# Computer Abstractions and Technology

Prof. Rajib Ranjan Maiti
CSIS, BITS-Pilani, Hyderabad

# Below Your Program

# Abstraction

- the hardware can
  - only executes extremely simple low-level instructions
- Complex applications
  - have to convert to those simple instructions
  - involves several layers of software
  - high-level operations interpret or translate into simple instructions
- One of  the greatest tools
  - abstraction

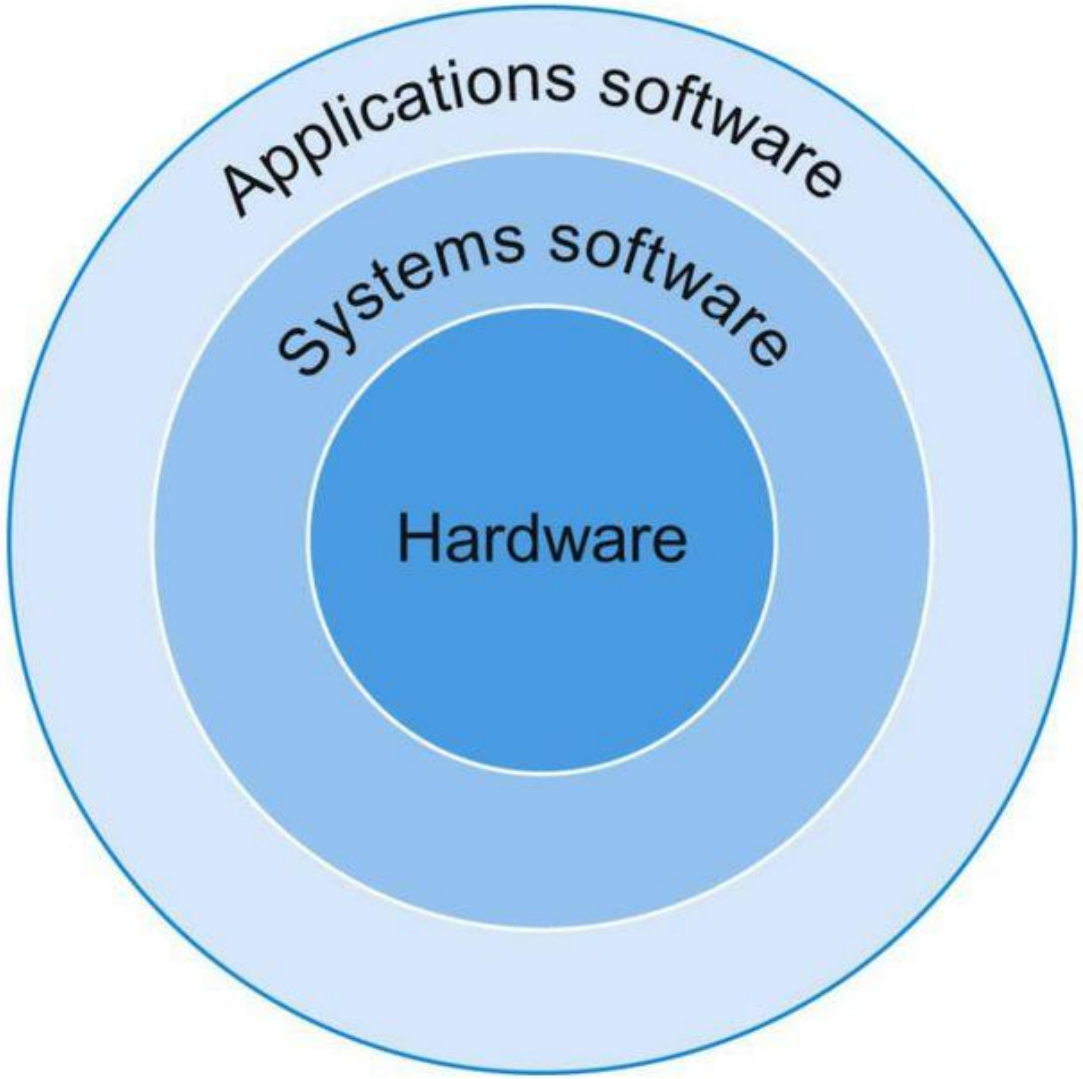



FIGURE 1.3 A simplified view of hardware and software as hierarchical layers, shown as concentric circles with hardware in the center and applications software outermost. In complex applications, there are often multiple layers of application software as well. For example, a database system may run on top of the systems software hosting an application, which in turn runs on top of the database.

# Understanding Program Performance

| Hardware or software component | How this component affects performance |
|---|---|
| Algorithm | Determines both the number of source-level statements and the number of I/O operations executed |
| Programming language, compiler, and architecture | Determines the number of machine instructions for each source-level statement |
| Processor and memory system | Determines how fast instructions can be executed |
| I/O system (hardware and operating system) | Determines how fast I/O operations may be executed |

Compilers being system software translate a program written in a high-level language, such as C, C , Java, or Visual Basic into instructions that the hardware can execute.

# System Software

## Operating System

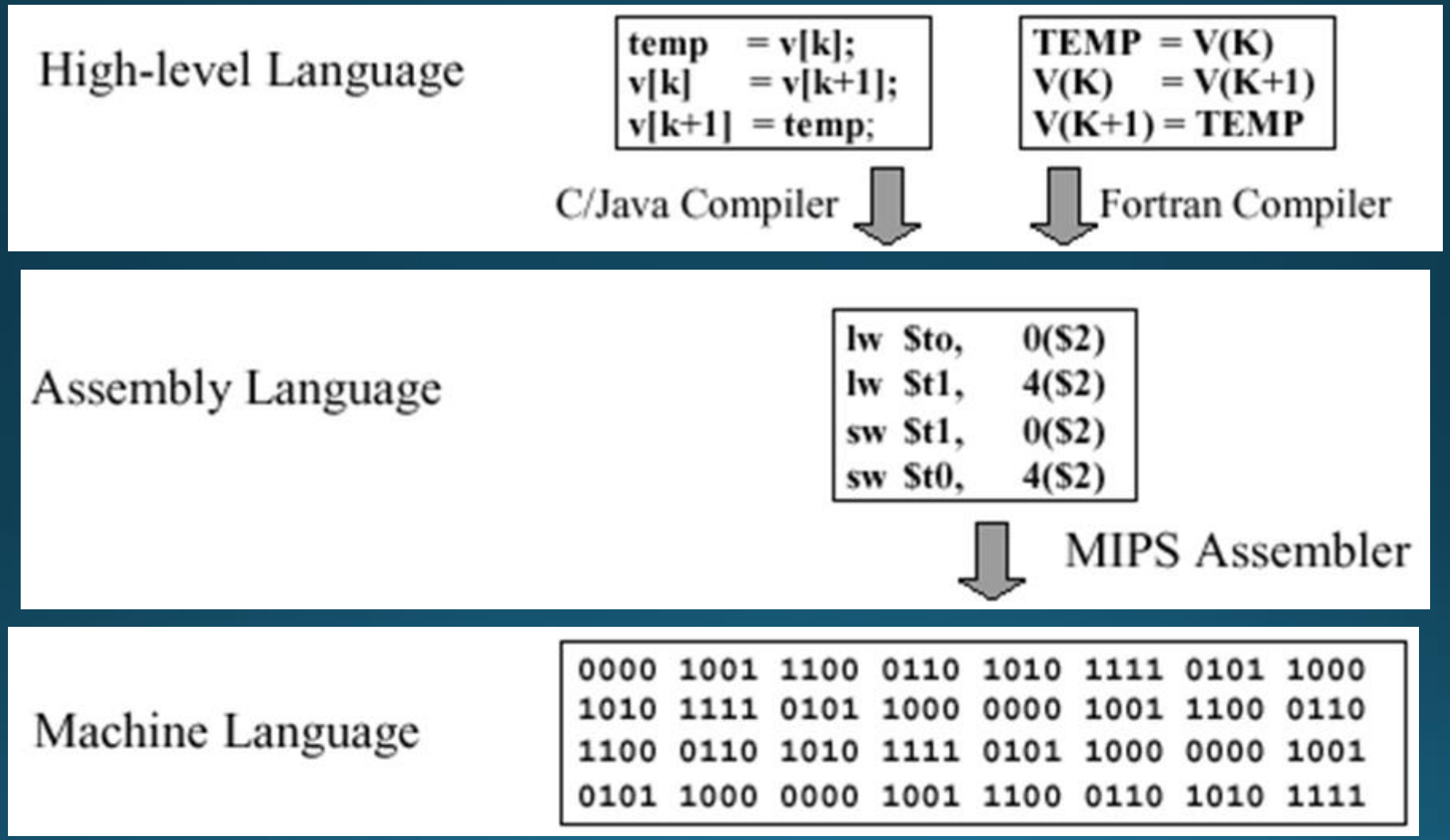
provides a variety of services and supervisory functions

## Compiler

translates a program written in a high-level language, such as C or Java, into instructions that the hardware can execute

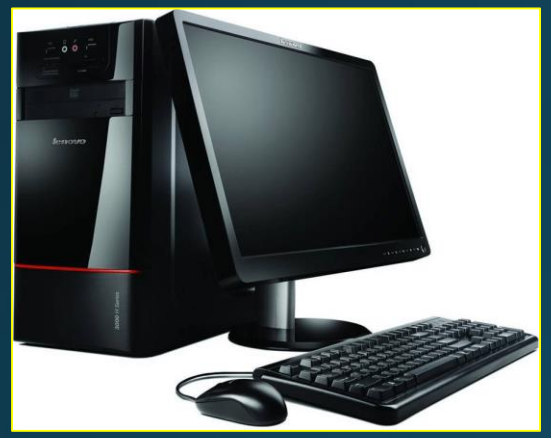
- **The Language of Hardware**
  - Machines understand ***on*** and ***off***,
  - So, the machine alphabet is just two letters : {0,1}

# Example Translation



# Performance Measures

# Performance



# Performance

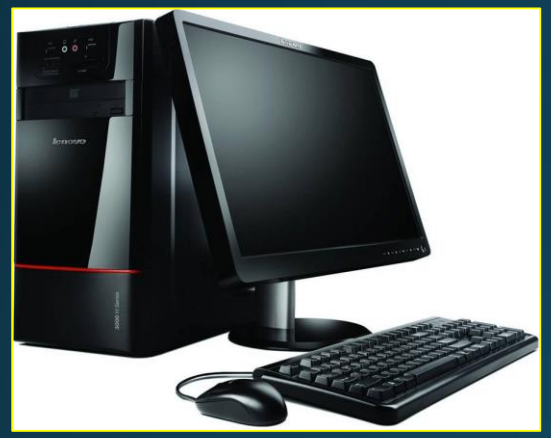
- An analogy with Airplane

| Airplane | Passenger capacity | Cruising range (miles) | Cruising speed (m.p.h.) | Passenger throughput (passengers x 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 |

**FIGURE 1.14 The capacity, range, and speed for a number of commercial airplanes.** The last column shows the rate at which the airplane transports passengers, which is the capacity times the cruising speed (ignoring range and takeoff and landing times).

- Which model is better?

# Performance



- If PC, then response time

**Response time or execution time:**
time for disk accesses
+ time for memory accesses
+ time for I/O activities
+ operating system overhead
+ CPU execution time
+ so on.

- If server, then #completed jobs/day

**Throughput or bandwidth:** the total amount of work done in a given time, useful in datacenters

Decreasing response time almost always improves throughput.

# Important Formulas

- To maximize performance
  - minimize response time or execution time for some task.

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

- Thus, for a computer X
  - performance and execution time are inversely proportional

- For two computers X and Y,
  - if the performance of X is greater than the performance of Y, then

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

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

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

- X is *n* times faster than Y:
  - $\eta = \frac{E_y}{E_x} = \frac{P_x}{P_y}$,
  - E ≡ Execution time, P ≡ Performance

# Example

- 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?

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

- Ans:
  - A is therefore 1.5 times as fast as B.

# Performance Measurement

- **CPU execution time** or **simply CPU time**
  - the time the CPU spends computing for this task
  - does not include time spent waiting for I/O or running other programs.

CPU time = CPU time spent in the program (called user CPU time) + CPU time spent in the operating system performing tasks on behalf of the program (called system CPU time)

# Clock cycles

- Almost all computers
  - constructed using a clock
- Clock creates discrete time intervals, called as
  - **clock cycles**, or **ticks**, or **clock ticks**, or **clock periods**, or **clocks**, or **cycles**.
- The length of a clock period
- For example,
  - clock period = 250 picoseconds, or 250 ps,
  - and the clock rate = 4 gigahertz, or 4 GHz)

CPU time n terms of clock cycles and clock cycle time

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

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

# Example

- 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:

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

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

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

# Example

- 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?

CPU time for B can be found using this equation:

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

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

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

# Instruction Performance

- the execution time must depend on the number of instructions in a program

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

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

# Example

- 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.
  - 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:

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

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

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

$$\text{CPU time}_B = I \times 1.2 \times 500 \text{ ps} = 600 \times I \text{ ps}$$

$$\frac{\text{CPU performance}_A}{\text{CPU performance}_B} = \frac{\text{Execution time}_B}{\text{Execution time}_A} = \frac{600 \times I \text{ ps}}{500 \times I \text{ ps}} = 1.2$$

# CPU Time

*CPU time* = Instruction count X Clock cycles per instruction X Clock cycle time

# Example: CPU Performance

- Suppose we made the following measurements:
  - Frequency of FP operations = 25%
  - Average CPI of FP operations = 4.0
  - Average CPI of other instructions = 1.33
  - Frequency of FSQRT = 2%
  - CPI of FSQRT = 20
- Assume that the two design alternatives
  - decrease the CPI of FSQRT to 2, or
  - decrease the average CPI of all FP operations to 2.5.
- Compare these two design alternatives using the processor performance equation (CPU time).

# Example: CPU Performance

- Suppose we made the following measurements:
  - Frequency of FP operations = 25%
  - Average CPI of FP operations = 4.0
  - Average CPI of other instructions = 1.33
  - Frequency of FSQRT = 2%
  - CPI of FSQRT = 20
- Assume that the two design alternatives
  - decrease the CPI of FSQRT to 2, or
  - decrease the average CPI of all FP operations to 2.5.
- Compare these two design alternatives using the processor performance equation (CPU time).

- Ans.
  - First, observe that only the CPI changes;
    - the clock rate and instruction count remain identical.
  - We start by finding the original CPI with neither enhancement:
    - $\text{CPI}_{\text{original}} = \sum_{i=1}^{n} CPI_i \times \left(\frac{IC_i}{IC}\right)$
    - = (4 X 25% )+ (1:33 X 75%) = 2.0

# Example Contd.

- Suppose we made the following measurements:
  - Frequency of FP operations = 25%
  - Average CPI of FP operations = 4.0
  - Average CPI of other instructions = 1.33
  - Frequency of FSQRT = 2%
  - CPI of FSQRT = 20
- Assume that the two design alternatives
  - decrease the CPI of FSQRT to 2, or
  - decrease the average CPI of all FP operations to 2.5.
- Compare these two design alternatives using the processor performance equation (CPU time).

- Ans (Contd).
  - The CPI for the enhanced FSQRT: the original CPI - the cycles saved from improvements:
    - $CPI_{\text{with new FPSQR}} = CPI_{\text{original}} - 2\% \, (CPI_{\text{old FPSQR}} - CPI_{\text{of new FPSQR only}})$
    - $= 2 - 2\% \, (20-2) = 1.64$
  - The CPI for the enhancement of all FP: CPI of all FP + CPI of all non-FP
    - $CPI_{\text{new FP}} = 75\% \text{ X } 1{:}33 + 25\% \text{ X } 2.5 = 1.625$

- Ans (Contd).
  - Since the CPI of the overall FP enhancement is slightly lower, its performance will be marginally better.
  - So, the speedup for the overall FP enhancement
    - $\text{Speedup}_{\text{new FP}} = \frac{CPI_{\text{original}}}{CPI_{\text{new FP}}} = 2/1.625 = 1.23$

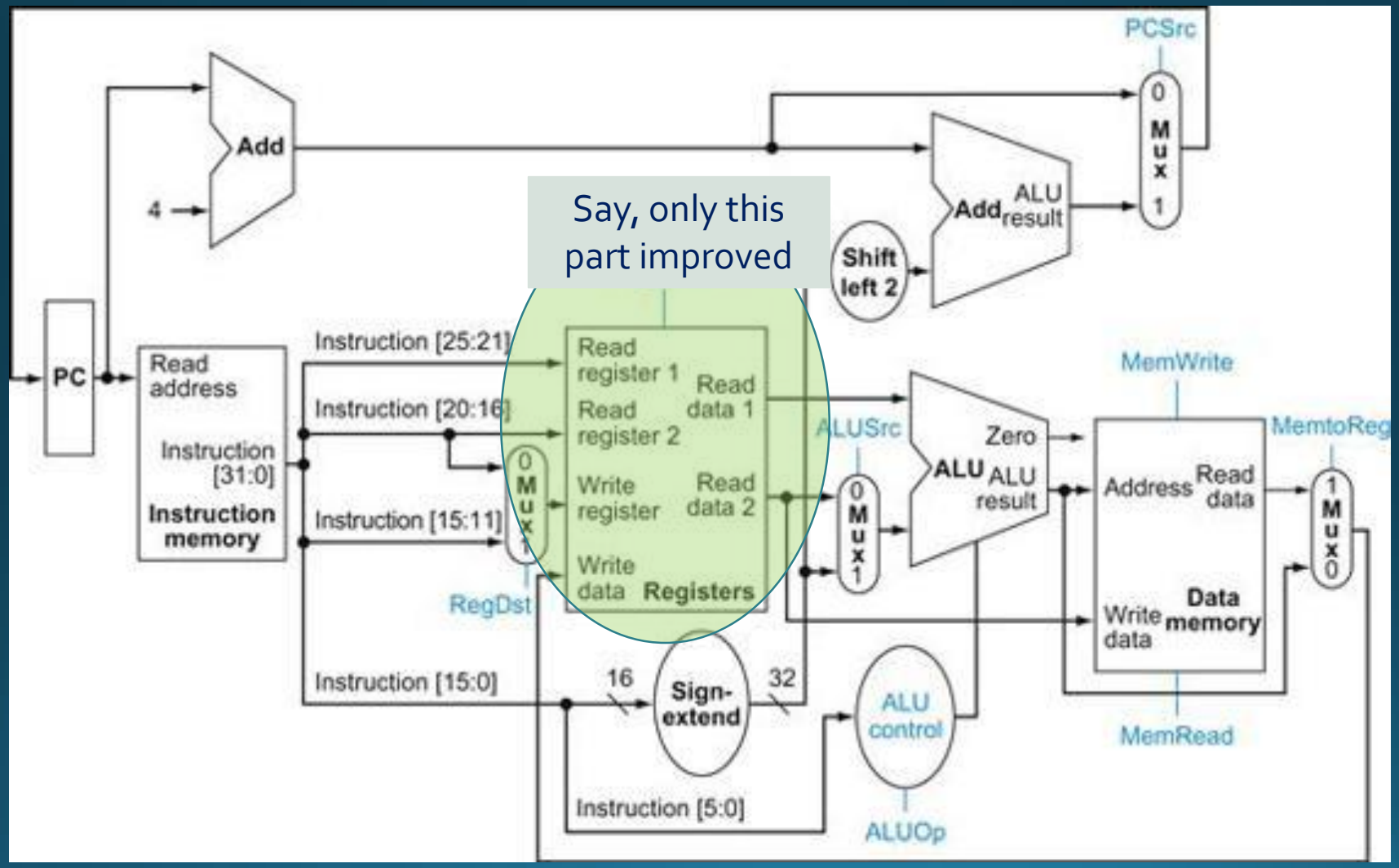
# Amdahl's Law

- Amdahl's Law defines
  - the **speedup** that can be gained by **using a particular feature**
- What is speedup?
  - Suppose that we can make an enhancement to a computer that will improve performance when it is used.
  - Speedup = $\frac{\text{Performance for entire task using the enhancement when possible}}{\text{Performance for entire task without using the enhancement}}$
  - Alternately,
  - Speedup = $\frac{\text{Execution time for entire task without using the enhancement}}{\text{Execution time for entire task using the enhancement when possible}}$
- Speedup tells us how much faster a task will run using the computer with the enhancement contrary to the original computer

# Amdahl's Law

- *Amdahl's Law:*
  - 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
  - $S_0 = \frac{E_{old}}{E_{new}} = \frac{1}{(1 - Fr_e) + \frac{Fr_{enh}}{s_{enh}}}$ ,
    - $S_o$ ≡ Speedup Overall,
    - $E_{old}$ = Execution time old,
    - $E_{new}$ = Execution new,
    - $Fr_{enh}$ ≡ Fraction enhanced,
    - $S_{enh}$ ≡ Speedup Enhanced

# Application of Amdhal's law on Architecture



# Example : Amdahl's Law

- Suppose that we want to enhance the processor used for web serving.
  - The new processor is 10 times faster on computation in the web serving application than the old processor
  - Assuming that the original processor is busy with computation 40% of the time and is waiting for I/O 60% of the time, what is the overall speedup gained by incorporating the enhancement?

# Example : Amdahl's Law

- Suppose that we want to enhance the processor used for web serving.
  - The new processor is 10 times faster on computation in the web serving application than the old processor
  - Assuming that the original processor is busy with computation 40% of the time and is waiting for I/O 60% of the time, what is the overall speedup gained by incorporating the enhancement?

- Ans.
  - $\text{Fraction}_{\text{enhanced}} = 0.4;$
  - $\text{Speedup}_{\text{enhanced}} = 10;$
  - $\text{Speedup}_{\text{overall}} = \frac{1}{(1-0.4) + \frac{0.4}{10}} = \frac{1}{0.64} = 1.56$


Thank You