

O Computador Neander

► A Arquitetura: conjunto de instruções

código	instrução	comentário
0000	NOP	Nenhuma operação
0001	STA end	$\text{MEM}(\text{end}) \leftarrow \text{AC}$
0010	LDA end	$\text{AC} \leftarrow \text{MEM}(\text{end})$
0011	ADD end	$\text{AC} \leftarrow \text{MEM}(\text{end}) + \text{AC}$
0100	OR end	$\text{AC} \leftarrow \text{MEM}(\text{end}) \text{ OR } \text{AC}$
0101	AND end	$\text{AC} \leftarrow \text{MEM}(\text{end}) \text{ AND } \text{AC}$
0110	NOT	$\text{AC} \leftarrow \text{NOT } \text{AC}$
1000	JMP end	$\text{PC} \leftarrow \text{end}$
1001	JN end	IF N=1 THEN $\text{PC} \leftarrow \text{end}$
1010	JZ end	IF Z=1 THEN $\text{PC} \leftarrow \text{end}$
1111	HLT	pára processamento

O Computador Neander

► A Organização: transferências necessárias

Analizando todas as descrições RT, a agrupando pelo registrador destino, tem-se:

$RI \leftarrow RDM$

$RDM \leftarrow AC$

Write

Read

$AC \leftarrow RDM$; atualiza N e Z

$AC \leftarrow AC + RDM$; atualiza N e Z

$AC \leftarrow AC \text{ OR } RDM$; atualiza N e Z

$AC \leftarrow AC \text{ AND } RDM$; atualiza N e Z

$AC \leftarrow \text{NOT}(AC)$; atualiza N e Z

$PC \leftarrow RDM$

$PC \leftarrow PC + 1$

$REM \leftarrow PC$

$REM \leftarrow RDM$

O Computador Neander

▶ A Organização: registradores

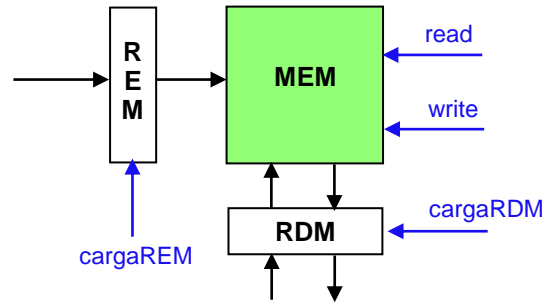
- AC: um registrador de 8 bits
- PC: um registrador de 8 bits (registrador-contador)
- RI: um registrador de 4 bits (ou 8)
- RDM: um registrador de 8 bits (largura do dado)
- REM: um registrador de 8 bits (largura do endereço)
- N: um flip-flop para o código de condição N
- Z: um flip-flop para o código de condição Z
- Uma memória de 256 posições (endereços) x 8 bits

O Computador Neander

► Organização do Sistema de Memória

Associados à Memória:

- RDM (dados)
- REM (endereços)
- sinal de escrita (write)
- sinal de leitura (read)

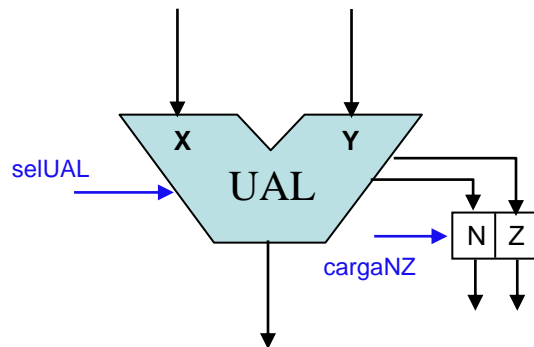


Cada registrador é controlado por um sinal de carga

► Organização da Unid. Aritmética e Lógica

Associados à UAL:

- 4 operações (ADD, AND, OR, NOT)
- sinal de controle (seleção)
- sinais de condição (N,Z)



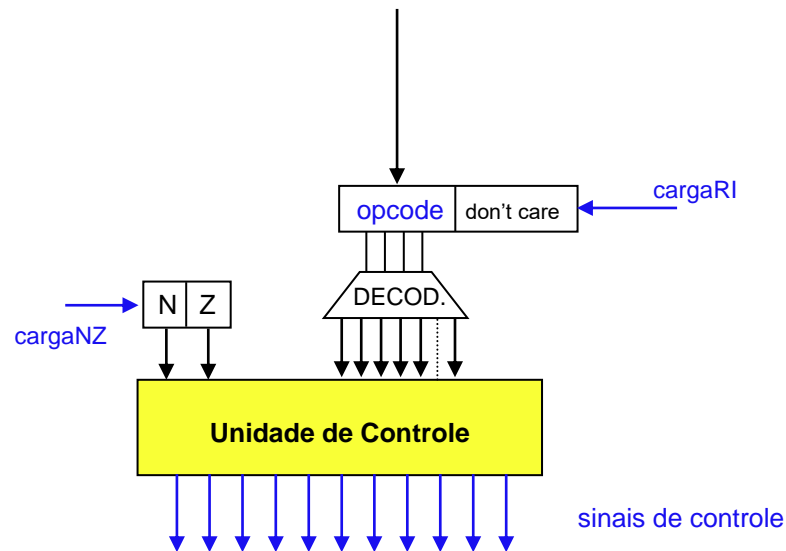
Flip-Flops devem ter sinal de carga

O Computador Neander

► Organização do Registrador de Instrução

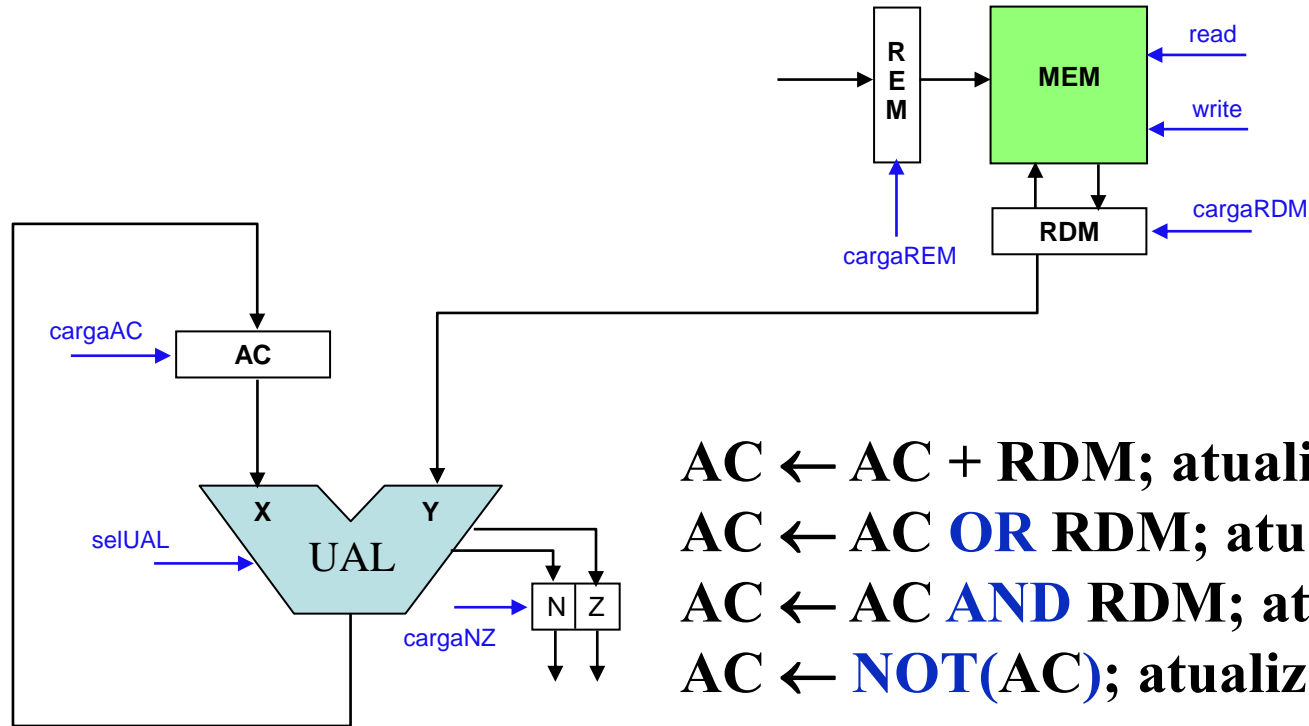
Associados ao Reg. de Instruções (4 ou 8 bits??):

- Decodificador (4 bits para 16 instruções)
- sinais de condição (N,Z) (para JN e JZ)
- registrador deve ter sinal de carga



O Computador Neander

▶ Operações na UAL



AC \leftarrow AC + RDM; atualiza N e Z

AC \leftarrow AC **OR** RDM; atualiza N e Z

AC \leftarrow AC **AND** RDM; atualiza N e Z

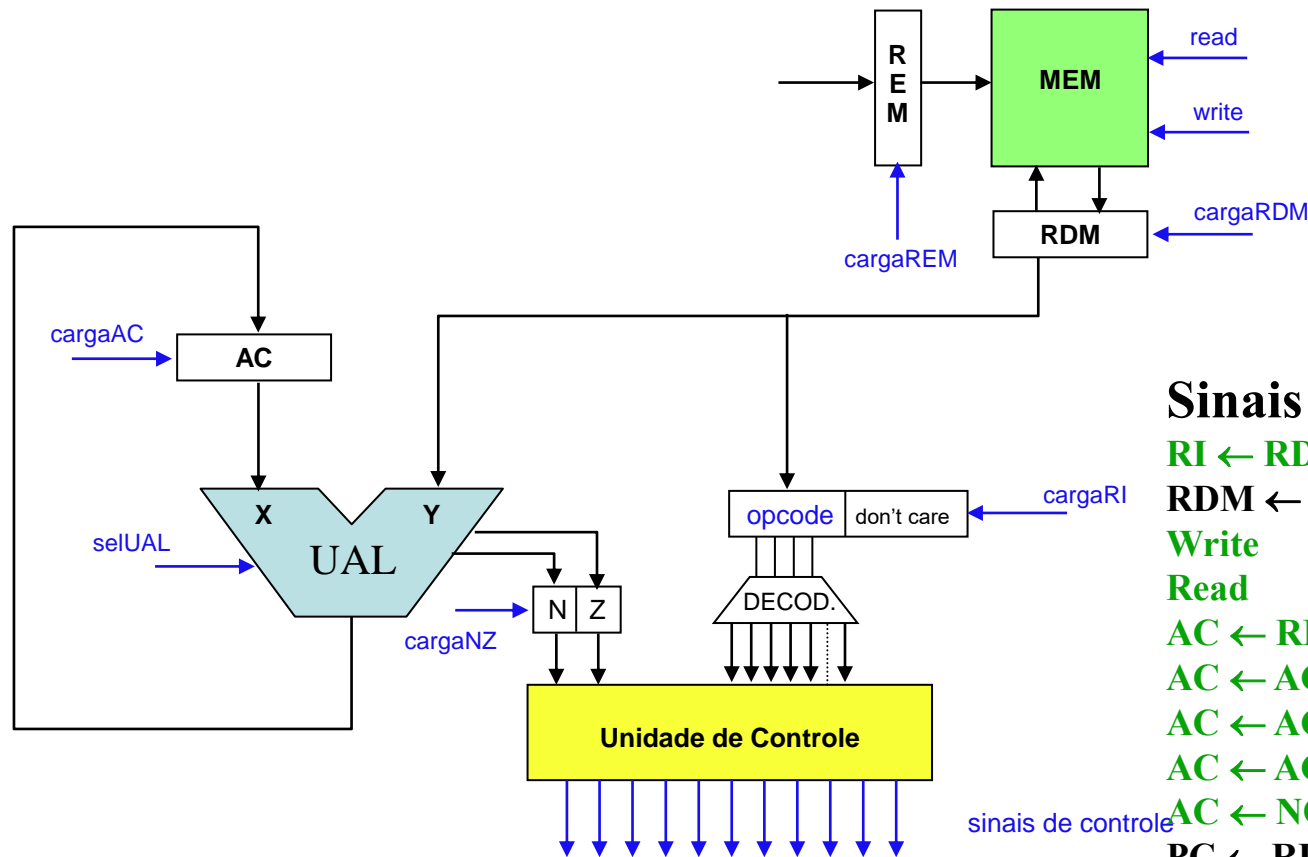
AC \leftarrow **NOT**(AC); atualiza N e Z

Dúvida:

AC \leftarrow RDM; atualiza N e Z (via UAL)

O Computador Neander

► Situação até aqui



Sinais de Controle

$RI \leftarrow RDM$

$RDM \leftarrow AC$

Write

Read

$AC \leftarrow RDM$; atualiza N e Z

$AC \leftarrow AC + RDM$; atualiza N e Z

$AC \leftarrow AC \text{ OR } RDM$; at. N e Z

$AC \leftarrow AC \text{ AND } RDM$; at. N e Z

$AC \leftarrow \text{NOT}(AC)$; atualiza N e Z

$PC \leftarrow RDM$

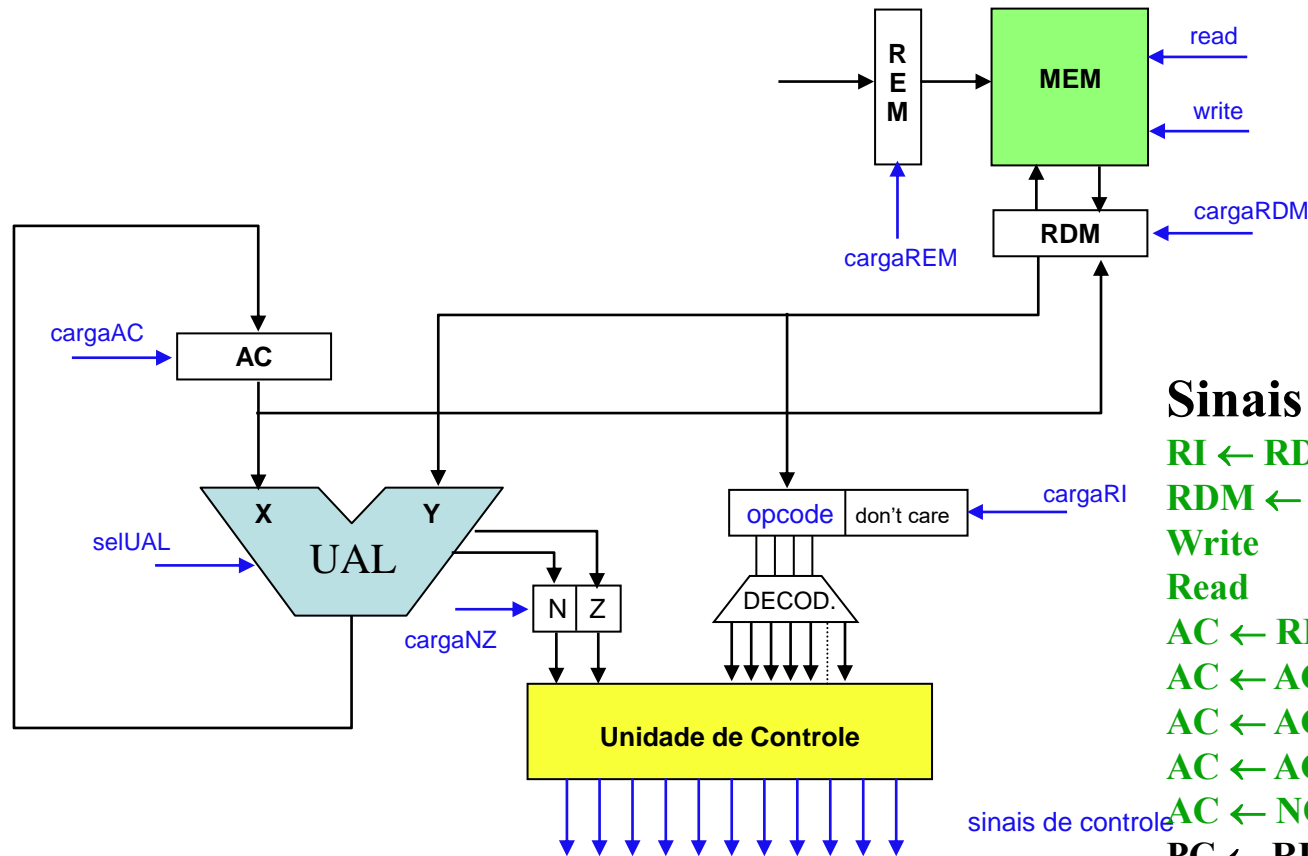
$PC \leftarrow PC + 1$

$REM \leftarrow PC$

$REM \leftarrow RDM$

O Computador Neander

► Acrescentado Escrita do AC



Sinais de Controle

$RI \leftarrow RDM$

$RDM \leftarrow AC$

Write

Read

$AC \leftarrow RDM$; atualiza N e Z

$AC \leftarrow AC + RDM$; atualiza N e Z

$AC \leftarrow AC \text{ OR } RDM$; at. N e Z

$AC \leftarrow AC \text{ AND } RDM$; at. N e Z

$AC \leftarrow \text{NOT}(AC)$; atualiza N e Z

$PC \leftarrow RDM$

$PC \leftarrow PC + 1$

$REM \leftarrow PC$

$REM \leftarrow RDM$

► **Acréscimo Program Counter (PC)**

O incremento do PC pode ser feito:

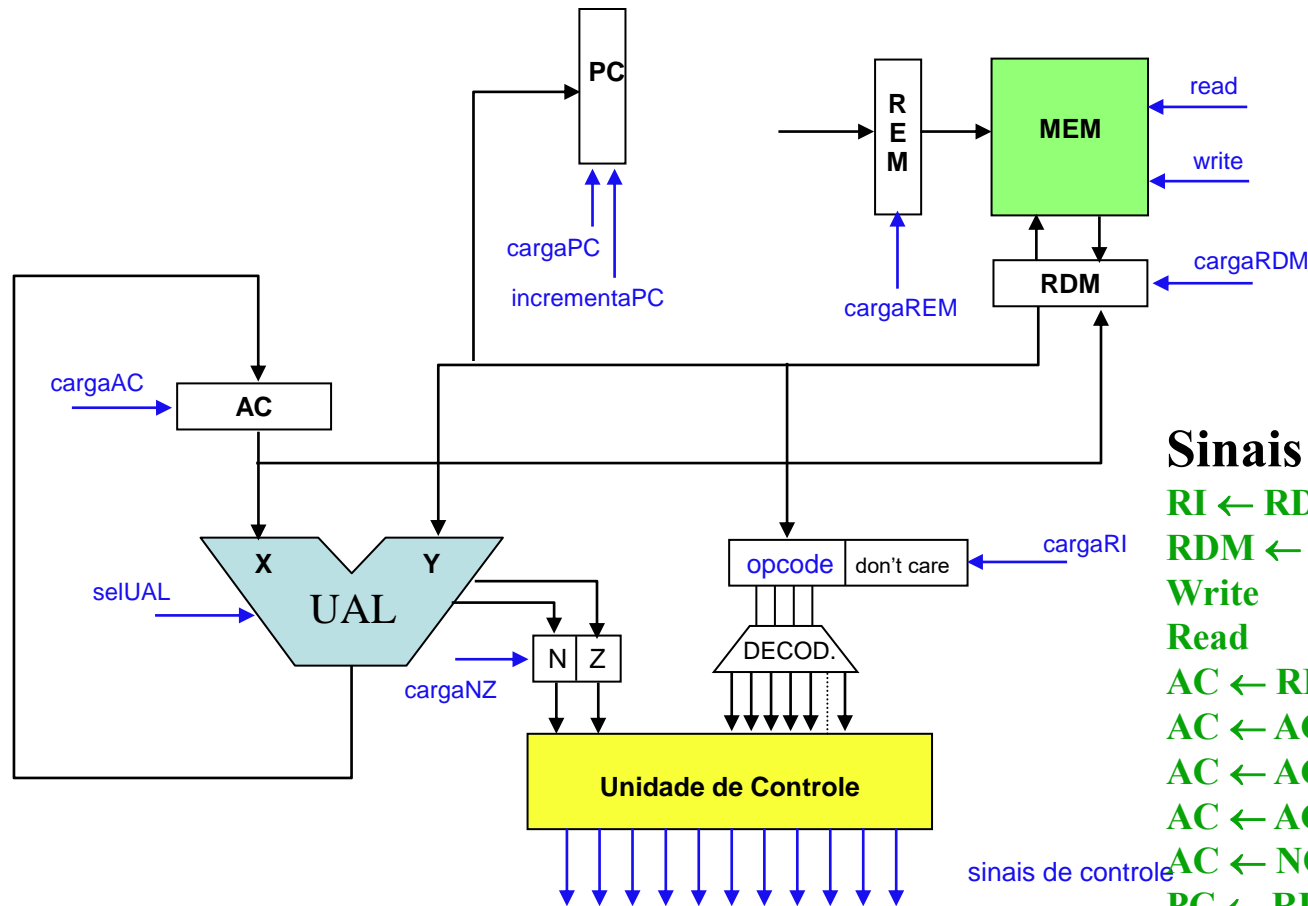
**Por meio de um somador
dedicado**

Usando a ULA

**Por meio de um registrador-
contador**

O Computador Neander

► Acrescentado Program Counter (PC)



Sinais de Controle

$RI \leftarrow RDM$

$RDM \leftarrow AC$

Write

Read

$AC \leftarrow RDM$; atualiza N e Z

$AC \leftarrow AC + RDM$; atualiza N e Z

$AC \leftarrow AC \text{ OR } RDM$; at. N e Z

$AC \leftarrow AC \text{ AND } RDM$; at. N e Z

$AC \leftarrow \text{NOT}(AC)$; atualiza N e Z

$PC \leftarrow RDM$

$PC \leftarrow PC + 1$

$REM \leftarrow PC$

$REM \leftarrow RDM$

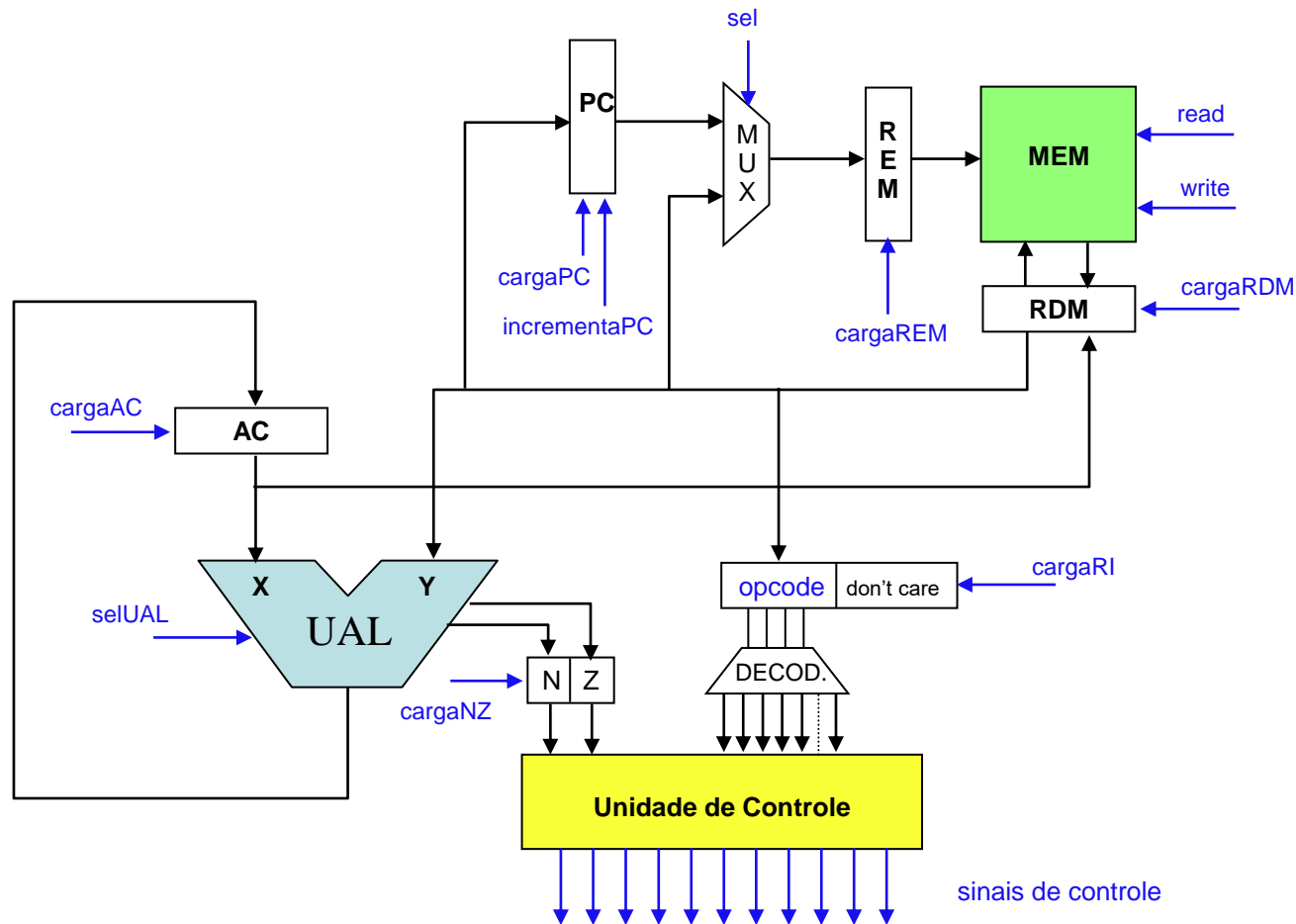
O Computador Neander

► Valores para o REM

- Existem duas transferências para o REM
$$\text{REM} \leftarrow \text{PC}$$
$$\text{REM} \leftarrow \text{RDM}$$
- O único registrador que recebe dados de duas fontes é o REM
- Para solucionar este conflito usa-se um multiplexador

O Computador Neander

► Organização final



O Computador Neander

► A Organização: sinais de controle para cada transferência

Transferência	Sinais de controle
$REM \leftarrow PC$	sel=0, cargaREM
$PC \leftarrow PC + 1$	incrementaPC
$RI \leftarrow RDM$	cargaRI
$REM \leftarrow RDM$	sel=1, cargaREM
$RDM \leftarrow AC$	cargaRDM
$AC \leftarrow RDM$; atualiza N e Z	selUAL(Y), cargaAC, cargaNZ
$AC \leftarrow AC + RDM$; atualiza N e Z	selUAL(ADD), cargaAC, cargaNZ
$AC \leftarrow AC \text{ AND } RDM$; atualiza N e Z	selUAL(AND), cargaAC, cargaNZ
$AC \leftarrow AC \text{ OR } RDM$; atualiza N e Z	selUAL(OR), cargaAC, cargaNZ
$AC \leftarrow \text{NOT}(AC)$; atualiza N e Z	selUAL(NOT), cargaAC, cargaNZ
$PC \leftarrow RDM$	cargaPC

O Computador Neander

Temporização dos sinais de controle (parte 1)

tempo	STA	LDA	ADD	OR	AND	NOT
t0	sel=0, carga REM	sel=0, carga REM	sel=0, carga REM	sel=0, carga REM	sel=0, carga REM	sel=0, carga REM
t1	Read, incrementa PC	Read, incrementa PC	Read, incrementa PC	Read, incrementa PC	Read, incrementa PC	Read, incrementa PC
t2	carga RI	carga RI	carga RI	carga RI	carga RI	carga RI
t3	sel=0, carga REM	sel=0, carga REM	sel=0, carga REM	sel=0, carga REM	sel=0, carga REM	UAL(NOT), carga AC, carga NZ, goto t0
t4	Read, incrementa PC	Read, incrementa PC	Read, incrementa PC	Read, incrementa PC	Read, incrementa PC	
t5	sel=1, carga REM	sel=1, carga REM	sel=1, carga REM	sel=1, carga REM	sel=1, carga REM	
t6	carga RDM	Read	Read	Read	Read	
t7	Write, goto t0	UAL(Y), carga AC, carga NZ, goto t0	UAL(ADD), carga AC, carga NZ, goto t0	UAL(OR), carga AC, carga NZ, goto t0	UAL(AND, carga AC, carga NZ, goto t0	

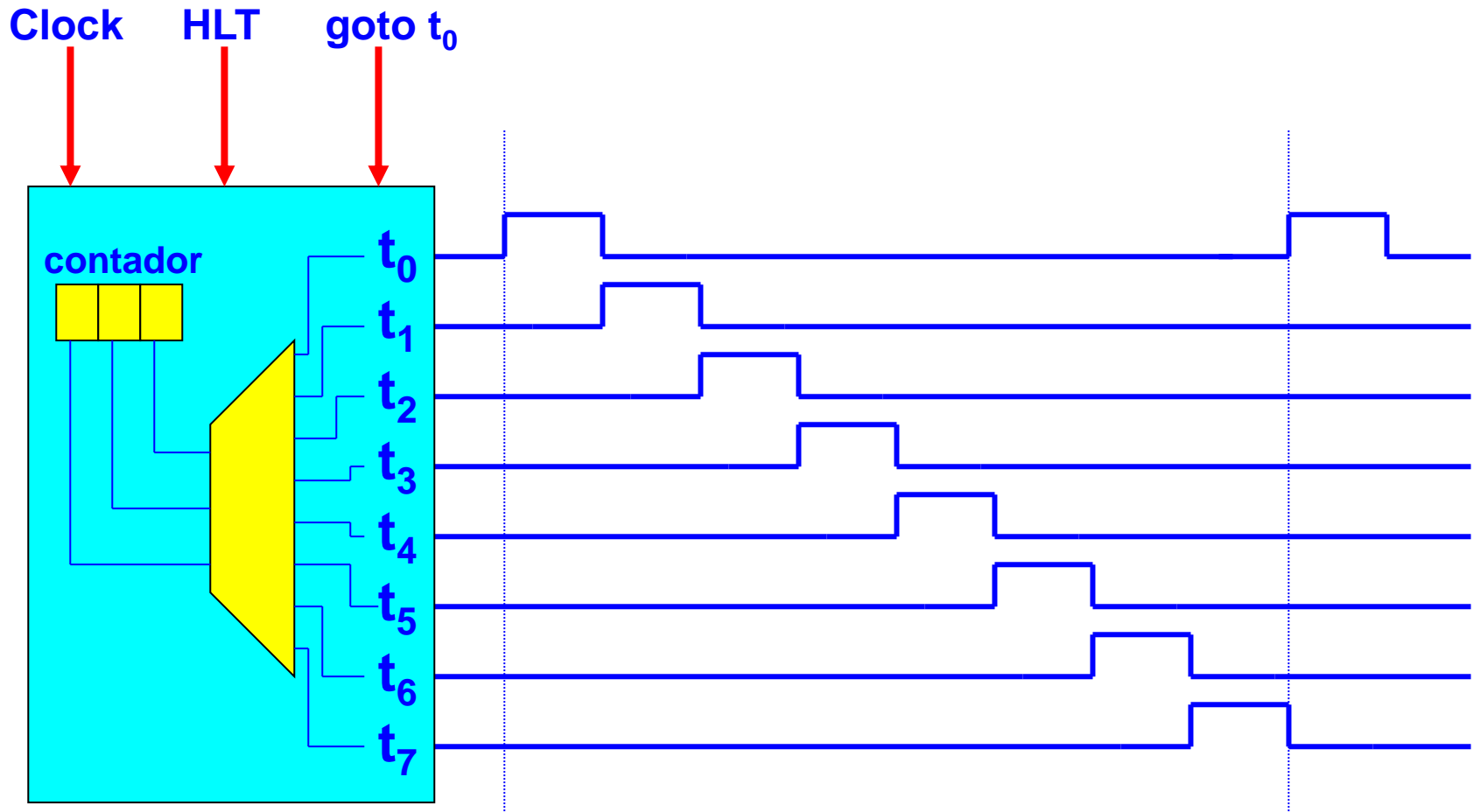
O Computador Neander

Temporização dos sinais de controle (parte 2)

tempo	JMP	JN, N=1	JN, N=0	JZ, Z=1	JZ, Z=0	NOP	HLT
t0	sel=0, carga REM	sel=0, carga REM	sel=0, carga REM	sel=0, carga REM	sel=0, carga REM	sel=0, carga REM	sel=0, carga REM
t1	Read, incrementa PC	Read, incrementa PC	Read, incrementa PC	Read, incrementa PC	Read, incrementa PC	Read, incrementa PC	Read, incrementa PC
t2	carga RI	carga RI	carga RI	carga RI	carga RI	carga RI	carga RI
t3	sel=0, carga REM	sel=0, carga REM	incrementa PC, goto t0	sel=0, carga REM	incrementa PC, goto t0	goto t0	Halt
t4	Read	Read		Read			
t5	carga PC, goto t0	carga PC, goto t0		carga PC, goto t0			
t6							
t7							

O Computador Neander

Gerador dos sinais de temporização



O Computador Neander

Expressões booleanas dos sinais de controle

carga REM = $t_0 + t_3.(STA+LDA+ADD+OR+AND+JMP+JN.N+JZ.Z) + t_5.(STA+LDA+ADD+OR+AND)$

incrementa PC = $t_1 + t_4.(STA+LDA+ADD+OR+AND) + t_3.(JN.N' + JZ.Z')$

carga RI = t_2

sel = $t_5.(STA+LDA+ADD+OR+AND)$

carga RDM = $t_6.STA$

Read = $t_1 + t_4.(STA+LDA+ADD+OR+AND+JMP+JN.N+JZ.Z) + t_6.(LDA+ADD+OR+AND)$

Write = $t_7.STA$

UAL(Y) = $t_7.LDA$

UAL(ADD) = $t_7.ADD$

UAL(OR) = $t_7.OR$

UAL(AND) = $t_7.AND$

UAL(NOT) = $t_3.NOT$

carga AC = $t_7.(LDA+ADD+OR+AND) + t_3.NOT$

carga NZ = $t_7.(LDA+ADD+OR+AND) + t_3.NOT = \text{carga AC}$

carga PC = $t_5.(JMP+JN.N+JZ.Z)$

goto t0 = $t_7.(STA+LDA+ADD+OR+AND) + t_3.(NOP+NOT+JN.N'+JZ.Z') + t_5.(JMP+JN.N+JZ.Z)$