Atividade 4 Salto incondicional e Desvio condicional (a)

Modificações necessárias no código:

(1) Acrescentar os estados **sJMP** e **sBRANCH** na lista de estados

```
type stateType is (sFETCH, sEXE, sREAD,
```

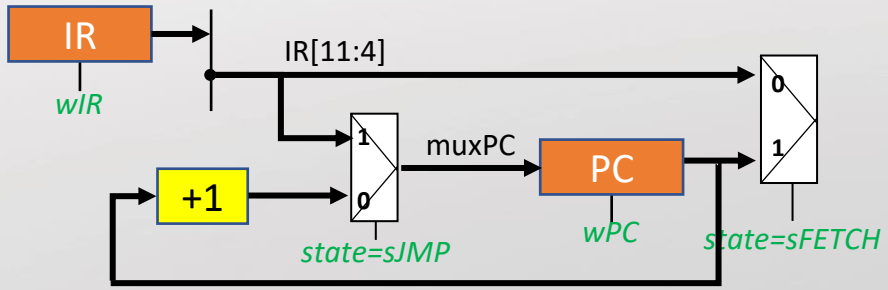
Instrução	15:12	11:8	7:4	3:0
iREAD	0	endereço		RS2
iWRITE	1	endereço		RS2
iJMP	2	endereço		0
iBRANCH	3	endereço		RS2

(2) Completar o *muxPC* para considerar os saltos. Note que os registradores têm 16 bits, logo é necessário concatenar com 8 bits zerados

```
muxPC <= x"00" & IR(11 downto 4) when state = sJMP or (state = sBRANCH and RS2(0) = '1') else PC + 1;
```

(3) Decodificar as instruções **iJMP** e **iBRANCH**

```
inst <= iREAD      when ir(15 downto 12) = x"0" else
        iWRITE     when ir(15 downto 12) = x"1" else
        iJMP
        iBRANCH
```



(4) Acrescentar no **wPC** os estados **sJMP** e **sBRANCH**

```
wPC  <= '1' when state = sREAD or state = sALU or
```

(5) Acrescentar na FSM os estados **sJMP** e **sBRANCH**

```
elsif inst = iREAD then
    state <= sREAD;
elsif inst = iWRITE then
    state <= sWRITE;
elsif inst = iJMP then
```

Atividade 4: Salto incondicional e Desvio condicional (b)

Lembrando: o R2 tem $(0001)_{16}$

Após o armazenamento de R2:

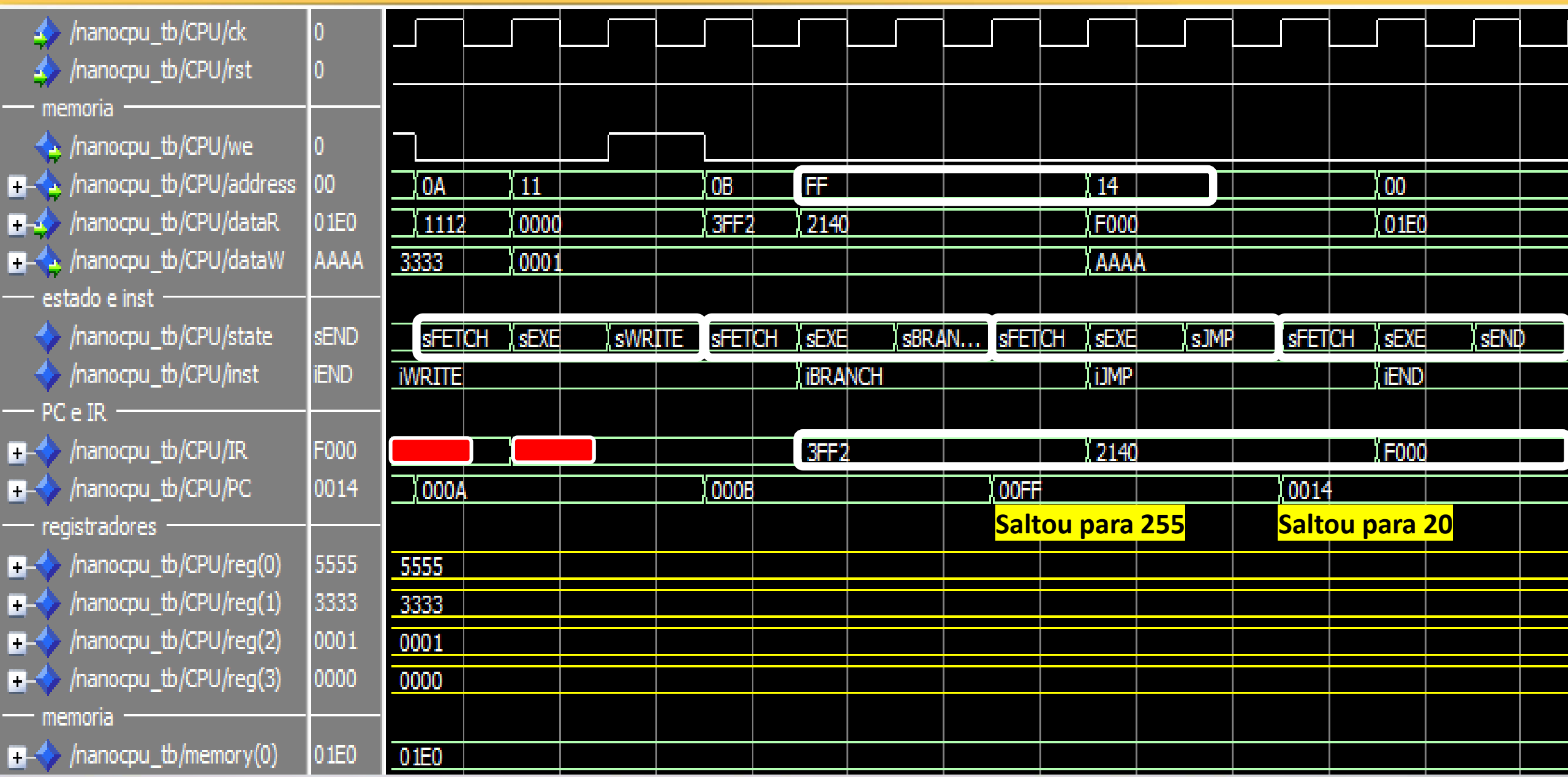
- Desvio condicional para endereço 255 se R2=1 (`11 => X"3FF2"`)
- No endereço 255 desvia para endereço 20 $(14)_{16}$ (`255 => X"2140"`)
- No endereço 32 termina o programa (`20 => X"F000"`)

Instrução	15:12	11:8	7:4	3:0
iREAD	0	endereço		RS2
iWRITE	1	endereço		RS2
iJMP	2	endereço		0
iBRANCH	3	endereço		RS2

```
    10 => .....,        -- store R2 em 11
    11 => X"3FF2",      -- desvia para 255 (xFF) se R2=1
    20 => X"F000",      -- FIM
    30 => X"1111",
    31 => X"2222",
    32 => X"3333",
    33 => X"4444",
    255=> X"2140",      -- salto incondicional para 20 (x14)
    others => (others => '0')
);
```

PUCRS ESCOLA POLITÉCNICA

Atividade 4: Salto incondicional e Desvio condicional (c)



Atividade 5 Incluir instruções INC e DEC (a)

- Incluir duas instruções – **iINC** e **iDEC**
- Usar os campos 8 e 9 para as instruções
  - iINC: Rt ← RS1 +1, exemplo X"8110" R1 ← R1 +1
  - iDEC: Rt ← RS1 - 1, exemplo X"9330" R3 ← R3 – 1

Instrução	15:12	11:8	7:4	3:0	Operação
iREAD	0	endereço		RS2	RS2 ← PMEM(end)
iWRITE	1	endereço		RS2	PMEM(end) ← RS2
iJMP	2	endereço		0	PC ← end
iBRANCH	3	endereço		RS2	PC ← end **se** RS2=1
iXOR	4	**Rt**	RS1	RS2	Rt ← RS1 xor RS2
iSUB	5	**Rt**	RS1	RS2	Rt ← RS1 - RS2
iADD	6	**Rt**	RS1	RS2	Rt ← RS1 + RS2
iLESS	7	**Rt**	RS1	RS2	Rt ← 1 **se** RS1 < RS2 senão 0
...	...				*a ser incluído pelos alunos*
iEND	15 $(F)_{16}$	0	0	0	termina a execução

- Ações:

1. Inserir iINC e iDEC na lista de instruções instType

```
type instType is (iREAD, iWRITE, iJMP, iBRANCH,
```

2. Decodificar INC e DEC

```
inst <= iREAD      when ir(15 downto 12) = x"0" else
        iWRITE     when ir(15 downto 12) = x"1" else
        iLESS      when ir(15 downto 12) = x"7" else
        iINC                                    else
        iDEC                                    else
        iEND;
```

3. Alterar a ALU

```
outalu <=   RS2           when inst = iWRITE else
            RS1 xor RS2   when inst = iXOR   else
            RS1 - RS2     when inst = iSUB   else
            less          when inst = iLESS  else
```

4. Programa de teste

```
11 => X"3FF2",   -- salta para 255 (xFF) se R2=1
20 => X"8000",   -- INC R0
21 => X"8110",   -- INC R1
22 => X"9220",   -- DEC R2
23 => X"9330",   -- DEC R3
24 => X"F000",
....
```

PUCRS ESCOLA POLITÉCNICA

Atividade 5: Incluir instruções INC e DEC (b)