CAPÍTULO 5
DESEMPENHO

- Desempenho/Performance
- Tempo de Execução
- Ciclos de Clock
- Escala de integração e Desempenho
- CPI (Ciclos de Clock por Instrução)
- Número de Instruções
- MIPS (Milhões de Instruções por Segundo)
- Benchmarks
- Lei de Moore
- Lei de Amdahl
- Exercícios e Exemplos Resolvidos
- Gargalo de Von Newmann
- Influência dos Dispositivos de I/O, memória e Barramentos
- Como melhorar a arquitetura
- Exercícios

Desempenho/Performance

- A medida de desempenho é importante para:
  - Reportar e sumarizar performance
  - Marketing
  - Comparação de arquiteturas
- Questões:
  - Por que alguns hardwares são melhores que outros para diferentes programas?
  - Quais fatores de desempenho são ligados ao hardware? (i.e., preciso de uma nova máquina ou de um novo sistema operacional?)
  - Como o conjunto de instruções pode afetar o desempenho da máquina?
  - O que significa dizer que o computador A é N vezes mais rápido que o computador B?
  - A visão de um usuário normal e a visão de um administrador da Amazon.com (p. ex.) seriam a mesma?

Desempenho/Performance

- Qual desses aviões tem o melhor desempenho?

Avião	Passageiros	Alcance (mi)	Velocidade (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

- Qual o melhor avião?
  - Quanto o Concorde é mais rápido 747?
  - Quanto o 747 é maior que Douglas DC-8?
- Questão-chave: Como comparar o desempenho de arquiteturas diferentes?
  - Que parâmetros devem ser utilizados?
  - Como isolar o efeito de cada parâmetro ou métrica usada?

Desempenho do Computador

- Tempo de Resposta (latência)
  - Quanto tempo leva minha tarefa para rodar?
  - Quanto tempo leva a execução da minha tarefa?
  - Quanto tempo devo esperar para uma consulta a uma base de dados?
- Vazão (Throughput)
  - Quantas tarefas a máquina pode rodar por vez?
  - Qual é a taxa de execução?
  - Quanto trabalho é feito?

- Um upgrade em uma máquina com um novo processador melhora o quê?
- Uma nova máquina na rede do laboratório melhora o quê?

Tempo de Execução

- Tempo gasto
  - Leva em conta “tudo” (acesso a disco e memória, I/O , etc.)
  - Um número útil, mas às vezes não tão bom para propósitos de comparação
- Tempo de CPU (CPU time)
  - Não conta tempo de I/O nem tempo gasto em outros programas
  - Pode ser dividido em tempo do sistema e tempo do usuário
- NOSSO FOCO: Tempo de CPU do usuário
  - Tempo gasto apenas na execução das instruções que estão dentro do programa (código compilado)

Definição de Desempenho

- Para algum programa rodando na máquina X:

$$desempenho_X = \frac{1}{tempo\ de\ execução_X}$$

- “X é N vezes mais rápida que Y"

$$\frac{desempenho_X}{desempenho_Y} = N$$

OU

$$\frac{tempo\ de\ execução_Y}{tempo\ de\ execução_X} = N$$

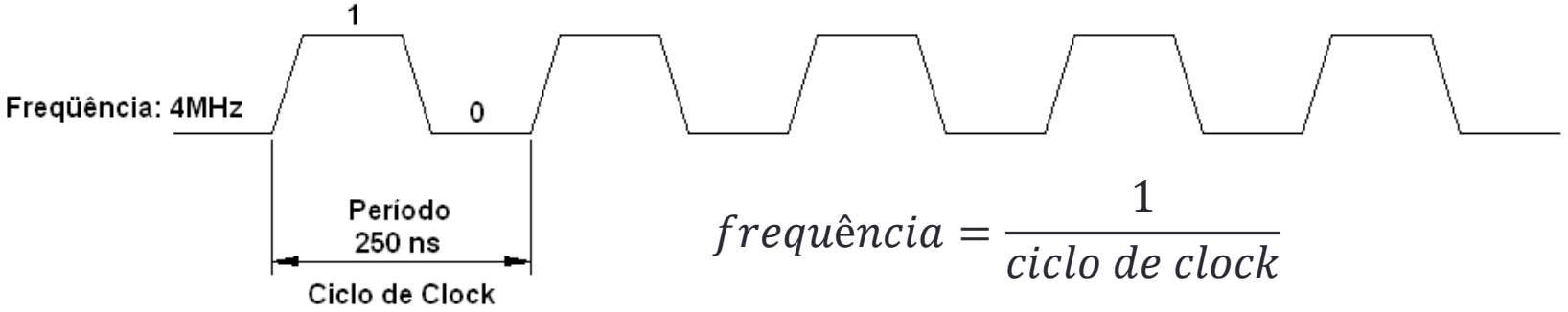
- Problema:
  - Máquina A roda um programa em 20s
  - Máquina B roda o mesmo programa em 25s
  - Como comparar os desempenhos?

Ciclos de Clock

- Para avaliarmos um tempo de execução levando em conta diferentes instruções e assim podermos caracterizar um workload, usamos o ciclo de clock por instrução:

$$\frac{segundos}{programa} = \frac{ciclos}{programa} \times \frac{segundos}{ciclo} =$$
$$= \frac{instruções}{programa} \times \frac{ciclos}{instruções} \times \frac{segundos}{ciclo}$$

- Ciclos de clock indicam quando as tarefas iniciam e terminam. O processador é um circuito síncrono.

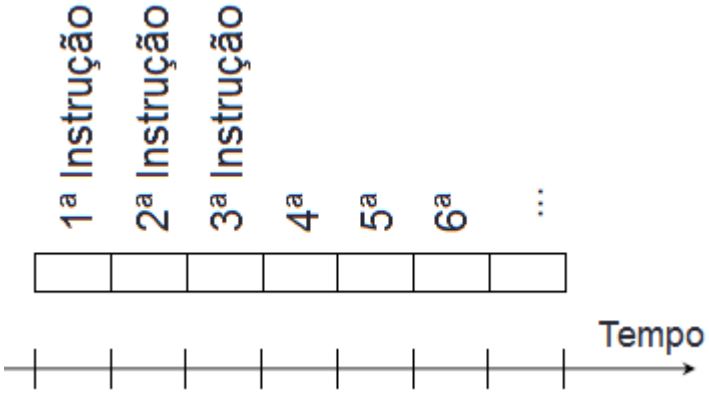


$$frequência = \frac{1}{ciclo\ de\ clock}$$

Como Melhorar o Desempenho?

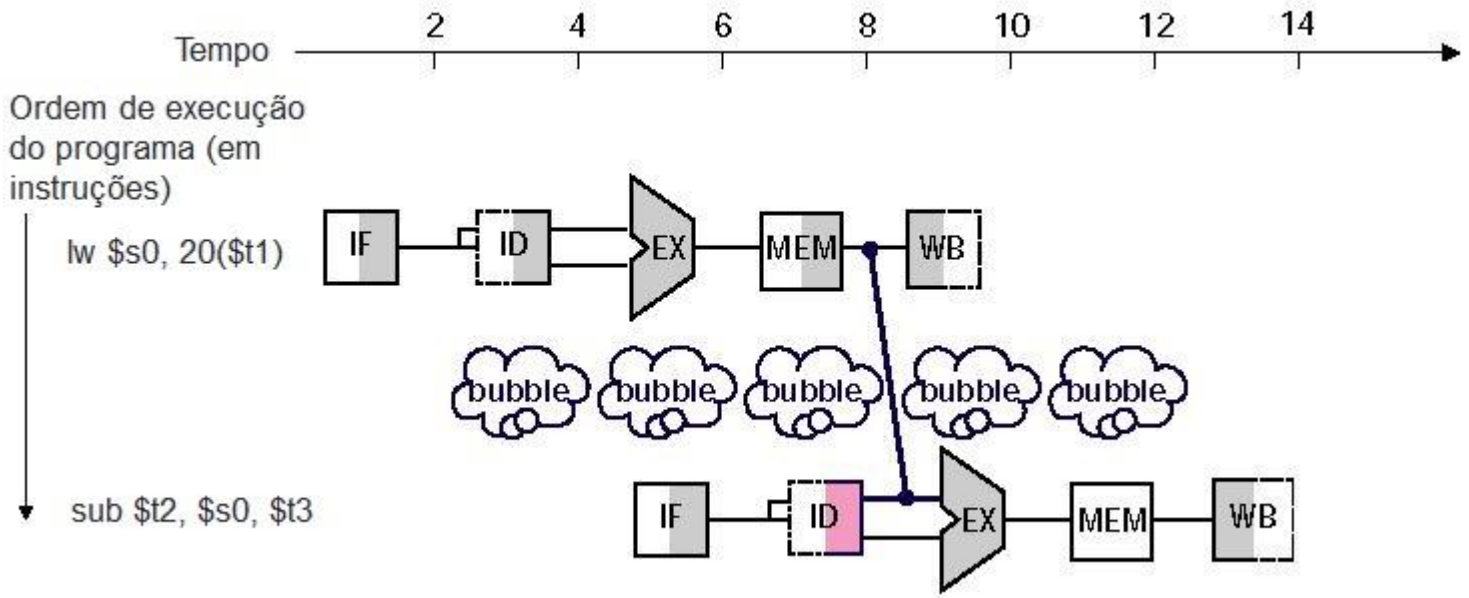
$$\frac{segundos}{programa} = \frac{ciclos}{programa} \times \frac{segundos}{ciclo}$$

- Duas formas de se reduzir o tempo de execução seria reduzir:
  - O Número de clocks requeridos pelo programa, ou
  - Tempo do ciclo de clock (ou seja, aumentar a taxa de clock!)
- Podemos assumir que número de ciclos = número de instruções?



Como Melhorar o Desempenho?

- O número de ciclos não é, necessariamente, igual ao número de instruções.
- Lembre-se que estas são **instruções de máquina**, não linhas de código em C.



Variabilidade do CPI

- Multiplicação gasta mais tempo que adição
- Operações de ponto flutuante são mais lentas que as de inteiros
- Acesso à memória gasta mais tempo que acesso a registradores
- Mudanças no tempo de ciclo podem alterar o número de ciclos exigidos para executar cada uma das instruções

Exemplo

- Um programa roda em 10s no computador A, o qual tem um clock de 400 MHz. Nós estamos tentando ajudar um projetista de computadores a construir uma nova máquina B, que irá rodar o mesmo programa em 6s. O projetista pode usar uma nova tecnologia (gastar muito mais) para aumentar substancialmente a taxa de clock. Porém, ele nos informa que isso irá afetar todo o projeto do restante da CPU, de forma que máquina B irá exigir 1.2 vezes ciclos a mais de clock que a máquina A para execução do programa. Qual taxa de clock deveria ser “buscada” pelo projetista?

Solução:

$$\frac{segundos}{programa} = \frac{instruções}{programa} \times \frac{ciclos}{instruções} \times \frac{segundos}{ciclo}$$

$$tempo = número\ de\ instruções \times CPI \times \frac{1}{freq}$$

Para o computador A:

$$10 = número\ de\ instruções \times CPI_A \times \frac{1}{400 \times 10^6}$$

$$número\ de\ instruções \times CPI_A = 10 \times 400 \times 10^6$$

Para o computador B:

$$6 = número\ de\ instruções \times CPI_B \times \frac{1}{freq_B}$$

$$6 = número\ de\ instruções \times 1{,}2 \times CPI_A \times \frac{1}{freq_B}$$

$$freq_B = \frac{1{,}2}{6} \times número\ de\ instruções \times CPI_A$$

$$freq_B = \frac{1{,}2}{6} \times 10 \times 400 \times 10^6 = 800 \times 10^6$$

$$\boldsymbol{freq_B = 800\ MHz}$$

Agora sabemos o que ciclos representam

- Um certo programa irá exigir:
  - Um certo no. de instruções (instruções de máquina)
  - Um certo no. de ciclos
  - Um certo no. de segundos
- Nós temos um vocabulário para relatar essas quantidades:
  - Tempo de ciclo (segundos / ciclo)
  - Taxa de clock (ciclos / segundo)
  - CPI (Ciclos Por Instrução) . Uma aplicação com uso intensivo de operações de ponto flutuante vai ter CPI alto
- MIPS (Milhões de Instruções Por Segundo). Poderia ser bem elevado para programas usando mais instruções simples do que mais complexas

MIPS

- MIPS pode ser calculado através da frequência em MHz e da CPI média:

$$MIPS = \frac{freq\ (MHz)}{CPI} = \frac{Clocks/_{segundo}}{Clocks/_{instruções}}$$

- Historicamente:
  - PDP-11, VAX, Intel 8086 CPI>1
  - Maquinas RISC Load/Store
    MIPS, SPARC, PowerPC, miniMIPS CPI=1
  - CPUs modernas, Pentium, Athlon CPI<1

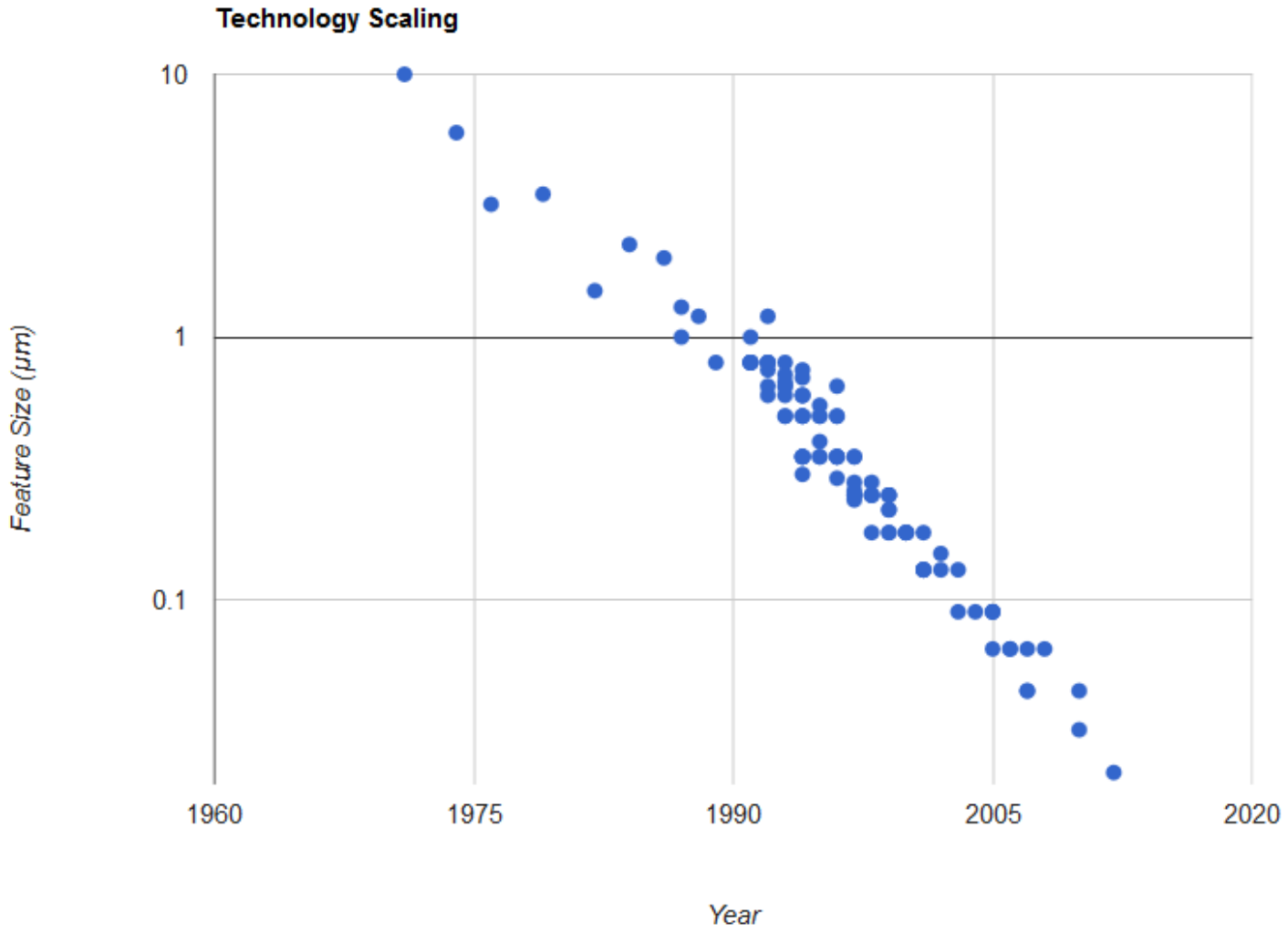
Desempenho

- Desempenho está intimamente ligado ao tempo de execução
- Quais outras variáveis também medem desempenho?
  - No. de ciclos para executar um programa?
  - No. de instruções um programa?
  - No. de ciclos por segundo?
  - No. médio de ciclos por instrução (CPI)?
  - No. médio de instruções por segundo?
- Erro comum: pensar que APENAS UMA destas variáveis é indicativa de desempenho quando ela sozinha realmente NÃO É.

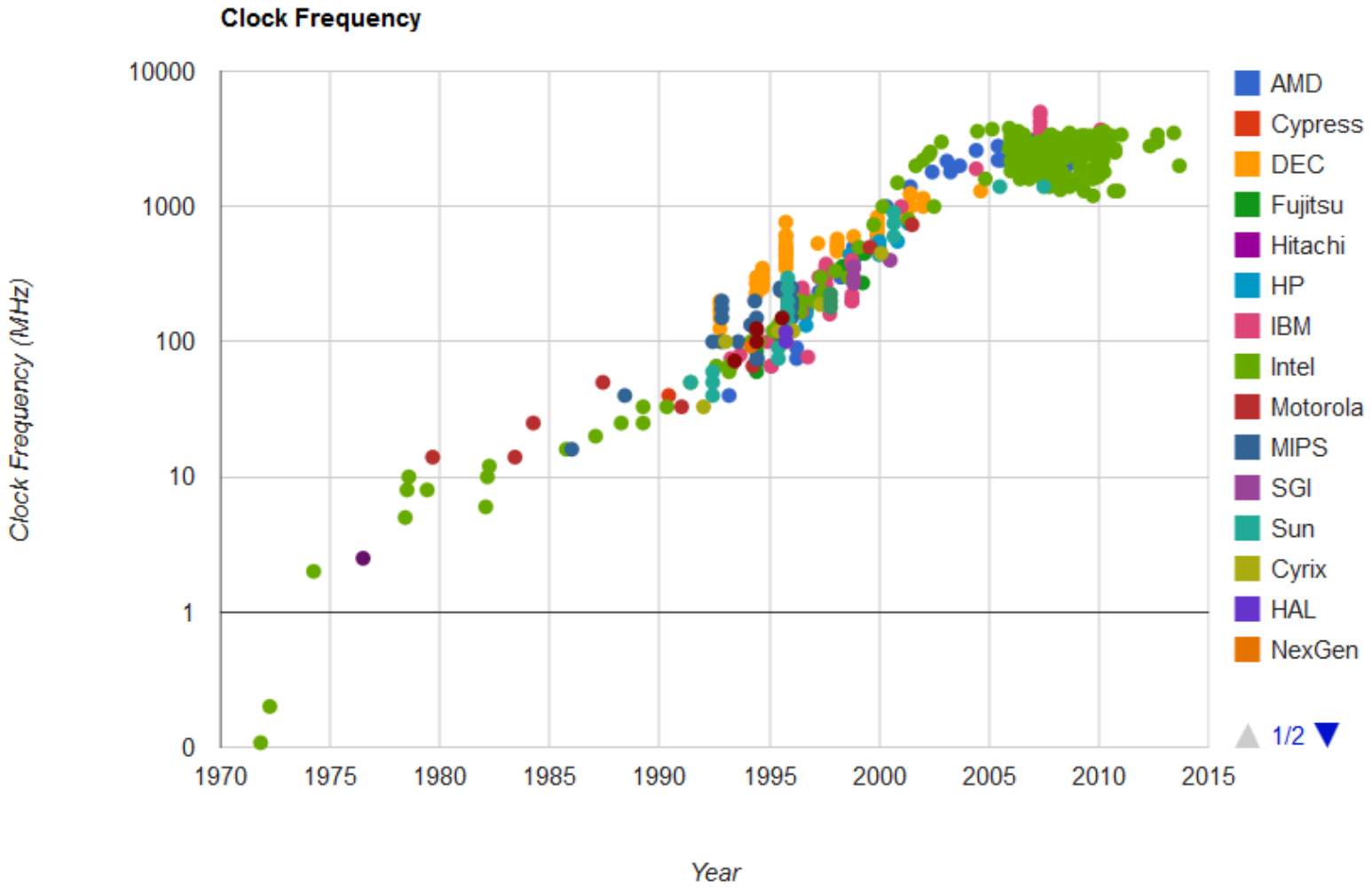
Escala e Desempenho

- Em Novembro de 1971, a Intel lançou o primeiro microprocessador em um único chip do mundo, o Intel 4004. Ele tinha 2.300 transistores, rodava com um clock de 740 KHz e executava 60.000 instruções por segundo, enquanto dissipava 0,5 W.
- Os chips microprocessadores atuais empregam bilhões de transistores, incluem múltiplos núcleos de processamento, rodam com clocks da ordem de Giga Hertz, e exibem um desempenho superior a 4 milhões de vezes ao do 4004 original.
- Computação móvel, ubíqua, sensores. Como comparar arquiteturas tão diferentes?
- Dê uma olhada em **http://cpudb.stanford.edu/**

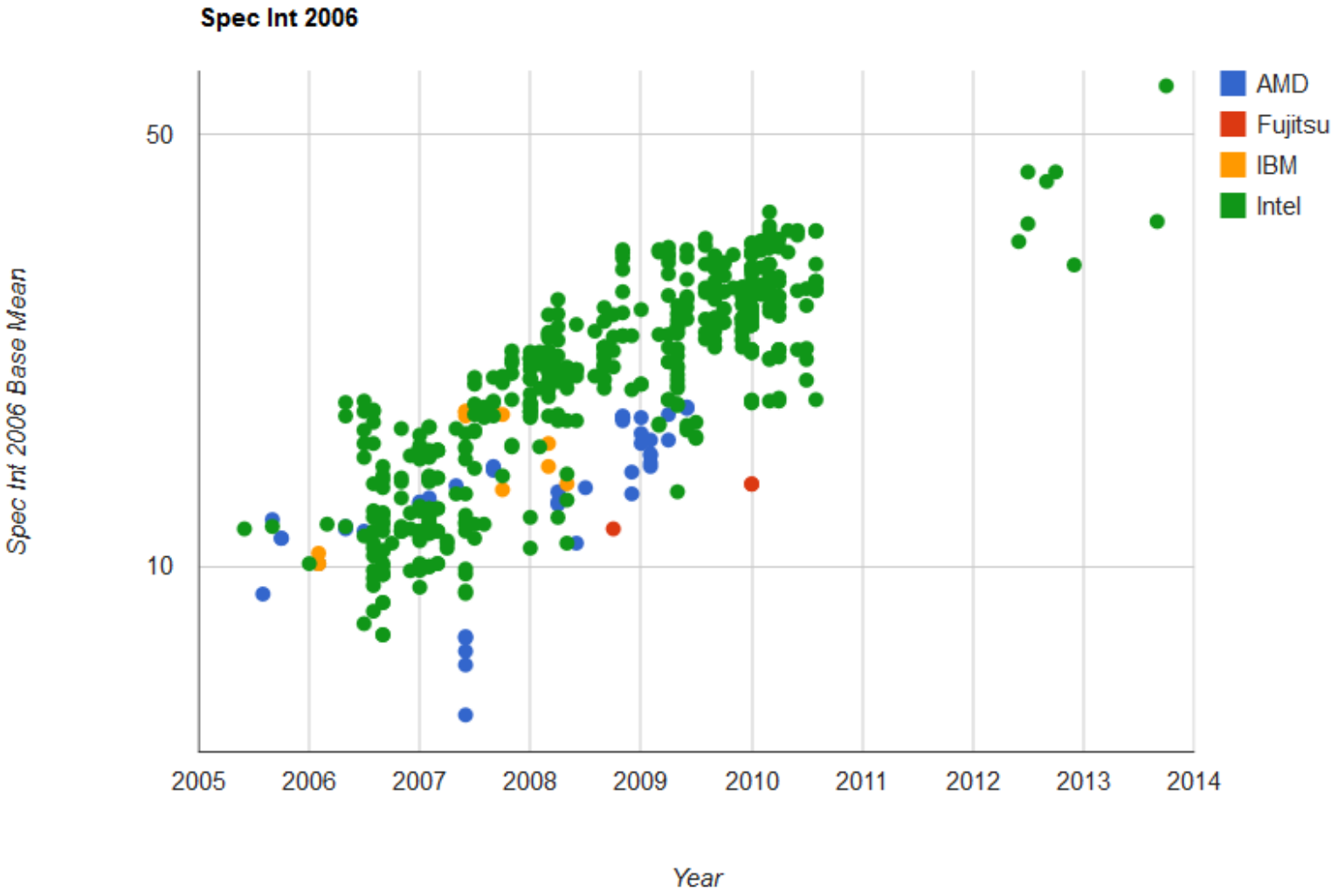
Escala da Tecnologia (CPUDB)



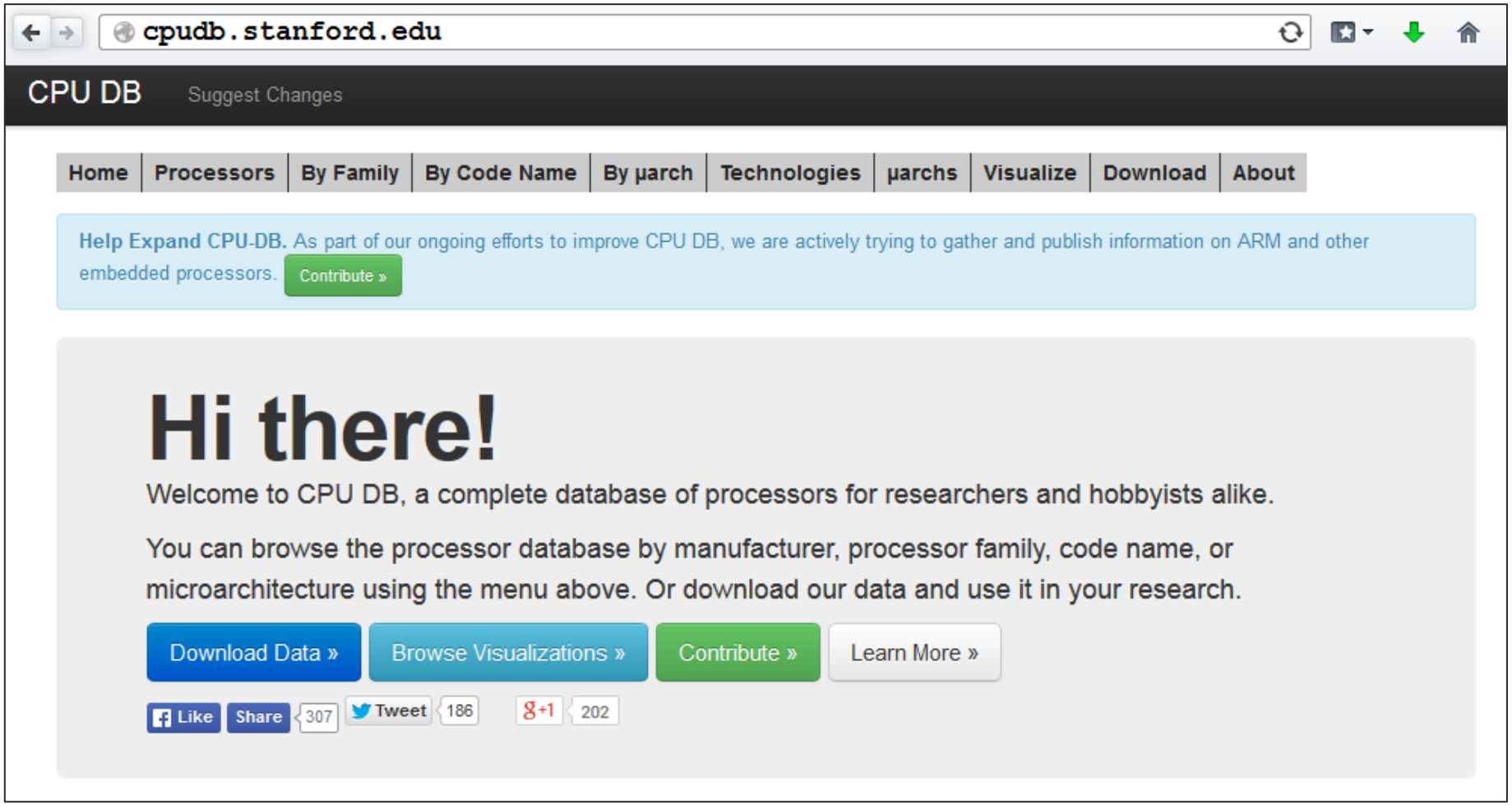
Frequência de Clock (CPUDB)



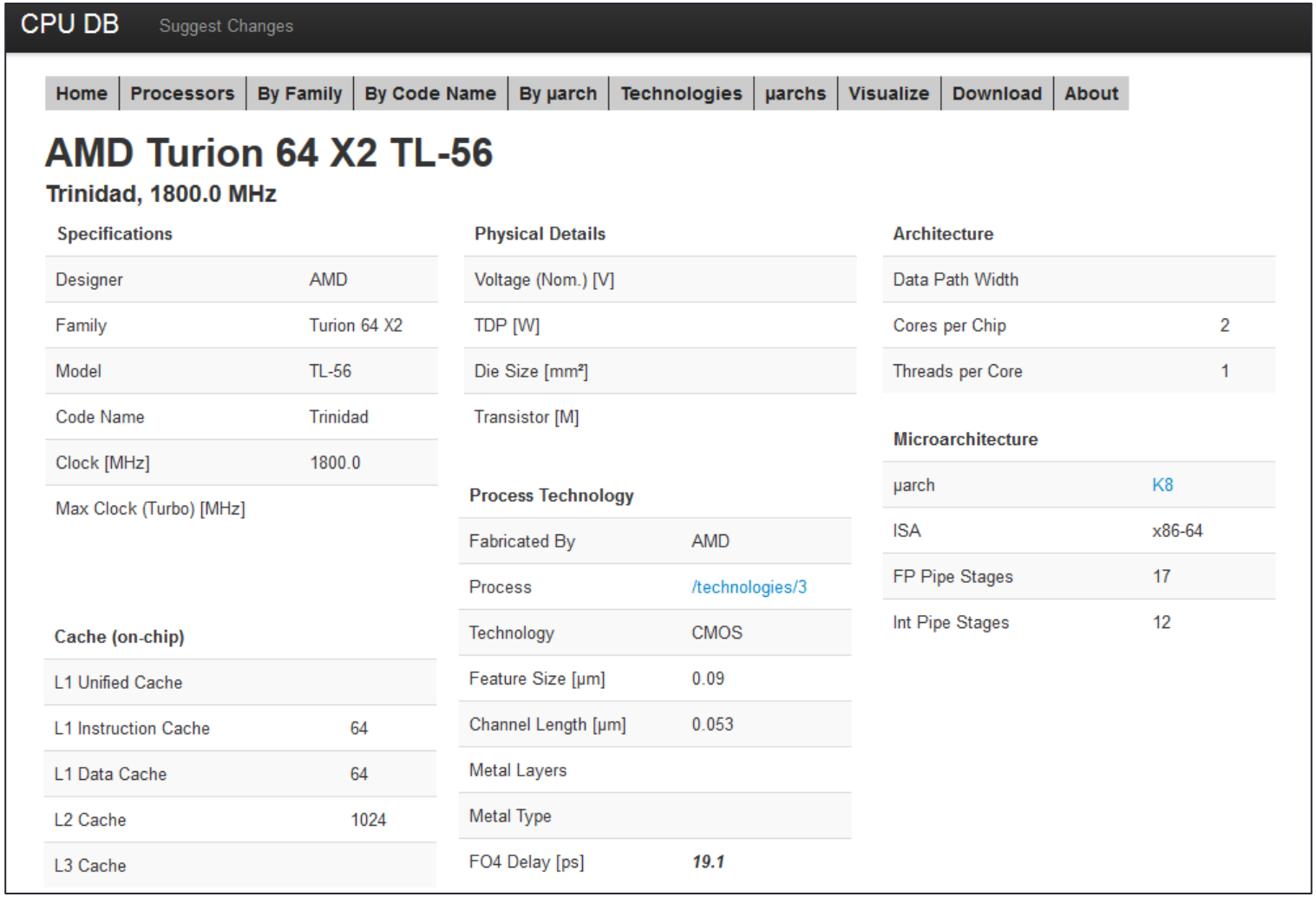
Spec Int 2006 (CPUDB)



CPUDB


cpudb.stanford.edu
CPU DB
Suggest Changes
Home
Processors
By Family
By Code Name
By µarch
Technologies
µarchs
Visualize
Download
About
Help Expand CPU-DB. As part of our ongoing efforts to improve CPU DB, we are actively trying to gather and publish information on ARM and other embedded processors.
Contribute »
Hi there!
Welcome to CPU DB, a complete database of processors for researchers and hobbyists alike.
You can browse the processor database by manufacturer, processor family, code name, or microarchitecture using the menu above. Or download our data and use it in your research.
Download Data »
Browse Visualizations »
Contribute »
Learn More »
Like
Share
307
Tweet
186
+1
202

CPUDB – Exemplo AMD Turion



CPU DB Suggest Changes

Home | Processors | By Family | By Code Name | By µarch | Technologies | µarchs | Visualize | Download | About

AMD Turion 64 X2 TL-56

Trinidad, 1800.0 MHz

Specifications

Designer	AMD
Family	Turion 64 X2
Model	TL-56
Code Name	Trinidad
Clock [MHz]	1800.0
Max Clock (Turbo) [MHz]	

Cache (on-chip)

L1 Unified Cache	
L1 Instruction Cache	64
L1 Data Cache	64
L2 Cache	1024
L3 Cache	

Physical Details

Voltage (Nom.) [V]	
TDP [W]	
Die Size [mm²]	
Transistor [M]	

Process Technology

Fabricated By	AMD
Process	/technologies/3
Technology	CMOS
Feature Size [µm]	0.09
Channel Length [µm]	0.053
Metal Layers	
Metal Type	
FO4 Delay [ps]	***19.1***

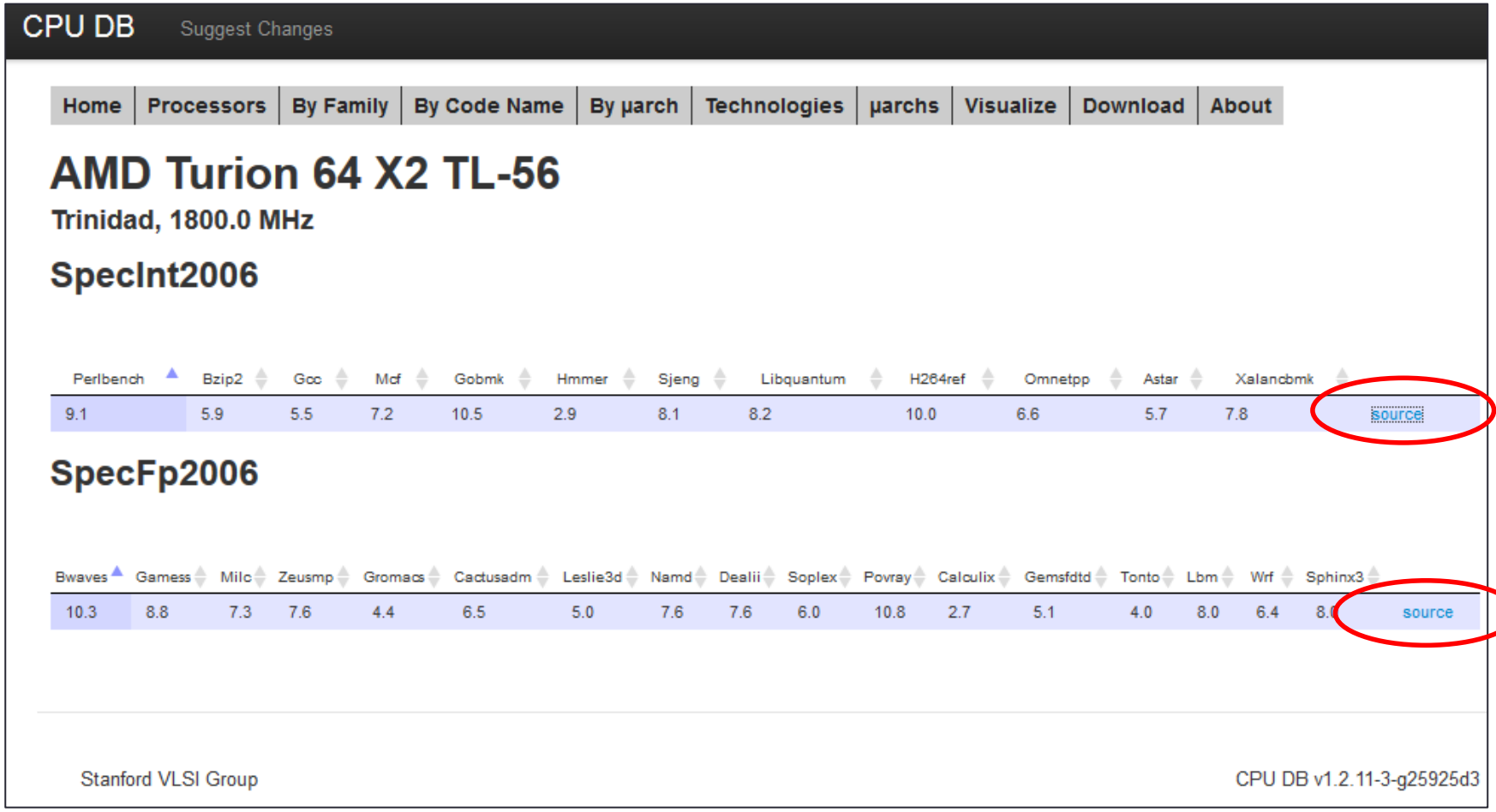
Architecture

Data Path Width	
Cores per Chip	2
Threads per Core	1

Microarchitecture

µarch	K8
ISA	x86-64
FP Pipe Stages	17
Int Pipe Stages	12

CPUDB – Exemplo AMD Turion



CPU DB Suggest Changes

Home | Processors | By Family | By Code Name | By µarch | Technologies | µarchs | Visualize | Download | About

AMD Turion 64 X2 TL-56

Trinidad, 1800.0 MHz

SpecInt2006

Perlbench	Bzip2	Gcc	Mcf	Gobmk	Hmmer	Sjeng	Libquantum	H264ref	Omnetpp	Astar	Xalancbmk	
9.1	5.9	5.5	7.2	10.5	2.9	8.1	8.2	10.0	6.6	5.7	7.8	source

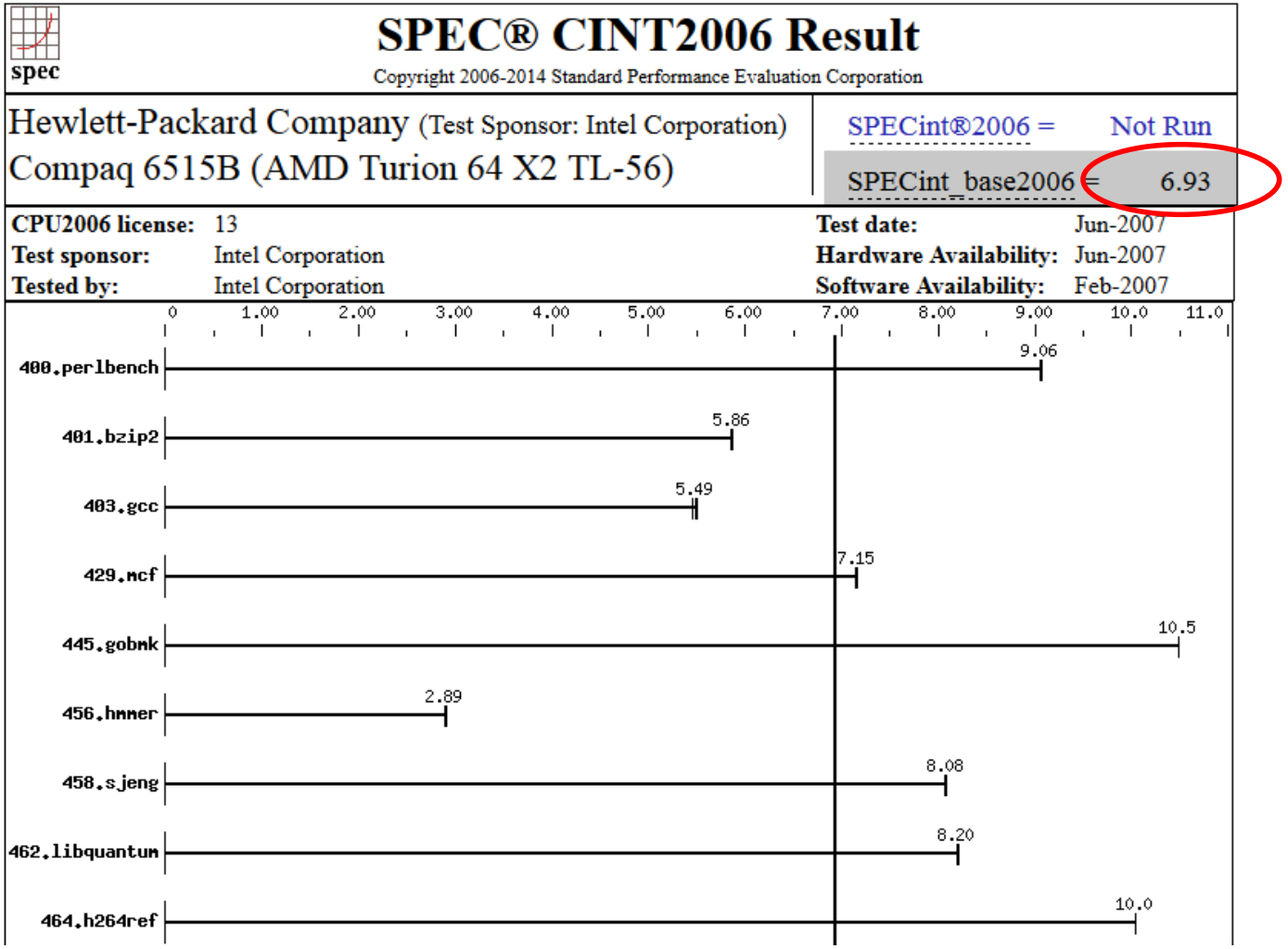
SpecFp2006

Bwaves	Gamess	Milc	Zeusmp	Gromacs	Cactusadm	Leslie3d	Namd	Dealii	Soplex	Povray	Calculix	Gemsfdtd	Tonto	Lbm	Wrf	Sphinx3	
10.3	8.8	7.3	7.6	4.4	6.5	5.0	7.6	7.6	6.0	10.8	2.7	5.1	4.0	8.0	6.4	8.0	source

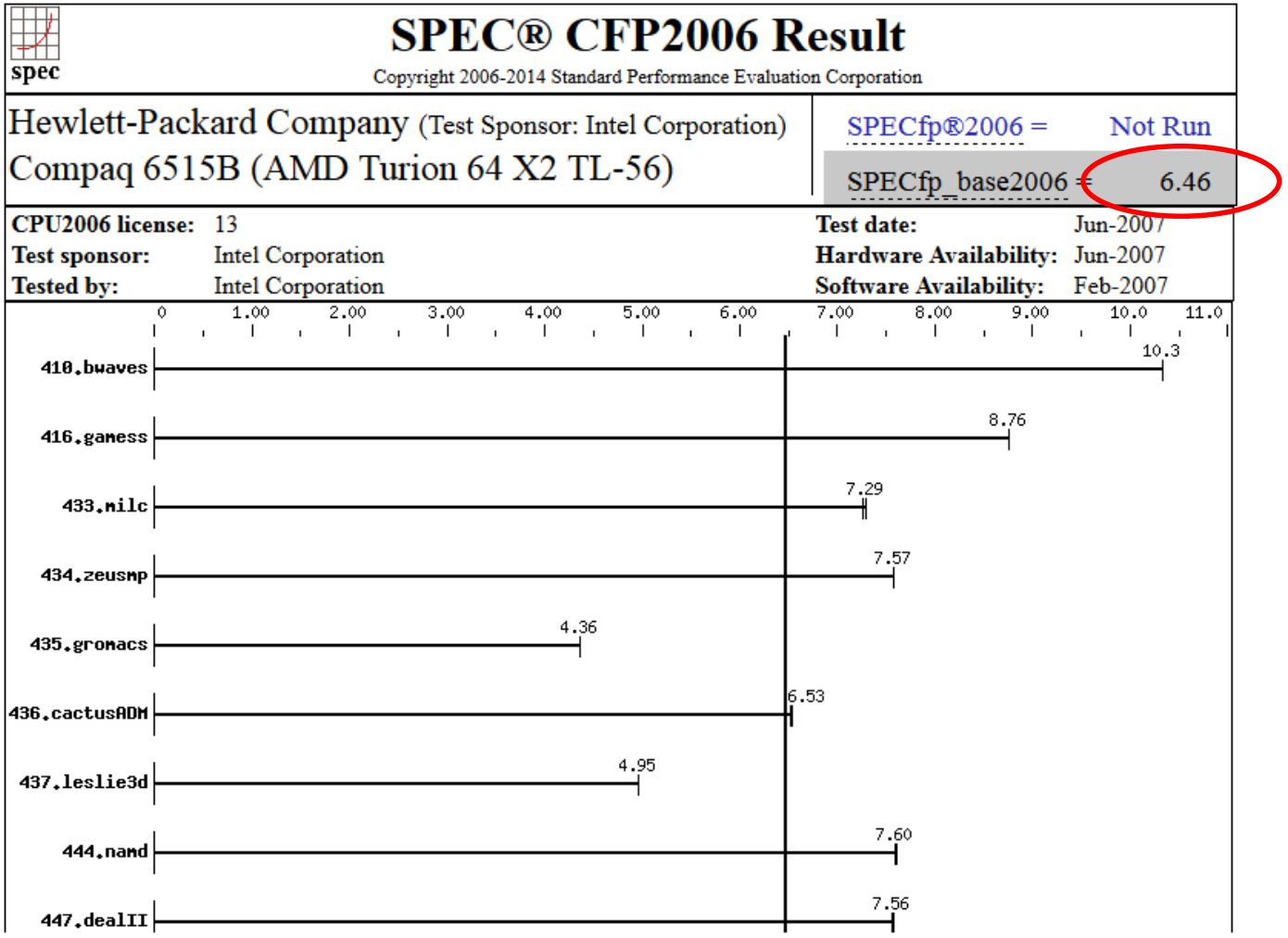
Stanford VLSI Group

CPU DB v1.2.11-3-g25925d3

CPUDB – Exemplo AMD Turion



CPUDB – Exemplo AMD Turion



Exemplo de CPI

- Suponha que existam 2 implementações possíveis para um mesmo Conjunto de Instruções (*Instruction Set Architecture* ou ISA). Para um certo programa:
  - Maq. A tem um tempo de ciclo de clock de 10 ns e um CPI de 2.0
  - Maq. B tem um tempo de ciclo de clock 20 ns e um CPI de 1.2

1. Qual máquina roda mais rápido o programa, e o quanto é mais rápida?
2. Se 2 máquinas tiverem o mesmo ISA, qual das variáveis (clock rate, CPI, tempo de execução, No.de instruções, MIPS) será sempre idêntica?

Solução

1)

$t_A = Ix2x10 = 20I$

$t_B = Ix1,2x20 = 24I$

$t_B/t_A = 24/20 = 1,2$

A maquina A é 1,2 vezes mais rápida que a maquina B

ou

A máquina A é 20% mais rápida que a maquina B

2)

Somente o número de instruções

Exemplo do Número de instruções

- Um projetista de compilador deve decidir entre 2 sequências de código para uma máquina X. Baseado na implementação do HW, existem 3 classes de instruções: Classe A, Classe B e Classe C, que requerem 1, 2 ou 3 ciclos, respectivamente.
  - A 1a. sequência de código tem 5 instruções: 2 de A, 1 de B e 2 de C
  - A 2a. sequência tem 6 instruções: 4 de A, 1 de B e 1 de C

1. Qual sequência é mais rápida? Em quanto?
2. Qual é o CPI para cada sequência?

Exemplo MIPS

- Dois diferentes compiladores estão sendo testados para uma máquina com 100MHz, com 3 diferentes classes de instruções: Classe A, Classe B e Classe C, as quais requerem 1, 2 e 3 ciclos respectivamente. Ambos compiladores são usados para produzir um código para um software grande porte.
  - O Código do 1o. compilador usa 5 Milhões de instruções Classe A, 2 Milhões de Classes B e 2 Milhões de Classes C.
  - O Código do 2o. compilador usa 10 Milhões de instruções Classe A, 1 Milhão de Classes B e 1 Milhão de Classes C.

1. Qual é a mais rápida de acordo com o tempo de execução?
2. Qual é a sequência mais rápida de acordo com o MIPS?

Solução:

CPI de cada classe:

A	B	C
1	2	3

Frequência = 100MHz

t1 = (5x1 + 2x2 + 2x3)/100 = 15/100 = 0,15 s
t2 = (10x1 + 1x2 + 1x3)/100 = 15/100 = 0,15 s

1) igual

2)
MIPS1 = (5+2+2)/0,15 = 60
MIPS2 = (10+1+1)/0,15 = 80

Exercício

Suponha uma implementação de um processador sem hardware para ponto flutuante que roda com uma frequência de 4GHz. O compilador deve implementar cada operação de ponto flutuante usando somente o conjunto de instruções de inteiros básico do processador. Na compilação de um programa que usaria única e exclusivamente 40 milhões de operações com precisão de ponto flutuante, a compilação gera um programa com o seguinte *mix* de instruções:

Tipo de Instrução	CPI	Frequência
BRANCHES/JUMPS	2	15%
ALU	3	50%
STORE	4	10%
LOAD	5	25%

Se o tempo para executar o programa compilado para esse processador for de **1,38 segundos**, quantas instruções de inteiros são necessárias, na média, para traduzir uma instrução de ponto flutuante? Justifique sua resposta.

Exercício

Considere o código abaixo:

```
        li $a0,500              # x=500
        li $a1,1000             # y=1000
        li $a2,11               # n=11
        add $v0,$a0,$a1         # soma $a0 e $a1
        move $t1,$v0            # resultado em $t1 será multiplicado por n
        li $t0,0                # $t0=0
        li $v0,0                # $v0=0
loop:   beq $a2,$t0,exit        # $se $t0==$a2 vai para exit
        add $v0,$v0,$t1         # soma $t1 n vezes
        addi $t0,$t0,1          # $t0=$t0+1
        j loop                  # jump incondicional para loop
exit:   move $a0,$v0            # coloca resultado em $a0 para ser impresso
        li $v0,1                # imprime resultado
        syscall                 # syscall código 1
        li $v0,10               # fim
        syscall                 # syscall código 10
```

Exercício

Verifique qual é o CPI médio desse programa, considerando a tabela abaixo:

Instrução	Ciclos de Clock
Todas as instruções da ALU	1,0
load/store	1,4
Desvios condicionais seguidos	2,0
Desvios condicionais não seguidos	1,5
Saltos incondicionais	1,2

Benchmarks

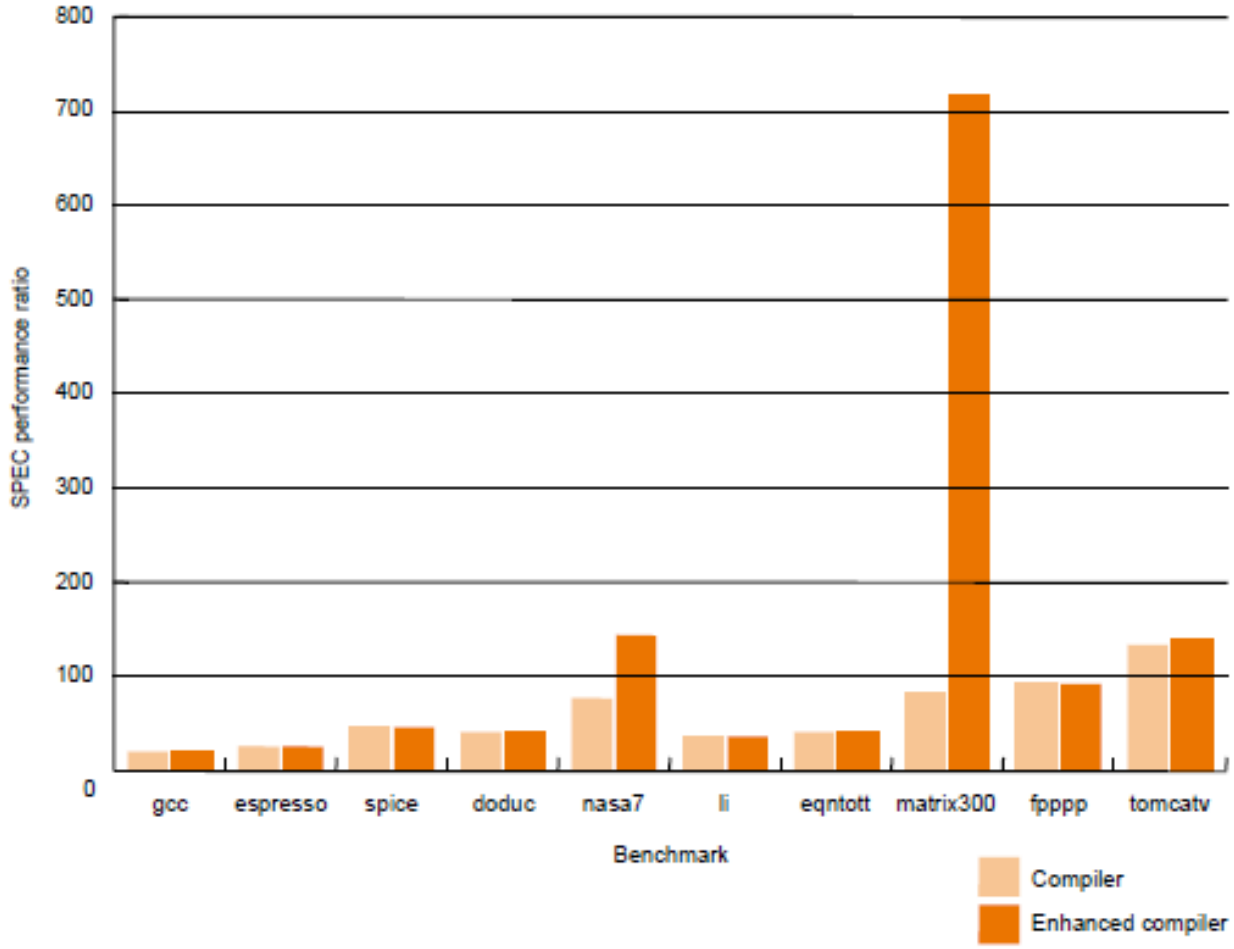
- Desempenho é melhor estimado usando-se uma aplicação real
  - Usar programas com workloads típicos
  - Ou, usar classes de aplicações esperadas típicas, i.e. compiladores, editores, aplicações científicas, gráficas, etc.
- Pequenos benchmarks
  - Interessantes para arquitetos e designers, simples para padronizar e seu uso é livre por todos. Ex.: Sieve de Eratosthenes, bom para comparar o tempo de execução de vários compiladores no mesmo computador.
- SPEC (System Performance Evaluation Cooperative)
  - "...founded in 1988 by a small number of workstation vendors who realized that the marketplace was in desperate need of realistic, standardized performance tests" (http://www.spec.org/spec/spec.html)
  - 1ª geração SPEC CPU89
  - Fabricantes entraram num acordo para definir um conjunto de programas e entradas reais para avaliação
  - Usado livremente por todos
  - Indicador valioso de desempenho (e tecnologia de compilação)

Benchmarks

- SPEC CPU2006 (Standard Performance Evaluation Corporation)
  - URL: http://www. spec.org/cpu2006/results/
  - 5ª geração de pacotes SPEC
  - Aplicações com uso intensivo do processador
  - 17 programas FP (C, C++, Fortran) e 12 INT (C, C++)
- Outros SPECs [http://www.spec.org/spec/]:
  - SPECjvm98: Java
  - SPECweb99: servidores WWW
  - SPECmail2001: servidor de correio eletrônico

SPEC'89

- Melhorias no compilador vs desempenho

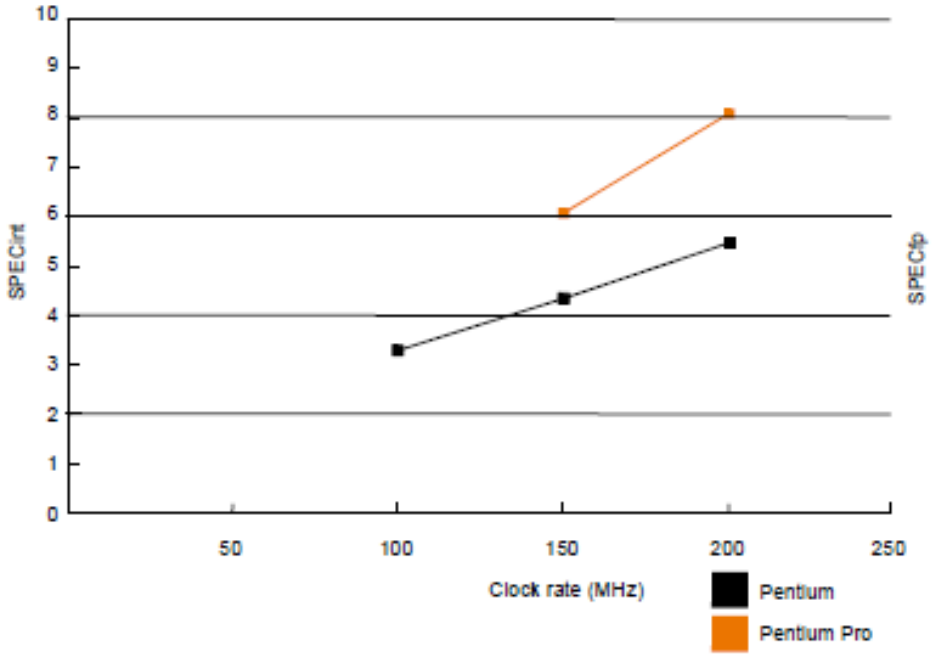


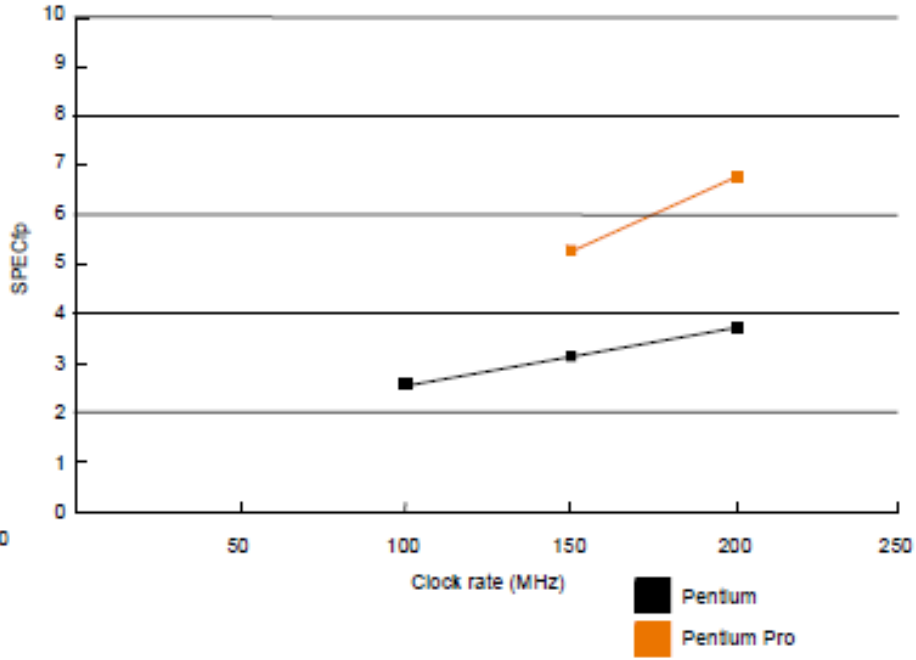
SPEC'95

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

- Dobrar a taxa de clock implica em dobrar o desempenho?
- É possível que uma máquina com um clock mais baixo tenha desempenho superior a uma máquina com clock mais alto?





Benchmarks para Sistemas Embarcados

“The Embedded Microprocessor Benchmark Consortium (EEMBC) develops benchmark software to help system designers select the optimal processors, and benchmark tools to help consumers and IT professionals select the appropriate smart phones/tablets and networking firewall appliances. EEMBC organizes its benchmark suites targeting Automotive, Digital Media, Java, Multicore Processors, Networking, Office Automation, Signal Processing, Smartphones/Tablets and Browsers.”

(http://www.eembc.org/)

Desempenho: Leis Conhecidas

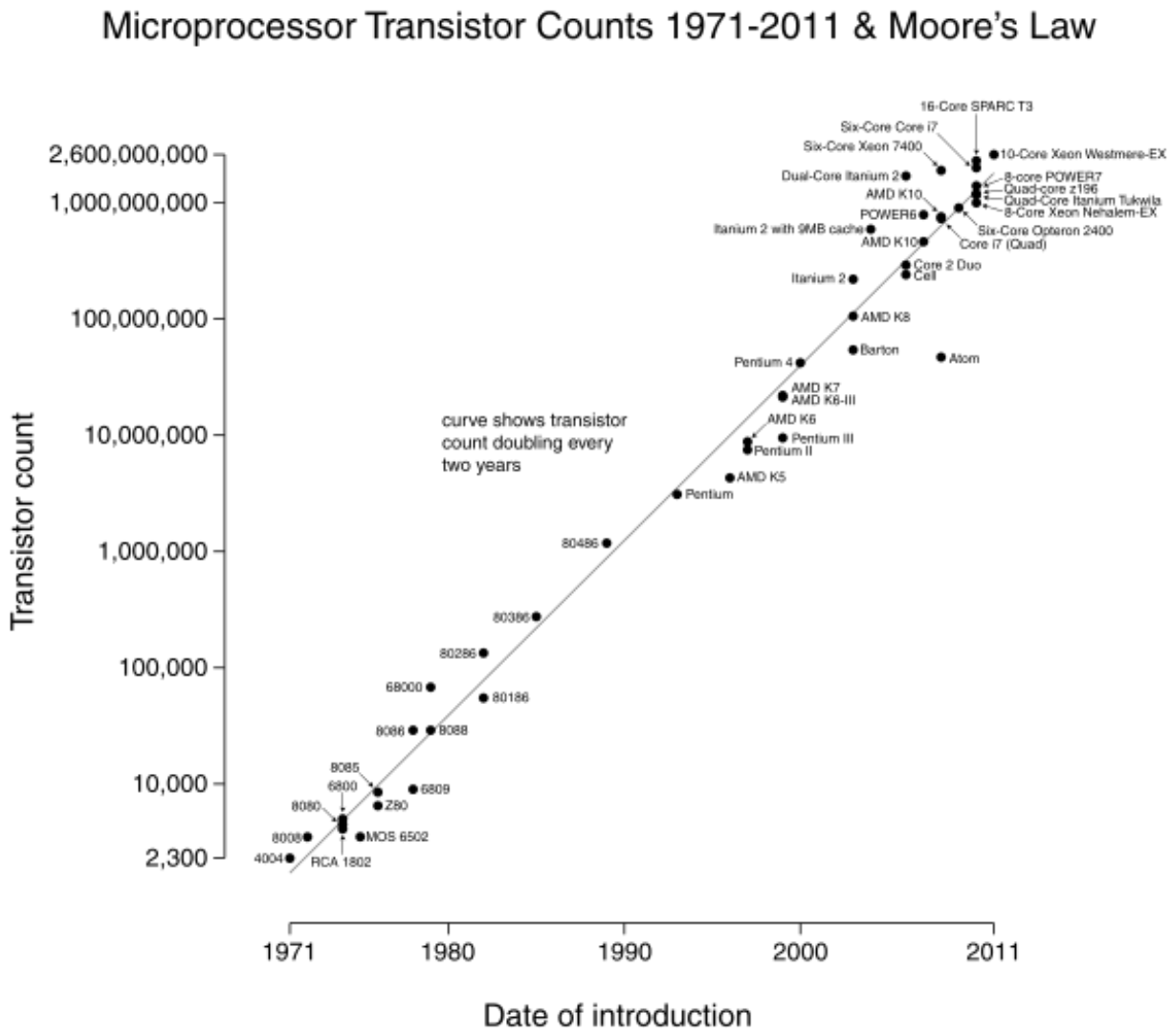
- Durante os anos, várias “leis” foram estabelecidas para se tentar prever o impacto dos ganhos tecnológico nas futuras gerações de arquitetura. Duas das mais importantes:

1. Lei de Moore
2. Lei de Amdahl

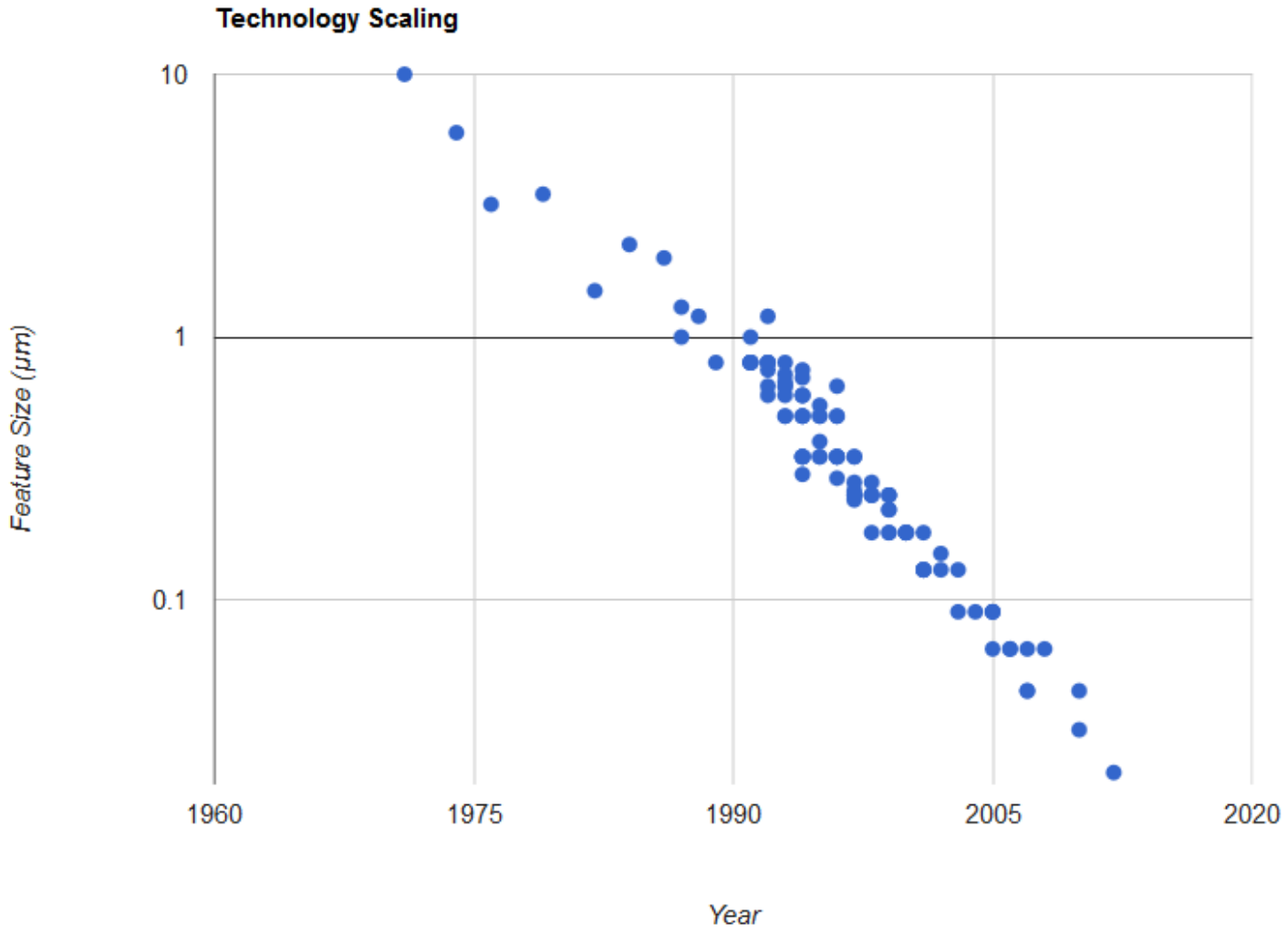
Lei de Moore

- Aumento na densidade de componentes no chip
  - Relação indireta com o desempenho
  - Gordon Moore: cofundador da Intel
- Número de transistores num chip dobrará a cada ano
  - Desde os anos 70, o desenvolvimento tem sido um pouco mais lento
  - No. de transistores 2x a cada 18 meses (1,5 ano)
  - Segundo a wikipedia, aproximadamente a cada 2 anos
- Custo do chip tem se mantido o mesmo
  - Maior densidade “empacotamento”. Menor distância para sinais elétricos. Maior velocidade da lógica. Desempenho maior.
  - Menor tamanho. Maior flexibilidade
  - Menos conexões. Mais confiabilidade
- A lei de Moore passou a ser usada como guia na indústria de semicondutores no planejamento a longo prazo e como meta na pesquisa e desenvolvimento.

Lei de Moore


Microprocessor Transistor Counts 1971-2011 & Moore's Law
Transistor count
2,600,000,000
1,000,000,000
100,000,000
10,000,000
1,000,000
100,000
10,000
2,300
curve shows transistor count doubling every two years
16-Core SPARC T3
Six-Core Core i7
Six-Core Xeon 7400
10-Core Xeon Westmere-EX
Dual-Core Itanium 2
8-core POWER7
AMD K10
Quad-core z196
Quad-Core Itanium Tukwila
POWER6
8-Core Xeon Nehalem-EX
Itanium 2 with 9MB cache
Six-Core Opteron 2400
AMD K10
Core i7 (Quad)
Core 2 Duo
Cell
Itanium 2
AMD K8
Barton
Atom
Pentium 4
AMD K7
AMD K6-III
AMD K6
Pentium III
Pentium II
AMD K5
Pentium
80486
80386
80286
68000
80186
8086
8088
8085
6800
6809
8080
Z80
8008
MOS 6502
4004
RCA 1802
1971
1980
1990
2000
2011
Date of introduction

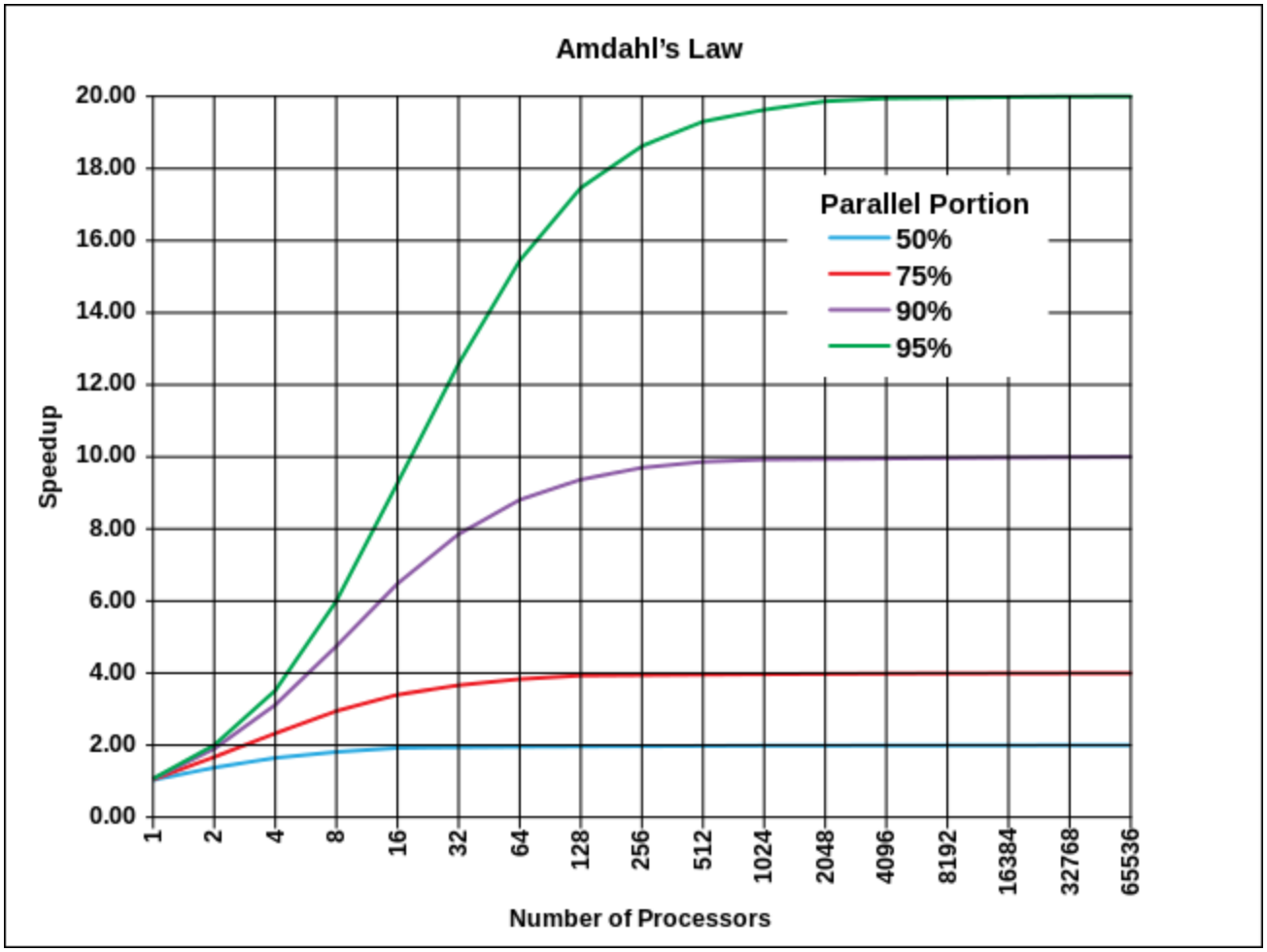
Escala


Technology Scaling
10
1
0.1
Feature Size (μm)
1960
1975
1990
2005
2020
Year

Lei de Amdahl

- Gene Amdahl (1967)
- É usada para se encontrar a máxima melhoria no desempenho de um sistema quando a melhoria se aplica a somente parte do sistema
- É muito usada em computação paralela para se prever o ganho máximo (speedup) usando-se múltiplos processadores
- O speedup de um programa usando múltiplos processadores é limitado pela fração do programa que só pode ser executada sequencialmente

Lei de Amdahl



Fonte: wikipedia

Lei de Amdahl

Calculo do speedup:

$$speedup = \frac{desempenho\ total\ com\ a\ melhoria}{desempenho\ original\ sem\ a\ melhoria}$$

Ou, alternativamente:

$$speedup = \frac{tempo\ original\ sem\ a\ melhoria}{tempo\ total\ com\ a\ melhoria}$$

O cálculo do *speedup* geralmente envolve duas grandezas:

- **Speedup da melhoria**: é a aceleração somente do trecho melhorado em relação ao seu tempo original. Por exemplo: Com uma nova unidade de aceleração de operações de ponto flutuante, a execução desse tipo de operação em um determinado programa passou de 5 segundos para 2 segundos. Assim: $speedup_{melhoria} = {}^{5}/_{2} = 2{,}5$
- **Fração da melhoria**: é a fração do tempo de computação na máquina original que pode ser convertida para tirar proveito da melhoria. Por exemplo: 20 segundos do tempo de execução de um programa que leva 60 segundos é gasto exclusivamente com operações de ponto flutuante. Se melhorarmos a unidade de ponto flutuante, a fração afetada será de: $fração_{melhoria} = {}^{20}/_{60} = 0{,}333$
- A $fração_{melhoria}$ é sempre menor do que 1 e o $speedup_{melhoria}$ é sempre maior que 1.

Lei de Amdahl

- Com a definição de $fração_{melhoria}$ e $speedup_{melhoria}$, temos as seguintes equações muito úteis:

$$tempo\ exec_{novo} = tempo\ exec_{antigo} \times \left((1 - fração_{melhoria}) + \frac{fração_{melhoria}}{speedup_{melhoria}} \right)$$

$$speedup_{global} = \frac{tempo\ exec_{antigo}}{tempo\ exec_{novo}} = \frac{1}{(1 - fração_{melhoria}) + \frac{fração_{melhoria}}{speedup_{melhoria}}}$$

Exemplo 1

Suponha que estejamos considerando um aperfeiçoamento para o processador de um sistema servido para Web. A nova CPU é 10 vezes mais rápida em computação na aplicação do serviço Web que o processador original. Supondo que a CPU original esteja ocupada com a computação 40% do tempo e que fique esperando por E/S durante 60% do tempo, qual será o *speedup* global obtido com a troca do processador?

Solução:

Dados:

$$fração_{melhoria} = 40\% = 0{,}4$$

$$speedup_{melhoria} = 10$$

Speedup global:

$$speedup_{global} = \frac{1}{(1 - 0{,}4) + \frac{0{,}4}{10}} = \frac{1}{0{,}6 + \frac{0{,}4}{10}} = \frac{1}{0{,}64} \approx 1{,}56$$

Exemplo 2

Suponha que um programa roda em 100s em uma máquina, com operações de multiplicação responsáveis por 80s deste tempo. Quanto se deve aumentar a “velocidade da multiplicação” para que o programa rode 4 vezes mais rápido? E para fazê-lo rodar 5 vezes mais rápido?

Solução:

- Se speedup global for 4:

$$4 = \frac{1}{(1 - 0{,}8) + \frac{0{,}8}{speedup_{melhoria}}} = \frac{1}{0{,}2 + \frac{0{,}8}{speedup_{melhoria}}}$$

$$0{,}2 + \frac{0{,}8}{speedup_{melhoria}} = \frac{1}{4} = 0{,}25$$

$$\frac{0{,}8}{speedup_{melhoria}} = 0{,}05$$

$$speedup_{melhoria} = \frac{0{,}8}{0{,}05} = 16$$

Solução

- Se speedup global for 5: impossível! O speedup da melhoria tenderia ao infinito. Veja:
- Com um speedup global de 5, significa que o tempo original de 100 segundos deve ser dividido por 5, ou seja, o tempo de execução final deverá ser 20s. Mas como existe uma parte do código que não é afetado pela melhoria e essa parte já executa em 20s, a outra parte, afetada pela melhoria, que originalmente executa em 80s, deverá executar em ZERO segundos. Podemos ver isso na equação:

$$speedup_{global} = \frac{1}{0{,}2 + \frac{0{,}8}{speedup_{melhoria}}}$$

Solução:

$$speedup_{global} = \frac{1}{0{,}2 + \frac{0{,}8}{speedup_{melhoria}}}$$

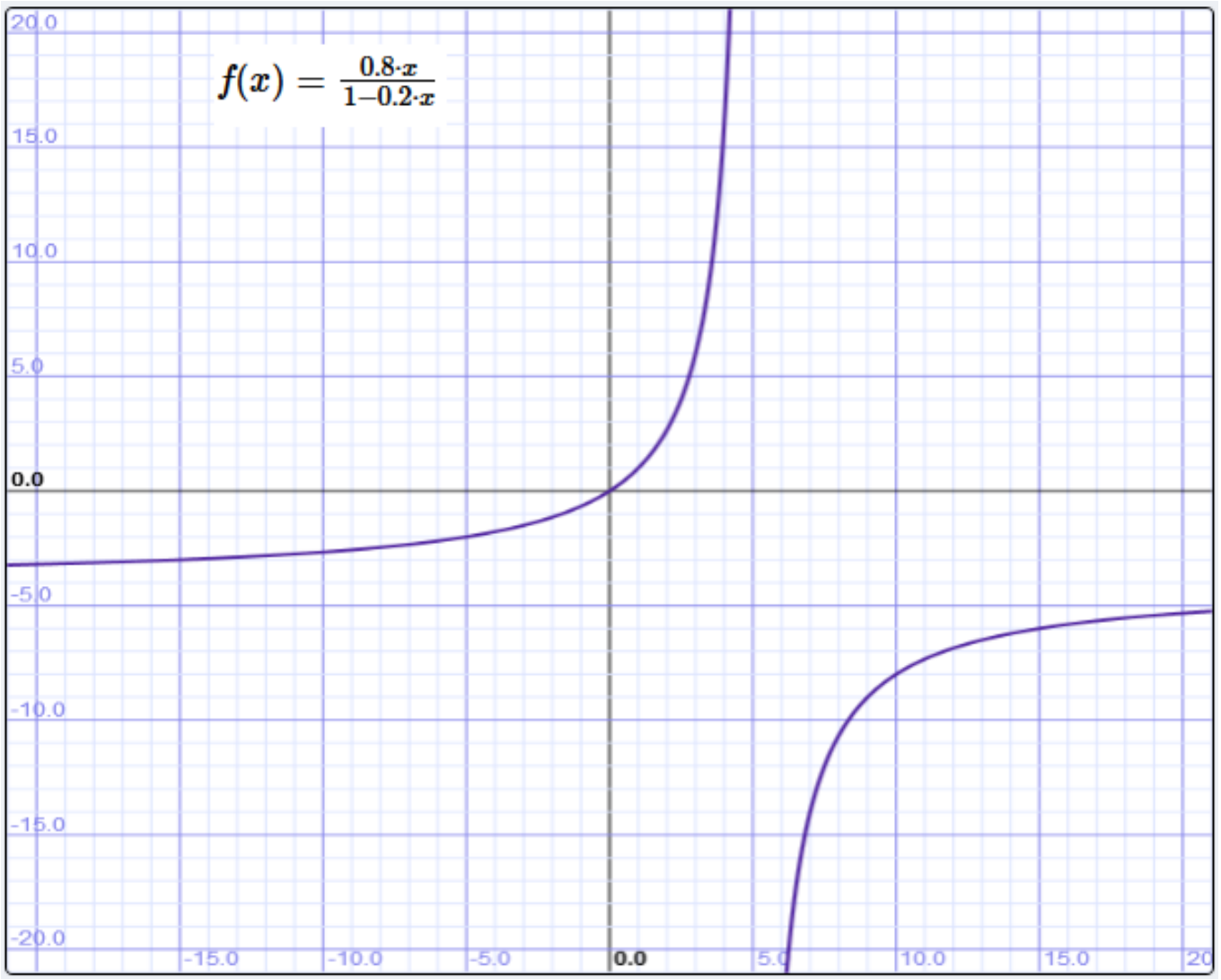
$$x = \frac{1}{0{,}2 + \frac{0{,}8}{y}}$$

$$0{,}2 + \frac{0{,}8}{y} = \frac{1}{x}$$

$$\frac{0{,}8}{y} = \frac{1}{x} - 0{,}2$$

$$\frac{0{,}8}{y} = \frac{1 - 0{,}2x}{x}$$

$$\boxed{y = \frac{0{,}8x}{1 - 0{,}2x}}$$



Lei de Amdahl e Princípio da Localidade

- Um princípio fundamental da Lei de Amdahl é o seguinte:

Tornar "o que é mais comum" o mais rápido possível!

- Princípio da Localidade:
  - Uma propriedade importante que exploramos nos sistemas é o chamado princípio de localidade, que diz que os programas tendem a reutilizar dados e instruções que usaram recentemente. Uma regra prática aceita amplamente é a regra 90/10. Um programa gasta 90% do seu tempo de execução em apenas 10% do código.
  - Assim pode-se prever as instruções que um programa vai executar com base na sua execução passada.
  - O princípio de localidade pode ser levado em conta na análise da Lei de Amdahl

Mais Exemplos

- Suponha que queremos que uma máquina execute todas as instruções de ponto-flutuante 5x mais rápido que a sua versão anterior. Se o tempo de execução de um *benchmark* antes do ganho na parte de ponto-flutuante é igual a 10s, qual será o *speedup* se o programa gasta a metade dos 10s total executando instruções de ponto-flutuante?
- Reflexão: Qual o *speedup* máximo que poderia ser alcançado por esse tipo de melhoria, considerando esse benchmark?

Solução:

$$speedup_{global} = \frac{1}{(1 - 0{,}5) + \frac{0{,}5}{5}} = \frac{1}{0{,}5 + 0{,}1} = \frac{1}{0{,}6} \approx 1{,}667$$

Speedup máximo:

- O speedup global máximo é encontrado considerando que a parte melhorada vai executar em tempo zero. Neste caso o máximo seria 2.
- É claro que nada pode ser realizado em tempo zero. Mas suponha o speedup da melhoria seja 1000.

$$speedup_{global} = \frac{1}{(1 - 0{,}5) + \frac{0{,}5}{1000}} = \frac{1}{0{,}5 + 0{,}0005} = \frac{1}{0{,}5005}$$

$$\approx 1{,}998$$

Mais Exemplos

- Nós agora usamos um benchmark para avaliar a unidade ponto-flutuante descrita anteriormente e queremos que o *benchmark* global verifique um *speedup* de 3. O *benchmark* que estamos considerando roda em 100s com o antigo hardware de ponto-flutuante. Quanto do tempo de execução (percentual) deveria estar associado a operações ponto-flutuante de forma a alcançar o *speedup* desejado com o uso deste *benchmark*?

Solução:

$$speedup_{global} = \frac{1}{(1 - fração_{melhoria}) + \frac{fração_{melhoria}}{speedup_{melhoria}}}$$

$$3 = \frac{1}{(1 - fração_{melhoria}) + \frac{fração_{melhoria}}{5}} = \frac{1}{1 - \frac{4fração_{melhoria}}{5}}$$

$$3 = \frac{5}{5 - 4fração_{melhoria}}$$

$$5 - 4fração_{melhoria} = \frac{5}{3}$$

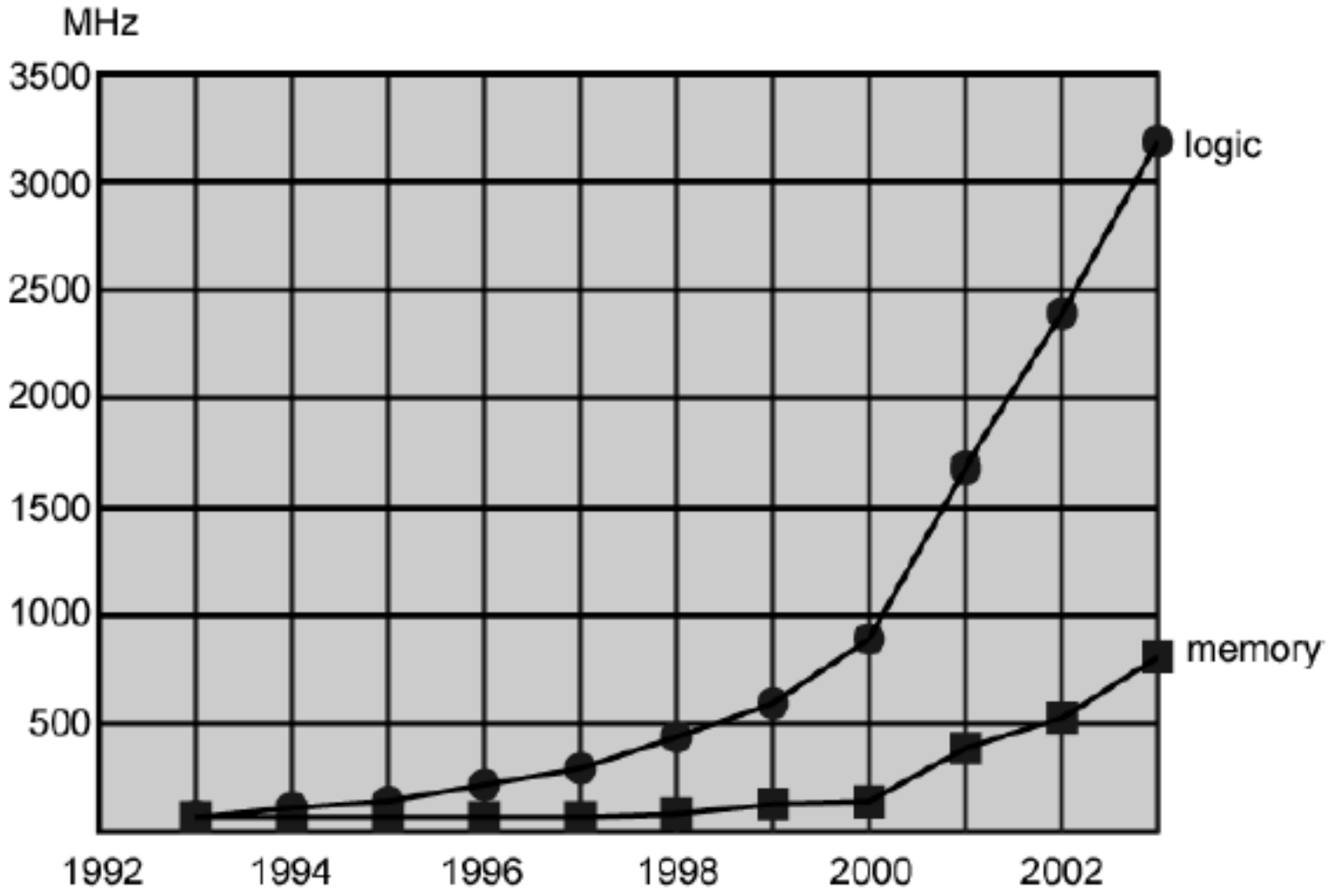
$$4fração_{melhoria} = 5 - \frac{5}{3} = \frac{10}{3}$$

$$fração_{melhoria} = \frac{10}{12} = 0{,}8333 = 83{,}3\%$$

Gargalo de desempenho

- Velocidade do Processor aumenta
- Capacidade da Memória aumenta
- Ganho de velocidade da memória << Ganho da velocidade da CPU
- Conhecido como **Gargalo de von Neumann**
- Outros fatores de impacto (menos dependentes do HW)
  - Compilador
  - SO (multitarefas, escalonamento, etc…)
  - Máquina virtual
  - I/O

CPU vs Memória



Fonte: STALLINGS, W. Arquitetura de Computadores, 5ª Edição, Prentice Hall do Brasil.

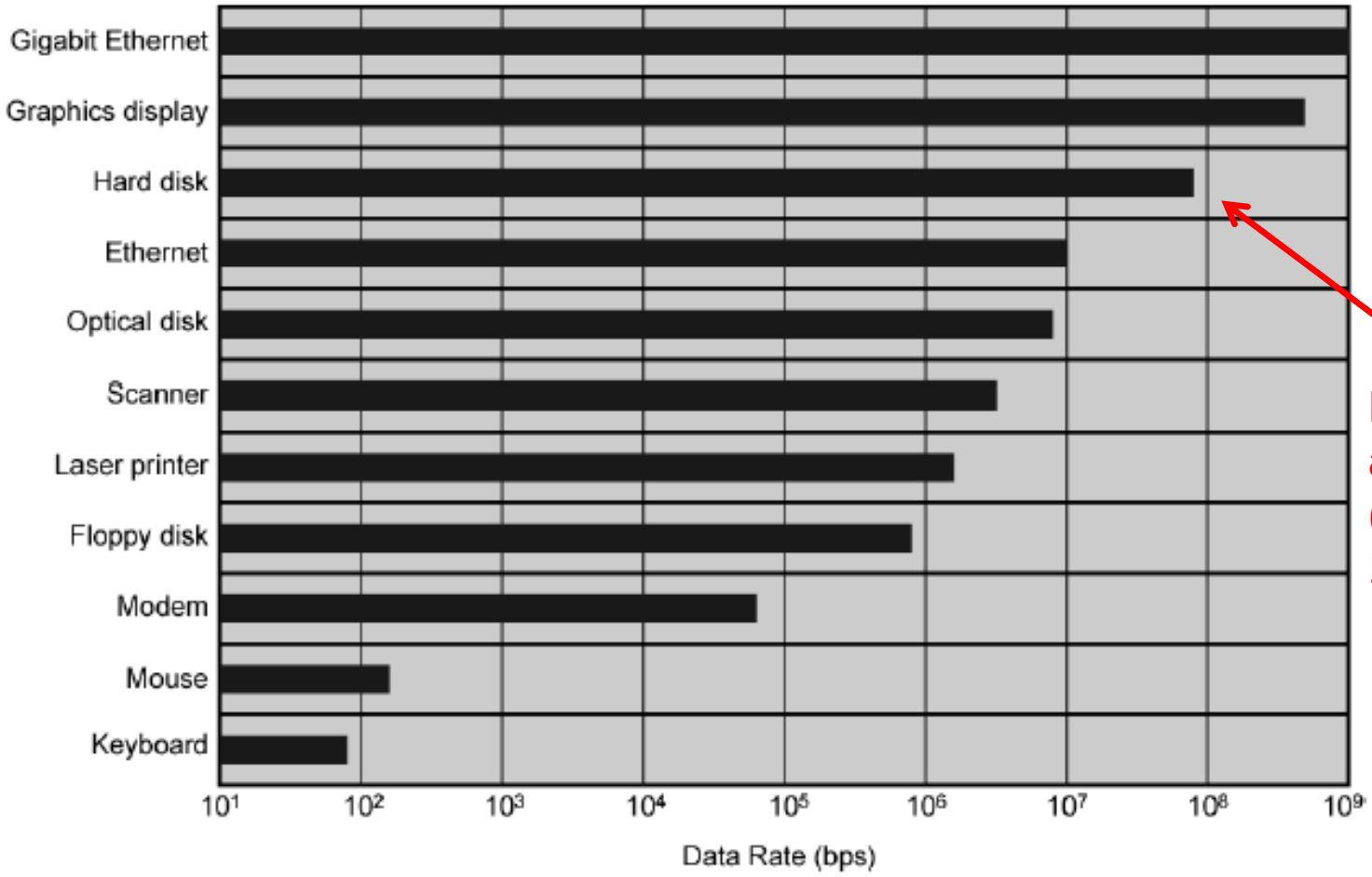
Soluções (para o gargalo de Von Newmann)

- Aumentar o número de bits recuperados / unidade de tempo
  - Lembram-se da vazão (throughput)?
  - Fazer a DRAM ficar “mais larga” ao invés de “mais profunda”
- Mudar a interface com a DRAM
  - Caches (vários níveis)
- Reduzir a frequência de acessos à memória
  - Caches mais complexas e cache on chip
  - Aumentar a banda de interconexão
  - Barramentos de alta velocidade
  - Hierarquia de barramentos

Dispositivos de I/O

- Periféricos com demandas intensivas de I/O
- Altíssima vazão de dados
- Processadores poderiam manipular isso (ex. placas gráficas)
- Movimentação de dados é um problema
- Soluções:
  - Caching
  - Buffering
  - Barramentos de alta velocidade
  - Estruturas de barramentos mais elaboradas
  - Variação na configuração de processadores
  - DMA

Taxas típicas de dados de dispositivos I/O



Hoje SATA já atingiu 6Gbps (Ver 3.0) e 16Gbps (Ver 3.2)

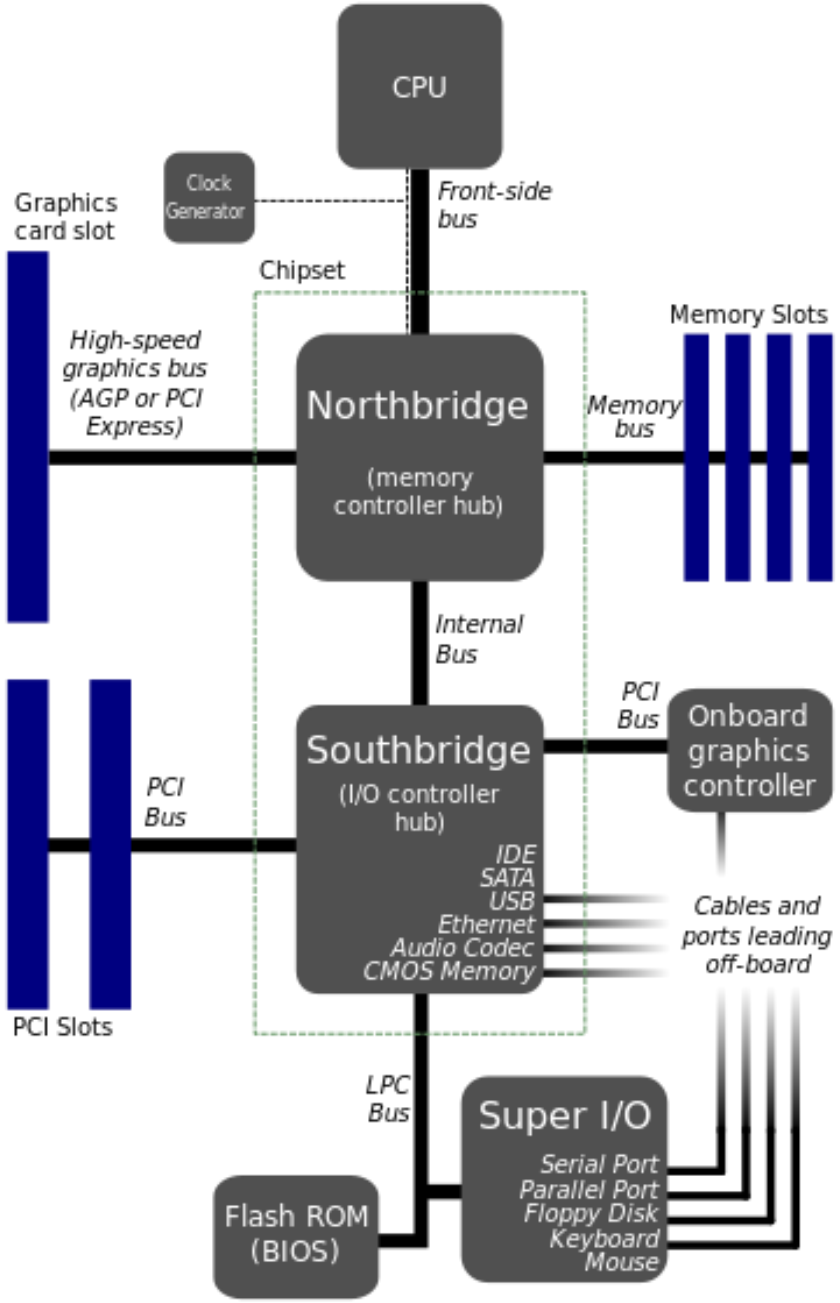
Fonte: STALLINGS, W. Arquitetura de Computadores, 5ª Edição, Prentice Hall do Brasil.

Balanceamento é a chave para desempenho

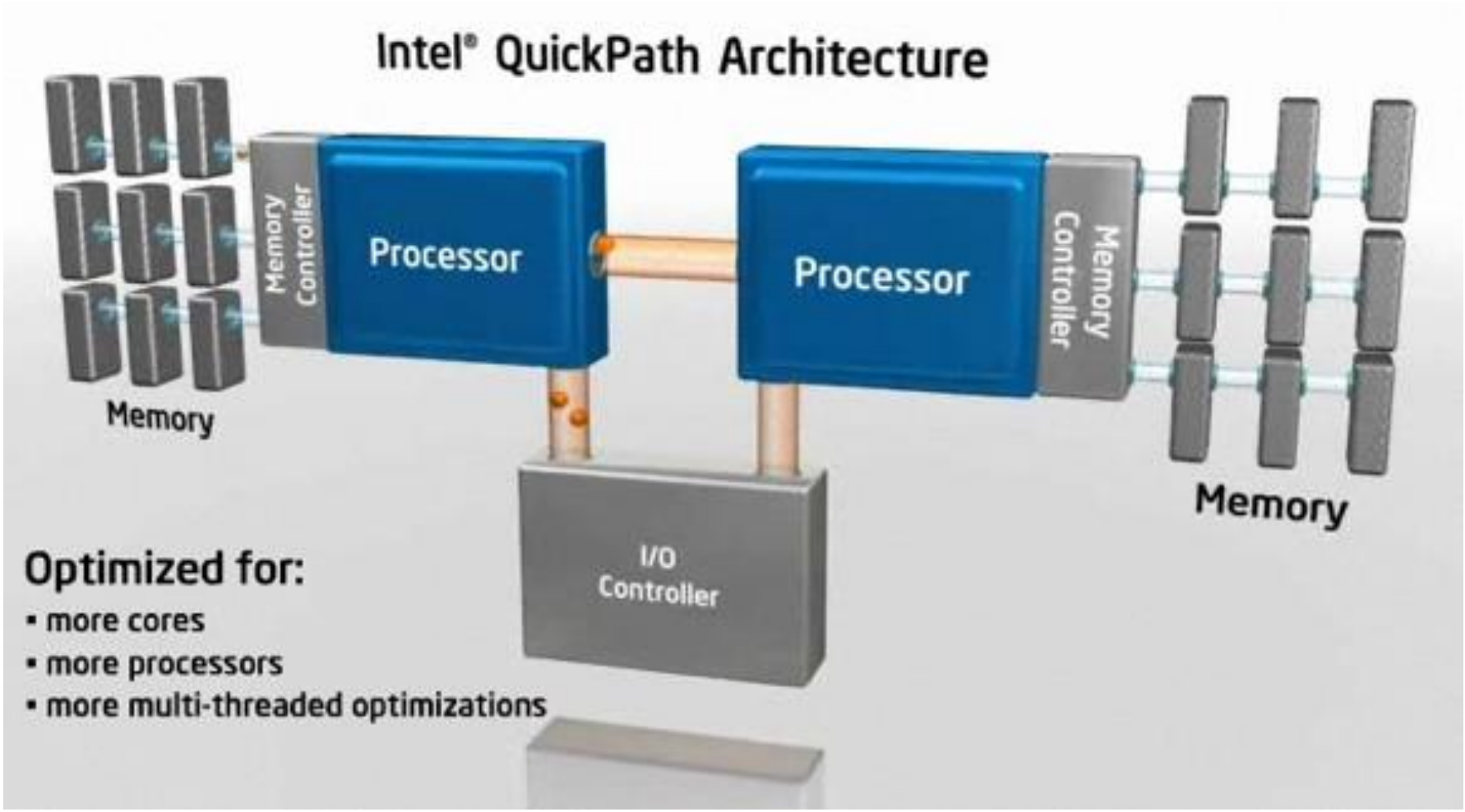
- Componentes do Processador
- Memória principal
- Dispositivos de E/S
- Estruturas de interconexão (barramentos)

- Todas as partes de um sistema evoluem para acompanhar o crescimento do desempenho das CPUs.
- Por exemplo, até 2008 era utilizado o FSB (Front Side Bus) para conexão com a memória
- Atualmente a tecnologia utilizada pela Intel é o QPA (QuickPath Interconnect)
- Um processador de hoje com o FSB não aproveitaria todo seu desempenho (Lei de Amdahl)

Front Side Bus



QuickPath Interconnect



Melhorias na Arquitetura e Organização dos Chips

- Aumento da velocidade do HW devido à redução das portas lógicas (tamanho)
  - Mais portas, mais empacotamento, maior taxa de clock
  - Porém, maior consumo de energia!
  - Tempo de propagação do sinal reduzido
- Aumento de tamanho e velocidade das caches
  - Parte da CPU é dedicada às caches
  - Tempos de acesso à cache caem significativamente
- Mudança na arquitetura e organização da CPU
  - Unidades funcionais
  - Ganho efetivo na velocidade de execução
  - Paralelismo

Problemas com Taxa de Clock e Densidade da Lógica

- Potência
  - Densidade de potência aumenta com o aumento da densidade de lógica e da velocidade do clock
  - Dissipação é um problema
- Atraso “RC”
  - Velocidade do fluxo de elétrons é limitada pela Resistência (R) e Capacitância (C) dos materiais que constroem a lógica
  - Aumento de *delay* proporcional à carga RC
  - Conexões mais “finas” → mais resistência
  - Conexões mais “próximas” → mais capacitância
- Latência das Memórias
- Solução: Ênfase maior nas abordagens que focam na Arquitetura e Organização

Aumento da Capacidade das Caches

- Tipicamente de 2 a 3 níveis de cache entre memória e CPU
- Densidade de Chip aumenta
  - Mais cache on chip → acesso mais rápido à cache → maior custo.

- Pentium: cerca de 10% do chip para cache
- Pentium 4: cerca de 50%

Lógica de Execução Mais Complexa

- Execução paralela de instruções
- Pipeline trabalha como linha de montagem
  - Diferentes estágios de execução sendo utilizados por diferentes instruções ao longo do pipeline
- Arquiteturas superescalares permitem vários pipelines dentro de um único processador
  - Instruções que não dependem umas das outras podem ser executadas em paralelo dentro da CPU

Retornos “não tão grandes”

- Organização interna da complexidade de processadores
  - –Nível de paralelismo a ser tratado
  - –Ganhos obtidos podem ser significativamente modestos
- Benefícios da cache são “limitados”
- Aumentar o clock implica em complicar a dissipação de potência
  - Alguns limites da física estarão sendo atingidos em pouco tempo
  - Computação quântica?
  - Outras soluções?

Observações Finais

- Desempenho é específico para determinado(s) programa(s)
  - Tempo Total de Execução é um “sumário” consistente de medida de desempenho global
- Para uma dada arquitetura, o desempenho pode ser aumentado:
  - Aumentando a taxa de clock (desconsiderando efeitos no CPI)
  - Melhorando a organização do processador, que baixa o CPI
  - Melhorando o compilador que reduz o CPI e/ou o no. de instruções
- ATENÇÃO: Tais ganhos de desempenho são apenas um dos aspectos do desempenho da máquina que afeta o desempenho total

Melhoria de desempenho mais recentes

- Pipelining
- Cache on chip (dentro da CPU)
- Diferentes níveis de cache (L1, L2, ...)
- Predição de desvio (para pipelines)
- Análise do fluxo de dados (redução de dependências)
- Execução especulativa (ocupação das unidades funcionais)

Exercícios

1) Um programa de benchmark é executado em um processador a 40 MHz. O programa consiste das seguintes instruções:

Tipo de instrução	Quantidade de instruções	CPI
Aritmética de inteiros	45.000	1
Transferência de dados	32.000	2
Ponto flutuante	15.000	2
Transferência de controle	8.000	2

Determine o CPI efetivo, a taxa de MIPS e o tempo de execução para esse programa.

Exercícios

2) Os primeiros exemplos de projetos CISC e RISC são o VAX 11/780 e o IBM RS/6000, respectivamente. Usando um programa de benchmark típico, o resultado são as seguintes características de máquina:

Processador	Frequência de clock	Desempenho	Tempo de CPU
VAX 11/780	5 MHz	1 MIPS	$12x$ segundos
IBM RS/6000	25 MHz	18 MIPS	$1x$ segundos

A coluna final mostra que o VAX exigia 12 vezes mais tempo que o IBM, medido em tempo de CPU.

a) Qual é o tamanho relativo da quantidade de instruções do código de máquina para esse programa de benchmark rodando nas duas máquinas?

b) Quais são os valores de CPI para as duas máquinas?

Exercícios

3) No exercício 1, uma melhoria na arquitetura da unidade de ponto flutuante reduz à metade o tempo de execução desse tipo de operação. Qual será o *speedup* final para o programa?

4) Também no exercício 1, Se todas as transferências, de dados e controle, ficarem 4 vezes mais rápidas, qual será o *speedup* final para o programa?

Exercícios

5) Um software de busca de padrões em cadeias de DNA de proteínas leva 4 dias de tempo de execução em um determinado computador. Desse tempo de execução, 20% é gasto com instruções de inteiros e 35% gasto realizando operações de E/S. Qual é a melhor opção para reduzirmos o tempo de execução:

A. Otimização do compilador que reduz o número de instruções de inteiros em 25% (assuma que todas as instruções de inteiros tem o mesmo CPI).
B. Otimização do hardware que reduz a latência de cada operação de E/S de 6μs para 5μs.

Exercícios

6) Assuma que o cálculo da raiz quadrada é responsável por 20% do tempo gasto por um *benchmark* gráfico. Considere as seguintes alternativas:

i. Tornar o cálculo da raiz quadrada 10 vezes mais rápido;
ii. Tornar todas as instruções de ponto flutuante 60% mais rápidas.

As instruções de ponto flutuante perfazem 50% do tempo total de execução. Compare as duas alternativas.