Avaliação do Desempenho

Arquitectura de Computadores
Lic. em Engenharia de Sistema e Informática
2008/09
Luís Paulo Santos

Avaliação do Desempenho

Conteúdos	7.1 – Tempo de Execução	C1
	7.2 – Ciclos por Instrução (CPI)	C1
	7.3 – Ciclos por Elemento (CPE)	C1
Resultados de Aprendizagem	R7.1 – Identificar e caracterizar as métricas relativas ao desempenho da execução de programas	C1
	R7.2 – Utilizar modelos quantitativos para prever/avaliar o desempenho da máquina	C1

Desempenho: o que é?

- Quando escolhemos o melhor sistema (i.e., com melhor desempenho) é necessário especificar o melhor **em quê**!
- Sistemas de Computação (hw + sw):
  - Tempo de execução
  - Débito (*throughput*)
- Aceitemos o desempenho de um sistema de computação X como estando relacionado com o tempo de execução de X (X implica equipamento, *hw*, e aplicação, *sw*):

$$\text{Desempenho}_X = \frac{1}{T_{\text{exec X}}}$$

donde, se $\text{Desempenho}_X > \text{Desempenho}_Y$, então

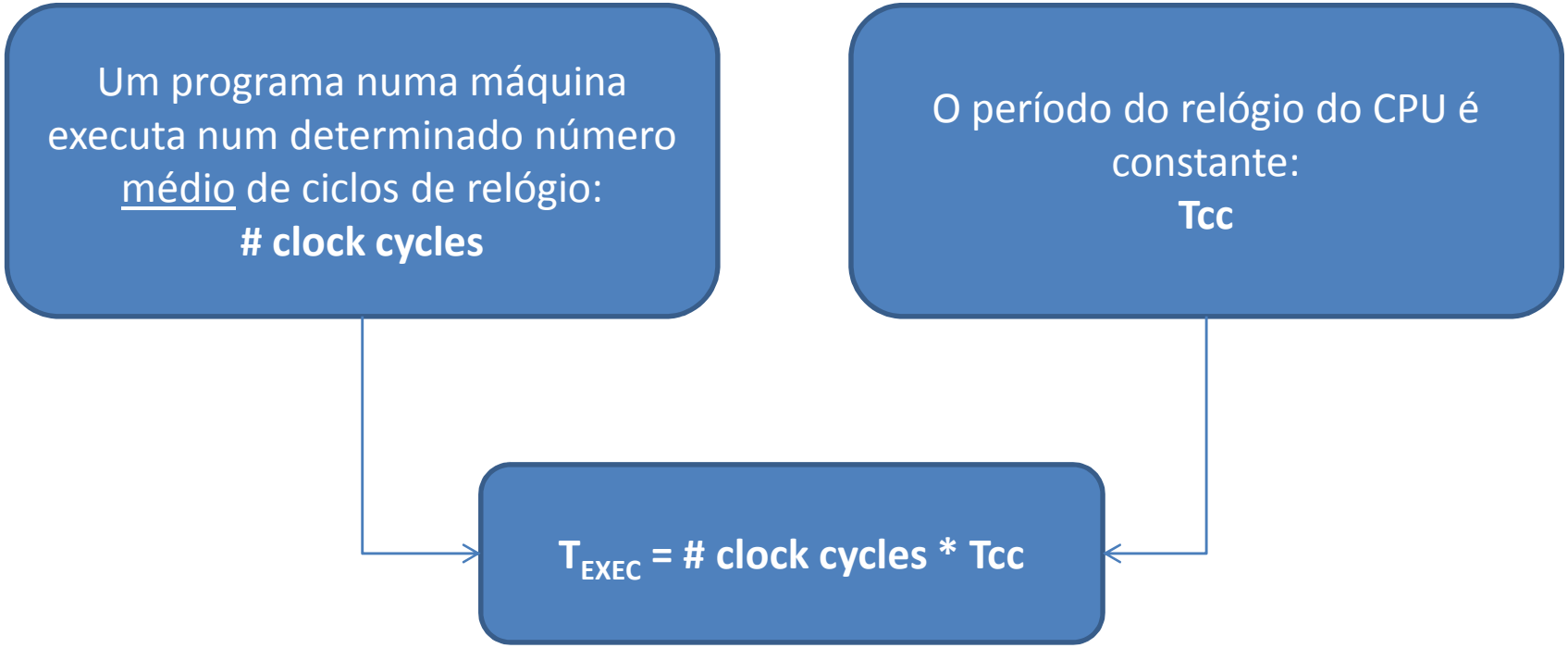
$$T_{\text{exec X}} < T_{\text{exec Y}}$$

Avaliação do Desempenho: para quê?

- Desenho de sistemas de computação/processadores
- Optimização de software
- Selecção/aquisição de um sistema de computação

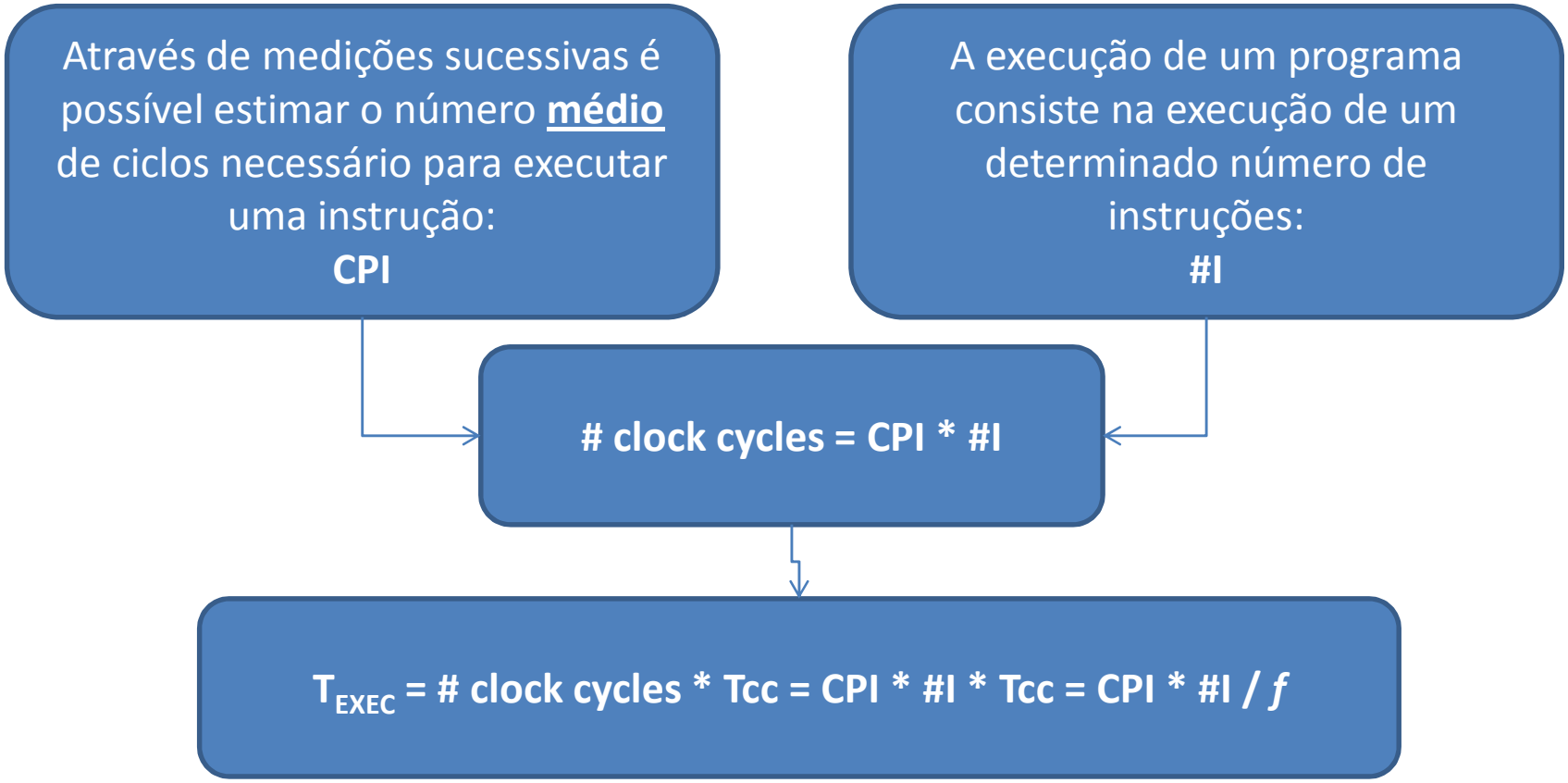
Desempenho do CPU

- Para prever o desempenho (T_{EXEC} – tempo de execução) de um dado programa num determinado CPU é necessário um **modelo** que relacione o desempenho com as características do sistema de computação (*hw+sw*)



Desempenho do CPU

- De que depende o número médio de ciclos necessários para executar um programa?



Desempenho do CPU

$$T_{EXEC} = CPI * \#I / f$$

O CPI é um valor médio, logo pode ser medido com diferentes precisões.

A aproximação mais grosseira será dizer que uma máquina apresenta um determinado CPI, independentemente do tipo de instruções.

Diferentes tipos de instruções exibem valores de CPI diferentes:

- Divisões exigem mais ciclos do que adições ou multiplicações
- Acessos à memória exigem mais ciclos do que acessos a registos
- Operações em vírgula flutuante podem exigir mais ciclos do que operações com inteiros

Desempenho do CPU

- Um programador quer escolher entre dois segmentos de código diferentes para um mesmo algoritmo. Qual o mais rápido?

Tipo de Instrução	CPI
A	1
B	2
C	3

Código	Número de Instruções		
	A	B	C
1	2000	1000	100
2	100	1000	1000

$$T_{EXEC1} = (1*2000 + 2*1000 + 3*100)/f = 4300/f$$

$$T_{EXEC2} = (1*100 + 2*1000 + 3*1000)/f = 5100/f$$

Desempenho do CPU

- Calcule o tempo de execução do programa abaixo numa máquina com um relógio de 2 GHz e CPI=1.5

```
  movl 10, %eax
  movl 0, %ecx
ciclo:
  addl %eax, %ecx
  decl %eax
  jnz ciclo
```

#I = 32

NOTA: O número de instruções a considerar é o número de instruções **executadas.**

Texec = 32 * 1.5 / 2E9 = 24E-9 s = 24 ns

Relação entre as métricas

$$T_{EXEC} = CPI * \#I / f$$

- #I – depende do algoritmo, do compilador e da arquitectura (ISA)
- CPI – depende da arquitectura (ISA), da mistura de instruções efectivamente utilizadas, da organização do processador e da organização dos restantes componentes do sistema (ex., memória)
- *f* – depende da organização do processador e da tecnologia utilizada

"A única métrica completa e fiável para avaliar o desempenho de um computador é o tempo de execução"

As métricas CPI, *f* e #I não podem ser avaliadas isoladamente, devendo ser sempre consideradas em conjunto, pois dependem umas das outras.

Relação entre as métricas

Exemplo 1 : Aumentar a frequência do relógio (diminuir Tcc) implica frequentemente um aumento do CPI!

Explicação: Entre outros factores, deve-se considerar o tempo de acesso à memória (Tmem). Se Tcc diminui, mas Tmem se mantém, então serão necessários mais ciclos para aceder à memória.

$f_1 = 1GHz$

$T_{cc1} = 1ns$

$T_{mem} = 40ns$

$Ciclos_{mem1} = 40$

$f_2 = 2GHz$

$T_{cc2} = 0.5ns$

$T_{mem} = 40ns$

$Ciclos_{mem2} = 80$

Conclusão: Apesar de Tcc diminuir para metade, Texec não diminui para metade, pois o número de ciclos de acesso à memória aumenta.

Relação entre as métricas

Exemplo 2 : Diminuir o número de instruções (#I) recorrendo a instruções mais complexas resulta num aumento do CPI!

Explicação: As instruções mais complexas realizam o trabalho de várias instruções simples, mas podem necessitar de mais ciclos para o completar, resultando num aumento do CPI. Este é um dos argumentos dos defensores de arquitecturas RISC.

Conclusão: O número de instruções diminui, mas o ganho em tempo de execução não diminui na mesma proporção, devido ao aumento do CPI.

Desempenho do CPU - MIPS

MIPS (milhões de instruções por segundo) – uma métrica enganadora

$$\text{MIPS nativo} = \frac{\#I}{T_{exec} * 10^6}$$

1. MIPS especifica a taxa de execução das instruções, mas não considera o trabalho feito por cada instrução. CPUs com diferentes *instruction sets* não podem ser comparados.
2. MIPS varia entre diferentes programas no mesmo CPU
3. MIPS pode variar inversamente com o desempenho

Esta métrica pode ser usada para comparar o desempenho do mesmo programa em CPUs com o mesmo conjunto de instruções, mas micro-arquitecturas e/ou frequências do relógio diferentes.

Desempenho do CPU - MIPS

- Considere os seguintes segmentos de código executados numa máquina com *f* = 1 GHz. Qual o que exibe melhor desempenho de acordo com as métricas Texec e MIPS?

Código	Número de Instruções		
	A (CPI=1)	B (CPI=2)	C (CPI=3)
1	5	1	1
2	10	1	1

$$T_{exec1} = \frac{5+2+3}{10^9} = 10ns \qquad T_{exec2} = \frac{10+2+3}{10^9} = 15ns$$

$$MIPS_1 = \frac{7}{10*10^{-9}*10^6} = 700 \qquad MIPS_2 = \frac{12}{15*10^{-9}*10^6} = 800$$

Esta métrica favorece programas com muitas instruções simples e rápidas, pois não tem em consideração a quantidade de trabalho feita por cada uma.

Desempenho do CPU - MIPS

MIPS de pico (ou *peak* MIPS) – máxima taxa de execução de instruções

É a métrica mais enganadora, pois corresponde a sequências de código que apenas tenham instruções com o CPI mais baixo possível.
Este tipo de sequências de instruções não realizam, regra geral, trabalho útil; consistem apenas em operações elementares com operandos em registos.

Pode ser visto como “a velocidade da luz” do CPU, e portanto, inatingível.

O principal problema é que é muitas vezes publicitada pelos fabricantes/vendedores como uma medida de desempenho das suas máquinas!

Desempenho - CPE

- As métricas CPI e MIPS dependem do número de instruções máquina efectivamente executadas
- Para guiar um programador de uma linguagem de alto nível são necessárias métricas mais próximas do problema que se pretende resolver
- **CPE – Ciclos Por Elemento**

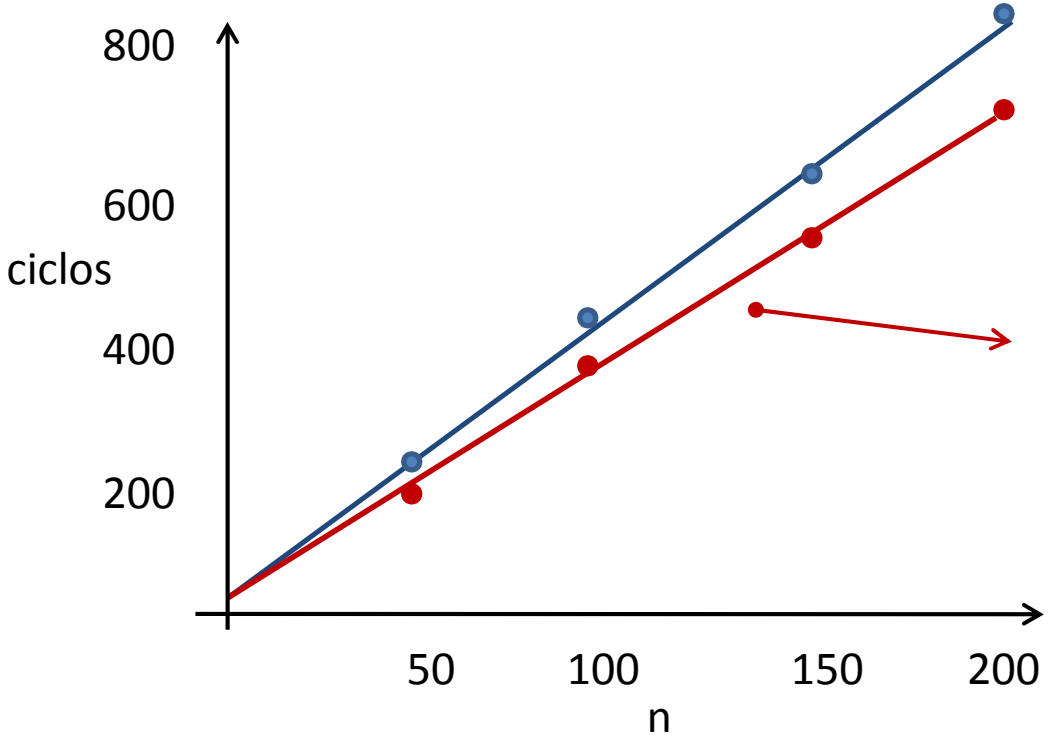
  “número médio de ciclos necessários para processar um elemento de dados”

  Ajuda a perceber o desempenho do ciclo de um programa iterativo
  Apropriada para expressar o desempenho de um programa que realiza uma operação repetitiva sobre diferentes elementos de dados:
  - Processar pixels numa imagem
  - Computar os elementos de uma matriz

Desempenho - CPE

```
void metade1 (int  *a, int n) {
  for (int i=0 ; i<n ; i++)
    a[i] = a[i] /2;
}
```

```
void metade2 (int  *a, int n) {
  for (int i=0 ; i<n ; i++)
    *a >>= 1;
}
```



Declive = CPE = 4.0

ClockCycles = 20 + **4.0** * n

Declive = CPE = 3.5

ClockCycles = 20 + **3.5** * n

NOTA: valores fictícios!

Desempenho - CPE

```
void metade1 (int  *a, int n) {
  for (int i=0 ; i<n ; i++)
    a[i] = a[i] /2;
}
```

Para n = 1000 -> ciclos = 4020

Qual o CPE?

Quantos ciclos por iteração?

```
void metade3 (int  *a, int n) {
  for (int i=0 ; i<n ; i+=2) {
    a[i] = a[i] /2;
    a[i+1] = a[i+1] /2;
  }
}
```

Para n = 1000 -> ciclos = 3820

Qual o CPE?

Quantos ciclos por iteração?

A utilização de **ciclos por elemento** e não **ciclos por iteração** dá uma indicação do tempo necessário para processar um vector de tamanho *n* independentemente da implementação.

Metodologia Medição de Desempenho

- Princípios
  - Isolar o mais possível factores externos
    - Considerar a sobrecarga (intrusão) do próprio processo de medição
    - Repetir várias vezes a medição
  - Documentar a experiência para que seja reprodutível por outros
    - Equipamento, versão do *software*, estado do sistema, …
  - Atenção ao relógio usado
    - Precisão: diferença entre o tempo medido e o tempo real
    - Resolução: unidade de tempo entre dois *ticks* do relógio

Metodologia Medição de Desempenho

- Qual o tempo a medir?
  - *Wall Time*
    - Tempo decorrido desde o início até ao fim do programa
    - Depende da carga do sistema (E/S, outros processos,…)
  - Tempo de CPU
    - Tempo efectivamente dedicado a este processo
    - Menos sensível à carga do sistema

- Comando `time`

  `time <comando>`

  `0.820u 0.300s 0:01.32 …`
  - 0,82 seg em “u”ser time
  - 0,30 seg em “s”ystem time
  - 1,32 seg de *wall time*

Metodologia Medição de Desempenho

Combinar o resultado de várias medições:

- Média das várias medições
  - Valores muito alto/baixos influenciam a média
  - Analisar também o desvio padrão (e.g., variações entre medições)
- Melhor medição
  - Valor obtido nas condições ideais
- Média das K-melhores medições
  - Média das k melhores execuções vezes
- Mediana
  - Mais robusto a variações nas medições

Metodologia Medição de Desempenho

Opções para medição do tempo

- `gettimeofday()`
  - Retorna o número de segundos desde 1-Jan-1970
  - Usa “Timer”ou o contador dos ciclos (depende da plataforma)
  - Resolução no melhor caso de 1us
- Contador de ciclos
  - Usa o *time stamp counter* do próprio processador, contando ciclos do relógio
  - Mede o *wall time*
  - Precisão muito elevada
  - Utilizar para medições <<1s
- Time (linha de comando)
  - Apenas usável para medições >>1seg