ICS 233
COMPUTER ARCHITECTURE

Computer Performance

Lecture Outline

- ❑ Performance Issues
- ❑ Performance Metrics

ICS 233
COMPUTER ARCHITECTURE

Computer Performance

Lecture 17

Performance

Two factors are to be taken into account when considering the performance of a computer system:

- **Response Time** -The time between the start and completion of a task. Also referred to as **Execution Time**

- **Throughput** - Total amount of work done in a given time

Performance

To improve performance

- reduce Response time

- increase Throughput

Performance

Question 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, handling an airline reservation system.

Performance

Answer

Decreasing the response time always improves performance

Hence,

In case a) both response time and throughput are improved

In case b) no one task gets work done faster, so only throughput increases

Maximize Performance

To maximize performance, response time or execution time is to be minimized

Hence, we relate Performance & Execution time for a machine X as

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

Maximize Performance

For two machines X & Y :

If performance of X is greater than performance of Y,

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

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

Or $\text{Execution Time}_Y > \text{Execution Time}_X$

If X is ‘n’ times faster than Y, then

$$\frac{\text{Performance}_X}{\text{Performance}_Y} = n$$

If X is ‘n’ times faster than Y, then the execution time on Y is ‘n’ times longer than X

$$\frac{\text{Performance}_X}{\text{Performance}_Y} = \frac{\text{Execution Time}_Y}{\text{Execution Time}_X} = n$$

Maximize Performance

Question 2:

If machine A runs a program in 10 seconds and machine B runs the same program in 15 seconds, how much faster is A than B?

We say, machine A is ‘n’ times faster than B if

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

or

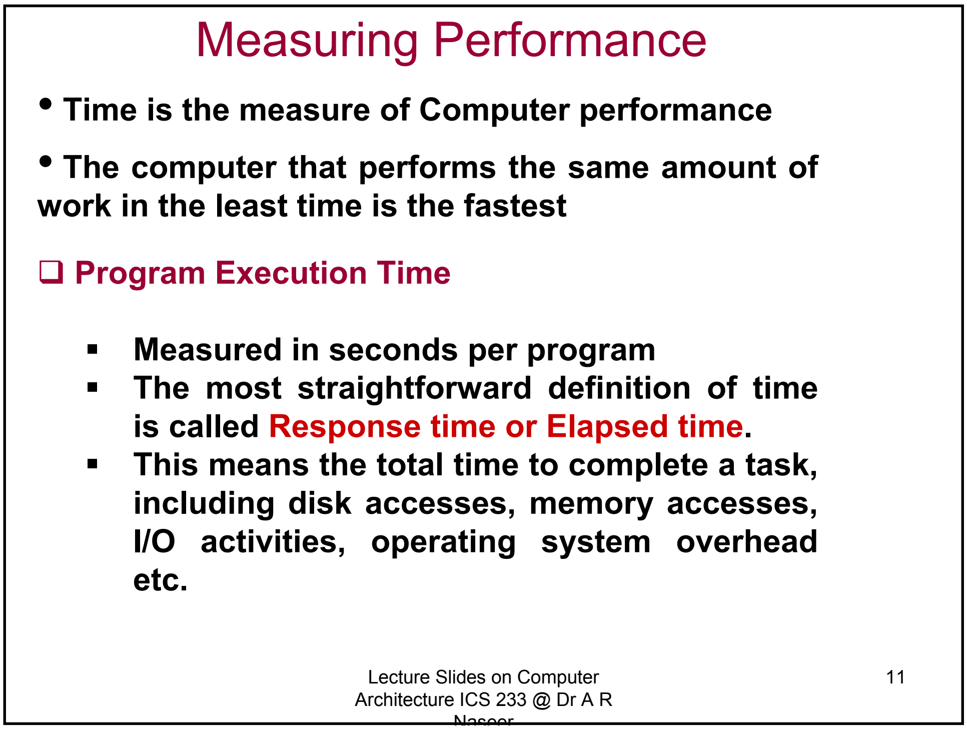
$$\frac{\text{Execution Time}_B}{\text{Execution Time}_A} = n$$

The performance ratio is $\frac{15}{10} = 1.5$

and <u>A is therefore 1.5 times faster than B.</u>

We could also say that Machine B is 1.5 times slower than machine A,

$$\frac{\text{Performance}_A}{\text{Performance}_B} = 1.5$$



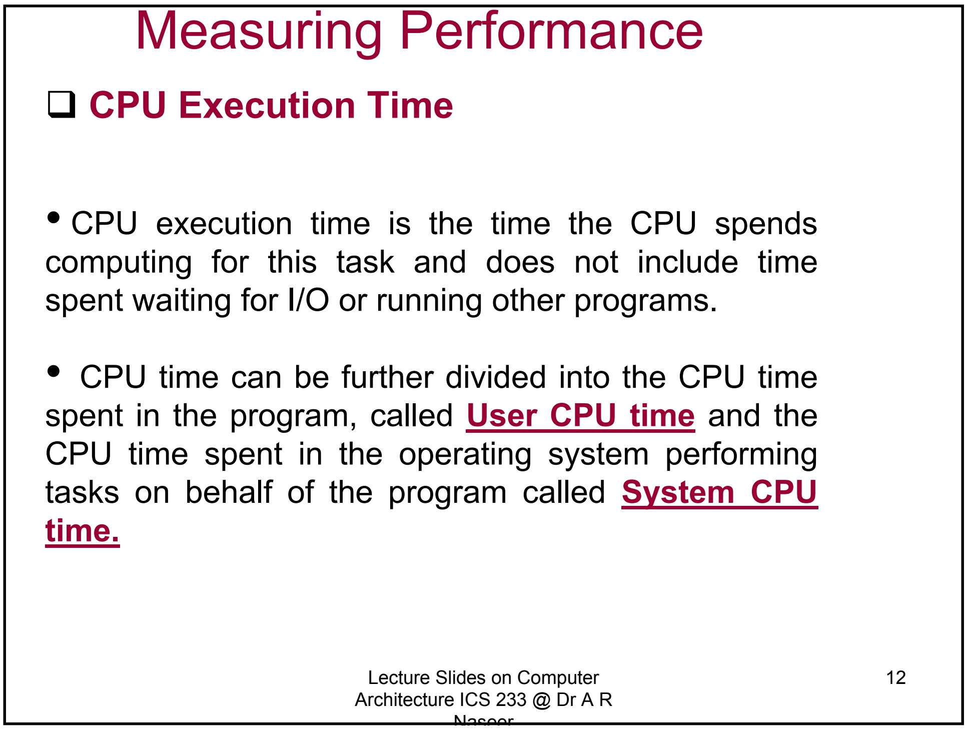
Measuring Performance

- **Time is the measure of Computer performance**

- **The computer that performs the same amount of work in the least time is the fastest**

❑ **Program Execution Time**

- **Measured in seconds per program**
- **The most straightforward definition of time is called Response time or Elapsed time.**
- **This means the total time to complete a task, including disk accesses, memory accesses, I/O activities, operating system overhead etc.**

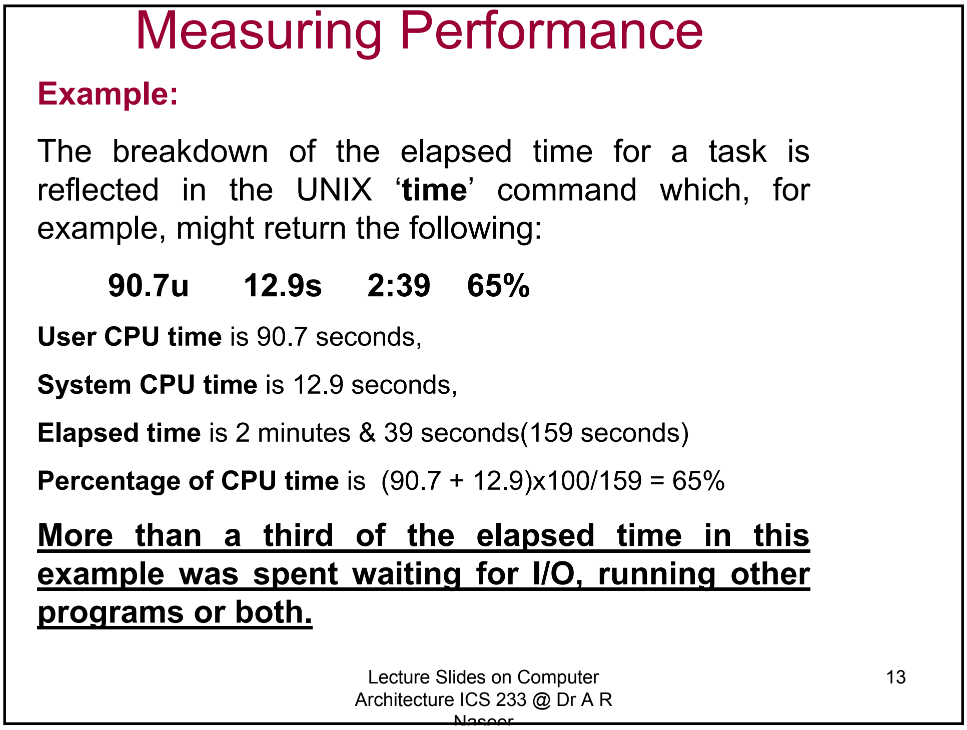


Measuring Performance

❑ **CPU Execution Time**

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

- CPU time can be further divided into the CPU time spent in the program, called **User CPU time** and the CPU time spent in the operating system performing tasks on behalf of the program called **System CPU time.**



Measuring Performance

Example:

The breakdown of the elapsed time for a task is reflected in the UNIX ‘**time**’ command which, for example, might return the following:

90.7u 12.9s 2:39 65%

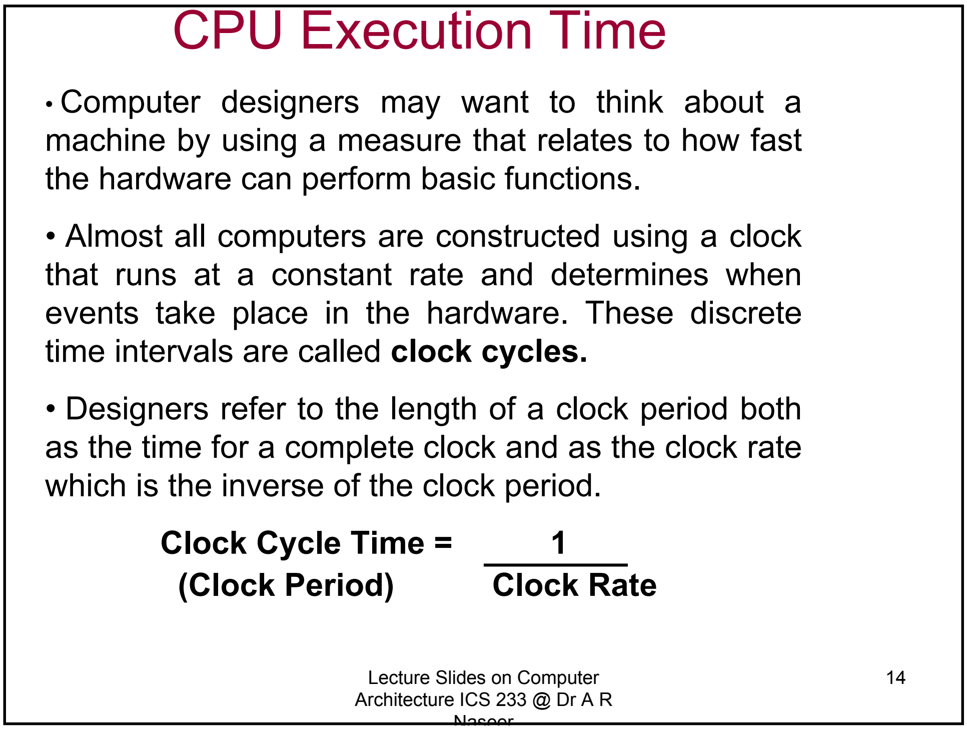
User CPU time is 90.7 seconds,

System CPU time is 12.9 seconds,

Elapsed time is 2 minutes & 39 seconds(159 seconds)

Percentage of CPU time is (90.7 + 12.9)x100/159 = 65%

<u>More than a third of the elapsed time in this example was spent waiting for I/O, running other programs or both.</u>



CPU Execution Time

- Computer designers may want to think about a machine by using a measure that relates to how fast the hardware can perform basic functions.
- Almost all computers are constructed using a clock that runs at a constant rate and determines when events take place in the hardware. These discrete time intervals are called **clock cycles.**
- Designers refer to the length of a clock period both as the time for a complete clock and as the clock rate which is the inverse of the clock period.

$$\textbf{Clock Cycle Time (Clock Period)} = \frac{\textbf{1}}{\textbf{Clock Rate}}$$

CPU Execution Time

❑ A simple formula that relates the most basic metrics (clock cycles & clock cycle time) to CPU Execution Time is

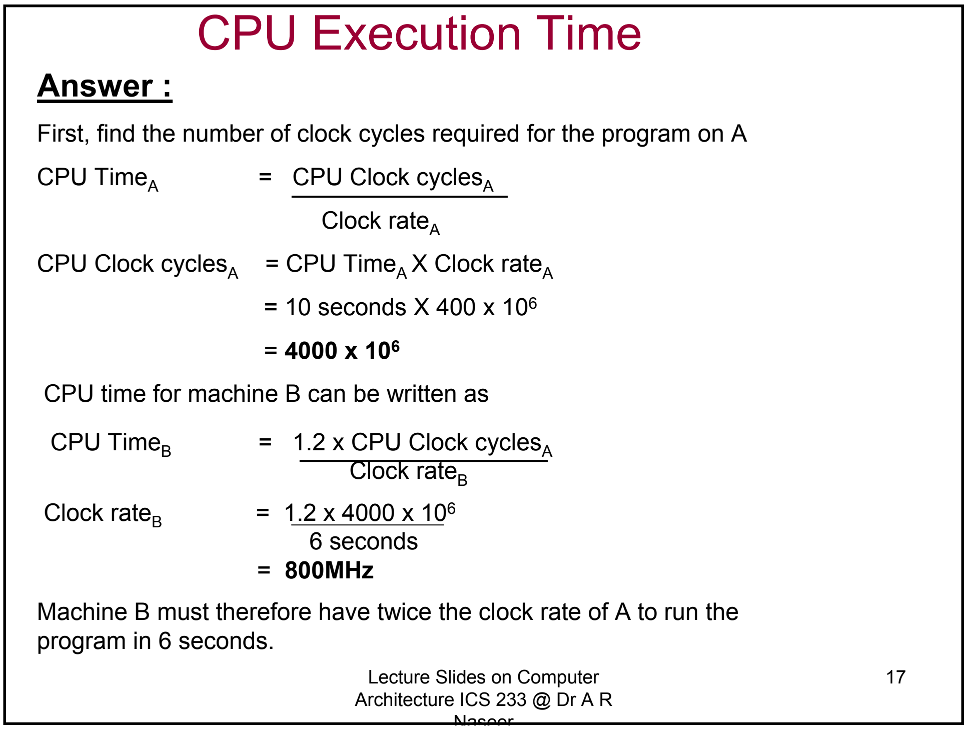
$$\text{CPU Execution Time} = \underset{\text{for a program}}{\text{CPU Clock cycles}} \times \text{Clock Cycle time}$$

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

CPU Execution Time

Question 3:

A program runs in 10 seconds on computer A, which has a 400 MHz clock. A computer designer wants to build a machine B that 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 machine B to require 1.2 times as many clock cycles as machine A for this program. What clock rate should we tell the designer to target?



CPU Execution Time

Answer :

First, find the number of clock cycles required for the program on A

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

$$\text{CPU Clock cycles}_A = \text{CPU Time}_A \text{ X Clock rate}_A$$

$$= 10 \text{ seconds X } 400 \times 10^6$$

$$= \mathbf{4000 \times 10^6}$$

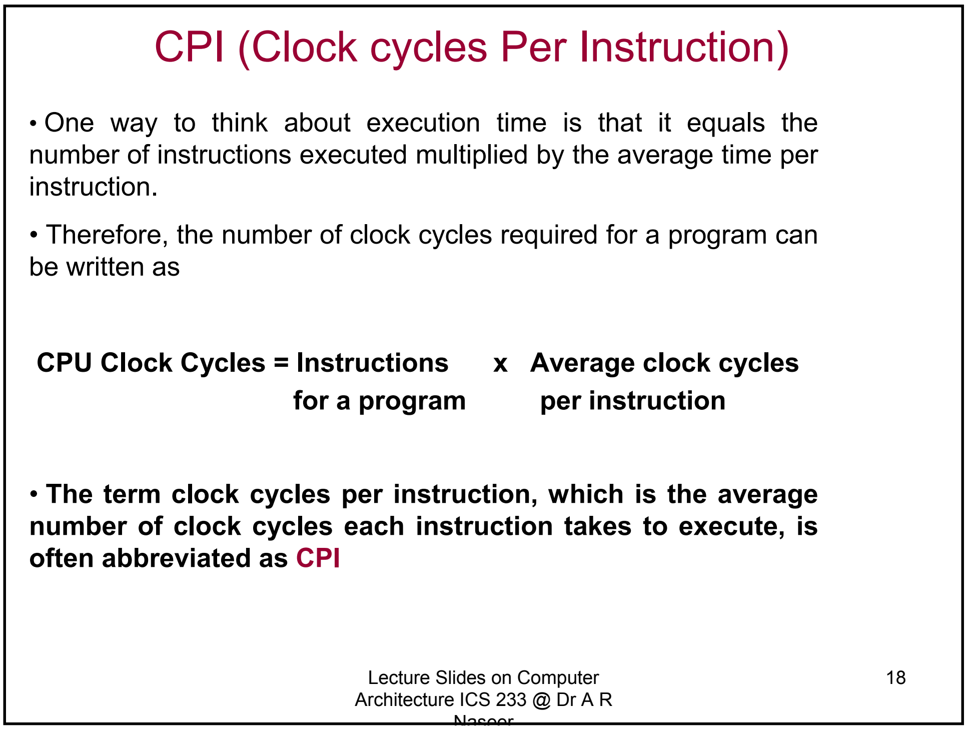
CPU time for machine B can be written as

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

$$\text{Clock rate}_B = \frac{1.2 \times 4000 \times 10^6}{6 \text{ seconds}}$$

$$= \mathbf{800MHz}$$

Machine B must therefore have twice the clock rate of A to run the program in 6 seconds.



CPI (Clock cycles Per Instruction)

- One way to think about execution time is that it equals the number of instructions executed multiplied by the average time per instruction.
- Therefore, the number of clock cycles required for a program can be written as

CPU Clock Cycles = Instructions for a program x Average clock cycles per instruction

- **The term clock cycles per instruction, which is the average number of clock cycles each instruction takes to execute, is often abbreviated as CPI**

CPI (Clock cycles Per Instruction)

CPU Clock Cycles = Instruction Count x CPI

$$\textbf{CPI} = \frac{\textbf{CPU Clock Cycles}}{\textbf{Instruction Count}}$$

- Since different instructions may take different amounts of time depending on what they do, CPI is an average of all the instructions executed in the program.
- CPU provides one way of comparing two different implementations of the same instruction set architecture, since the instruction count required for a program will, of course, be the same.

CPI (Clock cycles Per Instruction)

Question 4:

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?

CPI (Clock cycles Per Instruction)

We know that each computer executes the same number of instructions for the program, let us call this number N.

First, find the number of processor clock cycles for each computer:

CPU Clock Cycles$_A$ = N x 2.0

CPU Clock Cycles$_B$ = N x 1.2

Now we can compute the CPU time for each computer:

CPU Time$_A$ = CPU Clock Cycles$_A$ x Clock Cycle Time$_A$

= N x 2.0 x 250 ps

= 500 x N ps

Similarly for B,
CPU Time$_B$ = CPU Clock Cycles$_B$ x Clock Cycle Time$_B$
= N x 1.2 x 500 ps
= 600 x N ps
Clearly Computer A is faster.

The amount faster is given by the ratio of the execution times:

$$\frac{\text{Performance}_A}{\text{Performance}_B} = \frac{\text{Execution Time}_B}{\text{Execution Time}_A} = \frac{600 \times N \text{ ps}}{500 \times N \text{ ps}} = 1.2$$

We can conclude that computer A is 1.2 times faster than B for this program.

Basic Performance Equation

- The basic performance equation can be written in terms of Instruction Count (the number of instructions executed by the program), CPI and Clock cycle time as follows:

CPU time = Instruction Count x CPI x Clock cycle time

$$\textbf{CPU time} = \frac{\textbf{Instruction Count x CPI}}{\textbf{Clock Rate}}$$

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

- Instruction Count (IC): Number of instructions/program
- Cycles per instruction (CPI)
  – Sometimes the reciprocal is used: Instructions per cycle (IPC)
- The number of seconds per cycle is the clock period
  – clock rate is the multiplicative inverse of the clock period

Aspects of CPU Performance

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

- CPU performance is dependent upon three characteristics :

  - Clock cycle time (clock rate), Cycles per instruction, and Instruction count

It is difficult to change one parameter in complete isolation from others because the basic technologies involved in changing each characteristics are interdependent

	Instruction Count	CPI	Clock rate
Program	X		
Compiler	X		
Instruction Set Architecture	X	X	
Organization		X	X
Hardware Technology			X

Basic Performance Equation

• Sometimes it is possible to compute the CPU clock cycles by looking at the different types of instructions and using their individual clock cycle counts.

In such cases, the following formula is useful:

$$\textbf{CPU Clock Cycles} = \sum_{i=1}^{n} (\text{CPI}_i \text{ x } C_i)$$

where C_i is the count of the number of instructions of class i executed,

CPI_i is the average number of cycles per instruction for that instruction class, and

n is the number of instruction classes.

• Remember that overall CPI for a program will depend on both the number of cycles for each instruction type and the frequency of each instruction type in the program execution.

Performance

Question 5:

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

Instruction Class	CPI for this instruction Class
A	1
B	2
C	3

For a particular high-level-language statement, the compiler writer is considering two code sequences that require the following instruction counts:

Code Sequence	Instruction Counts for Instruction Class		
	A	B	C
1	2	1	2
2	4	1	1

Which code sequence executes most instructions? Which will be faster? What is the CPI for each code sequence

Answer:

Code Sequence 1 executes 2+1+2 = 5 instructions

Code sequence 2 executes 4+1+1 = 6 instructions

So 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:

$$\text{CPU Clock Cycles} = \sum_{i=1}^{n} (\text{CPI}_i \times C_i)$$

This yields

CPU clock cycles$_1$ = (1x2) + (2x1) + (3x2) = 2+2+6 =10 cycles

CPU clock cycles$_2$ = (1x4) + (2x1) + (3x1) = 4+2+3 = 9 cycles

So code sequence 2 is faster, even though it actually executes one extra instruction.

Answer (continued):

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

$$\text{CPI} = \frac{\text{CPU Clock cycles}}{\text{Instruction Count}}$$

$$\text{CPI}_1 = \frac{\text{CPU Clock cycles}_1}{\text{Instruction Count}_1} = \frac{10}{5} = 2$$

$$\text{CPI}_2 = \frac{\text{CPU Clock cycles}_2}{\text{Instruction Count}_2} = \frac{9}{6} = 1.5$$

Quantitative Principles of Computer Design

❑ **Make the Common Case Fast**

➢Most important and pervasive principle of Computer design

➢ **One of the principles behind RISC: Reduced Instruction Set Computers**

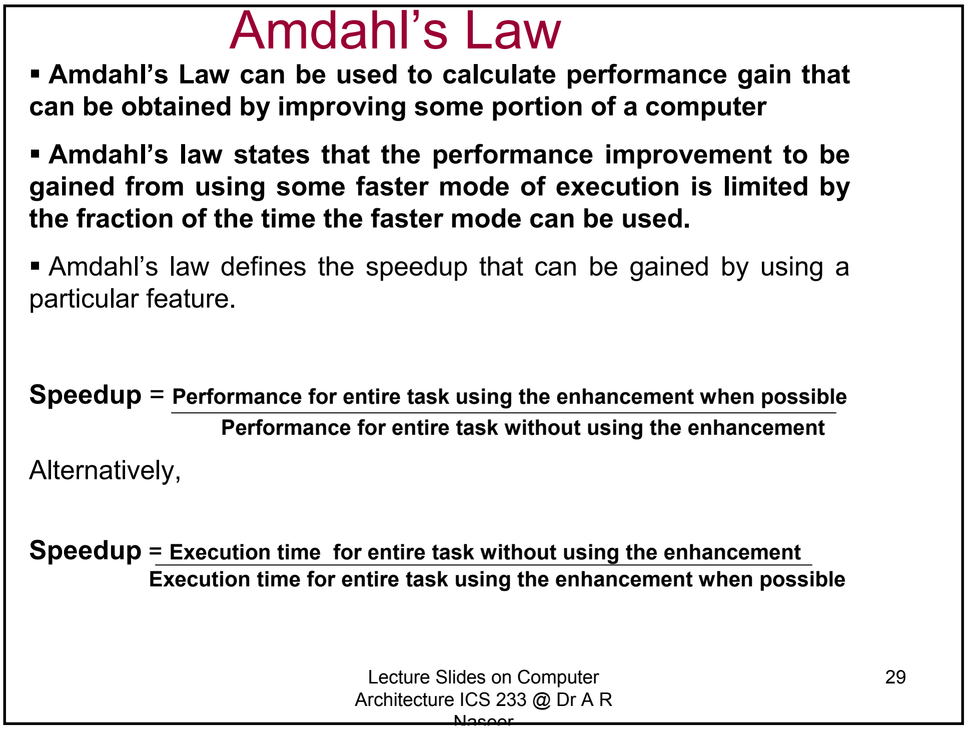
- **Identify most frequently-used instructions**
  - **Implement them in hardware**
- **Emulate other instructions in software**

➢ **Pretty much every technique used in current-day microprocessors**

■ **Is there a way of quantifying the gains one is likely to see by improving some portion of the design?**

– What is the best one can hope to do?

■ **A fundamental law, called Amdahl's Law can be used to quantify this principle.**



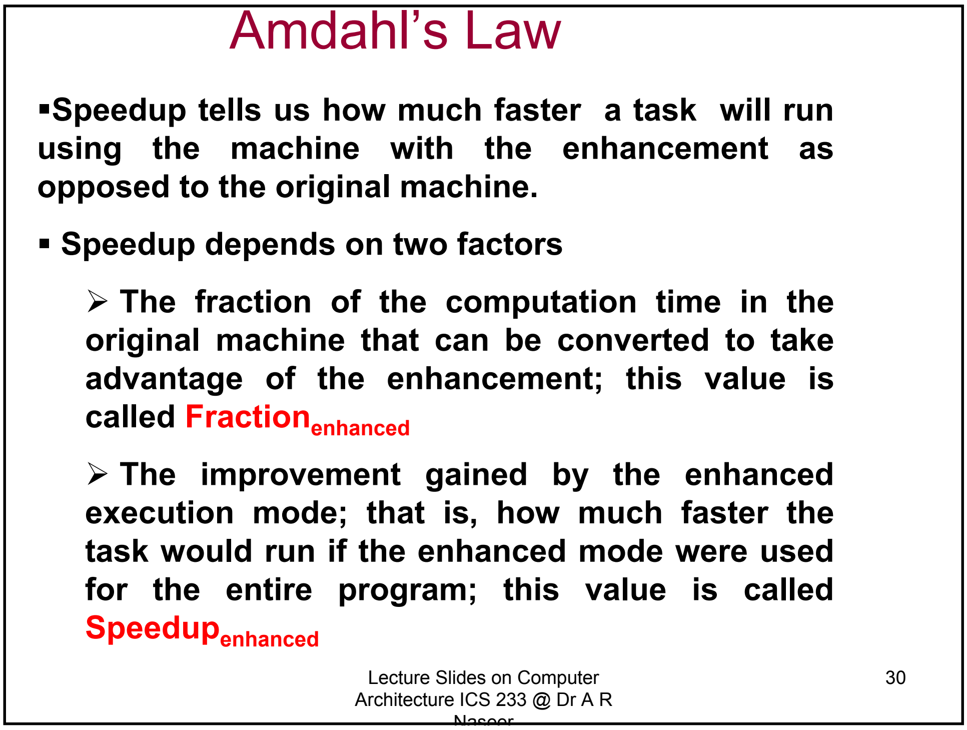
Amdahl’s Law

- **Amdahl’s Law can be used to calculate performance gain that can be obtained by improving some portion of a computer**
- **Amdahl’s law states that the performance improvement to be gained from using some faster mode of execution is limited by the fraction of the time the faster mode can be used.**
- Amdahl’s law defines the speedup that can be gained by using a particular feature.

$$\textbf{Speedup} = \frac{\text{Performance for entire task using the enhancement when possible}}{\text{Performance for entire task without using the enhancement}}$$

Alternatively,

$$\textbf{Speedup} = \frac{\text{Execution time for entire task without using the enhancement}}{\text{Execution time for entire task using the enhancement when possible}}$$



Amdahl’s Law

- **Speedup tells us how much faster a task will run using the machine with the enhancement as opposed to the original machine.**
- **Speedup depends on two factors**
  - **The fraction of the computation time in the original machine that can be converted to take advantage of the enhancement; this value is called $\textbf{Fraction}_{\textbf{enhanced}}$**
  - **The improvement gained by the enhanced execution mode; that is, how much faster the task would run if the enhanced mode were used for the entire program; this value is called $\textbf{Speedup}_{\textbf{enhanced}}$**

Amdahl’s Law

▪The execution time using the original machine with the enhanced mode will be the time spent using the unenhanced portion of the machine plus the time spent using the enhancement.

$$\text{Execution time}_{new} = \text{Executiontime}_{old} \times \left[(1 - \text{Fraction}_{enhanced}) + \frac{\text{Fraction}_{enhanced}}{\text{Speedup}_{enhanced}} \right]$$

The overall speedup is the ratio of the execution times :

$$\text{Speedup}_{overall} = \frac{\text{Execution time}_{old}}{\text{Execution time}_{new}} = \frac{1}{(1 - \text{Fraction}_{enhanced}) + \frac{\text{Fraction}_{enhanced}}{\text{Speedup}_{enhanced}}}$$

Amdahl’s Law

Question :

Suppose that we are considering an enhancement to the processor of a server system used for Web Serving. The new CPU is 10 times faster on computation in the Web serving application than the original processor. Assuming that the original CPU 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?

Answer :

$\text{Fraction}_{enhanced} = 0.4$

$\text{Speedup}_{enhanced} = 10$

$$\text{Speedup}_{overall} = \frac{1}{(1 - \text{Fraction}_{enhanced}) + \frac{\text{Fraction}_{enhanced}}{\text{Speedup}_{enhanced}}}$$

$$= \frac{1}{0.6 + \frac{0.4}{10}} = \frac{1}{0.64} = 1.56$$

Amdahl's Law

- Expresses the law of diminishing returns

- **The incremental improvement in speedup gained by an additional improvement in the performance of just a portion of the computation diminishes as improvements are added.**

Amdahl's Law

Question :

A common transformation required in graphics engines is square root. Implementations of floating-point (FP) square root vary significantly in performance, especially among processors designed for graphics.

i) Suppose FP square root (FPSQR) is responsible for 20% of the execution time of a critical graphics benchmark. One proposal is to enhance the FPSQR hardware and speed up this operation by a factor of 10.

ii) The other alternative is just to try to make all FP instructions in the graphics processor run faster by a factor of 1.6; FP instructions are responsible for a total of 50% of the execution time for the application. The design team believes that they can make all FP instructions run 1.6 times faster with the same effort as required for the fast square root.

Compare these two design alternatives.

Amdahl’s Law

Answer :

We can compare these two alternatives by comparing the speedups:

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

$$\text{Speedup}_{\text{FP}} = \frac{1}{(1- 0.5) + \frac{0.5}{1.6}} = \frac{1}{0.8125} = 1.23$$

- **Improving the performance of the FP operations overall is slightly better because of the higher frequency.**