Organização de Computadores Hardware

Professor Marcus Vinícius Midena Ramos

Colegiado de Engenharia de Computação

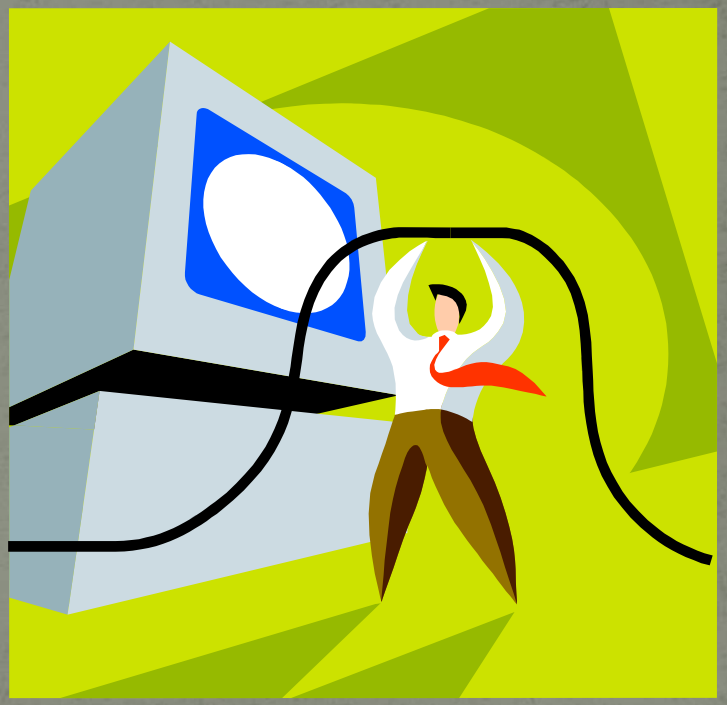
(74)3614.1936

marcus.ramos@univasf.edu.br

www.univasf.edu.br/~marcus.ramos

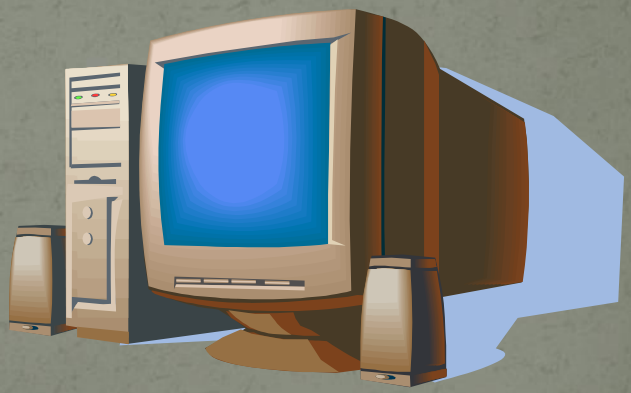
Computador

- Ferramenta indispensável;
- Faz parte das nossas vidas;
- Por si só não faz nada de útil;
- Grande capacidade de resolução de problemas;
- Necessita ser instruído.



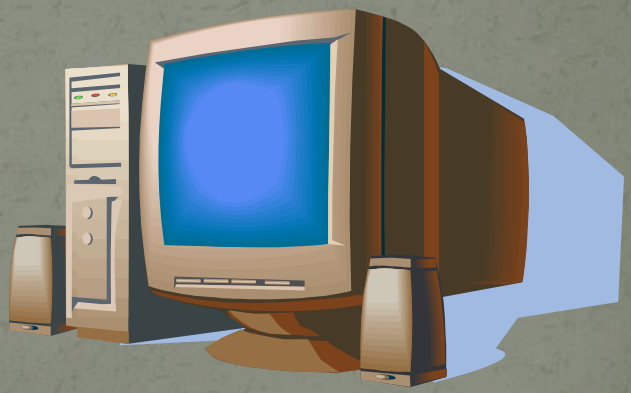
Computador

- Capaz apenas de excutar poucas tarefas básicas distintas, todas muito simples;
- É extremamente rápido;
- Possui um comportamento previsível;
- É excelente para reproduzir “roteiros” pré-concebidos;
- Não se cansa e pode ser usado à exaustão.



Computador = Hardware *(corpo)* + Software *(alma)*

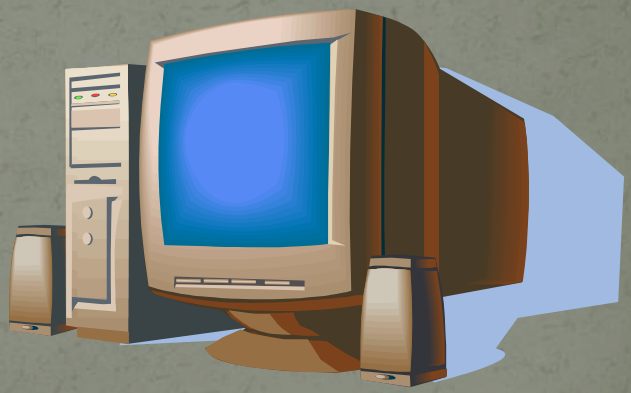
- ✓ O corpo fornece suporte para a alma.
- ✓ O corpo procura suprir as necessidades da alma.
- ✓ O corpo pode criar novas possibilidades para a alma, ou então estabeldcer limitações.
- ✓ A alma se expressa através do corpo.
- ✓ A alma usa os recursos do corpo.



Computador

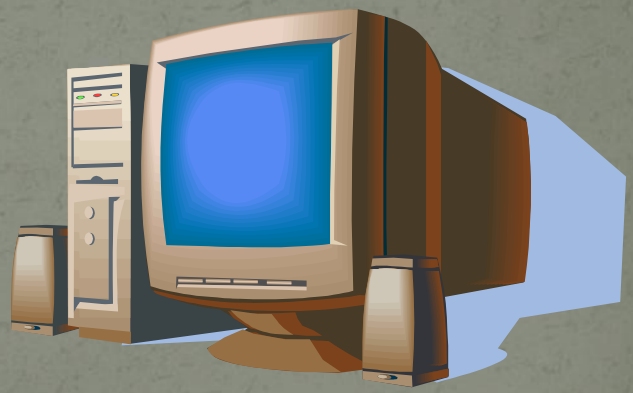
- Hardware:

  - ✓ Parte física: placas, periféricos, circuitos, cabos e componentes.
  - ✓ Quanto mais usado, mais propenso à falhas.
  - ✓ Sozinho, não serve para nada.
  - ✓ Vem pronto da fábrica.



Computador

- Software:
  - ✓ Parte intangível: conhecimentos e idéias que fazem o hardware exibir um certo comportamento.
  - ✓ Quanto mais usado, menos propenso à falhas.
  - ✓ Confere funcionalidade ao hardware.
  - ✓ Pode ser adquirido ou desenvolvido.

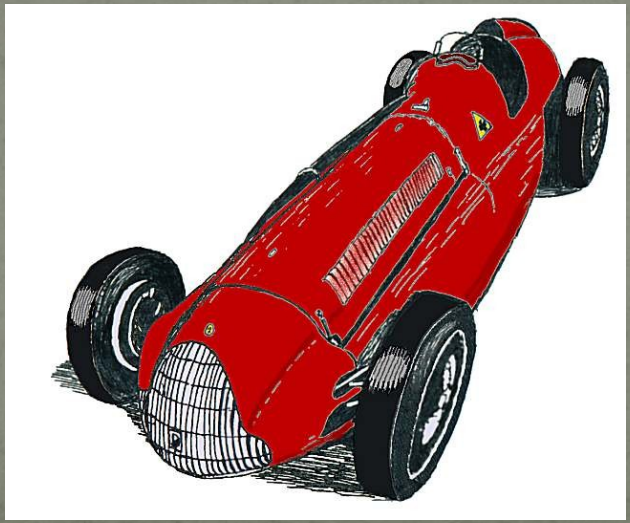


Modelo de von Neumann

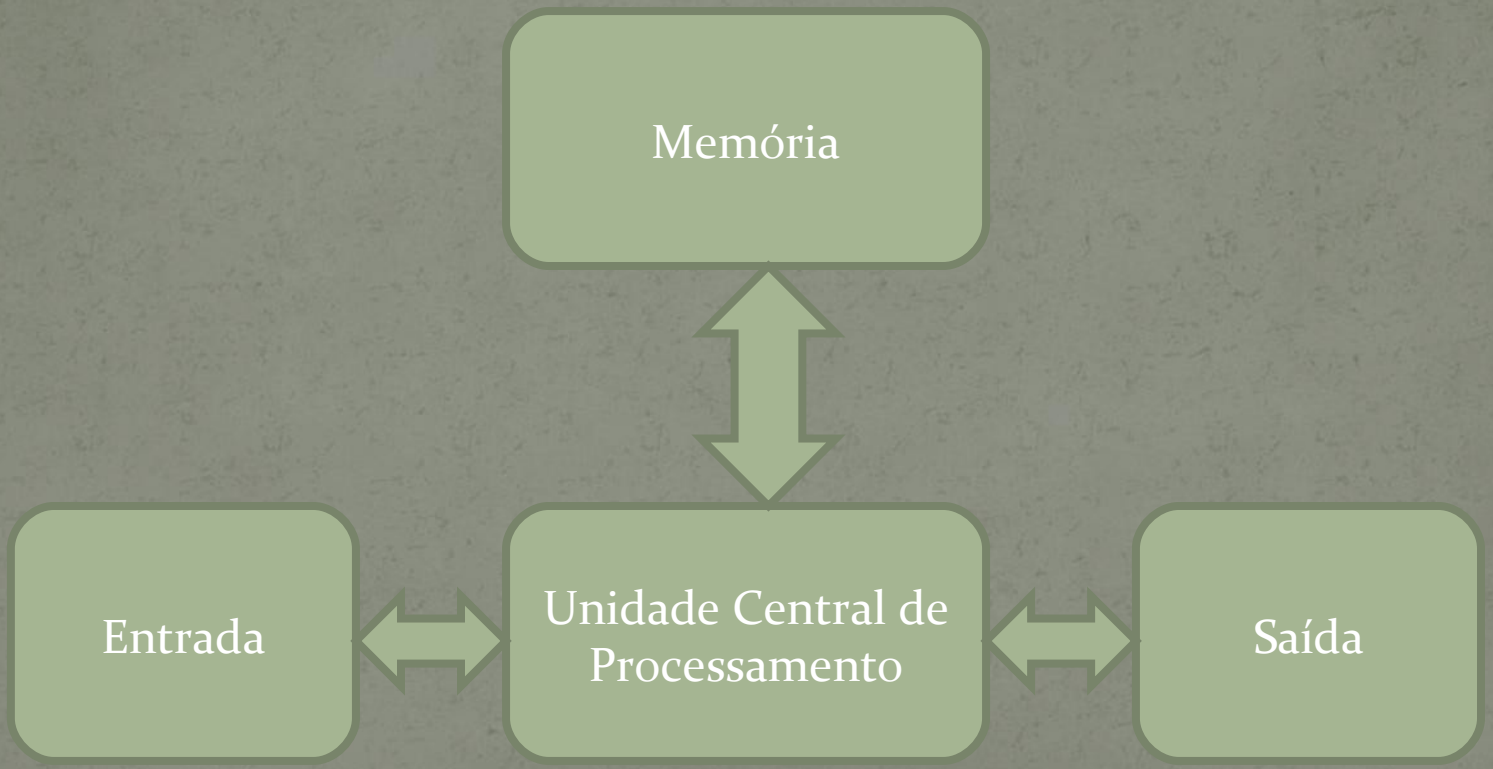
- Matemático húngaro;
- Propôs um modelo de arquitetura composto por quatro partes distintas;
- Nesse modelo, o programa a ser executado pelo computador ficava armazenado na memória do mesmo;
- Grande flexibilidade e rapidez, pois o hardware não precisa ser modificado e não precisava ser lido em cartões perfurados;
- A maioria dos computadores modernos adota essa arquitetura.

Modelo de von Neumann

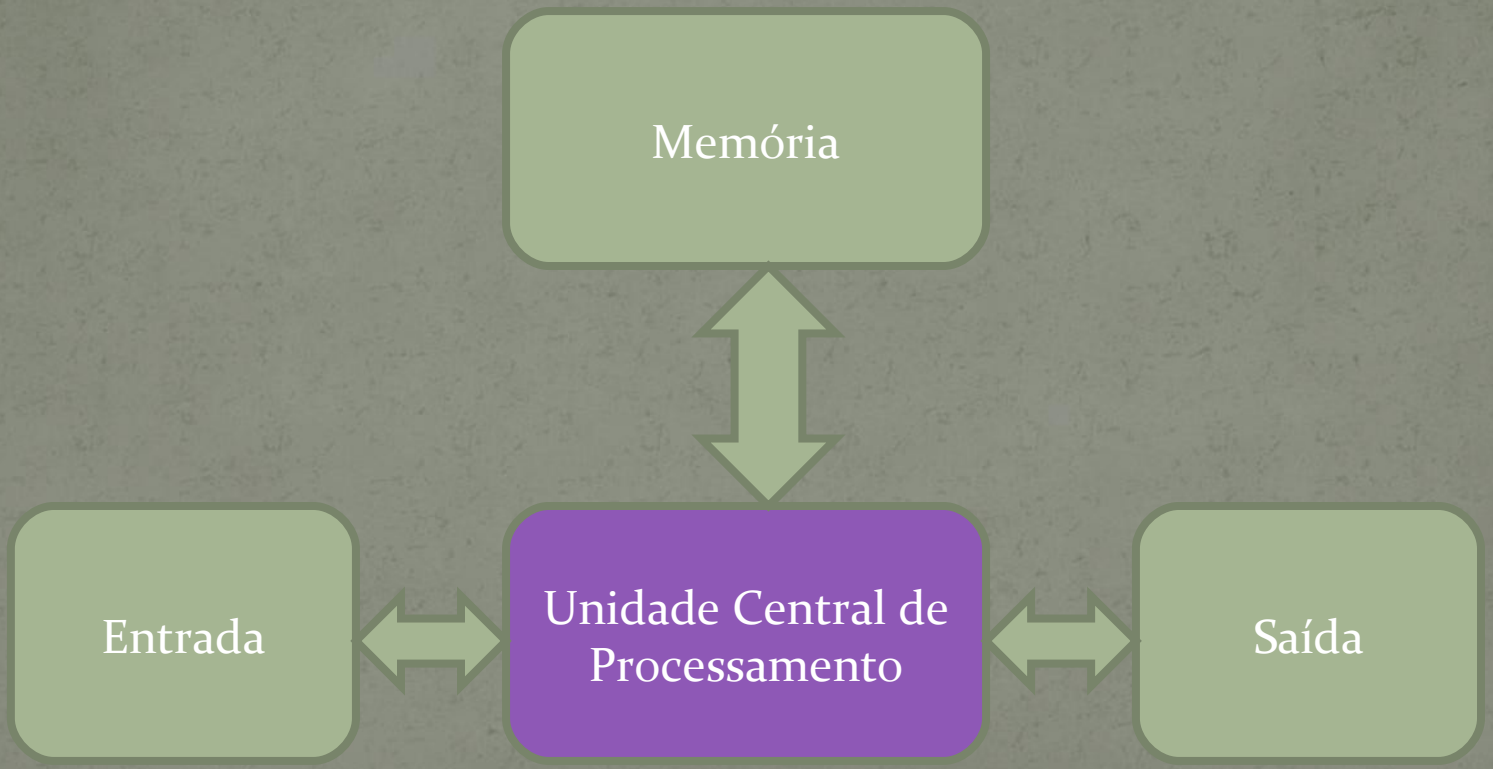
- Ciência x Tecnologia
- Evolução científica x tecnológica
- Exemplo do automóvel



Modelo de von Neumann



Modelo de von Neumann

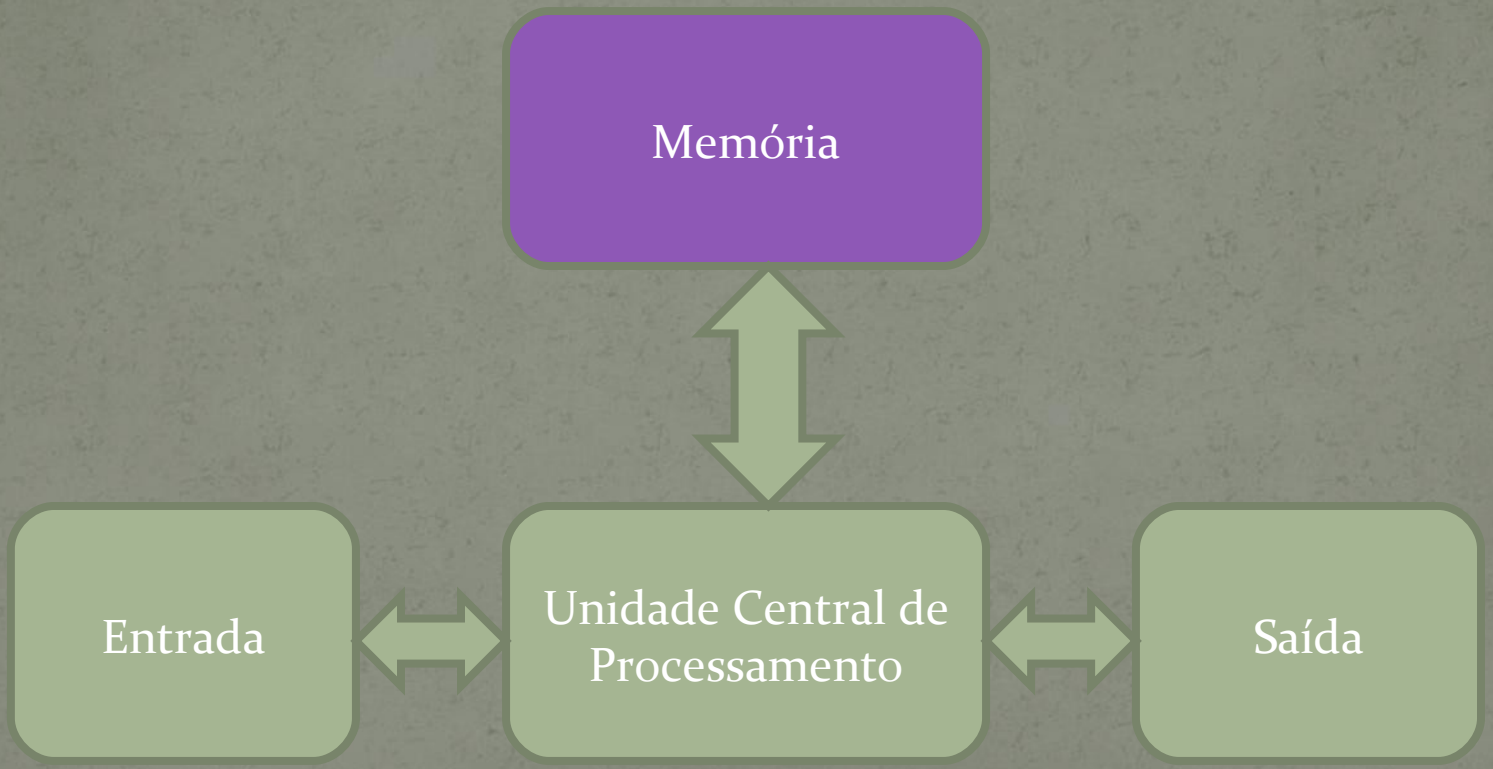


Modelo de von Neumann

Unidade Central de Processamento (UCP ou CPU):

- Lê, decodifica e executa as instruções armazenadas na memória;
- Realiza operações lógicas e aritméticas;
- Pode ler modificar o valor de alguma posição de memória;
- Pode transferir valores da entrada para a memória e da memória para a saída;
- Executa as instruções seqüencialmente, até o término do programa ou até encontrar algum desvio explícito.

Modelo de von Neumann

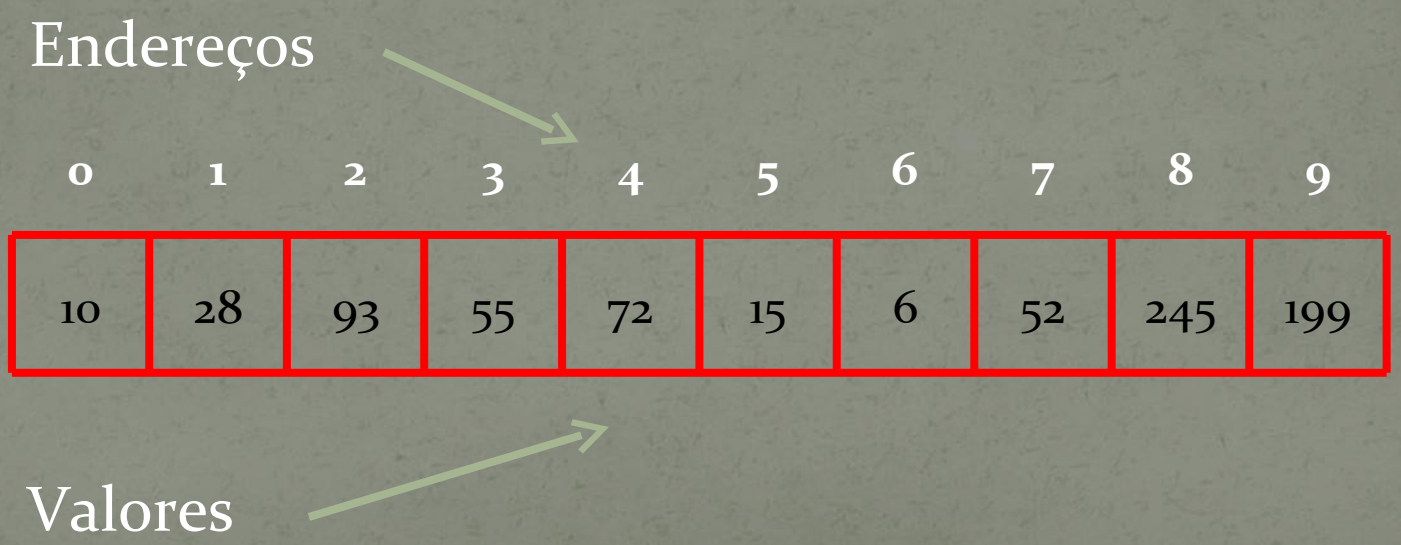


Modelo de von Neumann

Memória:

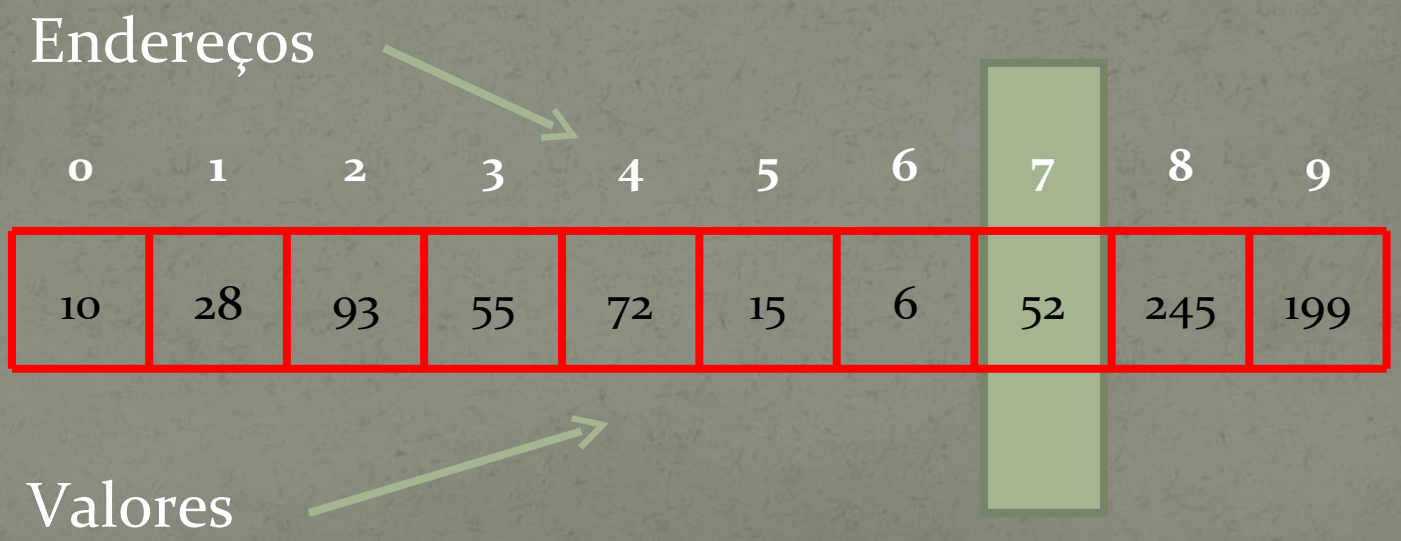
- É dividida em células (bytes)
- Cada célula armazena um valor e possui um endereço;
- É comandada pela CPU, que informa em qual endereço deseja executar uma operação (de leitura ou escrita);
- Operações de leitura não modificam os valores armazenados;
- Operações de escrita apagam os valores anteriores, substituindo-os pelos novos valores;

- Exemplo de memória com 10 posições;
- As posições são numeradas 0 a 9;
- Valores permanecem até que um novo valor seja armazenado na mesma posição ou até que a memória seja desligada.



Operação de leitura

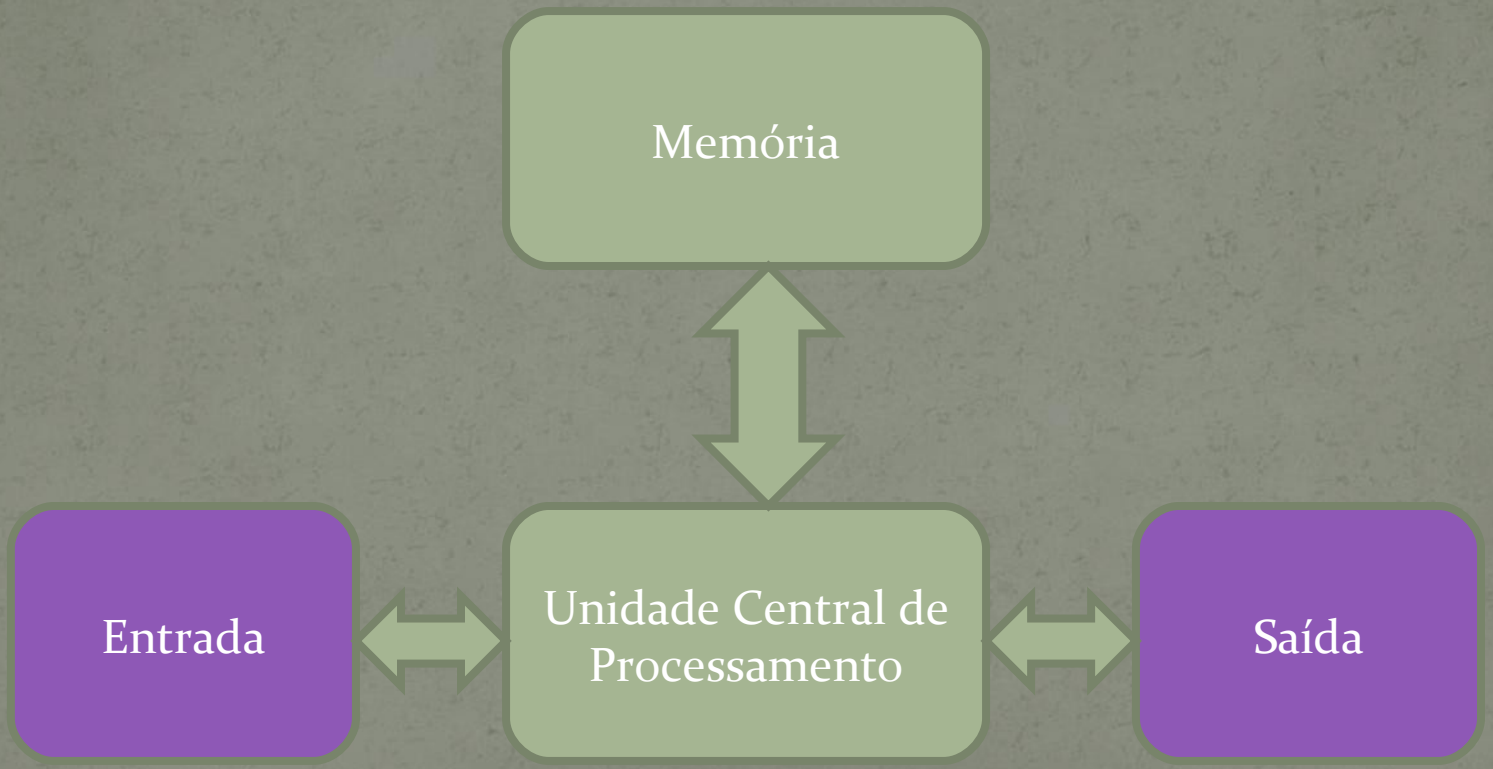
- Obter o valor armazenado na posição (endereço) 7
- Resultado: 52



Operação de escrita

- Armazenar o valor 31 na posição (endereço) 7

Modelo de von Neumann



Modelo de von Neumann

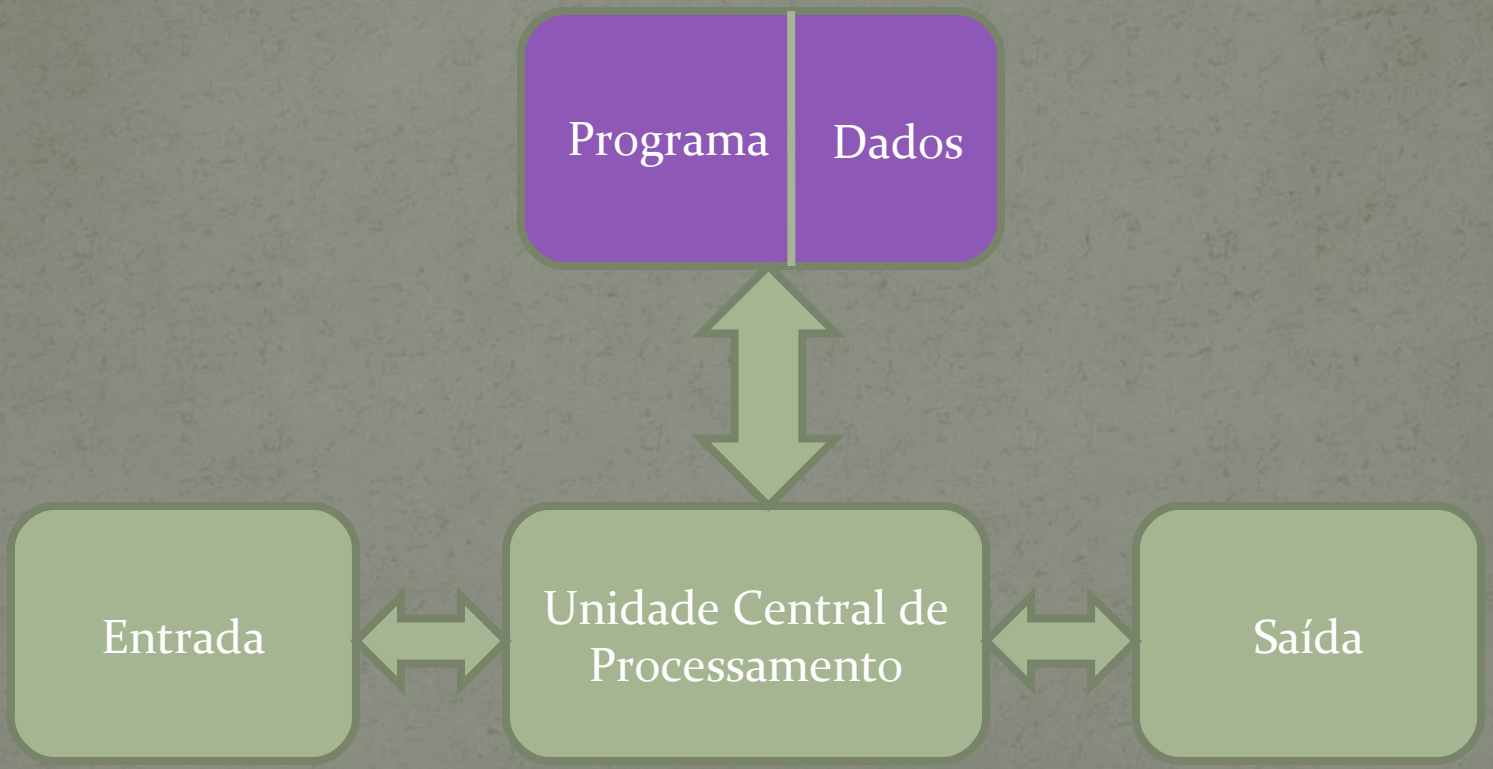
Entrada e Saída:

- Sob comando da CPU, é possível ler valores de um dispositivo de entrada, armazenando-os na memória;
- Sob comando da CPU, é possível ler valores da memória, enviando-os para um dispositivo de saída;
- Existe uma grande variedade de dispositivos de entrada e saída;
- Conceitualmente, no entanto, eles são passivos e obedecem ao comando da CPU.

Modelo de von Neumann

- Os dados e os programas são armazenados na memória, em regiões distintas;
- Um programa é composto por uma coleção de instruções que são lidas e executadas em seqüência.

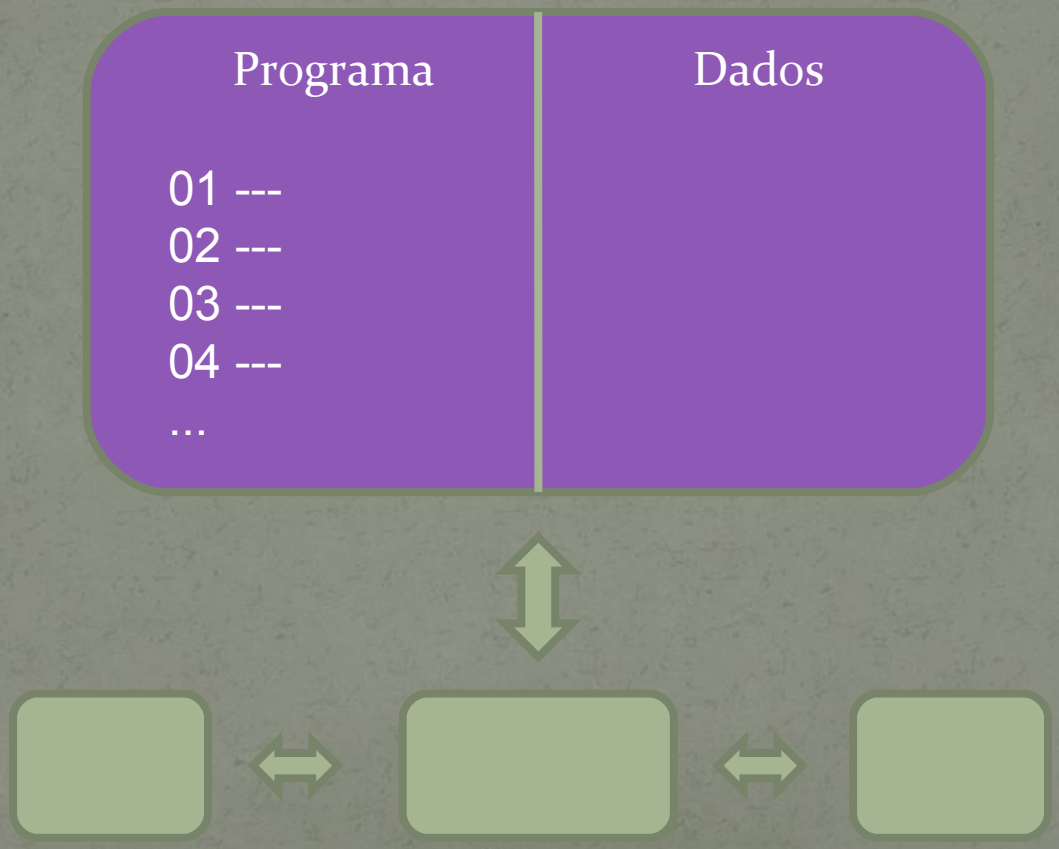
Modelo de von Neumann



Modelo de von Neumann

- Os programas são formados, essencialmente, por comandos (instruções sobre o que fazer);
- As instruções são muito simples e executadas muito rapidamente;

Modelo de von Neumann



Modelo de von Neumann

- As instruções são lidas seqüencialmente da memória, uma após a outra;
- Uma instrução pode (i) ler um valor da entrada; (ii) enviar um valor para a saída; (iii) gerar um novo valor; (iv) indicar o endereço da próxima instrução a ser executada.

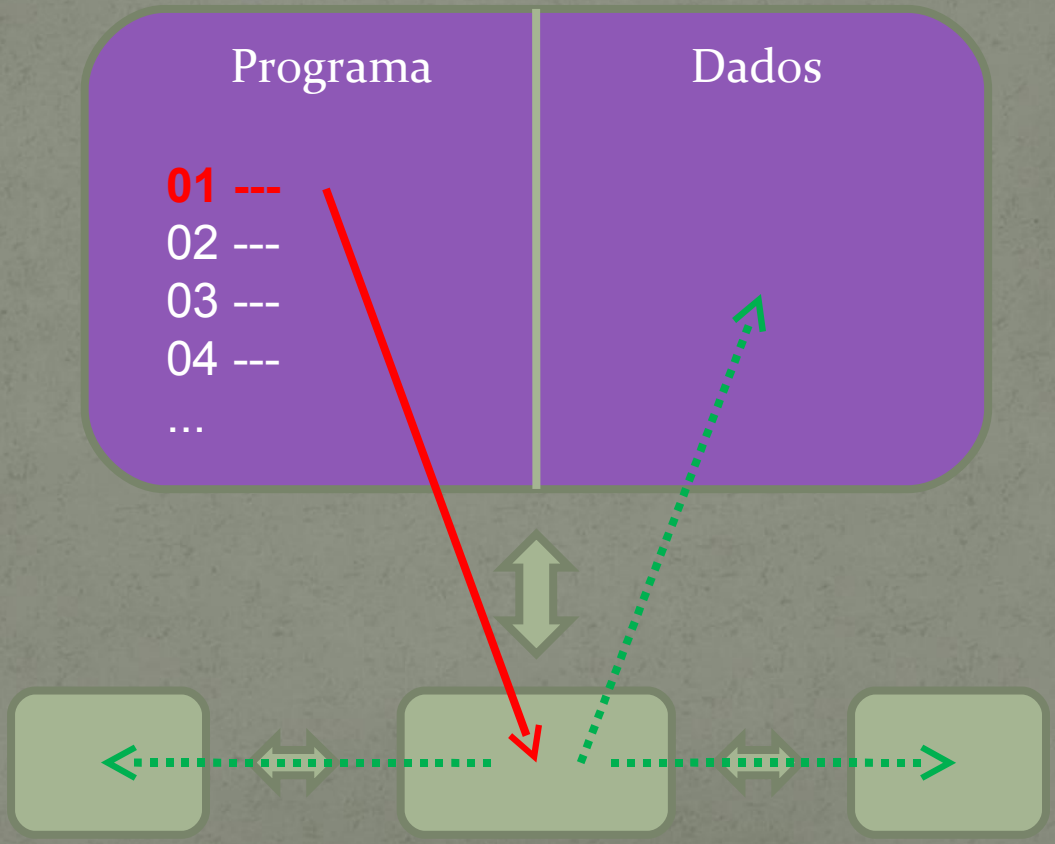
Modelo de von Neumann

- A execução de um novo comando inicia apenas depois que a execução do anterior tiver terminado (execução seqüencial);
- Se não houver nenhuma indicação explícita, a instrução armazenada no endereço de memória seguinte é executada.

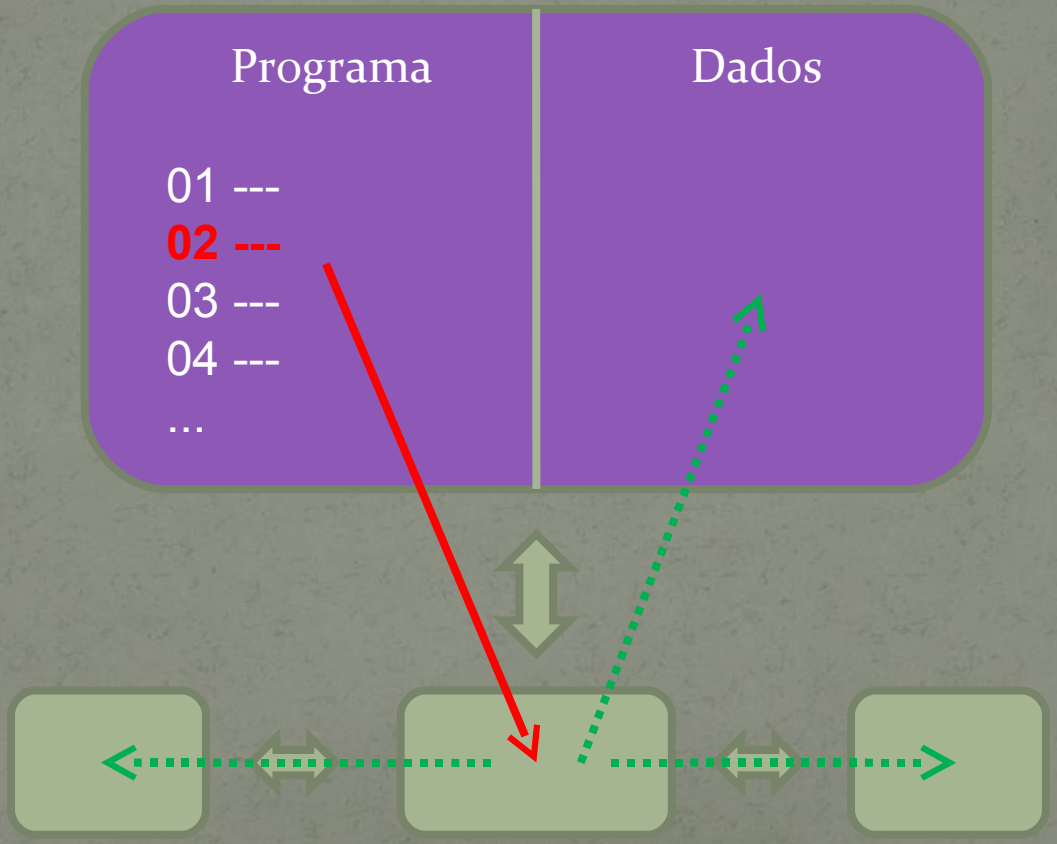
Modelo de von Neumann

- Eventualmente, um comando pode modificar o valor de um dado existente na memória, solicitar novos dados ao usuário ou enviar dados para a saída.

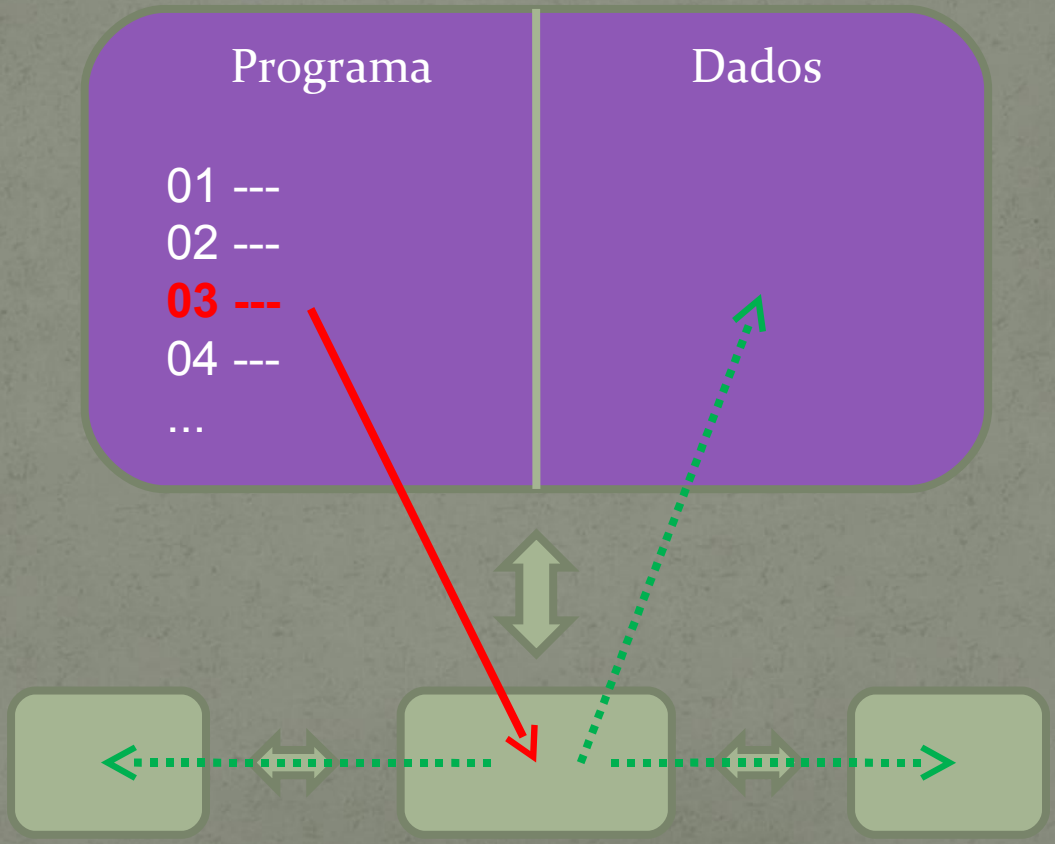
Modelo de von Neumann



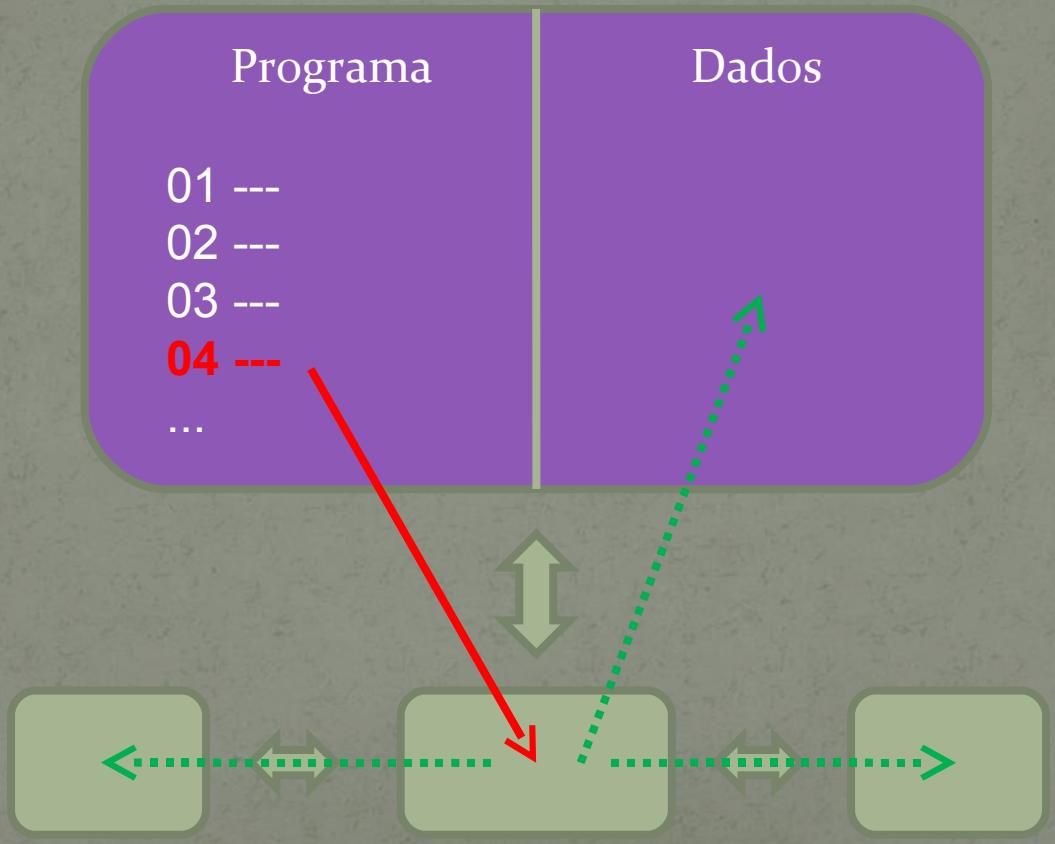
Modelo de von Neumann



Modelo de von Neumann



Modelo de von Neumann



Modelo de von Neumann

Exemplos de evolução tecnológica:

- Velocidade da CPU;
- Quantidade de registradores;
- Quantidade de bits maniupulados simultaneamente;
- Quantidade de posições de memória;
- Velocidade de acesso à memória;
- Tamanho do computador;
- Consumo de energia;
- Preço do computador;
- Possibilidade de expansão do computador;
- Diversificação dos dispositivos de entrada e saída.

Exemplo

Simple Computer Simulator:

- 13 instruções
- 100 posições de memória
- As posições de memória são endereçadas 00 a 99
- Funciona no browser: http://community.vcsu.edu/facultypages/curt.hill/My_Webpage/simulators.htm

Exemplo

Simple Computer Simulator:

- AX: registrador interno à CPU (único);
- PC: contém o endereço da primeira instrução a ser executada;
- Delay: intervalo de tempo entre a execução de instruções consecutivas;
- Run: executa o programa até o fim;
- Single step: executa apenas uma instrução (indicada pelo PC).

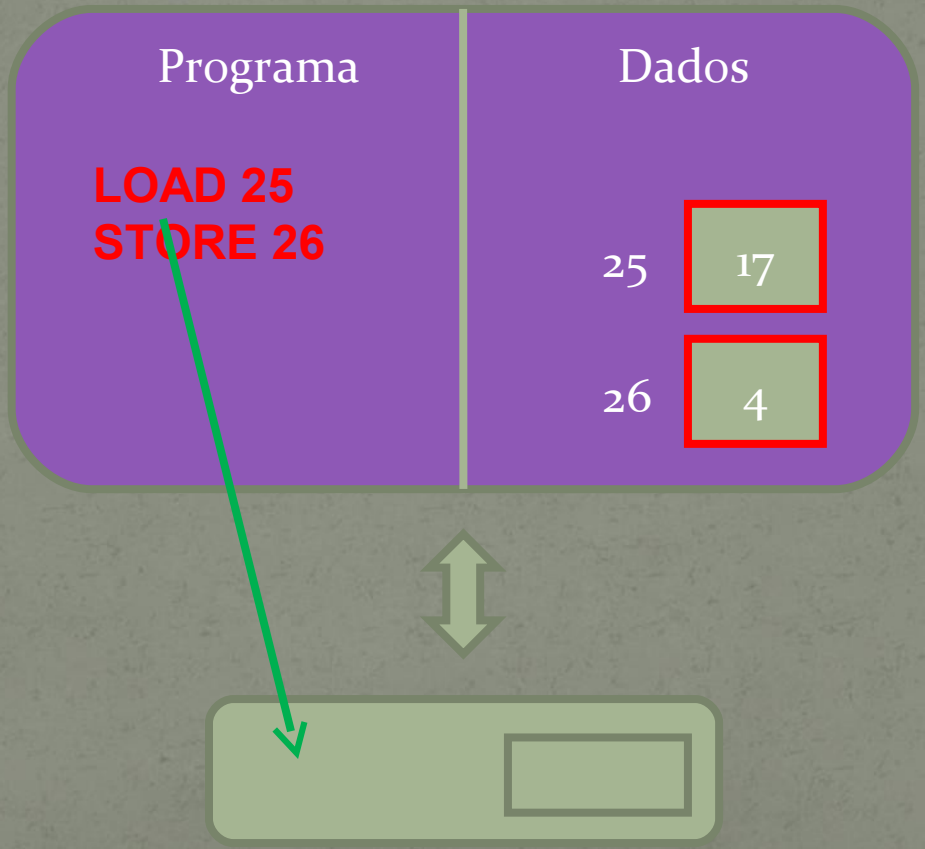
Instruções

- Cada instrução tem um nome;
- Para a máquina, nomes nada significam;
- Nomes precisam ser traduzidos para números;
- Linguagem de Montagem x Linguagem de Máquina;
- Cada instrução tem um número;
- A maioria das instruções tem um complemento (dois números), outras usam apenas um número;

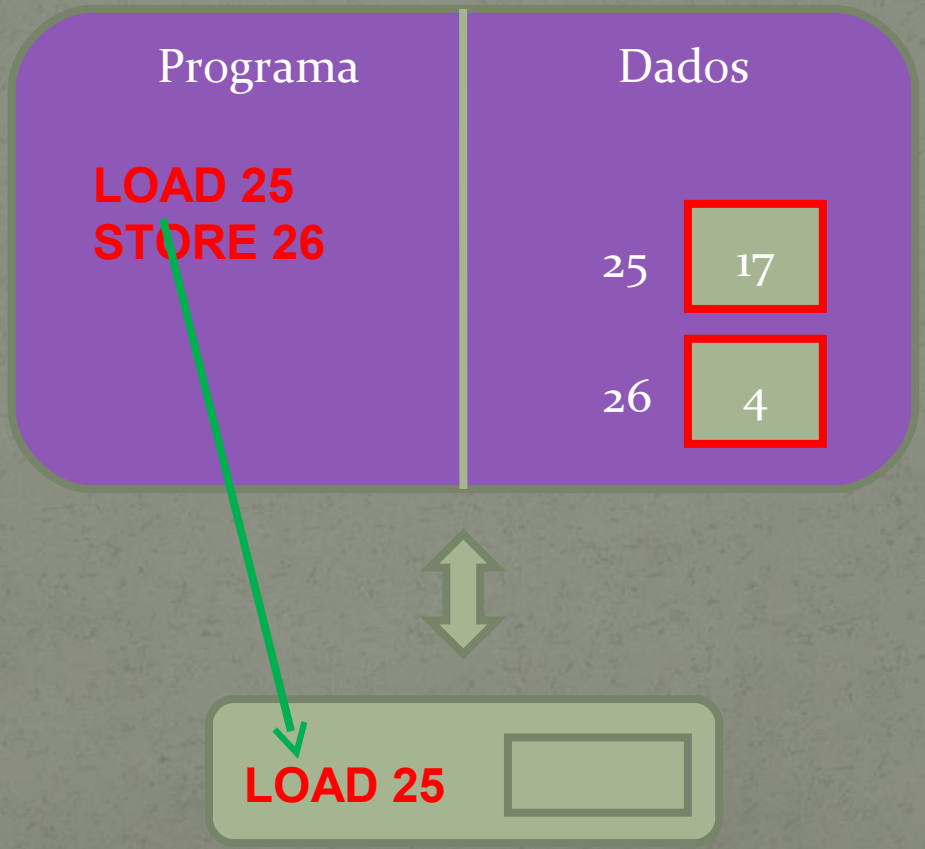
LOAD

- Código 161;
- Usa número adicional;
- Formato: 161 <endereço>
- Copia o valor armazenado na posição de memória <endereço> para o registrador AX;
- O valor anterior de AX é perdido;
- O valor da posição de memória <endereço> permanece inalterado;
- Exemplo:
- 161 25
  ou
  LOAD 25

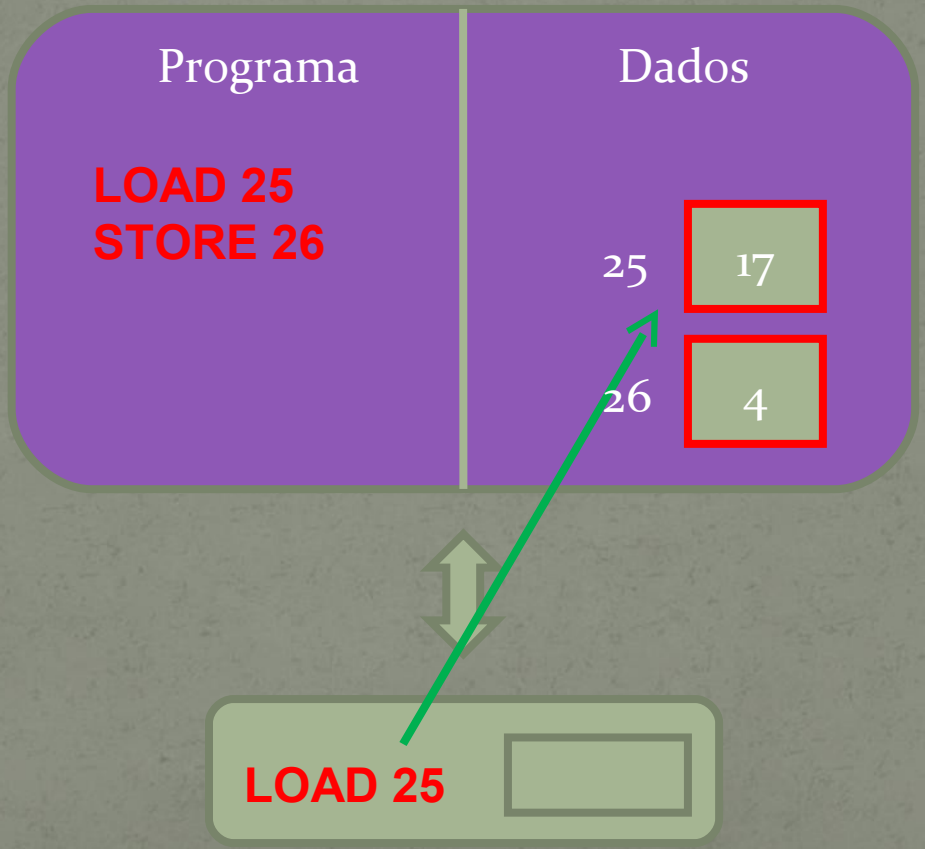
Modelo de von Neumann



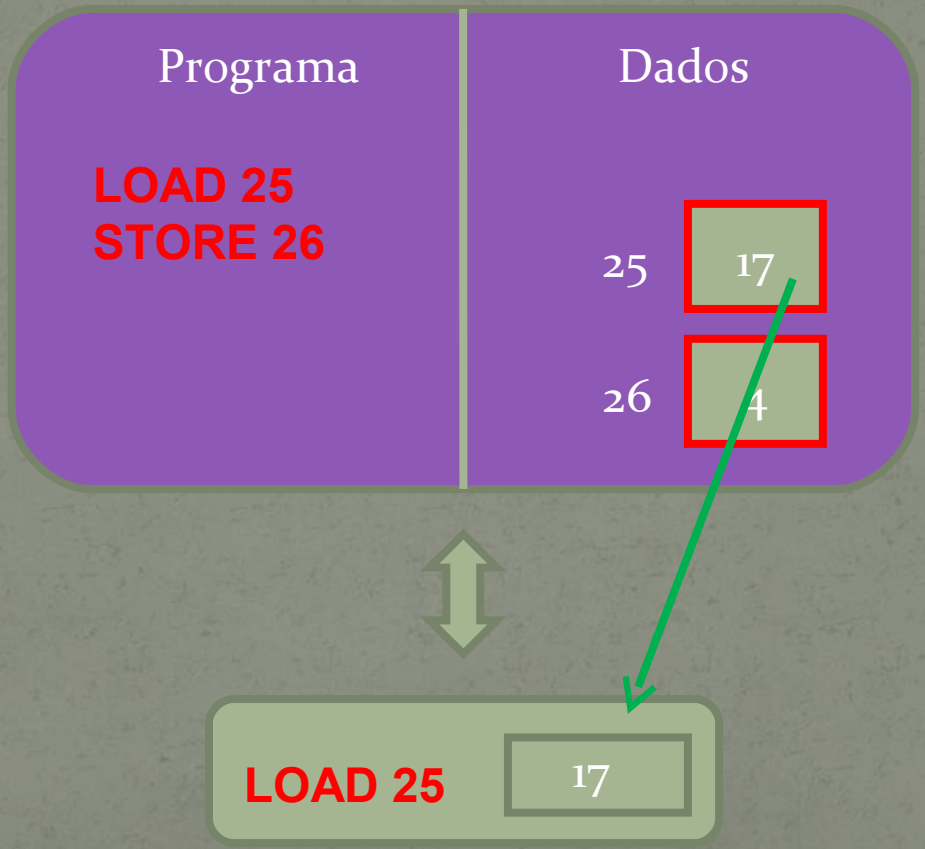
Modelo de von Neumann



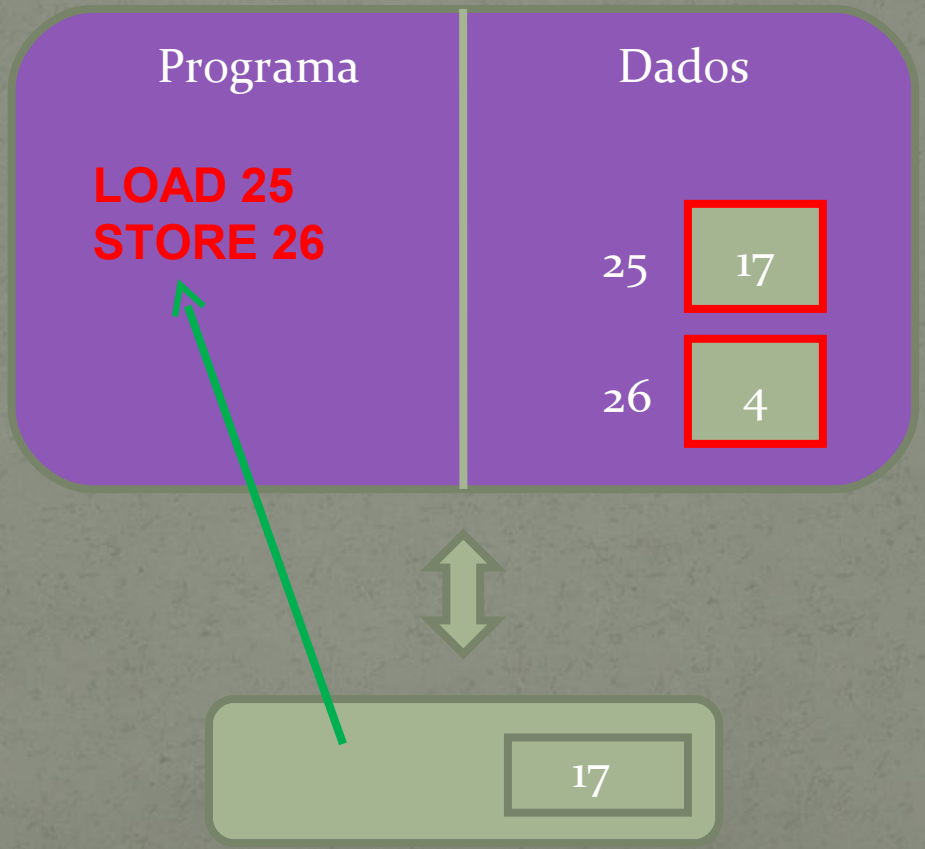
Modelo de von Neumann



Modelo de von Neumann



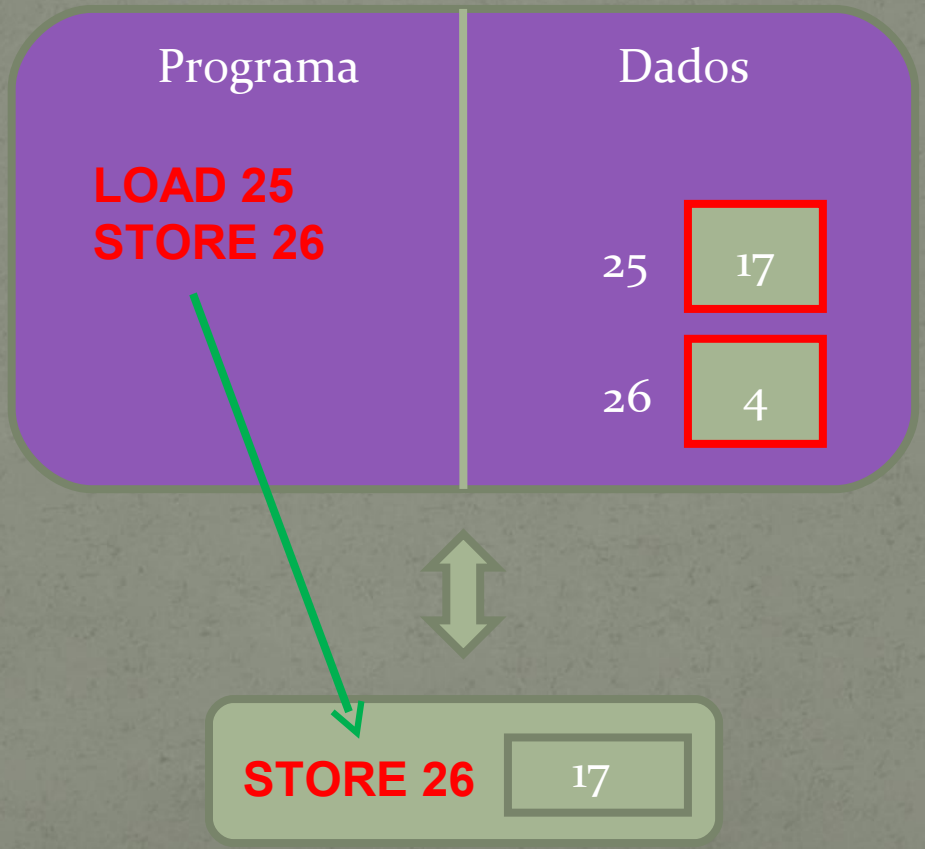
Modelo de von Neumann



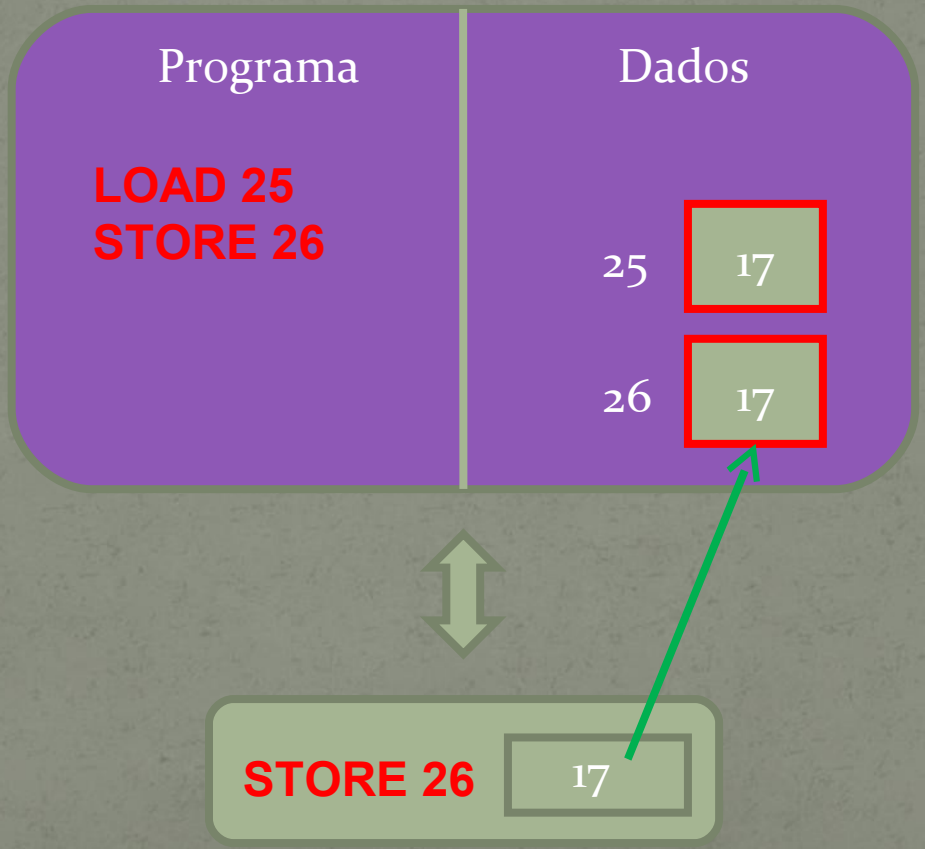
STORE

- Código 160;
- Usa número adicional;
- Formato: 160 <endereço>
- Copia o valor armazenado no registrador AX para a posição de memória <endereço>;
- O valor anterior da posição de memória <endereço> é perdido;
- O valor de AX permanece inalterado;
- Exemplo:
- 160 25
  ou
  STORE 25

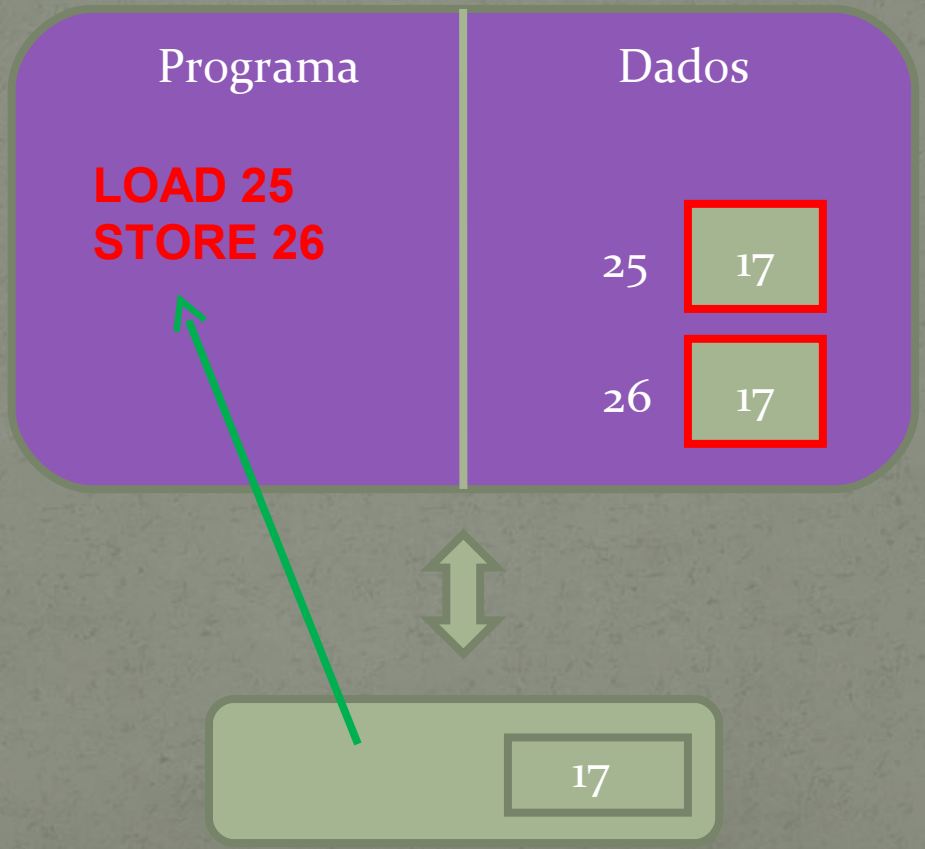
Modelo de von Neumann



Modelo de von Neumann



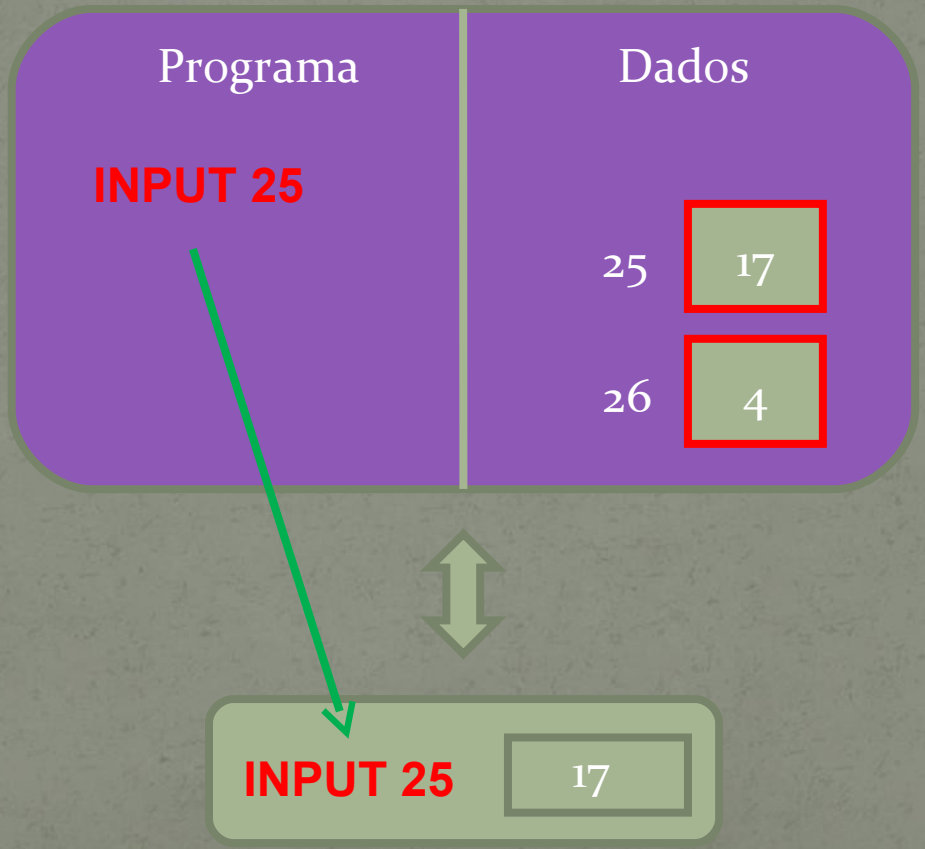
Modelo de von Neumann



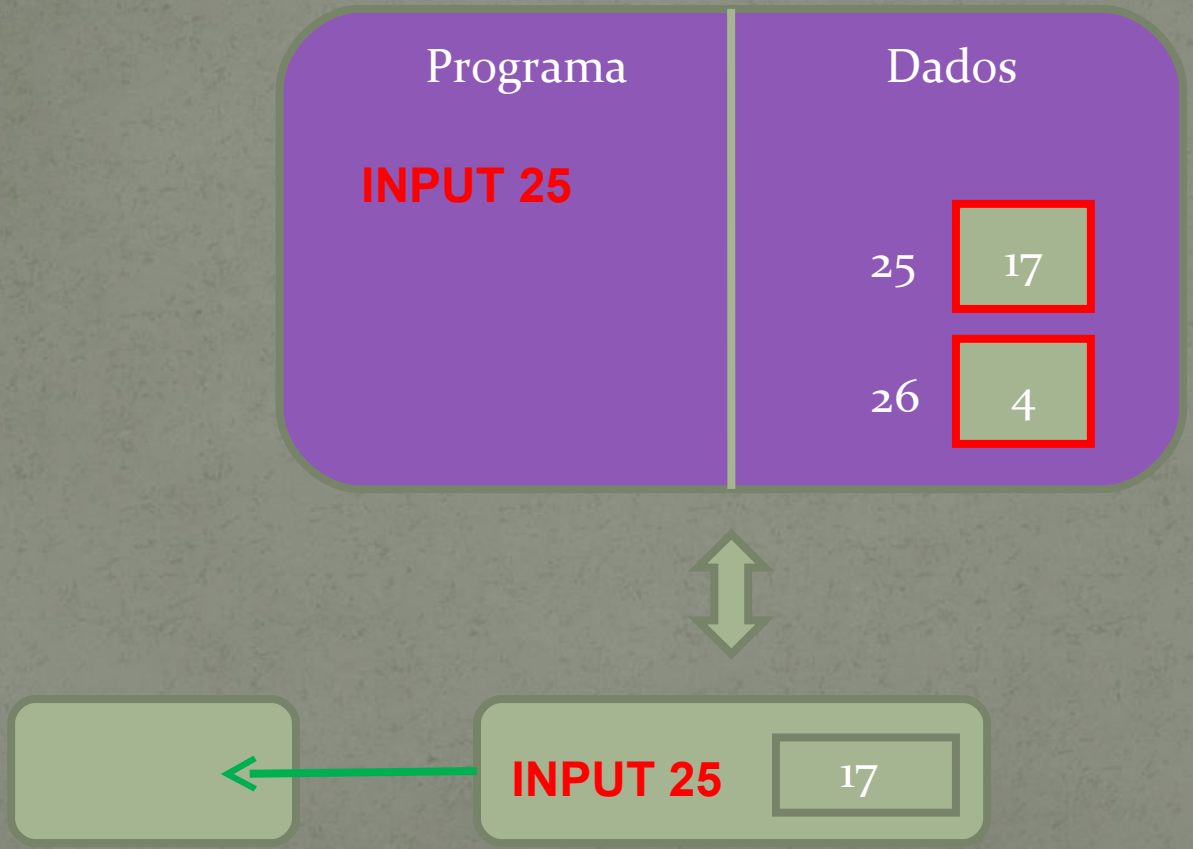
INPUT

- Código 71;
- Usa número adicional;
- Formato: 71 <endereço>
- Aguarda o usuário digitar um número e depois armazena o mesmo na posição de memória <endereço>;
- O valor anterior da posição de memória <endereço> é perdido;
- O valor de AX permanece inalterado;
- Exemplo:
- 71 25
  ou
  INPUT 25

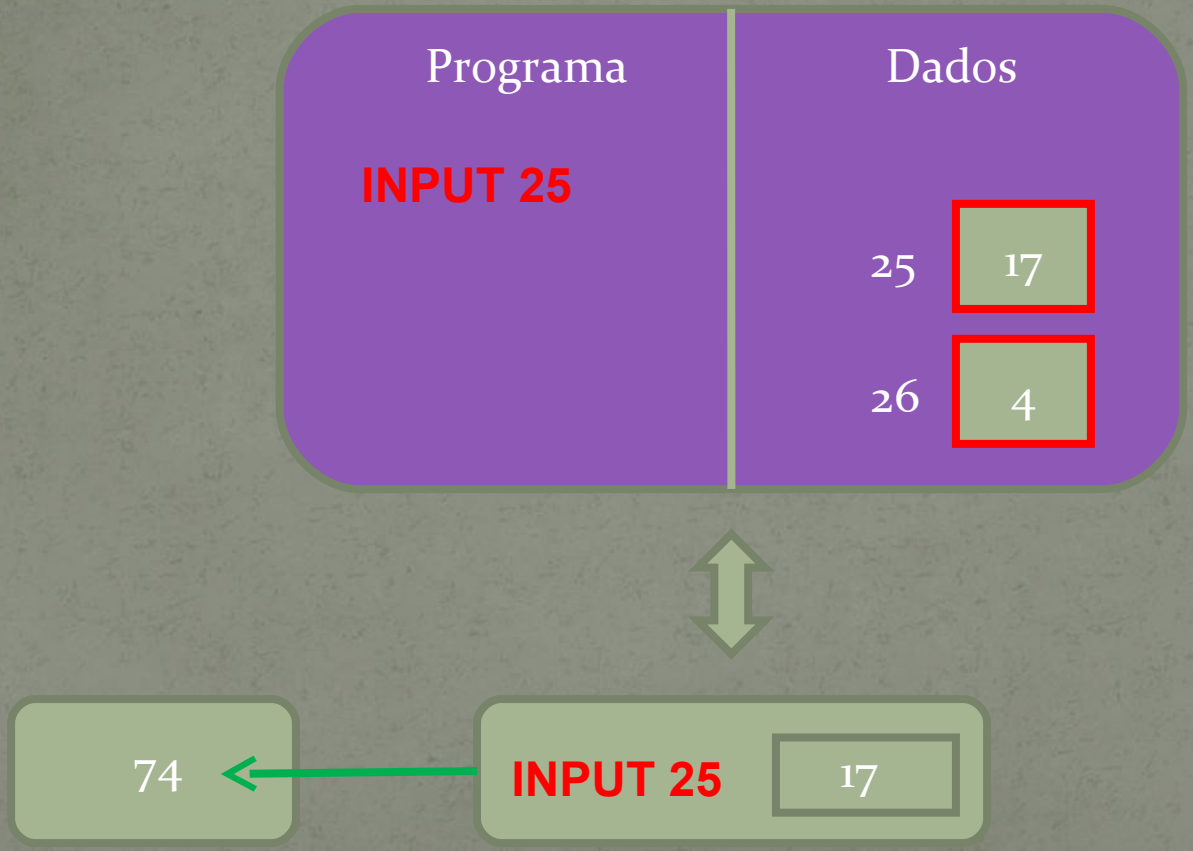
Modelo de von Neumann



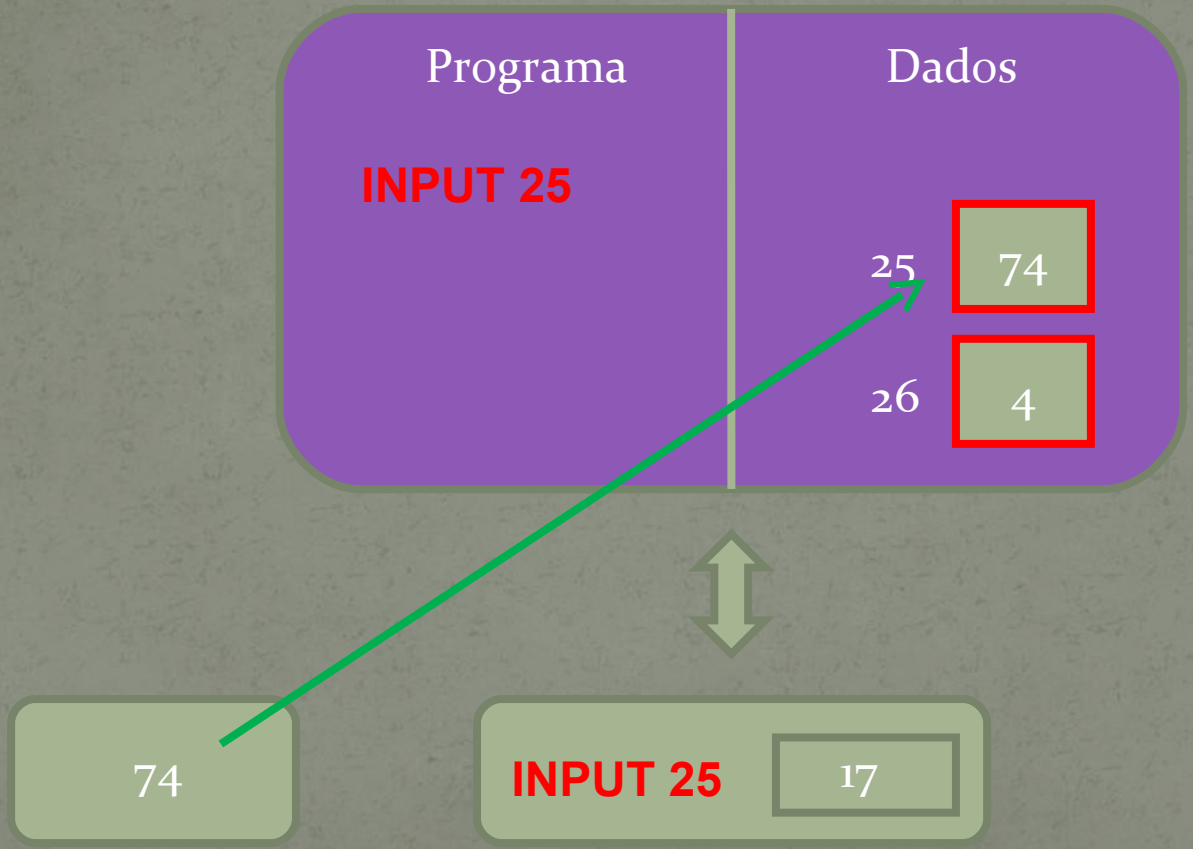
Modelo de von Neumann



Modelo de von Neumann



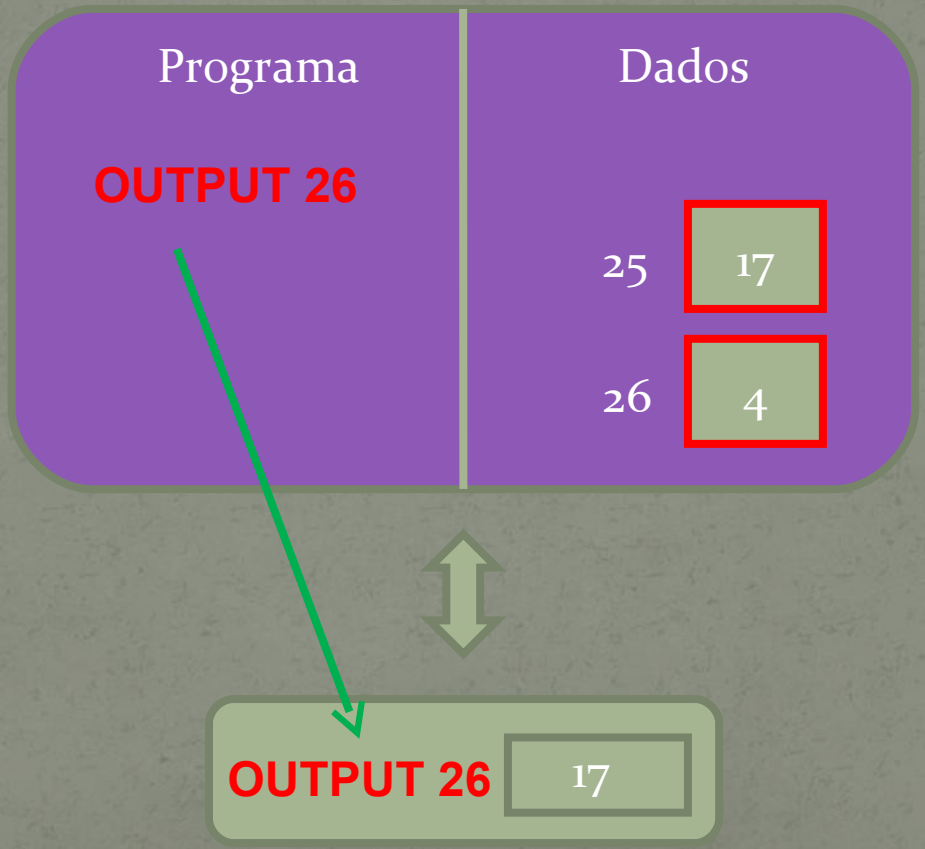
Modelo de von Neumann



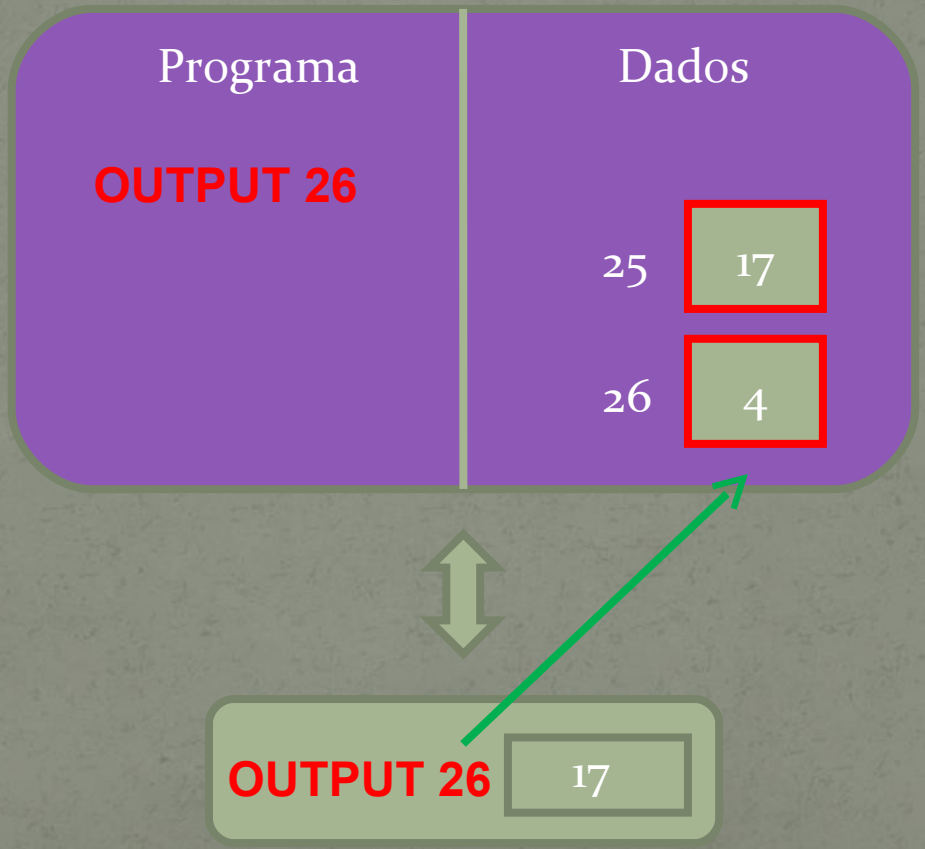
OUTPUT

- Código 72;
- Usa número adicional;
- Formato: 72 <endereço>
- Envia o valor armazenado na posição de memória <endereço> para o dispositivo de saída;
- O valor da posição de memória <endereço> permanece inalterado;
- O valor de AX permanece inalterado;
- Exemplo:
- 72 25
  ou
  OUPUT 25

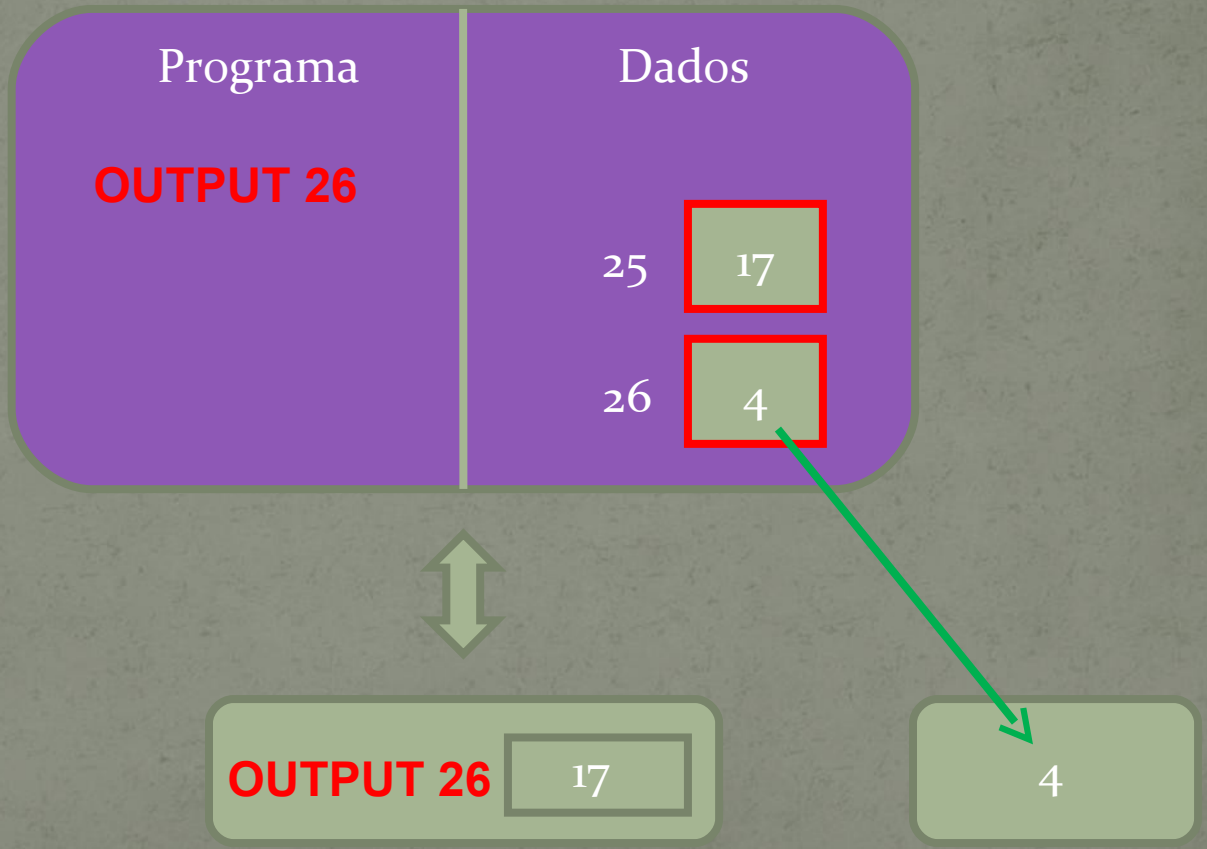
Modelo de von Neumann



Modelo de von Neumann



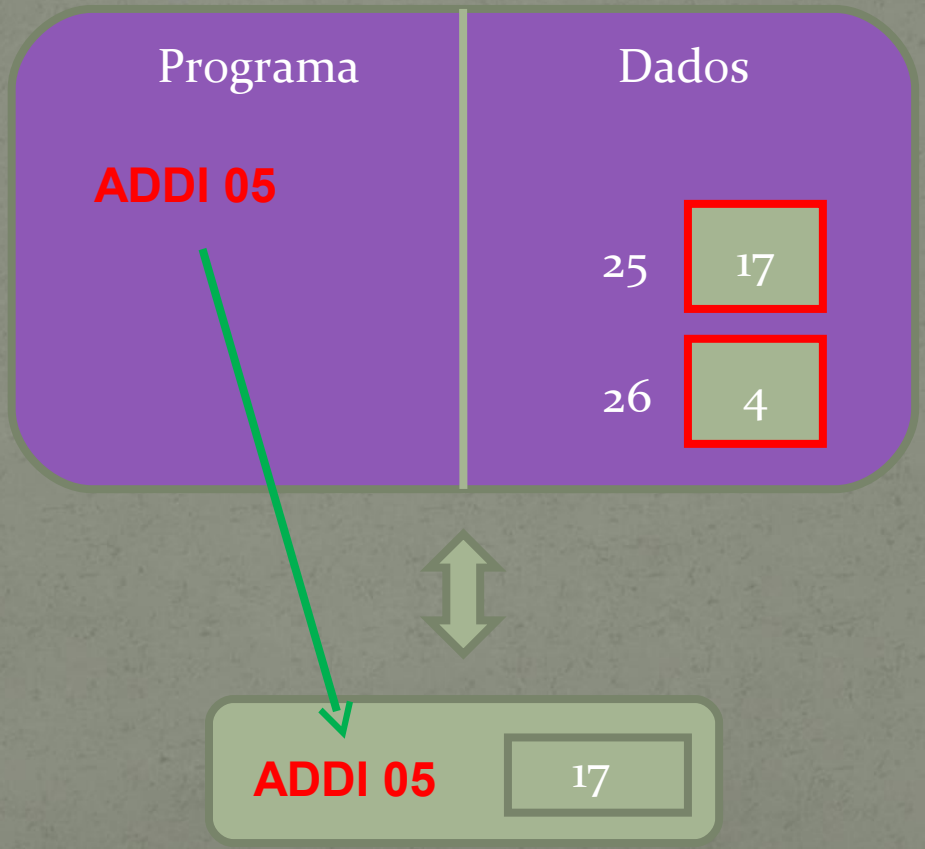
Modelo de von Neumann



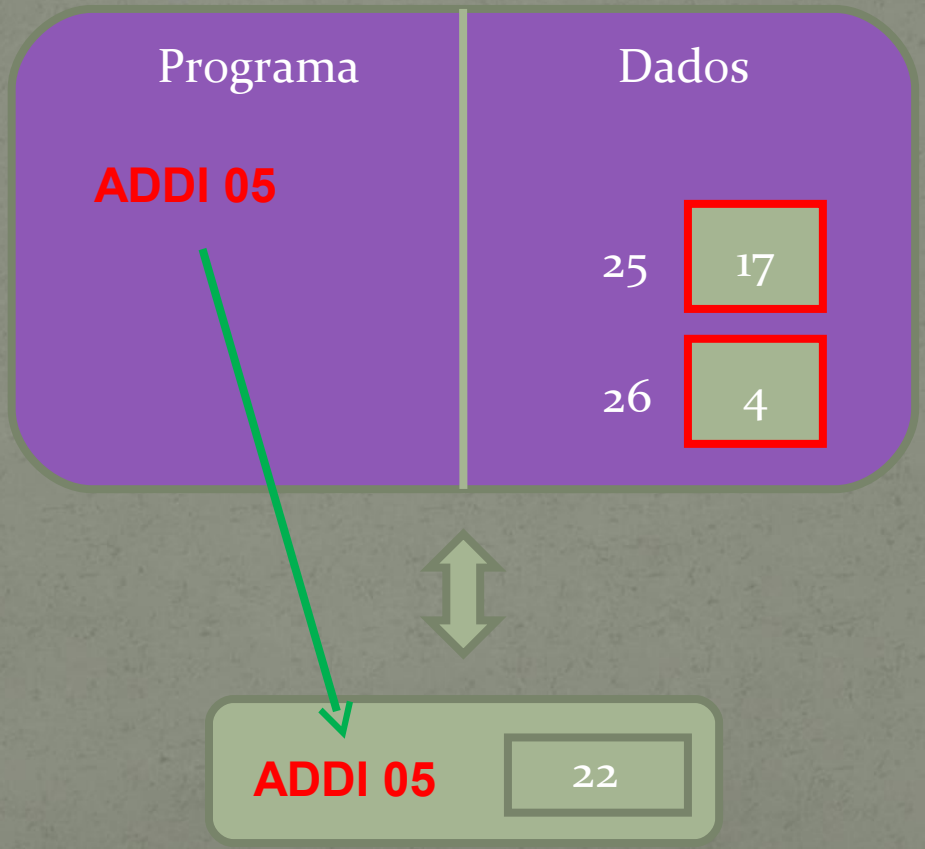
ADDI

- Código 44;
- Usa número adicional;
- Formato: 44 <valor>
- Soma o conteúdo corrente do registrador AX com <valor>, e deixa o resultado no próprio AX;
- Exemplo:
- 44 01
  ou
  ADDI 01

Modelo de von Neumann



Modelo de von Neumann



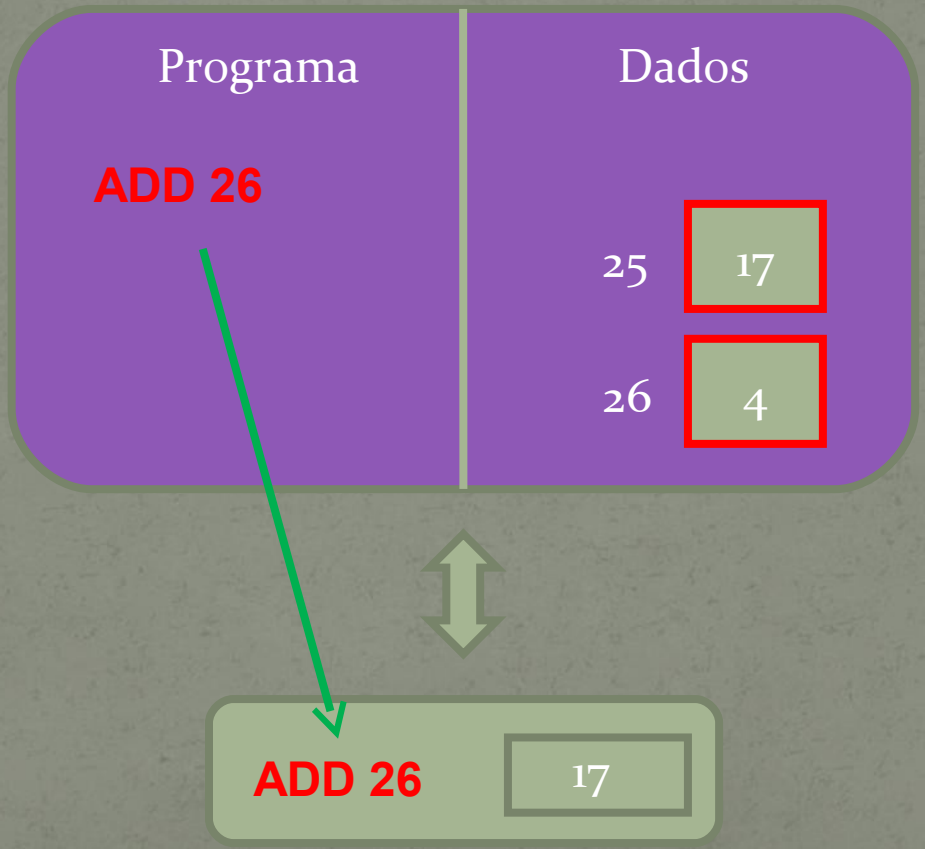
SUBI

- Código 45;
- Usa número adicional;
- Formato: 45 <valor>
- Subtrai do conteúdo corrente do registrador AX <valor>, e deixa o resultado no próprio AX;
- Exemplo:
- 45 01
  ou
  SUBI 01

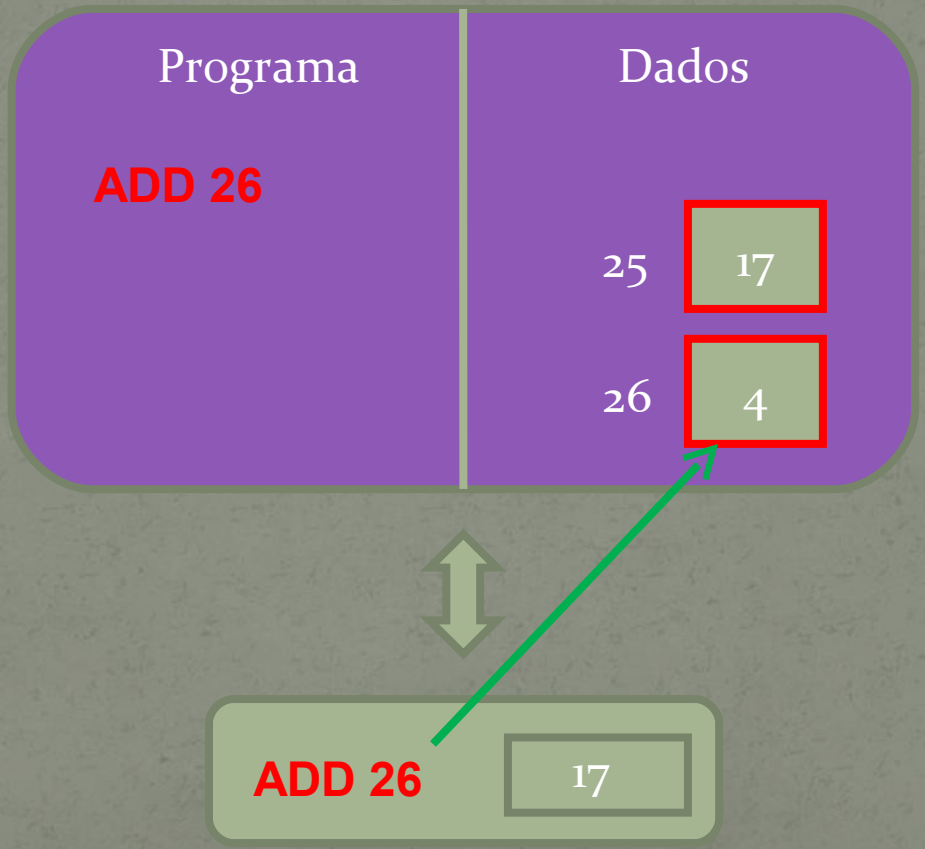
ADD

- Código 50;
- Usa número adicional;
- Formato: 50 <endereço>
- Soma o conteúdo corrente do registrador AX com o valor armazenado na posição de memória <endereço>, e deixa o resultado no próprio AX;
- O conteúdo da posição de memória <endereço> permanece inalterado;
- Exemplo:
- 50 25
  ou
  ADD 25

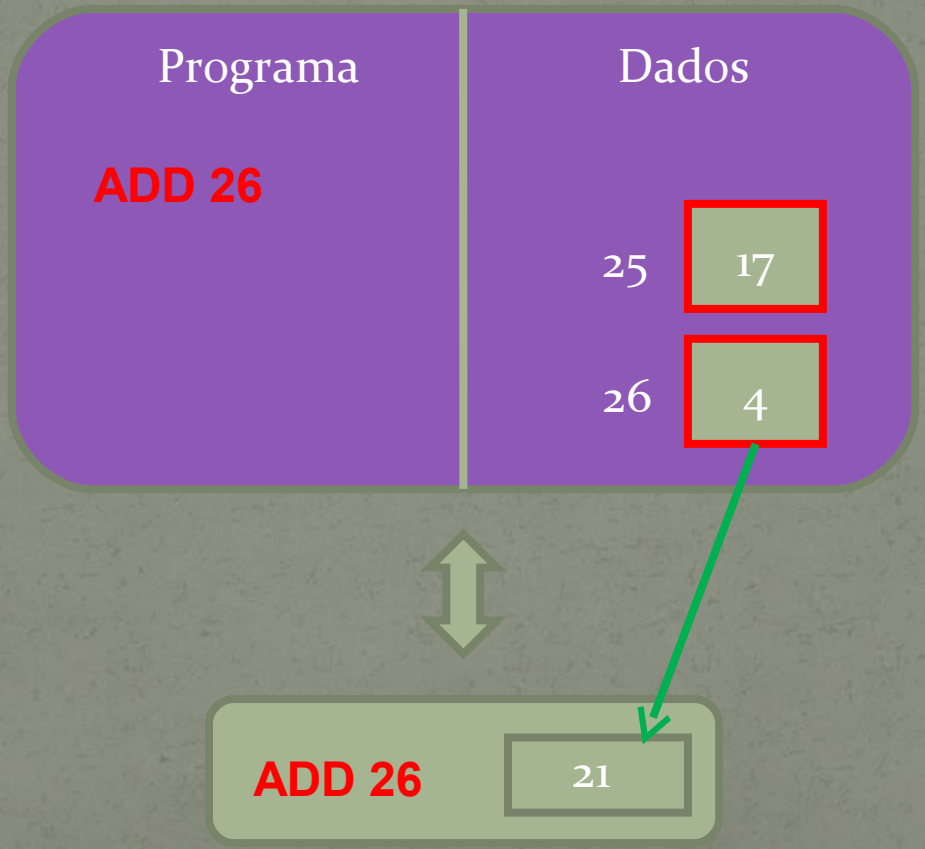
Modelo de von Neumann



Modelo de von Neumann



Modelo de von Neumann



SUB

- Código 51;
- Usa número adicional;
- Formato: 51 <endereço>
- Subtrai do conteúdo corrente do registrador AX o valor armazenado na posição de memória <endereço>, e deixa o resultado no próprio AX;
- O conteúdo da posição de memória <endereço> permanece inalterado;
- Exemplo:
- 51 25
  ou
  SUB 25

MUL

- Código 52;
- Usa número adicional;
- Formato: 52 <endereço>
- Multiplica o conteúdo corrente do registrador AX pelo valor armazenado na posição de memória <endereço>, e deixa o resultado no próprio AX;
- O conteúdo da posição de memória <endereço> permanece inalterado;
- Exemplo:
- 52 25
  ou
  MUL 25

DIV

- Código 53;
- Usa número adicional;
- Formato: 53 <endereço>
- Divide o conteúdo corrente do registrador AX pelo valor armazenado na posição de memória <endereço>, e deixa o resultado no próprio AX;
- O conteúdo da posição de memória <endereço> permanece inalterado;
- Exemplo:
- 53 25
  ou
  DIV 25

HALT

- Código 00;
- Não usa número adicional;
- Formato: 00
- Encerra a execução do programa e devolve o controle para o sistema operacional;
- Exemplo:
- 00
  ou
  HALT

Conjunto completo de instruções

Mnemonic	Code (1st word)	Code (2nd word)	Description
STORE	160	n	Store the contents of AX in memory location n
LOAD	161	n	Load AX with the contents of location n
ADDI	44	n	Add n to AX (add immediate)
SUBI	45	n	Subtract n from AX (subtract immediate)
ADD	50	n	Add the contents of location n to AX
SUB	51	n	Subtract the contents of location n from AX
MUL	52	n	Multiply the contents AX by the contents of location n
DIV	53	n	Divide the contents of AX by the contents of location n
INPUT	71	n	Enable input button and accept input into location n
OUTPUT	72	n	Output the contents of location n
JPOS	127	n	Jump to the instruction in location n if AX > 0
JZERO	128	n	Jump to the instruction in location n if AX = 0
HALT	0		Halt the execution

Exemplo de programa

Linguagem de montagem		Linguagem de máquina	
INPUT	25	71	25
INPUT	26	71	26
LOAD	25	161	25
ADD	26	50	26
ADDI	01	44	01
STORE	27	160	27
OUTPUT	27	72	27
HALT		00	

Instructions: STORE, LOAD, ADDI, SUBI, ADD, SUB, MUL, DIV, INPUT, OUTPUT, JPOS, JZERO, HALT

MEM	0	1	2	3	4	5	6	7	8	9
0	INPUT	25	INPUT	26	LOAD	25	ADD	26	ADDI	1
1	STORE	27	OUTPUT	27	HALT					
2										
3										
4										
5										
6										
7										
8										
9										

Bus Activity Idle

CPU internal phase CPU is idle

PC 0 AX 0 Run Single Step Halt Clear Status Area

Delay (millisec): 500 Sample programs: Count to 5 Calculate 4 factorial Sum 5 numbers

Input: Output:

Instructions: STORE, LOAD, ADDI, SUBI, ADD, SUB, MUL, DIV, INPUT, OUTPUT, JPOS, JZERO, HALT

MEM	0	1	2	3	4	5	6	7	8	9
0	71	25	71	26	161	25	50	26	44	1
1	160	27	72	27	00					
2										
3										
4										
5										
6										
7										
8										
9										

Bus Activity Fetch 0

CPU internal phase Executing instruction

PC 0 AX Run Single Step Halt Clear Single step

Delay (millisec): 500 Sample programs: Count to 5 Calculate 4 factorial Sum 5 numbers

Input: Output:

Exemplo de programa

Linguagem de montagem		Linguagem de máquina	
INPUT	**25**	**71**	**25**
INPUT	26	71	26
LOAD	25	161	25
ADD	26	50	26
ADDI	01	44	01
STORE	27	160	27
OUTPUT	27	72	27
HALT		00	

Instructions: STORE, LOAD, ADDI, SUBI, ADD, SUB, MUL, DIV, INPUT, OUTPUT, JPOS, JZERO, HALT

MEM	0	1	2	3	4	5	6	7	8	9
0	INPUT	25	INPUT	26	LOAD	25	ADD	26	ADDI	1
1	STORE	27	OUTPUT	27	HALT					
2										
3										
4										
5										
6										
7										
8										
9										

Bus Activity Idle

CPU internal phase Executing instruction

PC 1 AX 0 Run Single Step Halt Clear Suspended awaiting input

Delay (millisec): 500 Sample programs: Count to 5 Calculate 4 factorial Sum 5 numbers

Input: 3 Enter input. Output:

Instructions: STORE, LOAD, ADDI, SUBI, ADD, SUB, MUL, DIV, INPUT, OUTPUT, JPOS, JZERO, HALT

MEM	0	1	2	3	4	5	6	7	8	9
0	INPUT	25	INPUT	26	LOAD	25	ADD	26	ADDI	1
1	STORE	27	OUTPUT	27	HALT					
2						3				
3										
4										
5										
6										
7										
8										
9										

Bus Activity Fetch 3

CPU internal phase Executing instruction

PC 2 AX 0 Run Single Step Halt Clear Single step

Delay (millisec): 500 Sample programs: Count to 5 Calculate 4 factorial Sum 5 numbers

Input: Output:

Exemplo de programa

Linguagem de montagem		Linguagem de máquina	
INPUT	25	71	25
INPUT	**26**	**71**	**26**
LOAD	25	161	25
ADD	26	50	26
ADDI	01	44	01
STORE	27	160	27
OUTPUT	27	72	27
HALT		00	

Instructions: STORE, LOAD, ADDI, SUBI, ADD, SUB, MUL, DIV, INPUT, OUTPUT, JPOS, JZERO, HALT

MEM	0	1	2	3	4	5	6	7	8	9
0	INPUT	25	INPUT	26	LOAD	25	ADD	26	ADDI	1
1	STORE	27	OUTPUT	27	HALT					
2						3	4			
3										
4										
5										
6										
7										
8										
9										

Bus Activity Fetch 4

CPU internal phase Executing instruction

PC 4 AX 0 Run Single Step Halt Clear Single step

Delay (millisec): 500 Sample programs: Count to 5 Calculate 4 factorial Sum 5 numbers

Input: Output:

Exemplo de programa

Linguagem de montagem		Linguagem de máquina	
INPUT	25	71	25
INPUT	26	71	26
LOAD	**25**	**161**	**25**
ADD	26	50	26
ADDI	01	44	01
STORE	27	160	27
OUTPUT	27	72	27
HALT		00	

Instructions: STORE, LOAD, ADDI, SUBI, ADD, SUB, MUL, DIV, INPUT, OUTPUT, JPOS, JZERO, HALT

MEM	0	1	2	3	4	5	6	7	8	9
0	INPUT	25	INPUT	26	LOAD	25	ADD	26	ADDI	1
1	STORE	27	OUTPUT	27	HALT					
2						3	4			
3										
4										
5										
6										
7										
8										
9										

Bus Activity Idle

CPU internal phase Executing instruction

PC 6 AX 3 Run Single Step Halt Clear Single step

Delay (millisec): 500 Sample programs: Count to 5 Calculate 4 factorial Sum 5 numbers

Input: Output:

Exemplo de programa

Linguagem de montagem		Linguagem de máquina	
INPUT	25	71	25
INPUT	26	71	26
LOAD	25	161	25
ADD	**26**	**50**	**26**
ADDI	01	44	01
STORE	27	160	27
OUTPUT	27	72	27
HALT		00	

Instructions: STORE, LOAD, ADDI, SUBI, ADD, SUB, MUL, DIV, INPUT, OUTPUT, JPOS, JZERO, HALT

MEM	0	1	2	3	4	5	6	7	8	9
0	INPUT	25	INPUT	26	LOAD	25	ADD	26	ADDI	1
1	STORE	27	OUTPUT	27	HALT					
2						3	4			
3										
4										
5										
6										
7										
8										
9										

Bus Activity Idle

CPU internal phase Executing instruction

PC 8 AX 7 Run Single Step Halt Clear Single step

Delay (millisec): 500 Sample programs: Count to 5 Calculate 4 factorial Sum 5 numbers

Input: Output:

Exemplo de programa

Linguagem de montagem		Linguagem de máquina	
INPUT	25	71	25
INPUT	26	71	26
LOAD	25	161	25
ADD	26	50	26
ADDI	**01**	**44**	**01**
STORE	27	160	27
OUTPUT	27	72	27
HALT		00	

Instructions: STORE, LOAD, ADDI, SUBI, ADD, SUB, MUL, DIV, INPUT, OUTPUT, JPOS, JZERO, HALT

MEM	0	1	2	3	4	5	6	7	8	9
0	INPUT	25	INPUT	26	LOAD	25	ADD	26	ADDI	1
1	STORE	27	OUTPUT	27	HALT					
2						3	4			
3										
4										
5										
6										
7										
8										
9										

Bus Activity Idle

CPU internal phase Executing instruction

PC 10 AX 8 Run Single Step Halt Clear Single step

Delay (millisec): 500 Sample programs: Count to 5 Calculate 4 factorial Sum 5 numbers

Input: Output:

Exemplo de programa

Linguagem de montagem		Linguagem de máquina	
INPUT	25	71	25
INPUT	26	71	26
LOAD	25	161	25
ADD	26	50	26
ADDI	01	44	01
STORE	**27**	**160**	**27**
OUTPUT	27	72	27
HALT		00	

Instructions: STORE, LOAD, ADDI, SUBI, ADD, SUB, MUL, DIV, INPUT, OUTPUT, JPOS, JZERO, HALT

MEM	0	1	2	3	4	5	6	7	8	9
0	INPUT	25	INPUT	26	LOAD	25	ADD	26	ADDI	1
1	STORE	27	OUTPUT	27	HALT					
2						3	4	8		
3										
4										
5										
6										
7										
8										
9										

Bus Activity Idle

CPU internal phase Executing instruction

PC 12 AX 8 Run Single Step Halt Clear Single step

Delay (millisec): 500 Sample programs: Count to 5 Calculate 4 factorial Sum 5 numbers

Input: Output:

Exemplo de programa

Linguagem de montagem		Linguagem de máquina	
INPUT	25	71	25
INPUT	26	71	26
LOAD	25	161	25
ADD	26	50	26
ADDI	01	44	01
STORE	27	160	27
OUTPUT	**27**	**72**	**27**
HALT		00	

Instructions: STORE, LOAD, ADDI, SUBI, ADD, SUB, MUL, DIV, INPUT, OUTPUT, JPOS, JZERO, HALT

MEM	0	1	2	3	4	5	6	7	8	9
0	INPUT	25	INPUT	26	LOAD	25	ADD	26	ADDI	1
1	STORE	27	OUTPUT	27	HALT					
2						3	4	8		
3										
4										
5										
6										
7										
8										
9										

Bus Activity Idle

CPU internal phase Executing instruction

PC 14 AX 8 Run Single Step Halt Clear Single step

Delay (millisec): 500 Sample programs: Count to 5 Calculate 4 factorial Sum 5 numbers

Input: Output: 8

Desvio do fluxo

Fluxo normal: seqüencial

00 --- (primeiro esta)
01 --- (depois esta)
02 --- (etc)
03 ---
04 ---
05 ---
06 ---
07 ---

Desvio do fluxo

Fluxo desviado: uma instrução indica qual deve ser a próxima a ser executada

00 --- (primeiro esta)
01 --- (depois esta)
02 --- (executa a instrução que está no endereço 5)
03 ---
04 ---
05 --- (depois esta!!)
06 ---
07 ---

Desvio do fluxo

Fluxo desviado: pode ser CONDICIONAL ou INCONDICIONAL

00 --- (primeiro esta)
01 --- (depois esta)
02 --- (VÁ PARA a instrução que está no endereço 5)
03 ---
04 ---
05 --- (depois esta!!)
06 ---
07 ---

Desvio do fluxo

Fluxo desviado: pode ser CONDICIONAL ou INCONDICIONAL

00 --- (primeiro esta)
01 --- (depois esta)
02 --- (VÁ PARA a instrução que está no endereço 5 SE...)
03 ---
04 ---
05 --- (depois esta!!)
06 ---
07 ---

JPOS

- Código 127;
- Usa número adicional;
- Formato: 127 <endereço>
- Se AX>0, faz com que a próxima instrução a ser executada seja aquela que está armazenada na posição de memória <endereço>; caso contrário, executa a instrução seguinte;
- Exemplo:
- 127 25
  ou
  JPOS 25

JZERO

- Código 128;
- Usa número adicional;
- Formato: 128 <endereço>
- Se AX=0, faz com que a próxima instrução a ser executada seja aquela que está armazenada na posição de memória <endereço>; caso contrário, executa a instrução seguinte;
- Exemplo:
- 128 25
  ou
  JZERO 25

Exemplo de programa

00	INPUT 25	71	25
02	INPUT 26	71	26
04	LOAD 25	161	25
06	SUB 26	51	26
08	JPOS 13	127	13
10	OUPUT 26	72	26
12	HALT	00	
13	OUTPUT 25	72	25
15	HALT	00	

Determina o maior de dois números informados pelo usuário

Diferenças para um computador real

- Quantidade de posições de memória;
- Quantidade de bits de cada posição de memória;
- Quantidade e variedade de instruções;
- Velocidade de execução;
- Implementação em hardware;
- Variedade de dispositivos de entrada e de saída.

Unidades

- Sistema binário
- 1 bit = 0 ou 1 (menor unidade de informação)
- 1 byte = 8 bits
- 1 KiloByte = 1KB = 1.024 bytes
- $1.024 = 2^{10}$
- 1 MegaByte = 1MB = 1.024 KiloBytes = 1.024*1.024 bytes ~ 1 milhão de bytes
- 1 GigaByte = 1 GB = 1.024 MegaBytes = 1.024*1.024*1.024 bytes ~ 1 bilhão de bytes
- Bits, Bytes, Herz, ...
- Kilo, Mega, Giga, Tera...

Sistemas de numeração

- Sistema decimal:

  xyz =

  $x*100 + y*10 + z*1 =$

  $x*10^2 + x*10^1 + x*10^0$

Sistema binário:

  xyz =

  $x*4 + y*2 + z*1 =$

  $x*2^2 + x*2^1 + x*2^0$

Sistemas de numeração

- Sistema decimal:

  xyz =

  $x*100 + y*10 + z*1 =$

  $x*10^2 + x*10^1 + x*10^0$

Sistema binário:

  xyz =

  $x*4 + y*2 + z*1 =$

  $x*2^2 + x*2^1 + x*2^0$

Sistemas de numeração

- Sistema decimal:

  518_{10} =

  $5*100 + 1*10 + 8*1$ =

  $5*10^2 + 1*10^1 + 8*10^0$

Sistema binário:

101_2 =

$1*4 + 0*2 + 1*1$ =

$1*2^2 + 0*2^1 + 1*2^0 = 5_{10}$

Exercício

- Fazer um programa para:
  Calcular a quantidade de ração necessária para alimentar X vacas e Y bezerros durante Z dias, sabendo que cada vaca consome M Kg por dia e cada bezerro consome N Kg por dia;
- Ligar o computador;
- Carregar o simulador;
- Digitar o programa;
- Conferir o resultado da execução com os seguintes valores:
  10 vacas, 6 bezerros, 3Kg/vaca/dia, 1 Kg/bezerro/dia, 30 dias.