

Desempenho de CPUs

Noções básicas

João Canas Ferreira

Janeiro de 2020



Tópicos

- 1 Conceitos básicos
- 2 Equação fundamental do desempenho
- 3 Lei de Amdahl

Contém figuras de “Computer Organization and Design”, D. Patterson & J. Hennessey, 3ª. ed., MKP

1 Conceitos básicos

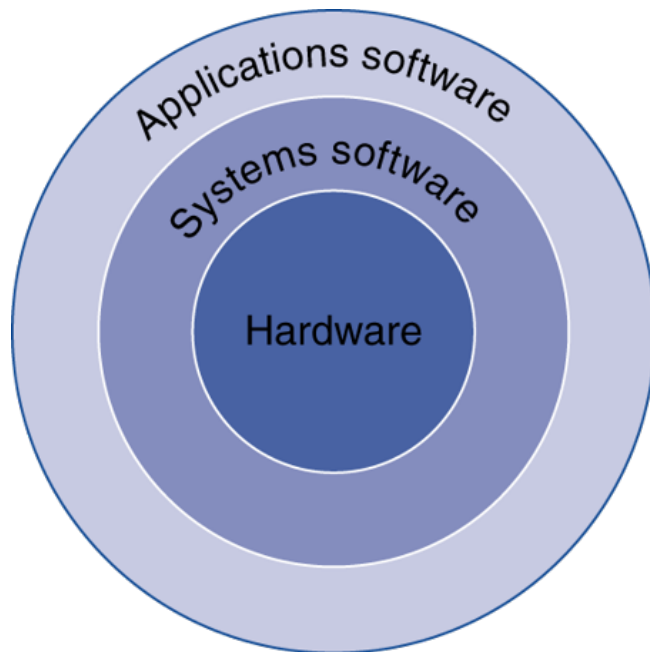
2 Equação fundamental do desempenho

3 Lei de Amdahl

O que influencia o desempenho de um CPU?

- Algoritmo usado
Determina o número de operações executadas
- Linguagem de programação, compilador, arquitetura
Determina o número de instruções executadas por operação
- Processador e sistema de memória
Determina a rapidez de execução das instruções
- Sistema de E/S (incluindo o sistema operativo)
Determina a rapidez das operações de E/S

Da aplicação ao suporte físico



- Aplicação
 - Escrita numa linguagem de alto nível
- Software de sistema
 - Compilador
 - Traduz código de linguagem de alto nível para código-máquina
 - Sistema operativo: serviços
 - Gestão de entradas/saídas de dados
 - Gestão de memória e armazenamento
 - Escalonamento de tarefas e partilha de recursos
- Suporte físico (*hardware*)
 - Processador, memória, controladores de E/S

Níveis de código

- Linguagem de alto nível
 - Nível de abstração mais perto do domínio do problema
 - Maior produtividade e portabilidade
- Linguagem *assembly*
 - Representação textual das instruções do CPU
- Código-máquina
 - Dígitos binários (bits)
 - Codificação de dados e instruções

High-level
language
program
(in C)

```
swap(int v[], int k)
{int temp;
  temp = v[k];
  v[k] = v[k+1];
  v[k+1] = temp;
}
```

Compiler

Assembly
language
program
(for MIPS)

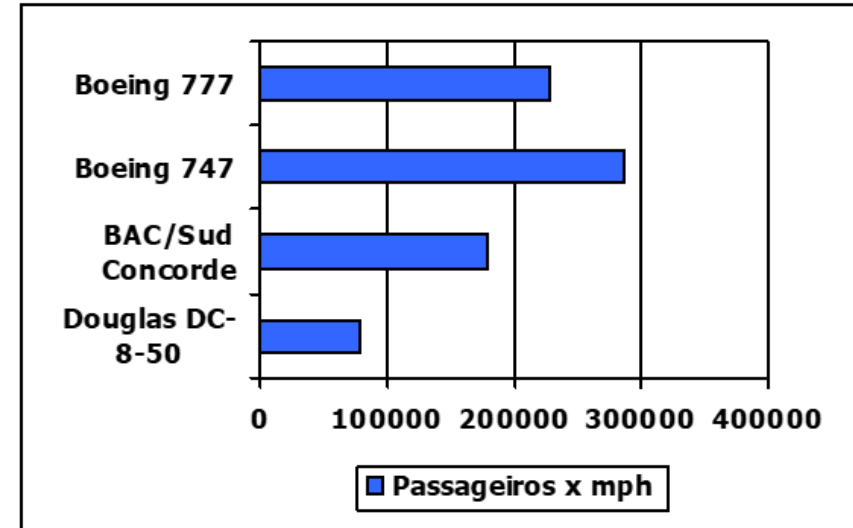
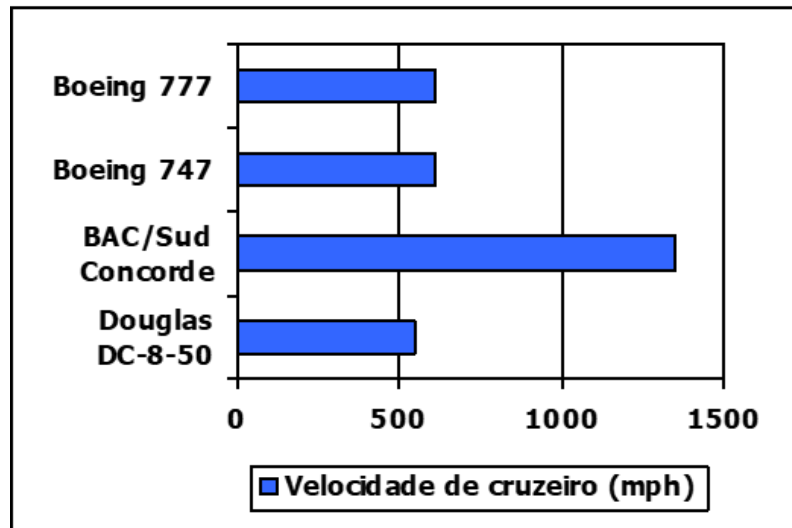
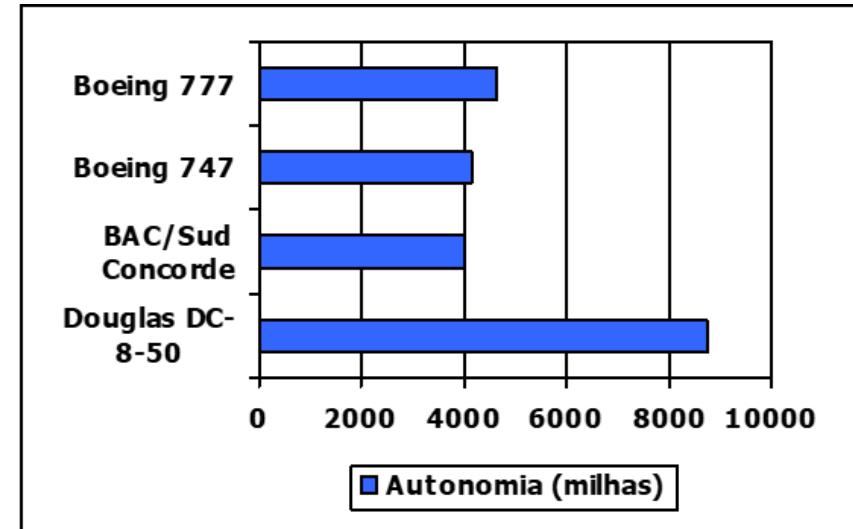
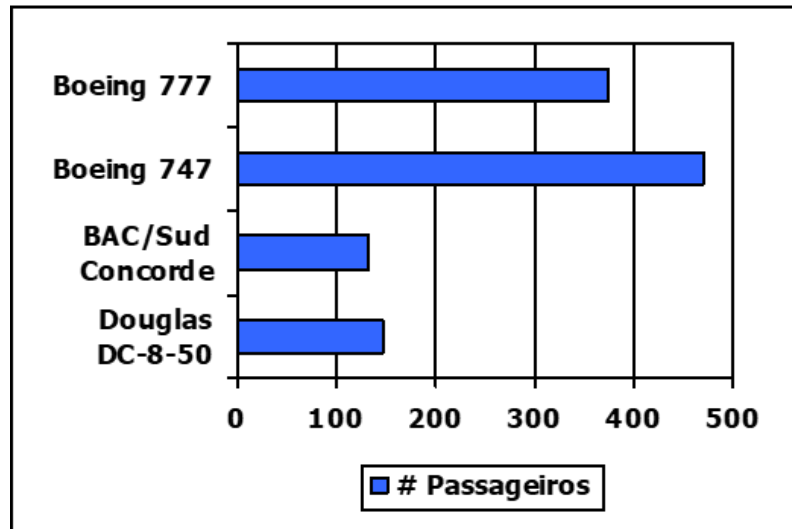
```
swap:
    muli $2, $5, 4
    add  $2, $4, $2
    lw   $15, 0($2)
    lw   $16, 4($2)
    sw   $16, 0($2)
    sw   $15, 4($2)
    jr   $31
```

Assembler

Binary machine
language
program
(for MIPS)

```
00000000101000010000000000011000
000000000000110000001100000100001
10001100011000100000000000000000
10001100111100100000000000000100
10101100111100100000000000000000
10101100011000100000000000000100
00000011111000000000000000001000
```

Critérios de desempenho



Fonte: [COD4]

👉 Qual é o melhor avião?

Tempo de resposta e débito

■ *Tempo de resposta ou latência*

Def: O tempo que o CPU demora a executar uma tarefa

– Tempo necessário para começar a serem produzidos resultados contado a partir do início da tarefa

■ *Débito*

Def: Trabalho efetuado por unidade de tempo

– Exemplos: número de tarefas por hora; número de instruções executadas por segundo.

■ Questão: Como é que débito e tempo de resposta são afetados ao:

– substituir-se um processador por outro mais rápido?

– acrescentar-se mais processadores (cada um pode executar uma tarefa)?

■ Foco desta introdução: **tempo de resposta** como medida de desempenho

1 Conceitos básicos

2 Equação fundamental do desempenho

3 Lei de Amdahl

Desempenho e desempenho relativo

▀ Definição: **Desempenho = 1 / (Tempo de execução)** (unidades: s^{-1})

▀ **Desempenho relativo**

O sistema X é n vezes mais rápido que o sistema Y

$$n = \frac{\text{Tempo de execução}_Y}{\text{Tempo de execução}_X} = \frac{\text{Desempenho}_X}{\text{Desempenho}_Y}$$

▀ Exemplo:

Tempo de execução de um dado programa: 10 s no computador A, 15 s em B

$$n = \frac{\text{Tempo de execução}_B}{\text{Tempo de execução}_A} = \frac{15 \text{ s}}{10 \text{ s}} = 1,5$$

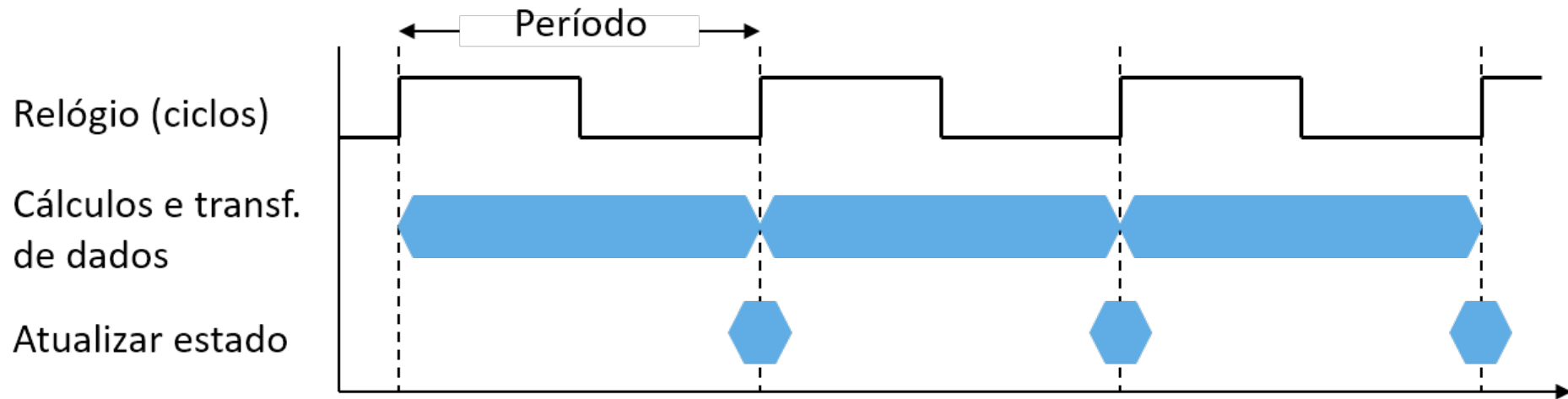
▀ O computador é 1,5 vezes mais rápido que B (para este programa!)

Tempo de execução

- Tempo transcorrido (*wall-clock time*): tempo de resposta **total**
 - Inclui processamento, E/S, *overhead* do sistema operativo, inatividade, tempo gasto pelo sistema com outras tarefas...
 - Determina o desempenho do sistema (mas depende de muitos fatores e não apenas do CPU)
- Tempo de CPU (*CPU time*):
 - Tempo gasto pelo CPU no processamento de uma dada tarefa
 - Não inclui tempo E/S, nem tempo de outras tarefas
 - Tem duas componentes:
 - tempo de utilizador (*user time*): tempo gasto pelo CPU a executar código do "utilizador"
 - tempo de sistema (*system time*); tempo gasto pelo sistema operativo a executar trabalho para a tarefa (aceder a ficheiros, etc.)
 - Cada programa pode ser afetado de maneira diferente por cada componente
- O tempo despendido numa tarefa depende do CPU, do código da tarefa (algoritmo, compilador) e dos **dados de entrada**

CPU como circuito sequencial síncrono

➡ Operação de circuitos síncronos é governada por um sinal de relógio (de período constante).



- Período: duração de um ciclo de relógio
p. ex., $250 \text{ ps} = 0,25 \text{ ns} = 250 \times 10^{-12} \text{ s}$
- Frequência de relógio: ciclos por segundo
p. ex., $4,0 \text{ GHz} = 4000 \text{ MHz} = 4,0 \times 10^9 \text{ Hz}$

Tempo de CPU

Para um dado programa com um conjunto específico de dados de entrada:

$$\begin{aligned}\text{Tempo de CPU} &= \text{Número de ciclos de execução} \times \text{Período de um ciclo} \\ &= \frac{\text{Número de ciclos de execução}}{\text{Frequência do relógio}}\end{aligned}$$

- Desempenho numa tarefa pode ser melhorado:
 - ① Reduzindo o número de ciclos de relógio necessários
 - Como? Aumentando a quantidade de trabalho efetuado em cada ciclo.
 - ② Aumentando a frequência do relógio
- Em geral, os dois objetivos são antagónicos!
 - Projetista do CPU deve obter um compromisso

Tempo de CPU: Exemplo

- ➡ Computador A: relógio de 2 GHz, tarefa demora 10 s
- ➡ Com um novo computador B: tarefa deve demorar 6 s
Pode-se usar relógio mais rápido, mas tarefa passa a requerer $1,2\times$ mais ciclos.

Qual deve ser a frequência de relógio do computador B?

$$\text{Frequência}_B = \frac{\text{Ciclos de execução}_B}{\text{Tempo de CPU}_B} = \frac{1,2 \times \text{Ciclos de execução}_A}{6 \text{ s}}$$

$$\begin{aligned}\text{Ciclos de execução}_A &= \text{Tempo de CPU}_A \times \text{Frequência}_A \\ &= 10 \text{ s} \times 2 \text{ GHz} = 20 \times 10^9\end{aligned}$$

$$\text{Frequência}_B = \frac{1,2 \times 20 \times 10^9}{6 \text{ s}} = \frac{24 \times 10^9}{6 \text{ s}} = 4 \text{ GHz}$$

Número de instruções e CPI

- ➡ CPI: número **médio** de ciclos por instrução
- ➡ A média deve ser válida para o programa cujo desempenho está a ser avaliado.

$$\# \text{ de ciclos executados} = \# \text{ instruções executadas} \times \text{CPI}$$

$$\begin{aligned} \text{Tempo de CPU} &= \# \text{ instruções executadas} \times \text{CPI} \times \text{período de relógio} \\ &= \frac{\# \text{ instruções executadas} \times \text{CPI}}{\text{Frequência}} \end{aligned}$$

$$T_{\text{exec}} = N \times \text{CPI} \times T = \frac{N \times \text{CPI}}{F} \quad (\text{equação fundamental})$$

- # instruções executadas de um programa:
 - Determinado pelo programa, conjunto de instruções e compilador
- Número de ciclos por instrução (CPI)
 - Para uma instrução: Determinado pela implementação
 - Para programa: o CPI médio depende da distribuição de instruções

Influência de CPI (exemplo)

- Computador A: Período = 250 ps, CPI = 2,0
Computador B: Período = 500 ps, CPI = 1,2
Mesmo conjunto de instruções e mesmo programa.
- Qual computador é mais rápido e por quanto?

$$\begin{aligned}\text{Tempo de CPU}_A &= \# \text{ instruções} \times \text{CPI} \times \text{Período de relógio}_A \\ &= N \times 2,0 \times 250 \text{ ps} = 500 \text{ ps}\end{aligned}$$

$$\begin{aligned}\text{Tempo de CPU}_B &= \# \text{ instruções} \times \text{CPI} \times \text{Período de relógio}_B \\ &= N \times 1,2 \times 500 \text{ ps} = N \times 600 \text{ ps}\end{aligned}$$

$$\frac{\text{Tempo de CPU}_B}{\text{Tempo de CPU}_A} = \frac{N \times 600 \text{ ps}}{N \times 500 \text{ ps}} = 1,2$$

- O computador A é mais **1,2** vezes mais rápido que o computador B.

Diferentes classes de instruções

▀ Quando diferentes classes de instruções requerem diferente número de ciclos:

$$\# \text{ ciclos} = \sum_{i=1}^n (\text{CPI}_i \times \# \text{ instruções de classe } i)$$

▀ **CPI médio** (média pesada pela taxa de ocorrência de cada classe):

$$\text{CPI} = \frac{\# \text{ ciclos}}{\# \text{ instruções}} = \sum_{i=1}^n \left(\text{CPI}_i \times \frac{\# \text{ instruções de classe } i}{\# \text{ instruções}} \right)$$

▀ Exemplo: Duas pequenas sequências de código alternativas usando instruções das classes A, B e C.

Classe	A	B	C
CPI	1	2	3
Sequência 1 (N=5)	2	1	2
Sequência 2 (N=6)	4	1	1

▀ Sequência 1:

ciclos = 10 → CPI médio = 10/5 = 2

▀ Sequência 2:

ciclos = 9 → CPI médio = 9/6 = 1,5

Desempenho: Equação fundamental

▣ Equação fundamental de desempenho:

$$\text{Tempo de CPU} = \# \text{ instruções} \times \frac{\# \text{ ciclos}}{\# \text{ instruções}} \times \frac{\# \text{ segundos}}{\text{ciclo}}$$

$$T_{\text{exec}} = N \times \text{CPI} \times T$$

▣ Desempenho de CPU depende de:

- Algoritmo: afeta # instruções, possivelmente também o CPI
- Linguagem de programação: # instruções, CPI
- Compilador: afeta # instruções, CPI
- Arquitetura do conjunto de instruções: afeta # instruções, período, CPI

Avaliação global de desempenhos por “benchmarks”

A equação fundamental determina o desempenho de um CPU executando **um** programa específico para dados de entrada definidos.

- Benchmarks: Programas usados para medir o desempenho
 - Programas devem ser típicos do domínio de utilização
- Organismo Standard Performance Evaluation Corp. (SPEC)
 - Desenvolve “benchmarks” para CPU, E/S, Web,...
- “Benchmarks” SPEC CPU2006, CPU2017
 - Tempo transcorrido na execução do programa
 - Débito para processadores multi-core (versão CPU2017)
 - E/S negligenciável: foco no desempenho de CPU
 - **Valores normalizados relativo a um computador de referência**
 - Programas com e sem vírgula flutuante
 - Sumário: **média geométrica das razões de desempenho**

$$\sqrt[n]{\prod_{i=1}^n (\text{razão dos tempos de execução de benchmark } i)}$$

CINT2006 para Intel Core i7 920

Description	Name	Instruction Count x 10 ⁹	CPI	Clock cycle time (seconds x 10 ⁻⁹)	Execution Time (seconds)	Reference Time (seconds)	SPECratio
Interpreted string processing	perl	2252	0.60	0.376	508	9770	19.2
Block-sorting compression	bzip2	2390	0.70	0.376	629	9650	15.4
GNU C compiler	gcc	794	1.20	0.376	358	8050	22.5
Combinatorial optimization	mcf	221	2.66	0.376	221	9120	41.2
Go game (AI)	go	1274	1.10	0.376	527	10490	19.9
Search gene sequence	hmmer	2616	0.60	0.376	590	9330	15.8
Chess game (AI)	sjeng	1948	0.80	0.376	586	12100	20.7
Quantum computer simulation	libquantum	659	0.44	0.376	109	20720	190.0
Video compression	h264avc	3793	0.50	0.376	713	22130	31.0
Discrete event simulation library	omnetpp	367	2.10	0.376	290	6250	21.5
Games/path finding	astar	1250	1.00	0.376	470	7020	14.9
XML parsing	xalancbmk	1045	0.70	0.376	275	6900	25.1
Geometric mean	–	–	–	–	–	–	25.7

Fonte: [COD4]

➡ <https://www.spec.org/benchmarks.html>

1 Conceitos básicos

2 Equação fundamental do desempenho

3 Lei de Amdahl

“Lei” de Amdahl

▀ Melhorar o desempenho de um aspeto de um computador **não leva** a um melhoramento proporcional do desempenho global

Porquê? O melhoramento não pode ser utilizado durante todo o tempo de execução original do programa.

▀ O tempo de execução do programa com o melhoramento é:

$$T_{\text{melhorado}} = \frac{T_{\text{afetado}}}{\text{fator de melhoramento}} + T_{\text{não afetado}}$$

▀ Exemplo: programa gasta 80 s de 100 s a fazer multiplicações

De quanto é preciso melhorar o desempenho do multiplicador para obter uma desempenho total 5× maior?

$$20 = \frac{80}{f} + 20 \quad \text{Impossível para } f > 0$$

Lei de Amdahl: aumento de rapidez

▣➡ Aumento de rapidez (*speedup*): fator de melhoramento do desempenho

$$S = \text{Speedup} = \frac{\text{Desempenho com melhoramento}}{\text{Desempenho original}}$$

▣➡ **Cenário de utilização parcial:** Melhoramento com *speedup* S_{parcial} é usado durante uma percentagem p do tempo de execução ($0 \leq p \leq 1$).

Qual é o aumento de rapidez **global**?

$$S_{\text{global}} = \frac{\text{Tempo de execução original}}{\text{Tempo de execução melhorado}} = \frac{1}{\frac{p}{S_{\text{parcial}}} + (1 - p)}$$

(Demonstrar!)

Aumento de rapidez: exemplo

▢ Um programa gasta 80 s de 100 s em multiplicações.

▢ Um novo multiplicador é 2× mais rápido que o original. Qual é o fator de melhoramento (*speedup*)?

$$S_{\text{global}} = \frac{1}{\frac{0,8}{2} + (1 - 0,8)} = \frac{1}{0,6} = \frac{10}{6} \approx 1,67$$

▢ De quanto é preciso melhorar o desempenho do multiplicador para obter um desempenho total 2× maior?

$$2 = \frac{1}{\frac{0,8}{S_{\text{parcial}}} + (1 - 0,8)} \Rightarrow S_{\text{parcial}} = \frac{16}{6} = 2\frac{2}{3} \approx 2,67$$

Erro metodológico: Usar medidas parciais de desempenho

➡ MIPS: *millions of instructions per second*

➡ Não tem em conta:

- Diferenças entre conjuntos de instruções
- Diferenças de complexidade entre instruções (→ CPI)

$$\begin{aligned}\text{MIPS} &= \frac{\# \text{ instruções}}{\text{Tempo de execução} \times 10^6} \\ &= \frac{\# \text{ instruções}}{\frac{\# \text{ instruções} \times \text{CPI}}{\text{Frequência de relógio}}} \times 10^6 = \frac{\text{Frequência de relógio}}{\text{CPI} \times 10^6}\end{aligned}$$

☞ Mas para o mesmo CPU, o número de instruções executadas varia entre programas (ou para o mesmo programa com dados diferentes)!

Referências

COD4 D. A. Patterson & J. L. Hennessey, Computer Organization and Design, 4 ed.

COD3 D. A. Patterson & J. L. Hennessey, Computer Organization and Design, 3 ed.

Os tópicos tratados nesta apresentação são descritos nas seguintes secções de [COD4]:

- 1.4, 1.7–1.8

Também são tratados nas seguintes secções de [COD3]:

- 4.1–4.5