CSE-413 Computer Architecture
Lecture 2

Performance

Defining Performance

- If you were running a program on two different desktop computers, you'd say that the faster one is the desktop computer that gets the job done first.

- If you were running a datacenter that had several servers running jobs submitted by many users, you'd say that the faster computer was the one that completed the most jobs during a day.

Cont.

- As an individual computer user, you are interested in reducing **response time—the** time between the start and completion of a task—also referred to as **execution time.**
- This includes disk accesses, memory accesses, I/O activities, operating system overhead, CPU execution time, etc.
- Datacenter managers are often interested in increasing **throughput** the total amount of work done in a given time.
- Decreasing response time almost always improves throughput.

Cont.

To maximize performance, we want to minimize response time or execution time for some task. Thus, we can relate performance and execution time for a computer X:

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

Cont.

This means that for two computers X and Y, if the performance of X is greater than the performance of Y, we have

$$\text{Performance}_X > \text{Performance}_Y$$

$$\frac{1}{\text{Execution time}_X} > \frac{1}{\text{Execution time}_Y}$$

$$\text{Execution time}_Y > \text{Execution time}_X$$

That is, the execution time on Y is longer than that on X, if X is faster than Y.

Cont.

In discussing a computer design, we often want to relate the performance of two different computers quantitatively. We will use the phrase “X is *n times faster* Than Y”—or equivalently “X is *n times as fast as* Y”—*to mean*

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

If X is n times faster than Y, then the execution time on Y is n times longer than it is on X:

$$\frac{\text{Performance}_X}{\text{Performance}_Y} = \frac{\text{Execution time}_Y}{\text{Execution time}_X} = n$$

Example 1

If computer A runs a program in 10 seconds and computer B runs the same program in 15 seconds, how much faster is A than B?

Solution

We know that *A is n times faster than B if*

$$\frac{\text{Performance}_A}{\text{Performance}_B} = \frac{\text{Execution time}_B}{\text{Execution time}_A} = n$$

Solution-Cont.

Thus the performance ratio is

$$\frac{15}{10} = 1.5$$

In the above example, we could also say that computer B is 1.5 times *slower than* computer *A*, since

$$\frac{\text{Performance}_A}{\text{Performance}_B} = 1.5$$

means that

$$\frac{\text{Performance}_A}{1.5} = \text{Performance}_B$$

Measuring Performance

- Time is the measure of computer performance: the computer that performs the same amount of work in the least time is the fastest.

- Program *execution time is* measured in seconds per program.

- However, time can be defined in different ways, depending on what we count.

- The most straightforward definition of time is called *wall clock time, response time, or elapsed time.*

- *These terms mean the total time* to complete a task, including disk accesses, memory accesses, input/output (I/O) activities, operating system overhead—everything.

Measuring Performance-Cont.

- Computers are often shared, however, and a processor may work on several programs simultaneously.

- In such cases, the system may try to optimize throughput rather than attempt to minimize the elapsed time for one program.

- Hence, we often want to distinguish between the elapsed time and the time that the processor is working on our behalf.

- **CPU execution time or simply CPU time**, which recognizes this distinction, is the time the CPU spends computing for this task and does not include time spent waiting for I/O or running other programs.

- The response time experienced by the user will be the elapsed time of the program, not the CPU time.

Measuring Performance-Cont.

- CPU time- 1. User CPU time
  2. System CPU time

- CPU time spent in the program is called User CPU time and the CPU time spent in the operating system performing tasks requested by the program is called System CPU time.
- Suppose, after executing a program we get the following measurements:

  User CPU time – 90.7 seconds

  System CPU time – 12.9 seconds

  Elapsed time – 159 seconds

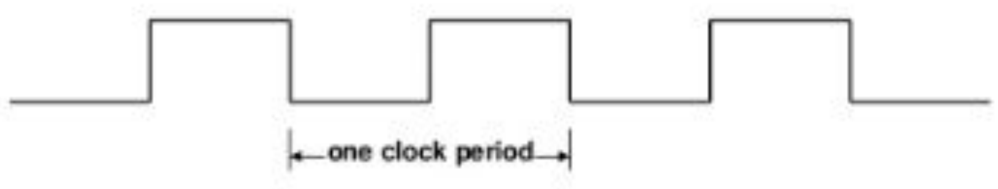
Measuring Performance-Cont.

- The term **system performance** is used to refer to elapsed time on an unloaded system while **CPU performance** refers to user CPU time on an unloaded system.

Clock Cycle

- Almost all computers are constructed using a clock that runs at a constant rate and determines when events take place in the hardware.

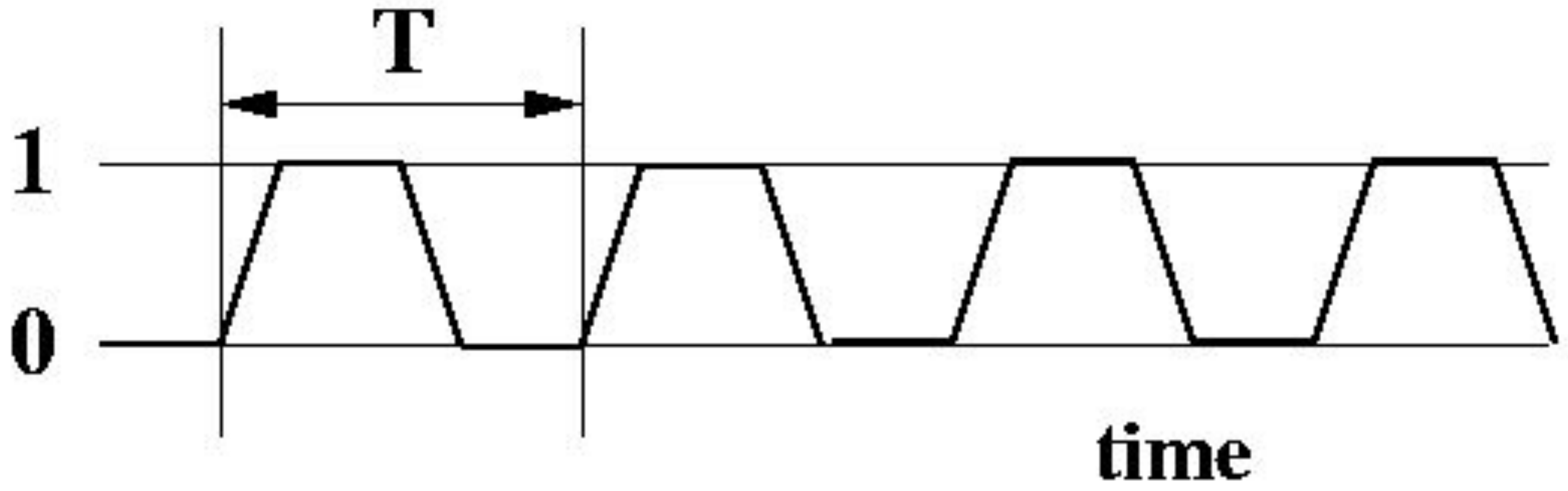
- Designers refer to the length of a **clock period** both as the time for a complete *clock cycle (e.g., 250 picoseconds, or 250 ps)*



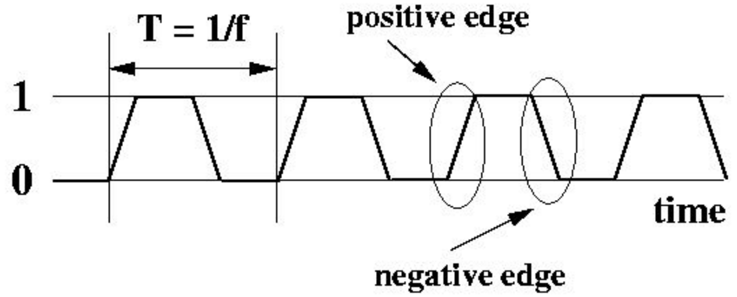
- The *clock rate (e.g., 4 gigahertz, or 4 GHz), is the inverse of the clock period.*

A more realistic Clock Cycle

A clock can't really instantaneously go from 0 to 1 and back to 0. A more realistic diagram looks like:



Features of a Clock



period This is the time it takes for a clock signal to repeat. Its unit of measurement is seconds. The symbol for the period is **T**.

frequency Frequency is **1/T**. The unit of measurement is **s**$^{-1}$, which is inverse seconds. This is also sometimes called **Hertz** (abbreviated **Hz**). Sometimes people say "cycles per second" instead of **Hz** or **s**$^{-1}$. They all mean the same. When people refer to the "speed" of a CPU, they are usually talking about the clock frequency.

positive edge When the signal transitions from 0 to 1. It's circled above.

negative edge When the signal transitions from 1 to 0. It's circled above. Each positive clock edge appears once per clock period. Each negative clock edge appears once per clock period.

Why clock used for?

- We use a clock to synchronize the events of a CPU. Devices in a CPU can be categorized as sequential circuits or combinational circuits. **Sequential logic** devices use a clock.

- Basically, sequential logic devices can only change outputs once a period, usually during a positive or a negative edge.

- To give you an analogy, imagine a music conductor tapping a beat with a baton (a stick) at regular intervals. Suppose you are playing a piano, and you're told to play a new note each time he taps his baton. Thus, how fast the conductor taps the baton controls how often you play notes.

- Similarly, the rate at which the positive clock edge appears controls how fast a sequential logic device can change outputs.

- It turns out that by using a clock, we can design a CPU more easily than if we don't use a clock. This is primary reason to use a clock.

Can't we make it faster?

If a CPU will run faster with a faster clock, then why not run it faster? The problem is that it takes time for signals to move around in a circuit. You can reduce this time in several ways. The main way is simply to make the circuit smaller. When the circuit is smaller, signals have less distance to travel, and therefore everything can happen more quickly.

CPU Performance and Its Factors

- CPU time for a program can be expressed in two ways:

$$\text{CPU execution time for a program} = \text{CPU clock cycles for a program} \times \text{Clock cycle time}$$

- Alternatively, because clock rate and clock cycle time are inverses,

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

This formula makes it clear that the hardware designer can improve performance by reducing the number of clock cycles required for a program or the length of the clock cycle.

Example 2

Our favourite program runs in 10 seconds on computer A, which has a 2 GHz clock. We are trying to help a computer designer build a computer, B, which will run this program in 6 seconds. The designer has determined that a substantial increase in the clock rate is possible, but this increase will affect the rest of the CPU design, causing computer B to require 1.2 times as many clock cycles as computer A for this program. What clock rate should we tell the designer to target?

Solution

Let's first find the number of clock cycles required for the program on A:

$$\text{CPU time}_A = \frac{\text{CPU clock cycles}_A}{\text{Clock rate}_A}$$

$$10 \text{ seconds} = \frac{\text{CPU clock cycles}_A}{2 \times 10^9 \dfrac{\text{cycles}}{\text{second}}}$$

$$\text{CPU clock cycles}_A = 10 \text{ seconds} \times 2 \times 10^9 \frac{\text{cycles}}{\text{second}} = 20 \times 10^9 \text{ cycles}$$

Solution-Cont.

CPU time for B can be found using this equation:

$$\text{CPU time}_B = \frac{1.2 \times \text{CPU clock cycles}_A}{\text{Clock rate}_B}$$

$$6 \text{ seconds} = \frac{1.2 \times 20 \times 10^9 \text{ cycles}}{\text{Clock rate}_B}$$

$$\text{Clock rate}_B = \frac{1.2 \times 20 \times 10^9 \text{ cycles}}{6 \text{ seconds}} = \frac{0.2 \times 20 \times 10^9 \text{ cycles}}{\text{second}} = \frac{4 \times 10^9 \text{ cycles}}{\text{second}} = 4 \text{ GHz}$$

To run the program in 6 seconds, B must have twice the clock rate of A.

Instruction Performance

- Since the compiler clearly generated instructions to execute, and the computer had to execute the instructions to run the program, the execution time must depend on the number of instructions in a program.

- One way to think about execution time is that it equals the number of instructions executed multiplied by the average time per instruction.

- Therefore, the number of clock cycles required for a program can be written as

$$\text{CPU clock cycles} = \text{Instructions for a program} \times \begin{array}{c}\text{Average clock cycles}\\ \text{per instruction}\end{array}$$

Instruction Performance –Cont.

- The term **clock cycles per instruction, which is the average number of clock** cycles each instruction takes to execute, is often abbreviated as **CPI**.

- Since different instructions may take different amounts of time depending on what they do, CPI is an average of all the instructions executed in the program.

- CPI provides one way of comparing two different implementations of the same instruction set architecture, since the number of instructions executed for a program will, of course, be the same.

Example 3

Suppose we have two implementations of the same instruction set architecture. Computer A has a clock cycle time of 250 ps and a CPI of 2.0 for some program, and computer B has a clock cycle time of 500 ps and a CPI of 1.2 for the same program.

Which computer is faster for this program and by how much?

Solution

We know that each computer executes the same number of instructions for the program; let's call this number I. First, find the number of processor clock cycles for each computer:

$$\text{CPU clock cycles}_A = I \times 2.0$$

$$\text{CPU clock cycles}_B = I \times 1.2$$

We know that each computer executes the same number of instructions for the program; let's call this number I. First, find the number of processor clock cycles for each computer:

Now we can compute the CPU time for each computer:

$$\begin{aligned}\text{CPU time}_A &= \text{CPU clock cycles}_A \times \text{Clock cycle time}\\ &= I \times 2.0 \times 250\ \text{ps} = 500 \times I\ \text{ps}\end{aligned}$$

Cont.

Likewise, for B: $\text{CPU time}_B = I \times 1.2 \times 500 \text{ ps} = 600 \times I \text{ ps}$

Clearly, computer A is faster. The amount faster is given by the ratio of the execution times:

$$\frac{\text{CPU performance}_A}{\text{CPU performance}_B} = \frac{\text{Execution time}_B}{\text{Execution time}_A} = \frac{600 \times I \text{ ps}}{500 \times I \text{ ps}} = 1.2$$

We can conclude that computer A is 1.2 times as fast as computer B for this program.

CPU Execution time

We have,

$$\frac{\text{CPU execution time}}{\text{for a program}} = \frac{\text{CPU clock cycles}}{\text{for a program}} \times \text{Clock cycle time}$$

Also,

$$\text{CPU clock cycles} = \text{Instructions for a program} \times \frac{\text{Average clock cycles}}{\text{per instruction}}$$

We can now write this basic performance equation in terms of **instruction count** (the number of instructions executed by the program), CPI, and clock cycle time:

$$\text{CPU time} = \text{Instruction count} \times \text{CPI} \times \text{Clock cycle time}$$

CPU Execution time

$$\text{CPU time} = \text{Instruction count} \times \text{CPI} \times \text{Clock cycle time}$$

or, since the clock rate is the inverse of clock cycle time:

$$\text{CPU time} = \frac{\text{Instruction count} \times \text{CPI}}{\text{Clock rate}}$$

Example 4

A compiler designer is trying to decide between two code sequences for a particular computer. The hardware designers have supplied the following facts:

	CPI for each instruction class		
	A	B	C
CPI	1	2	3

For a particular high-level language statement, the compiler writer is considering two code sequences that require the following instruction counts:

	Instruction counts for each instruction class		
Code sequence	A	B	C
1	2	1	2
2	4	1	1

Which code sequence executes the most instructions? Which will be faster? What is the CPI for each sequence?

Solution

- Sequence 1 executes 2 + 1 + 2 = 5 instructions.
- Sequence 2 executes 4 + 1 + 1 = 6 instructions.
- Therefore, sequence 1 executes fewer instructions.

$$\text{CPU clock cycles} = \sum_{i=1}^{n} (\text{CPI}_i \times \text{C}_i)$$

This yields

$$\text{CPU clock cycles}_1 = (2 \times 1) + (1 \times 2) + (2 \times 3) = 2 + 2 + 6 = 10 \text{ cycles}$$

$$\text{CPU clock cycles}_2 = (4 \times 1) + (1 \times 2) + (1 \times 3) = 4 + 2 + 3 = 9 \text{ cycles}$$

So code sequence 2 is faster, even though it executes one extra instruction.

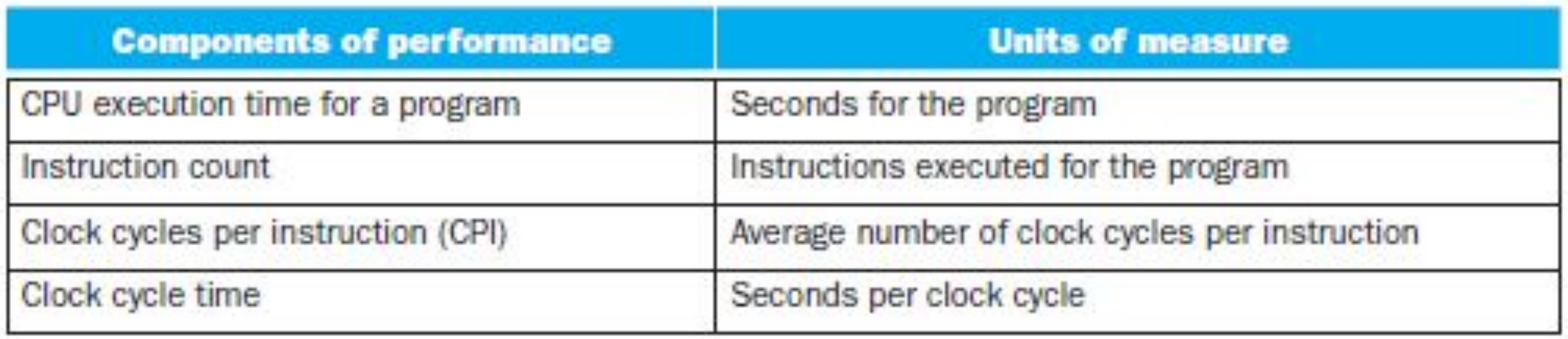
Cont.

$$\text{CPI} = \frac{\text{CPU clock cycles}}{\text{Instruction count}}$$

$$\text{CPI}_1 = \frac{\text{CPU clock cycles}_1}{\text{Instruction count}_1} = \frac{10}{5} = 2.0$$

$$\text{CPI}_2 = \frac{\text{CPU clock cycles}_2}{\text{Instruction count}_2} = \frac{9}{6} = 1.5$$

Summary



Components of performance	Units of measure
CPU execution time for a program	Seconds for the program
Instruction count	Instructions executed for the program
Clock cycles per instruction (CPI)	Average number of clock cycles per instruction
Clock cycle time	Seconds per clock cycle

FIGURE The basic components of performance and how each is measured.