SECR2033

Computer Organization and Architecture

Lecture slides prepared by “Computer Organization and Architecture”, 9/e, by William Stallings, 2013.

Module 8

Performance Measurement and Analysis

Objectives:

- To understand the key performance issues that relate to computer design.
- Explain the reasons for the move to multicore organization, and understand the trade-off between cache and processor resources on a single chip.
- To summarize some of the issues in computer performance assessment.
- To discuss the benchmarks in general.

Module 8

Performance Measurement and Analysis

8.1 Introduction
8.2 Defining Performance
8.3 Basic Measures of Computer Performance
8.4 Comparing Performance
8.5 Benchmarking
8.6 Summary

8.1 Introduction

8

- Hardware performance is often key to the effectiveness of an entire system of hardware and software.
- For different types of applications, different performance metrics may by appropriate, and different aspects of a computer systems may be the most significant factor in determining overall performance.

- Understanding how best to measure performance and limitations of performance is important when selecting a computer system.

- Assessing the performance of computers can be quite challenging, together with the wide range of performance improvement techniques employed by hardware designers, have made performance assessment much more difficult.
- To understand the issues of assessing performance:

Why a piece of software performs as it does?

How some hardware feature affects performance?

Why one instruction set can be implemented to perform better than another?

Module 8
Performance Measurement and Analysis

- Overview
- Throughput and Response Time
- Relative Performance
- Performance Metrics

Overview

- When we say one computer has <u>better performance</u> than another, what do we mean?
- Although this question might seem simple, an analogy with passenger airplanes shows how subtle the question of performance can be (next slide).

*Which of these airplanes has the **best performance**?*

*The rate at which the airplane transports passengers, which is the **capacity times** the cruising speed.*

Airplane	Passenger capacity	Cruising range (miles)	Cruising speed (m.p.h.)	Passenger throughput (passengers × m.p.h.)
Boeing 777	375	4630	610	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

- Considering different measures of performance, we see that :
  - the plane with the highest cruising speed was the Concorde (retired from service in 2003),
  - the plane with the longest range is the Douglas DC-8-50,
  - the plane with the largest capacity is the Boeing 747.

Patterson, D.A. and Hennessy, J.L. (2014). *Computer Organization and Design: The Hardware/Software Interface* (5th Edition). United States: Elsevier, p.29.

- Let's suppose we define performance in terms of speed.
- Two possible definitions.

Performance based on *speed*:

You could define the fastest plane as the one with the highest cruising speed, taking a single passenger from one point to another in the least time.

Concord

Performance based on *throughput*:

If you were interested in transporting 450 passengers from one point to another, in the fastest time (as the last column of the figure shows).

Boeing 747

- Similarly, we can define computer performance in several different ways.

Patterson, D.A. and Hennessy, J.L. (2014). *Computer Organization and Design: The Hardware/Software Interface* (5th Edition). United States: Elsevier, p.29.

Throughput & Response Time

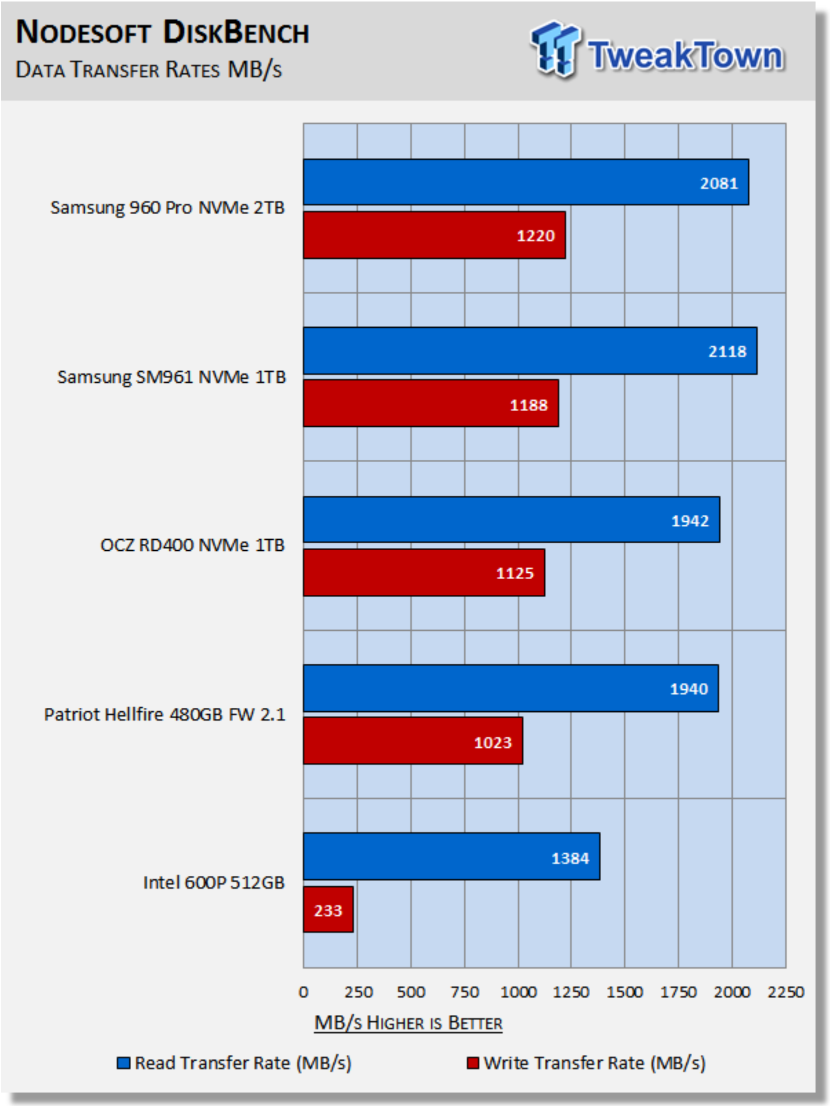
- We might interested in:

reducing *response time* (the time between the start and completion of a task) → also referred to as *execution time.*	increasing *throughput* or *bandwidth* → the total amount of work done in a given time.

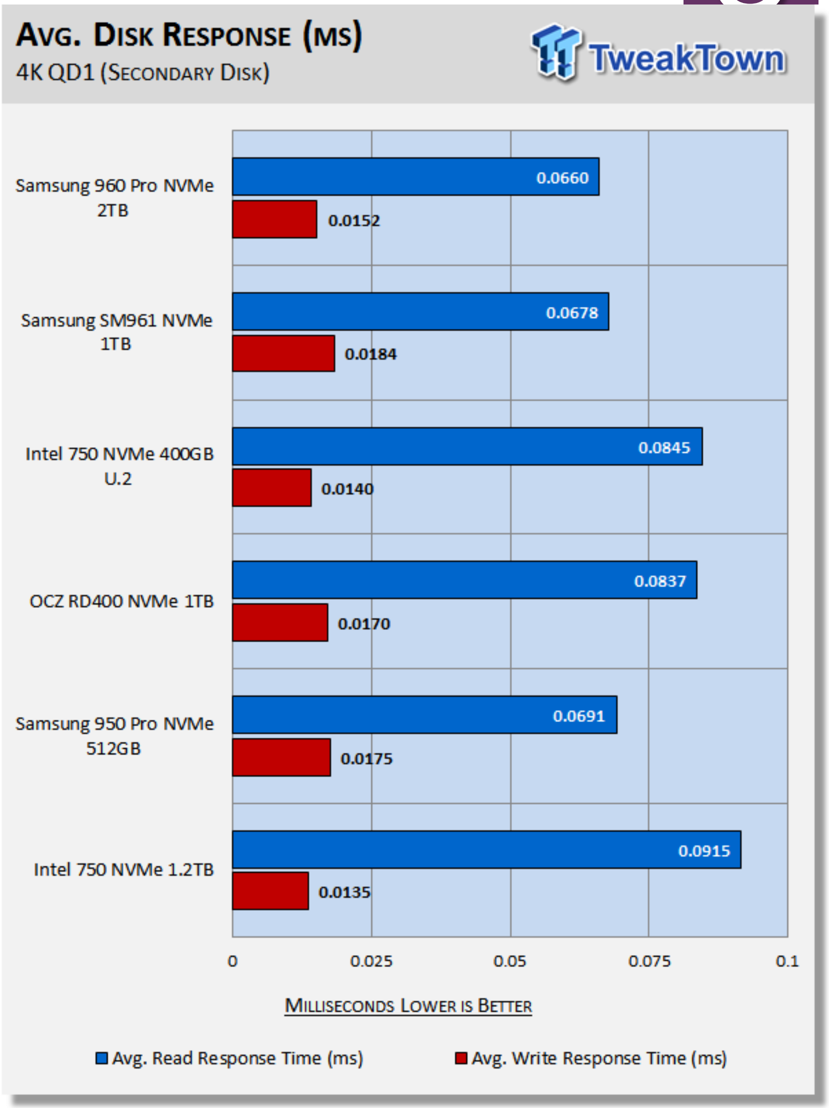
- Hence, in most cases, we need different performance metrics and sets of applications to benchmark personal mobile devices, which are more focused on *response time*, versus servers, which are more focused on *throughput*.

Patterson, D.A. and Hennessy, J.L. (2014). *Computer Organization and Design: The Hardware/Software Interface* (5th Edition). United States: Elsevier, p.30.

- Computer performance measures:



(a) Throughput



(b) Response Time

Example 1:

Do the following changes to a computer system increase throughput, decrease response time, or both?

(a) Replacing the processor in a computer with a faster version.

(b) Adding additional processors to a system that uses multiple processors for separate tasks, for example, searching the web

Solution :

(a) Decreasing response time almost always improves throughput. Hence, both response time and throughput are improved.

(b) No one task gets work done faster, so only throughput increases.

Patterson, D.A. and Hennessy, J.L. (2014). *Computer Organization and Design: The Hardware/Software Interface (5th* Edition). United States: Elsevier, p.30.

Relative Performance

- To maximize performance, the response time or execution time for some task should be minimized.
- Thus, the relationship between performance and execution time for a computer X:

$$Performance_X = \frac{1}{Execution\ time_X}$$

The lower the execution time, the higher the performance

Patterson, D.A. and Hennessy, J.L. (2014). *Computer Organization and Design: The Hardware/Software Interface* (5th Edition). United States: Elsevier, p.30.

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

$$Performance_X > Performance_Y$$

$$\frac{1}{Execution\ time_X} > \frac{1}{Execution\ time_Y}$$

$$Execution\ time_Y > Execution\ time_X$$

The execution time on Y is longer than X, if X is faster than Y.

- If X is n times as fast as Y, then the execution time on Y is n times as long as it is on X:

$$\frac{Performance_X}{Performance_Y} = \frac{Execution\ time_Y}{Execution\ time_X} = n$$

Example 2:

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 :

A is therefore 1.5 times as fast as B.

We know that A is n times as fast as B if:

$$\frac{Performance_A}{Performance_B} = \frac{Execution\ time_B}{Execution\ time_A} = n$$

Thus the performance ratio is $n = \frac{Execution\ time_B}{Execution\ time_A} = \frac{15}{10} = 1.5$

Patterson, D.A. and Hennessy, J.L. (2014). *Computer Organization and Design: The Hardware/Software Interface (5th* Edition). United States: Elsevier, p.31.

8 Performance Metrics

MB/s, Mb/s	: Megabytes, Megabits Per Second
MIPS	: Millions of Instructions Per Second
CPI	: Clock Cycles Per Instruction
IPC	: Instructions Per Clock cycle
Hz	: (processor clock frequency) cycles Per Second
LIPS	: Logical Interference Per Second
FLOPS	: Floating-Point arithmetic Operations Per Second

Module 8
Performance Measurement and Analysis

- Overview
- Clock Speed
- Instruction Execution Rate
- CPU Performance
- Classic CPU - Instruction Count

Overview

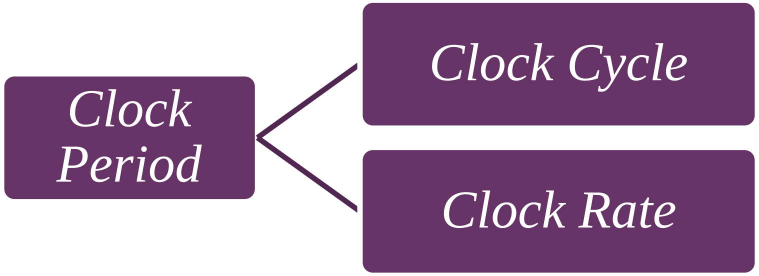
- In evaluating processor hardware and setting requirements for new systems, *performance* is one of the key parameters to consider, along with cost, size, security, reliability, and, in some cases, power consumption.
- It is difficult to make meaningful performance comparisons among different processors, even among processors in the same family.
- Next we look at some traditional measures of processor speed.

Clock Speed

- *Operations performed by a processor, such as fetching an instruction, decoding the instruction, performing an arithmetic operation, and so on, are governed by a system clock.

- These discrete time intervals are called *clock cycles* (or *ticks*, *clock ticks*, *clock periods*, *clocks, cycles*).

- *Clock period* is the length of each *clock cycle*, or *clock rate*, which inverse of the clock period.



(*e.g.*, 250 picoseconds, or 250 ps)

(*e.g.*, 4 Gigahertz, or 4 GHz),

* William Stallings (2016). *Computer Organization and Architecture: Designing for Performance* (10th Edition). United States: Pearson Education Limited, p.57.
Patterson, D.A. and Hennessy, J.L. (2014). *Computer Organization and Design: The Hardware/Software Interface* (5th Edition). United States: Elsevier, p.33.

Clock Speed

Clock cycle = the amount of time between two pulses of an oscillator. It is a single increment of the cpu clock during which the smallest unit of processor activity is carried out

Clock rate = clock frequency = clock speed = how many clock cycles a cpu can perform in 1 second
Measured in Hertz. 4 GHz= 4 billion hertz = a cpu with 4Ghz can perform 4 billion clock cycle in 1 second

Clock Period

- *Clock Cycle* (*e.g.*, 250 picoseconds, or 250 ps)
- *Clock Rate* (*e.g.*, 4 Gigahertz, or 4 GHz),

* William Stallings (2016). *Computer Organization and Architecture: Designing for Performance* (10th Edition). United States: Pearson Education Limited, p.57.
Patterson, D.A. and Hennessy, J.L. (2014). *Computer Organization and Design: The Hardware/Software Interface* (5th Edition). United States: Elsevier, p.33.

Instruction Execution Rate

- A processor is driven by a clock with a constant frequency, f.
- Define the instruction count, I_c, for a program as the number of machine instructions executed for that program.
- An important parameter is the average *Cycles Per Instruction* (*CPI*) for a program.

- If all instructions required the same number of clock cycles, then *CPI* would be a constant value for a processor.

William Stallings (2016). *Computer Organization and Architecture: Designing for Performance* (10th Edition). United States: Pearson Education Limited, p.58.

- However, on any given processor, the number of clock cycles required varies for different types of instructions, such as *load*, *store*, *branch*, and so on.
- Let CPI_i be the number of cycles required for instruction type i, and I_i be the number of executed instructions of type i for a given program. Then we can calculate an overall CPI as :

$$CPI = \frac{\sum_{i=1}^{n} (CPI_i \times I_i)}{I_c}$$

***Note**: I_c is number of instruction executions, not the number of instructions in the object code of the program.*

- A common measure of performance for a processor is the rate at which instructions are executed, expressed as *Millions of Instructions Per Second* (*MIPS*), referred to as the *MIPS rate*.

- We can express the MIPS rate in terms of the *clock rate* and *CPI* as follows:

No of executed instructions

$$MIPS\ rate = \frac{I_c}{T \times 10^6} = \frac{f}{CPI \times 10^6}$$

T → *Processor time*

Example 3:

Consider the execution of a program that results in the execution of 2 million instructions on a 400-MHz processor. The program consists of four major types of instructions. The instruction mix and the *CPI* for each instruction type are given below, based on the result of a program trace experiment:

Instruction Type	***CPI***	**Instruction Mix (%)**
Arithmetic and logic	1	60
Load/store with cache hit	2	18
Branch	4	12
Memory reference with cache miss	8	10

Calculate the corresponding *MIPS rate* for the processor.

William Stallings (2016). *Computer Organization and Architecture: Designing for Performance* (10th Edition). United States: Pearson Education Limited, p.59.

Instruction Type	*CPI*	Instruction Mix (%)
Arithmetic and logic	1	60
Load/store with cache hit	2	18
Branch	4	12
Memory reference with cache miss	8	10

CPI (Cycles Per Instruction)
MIPS (Millions of Instructions Per Second)

Solution :

The average *CPI* when the program is executed on a uniprocessor with the above trace results is:

$$
\begin{aligned}
CPI &= (1 \times 60\%) + (2 \times 18\%) + (4 \times 12\%) + (8 \times 10\%) \\
&= (1 \times 0.6) + (2 \times 0.18) + (4 \times 0.12) + (8 \times 0.1) \\
&= 0.6 + 0.36 + 0.48 + 0.8 \\
&= 2.24
\end{aligned}
$$

For this particular program, the cpu can execute 178 millions instructions per second

The corresponding *MIPS* rate is:

$$
MIPS\ rate = \frac{f}{CPI \times 10^6} = \frac{(400 \times 10^6)}{(2.24 \times 10^6)} \approx 178
$$

CPU Performance

- Users and designers often examine performance using different metrics.
- If we could relate these different metrics, we could determine the effect of a design change on the performance as experienced by the user.
- CPU performance as the bottom-line performance measure is *CPU execution time*.

$$\text{Clock cycle time} = \frac{1}{\text{Clock rate}}$$

- A simple formula relates the most basic metrics (clock cycles and clock cycle time) to CPU time for a program:

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

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

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

- This formula makes it clear that the performance can be improved by **reducing**:
  - the number of clock cycles required for a program, or
  - the length of the clock cycle.

Patterson, D.A. and Hennessy, J.L. (2014). *Computer Organization and Design: The Hardware/Software Interface* (5th Edition). United States: Elsevier, p.34.

Example 4: Improving performance

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?

8

Solution : Improving performance

(Method 1)

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

$$CPU\ time_A = \frac{CPU\ clock\ cycles_A}{Clock\ rate_A}$$

$$\begin{aligned} CPU\ clock\ cycles_A &= 10 \times (2 \times 10^9) \\ &= 20 \times 10^9\, cycles \end{aligned}$$

Patterson, D.A. and Hennessy, J.L. (2014). *Computer Organization and Design: The Hardware/Software Interface (5th* Edition). United States: Elsevier, p.34.

- CPU time for *B* can be found using this equation:

$$CPU\ time_B = \frac{1.2 \times CPU\ clock\ cycles_A}{Clock\ rate_B}$$

$$Clock\ rate_B = \frac{1.2 \times 20 \times 10^9 cycles}{6\ sec\,onds}$$

$$= 4 \times 10^9\ \frac{cycles}{sec\,onds}$$

$$= 4\,GHz$$

The clock rate we should tell the designer to target for computer B.

Patterson, D.A. and Hennessy, J.L. (2014). *Computer Organization and Design: The Hardware/Software Interface* (5th Edition). United States: Elsevier, p.34.

8

Solution : Improving performance

(Method 2)

Computer *A*:
Execution time$_A$ = 10 seconds
Clock rate$_A$ = 2 GHz = 2 × 10^9 Hz

Computer *B*:
Execution time$_B$ = 6 seconds

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

$$\begin{aligned} CPU\ clock\ cycle_A &= Execution\ time_A \times Clock\ rate_A \\ &= 10\ seconds \times (2 \times 10^9)\ Hz \\ &= 20 \times 10^9\ cycles \end{aligned}$$

- Clock rate for B can be found using this equation with 1.2 times more clock cycle than A:

$$1.2\,(CPU\ clock\ cycle_A) = Execution\ time_B \times Clock\ rate_B$$

$$Clock\ rate_B = \frac{1.2 \times CPU\ clock\ cycle_A}{Execution\ time_B}$$

$$= \frac{1.2 \times (2 \times 10^9)\ cycles}{6\ sec\,onds}$$

$$= 4 \times 10^9\ \frac{cycles}{sec\,onds}$$

$$= 4\ GHz$$

The clock rate we should tell the designer to target for computer B.

Activity 1

8

Exercise 8.1:

Our favourite program runs in 20 seconds on computer P, which has a 8 GHz clock.

We are trying to help a computer designer build a computer, Q, which will run this program in 5 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 Q to require 1.5 times as many clock cycles as computer P for this program.

What *clock rate* should we tell the designer to target?

8

Instruction Performance

- The performance equations previously did not include any reference to the number of instructions needed for the program.
- The execution time of a computer must depend on the number of instructions in a program.
- Therefore, the <u>number of clock cycles</u> required for a program can be written as:

$$\text{CPU clock cycles} = \begin{matrix}\text{Instruction executed} \\ \text{for a program}\end{matrix} \times \begin{matrix}\text{Average clock cycles} \\ \text{per instruction}\end{matrix}$$

$$\text{CPU clock cycles} = I \times CPI$$

Example 5: Using performance equation

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?

$$CPU\ execution\ time = CPU\ clock\ cycles \times Clock\ cycle\ time$$

$$CPU\ clock\ cycles = \underset{\text{a program.}}{Instruction\ for} \times \underset{\text{per instruction}}{Average\ clock\ cycles}$$

$$CPU\ clock\ cycles = I \times CPI$$

Patterson, D.A. and Hennessy, J.L. (2014). *Computer Organization and Design: The Hardware/Software Interface (5th* Edition). United States: Elsevier, p.35.

Solution : Using performance equation

- 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:

$$CPU\ clock\ cycle_A = I \times 2.0$$
$$CPU\ clock\ cycle_B = I \times 1.2$$

- Now we can compute the CPU time for each computer:

$$\begin{aligned} CPU\ time_A &= CPU\ clock\ cycle_A \times Clock\ cycle\ time_A \\ &= (I \times 2.0) \times 250ps \\ &= 500 \times I\ ps \end{aligned}$$

- The CPU time for computer B:

$$
\begin{aligned}
CPU\ time_B &= CPU\ clock\ cycle_B \times Clock\ cycle\ time_B \\
&= (I \times 1.2) \times 500ps \\
&= 600 \times I\ ps
\end{aligned}
$$

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

$$
\begin{aligned}
\frac{CPU\ performance_A}{CPU\ performance_B} &= \frac{Execution\ timeB}{Execution\ timeA} \\
&= \frac{600 \times I\ ps}{500 \times I\ ps} \\
&= 1.2
\end{aligned}
$$

Computer A is 1.2 times as fast as computer B for this program.

Patterson, D.A. and Hennessy, J.L. (2014). *Computer Organization and Design: The Hardware/Software Interface* (5th Edition). United States: Elsevier, p.36.

$$CPU\ execution\ time = CPU\ clock\ cycles \times Clock\ cycle\ time$$

$$CPU\ clock\ cycles = \underset{a\ program.}{Instruction\ for} \times \underset{per\ instruction}{Average\ clock\ cycles}$$

$$CPU\ clock\ cycles = I \times CPI$$

Classic CPU - Instruction Count

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

$$CPU\ time = Instruction\ count \times CPI \times Clock\ cycle\ time$$

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

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

The three key factors that affect performance.

Patterson, D.A. and Hennessy, J.L. (2014). *Computer Organization and Design: The Hardware/Software Interface* (5th Edition). United States: Elsevier, p.36.

Example 6: Comparing code segments

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

	CPI for each instruction class		
CPI	1	2	3

The compiler writer is considering two code sequences that require the following instruction counts:

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

(a) Which code sequence execute the most instructions?
(b) Which will be faster?
(c) What is the CPI for each sequence?

Patterson, D.A. and Hennessy, J.L. (2014). *Computer Organization and Design: The Hardware/Software Interface* (5th Edition). United States: Elsevier, p.37.

Solution : Comparing code segments

Code sequence 2 executes the most instruction count.

(a) $$Code\ sequence_1 = 2 + 1 + 2 = 5\ instructions\ count_1$$

$$Code\ sequence_2 = 4 + 1 + 1 = 6\ instructions\ count_2$$

(b) Get the CPU clock cycle for each code sequence using the equation:

$$CPU\ clock\ cycle = \sum_{i=1}^{n} (CPI_i \times I_i)$$

Code sequence 2 executes the instructions faster.

$$CPU\ clock\ cycle_1 = (1 \times 2) + (2 \times 1) + (3 \times 2)$$
$$= 2 + 2 + 6 = 10\ cycles$$

$$CPU\ clock\ cycle_2 = (1 \times 4) + (2 \times 1) + (3 \times 1)$$
$$= 4 + 2 + 3 = 9\ cycles$$

Patterson, D.A. and Hennessy, J.L. (2014). *Computer Organization and Design: The Hardware/Software Interface* (5th Edition). United States: Elsevier, p.37.

(c) The *CPI* values for each code sequence can be computed by:

$$CPI = \frac{CPU\ clock\ cycle}{Instruction\ count}$$

$$CPI_1 = \frac{CPU\ clock\ cycle_1}{Instruction\ count_1} = \frac{10}{5} = 2.0$$

$$CPI_2 = \frac{CPU\ clock\ cycle_2}{Instruction\ count_2} = \frac{9}{6} = 1.5$$

Since code sequence 2 takes fewer overall clock cycle but more instructions, it must has a lower CPI .

The BIG Picture

- The table shows the basic measurement at different levels in the computer and what is being measured in each case.

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

Patterson, D.A. and Hennessy, J.L. (2014). *Computer Organization and Design: The Hardware/Software Interface* (5th Edition). United States: Elsevier, p.38.

Factors in CPU performance

The BIG Picture

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

$$\text{Clock cycle time} = \frac{1}{\text{Clock rate}}$$

- We can see how these factors are combined to yield execution time in seconds per program:

$$\text{Time} = \text{Instruction Count} \times \text{CPI} \times \text{Cycle Time}$$

OR

$$\text{Time} = \frac{\text{Seconds}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock cycle}}$$

- Higher instruction count → faster clock cycle time or lower CPI or higher CPU frequency
- Lower instruction count → slower clock cycle time or higher CPI or higher CPU frequency

CPI depends on type of instructions executed, less instructions may not be the fastest.

Understanding Program Performance

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 divides, 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.

Aspects of CPU Performance

$$\text{cpu time} = \frac{\text{seconds}}{\text{program}} = \frac{\text{instructions}}{\text{program}} \times \frac{\text{cycles}}{\text{instruction}} \times \frac{\text{seconds}}{\text{cycle}}$$

	Instruction Count	CPI	Clock cycle time
Program	X	X	
Compiler	X	X	
Instruction Set	X	X	
Organization		X	X
Technology			X

Activity 2

Exercise 8.2:

A compiler designer is trying to decide between two code sequences for a particular computer. Two code sequences need to be considered with the following facts :

Instruction	CPI	Code seq 1	Code seq 2
A	1	5	3
B	2	3	2
C	5	1	2

(a) Which code sequence execute the most instructions?
(b) Which will be faster?
(c) What is the CPI for each sequence?

Activity 3

8

Exercise 8.3:

A given application written in Java runs 15 seconds on a desktop processor. A new java compiler is released the requires only 0.6 as many instructions as the old compiler. Unfortunately, it increased the CPI by 1.1.

How fast can we expect the application to run using this new compiler?

Exercise 8.4:

Activity 4

CPU X runs a program/code sequence which consists of 100 instructions. Calculate and fill in the table:

(a) The CPI for each instruction class given below.

(b) The execution time for each instruction class, given a clock cycle time is 0.25 miliseconds.

(c) The CPU X's execution time.

(d) The CPU X's clock rate.

Instruction	Instructions count	Clock Cycles	(a) CPI	(b) Execution time
A	20	3		
B	25	1		
C	10	2		
D	30	2		
E	10	3		
F	5	4		