

# “Arquitetura e Organização de Computadores I – Aula\_06 – Desempenho de Computadores”

Prof. Dr. Emerson Carlos Pedrino  
DC/UFSCar  
São Carlos





# Objetivos

- Definição de desempenho
- Aplicação na avaliação de computadores



# Apresentação

- O desempenho dos computadores tem melhorado ao longo do tempo.
- Definindo o desempenho, apresentando formas de medi-lo, ou comparando computadores quanto a seu desempenho, podemos assim entender por que alguns *hardwares* são melhores que outros para programas diferentes, quais fatores de desempenho do sistema são relacionados ao *hardware*, e quais, ao SO.



# Outra Questão

- Como a arquitetura do conjunto de instruções (*Instruction Set Architecture - ISA*) de uma máquina afeta o desempenho.



# Comparações de Aviões Comerciais

- Na Tabela 1 a seguir, alguns aviões comerciais (não necessariamente em atividade) são comparados entre si quanto à capacidade de assentos para passageiros, autonomia de vôo em milhas, velocidade em milhas por hora.



# Tabela 1: Capacidade, autonomia e velocidade de aviões comerciais

| Avião            | Capacidade (passageiros) | Autonomia (milhas) | Velocidade (milhas/hora) |
|------------------|--------------------------|--------------------|--------------------------|
| Boeing 777       | 375                      | 4.630              | 610                      |
| Boeing 747       | 470                      | 4.150              | 610                      |
| BAC/Sud Concorde | 132                      | 4.000              | 1.350                    |
| Douglas DC-8-50  | 146                      | 8.720              | 544                      |

Fonte: adaptada de Hennessy & Patterson (2005).



# *Throughput*

- Na Tabela 2, a seguir, é apresentado o *throughput* em passageiros x milha por hora.
- *Throughput* é uma medida de tarefa realizada por unidade de tempo. No exemplo a seguir, é a quantidade de passageiros transportados numa distância correspondente a uma milha, no tempo correspondente a uma hora.



## Tabela 2: *Throughput* para os aviões comerciais da Tabela 1.

| Avião            | <i>Throughput</i><br>(passageiros x milhas/hora) |
|------------------|--|
| Boeing 777       | 228.750  |
| Boeing 747       | 286.700  |
| BAC/Sud Concorde | 178.200  |
| Douglas DC-8-50  | 79.424   |

Obs.: É possível obter essa medida multiplicando a capacidade de passageiros pela velocidade (ver Tabela 1).





# Qual é o melhor avião?

- Essa resposta não é trivial. Se o critério de comparação é a velocidade, o Concorde é o melhor. Se o critério é a capacidade de passageiros, o Boeing 747 é o melhor. E se levar em consideração a autonomia de vôo, o DC-8-50 é o vencedor.



# Outra Questão

- Outra questão é a medida de quão rápido é um avião em relação a outro, ou quanto um avião é maior em capacidade de passageiros em relação a outro.
- Tais questões serão analisadas em relação aos computadores.



# Desempenho de Computadores

- Principais conceitos para a medida de desempenho em computadores: *benchmark* e a lei de Amdahl, que permite obter o ganho em *speed-up*, em função de uma melhoria de desempenho numa parte do computador.



# Medida de Tempo

- Medida de tempo de resposta -> Latência.
- Exemplos de medidas de tempo em computação:
  - tempo para executar a tarefa do usuário;
  - tempo gasto para executar uma tarefa;
  - tempo dispendido para uma consulta a um banco de dados.



# *Throughput*

- Quantidade de tarefas realizadas.
- Exemplos:
  - quantas tarefas a máquina pode executar de uma vez;
  - taxa média de execução;
  - quantidade de trabalho realizado.



# Impacto nas medidas

- Se atualizarmos uma máquina com um novo processador.
- Se adicionarmos uma nova máquina no laboratório.



# Tempo decorrido

- Medida de tempo que conta todas as operações feitas no computador, envolvendo acesso à memória e disco, E/S (Entrada/Saída), etc.
- Tal medida é útil, mas nem sempre boa para o propósito de comparação.



# Tempo de CPU

- Não conta a E/S ou o tempo dispendido rodando outros programas e pode ser dividido em tempo do sistema e tempo do usuário.
- Portanto, é o tempo dispendido executando linhas de código que estão no programa do usuário.



# Medida de Desempenho

- Desempenho -> inverso do tempo de execução.
- Assim, para um certo programa que roda numa máquina X, temos:

$$desempenho_x = \frac{1}{tempo\ de\ execu\c{c}{\tilde{o}}_x}$$



# Medida de Desempenho

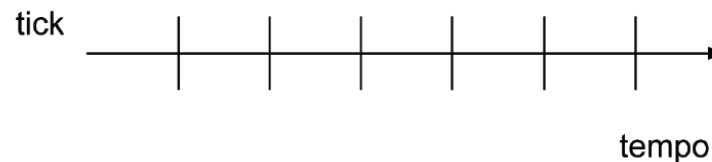
- Comparando de quanto uma máquina X é mais rápida que uma máquina Y:

$$n = \frac{\text{desempenho}_x}{\text{desempenho}_y}$$

- n: quantidade de vezes que X é mais rápida que Y. Se  $n < 1$ , X é mais lenta que Y.

# Ciclos de *clock*

- Tempo de processador: função de um conceito conhecido como ciclo de *clock*.
- O *clock* é uma sequência de pulsos que determina o ciclo básico da máquina.
- *Tick* de *clock*: indica quando se inicia uma atividade.





# Ciclos de *clock*

- Período do ciclo: tempo em segundos entre *ticks*.
- Taxa de *clock*: frequência que representa o inverso do tempo de ciclo (dada em hertz).

$$\text{taxa de clock} = \frac{1}{\text{tempo de ciclo}}.$$

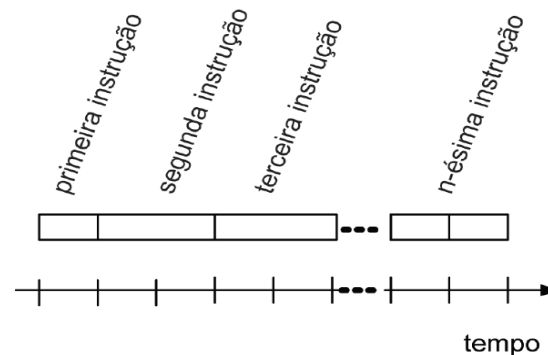
# Tempo de Processador

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

- O tempo de execução de um programa, em segundos, é igual ao produto do número de ciclos do programa pelo tempo de um ciclo em segundos.

# Ciclos por Programa

- Diferentes instruções podem gastar diferentes quantidades de tempo.



- Exemplos:
  - a multiplicação leva mais tempo que a soma;
  - operações de ponto flutuante levam mais tempo que operações de números inteiros;
  - acessar a memória leva mais tempo que acessar os registradores.



# Ciclos por Programa

- Também se tentarmos diminuir o tempo de ciclo, muitas instruções podem necessitar de mais ciclos para a sua execução.

# Exercício

- Um programa é executado em 10 s num computador A com  $clk = 400$  MHz. Um projetista deve construir uma máquina B, para executar o mesmo programa em 6 s. Também, deverá ser utilizada uma nova tecnologia para aumentar substancialmente a taxa de  $clk$ , que afetará o resto do projeto da CPU, obrigando a máquina B a ter 1,2 x o número de ciclos de  $clk$  que A para o mesmo programa. Que taxa de  $clk$  o projetista deverá usar como meta?



# Solução

- Aplicar a equação que relaciona o tempo de execução de um programa em termos de ciclos de *clk*.

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

$$\text{A:} \quad 10 \text{ s} = \frac{\text{ciclos}}{\text{programa}} \cdot \frac{1}{400 \text{ MHz}} \rightarrow \frac{\text{ciclos}}{\text{programa}} = 10 \text{ s} \cdot 400 \text{ MHz}$$

$$\text{B:} \quad 6 \text{ s} = \frac{\text{ciclos}}{\text{programa}} \times \frac{1}{\text{taxa de clock}} \rightarrow 6 \text{ s} = 1,2 \cdot (10 \text{ s} \cdot 400 \text{ MHz}) \cdot \frac{1}{\text{taxa de clock}}$$

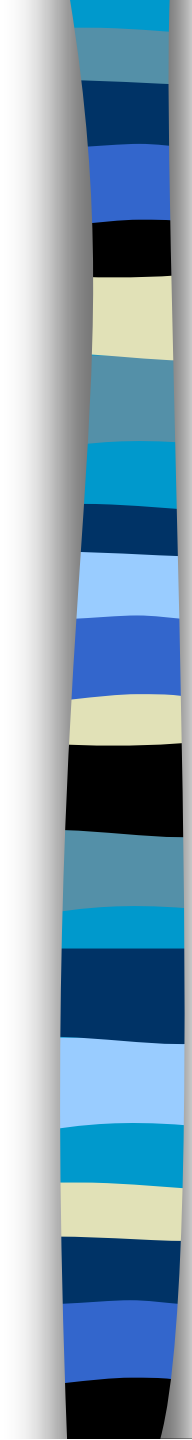


$$\text{taxa de clock} = 1,2 \cdot (10 \text{ s} \cdot 400 \text{ MHz}) \cdot \frac{1}{6 \text{ s}} = 800 \text{ MHz}.$$



# Ciclos por Instrução (CPI)

- Medida muito útil na análise de desempenho.
- Um programa com intenso uso de instruções de ponto flutuante deve ter alto CPI.
- Quanto maior o CPI, menor a quantidade de instruções executadas pelo computador por unidade de tempo.



MIPS (milhões de instruções por segundo):  
 $\text{num\_instrs}/(\text{temp\_exec} * 10^6)$  – Tentativa de padronizar uma medida de desempenho de um computador

- *Throughput* de instruções.
- Quanto mais simples forem as instruções, maior será o valor do MIPS.
- Obs.: Não confundir a métrica MIPS com o computador MIPS. Não se pode comparar computadores com diferentes *conjuntos de instruções* (não leva em consideração o tipo de instrução). Varia entre programas num mesmo computador (não é possível ter um único índice MIPS para todos os programas). Por fim, também pode variar inversamente com o desempenho.



# Exercício sobre CPI

- Considere duas implementações (A e B) com o mesmo conjunto de instruções. Para um dado programa, a máquina A tem um tempo de ciclo de  $10\text{ ns}$  e CPI de 2.0, e a máquina B tem um tempo de ciclo de  $20\text{ ns}$  e CPI de 1.2. Qual máquina é mais rápida para esse programa, e quanto?



# Solução

## ■ Considerações Iniciais:

- As duas máquinas têm o mesmo conjunto de instruções -> o número  $X$  de instruções será igual para A e B rodando esse programa.
- CPI -> indica o número médio de ciclos por instrução, assim, é possível calcular o número de ciclos em função de  $X$ .

# Solução

A é mais rápida.

*tempo de programa A* =  $X \cdot 2,0 \cdot 10 \text{ ns} = X \cdot 20 \text{ ns}$ ,

*tempo de programa B* =  $X \cdot 1,2 \cdot 20 \text{ ns} = X \cdot 24 \text{ ns}$ .

$$\frac{\text{desempenho A}}{\text{desempenho B}} = \frac{\frac{1}{\text{tempo de programa A}}}{\frac{1}{\text{tempo de programa B}}} = \frac{X \cdot 24 \text{ ns}}{X \cdot 20 \text{ ns}} = 1,2.$$

Quanto?



# Exercício sobre o número de instruções

- Um projetista está tentando decidir entre duas sequências de código para uma máquina particular. Numa implementação de *hardware*, existem três classes diferentes de instruções: A, B e C, que requerem 1, 2 e 3 ciclos, respectivamente. A primeira sequência tem 5 instruções: 2 de A, 1 de B e 2 de C, e a segunda sequência tem 6 instruções: 4 de A, 1 de B e 1 de C. Qual sequência será mais rápida? Quanto? Qual o CPI para cada sequência?

# Solução

- $\text{num\_ciclos\_seq\_1}: 2 \times 1 + 1 \times 2 + 2 \times 3 = 10$  ciclos.
- $\text{num\_ciclos\_seq\_2}: 4 \times 1 + 1 \times 2 + 1 \times 3 = 9$  ciclos.
- Portanto, a sequência 2 é mais rápida.
- Quanto?  $10/9=1,1$ .
- $\text{CPI\_1}=10/5=2$  e  $\text{CPI\_2}=9/6=1,5$ .
- Obs.: o exercício mostra o risco de usar apenas um fator ( $\text{num\_instr}$ ,  $\text{CPI}$  e  $\text{clk}$ ) para avaliar o desempenho. O tempo sempre será a medida mais confiável para avaliar o desempenho de um computador.  $\text{Tempo} = \text{seg/prog} = \text{instr/prog} \times \text{ciclos\_clk/instr} \times \text{seg/ciclos\_clk}$ .





# Exercício sobre MIPS

- Dois diferentes compiladores estão sendo testados numa máquina de 100 MHz com 3 diferentes classes de instruções: A, B e C, que requerem 1, 2 e 3 ciclos, respectivamente. Ambos os compiladores são usados para produzir códigos para uma parte grande de *software*. O primeiro código compilado usa 5 milhões de instruções A, 1 milhão de instruções B e 1 milhão de C. O segundo código compilado usa 10 milhões de instruções A, 1 milhão de instruções B e 1 milhão de C. Qual sequência será mais rápida, levando-se em consideração a métrica MIPS? E em relação ao tempo de execução?



# Solução

- Calculando o número de ciclos para os compiladores:
  - C1:  $5 \text{ milhões} \times 1 + 1 \text{ milhão} \times 2 + 1 \text{ milhão} \times 3 = 10 \text{ milhões de ciclos.}$
  - C2:  $10 \text{ milhões} \times 1 + 1 \text{ milhão} \times 2 + 1 \text{ milhão} \times 3 = 15 \text{ milhões de ciclos.}$



# Solução

- Calculando o tempo de programa para cada código usando:  
(seg/progr=ciclos/progr x 1/taxa\_ciclo)
  - C1:  $\text{seg/progr} = 10 \text{ milhões} \times 1 / 100 \text{ MHz}$   
 $= 0,1 \text{ s.}$
  - C2:  $\text{seg/progr} = 15 \text{ milhões} \times 1 / 100 \text{ MHz}$   
 $= 0,15 \text{ s.}$



# Solução

- Usando a métrica MIPS:

- C1: MIPS = 7 milhões / 0,1 s = 70.
- C2: MIPS = 12 milhões / 0,15 s = 80.

Portanto, o código 2 é mais rápido de acordo com a métrica MIPS. Entretanto, de acordo com o tempo de execução, o código 1 é mais rápido.



# *Benchmarks*

- *Benchmarks*: programas de teste usados para determinação de desempenho, executando aplicações reais, típicos de carga de trabalho (*workload*), ou programas típicos de classes de aplicações desejadas, como compiladores/editores, aplicações científicas, gráficos, etc.



# *Benchmarks*

- A melhor escolha de *benchmarks* para medir o desempenho são as aplicações reais, como um compilador, por exemplo.
- As tentativas de executar programas muito mais simples podem levar a armadilhas de desempenho.
- O arquiteto do compilador pode conspirar para fazer com que o computador pareça mais rápido nesses programas.
- *Benchmarks* pequenos são atraentes no início do projeto, já que são pequenos o bastante para compilar e simular facilmente. Algumas vezes, na ânsia de produzir código altamente otimizado para *benchmarks*, podem ser introduzidas otimizações errôneas.



# *Benchmarks*

- Um benchmark muito usado é o SPEC (System Performance Evaluation Cooperative).
- Primeiro turno: 1989. 10 programas produzindo um único número “SPECmarks”.
- Segundo turno: 1992. Programas SPECint92 (6 programas em inteiros) e SPECfp92 (14 programas em ponto flutuante).



# *Benchmarks*

- Terceiro turno: 1995. SPECint95 (8 programas em inteiros) e SPECfp95 (10 programas em ponto flutuante).
- Os quarto e quinto turnos foram lançados em 2000 e 2006, respectivamente, atualizando o conjunto de programas anteriores.





# *Benchmarks*

- CINT2006: serve para medir e comparar o desempenho de processamento de inteiros.
- CPF2006: serve para medir e comparar o desempenho de processamento de ponto flutuante.
  - Referências: [www.spec.org](http://www.spec.org)



# Lei de Amdahl

- **Define o tempo de execução novo** (quando se faz um aperfeiçoamento para melhorar o desempenho de uma máquina), **em função do tempo de execução antigo** (dividido em tempo relativo à parte não afetada pelo aperfeiçoamento e à parte afetada).



# Observações

- Tempo de execução antigo da parte não afetada: contribui da mesma forma no tempo de execução novo.
- Tempo de execução antigo da parte afetada: é dividido pela razão de melhoramento, que corresponde a quanto o aperfeiçoamento melhorou o desempenho.

# Fórmula

$$\begin{aligned} \textit{Tempo de Execução}_{\textit{novo}} = & \textit{Tempo de Execução}_{\textit{parte não afetada}} \\ & + \frac{\textit{Tempo de Execução}_{\textit{parte afetada}}}{\textit{Razão de Melhoramento}} \end{aligned}$$



# Exercício

- Suponha que um programa é executado em 100 s numa máquina, com instruções de multiplicação responsáveis por 80 s desse tempo. Quanto devemos melhorar a velocidade das instruções de multiplicação se desejamos que o programa seja executado 4 x mais rápido? E para executá-lo 5 x mais rápido?



# Solução

- Tempo de execução da parte afetada = 80 s.
- Tempo de execução da parte não afetada = 20 s.
- Velocidade de execução nova = 4 x mais rápida =  $100 / 4 = 25$  s.

# Solução

$$25 \text{ s} = 20 \text{ s} + \frac{80 \text{ s}}{\text{razão de melhoramento}}.$$



$$\text{razão de melhoramento} = \frac{80 \text{ s}}{25 \text{ s} - 20 \text{ s}} = \frac{80 \text{ s}}{5 \text{ s}} = 16.$$



E para executar o programa 5 x mais rápido?

$$\textit{raz\~ao de melhoramento} = \frac{80\text{ s}}{20\text{ s} - 20\text{ s}} = \frac{80\text{ s}}{0\text{ s}} = \infty,$$

- Tal resultado mostra que existe um limite para a reduo do tempo, acima do qual chegamos a uma razo de melhoramento impossvel.





## \*Exercícios Complementares

- 1. Suponha que uma máquina foi melhorada fazendo todas as instruções de ponto flutuante serem executadas 5 x mais rápido. Se o tempo de execução de certo *benchmark* antes do melhoramento é de 10 s, qual seria o *speed-up* se metade dos 10 s é dispendida em instruções de ponto flutuante?



## \*Exercícios Complementares

- 2. Estamos procurando um *benchmark* para testar a nova unidade de ponto flutuante acima, e queremos que este mostre um *speed-up* de 3. Um *benchmark* é executado em 100 s, com o antigo *hardware* de ponto flutuante. A quanto do tempo de execução as instruções de ponto flutuante devem corresponder nesse programa para que possamos produzir o *speed-up* desejado nesse *benchmark*?



# \*Exercícios Complementares

- Resolver a seguinte lista de exercícios do livro texto:
  - 4.1, 4.2, 4.7, 4.8 e 4.10.