UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

INSTITUTO DE INFORMATICA

Aula 31/1/23 - Sistemas Digitais

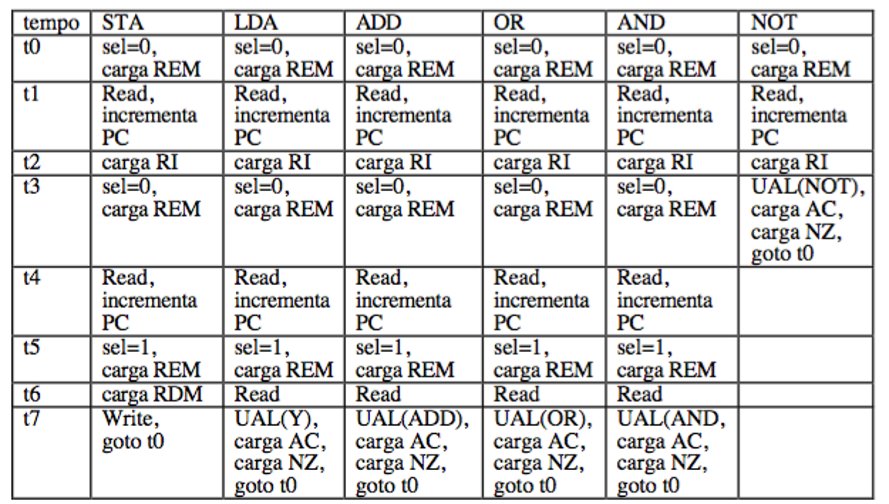
Prof. Fernanda Kastensmit

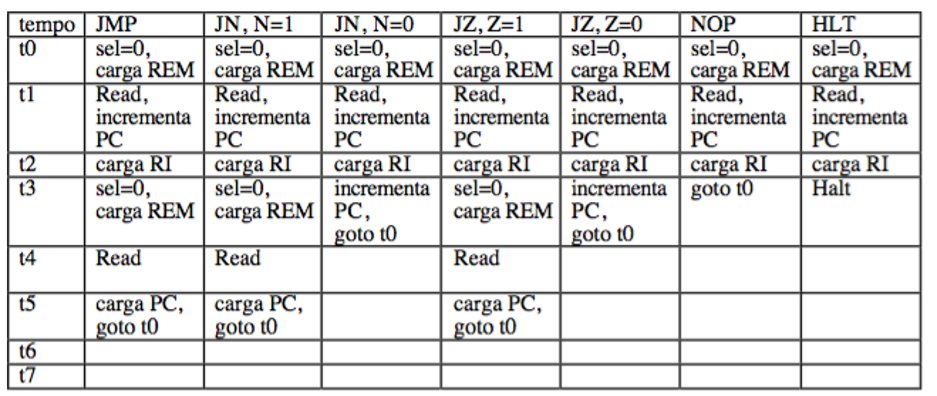
**Objetivo:** projetar e descrever em VHDL o processador Ahmes, implementar 2 programas em sua memória e mostrar através de simulação lógica sem e com atraso o funcionamento.

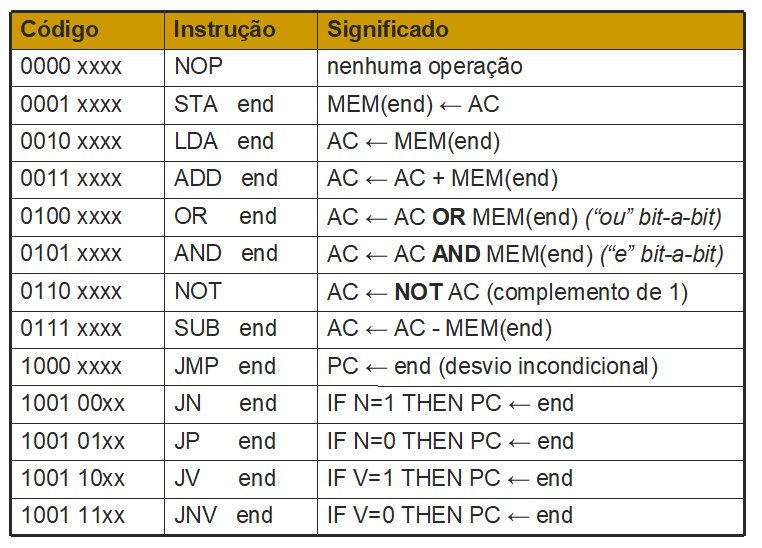
**AULA 3 (31/1/2022)**

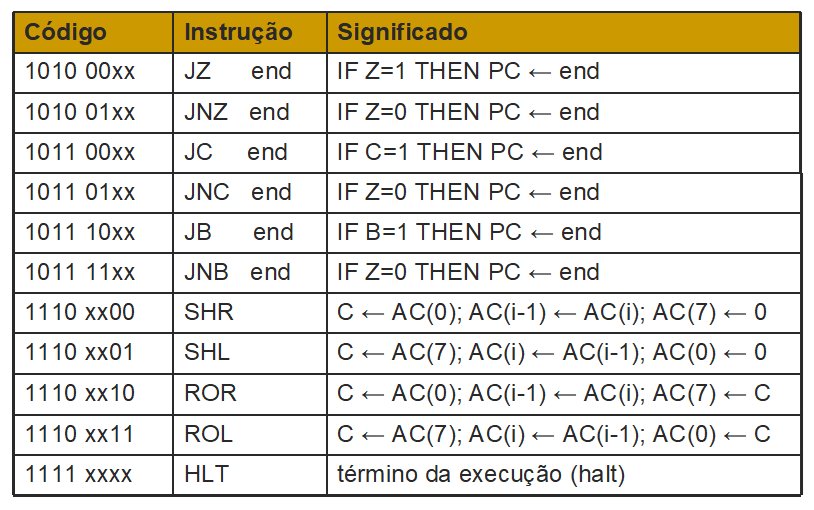
Descrever a parte de controle do AHMES em VHDL como uma maquina de estados.

**Passo 1:** Dada as tabelas com as instruções do Neander por estado da máquina de estrados









Completar a tabela a seguir com as instruções AHMES que não tem no NEANDER e as instruções novas de Desvio

| Tempo | SHR | SHL | ROR | ROL | SUB | **JV, V=1** | **JNV, V=0** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| T0 | sel = 0,  load REM | Sel = 0,  load REM | Sel = 0,  load REM | Sel = 0,  load REM | Sel = 0,  load REM | Sel = 0,  load REM | Sel = 0,  load REM |
| T1 | Read,  Inc PC | Read,  Inc PC | Read,  Inc PC | Read,  Inc PC | Read,  Inc PC | Read,  Inc PC | Read,  Inc PC |
| T2 | load RI | load RI | load RI | load RI | Load RI | Load RI | Load RI |
| T3 | ULA(SHR),  load AC,  load flags,  Goto t0 | ULA(SHL),  load AC,  load flags,  Goto t0 | ULA(ROR),  load AC,  load flags,  Goto t0 | ULA(ROL),  load AC,  load flags,  Goto t0 | Sel = 0,  Load REM | Sel = 0,  Load REM | Sel = 0,  Load REM |
| T4 |  |  |  |  | Read,  Inc PC | Read | Read |
| T5 |  |  |  |  | Sel = 1,  Load REM | Load PC,  Goto t0 | Load PC,  Goto t0 |
| T6 |  |  |  |  | Read |  |  |
| T7 |  |  |  |  | ULA(SUB), load AC,  Load NZBC |  |  |

| Tempo | **JNZ, Z=0** | **JP, Z=0** | **JC, C=1** | **JNC, C=0** | **JB, B=1** | **JNB, B=1** |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| T0 | Sel = 0,  load REM | Sel = 0,  load REM | Sel = 0,  load REM | Sel = 0,  load REM | Sel = 0,  load REM | Sel = 0,  load REM |  |
| T1 | Read,  Inc PC | Read,  Inc PC | Read,  Inc PC | Read,  Inc PC | Read,  Inc PC | Read,  Inc PC |  |
| T2 | Load RI | Load RI | Load RI | Load RI | Load RI | Load RI |  |
| T3 | Sel = 0,  Load REM | Sel = 0,  Load REM | Sel = 0,  Load REM | Sel = 0,  Load REM | Sel = 0,  Load REM | Sel = 0,  Load REM |  |
| T4 | Read | Read | Read | Read | Read | Read |  |
| T5 | Load PC,  Goto t0 | Load PC,  Goto t0 | Load PC,  Goto t0 | Load PC,  Goto t0 | Load PC,  Goto t0 | Load PC,  Goto t0 |  |
| T6 |  |  |  |  |  |  |  |
| T7 |  |  |  |  |  |  |  |

*library IEEE;*

*use IEEE.STD\_LOGIC\_1164.ALL;*

*use IEEE.NUMERIC\_STD.ALL;*

*entity control\_ahmes is*

*Port(*

*CLOCK: in std\_logic;*

*RESET: in std\_logic;*

*-- controle dos registradores*

*inc\_pc: out std\_logic;*

*load\_ac: out std\_logic;*

*load\_pc: out std\_logic;*

*load\_REM: out std\_logic;*

*load\_RDM: out std\_logic;*

*load\_RI: out std\_logic;*

*load\_N: out std\_logic;*

*load\_Z: out std\_logic;*

*load\_V: out std\_logic;*

*load\_C: out std\_logic;*

*load\_B: out std\_logic;*

*-- seletores*

*sel\_ULA: out std\_logic\_vector(3 downto 0);*

*sel\_MUXREM: out std\_logic;*

*sel\_MUXRDM: out std\_logic;*

*-- controle da memoria*

*--mem\_read: out std\_logic; -- nao utilizado*

*mem\_write: out std\_logic;*

*-- flags de estado*

*reg\_N: in STD\_LOGIC;*

*reg\_Z: in STD\_LOGIC;*

*reg\_V: in STD\_LOGIC;*

*reg\_C: in STD\_LOGIC;*

*reg\_B: in STD\_LOGIC;*

*-- flags das operacoes*

*op\_nop: in std\_logic;*

*op\_sta: in std\_logic;*

*op\_lda: in std\_logic;*

*op\_add: in std\_logic;*

*op\_or: in std\_logic;*

*op\_and: in std\_logic;*

*op\_not: in std\_logic;*

*op\_sub: in std\_logic;*

*op\_jmp: in std\_logic;*

*op\_jn: in std\_logic;*

*op\_jp: in std\_logic;*

*op\_jv: in std\_logic;*

*op\_jnv: in std\_logic;*

*op\_jz: in std\_logic;*

*op\_jnz: in std\_logic;*

*op\_jc: in std\_logic;*

*op\_jnc: in std\_logic;*

*op\_jb: in std\_logic;*

*op\_jnb: in std\_logic;*

*op\_shr: in std\_logic;*

*op\_shl: in std\_logic;*

*op\_ror: in std\_logic;*

*op\_rol: in std\_logic;*

*op\_hlt: in std\_logic);*

*end control\_ahmes;*

*architecture Behavioral of control\_ahmes is*

*type state\_type is (S0, S1, S2, S3, S4, S5, S6, S7, S8);*

*signal next\_state, current\_state: state\_type;*

*begin*

*process(CLOCK, RESET) -- controle de estados*

*begin*

*if(RESET = '1') then*

*current\_state <= S0;*

*elsif(rising\_edge(CLOCK)) then*

*current\_state <= next\_state;*

*else*

*current\_state <= current\_state;*

*end if;*

*end process;*

*process(*

*next\_state,*

*-- flags gerais*

*reg\_N,*

*reg\_Z,*

*reg\_V,*

*reg\_C,*

*reg\_B,*

*--flags das ops*

*op\_nop,*

*op\_sta,*

*op\_lda,*

*op\_add,*

*op\_or,*

*op\_and,*

*op\_not,*

*op\_sub,*

*op\_jmp,*

*op\_jn,*

*op\_jp,*

*op\_jv,*

*op\_jnv,*

*op\_jz,*

*op\_jnz,*

*op\_jc,*

*op\_jnc,*

*op\_jb,*

*op\_jnb,*

*op\_shr,*

*op\_shl,*

*op\_ror,*

*op\_rol,*

*op\_hlt)*

*begin*

*-- reseta sinais de carga*

*inc\_pc <= '0';*

*load\_ac <= '0';*

*load\_pc <= '0';*

*load\_N <= '0';*

*load\_Z <= '0';*

*load\_V <= '0';*

*load\_C <= '0';*

*load\_B <= '0';*

*load\_RDM <= '0';*

*load\_REM <= '0';*

*sel\_MUXREM <= '0';*

*sel\_MUXRDM <= '0';*

*sel\_ULA <= "0000";*

*mem\_write <= '0';*

*case current\_state is*

*when S0 =>*

*load\_RDM <= '1';*

*next\_state <= S1;*

*when S1 =>*

*load\_REM <= '0';*

*inc\_pc <= '0';*

*next\_state <= S2;*

*when S2 =>*

*load\_RDM <= '1';*

*inc\_pc <= '1';*

*next\_state <= S3;*

*when S3 =>*

*inc\_pc <= '0';*

*load\_RDM <= '0';*

*if(op\_nop = '1') then -- NOP*

*next\_state <= S0;*

*elsif(op\_not = '1') then -- NOT*

*sel\_ULA <= "0011";*

*load\_AC <= '1';*

*load\_N <= '1';*

*load\_Z <= '1';*

*load\_V <= '1';*

*load\_C <= '1';*

*load\_B <= '1';*

*next\_state <= S0;*

*elsif(op\_jn = '1' and reg\_N = '0') then -- JN quando n=0*

*inc\_PC <= '1';*

*next\_state <= S0;*

*elsif(op\_jp = '1' and reg\_N = '1') then -- JP quando n=1*

*inc\_PC <= '1';*

*next\_state <= S0;*

*elsif(op\_jv = '1' and reg\_V = '0') then -- JV quando v=0*

*inc\_PC <= '1';*

*next\_state <= S0;*

*elsif(op\_jv = '1' and reg\_V = '1') then -- JNV quando v=1*

*inc\_PC <= '1';*

*next\_state <= S0;*

*elsif(op\_jz = '1' and reg\_Z = '0') then -- JZ quando z=0*

*inc\_PC <= '1';*

*next\_state <= S0;*

*elsif(op\_jz = '1' and reg\_Z = '1') then -- JNZ quando z=1*

*inc\_PC <= '1';*

*next\_state <= S0;*

*elsif(op\_jc = '1' and reg\_C = '0') then -- JC quando c=0*

*inc\_PC <= '1';*

*next\_state <= S0;*

*elsif(op\_jb = '1' and reg\_C = '1') then -- JNC quando c=1*

*inc\_PC <= '1';*

*next\_state <= S0;*

*elsif(op\_jb = '1' and reg\_B = '0') then -- JB quando b=0*

*inc\_PC <= '1';*

*next\_state <= S0;*

*elsif(op\_jb = '1' and reg\_B = '1') then -- JNB quando b=1*

*inc\_PC <= '1';*

*next\_state <= S0;*

*elsif(op\_shr = '1') then -- SHR*

*sel\_ULA <= "0101";*

*load\_AC <= '1';*

*load\_N <= '1';*

*load\_Z <= '1';*

*load\_V <= '1';*

*load\_C <= '1';*

*load\_B <= '1';*

*next\_state <= S0;*

*elsif(op\_shr = '1') then -- SHL*

*sel\_ULA <= "0101";*

*load\_AC <= '1';*

*load\_N <= '1';*

*load\_Z <= '1';*

*load\_V <= '1';*

*load\_C <= '1';*

*load\_B <= '1';*

*next\_state <= S0;*

*elsif(op\_shr = '1') then -- ROR*

*sel\_ULA <= "0101";*

*load\_AC <= '1';*

*load\_N <= '1';*

*load\_Z <= '1';*

*load\_V <= '1';*

*load\_C <= '1';*

*load\_B <= '1';*

*next\_state <= S0;*

*elsif(op\_shr = '1') then -- ROL*

*sel\_ULA <= "0101";*

*load\_AC <= '1';*

*load\_N <= '1';*

*load\_Z <= '1';*

*load\_V <= '1';*

*load\_C <= '1';*

*load\_B <= '1';*

*next\_state <= S0;*

*elsif(op\_hlt = '1') then -- HLT*

*inc\_PC <= '0';*

*next\_state <= S8;*

*else -- qualquer outra operacao*

*sel\_MUXREM <= '0';*

*load\_REM <= '1';*

*next\_state <= S4;*

*end if;*

*when S4 =>*

*sel\_MUXREM <= '0';*

*inc\_PC <= '0';*

*load\_AC <= '0';*

*load\_REM <= '0';*

*load\_AC <= '0';*

*load\_N <= '0';*

*load\_Z <= '0';*

*load\_V <= '0';*

*load\_C <= '0';*

*load\_B <= '0';*

*if(op\_sta = '1' or*

*op\_lda = '1' or*

*op\_add = '1' or*

*op\_or = '1' or*

*op\_and = '1' or*

*op\_sub = '1' or*

*op\_jmp = '1' or*

*op\_jn = '1' or*

*op\_jp = '1' or*

*op\_jv = '1' or*

*op\_jnv = '1' or*

*op\_jz = '1' or*

*op\_jnz = '1' or*

*op\_jc = '1' or*

*op\_jnc = '1' or*

*op\_jb = '1' or*

*op\_jnb = '1') then*

*inc\_PC <= '1';*

*end if;*

*next\_state <= S5;*

*when S5 =>*

*inc\_PC <= '0';*

*if(op\_sta = '1' or*

*op\_lda = '1' or*

*op\_add = '1' or*

*op\_or = '1' or*

*op\_and = '1' or*

*op\_sub = '1' or*

*op\_jmp = '1' or*

*op\_jn = '1' or*

*op\_jp = '1' or*

*op\_jv = '1' or*

*op\_jnv = '1' or*

*op\_jz = '1' or*

*op\_jnz = '1' or*

*op\_jc = '1' or*

*op\_jnc = '1' or*

*op\_jb = '1' or*

*op\_jnb = '1') then*

*sel\_MUXREM <= '1';*

*load\_REM <= '1';*

*next\_state <= S6;*

*else*

*load\_PC <= '1';*

*next\_state <= S0;*

*end if;*

*when S6 =>*

*inc\_PC <= '0';*

*sel\_MUXREM <= '0';*

*load\_REM <= '0';*

*load\_PC <= '0';*

*next\_state <= S7;*

*if(op\_sta = '1') then -- STA*

*load\_RDM <= '1';*

*end if;*

*when S7 =>*

*if(op\_sta = '1') then*

*mem\_write <= '1';*

*elsif(op\_lda = '1') then*

*sel\_ULA <= "0100"; -- NOP(ULA Y)*

*elsif(op\_add = '1') then*

*sel\_ULA <= "0000";*

*elsif(op\_or = '1') then*

*sel\_ULA <= "0001";*

*elsif(op\_and = '1') then*

*sel\_ULA <= "0010";*

*elsif(op\_not = '1') then*

*sel\_ULA <= "0011";*

*elsif(op\_sub = '1') then*

*sel\_ULA <= "0100";*

*end if;*

*when S8 =>*

*next\_state <= S8;*

*when others =>*

*next\_state <= S0;*

*end case;*

*end process;*

*end Behavioral;*

**\*\*\*ENTREGAR DIA 03/2/2023\*\*\*\* Editando esse DOC e submetendo no MS-TEAMS**