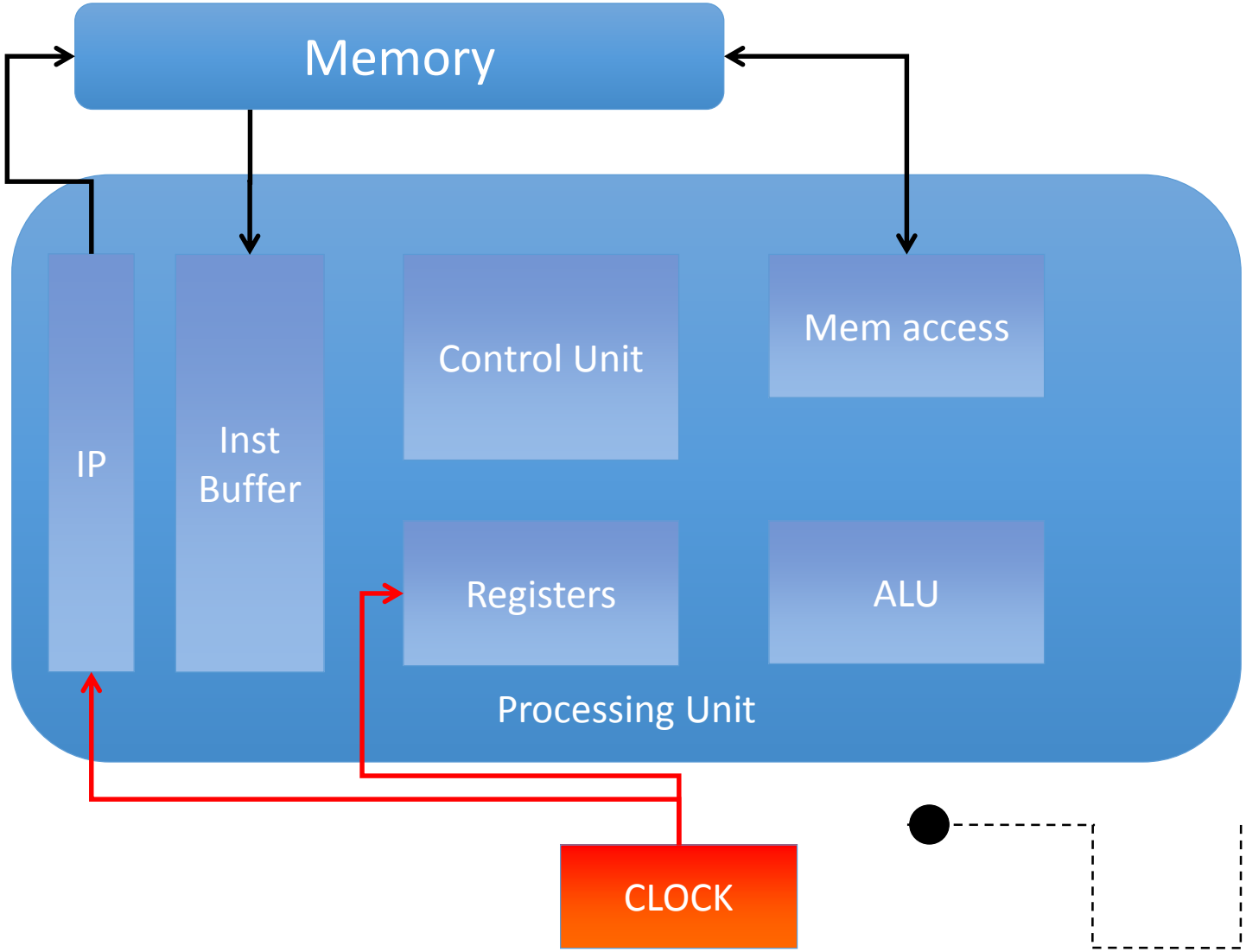
02 – Avaliação do Desempenho

Luís Paulo Santos
Arquitectura de Computadores
Universidade do Minho

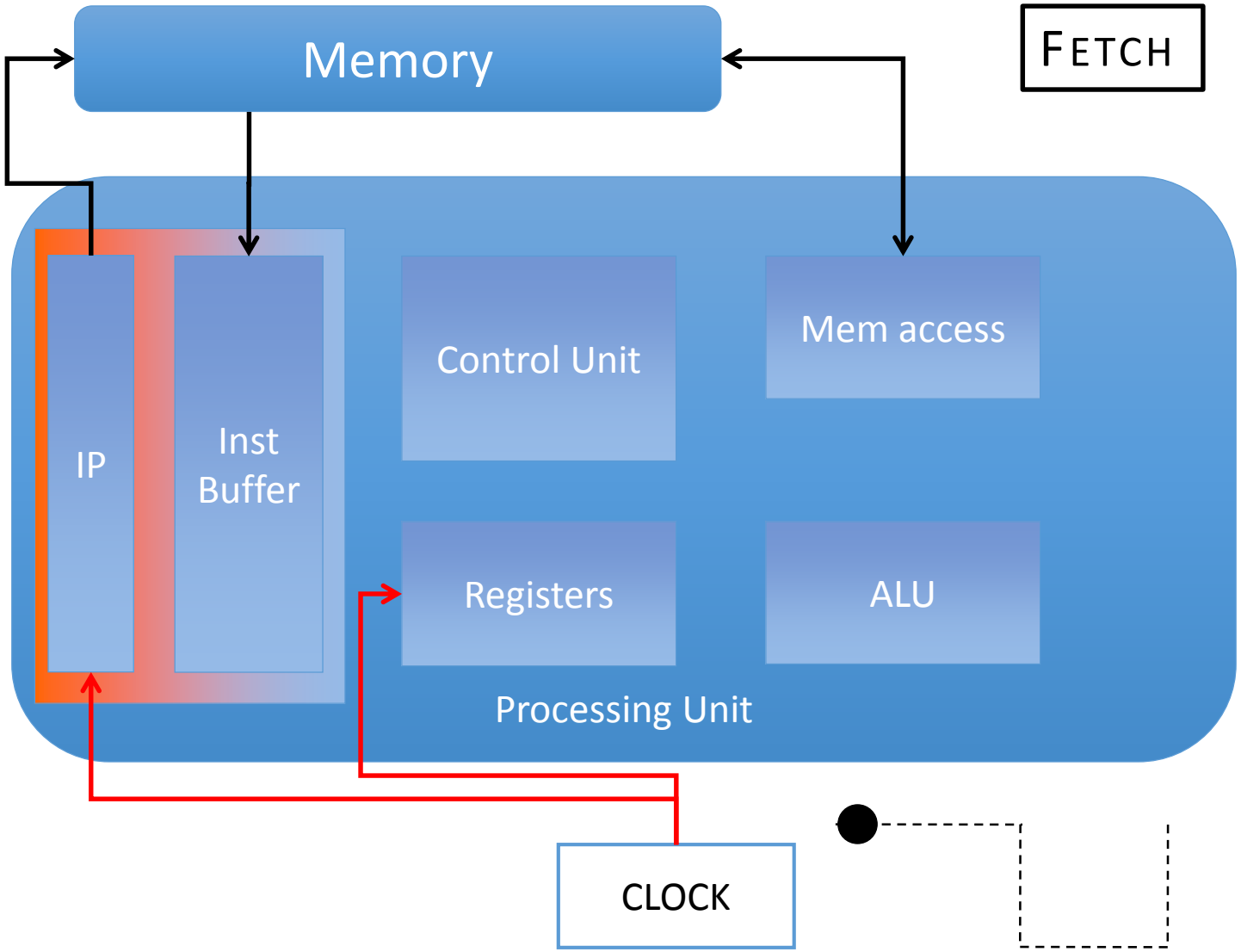
Material de apoio

- "*Computer Organization and Design: The Hardware / Software Interface*"
  David A. Patterson, John L. Hennessy
  5th Edition, 2013
  - Secção 1.6 (pags. 28 .. 40) – Performance
    Secção 1.10 (pags. 59 .. 51) – Fallacies and Pitfalls

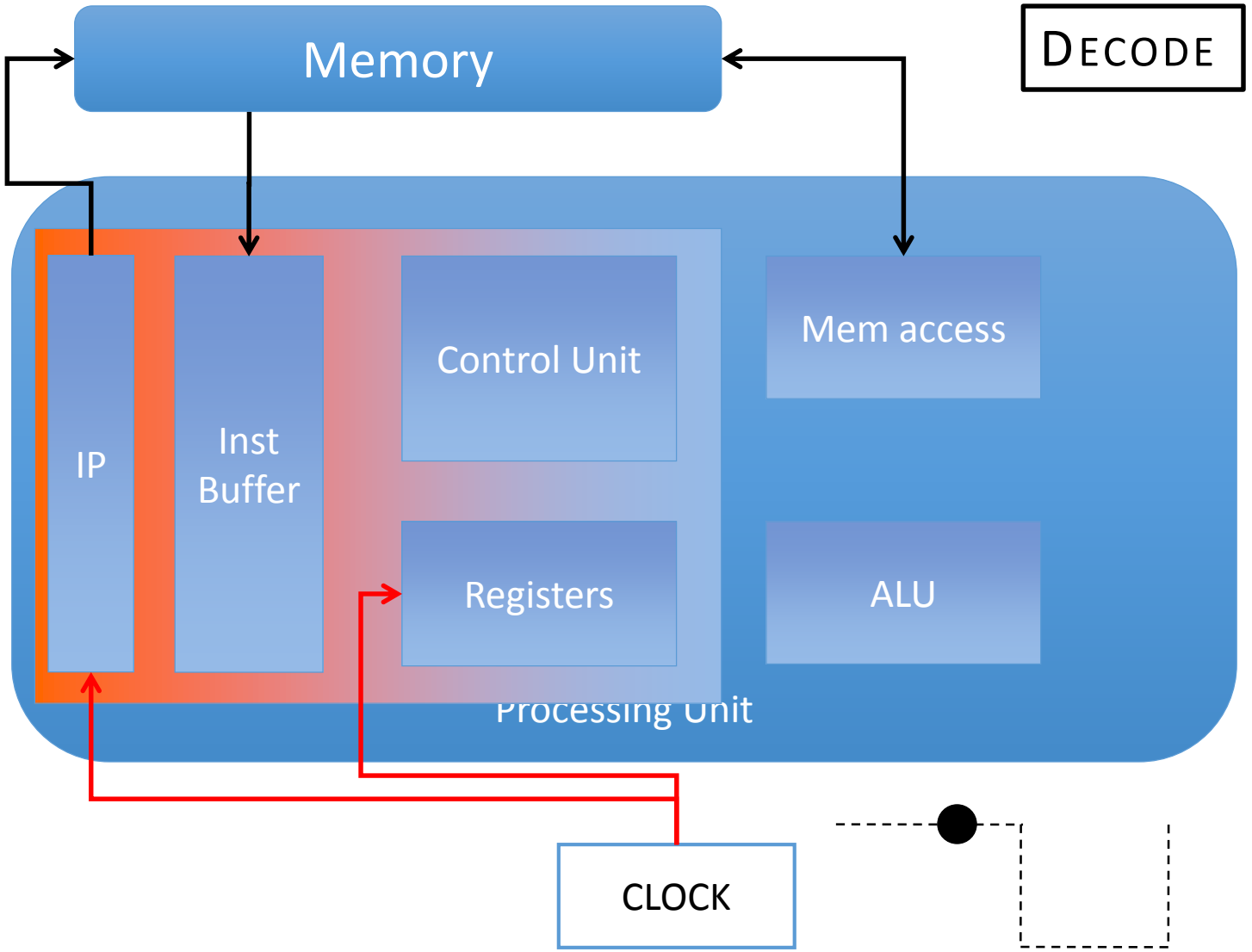
Avaliação Desempenho



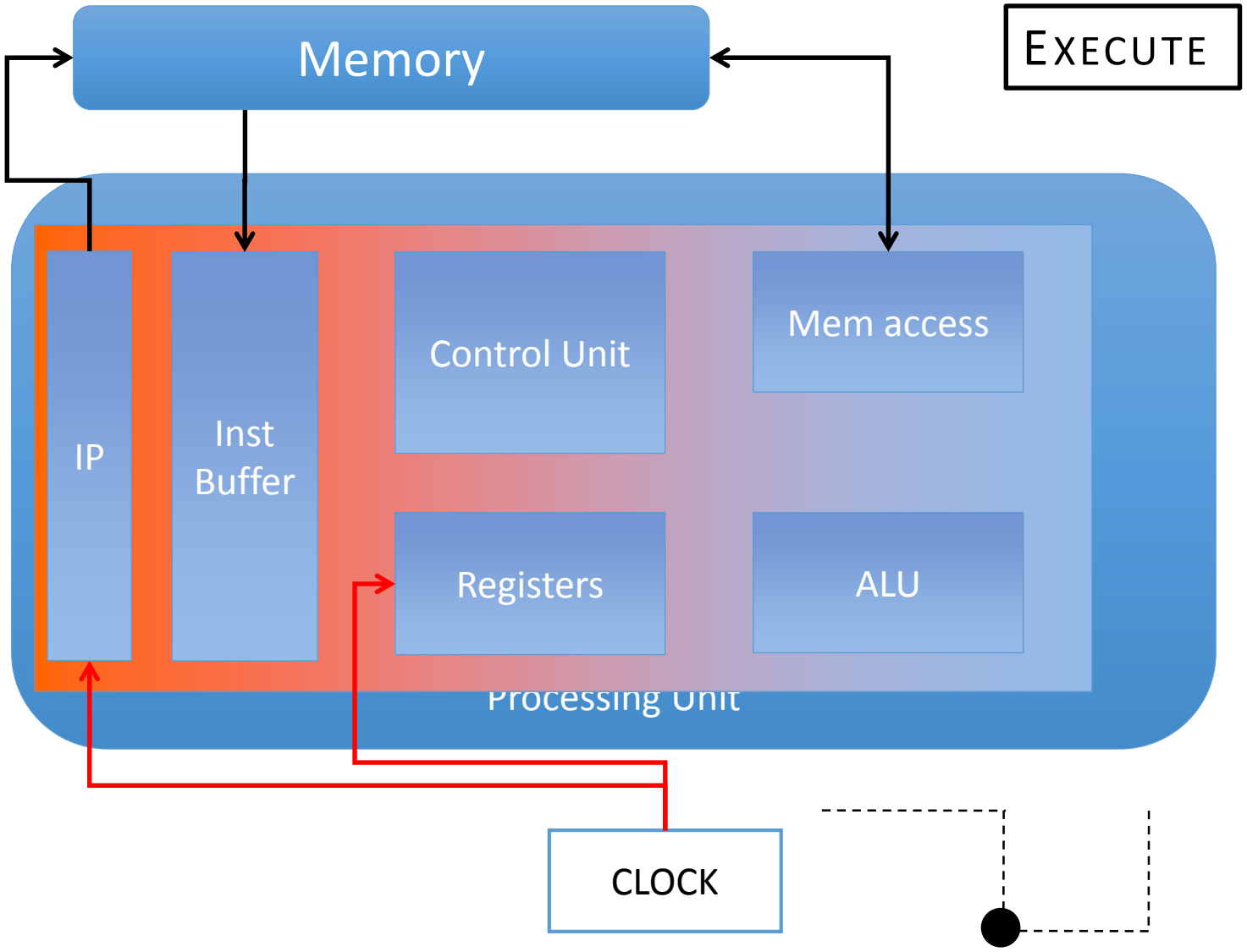
Avaliação Desempenho



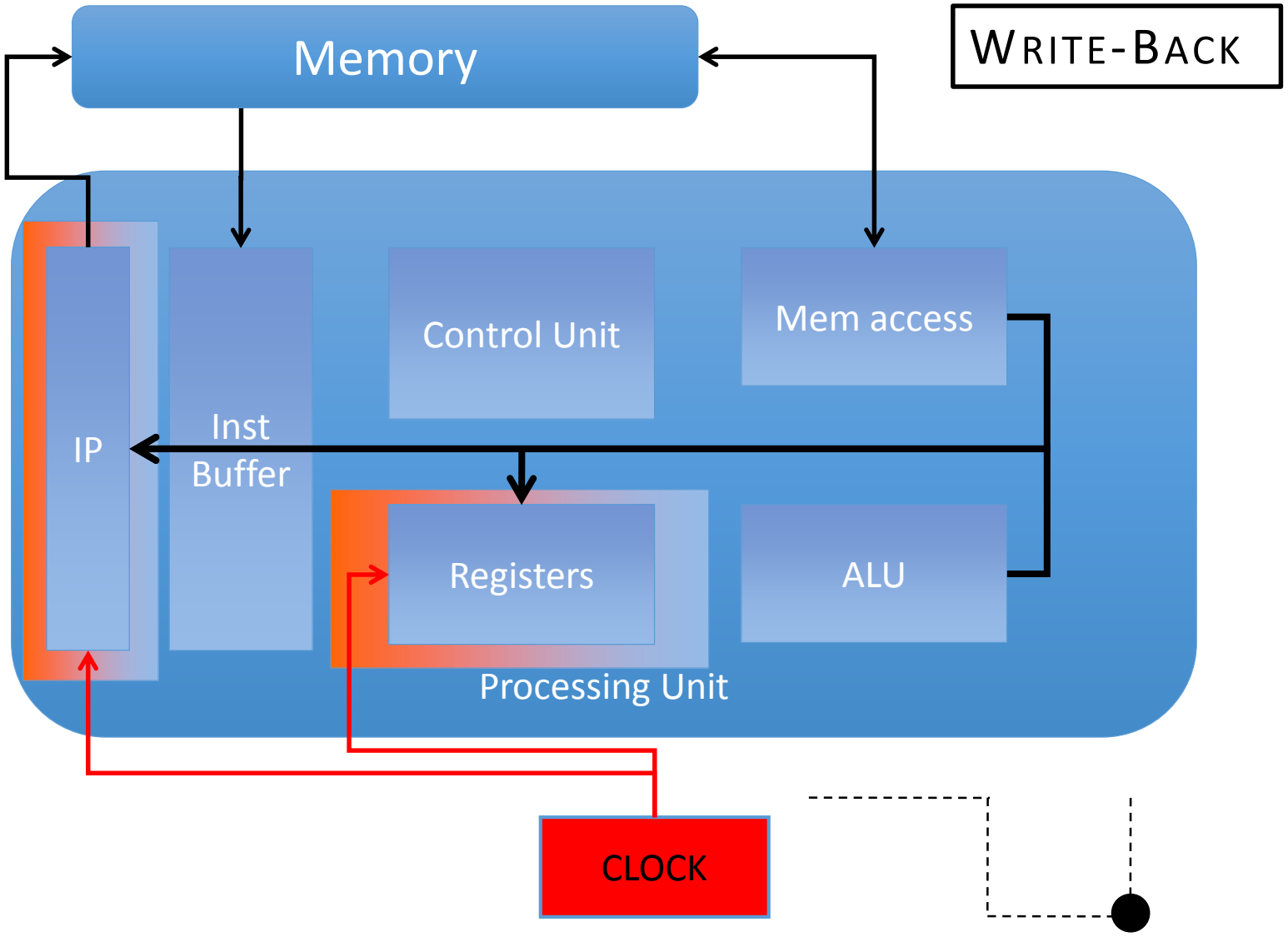
Avaliação Desempenho



Avaliação Desempenho

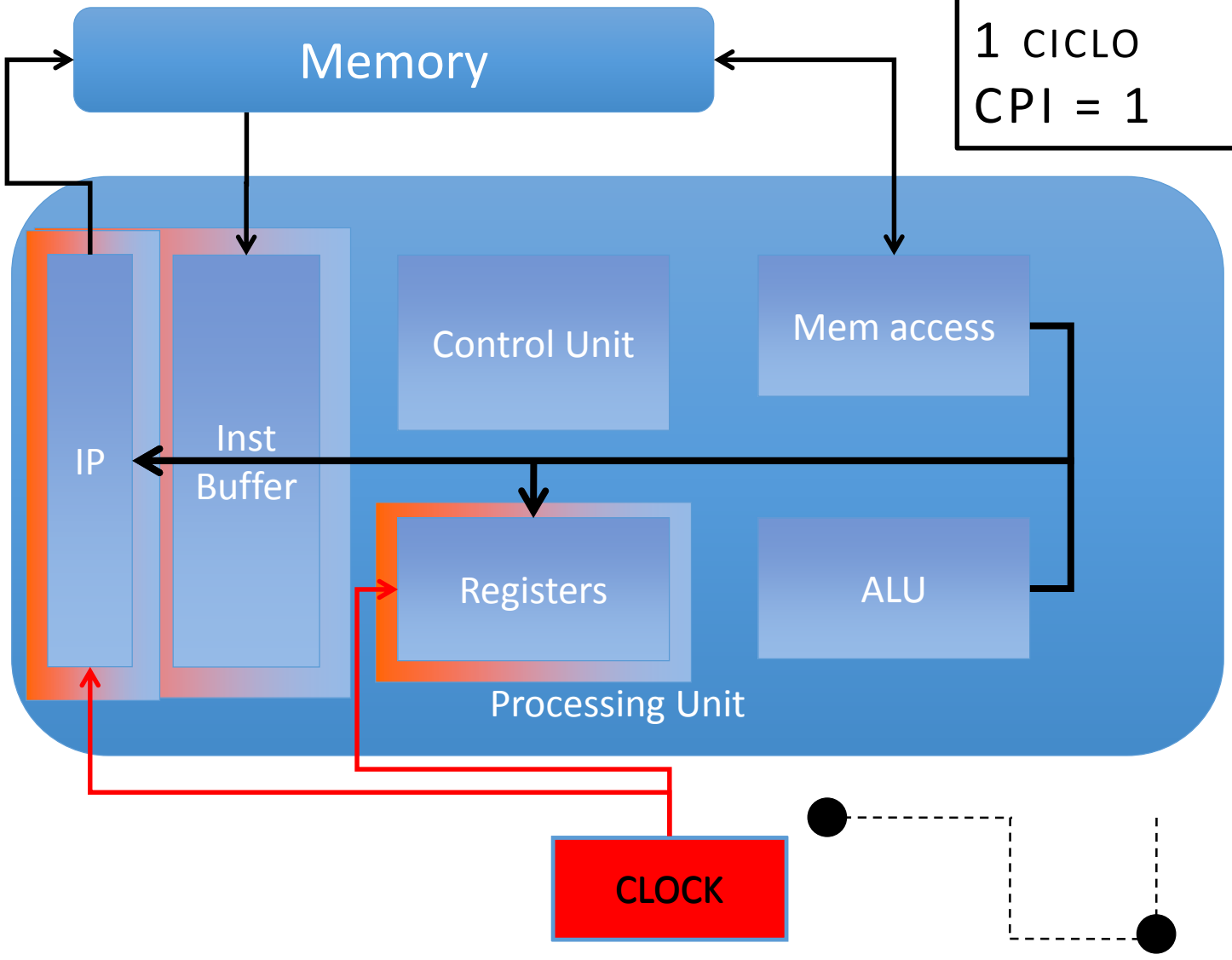


Avaliação Desempenho



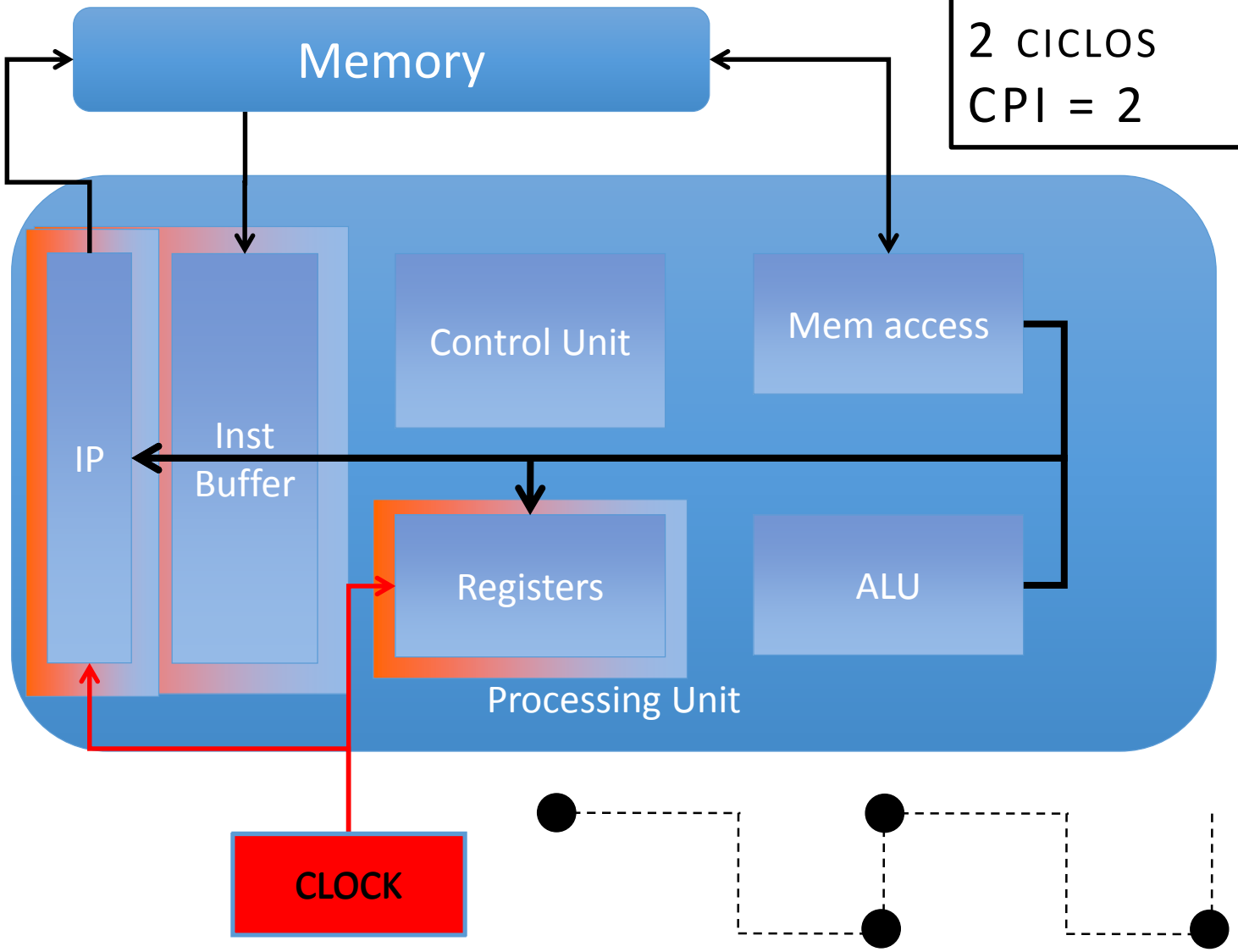
Avaliação Desempenho

NESTE CASO:
1 INSTRUÇÃO
1 CICLO
CPI = 1



Avaliação Desempenho

NESTE CASO:
1 INSTRUÇÃO
2 CICLOS
CPI = 2

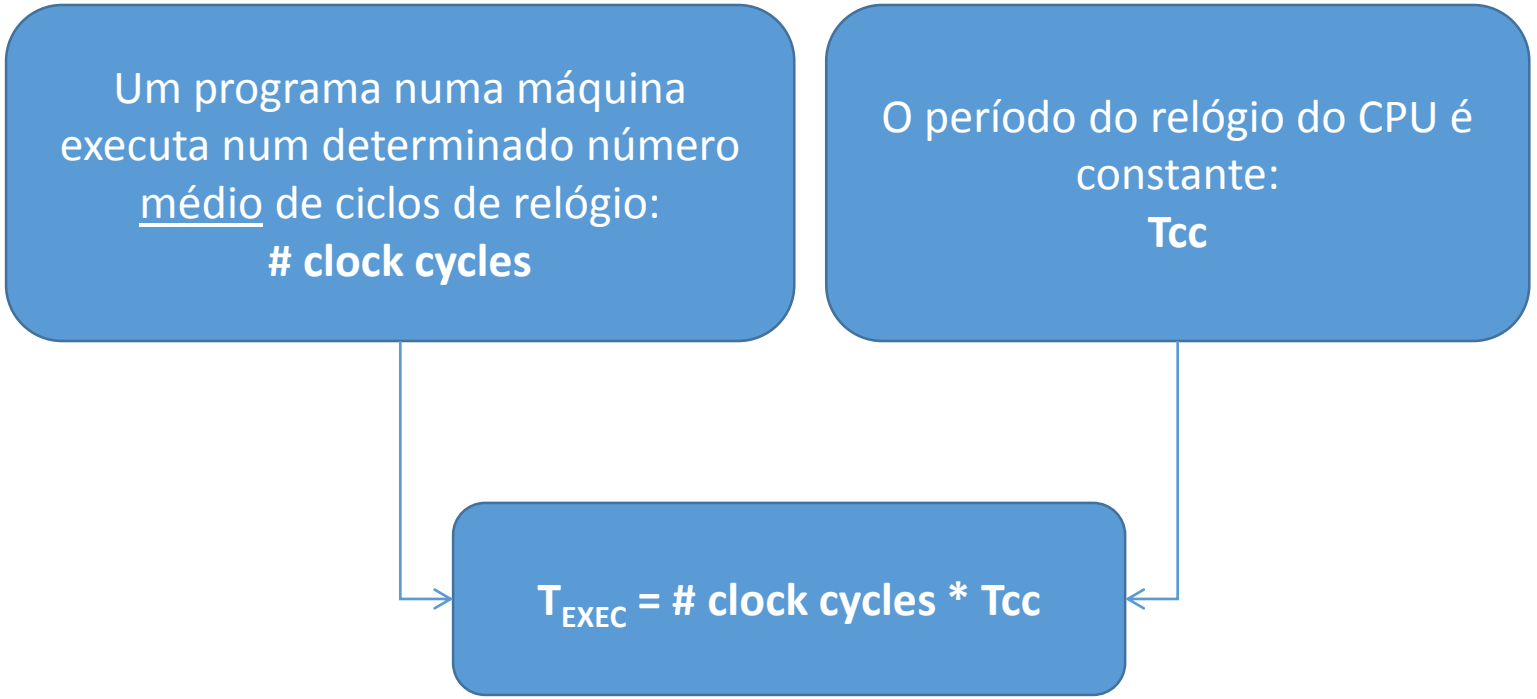


CPI – cycles per instruction

- Diferentes tipos de instruções exibem diferentes CPI:
  - CPI divisões > CPI adições
  - CPI acessos à memória > CPI acessos a registos
  - CPI operações vírgula flutuante >= CPI operações inteiras
- A mesma instrução pode requerer um número de ciclos diferente para diferentes estados da máquina
- CPI é um valor médio
- Pode ser medido com diferentes precisões.

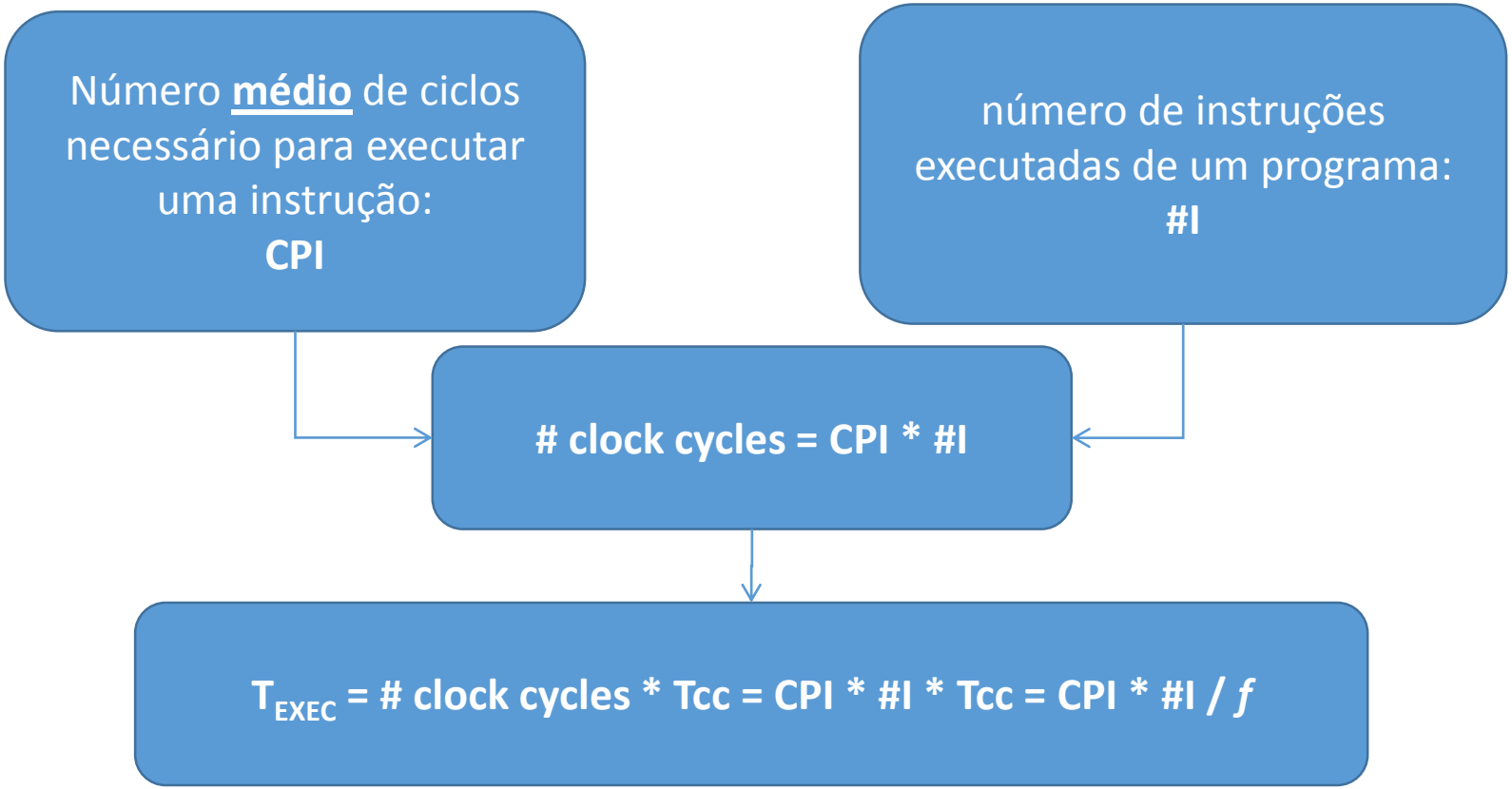
Desempenho do CPU

Previsão do tempo de execução (T_{EXEC}) de um programa numa máquina - requer 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

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

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

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

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

$$Ganho = \frac{T_{EXEC2}}{T_{EXEC1}} = \frac{5100}{4300} = 1{,}186$$

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: 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 *f* (diminuir Tcc) implica frequentemente um aumento do CPI!

Explicação: Se Tcc diminui, mas o tempo de acesso à memória (Tmem) se mantém, são 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– máxima taxa **possível** 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
- Um programador de uma linguagem de alto nível necessita de métricas mais próximas do problema que se pretende resolver
- **CPE – Ciclos Por Elemento**

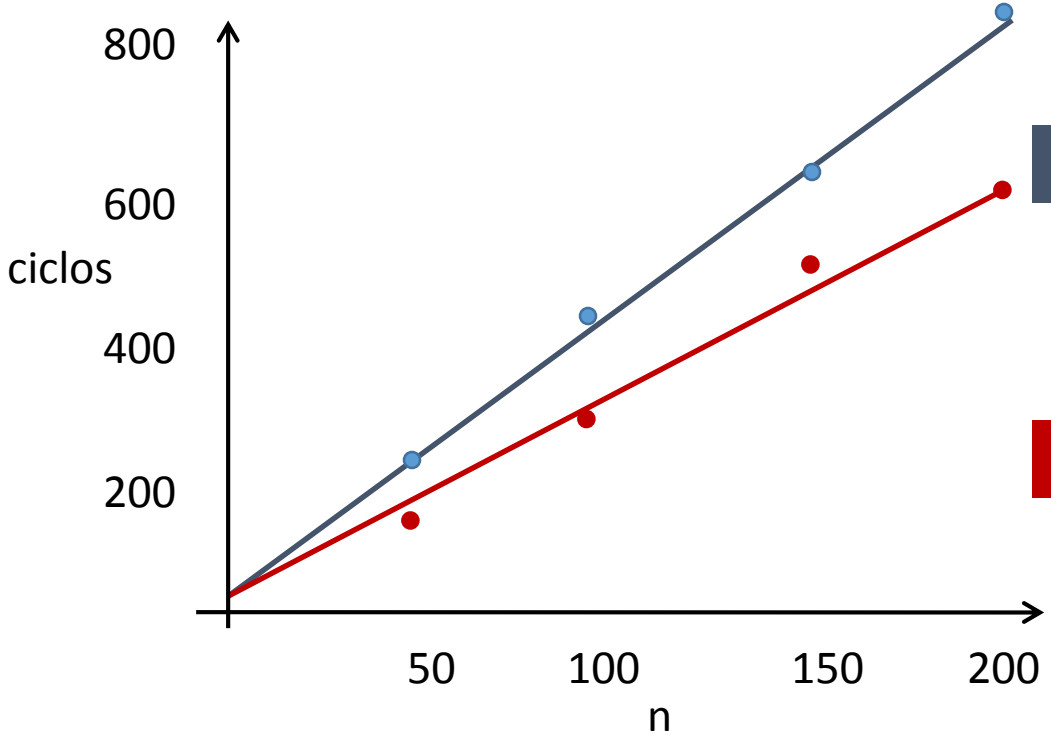
  “número médio de ciclos necessários para processar um elemento de dados”

  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:

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[i] >>= 1;
}
```



Declive = CPE = 4.0

ClockCycles = 20 + **4.0** * n

Declive = CPE = 3.0

ClockCycles = 20 + **3.50*** 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?

```
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;
  }
}
```

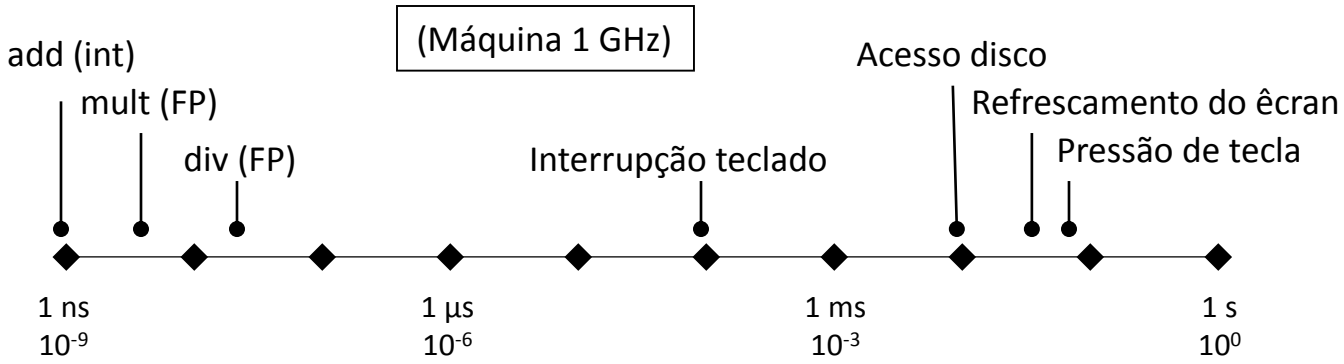
Para n = 1000 -> ciclos = 2520

Qual o CPE?

A utilização de **ciclos por elemento** dá uma indicação do tempo necessário para processar um vector de tamanho *n*.

Metodologia: Medição de Desempenho

```
before = ReadTimer();
<<< Code Segment>>>
after = ReadTimer();
ElapsedTime = after - before;
```



- **Resolução** do relógio: unidade de tempo entre 2 incrementos do contador

  não é possível medir eventos com duração inferior à resolução
- **Precisão** do relógio: diferença entre o valor medido e o tempo efectivamente decorrido

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

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

Contadores de eventos

- Lógica incluída nos processadores (modernos) para contagem de eventos específicos
- Actualizados a cada ciclo de relógio

- Vantagens:
  - Não intrusivos / baixa sobrecarga (disponibilizados pelo hardware)
  - Elevada resolução (relógio do processador)
- Pontos fracos:
  - Específicos de cada processador / não existe um “standard”
  - Não são apropriados para serem usados por utilizador “comum”

Metodologia: Medição de Desempenho

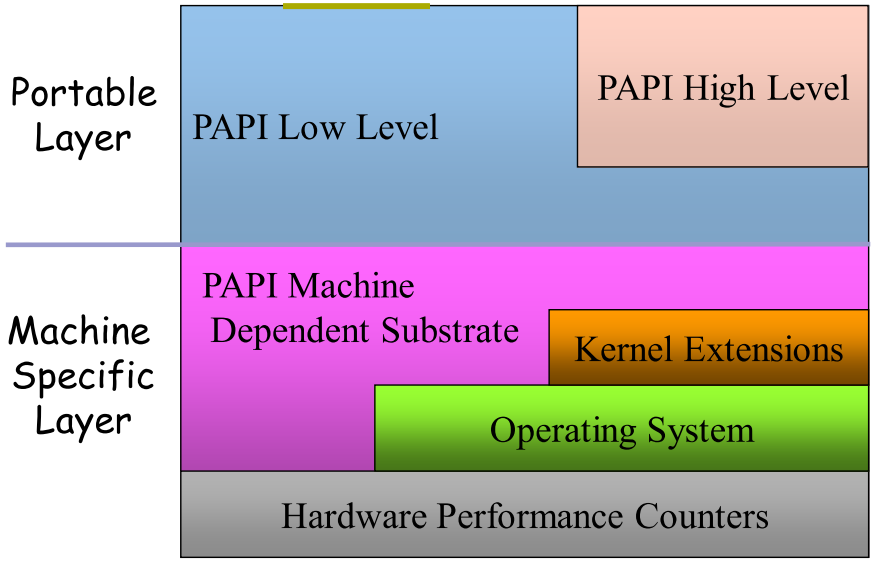
Contadores de desempenho

- Eventos típicos
  - Ciclos de relógio / número de instruções
  - Instruções de vírgula flutuante
  - Instruções sobre valores inteiros (add, sub, etc)
  - *Load/stores*
  - *Cache misses* (L1, L2, etc)

Metodologia: Medição de Desempenho

Performance **A**pplication **P**rogramming **I**nterface

- Interface para acesso aos contadores de desempenho
- Inclui rotinas para contagem de tempo e para obter informação sobre o sistema
- http://icl.cs.utk.edu/papi/
- Arquitectura:



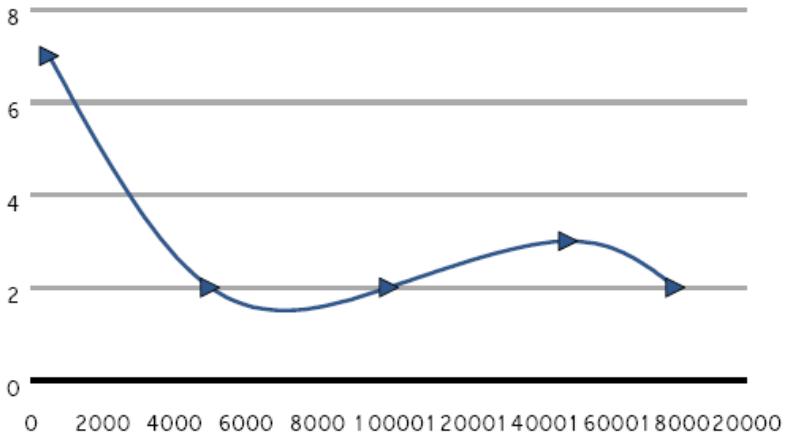
Metodologia: Medição de Desempenho

Apresentação dos resultados

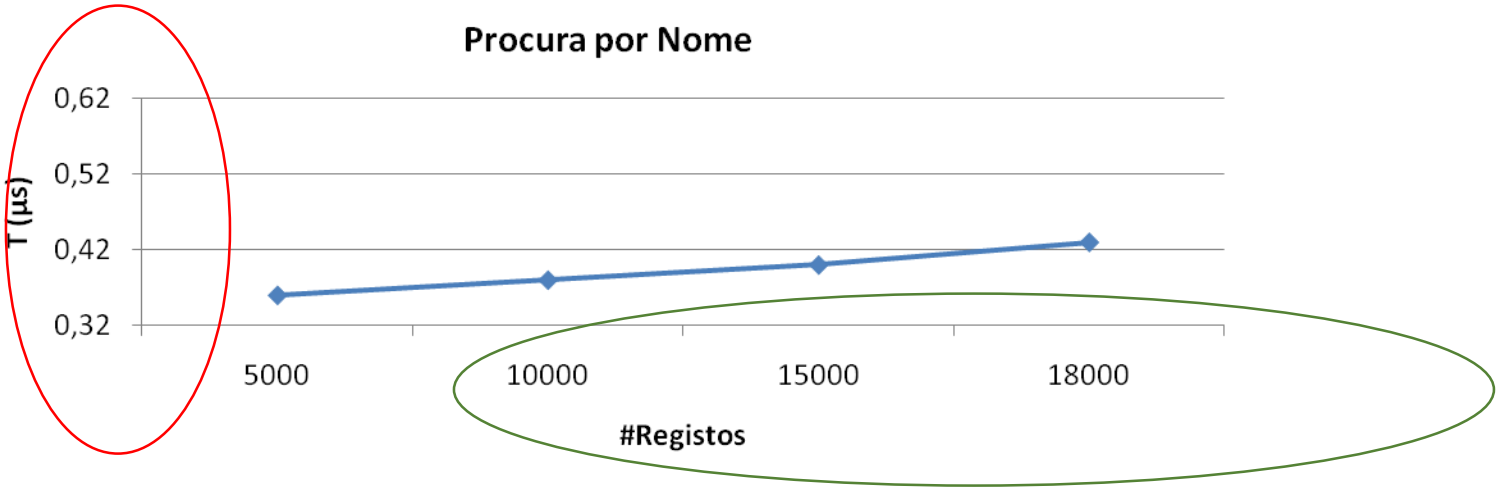
- Apresentar os resultados de forma compacta

Tempos de Execução				
	Nº de Clientes no Ficheiro			
Operações	**5000**	**10000**	**15000**	**18000**
Carregar Dados	10.019 ms	20.881 ms	32.027 ms	40.992 ms
Inserir Cliente	7.100 μs	7.400 μs	8.800 μs	9.500 μs
Procura por Nome	0.360 μs	0.380 μs	0.400 μs	0.430 μs
Procura por Nif	0.020 μs	0.020 μs	0.020 μs	0.020 μs
Percorrer Estrutura	0.092 ms	0.232 ms	0.470 ms	0.673 ms

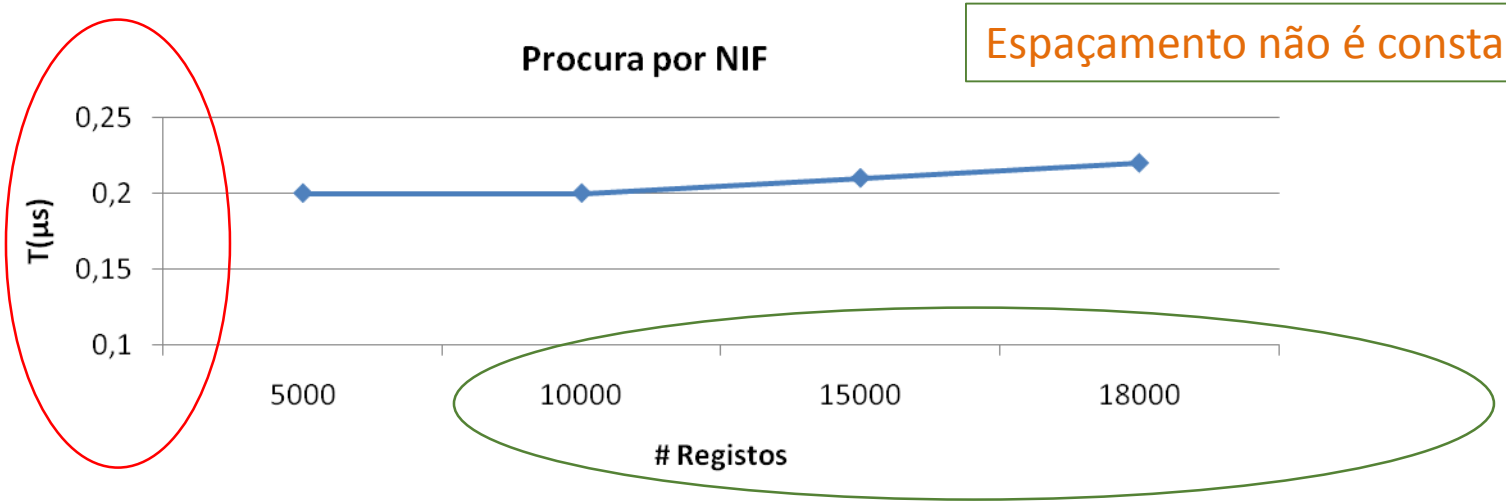
- Colocar legendas nas tabelas e gráficos



Metodologia: Medição de Desempenho

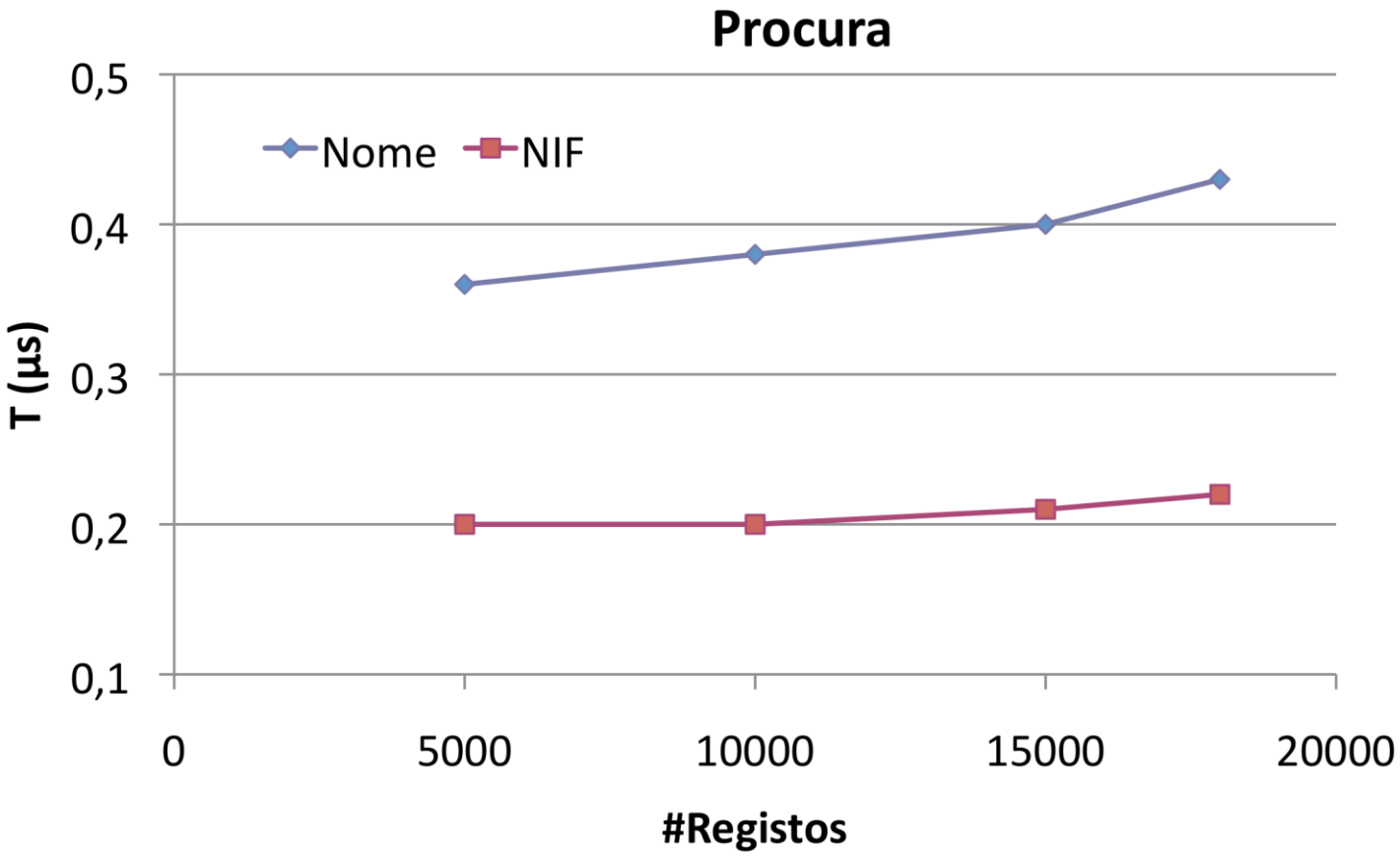


Espaçamento não é constante!



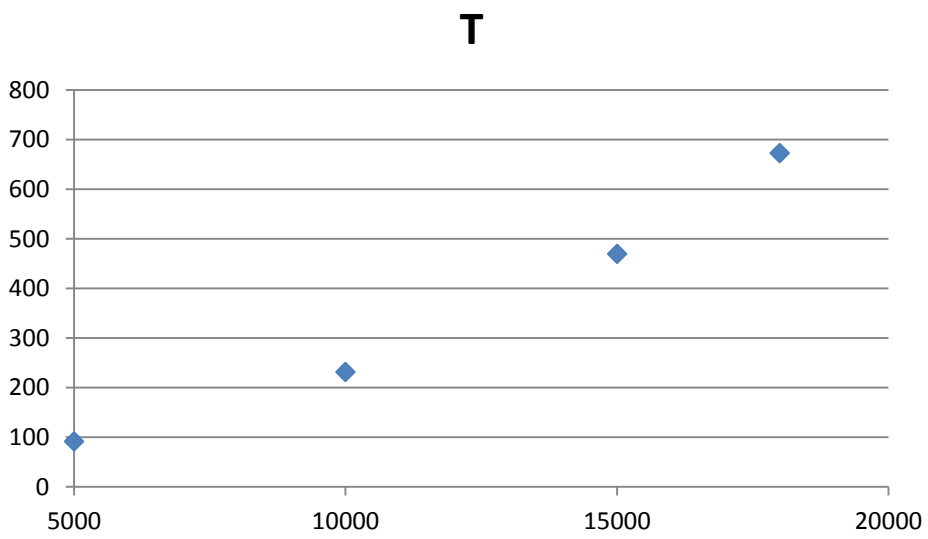
Escalas diferentes em comparações directas induzem em erro!

Metodologia: Medição de Desempenho



Complexidade – curve fitting

Tempo de Execução (µs)				
	Nº de registos			
Operação	**5000**	**10000**	**15000**	**18000**
Percorrer Estrutura	92.00	232.00	470.00	673.00



O processo de curve fitting permite determinar a equação da curva que melhor descreve os dados.

Complexidade – curve fitting

A opção “Trendline” do MS EXCEL determina a equação da curva dado um modelo: linear, polinomial, logaítmico, exponencial, etc.
O parâmetro R2 descreve a qualidade do fitting.
Quanto mais perto de 1 melhor.