Instruções do Computador Teórico NEANDER: Tabelas descritivas e suas respectivas expressões booleanas

1. NOP (*No Operation*):

NOP	b2	b1	b0	dec	barr/INC	barr/PC	ULA_op	PC_rw	AC_rw	MEM_rw	REM_rw	RDM_rw	RI_rw
BUSCA	0	0	0	0	1	1	000	0	0	0	1	0	0
	0	0	1	1	1	1	000	1	0	0	0	1	0
	0	1	0	2	1	1	000	0	0	0	0	0	1
EXECUÇÃO	0	1	1	3	1	1	000	0	0	0	0	0	0
	1	0	0	4	1	1	000	0	0	0	0	0	0
	1	0	1	5	1	1	000	0	0	0	0	0	0
	1	1	0	6	1	1	000	0	0	0	0	0	0
	1	1	1	7	1	1	000	0	0	0	0	0	0

Simplificações dos sinais de controle:

→ barrINC = '1'

→ barr/PC = '1'

→ ULA_op = "000"

→ PC_nrw = not b_2 and not b_1 and b_0

→ AC_nrw = '0'

→ MEM_nrw = '0'

→ REM_nrw = not b_2 and not b_1 and not b_0

→ RDM_nrw = not b_2 and not b_1 and b_0

→ RI_nrw = not b_2 and b_1 and not b_0

2. STA (*Store Accumulator*):

STA	b2	b1	b0	dec	barr/INC	barr/PC	ULA_op	PC_rw	AC_rw	MEM_rw	REM_rw	RDM_rw	RI_rw
BUSCA	0	0	0	0	1	1	000	0	0	0	1	0	0
	0	0	1	1	1	1	000	1	0	0	0	1	0
	0	1	0	2	1	1	000	0	0	0	0	0	1
EXECUÇÃO	0	1	1	3	1	1	000	0	0	0	1	0	0
	1	0	0	4	1	1	000	1	0	0	0	1	0
	1	0	1	5	1	0	000	0	0	0	1	0	0
	1	1	0	6	1	1	000	0	0	1	0	0	0
	1	1	1	7	1	1	000	0	0	0	0	0	0

Simplificações dos sinais de controle:

→ barrINC = '1'

→ barr/PC = not b_2 or b_1 or not b_0

→ ULA_op = "000"

→ PC_nrw = not b_1 and (b_2 xor b_0)

→ **AC_nrw = '0'**

→ **MEM_nrw = b_2 and b_1 and not b_0**

→ **REM_nrw = (not b_1 and (b_2 xnor b_0)) or (not b_2 and b_1 and b_0)**

→ **RDM_nrw = not b_1 and (b_2 xor b_0)**

→ **RI_nrw = not b_2 and b_1 and not b_0**

3. NOT (*Logical Not*):

NOT	b2	b1	b0	dec	barr/INC	barr/PC	ULA_op	PC_rw	AC_rw	MEM_rw	REM_rw	RDM_rw	RI_rw
BUSCA	0	0	0	0	1	1	100	0	0	0	1	0	0
	0	0	1	1	1	1	100	1	0	0	0	1	0
	0	1	0	2	1	1	100	0	0	0	0	0	1
EXECUÇÃO	0	1	1	3	1	1	100	0	0	0	0	0	0
	1	0	0	4	1	1	100	0	0	0	0	0	0
	1	0	1	5	1	1	100	0	0	0	0	0	0
	1	1	0	6	1	1	100	0	0	0	0	0	0
	1	1	1	7	1	1	100	0	1	0	0	0	0

Simplificações dos sinais de controle:

→ **barrINC = '1'**

→ **barr/PC = '1'**

→ **ULA_op = "100"**

→ **PC_nrw = not b_2 and not b_1 and b_0**

→ **AC_nrw = b_2 and b_1 and b_0**

→ **MEM_nrw = '0'**

→ **REM_nrw = not b_2 and not b_1 and not b_0**

→ **RDM_nrw = not b_2 and not b_1 and b_0**

→ **RI_nrw = not b_2 and b_1 and not b_0**

4. ADD (*Add*):

ADD	b2	b1	b0	dec	barr/INC	barr/PC	ULA_op	PC_rw	AC_rw	MEM_rw	REM_rw	RDM_rw	RI_rw
BUSCA	0	0	0	0	1	1	001	0	0	0	1	0	0
	0	0	1	1	1	1	001	1	0	0	0	1	0
	0	1	0	2	1	1	001	0	0	0	0	0	1
EXECUÇÃO	0	1	1	3	1	1	001	0	0	0	1	0	0
	1	0	0	4	1	1	001	1	0	0	0	1	0
	1	0	1	5	1	0	001	0	0	0	1	0	0
	1	1	0	6	1	1	001	0	0	0	0	1	0
	1	1	1	7	1	1	001	0	1	0	0	0	0

Simplificações dos sinais de controle:

→ barrINC = '1'

→ barr/PC = not b_2 or b_1 or not b_0

→ ULA_op = "001"

→ PC_nrw = not b_1 and (b_2 xor b_0)

→ AC_nrw = b_2 and b_1 and b_0

→ MEM_nrw = '0'

→ REM_nrw = (not b_1 and (b_2 xnor b_0)) or (not b_2 and b_1 and b_0)

→ RDM_nrw = (b_2 and not b_0) or (not b_2 and not b_1 and b_0)

→ RI_nrw = not b_2 and b_1 and not b_0

5. OR (*Logical Or*):

OR	b2	b1	b0	dec	barr/INC	barr/PC	ULA_op	PC_rw	AC_rw	MEM_rw	REM_rw	RDM_rw	RI_rw
	0	0	0	0	1	1	010	0	0	0	1	0	0
BUSCA	0	0	1	1	1	1	010	1	0	0	0	1	0
	0	1	0	2	1	1	010	0	0	0	0	0	1
	0	1	1	3	1	1	010	0	0	0	1	0	0
	1	0	0	4	1	1	010	1	0	0	0	1	0
EXECUÇÃO	1	0	1	5	1	0	010	0	0	0	1	0	0
	1	1	0	6	1	1	010	0	0	0	0	1	0
	1	1	1	7	1	1	010	0	1	0	0	0	0

Simplificações dos sinais de controle:

→ barrINC = '1'

→ barr/PC = not b_2 or b_1 or not b_0

→ ULA_op = "010"

→ PC_nrw = not b_1 and (b_2 xor b_0)

→ AC_nrw = b_2 and b_1 and b_0

→ MEM_nrw = '0'

→ REM_nrw = (not b_1 and (b_2 xnor b_0)) or (not b_2 and b_1 and b_0)

→ RDM_nrw = (b_2 and not b_0) or (not b_2 and not b_1 and b_0)

→ RI_nrw = not b_2 and b_1 and not b_0

6. AND (*Logical And*):

AND	b2	b1	b0	dec	barr/INC	barr/PC	ULA_op	PC_rw	AC_rw	MEM_rw	REM_rw	RDM_rw	RI_rw
	0	0	0	0	1	1	011	0	0	0	1	0	0
BUSCA	0	0	1	1	1	1	011	1	0	0	0	1	0
	0	1	0	2	1	1	011	0	0	0	0	0	1
	0	1	1	3	1	1	011	0	0	0	1	0	0
	1	0	0	4	1	1	011	1	0	0	0	1	0
EXECUÇÃO	1	0	1	5	1	0	011	0	0	0	1	0	0
	1	1	0	6	1	1	011	0	0	0	0	1	0
	1	1	1	7	1	1	011	0	1	0	0	0	0

Simplificações dos sinais de controle:

→ barrINC = '1'

→ barr/PC = not b_2 or b_1 or not b_0

→ ULA_op = "011"

→ PC_nrw = not b_1 and (b_2 xor b_0)

→ AC_nrw = b_2 and b_1 and b_0

→ MEM_nrw = '0'

→ REM_nrw = (not b_1 and (b_2 xnor b_0)) or (not b_2 and b_1 and b_0)

→ RDM_nrw = (b_2 and not b_0) or (not b_2 and not b_1 and b_0)

→ RI_nrw = not b_2 and b_1 and not b_0

7. JMP, JN e JZ (caso N = 1 e Z = 1) (*Jump*, *Jump if Negative* e *Jump if Zero*):

JMP	b2	b1	b0	dec	barr/INC	barr/PC	ULA_op	PC_rw	AC_rw	MEM_rw	REM_rw	RDM_rw	RI_rw
	0	0	0	0	1	1	000	0	0	0	1	0	0
BUSCA	0	0	1	1	1	1	000	1	0	0	0	1	0
	0	1	0	2	1	1	000	0	0	0	0	0	1
	0	1	1	3	1	1	000	0	0	0	1	0	0
	1	0	0	4	1	1	000	0	0	0	0	1	0
EXECUÇÃO	1	0	1	5	0	1	000	1	0	0	0	0	0
	1	1	0	6	1	1	000	0	0	0	0	0	0
	1	1	1	7	1	1	000	0	0	0	0	0	0

Simplificações dos sinais de controle:

→ barrINC = not b_2 or b_1 or not b_0

→ barr/PC = '1'

→ ULA_op = "000"

→ PC_nrw = not b_1 and b_0

→ **AC_nrw = '0'**

→ **MEM_nrw = '0'**

→ **REM_nrw = not b_2 and (b_1 xnor b_0)**

→ **RDM_nrw = not b_1 and (b_2 xor b_0)**

→ **RI_nrw = not b_2 and b_1 and not b_0**

8. JN e JZ (caso N = 0 e Z = 0) (*Jump*, *Jump if Negative* e *Jump if Zero*):

JN (N=0)	b2	b1	b0	dec	barr/INC	barr/PC	ULA_op	PC_rw	AC_rw	MEM_rw	REM_rw	RDM_rw	RI_rw
	0	0	0	0	1	1	000	0	0	0	1	0	0
BUSCA	0	0	1	1	1	1	000	1	0	0	0	1	0
	0	1	0	2	1	1	000	0	0	0	0	0	1
	0	1	1	3	1	1	000	1	0	0	0	0	0
	1	0	0	4	1	1	000	0	0	0	0	0	0
EXECUÇÃO	1	0	1	5	1	1	000	0	0	0	0	0	0
	1	1	0	6	1	1	000	0	0	0	0	0	0
	1	1	1	7	1	1	000	0	0	0	0	0	0

JZ (Z =0)	b2	b1	b0	dec	barr/INC	barr/PC	ULA_op	PC_rw	AC_rw	MEM_rw	REM_rw	RDM_rw	RI_rw
	0	0	0	0	1	1	000	0	0	0	1	0	0
BUSCA	0	0	1	1	1	1	000	1	0	0	0	1	0
	0	1	0	2	1	1	000	0	0	0	0	0	1
	0	1	1	3	1	1	000	1	0	0	0	0	0
	1	0	0	4	1	1	000	0	0	0	0	0	0
EXECUÇÃO	1	0	1	5	1	1	000	0	0	0	0	0	0
	1	1	0	6	1	1	000	0	0	0	0	0	0
	1	1	1	7	1	1	000	0	0	0	0	0	0

Simplificações dos sinais de controle:

→ **barrINC = '1'**

→ **barr/PC = '1'**

→ **ULA_op = "000"**

→ **PC_nrw = not b_2 and b_0**

→ **AC_nrw = '0'**

→ **MEM_nrw = '0'**

→ **REM_nrw = not b_2 and not b_1 and not b_0**

→ **RDM_nrw = not b_2 and not b_1 and b_0**

→ **RI_nrw = not b_2 and b_1 and not b_0**

9. HLT (*Halt*):

HLT	b2	b1	b0	dec	barr/INC	barr/PC	ULA_op	PC_rw	AC_rw	MEM_rw	REM_rw	RDM_rw	RI_rw
	0	0	0	0	0	0	000	0	0	0	0	0	0
BUSCA	0	0	1	1	0	0	000	0	0	0	0	0	0
	0	1	0	2	0	0	000	0	0	0	0	0	0
	0	1	1	3	0	0	000	0	0	0	0	0	0
	1	0	0	4	0	0	000	0	0	0	0	0	0
EXECUÇÃO	1	0	1	5	0	0	000	0	0	0	0	0	0
	1	1	0	6	0	0	000	0	0	0	0	0	0
	1	1	1	7	0	0	000	0	0	0	0	0	0

Simplificações dos sinais de controle:

→ barrINC = '0'

→ barr/PC = '0'

→ ULA_op = "000"

→ PC_nrw = '0'

→ AC_nrw = '0'

→ MEM_nrw = '0'

→ REM_nrw = '0'

→ RDM_nrw = '0'

→ RI_nrw = '0'

10. LDA (*Load Accumulator*):

LDA	b2	b1	b0	dec	barr/INC	barr/PC	ULA_op	PC_rw	AC_rw	MEM_rw	REM_rw	RDM_rw	RI_rw
	0	0	0	0	1	1	000	0	0	0	1	0	0
BUSCA	0	0	1	1	1	1	000	1	0	0	0	1	0
	0	1	0	2	1	1	000	0	0	0	0	0	1
	0	1	1	3	1	1	000	0	0	0	1	0	0
	1	0	0	4	1	1	000	1	0	0	0	1	0
EXECUÇÃO	1	0	1	5	1	0	000	0	0	0	1	0	0
	1	1	0	6	1	1	000	0	0	0	0	1	0
	1	1	1	7	1	1	000	0	1	0	0	0	0

Simplificações dos sinais de controle:

→ barrINC = '1'

→ barr/PC = '1'

→ ULA_op = "000"

→ PC_nrw = not b_1 and (b_2 xor b_0)

→ AC_nrw = b_2 and b_1 and b_0

→ MEM_nrw = '0'

→ REM_nrw = (not b_1 and (b_2 xnor b_0)) or (not b_2 and b_1 and b_0)

→ RDM_nrw = (b_2 and not b_0) or (not b_2 and not b_1 and b_0)

→ RI_nrw = not b_2 and b_1 and not b_0