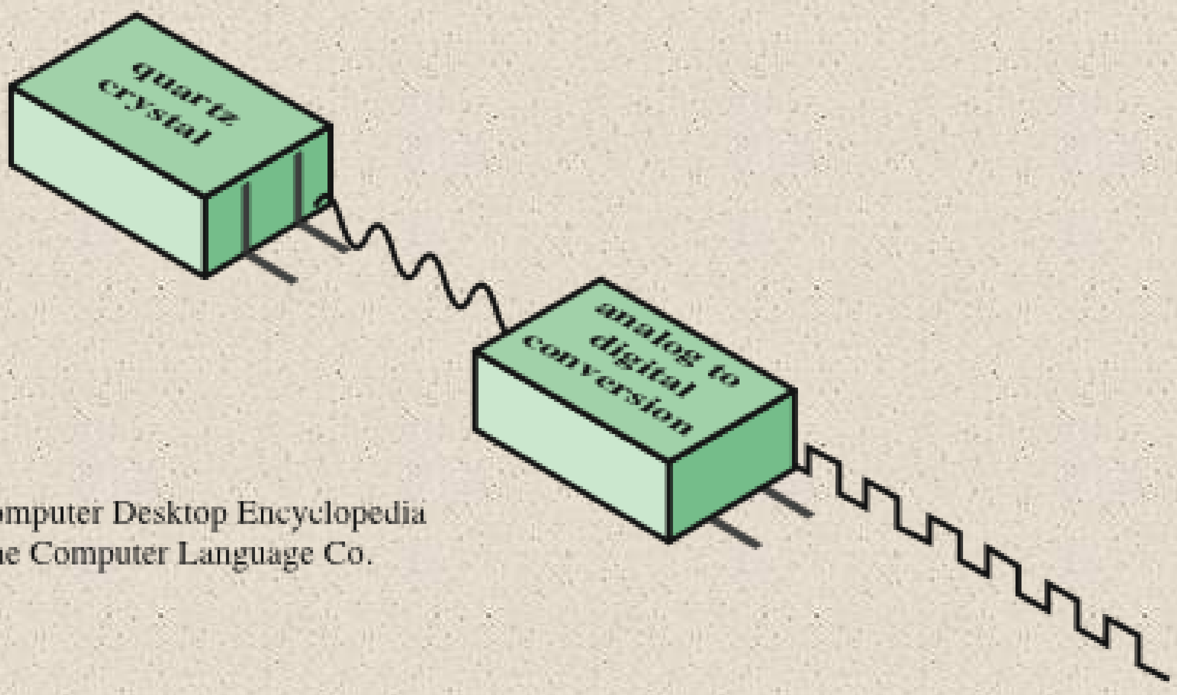
System Clock



From Computer Desktop Encyclopedia
1998, The Computer Language Co.

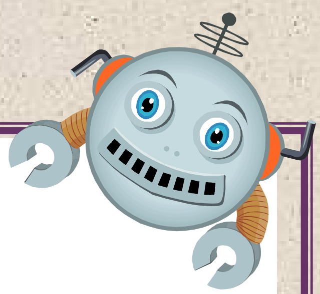
Figure 2.13 System Clock

Performance Factors and System Attributes

Table 2.9

	I_c	p	m	k	τ
Instruction set architecture	X	X			
Compiler technology	X	X	X		
Processor implementation		X			X
Cache and memory hierarchy				X	X

Performance



Cycle Time ---- $\tau = \frac{1}{f}$

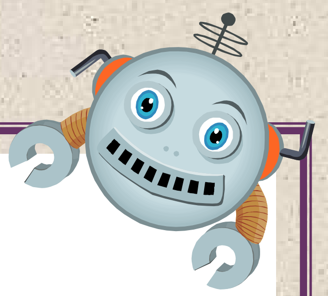
Instruction Count ---- I_C, number of instruction executions.

Average cycle per instruction = CPI, if it is equal to all instruction that mean constant.

CPI_i : varies from instruction to another.

$$CPI = \frac{\sum_{i=1}^{n}(CPI_i \times I_i)}{I_C}$$

Performance



Processor time T needed to execute a given program:

$$T = I_C \times CPI \times \tau = I_C \times [P + (m \times k)] \times \tau$$

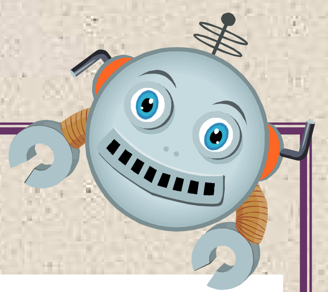
P = *number of processor cycle needed to decode and execute instruction.*

m = *number of memory references needed*

k = *ratio between memory cycle time and processor cycle time*

$$MIPS\ rate = \frac{I_C}{T \times 10^6} = \frac{f}{CPI \times 10^6}$$

Performance

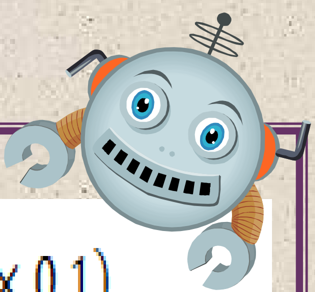


Example: Consider the execution of program that result in the execution of 2 million instructions on 400 MHz processor. The program consists of 4 major types of instructions. The instruction mix and CPI for each instruction type are given:

Instruction Type	CPI	Instruction Mix. %
Arithmetic and Logic	1	60
Load/Store with cache	2	18
Branch	4	12
Memory reference with cache miss	8	10

Find MIPS rate, and execution time????

Performance



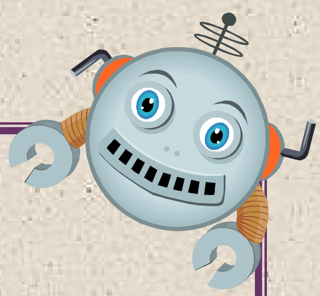
$$Average\ CPI\ uniprocessor = (1 \times 0.6) + (2 \times 0.18) + (4 \times 0.12) + (8 \times 0.1)$$
$$= 2.24$$

$$MIPS\ rate = \frac{400 \times 10^6}{2.24 \times 10^6} = 178$$

$$Execution\ time = \frac{\#\ of\ instruction}{MIPS\ rate} = \frac{2 \times 10^6}{178 \times 10^6} = 11.2359\ ms$$

$$Or\ Execution\ time\ (T) = I_C \times CPI \times \tau = \frac{I_C \times CPI}{f} = \frac{2 \times 10^6 \times 2.24}{400 \times 10^6} = 11.2359\ ms$$

Benchmarks



For example, consider this high-level language statement:

A = B + C /* assume all quantities in main memory */

With a traditional instruction set architecture, referred to as a complex instruction set computer (CISC), this instruction can be compiled into one processor instruction:

add mem(B), mem(C), mem (A)

On a typical RISC machine, the compilation would look something like this:

load mem(B), reg(1);
load mem(C), reg(2);
add reg(1), reg(2), reg(3);
store reg(3), mem (A)

Desirable Benchmark Characteristics

- Written in a high-level language, making it portable across different machines
- Representative of a particular kind of programming style, such as system programming, numerical programming, or commercial programming
- Can be measured easily
- Has wide distribution

System Performance Evaluation Corporation (SPEC)

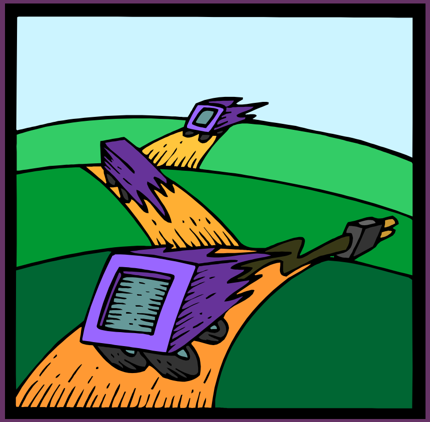
- Benchmark suite
  - A collection of programs, defined in a high-level language
  - Attempts to provide a representative test of a computer in a particular application or system programming area
- SPEC
  - An industry consortium
  - Defines and maintains the best known collection of benchmark suites
  - Performance measurements are widely used for comparison and research purposes

SPEC CPU2006

- Best known SPEC benchmark suite
- Industry standard suite for processor intensive applications
- Appropriate for measuring performance for applications that spend most of their time doing computation rather than I/O
- Consists of 17 floating point programs written in C, C++, and Fortran and 12 integer programs written in C and C++
- Suite contains over 3 million lines of code
- Fifth generation of processor intensive suites from SPEC

+

Amdahl's Law



- Gene Amdahl [AMDA67]
- Deals with the potential speedup of a program using multiple processors compared to a single processor
- Illustrates the problems facing industry in the development of multi-core machines
  - Software must be adapted to a highly parallel execution environment to exploit the power of parallel processing
- Can be generalized to evaluate and design technical improvement in a computer system

Amdahl's Law

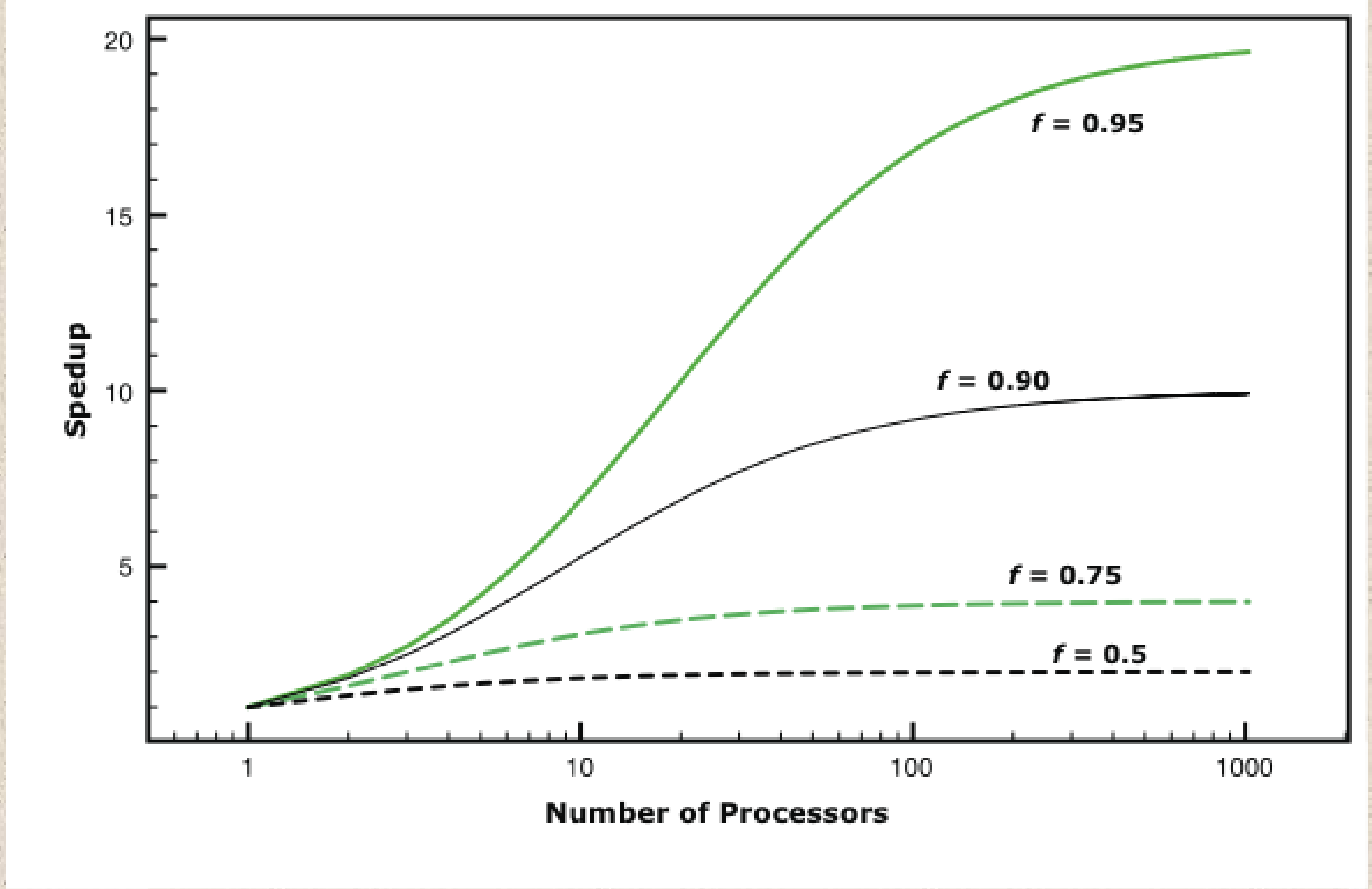


Figure 2.14 Amdahl's Law for Multiprocessors

Little's Law

- Fundamental and simple relation with broad applications
- Can be applied to almost any system that is statistically in steady state, and in which there is no leakage
- Queuing system
  - If server is idle an item is served immediately, otherwise an arriving item joins a queue
  - There can be a single queue for a single server or for multiple servers, or multiples queues with one being for each of multiple servers
- Average number of items in a queuing system equals the average rate at which items arrive multiplied by the time that an item spends in the system
  - Relationship requires very few assumptions
  - Because of its simplicity and generality it is extremely useful

+ # Summary

Chapter 2

Computer Evolution and Performance

- First generation computers
  - Vacuum tubes
- Second generation computers
  - Transistors
- Third generation computers
  - Integrated circuits
- Performance designs
  - Microprocessor speed
  - Performance balance
  - Chip organization and architecture
- Multi-core
- MICs
- GPGPUs
- Evolution of the Intel x86
- Embedded systems
- ARM evolution
- Performance assessment
  - Clock speed and instructions per second
  - Benchmarks
  - Amdahl's Law
  - Little's Law