

# Performance

Ref: Chapter 2

1

---

---

---

---

---

---

---

## What is performance?

- Which computer performs better?
  - It depend on what you want to perform.
- What does *faster* mean?
  - response time vs. throughput

2

---

---

---

---

---

---

---

## You go to the grocery store...

- What is important?
  - response time: how fast the cashier rings you up once you've made it to the cashier.
  - throughput: how many customers are processed per hour.

3

---

---

---

---

---

---

---

For now – focus on response time.

- Response time is also called execution time.
- Maximum performance means minimum execution time:

$$Performance_x = \frac{1}{ExecutionTime_x}$$

4

---

---

---

---

---

---

---

---

## Comparing Performance

“ $x$  is  $n$  times faster than  $y$ ”

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

5

---

---

---

---

---

---

---

---

## Measuring Performance

- My cashier is very speedy, except she doesn't know what to do with fruit.
- Her performance (relative to other cashiers) depends on what is in my cart.

$x$  is  $n$  times faster than  $y$ , but at what? fruit?

6

---

---

---

---

---

---

---

---

## Time

- Elapsed Time: wall clock time.
- CPU time: the time the CPU spends on the program.
  - user CPU time: the time spent in my program.
  - system CPU time: time spent in the O.S. (in service of my program).
  - CPU times do not include time waiting for I/O.

7

---

---

---

---

---

---

---

## The Unix `time` command

`time` tells you the breakdown of your program in user, system and elapsed times.

```
> time gcc -o exec exec.c
0.13user 0.09system 0:03.25elapsed 6%CPU
```

8

---

---

---

---

---

---

---

## User vs. Designer

Users worry about how long a program will take.

Designers worry about how long individual operations (addition, multiplication, etc) take.

Programmers must worry about both!

9

---

---

---

---

---

---

---

## Computer Clock

- A clock in the computer runs at a constant rate, this is used as the base unit of time for everything the computer does.
- Abstraction: *everything is bits*
  - The clock is a never-ending sequence of alternating **1s** and **0s**.
  - The clock rate depends on how long it takes between **1s** (or **0s**).

10

---

---

---

---

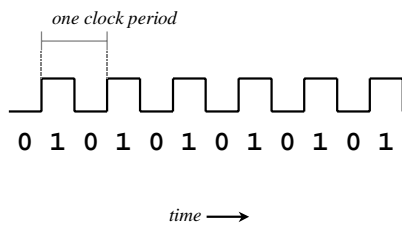
---

---

---

---

## A graphical view of a clock



11

---

---

---

---

---

---

---

---

## Clock Cycle and Clock Rate

- Clock Period or Cycle is the basic unit of measure – the time interval between when the clock changes to a 1 and the next time it changes to a 1.
- Clock rate is how many clock cycles happen in one second.  
Hertz (Hz) is *cycles per second*.

12

---

---

---

---

---

---

---

---

## So fast it hertz

<u>Clock Cycle</u>	<u>Clock Rate</u>
1ms	1000 Hz
1us	1 MHz
5ns	200 MHz
1ns	1 GHz

13

---

---

---

---

---

---

---

## Programs and cycles

*CPU Execution Time =*

*CPU cycles in program \* cycle time*

↓  
how many cycles will  
the program take

↓  
how long does  
each cycle take

14

---

---

---

---

---

---

---

## Quiz

Computer A has a 400MHz clock.  
Our program takes 10 seconds on A.

We want to make the program run in 6  
seconds – how fast should the clock be?

15

---

---

---

---

---

---

---

Find the total number of clock cycles

$$\#cycles = CPUtime * clock\ rate$$

$$\#cycles = 10s * 400x10^6$$

$$\#cycles = 4000x10^6$$

16

---

---

---

---

---

---

---

---

Use same equation – solve for new clock rate

$$\#cycles = CPUtime * clock\ rate$$

$$4000x10^6 = 6s * clock\_rate_{new}$$

$$clock\_rate_{new} = 666MHz$$

17

---

---

---

---

---

---

---

---

A realistic twist on the problem

- Suppose in order to move to a faster clock we need to make a change in CPU design.
- The change will mean that we have 1.2 times as many cycles to complete the same program.
- What does the clock rate need to be to make the program run in 6 seconds?
  - should this be more or less than 666MHz?

18

---

---

---

---

---

---

---

---

## Instruction Set

- Each processor has a set of instructions that it can execute.
  - programs are just sequences of these instructions.
- Some instructions take longer than others (some do more *work!*).
- Instruction time is often measured in cycles.

19

---

---

---

---

---

---

---

---

## CPI: Cycles per Instruction

- CPI is the *average* number of cycles per instruction.
  - average over the instructions executed in a specific program.
  - You can't just take the average over all the instructions in the instruction set!

20

---

---

---

---

---

---

---

---

## IPS: Items per Scan

- Average # of items the cashier can process per scan.
  - 2 identical cans of spam might take only 1 scan (for an experienced spam scanner).
  - a bunch of bananas might take only 1 scan.
- Given a cart full of items, the IPS is the total number of items divided by the total number of scans.
- IPS will depend on what is in the cart!
- CPI depends on what is in the program!

21

---

---

---

---

---

---

---

---

## New Relationship

$$\text{CPU Execution Time} = \frac{\# \text{Instructions in program} * \text{CPI}}{\text{Clock Rate}}$$

22

---

---

---

---

---

---

---

## Another Problem

- 2 Computers: A and B.
- Same instruction set (same program will run on both computers).
- A has cycle time of 1ns and CPI of 2.0 for program P.
- B has cycle time of 2ns and CPI of 1.2 for program P.
- Which machine is faster (and by how much)?

23

---

---

---

---

---

---

---

## Computer A

$$\begin{aligned} \text{CPU Execution Time} &= \\ \frac{\# \text{Instructions in program} * \text{CPI}}{\text{Clock Rate}} &= \\ \# \text{Instructions in program} * \text{CPI} * \text{Clock Cycle} & \end{aligned}$$

$$\text{Time}_A = p * 2.0 * 1\text{ns} = 2.0p \text{ ns}$$

NOTE: # instructions in program P is  $p$

24

---

---

---

---

---

---

---

## Computer B

CPU Execution Time =

$$\frac{\# \text{Instructions in program} * \text{CPI}}{\text{Clock Rate}} =$$

$$\# \text{Instructions in program} * \text{CPI} * \text{Clock Cycle}$$

$$\text{Time}_B = p * 1.2 * 2 \text{ ns} = 2.4p \text{ ns}$$

25

---

---

---

---

---

---

---

---

## A vs. B on program P

A:  $2.0p$  ns

B:  $2.4p$  ns

$$\frac{\text{Performance}_A}{\text{Performance}_B} = \frac{\text{Time}_B}{\text{Time}_A} = \frac{2.4}{2.0} = 1.2$$

A is 1.2 times faster than B on program P

26

---

---

---

---

---

---

---

---

## Important Relationship!

$$\text{Time} = \frac{\text{Instructions}}{\text{program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Cycle}}$$

Knowing only one of these doesn't tell you all you need to know!

*My computer has a 100GHz clock!  
(the only instruction is a NOP)*

27

---

---

---

---

---

---

---

---

$$Time = \frac{\text{Instructions}}{\text{program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Cycle}}$$

Called the *Instruction Count* or **IC**

Depends on the instruction set architecture and on the compiler.

The same 'C' program could result in dramatically different numbers of instructions on different machines/compilers!

28

---

---

---

---

---

---

---

---

## A Relevant Question

Given a program, how can we tell what the instruction count is?

### An Irrelevant Question

Which is the coolest form of transportation?

- a skateboard with blood on it.
- a new VW Beetle with a daisy in the vase.
- one of those new-fangled motorized scooters.

29

---

---

---

---

---

---

---

---

## Instruction Count

IC is a count of the instructions executed at runtime!

You have to watch the program flow when running (called profiling) or simulate the entire program.

You can't tell just by looking at the program (even at the machine code).

Yell if you don't understand this...(it's important!)

30

---

---

---

---

---

---

---

---