

MO401

Arquitetura de Computadores I

2006

Prof. Paulo Cesar Centoducatte

ducatte@ic.unicamp.br

www.ic.unicamp.br/~ducatte

MO401

Arquitetura de Computadores I

Revisão

Resumo #1/3: Pipelining & Desempenho

- Sobreposição de tarefas; fácil se as tarefas são independentes
- Speed Up \leq Pipeline Depth; Se CPI ideal for 1, então:

$$\text{Speedup} = \frac{\text{Pipeline depth}}{1 + \text{Pipeline stall CPI}} \times \frac{\text{Cycle Time}_{\text{unpipelined}}}{\text{Cycle Time}_{\text{pipelined}}}$$

- Hazards limita o desempenho nos computadores:
 - Estrutural: é necessário mais recursos de HW
 - Dados (RAW, WAR, WAW): forwarding, compiler scheduling
 - Controle: delayed branch, prediction
- Tempo é a medida de desempenho: latência ou throughput
- CPI Law:

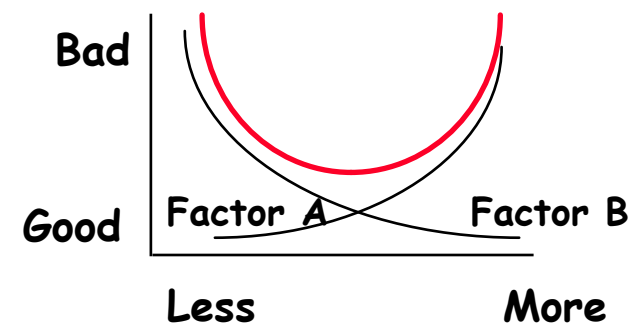
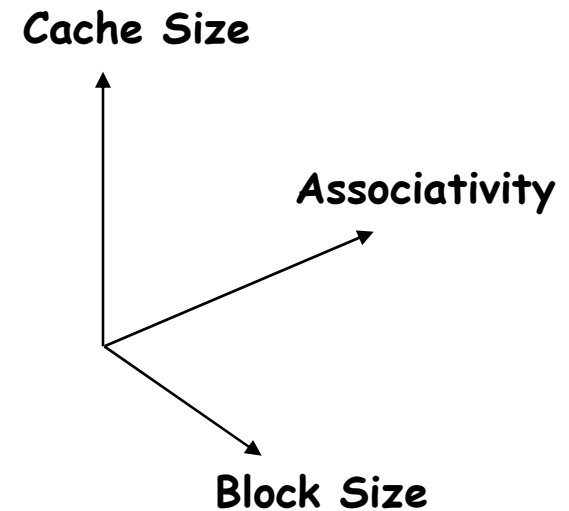
CPU time	=	$\frac{\text{Seconds}}{\text{Program}}$	=	$\frac{\text{Instructions}}{\text{Program}}$	x	$\frac{\text{Cycles}}{\text{Instruction}}$	x	$\frac{\text{Seconds}}{\text{Cycle}}$
----------	---	---	---	--	---	--	---	---------------------------------------

Resumo #2/3: Caches

- Princípio da Localidade:
 - Programas acessam relativamente uma pequena porção do espaço de endereçamento em um dado instante de tempo.
 - » Localidade Temporal : Localidade no Tempo
 - » Localidade Espacial : Localidade no Espaço
- Cache Misses: 3 categorias
 - Compulsory Misses
 - Capacity Misses
 - Conflict Misses
- Políticas de Escrita:
 - Write Through: (write buffer)
 - Write Back

Resumo #3/3: Cache Design

- **Várias Dimensões interagindo**
 - Tamanho da cache
 - Tamanho do block
 - associatividade
 - Política de "replacement"
 - Write-through vs Write-back
- Solução "ótima" é um compromisso
 - Depende da característica dos acessos
 - » workload
 - » I-cache, D-cache, TLB
 - Depende da razão tecnologia / custo



MO401

Arquitetura de Computadores I

Fundamentos

"Computer Architecture: A Quantitative Approach" - (Capítulo 1)

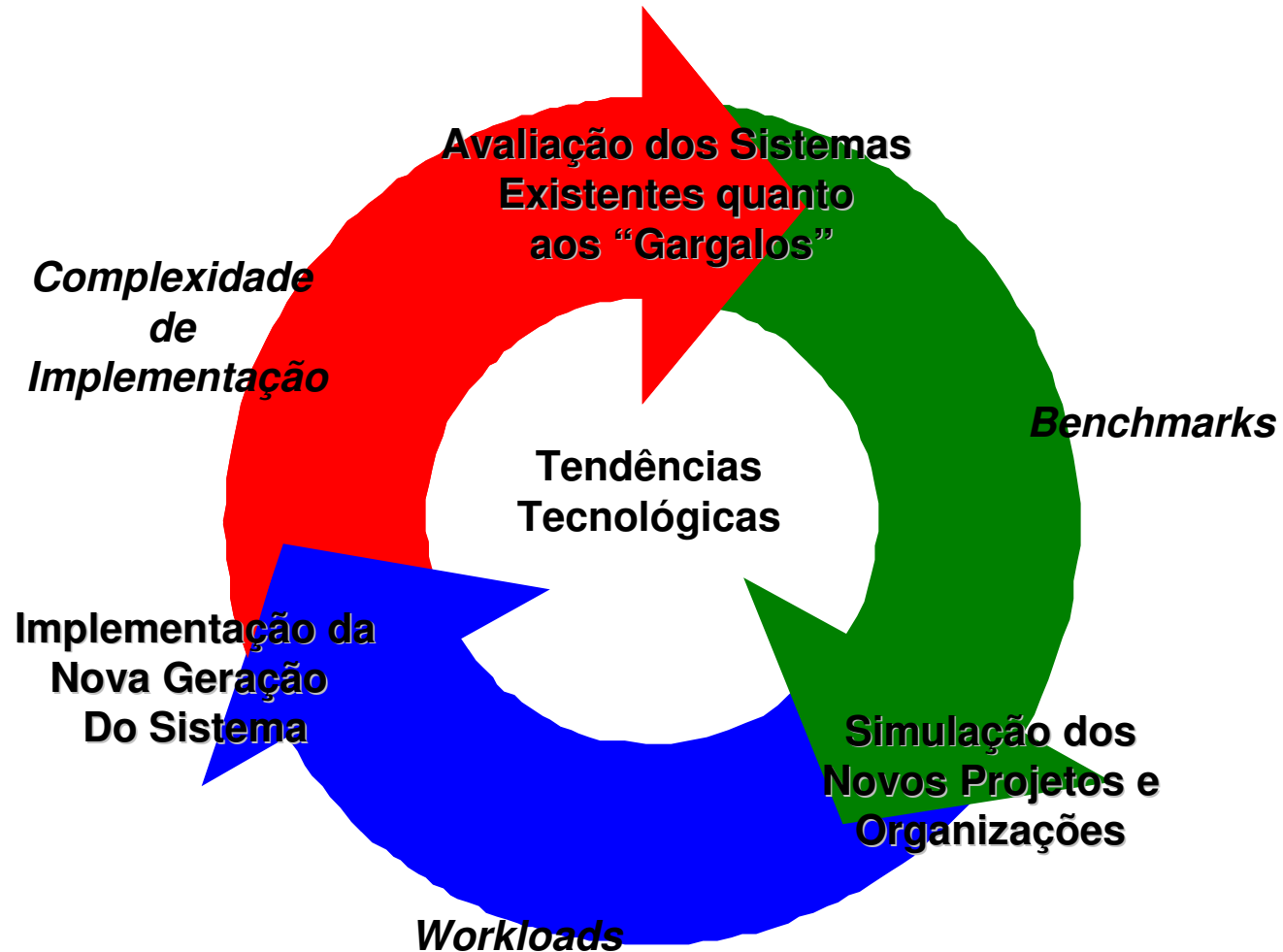
Sumário

- **Introdução**
 - O que é Arquitetura de Computadores?
- **Tarefas do Projetista**
- **Tecnologia e Tendências na Computação**
- **Custo, Preço e suas Tendências**
- **Medidas**
- **Princípios Quantitativos**
- **Outros Aspectos**

O que é Arquitetura de Computadores (AC)?

- **1950s a 1960s: Cursos de AC?**
Aritmética Computacional
- **1970s a meados dos anos 1980s: Cursos de AC?**
Projeto do Conjunto de Instruções (ISA), especialmente voltado para compiladores
- **1990s a 2000s: Cursos de AC?**
Projeto de CPU, Sistemas de Memórias, Sistemas de I/O, Multiprocessadores.
 - Enfoque Baseado em Desempenho
- **200? : Cursos de AC?**
Multi-Core, Embedded System
 - Enfoque Baseado em Desempenho e Consumo

Tarefas do Projetista



Tendências

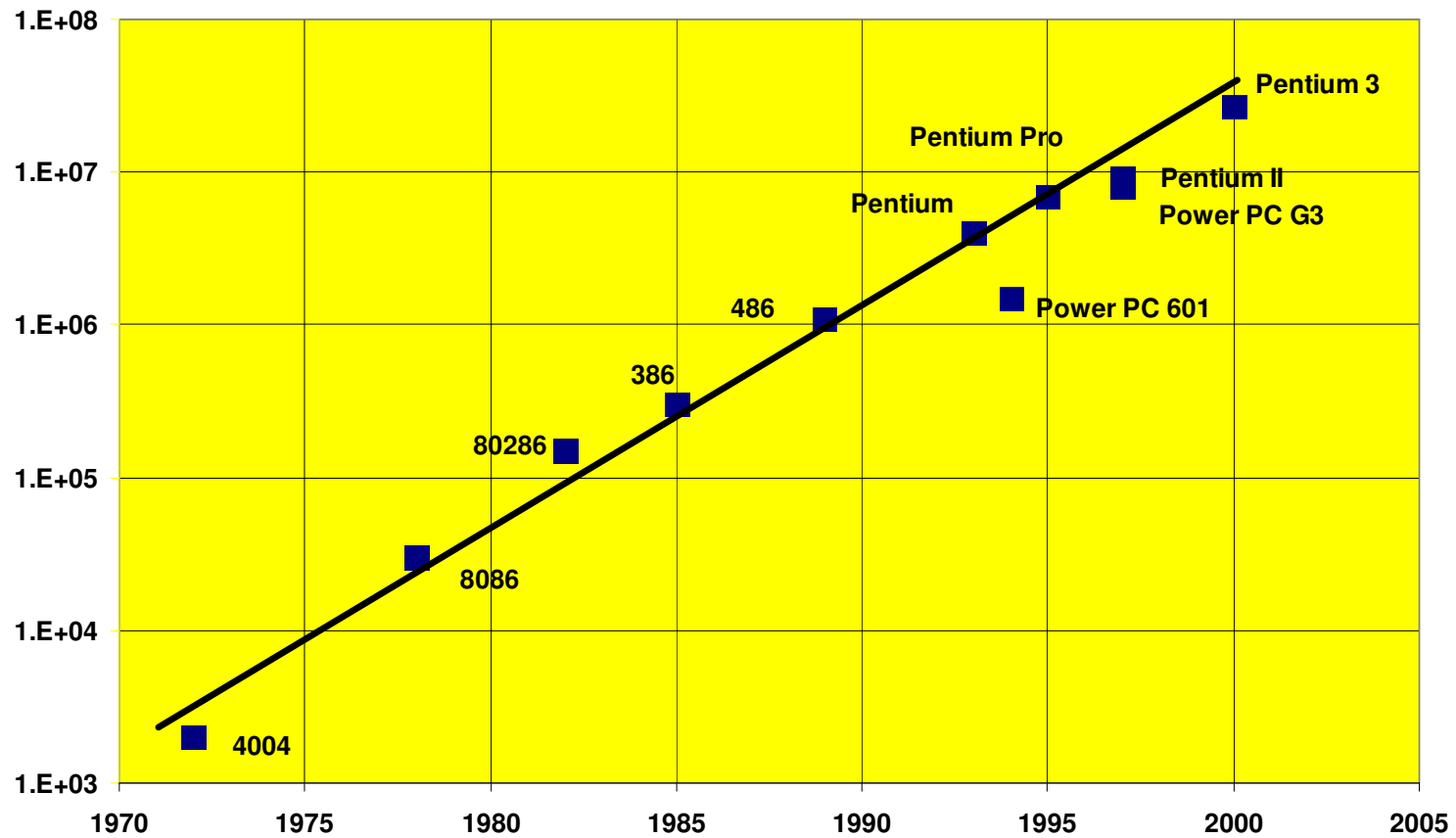
- Gordon Moore (fundador da Intel), em 1965 observou que o número de transistores em um chip dobrava a cada ano (Lei de Moore)
Continua valida até os dias de hoje !!!???
- O desempenho dos processadores, medidos por diversos **benchmarks**, também tem crescido de forma acelerada.
- A capacidade das memórias tem aumentado significativamente nos últimos 20 anos
(E o custo reduzido)

Quais as Razões Desta Evolução nos Últimos Anos?

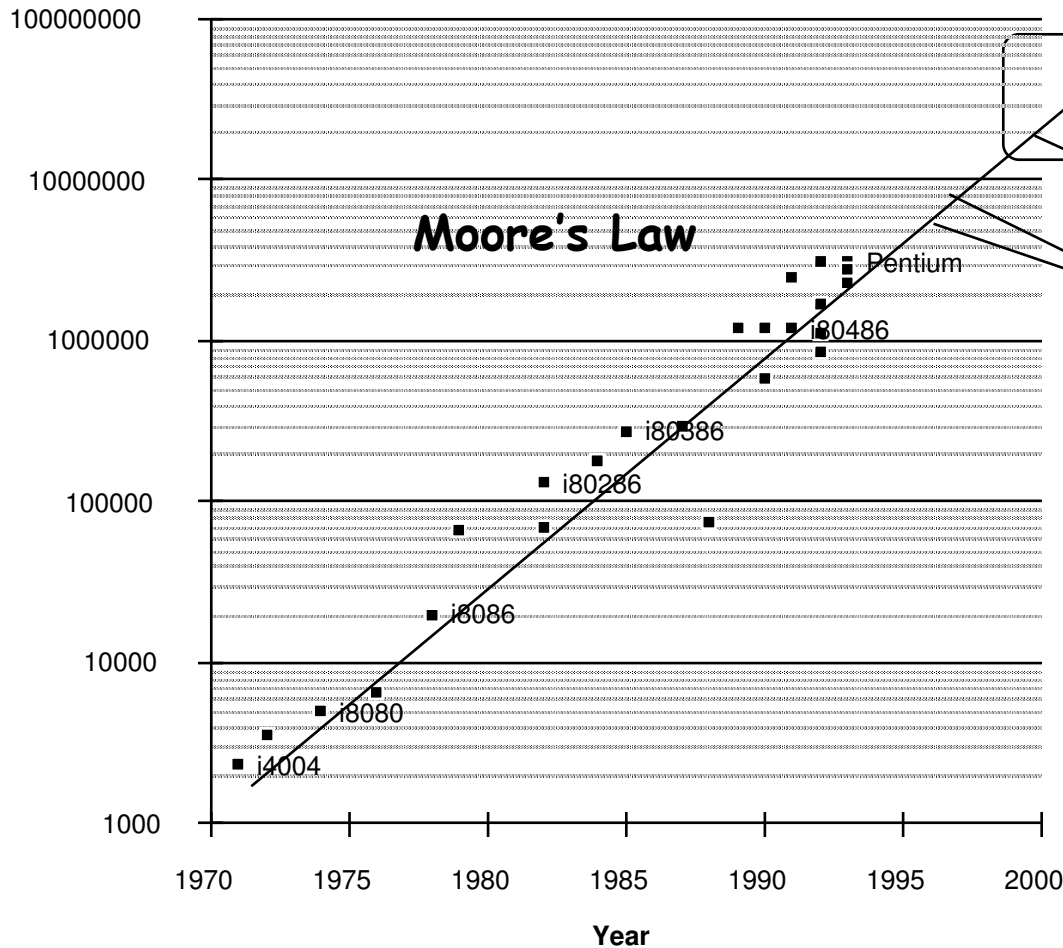
- **Desempenho**
 - Avanços tecnológicos
 - » Domínio de CMOS sobre as tecnologias mais antigas (TTL, ECL, ...) em custo e desempenho
 - Avanços nas arquiteturas
 - » RISC, superscalar, VLIW,
 - » DRAM, SDRAM, ...
 - » RAID, ...
- **Preço: Baixo custo devido a:**
 - Desenvolvimento mais simples
 - » CMOS VLSI => sistemas menores, menos componentes
 - Alto volume (escala)
-

Tendências Lei de Moore

Transistors Per Chip



Tendência Tecnológica: Capacidade Microprocessadores



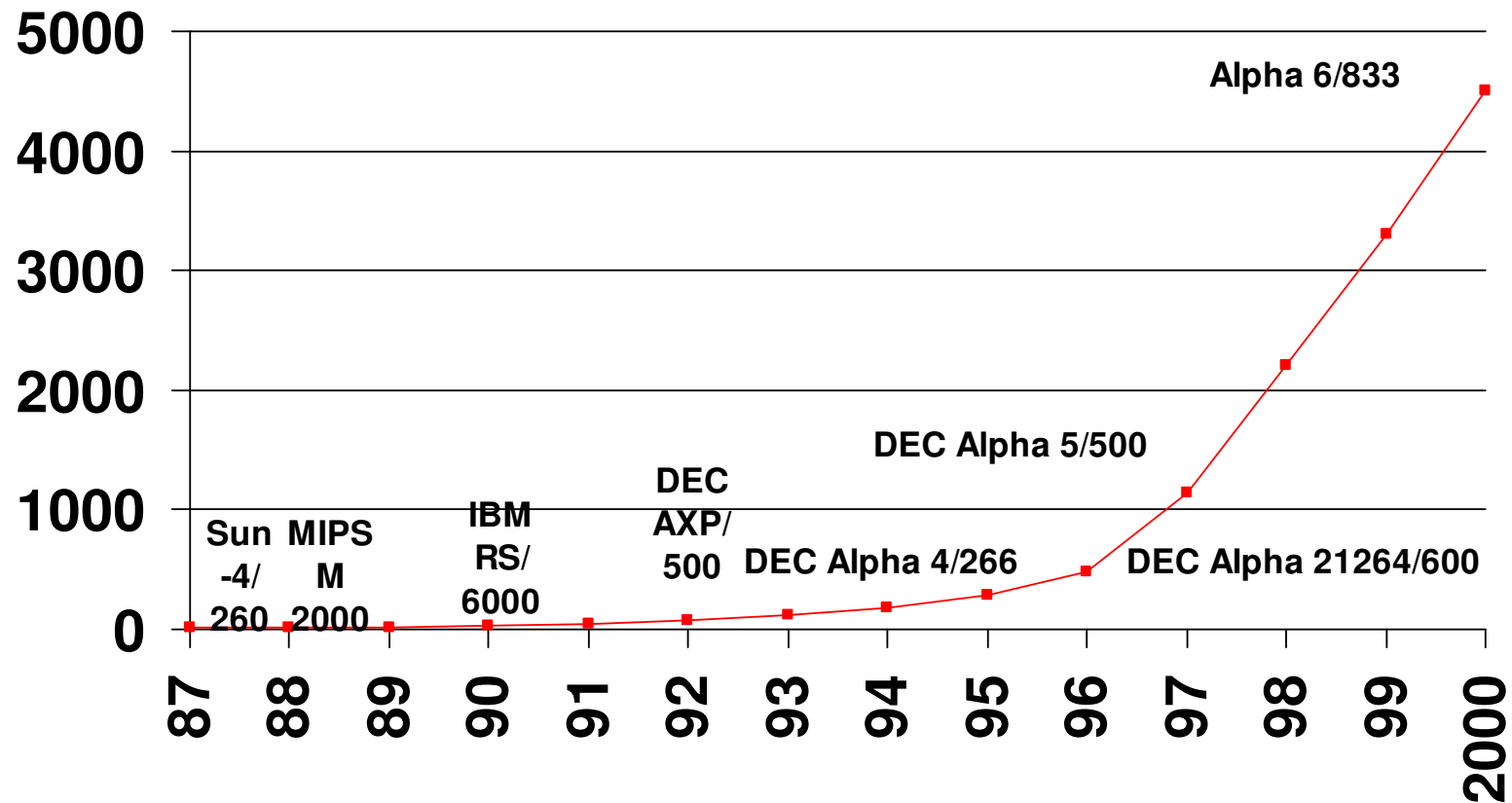
Alpha 21264: 15 milhões
Pentium Pro: 5.5 milhões
PowerPC 620: 6.9 milhões
Alpha 21164: 9.3 milhões
Sparc Ultra: 5.2 milhões

CMOS:

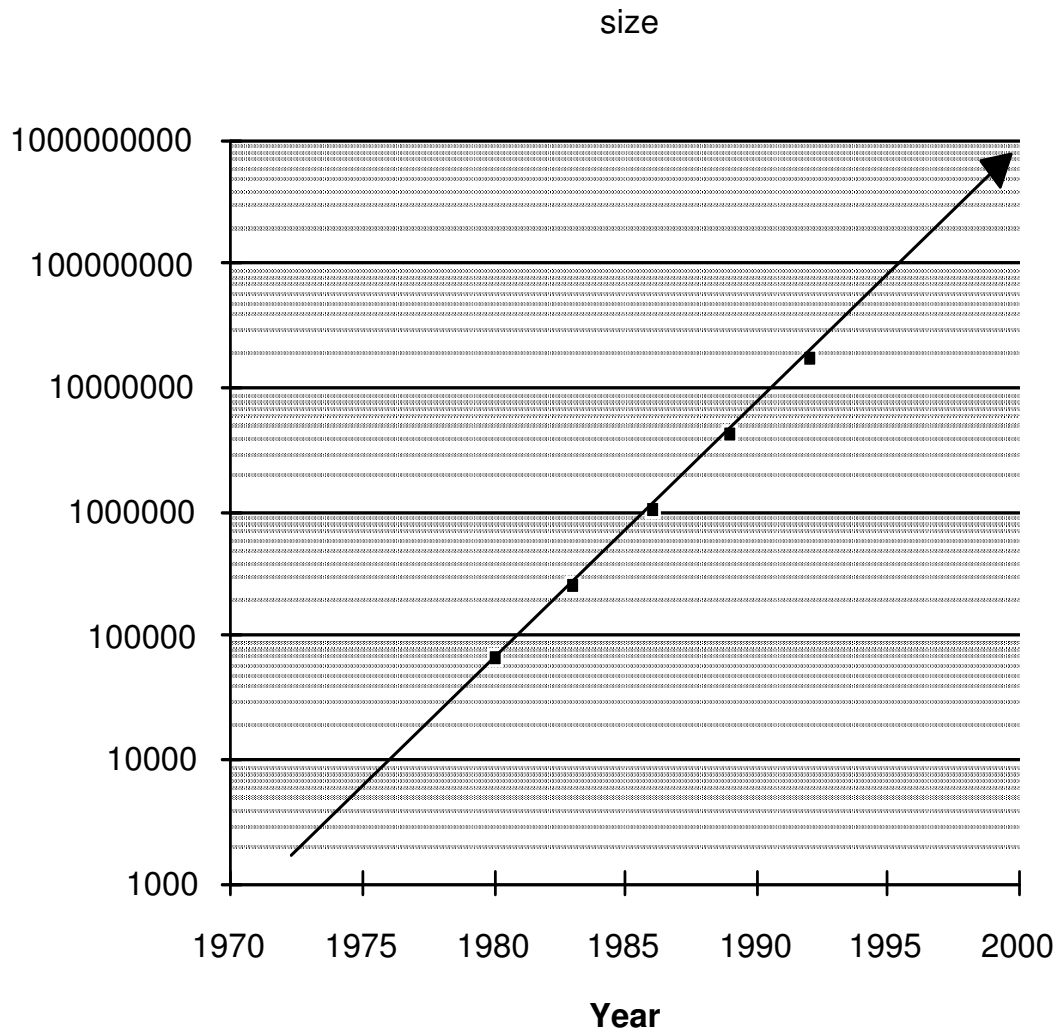
- Die size: 2X a cada 3 anos

Tendências

Desempenho dos Processadores



Tendências Capacidade das Memórias



ano	Mbyte	cycle time
1980	0.0625	250 ns
1983	0.25	220 ns
1986	1	190 ns
1989	4	165 ns
1992	16	145 ns
1996	64	120 ns
2000	256	100 ns

Tendências Velocidade

- Para a CPU o crescimento da velocidade tem sido muito acelerado
- Para Memória e disco o crescimento da velocidade tem sido modesto

Isto tem levado a mudanças significativas nas arquiteturas, SO e mesmo nas práticas de programação.

	<u>Capacidade</u>	<u>Speed (latency)</u>
Lógica	2x em 3 anos	2x em 3 anos
DRAM	4x em 3 anos	2x em 10 anos
Disco	4x em 3 anos	2x em 10 anos

Medidas ?

- Como descrever em forma numérica o desempenho dos computadores?
- Quais ferramentas (ou qual ferramental) usar para realizar e apresentar as medidas?

Métricas

Plane	DC to Paris	Speed	Passengers	Throughput (pmp)
Boeing 747	6.5 hours	610 mph	470	286,700
BAD/Sud Concorde	3 hours	1350 mph	132	178,200

- Tempo para executar uma tarefa (ExTime)
 - Execution time, response time, latency
- Tarefas por dia, hora, semana, segundo, ns, ... (Desempenho)
 - Throughput, bandwidth

Métricas Comparação

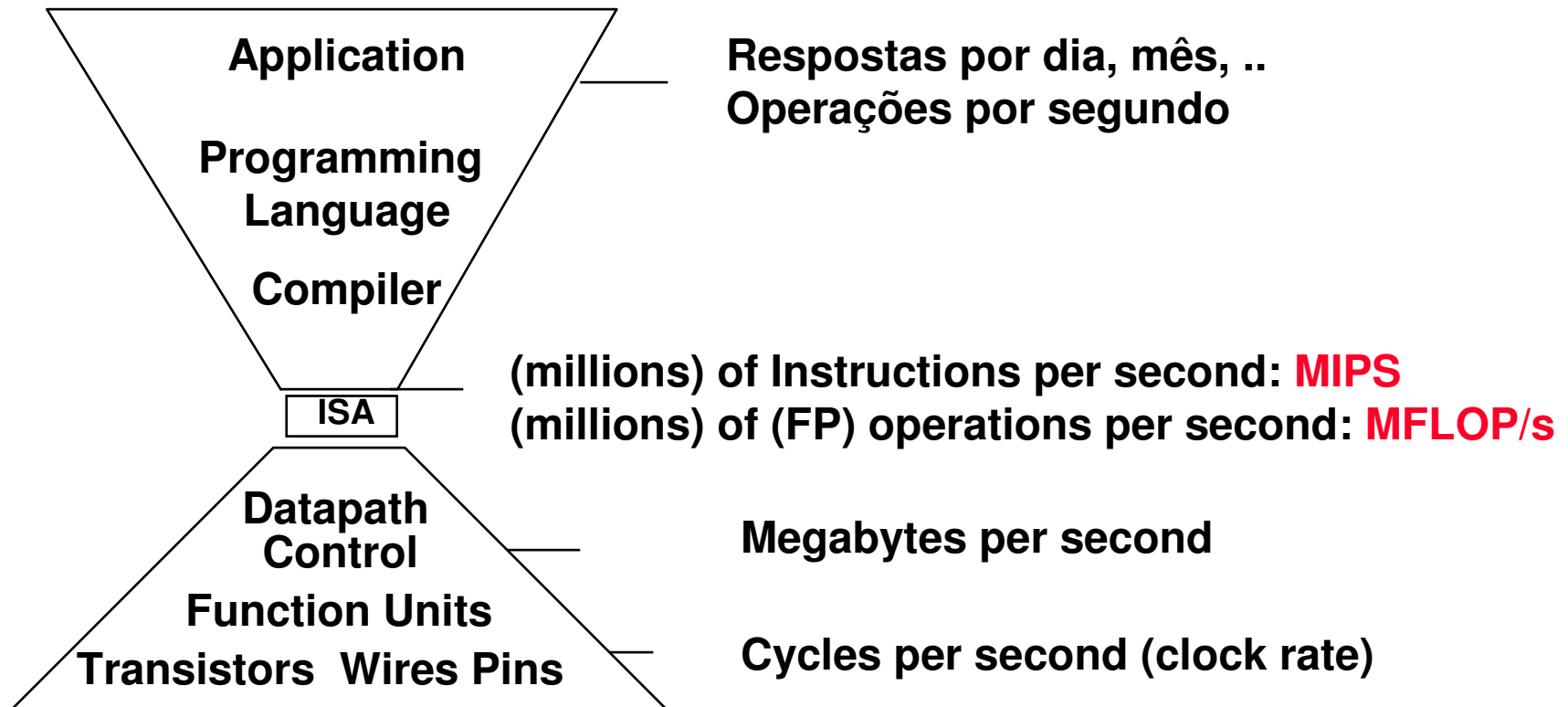
"X é n vezes mais rápido que Y" significa:

$$\frac{\text{ExTime}(Y)}{\text{ExTime}(X)} = \frac{\text{Performance}(X)}{\text{Performance}(Y)}$$

Velocidade do Concorde vs. Boeing 747

Throughput do Boeing 747 vs. Concorde

Métricas Throughput



Métodos para Estimar o Desempenho

- Benchmarks, Traces, Mixes
- Hardware: custo, delay, área, consumo de energia
- Simulação (vários níveis)
 - ISA, RT, Gate, Circuito
- Teoria das Filas
- Regras Práticas
- “Leis”/Princípios

Benchmarks

- **Aplicações Reais**
 - Compiladores, processadores de texto, ...
 - Problema de portabilidade, difícil medir o tempo de execução
- **Aplicações Modificadas**
 - Melhora a portabilidade, pode ser refinado para medir um certo aspecto de interesse (exp: tempo de cpu)
- **Kernels**
 - Usados para avaliar características específicas
 - Livermore Loops, Linpack
- **Toy Benchmarks**
 - 10 a 100 linhas de código, fácil de programar, avaliação inicial
- **Benchmarks Sintéticos**
 - Semelhantes aos Kernels
 - Whetstone, Dhrystone

Benchmarks

- **Desktop**
 - SPEC (<http://www.spec.org>)
- **Servidores**
 - SPEC
- **Sistemas Embarcados**
 - EEMBC (Embedded Microprocessor Benchmark Consortium) (<http://www.eembc.org>)
 - » Automotivo
 - » Consumidor
 - » Rede
 - » Automação de Escritório
 - » telecomunicações

Benchmarks

SPEC: Standard Performance Evaluation Corporative
(<http://www.spec.org>)

- **Primeira Versão - 1989**
 - 10 programas ("SPECmarks")
- **Segunda Versão - 1992**
 - SPECInt92 (6 programas)
 - SPECfp92 (14 programas)
 - » Compiler Flags: livre
- **Terceira Versão - 1995**
 - SPECint95 (8 programas)
 - SPECfp95 (10 programas)
 - SPECint_base95, SPECfp_base95
 - » "benchmarks útil por 3 anos"
 - » Compiler Flags: controladas

Benchmarks

SPEC CPU2000 - CINT2000

Programa	Linguagem	Finalidade
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.perlbnk	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

Benchmarks

SPEC CPU2000 - CFP2000

Program	Linguagem	Finalidade
168.wupwise	Fortran 77	Physics / Quantum Chromodynamics
171.swim	Fortran 77	Shallow Water Modeling
172.mgrid	Fortran 77	Multi-grid Solver: 3D Potential Field
173.applu	Fortran 77	Parabolic / Elliptic Differential Equations
177.mesa	C	3-D Graphics Library
178.galgel	Fortran 90	Computational Fluid Dynamics
179.art	C	Image Recognition / Neural Networks
183.quake	C	Seismic Wave Propagation Simulation
187.facerec	Fortran 90	Image Processing: Face Recognition
188.amp	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 Physics Accelerator Design
301.apsi	Fortran 77	Meteorology: Pollutant Distribution

<http://www.spec.org/osg/cpu2000/CFP2000/>

Benchmarks - Exemplo de Resultado para SpecINT2000

<http://www.spec.org/osg/cpu2000/results/res2000q3/cpu2000-20000718-00168.asc>

Benchmarks	Base	Base	Base	Peak	Peak	Peak
	Ref Time	Run Time	Ratio	Ref Time	Run Time	Ratio
164.gzip	1400	277	505*	1400	270	518*
175.vpr	1400	419	334*	1400	417	336*
176.gcc	1100	275	399*	1100	272	405*
181.mcf	1800	621	290*	1800	619	291*
186.crafty	1000	191	522*	1000	191	523*
197.parser	1800	500	360*	1800	499	361*
252.eon	1300	267	486*	1300	267	486*
253.perlbnk	1800	302	596*	1800	302	596*
254.gap	1100	249	442*	1100	248	443*
255.vortex	1900	268	710*	1900	264	719*
256.bzip2	1500	389	386*	1500	375	400*
300.twolf	3000	784	382*	3000	776	387*
SPECint_base2000			438			
SPECint2000						442

Intel OR840(1 GHz
Pentium III processor)

Benchmarks

Como Apresentar o Desempenho?

Gerentes gostam de números.

Técnicos querem mais:

- Reprodutibilidade - informações que permitam que o experimento seja repetido (reproduzido)
- Consistência nos dados, ié se o experimento é repetido os dados devem ser compatíveis entre si

Como Apresentar os Dados?

	Computador A	Computador B	Computador C
Programa P1 (secs)	1	10	20
Programa P2 (secs)	1000	100	20
Total Time (secs)	1001	110	40

Como Apresentar os Dados

- Média Aritmética (média aritmética ponderada)

$$\Sigma(T_i)/n \text{ or } \Sigma(W_i * T_i)$$

- Média Harmônica (média harmônica ponderada)

$$n/\Sigma(1/R_i) \text{ or } n/\Sigma(W_i/R_i)$$

- Média geométrica $(\prod T_j / N_j)^{1/n}$

- Tempo de execução normalizado (e.g., X vezes melhor que SPARCstation 10 - **Spec**)

- Não use média aritmética para tempos de execução normalizado (o resultado, quando comparado n máquinas, depende de qual máquina é usada como referência), **use média geométrica**

Como Apresentar os Dados

máquina	A	B
programa 1	10 => t1A	20 => t1B
programa 2	30 => t2A	5 => t2B

Média aritmética normalizada em A:

$$(t1A/t1A + t2A/t2A)/2 = 1 < (t1B/t1A + t2B/t2A)/2 = 13/12$$

Média aritmética normalizada em B:

$$(t1A/t1B + t2A/t2B)/2 = 13/4 > (t1B/t1B + t2B/t2B)/2 = 1$$

CONTRADIÇÃO!!!!

Média Geométrica :

$$((t1A * t2A)/(t1A * t2A))^{-.5} = 1 > ((t1B * t2B)/(t1A * t2A))^{-.5} = (1/3)^{-.5} \Rightarrow$$

normalizado em A

$$((t1A * t2A)/(t1B * t2B))^{-.5} = 3^{-.5} > ((t1B * t2B)/(t1B * t2B))^{-.5} = 1 \Rightarrow$$

normalizado em B

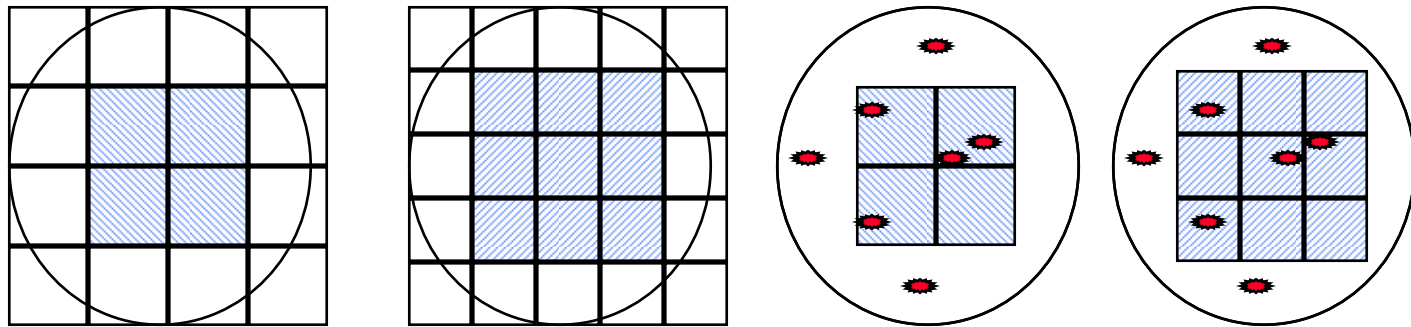
Custo, Preço e suas tendências

Custo de Circuito Integrado (IC)

$$\text{IC cost} = \frac{\text{Die cost} + \text{Testing cost} + \text{Packaging cost}}{\text{Final test yield}}$$

$$\text{Die cost} = \frac{\text{Wafer cost}}{\text{Dies per Wafer} \times \text{Die yield}}$$

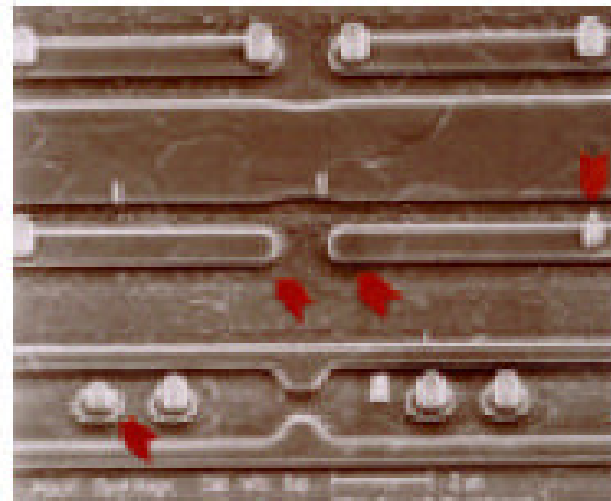
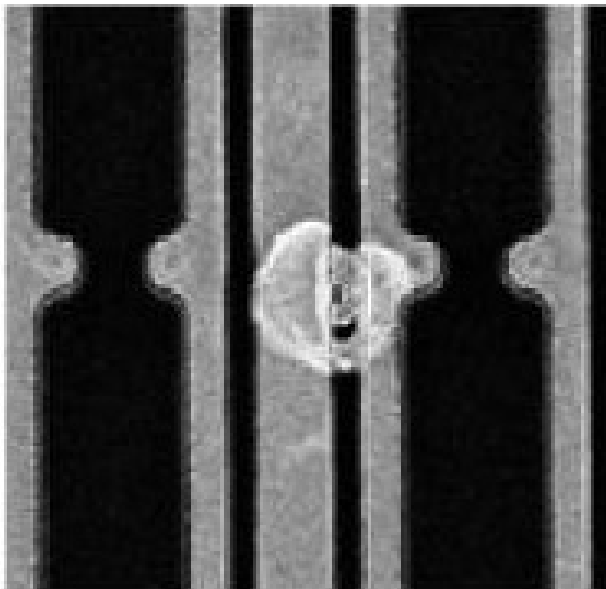
$$\text{Dies per wafer} = \frac{\pi (\text{Wafer_diam}/2)^2}{\text{Die_Area}} - \frac{\pi \times \text{Wafer_diam}}{\sqrt{2} \cdot \text{Die_Area}} - \text{Test_Die}$$



$$\text{Die Yield} = \text{Wafer_yield} \times \left\{ 1 + \left(\frac{\text{Defect_Density} \times \text{Die_area}}{\alpha} \right)^{-\alpha} \right\}$$

Custo do Die é proporcional à (área do die)⁴

Integrated Circuits Yield - Defects



ECE888/Koren Part.1 .20

Copyright 2004 Koren UMass

Exemplos Reais

Chip	Metal layers	Line width	Wafer cost	Defect /cm ²	Area mm ²	Dies/wafer	Yield	Die Cost
386DX	2	0.90	\$900	1.0	43	360	71%	\$4
486DX2	3	0.80	\$1200	1.0	81	181	54%	\$12
PowerPC 601	4	0.80	\$1700	1.3	121	115	28%	\$53
HP PA 7100	3	0.80	\$1300	1.0	196	66	27%	\$73
DEC Alpha	3	0.70	\$1500	1.2	234	53	19%	\$149
SuperSPARC	3	0.70	\$1700	1.6	256	48	13%	\$272
Pentium	3	0.80	\$1500	1.5	296	40	9%	\$417

– From "Estimating IC Manufacturing Costs," by Linley Gwennap, *Microprocessor Report*, August 2, 1993, p. 15

Abordagem Quantitativa

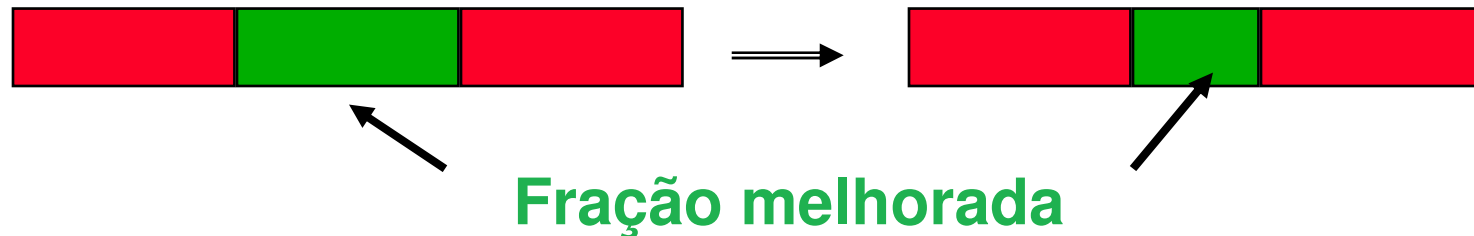
- Faça o caso comum ser mais rápido
- Amdahl's Law:
 - Relaciona o speedup total de um sistema com o speedup de uma porção do sistema

O speedup no desempenho obtido por uma melhoria é limitado pela fração do tempo na qual a melhoria é utilizada

Abordagem Quantitativa Amdahl's Law

Speedup devido a uma melhoria E:

$$\text{Speedup}(E) = \frac{\text{Execution_Time_Without_Enhancement}}{\text{Execution_Time_With_Enhancement}} = \frac{\text{Performance_With_Enhancement}}{\text{Performance_Without_Enhancement}}$$



Abordagem Quantitativa Amdahl's Law

Suponha que a melhoria **E** acelera a execução de uma fração **F** da tarefa de um fator **S** e que o **restante da tarefa** não é afetado pela melhoria **E**. Qual o speedup?

$$\begin{array}{l} T_{Old} = T_F + T_{nF} \\ T_{New} = T_F/S + T_{nF} \end{array} \longrightarrow \frac{T_{Old}}{T_{New}} = \frac{T_F + T_{nF}}{\frac{T_F}{S} + T_{nF}} = \frac{T_F + T_{nF}}{\frac{T_F + ST_{nF}}{S}}$$

$$Speedup = \frac{S(T_F + T_{nF})}{T_F + ST_{nF}}$$

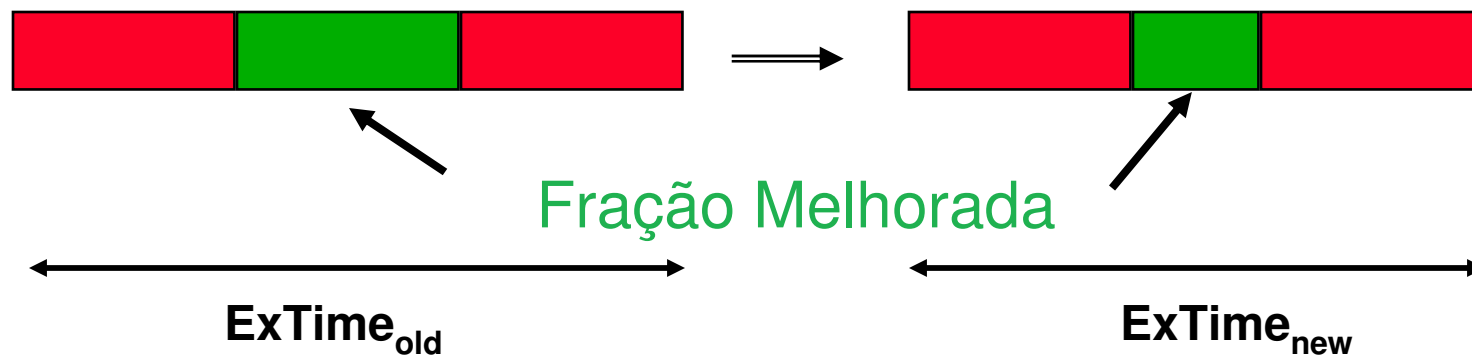
Lim $T_{nF} \rightarrow 0$?

Lim $F \rightarrow 0$?

Abordagem Quantitativa Amdahl's Law

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

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



Abordagem Quantitativa

Amdahl's Law

- Exemplo: Suponha que as instruções de ponto flutuante foram melhoradas e executam 2 vezes mais rápidas, porém somente 10% das instruções, em um programa, são FP

$$\text{ExTime}_{\text{new}} = \text{ExTime}_{\text{old}} \times (0.9 + 0.1/2) = 0.95 \times \text{ExTime}_{\text{old}}$$

$$\text{Speedup}_{\text{overall}} = \frac{1}{0.95} = 1.053$$

Amdahl's Law

Execução de um programa em N processadores

Fraction_{enhanced} = parallelizable part of program

Speedup_{enhanced} = n

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

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

$$\lim_{n \rightarrow \infty} \text{Speedup}_{\text{overall}} = 1 / (1 - \text{Fraction}_{\text{enhanced}})$$

Amdahl's Law - Graph

Law of Diminishing Returns

