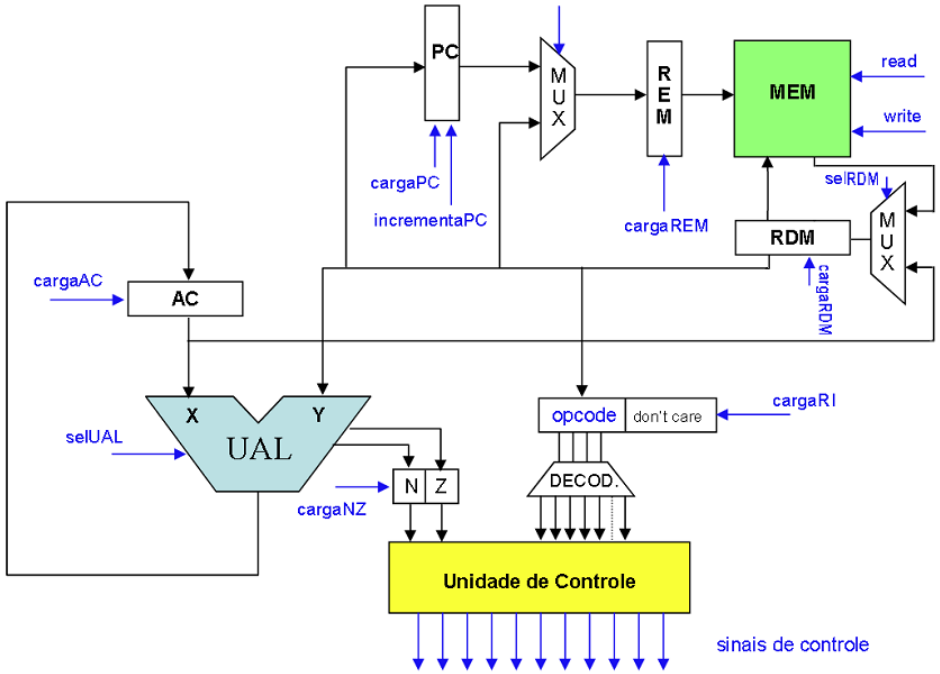
Organização e Arquitetura de Computadores

Arquitetura Neander

Lúcio Renê Prade
Rodrigo Marques de Figueiredo

Processador Neander

- Trata-se de um **protoprocessador** (processador simples/primitivo) utilizado para fins acadêmicos;
- Projetado para foco em sua arquitetura de processamento e não de interface.



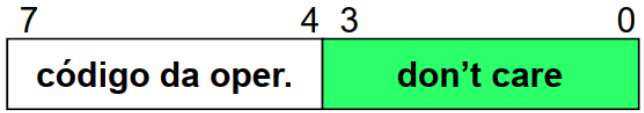
Processador Neander - Projeto

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

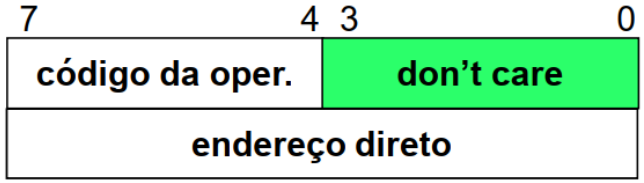
Processador Neander - Projeto

As instruções do Neander possuem um ou dois bytes (ocupam uma ou duas posições de memória)

Instruções com um byte:
NOP, NOT



Instruções com dois bytes:
STA, LDA, ADD, OR, AND, JMP, JN, JZ



Processador Neander - Projeto

- **Largura de dados e endereços de 8 bits**
- **Dados representados em complemento de 2**
- **1 acumulador de 8 bits (AC)**
- **1 apontador de programa de 8 bits (PC)**
- **1 registrador de estado com 2 códigos de condição: negativo (N) e zero (Z)**
- **Um registrador de 8 bits para servir de acumulador**
- **Um registrador de 8 bits para o PC (registrador-contador)**
- **Dois flip-flops: um para o código de condição N e outro para Z**
- **Uma memória de 256 posições (endereços) x 8 bits**

Processador Neander - Premissas

Ciclo de *fetch*

Ciclo de Execução

A fase de busca: é igual para todas as instruções

RI ← MEM(PC)

PC ← PC + 1

- **Novo elemento é necessário: o registrador de instrução (RI)**
- **MEM(PC) corresponde a um acesso à memória, usando o conteúdo do PC como fonte do endereço**

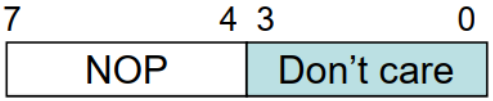
Processador Neander - Premissas

Ciclo de Execução

Instrução NOP (nenhuma operação)

Simbólico: NOP

RT: -



Passos no nível RT:

Busca:	**RI ← MEM(PC)**
	PC ← PC + 1
Execução:	**nenhuma operação**

As transferências indicam quais caminhos de dados devem existir, mas não indicam os caminhos físicos reais entre os elementos (registradores e ULA)

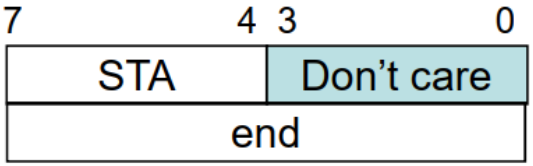
Processador Neander - Premissas

Ciclo de Execução

Instrução STA (armazena acumulador)

Simbólico: STA end

RT: MEM(end) ← AC



Passos no nível RT:

Busca:	RI ← MEM(PC)
	PC ← PC + 1
Execução:	end ← MEM(PC)
	PC ← PC + 1
	MEM(end) ← AC

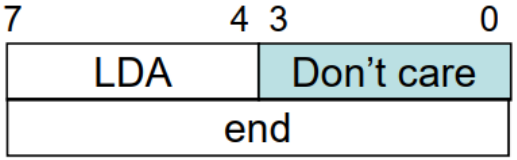
Processador Neander - Premissas

Ciclo de Execução

Instrução LDA (carrega acumulador)

Simbólico: LDA end

RT: AC ← MEM(end)



Passos no nível RT:

Busca:	**RI ← MEM(PC)**
	PC ← PC + 1
Execução:	**end ← MEM(PC)**
	PC ← PC + 1
	AC ← MEM(end); atualiza N e Z

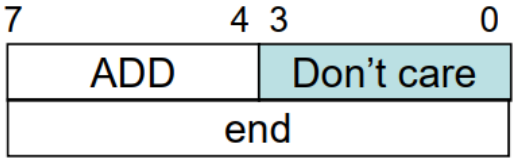
Processador Neander - Premissas

Ciclo de Execução

Instrução ADD (soma)

Simbólico: ADD end

RT: AC ← MEM(end) + AC



Passos no nível RT:

Busca:	RI ← MEM(PC)
	PC ← PC + 1
Execução:	end ← MEM(PC)
	PC ← PC + 1
	AC ← AC + MEM(end); atualiza N e Z

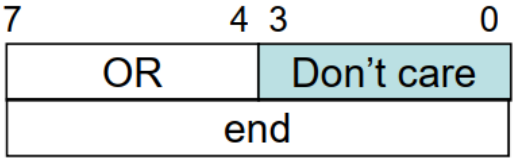
Processador Neander - Premissas

Ciclo de Execução

Instrução OR ("ou" lógico, bit a bit)

Simbólico: OR end

RT: AC ← MEM(end) OR AC



Passos no nível RT:

Busca:	RI ← MEM(PC)
	PC ← PC + 1
Execução:	end ← MEM(PC)
	PC ← PC + 1
	AC ← AC OR MEM(end); atualiza N e Z

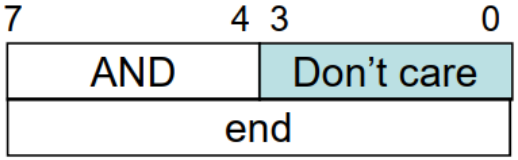
Processador Neander - Premissas

Ciclo de Execução

Instrução AND ("e" lógico, bit a bit)

Simbólico: AND end

RT: AC ← MEM(end) AND AC



Passos no nível RT:

Busca:	RI ← MEM(PC)
	PC ← PC + 1
Execução:	end ← MEM(PC)
	PC ← PC + 1
	AC ← AC AND MEM(end); atualiza N e Z

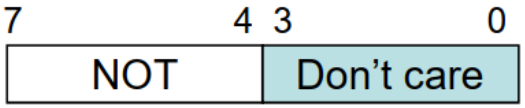
Processador Neander - Premissas

Ciclo de Execução

Instrução NOT (complementa acumulador)

Simbólico: NOT

RT: AC ← NOT AC



Passos no nível RT:

Busca:	RI ← MEM(PC)
	PC ← PC + 1
Execução:	AC ← NOT(AC); atualiza N e Z

Processador Neander - Premissas

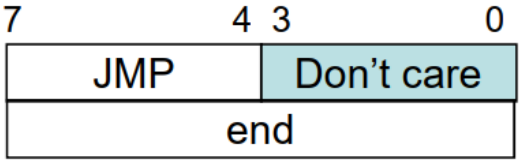
Ciclo de Execução

Instrução JMP (desvio incondicional - jump)

Simbólico: JMP end

RT: PC ← end

Passos no nível RT:



Busca:	RI ← MEM(PC)
	PC ← PC + 1
Execução:	end ← MEM(PC)
	PC ← end

Processador Neander - Premissas

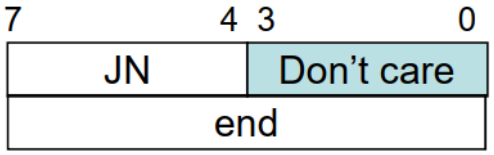
Ciclo de Execução

Instrução JN (desvio condicional - jump on negative)

Simbólico: JN end

RT: IF N = 1 THEN PC ← end

Passos no nível RT:



Se N=1 (desvio ocorre)

Busca:	RI ← MEM(PC)
	PC ← PC + 1
Execução:	end ← MEM(PC)
	PC ← end

Se N=0 (desvio não ocorre)

Busca:	RI ← MEM(PC)
	PC ← PC + 1
Execução:	end ← MEM(PC)
	PC ← PC + 1

a rigor, desnecessário

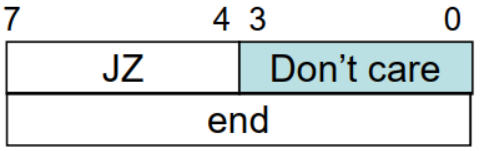
Processador Neander - Premissas

Ciclo de Execução

Instrução JZ (desvio condicional - jump on zero)

Simbólico: JZ end

RT: IF Z = 1 THEN PC ← end



Passos no nível RT:

Se Z=1 (desvio ocorre)

Busca:	RI ← MEM(PC)
	PC ← PC + 1
Execução:	end ← MEM(PC)
	PC ← end

Se Z=0 (desvio não ocorre)

Busca:	RI ← MEM(PC)
	PC ← PC + 1
Execução:	end ← MEM(PC)
	PC ← PC + 1

a rigor, desnecessário

Processador Neander - Premissas

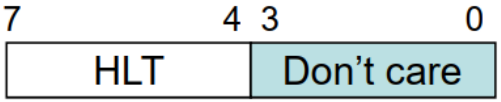
Ciclo de Execução

Instrução HLT (término de execução - halt)

Simbólico: HLT

RT: --

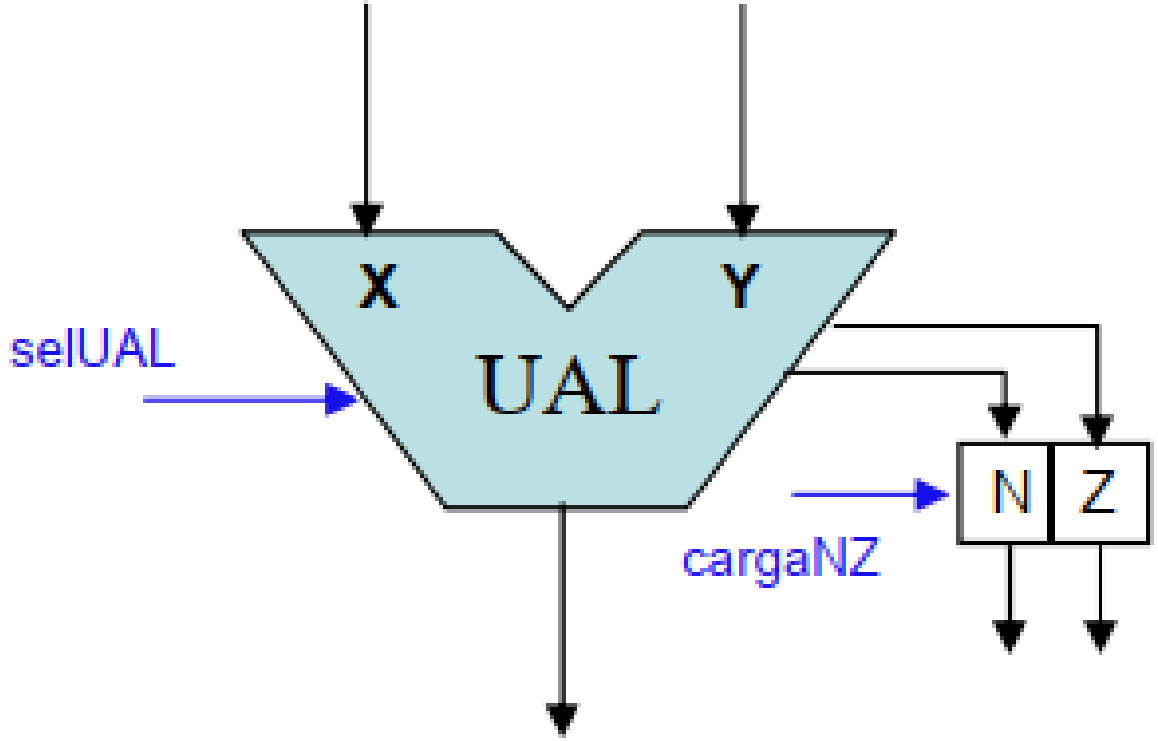
Passos no nível RT:



Busca:	**RI ← MEM(PC)**
	PC ← PC + 1
Execução:	**parar o processamento**

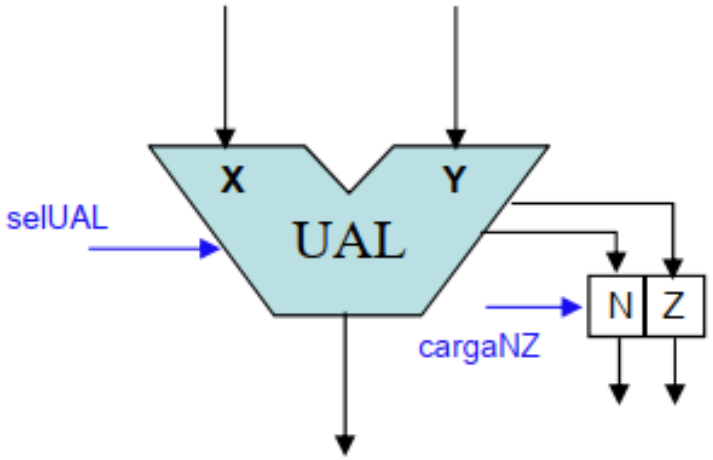
Processador Neander – Organização da Arquitetura

ULA – Unidade Lógica Aritmética



Processador Neander – Organização da Arquitetura

UAL – Unidade Aritmética e Lógica



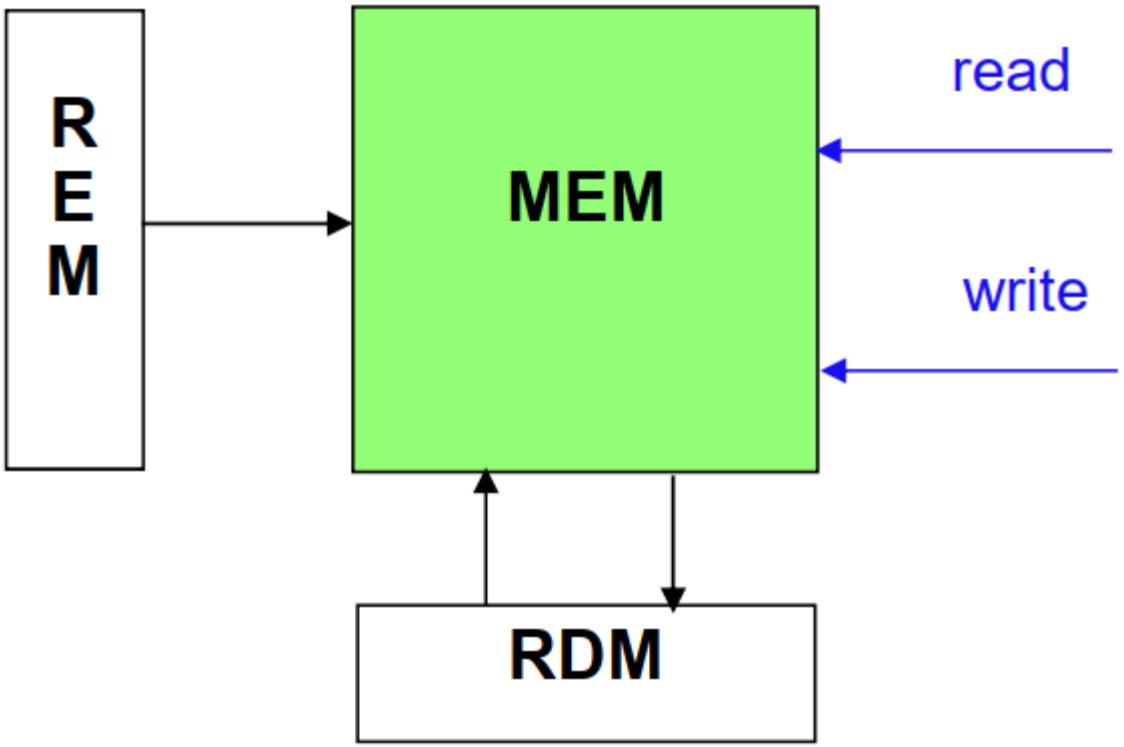
Associados à UAL:

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

Flip-Flops devem ter sinal de carga

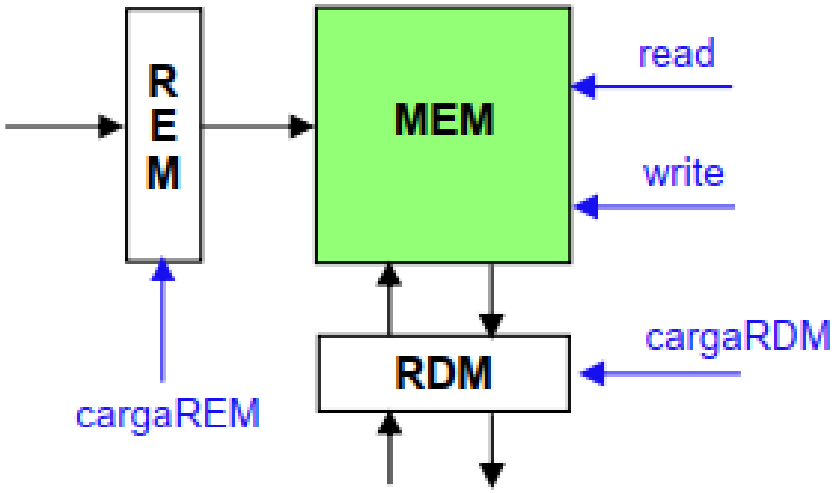
Processador Neander – Organização da Arquitetura

Sistema de Memória



Processador Neander – Organização da Arquitetura

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

Processador Neander – Organização da Arquitetura

Sistema de Memória

Operações com a memória

x ← MEM(y) descreve uma leitura da memória, que é realizada pelos seguintes passos:

1. REM ← y	copia y (que é um endereço) para o REM
2. Read	ativação de uma operação de leitura da memória
3. x ← RDM	copia o conteúdo de RDM para x

- REM é o registrador de endereços da memória
- RDM é o registrador de dados da memória

Processador Neander – Organização da Arquitetura

Sistema de Memória

Operações com a memória

MEM(y) ← x descreve uma escrita da memória, que é realizada pelos seguintes passos:

1. REM ← y	copia y (que é um endereço) para o REM
2. RDM ← x	copia x (que é um dado) para o RDM
3. write	ativação de uma operação de escrita na memória

Processador Neander – Organização da Arquitetura

Sistema de Memória

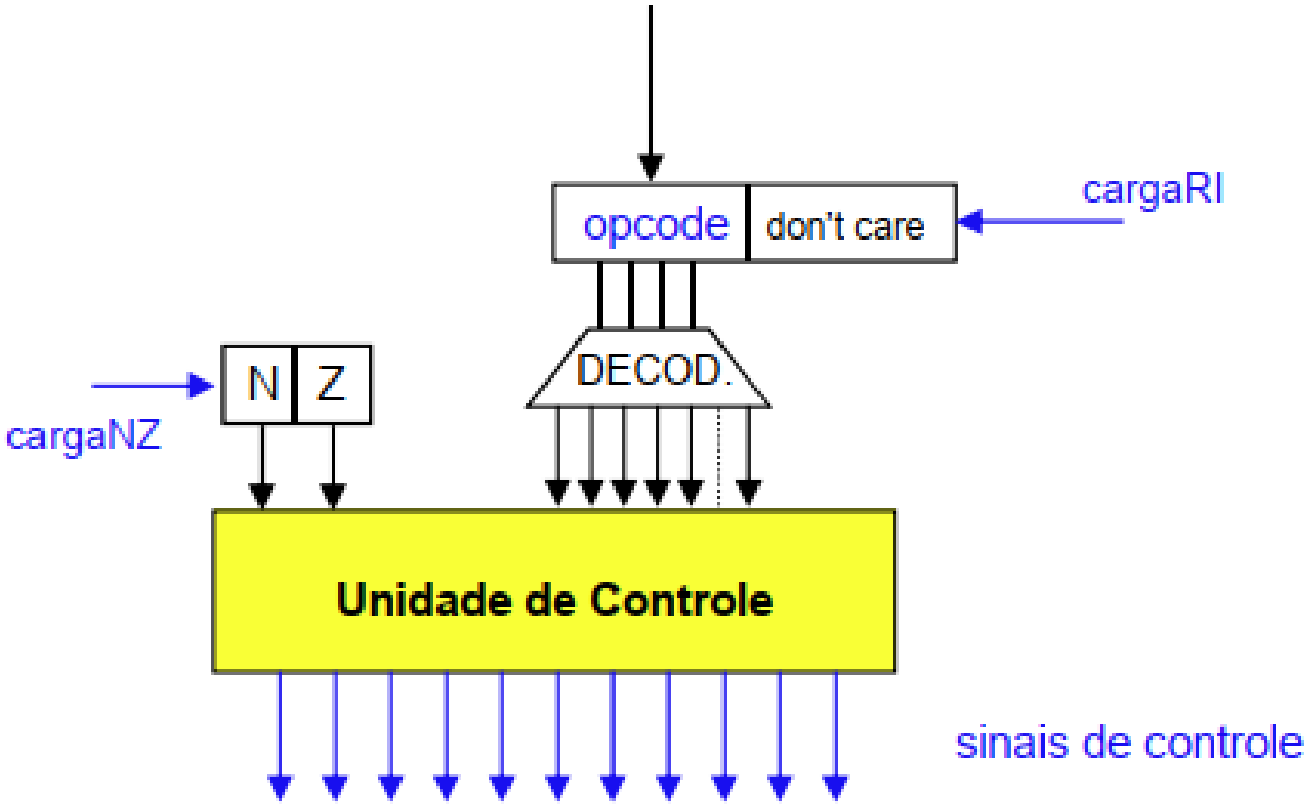
Operações com a memória

Observações

- Após a leitura do PC, seu conteúdo deve ser incrementado, para apontar para a próxima posição
- O incremento do PC pode ser feito a qualquer instante após a transferência do PC para o REM
- O incremento do PC pode ser feito em paralelo com outras operações
- Um desvio condicional que não se realiza não necessita ler o valor do endereço de desvio
- Ou seja, basta incrementar o PC

Processador Neander – Organização da Arquitetura

Registrador de Instrução

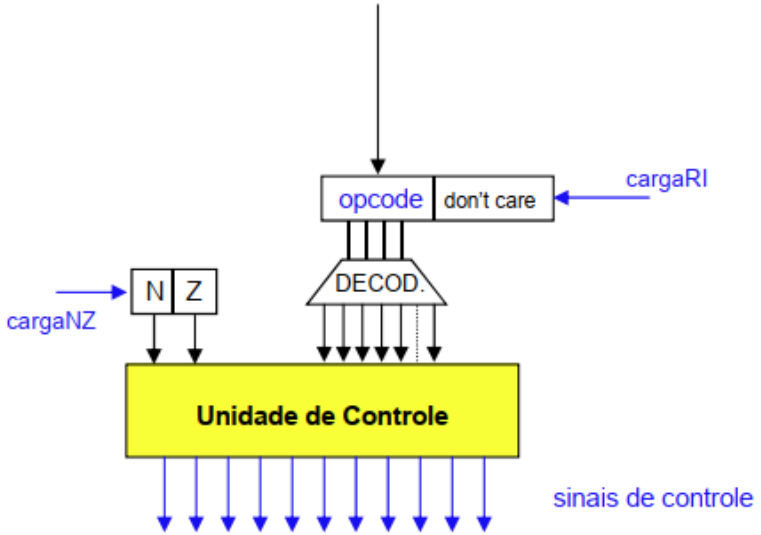


Processador Neander – Organização da Arquitetura

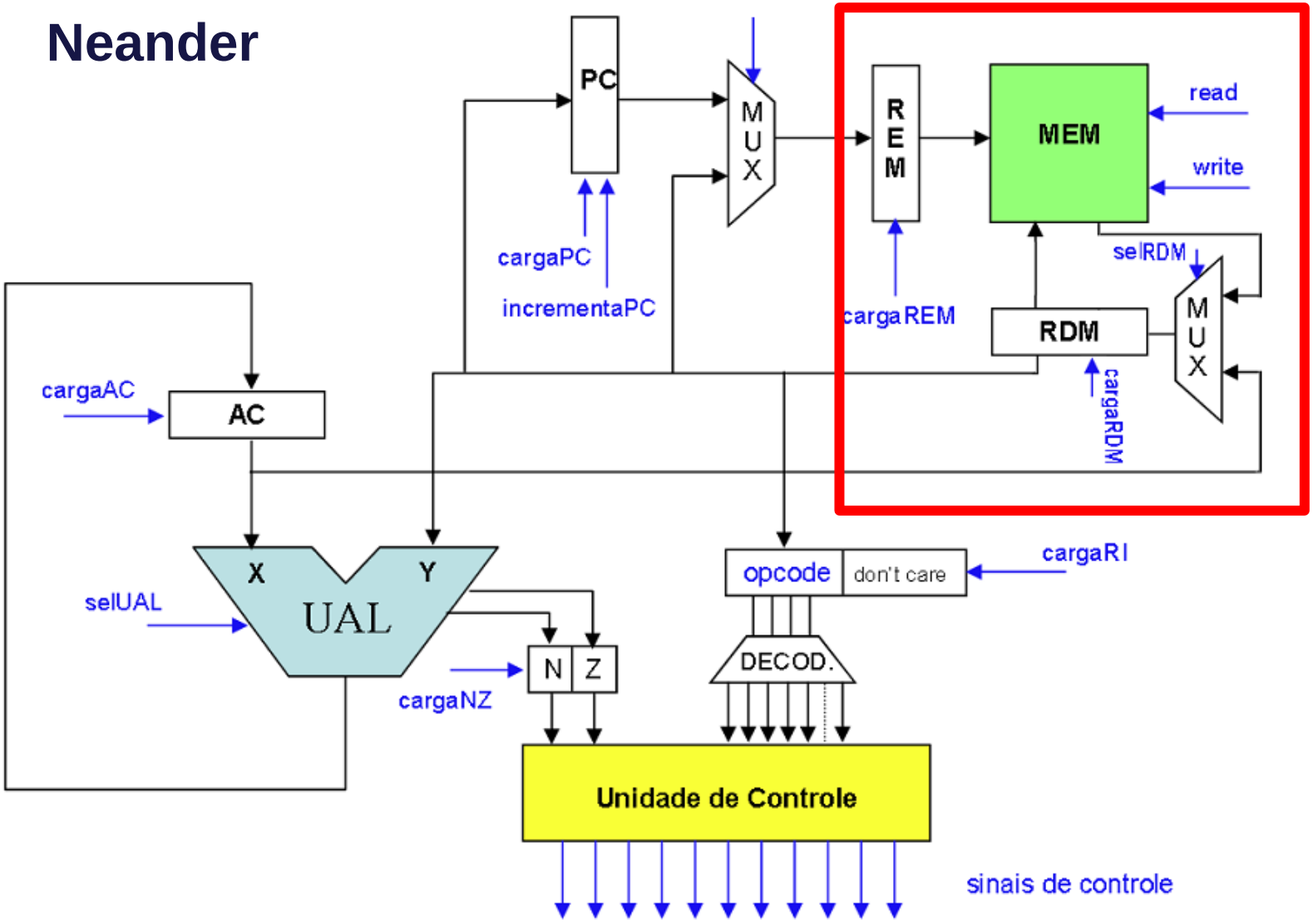
Registrador de Instrução

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

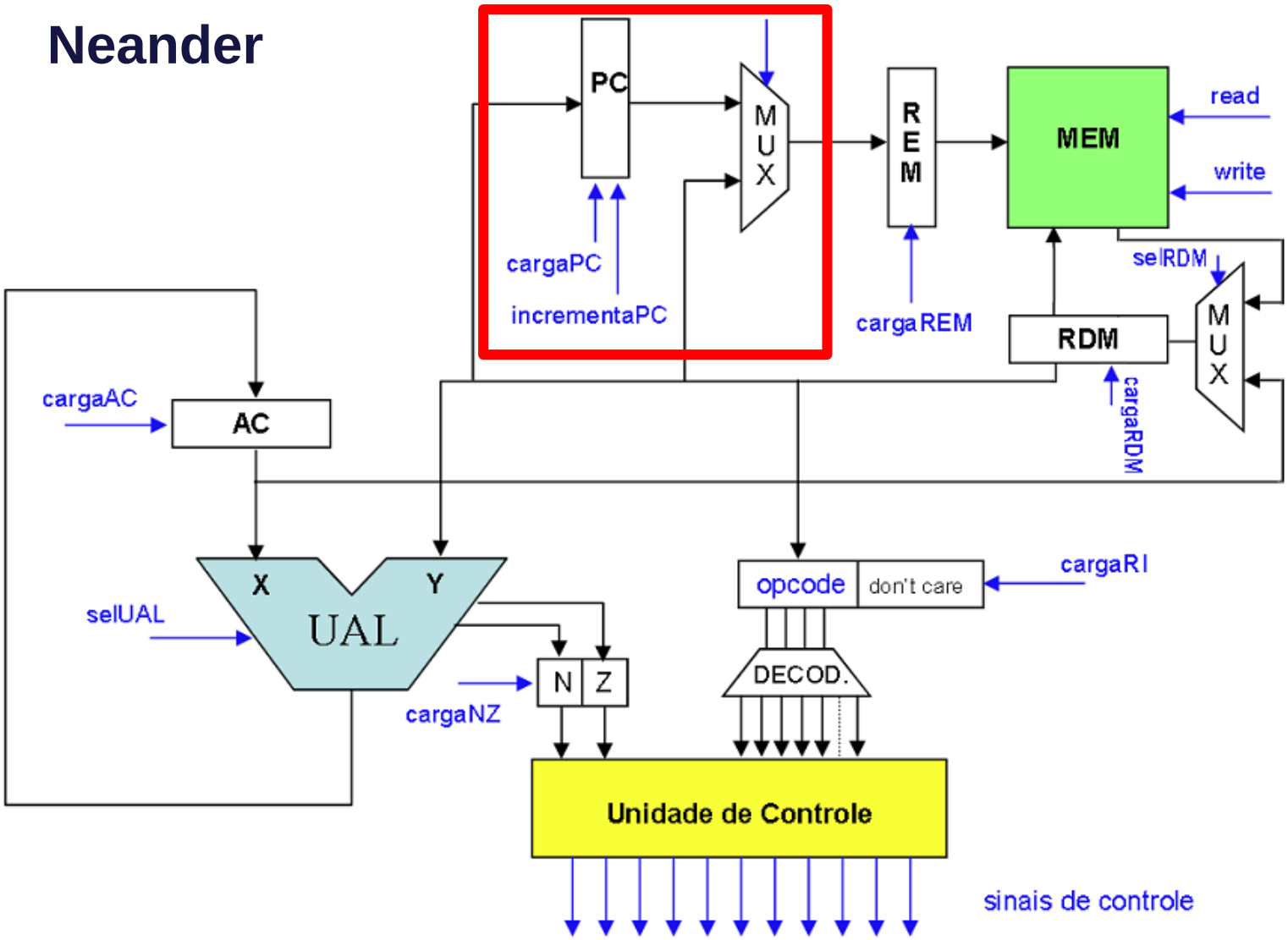
- **Decodificador (4 bits para 16 instruções)**
- **sinais de condição (N,Z) (para JN e JZ)**
- **registrador deve ter sinal de carga**



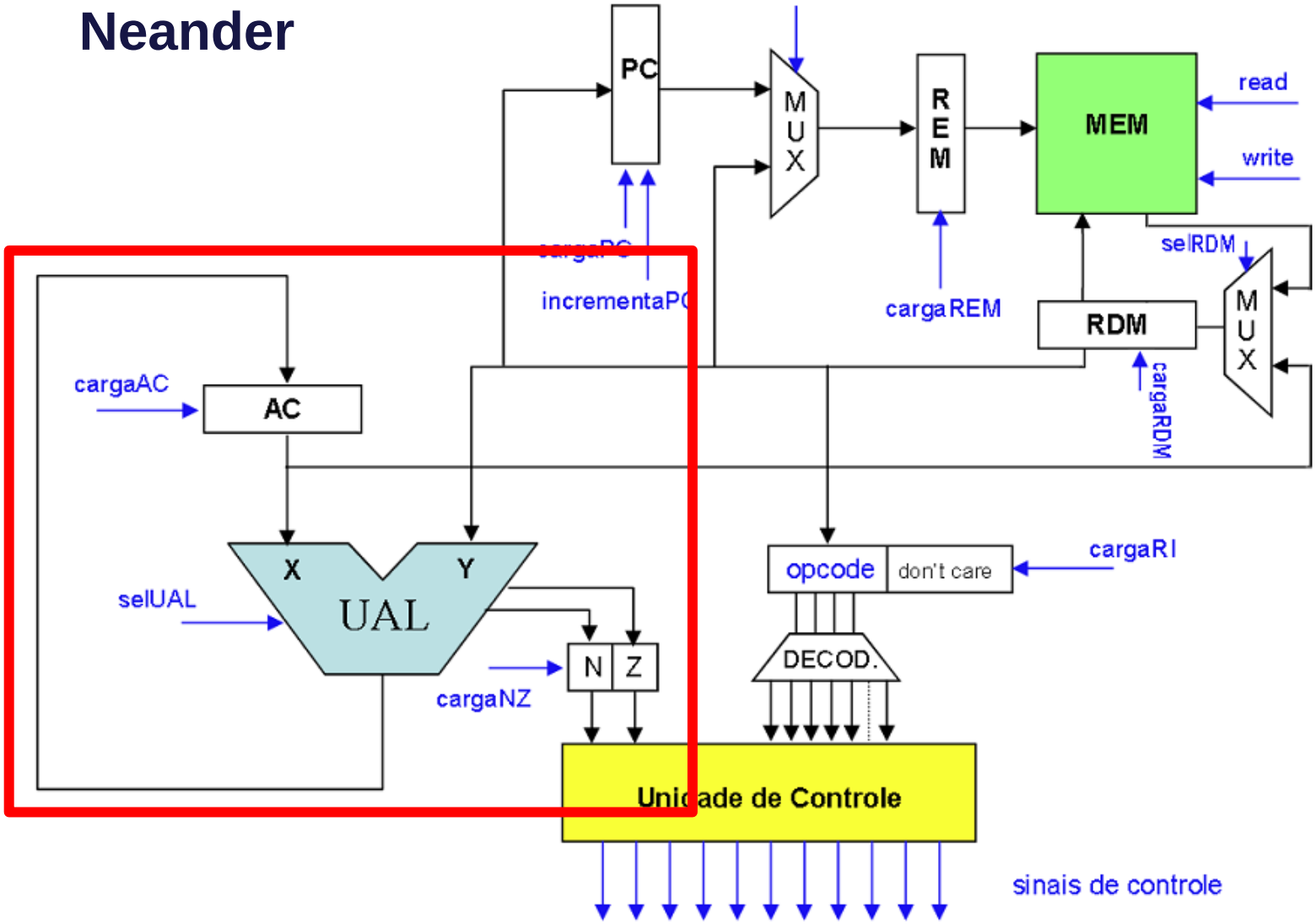
Neander


PC
MUX
REM
MEM
read
write
selRDM
MUX
RDM
cargaRDM
cargaPC
incrementaPC
cargaREM
cargaAC
AC
X
Y
UAL
selUAL
N
Z
cargaNZ
opcode
don't care
cargaRI
DECOD.
Unidade de Controle
sinais de controle

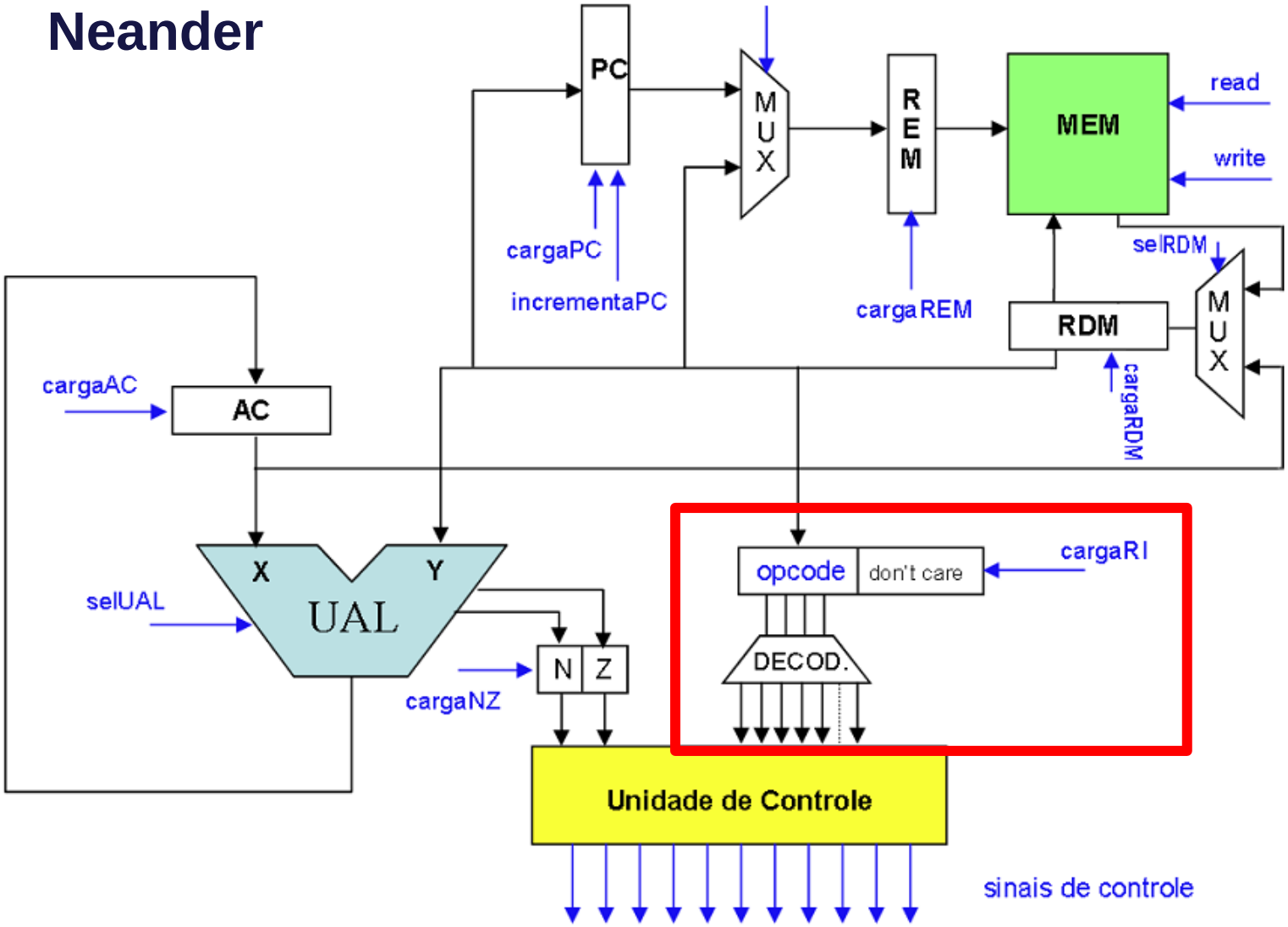
Neander


PC
MUX
REM
MEM
read
write
cargaPC
incrementaPC
cargaREM
selRDM
RDM
MUX
cargaRDM
cargaAC
AC
X
Y
UAL
selUAL
N
Z
cargaNZ
opcode
don't care
cargaRI
DECOD.
Unidade de Controle
sinais de controle

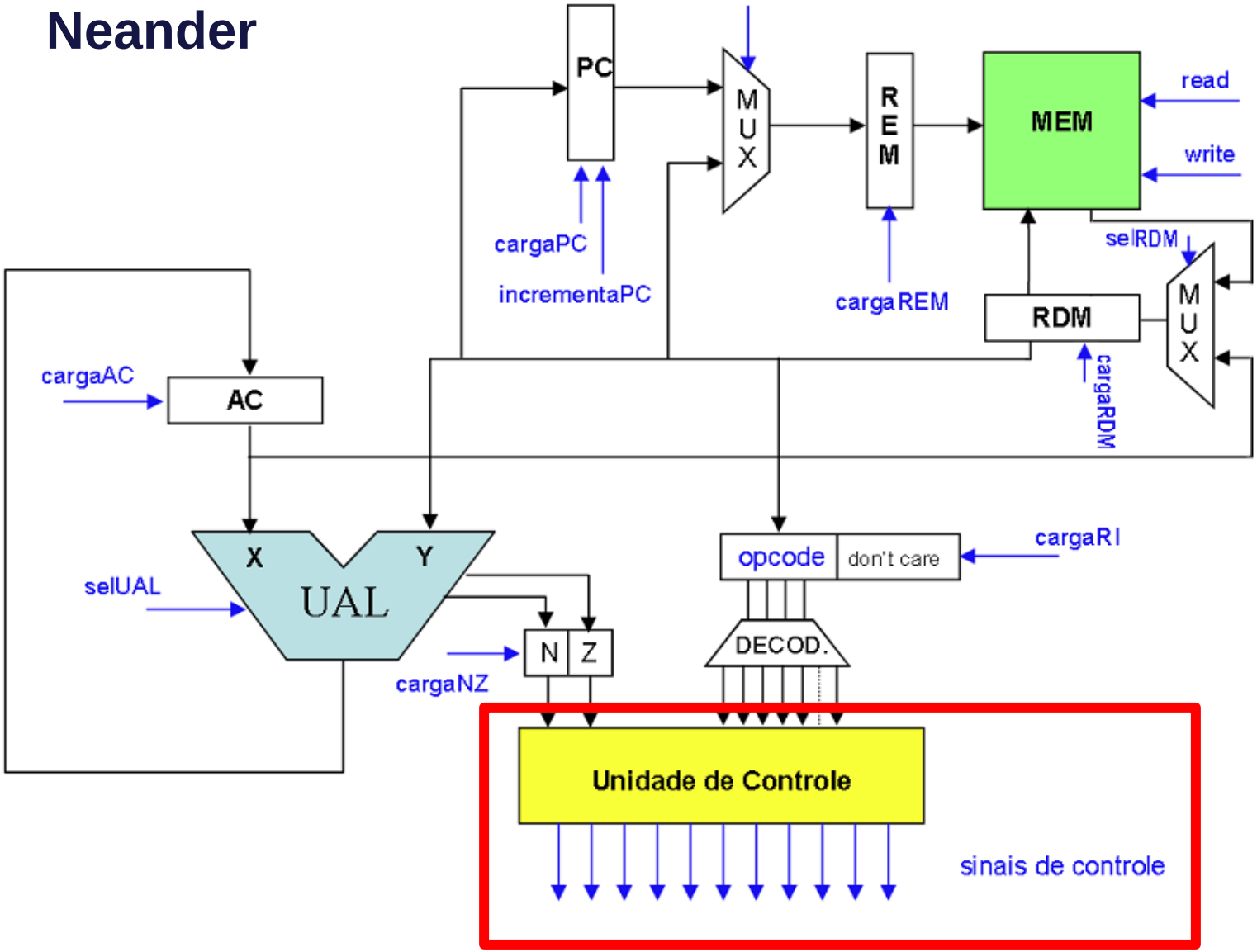
Neander


PC
MUX
REM
MEM
read
write
selRDM
MUX
RDM
cargaREM
cargaRDM
incrementaPC
cargaAC
AC
X
Y
UAL
selUAL
N
Z
cargaNZ
opcode
don't care
cargaRI
DECOD.
Unidade de Controle
sinais de controle

Neander


PC
MUX
REM
MEM
read
write
cargaPC
incrementaPC
cargaREM
selRDM
RDM
MUX
cargaRDM
cargaAC
AC
X
Y
UAL
selUAL
N
Z
cargaNZ
opcode
don't care
cargaRI
DECOD.
Unidade de Controle
sinais de controle

Neander


PC
MUX
REM
MEM
read
write
cargaPC
incrementaPC
cargaREM
selRDM
RDM
MUX
cargaRDM
cargaAC
AC
X
Y
UAL
selUAL
N
Z
cargaNZ
opcode
don't care
cargaRI
DECOD.
Unidade de Controle
sinais de controle

Neander: Sinais de Controle para cada Transferência

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

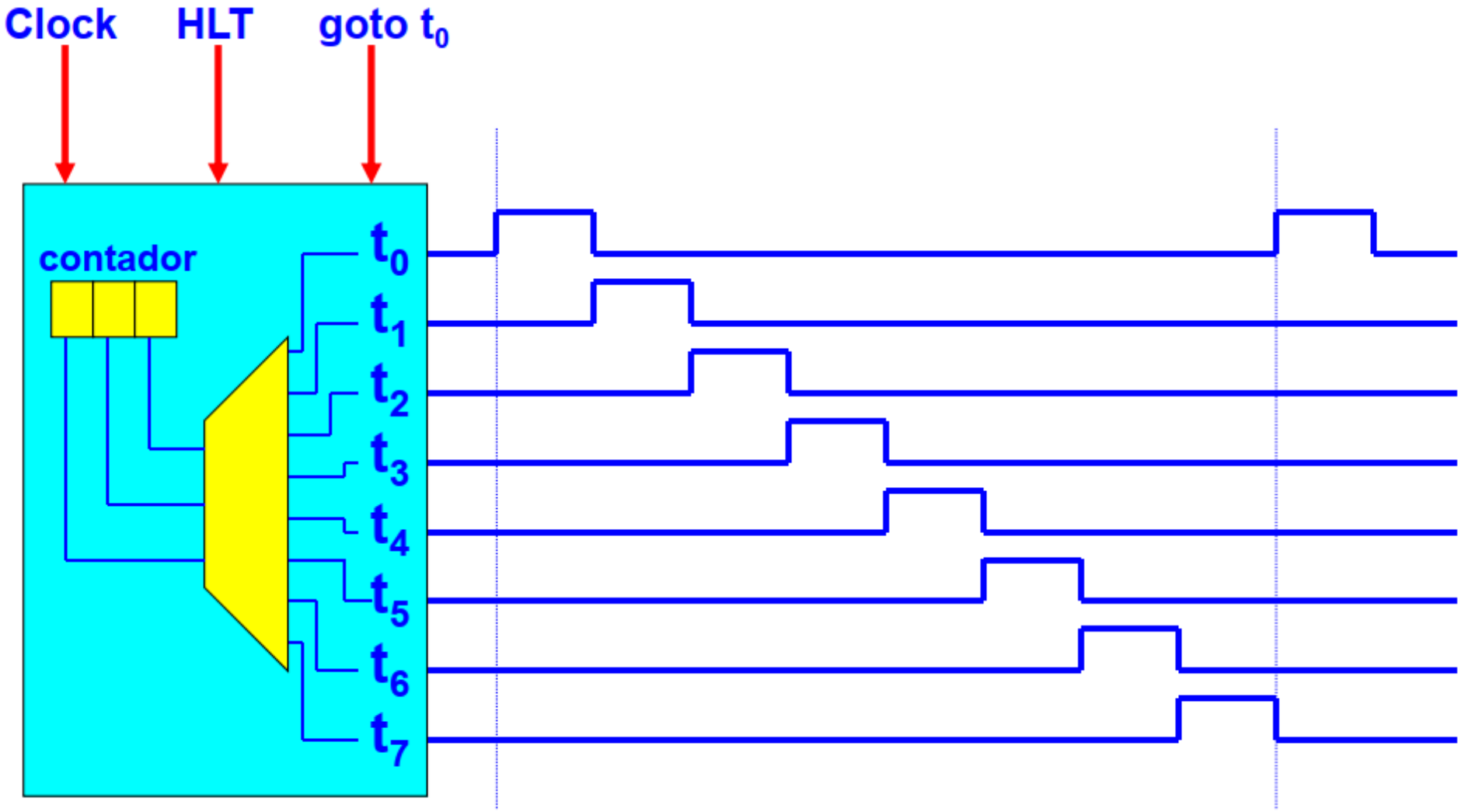
Neander: Temporização dos Sinais de Controle

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	

Neander: Temporização dos Sinais de Controle

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							

Neander: Temporização dos Sinais de Controle



Neander: Expressões Booleana dos Sinais de Controle

carga REM = t0 + t3.(STA+LDA+ADD+OR+AND+JMP+JN.N+JZ.Z + t5.(STA+LDA+ADD+OR+AND)

incrementa PC = t1 + t4.(STA+LDA+ADD+OR+AND) + t3.(JN.N' + JZ.Z')

carga RI = t2

sel = t5.(STA+LDA+ADD+OR+AND)

carga RDM = t6.STA

Read = t1 + t4.(STA+LDA+ADD+OR+AND+JMP+JN.N+JZ.Z) + t6.(LDA+ADD+OR+AND)

Write = t7.STA

UAL(Y) = t7.LDA

UAL(ADD) = t7.ADD

UAL(OR) = t7.OR

UAL(AND) = t7.AND

UAL(NOT) = t3.NOT

carga AC = t7.(LDA+ADD+OR+AND) + t3.NOT

carga NZ = t7.(LDA+ADD+OR+AND) + t3.NOT = carga AC

carga PC = t5.(JMP+JN.N+JZ.Z)

goto t0 = t7.(STA+LDA+ADD+OR+AND) + t3.(NOP+NOT+JN.N'+JZ.Z') + t5.(JMP+JN.N+JZ.Z)