

Organização do Neander

10.1 Elementos necessários

Para definir uma organização para o computador NEANDER, é necessário inicialmente definir quais os elementos necessários. Estes dados podem ser retirados das próprias características do NEANDER:

- Largura de dados e endereços de 8 bits
- Dados representados em complemento de dois
- 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)

Assim, os seguintes elementos são necessários:

- Um registrador de 8 bits para o acumulador
- Um registrador de 8 bits para o PC (e possivelmente um registrador contador)
- Dois flip-flops: um para o código de condição N e outro para Z
- Uma memória de 256 posições de 8 bits cada

10.2 Fluxo de dados

O conjunto de instruções do NEANDER fornece mais detalhes sobre o uso e as interconexões necessárias entre estes elementos:

Código	Instrução	Execução
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	término de execução - (halt)

Tabela 10.1 - Conjunto de instruções do NEANDER

A fase de busca de cada instrução não está mostrada na Tabela 10.1, mas é igual para todas as instruções:

$\text{RI} \leftarrow \text{MEM}(\text{PC})$
 $\text{PC} \leftarrow \text{PC} + 1$

Com isto introduz-se um novo elemento, o Registrador de Instruções (RI), que deve apresentar tamanho suficiente para armazenar uma instrução completa.

A seguir estão descritas as transferências entre os diversos elementos de armazenamento que formam a organização do NEANDER. Note-se que estas transferências já indicam quais os caminhos de dados necessários (qual saída de um registrador deve ser levada até qual entrada de outro registrador), mas não indicam qual o caminho físico exato utilizado para a transferência.

Instrução NOP

Busca: $RI \leftarrow MEM(PC)$
 $PC \leftarrow PC + 1$
 Execução: Nenhuma operação necessária

Instrução STA

Busca: $RI \leftarrow MEM(PC)$
 $PC \leftarrow PC + 1$
 Execução: $end \leftarrow MEM(PC)$
 $PC \leftarrow PC + 1$
 $MEM(end) \leftarrow AC$

Instrução LDA

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

Instrução ADD

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

Instrução OR

Busca: $RI \leftarrow MEM(PC)$
 $PC \leftarrow PC + 1$
 Execução: $end \leftarrow MEM(PC)$
 $PC \leftarrow PC + 1$
 $AC \leftarrow AC \text{ or } MEM(end)$; atualiza N e Z

Instrução AND

Busca: $RI \leftarrow MEM(PC)$
 $PC \leftarrow PC + 1$
 Execução: $end \leftarrow MEM(PC)$
 $PC \leftarrow PC + 1$
 $AC \leftarrow AC \text{ and } MEM(end)$; atualiza N e Z

Instrução NOT

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

Instrução JMP

Busca: $RI \leftarrow MEM(PC)$
 $PC \leftarrow PC + 1$
 Execução: $end \leftarrow MEM(PC)$
 $PC \leftarrow end$

Instrução JN, caso em que $N=1$

Busca: $RI \leftarrow MEM(PC)$
 $PC \leftarrow PC + 1$
Execução: $end \leftarrow MEM(PC)$
 $PC \leftarrow end$

Instrução JN, caso em que $N=0$

Busca: $RI \leftarrow MEM(PC)$
 $PC \leftarrow PC + 1$
Execução: $end \leftarrow MEM(PC)$ (esta transferência a rigor é desnecessária)
 $PC \leftarrow PC + 1$

Instrução JZ, caso em que $Z=1$

Busca: $RI \leftarrow MEM(PC)$
 $PC \leftarrow PC + 1$
Execução: $end \leftarrow MEM(PC)$
 $PC \leftarrow end$

Instrução JZ, caso em que $Z=0$

Busca: $RI \leftarrow MEM(PC)$
 $PC \leftarrow PC + 1$
Execução: $end \leftarrow MEM(PC)$ (esta transferência a rigor é desnecessária)
 $PC \leftarrow PC + 1$

Instrução HLT

Busca: $RI \leftarrow MEM(PC)$
 $PC \leftarrow PC + 1$
Execução: Parar o processamento

Uma transferência do tipo $x \leftarrow MEM(y)$ descreve uma leitura de memória. Esta operação pode ser decomposta em três fases:

1. $REM \leftarrow y$ Transferência do endereço y para o Reg. de Endereços da Memória
2. Read Ativação de uma operação de Leitura da Memória
3. $x \leftarrow RDM$ Transferência do Reg. de Dados da Memória para x

Por outro lado, uma transferência do tipo $MEM(y) \leftarrow x$ descreve uma escrita de memória. Esta operação também pode ser decomposta em três fases:

1. $REM \leftarrow y$ Transferência do endereço y para o Reg. de Endereços da Memória
2. $RDM \leftarrow x$ Transferência do dado x para o Reg. de Dados da Memória
3. Write Ativação da operação de Escrita na Memória

Além disto, as seguintes observações podem ser feitas:

1. Após uma leitura de memória na posição do PC, o conteúdo deste registrador deve ser incrementado, para apontar para a posição seguinte. Esta operação pode ser feita a qualquer instante de tempo após a transferência do PC para o REM. E o incremento pode ser feito em paralelo com outras operações. Nas seqüências descritas a seguir, este incremento é feito sempre ao mesmo tempo que a operação na memória (Read ou Write).
2. Um desvio condicional que não se realize não necessita ler o valor do endereço de desvio. Assim, basta incrementar mais uma vez o valor do PC, para que este “pule” sobre a posição de memória que contém este endereço e passe a apontar para a instrução seguinte.

Com isto, obtêm-se as seguintes seqüências:

Instrução NOP:

Busca: $REM \leftarrow PC$
Read; $PC \leftarrow PC + 1$
 $RI \leftarrow RDM$

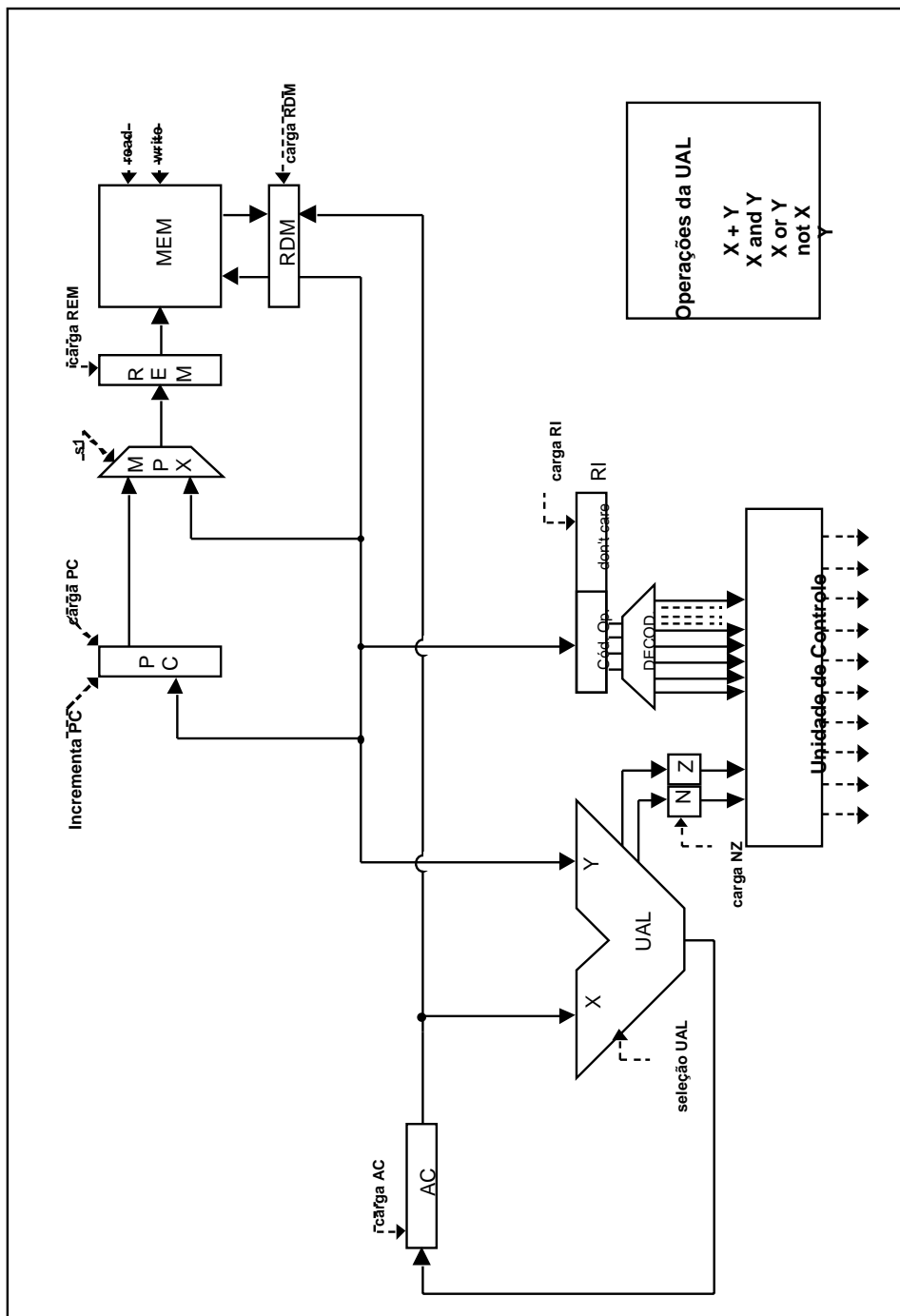
	Execução:	Nenhuma operação
Instrução STA		
	Busca:	$REM \leftarrow PC$ Read; $PC \leftarrow PC + 1$ $RI \leftarrow RDM$
	Execução:	$REM \leftarrow PC$ Read; $PC \leftarrow PC + 1$ $REM \leftarrow RDM$ $RDM \leftarrow AC$ Write
Instrução LDA		
	Busca:	$REM \leftarrow PC$ Read; $PC \leftarrow PC + 1$ $RI \leftarrow RDM$
	Execução:	$REM \leftarrow PC$ Read; $PC \leftarrow PC + 1$ $REM \leftarrow RDM$ Read $AC \leftarrow RDM$; Atualiza N e Z
Instrução ADD		
	Busca:	$REM \leftarrow PC$ Read; $PC \leftarrow PC + 1$ $RI \leftarrow RDM$
	Execução:	$REM \leftarrow PC$ Read; $PC \leftarrow PC + 1$ $REM \leftarrow RDM$ Read $AC \leftarrow AC + RDM$; Atualiza N e Z
Instrução OR		
	Busca:	$REM \leftarrow PC$ Read; $PC \leftarrow PC + 1$ $RI \leftarrow RDM$
	Execução:	$REM \leftarrow PC$ Read; $PC \leftarrow PC + 1$ $REM \leftarrow RDM$ Read $AC \leftarrow AC \text{ or } RDM$; Atualiza N e Z
Instrução AND		
	Busca:	$REM \leftarrow PC$ Read; $PC \leftarrow PC + 1$ $RI \leftarrow RDM$
	Execução:	$REM \leftarrow PC$ Read; $PC \leftarrow PC + 1$ $REM \leftarrow RDM$ Read $AC \leftarrow AC \text{ and } RDM$; Atualiza N e Z
Instrução NOT		
	Busca:	$REM \leftarrow PC$ Read; $PC \leftarrow PC + 1$ $RI \leftarrow RDM$
	Execução:	$AC \leftarrow \text{not}(AC)$; Atualiza N e Z

Instrução JMP		
Busca:	REM \leftarrow PC	
	Read; PC \leftarrow PC + 1	
	RI \leftarrow RDM	
Execução:	REM \leftarrow PC	
	Read	
	PC \leftarrow RDM	
Instrução JN quando N=1		
Busca:	REM \leftarrow PC	
	Read; PC \leftarrow PC + 1	
	RI \leftarrow RDM	
Execução:	REM \leftarrow PC	
	Read	
	PC \leftarrow RDM	
Instrução JN quando N=0		
Busca:	REM \leftarrow PC	
	Read; PC \leftarrow PC + 1	
	RI \leftarrow RDM	
Execução:	PC \leftarrow PC + 1	
Instrução JZ quando Z=1		
Busca:	REM \leftarrow PC	
	Read; PC \leftarrow PC + 1	
	RI \leftarrow RDM	
Execução:	REM \leftarrow PC	
	Read	
	PC \leftarrow RDM	
Instrução JZ quando Z=0		
Busca:	REM \leftarrow PC	
	Read; PC \leftarrow PC + 1	
	RI \leftarrow RDM	
Execução:	PC \leftarrow PC + 1	
Instrução HLT		
Busca:	REM \leftarrow PC	
	Read; PC \leftarrow PC + 1	
	RI \leftarrow RDM	
Execução:	Parar o processamento	

A Figura 10.1 ilustra uma possível interconexão entre os elementos de armazenamento e os elementos combinacionais necessários para implementar a arquitetura do NEANDER. Ela é derivada quase que diretamente do fluxo de dados mostrado acima.

Para a organização do NEANDER mostrada na Figura 10.1 as seguintes considerações podem ser feitas:

1. O incremento do PC poderia ser feito de várias maneiras. Entre elas, podem ser citadas a soma feita através da UAL, a soma feita através de um somador próprio e o uso de um registrador contador. Para a organização optou-se por esta última.
2. Para cada registrador é necessário um sinal de “carga” correspondente, que indica quando o valor da sua entrada deve ser armazenado.



Sinais de Controle para a UCP

Figura 10.1 - Organização para o NEANDER

3. Para atualizar os códigos de N e Z durante a operação de LDA, acrescentou-se uma operação de transferência através da UAL. Com isto, a UAL realiza cinco operações possíveis.
4. As entradas X e Y da UAL, assim como as operações de NOT(X) e Y, foram escolhidas de forma a simplificar as transferências. Com isto, a entrada X está permanentemente ligada à saída da UAL, e a entrada Y da UAL está ligada ao RDM.

5. O único registrador que recebe dados de duas fontes possíveis é o REM. Para solucionar este conflito utiliza-se um multiplexador que seleciona entre o PC (sel=0) e o RDM (sel=1).

10.3 Sinais de controle

Todos os sinais de controle da Figura 10.1 são gerados nos instantes de tempo adequados pela Unidade de Controle. A Tabela 10.2 mostra a equivalência entre as transferências realizadas e os sinais de controle a serem ativados.

Transferência	Sinais de controle
$REM \leftarrow PC$	sel=0, carga REM
$PC \leftarrow PC + 1$	incrementa PC
$RI \leftarrow RDM$	carga RI
$REM \leftarrow RDM$	sel =1, carga REM
$RDM \leftarrow AC$	carga RDM
$AC \leftarrow RDM$; Atualiza N e Z	UAL(Y), carga AC, carga NZ
$AC \leftarrow AC + RDM$; Atualiza N e Z	UAL(ADD), carga AC, carga NZ
$AC \leftarrow AC \text{ or } RDM$; Atualiza N e Z	UAL(OR), carga AC, carga NZ
$AC \leftarrow AC \text{ and } RDM$; Atualiza N e Z	UAL(AND), carga AC, carga NZ
$AC \leftarrow \text{not}(AC)$; Atualiza N e Z	UAL(NOT), carga AC, carga NZ
$PC \leftarrow RDM$	carga PC

Tabela 10.2 - Sinais de controle para as transferências do NEANDER

Com isto, as seqüências de transferências entre registradores analisadas anteriormente para cada instrução podem ser transformadas em seqüências de sinais de controle. A Tabela 10.3 mostra estas seqüências para as instruções STA, LDA, ADD, OR, AND e NOT. A Tabela 10.4 mostra as seqüências para as instruções NOP, JMP, JN, JZ e HLT

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	

Tabela 10.3- Sinais de controle para STA, LDA, ADD, OR, AND e NOT

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							

Tabela 10.4- Sinais de controle para JMP, JN, JZ, NOP e HLT

Para o sequenciamento de todas as instruções são necessários 8 tempos distintos, numerados de t0 a t7. Os três primeiros, t0, t1 e t2, servem para a fase de busca das instruções. Os demais (t3 a t7) são necessários para a fase de execução. Observe-se que, quando termina a execução de uma instrução, existe um sinal de controle explícito para voltar ao tempo t0 (goto t0). Das Tabelas 10.3 e 10.4 podem ser derivadas as equações booleanas para cada um dos sinais de controle. Estas equações são indicadas a seguir, em forma não necessariamente otimizada:

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

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

$$\text{carga RI} = t2$$

$$\text{sel} = t5.(STA+LDA+ADD+OR+AND)$$

$$\text{carga RDM} = t6.STA$$

$$\text{Read} = t1 + t4.(STA+LDA+ADD+OR+AND+JMP+JN.N+JZ.Z) + t6.(LDA+ADD+OR+AND)$$

$$\text{Write} = t7.STA$$

$$\text{UAL(Y)} = t7.LDA$$

$$\text{UAL(ADD)} = t7.ADD$$

$$\text{UAL(OR)} = t7.OR$$

$$\text{UAL(AND)} = t7.AND$$

$$\text{UAL(NOT)} = t3.NOT$$

$$\text{carga AC} = t7.(LDA+ADD+OR+AND) + t3.NOT$$

$$\text{carga NZ} = t7.(LDA+ADD+OR+AND) + t3.NOT = \text{carga AC}$$

$$\text{carga PC} = t5.(JMP+JN.N+JZ.Z)$$

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