Chapter 1
Computer Abstraction and Technology

Dr. Md Abu Sayeed
EET 340

Classes of Computers

Personal Computer: A **personal computer** is a general-purpose, cost-effective computer that is designed to be used by a single end-user. Example: Desktop computer, laptop computer

Server : A computer used for running larger programs for multiple users, often simultaneously, and typically accessed only via a network.

Supercomputer: A supercomputer is a large array of small computers. A class of computers with the highest performance. It is built to solve complex problem. Supercomputers are usually used for high-end scientific and engineering calculations, such as weather forecasting, oil exploration, protein structure, and other large-scale problems. Supercomputers are expensive and typically cost tens to hundreds of millions of dollars.

Classes of Computers

Embedded Computer: An **embedded computer** is a combination of hardware and software that is designated to perform a specific task. Example: Television, printer, projector.

Personal Mobile Devices (PMDs): PMDs are small wireless device to connect to the internet. It is both portable and capable of storing, transmitting or processing electronic data or images. They rely on batteries for power, and software is installed by downloading app. Example: smartphone.

Cloud Computing: It refers to large collections of servers that provide service over internet. Instead of buying, owning, and maintaining physical data centers and servers, you can access technology services, such as computing power, storage, and databases, on an as-needed basis from a cloud provider. Example: Amazon AWS, Epsagon

High Level Languages

•There are many languages for writing programs such as : C, C++, Java, Pascal, FORTRAN etc.

•High level languages resemble human languages in many ways. They are designed to be easy for human beings to write programs in and to be easy for human beings to read.

Example of a high-level language:

```
X = Y + Z ;
```

This line of instruction clearly indicates that X is equal to the addition of Y and Z

Low Level Languages

- The kind of language a computer can understand is called a low-level Language.

  Example of a Low-level language:
  ADD X9, X20, X21;

- This instruction means adding the value of X20 and X21, and storing the resulting addition on X9, in other words, it is equivalent to high level instruction X= Y+Z .
- This line of instruction is written in assembly language which is a low-level language. Assembly language is a symbolic representation of machine instructions. This instruction may be unclear to human user. Low level languages do not resemble human languages. However, it is understood by the computer

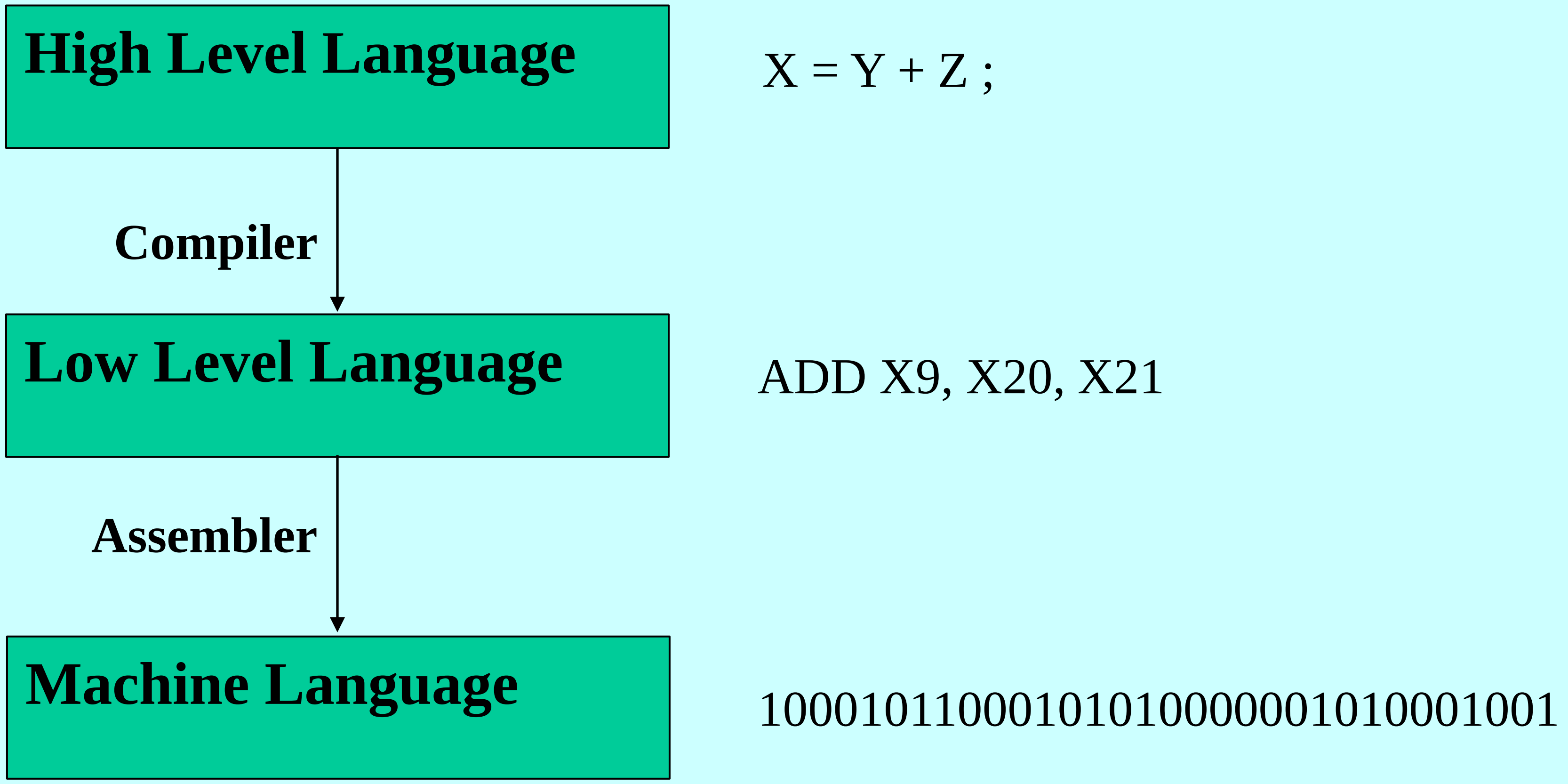
Machine Languages

- Programs written in the form of zeros and ones are called machine languages, because that is the version of the program that computer reads and follows.
- Assembly language needs to be translated into string of zeros and ones (machine languages), so that computer can read.
- Any high-level language must be translated into machine language before computer can read and understand the program.

```
ADD X9, X20, X21        ← Assembly Language

10001011000101010000001010001001← Machine Language
```

High Level Language to Machine Language



Compiler : A program that translate high-level language statements into assembly language.

Assembler : A program that translate a symbolic version of instructions into the binary version.

Components of a Computer

Five components: Input, Output, Memory, Control, Datapath

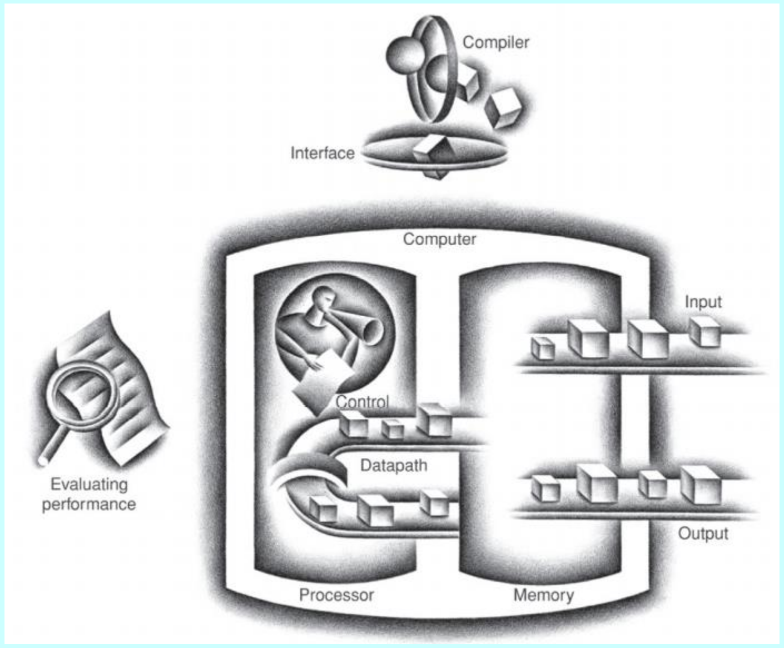
Input: Input feeds computer with data. Example: Keyboard, touchscreen.

Output: Output is a result of computation sent to the user. Example: Display, speaker

Memory: Store data and instructions. Example: Hard disk, CD/DVD, flash

Control: Commands the datapath, memory, and I/O devices according to instructions of the program.

Datapath: Moves data and perform arithmetic operations.



Response Time and Throughput

What is volatile and non-volatile memory?

Volatile Memory: Loses instruction and data when power is off. Example: DRAM.

Non-volatile Memory: It does not lose instruction and data when power is off. Example: DVD, Magnetic disk, Flash Memory.

Response Time : The time required for CPU to complete a task is called response Time. This is also called execution time.

Throughput: The total number of work done in a given time.

How to improve response time and throughput?

– Replacing the processor with a faster version
– Adding more processors

CPU Clocking

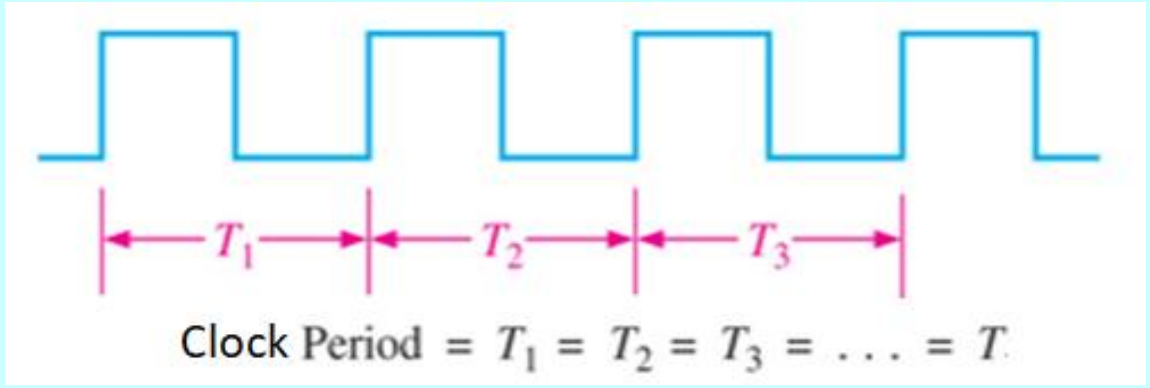


FIGURE Examples of periodic digital waveforms (square wave).

Clock Period or Clock cycle time (T) : Duration of a clock cycle. The clock period or cycle time, T, is the time between rising edges of a repetitive clock signal.

Clock frequency or Clock rate (f): Clock frequency is the reciprocal of the period. It refers to the number of clock cycles CPU executes per second.

$$f = \frac{1}{T}$$

CPU Time

The actual time the CPU spends computing for a specific task is known as CPU time. It is also called CPU execution time.

$$\text{CPU Time} = \text{CPU Clock Cycles X Clock Cycle Time}$$

$$= \frac{\text{CPU Clock Cycles}}{\text{Clock Rate}}$$

• Performance improved by

– Reducing number of clock cycles

– Increasing clock rate

– Hardware designer must often trade off clock rate against cycle count

Problem: A program takes 1000 clock-cycles to run a processor running at 2 GHZ. What is the time spent on the CPU by the program.

Solution :

Clock cycles = 1000
Clock rate = 2 GHz
CPU Time = 1000 / 2 x 10^{9} = 0.5 x 10^{-6} = 0.5µs

Performance

We generally evaluate performance in terms of response/execution time. 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;

$$\text{Performance}_x = \frac{1}{\text{Execution Time}_x}$$

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

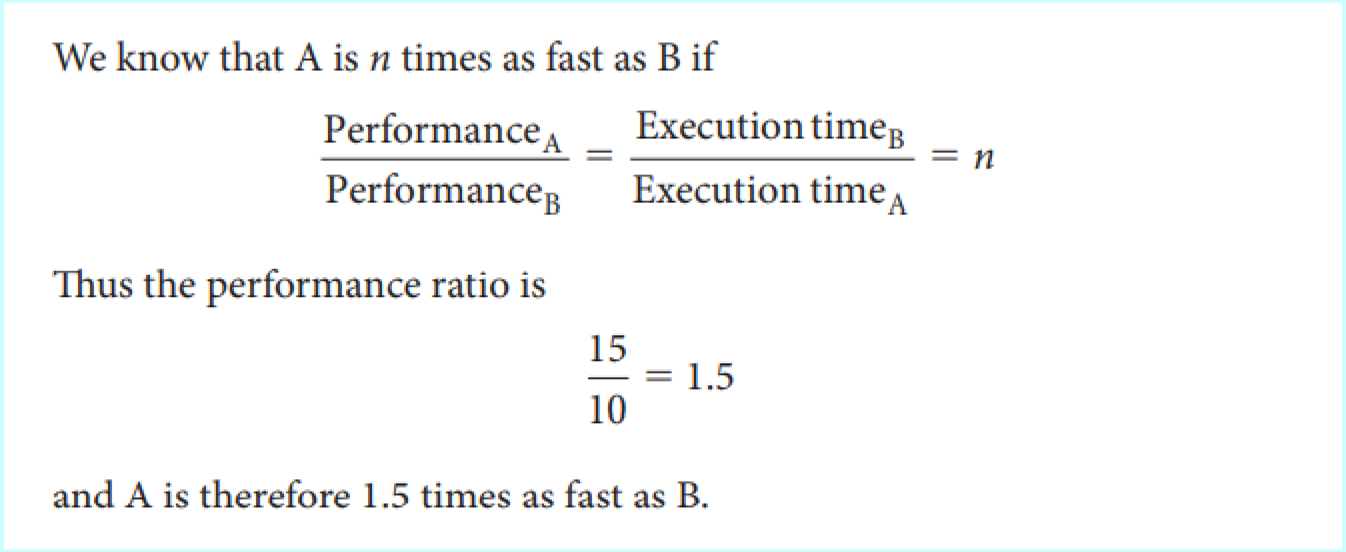
$$\text{Performance}_X > \text{Performance}_Y$$

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

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

Problem: 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 as fast as B if

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

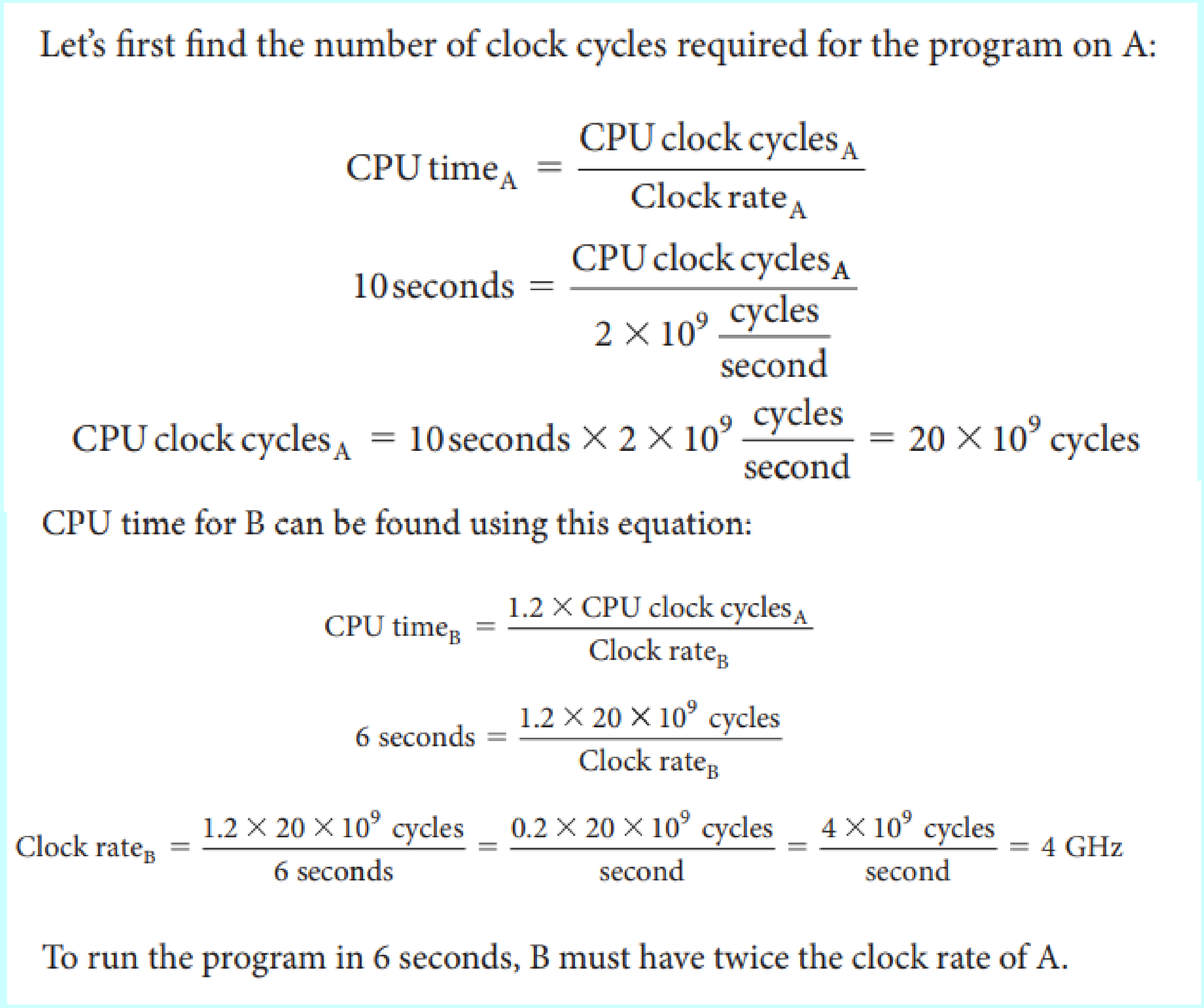
Thus the performance ratio is

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

and A is therefore 1.5 times as fast as B.

Problem: Our favorite program runs in 10 seconds on computer A, which has a 2GHz 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:

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

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

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

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 Count and CPI

Instruction Count: The number of instructions executed by the program.
CPI (Clock cycles per instruction): Average number of clock cycles per instruction for a program.

The number of clock cycles required for a program can be written as:

$$\textbf{CPU Clock Cycles} = \textbf{Instruction Count X Clock Cycles per instruction}$$

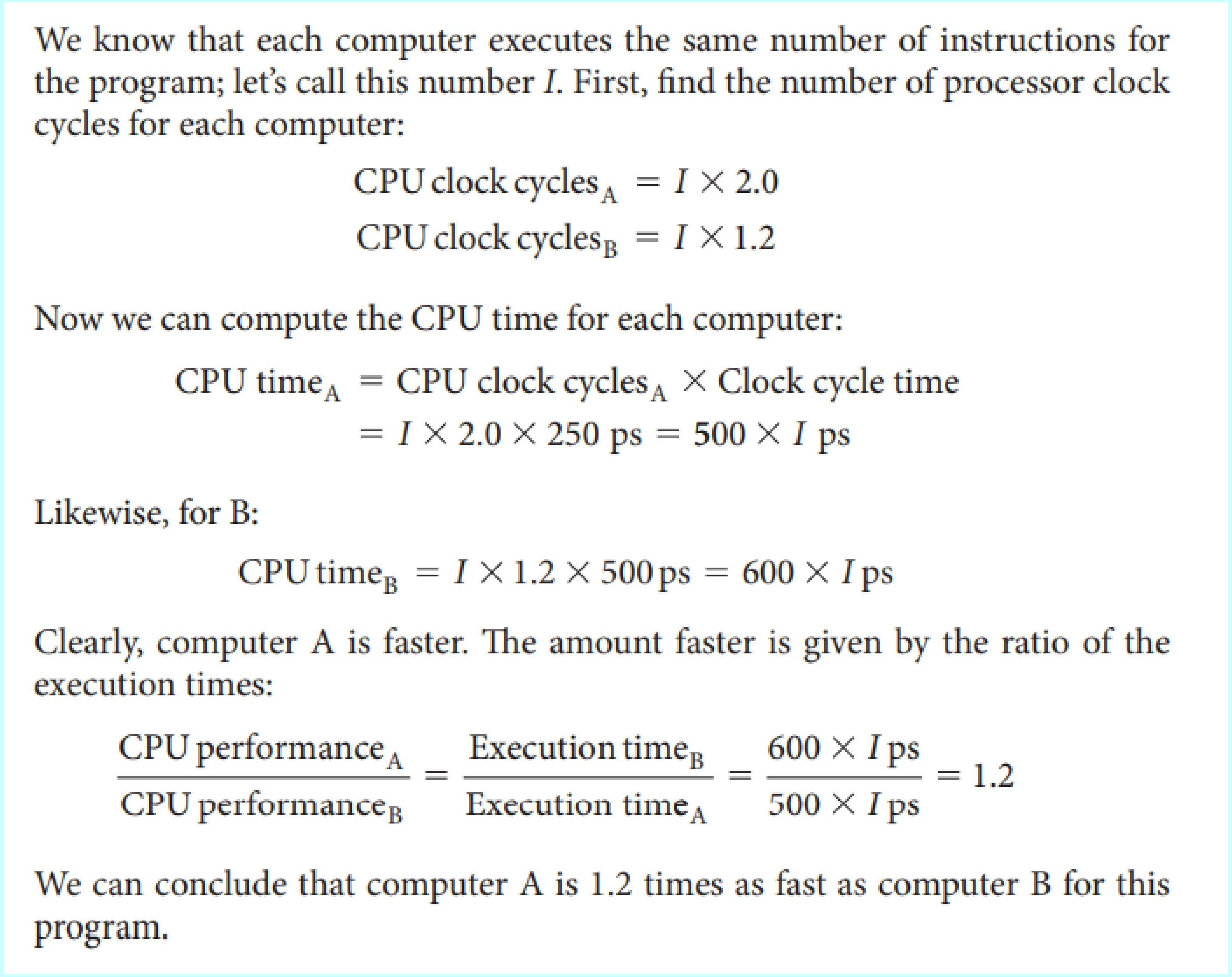
The performance equation can be written in terms of instruction count, CPI, and clock cycles.

$$\textbf{CPU Time} = \textbf{Instruction Count X CPI X Clock Cycle Time}$$

$$= \frac{\textbf{Instruction Count X CPI}}{\textbf{Clock Rate}}$$

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

$$\text{CPU clock cycles}_A = I \times 2.0$$
$$\text{CPU clock cycles}_B = I \times 1.2$$

Now we can compute the CPU time for each computer:

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

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.

CPI in More Detail

- **If different instruction classes take different numbers of cycles**

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

CPI is the clock cycle per instruction and C is the instruction count.

- **Weighted average CPI**

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

Problem: A compiler designer is trying to decide between two code sequences for a computer. For a particular high-level language, the compiler writer is considering two code sequences that requires the following instruction counts. Which code sequence executes the most instructions? Which will be faster? What is the CPI for each sequence?

Class	**A**	**B**	**C**
CPI	1	2	3
IC in code sequence 1	2	1	2
IC in code sequence 2	4	1	1

Solution : on the next page

Sequence 1 executes 2 + 1 + 2 = 5 instructions. Sequence 2 executes 4 + 1 + 1 = 6 instructions. Therefore, 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:

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

This yields

$$\text{CPU clock cycles}_1 = (2 \times 1) + (1 \times 2) + (2 \times 3) = 2 + 2 + 6 = 10 \text{ cycles}$$
$$\text{CPU clock cycles}_2 = (4 \times 1) + (1 \times 2) + (1 \times 3) = 4 + 2 + 3 = 9 \text{ cycles}$$

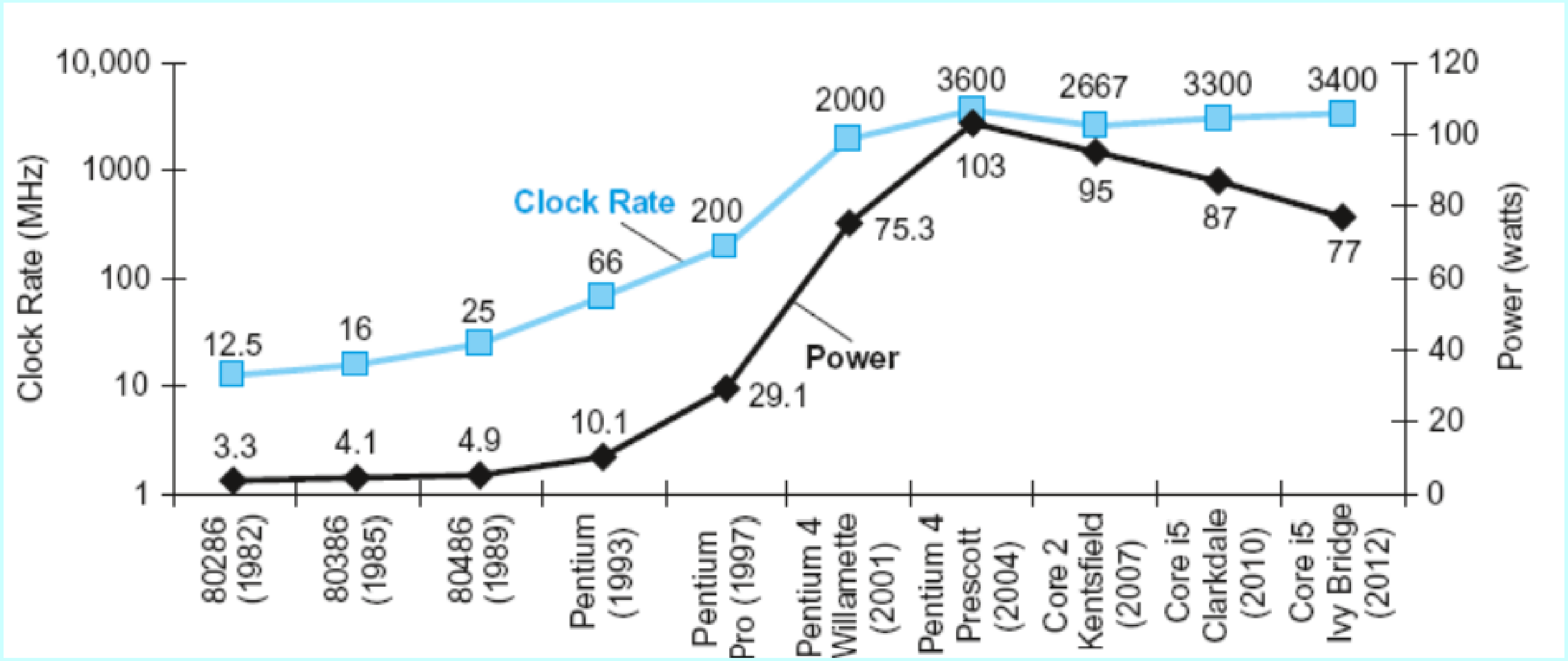
So code sequence 2 is faster, even though it executes one extra instruction. Since code sequence 2 takes fewer overall clock cycles but has more instructions, it must have a lower CPI. The CPI values can be computed by

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

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

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

Power Trends

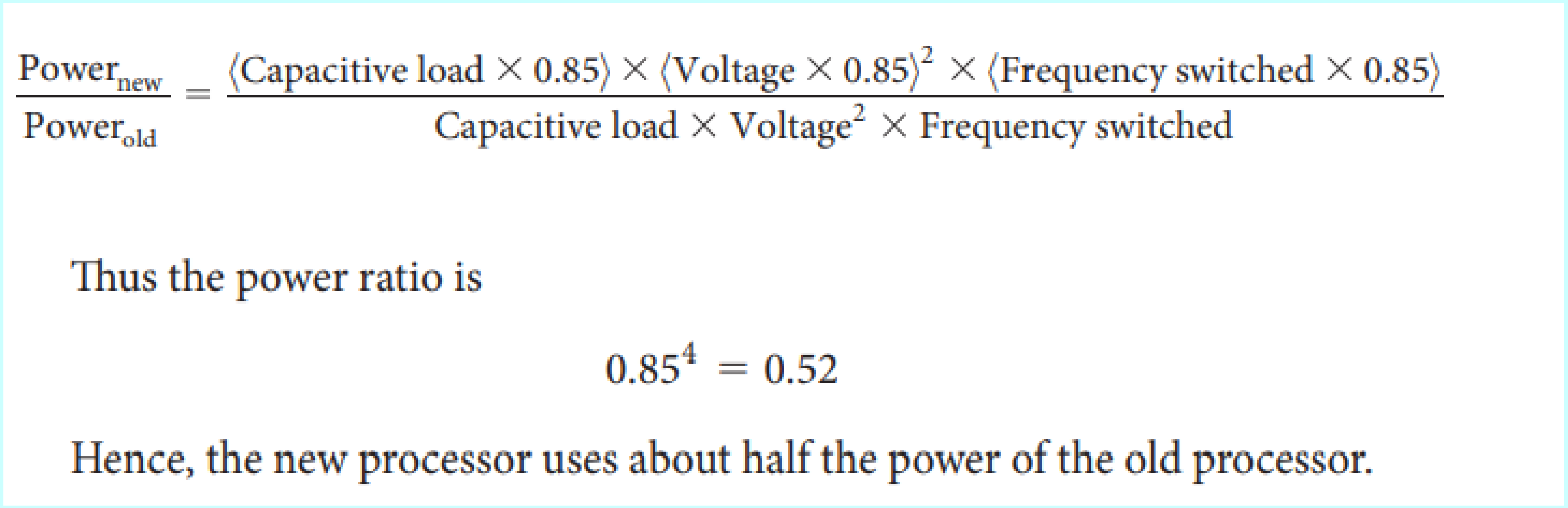


Power = Capacitive load X Voltage2 X Frequency

Reducing Power

Problem: Suppose we developed a new, simpler processor that has 85% of the capacitive load of the more complex older processor. Further, assume that it can adjust voltage so that it can reduce voltage 15% compared to processor B, which results in a 15% shrink in frequency. What is the impact on dynamic power?

Solution :



$$\frac{\text{Power}_{\text{new}}}{\text{Power}_{\text{old}}} = \frac{\langle \text{Capacitive load} \times 0.85 \rangle \times \langle \text{Voltage} \times 0.85 \rangle^2 \times \langle \text{Frequency switched} \times 0.85 \rangle}{\text{Capacitive load} \times \text{Voltage}^2 \times \text{Frequency switched}}$$

Thus the power ratio is

$$0.85^4 = 0.52$$

Hence, the new processor uses about half the power of the old processor.

Amdahl’s Law

This states that the overall performance improvement gained by optimizing a single part of a system is limited by the fraction of time that the improved part is actually used.

The execution time of a program after making the improvement is given by the following simple equation known as Amdahl’s law.

$$\text{Execution time after improvement} = \frac{\text{Execution time affected by improvement}}{\text{Amount of Improvement}} + \mathrm{T}_{\text{unaffected}}$$

$$T_{\text{improved}} = \frac{\mathrm{T}_{affected}}{\text{Improvement } Factor} + \mathrm{T}_{\text{unaffected}}$$

Problem: Suppose a program runs in 100 seconds in a computer, with multiply operations responsible for 80 seconds ? How much do I need to improve the speed of the multiplication if I want my program run 2 times faster.

Solution :

$$\mathrm{T}_{\mathbf{improved}} = \frac{\mathbf{T_{affected}}}{\mathbf{Improvement\ Factor}} + \mathbf{T_{unaffected}}$$

$$\mathbf{50 = \frac{80}{n} + 20}$$

$$\mathbf{\frac{80}{n} = 50 - 20}$$

$$\mathbf{n = \frac{80}{30} = 2.66}$$

Multiply operations should be 2.66 times faster

Problem: Suppose a program runs in 100 seconds in a computer, with multiply operations responsible for 80 seconds ? How much do I need to improve the speed of the multiplication if I want my program run 5 times faster.

Solution :

$$\mathrm{T}_{\mathbf{improved}} = \frac{\mathbf{T}_{\mathbf{affected}}}{\mathbf{Improvement\ Factor}} + \mathbf{T}_{\mathbf{unaffected}}$$

$$\mathbf{20 = \frac{80}{n} + 20}$$

$$\mathbf{\frac{80}{n} = 20 - 20}$$

$$\mathbf{n = \frac{80}{0}}$$

Speed up is not possible

Note: The figures, text etc included in slides are borrowed from books, other sources for academic purpose only. The author does not claim any originality.

Source:

1.Computer Organization and Design (ARM Edition) by David A. Patterson