+

William Stallings Computer Organization and Architecture 10th Edition

Chapter 2

Performance Issues

+

Designing for Performance

- The **cost** of computer systems continues **to drop** dramatically, while the **performance and capacity** of those systems continue to **rise** equally dramatically
- Today's laptops have the **computing power** of an IBM mainframe from 10 or 15 years ago
- Processors are so **inexpensive** that we now have microprocessors we throw away

+
Designing for Performance

- Desktop **applications** that require the great power of today's microprocessor-based systems include:
  - Image processing
  - Three-dimensional rendering
  - Speech recognition
  - Videoconferencing
  - Multimedia authoring
  - Voice and video annotation of files
  - Simulation modeling
- Businesses are depend on **increasingly powerful servers** to handle transaction and database processing and to support massive client/server networks
- Cloud service providers use massive high-performance banks of servers to satisfy high-volume, high-transaction-rate applications.

+
Microprocessor Speed

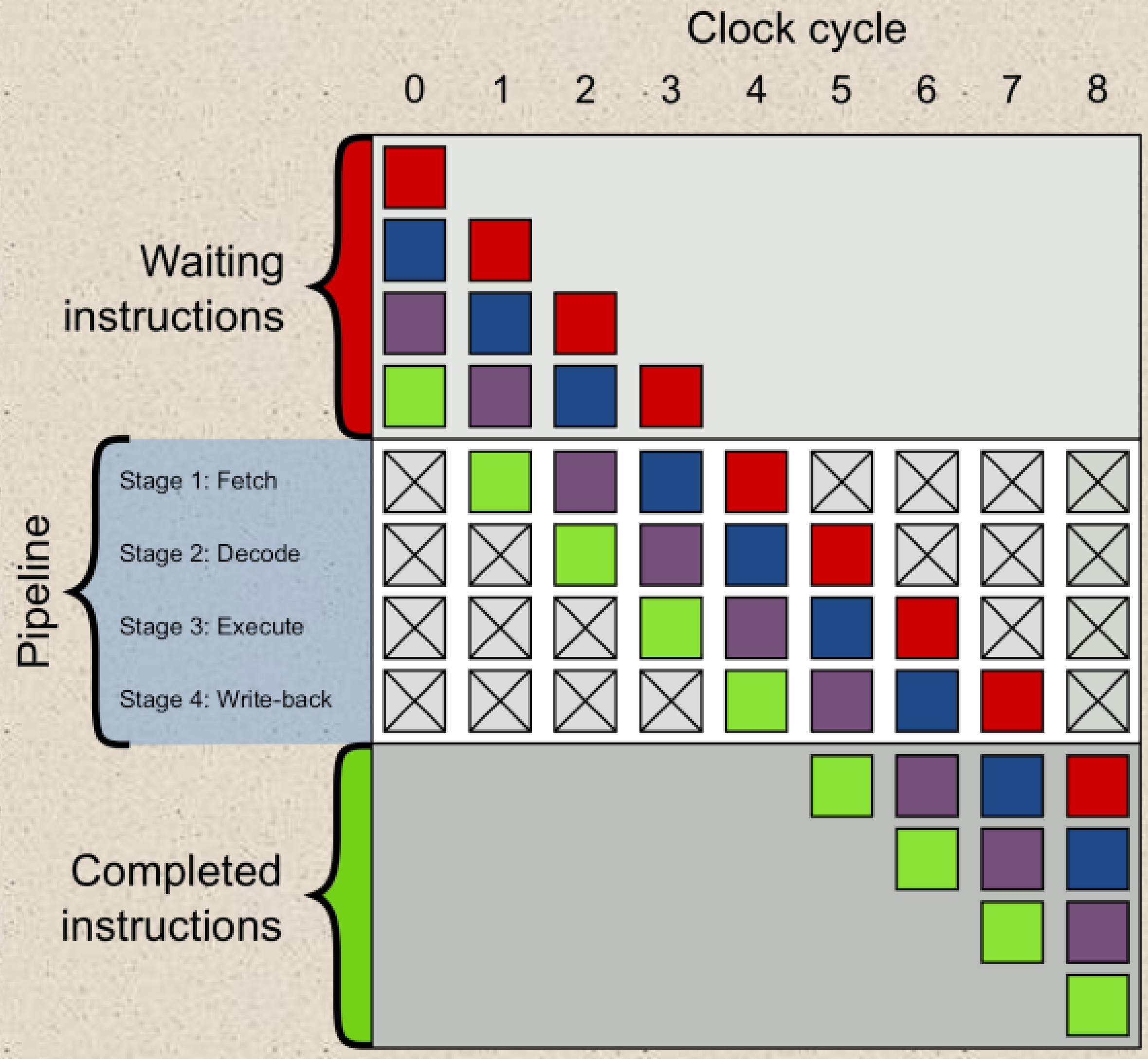
Techniques built into modern processors include:

- Pipelining
- Branch prediction
- Superscalar execution
- Data flow analysis
- Speculative execution

+ Pipelining

- Processor moves data or instructions into a conceptual pipe with all stages of the pipe processing simultaneously

+ Pipelining



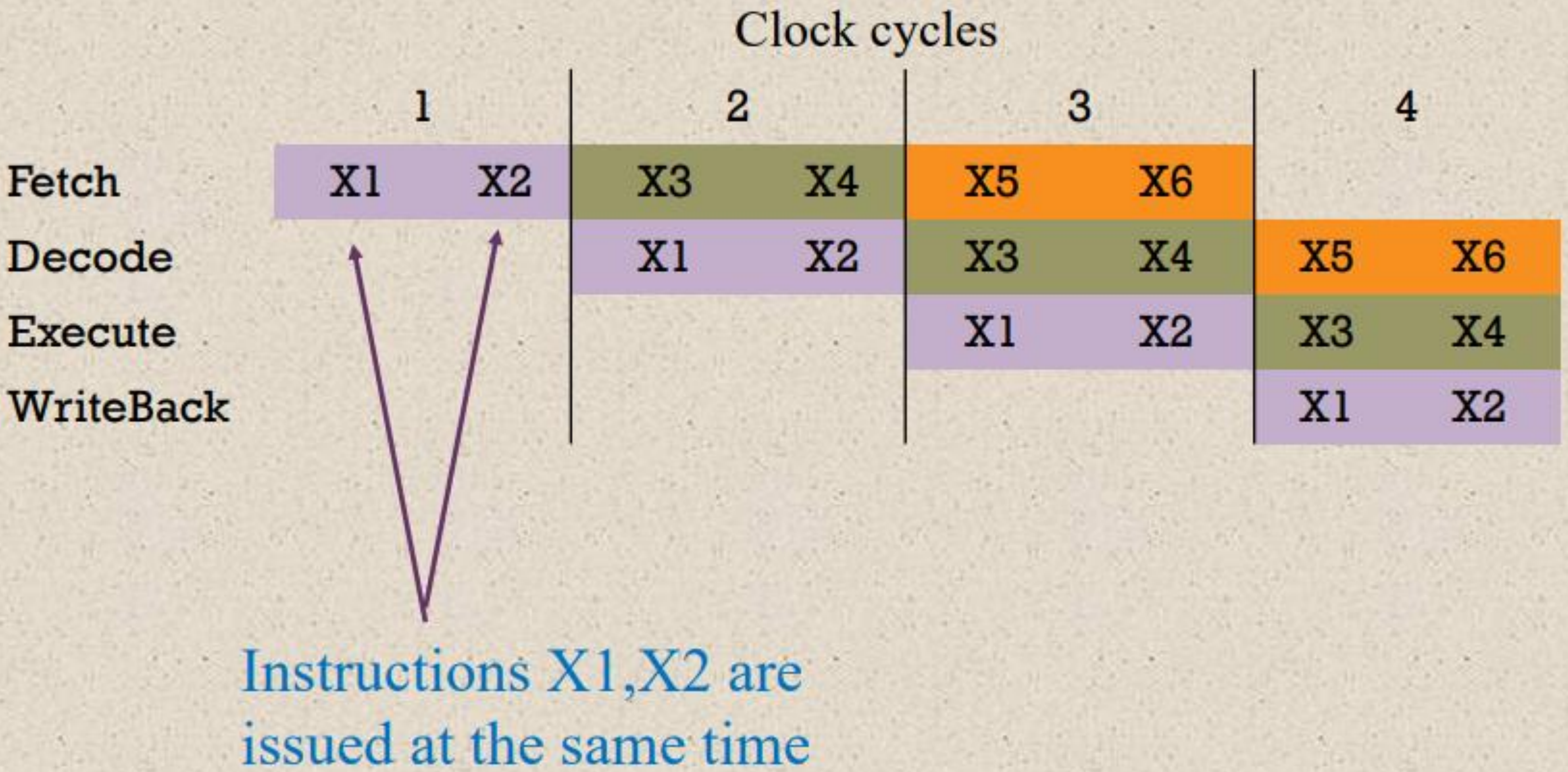
Pipelining

+ Branch prediction

- Processor looks ahead in the instruction code fetched from memory and **predicts which** branches, or groups of **instructions**, are likely to be **processed next**
- The processor fetches the predicted instructions into the pipeline.
- If the prediction is incorrect the predicted instructions are removed from the pipeline and the correct instructions are fetched.

+ Superscalar execution

- This is the ability to **issue more than one instruction** in every processor clock cycle.
- In effect, **multiple** parallel pipelines are used.



	1	1	2	2	3	3	4	4
Fetch	X1	X2	X3	X4	X5	X6		
Decode			X1	X2	X3	X4	X5	X6
Execute					X1	X2	X3	X4
WriteBack							X1	X2

+ Data flow analysis

- Processor analyzes which instructions are dependent on each other’s results, or data, to **create an optimized schedule (order) for executing instructions**

+
Speculative execution

- Using branch prediction and data flow analysis, some processors speculatively execute instructions ahead of their actual appearance in the program execution,
- holding the results in temporary locations (registers) until be used.
- The objective is to keep processor as busy as possible

+ Performance Balance

Adjust the organization and architecture to **compensate for the mismatch among the capabilities** of the various components.
particularly critical at the interface between processor and main memory

- **Architectural examples include:**

- **Increase the number of bits that are retrieved at one time by making RAMs "wider" rather than "deeper" and by using wide bus data paths**

- **Change the RAM interface to make it more efficient by including a cache or other buffering scheme on the RAM chip itself**

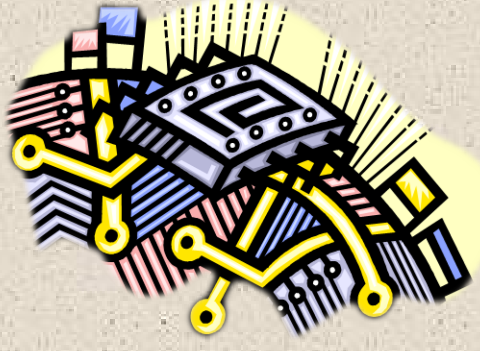
- **Reduce the frequency of memory access by using efficient cache structures between the processor and main memory**

- **Increase the interconnect bandwidth between processors and memory by using higher speed buses**

+

Improvements in Chip Organization and Architecture

- **Increase hardware speed of processor**
  - by reduction of logic gate size (**increase logic density**)
    - More gates, packed more tightly, **increasing clock rate**
    - **Reduce Propagation time for signals**
- **Increase size and speed of caches**
  - Dedicating part of processor chip for Cache **drops access times**
  - .
- **Change processor organization and architecture**
  - Increase effective speed of instruction execution
  - Parallelism



+

Problems with Clock Speed and Logic Density

- **Power**
  - Power density increases with density of logic and clock speed
  - Dissipating heat
- **Memory latency**
  - Memory speeds lag processor speeds
  - So if the clock ticks are too quick, the memory transfer operations may not be completed, **causing data loss**.

Improving performance by:

1- Multicore

The use of multiple processors on the same chip increases performance without increasing the clock rate

Strategy is to use two simpler processors on the chip rather than one more complex processor

With two processors larger caches are justified

As caches became larger it made performance sense to create two and then three levels of cache on a chip

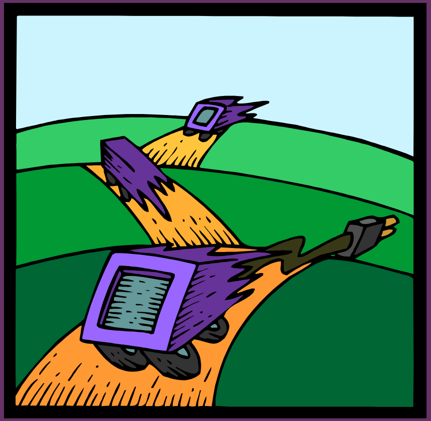
+

2-Many Integrated Core (MIC) 3-Graphics Processing Unit (GPU)

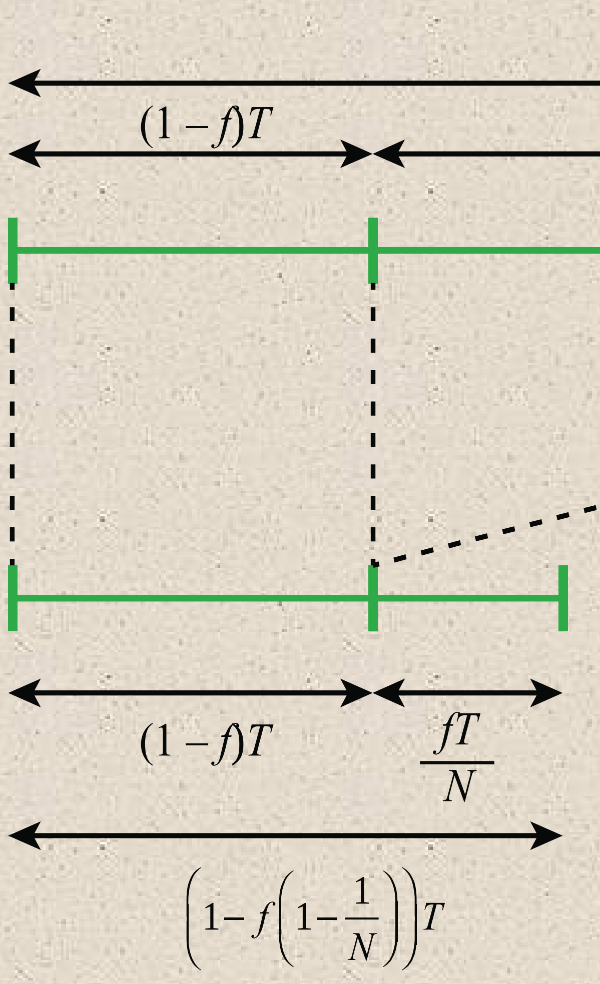
MIC	GPU
▪ The technology of integrating a large number of cores e.g. 50 or more in a processor ▪ The multicore and MIC strategy involves a homogeneous collection of general purpose processors on a single chip	▪ **Core** designed to perform parallel operations on **graphics data** ▪ Traditionally found on a plug-in graphics card, it is used to encode and render 2D and 3D graphics as well as process video ▪ Used as vector processors for a variety of applications that require repetitive computations

+

Amdahl's Law



- Gene Amdahl
- Deals with the potential speedup of a program using multiple processors compared to a single processor
- Illustrates the problems facing industry in the development of multi-core machines
  - Software must be adapted to a highly parallel execution environment to achieve the power of parallel processing
- Can be generalized to evaluate and design technical improvement in a computer system



f :a fraction that consist of code using a parallel processor with no scheduling overhead.
T : the total execution time of the program using a single processor.
N : number of processors

Figure 2.3 Illustration of Amdahl's Law

$$\text{Speedup} = \frac{\text{Time to execute program on a single processor}}{\text{Time to execute program on } N \text{ parallel processors}}$$

$$= \frac{T(1-f) + Tf}{T(1-f) + \frac{Tf}{N}} = \frac{1}{(1-f) + \frac{f}{N}}$$

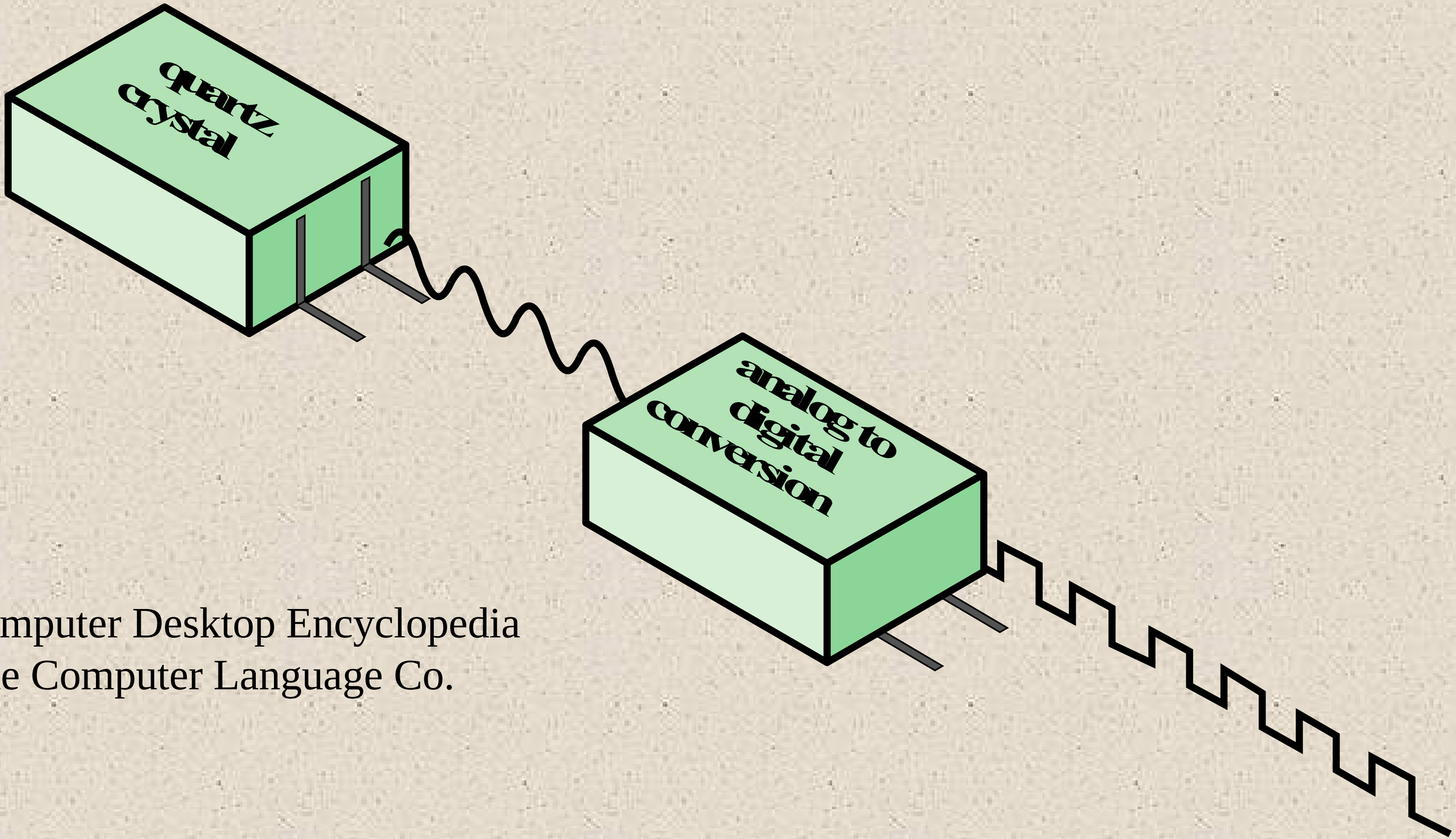
$$\text{Speedup} = \frac{\text{Performance after enhancement}}{\text{Performance before enhancement}} = \frac{\text{Execution time before enhancement}}{\text{Execution time after enhancement}}$$

$$\text{Speedup} = \frac{1}{(1-f) + \frac{f}{SU_f}}$$

SU_f : the speedup after enhancement.

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.**
- For example, a 1-GHz processor receives 1 billion pulses per second.
- The rate of pulses is known as the **clock rate**, or **clock speed**.
- One increment, or pulse, of the clock is referred to as a **clock cycle**, or a clock tick.
- The time between pulses is the **cycle time**.



From Computer Desktop Encyclopedia
1998, The Computer Language Co.

Figure 2.5 System Clock

Cycles-per-Instruction (CPI)

- This is clock cycles for executing one instruction.
- CPI depends on the type of the instruction (such as load, store, branch)
- Clock cycles required for instruction type i is CPI_i
- Average CPI is given as

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

  - I_i : The number of instructions of type i in the program
  - I_c : The total number of instructions in 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:

$$\text{MIPS rate} = \frac{I_c}{T \times 10^6} = \frac{f}{CPI \times 10^6} \qquad \textbf{(2.3)}$$

EXAMPLE 2.2 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

The average *CPI* when the program is executed on a uniprocessor with the above trace results is $CPI = 0.6 + (2 \times 0.18) + (4 \times 0.12) + (8 \times 0.1) = 2.24$. The corresponding MIPS rate is $(400 \times 10^6)/(2.24 \times 10^6) \approx 178$.

Calculating the Mean

The use of benchmarks to compare systems involves calculating the mean value of a set of data points related to execution time

The three common formulas used for calculating a mean are:

- Arithmetic
- Geometric
- Harmonic

Arithmetic Mean

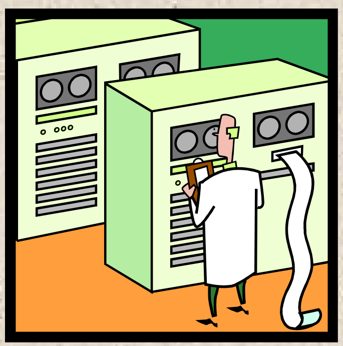
- An Arithmetic Mean (AM) is an appropriate measure if the sum of all the measurements is a meaningful and interesting value
- The AM is a good candidate for comparing the execution time performance of several systems

$$AM = \frac{x_1 + \cdots + x_n}{n} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

For example, suppose we were interested in using a system for large-scale simulation studies and wanted to evaluate several alternative products. On each system we could run the simulation multiple times with different input values for each run, and then take the average execution time across all runs. The use of multiple runs with different inputs should ensure that the results are not heavily biased by some unusual feature of a given input set. The AM of all the runs is a good measure of the system's performance on simulations, and a good number to use for system comparison.

+

- The AM used for a time-based variable, such as program execution time, has the important property that it is directly proportional to the total time
  - If the total time doubles, the mean value doubles



Example1:

Our favorite 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?

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

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

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

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

CPU time for B can be found using this equation:

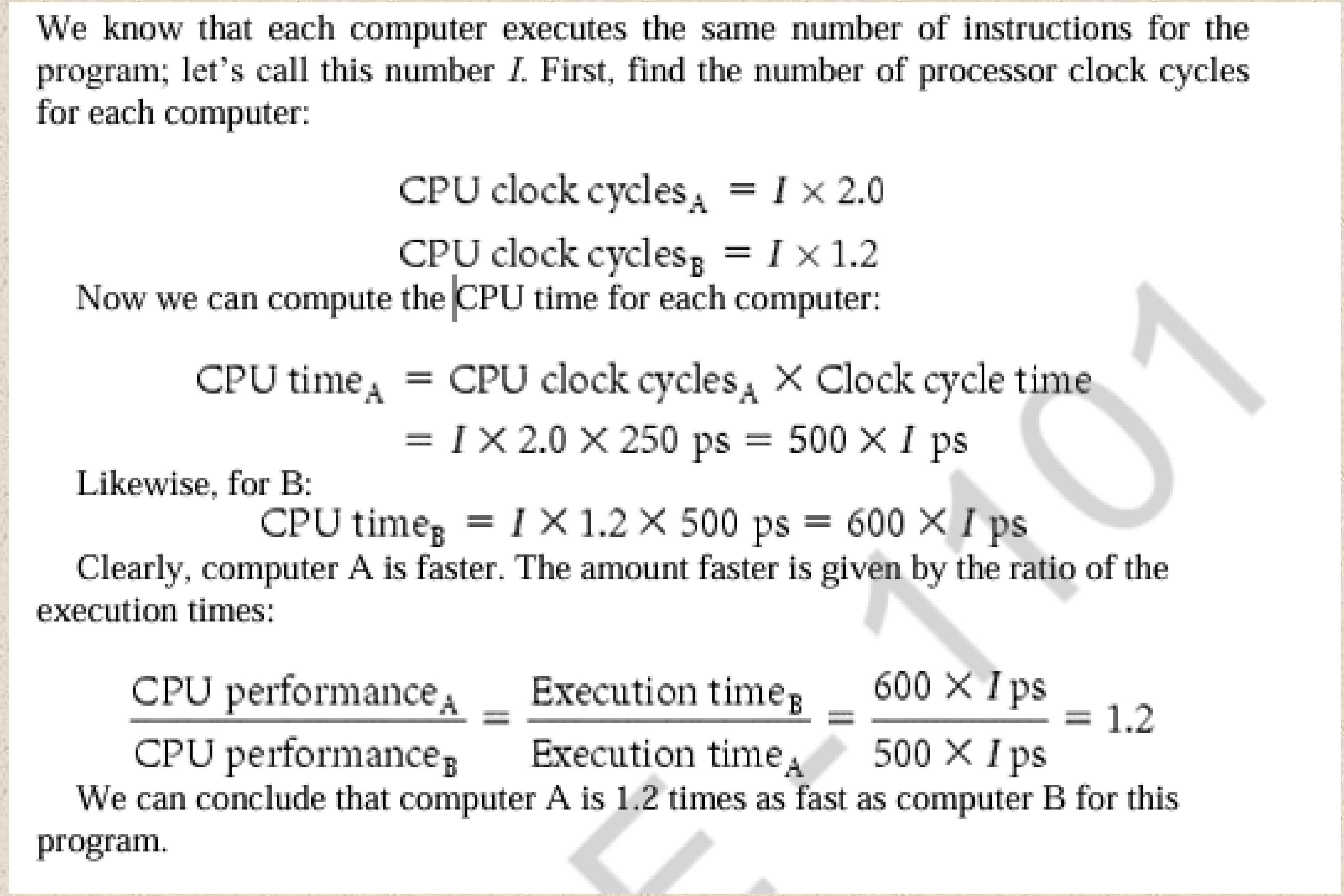
$$\text{CPU time}_B = \frac{1.2 \times \text{CPU clock cycles}_A}{\text{Clock rate}_B}$$

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

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

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

Example 2: 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:

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

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.

Example 3: Consider three different processors P1, P2, and P3 executing the same instruction set. P1 has 3 GHz clock rate and a CPI of 1.5. P2 has a 2.5 GHz clock rate and a CPI of 1.0. P3 has a 4.0 GHz clock rate and has a CPI of 2.2.

a. Which processor has the highest performance expressed in instructions per second?

b. If the processors each execute a program in 10 seconds, find the number of cycles and the number of instructions.

c. We are trying to reduce the execution time by 30% but this leads to an increase of 20% in the CPI. What clock rate should we have to get this time reduction?

Ans:

1.P1: 3GHz / 1.5 = 2 * 10^9 instructions per second
 P2: 2.5GHz / 1.0 = 2.5 * 10^9 instructions per second
 P3: 4GHz / 2.2 = 1.82 * 10^9 instructions per second
So P2 has the highest performance among the three.

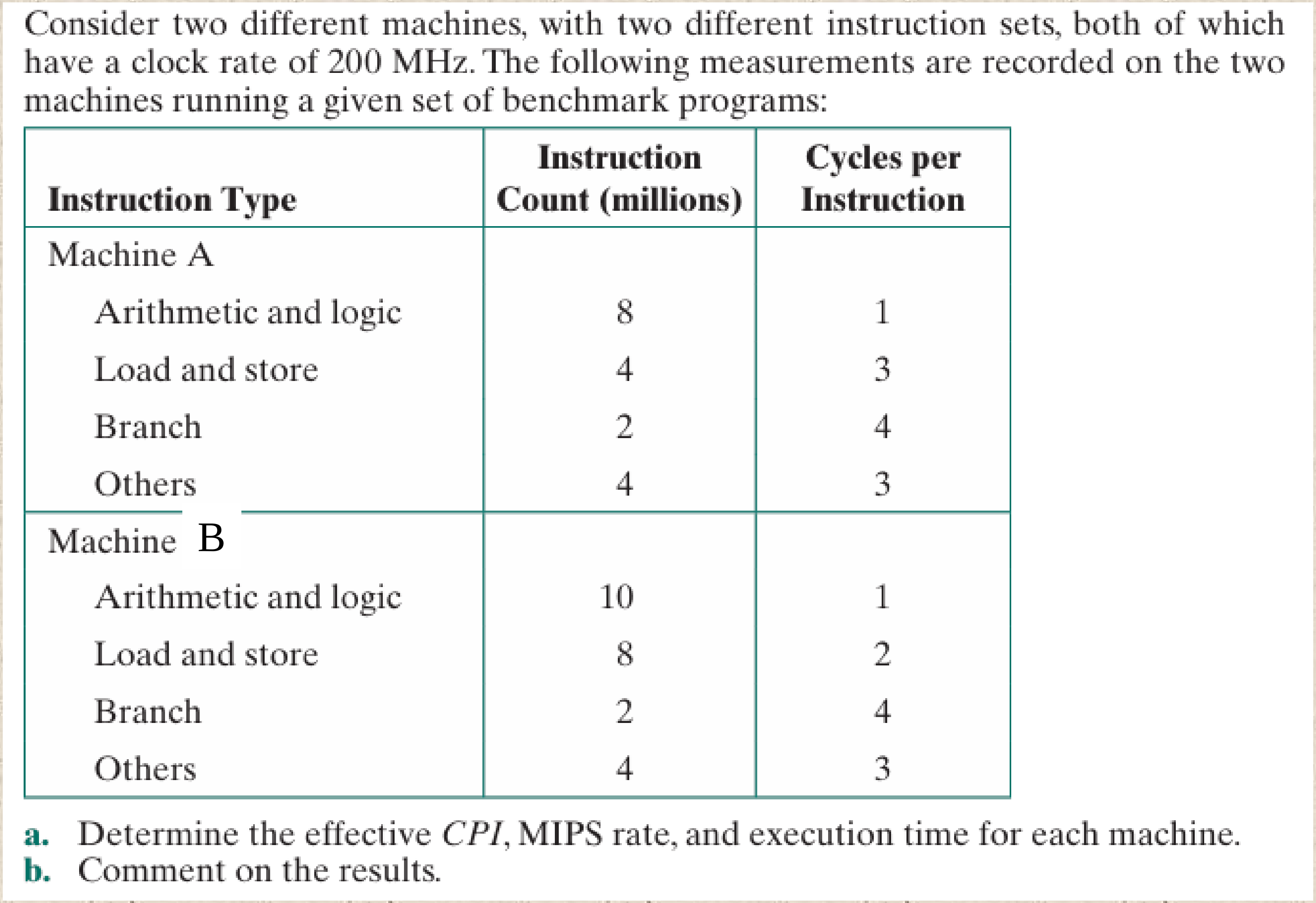
2. Cycles: P1: 3GHz * 10 = 3 * 10^10 cycles
P2: 2.5GHz * 10 = 2.5 * 10^10 cycles
P3: 4GHz * 10 = 4 * 10^10 cycles

3.Num of instructions: P1: 3GHz * 10 / 1.5 = 2 * 10^10 instructions
P2: 2.5GHz * 10 / 1.0 = 2.5 * 10^10 instructions
P3: 4GHz * 10 / 2.2 = 1.82 * 10^10 instructions

4. Execution time = (Num of instructions * CPI) / (Clock rate)
So if we want to reduce the execution time by 30%, and CPI increases by 20%, we have: Execution time * 0.7 = (Num of instructions * CPI * 1.2) / (New Clock rate)
New Clock rate = Clock rate * 1.2 / 0.7 = 1.71 * Clock rate
New Clock rate for each processor: P1: 3GHz * 1.71 = 5.13 GHz
P2: 2.5GHz * 1.71 = 4.27 GHz
P3: 4GHz * 1.71 = 6.84 GHz

Write down the five stages of Instruction Executions

- Stage 1 (Instruction Fetch) In this stage the CPU reads instructions from the address in the memory whose value is present in the program counter.
- Stage 2 (Instruction Decode) In this stage, instruction is decoded and the register file is accessed to get the values from the registers used in the instruction.
- Stage 3 (Instruction Execute) In this stage, ALU operations are performed.
- Stage 4 (Memory Access) In this stage, memory operands are read and written from/to the memory that is present in the instruction.
- Stage 5 (Write Back) In this stage, computed/fetched value is written back to the register present in the instructions



Consider two different machines, with two different instruction sets, both of which have a clock rate of 200 MHz. The following measurements are recorded on the two machines running a given set of benchmark programs:

Instruction Type	**Instruction Count (millions)**	**Cycles per Instruction**
Machine A		
Arithmetic and logic	8	1
Load and store	4	3
Branch	2	4
Others	4	3
Machine B		
Arithmetic and logic	10	1
Load and store	8	2
Branch	2	4
Others	4	3

a. Determine the effective *CPI*, MIPS rate, and execution time for each machine.
b. Comment on the results.

Review Questions

1- List and briefly define some of the techniques used in contemporary processors to increase speed.
2- Explain the concept of performance balance.
3- Explain the differences among multicore systems, MICs, and GPUs.
4- What are clock and clock cycles?
5- What is instruction register?
6- What is program counter?
7- List out the methods used to improve system performance.
8- What is pipelining?
9- What are the advantages of pipelining?
10- What are the advantages and disadvantages of multicore processor?