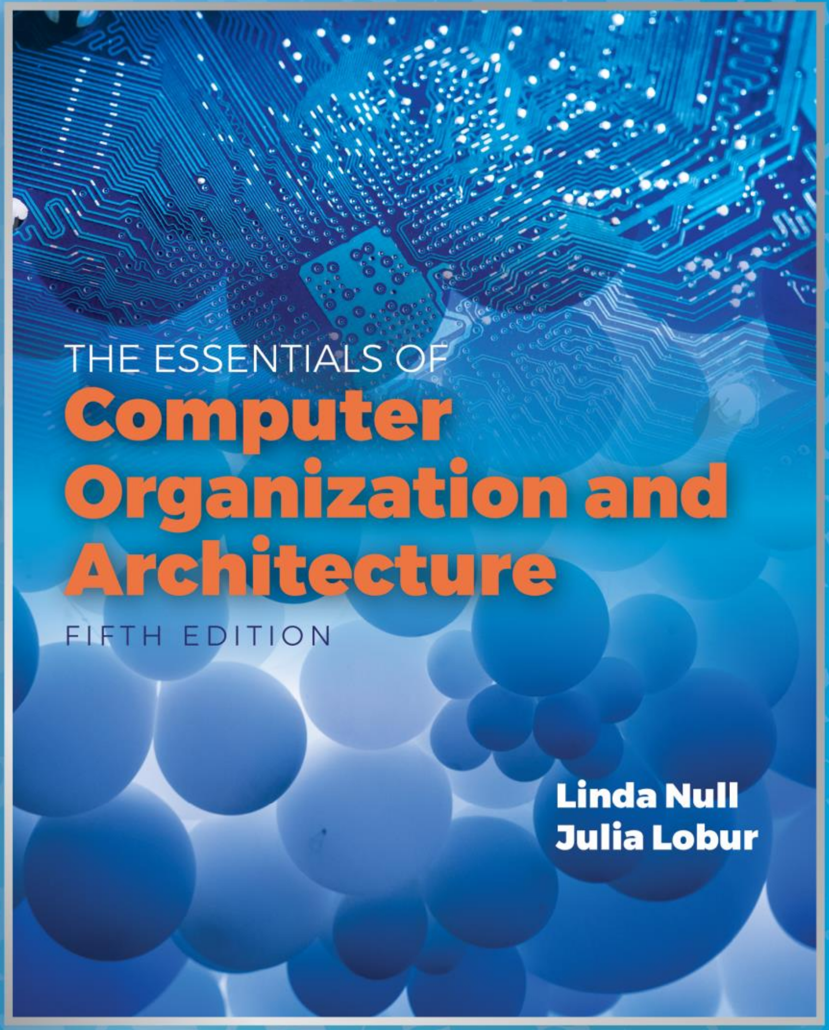
This is the first lecture of Chapter 11

# Chapter 11

## Performance Measurement and Analysis (A)



## Objectives

- Understand the ways in which computer performance is measured.
- Be able to describe common benchmarks and their limitations.
- Become familiar with factors that contribute to improvements in CPU and disk performance.

## 11.1 Introduction

- The ideas presented in this chapter will help you to understand various measurements of computer performance.
- You will be able to use these ideas when you are purchasing a large system, or trying to improve the performance of an existing system.
- We will discuss a number of factors that affect system performance, including some tips that you can use to improve the performance of programs.

## 11.2 Computer Performance Equations (1 of 3)

- The basic computer performance equation has been useful in our discussions of RISC versus CISC:

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

- To achieve better performance, RISC machines reduce the number of cycles per instruction, and CISC machines reduce the number of instructions per program.

## 11.2 Computer Performance Equations (2 of 3)

- We have also learned that CPU efficiency is not the sole factor in overall system performance. Memory and I/O performance are also important.
- Amdahl’s Law tells us that the system performance gain realized from the speedup of one component depends not only on the speedup of the component itself, but also on the fraction of work done by the component:

$$S = \frac{1}{(1 - f) + (f / k)}$$

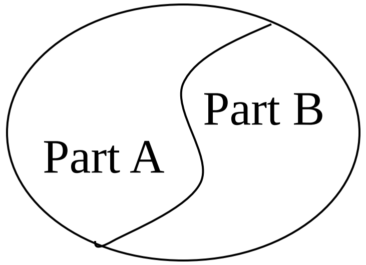
## 11.2 Computer Performance Equations (3 of 3)

- In short, using Amdahl’s Law we know that we need to make the common case fast.
- So if our system is CPU bound, we want to make the CPU faster.
- A memory bound system calls for improvements in memory management.
- The performance of an I/O bound system will improve with an upgrade to the I/O system.

Of course, fixing a performance problem in one part of the system can expose a weakness in another part of the system!

# Quantitative Principles of Computer Design

$$\text{Speedup} = \frac{\text{Execution time without using the enhancement}}{\text{Execution time using the enhancement}}$$



## Amdahl's Law

The performance improvement to be gained from using some fast mode of execution is limited by the fraction of the time the fast mode can be used.

$$Speedup = \frac{Time_{old}}{Time_{new}} = \frac{Time_A + Time_B}{Time_A + Time_{enhancedB}}$$

$$= \frac{1}{\frac{Time_A}{Time_A + Time_B} + \frac{Time_B}{Time_A + Time_B} \times \frac{Time_{enhancedB}}{Time_B}}$$

$$= \frac{1}{(1 - Fraction_{enhanced}) + \frac{Fraction_{enhanced}}{Speedup_{enhanced}}}$$

$$Fraction_{enhanced} = \frac{Time_B}{Time_A + Time_B}$$

$$Speedup_{enhanced} = \frac{Time_B}{Time_{enhancedB}}$$

Make the common case fast

# An Example

A processor contains a Floating Point (FP) unit, which, in turn, contains an FP Square Root (SQR) instruction.

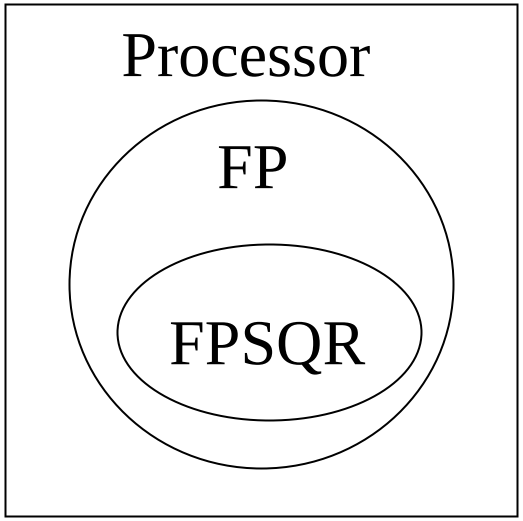
For a typical program,

Frequency of FPSQR = 20%

Frequency of FP = 50%

Two improvement selections:

(1) $\text{Speedup}_{SQR}=10$. (2) $\text{Speedup}_{FP}=2$



$$Speedup(1) = \frac{1}{(1-0.2)+\frac{0.2}{10}} = \frac{1}{0.82} \approx 1.22$$

$$Speedup(2) = \frac{1}{(1-0.5)+\frac{0.5}{2}} = \frac{1}{0.75} \approx 1.33$$

$\Rightarrow$ Improvement selection (2) is better than Improvement selection (1).

# Parallel Speedup

- In the context of parallel processing, speedup can be computed by

$$\text{Speedup} = \frac{T_{\text{Sequential}}}{T_{\text{Parallel}}}$$

- Amdahl's law, for *p* processors and a fraction *f* of unparallelizable code:

$$\text{Speedup} = \frac{1}{f + \frac{1-f}{p}}$$

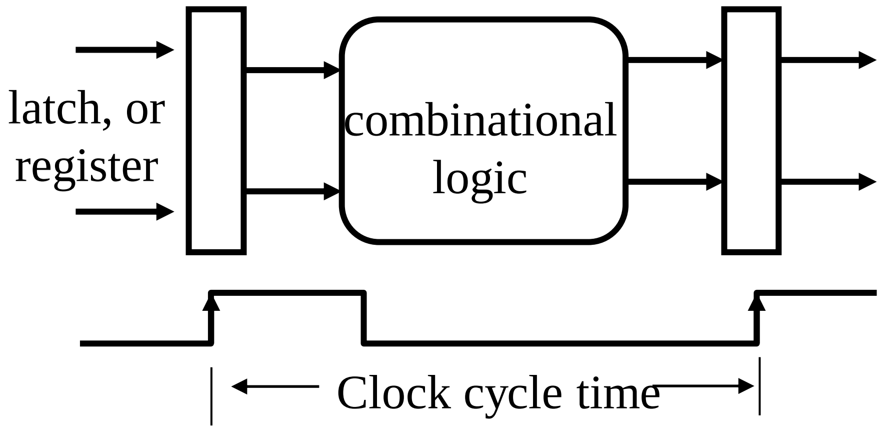
- For example, if *f* = 10% of the operations must be performed sequentially, then speedup can be no greater than 10 regardless of how many processors are used:

$$\text{Speedup} = \frac{1}{0.1 + \frac{0.9}{10}} \cong 5.3 \qquad \text{Speedup} = \frac{1}{0.1 + \frac{0.9}{\infty}} = 10$$

P = 10 processors

P = ∞ processors

# The CPU Performance Equation



- Clock cycle time (clock period)
- Clock frequency (clock rate)

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

$$CPU\ \text{time} = \text{Clock cycles} \times \text{Clock period} = \frac{\text{Clock Cycles}}{\text{Clock rate}}$$

$$\text{Clock cycles per instruction } (CPI) = \frac{\text{Clock cycles}}{\text{Instruction count } (IC)}$$

$$CPU\ \text{time} = CPI \times IC \times \text{Clock cycle time} = \frac{CPI \times IC}{\text{Clock rate}}$$

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

# Three characteristics

$$\text{CPU time} = \frac{\text{Seconds}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Cycle}}$$

| | Inst Count | CPI | Clock period |
|---|---|---|---|
| Program | X | | |
| Compiler | X | (X) | |
| Inst. Set. | X | X | (X) |
| Organization | | X | X |
| Technology | | | X |

# Average Clock Cycles per Instruction

$$CPU \text{ clock cycles} = \sum_{i=1}^{n} CPI_i \times IC_i$$

$$CPI = \frac{\sum_{i=1}^{n} CPI_i \times IC_i}{IC} = \sum_{i=1}^{n} \left( \frac{IC_i}{IC} \right) \times CPI_i$$

$$CPU \text{ times} = \left( \sum_{i=1}^{n} CPI_i \times IC_i \right) \times \text{Clock period}$$

**Example: Calculating CPI bottom up**

| Op | Freq | $CPI_i$ | $Freq*CPI_i$ | (% Time) |
|---|---|---|---|---|
| ALU | 50% | 1 | .5 | (33%) |
| Load | 20% | 2 | .4 | (27%) |
| Store | 10% | 2 | .2 | (13%) |
| Branch | 20% | 2 | .4 | (27%) |

CPI = 1.5

Typical mix of instruction types in program

## An Example

Measurements have shown that for an unpipelined machine A, the instruction mix and the average Clock cycle Per Instruction (CPI) are as shown in the following table.

| Operation | Frequency | CPI |
|---|---|---|
| ALU instruction | 40% | 4 |
| Load/Store | 40% | 5 |
| Branch | 20% | 4 |

One improvement on the memory system can reduce the CPI for loads and stores to 3. This improvement, however, leads to a 5% increase in the clock cycle time. Is this improvement useful?

$\text{CPU time}_{old} = (40\%\times4+40\%\times5+20\%\times4)\times\text{IC}\times\text{Cycle time} = 4.4\times\ \text{IC}\times\text{Cycle time}$

$\text{CPU time}_{new} = (40\%\times4+40\%\times3+20\%\times4)\times\text{IC}\times1.05\times\text{Cycle time} = 3.78\times\text{IC}\times\text{Cycle time}$

This improvement is useful because it reduces the CPU time.

# 11.3 Mathematical Preliminaries (1 of 17)

- Measures of system performance depend upon one’s point of view.
  - A computer user is most often concerned with response time: How long does it take the system to carry out a task?
  - System administrators are usually more concerned with throughput: How many concurrent tasks can the system handle before response time is adversely affected?
- These two ideas are related: If a system carries out a task in k seconds, then its throughput is 1/*k* of these tasks per second.