Two notions of performance

Aircraft	DC to Paris	Passengers
747	6 hours	500
Concorde	3 hours	125

- Which has higher performance…
  - from a passenger's viewpoint?
  - from an airline's viewpoint?

Two notions of performance

Aircraft	DC to Paris	Passengers
747	6 hours	500
Concorde	3 hours	125

- Latency vs. throughput
  - Passenger's viewpoint: hours per flight
    - time to do the task (latency, execution time, response time)
  - From an airline's viewpoint: passengers per hour
    - tasks per unit time (throughput, bandwidth)
- Latency and throughput are often in opposition

Some Definitions

- Latency is time per task (e.g. hours per flight)
- If we are primarily concerned with latency,

$$\text{Performance}(x) = \frac{1}{\text{execution\_time}(x)}$$

Bigger is better

- Throughput is number of tasks per unit time (e.g. passengers per hour)

Performance(x) = throughput(x)

Again, bigger is better

- Relative performance: "x is *N* times faster than y"

$$N = \frac{\text{Performance}(x)}{\text{Performance}(y)}$$

CPU performance

- The obvious metric: how long does it take to run a test program?
  - Aircraft analogy: how long does it take to transport 1000 passengers?

Our vocabulary	Aircraft analogy
N instructions	N passengers
c cycles per instruction	$(1/c)$ passengers per flight
t seconds per cycle	t hours per flight
Time = $N \times c \times t$ seconds	Time = $N \times c \times t$ hours

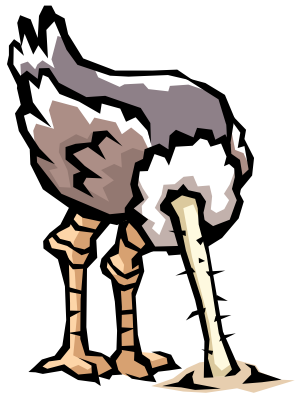
$$\text{CPU time}_{X,P} = \text{Instructions executed}_P * \text{CPI}_{X,P} * \text{Clock cycle time}_X$$

Cycles Per Instruction

Instructions Executed

- Instructions executed:
  - We are not interested in the static instruction count, or how many lines of code are in a program.
  - Instead we care about the dynamic instruction count, or how many instructions are actually executed when the program runs.
- There are three lines of code below, but the number of instructions executed would be 2001.

```
          li   $a0, 1000
ostrich:  sub  $a0, $a0, 1
          bne  $a0, $0, ostrich
```



CPI

- The average number of clock cycles per instruction, or CPI, is a function of the machine and program.
  - The CPI depends on the actual instructions appearing in the program—a floating-point intensive application might have a higher CPI than an integer-based program.
  - It also depends on the CPU implementation. For example, a Pentium can execute the same instructions as an older 80486, but faster.
- In CS231, we assumed each instruction took one cycle, so we had CPI = 1.
  - The CPI can be >1 due to memory stalls and slow instructions.
  - The CPI can be <1 on machines that execute more than 1 instruction per cycle (superscalar).

Clock cycle time

- One “cycle” is the minimum time it takes the CPU to do any work.
  - The clock cycle time or clock period is just the length of a cycle.
  - The clock rate, or frequency, is the reciprocal of the cycle time.
- Generally, a higher frequency is better.
- Some examples illustrate some typical frequencies.
  - A 500MHz processor has a cycle time of 2ns (nanoseconds).
  - A 2GHz (2000MHz) CPU has a cycle time of just 0.5ns

Execution time, again

$$\text{CPU time}_{X,P} = \text{Instructions executed}_P * \text{CPI}_{X,P} * \text{Clock cycle time}_X$$

- The easiest way to remember this is match up the units:

$$\frac{\text{Seconds}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} * \frac{\text{Clock cycles}}{\text{Instructions}} * \frac{\text{Seconds}}{\text{Clock cycle}}$$

- Make things faster by making any component smaller!!

	Program	Compiler	ISA	Organization	Technology
Instruction Executed					
CPI					
Clock Cycle TIme					

- Often easy to reduce one component by increasing another

Example 1: ISA-compatible processors

- Let's compare the performances two x86-based processors.
  - An 800MHz AMD Duron, with a CPI of 1.2 for an MP3 compressor.
  - A 1GHz Pentium III with a CPI of 1.5 for the same program.
- Compatible processors implement identical instruction sets and will use the same executable files, with the same number of instructions.
- But they implement the ISA differently, which leads to different CPIs.

$$\text{CPU time}_{AMD,P} = \text{Instructions}_P * \text{CPI}_{AMD,P} * \text{Cycle time}_{AMD}$$

$$=$$

$$=$$

$$\text{CPU time}_{P3,P} = \text{Instructions}_P * \text{CPI}_{P3,P} * \text{Cycle time}_{P3}$$

$$=$$

$$=$$

Example 2: Comparing across ISAs

- Intel's Itanium (IA-64) ISA is designed facilitate executing multiple instructions per cycle. If an Itanium processor achieves an average CPI of .3 (3 instructions per cycle), how much faster is it than a Pentium4 (which uses the x86 ISA) with an average CPI of 1?

a) Itanium is three times faster
b) Itanium is one third as fast
c) Not enough information

Improving CPI

- Some processor design techniques improve CPI
  - Often they only improve CPI for certain types of instructions

$$CPI = \sum_{i=1}^{n} CPI_i \times F_i$$

where F_i = fraction of instructions of type *i*

First Law of Performance:

Make the common case fast

Example: CPI improvements

- Base Machine:

Op Type	Freq (F_i)	CPI_i	contribution to CPI
ALU	50%	3	
Load	20%	6	
Store	20%	3	
Branch	10%	2	

- How much faster would the machine be if:
  - we added a cache to reduce average load time to 3 cycles?
  - we added a branch predictor to reduce branch time by 1 cycle?
  - we could do two ALU operations in parallel?

Amdahl's Law

- Amdahl's Law states that optimizations are limited in their effectiveness.

$$\text{Execution time after improvement} = \frac{\text{Time affected by improvement}}{\text{Amount of improvement}} + \text{Time unaffected by improvement}$$

- For example, doubling the speed of floating-point operations sounds like a great idea. But if only 10% of the program execution time T involves floating-point code, then the overall performance improves by just 5%.

$$\text{Execution time after improvement} = \frac{0.10\ T}{2} + 0.90\ T = 0.95\ T$$

- Second Law of Performance:

Make the fast case common

Summary

- Performance is one of the most important criteria in judging systems.

- Our main performance equation explains how performance depends on several factors related to both hardware and software.

$$\text{CPU time}_{X,P} = \text{Instructions executed}_{P} * \text{CPI}_{X,P} * \text{Clock cycle time}_{X}$$

- It can be hard to measure these factors in real life, but this is a useful guide for comparing systems and designs.

- Amdahl's Law also tells us how much improvement we can expect from specific enhancements.