90.7u', '12.9s', and '2:39' values to the list items below.

- user CPU time? (time CPU spends running your code)
- total CPU time (user + kernel)? (includes op. sys. code)
- Wallclock time? (total elapsed time)
 - Includes time spent waiting for I/O, other users, ...
- Answer depends ...
 - For measuring processor speed, we can use total CPU.
 - If no I/O or interrupts, wallclock may be better
 - more precise (microseconds rather than 1/100 sec)
 - can measure individual sections of code

Performance

- For “performance”, larger should be better.
 - Time is backwards - larger execution time is *worse*.

CPU performance = 1 / total CPU time

System performance = 1 / wallclock time

- These terms only make sense if you know what program is measured ...
 - e.g. “The performance on Linpack was 200 MFLOP/S”
- and if CPU or system only works on 1 program at a time.
 - This may all change in the next few years!
- Performance’s units, “inverse seconds”, can be awkward
 - Can answer “What was performance?” by “It took 15 seconds.”

A brief study of time

CPU Time = CPU cycles executed * Cycle times

- Every conventional processor has a clock with a fixed cycle time or clock rate


Rate often measured in MHz = millions of cycles/second

Time often measured in ns (nanoseconds)

X MHz corresponds to 1000/X ns (e.g. 500 MHz \leftrightarrow 2 ns clock)

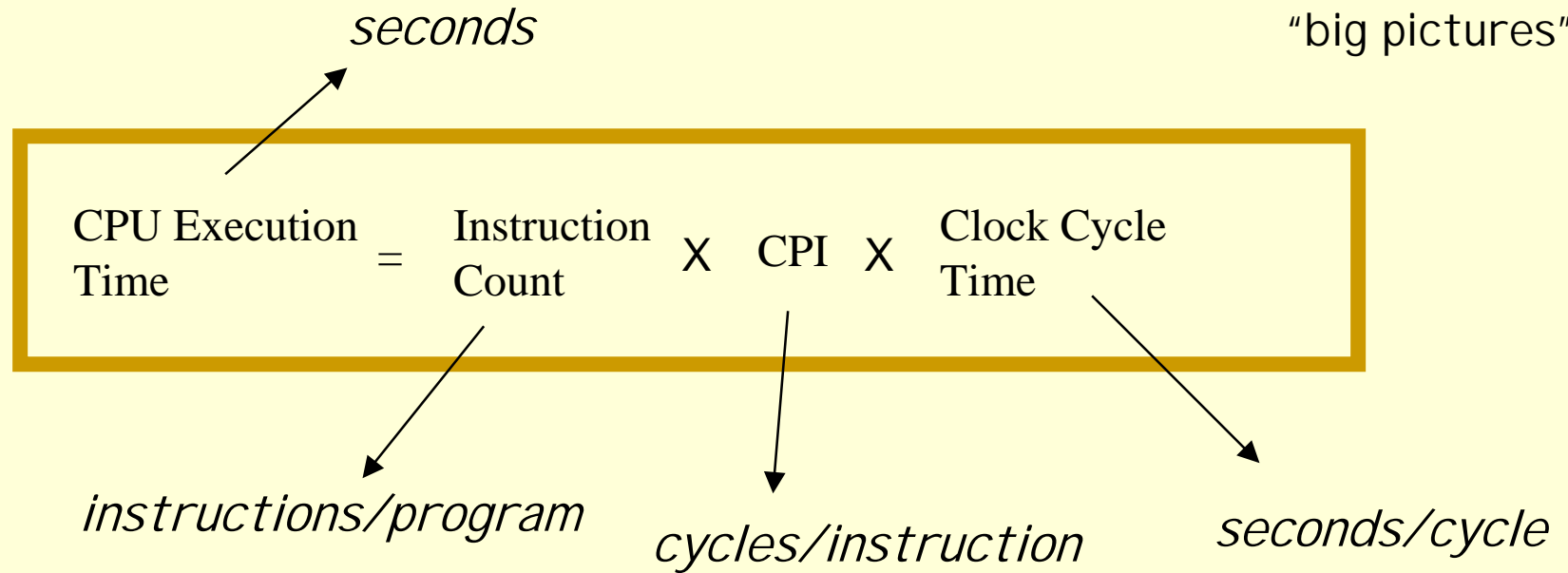
CPU cycles = Instructions executed * CPI

Average Clock Cycles per Instruction



Putting it all together

One of P&H's
"big pictures"



Note: CPI is somewhat artificial

(it's computed from the other numbers using this formula)

but it's an intuitive and useful concept.

Note: Use dynamic instruction count (#instructions executed), not static (#instructions in compiled code)

Explaining performance variation

$$\text{CPU Execution Time} = \text{Instruction Count} \times \text{CPI} \times \text{Clock Cycle Time}$$

Same machine,
different programs

Same program,
different machines,
but same ISA

Same program,
different ISA's

Comparing performance

The fundamental question:

Will computer A will run program P
faster than computer B?

- Compare clock rates?
 - Will a 1.7 GHz PC be faster than a 867 MHz Mac??
 - Not necessarily - CPI or Instruction Count may differ.
 - see <http://www.apple.com/g4/myth> (Photoshop benchmark)
- Peak MIPS rate? (MIPS = Millions of Instructions / sec)
 - PowerPC G4 *can* execute 4 instruction/cycle (CPI = 1/4)
 - 867 MHz clock → 3468 MIPS peak
 - But it doesn't necessarily execute that quickly.

Comparing performance

The fundamental question:

Will computer A will run program P
faster than computer B?

- Compare actual MIPS rate on program P?
 - $MIPS = 1 / (CPI \times \text{Cycle time})$ (in microseconds)
 - If Instruction Counts are the same, this is OK
 - E.g., comparing two implementations of same ISA
 - Otherwise, actual MIPS doesn't answer question.

Comparing performance

The fundamental question:

Will computer A will run program P
faster than computer B?

- Relative MIPS ?
 - Defined as, "How much faster is this computer than a Vax 11 model 780 (on some benchmark programs)"
 - If the benchmark is similar to P, this may give the right answer.

What about MFLOP/S?

- Millions of Floating Point Ops per Second
 - Often written MFLOPS.
- “Peak MFLOP/S” (like peak MIPS) is useless.
 - maximum float ops per cycle / cycle time (in microseconds)
- “Normalized MFLOP/S” uses conventions (e.g. “divide counts as three float ops”) so “flop count” of a program is machine-independent.
 - OK for floating-point intensive programs
 - Depends on program - a better MFLOP/S rate on program P doesn't guarantee better performance on Q.

Relative Performance

- “Computer X is r times faster than Y” means

$$\text{Perf}(X) / \text{Perf}(Y) = r \quad (\text{i.e. } \text{Time}(Y) / \text{Time}(X) = r)$$

Note the swapping of which goes on top when you use times

Comparing speeds ...

- “times faster than” (or “times as fast as”) means there’s a multiplicative factor relating quantities
 - “X was 3 time faster than Y” \rightarrow $\text{speed}(X) = 3 \text{ speed}(Y)$
- “percent faster than” implies an additive relationship
 - “X was 25% faster than Y” \rightarrow $\text{speed}(X) = (1+25/100) \text{ speed}(Y)$
- “percent slower than” implies subtraction
 - “X was 5% slower than Y” \rightarrow $\text{speed}(X) = (1-5/100) \text{ speed}(Y)$
 - “100% slower” means it doesn’t move at all !
- “times slower than” or “times as slow as” is awkward.
 - “X was 3 times slower than Y” means $\text{speed}(X) = 1/3 \text{ speed}(Y)$
 - It hints at having a measure of “slowness”
 - I’ll mostly avoid using this.

Percentages aren't intuitive!

- If X is p% faster than Y, is Y p% slower than X?
 - X is p% faster \rightarrow $\text{speed}(X) = (1+p/100) \text{ speed}(Y)$
 - so $\text{speed}(Y) = 1/(1+p/100) \text{ speed}(X)$
 - Y is p% slower \rightarrow $\text{speed}(Y) = (1-p/100) \text{ speed}(X)$

No! $1/(1+p/100)$ is not $(1 - p/100)$ (unless $p=0$)

- Suppose X is p% faster than Y and Y q% faster than Z.
Is X (p+q)% faster than Z ??

"Times faster" is easier!

X is r times faster than Y \rightarrow $\text{speed}(X) = r \text{ speed}(Y)$

$\rightarrow \text{speed}(Y) = 1/r \text{ speed}(X)$

\rightarrow Y is r times slower than X

X is r times faster than Y, & Y is s times faster than Z

$\rightarrow \text{speed}(X) = r \text{ speed}(Y) = rs \text{ speed}(Z)$

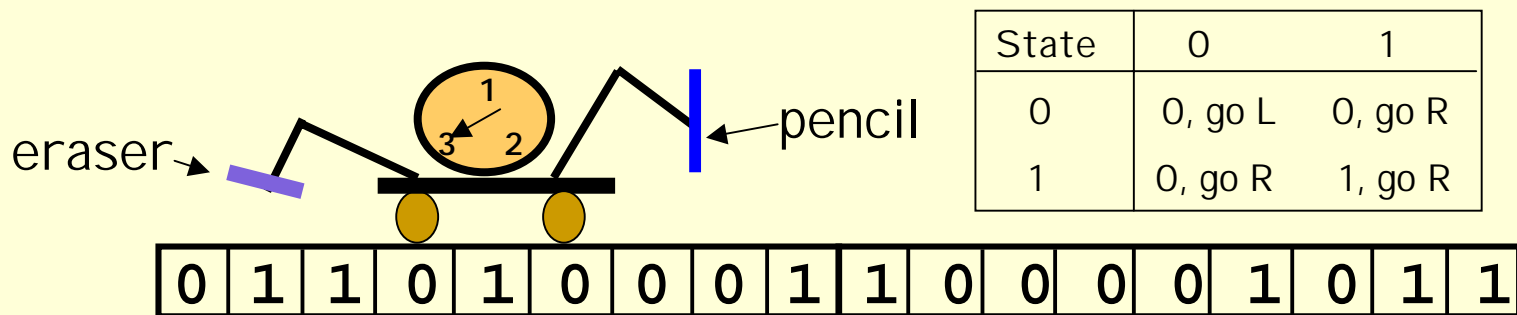
\rightarrow X is rs faster than Z

Advice: Convert "% faster" to "times faster"
then do calculation and convert back if needed.

Example: change "25% faster" to "5/4 times faster".

Machine of the day: Turing Machine

- Published 1936 by Alan Turing
- Extremely simple ISA
- “Universal” Turing machine (with about 20 states and 4 symbols) can do any computable function.
 - Program and data are written on the same tape



•Footnotes: Turing went on to work on real computer

Machine of the day: Turing Machine

- Used to prove some functions are uncomputable
- Turing machine only of theoretical interest
 - still remarkable – had elements of real computer
- Turing worked on “Bombe” computer during WW II
 - cracked German codes; greatly helped Allied victory
- After war, designed a general purpose computer (not built), proposed ideas of programming languages, neural nets, and the “Turing test”.
- Turing persecuted as homosexual; committed suicide