

# [Aula 06] Avaliando o desempenho 2

Prof. João F. Mari  
*joaof.mari@ufv.com*

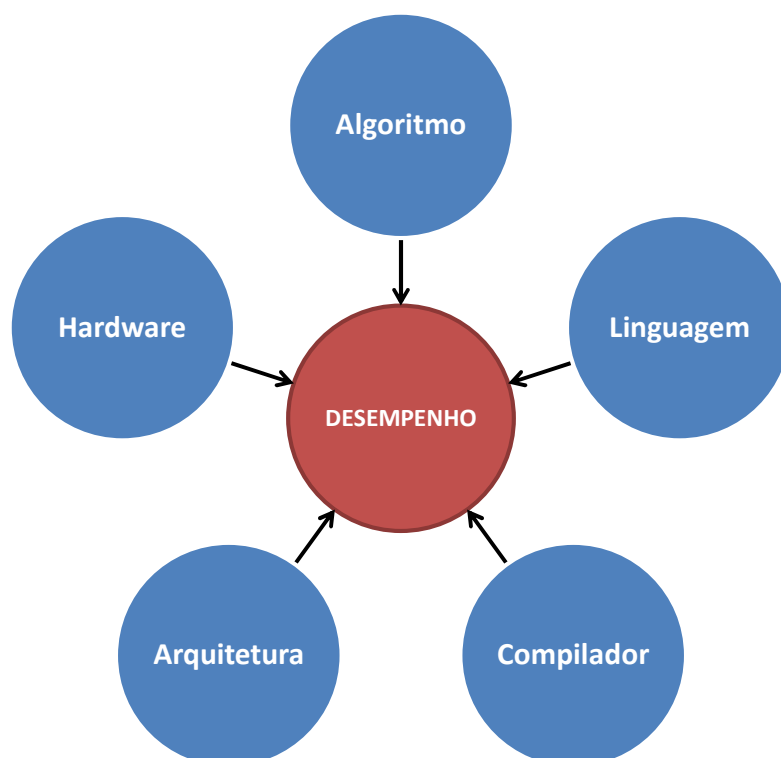
## Aula 06 – Avaliando o desempenho 2

### Roteiro

- Entendendo o desempenho dos programas
- **[EX]** Comparando segmentos de códigos
- Avaliando o desempenho
- Comparando e resumindo o desempenho
- Tempo de execução total
- Benchmark SPEC
- Lei de Amdahl
- MIPS – Milhões de instruções por segundo
- **[EX]** MIPS como medida de desempenho

## ENTENDENDO O DESEMPENHO DOS PROGRAMAS

## Entendendo o desempenho dos programas



# Entendendo o desempenho dos programas

- Algoritmo
  - Afeta a contagem de instruções e possivelmente o CPI.
    - *O algoritmo determina o número de instruções do programa fonte executadas e, portanto, o número de instruções do processador executadas.*
    - *Pode afetar o CPI, favorecendo instruções mais lentas ou mais rápidas. Por exemplo, se o algoritmo usar mais operações de ponto flutuante, ele terá um CPI mais alto.*
- Linguagem de programação
  - Contagem de instruções, CPI.
    - *As instruções da linguagem são traduzidas em instruções do processador, que determinam a contagem de instruções.*
    - *Pode afetar o CPI devido aos seus recursos; por exemplo, uma linguagem com pesado suporte a abstrações de dados (como Java) exigirá chamadas indiretas, que usarão instruções de CPI mais altos.*

# Entendendo o desempenho dos programas

- Compilador
  - Contagem de instruções, CPI.
    - *O compilador determina a tradução das instruções da linguagem fonte para instruções do computador.*
- Conjunto de instruções
  - Contagem de instruções, velocidade de clock, CPI
    - *Afeta os três aspectos do desempenho da CPU, uma vez que ele afeta as instruções necessárias para uma função, o custo em ciclos de cada instrução e a velocidade geral do processador.*

## [EX] Comparando segmentos de código

- O projetista de um compilador precisa decidir entre duas sequencias de código para um determinado computador:

CPI para cada classe de instruções			
	A	B	C
CPI	1	2	3

- Para uma determinada instrução da linguagem de alto nível, o projetista do compilador está considerando duas sequencias de código que exigem as seguintes contagens de instrução:

Contagens de instrução para classe de instrução			
Sequencia de código	A	B	C
1	2	1	2
2	4	1	1

- Qual sequencia de código executa mais instruções? Qual será a mais rápida? Qual é o CPI para cada sequencia?

SIN 252 – Arquitetura de computadores (2021-2)

## [EX] Comparando segmentos de código

### RESPOSTA:

- A sequencia 1 executa  $2 + 1 + 2 = 5$  instruções
- A sequencia 2 executa  $4 + 1 + 1 = 6$  instruções
  - Logo, a sequencia 2 executa mais instruções
- Equação para os ciclos de clock de CPU
  - Contagem de instruções e CPI

$$CiclosDeClockDeCPU = \sum_{i=1}^n (CPI_i \times C_i)$$

- Isso resulta em:
  - Ciclos de clock da CPU<sub>1</sub> =  $(1 \times 2) + (2 \times 1) + (3 \times 2) = 2 + 2 + 6 = 10$  ciclos
  - Ciclos de clock de CPU<sub>2</sub> =  $(1 \times 4) + (2 \times 1) + (3 \times 1) = 4 + 2 + 3 = 9$  ciclos

SIN 252 – Arquitetura de computadores (2021-2)

## [EX] Comparando segmentos de código

### RESPOSTA:

- A sequência de código 2 é a mais rápida, mesmo executando uma instrução extra
  - Os valores de CPI podem ser calculados por:

$$CPI = \frac{CiclosDeClockDaCPU}{ContagemDeInstruções}$$

$$CPI_1 = \frac{CiclosDeClockDaCPU_1}{ContagemDeInstruções_1} = \frac{10}{5} = 2$$

$$CPI_2 = \frac{CiclosDeClockDaCPU_2}{ContagemDeInstruções_2} = \frac{9}{6} = 1,5$$

## AVALIANDO O DESEMPENHO – WORKLOAD E BENCHMARK

# Avaliando o desempenho

- **Workload:**
  - Um conjunto de programas executados em um computador
- **Benchmark:**
  - Programas especificamente escolhidos para medir desempenho
  - Formam um *workload* (carga de trabalho)
  - O melhor tipo de programa para *benchmark* são as aplicações reais
  - Diferentes classes e aplicações de computadores exigirão diferentes tipos de benchmarks
    - **Desktop** → desempenho de CPU ou alguma tarefa específica
    - **Servidores Científicos** → orientados a CPU (tempo de resposta)
    - **Servidores** → serviços Web, compartilhamento de arquivos, banco de dados (vazão)
    - **Computação embarcada** → poucos benchmarks, próprias aplicações
  - Reprodutibilidade

# Avaliando o desempenho

- Selecionados os programas a serem usados como benchmarks
  - Decidido entre medir tempo de execução e vazão
- Como resumir o desempenho de um grupo de benchmarks

	Computador A	Computador B
Programa 1 (seg.)	1	10
Programa 2 (seg.)	1000	100
Tempo total (seg.)	1001	110

- A é 10 vezes mais rápido que B para o prog. 1
- B é 10 vezes mais rápido que A para o prog. 2
- O desempenho relativo dos computadores A e B não é claro

# Tempo de execução total

- Resumir desempenho relativo
  - Usar o tempo de execução total dos dois programas

$$\frac{Desempenho_B}{Desempenho_A} = \frac{TempoDeExecução_A}{TempoDeExecução_B} = \frac{1001}{110} = 9,1$$

- B é 9,1 vezes mais rápido do que A para os programas 1 e 2 juntos
  - Média aritmética (MA):
    - Diretamente proporcional ao tempo de execução total
- $$MA = \frac{1}{n} \sum_{i=1}^n Tempo_i$$
- Média Aritmética Ponderada:
    - Utiliza pesos destinados a refletir a presença dos programas em um *workload*

# Benchmark SPEC

- Benchmark SPEC CPU (*System Performance Evaluation Corporation*)
  - [www.spec.org](http://www.spec.org)
  - Última versão é o pacote SPEC CPU2006
    - 12 programas de inteiros e 19 de ponto flutuante
  - Substitui a versão SPEC CPU2000
    - 12 programas de inteiros e 14 de ponto flutuante
- SPECweb2009 (descontinuado)
  - Benchmark de vazão para servidores web
  - Sistemas com múltiplos processadores
  - Depende de características de sistema (disco, memória, etc.)
- Outras classes de benchmark
  - CPU, Processamento gráfico, computação de alto-desempenho, cliente-servidor Java, servidores de e-mail, servidores de arquivos, virtualização etc.

# Lei de Amdahl

- O aumento de desempenho possível com uma determinada melhoria é limitado pela quantidade de uso do recurso melhorado

$$\begin{aligned} & \text{TempoDeExecuçãoApósMelhoria} \\ &= \frac{\text{TempoDeExecuçãoAfetadoPelaMelhoria}}{\text{QualidadeDaMelhoria}} \\ &+ \text{TempoDeExecuçãoNãoAfetado} \end{aligned}$$

- Consequência da lei de Amdahl: torne o caso comum rápido
  - A frequência de alguns eventos é muito maior do que outros.
  - Geralmente o caso comum é mais simples.

## MIPS – *Milhões de instruções por segundo*

- Alternativa para o tempo com métrica de desempenho
  - Baseado no número de milhões de instruções

$$MIPS = \frac{\text{ContadorDeInstruções}}{\text{TempoDeExecução} \times 10^6}$$

- Computadores mais rápidos possuem um índice MIPS mais alto
  - Mais fácil de ser entendido
- **Problemas:**
  - Considera a taxa de execuções de instruções
    - Mas desconsidera as capacidades das instruções
    - Não permite comparar computadores com diferentes conjuntos de instruções
  - O MIPS varia entre programas no mesmo computador
  - Pode variar inversamente com o desempenho!



## [EX] MIPS como medida de desempenho

- Computador com três classes de instruções e medições de CPI

CPI para está classe de instrução			
	A	B	C
CPI	1	2	3

- Medindo o código gerado por dois compiladores diferentes para o mesmo programa

Contagens de instrução para classe de instrução			
Sequencia de código	A	B	C
1	5	1	1
2	10	1	1

- Velocidade de clock de 4GHz.
- Qual sequencia de código será executada mais rápido de acordo com o MIPS? E de acordo com o tempo de execução?

## [EX] MIPS como medida de desempenho

- Encontramos o tempo de execução

$$\text{TempoDeExecução} = \frac{\text{CiclosDeClockdaCPU}}{\text{VelocidadeDeClock}}$$

- Para os ciclos de clock da CPU:

$$\text{CiclosDeClockDaCPU} = \sum_{i=1}^n (\text{CPI}_i \times C_i)$$

$$\text{CiclosDeClockDaCPU}_1 = (1 \times 5 + 2 \times 1 + 3 \times 1) \times 10^9 = 10 \times 10^9$$

$$\text{CiclosDeClockDaCPU}_2 = (1 \times 10 + 2 \times 1 + 3 \times 1) \times 10^9 = 15 \times 10^9$$

- Tempo de execução para os dois compiladores:

$$\text{TempoDeExecução}_1 = \frac{10 \times 10^9}{4 \times 10^9} = 2,5 \text{ seg.}$$

$$\text{TempoDeExecução}_2 = \frac{15 \times 10^9}{4 \times 10^9} = 3,75 \text{ seg.}$$

- De acordo com o **tempo de execução** o compilador 1 gera o programa mais rápido

## [EX] MIPS como medida de desempenho

- O índice MIPS para cada versão do programa

$$MIPS = \frac{ContagemDeInstruções}{TempoDeExecução \times 10^6}$$

$$MIPS_1 = \frac{(5 + 1 + 1) \times 10^9}{2,5 \times 10^6} = 2.800$$

$$MIPS_2 = \frac{(10 + 1 + 1) \times 10^9}{3,75 \times 10^6} = 3.200$$

- Código do compilador 2 possui índice MIPS mais alto
- Código do compilador 1 é executado mais rápido
  - MIPS pode falhar em fornecer o desempenho

## Apêndice – Frações de segundo

Unidade	Em segundos
Segundo (s)	1 s
Milisegundos (ms)	0,001 s ( $1 \times 10^{-3}$ s)
Microsegundos ( $\mu$ s)	0,000001 s ( $1 \times 10^{-6}$ s)
Nanosegundos (ns)	0,000000001 s ( $1 \times 10^{-9}$ s)
Picosegundos (ps)	0,000000000001 s ( $1 \times 10^{-12}$ s)

# Bibliografia

1. PATTERSON, D.A; HENNESSY, J.L. **Organização e Projeto de Computadores: A Interface Hardware/Software**. 3a. Ed. Elsevier, 2005.
  - Capítulo 2
2. Notas de aula do prof. Luciano J. Senger:
  - <http://www.ljsenger.net/classroom.html>



## FIM – Aula 06

- FIM:
  - Aula 06 – Avaliando o desempenho 2
- Próxima aula:
  - **Aula 07 – Caminho de dados e controle 1**