

# Acknowledgements

---

Class notes based upon

- Patterson & Hennessy: Book & Lecture Notes

- Patterson's 1997 course notes (U.C. Berkeley CS 152, 1997)

- Tom Fountain 2000 course notes (Stanford EE182)

- Michael Wahl 2000 lecture notes (U. of Siegen CS 3339)

- Ben Dugan 2001 lecture notes (UW-CSE 378)

- Professor Scott Hauck lecture notes (UW EE 471)

# Why are you here?

---

# What is Computer Architecture?

---

# What things are important when buying a computer?

---

(What features do you look for when buying one?)

- Price

- Power / Heat

→ I/O

in and out

→ storage space(B)

- Speed → Frames/s → Physical Dimensions

In/s

Disk speed → rotational velocity  
access time.

- Amount of memory (vs. disk in latency)  
types of memory (CPU cache, video mem, HD)

- Good sound.

- Features / complexity of CPU. → instruction set

- Compatibility. → Noise

- Reliability. → use case?

# Computer “Performance”

---

MIPS (Million Instructions Per Second) vs. MHz (Million Cycles Per Second)

Throughput (jobs/seconds) vs. Latency (time to complete a job)

Measuring, Metrics, Evaluation – what is “best”?



3.09 GHz  
Pentium 4

The PowerBook G4 outguns Pentium III-based notebooks by up to 30 percent.\*

\* Based on Adobe Photoshop tests comparing a 500MHz PowerBook G4 to 850MHz Pentium III-based portable computers



Hyper  
Pipelined  
Technology

## Performance Example: Planes

---

Airplane	Passenger Capacity	Cruising Range (miles)	Cruising Speed (mph)	Passenger Throughput (passengermile/hour)
Boeing 777	375	4630	610	228,750
Boeing 747	470	4150	610	286,700
Concorde	132	4000	1350	178,200
Douglas DC-8	146	8720	544	79,424

Which is the “best” plane?

Which gets one passenger to the destination first?

Which moves the most passengers?

Which goes the furthest?

Which is the “speediest” plane (between Seattle and NY for example)?

Latency: how fast is one person moved?

Throughput: number of people per time moved?

# Computer Performance

---

Primary goal: execution time (time from program start to program completion)

$$Performance = \frac{1}{ExecutionTime}$$

To compare machines, we say “X is n times faster than Y”

$$n = \frac{Performance_x}{Performance_y} = \frac{ExecutionTime_y}{ExecutionTime_x}$$

Example: Machine *Orange* and *Grape* run a program

Orange takes 5 seconds, Grape takes 10 seconds

$$n = 10/5 = 2$$

Orange is 2 times faster than Grape

---

# Execution Time

---

## 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

## Example: Unix “time” command

```
fpga.olin.edu> time javac CircuitViewer.java  
3.370u 0.570s 0:12.44 31.6%
```

## Our focus: user CPU time

time spent executing the lines of code that are "in" our program



# CPU Time

---

$$\text{CPU execution time for a program} = \text{CPU clock cycles for a program} * \text{Clock period}$$

$$\text{CPU execution time for a program} = \text{CPU clock cycles for a program} * \frac{1}{\text{Clock rate}}$$

Application example:

A program takes 10 seconds on computer *Orange*, with a 400MHz clock. Our design team is developing a machine *Grape* with a much higher clock rate, but it will require 1.2 times as many clock cycles. If we want to be able to run the program in 6 second, how fast must the clock rate be?

ORANGE  $10s = \frac{x}{400}$

ERIC SEZ  $\rightarrow 800 \text{ MHz}$

# CPI

---

How do the # of instructions in a program relate to the execution time?

$$\begin{array}{ccccc} \text{CPU clock cycles} & & & & \text{Average Clock} \\ \text{for a program} & = & \text{Instructions} & * & \text{Cycles per Instruction} \\ & & \text{for a program} & & \text{(CPI)} \end{array}$$

$$\begin{array}{ccccccc} \text{CPU execution time} & & & & & & 1 \\ \text{for a program} & = & \text{Instructions} & * & \text{CPI} & * & \text{Clock rate} \\ & & \text{for a program} & & & & \end{array}$$

## CPI Example

---

Suppose we have two implementations of the same instruction set (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?

$$\text{CPU Clock Cycles}_A = I \times 2.0$$

$$\text{CPU Clock Cycles}_B = I \times 1.2$$

$$\text{CPU Time}_A = I \times 2.0 \times 10\text{ns} = 20 \times I\text{ns}$$

$$B = I \times 1.2 \times 20\text{ns} = 24 \times I\text{ns}$$

$$\frac{24 \times I\text{ns}}{20 \times I\text{ns}} = \boxed{1.2 \times}$$

# Computing CPI

---

Different types of instructions can take very different amounts of cycles  
Memory accesses, integer math, floating point, control flow

$$CPI = \sum_{types} (Cycles_{type} * Frequency_{type})$$

Instruction Type	Type Cycles	Type Frequency	Cycles * Freq
ALU	1	50%	.5
Load	5	20%	1.0
Store	3	10%	0.3
Branch	2	20%	0.4
CPI:			2.2

## CPI & Processor Tradeoffs

---

Instruction Type	Type Cycles	Type Frequency
ALU	1	50%
Load	5	20%
Store	3	10%
Branch	2	20%

How much faster would the machine be if:

1. A data cache reduced the average load time to 2 cycles?

$$\begin{array}{l} \text{OLD} = 2.2 \\ \text{NEW} = 1.6 \end{array} \quad = 1.375 \times$$

2. Branch prediction shaved a cycle off the branch time?

$$\frac{2.2}{2} = 1.1 \times$$

3. Two ALU instructions could be executed at once?

$$\frac{2.2}{1.8} = 1.22 \times$$

## Warning 1: Amdahl's Law

---

The impact of a performance improvement is limited by what is NOT improved:

$$\text{Execution time after improvement} = \text{Execution time of unaffected} + \text{Execution time affected} * \frac{1}{\text{Amount of improvement}}$$

Example: Assume a program runs in 100 seconds on a machine, with multiply responsible for 80 seconds of this time. How much do we have to speed up multiply to make the program run 4 times faster?

$$\text{OLD} = 100s \rightarrow 25s. \Rightarrow 20s + 80/N$$

$$N = 16.$$

5 times faster?

20s - can't do.