Relatório de Arquitetura de Computadores

Primeiro projeto prático

Professores :

Dionísio Barros
Sofia Inácio
Dino Vasconcelos
Pedro Camacho

Trabalho realizado por :

Paulo Alexandre Rodrigues Alves Nº2120722
Renato Gabriel Silva Pêssego Nº2121922

Funchal, março de 2024

Índice

1.Introdução

Este trabalho serve como ponto de partida da cadeira de Arquitetura de Computadores. O desafio proposto foi a construção de um processador (PEPE-8), uma peça fundamental na sociedade atual. Além disso, este processador, em conjunto com outros dois módulos, será integrado para formar a placa-mãe

O principal objetivo deste projeto foi conseguir implementar um processador em linguagem de descrição de hardware (VHDL), utilizando a “ISE Design Suite” da Xilinx para simulação. Após essa etapa crucial de simulação, tivemos a oportunidade de dar continuidade ao projeto, escolhendo uma FPGA entre as disponíveis (SPARTAN 3E, SPARTAN 6, ARTIX 7 e NEXYS A7).

Este processo não apenas enfatiza a aplicação prática dos conceitos aprendidos, mas também proporciona uma transição significativa do ambiente de simulação para o mundo real das FPGAs. A escolha da FPGA oferece uma experiência prática, permitindo que os alunos compreendam os desafios e as considerações práticas associadas à implementação de um processador.

2.Objetivos

- Dotar os alunos de conhecimentos sobre o funcionamento de um processador e como o construir de raiz, sendo este de 8 bits.
- Integrar o processador na placa-mãe com suplemento de outros dois módulos.
- Fazer as respectivas simulações no “ISIM”
- Implementar o que foi feito no “ISE” da XLINX numa “FPGA”
- Refletir de como o processador é uma peça fulcral na elaboração de computadores/sistemas

3.Desenvolvimento

Primeiramente, construímos todos os módulos com as respectivas entradas e saídas. Começando no periférico de entrada, e seguindo o guião até à memória de dados.

No **periférico de entrada** o utilizador introduz os dados que serão utilizados para, posteriormente, realizar as operações com estes. Este periférico é controlado pelo sinal ESCR_P, de 1 bit, que quando está a '0' faz com que seja realizada uma leitura aos dados de entrada, PIN, de 8 bits, colocando-os na saída do módulo, Dados_IN, de 8 bits.

No **periférico de saída** o utilizador vê os resultados dos programas executados pelo processador. Este módulo é também controlado pelo sinal ESCR_P, que quando está a '1', na transição ascendente do relógio (clk), escreve no sinal de saída, POUT, de 8 bits, o valor do sinal à entrada do módulo, Operando1, de 8 bits.

O **multiplexer do banco de registos (Mux R)** é responsável por encaminhar um sinal dos quatro disponíveis, de 8 bits, à sua entrada, sendo estes, Constante, Dados_M, Dados_IN e Resultado, para apresentar na sua saída, Dados_R, também de 8 bits. O sinal que será encaminhado depende do valor na entrada de seleção, o sinal SEL_Dados, de 2 bits.

O **banco de registos** possui 8 registos de 8 bits cada (R0, R1, R2, R3, R4, R5, R6 e R7), sendo a escrita nestes registos controlada pelo sinal ESCR_R, de 2 bits, que quando o seu bit menos significativo estiver a '1' permite que valor no sinal de entrada, Dados_R, seja guardado no registo especificado pelos três bits menos significativos do sinal SEL_R, de 6 bits, isto quando ocorre uma transição ascendente do sinal de relógio (clk).
Além da escrita, o banco de registos efetua leituras continuamente. A saída Operando1, apresentará o valor do registo especificado pelos pelos três bits menos significativos do sinal SEL_R, e a saída Operando2, também de 8 bits, apresentará o valor do registo especificado pelos três bits mais significativos do sinal SEL_R.

A **unidade lógica e aritmética (ALU)** é responsável pela realização das operações aritméticas e lógicas. Neste projeto a ALU desenvolvida é capaz de realizar as operações apresentadas no guia, com os dois sinais de entrada de 8 bits, Operando1 e Operando2, representando números inteiros com sinal, sendo que os sinais de saída da ALU são determinados pelo sinal de seleção SEL_ALU de 4 bits. A saída, Resultado, será atualizada no caso das operações de soma, subtração, AND, NAND, OR, NOR, XOR, XNOR. Já a saída E_FLAG, de 5 bits, será atualizada apenas quando é executada uma comparação. Cada um dos seus bits indica o resultado de cada uma das cinco comparações.

O **registo de flags** tem o objetivo de guardar o valor do sinal de entrada E_FLAG, de 5 bits, quando o sinal de relógio (clk) estiver na transição ascendente e quando o bit mais significativo do sinal de entrada ESCR_R, estiver a '1'. Este

registo encaminha um dos bits guardados para o sinal de saída, S_FLAG, de 1 bit. Esse bit é determinado através do sinal de seleção SEL_FLAG, de 3 bits.

O **contador de programa (PC)** indica qual é a posição atual da sequência de execução de um programa. Assim, este envia, na transição ascendente de cada ciclo de relógio, a sua saída Endereco, de 8 bits, à Memória de Instruções. A sequência de execução será incrementada de um em um endereço, caso a entrada ESCR_PC, de 1 bit, esteja a '0', caso contrário (ESCR_PC a '1'), a saída do PC corresponderá ao valor da entrada Constante, de 8 bits, e ocorrerá um salto para o endereço de instrução indicado por esse valor. A entrada Reset, de 1 bit, quando ativa, permite voltar ao início do programa.

A **porta NOR** permite que seja passado para a entrada do multiplexer do program counter (MUX_PC) ou '1' ou '0' dependendo do resultado da operação NOR dos 8 bits do Operando1.

O **multiplexer do program counter (MUX_PC)** tem o objetivo de indicar ao PC se é para realizar um salto ou incrementar o contador, através do sinal de saída ESCR_PC, de 1 bit, a '1' ou a '0', respetivamente. Este apresenta um sinal de seleção SEL_PC, de 3 bits, que indica qual dos valores de entrada deve passar para a saída.

A **ROM de descodificação (ROM)** é responsável por fornecer aos restantes blocos os sinais de controlo. Esta recebe o sinal opcode da memória de instruções, de 5 bits, e coloca na sua saída os valores correspondentes aos seguintes sinais de controlo: SEL_PC, de 3 bits, SEL_F, de 3 bits, SEL_ALU, de 4 bits, ESCR_R, de 2 bits, SEL_Dados, de 2 bits, ESCR_P, de 1 bit, WR, de 1 bit. Na Tabela seguinte encontra-se a relação entre o sinal opcode, e os sinais de controlo. É também indicada em assembly, a instrução correspondente. O registo Ri corresponde ao registo indicado pelos 3 bits menos significativos do sinal SEL_R, e o registo Rj corresponde aos 3 três mais significativos do sinal SEL_R. Nestas instruções pretende-se que o resultado fique guardado no Registo Ri (é necessário haver uma escrita no registo) e portanto o sinal SEL_R terá de tomar o valor apresentado na Tabela 2.

Tabela 7 – Relação entre o sinal *opcode* e os sinais de saída da ROM.

Opcode	Instrução	SEL_ALU	ESCR_P	SEL_Dados	ESCR_R	WR	SEL_PC	SEL_FLAG
Periféricos								
00000	LDP *Ri*	XXXX	0	01	01	0	000	XXX
00001	STP *Ri*	XXXX	1	XX	00	0	000	XXX
Leitura e Escrita								
00010	LD *Ri*, constante	XXXX	0	11	01	0	000	XXX
00011	LD *Ri*, [constante]	XXXX	0	10	01	0	000	XXX
00100	ST [constante], *Ri*	XXXX	0	XX	00	1	000	XXX
Lógica e Aritmética								
00101	ADD *Ri, Rj*	0000	0	00	01	0	000	XXX
00110	SUB *Ri, Rj*	0001	0	00	01	0	000	XXX
00111	AND Ri, Rj	0010	0	00	01	0	000	XXX
01000	NAND *Ri, Rj*	0011	0	00	01	0	000	XXX
01001	OR Ri, Rj	0100	0	00	01	0	000	XXX
01010	NOR Ri, Rj	0101	0	00	01	0	000	XXX
01011	XOR Ri, Rj	0110	0	00	01	0	000	XXX
01100	XNOR Ri, Rj	0111	0	00	01	0	000	XXX
01101	CMP *Ri, Rj*	1000	0	XX	10	0	000	XXX
Salto								
01110	JL constante	XXXX	0	XX	00	0	010	000
01111	JLE constante	XXXX	0	XX	00	0	010	001
10000	JE constante	XXXX	0	XX	00	0	010	010
10001	JGE constante	XXXX	0	XX	00	0	010	011
10010	JG constante	XXXX	0	XX	00	0	010	100
10011	JMP constante	XXXX	0	XX	00	0	001	XXX
10100	JZ *Ri*, constante	XXXX	0	XX	00	0	100	XXX
10101	JN *Ri*, constante	XXXX	0	XX	00	0	011	XXX
Outros								
Outros	NOP	XXXX	0	XX	00	0	000	XXX

Na **memória de instruções** ficam armazenadas as instruções do programa a ser executado. A sua dimensão é de 19 bits, sendo endereçada pelo sinal Endereco, de 8 bits, e disponibiliza à sua saída o opcode, de 5 bits, o sinal SEL_R, de 6 bits e o sinal Constante, de 8 bits.

Na **memória de dados** permite guardar os dados presentes no sinal de entrada Operando1, de 8 bits, quando o sinal WR estiver a '1', na transição ascendente do sinal de relógio (clk), na posição de memória indicada pelo sinal de entrada Constante, de 8 bits. Quando o sinal WR está a '0' é realizada uma leitura à posição de memória indicada pelo sinal Constante e esse valor é atribuído ao sinal de saída Dados_M, de 8 bits.

Depois de feitos os módulos acima descritos, criamos a struct do **processador,** que consiste em definir em componentes todos os módulos que o constituem, e fazer os seus port map. Ao finalizar a componente do processador bastou criar uma outra struct mas desta vez da **placa-mãe** muito similar ao que foi feito no processador com o módulo do processador,da memória de instruções e da memória de dados.

5.Discussão de resultados

Ao analisar as instruções dadas no enunciado do projeto percebemos que os resultados esperados são os seguintes:

Se PIN >= 20, o POUT esperado será o valor do PIN menos 15.
Se PIN = 0, o POUT esperado é -1.
Se PIN < -20, o POUT esperado é 1 ou 0, sendo 1 se o valor decimal do PIN for ímpar e, 0 se o mesmo for par.
Se -20 =<PIN <20 e PIN != 0, o POUT esperado é PIN * 8.

Nota-se que no caso dos valores 16,17,18,19, 20, -20,-19,-18,-17 do PIN ocorre overflow, visto que este programa usa valores de 8 bits, sendo o mais significativo o de sinal, assim sendo, permite valores entre -128 e 127 e, nestes casos multiplicando o valor do PIN por 8 ultrapassa-se esse intervalo de valores. É possível perceber que o overflow ocorre na prática, pois um número multiplicado por 8 teria de manter o mesmo sinal mas, nestes casos, isso não ocorre.

6.Conclusão

Neste projeto prático de Arquitetura de Computadores, foi possível adquirir conhecimentos fundamentais sobre como construir um processador de 8 bits(PEPE-8), utilizando a linguagem de descrição de hardware VHDL. O desenvolvimento do processador envolveu a criação e interligação de vários módulos.

A simulação no “ISim” permitiu verificar o funcionamento do processador, garantindo o bom funcionamento do processador e seus componentes. Foram criados outros dois módulos, a memória de dados e a memória de instruções, para criar a placa-mãe.

A fase de implementação em FPGA acrescentou uma parte prática ao projeto, possibilitando a execução do processador numa FPGA. Sendo que no nosso caso, escolhemos a SPARTAN3E.

Ofereceu também uma experiência valiosa na transição do ambiente de simulação para a implementação real.

O teste do processador demonstrou a capacidade do mesmo, em realizar operações de maneira eficaz e prática.

Podemos concluir, que o projeto proporcionou uma compreensão mais profunda da arquitetura de um processador e, o seu funcionamento interno. Este trabalho prático não apenas reforçou os conceitos teóricos abordados na disciplina, mas também permitiu a aplicação prática desses conhecimentos, proporcionando uma experiência valiosa no desenvolvimento de processadores.

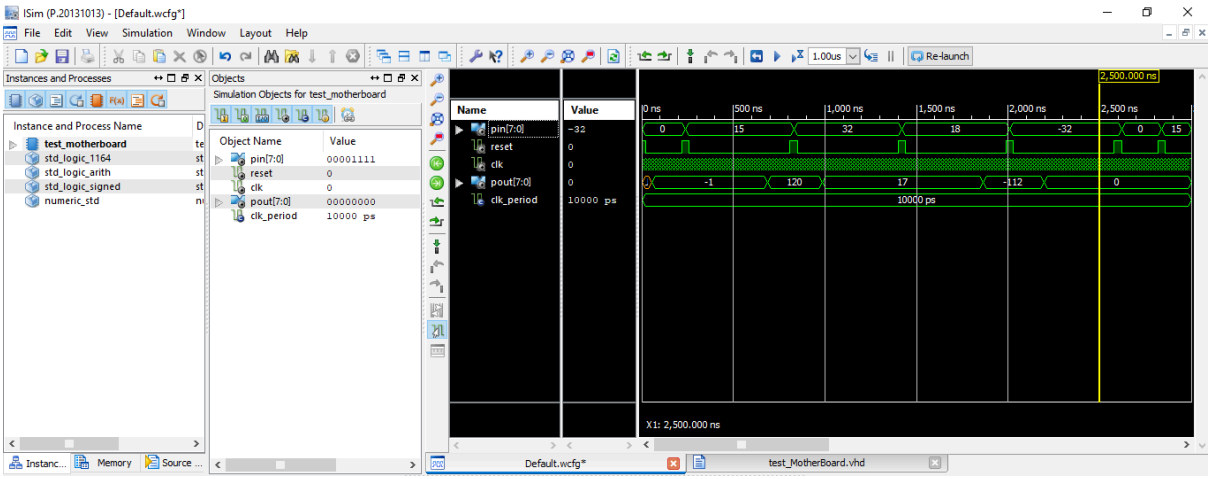
7.Bibliografia

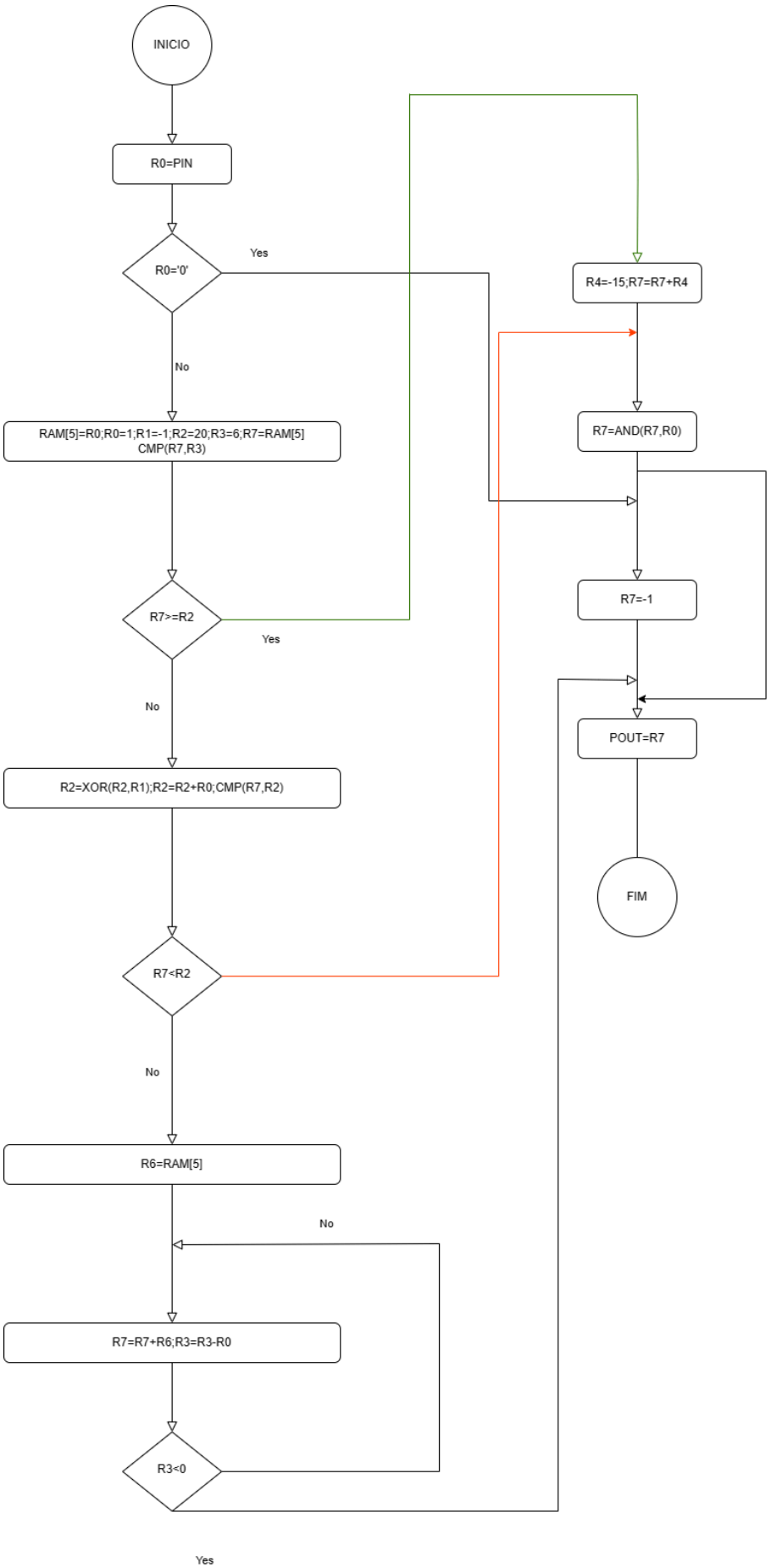
Guia Componente de Avaliação P1 de Arquitetura de Computadores
J. Delgado e C. Ribeiro, Arquitectura de Computadores, FCA, 2007
D. M. Harris and S. L. Harris, Digital Design and Computer Architecture, Elsevier MK, 2007

Anexo A

Endereço	Instrução (assembly)	Instrução (código máquina)		
		OPCODE	SEL_R	Constante
00000000	LDP R0	00000	XXX000	XXXXXXXX
00000001	JZ R0, 24	10100	XXX000	00011000
00000010	ST [5], R0	00100	XXX000	00000101
00000011	LD R0, 1	00010	XXX000	00000001
00000100	LD R1, -1	00010	XXX001	11111111
00000101	LD R2, 20	00010	XXX010	00010100
00000110	LD R3, 6	00010	XXX011	00000110
00000111	LD R7, [5]	00011	XXX111	00000101
00001000	CMP R7, R2	01101	010111	XXXXXXXX
00001001	JGE 19	10001	XXXXXX	00010011
00001010	XOR R2, R1	01011	001010	XXXXXXXX
00001011	ADD R2, R0	00101	000010	XXXXXXXX
00001100	CMP R7, R2	01101	010111	XXXXXXXX
00001101	JL 22	01110	XXXXXX	00010110
00001110	LD R6, [5]	00011	XXX110	00000101
00001111	ADD R7, R6	00101	110111	XXXXXXXX
00010000	SUB R3, R0	00110	000111	XXXXXXXX
00010001	JN R3, 25	10101	XXX011	00011001
00010010	JMP 15	10011	XXXXXX	00001111
00010011	LD R4, -15	00010	XXX100	11110001
00010100	ADD R7, R4	00101	100111	XXXXXXXX

00010101	JMP 25	10011	XXXXXX	00011001
00010110	AND R7, R0	00111	000111	XXXXXXXX
00010111	JMP 25	10011	XXXXXX	00011001
00011000	LD R7, -1	00010	XXX111	11111111
00011001	STP R7	00001	XXX111	XXXXXXXX
00011010	JMP 26	10011	XXXXXX	00011010
Others	NOP	XXXXX	XXXXXX	XXXXXXXX




INICIO
R0=PIN
R0='0'
Yes
No
RAM[5]=R0;R0=1;R1=-1;R2=20;R3=6;R7=RAM[5]
CMP(R7,R3)
R7>=R2
Yes
No
R2=XOR(R2,R1);R2=R2+R0;CMP(R7,R2)
R7<R2
No
R6=RAM[5]
No
R7=R7+R6;R3=R3-R0
R3<0
Yes
R4=-15;R7=R7+R4
R7=AND(R7,R0)
R7=-1
POUT=R7
FIM

Anexo B

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date:           13:57:40 03/05/2024
-- Design Name:
-- Module Name:        MotherBoard - Behavioral
-- Project Name:
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values


-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity MotherBoard is
        Port ( PIN : in  STD_LOGIC_VECTOR (7 downto 0);
        POUT : out  STD_LOGIC_VECTOR (7 downto 0);
        reset : in  STD_LOGIC;
        clk : in  STD_LOGIC);
```

```
end MotherBoard;

architecture Struct of MotherBoard is

Component Processador is
        Port ( PIN : in  STD_LOGIC_VECTOR (7 downto 0);
        POUT : out  STD_LOGIC_VECTOR (7 downto 0);
        clk : in  STD_LOGIC;
        reset : in  STD_LOGIC;
        Endereco : out  STD_LOGIC_VECTOR (7 downto 0);
        opcode : in  STD_LOGIC_VECTOR (4 downto 0);
        WR : out  STD_LOGIC;
        SEL_R : in  STD_LOGIC_VECTOR (5 downto 0);
        Constante : in  STD_LOGIC_VECTOR (7 downto 0);
                ConstanteCPU : out  STD_LOGIC_VECTOR (7 downto 0);
        Dados_M : in  STD_LOGIC_VECTOR (7 downto 0);
        Operando1 : out  STD_LOGIC_VECTOR (7 downto 0));
end Component;

Component Mem_Dados is
        Port ( clk : in  STD_LOGIC;
        ConstanteCPU : in  STD_LOGIC_VECTOR (7 downto 0);
        WR : in  STD_LOGIC;
        Dados_M : out  STD_LOGIC_VECTOR (7 downto 0);
        Operando1 : in  STD_LOGIC_VECTOR (7 downto 0));
end Component;

Component Memoria_Instr is
        Port ( Endereco : in  STD_LOGIC_VECTOR (7 downto 0);
        opcode : out  STD_LOGIC_VECTOR (4 downto 0);
        Constante : out  STD_LOGIC_VECTOR (7 downto 0);
        SEL_R : out  STD_LOGIC_VECTOR (5 downto 0));
end Component;

signal WR : STD_LOGIC;
signal SEL_R: STD_LOGIC_VECTOR (5 downto 0);
signal Endereco,Constante,Dados_M,Operando1: STD_LOGIC_VECTOR (7 downto 0);
signal opcode : STD_LOGIC_VECTOR(4 downto 0);
signal ConstanteCPU: STD_LOGIC_VECTOR (7 downto 0);

begin
   CPU : Processador port
map(PIN,POUT,clk,reset,Endereco,opcode,WR,SEL_R,Constante,ConstanteCPU,Dados_M,
Operando1);
   Memoria_Dados : Mem_Dados port map(clk,ConstanteCPU,WR,Dados_M,Operando1);
   Memoria_Instrucoes : Memoria_Instr port map(Endereco,opcode,Constante,SEL_R);

end Struct;
```

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date:           11:48:45 03/05/2024
-- Design Name:
-- Module Name:        Processador - Struct
-- Project Name:
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity Processador is
        Port ( PIN : in  STD_LOGIC_VECTOR (7 downto 0);
        POUT : out  STD_LOGIC_VECTOR (7 downto 0);
        clk : in  STD_LOGIC;
        reset : in  STD_LOGIC;
        Endereco : out  STD_LOGIC_VECTOR (7 downto 0);
        opcode : in  STD_LOGIC_VECTOR (4 downto 0);
        WR : out  STD_LOGIC;
        SEL_R : in  STD_LOGIC_VECTOR (5 downto 0);
        Constante : in  STD_LOGIC_VECTOR (7 downto 0);
                ConstanteCPU : out  STD_LOGIC_VECTOR (7 downto 0);
        Dados_M : in  STD_LOGIC_VECTOR (7 downto 0);
        Operando1 : out  STD_LOGIC_VECTOR (7 downto 0));
end Processador;
```

```
architecture Struct of Processador is

    Component Peri_Entrada is
          Port ( PIN : in  STD_LOGIC_VECTOR (7 downto 0);
          ESCR_P : in  STD_LOGIC;
          Dados_IN : out  STD_LOGIC_VECTOR (7 downto 0));
    end Component;

    Component Peri_Saida is
          Port ( clk : in  STD_LOGIC;
          ESCR_P : in  STD_LOGIC;
          Operando1 : in  STD_LOGIC_VECTOR (7 downto 0);
          POUT : out  STD_LOGIC_VECTOR (7 downto 0));
    end Component;

    Component ALU is
          Port ( Operando1 : in  STD_LOGIC_VECTOR (7 downto 0);
          Operando2 : in  STD_LOGIC_VECTOR (7 downto 0);
          SEL_ALU : in  STD_LOGIC_VECTOR (3 downto 0);
          Resultado : out  STD_LOGIC_VECTOR (7 downto 0);
                    E_FLAG : out STD_LOGIC_VECTOR (4 downto 0));
    end Component;

    Component Reg_Flags is
          Port ( clk : in  STD_LOGIC;
          E_FLAG : in  STD_LOGIC_VECTOR (4 downto 0);
          ESCR_R : in  STD_LOGIC_VECTOR (1 downto 0);
          SEL_FLAG : in  STD_LOGIC_VECTOR (2 downto 0);
          S_FLAG  : out  STD_LOGIC);
    end Component;

    Component Porta_NOR is
    Port ( Operando1 : in  STD_LOGIC_VECTOR (7 downto 0);
          saida_nor : out  STD_LOGIC);
    end Component;

    Component Mux_PC is
          Port ( S_FLAG : in  STD_LOGIC;
          Operando1 : in  STD_LOGIC_VECTOR (7 downto 0);
          SEL_PC : in  STD_LOGIC_VECTOR (2 downto 0);
          ESCR_PC : out  STD_LOGIC;
                    zero : in  STD_LOGIC;
          um : in  STD_LOGIC;
                    saida_nor: in STD_LOGIC);
    end Component;

    Component Banco_Reg is
          Port ( ESCR_R : in  STD_LOGIC_VECTOR (1 downto 0);
```

```
        clk : in  STD_LOGIC;
        SEL_R : in  STD_LOGIC_VECTOR (5 downto 0);
        Dados_R : in  STD_LOGIC_VECTOR (7 downto 0);
        Operando1 : out  STD_LOGIC_VECTOR (7 downto 0);
        Operando2 : out  STD_LOGIC_VECTOR (7 downto 0));
    end Component;

    Component MUX_R is
        Port ( Constante : in  STD_LOGIC_VECTOR (7 downto 0);
        Dados_M : in  STD_LOGIC_VECTOR (7 downto 0);
        Dados_IN : in  STD_LOGIC_VECTOR (7 downto 0);
        Resultado : in  STD_LOGIC_VECTOR (7 downto 0);
        Dados_R : out  STD_LOGIC_VECTOR (7 downto 0);
        SEL_Dados : in  STD_LOGIC_VECTOR (1 downto 0));
    end Component;

    Component PC is
        Port ( clk : in  STD_LOGIC;
        reset : in  STD_LOGIC;
        ESCR_PC : in  STD_LOGIC;
        Constante : in  STD_LOGIC_VECTOR (7 downto 0);
        Endereco : out  STD_LOGIC_VECTOR (7 downto 0));
    end Component;

    Component ROM_Descod is
    Port ( opcode : in  STD_LOGIC_VECTOR (4 downto 0);
        WR : out  STD_LOGIC;
        ESCR_P : out  STD_LOGIC;
        SEL_Dados : out  STD_LOGIC_VECTOR (1 downto 0);
        ESCR_R : out  STD_LOGIC_VECTOR (1 downto 0);
        SEL_ALU : out  STD_LOGIC_VECTOR (3 downto 0);
        SEL_FLAG : out  STD_LOGIC_VECTOR (2 downto 0);
        SEL_PC : out  STD_LOGIC_VECTOR (2 downto 0));
    end Component;

    signal ESCR_P,S_FLAG,ESCR_PC,saida_nor : STD_LOGIC;
    signal Dados_IN,Resultado : STD_LOGIC_VECTOR (7 downto 0);
    signal SOperando1,Operando2,Dados_R : STD_LOGIC_VECTOR (7 downto 0);
    signal SEL_ALU : STD_LOGIC_VECTOR (3 downto 0);
    signal ESCR_R,SEL_Dados : STD_LOGIC_VECTOR (1 downto 0);
    signal E_FLAG : STD_LOGIC_VECTOR (4 downto 0);
    signal SEL_FLAG,SEL_PC: STD_LOGIC_VECTOR (2 downto 0);

begin
    ConstanteCPU <= Constante;
    PerifericoEntrada: Peri_Entrada port map (PIN, ESCR_P, Dados_IN);
    PerifericoSaida: Peri_Saida port map (clk, ESCR_P, SOperando1,POUT);
    ALU_op: ALU port map (SOperando1, Operando2, SEL_ALU,Resultado,E_FLAG);
```

```
    Registo_Flags: Reg_Flags port map (clk, E_FLAG, ESCR_R,SEL_FLAG,S_FLAG);
    PortaNor: Porta_NOR port map (SOperando1,saida_nor);
    MUXPC: Mux_PC port map (S_FLAG, SOperando1,
SEL_PC,ESCR_PC,'0','1',saida_nor);
    Banco_Registos: Banco_Reg port map (ESCR_R, clk,
SEL_R,Dados_R,SOperando1,Operando2);
    MUXR: MUX_R port map (Constante, Dados_M,
Dados_IN,Resultado,Dados_R,SEL_Dados);
    ProgCounter: PC port map (clk, reset, ESCR_PC,Constante,Endereco);
    ROMDescodificacao: ROM_Descod port map (opcode, WR,
ESCR_P,SEL_Dados,ESCR_R,SEL_ALU,SEL_FLAG,SEL_PC);

    Operando1 <= SOperando1;


end Struct;


----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date:           21:37:58 02/25/2024
-- Design Name:
-- Module Name:        Peri_Entrada - Behavioral
-- Project Name:
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;
```

```
entity Peri_Entrada is
        Port ( PIN : in  STD_LOGIC_VECTOR (7 downto 0);
        ESCR_P : in  STD_LOGIC;
        Dados_IN : out  STD_LOGIC_VECTOR (7 downto 0));
end Peri_Entrada;

architecture Behavioral of Peri_Entrada is

begin
   process(PIN,ESCR_P)
         begin
                if ESCR_P = '0' then
                        Dados_IN <= PIN;
                end if;
   end process;
end Behavioral;
```

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date:           21:43:23 02/25/2024
-- Design Name:
-- Module Name:        Peri_Saida - Behavioral
-- Project Name:
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
```

```
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity Peri_Saida is
	Port ( clk : in  STD_LOGIC;
	ESCR_P : in  STD_LOGIC;
	Operando1 : in  STD_LOGIC_VECTOR (7 downto 0);
	POUT : out  STD_LOGIC_VECTOR (7 downto 0));
end Peri_Saida;

architecture Behavioral of Peri_Saida is

begin
   process(clk)
	 begin
		if rising_edge(clk) and ESCR_P = '1' then
			POUT <= Operando1;
		end if;

   end process;


end Behavioral;


----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date:           21:52:55 02/25/2024
-- Design Name:
-- Module Name:        ALU - Behavioral
-- Project Name:
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_signed.all;
```

```
-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity ALU is
	Port ( Operando1 : in  STD_LOGIC_VECTOR (7 downto 0);
	Operando2 : in  STD_LOGIC_VECTOR (7 downto 0);
	SEL_ALU : in  STD_LOGIC_VECTOR (3 downto 0);
	Resultado : out  STD_LOGIC_VECTOR (7 downto 0);
		E_FLAG : out STD_LOGIC_VECTOR (4 downto 0));


end ALU;

architecture Behavioral of ALU is
begin
   process(Operando1, Operando2, SEL_ALU)

   begin
	 case SEL_ALU is
		when "0000" => Resultado <= Operando1 +
Operando2;E_FLAG<="XXXXX";
		when "0001" => Resultado <= Operando1 - Operando2;E_FLAG<="XXXXX";
		when "0010" => Resultado <= Operando1 and
Operando2;E_FLAG<="XXXXX";
		when "0011" => Resultado <= Operando1 nand
Operando2;E_FLAG<="XXXXX";
		when "0100" => Resultado <= Operando1 or
Operando2;E_FLAG<="XXXXX";
		when "0101" => Resultado <= Operando1 nor
Operando2;E_FLAG<="XXXXX";
		when "0110" => Resultado <= Operando1 xor
Operando2;E_FLAG<="XXXXX";
		when "0111" => Resultado <= Operando1 xnor
Operando2;E_FLAG<="XXXXX";
		when "1000" =>
				if Operando1 = Operando2 then
					E_FLAG <= (1 => '1', 2 => '1', 3 => '1', others => '0');
				end if;
				if Operando1 < Operando2 then
					E_FLAG <= (0 => '1', 1 => '1', others => '0');
				end if;
				if Operando1 > Operando2 then
					E_FLAG <= (4 => '1', 3 => '1', others => '0');
				end if;
```

```
                when others => Resultado <= (others => 'X'); E_FLAG <= (others =>
'X');--estado alta impedancia
            end case;

    end process;


end Behavioral;
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date:           21:54:40 02/25/2024
-- Design Name:
-- Module Name:        Reg_Flags - Behavioral
-- Project Name:
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.std_logic_signed.all;
-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity Reg_Flags is
            Port ( clk : in  STD_LOGIC;
            E_FLAG : in  STD_LOGIC_VECTOR (4 downto 0);
            ESCR_R : in  STD_LOGIC_VECTOR (1 downto 0);
            SEL_FLAG : in  STD_LOGIC_VECTOR (2 downto 0);
            S_FLAG  : out  STD_LOGIC);
end Reg_Flags;

architecture Behavioral of Reg_Flags is
```

```
begin
   process(clk, ESCR_R, E_FLAG, SEL_FLAG)
   variable mem : STD_LOGIC_VECTOR (4 downto 0);
         begin
               case SEL_FLAG is
                     when "000" => S_FLAG <= mem(0);
                     when "001" => S_FLAG <= mem(1);
                     when "010" => S_FLAG <= mem(2);
                     when "011" => S_FLAG <= mem(3);
                     when "100" => S_FLAG <= mem(4);
                     when others => S_FLAG <= 'X';
               end case;
               if ESCR_R(1) = '1' and rising_edge(clk) then
                     mem := E_FLAG;
               end if;
   end process;


end Behavioral;

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity Porta_NOR is
         Port ( Operando1 : in  STD_LOGIC_VECTOR (7 downto 0);
         saida_nor : out  STD_LOGIC);
end Porta_NOR;

architecture Behavorial of Porta_NOR is
   begin
                     saida_nor <= not(Operando1(7) or Operando1(6) or Operando1(5) or
Operando1(4) or Operando1(3) or Operando1(2) or Operando1(1) or Operando1(0));

end Behavorial;


----------------------------------------------------------------------------------
-- Company:
-- Engineer:
```

```
--
-- Create Date:           22:05:39 02/25/2024
-- Design Name:
-- Module Name:        Mux_PC - Behavioral
-- Project Name:
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity Mux_PC is
          Port ( S_FLAG : in  STD_LOGIC;
          Operando1 : in  STD_LOGIC_VECTOR (7 downto 0);
          SEL_PC : in  STD_LOGIC_VECTOR (2 downto 0);
          ESCR_PC : out  STD_LOGIC;
                    zero : in  STD_LOGIC;
          um : in  STD_LOGIC;
                    saida_nor: in STD_LOGIC);
end Mux_PC;

architecture Behavioral of Mux_PC is

begin
   process(SEL_PC,S_FLAG,Operando1)
           begin
                    case SEL_PC is
                              when "000" => ESCR_PC <= zero;
                              when "001" => ESCR_PC <= um;
                              when "010" => ESCR_PC <= S_FLAG;
                              when "011" => ESCR_PC <= Operando1(7);
```

```
                        when "100" => ESCR_PC <= saida_nor;
                        when others => ESCR_PC <= 'X';
                    end case;
    end process;
end Behavioral;


----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date:           21:51:34 02/25/2024
-- Design Name:
-- Module Name:        Banco_Reg - Behavioral
-- Project Name:
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity Banco_Reg is
        Port ( ESCR_R : in  STD_LOGIC_VECTOR (1 downto 0);
        clk : in  STD_LOGIC;
        SEL_R : in  STD_LOGIC_VECTOR (5 downto 0);
        Dados_R : in  STD_LOGIC_VECTOR (7 downto 0);
        Operando1 : out  STD_LOGIC_VECTOR (7 downto 0);
        Operando2 : out  STD_LOGIC_VECTOR (7 downto 0));
end Banco_Reg;
```

```
architecture Behavioral of Banco_Reg is

    signal R0 : STD_LOGIC_VECTOR (7 downto 0);
    signal R1 : std_logic_vector (7 downto 0);
    signal R2 : std_logic_vector (7 downto 0);
    signal R3 : std_logic_vector (7 downto 0);
    signal R4 : std_logic_vector (7 downto 0);
    signal R5 : std_logic_vector (7 downto 0);
    signal R6 : std_logic_vector (7 downto 0);
    signal R7 : std_logic_vector (7 downto 0);

begin
    process(clk)
    begin
    case SEL_R(2 downto 0) is
            when "000" => Operando1 <= R0;
            when "001" => Operando1 <= R1;
            when "010" => Operando1 <= R2;
            when "011" => Operando1 <= R3;
            when "100" => Operando1 <= R4;
            when "101" => Operando1 <= R5;
            when "110" => Operando1 <= R6;
            when "111" => Operando1 <= R7;
            when others => null;
    end case;
            case SEL_R(5 downto 3) is
            when "000" => Operando2 <= R0;
            when "001" => Operando2 <= R1;
            when "010" => Operando2 <= R2;
            when "011" => Operando2 <= R3;
            when "100" => Operando2 <= R4;
            when "101" => Operando2 <= R5;
            when "110" => Operando2 <= R6;
            when "111" => Operando2 <= R7;
            when others => null;
    end case;
    if rising_edge(clk) then
                                if ESCR_R(0)='1' then
                                        case SEL_R(2 downto 0) is
                                                when "000" => R0 <= Dados_R;
                                                when "001" => R1 <= Dados_R;
                                                when "010" => R2 <= Dados_R;
                                                when "011" => R3 <= Dados_R;
                                                when "100" => R4 <= Dados_R;
                                                when "101" => R5 <= Dados_R;
                                                when "110" => R6 <= Dados_R;
                                                when "111" => R7 <= Dados_R;
                                                when others => null;
```

```
                                    end case;
                                end if;
    end if;

end process;

end Behavioral;

----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date:           21:46:17 02/25/2024
-- Design Name:
-- Module Name:        MUX_R - Behavioral
-- Project Name:
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity MUX_R is
        Port ( Constante : in  STD_LOGIC_VECTOR (7 downto 0);
        Dados_M : in  STD_LOGIC_VECTOR (7 downto 0);
        Dados_IN : in  STD_LOGIC_VECTOR (7 downto 0);
        Resultado : in  STD_LOGIC_VECTOR (7 downto 0);
        Dados_R : out  STD_LOGIC_VECTOR (7 downto 0);
        SEL_Dados : in  STD_LOGIC_VECTOR (1 downto 0));
end MUX_R;
```

```
architecture Behavioral of MUX_R is

begin
   process(Dados_IN, Dados_M, Constante, SEL_Dados, Resultado)
         begin
                  case SEL_Dados is
                           when "00" => Dados_R <= Resultado;
                           when "01" => Dados_R <= Dados_IN;
                           when "10" => Dados_R <= Dados_M;
                           when "11" => Dados_R <= Constante;
                           when others => Dados_R <= (others => 'X');
                  end case;
   end process;
end Behavioral;



----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date:          22:00:09 02/25/2024
-- Design Name:
-- Module Name:        PC - Behavioral
-- Project Name:
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_signed.ALL;
-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;
```

```
entity PC is
	Port ( clk : in  STD_LOGIC;
	reset : in  STD_LOGIC;
	ESCR_PC : in  STD_LOGIC;
	Constante : in  STD_LOGIC_VECTOR (7 downto 0);
	Endereco : out  STD_LOGIC_VECTOR (7 downto 0));
end PC;

architecture Behavioral of PC is
signal contador : STD_LOGIC_VECTOR(7 downto 0) := (others => '0');
begin

  process(clk)
  begin
	if rising_edge(clk) then
		if reset = '1' then
			contador <= (others => '0');
		else
			if ESCR_PC = '0' then
				contador <= contador + "00000001";
			else
				contador <= Constante;
			end if;
		end if;
	 end if;

    end process;
Endereco <= contador;
end Behavioral;



----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date:           22:16:57 02/25/2024
-- Design Name:
-- Module Name:        ROM_Descod - Behavioral
-- Project Name:
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
```

```
-- Additional Comments:
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity ROM_Descod is
    Port ( opcode : in  STD_LOGIC_VECTOR (4 downto 0);
           WR : out  STD_LOGIC;
           ESCR_P : out  STD_LOGIC;
           SEL_Dados : out  STD_LOGIC_VECTOR (1 downto 0);
           ESCR_R : out  STD_LOGIC_VECTOR (1 downto 0);
           SEL_ALU : out  STD_LOGIC_VECTOR (3 downto 0);
           SEL_FLAG : out  STD_LOGIC_VECTOR (2 downto 0);
           SEL_PC : out  STD_LOGIC_VECTOR (2 downto 0));
end ROM_Descod;

architecture Behavioral of ROM_Descod is

begin
process(opcode)
    begin

            case opcode is
            --Periféricos
            -- LDP Ri
            -- STP Ri
                    when "00000" => SEL_ALU <= "XXXX";  ESCR_P <= '0';  SEL_Dados <=
"01";  ESCR_R <= "01";  WR <= '0';  SEL_PC <= "000";   SEL_FLAG <= "XXX";
                    when "00001" => SEL_ALU <= "XXXX";  ESCR_P <= '1';  SEL_Dados <=
"XX";  ESCR_R <= "00";  WR <= '0';  SEL_PC <= "000";   SEL_FLAG <= "XXX";

            --Leitura e Escrita
            -- LD Ri, constante
            -- LD Ri, [constante]
            -- ST [constante], Ri
                    when "00010" => SEL_ALU <= "XXXX";  ESCR_P <= '0';  SEL_Dados <=
"11";  ESCR_R <= "01";  WR <= '0';  SEL_PC <= "000";   SEL_FLAG <= "XXX";
```

```
                when "00011" => SEL_ALU <= "XXXX";  ESCR_P <= '0';  SEL_Dados <=
"10";  ESCR_R <= "01";  WR <= '0';  SEL_PC <= "000";   SEL_FLAG <= "XXX";
                when "00100" => SEL_ALU <= "XXXX";  ESCR_P <= '0';  SEL_Dados <=
"XX";  ESCR_R <= "00";  WR <= '1';  SEL_PC <= "000";   SEL_FLAG <= "XXX";

        --Lógica e Aritmética
        -- ADD Ri, Rj
        -- SUB Ri, Rj
        -- AND Ri, Rj
        -- NAND Ri, Rj
        -- OR Ri, Rj
        -- NOR Ri, Rj
        -- XOR Ri, Rj
        -- XNOR Ri, Rj
        -- CMP Ri, Rj
                when "00101" => SEL_ALU <= "0000";  ESCR_P <= '0';  SEL_Dados <=
"00";  ESCR_R <= "01";  WR <= '0';  SEL_PC <= "000";       SEL_FLAG <= "XXX";
                when "00110" => SEL_ALU <= "0001";  ESCR_P <= '0';  SEL_Dados <=
"00";  ESCR_R <= "01";  WR <= '0';  SEL_PC <= "000";       SEL_FLAG <= "XXX";
                when "00111" => SEL_ALU <= "0010";  ESCR_P <= '0';  SEL_Dados <=
"00";  ESCR_R <= "01";  WR <= '0';  SEL_PC <= "000";       SEL_FLAG <= "XXX";
                when "01000" => SEL_ALU <= "0011";  ESCR_P <= '0';  SEL_Dados <=
"00";  ESCR_R <= "01";  WR <= '0';  SEL_PC <= "000";    SEL_FLAG <= "XXX";
                when "01001" => SEL_ALU <= "0100";  ESCR_P <= '0';  SEL_Dados <=
"00";  ESCR_R <= "01";  WR <= '0';  SEL_PC <= "000";   SEL_FLAG <= "XXX";
                when "01010" => SEL_ALU <= "0101";  ESCR_P <= '0';  SEL_Dados <=
"00";  ESCR_R <= "01";  WR <= '0';  SEL_PC <= "000";   SEL_FLAG <= "XXX";
                when "01011" => SEL_ALU <= "0110";  ESCR_P <= '0';  SEL_Dados <=
"00";  ESCR_R <= "01";  WR <= '0';  SEL_PC <= "000";   SEL_FLAG <= "XXX";
                when "01100" => SEL_ALU <= "0111";  ESCR_P <= '0';  SEL_Dados <= "00";
ESCR_R <= "01";  WR <= '0';  SEL_PC <= "000";   SEL_FLAG <= "XXX";
                when "01101" => SEL_ALU <= "1000";  ESCR_P <= '0';  SEL_Dados <=
"XX";  ESCR_R <= "10";  WR <= '0';  SEL_PC <= "000";   SEL_FLAG <= "XXX";

        --Salto
        -- JL constante
        -- JLE constante
        -- JE constante
        -- JGE constante
        -- JG constante
        -- JMP constante
        -- JZ Ri, constante
        -- JN Ri, constante

                when "01110" => SEL_ALU <= "XXXX";  ESCR_P <= '0';  SEL_Dados <=
"XX";  ESCR_R <= "00";  WR <= '0';  SEL_PC <= "010";   SEL_FLAG <= "000";
                when "01111" => SEL_ALU <= "XXXX";  ESCR_P <= '0';  SEL_Dados <=
"XX";  ESCR_R <= "00";  WR <= '0';  SEL_PC <= "010";   SEL_FLAG <= "001";
```

```
                when "10000" => SEL_ALU <= "XXXX";  ESCR_P <= '0';  SEL_Dados <=
"XX";  ESCR_R <= "00";  WR <= '0';  SEL_PC <= "010";   SEL_FLAG <= "010";
                when "10001" => SEL_ALU <= "XXXX";  ESCR_P <= '0';  SEL_Dados <=
"XX";  ESCR_R <= "00";  WR <= '0';  SEL_PC <= "010";   SEL_FLAG <= "011";
                when "10010" => SEL_ALU <= "XXXX";  ESCR_P <= '0';  SEL_Dados <=
"XX";  ESCR_R <= "00";  WR <= '0';  SEL_PC <= "010";   SEL_FLAG <= "100";
                when "10011" => SEL_ALU <= "XXXX";  ESCR_P <= '0';  SEL_Dados <=
"XX";  ESCR_R <= "00";  WR <= '0';  SEL_PC <= "001";   SEL_FLAG <= "XXX";
                when "10100" => SEL_ALU <= "XXXX";  ESCR_P <= '0';  SEL_Dados <=
"XX";  ESCR_R <= "00";  WR <= '0';  SEL_PC <= "100";   SEL_FLAG <= "XXX";
                when "10101" => SEL_ALU <= "XXXX";  ESCR_P <= '0';  SEL_Dados <=
"XX";  ESCR_R <= "00";  WR <= '0';  SEL_PC <= "011";   SEL_FLAG <= "XXX";

        --Outros
        -- NOP
                when others  => SEL_ALU <= "XXXX";  ESCR_P <= '0';  SEL_Dados <=
"XX";  ESCR_R <= "00";  WR <= '0';  SEL_PC <= "000";   SEL_FLAG <= "XXX";

        end case;
    end process;


end Behavioral;

----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date:          22:22:30 02/25/2024
-- Design Name:
-- Module Name:        Mem_Dados - Behavioral
-- Project Name:
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
```

```
-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity Mem_Dados is
	Port ( clk : in  STD_LOGIC;
	ConstanteCPU : in  STD_LOGIC_VECTOR (7 downto 0);
	WR : in  STD_LOGIC;
	Dados_M : out  STD_LOGIC_VECTOR (7 downto 0);
	Operando1 : in  STD_LOGIC_VECTOR (7 downto 0));
end Mem_Dados;

architecture Behavioral of Mem_Dados is
type mem is array (0 to 255) of STD_LOGIC_VECTOR(7 downto 0);
signal memoria : mem;
begin
process(clk)

	begin
	if rising_edge(clk) and WR = '1' then
		memoria(to_integer(unsigned(ConstanteCPU))) <= Operando1;
	end if;
	if WR = '0' then
		Dados_M <= memoria(to_integer(unsigned(ConstanteCPU)));
	end if;

   end process;

end Behavioral;
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date:           22:15:16 02/25/2024
-- Design Name:
-- Module Name:        Memoria_Instr - Behavioral
-- Project Name:
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
```

```
-- Additional Comments:
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity Memoria_Instr is
        Port ( Endereco : in  STD_LOGIC_VECTOR (7 downto 0);
        opcode : out  STD_LOGIC_VECTOR (4 downto 0);
        Constante : out  STD_LOGIC_VECTOR (7 downto 0);
        SEL_R : out  STD_LOGIC_VECTOR (5 downto 0));
end Memoria_Instr;

architecture Behavioral of Memoria_Instr is

begin
   process(Endereco)
         begin
                case Endereco is
                        when "00000000" => Opcode <= "00000";SEl_R <=
"XXX000";Constante <= "XXXXXXXX"; --LDP R0
                        when "00000001" => Opcode <= "10100";SEl_R <=
"XXX000";Constante <= "00011000";--JZ R0; 24
                        when "00000010" => Opcode <= "00100";SEl_R <=
"XXX000";Constante <= "00000101";--ST [5]; R0
                        when "00000011" => Opcode <= "00010";SEl_R <=
"XXX000";Constante <= "00000001";--LD R0; 1
                        when "00000100" => Opcode <= "00010";SEl_R <=
"XXX001";Constante <= "11111111";--LD R1; -1
                        when "00000101" => Opcode <= "00010";SEl_R <=
"XXX010";Constante <= "00010100";--LD R2; 20
                        when "00000110" => Opcode <= "00010";SEl_R <=
"XXX011";Constante <= "00000110";--LD R3; 6
                        when "00000111" => Opcode <= "00011";SEl_R <=
"XXX111";Constante <= "00000101";--LD R7; [5]
                        when "00001000" => Opcode <= "01101";SEl_R <=
"010111";Constante <= "XXXXXXXX";--CMP R7; R2
                        when "00001001" => Opcode <= "10001";SEl_R <=
"XXXXXX";Constante <= "00010011";--JGE 19
```

```
                        when "00001010" => Opcode <= "01011";SEl_R <=
"001010";Constante <= "XXXXXXXX";--XOR R2; R1
                        when "00001011" => Opcode <= "00101";SEl_R <=
"000010";Constante <= "XXXXXXXX";--ADD R2; R0
                        when "00001100" => Opcode <= "01101";SEl_R <=
"010111";Constante <= "XXXXXXXX";--CMP R7; R2
                        when "00001101" => Opcode <= "01110";SEl_R <=
"XXXXXX";Constante <= "00010110";--JL 22
                        when "00001110" => Opcode <= "00011";SEl_R <=
"XXX110";Constante <= "00000101";--LD R6; [5]
                        when "00001111" => Opcode <= "00101";SEl_R <=
"110111";Constante <= "XXXXXXXX";--ADD R7; R6
                        when "00010000" => Opcode <= "00110";SEl_R <=
"000011";Constante <= "XXXXXXXX";--SUB R3; R0
                        when "00010001" => Opcode <= "10101";SEl_R <=
"XXX011";Constante <= "00011001";--JN R3; 25
                        when "00010010" => Opcode <= "10011";SEl_R <=
"XXXXXX";Constante <= "00001111";--JMP 15
                        when "00010011" => Opcode <= "00010";SEl_R <=
"XXX100";Constante <= "11110001";--LD R4; -15
                        when "00010100" => Opcode <= "00101";SEl_R <=
"100111";Constante <= "XXXXXXXX";--ADD R7; R4
                        when "00010101" => Opcode <= "10011";SEl_R <=
"XXXXXX";Constante <= "00011001";--JMP 25
                        when "00010110" => Opcode <= "00111";SEl_R <=
"000111";Constante <= "XXXXXXXX";--AND R7; R0
                        when "00010111" => Opcode <= "10011";SEl_R <=
"XXXXXX";Constante <= "00011001";--JMP 25
                        when "00011000" => Opcode <= "00010";SEl_R <=
"XXX111";Constante <= "11111111";--LD R7; -1
                        when "00011001" => Opcode <= "00001";SEl_R <=
"XXX111";Constante <= "XXXXXXXX";--STP R7
                        when "00011010" => Opcode <= "10011";SEl_R <=
"XXXXXX";Constante <= "00011010";--JMP 26
                        when others => Opcode <= "XXXXX";SEl_R <= "XXXXXX";Constante
<= "XXXXXXXX";

            end case;

    end process;


end Behavioral;
```