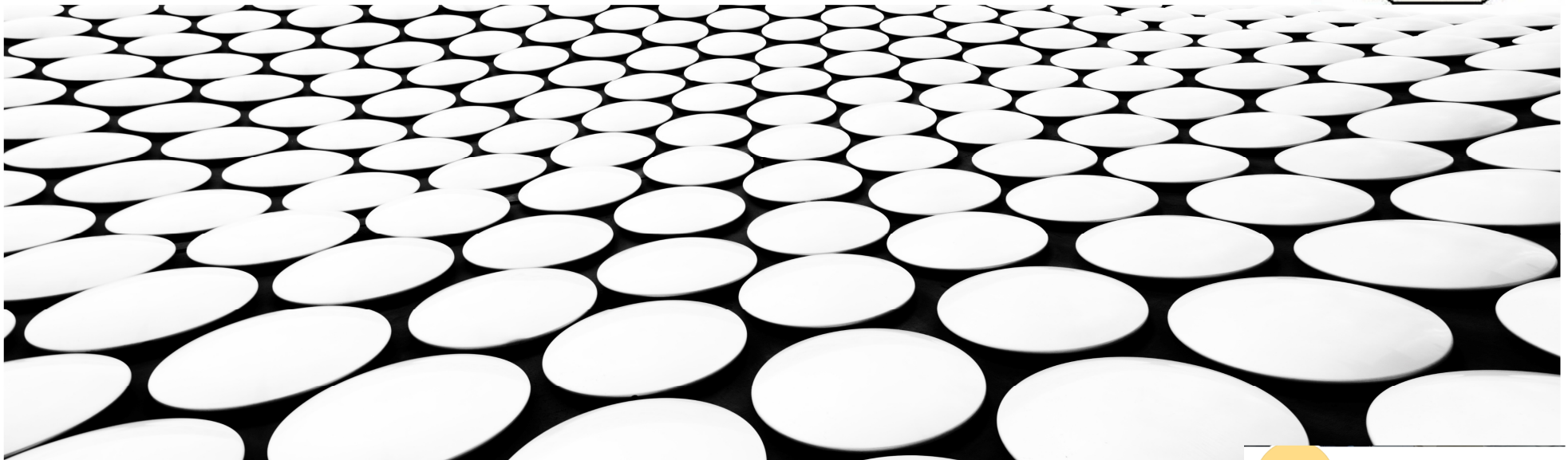


Fall 2022

CSE 2213 - COMPUTER ARCHITECTURE

LECTURE 2- PERFORMANCE MEASUREMENT

PROF. DR. MD. SHAMIM AKHTER



AISIP-Applied Intelligent
System and Information
Processing.

PERFORMANCE METRICS

- Purchasing perspective
 - given a collection of machines, which has the
 - best performance ?
 - least cost ?
 - best cost/performance?
- Design perspective
 - faced with design options, which has the
 - best performance improvement ?
 - least cost ?
 - best cost/performance?
- Both require
 - basis for comparison
 - metric for evaluation

-
- Our goal is to understand what factors in the architecture contribute
 - to overall system performance
 - relative importance (and cost) of these factors.

What ways can be used to determine performance on a desktop PC?

What ways can be used to determine performance on a server?

COMPUTER PERFORMANCE: TIME, TIME, TIME!!!

- *Response Time (elapsed time, latency):*

- how long does it take for *my* job to run?
- how long does it take to execute (start to finish) *my* job?
- how long must I wait for the database query?

Individual user
concerns...

- *Throughput:*

- how *many* jobs can the machine run at once?
- what is the *average* execution rate?
- how *much* work is getting done?

Systems manager
concerns...

WHAT IS THE EXECUTION TIME??

■ Elapsed Time

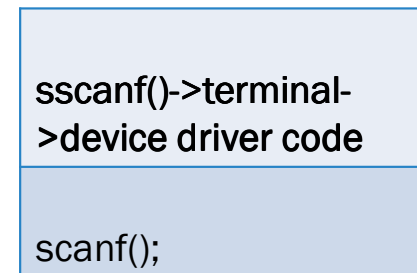
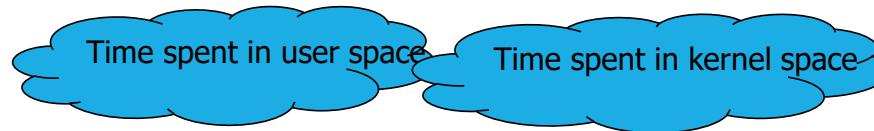
- counts everything (*disk and memory accesses, waiting for I/O, running other programs, etc.*) from start to finish
- a useful number, **but often not good for comparison purposes**
elapsed time = CPU time + wait time (I/O, other programs, etc.)

■ CPU time

- doesn't count waiting for I/O or time spent running other programs
- can be divided into **user CPU time** and **system CPU time** (OS calls)

CPU time = user CPU time + system CPU time

⇒ elapsed time = user CPU time + system CPU time + wait time



Kernel Space

User Space

- **Our focus: user CPU time** (CPU execution time or, simply, execution time)
 - time spent executing the lines of code that are *in our program*

CLOCK CYCLES

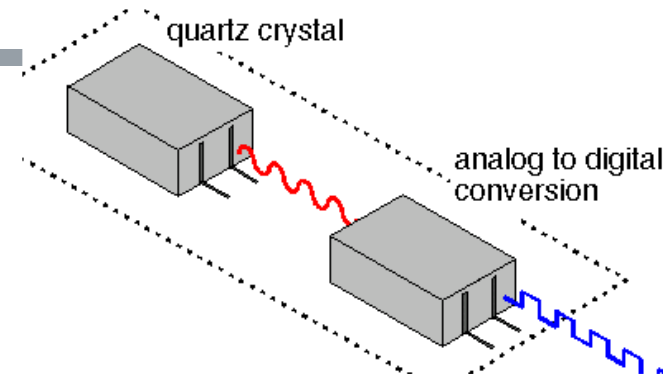
How do you measure execution time?

X seconds
↑ But cycles.

A clock is a circuit that produces a periodic signal, usually at constant frequency or rate.

A 1-GHz processor receives 1 billion pulses per second.

The rate of pulses is known as the clock rate, One increment, or a pulse is called a cycle or clock tick.

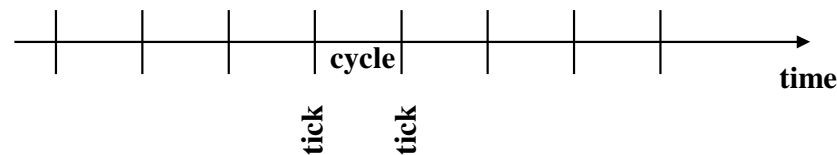


- Hardware events progress cycle by cycle:

- in other words, each event, e.g., multiplication, addition, etc., is a sequence of cycles

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

- Clock ticks indicate start and end of cycles:



Cycle time

- cycle time = time between ticks = seconds per cycle

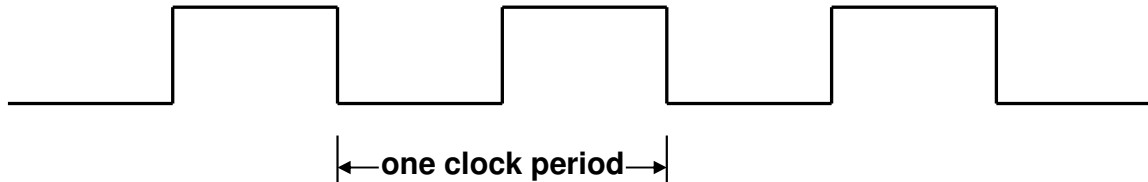
NB: Clock ticks = smallest unit of time recognized by a device.

Ex. 66MHz = 66 million clock ticks per second.

CLOCK RATE

- Clock rate (MHz, GHz) is inverse of clock cycle time (clock period)

$$CR = 1 / CT$$



clock rate (frequency) = cycles per second (1 Hz. = 1 cycle/sec,
1 MHz. = 10^6 cycles/sec)

Example: A 200 Mhz. clock has a cycle time

$$\frac{1}{200 \times 10^6} \times 10^9 = 5 \text{ nanosecor}$$

DEFINITION OF PERFORMANCE



**Improves
by reducing
execution time**

- For some program running on machine X:

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

X is n times faster than Y means:

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

PERFORMANCE EQUATION I

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

equivalently

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

- So, to improve performance one can either:
 - reduce execution time
 - reduce program cycles, or
 - reduce the clock cycle time, or, equivalently, increase the clock rate

EXAMPLE

- Our favorite program runs in 10 seconds on computer A, which has a 400Mhz. clock.
- We are trying to help a computer designer to build a new machine B, that will run this program in 6 seconds. The designer can use new (or perhaps more expensive) technology to substantially increase the clock rate, but has informed us that this increase will affect the rest of the CPU design, causing machine B to require 1.2 times as many clock cycles as machine A for the same program.
- *What clock rate should we tell the designer to target?*

SOLUTION:

- $\text{CPU Time}_A = \text{CPU Clock Cycles}_A / \text{Clock Rate}_A$
- $10 \text{ CR}_A = \text{CC}_A$
- $\text{CPU Time}_B = \text{CPU CC}_B / \text{CR}_B$
- $6 = 1.2 \text{CC}_A / \text{CR}_B$
- $\text{CR}_B = 1.2 \times 10 \times \text{CR}_A / 6 = 2 \times 400 \times 10^6$
- 800MHz

Our favorite program runs in 10 seconds on computer A, which has a 4 GHz clock. We are trying to help a computer designer build a computer, B, that 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?

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

$$\text{CPU time}_A = \frac{\text{CPU clock cycles}_A}{\text{Clock rate}_A}$$
$$10 \text{ seconds} = \frac{\text{CPU clock cycles}_A}{4 \times 10^9 \frac{\text{cycles}}{\text{second}}}$$

$$\text{CPU clock cycles}_A = 10 \text{ seconds} \times 4 \times 10^9 \frac{\text{cycles}}{\text{second}} = 40 \times 10^9 \text{ cycles}$$


CPU time for B can be found using this equation:

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

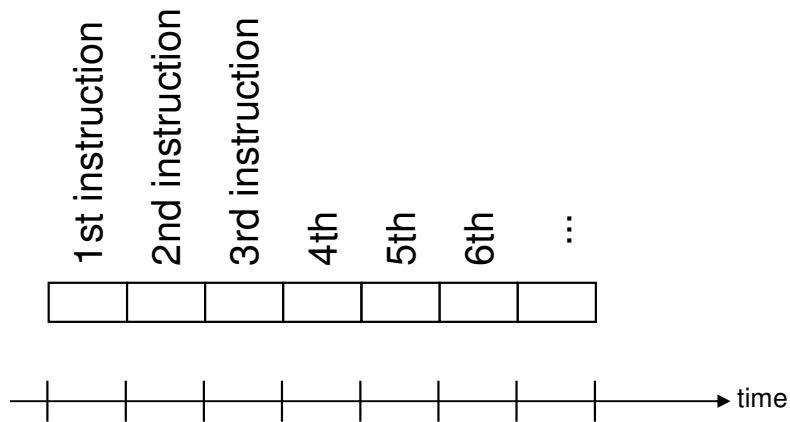
$$6 \text{ seconds} = \frac{1.2 \times 40 \times 10^9 \text{ cycles}}{\text{Clock rate}_B}$$

$$\text{Clock rate}_B = \frac{1.2 \times 40 \times 10^9 \text{ cycles}}{6 \text{ seconds}} = \frac{8 \times 10^9 \text{ cycles}}{\text{second}} = 8 \text{ GHz}$$

computer B must therefore have twice the clock rate of A to run the program in 6 seconds.

- 
- Can we predict the execution time, without running in CPU?
 - Assume: # of instruction = # of cycles

ASSUMPTION IS INCORRECT!



- Could assume that # of cycles = # of instructions

■ Because:

- Different instructions take different amounts of time (cycles)
- Multiplication takes more time than addition
- Floating point operations take longer than integer ones
- Accessing memory takes more time than accessing registers

They have different execution steps [Datapath]

SOLUTION:

- Execution time is equals to
- # of instructions executed \times the average time per instruction

PERFORMANCE EQUATION II

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

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

$$\text{CPU execution time for a program} = \text{Instruction count for a program} \times \text{average CPI} \times \text{Clock cycle time}$$

□ **Clock cycles per instruction (CPI)** – the average number of clock cycles each instruction takes to execute

- A way to compare two different implementations of the same ISA

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

CPI EXAMPLE I

- 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
- Which machine is faster for this program, and by how much?

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?

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:

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

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

Now we can compute the CPU time for each computer:

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

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.

1 milliseconds To Seconds

- millisecond $10^{-3} = 1/1,000$
- microsecond $10^{-6} = 1/1,000,000$
- nanosecond $10^{-9} = 1/1,000,000,000$
- picosecond $10^{-12} = 1/1,000,000,000,000$
- femtosecond $10^{-15} = 1/1,000,000,000,000,000$

EFFECTIVE CPI

- Computing the overall effective CPI is done by looking at the **different types of instructions and their individual cycle counts and averaging**

$$\text{Overall effective CPI} = \sum_{i=1}^n (\text{CPI}_i \times C_i)$$

Where C_i is the count **(percentage)** of the number of instructions of class i executed

C_i not in %, the above equation will be divided by total I_c

CPI_i is the (average) number of clock cycles per instruction for that instruction class

n is the **number of instruction classes**

$$\begin{aligned}\text{CPI}_{\text{effective}} &= (C_1 \times \text{CPI}_1 + C_2 \times \text{CPI}_2 + \dots + C_n \times \text{CPI}_n) / (c_1 + c_2 + \dots + c_n) \\ &= C_1 \times \text{CPI}_1 / T + C_2 \times \text{CPI}_2 / T + \dots + C_n \times \text{CPI}_n / T\end{aligned}$$

$$\text{Clock Cycle Time} = I \times \text{CPI}_{\text{effective}} \text{ from performance eqn II}$$

A SIMPLE EXAMPLE

Op	Freq	CPI _i	Freq x CPI _i	Q1	Q2	Q3
ALU	50%	1	.5	.5	.5	.25
Load	20%	5	1.0	.4	1.0	1.0
Store	10%	3	.3	.3	.3	.3
Branch	20%	2	.4	.4	.2	.4
Overall effective CPI			$\Sigma = 2.2$	1.6	2.0	1.95

- How much faster would the machine be if a better data cache reduced and replaced the average load time to 2 cycles?
 $\text{CPU time new} = 1.6 \times \text{IC} \times \text{CCT}$ so $2.2/1.6$ means 37.5% faster
- How does this compare with using branch prediction to save a cycle off the branch time?
 $\text{CPU time new} = 2.0 \times \text{IC} \times \text{CCT}$ so $2.2/2.0$ means 10% faster
- What if two ALU instructions could be executed at once?
 $\text{CPU time new} = 1.95 \times \text{IC} \times \text{CCT}$ so $2.2/1.95$ means 12.8% faster

CPI EXAMPLE II

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

	CPI for this 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 requires the following instruction counts:

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

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

Sequence 1 executes $2 + 1 + 2 = 5$ instructions. Sequence 2 executes $4 + 1 + 1 = 6$ instructions. So sequence 1 executes fewer instructions.

We can use the equation for CPU clock cycles based on instruction count and CPI to find the total number of clock cycles for each sequence:

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

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 actually executes one extra instruction. Since code sequence 2 takes fewer overall clock cycles but has more instructions, it must have a lower CPI. The CPI values can be computed by

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

$$\text{CPI}_1 = \frac{\text{CPU clock cycles}_1}{\text{Instruction count}_1} = \frac{10}{5} = 2$$

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

$$\text{ExeA} = 5 \times 2 = 10$$

$$\text{ExeB} = 6 \times 1.5 = 9$$

$$\text{PB/PA} = \text{ExeA/ExeB} \\ = 10/9$$

$$= 1.1 = 10\%$$

DEPENDENCY

Performance	Depends on the algorithm, programming language, Compiler, ISA
Execution Time	Depends on instruction counts, CPI, Clock Cycle time
Instruction Counts	Depends on ISA, Programmer effective coding, Compiler code optimization
CPI	ISA, H/W Organization
Clock Cycle Time	H/W Organization, Implementation strategies

Hardware or software component	Affects what?	How?
Algorithm	Instruction count, possibly CPI	The algorithm determines the number of source program instructions executed and hence the number of processor instructions executed. The algorithm may also affect the CPI, by favoring slower or faster instructions. For example, if the algorithm uses more floating-point operations, it will tend to have a higher CPI.
Programming language	Instruction count, CPI	The programming language certainly affects the instruction count, since statements in the language are translated to processor instructions, which determine instruction count. The language may also affect the CPI because of its features; for example, a language with heavy support for data abstraction (e.g., Java) will require indirect calls, which will use higher-CPI instructions.
Compiler	Instruction count, CPI	The efficiency of the compiler affects both the instruction count and average cycles per instruction, since the compiler determines the translation of the source language instructions into computer instructions. The compiler's role can be very complex and affect the CPI in complex ways.
Instruction set architecture	Instruction count, clock rate, CPI	The instruction set architecture affects all three aspects of CPU performance, since it affects the instructions needed for a function, the cost in cycles of each instruction, and the overall clock rate of the processor.