Sistemas Digitais

Tema: Computador Teórico Neander e Instruções

Alunos:

Eric Klaus

Matheus Gaspar

Matheus Rogerio

Introdução:

O Neander é um máquina de von Neumann que possui um conjunto limitado de instruções e utiliza uma arquitetura de 8 bits. Ele possui três subáreas principais:

Unidade de Controle (UC):

A UC é responsável por buscar as instruções na memória, decodificá-las e controlar as operações do processador. Ela utiliza a instrução atual para determinar o próximo passo a ser executado.

Unidade de Memória (UM):

A UM é onde as instruções e os dados são armazenados. Ela é constituída por uma memória de acesso aleatório (RAM) que contém locais de memória endereçáveis, cada um com 8 bits.

Unidade Lógica e Aritmética (ULA):

A ULA é responsável por realizar as operações lógicas (AND, OR, NOT) e aritméticas (adição) entre os dados presentes nos registradores.

Instruções e Simplificações:

A seguir, apresentamos as principais instruções do Neander e as simplificações para cada uma delas:

NOP:

Tabela:

barr/Inc	barr/PC	ula_op	pc_rw	ac_rw	mem_rw	rem_rw	rdm_rw	ri_rw
1	1	000	0	0	0	1	0	0
1	1	000	1	0	0	0	1	0
1	1	000	0	0	0	0	0	1
1	1	000	0	0	0	0	0	0
1	1	000	0	0	0	0	0	0
1	1	000	0	0	0	0	0	0
1	1	000	0	0	0	0	0	0
1	1	000	0	0	0	0	0	0

Simplificação:

nBarrInc = 1;

nBarrPc = 1;

ula(2) = 0;

ula(1) = 0;

ula(0) = 0;

pc_nrw = not(counter(1)) and not(counter(2)) and counter(0);

ac_rnw = 0;

mem_nrw = 0;

rem_nrw = not(counter(2)) and not(counter(1)) and not(counter(0));

rdm_nrw = not(counter(1)) and not(counter(2)) and counter(0);

ri_nrw = not(counter(2)) and counter(1) and not(counter(0));

STA:

Tabela:

barr/Inc	barr/PC	ula_op	pc_rw	ac_rw	mem_rw	rem_rw	rdm_rw	ri_rw
1	1	000	0	0	0	1	0	0
1	1	000	1	0	0	0	1	0
1	1	000	0	0	0	0	0	1
1	1	000	0	0	0	1	0	0
1	1	000	1	0	0	0	1	0
1	0	000	0	0	0	1	0	0
1	1	000	0	0	1	0	0	0
1	1	000	0	0	0	0	0	0

Simplificação:

nBarrInc = 1;

nBarrPc = not(counter(2)) or not(counter(0)) or counter(1);

ula(2) = 0;

ula(1) = 0;

ula(0) = 0;

pc_nrw = (not(counter(2)) and not(counter(1)) and counter(0)) or (counter(2) and not(counter(1)) and not(counter(0)));

ac_nrw = 0;

mem_nrw = counter(2) and counter(1) and not(counter(0));

rem_nrw = (not(counter(2)) and not(counter(1)) and not(counter(0))) or (not(counter(2)) and counter(1) and counter(0)) or (counter(2) and not(counter(1)) and counter(0));

rdm_nrw = (not(counter(2)) and not(counter(1)) and counter(0)) or (counter(2) and not(counter(1)) and not(counter(0)));

ri_nrw = not(counter(2)) and counter(1) and not(counter(0));

LDA:

Tabela:

barr/Inc	barr/PC	ula_op	pc_rw	ac_rw	mem_rw	rem_rw	rdm_rw	ri_rw
1	1	000	0	0	0	1	0	0
1	1	000	1	0	0	0	1	0
1	1	000	0	0	0	0	0	1
1	1	000	0	0	0	1	0	0
1	1	000	1	0	0	0	1	0
1	0	000	0	0	0	1	0	0
1	1	000	0	0	0	0	1	0
1	1	000	0	1	0	0	0	0

Simplificação:

nBarrInc = 1;

nBarrPc = not(counter(2)) or counter(1) or not(counter(0));

ula(2) = 0;

ula(1) = 0;

ula(0) = 0;

pc_nrw = not(counter(1)) and (counter(2) xor counter(0));

ac_rnw = counter(0) and counter(1) and counter(2);

mem_nrw = 0;

rem_nrw = (not(counter(1)) and (counter(2) xnor counter(0))) or (not(counter(2)) and counter(1) and counter(0));

rdm_nrw = (counter(2) and not(counter(0))) or (not(counter(2)) and not(counter(1)) and counter(0));

ri_nrw = not(counter(2)) and counter(1) and not(counter(0));

ADD:

Tabela:

barr/Inc	barr/PC	ula_op	pc_rw	ac_rw	mem_rw	rem_rw	rdm_rw	ri_rw
1	1	001	0	0	0	1	0	0
1	1	001	1	0	0	0	1	0
1	1	001	0	0	0	0	0	1
1	1	001	0	0	0	1	0	0
1	1	001	1	0	0	0	1	0
1	0	001	0	0	0	1	0	0
1	1	001	0	0	0	0	1	0
1	1	001	0	1	0	0	0	0

Simplificação:

nBarrInc = 1;

nBarrPc = not(counter(2)) or not(counter(0)) or counter(1);

ula(2) = 0;

ula(1) = 0;

ula(0) = 1;

pc_nrw = (not(counter(2)) and not(counter(1)) and counter(0)) or (counter(2) and not(counter(1)) and not(counter(0)));

ac_rnw = counter(2) and counter(1) and counter(0);

mem_nrw = 0;

rem_nrw = (not(counter(2)) and not(counter(1)) and not(counter(0))) or (not(counter(2)) and counter(1) and counter(0)) or (counter(2) and not(counter(1)) and counter(0));

rdm_nrw = (counter(2) and not(counter(0))) or (not(qcounter(2)) and not(counter(1)) and counter(0));

ri_nrw = not(counter(2)) and counter(1) and not(counter(0));

AND:

Tabela:

barr/Inc	barr/PC	ula_op	pc_rw	ac_rw	mem_rw	rem_rw	rdm_rw	ri_rw
1	1	011	0	0	0	1	0	0
1	1	011	1	0	0	0	1	0
1	1	011	0	0	0	0	0	1
1	1	011	0	0	0	1	0	0
1	1	011	1	0	0	0	1	0
1	0	011	0	0	0	1	0	0
1	1	011	0	0	0	0	1	0
1	1	011	0	1	0	0	0	0

Simplificação:

nBarrInc = 1;

nBarrPc = not(counter(2)) or counter(1) or not(counter(0));

ula(2) = 0;

ula(1) = 1;

ula(0) = 1;

pc_nrw = (not(counter(2)) and not(counter(1)) and counter(0)) or (counter(2) and not(counter(1)) and not(counter(0)));

ac_rnw = counter(2) and counter(1) and counter(0);

mem_nrw = 0;

rem_nrw = (not(counter(2)) and not(counter(1)) and not(counter(0))) or (not(counter(2)) and counter(1) and counter(0)) or (counter(2) and not(counter(1)) and counter(0));

rdm_nrw = (counter(2) and not(counter(0))) or (not(counter(2)) and not(counter(1)) and counter(0));

ri_nrw = not(counter(2)) and counter(1) and not(counter(0));

OR:

Tabela:

barr/Inc	barr/PC	ula_op	pc_rw	ac_rw	mem_rw	rem_rw	rdm_rw	ri_rw
1	1	010	0	0	0	1	0	0
1	1	010	1	0	0	0	1	0
1	1	010	0	0	0	0	0	1
1	1	010	0	0	0	1	0	0
1	1	010	1	0	0	0	1	0
1	0	010	0	0	0	1	0	0
1	1	010	0	0	0	0	1	0
1	1	010	0	1	0	0	0	0

Simplificação:

nBarrInc = 1;

nBarrPc = not(counter(2)) or counter(1) or not(counter(0));

ula(2) = 0;

ula(1) = 1;

ula(0) = 0;

pc_nrw = (not(counter(2)) and not(counter(1)) and counter(0)) or (counter(2) and not(counter(1)) and not(counter(0)));

ac_rnw = counter(2) and counter(1) and counter(0);

mem_nrw = 0;

rem_nrw = (not(counter(2)) and not(counter(1)) and not(counter(0))) or (not(counter(2)) and counter(1) and counter(0)) or (counter(2) and not(counter(1)) and counter(0));

rdm_nrw = (counter(2) and not(counter(0))) or (not(counter(2)) and not(counter(1)) and counter(0));

ri_nrw = not(counter(2)) and counter(1) and not(counter(0));

NOT:

Tabela:

barr/Inc	barr/PC	ula_op	pc_rw	ac_rw	mem_rw	rem_rw	rdm_rw	ri_rw
1	1	100	0	0	0	1	0	0
1	1	100	1	0	0	0	1	0
1	1	100	0	0	0	0	0	1
1	1	100	0	0	0	0	0	0
1	1	100	0	0	0	0	0	0
1	1	100	0	0	0	0	0	0
1	1	100	0	0	0	0	0	0
1	1	100	0	1	0	0	0	0

Simplificação:

nBarrInc = 1;

nBarrPc = 1;

ula(2) = 1;

ula(1) = 0;

ula(0) = 0;

pc_nrw = not(counter(2)) and not(counter(1)) and counter(0);

ac_rnw = counter(2) and counter(1) and counter(0);

mem_nrw = 0;

rem_nrw = not(counter(2)) and not(counter(1)) and not(counter(0));

rdm_nrw = not(counter(2)) and not(counter(1)) and counter(0);

ri_nrw = not(counter(2)) and counter(1) and not(counter(0));

JMP:

Tabela:

barr/Inc	barr/PC	ula_op	pc_rw	ac_rw	mem_rw	rem_rw	rdm_rw	ri_rw
1	1	000	0	0	0	1	0	0
1	1	000	1	0	0	0	1	0
1	1	000	0	0	0	0	0	1
1	1	000	0	0	0	1	0	0
1	1	000	0	0	0	0	1	0
0	1	000	1	0	0	0	0	0
1	1	000	0	0	0	0	0	0
1	1	000	0	0	0	0	0	0

Simplificação:

nBarrInc = not(counter(2)) or not(counter(0)) or counter(1);

nBarrPc = 1;

ula(2) = 0;

ula(1) = 0;

ula(0) = 0;

pc_nrw = (not(counter(1)) and counter(0));

ac_nrw = 0;

mem_nrw = 0;

rem_nrw = (not(counter(2)) and not(counter(1)) and not(counter(0))) or (not(counter(2)) and counter(1) and counter(0));

rdm_nrw = (not(counter(2)) and not(counter(1)) and counter(0)) or (counter(2) and not(counter(1)) and not(counter(0)));

ri_nrw = not(counter(2)) and counter(1) and not(counter(0));

JN e JZ:

Essas funções não possuem tabelas nem simplificações, pois são resultado de um multiplexador utilizando sinais de FlagNZ e a JMP já mostrada anteriormente.

HLT:

Tabela:

barr/Inc	barr/PC	ula_op	pc_rw	ac_rw	mem_rw	rem_rw	rdm_rw	ri_rw
0	0	000	0	0	0	0	0	0
0	0	000	0	0	0	0	0	0
0	0	000	0	0	0	0	0	0
0	0	000	0	0	0	0	0	0
0	0	000	0	0	0	0	0	0
0	0	000	0	0	0	0	0	0
0	0	000	0	0	0	0	0	0
0	0	000	0	0	0	0	0	0

Simplificação:

nBarrInc = 0;

nBarrPc = 0;

ula(2) = 0;

ula(1) = 0;

ula(0) = 0;

pc_nrw = 0;

ac_rnw = 0;

mem_nrw = 0;

rem_nrw = 0;

rdm_nrw = 0;

ri_nrw = 0;