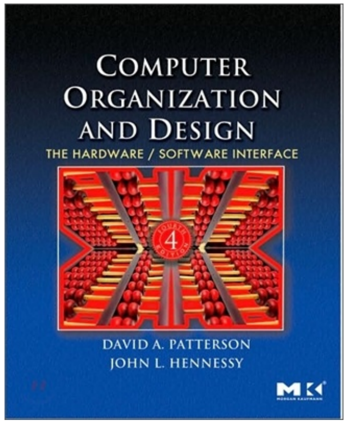
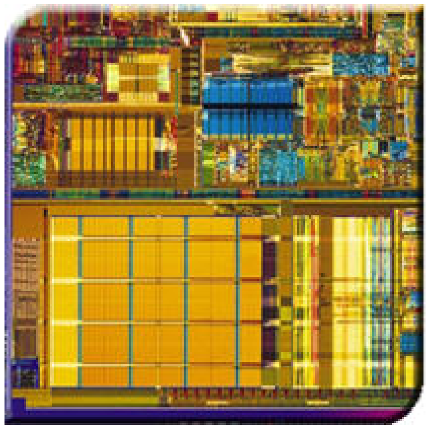
Computer Architecture

Lecture 3 Performance





Prof. Jongmyon Kim

울산대학교
UNIVERSITY OF ULSAN

Performance

- Measure, Report, and Summarize
- Make intelligent choices
- See through the marketing type
- Key to understanding underlying organizational motivation

Why is some hardware better than others for different programs?

What factors of system performance are hardware related?
(e.g., Do we need a new machine, or a new operating system?)

How does the machine's instruction set affect performance?

Which of these airplanes has the best performance?

Airplane	Passengers	Range (mi)	Speed (mph)
Boeing 737-100	101	630	598
Boeing 747	470	4150	610
BAC/Sud Concorde	132	4000	1350
Douglas DC-8-50	146	8720	544

- How much faster is the Concorde compared to the 747?
- How much bigger is the 747 than the Douglas DC-8?

Computer Performance: TIME, TIME, TIME

- Response Time (latency)
  - — How long does it take for my job to run?
  - — How long does it take to execute a job?
  - — How long must I wait for the database query?
- Throughput
  - — How many jobs can the machine run at once?
  - — What is the average execution rate?
  - — How much work is getting done?
- *If we upgrade a machine with a new processor what do we increase?*
- *If we add a new machine to the lab what do we increase?*

Execution Time

- Elapsed Time
  - counts everything *(disk and memory accesses, I/O , etc.)*
  - a useful number, but often not good for comparison purposes
- CPU time
  - doesn't count I/O or time spent running other programs
  - can be broken up into system time, and user time
- Our focus: user CPU time
  - time spent executing the lines of code that are "in" our program

Book's Definition of Performance

- For some program running on machine X,

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

- "X is n times faster than Y"

  $$\text{Performance}_X \;/\; \text{Performance}_Y = n$$

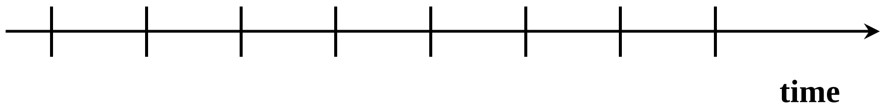
- Problem:
  - machine A runs a program in 20 seconds
  - machine B runs the same program in 25 seconds

Clock Cycles

- Instead of reporting execution time in seconds, we often use cycles

$$\frac{\text{seconds}}{\text{program}} = \frac{\text{cycles}}{\text{program}} \times \frac{\text{seconds}}{\text{cycle}}$$

- Clock “ticks” indicate when to start activities (one abstraction):



- cycle time = time between ticks = seconds per cycle
- clock rate (frequency) = cycles per second (1 Hz = 1 cycle/sec)

A 200 MHz clock has a ________________ cycle time

How to Improve Performance

$$\frac{\text{seconds}}{\text{program}} = \frac{\text{cycles}}{\text{program}} \times \frac{\text{seconds}}{\text{cycle}}$$

So, to improve performance (everything else being equal) you can either

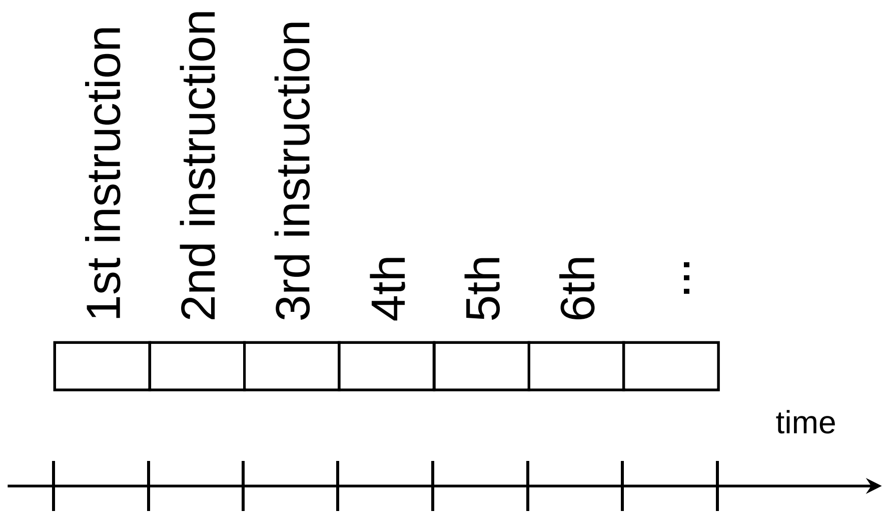
________ the # of required cycles for a program, or

________ the clock cycle time or, said another way,

________ the clock rate.

How many cycles are required for a program?

- Could assume that # of cycles = # of instructions

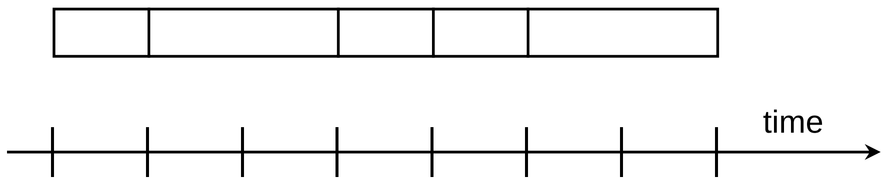


This assumption is incorrect,

different instructions take different amounts of time on different machines.

Why? hint: remember that these are machine instructions, not lines of C code

Different numbers of cycles for different instructions



- Multiplication takes more time than addition
- Floating point operations take longer than integer ones
- Accessing memory takes (in general) more time than accessing registers

- *Important point: changing the cycle time often changes the number of cycles required for various instructions (more later)*

Example

- Our favorite program runs in 10 seconds on computer A, which has a 400 MHz clock. We are trying to help a computer designer build a new machine B, that will run this program in 6 seconds. The designer can use new (or perhaps more expensive) technology to substantially increase the clock rate, but has informed us that 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 the same program. What clock rate should we tell the designer to target?"

- Don't Panic, can easily work this out from basic principles

$$\text{CPU time}_A = \frac{\text{CPU clock cycle}_A}{\text{Clock rate}_A} \rightarrow 10\sec = \frac{\text{CPU clock cycle}_A}{400\times 10^6 \frac{cycles}{second}}$$

$$\text{CPU clock cycle}_A = 10\sec \times 400\times 10^6 \frac{cycles}{second} = 4000\times 10^6\, cycles$$

$$\text{CPU time}_B = \frac{1.2\times \text{CPU clock cycle}_A}{\text{Clock rate}_B} \rightarrow 6\sec = \frac{1.2\times 4000\times 10^6\, cycle}{\text{Clock rate}_B}$$

$$\text{Clock rate}_B = \frac{1.2\times 4000\times 10^6\, cycles}{6\, seconds} = \frac{800\times 10^6\, cycles}{second} = 800 MHz$$

Now that we understand cycles

- A given program will require
  - some number of instructions (machine instructions)
  - some number of cycles
  - some number of seconds
- We have a vocabulary that relates these quantities:
  - cycle time (seconds per cycle)
  - clock rate (cycles per second)
  - CPI (cycles per instruction)

    a floating point intensive application might have a higher CPI
  - MIPS (millions of instructions per second)

    this would be higher for a program using simple instructions

Performance

- Performance is determined by execution time
- Do any of the other variables equal performance?
  - # of cycles to execute program?
  - # of instructions in program?
  - # of cycles per second? (frequency)
  - average # of cycles per instruction (CPI)?
  - average # of instructions per second?

- Common pitfall: thinking one of the variables is indicative of performance when it really isn't.

CPI Example

- Suppose we have two implementations of the same instruction set architecture (ISA).

  For some program,

  Machine A has a clock cycle time of 1 ns. and a CPI of 2.0
  Machine B has a clock cycle time of 2 ns. and a CPI of 1.2

  What machine is faster for this program, and by how much?

$$\text{CPU clock cycle}_A = I \times 2.0 \qquad \text{CPU time}_A = \text{CPU clock cycle}_A \times \text{CPU cycle time}_A = \text{I} \times 2.0 \times 1\text{ns}$$

$$\text{CPU clock cycle}_B = I \times 1.2 \qquad \text{CPU time}_B = \text{I} \times 1.2 \times 2\text{ns}$$

$$\frac{\text{CPU performance}_A}{\text{CPU performance}_B} = \frac{\text{Execution time}_B}{\text{Execution time}_A} = \frac{2.4 \times \text{Ins}}{2 \times \text{Ins}} = 1.2$$

- *If two machines have the same ISA which of our quantities (e.g., clock rate, CPI, execution time, # of instructions, MIPS) will always be identical?*

of Instructions Example

- A compiler designer is trying to decide between two code sequences for a particular machine. Based on the hardware implementation, there are three different classes of instructions: Class A, Class B, and Class C, and they require one, two, and three cycles (respectively).

  The first code sequence has 5 instructions: 2 of A, 1 of B, and 2 of C
  The second sequence has 6 instructions: 4 of A, 1 of B, and 1 of C.

  Which sequence will be faster? How much?
  What is the CPI for each sequence?

$$\text{CPU clock cycles} = \sum_{i=1}^{n} (\mathrm{I}_i \times C_i)$$

$$\text{CPU clock cycles}_1 = (2\times 1) + (1\times 2) + (2\times 3) = 10\text{cycles}$$

$$\text{CPU clock cycles}_2 = (4\times 1) + (1\times 2) + (1\times 3) = 9\text{cycles}$$

$$\mathrm{CPI}_1 = \frac{\text{CPU clock cycles}_1}{\text{Instruction count}_1} = {}^{10}\!/_{5} = 2$$

$$\mathrm{CPI}_2 = \frac{\text{CPU clock cycles}_2}{\text{Instruction count}_2} = {}^{9}\!/_{6} = 1.5$$

MIPS example

- Two different compilers are being tested for a 100 MHz. machine with three different classes of instructions: Class A, Class B, and Class C, which require one, two, and three cycles (respectively). Both compilers are used to produce code for a large piece of software.

  The first compiler's code uses 5 billion Class A instructions, 1 billion Class B instructions, and 1 billion Class C instructions.

  The second compiler's code uses 10 billion Class A instructions, 1 billion Class B instructions, and 1 billion Class C instructions.

- Which sequence will be faster according to MIPS?
- Which sequence will be faster according to execution time?

$$\text{CPU clock cycles}_1 = (5\times1 + 1\times2 + 1\times3)\times10^9 = 10\times10^9$$

$$\text{CPU clock cycles}_2 = (10\times1 + 1\times2 + 1\times3)\times10^9 = 15\times10^9$$

$$\text{Execution time}_1 = \frac{10\times10^9}{100\times10^6} = 100\text{ seconds} \qquad \text{MIPS}_1 = \frac{\text{Instruction Count}}{\text{Execution time}\times10^6} = \frac{(5+1+1)\times10^9}{100\times10^6} = 70$$

$$\text{Execution time}_2 = \frac{15\times10^9}{100\times10^6} = 150\text{ seconds} \qquad \text{MIPS}_2 = \frac{\text{Instruction Count}}{\text{Execution time}\times10^6} = \frac{(10+1+1)\times10^9}{150\times10^6} = 80$$

Benchmarks

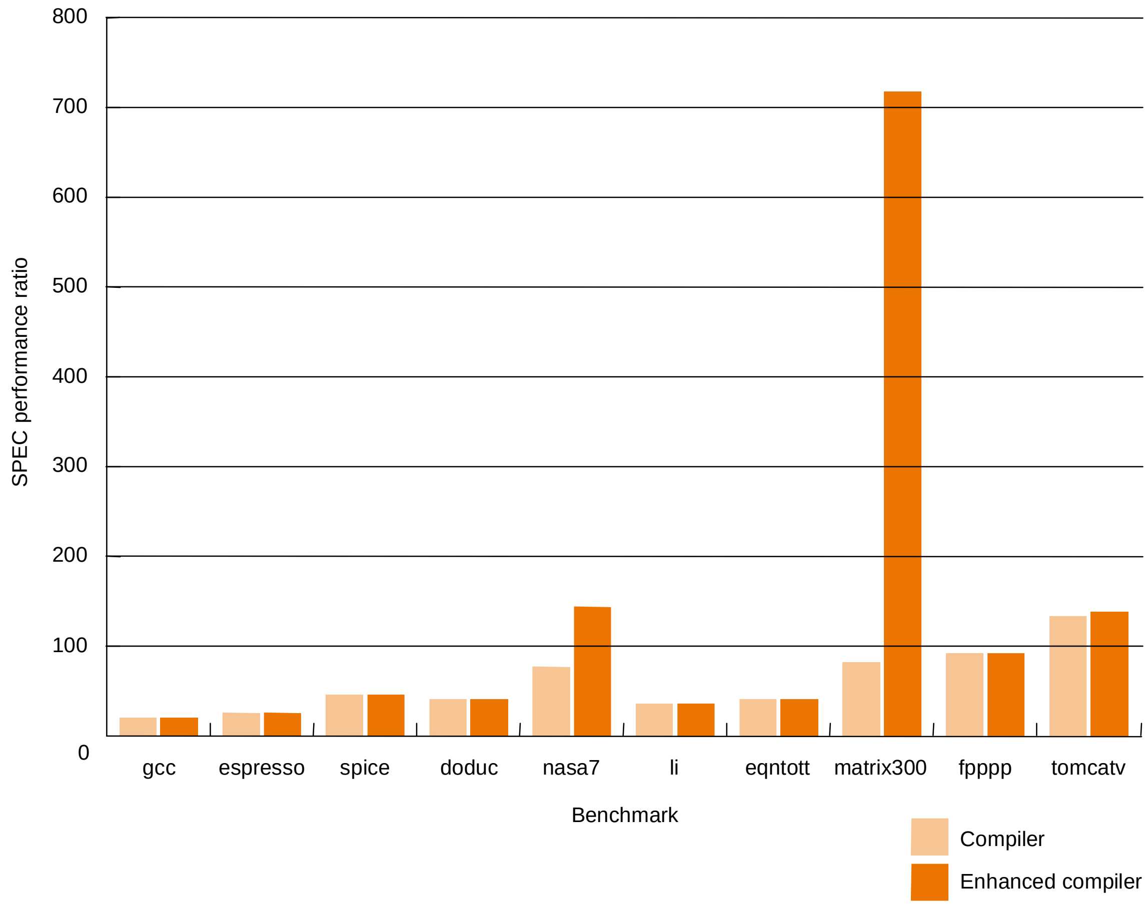
- Small benchmarks
  - nice for architects and designers
  - easy to standardize
  - can be abused
- Performance best determined by running a real application
  - Use programs typical of expected workload
  - Typical of representative class of applications
    - Media: MP3 decoder, iTune,
    - Games: Quake, Unreal, Call of Duty
    - Editing: Clone DVD, Photoshop, 3D Studio
- Synthetic benchmarks
  - Collection of common applications (i.e. Windows)
  - 3DMark, PC Mark, SiSoft Sandra,
- SPEC (System Performance Evaluation Cooperative)
  - companies have agreed on a set of real program and inputs
  - can still be abused (Intel's "other" bug)
  - valuable indicator of performance (and compiler technology)
  - Separate into integer benchmark and floating-point benchmark
  - Current version: SPEC2000

SPEC CPU2000 Benchmark

Benchmark	Source	Application
164.gzip	C	Compression
175.vpr	C	FPGA circuit placement and routing
176.gcc	C	C programming language compiler
181.mcf	C	Combinatorial optimization
186.crafty	C	Game playing: chess
197.parser	C	Word processing
252.eon	C++	Computer visualization
253.perlbmk	C	PERL programming language
254.gap	C	Group theory, interpreter
255.vortex	C	Object-oriented database
256.bzip2	C	Compression
300.twolf	C	Place and route simulator
168.wupwise	FORTRAN 77	Quantum chromodynamics
171.swim	FORTRAN 77	Shallow water modeling
172.mgrid	FORTRAN 77	Multi-grid solver
173.applu	FORTRAN 77	Parabolic/Elliptic PDEs
177.mesa	C	3D OpenGL graphics library
178.galgel	FORTRAN 90	Computational fluid dynamics
179.art	C	Image recognition / Neural networks
183.equake	C	Seismic wave propagation simulation
187.facerec	FORTRAN 90	Image processing: face recognition
188.ammp	C	Computational chemistry
189.lucas	FORTRAN 90	Number theory / primality testing
191.fma3d	FORTRAN 90	Finite-element crash simulation
200.sixtrack	FORTRAN 77	High energy nuclear physics accelerator design
301.apsi	FORTRAN 77	Meteorology: pollutant distribution

SPEC '89 (this one is really old)

- Compiler "enhancements" and performance



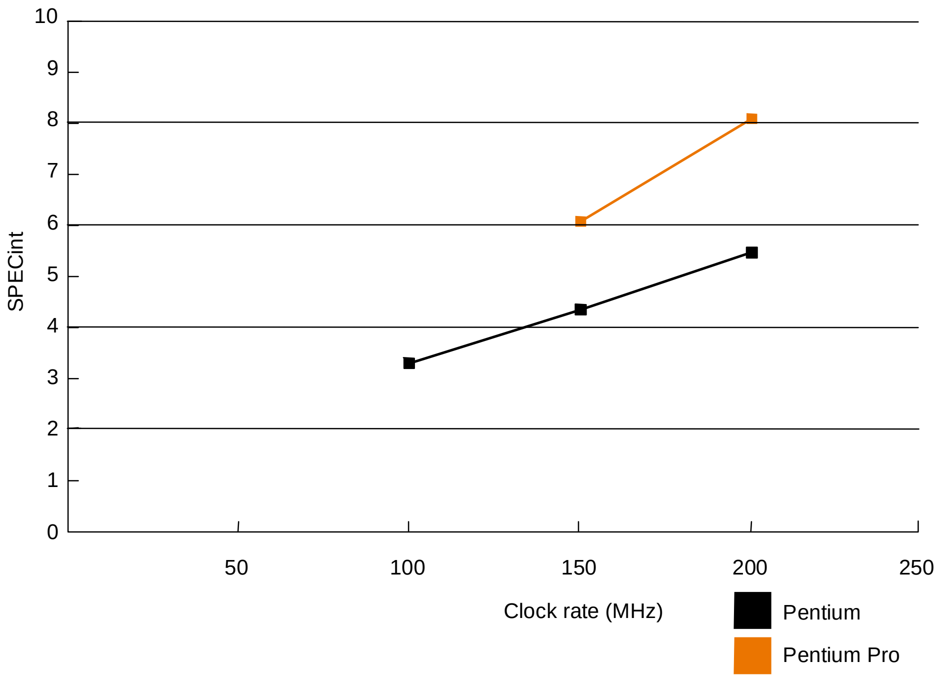
SPEC '95

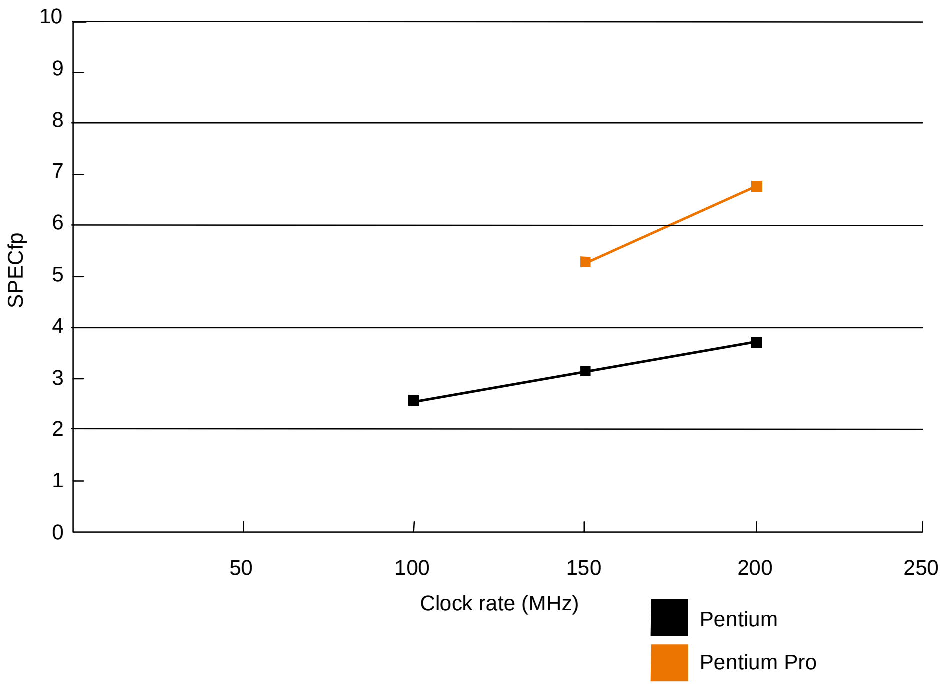
Benchmark	Description
go	Artificial intelligence; plays the game of Go
m88ksim	Motorola 88k chip simulator; runs test program
gcc	The Gnu C compiler generating SPARC code
compress	Compresses and decompresses file in memory
li	Lisp interpreter
ijpeg	Graphic compression and decompression
perl	Manipulates strings and prime numbers in the special-purpose programming language Perl
vortex	A database program
tomcatv	A mesh generation program
swim	Shallow water model with 513 x 513 grid
su2cor	quantum physics; Monte Carlo simulation
hydro2d	Astrophysics; Hydrodynamic Naiver Stokes equations
mgrid	Multigrid solver in 3-D potential field
applu	Parabolic/elliptic partial differential equations
trub3d	Simulates isotropic, homogeneous turbulence in a cube
apsi	Solves problems regarding temperature, wind velocity, and distribution of pollutant
fpppp	Quantum chemistry
wave5	Plasma physics; electromagnetic particle simulation

SPEC ‘95

Does doubling the clock rate double the performance?

Can a machine with a slower clock rate have better performance?

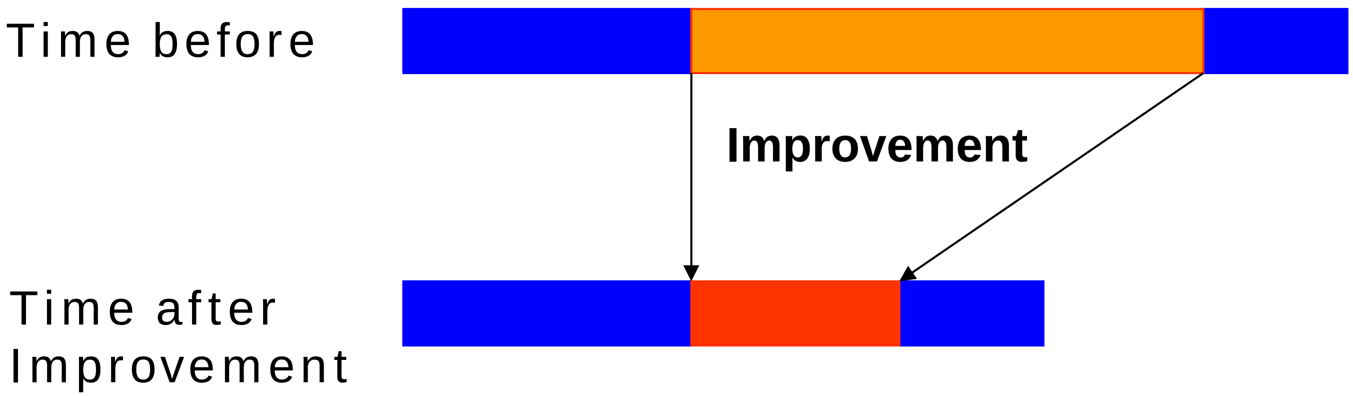




Amdahl's Law

Execution Time After Improvement =

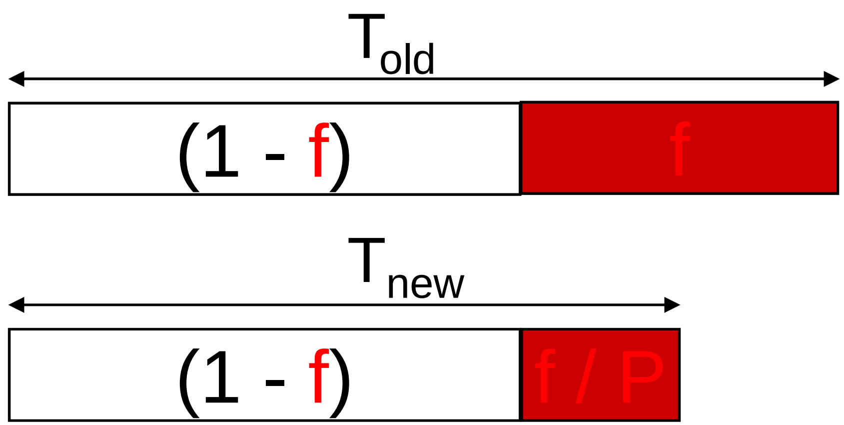
Execution Time Unaffected +(Execution Time Affected / Amount of Improvement)



Amdahl’s Law

- Speed-up =
  $Perf_{new} / Perf_{old} = Exec\_time_{old} / Exec\_time_{new} = \dfrac{1}{(1-f)+\dfrac{f}{P}}$

- Performance improvement from using faster mode is limited by the fraction the faster mode can be applied.



Example

- "Suppose a program runs in 100 seconds on a machine, with multiply responsible for 80 seconds of this time. How much do we have to improve the speed of multiplication if we want the program to run 4 times faster?"

$$\text{Execute time after improvement} = \frac{80\ \text{seconds}}{n} + (100 - 80\text{seconds})$$

$$25\ \text{seconds} = \frac{80\ \text{seconds}}{n} + 20\ \text{seconds}$$

$$n = 16$$

How about making it 5 times faster?

- *Principle: Make the common case fast*

Remember

- Performance is specific to a particular program/s
  - Total execution time is a consistent summary of performance
- For a given architecture performance increases come from:
  - increases in clock rate (without adverse CPI affects)
  - improvements in processor organization that lower CPI
  - compiler enhancements that lower CPI and/or instruction count
- Pitfall: expecting improvement in one aspect of a machine's

  performance to affect the total performance
- You should not always believe everything you read! Read carefully!