

Organização do Neander

Disciplina: Introdução à Arquitetura de Computadores

Luciano Moraes Da Luz Brum

Universidade Federal do Pampa – Unipampa – Campus Bagé

Email: lucianobrum18@gmail.com

- Largura de dados e endereços de 8 bits;
- Dados representados em complemento de dois;
- 1 acumulador de 8 bits (AC);
- 1 contador de programa de 8 bits (PC);
- 1 registrador de estado com 2 códigos de condição: negativo (N) e zero (Z);

➤ Modos de endereçamento: Direto.

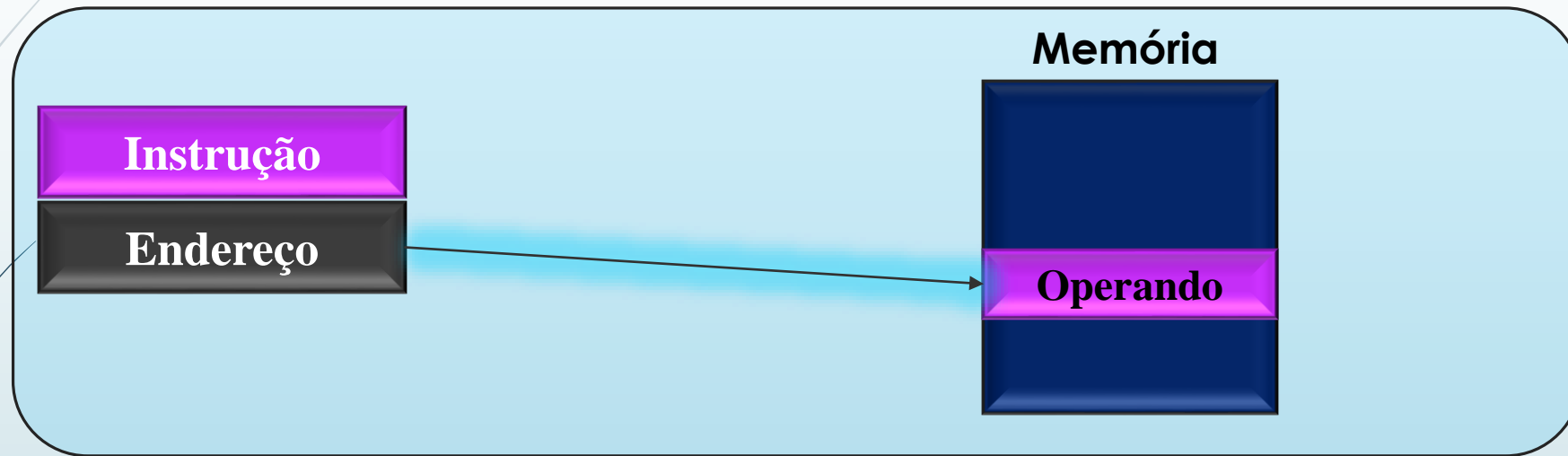


Figura 1: Endereçamento Direto. Fonte: Adaptado de Weber, 2001.

- Obs: em instruções de desvio, o endereço contido na instrução é a posição de memória onde está a instrução a ser executada.

Código	Instrução	Comentário
0000	NOP	nenhuma operação
0001	STA end	armazena acumulador - (store)
0010	LDA end	carrega acumulador - (load)
0011	ADD end	soma
0100	OR end	“ou” lógico
0101	AND end	“e” lógico
0110	NOT	inverte (complementa) acumulador
1000	JMP end	desvio incondicional - (jump)
1001	JN end	desvio condicional - (jump on negative)
1010	JZ end	desvio condicional - (jump on zero)
1111	HLT	término de execução - (halt)

Tabela 1: Conjunto de instruções do Neander. Fonte: Weber, 2001.

Instrução	Comentário
NOP	nenhuma operação
STA end	$\text{MEM}(\text{end}) \leftarrow \text{AC}$
LDA end	$\text{AC} \leftarrow \text{MEM}(\text{end})$
ADD end	$\text{AC} \leftarrow \text{MEM}(\text{end}) + \text{AC}$
OR end	$\text{AC} \leftarrow \text{MEM}(\text{end}) \text{ OR } \text{AC}$
AND end	$\text{AC} \leftarrow \text{MEM}(\text{end}) \text{ AND } \text{AC}$
NOT	$\text{AC} \leftarrow \text{NOT } \text{AC}$
JMP end	$\text{PC} \leftarrow \text{end}$
JN end	IF $\text{N}=1$ THEN $\text{PC} \leftarrow \text{end}$
JZ end	IF $\text{Z}=1$ THEN $\text{PC} \leftarrow \text{end}$

Tabela 2: Ações das instruções. Fonte: Weber, 2001.

- Utilizado pelas instruções: JN e JZ;
- N - negativo: sinal do resultado (1 se é negativo, 0 se é positivo ou nulo);
- Z – zero: indica se o resultado é zero (1 se é igual a 0, 0 se é diferente de 0);
- Apenas as instruções lógicas e aritméticas (ADD, NOT, AND, OR) e de transferência (LDA) afetam os códigos de condição N e Z;

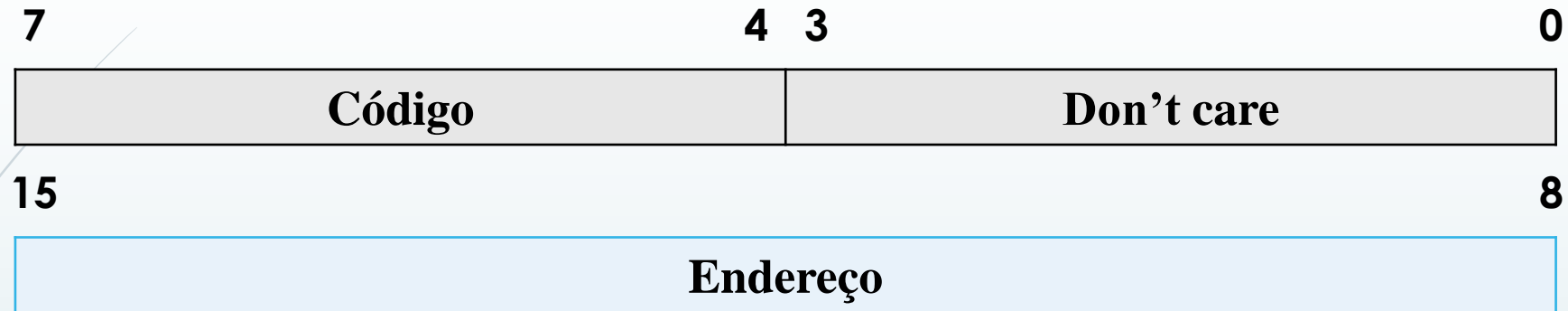


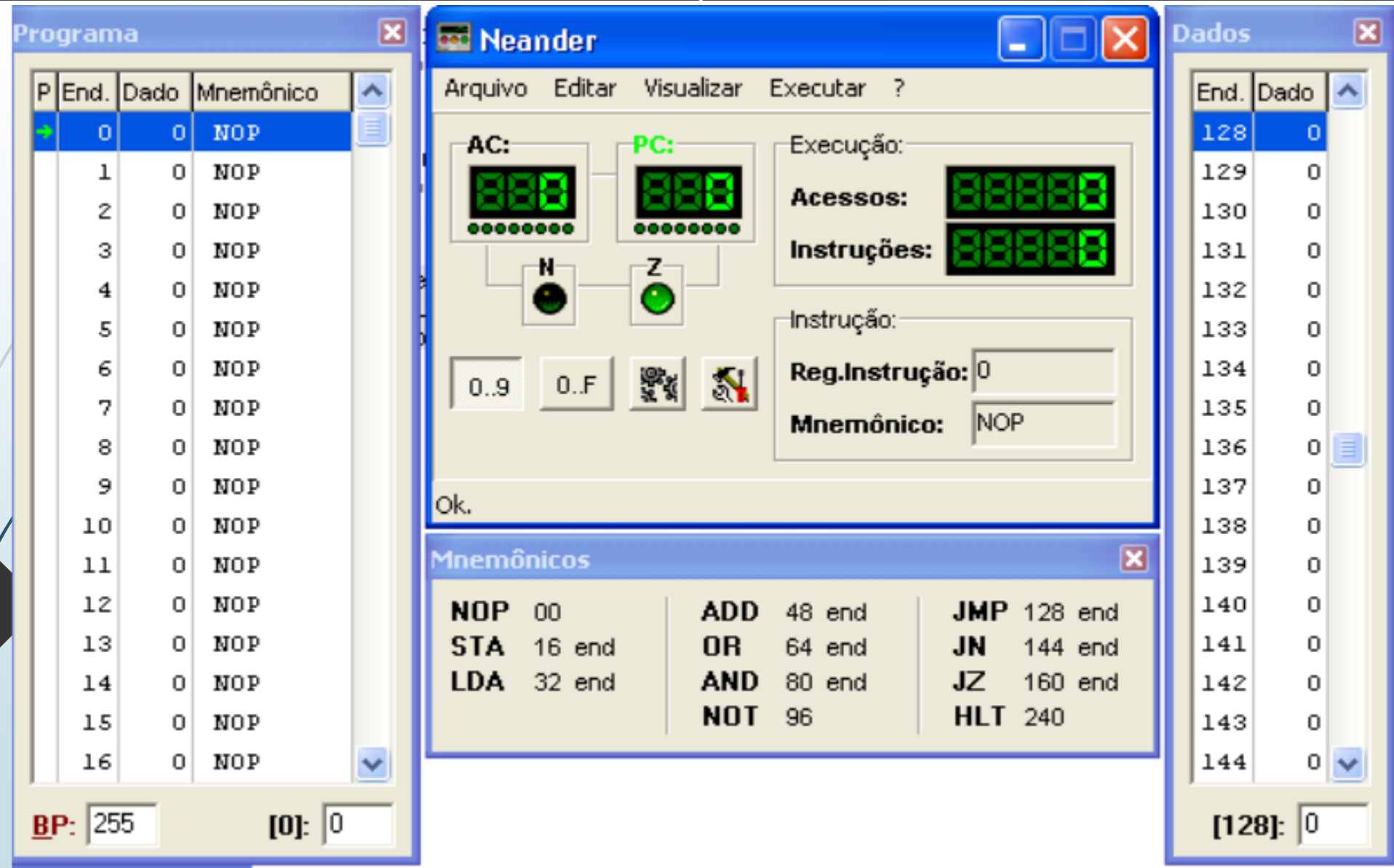
Figura 2: Formato das Instruções no Neander. Fonte: Adaptado de Weber, 2001.

- Nas instruções de 1 byte, os 4 bits mais significativos contém o código da instrução;
- Nas instruções de 2 bytes, os 4 bits mais significativos do 1º byte contém o código da instrução e o 2º byte contém um endereço;
- Instruções de 2 bytes são as que fazem referência a memória;

- A memória no Neander tem 256 posições;
- Por convenção, utiliza-se da posição 0-127 para instruções e da posição 128- 255 para dados;

Simbólico	Decimal	Comentários
LDA 128	32 128	Acumulador AC recebe conteúdo da posição 128;
ADD 129	48 129	Conteúdo do AC é somado ao conteúdo da posição 129;
ADD 130	48 130	Conteúdo do AC é somado ao conteúdo da posição 130;
STA 131	16 131	Conteúdo do AC é armazenado na posição 131 da memória;
HLT	240	Processador para, fim do programa;

Tabela 3: Exemplo de programação no Neander. Fonte: Adaptado de Weber, 2001.



The image shows a screenshot of the Neander simulator interface, which consists of three main windows: **Programa**, **Neander**, and **Dados**.

Programa Window: Displays a list of instructions in a table with columns P, End., Dado, and Mnemônico. The first instruction is at address 0, labeled NOP. The list continues up to address 16, all labeled NOP. At the bottom, there are input fields for **BP:** 255 and **[0]:** 0.

Neander Window: The central window showing the state of the processor. It includes a menu bar (Arquivo, Editar, Visualizar, Executar, ?), status indicators for **AC:** and **PC:** (both showing 000), and execution status (Acessos: 00000, Instruções: 00000). Below these are buttons for **N** and **Z** flags, and a section for the current instruction (Reg.Instrução: 0, Mnemônico: NOP). At the bottom, there is a table of mnemonics and their corresponding values.

Dados Window: Displays a list of memory addresses and their values. The first entry is at address 128, with a value of 0. The list continues up to address 144, all with a value of 0. At the bottom, there is an input field for **[128]:** 0.

Mnemonics Table:

Mnemonic	Value
NOP	00
STA	16 end
LDA	32 end
ADD	48 end
OR	64 end
AND	80 end
NOT	96
JMP	128 end
JN	144 end
JZ	160 end
HLT	240

Figura 3: Simulador Neander. Fonte: Elaborada pelo autor.

<http://courses.cs.vt.edu/~csonline/MachineArchitecture/Lessons/CPU/sumprogram.html>



<http://courses.cs.vt.edu/~csonline/MachineArchitecture/Lessons/CPU/countprogram.html>

Roteiro



- Componentes necessários.
- Fluxo de dados.
- Sinais de Controle.
- Contador de Programa (PC).
- Resumo.

- 
- Pensando nas necessidades do Neander, quais elementos ou componentes ele precisa ?

- Pensando nas necessidades do Neander, quais elementos ou componentes ele precisa ?
 - Memória de 256 posições de 8 bits cada;

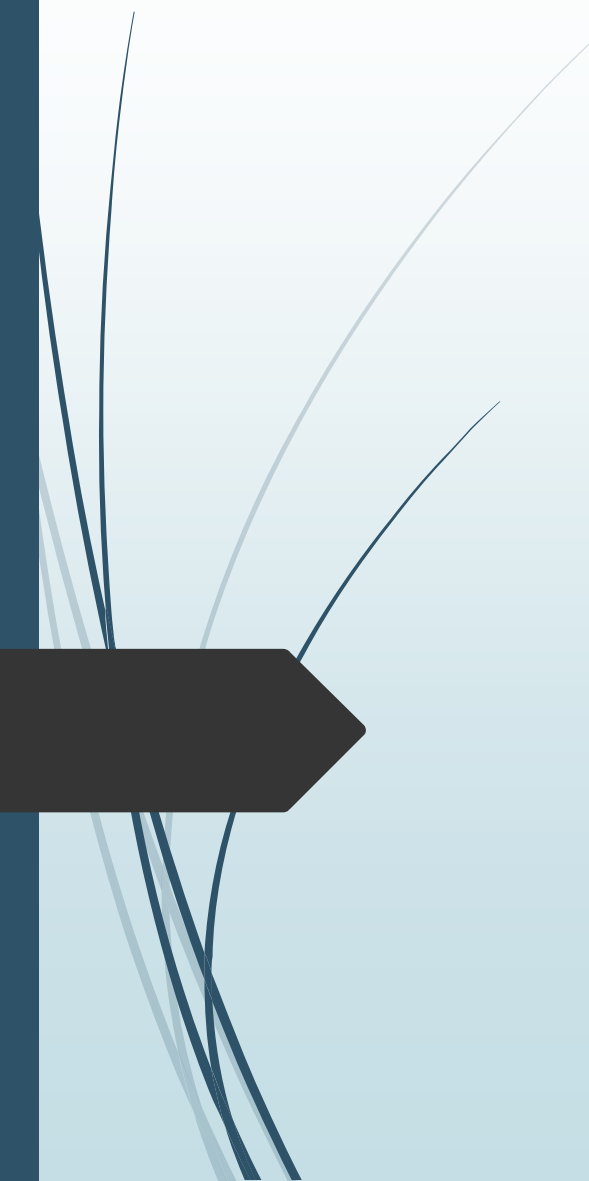
- Pensando nas necessidades do Neander, quais elementos ou componentes ele precisa ?
 - Memória de 256 posições de 8 bits cada;
 - Um registrador de 8 bits para o Acumulador;

- Pensando nas necessidades do Neander, quais elementos ou componentes ele precisa ?
 - Memória de 256 posições de 8 bits cada;
 - Um registrador de 8 bits para o Acumulador;
 - Um registrador de 8 bits para o Registrador de Instruções;

- Pensando nas necessidades do Neander, quais elementos ou componentes ele precisa ?
 - Memória de 256 posições de 8 bits cada;
 - Um registrador de 8 bits para o Acumulador;
 - Um registrador de 8 bits para o Registrador de Instruções;
 - Um registrador-contador de 8 bits para o Contador de Programa;

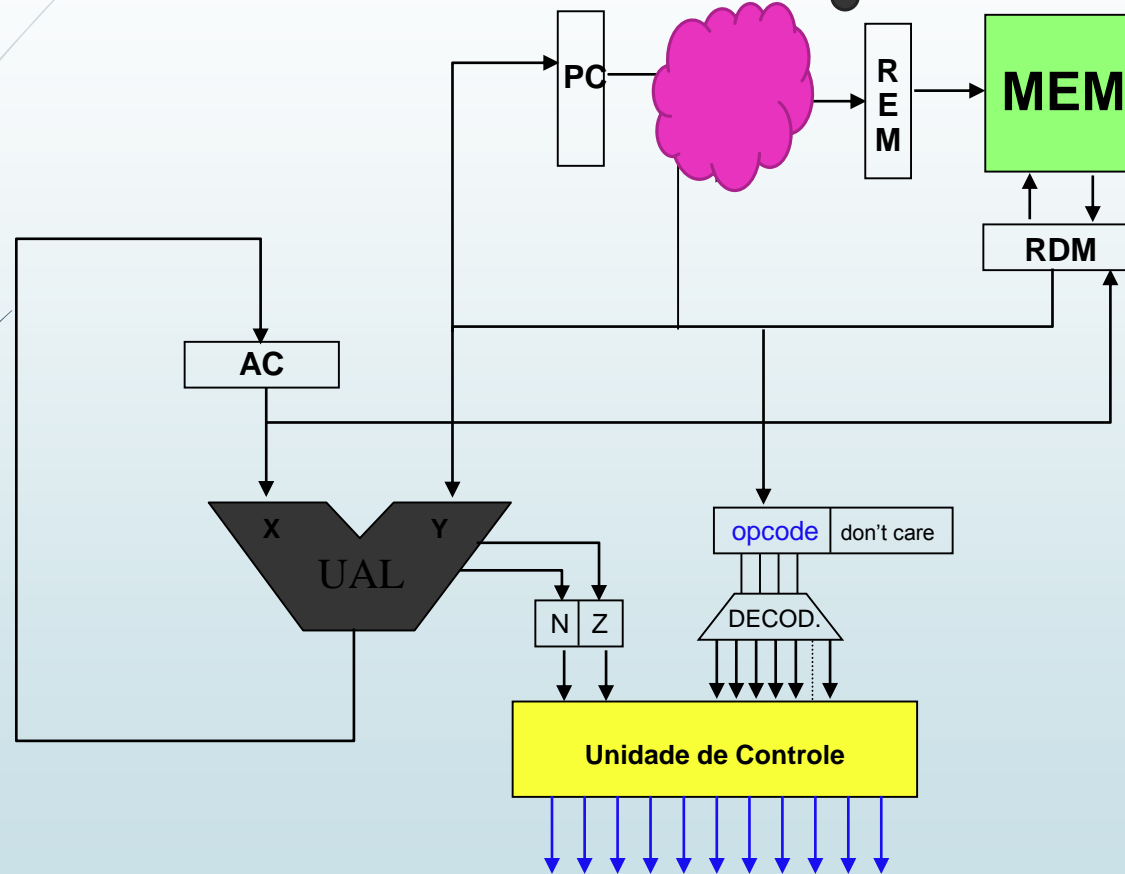
- Pensando nas necessidades do Neander, quais elementos ou componentes ele precisa ?
 - Memória de 256 posições de 8 bits cada;
 - Um registrador de 8 bits para o Acumulador;
 - Um registrador de 8 bits para o Registrador de Instruções;
 - Um registrador-contador de 8 bits para o Contador de Programa;
 - Dois flip-flops: um para cada código de condição (N e Z);

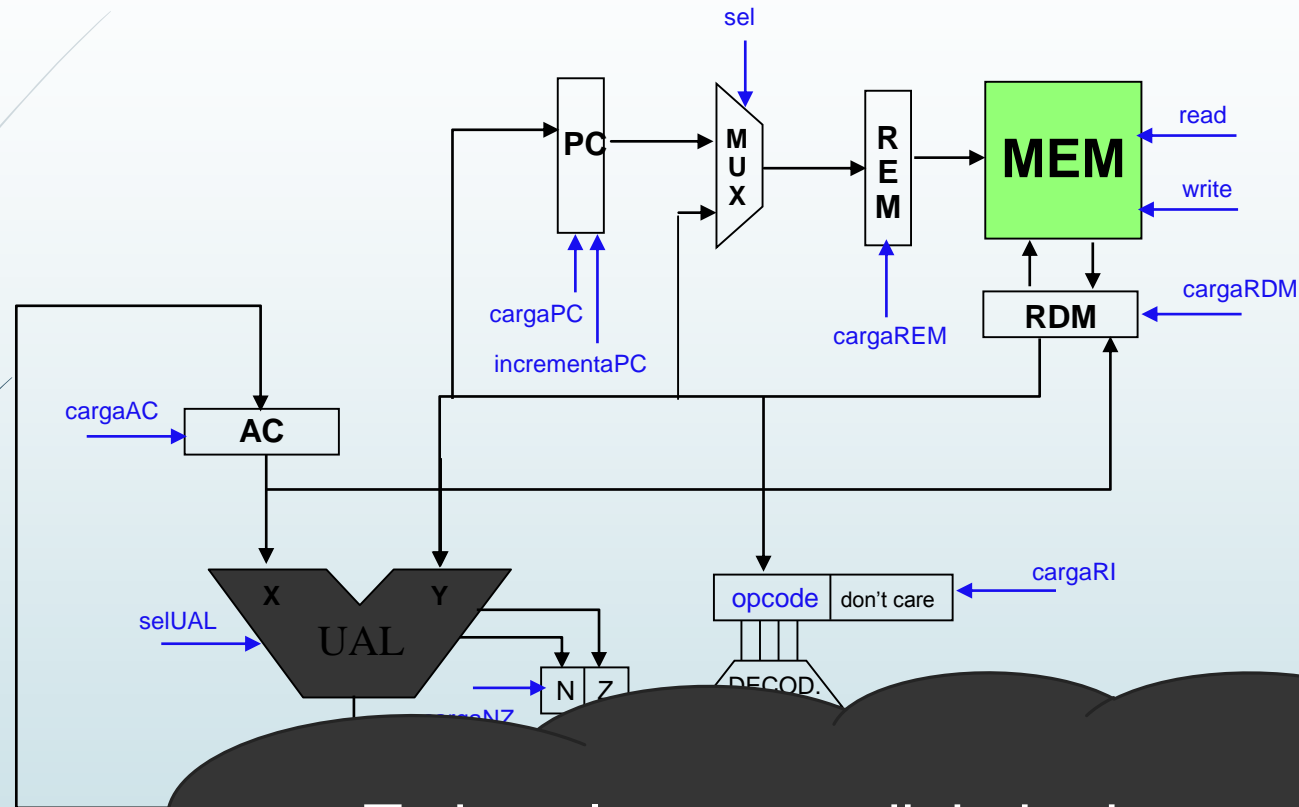
- Pensando nas necessidades do Neander, quais elementos ou componentes ele precisa ?
 - Memória de 256 posições de 8 bits cada;
 - Um registrador de 8 bits para o Acumulador;
 - Um registrador de 8 bits para o Registrador de Instruções;
 - Um registrador-contador de 8 bits para o Contador de Programa;
 - Dois flip-flops: um para cada código de condição (N e Z);
 - Uma Unidade Lógica e Aritmética;
 - Uma Unidade de Controle;



Temos os seguintes componentes...

**Endereço da próxima instrução
ou endereço que contém dados?**





Todos elementos digitais da organização precisam de um sinal de controle para sua ativação...

Roteiro



- ~~• Componentes necessários.~~
- Fluxo de dados.
- Sinais de Controle.
- Contador de Programa (PC).
- Resumo.

Transferências

$$RI \leftarrow RDM$$

RDM \leftarrow **AC**

Write

Read

AC \leftarrow RDM; atualiza N e Z

AC ← AC + RDM; atualiza N e Z

AC ← AC OR RDM; at. N e Z

AC ← AC AND RDM; at. N e Z

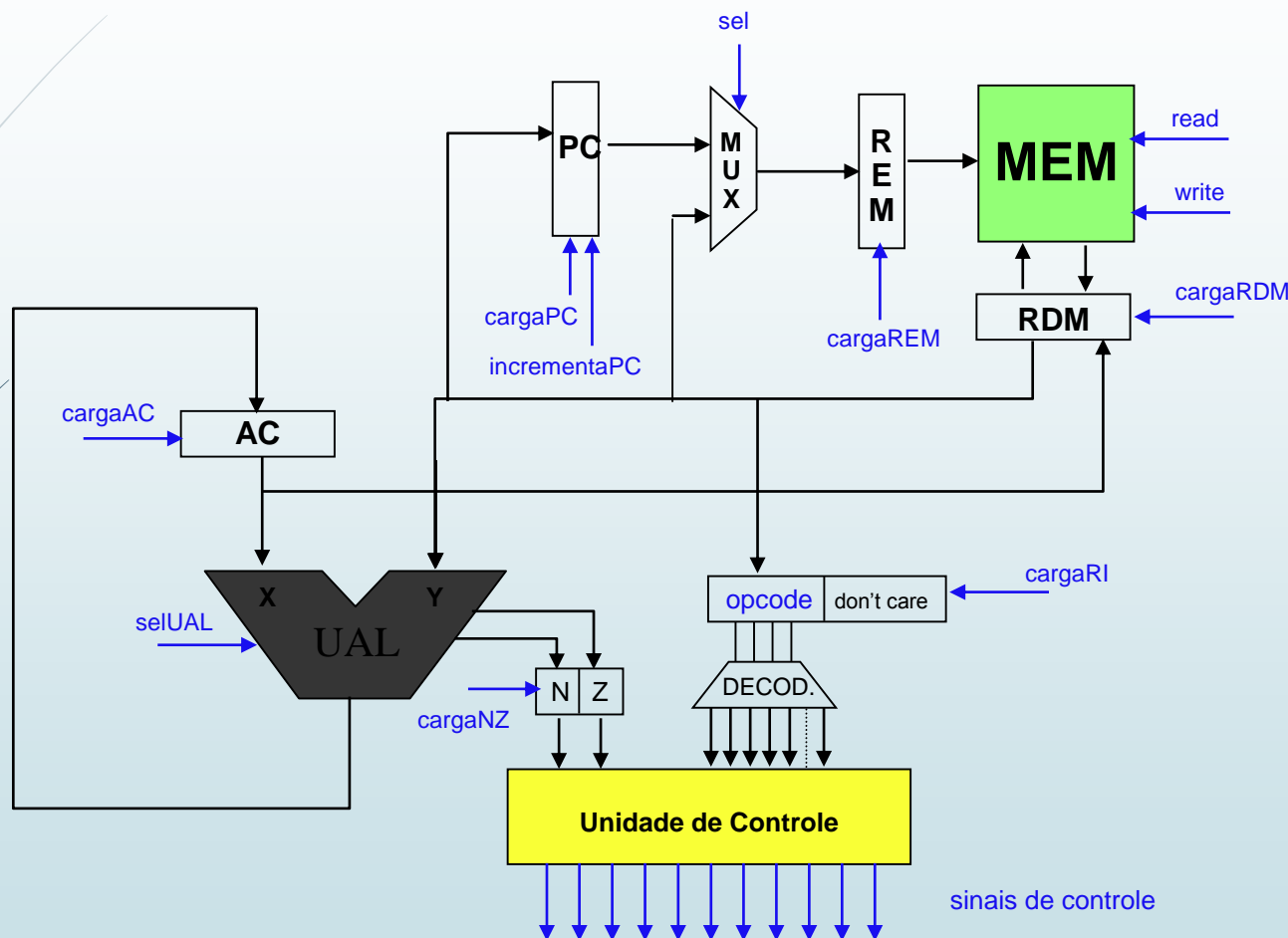
AC ← NOT(AC); atualiza N e Z

PC ← RDM

$$PC \leftarrow PC + 1$$

REM \leftarrow PC

REM \leftarrow RDM



A seguir serão descritas as transferências
entre os elementos da organização do
Neander durante a busca e execução de
algumas das instruções...

Figura 5: Instrução NOP e as transferências necessárias.

Instrução NOP	
Busca	REM \leftarrow PC
	Read (RDM \leftarrow MEM(PC)); PC \leftarrow PC + 1
	RI \leftarrow RDM
Execução	Nenhuma Operação !

Fonte: Adaptado de Weber, 2001.

Figura 6: Instrução NOT e as transferências necessárias.

Instrução NOT	
Busca	REM \leftarrow PC
	Read (RDM \leftarrow MEM(PC)); PC \leftarrow PC + 1
	RI \leftarrow RDM
Execução	AC \leftarrow NOT (AC); Atualiza Códigos de Condição N e Z

Fonte: Adaptado de Weber, 2001.

Figura 13: Instrução HLT e as transferências necessárias.

Instrução HLT	
Busca	REM \leftarrow PC
	Read (RDM \leftarrow MEM(PC)); PC \leftarrow PC + 1
	RI \leftarrow RDM
Execução	Parar o processamento !

Fonte: Adaptado de Weber, 2001.

Figura 7: Instrução STA e as transferências necessárias.

Instrução STA	
Busca	REM \leftarrow PC
	Read (RDM \leftarrow MEM(PC)); PC \leftarrow PC + 1
	RI \leftarrow RDM
Execução	REM \leftarrow PC
	Read (RDM \leftarrow MEM(PC)); PC \leftarrow PC + 1
	REM \leftarrow RDM
	RDM \leftarrow AC
	Write (MEM(REM) \leftarrow RDM)

Fonte: Adaptado de Weber, 2001.

Figura 8: Instrução LDA e as transferências necessárias.

Instrução LDA	
Busca	REM \leftarrow PC
	Read (RDM \leftarrow MEM(PC)); PC \leftarrow PC + 1
	RI \leftarrow RDM
Execução	REM \leftarrow PC
	Read (RDM \leftarrow MEM(PC)); PC \leftarrow PC + 1
	REM \leftarrow RDM
	Read (RDM \leftarrow MEM(REM))
	AC \leftarrow RDM; Atualiza Códigos de Condição N e Z

Fonte: Adaptado de Weber, 2001.

Figura 9: Instrução ADD e as transferências necessárias.

Instrução ADD	
Busca	REM \leftarrow PC
	Read (RDM \leftarrow MEM(PC)); PC \leftarrow PC + 1
	RI \leftarrow RDM
Execução	REM \leftarrow PC
	Read (RDM \leftarrow MEM(PC)); PC \leftarrow PC + 1
	REM \leftarrow RDM
	Read (RDM \leftarrow MEM(REM))
	AC \leftarrow AC + RDM; Atualiza Códigos de Condição N e Z

Fonte: Adaptado de Weber, 2001.

Figura 10: Instrução JN e as transferências necessárias.

Instrução JN (para N=0 e N=1)		
Busca	REM \leftarrow PC	
	Read (RDM \leftarrow MEM(PC)); PC \leftarrow PC + 1	
	RI \leftarrow RDM	
Execução	Para N = 1	Para N = 0
	REM \leftarrow PC	PC \leftarrow PC + 1
	Read (RDM \leftarrow MEM(REM))	
	PC \leftarrow RDM	

Fonte: Adaptado de Weber, 2001.

Figura 11: Instrução JZ e as transferências necessárias.

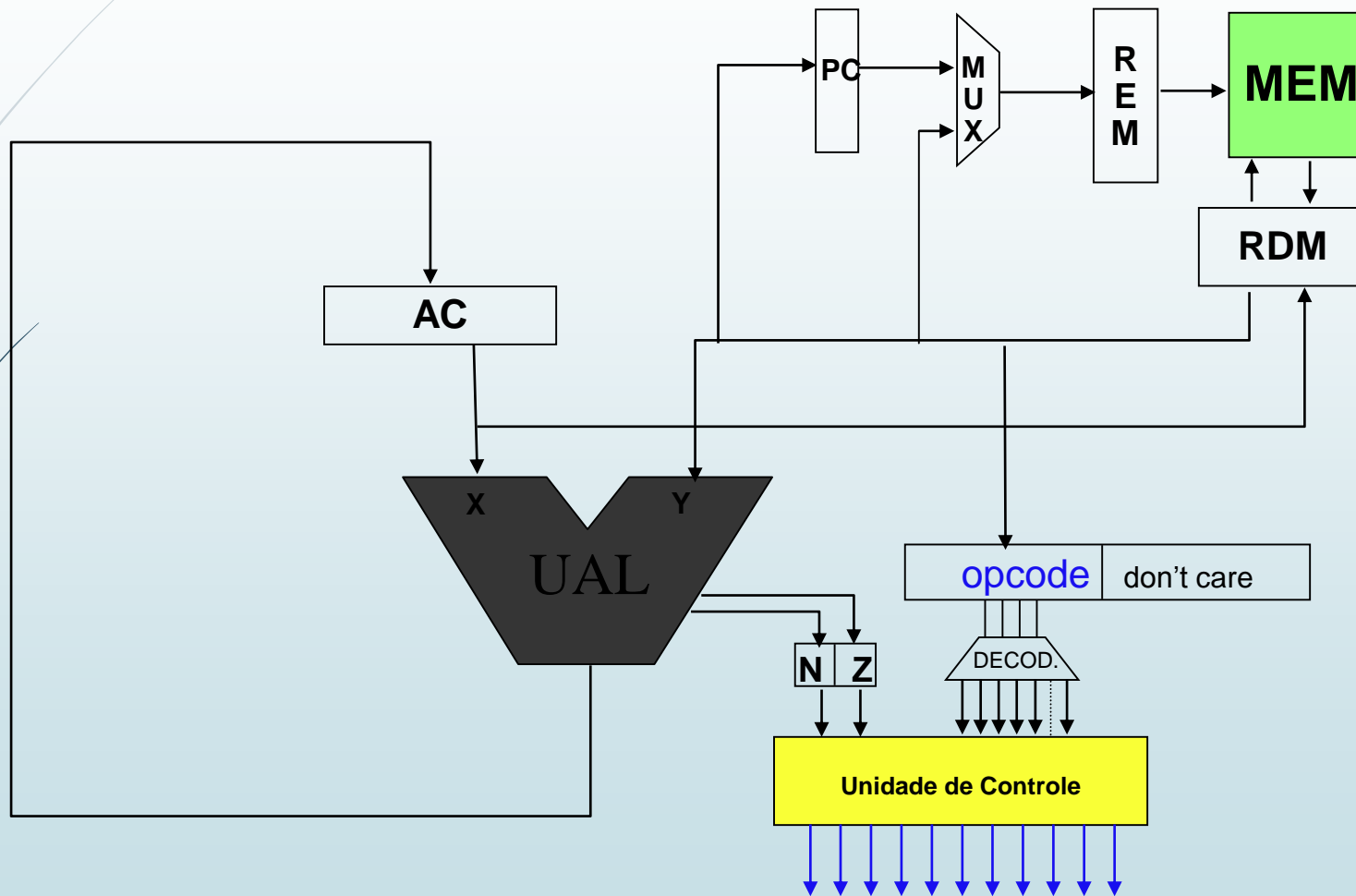
Instrução JZ (para Z=0 e Z=1)		
Busca	REM \leftarrow PC	
	Read (RDM \leftarrow MEM(PC)); PC \leftarrow PC + 1	
	RI \leftarrow RDM	
Execução	Para Z = 1	Para Z = 0
	REM \leftarrow PC	PC \leftarrow PC + 1
	Read (RDM \leftarrow MEM(REM))	
	PC \leftarrow RDM	

Fonte: Adaptado de Weber, 2001.

Figura 12: Instrução JMP e as transferências necessárias.

Instrução JMP	
Busca	REM \leftarrow PC
	Read (RDM \leftarrow MEM(PC)); PC \leftarrow PC + 1
	RI \leftarrow RDM
Execução	REM \leftarrow PC
	Read (RDM \leftarrow MEM(REM))
	PC \leftarrow RDM

Fonte: Adaptado de Weber, 2001.



Roteiro



- ~~Componentes necessários.~~
- ~~Fluxo de dados.~~
- Sinais de Controle.
- Contador de Programa (PC).
- Resumo.

Transferências

RI ← RDM

RDM \leftarrow **AC**

Write

Read

AC \leftarrow RDM; atualiza N e Z

AC \leftarrow AC + RDM; atualiza N e Z

AC ← AC OR RDM; at. N e Z

AC ← AC AND RDM; at. N e Z

AC ← NOT(AC); atualiza N e Z

PC ← RDM

$$PC \leftarrow PC + 1$$

REM ← PC

REM \leftarrow RDM

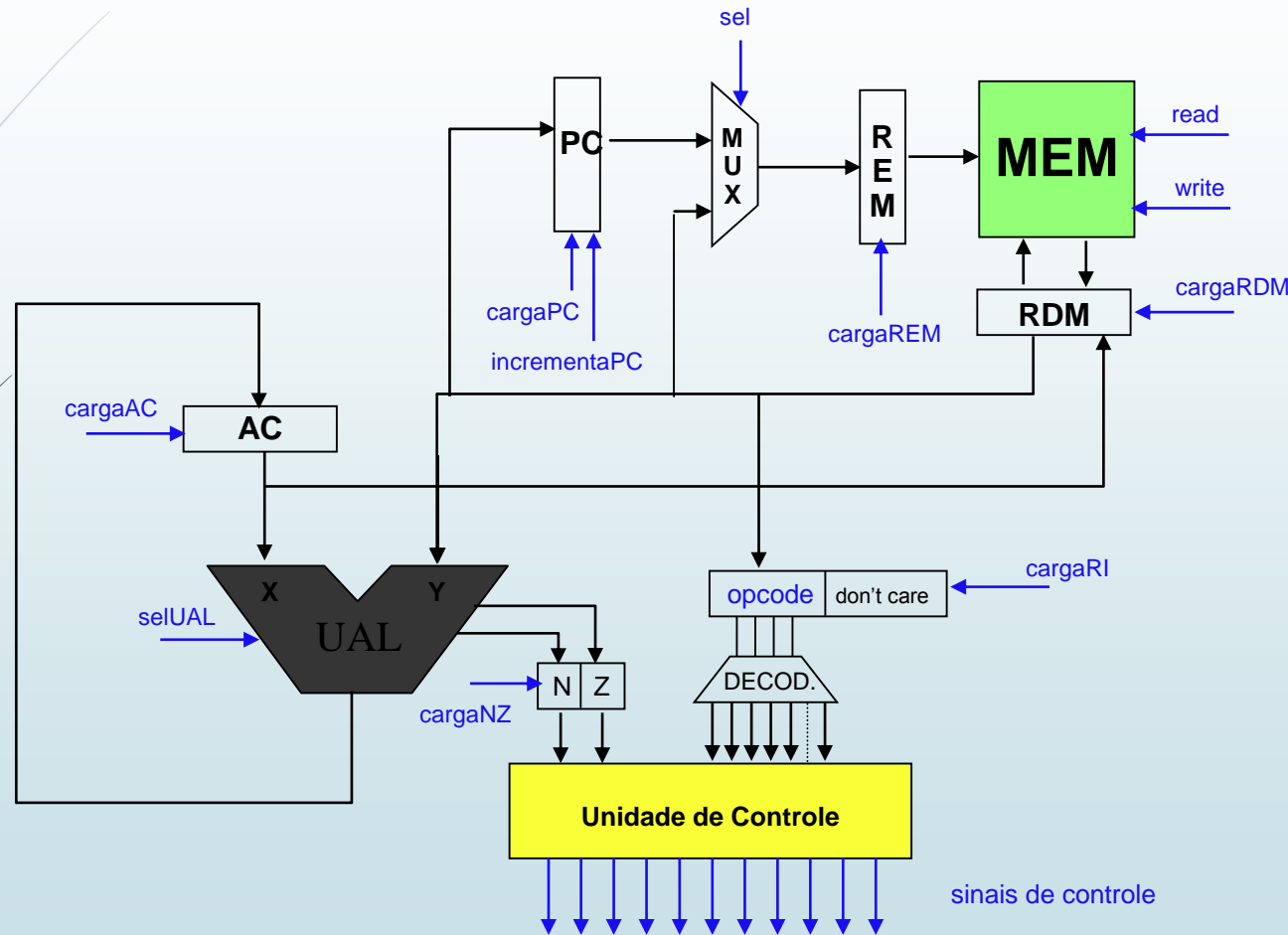


Figura 14: Transferências entre os componentes e os sinais de controle.

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

Fonte: Elaborado pelo autor.

Figura 15: 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	

Fonte: Adaptado de Weber, 2001.

Figura 16: Temporização dos sinais de controle.

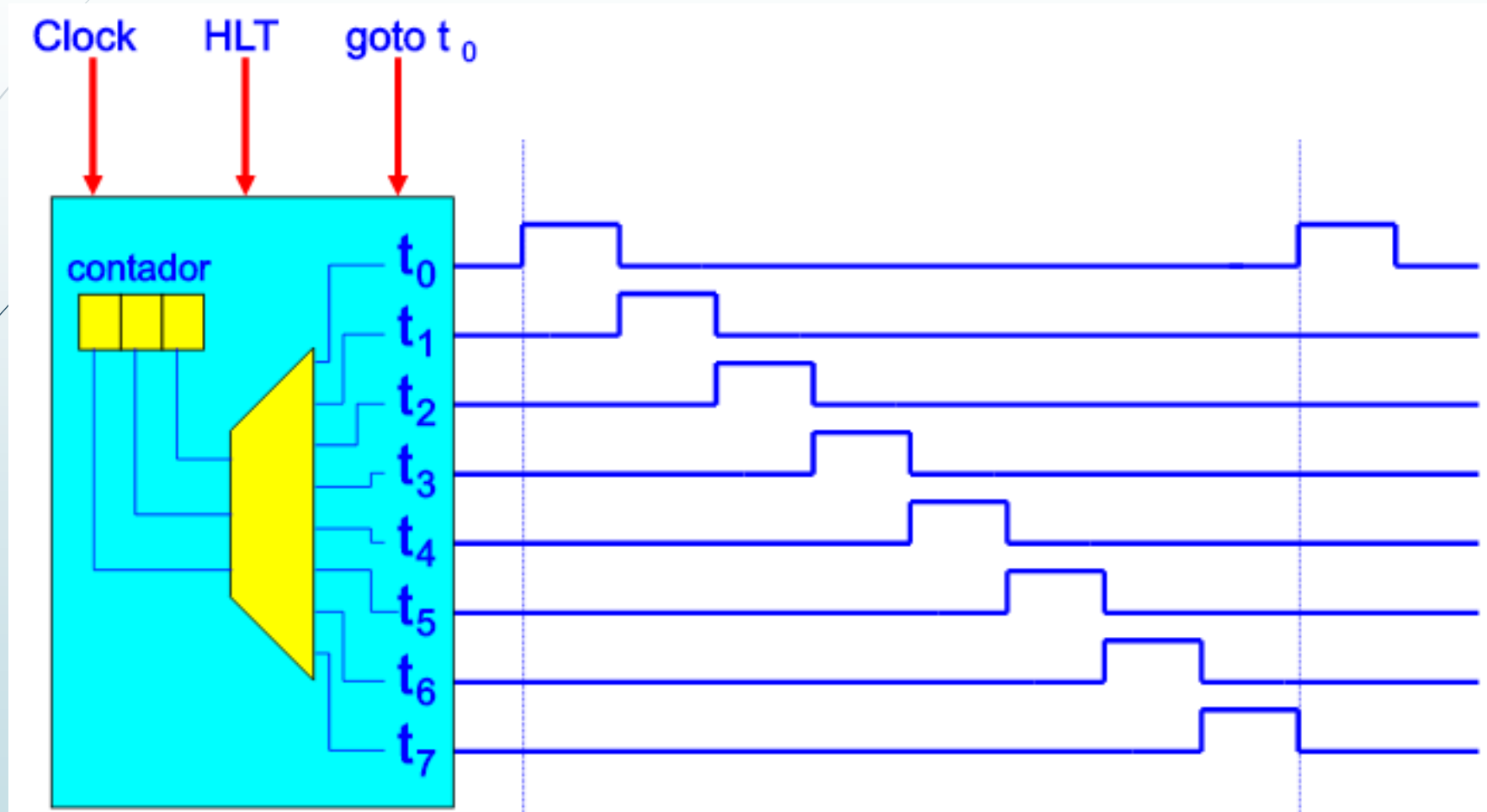
tempo	JMP	JN, N=1	JN, N=0	JZ, Z=1	JZ, Z=0	NOP	HLT
t0	sel=0, carga REM	sel=0, carga REM	sel=0, carga REM	sel=0, carga REM	sel=0, carga REM	sel=0, carga REM	sel=0, carga REM
t1	Read, incrementa PC	Read, incrementa PC	Read, incrementa PC	Read, incrementa PC	Read, incrementa PC	Read, incrementa PC	Read, 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							

Fonte: Adaptado de Weber, 2001.



Na Unidade de Controle...

Figura 17: Geração dos sinais de temporização



Fonte: Elaborado pelo autor, 2016.

- Percebemos que nos tempos t_0 , t_1 e t_2 é feita a busca da instrução.
- Nos tempos seguintes é feita a execução.
- Após a execução de uma instrução, se torna necessário um sinal de controle adicional para voltar no tempo t_0 .
- Como podemos implementar a Unidade de Controle?

- Um registrador contador pode ser usado para gerar os sinais de tempo (t_0 a t_7) a partir de um relógio básico (clock).
- Lógica combinacional para os códigos de condição, os sinais do decodificador e os sinais de tempo para gerar todos os sinais de controle necessários à organização do Neander.
- Como derivar as funções booleanas para cada um dos sinais de controle? Basta analisar a figura da temporização dos sinais de controle.

Figura 18: Sinais de controle em função dos tempos e instruções.

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

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

$\text{carga RI} = t_2$

$\text{sel} = t_5 \cdot (\text{STA} + \text{LDA} + \text{ADD} + \text{OR} + \text{AND})$

$\text{carga RDM} = t_6 \cdot \text{STA}$

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

$\text{Write} = t_7 \cdot \text{STA}$

$\text{UAL(Y)} = t_7 \cdot \text{LDA}$

$\text{UAL(ADD)} = t_7 \cdot \text{ADD}$

$\text{UAL(OR)} = t_7 \cdot \text{OR}$

$\text{UAL(AND)} = t_7 \cdot \text{AND}$

$\text{UAL(NOT)} = t_3 \cdot \text{NOT}$

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

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

$\text{carga PC} = t_5 \cdot (\text{JMP} + \text{JN.N} + \text{JZ.Z})$

Fonte: Adaptado de Weber, 2001.

Roteiro



- ~~Componentes necessários.~~
- ~~Fluxo de dados.~~
- ~~Sinais de Controle.~~
- Contador de Programa (PC).
- Resumo.

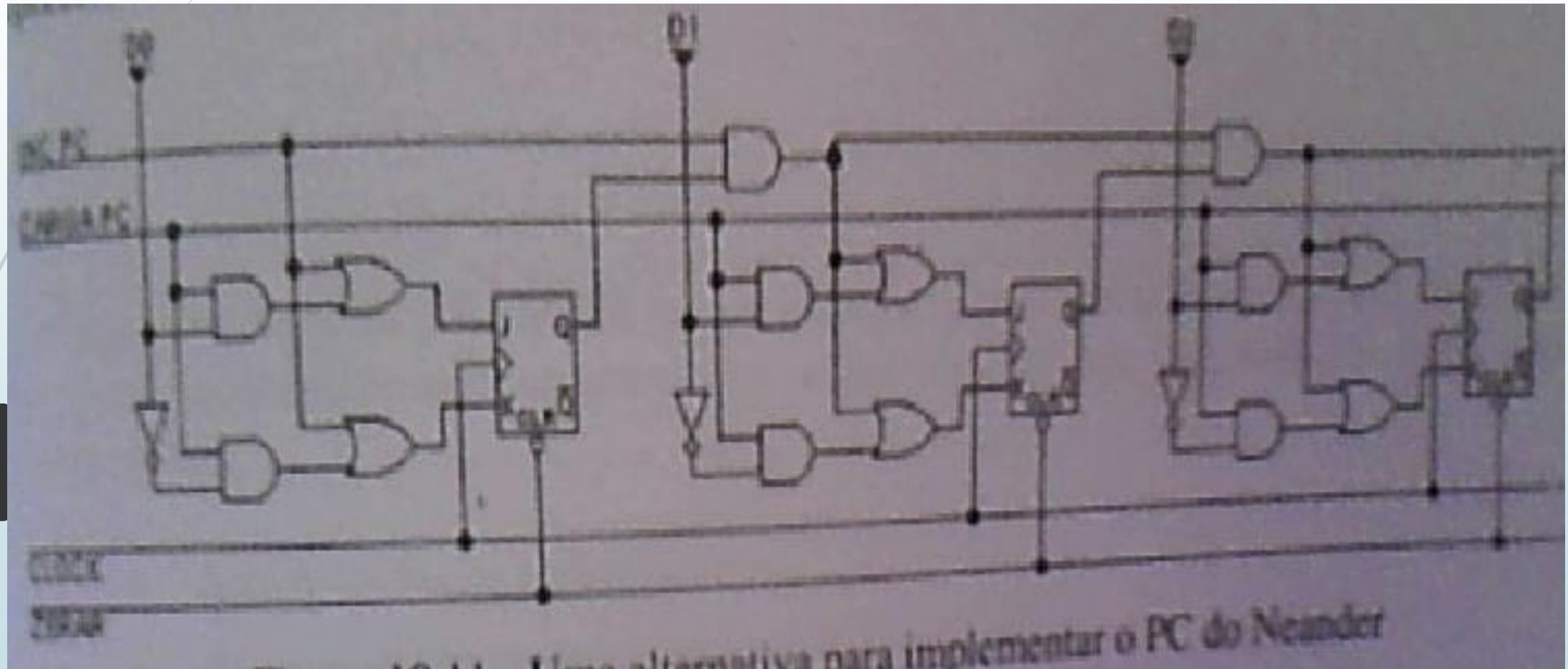
- O PC, além de um registrador, ele é um contador.
- O PC, quando não ocorrem desvios, incrementa o endereço contido nele mesmo em +1.
- Também devemos considerar que o PC recebe 2 sinais de controle:
 - O INCPC, que indica que o endereço no PC deve aumentar em +1;
 - O CARGAPC, que indica que o endereço no PC deve ser o endereço de desvio indicado por alguma instrução de desvio;
- Para relembrar, como são construídos registradores e contadores?

- Primeiramente:
 - Por ser um registrador contador que armazena endereços de 8 bits, teremos 8 flip-flops JK;
 - Para as entradas dos flip-flops temos:
 - 2 sinais de controle.
 - Sinal de pulso de clock.
 - Bit do dado, caso seja efetuado um desvio;
 - Vamos analisar os sinais de controle:

- Os sinais INCPC e CARGAPC poderão estar ativos simultaneamente?
- Os sinais INCPC e CARGAPC poderão estar inativos simultaneamente?

- Os sinais INCPC e CARGAPC poderão estar ativos simultaneamente?
- Não.
- Ou o PC armazena o endereço de desvio ou ele incrementa em +1.
- Os sinais INCPC e CARGAPC poderão estar inativos simultaneamente?
- Sim.
- O Neander pode estar na fase de execução de um ADD, por exemplo, em um momento onde o PC não é alterado de valor.
- Se o CARGAPC estiver em 1, o dado deve ser armazenado no PC.
- Se o INCPC estiver em 1, no PC deve ser somado +1. O bit menos significativo deve inverter o sinal e os posteriores corrigidos, através da análise da saída de cada um dos flip-flops em uma AND com o INCPC.

Figura 19: Possível implementação para o PC.



Fonte: Adaptado de Weber, 2001.

Roteiro



- ~~Componentes necessários.~~
- ~~Fluxo de dados.~~
- ~~Sinais de Controle.~~
- ~~Contador de Programa (PC).~~
- **Resumo.**

- Foram demonstrados os elementos da organização do Neander.
- Foram demonstradas as possíveis interações entre os componentes da organização (fluxo de dados).
- Foi demonstrado de forma mais detalhada, o funcionamento da Unidade de Controle e os sinais de controle.
- Foi demonstrada uma possibilidade de construção de contador de programa.



Dúvidas?

Dηλ!q92j