January 23

- Slides available as HTML
- A&S “Free Lunch Program”
- Book on reserve in THE library (Math/Physics)

Which of these airplanes has the best performance?

Airplane	Passengers	Range (mi)	Speed (mph)	Throughput
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

Which communications network is best?

- 56kb modem (56k bits / second, WW)
- Road Runner (1.5M bits / second, WW)
- Zip Disk + Sneakers (2G bits / 5 minutes, feet)
- DLT (Digital Linear Tape) + FedEx
  (1T bit/12 hours)

TIME is THE measure!

- **Response Time (latency)**
  - **— 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?**
- **Throughput**
  - **— How many jobs can the machine run at once?**
  - **— What is the average execution rate?**
  - **— How much work is getting done?**

If we upgrade a machine with a new processor what do we increase?

If we add a new machine to the lab what do we increase?

What kind of time?

- **Wall-clock Time**
  - **counts everything *(disk and memory accesses, I/O , etc.)***
  - **a useful number, but sometimes not good for comparison or analysis 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**

DANGER Will Robinson!

- All this focus on CPU time can SERIOUSLY distort our world view…
- SYSTEM designers (as opposed to CPU designers) need to focus on the USER EXPERIENCE.

“Performance”

- For some program running on machine X,

  $$\text{Performance}_X = 1 / \text{Execution time}_X$$

- "X is n times faster than Y“

  $$\text{Performance}_X \ / \ \text{Performance}_Y = n$$

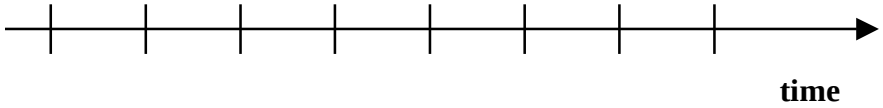
  Problem:
  - **machine A runs a program in 20 seconds**
  - **machine B runs the same program in 25 seconds**

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):



- cycle time = time between ticks = seconds per cycle
- clock rate (frequency) = cycles per second (1 Hz. = 1 cycle/sec)

A 200 MHz. clock has a $\frac{1}{200 \times 10^6} \times 10^9 = 5$ nanoseconds cycle time

How to improve performance?

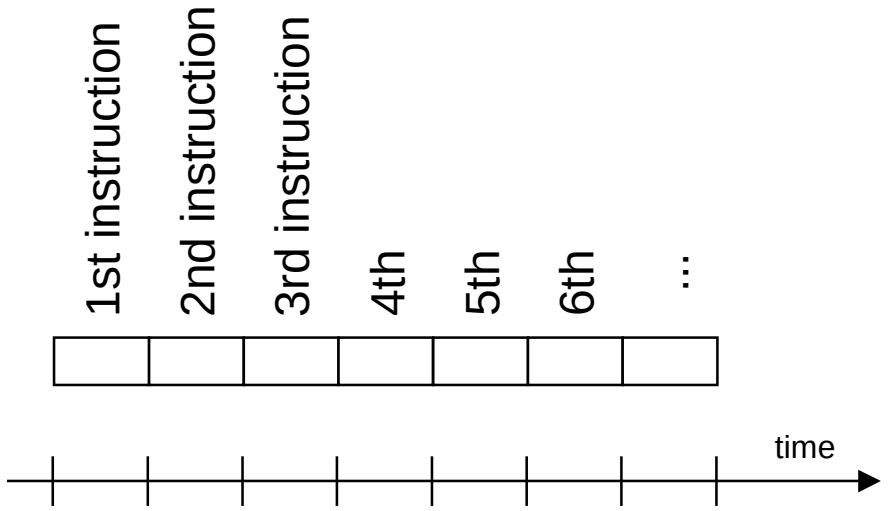
$$\frac{\text{seconds}}{\text{program}} = \frac{\text{cycles}}{\text{program}} \times \frac{\text{seconds}}{\text{cycle}}$$

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

_____ the # of required cycles for a program, or
_____ the clock cycle time or, said another way,
_____ the clock rate.

How many cycles are required for a program?

- Could assume that # of cycles = # of instructions

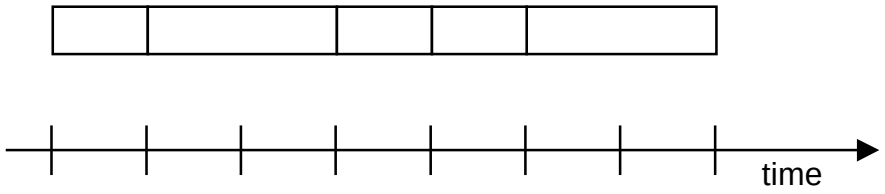


WRONG!

different instructions take different amounts of time on different machines.

WHY?

Instructions take differing numbers of cycles



- **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 (cycles per instruction)

    a floating point intensive application might have a higher CPI
  - MIPS (millions of instructions per second)

    this would be higher for a program using simple instructions

Do any of these equal performance?

# of cycles to execute program?

# of instructions in program?

# 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.

CPI Example

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

If two machines have the same ISA which of our quantities (e.g., clock rate, CPI, execution time, # of instructions, MIPS) will always be identical?

of instructions example

A compiler designer is trying to decide between two 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.

Which sequence will be faster? How much?
What is the CPI for each sequence?

MIPS example

- **Two different compilers are being tested for a 100 MHz. machine with three different classes of instructions: Class A, Class B, and Class 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, 1 million Class B instructions, and 1 million Class C instructions.

  The second compiler's code uses 10 million Class A instructions, 1 million Class B instructions, and 1 million Class C instructions.

- **Which sequence will be faster according to MIPS?**
- **Which sequence will be faster according to execution time?**