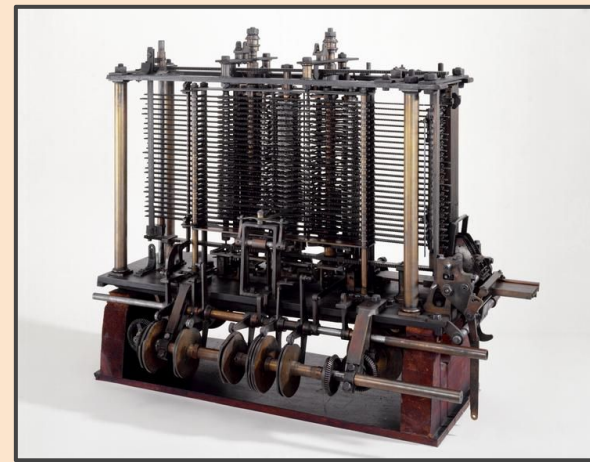

Conceitos de Hardwares

— Arquitetura de
Sistemas Computacionais —

Um pouco de história

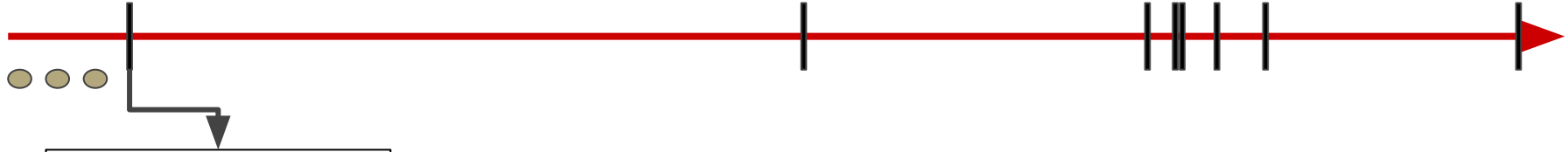


História (Resumida)



História - Dispositivos mecânicos

Atual



~1600 EC (Era Comum)

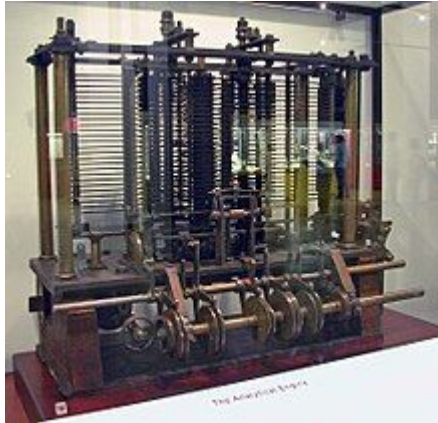
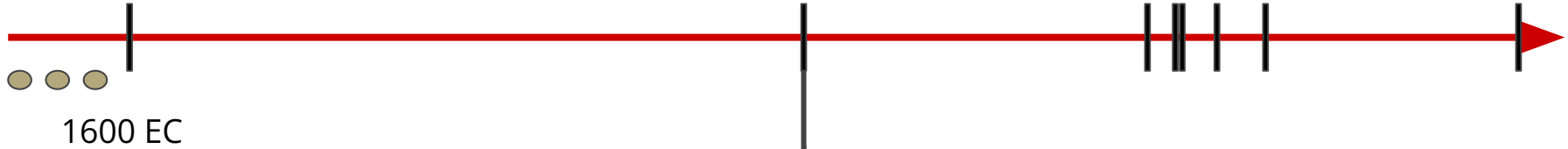
Wilhelm Schickard



Primeira máquina de calcular com
as 4 operações básicas

História - Dispositivos mecânicos

Atual



1837 EC

Charles
Babbage

Máquina Analítica

Primeira máquina de
propósito geral

Lia instruções de cartões
perfurados e as executava

Atual

Charles Babbage

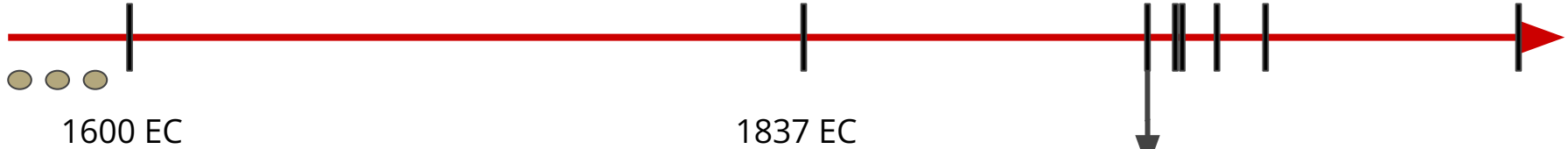
DIAGRAM BELONGING TO NOTE D

Number of Operations Nature of Operations		Variables for Data										Working Variables									
		$1V_0$	$1V_1$	$1V_2$	$1V_3$	$1V_4$	$1V_5$	$2V_0$	$2V_1$	$2V_2$	$2V_3$	$2V_4$	$2V_5$	$3V_0$	$3V_1$	$3V_2$	$3V_3$				
		+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+				
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
		m	n	d	m'	n'	d'														
1	x																				
2	x	m	n		m'	n'															
3	x																				
4	x		d																		
5	x	0																			
6	x		0				0														
7	x			0	0																
8	+							0	0												
9	+							0	0												
10	+																				
11	+																				

$$\frac{d'm - dm'}{mn - m'n} = \gamma$$

História - Dispositivos eletromecânicos

Atual



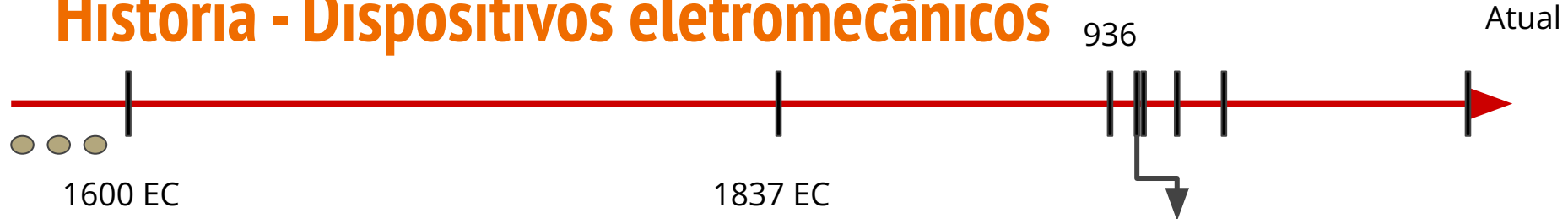
1936 EC

Konrad Zuse

Z1

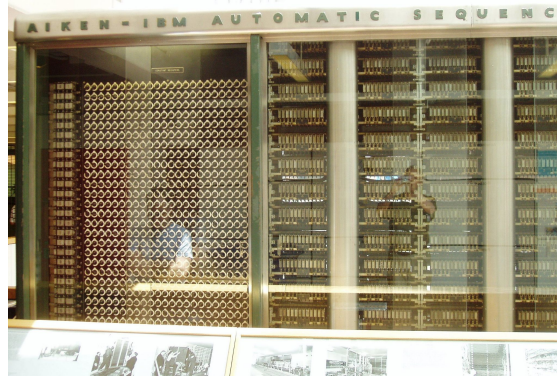


História - Dispositivos eletromecânicos



Harvard
Mark 1

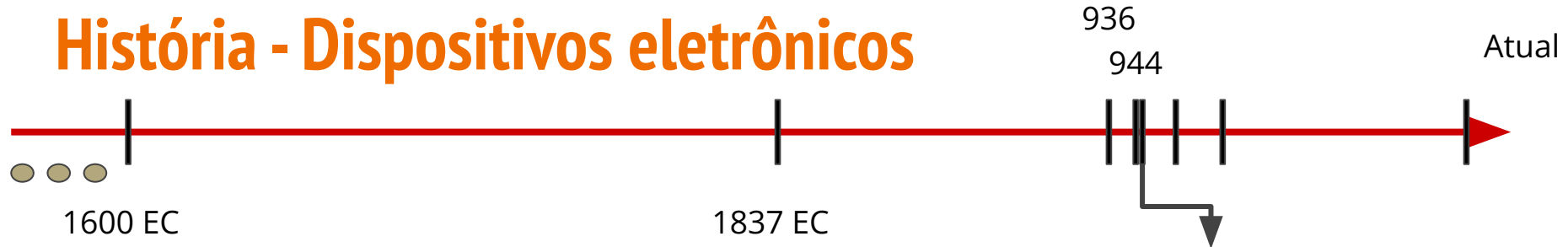
Capaz de multiplicar dois
números de dez dígitos em
três segundos!!!



1944 EC

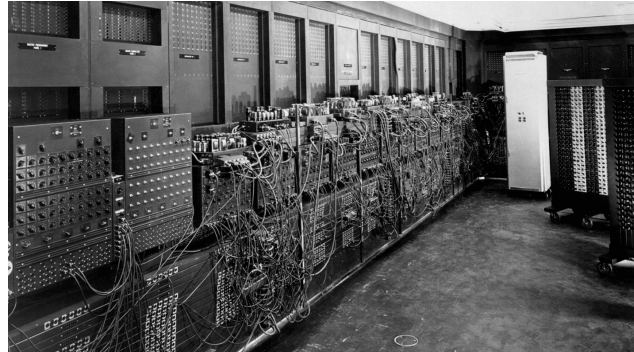
Marinha dos
EUA e Harvard

História - Dispositivos eletrônicos



ENIAC

500 multiplicações de 10
dígitos por segundo...



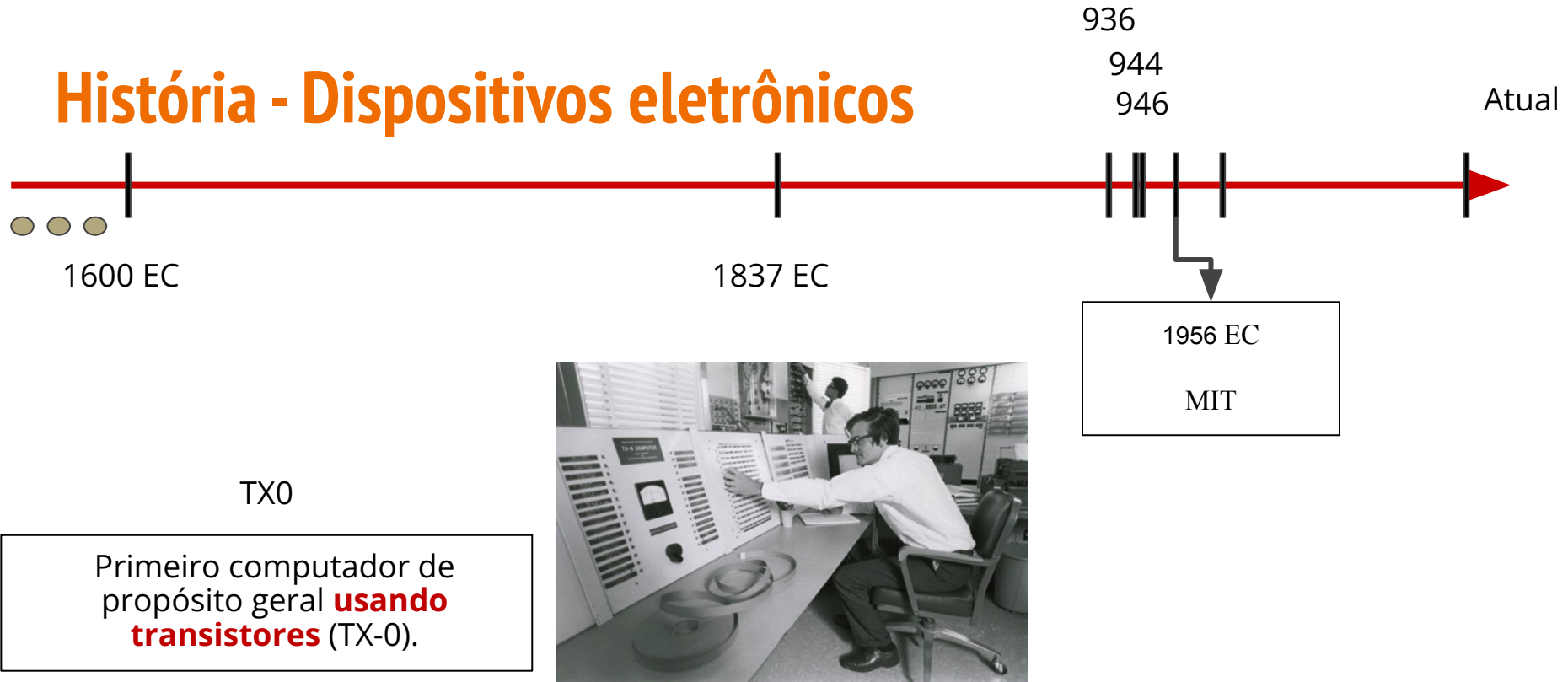
1946 EC

Exército dos EUA

Universidade da
Pennsylvania

von Neumann

História - Dispositivos eletrônicos

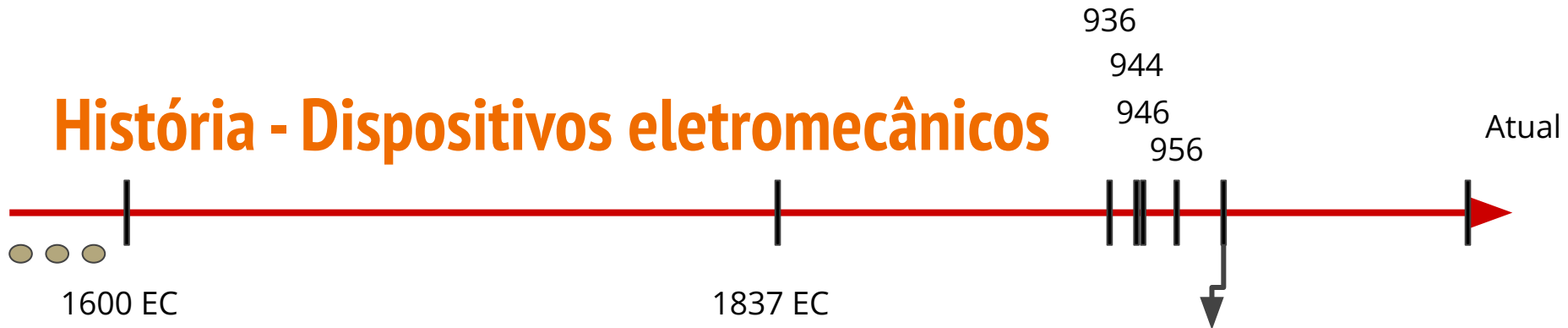


TX0

Primeiro computador de propósito geral **usando transistores** (TX-0).



História - Dispositivos eletromecânicos

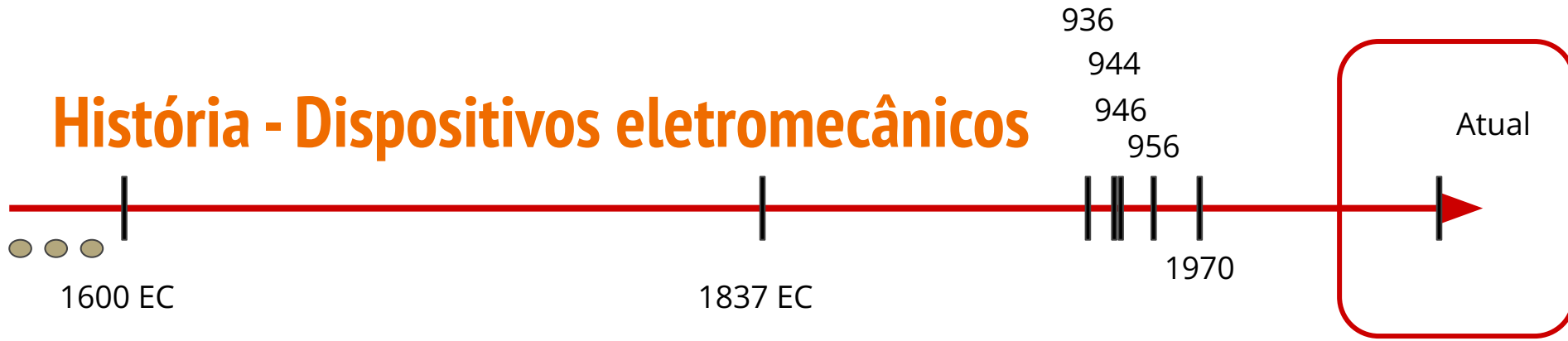


Intel 4004

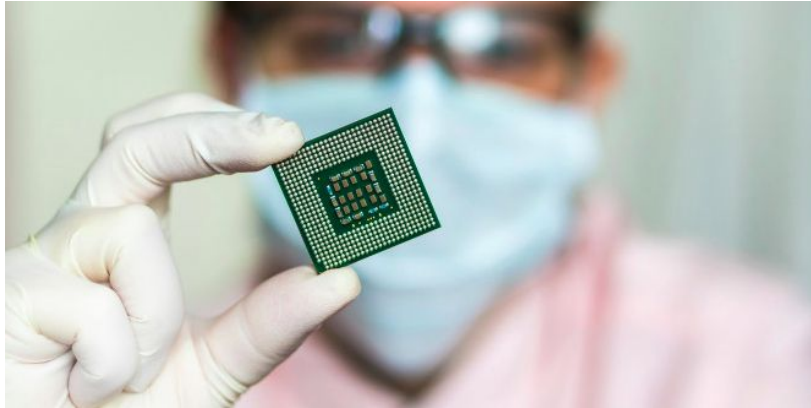
Introduz o primeiro
microprocessador com **2,250**
transistores



História - Dispositivos eletromecânicos



Atualmente – Intel
vende processadores com
+8 bilhões de transistores

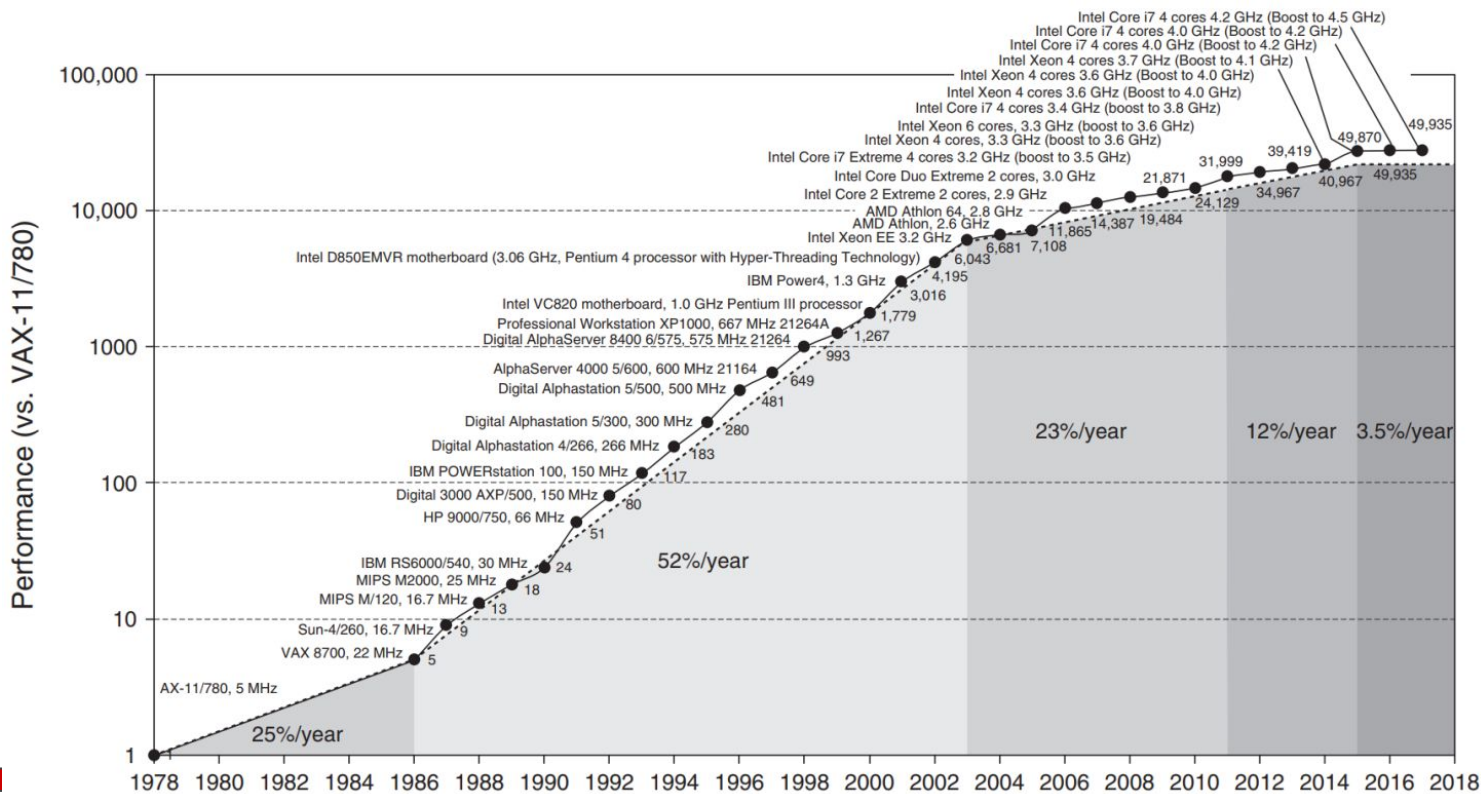


Evolução

1970 EC

Intel

**2,250
transistores**

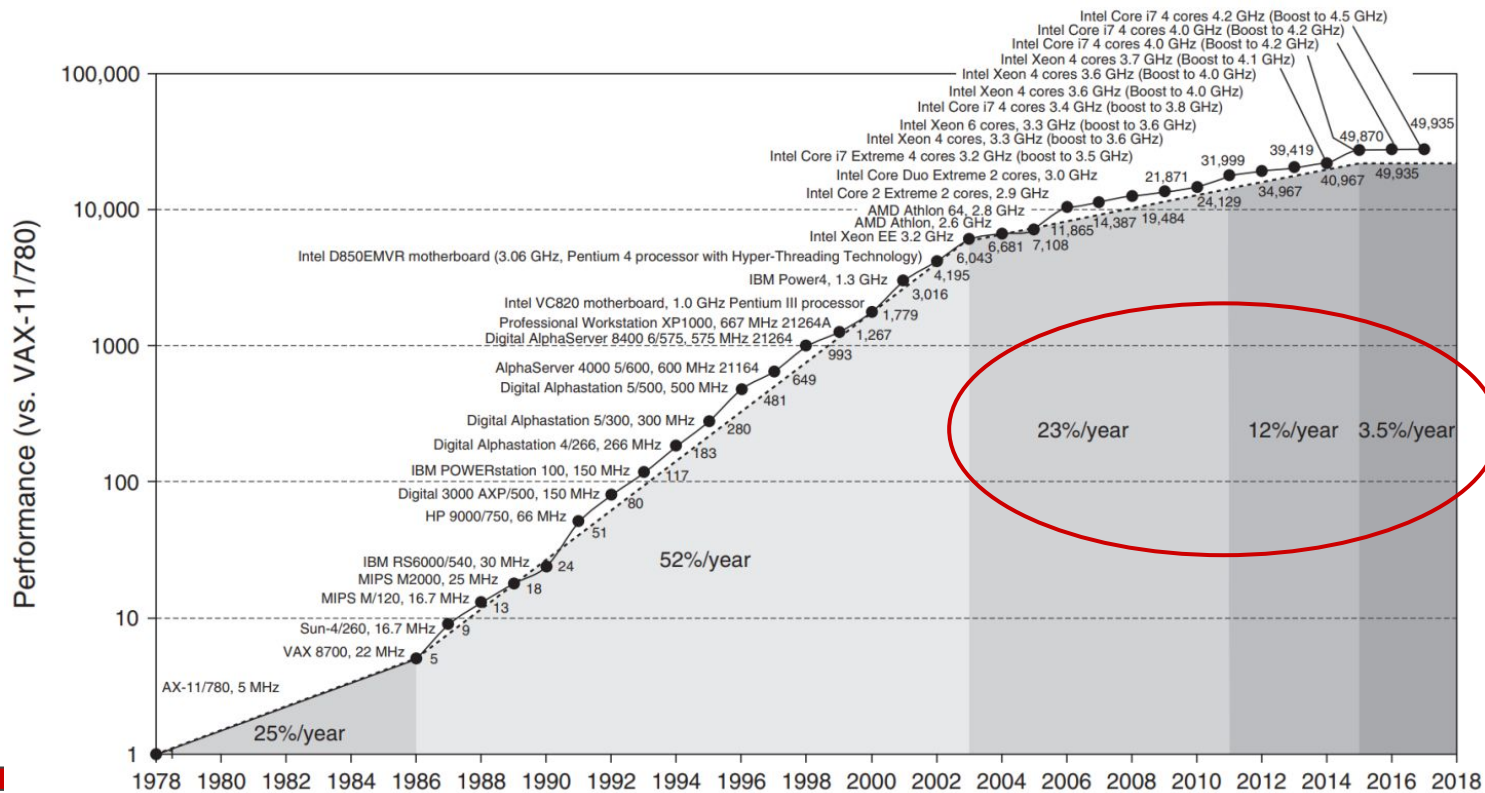


Evolução

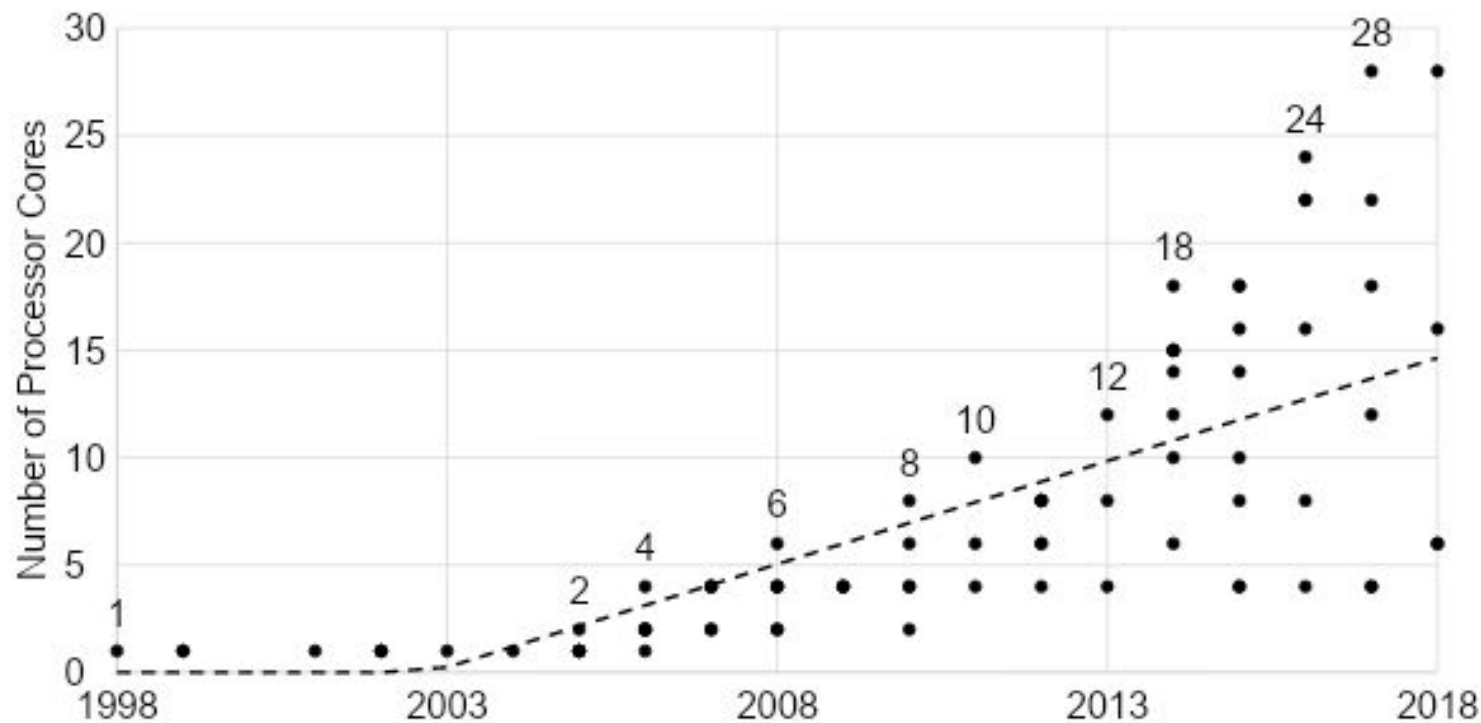
1970 EC

Intel

**2,250
transistores**



Evolução multicore



Revisão rápida

Álgebra booleana

Na álgebra booleana, as variáveis Booleanas só podem assumir um número finito de valores (0 ou 1).

Função booleana

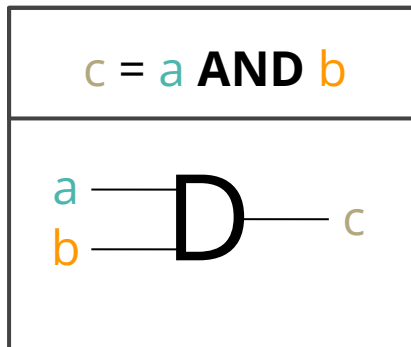
- Para um conjunto de entradas, produz uma única saída
- Operadores lógicos: AND, OR, NOT, XOR, NAND, NOR

Portas lógicas

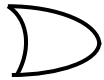




Portas ou **circuitos lógicos** são dispositivos que operam e trabalham com um ou mais sinais lógicos de entrada para produzir uma saída, dependente da função implementada no circuito.

Permitem a implementação de funções da álgebra booleana.

Operadores booleanos e portas lógicas



Outras portas

OR	
NOT	
XOR	
NAND	
NOR	

Exercício

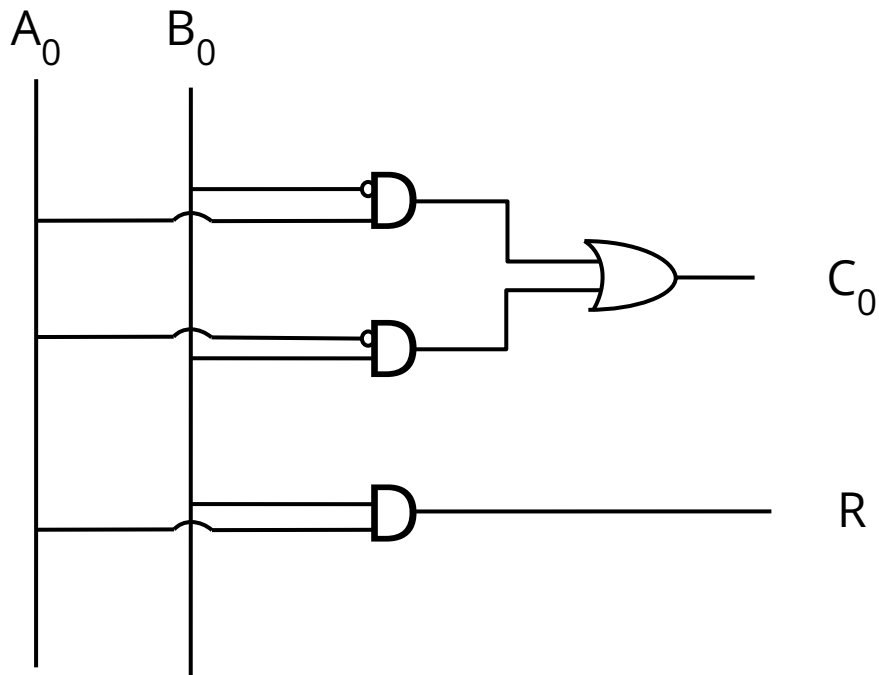
Construa o circuito de um somador de 1 bit.

Entrada: dois números de 1 bit $\rightarrow A_0$ e B_0

Saída: bit do resultado $\rightarrow C_0$ e um bit de resto R.

$$\begin{array}{r} + \quad A_0 \\ \quad B_0 \\ \hline R \quad C_0 \end{array}$$

Exercício - Resposta

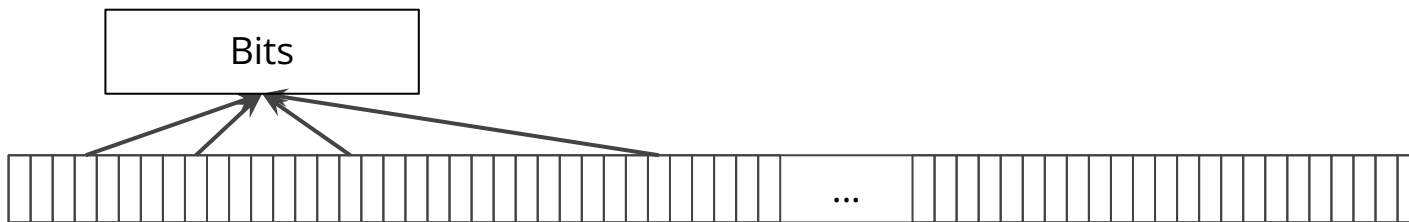


Componentes complexos

Registrador

Memória mais rápida e cara de um computador

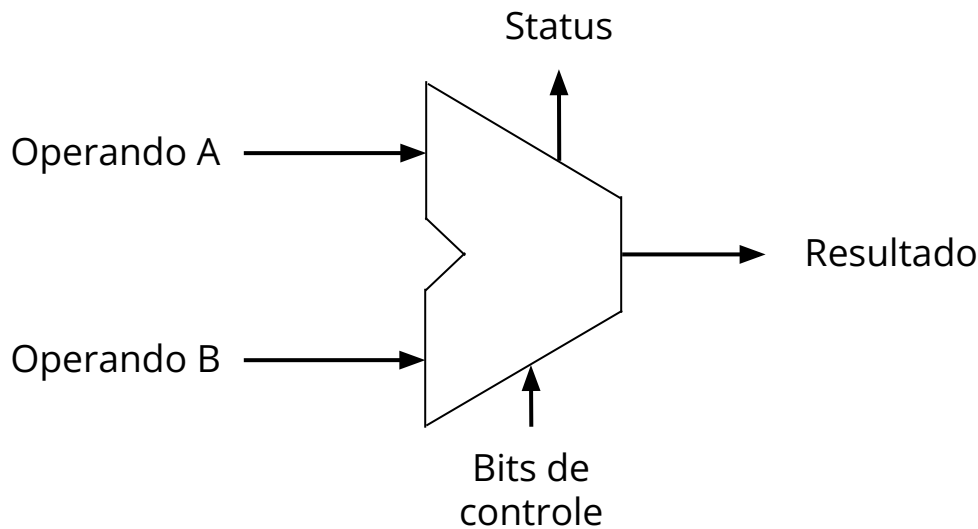
Armazena dados **dentro** da CPU



Unidade Lógica e Aritmética (ULA)

Componente capaz de realizar operações lógicas (AND, OR, NOT, ...) e aritméticas (+, -, *, /)

Arithmetic Logic Unit - ALU



MUX e DEMUX

Multiplexador (MUX)

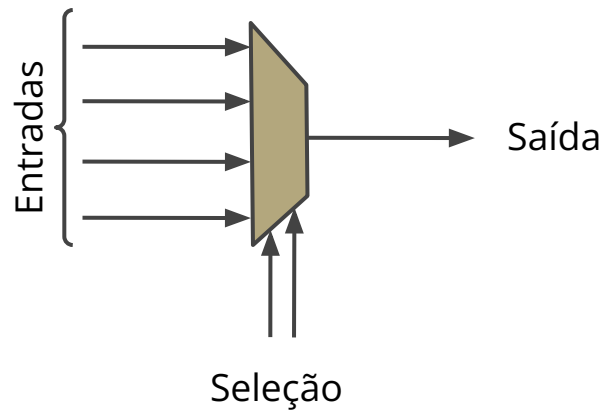
- Permite selecionar uma, dentre diversas entradas, para uma saída

Demultiplexador (DEMUX)

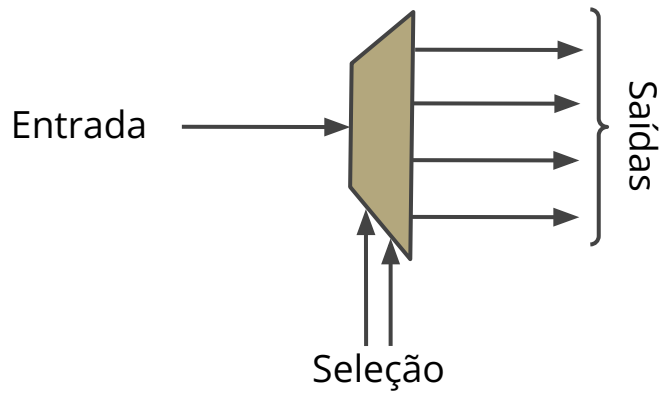
- Permite enviar uma entrada para uma de múltiplas saídas

MUX e DEMUX

Multiplexador (MUX)



Demultiplexador (DEMUX)



Sistemas modernos

Componentes importantes



Processador (CPU)

Componentes importantes

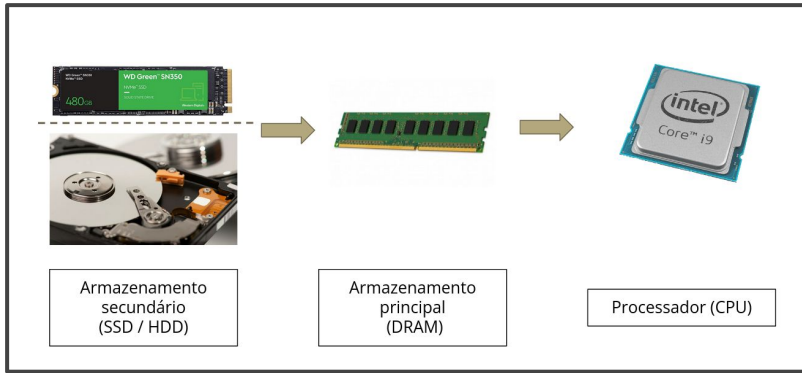


Armazenamento
secundário
(SSD / HDD)



Processador (CPU)

Componentes importantes



Comunicação entre os componentes

- Interconexões da placa mãe

Componentes importantes



Comunicação entre os componentes

- Interconexões da placa mãe

Dispositivos
periféricos

Como o hardware executa um programa?



Camadas de abstração - Código C

Linguagem de alto nível

```
int main() {  
    int a = 10;  
    a += 3;  
    return a;  
}
```

Camadas de abstração - Geração do assembly

Linguagem de alto nível

```
int main() {  
    int a = 10;  
    a += 3;  
    return a;  
}
```

Compilador



Linguagem de montagem

```
movl    $10, -4(%rbp)  
addl    $3, -4(%rbp)  
movl    -4(%rbp), %eax  
popq    %rbp  
ret
```

Camadas de abstração - Montagem

Linguagem de montagem

movl \$10, -4(%rbp)

addl \$3, -4(%rbp)

movl -4(%rbp), %eax

popq %rbp

ret

Montador
(Assembler)



Camadas de abstração - Geração do código de máquina

Linguagem de montagem

```
movl    $10, -4(%rbp)
addl    $3, -4(%rbp)
movl    -4(%rbp), %eax
popq    %rbp
ret
```

Montador
(Assembler)



Código de Máquina

```
11000111 01000101
11111100 00001010
00000000 00000000
00000000 10000011
01000101 11111100
00000011 10001011
01000101 11111100
01011101 11000011
```

Camadas de abstração - Execução

Código de Máquina

```
11000111 01000101  
11111100 00001010  
00000000 00000000  
00000000 10000011  
01000101 11111100  
00000011 10001011  
01000101 11111100  
01011101 11000011
```

Executa



Camadas de abstração - Internamente



Descreve o
caminho dos
dados



RTL - Register Transfer Level

$\text{REG}[0] \leftarrow \text{INST}[15..30]$

$\text{PC} \leftarrow \text{PC} + 4;$

.

.

.

Utilidade das camadas

Qual o objetivo de termos várias camadas de abstração?



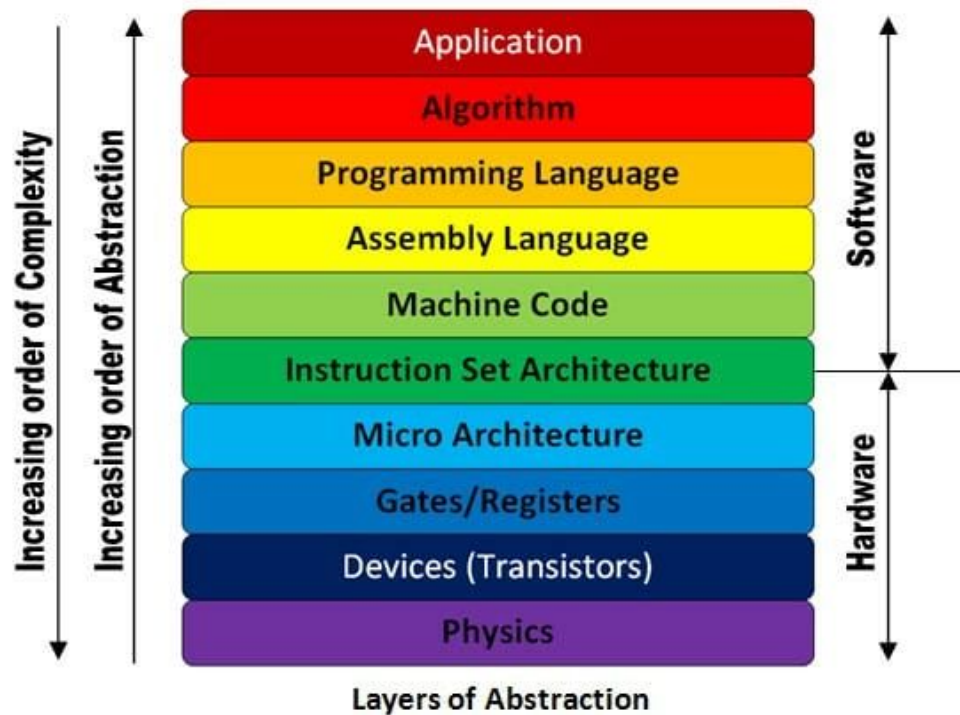
Utilidade das camadas

Qual o objetivo de termos várias camadas de abstração?

- Simplificar o desenvolvimento, dispensando detalhes desnecessários
- Desenvolver aplicações mais complexas, mais rapidamente



Camadas de abstração - Resumo



Hardware e Software

Hardware

- **Objetos tangíveis:** Circuitos integrados, circuito impresso, cabos, fontes, memórias

Software

- **Programas** → Conjuntos de instruções e dados usados para operar um computador e executar tarefas específicas
- Conjunto de instruções, não o meio físico no qual são armazenados

Equivalência entre Hardware e Software

1. Qualquer operação implementada em Software pode ser implementada em Hardware
 2. Qualquer operação implementada em Hardware pode ser implementada em Software
- Considerações: Custo, velocidade, frequência de atualizações, etc...

Arquitetura e Organização de Computadores

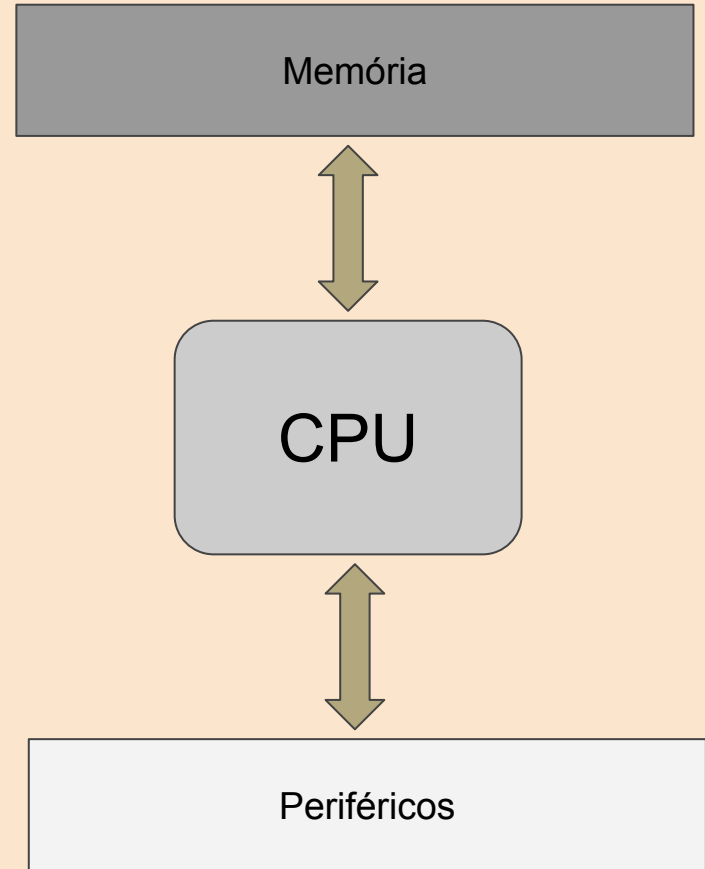
Arquitetura de Computadores

- Estudo sobre como projetar as partes do sistema visíveis ao programador
 - Características da ISA: tipos de dados, operações
 - Quantidade de memória disponível
 - Número de núcleos
- Determinar as necessidades do usuário e atendê-las da maneira mais eficaz dentro das restrições econômicas e tecnológicas
- **Responde à pergunta:** O que precisa ser feito?

Organização de Computadores

- Estudo dos aspectos de implementação da arquitetura, invisíveis ao programador
 - Sequência de portas lógicas utilizada
 - Algoritmos para operações aritméticas
 - Tecnologia de armazenamento dos bits na memória
- **Responde à pergunta:** Como será implementado?

Planejando um processador



Execução em hardware

```
int main() {  
    int a = 10, b = 3;  
    a += b;  
    return a;  
}
```


Execução em hardware

O que precisamos
para executar essa
instrução?

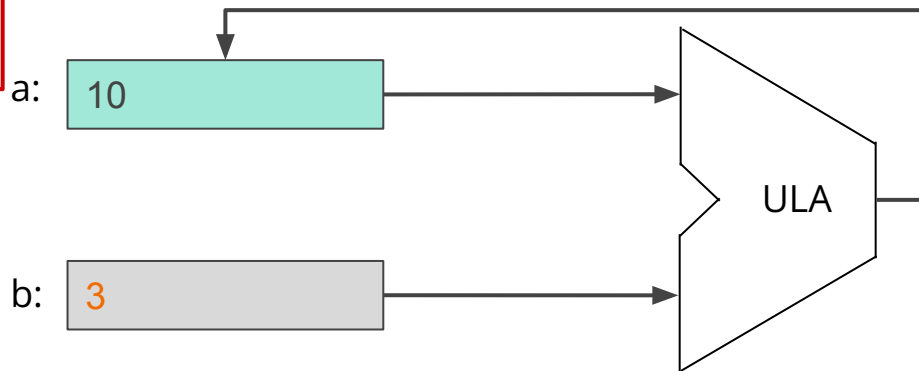
```
int main() {  
    int a = 10, b = 3;  
    a += b;  
    return a;  
}
```

The diagram illustrates the hardware execution of a specific instruction. A red box highlights the instruction `a += b;` within the `main` function. A red arrow points from this box to a separate box on the left that asks, "O que precisamos para executar essa instrução?" (What do we need to execute this instruction?).

Circuito inicial

O que precisamos para executar essa instrução?

`a += b;`

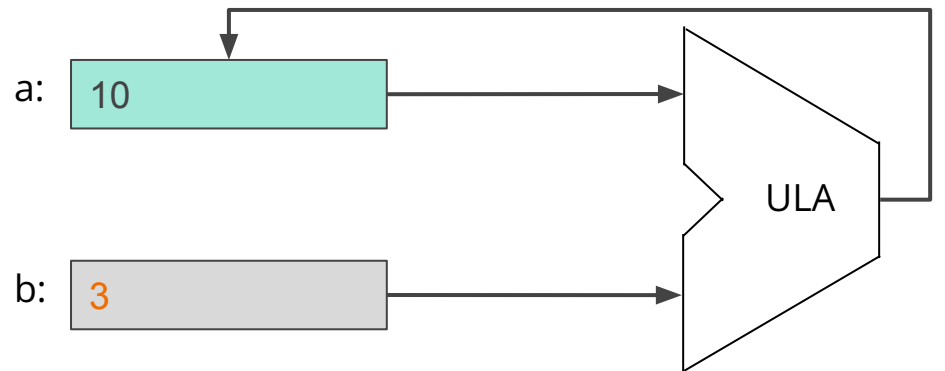


ULA - Unidade
Lógica e Aritmética

Circuito inicial - Generalizando

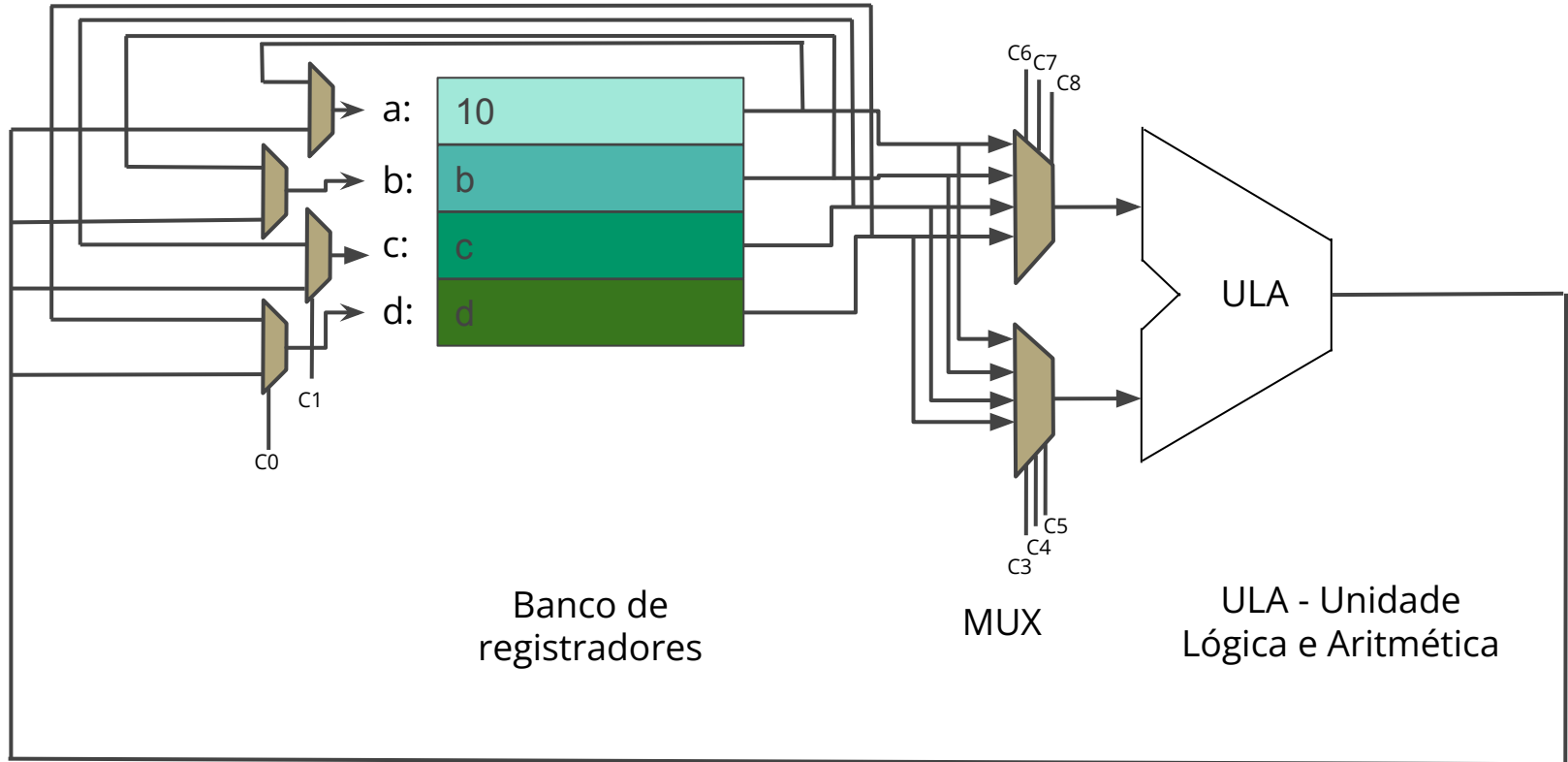
Como alterar o circuito para executar essas duas instruções?

- $a = b + c;$
- $b = a + d;$

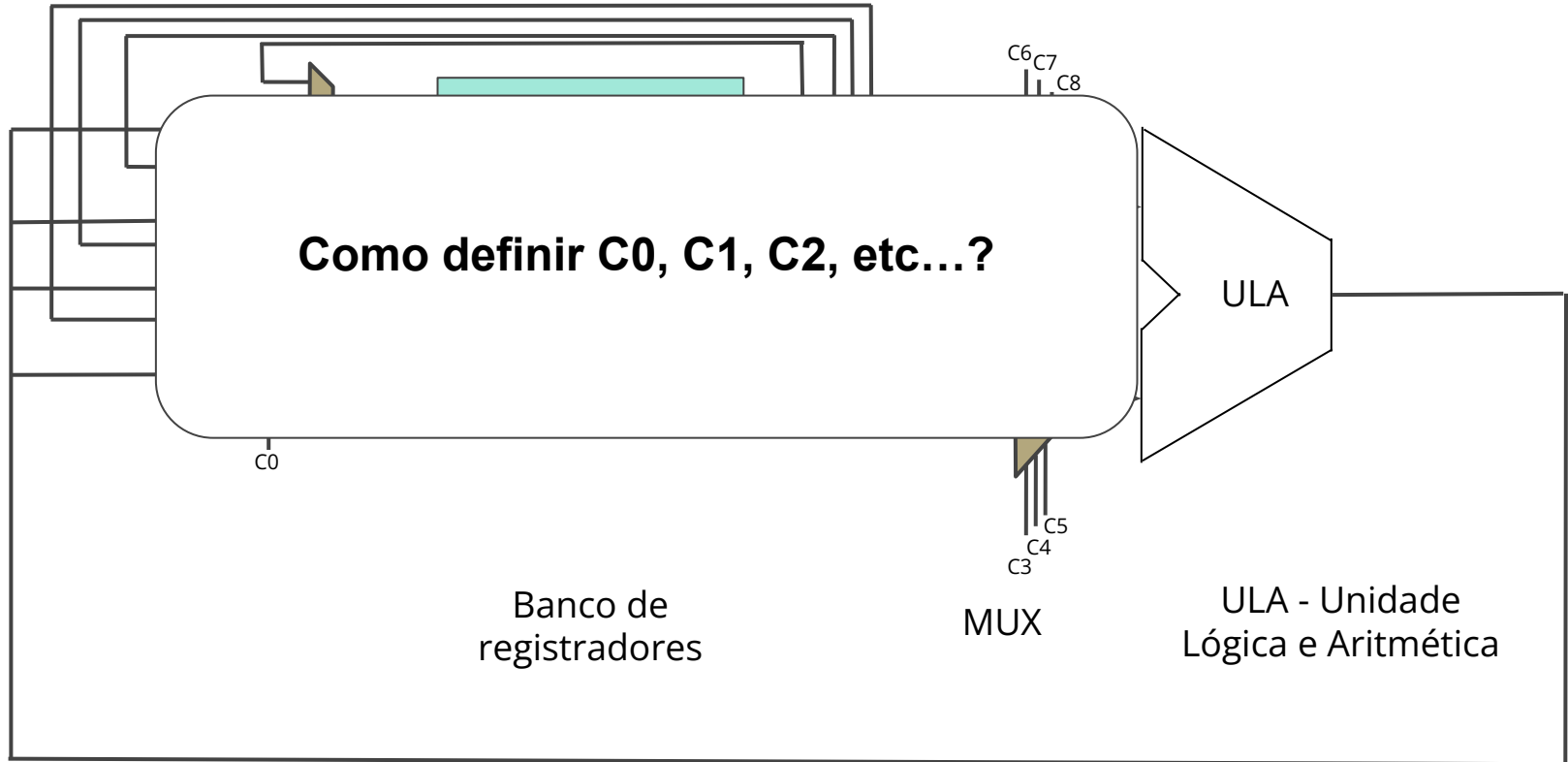


ULA - Unidade
Lógica e Aritmética

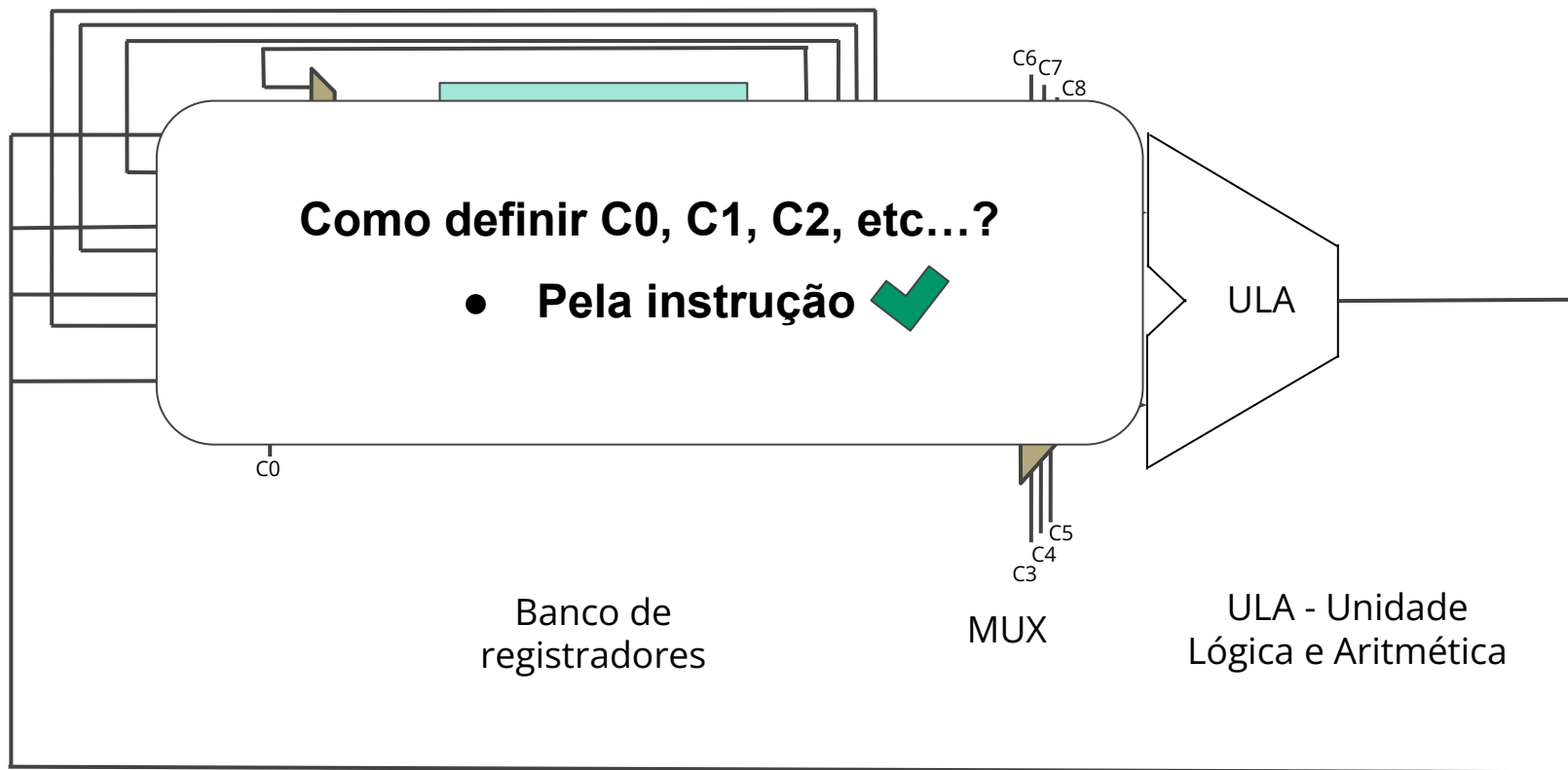
Circuito inicial - Banco de registradores



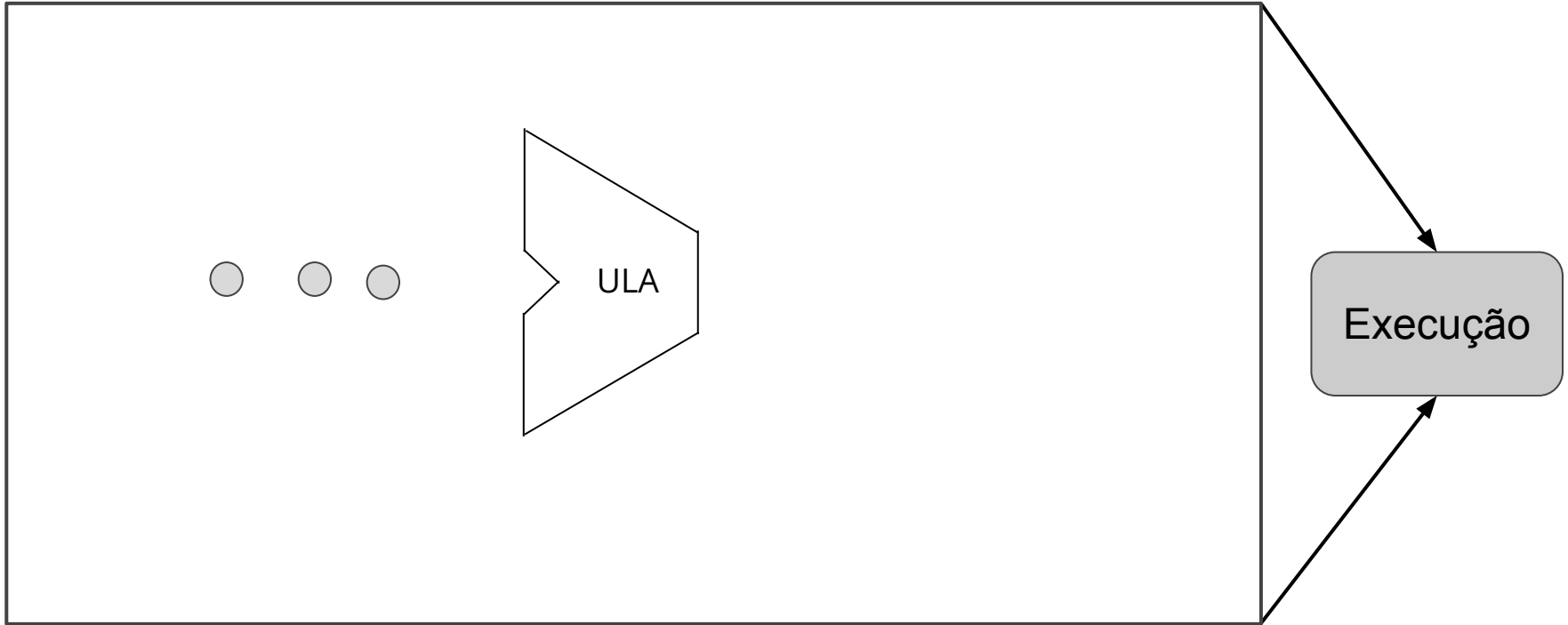
Circuito inicial - Banco de registradores



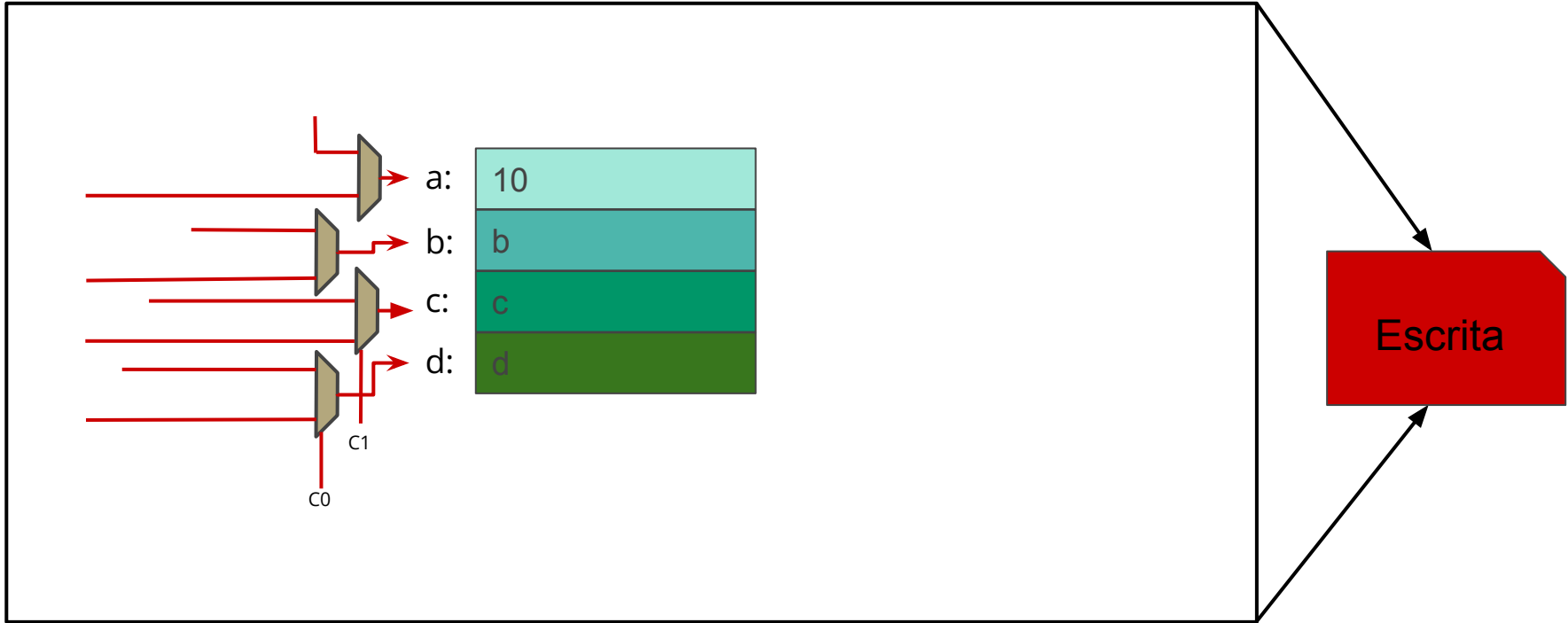
Circuito inicial - Banco de registradores



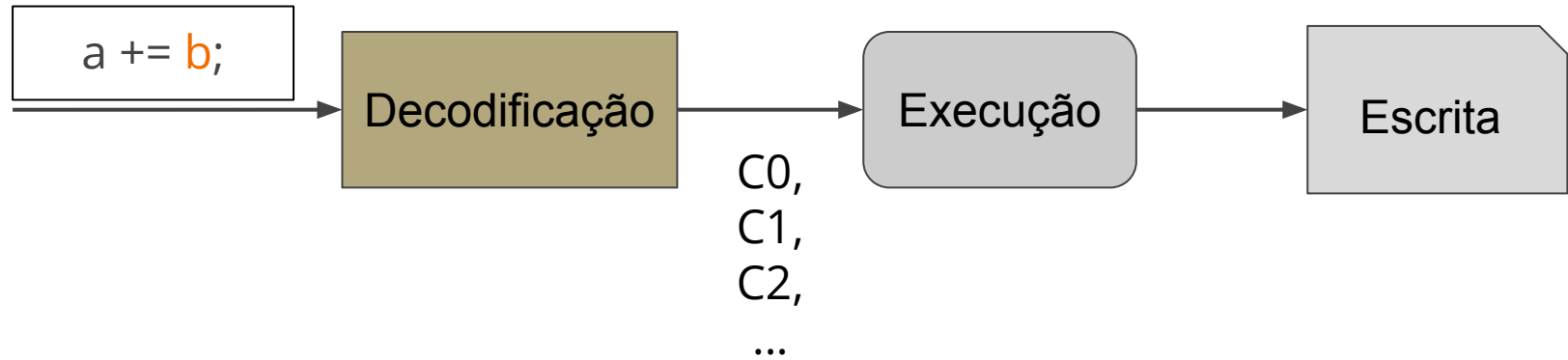
Circuito inicial - Estágio de execução



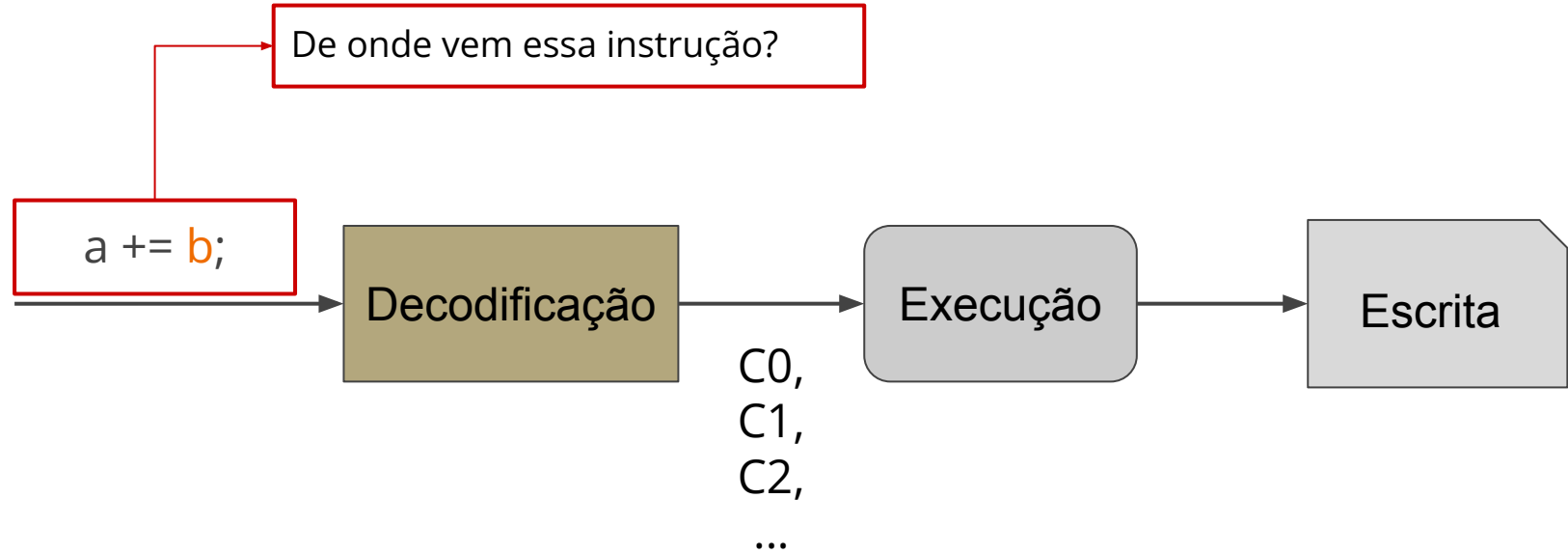
Circuito inicial - Estágio de execução



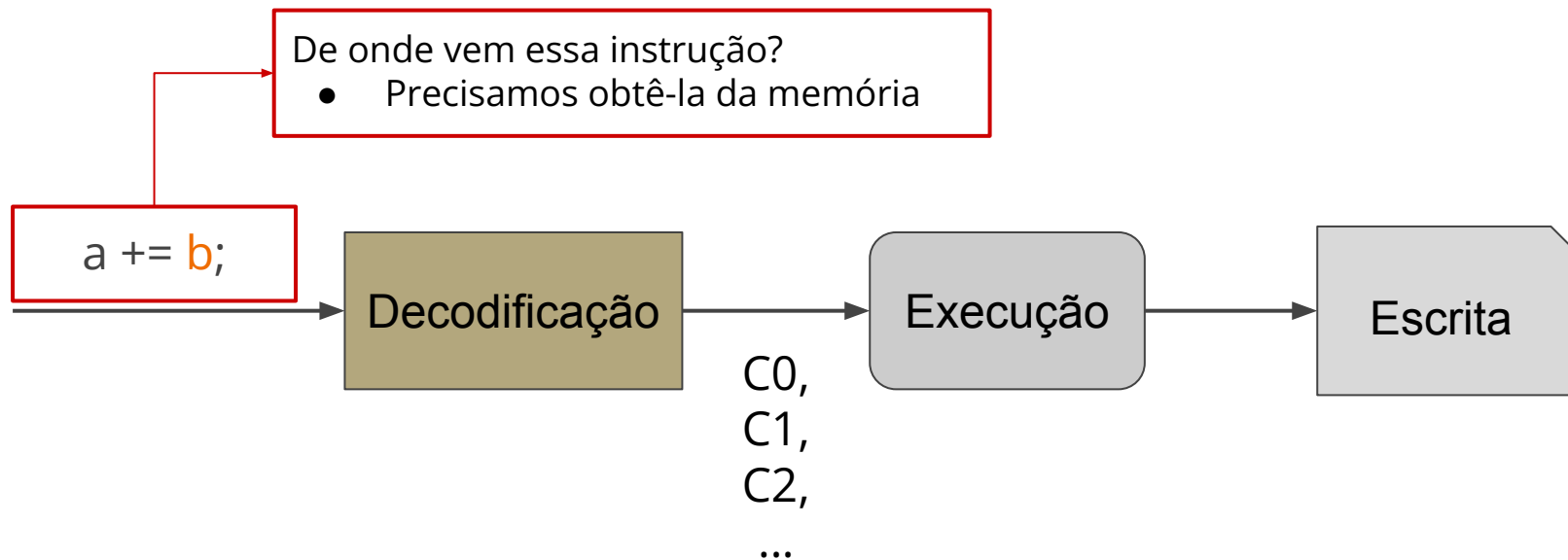
Circuito inicial - Estágio de decodificação



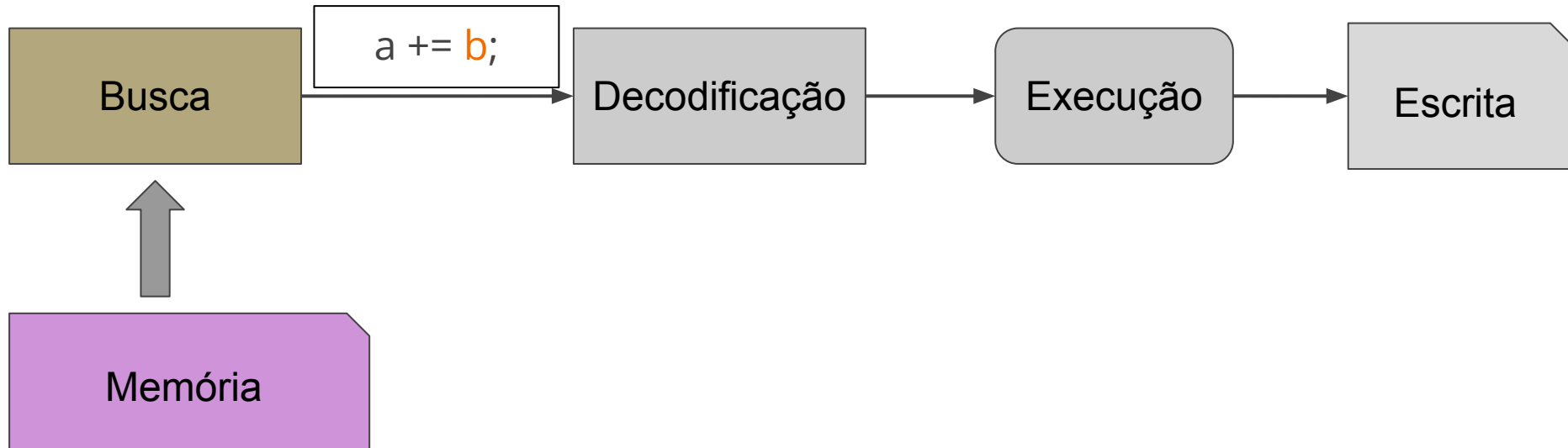
Circuito inicial - Busca de instruções



Circuito inicial - Busca de instruções



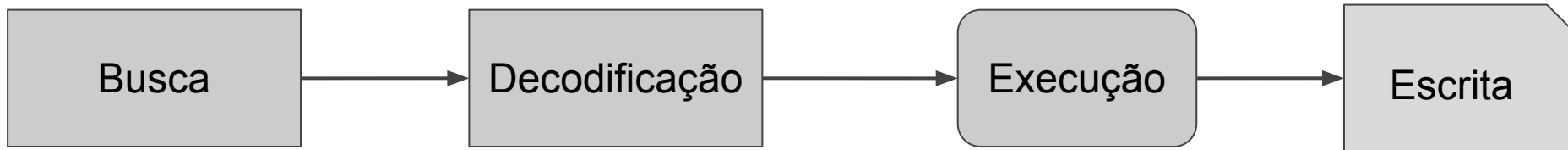
Circuito inicial - Busca de instruções



Circuito inicial - Busca de dados

Como executar as seguintes instruções?

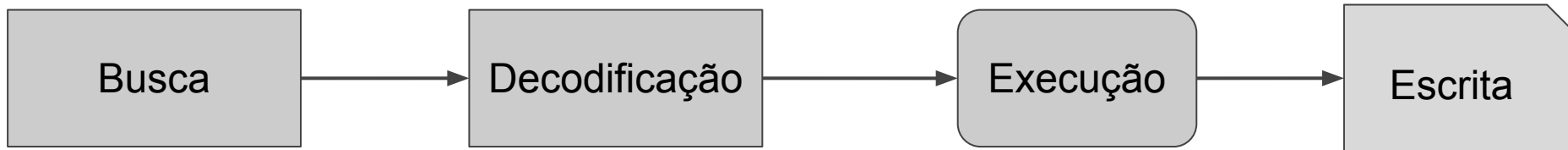
- `a = vetor [3];`
- `vetor [3] = a;`



Circuito inicial - Busca de dados

Como executar as seguintes instruções?

- `a = vetor [3];`
- `vetor [3] = a;`

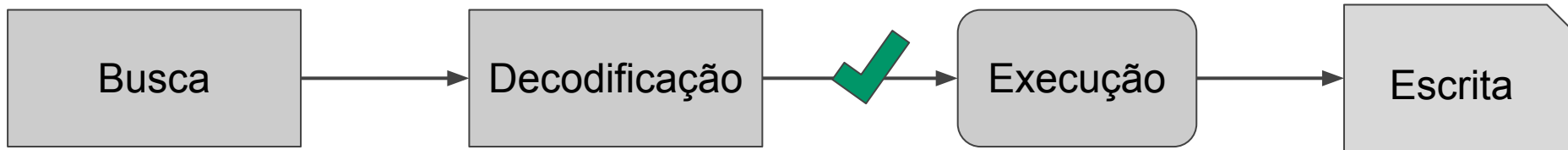


Circuito inicial - Busca de dados

Como executar as seguintes instruções?

- `a = vetor [3];`

Após decodificar a instrução, sabemos o endereço do dado a carregar!

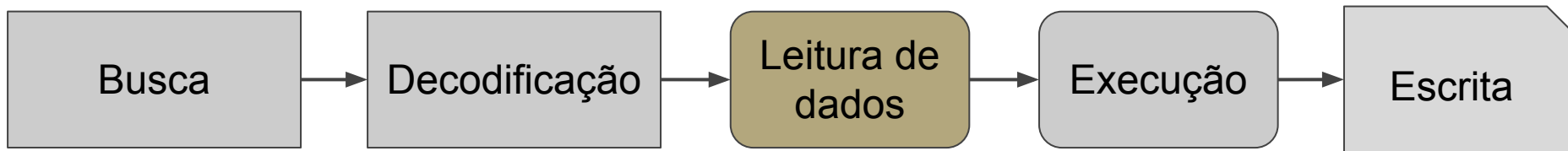


Circuito inicial - Busca de dados

Como executar as seguintes instruções?

- `a = vetor [3];`

Após decodificar a instrução, sabemos o endereço do dado a carregar!

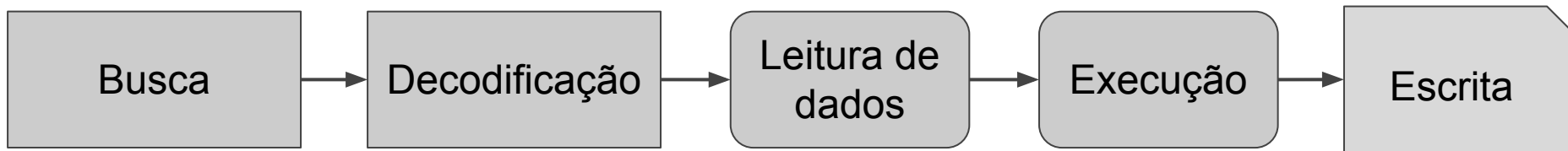


Leitura da Memória
Leitura do Banco de Registradores

Circuito inicial - Busca de dados

Como executar as seguintes instruções?

- `a = vetor [3];`
- `vetor [3] = a;`

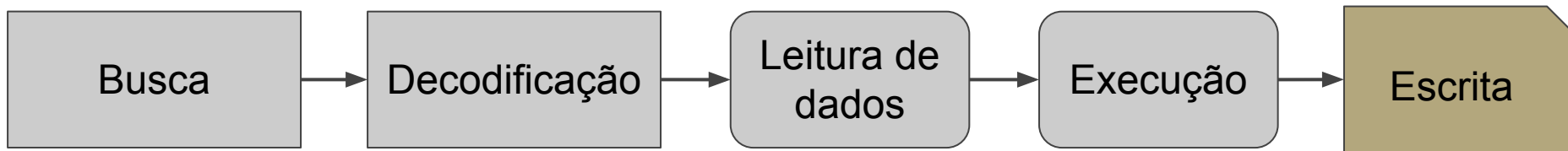


Circuito inicial - Busca de dados

Como executar as seguintes instruções?

- `vetor [3] = a;`

Podemos utilizar o estágio existente de escrita!

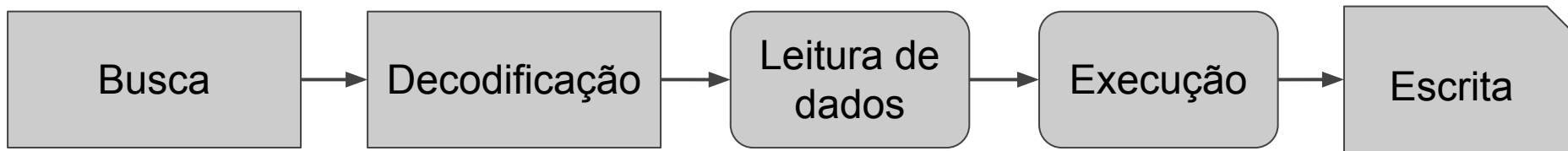


Processador Monociclo

Circuito inicial - Monociclo

Revisando:

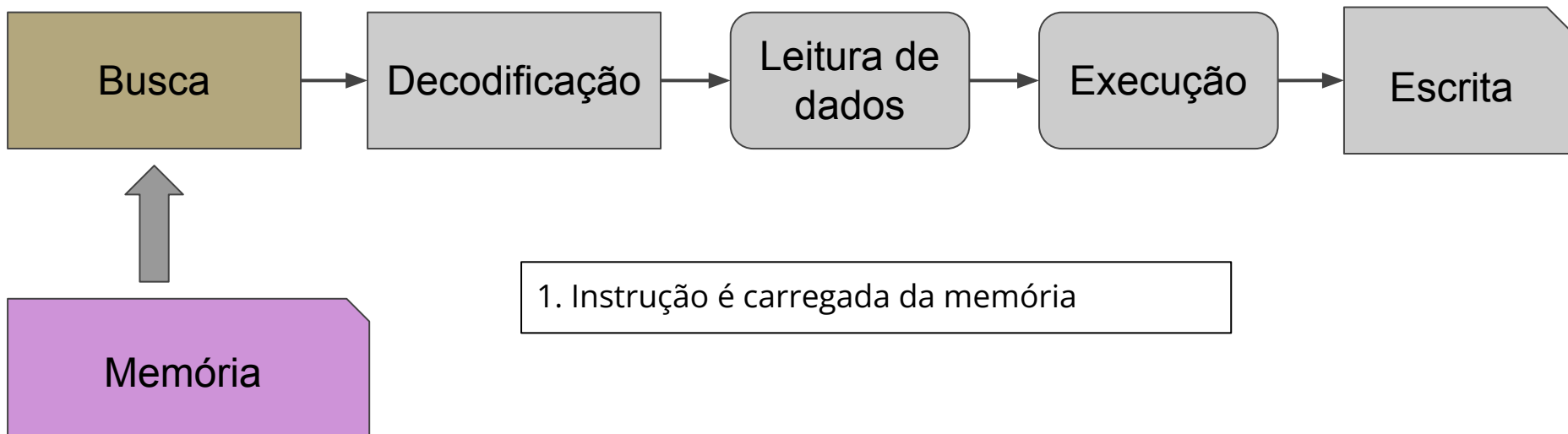
- $a = b + \text{vetor}[2];$



Circuito inicial - Busca de instrução

Revisando:

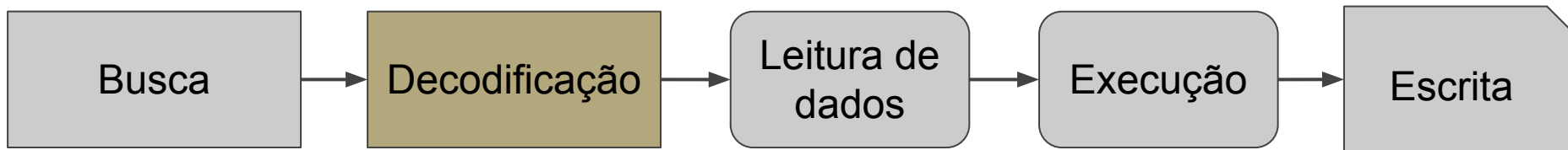
- `a = b + vetor[2];`



Circuito inicial - Decodificação da instrução

Revisando:

- $a = b + \text{vetor}[2];$

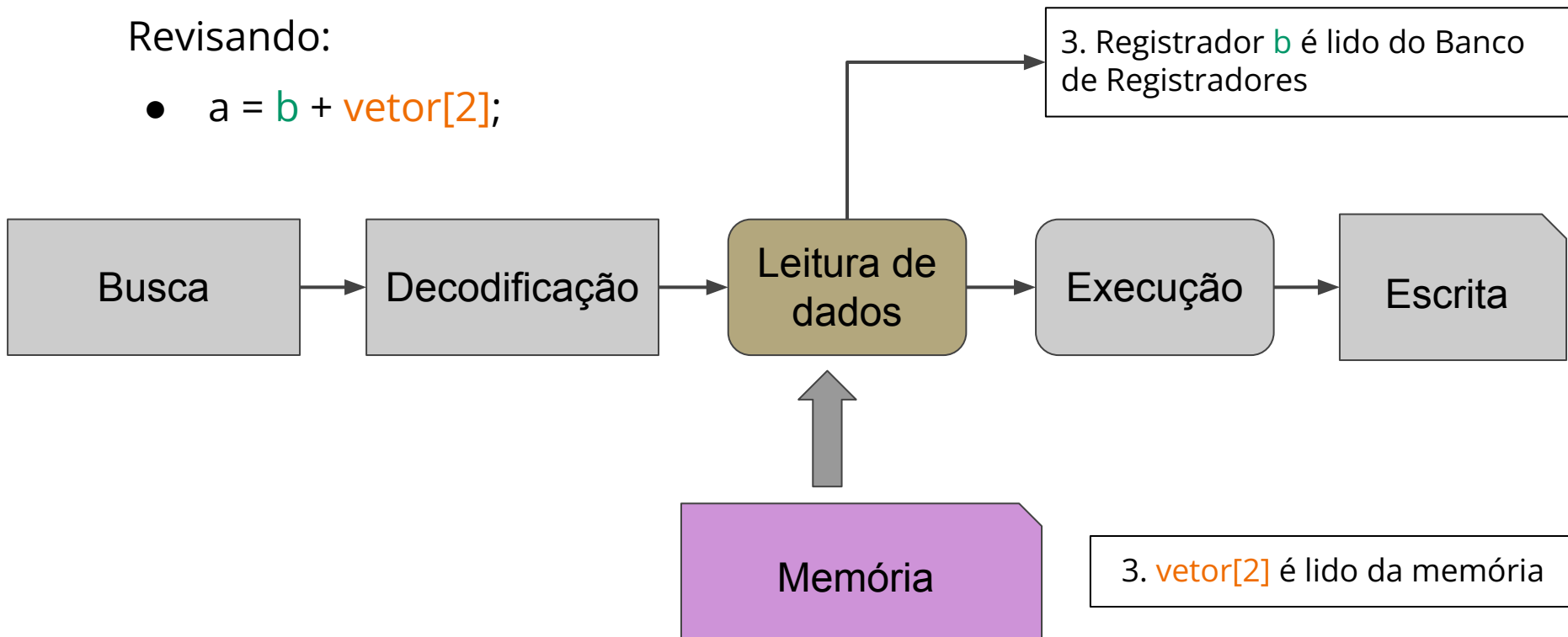


2. Sinais de controle são obtidos pela decodificação da instrução

Circuito inicial - Leitura de operandos

Revisando:

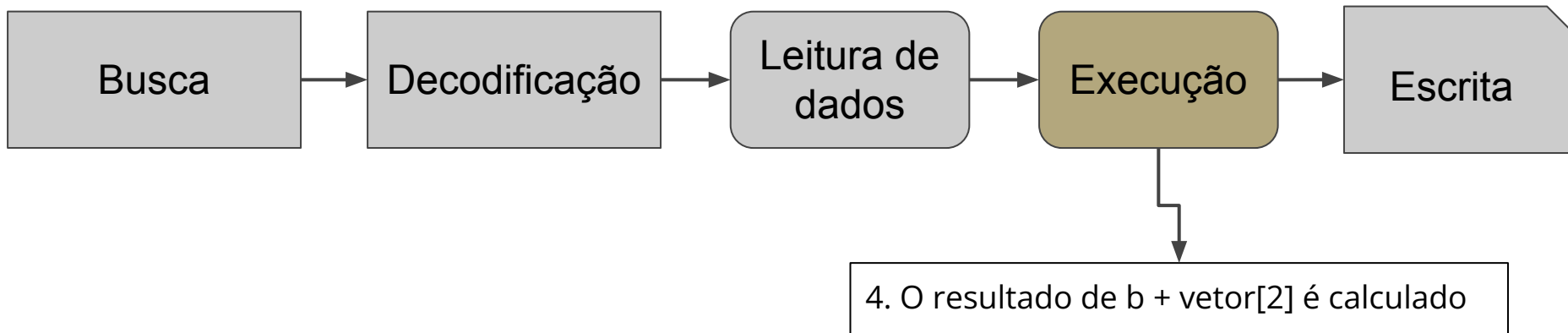
- $a = b + \text{vetor}[2];$



Circuito inicial - Execução da operação

Revisando:

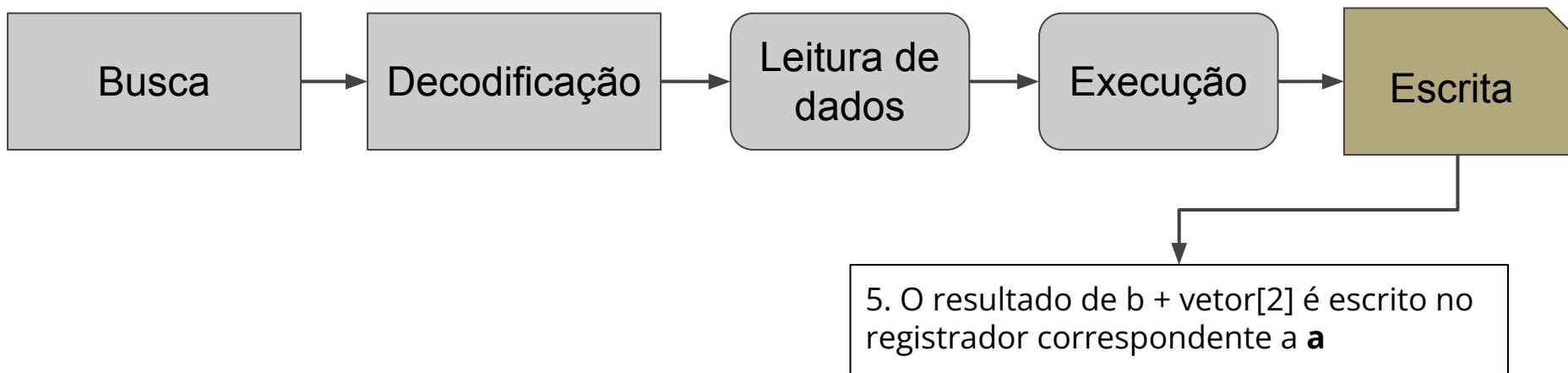
- $a = b + \text{vetor}[2];$



Circuito inicial - Escrita do resultado

Revisando:

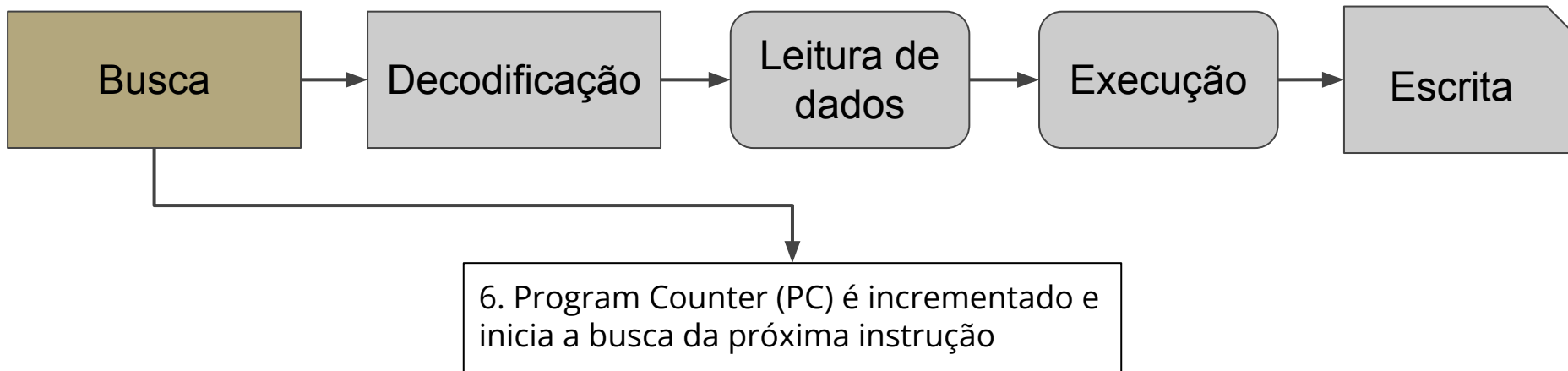
- $\mathbf{a} = b + \text{vetor}[2];$



Circuito inicial - Próxima instrução

Revisando:

- **a** = b + vetor[2];



Monociclo - Exercício 1

Considerando um processador monociclo com as seguintes latências:

Leitura em memória: 100 ns	Soma: 5 ns
Escrita em memória: 300 ns	Multiplicação: 15 ns
Decodificação: 10 ns	Banco de Registradores: 10 ns

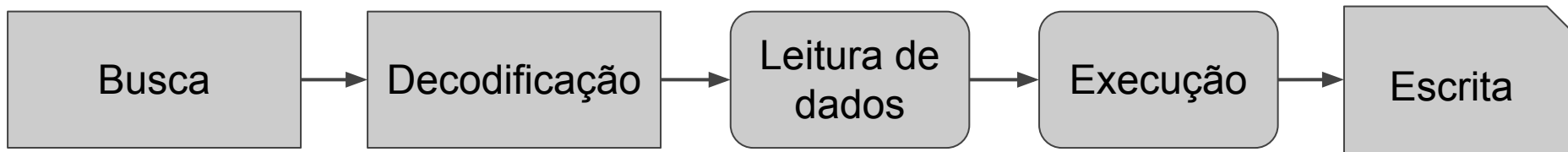
Qual o tempo necessário para executar cada uma das seguintes instruções?

- $a = \text{Vetor}[5] + e;$
- $b = c * d;$

Monociclo - Exercício 1 - Resolução (a)

$a = \text{Vetor}[5] + e;$

Tempo total:

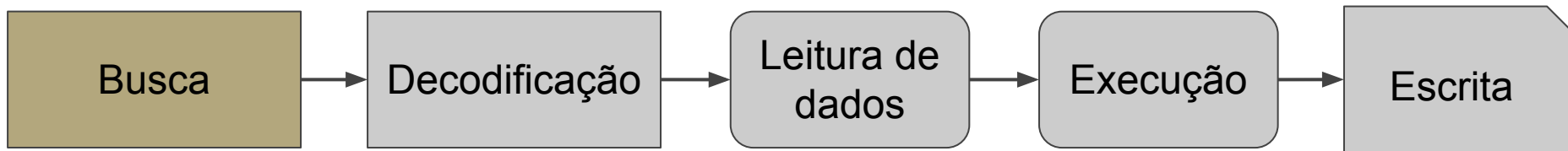


Leitura em memória: 100 ns	Soma: 5 ns
Escrita em memória: 300 ns	Multiplicação: 15 ns
Decodificação: 10 ns	Banco de Registradores: 10 ns

Monociclo - Exercício 1 - Resolução (a)

$a = \text{Vetor}[5] + e;$

Tempo total: 100



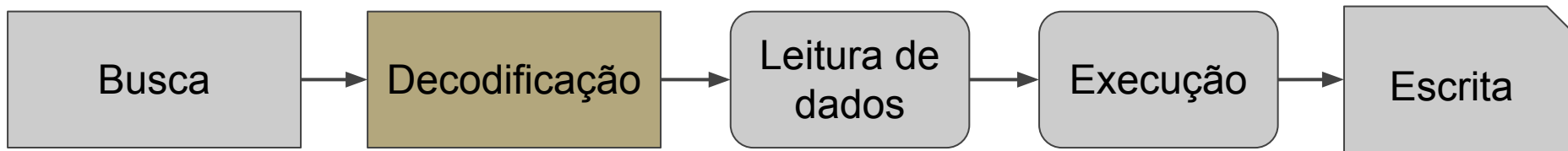
Leitura da memória: 100 ns

Leitura em memória: 100 ns	Soma: 5 ns
Escrita em memória: 300 ns	Multiplicação: 15 ns
Decodificação: 10 ns	Banco de Registradores: 10 ns

Monociclo - Exercício 1 - Resolução (a)

$a = \text{Vetor}[5] + e;$

Tempo total: 100 + 10



Decodificação: 10 ns

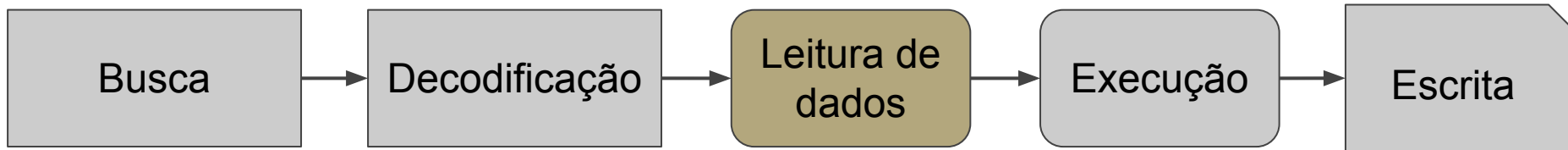
Leitura em memória: 100 ns	Soma: 5 ns
Escrita em memória: 300 ns	Multiplicação: 15 ns
Decodificação: 10 ns	Banco de Registradores: 10 ns

Monociclo - Exercício 1 - Resolução (a)

a = Vetor[5] + e;

Tempo total: 100 + 10

Leitura em memória: 100 ns	Soma: 5 ns
Escrita em memória: 300 ns	Multiplicação: 15 ns
Decodificação: 10 ns	Banco de Registradores: 10 ns



Leitura de memória : 100 ns

ou

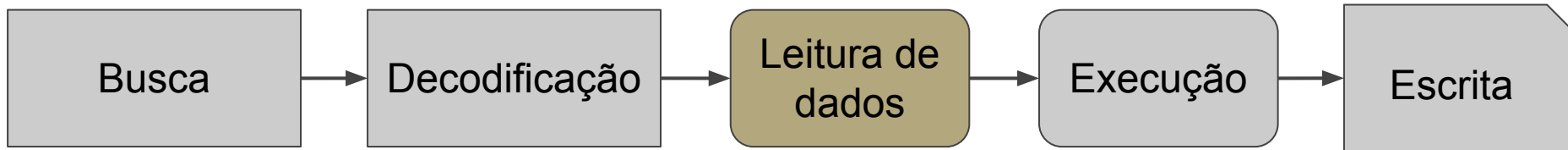
Leitura do Banco de Registradores: 10 ns

Monociclo - Exercício 1 - Resolução (a)

a = Vetor[5] + e;

Tempo total: 100 + 10 + 100

Leitura em memória: 100 ns	Soma: 5 ns
Escrita em memória: 300 ns	Multiplicação: 15 ns
Decodificação: 10 ns	Banco de Registradores: 10 ns



Leitura de memória : 100 ns

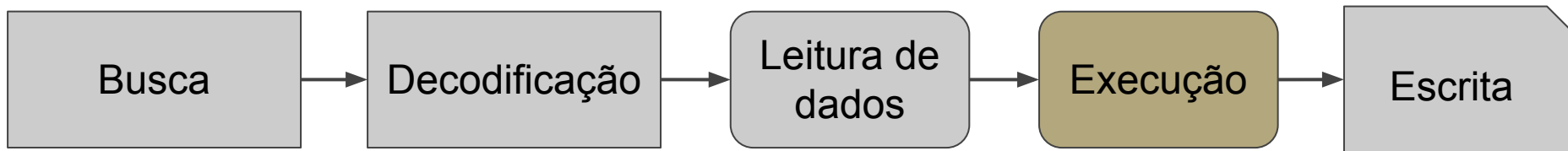
ou

Leitura do Banco de Registradores: 10 ns

Monociclo - Exercício 1 - Resolução (a)

$a = \text{Vetor}[5] + e;$

Tempo total: $100 + 10 + 100 + 5$



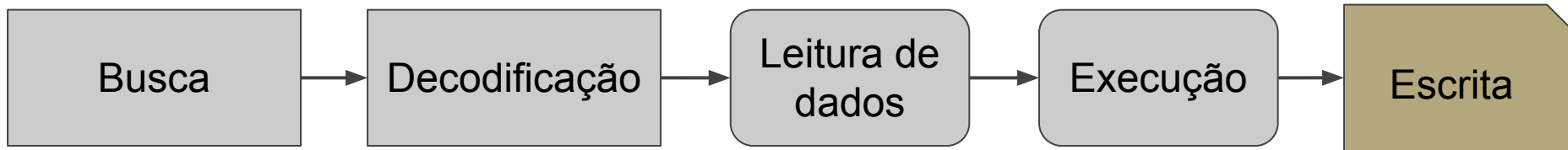
Soma: 5 ns

Leitura em memória: 100 ns	Soma: 5 ns
Escrita em memória: 300 ns	Multiplicação: 15 ns
Decodificação: 10 ns	Banco de Registradores: 10 ns

Monociclo - Exercício 1 - Resolução (a)

$a = \text{Vetor}[5] + e;$

Tempo total: $100 + 10 + 100 + 5 + 10$



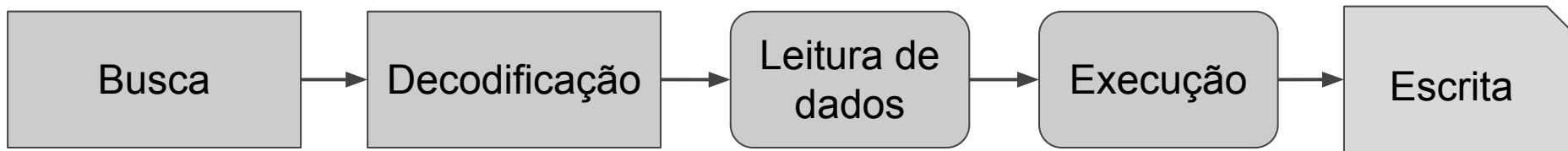
Escrita no Banco de
registradores: 10 ns

Leitura em memória: 100 ns	Soma: 5 ns
Escrita em memória: 300 ns	Multiplicação: 15 ns
Decodificação: 10 ns	Banco de Registradores: 10 ns

Monociclo - Exercício 1 - Resolução (a)

$a = \text{Vetor}[5] + e;$

Tempo total: $100 + 10 + 100 + 5 + 10 = \mathbf{225 \text{ ns}}$



Leitura em memória: 100 ns	Soma: 5 ns
Escrita em memória: 300 ns	Multiplicação: 15 ns
Decodificação: 10 ns	Banco de Registradores: 10 ns

Monociclo - Exercício 1

Considerando um processador monociclo com as seguintes latências:

Leitura em memória: 100 ns	Soma: 5 ns
Escrita em memória: 300 ns	Multiplicação: 15 ns
Decodificação: 10 ns	Banco de Registradores: 10 ns

Qual o tempo necessário para executar cada uma das seguintes instruções?

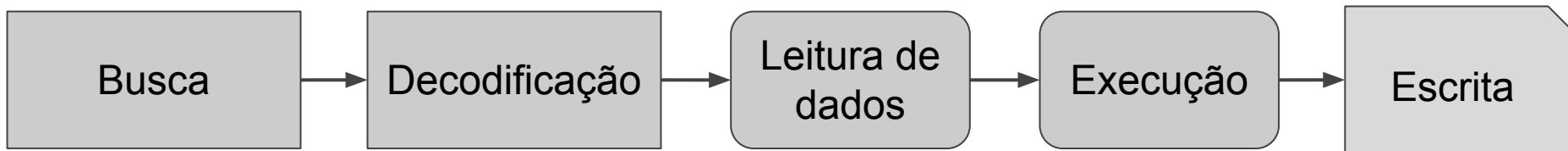
- $a = \text{Vetor}[5] + e; \rightarrow \mathbf{225 \text{ ns}}$
- $b = c * d; \rightarrow ?$

Monociclo - Exercício 1 - Resolução (b)

$b = c + d; \rightarrow ?$

Leitura em memória: 100 ns	Soma: 5 ns
Escrita em memória: 300 ns	Multiplicação: 15 ns
Decodificação: 10 ns	Banco de Registradores: 10 ns

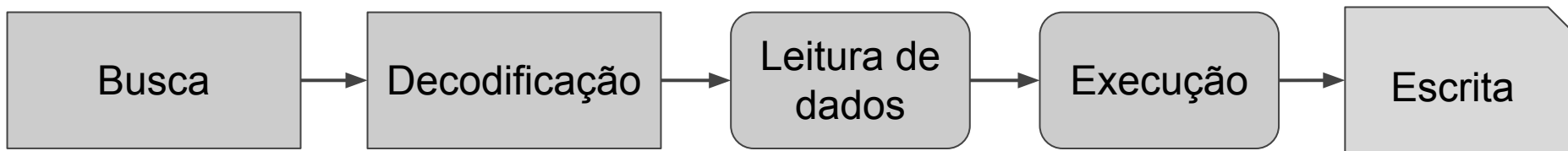
1. Tempo de Busca: 100 ns
2. Tempo de Decodificação: 10 ns
3. Tempo de leitura do Banco de Registradores: 10 ns
4. Multiplicação: 15 ns
5. Escrita no Banco de Registradores: 10 ns



Monociclo - Exercício 1 - Resolução (b)

$b = c + d$; \rightarrow Tempo total: $100 + 10 + 10 + 15 + 10 = \mathbf{145\ ns}$

1. Tempo de Busca: **100** ns
2. Tempo de Decodificação: **10** ns
3. Tempo de leitura do Banco de Registradores: **10** ns
4. Multiplicação: **15** ns
5. Escrita no Banco de Registradores: **10** ns



Monociclo - Exercício 4

Considerando um processador monociclo com as seguintes latências:

Leitura em memória: 100 ns	Soma: 5 ns
Escrita em memória: 300 ns	Multiplicação: 15 ns
Decodificação: 10 ns	Banco de Registradores: 10 ns

Qual o tempo necessário para executar o seguinte programa?

a = b + c;

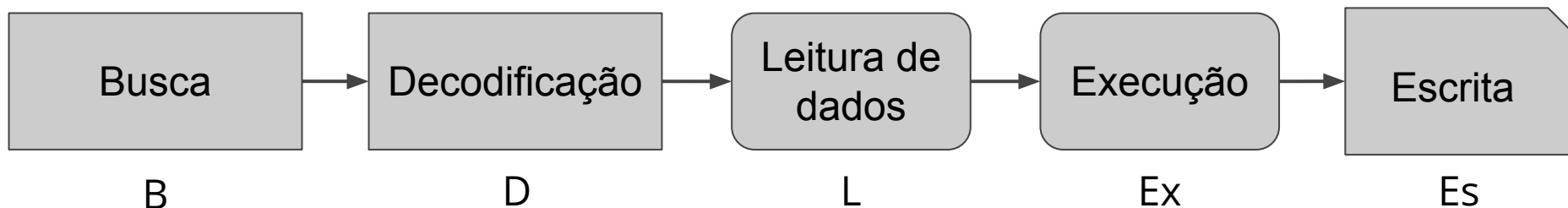
v[3] = a;

a = v[4] * d;

▼ v[0] = v[3] - a;

Otimizando nosso sistema

O que fazemos agora:



Otimizando nosso sistema

O que fazemos agora:

Programa
Instrução 0
Instrução 1
Instrução 2

Estágios

Otimizando nosso sistema

O que fazemos agora:

Programa

Instrução 0

Instrução 1

Instrução 2

Estágios														
B	D	L	Ex	Es										
					B	D	L	Ex	Es					

Otimizando nosso sistema

O que fazemos agora:

Programa

Instrução 0

Instrução 1

Instrução 2

Estágios														
B	D	L	Ex	Es										
					B	D	L	Ex	Es					
										B	D	L	Ex	Es

Otimizando nosso sistema

O que fazemos agora:

Programa

Instrução 0

Instrução 1

Instrução 2

Estágios														
B	D	L	Ex	Es										
					B	D	L	Ex	Es					
										B	D	L	Ex	Es

Tempo de execução

Otimizando nosso sistema

O que fazemos agora:

Programa

Instrução 0

Instrução 1

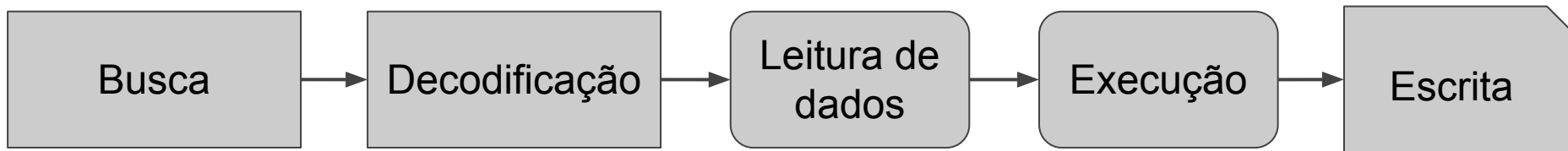
Instrução 2



Tempo de execução = Tempo da Instrução 0 + Tempo da Instrução 1 + Tempo da Instrução 2

Instrução

Aproveitando essa organização, como podemos acelerar a execução de múltiplas instruções?



Pipeline

Pipeline

Técnica de *hardware* que permite que a CPU execute simultaneamente diversas instruções

O processamento de cada instrução é subdividido em etapas, sendo que cada etapa é executada por uma porção especializada da CPU

Execução em Pipeline

Programa

Instrução 0

Instrução 1

Instrução 2

Estágios														
B	D	L	Ex	Es										
	B	D	L	Ex	Es									
		B	D	L	Ex	Es								



Tempo de execução = Tempo da Instrução 0 + 2 ciclos

Vantagens do pipeline

Pipelining **não reduz a latência** de uma instrução única.

Mas **aumenta o throughput (vazão)** de todo workload

Inst.	Estágio do pipeline					
1	B	D	L	Ex	Es	
2		B	D	L	Ex	Es
3			B	D	L	Ex
4				B	D	L
Ciclo de clock	1	2	3	4	5	6

Conceitos importantes

Frequência

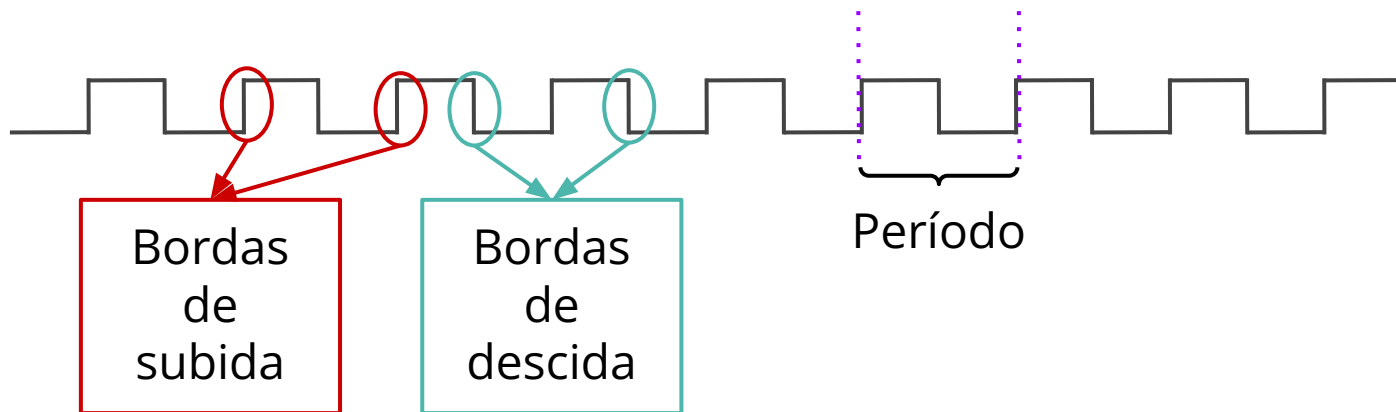
- Número de voltas realizadas por um objeto em movimento circular em um intervalo de tempo determinado.
- Quantidade de ondas geradas em um tempo específico.

Período

- Tempo necessário para que um objeto em movimento circular conclua uma volta
- Tempo necessário para que uma onda seja formada

Clock

- Sinal para sincronização dos componentes



Clock - Pipeline

Período de clock

- Maior ou igual ao tempo de execução do estágio mais demorado
- Instruções por ciclo (IPC) ideal = 1

Pipeline - Exercício 1

Suponha um pipeline operando a uma frequência de clock de 1 Hz.
Qual o tempo ideal de execução de 10 instruções nesse sistema?

Estágios do pipeline				
B	D	L	Ex	Es

frequência = 1 / período