
Chapter 4 – Assessing and Understanding Performance

Why know about performance

❑ Purchasing Perspective:

- **Given a collection of machines, which has the**
 - Best Performance?
 - Lowest Price?
 - Best Performance/Price?

❑ Design Perspective:

- **Faced with design options, which has the**
 - Best Performance Improvement?
 - Lowest Cost?
 - Best Performance/Cost ?

❑ Both require

- **Metric for evaluation**
- **Basis for comparison**

Computer Performance: TIME, TIME, TIME

□ 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?

Book's Definition of 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:**

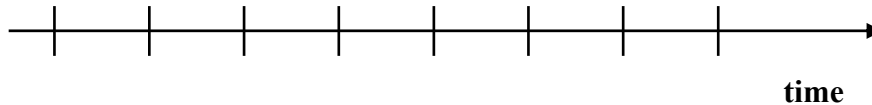
- machine A runs a program in 20 seconds
- machine B runs the same program in 25 seconds

Clock 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

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

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

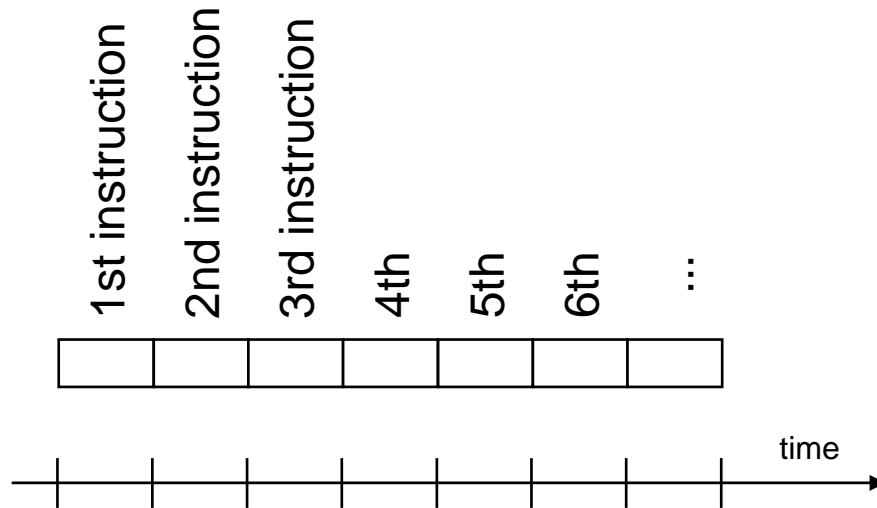
_____ ↓ the # of required cycles for a program, or

_____ ↓ the clock cycle time or, said another way,

_____ ↑ the clock rate.

How many cycles are for a program?

- ❑ Could assume that # of cycles = # of instructions

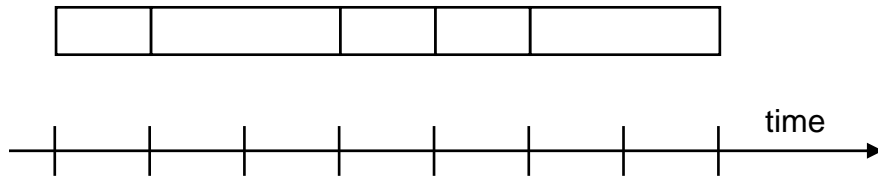


This assumption is incorrect,

different instructions take different amounts of time on different machines.

Why? hint: remember that these are machine instructions, not lines of C code

Different numbers of cycles for different instructions



- ❑ 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)