Arquitectura de Computadores I

Breve História dos Computadores

Miguel Barão

ESCOLA DE CIÊNCIAS E TECNOLOGIA
DEPARTAMENTO DE INFORMÁTICA

Resumo

Interface Hardware/Software

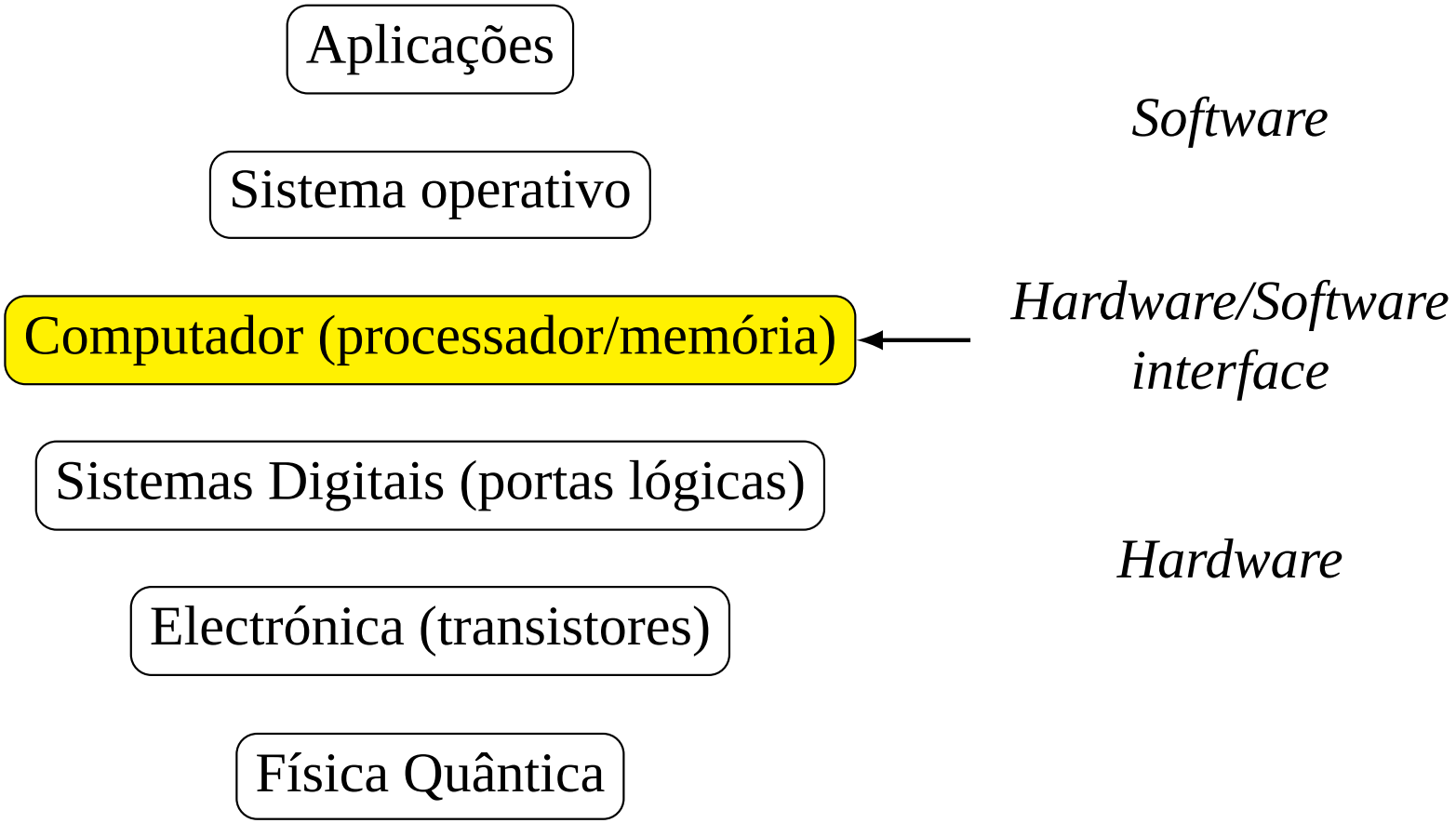
Um pouco de história

Computador e dispositivos de Input/Output (IO)

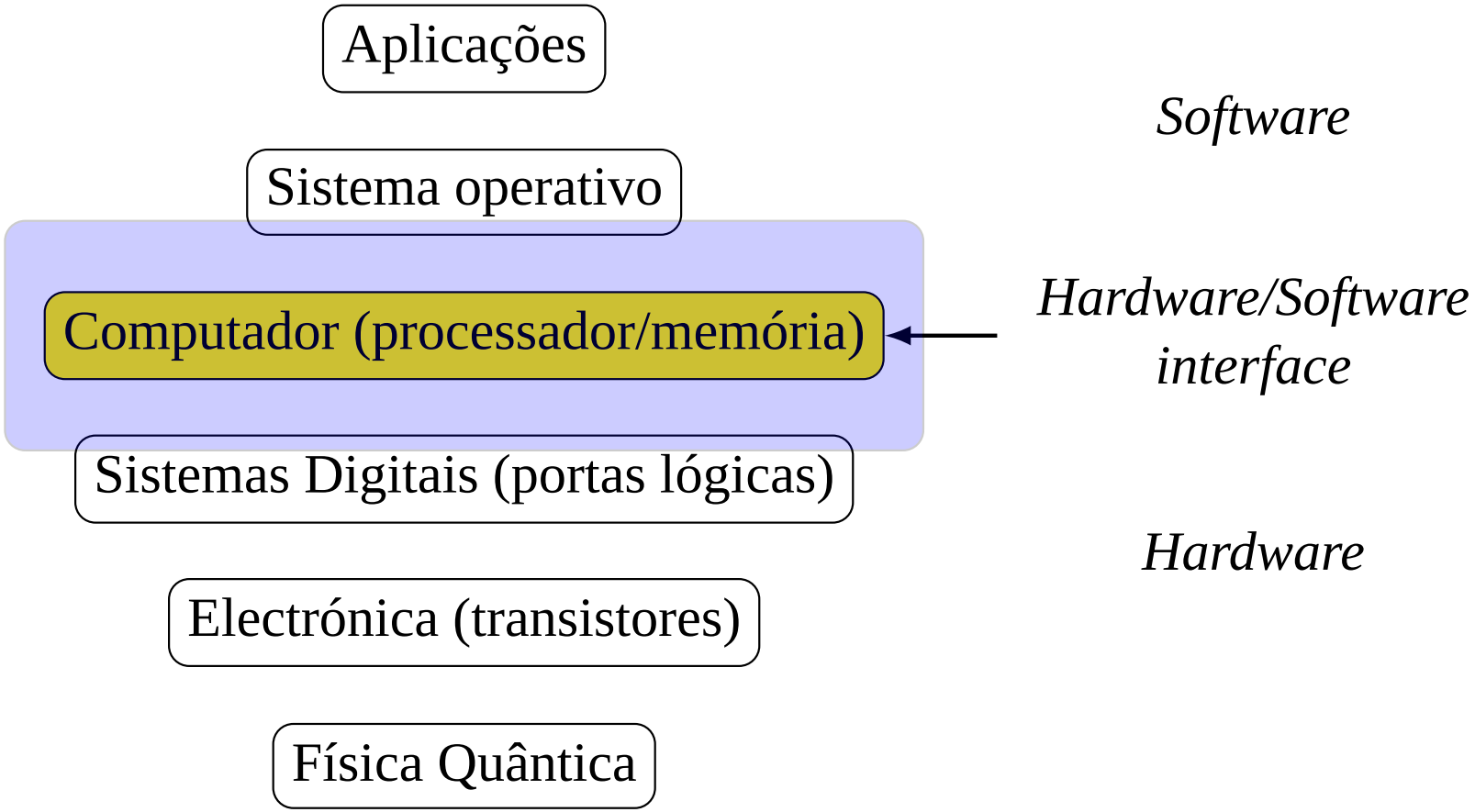
Arquitecturas Harvard e Von Neumann

Interface Hardware/Software

Interface Hardware/Software



Interface Hardware/Software

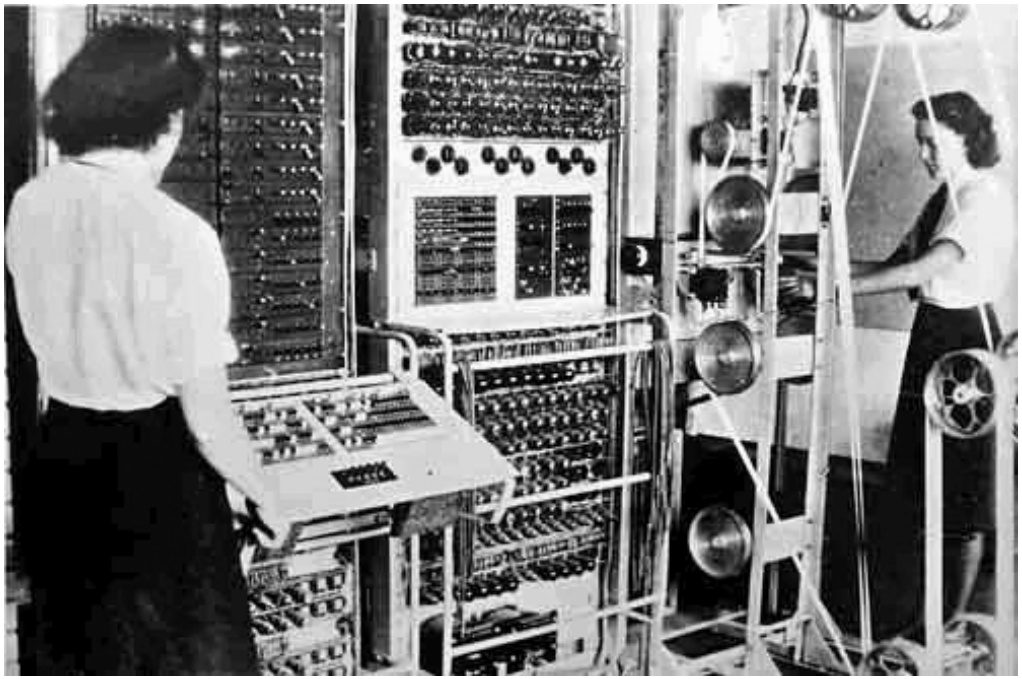


Vamos estudar o interface Hardware/Software: como o computador está organizado e como executa as instruções de um programa.

Um pouco de história

Colossus

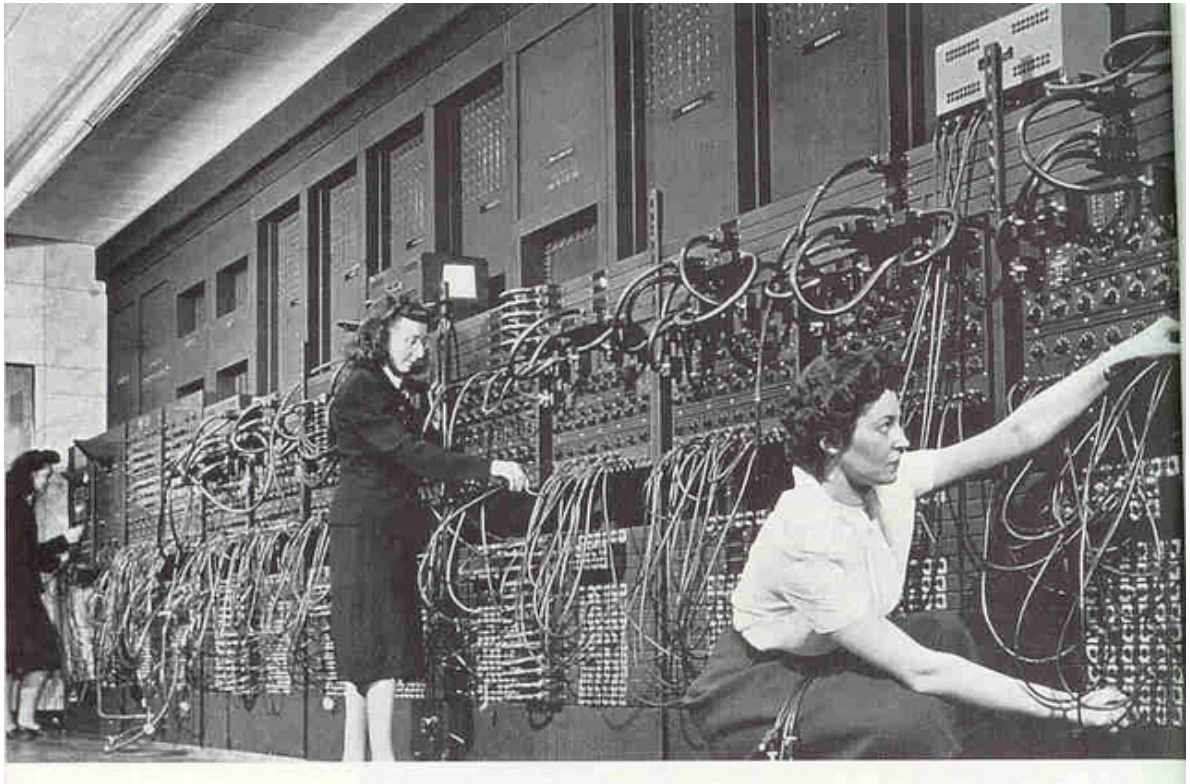
Desenvolvido pelos Britânicos para decifrar mensagens dos Alemães na 2ª Guerra Mundial, 1943–1945.



- Os primeiros computadores executavam programas fixos.
- Para correr um programa diferente era necessário modificar fisicamente um conjunto de interruptores e/ou a cablagem.
- As calculadoras básicas também executam programas fixos.

ENIAC (Electronic Numerical Integrator And Computer)

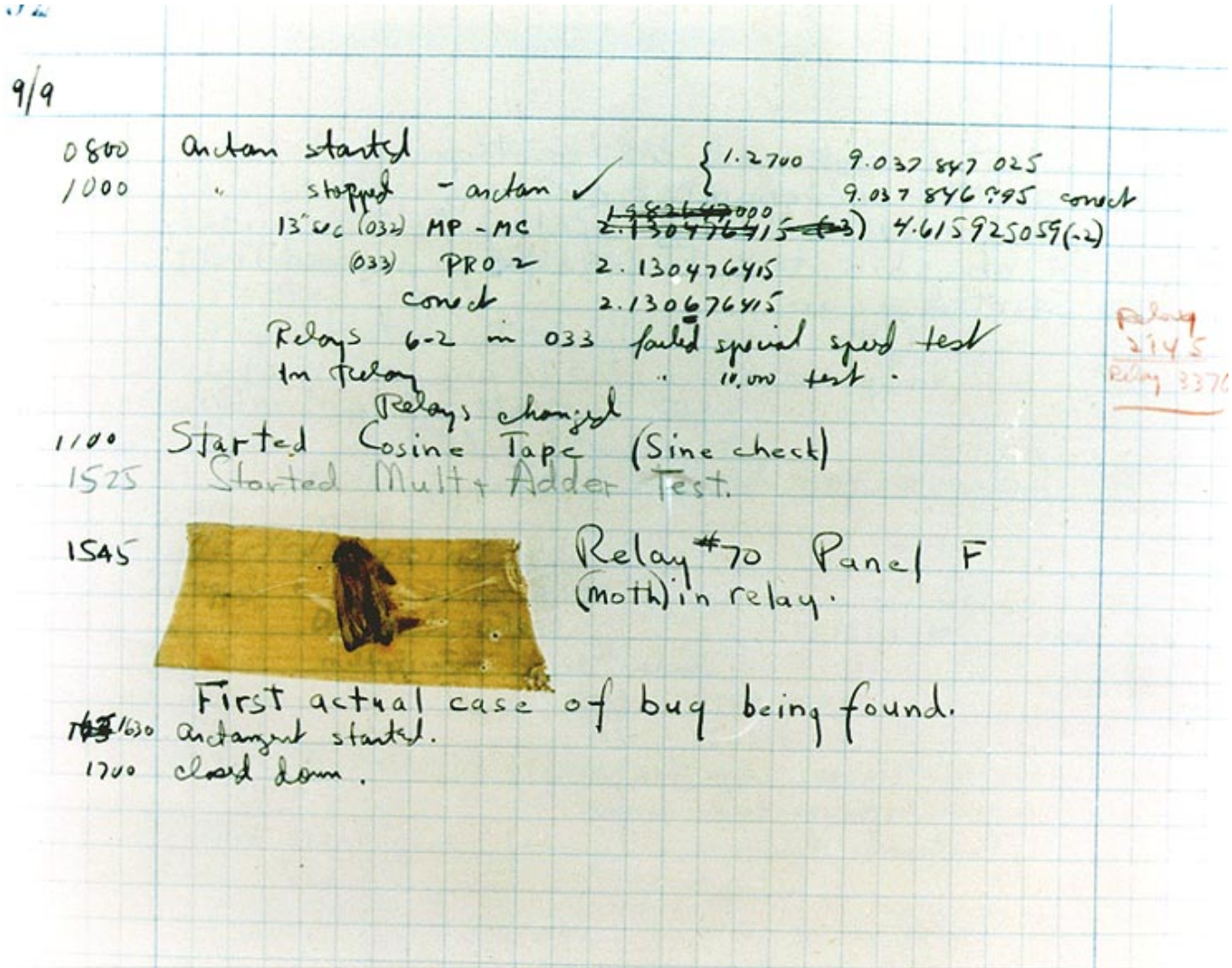
Desenvolvido pelos EUA para cálculos da bomba atómica e balística, 1945–1955.



- Executava também programas fixos, tal como o Colossus.
- Para correr um programa diferente era necessário modificar fisicamente um conjunto de interruptores e/ou a cablagem.

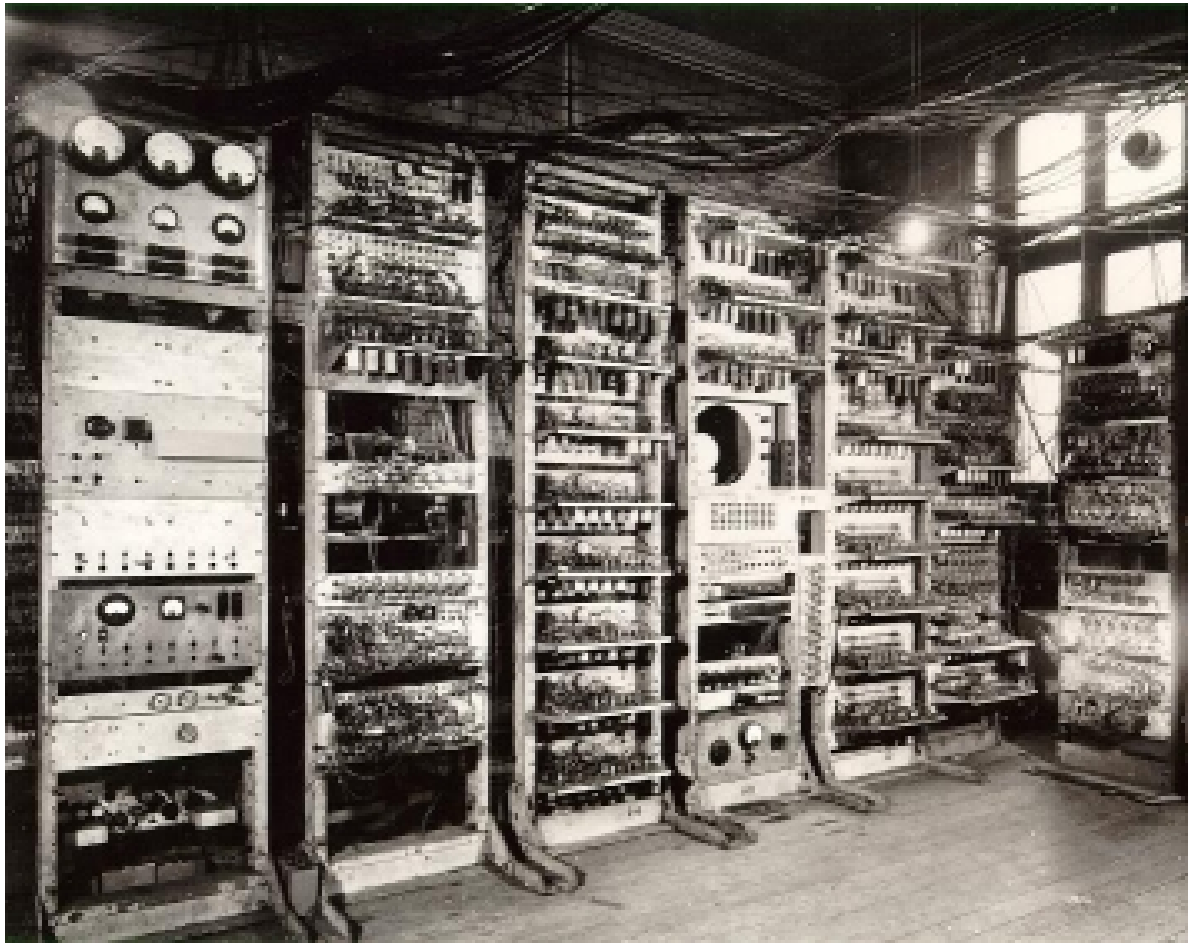
O primeiro “bug” de software que era realmente um bug

Uma traça encontrada num relé do computador Harvard Mark II em 1947



Manchester Mark 1

O primeiro “stored-program computer”, 1949



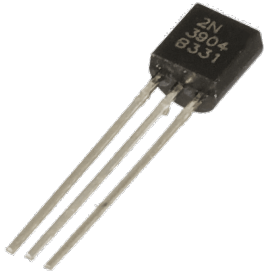
- Pela primeira vez, o programa são dados.
- 4050 válvulas, potência 25 kW.

Salto tecnológico

Válvula

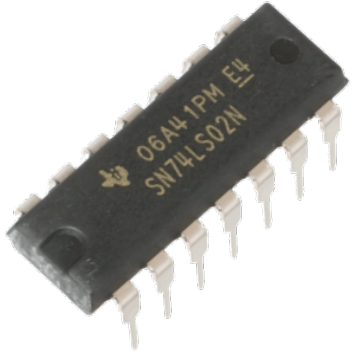
→

Transistor



→

Circuito Integrado



- Redução na energia consumida, tamanho e custo.
- Aumento do desempenho e fiabilidade.

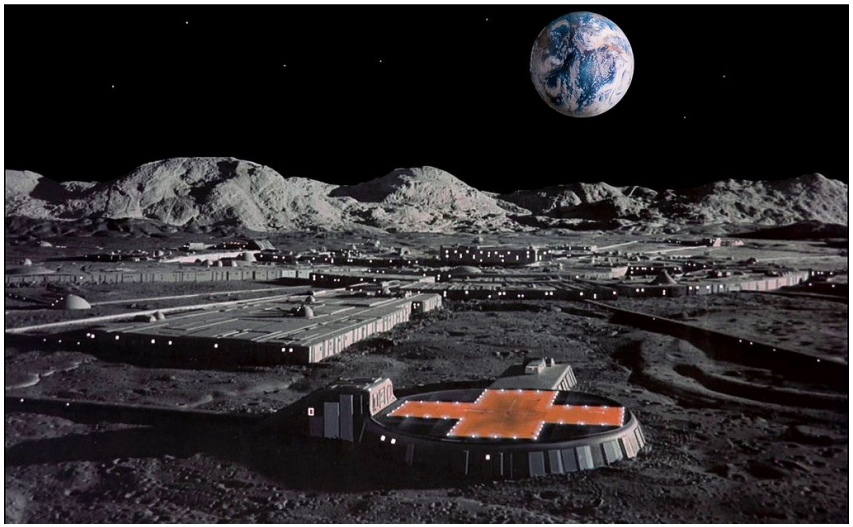
IBM System/360 Mainframe

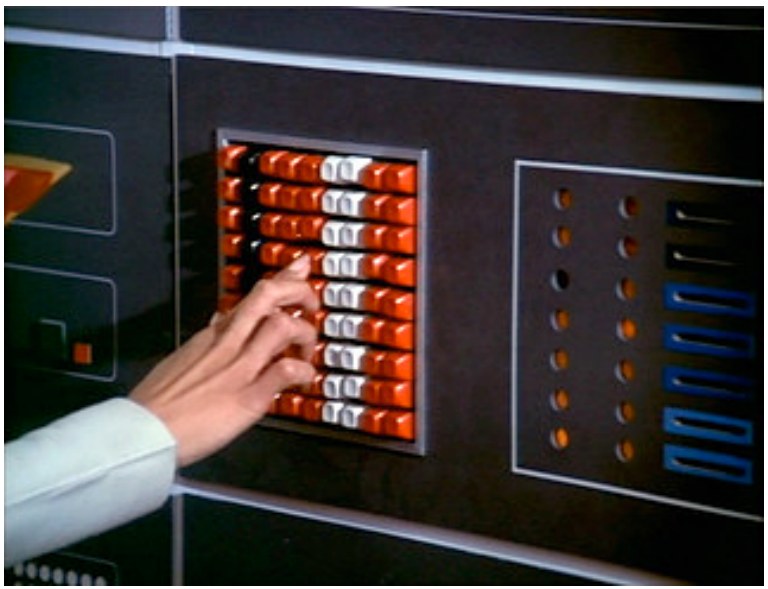
1964–1977

Influência na indústria:

- Byte de 8 bits.
- Memória endereçável ao nível do byte.
- Words de 32 bits.

Ficção científica em 1973 - Space 1999



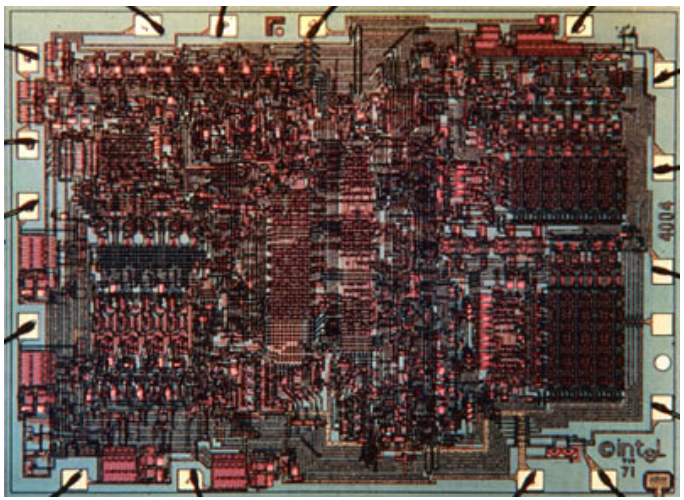


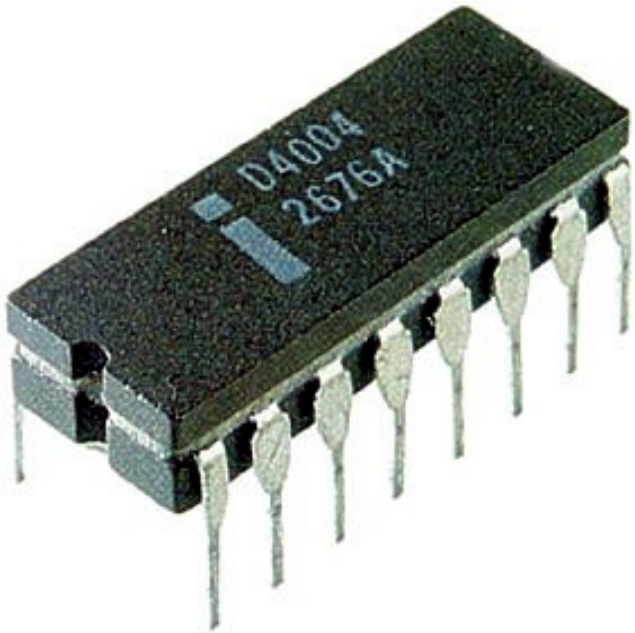
DEC PDP-11 (1970–1990)

- Considerado um Minicomputador (custo até 150.000 €).
- O primeiro Unix correu num PDP-11/70 em 1970.
- Utilização empresarial, médica, industrial, científica, educação.

Intel 4004

CPU de 4 bits desenvolvido em 1971.





- Primeiro microprocessador comercialmente disponível.
- Ocupa uma área de $12mm^2$.
- Executa cerca de 92000 instr./s.

MOS Tech 6502

CPU de 8 bits desenvolvido em 1975.

- Processador muito barato. Custava $25 na altura.
- Deu origem à revolução dos computadores pessoais.
- Usado em:
  - Microcomputadores: BBC, Apple IIe, Atari 400/800.
  - Plataformas de jogos: Atari 2600 e Nintendo Entertainment System (NES).

Revolução dos Microcomputadores pessoais

Apple II (1979–1982)
microprocessor 6502

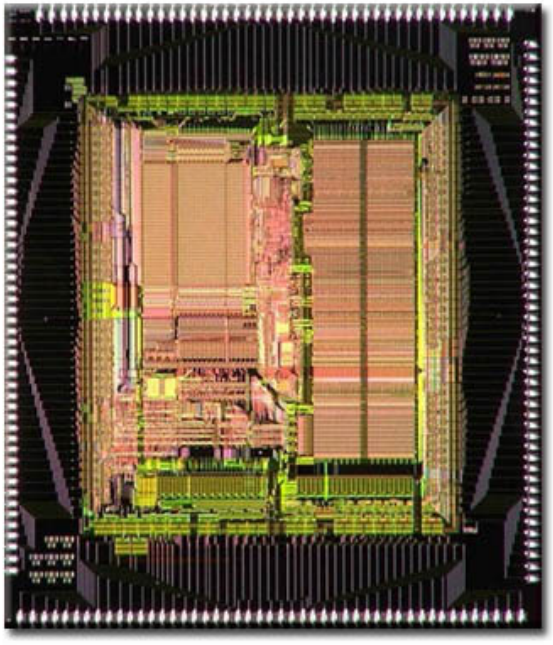
BBC (1982–1994)
microprocessor 6502

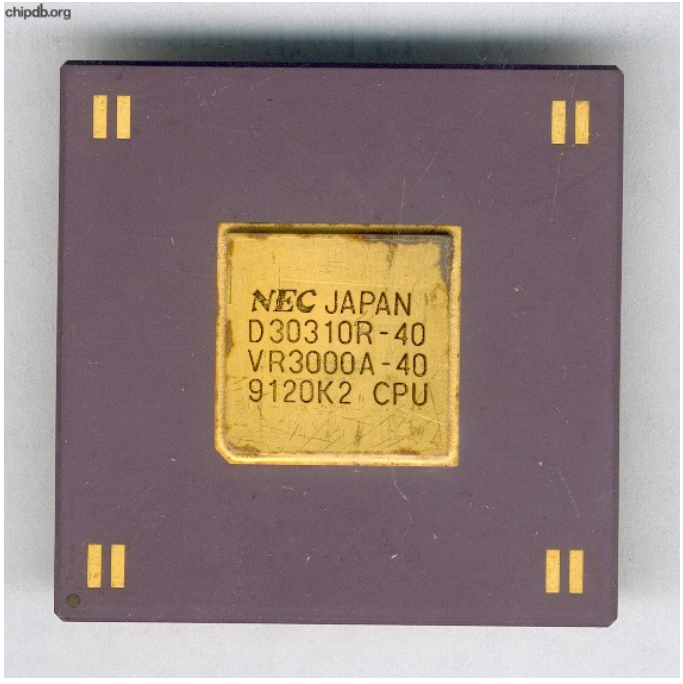
Comodore 64 (1982–1994)
microprocessor 6510

ZX Spectrum 48k (1982–1992)
microprocessor Z80

MIPS R2000

CPU de 32bits desenvolvido em 1986.

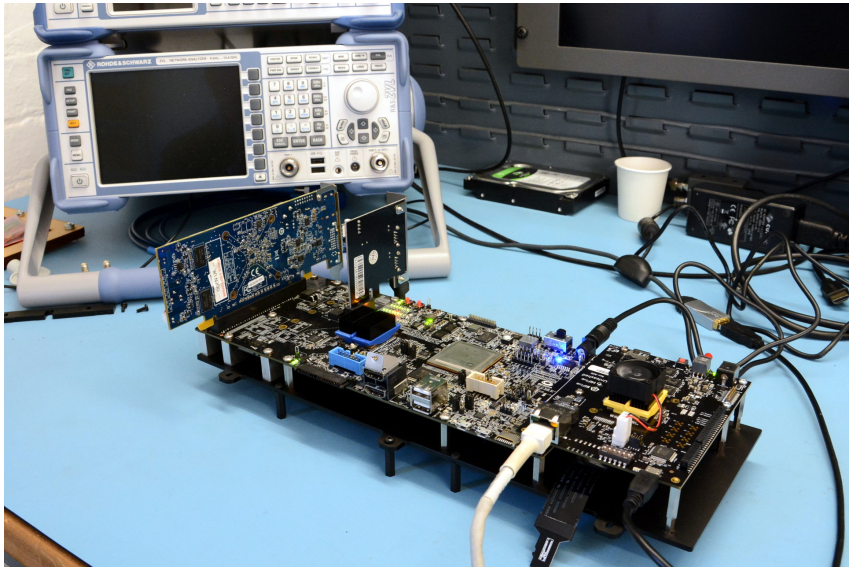




- Primeiro processador RISC disponível comercialmente.
- Execução de instruções em pipeline.
- Usado em workstations (Silicon Graphics, MIPS, DEC, etc).
- Processadores desta arquitectura são actualmente usados em dispositivos de rede, consolas de jogos, impressoras, set-top boxes, televisões digitais e modems de cabo/ADSL.

RISC V

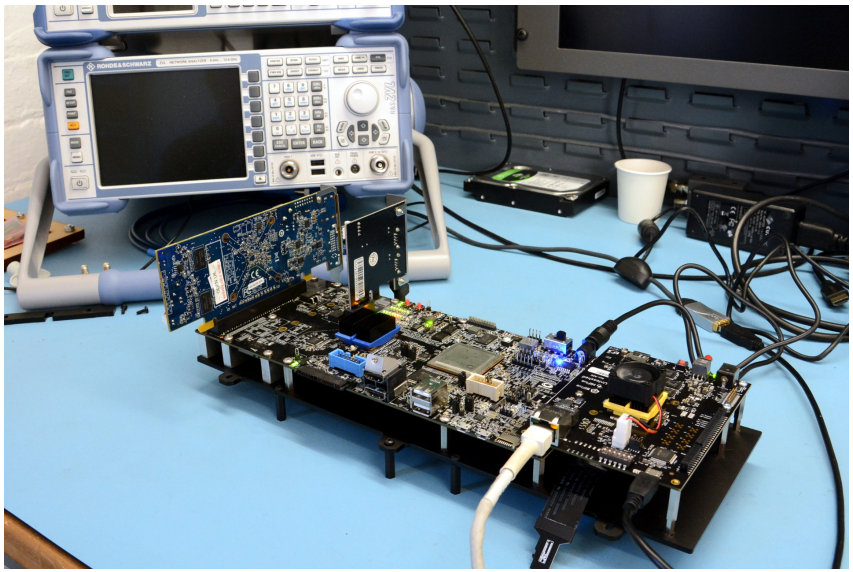
Arquitectura com licensa open-source. 2010



- Outras arquitecturas (ARM, MIPS, ...) cobram royalties, obrigam a assinatura de NDA e não partilham informação.
- A arquitectura RISC-V foi criada para resolver estes problemas e ser fácil de aprender.
- Pode ser usada desde pequenos computadores IoT até supercomputadores.
- Expansível. Apoio da indústria.

RISC V

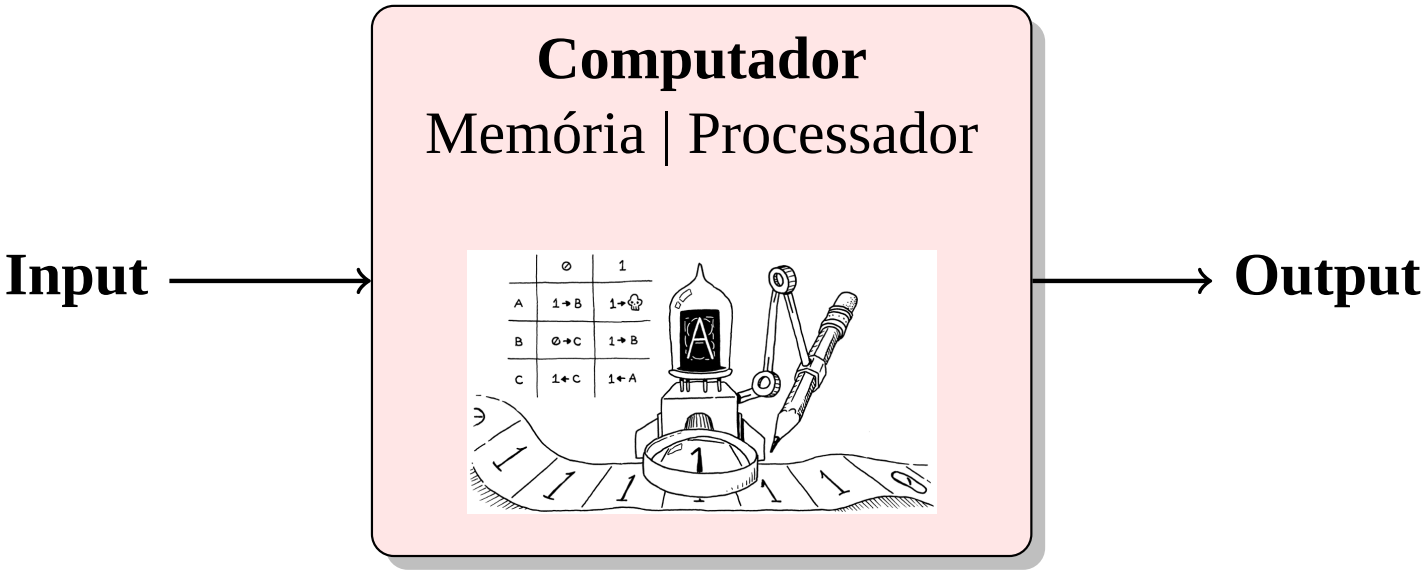
Arquitectura com licensa open-source. 2010



Esta é a aquitectura que vamos estudar este semestre!

Computador e dispositivos de Input/Output (IO)

Computador e Input/Output



Um sistema computacional:

- recebe informação do mundo (input)
- processa a informação (computação)
- envia o resultado para o mundo (output)

ASCII - American Standard Code for Information Interchange

	hex		hex		hex		hex		hex		hex		hex		hex
NUL	00	DLE	10		20	0	30	@	40	P	50	‘	60	p	70
SOH	01	DC1	11	!	21	1	31	A	41	Q	51	a	61	q	71
STX	02	DC2	12	”	22	2	32	B	42	R	52	b	62	r	72
ETX	03	DC3	13	#	23	3	33	C	43	S	53	c	63	s	73
EOT	04	DC4	14	$	24	4	34	D	44	T	54	d	64	t	74
ENQ	05	NAK	15	%	25	5	35	E	45	U	55	e	65	u	75
ACK	06	SYN	16	&	26	6	36	F	46	V	56	f	66	v	76
BEL	07	ETB	17	’	27	7	37	G	47	W	57	g	67	w	77
BS	08	CAN	18	(	28	8	38	H	48	X	58	h	68	x	78
TAB	09	EM	19	)	29	9	39	I	49	Y	59	i	69	y	79
LF	0a	SUB	1a	*	2a	:	3a	J	4a	Z	5a	j	6a	z	7a
VT	0b	ESC	1b	+	2b	;	3b	K	4b	[	5b	k	6b	{	7b
FF	0c	FS	1c	,	2c	<	3c	L	4c	\	5c	l	6c	|	7c
CR	0d	GS	1d	-	2d	=	3d	M	4d	]	5d	m	6d	}	7d
SO	0e	RS	1e	.	2e	>	3e	N	4e	^	5e	n	6e	~	7e
SI	0f	US	1f	/	2f	?	3f	O	4f	_	5f	o	6f	DEL	7f

- Desenvolvido nos anos 60 para comunicações de texto.
- Usado para o Input/Output dos computadores da altura.

Input/Output (IO) - Teleimpressora

Teleimpressora TTY

Computador

Output

Input

Input teclado envia caracteres ASCII para o computador.

Output computador envia caracteres ASCII para serem impressos num rolo de papel contínuo.

Input/Output (IO) - Videoterminal

Videoterminal

Computador

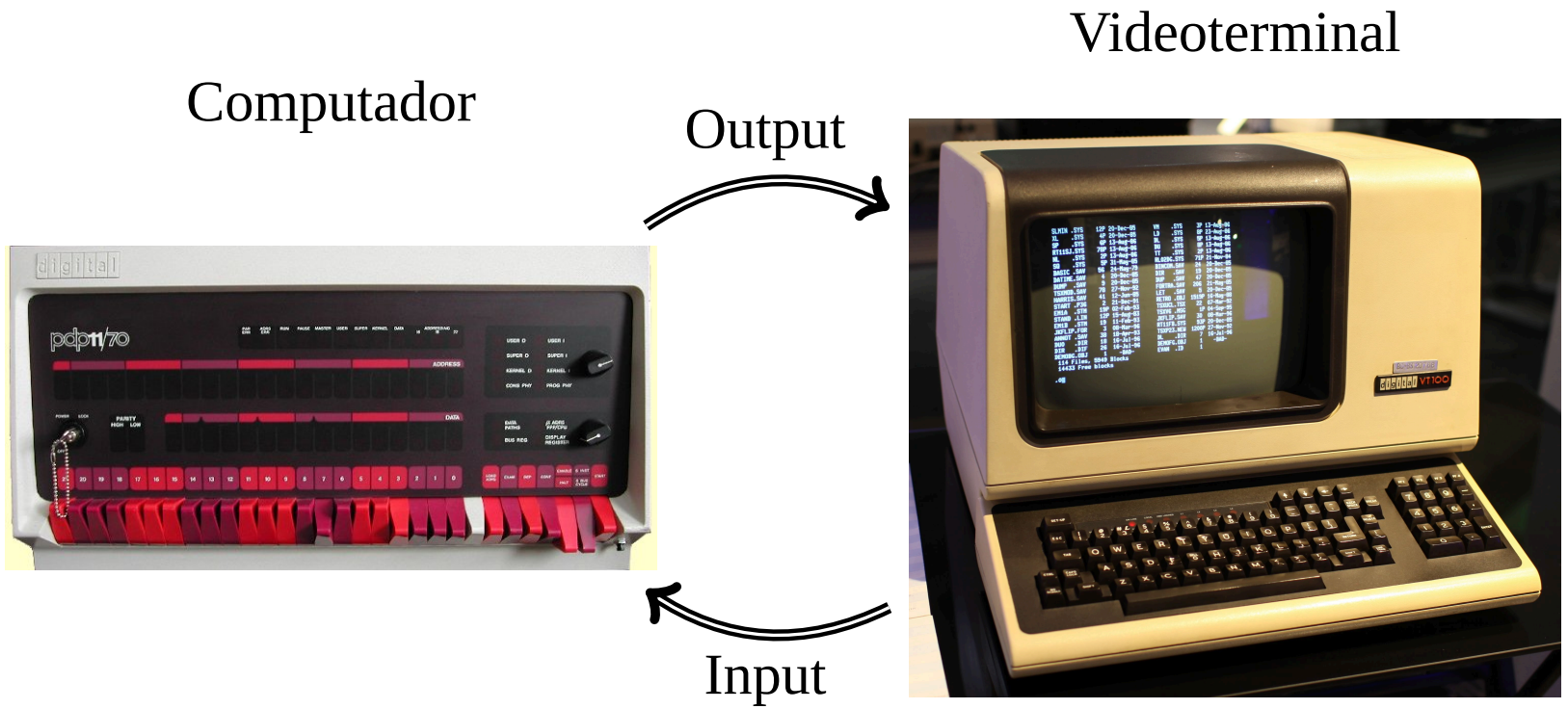
Output

Input

Input teclado envia caracteres ASCII para o computador.

Output computador envia caracteres ASCII para serem mostrados no ecrã.

Input/Output (IO) - Videoterminal



Input teclado envia caracteres ASCII para o computador.

Output computador envia caracteres ASCII para serem mostrados no ecrã.

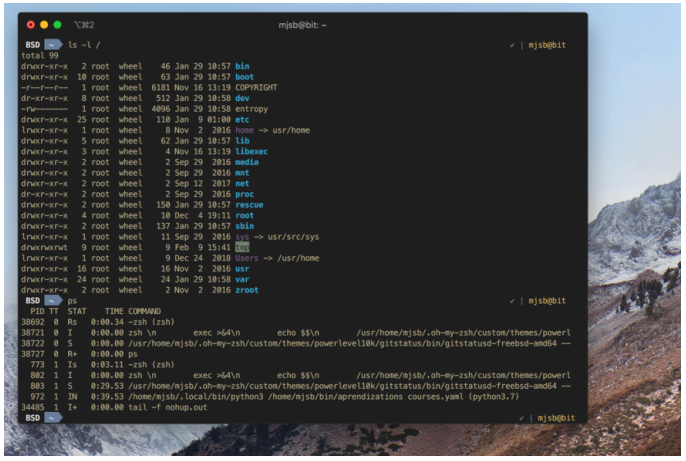
O ecrã substituiu a impressora da teleimpressora.

O texto desliza para cima tal como o papel da impressora!

Input/Output (IO) - Emulador de terminal

Computador

Emulador de terminal



- Computador e dispositivos de interface estão juntos.
- O comportamento dos videoterminais antigos passou a ser simulado por um programa chamado emulador de terminal.
- O emulador de terminal envia e recebe caracteres ASCII, e só mostra texto tal como os videoterminais.

Curiosidade: Sequências de escape

Se o input/output é só ASCII, como se codificam

- as setas, PageUp/PageDn, Home/End, F1, F2, ...

Curiosidade: Sequências de escape

Se o input/output é só ASCII, como se codificam

- as setas, PageUp/PageDn, Home/End, F1, F2, ...
- as cores, o ASCII não tem cores...

Curiosidade: Sequências de escape

Se o input/output é só ASCII, como se codificam

- as setas, PageUp/PageDn, Home/End, F1, F2, ...
- as cores, o ASCII não tem cores...

O ASCII tem um conjunto limitado de códigos. Outras funcionalidades são codificadas em sequências especiais de caracteres chamadas sequências de escape.

Curiosidade: Sequências de escape

Se o input/output é só ASCII, como se codificam

- as setas, PageUp/PageDn, Home/End, F1, F2, ...
- as cores, o ASCII não tem cores...

O ASCII tem um conjunto limitado de códigos. Outras funcionalidades são codificadas em sequências especiais de caracteres chamadas sequências de escape.

 `\e[A`  `\e[B`  `\e[C`  `\e[D`

Curiosidade: Sequências de escape

Se o input/output é só ASCII, como se codificam

- as setas, PageUp/PageDn, Home/End, F1, F2, ...
- as cores, o ASCII não tem cores...

O ASCII tem um conjunto limitado de códigos. Outras funcionalidades são codificadas em sequências especiais de caracteres chamadas sequências de escape.

 `\e[A`  `\e[B`  `\e[C`  `\e[D`

```
printf("\e[38;5;1mHello \e[38;5;2mWorld\n");
Hello World
```

`\e` representa o carácter Escape (código ASCII 27).


INTERMISSION

Arquitecturas Harvard e Von Neumann

Arquitecturas Harvard e Von Neumann

Stored Program Computer É um computador em que o programa está guardado em memória.

Arquitecturas Harvard e Von Neumann

Stored Program Computer É um computador em que o programa está guardado em memória.

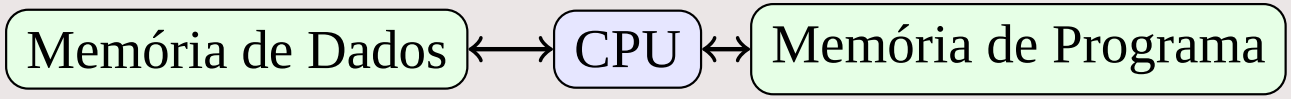
Arquitectura de Von Neumann Os programas e os dados estão guardados na mesma memória.

Arquitecturas Harvard e Von Neumann

Stored Program Computer É um computador em que o programa está guardado em memória.

Arquitectura de Von Neumann Os programas e os dados estão guardados na mesma memória.

Arquitectura de Harvard Os programas e os dados são guardados em memórias separadas.

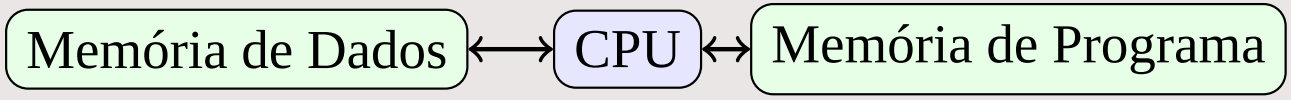


Arquitecturas Harvard e Von Neumann

Stored Program Computer É um computador em que o programa está guardado em memória.

Arquitectura de Von Neumann Os programas e os dados estão guardados na mesma memória.

Arquitectura de Harvard Os programas e os dados são guardados em memórias separadas.



Exemplos:

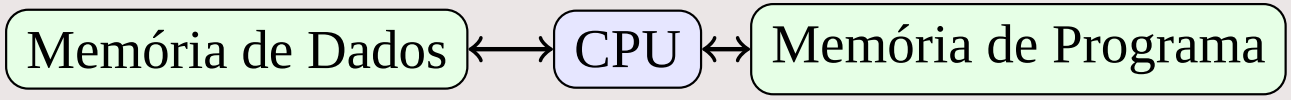
- Computadores pessoais, telemóveis?
- Microcontroladores, p. ex. arduino?
- Primeiros computadores, calculadoras?

Arquitecturas Harvard e Von Neumann

Stored Program Computer É um computador em que o programa está guardado em memória.

Arquitectura de Von Neumann Os programas e os dados estão guardados na mesma memória.

Arquitectura de Harvard Os programas e os dados são guardados em memórias separadas.



Exemplos:

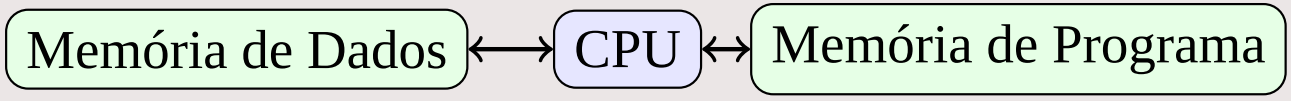
- Computadores pessoais, telemóveis? Von Neumann
- Microcontroladores, p. ex. arduino?
- Primeiros computadores, calculadoras?

Arquitecturas Harvard e Von Neumann

Stored Program Computer É um computador em que o programa está guardado em memória.

Arquitectura de Von Neumann Os programas e os dados estão guardados na mesma memória.

Arquitectura de Harvard Os programas e os dados são guardados em memórias separadas.



Exemplos:

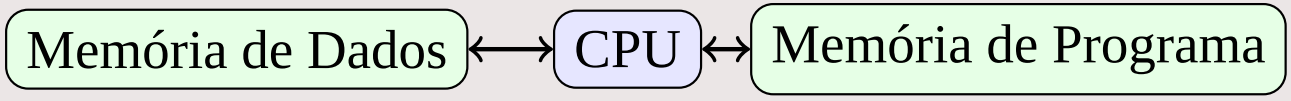
- Computadores pessoais, telemóveis? Von Neumann
- Microcontroladores, p. ex. arduino? Harvard
- Primeiros computadores, calculadoras?

Arquitecturas Harvard e Von Neumann

Stored Program Computer É um computador em que o programa está guardado em memória.

Arquitectura de Von Neumann Os programas e os dados estão guardados na mesma memória.

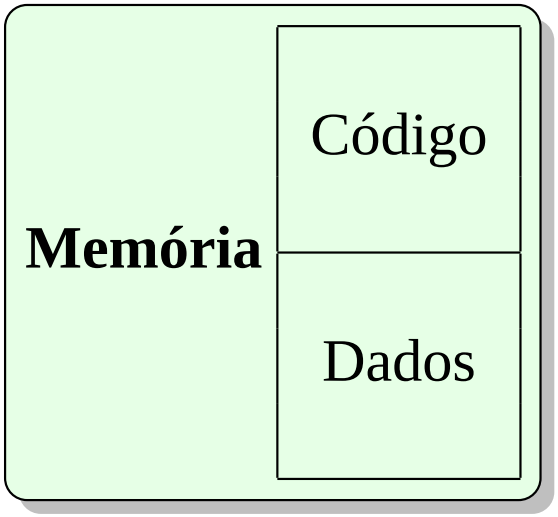
Arquitectura de Harvard Os programas e os dados são guardados em memórias separadas.

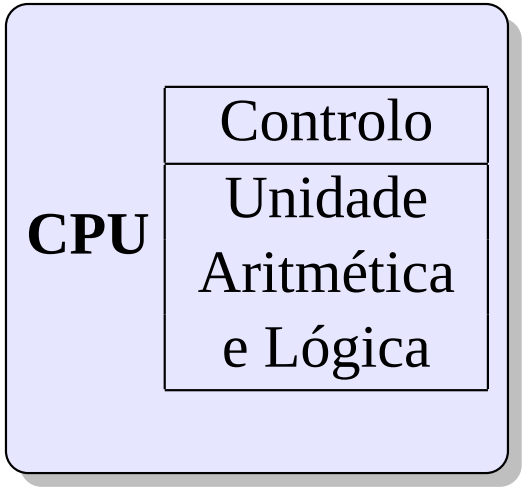


Exemplos:

- Computadores pessoais, telemóveis? Von Neumann
- Microcontroladores, p. ex. arduino? Harvard
- Primeiros computadores, calculadoras? Não é “stored program”

Von Neumann - Ciclo de uma instrução



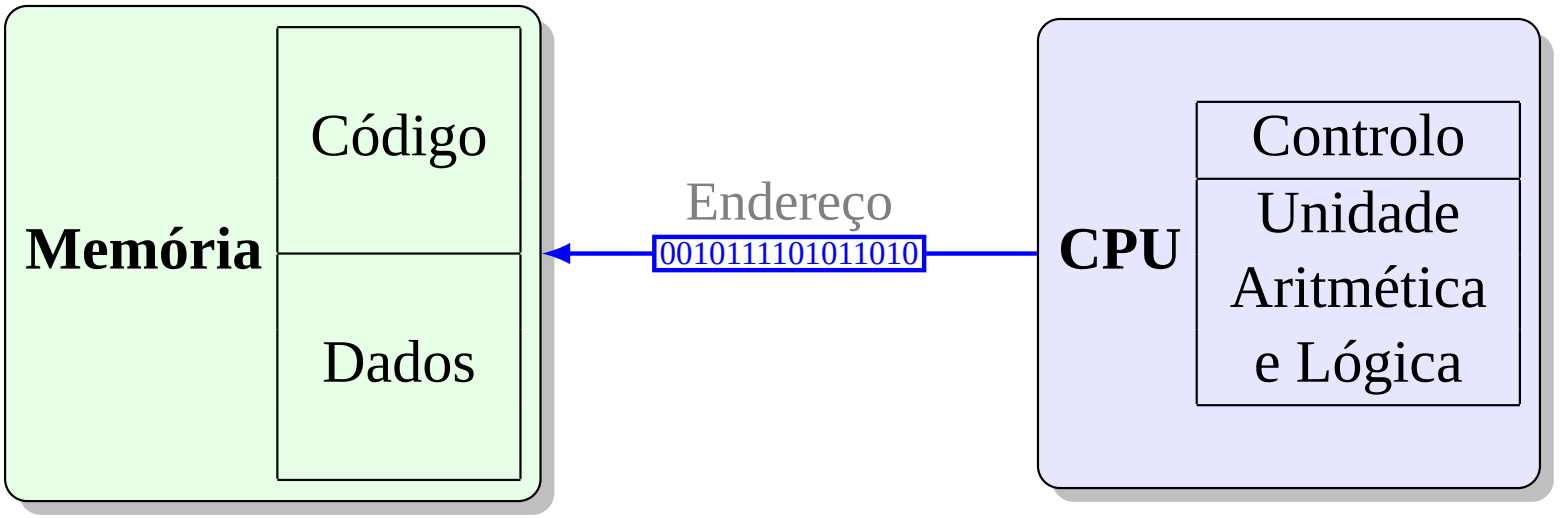


Fetch o processador pede à memória a próxima instrução a executar. A instrução vem codificada num número binário chamado código máquina.

Decode o processador descodifica o código máquina para se preparar para a execução.

Execute o processador executa a instrução. Pode ler ou escrever dados na memória.

Von Neumann - Ciclo de uma instrução

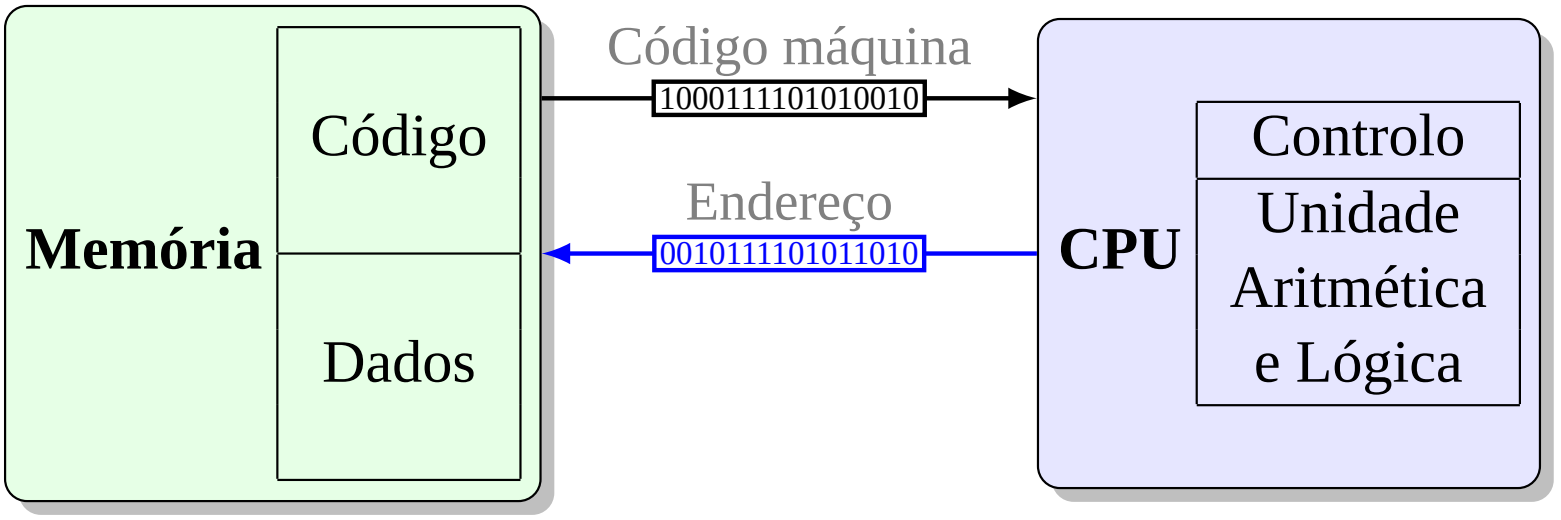


Fetch o processador pede à memória a próxima instrução a executar. A instrução vem codificada num número binário chamado código máquina.

Decode o processador descodifica o código máquina para se preparar para a execução.

Execute o processador executa a instrução. Pode ler ou escrever dados na memória.

Von Neumann - Ciclo de uma instrução

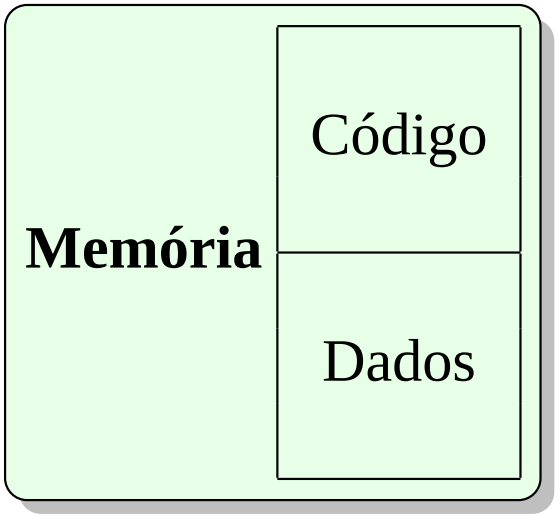


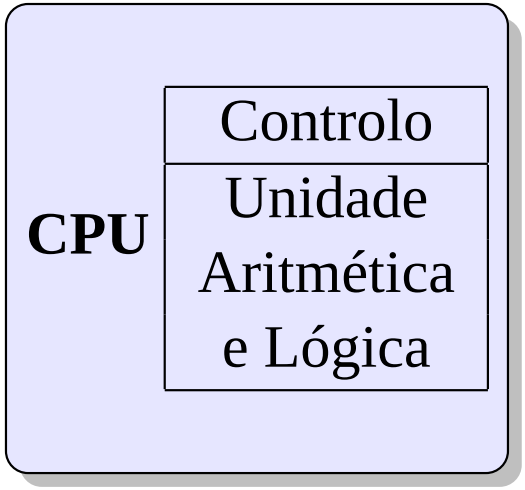
Fetch o processador pede à memória a próxima instrução a executar. A instrução vem codificada num número binário chamado código máquina.

Decode o processador descodifica o código máquina para se preparar para a execução.

Execute o processador executa a instrução. Pode ler ou escrever dados na memória.

Von Neumann - Ciclo de uma instrução



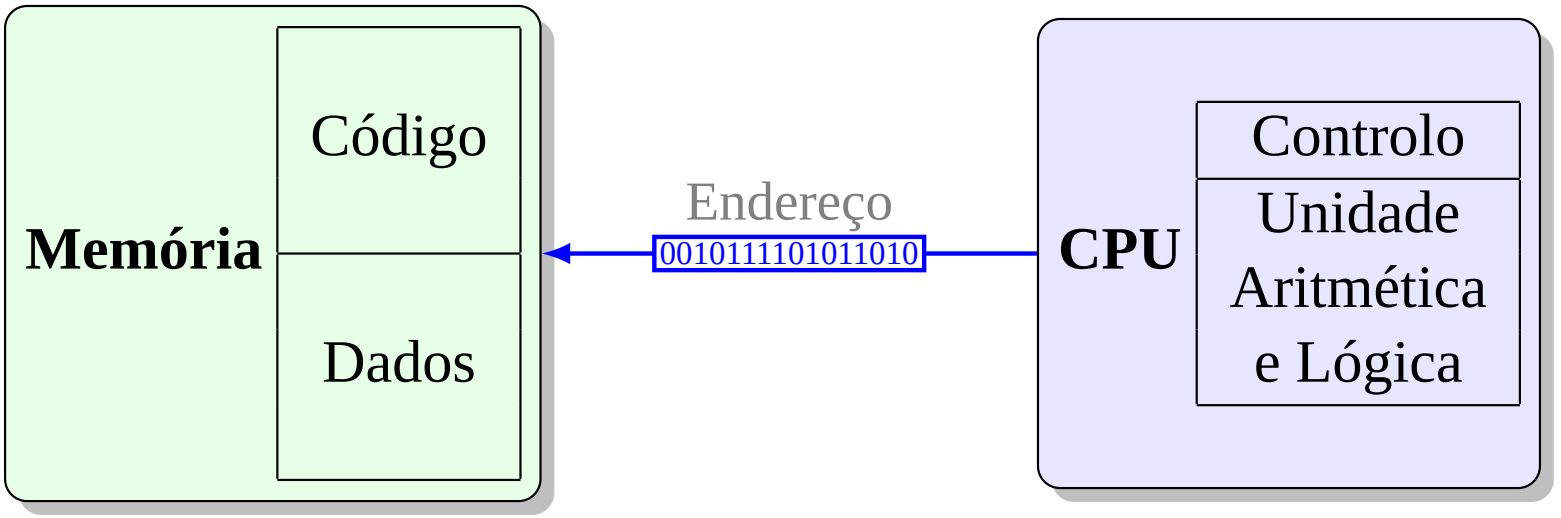


Fetch o processador pede à memória a próxima instrução a executar. A instrução vem codificada num número binário chamado código máquina.

Decode o processador descodifica o código máquina para se preparar para a execução.

Execute o processador executa a instrução. Pode ler ou escrever dados na memória.

Von Neumann - Ciclo de uma instrução

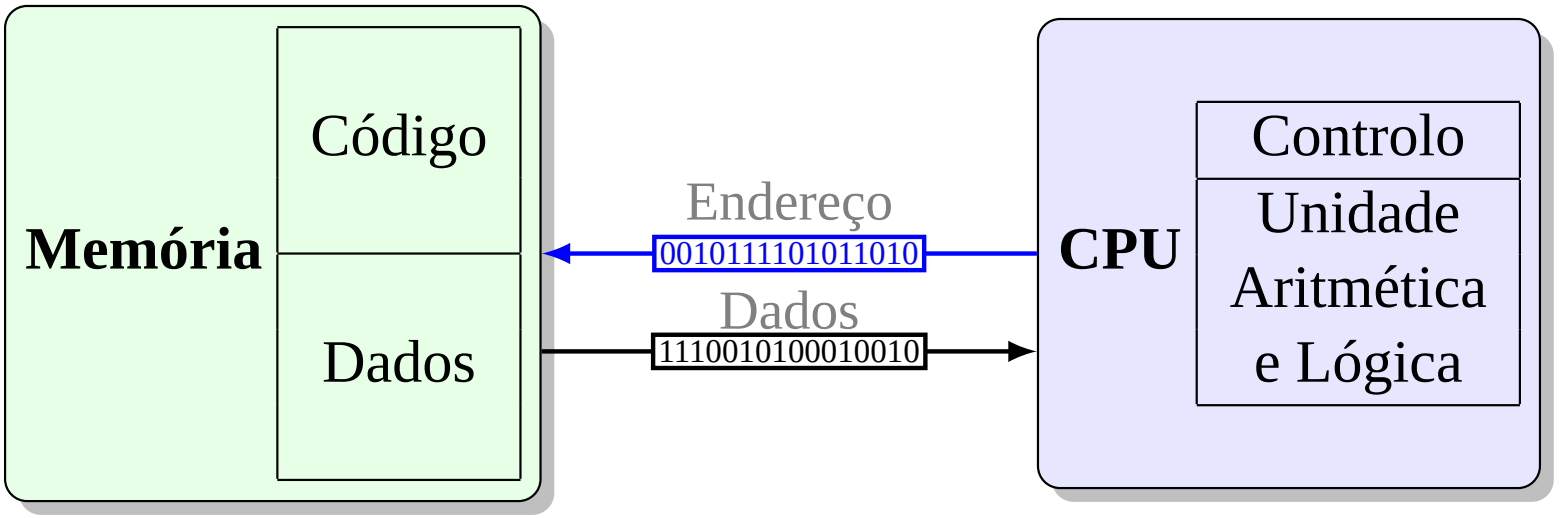


Fetch o processador pede à memória a próxima instrução a executar. A instrução vem codificada num número binário chamado código máquina.

Decode o processador descodifica o código máquina para se preparar para a execução.

Execute o processador executa a instrução. Pode ler ou escrever dados na memória.

Von Neumann - Ciclo de uma instrução

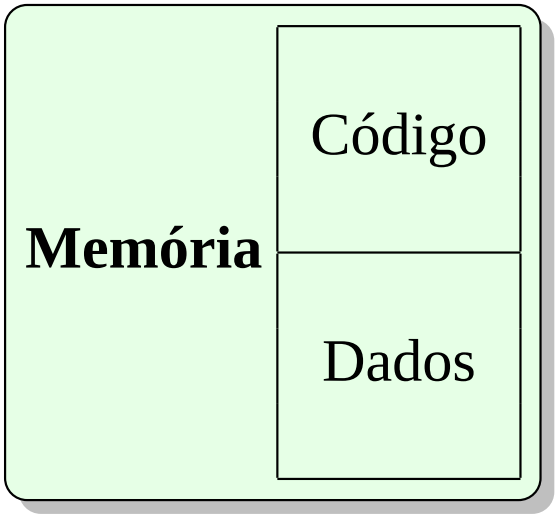


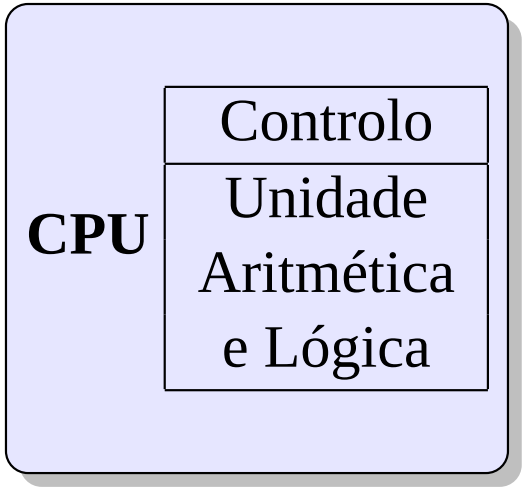
Fetch o processador pede à memória a próxima instrução a executar. A instrução vem codificada num número binário chamado código máquina.

Decode o processador descodifica o código máquina para se preparar para a execução.

Execute o processador executa a instrução. Pode ler ou escrever dados na memória.

Von Neumann - Ciclo de uma instrução



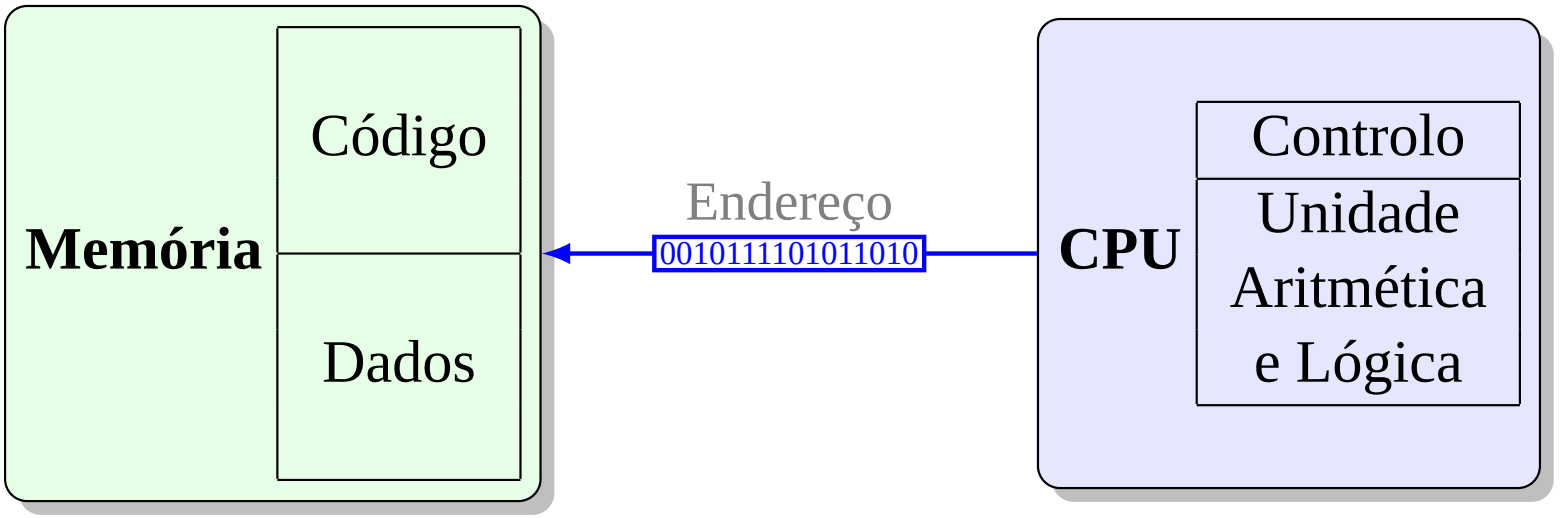


Fetch o processador pede à memória a próxima instrução a executar. A instrução vem codificada num número binário chamado código máquina.

Decode o processador descodifica o código máquina para se preparar para a execução.

Execute o processador executa a instrução. Pode ler ou escrever dados na memória.

Von Neumann - Ciclo de uma instrução

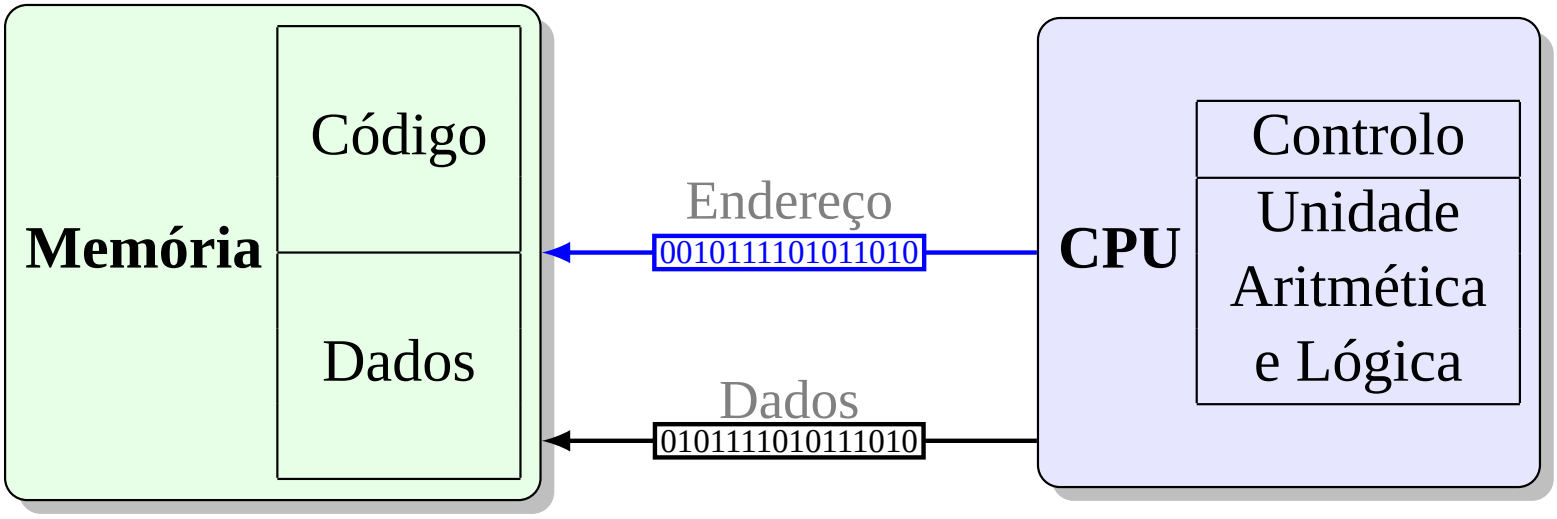


Fetch o processador pede à memória a próxima instrução a executar. A instrução vem codificada num número binário chamado código máquina.

Decode o processador descodifica o código máquina para se preparar para a execução.

Execute o processador executa a instrução. Pode ler ou escrever dados na memória.

Von Neumann - Ciclo de uma instrução

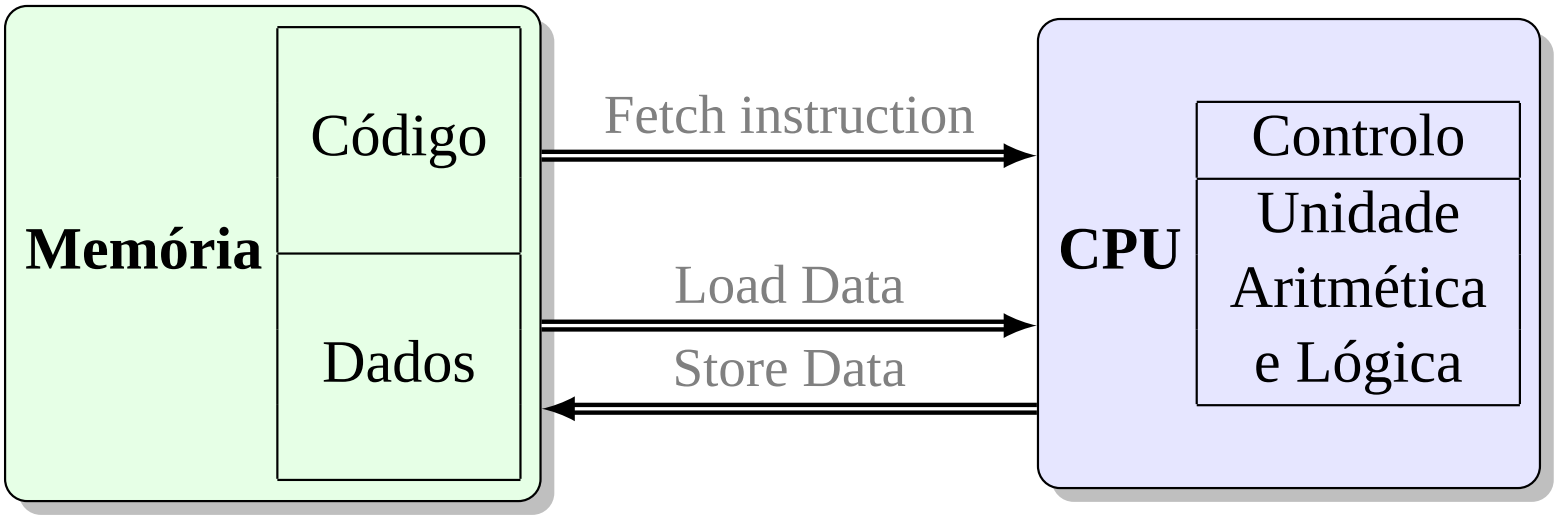


Fetch o processador pede à memória a próxima instrução a executar. A instrução vem codificada num número binário chamado código máquina.

Decode o processador descodifica o código máquina para se preparar para a execução.

Execute o processador executa a instrução. Pode ler ou escrever dados na memória.

Von Neumann - Ciclo de uma instrução



Fetch o processador pede à memória a próxima instrução a executar. A instrução vem codificada num número binário chamado código máquina.

Decode o processador descodifica o código máquina para se preparar para a execução.

Execute o processador executa a instrução. Pode ler ou escrever dados na memória.

Exemplo de execução de um programa

Endereço	Instruções e dados
. . .	. . .
100	17
101	5
102	214
. . .	. . .
201	Ler do endereço 100
202	Ler do endereço 101
203	Somar registos
204	Guardar no endereço 102
. . .	. . .

Memória

Instrução: “?”

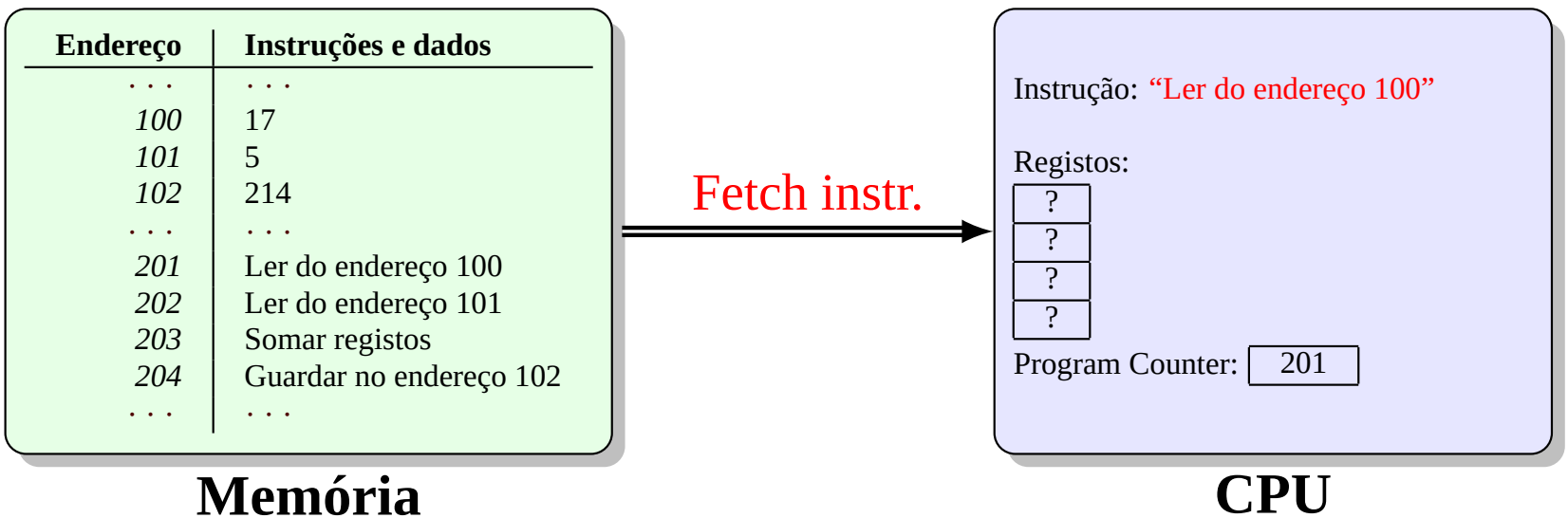
Registos:

?
?
?
?

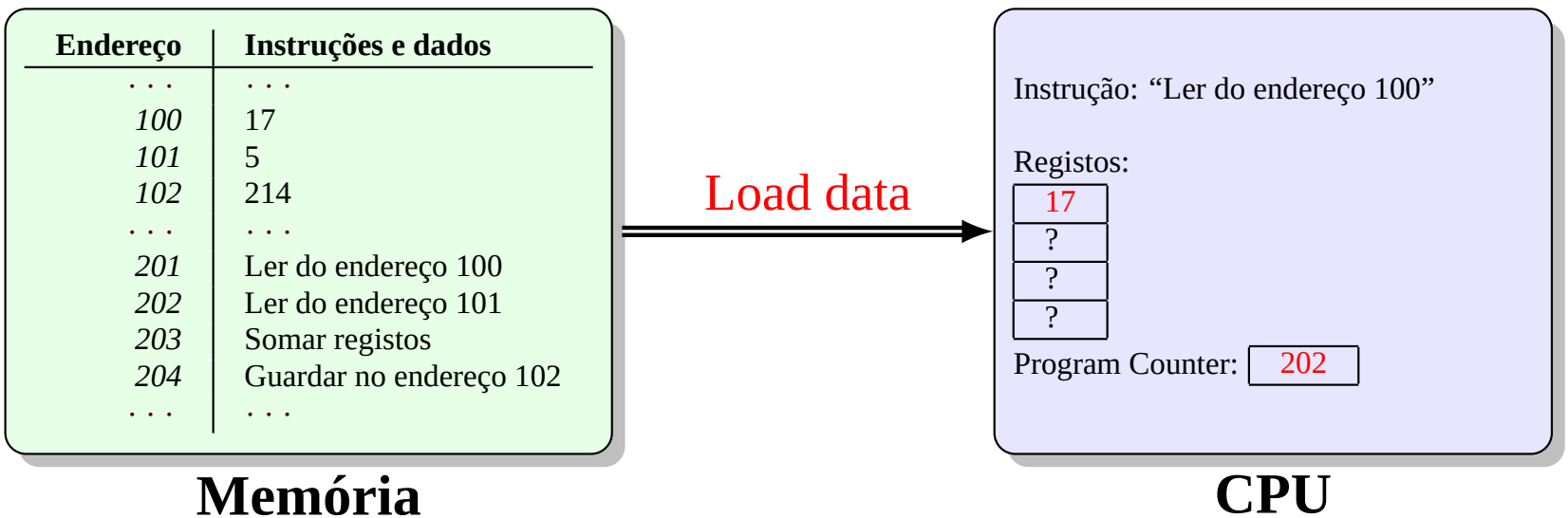
Program Counter: 201

CPU

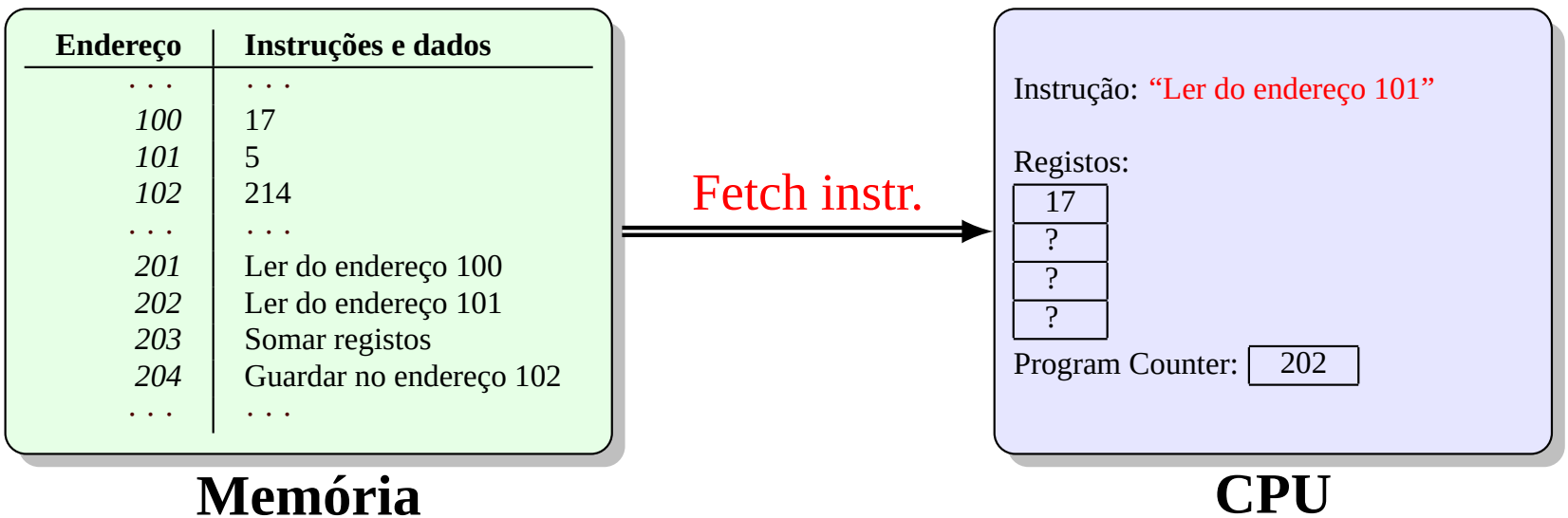
Exemplo de execução de um programa



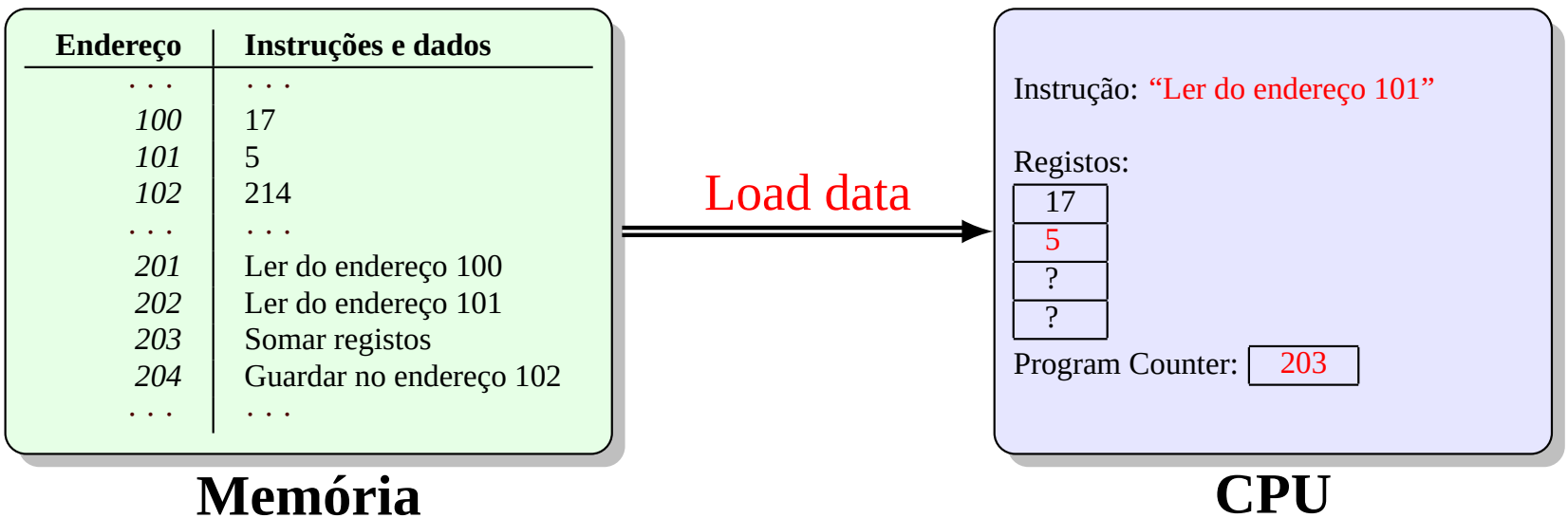
Exemplo de execução de um programa



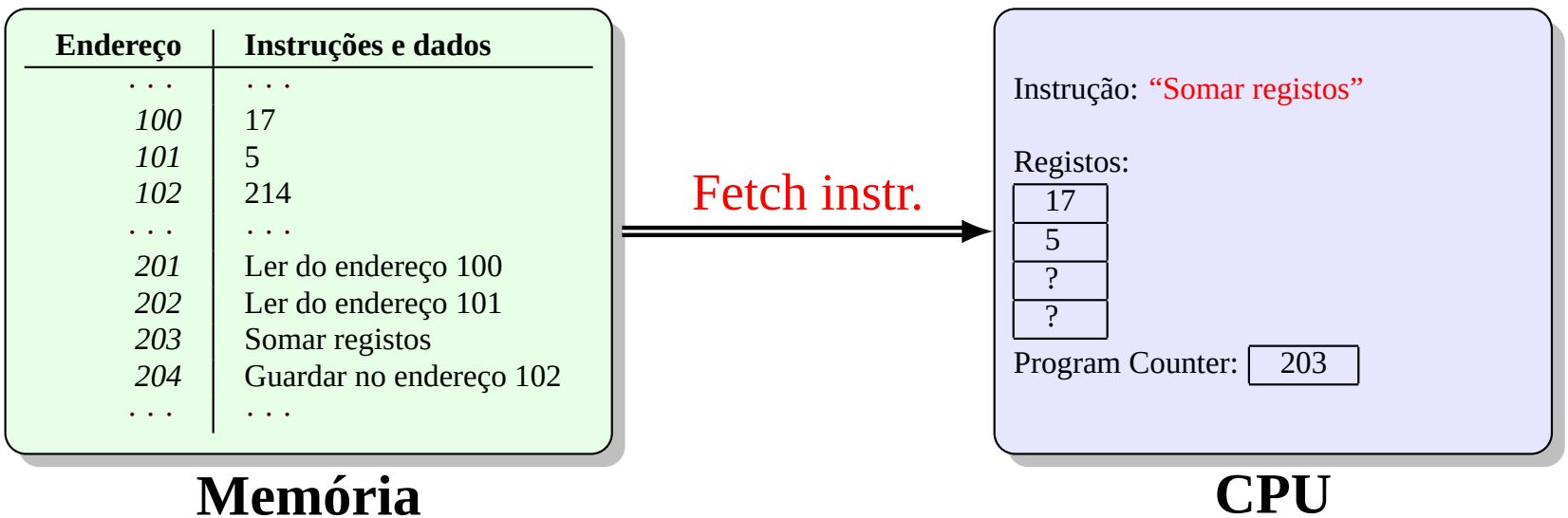
Exemplo de execução de um programa



Exemplo de execução de um programa



Exemplo de execução de um programa



Exemplo de execução de um programa

Endereço	Instruções e dados
. . .	. . .
100	17
101	5
102	214
. . .	. . .
201	Ler do endereço 100
202	Ler do endereço 101
203	Somar registos
204	Guardar no endereço 102
. . .	. . .

Memória

Instrução: “Somar registos”

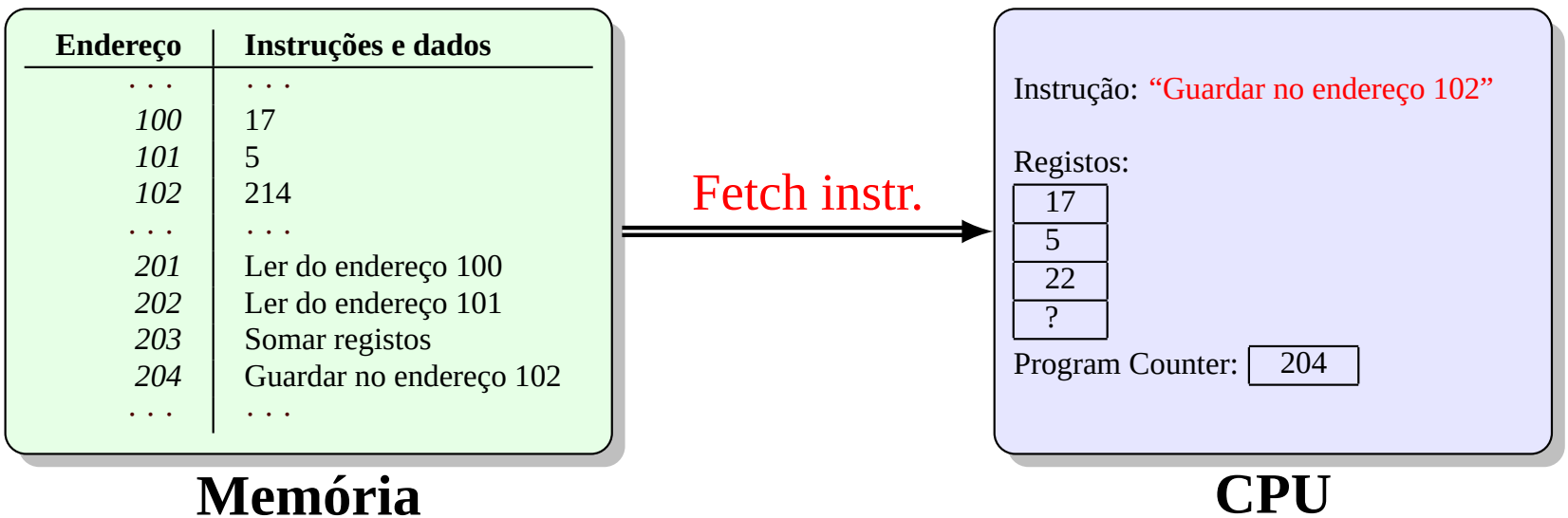
Registos:

17
5
22
?

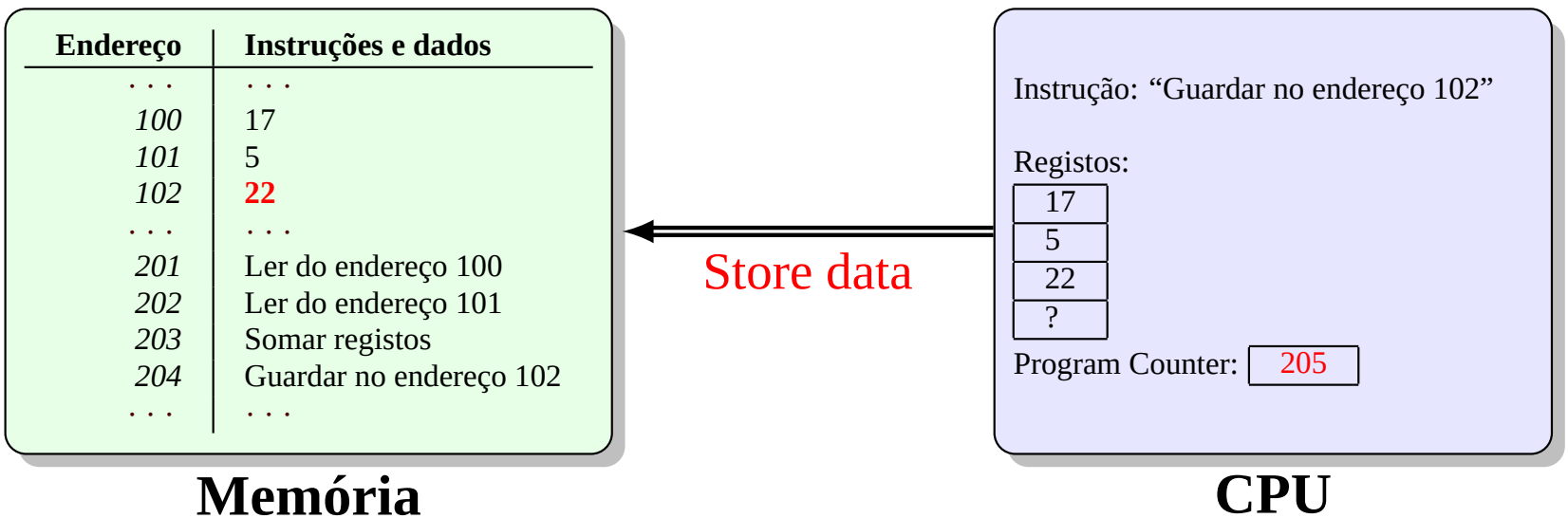
Program Counter: 204

CPU

Exemplo de execução de um programa

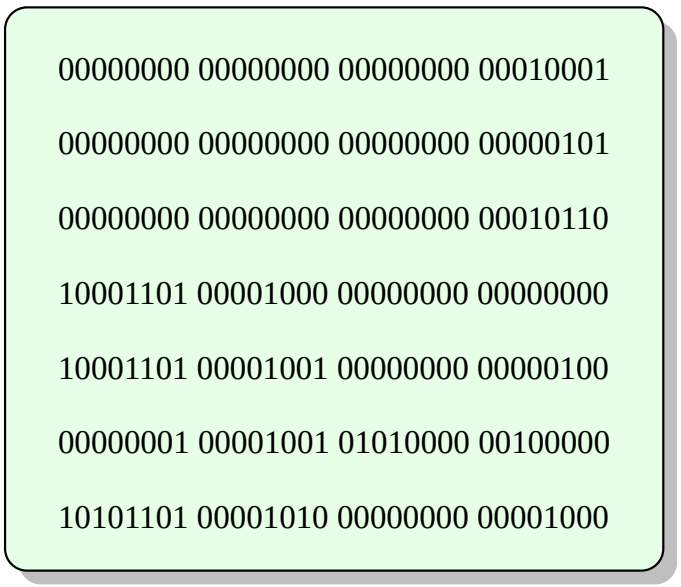


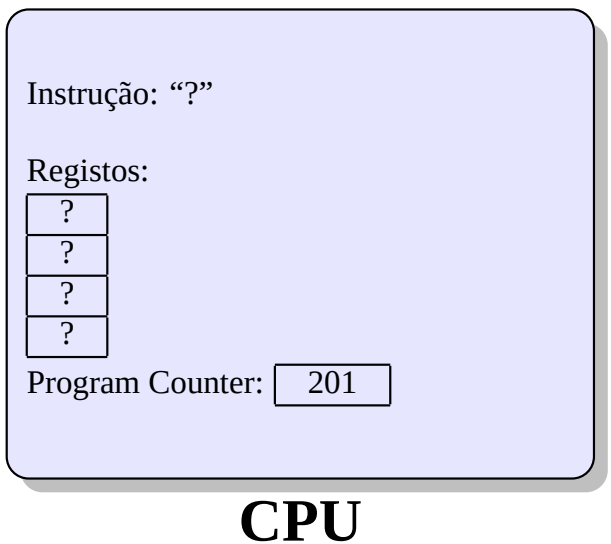
Exemplo de execução de um programa



Exemplo de execução de um programa

Um programa é um conjunto de números em memória que codificam as instruções a executar → código máquina.
O conteúdo da memória está em binário:



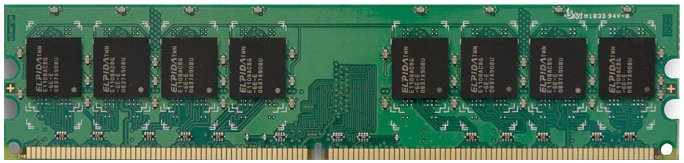


Na memória não há distinção entre dados e instruções!
São apenas números em binário.

Tipos de memória

SAM (Sequential Access Memory) apenas permite ler/escrever em sequência.

RAM (Random Access Memory) Permite acesso a qualquer posição de memória em qualquer momento.



ROM (Read Only Memory) só permitem leitura, o conteúdo é definido de fábrica. As EPROM e EEPROM podem ser “reprogramadas”, mas em operação normal são apenas de leitura.