Performance

Performance

- The most important measure : “Speed “
  - how quickly it can execute programs
- Depends on
  - instruction set
  - Hardware & its implementation technology
  - Software
  - Compiler
- Make intelligent choices
- See through the marketing hype

Performance

- **Technology**
  - **Very Large Scale Integration** (VLSI)
    - Fabricate processor on a single chip
    - Increases the speed of execution of machine instructions
  - **Smaller transistors switch faster**
    - The speed of switching between the 0 and 1 states in logic circuits is largely determined by the size of the transistors that implement the circuits.

Performance

- **Technology**
  - Advances in fabrication technology have reduced transistor sizes dramatically
  - This has two advantages:
    - instructions can be executed faster
    - more transistors can be placed on a chip
      - leading to more logic functionality
      - leading to more memory storage capacity

Performance

- **Parallelism**
  - Increase Performance
  - Parallelism can be implemented on many different levels
  - **Instruction-level Parallelism**
    - overlap the execution of the steps of successive instructions, total execution time will be reduced.
    - For example, the next instruction could be fetched from memory at the same time that an arithmetic operation is being performed on the register operands of the current instruction.
    - This form of parallelism is called *pipelining.*

Performance

- **Multicore Processors**
  - Multiple processing units can be fabricated on a single chip
    - *dual-core, quad-core, and octo-core*

Performance

- **Multiprocessors**
  - Many processors containing multiple cores
  - *These systems either execute a number* of different application tasks in parallel, or they execute subtasks of a single large task in parallel
  - All processors usually have access to all of the memory in such systems, and the term ***shared-memory multiprocessor*** *is often used to make this clear.*
  - *The high* performance of these systems comes with much higher complexity and cost,
    - This is due to multiple processors and memory units, along with more complex interconnection Networks.

Performance

- **Multiprocessors**
  - Use interconnected group of computers to achieve high total computational power
  - The computers normally have access only to their own memory units
  - When the tasks they are executing need to share data, they do so by exchanging *messages over a communication network*
  - *This property* distinguishes them from shared-memory multiprocessors, leading to the name ***messagepassing multicomputers***

Response Time and Throughput

- Response time / **execution time**
  - How long it takes to do a task
  - the time between the start and completion of a task
  - individual computer user are interested in this
  - personal mobile devices
- Throughput/ **bandwidth**
  - Total work done per unit time
    - e.g., tasks/transactions/… per hour
  - Data center managers are often interested in this
  - servers

Response Time and Throughput

- How are response time and throughput affected by
  - Replacing the processor with a faster version?
    - Decreasing response time almost always improves throughput
    - So both response time and throughput are improved
  - Adding additional processors to a system that uses multiple processors for separate tasks—for example, searching the web
    - no one task gets work done faster, so only throughput increases
- Note: increasing the throughput could also improve response time, if it reduce the waiting time in the queue

Relative Performance

- To maximize performance, we want to minimize response time or execution time

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

•Execution time on Y is longer than that on X,

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

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

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

Relative Performance

- "X is *n* time faster than Y"

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

- If X is *n times as fast as Y, then the execution time on Y is n times as long as it is* on X

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

Relative Performance

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?

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.

In the above example, we could also say that computer B is 1.5 times *slower than* computer A, since

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

means that

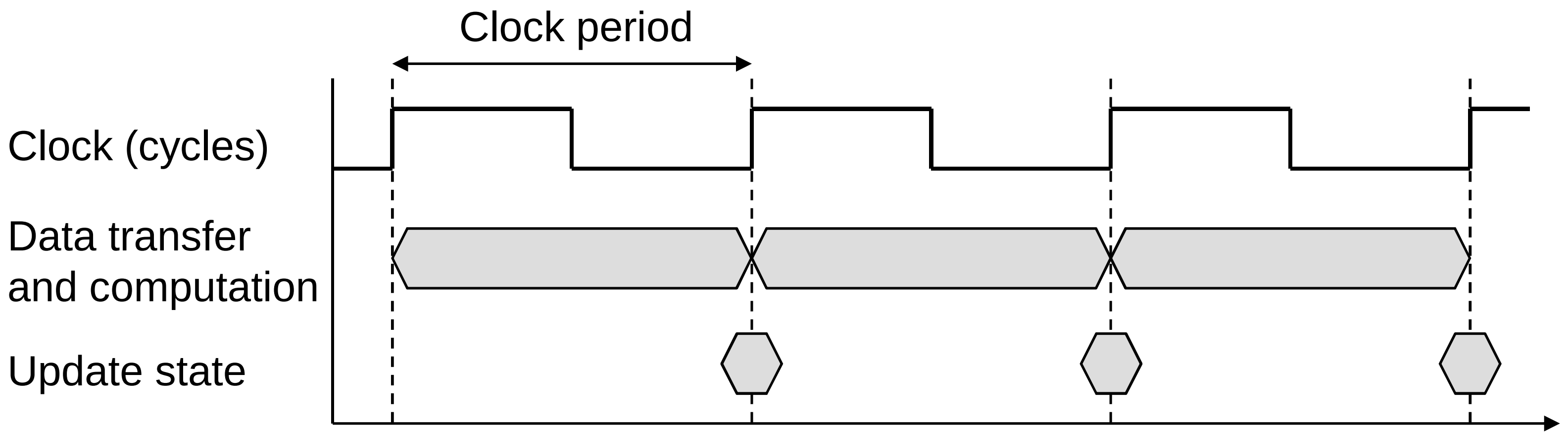
$$\frac{\text{Performance}_A}{1.5} = \text{Performance}_B$$

Measuring Execution Time

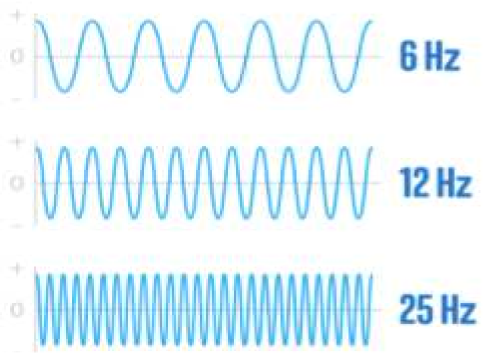
- Elapsed time
  - Total response time, including all aspects
    - Processing, I/O, OS overhead, idle time
  - Determines system performance
- CPU time
  - Time spent processing a given job
    - Discounts I/O time, other jobs’ shares
  - Comprises user CPU time and system CPU time
  - Different programs are affected differently by CPU and system performance

CPU Clocking

- Operation of digital hardware governed by a constant-rate clock



- Clock period: duration of a clock cycle
  - e.g., 250ps = 0.25ns = 250×10^{-12}s
- Clock frequency (rate): cycles per second
  - e.g., 4.0GHz = 4000MHz = 4.0×10^{9}Hz



CPU Time

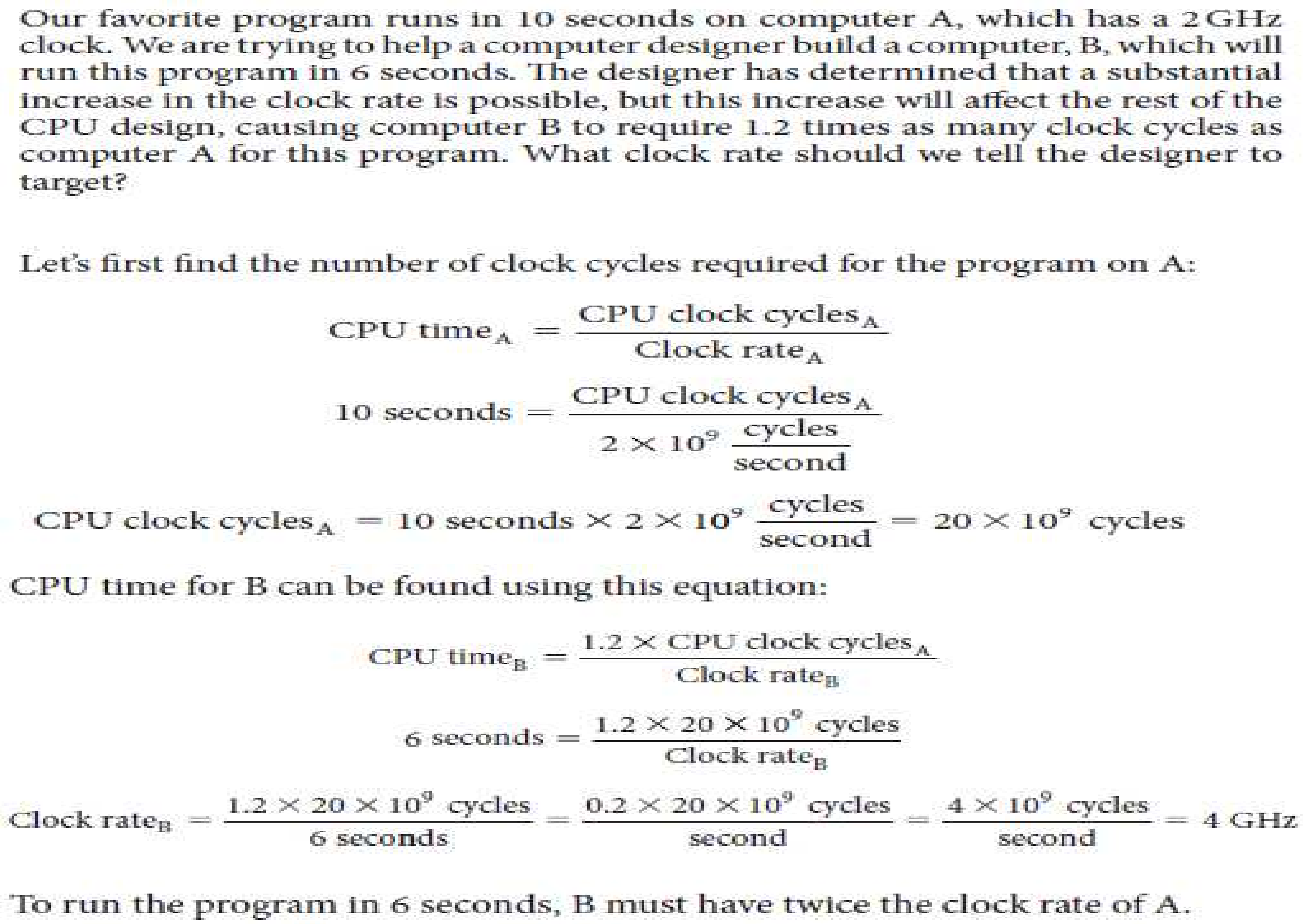
- Performance improved by
  - Reducing number of clock cycles
  - Increasing clock rate

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

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

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

CPU Time 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:

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

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

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

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

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

$$\text{CPU time} = \text{Instruction count} \times \text{CPI} \times \text{Clock cycle time}$$

$$\text{CPU time} = \frac{\text{Instruction count} \times \text{CPI}}{\text{Clock rate}}$$

- Instruction Count for a program
  - Determined by program, ISA and compiler
- Average cycles per instruction
  - Determined by CPU hardware
  - If different instructions have different CPI
    - Average CPI affected by instruction mix

Instruction Count and CPI

MOV	AX,90	→ 2 Cycles
MOV	BX,50	→ 2 Cycles
ADD	AX,BX	→ 1 Cycle
STA	AX,4500	→ 3 Cycles
HLT		→ 1 Cycle

- CPI = 9 /5 =1.8

CPI 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, 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:

$$\begin{aligned}\text{CPU clock cycles}_A &= I \times 2.0\\ \text{CPU clock cycles}_B &= I \times 1.2\end{aligned}$$

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.

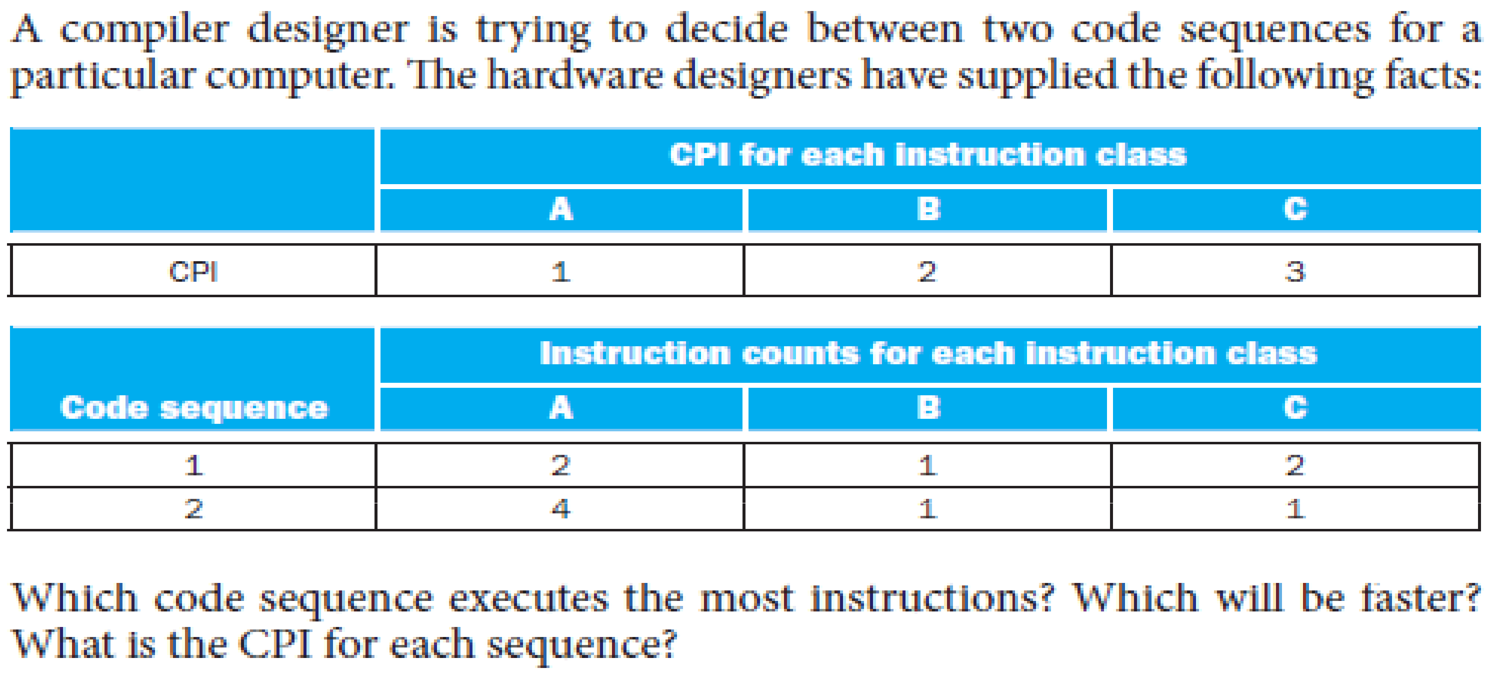
CPI in More Detail

- If different instruction classes take different numbers of cycles

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

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

CPI Example



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

	CPI for each instruction class		
	A	B	C
CPI	1	2	3

Code sequence	Instruction counts for each instruction class		
	A	B	C
1	2	1	2
2	4	1	1

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

CPI Example

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

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

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

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

Performance Summary

The BIG Picture

$$\text{Time} = \text{Seconds/Program} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock cycle}}$$

$$\text{CPU clock cycles} = \text{Instructions for a program} \times \begin{array}{c}\text{Average clock cycles}\\ \text{per instruction}\end{array}$$

$$\text{CPU time} = \text{Instruction count} \times \text{CPI} \times \text{Clock cycle time}$$

$$\text{CPU time} = \frac{\text{Instruction count} \times \text{CPI}}{\text{Clock rate}}$$

Components of performance	Units of measure
CPU execution time for a program	Seconds for the program
Instruction count	Instructions executed for the program
Clock cycles per instruction (CPI)	Average number of clock cycles per instruction
Clock cycle time	Seconds per clock cycle

Performance Summary

- Performance depends on
  - Algorithm: affects IC, possibly CPI
  - Programming language: affects IC, CPI
  - Compiler: affects IC, CPI
  - Instruction set architecture: affects IC, CPI, T_c

Amdahl's Law

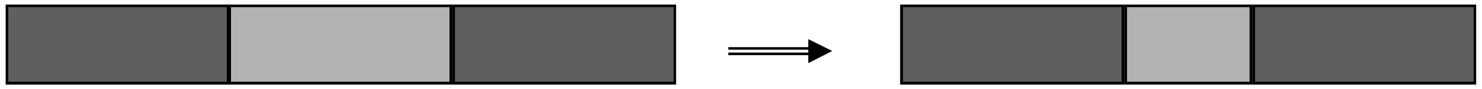
Execution Time After Improvement =

$$\text{Execution Time Unaffected} + \frac{\text{Execution Time Affected}}{\text{Amount of Improvement}}$$

Amdahl's Law

Speedup due to enhancement E:

$$\textbf{Speedup(E)} = \frac{\textbf{ExTime w/o E}}{\textbf{ExTime w/ E}} = \frac{\textbf{Performance w/ E}}{\textbf{Performance w/o E}}$$



Suppose that enhancement E accelerates a fraction F of the task by a factor S, and the remainder of the task is unaffected

Amdahl's Law

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

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