Uma imagem com sentado, preto, computador

Descrição gerada automaticamente

**Faculdade de Ciências Exatas e da Engenharia**

**2019/2020**

**Arquitetura de Computadores**

Licenciatura em Engenharia Informática

**1º Projeto – Processador Básico**

Uma imagem com camisa

Descrição gerada automaticamente

**Trabalho realizado por:**

Diego Briceño (nº 2043818)

Rúben Rodrigues (nº 2046018)

Funchal, 14 de março de 2020

Índice

[1. Introdução 3](#_Toc35267396)

[2. Objetivos 3](#_Toc35267397)

[3. Desenvolvimento 3](#_Toc35267398)

[3.1. Placa-mãe 3](#_Toc35267399)

[3.1.1. Memória de Dados (RAM) 3](#_Toc35267400)

[3.1.2. Memória de Instruções 3](#_Toc35267401)

[3.1.3. Processador 4](#_Toc35267402)

[3.1.3.1. Periférico de Entrada 4](#_Toc35267403)

[3.1.3.2. Periférico de Saída 4](#_Toc35267404)

[3.1.3.3. Multiplexer dos Registos (Mux R) 4](#_Toc35267405)

[3.1.3.4. Registos A e B 4](#_Toc35267406)

[3.1.3.5. Unidade Aritmética e Lógica (ALU) 5](#_Toc35267407)

[3.1.3.6. Comparação 5](#_Toc35267408)

[3.1.3.7. Contador de programa (PC) 6](#_Toc35267409)

[3.1.3.8. *Multiplexer* do *Program Counter* (Mux\_PC) 6](#_Toc35267410)

[3.1.3.9. ROM de descodificação (ROM) 6](#_Toc35267411)

[4. Discussão de Resultados 7](#_Toc35267412)

[5. Conclusão 7](#_Toc35267413)

[6. Bibliografia 7](#_Toc35267414)

[7. Anexo A 8](#_Toc35267415)

[7.1. Tabela de Instruções de teste 8](#_Toc35267416)

[7.2. Fluxograma 9](#_Toc35267417)

[7.3. Simulação com o PIN < 0 10](#_Toc35267418)

[7.4. Simulação com o PIN >= 40 10](#_Toc35267419)

[7.5. Simulação com o 0<=PIN<40 11](#_Toc35267420)

[8. Anexo B 12](#_Toc35267421)

[8.1. Placa-Mãe 12](#_Toc35267422)

[8.1.1. Processador 12](#_Toc35267423)

[8.1.1.1. Periférico de Entrada 14](#_Toc35267424)

[8.1.1.2. Periférico de Saída 14](#_Toc35267425)

[8.1.1.3. Multiplexer dos Registos (Mux R) 14](#_Toc35267426)

[8.1.1.4. Registos A e B 15](#_Toc35267427)

[8.1.1.5. Unidade Aritmética e Lógica (ALU) 15](#_Toc35267428)

[8.1.1.6. Comparação 16](#_Toc35267429)

[8.1.1.7. Contador de programa (PC) 16](#_Toc35267430)

[8.1.1.8. Multiplexer do Program Counter (Mux\_PC) 17](#_Toc35267431)

[8.1.1.9. ROM de descodificação (ROM) 17](#_Toc35267432)

[8.1.2. Memória de Instruções 18](#_Toc35267433)

[8.1.3. Memória de Dados (RAM) 19](#_Toc35267434)

1. **Introdução**

Este relatório apresentará os objetivos relacionados ao primeiro trabalho prático da unidade curricular de Arquitetura de Computadores assim como o seu desenvolvimento, discussão de resultados e a conclusão a que os alunos chegaram no fim.

Os processadores são as unidades centrais dos sistemas computacionais. Se comparássemos um sistema computacional a uma pessoa, o processador seria o “cérebro” pois é ele que executa as instruções de máquina, que são qualquer tarefa que o processador possa executar utilizando uma série de cálculos e decisões.

1. **Objetivos**

O objetivo deste trabalho é realizar um processador básico, com um conjunto mínimo de instruções, em linguagem de descrição de hardware (VHDL). Para este fim, utilizou-se o programa ISE da Xilinx com a simulação sendo efetuada no ISim e o teste em FPGA (Spartan 3E e Artix 7).

1. **Desenvolvimento**

O processador desenvolvido é constituído por vários módulos que, quando conectados com a memória de dados e a memória de instruções, formam a placa mãe. Cada módulo foi implementado em separado, como será descrito de seguida, de modo a facilitar a implementação, sendo só necessário no fim ligar os diferentes módulos.

* 1. **Placa-mãe**
     1. **Memória de Dados (RAM)**

A RAM (Random-Access Memory) é um tipo de memória que permite a escrita/leitura de dados, individual e aleatória (como o nome indica), através do seu endereço. É uma memória volátil, ou seja, os seus dados perdem-se quando a memória perde a alimentação elétrica.

No caso desta placa-mãe, a memória de dados guarda os dados presentes no sinal de entrada ***Operando1,*** de 8 bits, quando o sinal ***WR*** está a ‘1’ na transição ascendente do sinal de relógio (***clk***), no endereço indicado pelo sinal de entrada ***Constante***, de 8 bits. Quando o sinal ***WR*** está a ‘0’ é feita a leitura dos dados, na posição de memória indicada por ***Constante*** e o valor lido é atribuído ao sinal de saída ***Dados\_M***, de 8 bits.

* + 1. **Memória de Instruções**

É neste módulo que ficam armazenas as instruções do programa a ser executado. Apresenta uma dimensão de 14 bits, onde o endereço da instrução é determinado pelo sinal ***Endereço***, de 8 bits, e à saída é disponibilizado o ***opcode***, de 5 bits, o sinal ***SEL\_R***, de 1 bit, e o sinal ***Constante***, de 8 bits.

* + 1. **Processador**
       1. **Periférico de Entrada**

É neste módulo que é feita a comunicação do processador com o exterior, permitindo ao utilizador inserir dados para posteriormente serem realizadas operações com os mesmos. Alguns exemplos deste tipo de periféricos são o teclado e o rato.

Este módulo é controlado pelo sinal , de 1 bit, que quando está a ‘1’ é feita uma leitura dos dados de entrada, ***PIN***, de 8 bits, colocando-os na saída do periférico, ***Dados\_IN***, de 8 bits.

* + - 1. **Periférico de Saída**

Este módulo permite que o utilizador veja os dados e informações processados pelo computador. Alguns exemplos deste tipo de periféricos são o monitor, a impressora e colunas de som. Este periférico é controlado pelo sinal ***ESCR\_P***, de 1 bit, 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***, também de 8 bits.

* + - 1. **Multiplexer dos Registos (Mux R)**

Este módulo é responsável por encaminhar um dos quatro sinais disponíveis, de 8 bits, à sua entrada (***Resultado, Dados\_IN, Dados\_M e Constante***) para apresentar na sua saída, ***Dados\_R***, de 8 bits. O sinal a encaminhar depende do valor do sinal de entrada ***SEL\_Data***, de 2 bits.

|  |  |
| --- | --- |
| **SEL\_DATA** | **DADOS\_R** |
| 00 | Resultado |
| 01 | Dados\_IN |
| 10 | Dados\_M |
| 11 | Constante |

Tabela 1 Sinal de saída do Mux R em função do sinal SEL\_DATA.

* + - 1. **Registos A e B**

A escrita nos registos A e B é controlada pelo sinal ***ESCR\_R***, de 1 bit. Quando o sinal está a ‘1’ o valor presente no sinal de entrada ***Dados\_R***, de 8 bits, é guardado no registo especificado pelo sinal ***SEL\_R***, de 1 bit, na transição ascendente do sinal de relógio (clk).

Estes registos estão continuamente a efetuas leituras. As saídas ***Operando1*** e ***Operando2***, ambas de 8 bits, apresentam os valores guardados nos registos A e B, respetivamente.

|  |  |
| --- | --- |
| **SEL\_R** | **Registo a ser escrito** |
| 0 | Registo A |
| 1 | Registo B |

Tabela 2 Sinal de seleção de escrita nos registos A e B

* + - 1. **Unidade Aritmética e Lógica (ALU)**

Este módulo permite realizar operações aritméticas e lógicas, tal como o nome indica. No caso desta placa-mãe, a unidade aritmética e lógica do processador, é capaz de realizar as operações soma, subtração, AND, OR e XOR, com os sinais de entrada ***Operando1*** e ***Operando2***, ambos de 8 bits, que representam números inteiros com sinal. Os sinais de saída da ALU são determinados pelo sinal de seleção ***SEL\_ALU***, de 3 bits, A saída ***Resultado***, de 8 bits, será atualizada no caso de cada operação e, a saída ***COMP\_RES***, de 5 bits, será atualizada apenas quando é realizada uma comparação, cada um dos seus bits indicando o resultado de uma das cinco comparações apresentadas no módulo *Comparação*.

|  |  |
| --- | --- |
| **SEL\_ALU** | **Operação** |
| 000 | Operando1 **+** Operando2 |
| 001 | Operando1 **–** Operando2 |
| 010 | Operando1 **AND** Operando2 |
| 011 | Operando1 **OR** Operando2 |
| 100 | Operando1 **XOR** Operando3 |
| 101 | Operando1 > Operando2  Operando1 >= Operando2  Operando1 = Operando2  Operando1 <= Operando2  Operando1 < Operando2 |

Tabela 3 Operações da ALU

* + - 1. **Comparação**

O funcionamento deste módulo é semelhante ao módulo dos registos. Guarda o sinal de entrada, que neste caso é o sinal ***COMP\_RES***, sinal de 5 bits, quando o sinal ***COMP\_FLAG*** está a ‘1’ e o sinal de relógio encontra-se na transição ascendente. Este módulo está constantemente a efetuar leituras, mas apenas um dos 5 bits do sinal guardado é encaminhado para a saída, ***S\_FLAG***, de 1 bit. O sinal de seleção ***SEL\_COMP***, de 3 bits, determina qual o bit guardado que está disponível na saída, do modo apresentado na tabela abaixo.

|  |  |
| --- | --- |
| **SEL\_COMP** | **S\_FLAG** |
| 000 | COMP\_RES(0) (>) |
| 001 | COMP\_RES(1) (>=) |
| 010 | COMP\_RES(2) (=) |
| 011 | COMP\_RES(3) (<=) |
| 100 | COMP\_RES(4) (<) |

Tabela 4 Sinal de saída do multiplexer de comparação em função do sinal de seleção SEL\_COMP

* + - 1. **Contador de programa (PC)**

O contador de programa indica qual é a posição atual da sequência de execução de um programa. Na transição ascendente do relógio, a saída ***Endereço***, de 8 bits, é enviada à Memória de Instruções. A sequência de execução será incrementada de um em um quando a entrada ***ESCR\_PC***, de 1 bit, estiver a ‘0’, caso contrário, a saída do contador corresponderá ao valor da entrada ***Constante***, de 8 bits, e neste caso ocorrerá um salto para o endereço de instrução indicado por este sinal. A entrada ***Reset***, de 1 bit, permite voltar ao início do programa quando ativa.

* + - 1. ***Multiplexer* do *Program Counter* (Mux\_PC)**

Este *multiplexer* indica ao contador de programa se é para realizar um salto ou simplesmente incrementar o contador, através do sinal de saída ***ESCR\_PC***, de 1 bit, como já foi visto no módulo do Contador de programa.

O sinal de seleção deste *multiplexer* é o sinal ***SEL\_PC***, de 3 bits, que indica qual dos valores de entrada deve passar para a saída, como indicado na tabela seguinte.

|  |  |
| --- | --- |
| **SEL\_PC** | **ESCR\_PC** |
| 000 | ‘0’ |
| 001 | ‘1’ |
| 010 | S\_FLAG |
| 011 | Operando1(7) |
| 100 | NOT (Operando1(7) OR Operando1(6) OR Operando1(5) OR Operando1(4) OR Operando1(3) OR Operando1(2) OR Operando1(1) OR Operando1(0)) |

Tabela 5 Valor de saída do MUX\_PC em função do sinal de seleção SEL\_PC

* + - 1. **ROM de descodificação (ROM)**

Esta ROM é responsável por fornecer aos restantes módulos os seus sinais de controlo. Esta recebe o sinal ***opcode***, de 5 bits, da memória de instruções e coloca na sua saída os valores correspondentes aos seguintes sinais de controlo: ***SEL\_PC***, ***SEL\_COMP*** e ***SEL\_ALU***, de 3 bits, ***SEL\_Data***, de 2 bits, e os sinais ***COMP\_FLAG***, ***ESCR\_R***, ***ESCR\_P*** e ***WR***, de 1 bit. Na tabela 6 presente no enunciado encontra-se a relação entre o sinal *opcode* e os sinais de controlo, onde cada instrução está também indicada em linguagem *assembly*. De considerar que o *Ri* corresponde ao registo indicado pelo sinal ***SEL\_R***, de 1 bit.

No código é utilizado um *case* para implementar a tabela referida anteriormente.

1. **Discussão de Resultados**

Para saber o procedimento que o processador efetuará quando são executadas as instruções mostradas do teste no enunciado, cuja tabela encontra-se no Anexo A. Foi preciso traduzir as instruções que se encontravam em linguagem *assembly* para código máquina de modo a programar a memória de instruções.

O teste referido está feito de maneira a que o processador siga as seguintes operações:

* Quando o PIN a ser introduzido representa um valor negativo, o programa realiza a operação:

**POUT = - PIN**

* Esta situação encontra-se representada na simulação presente no Anexo A, ponto 7.3.
* Quando o PIN a ser introduzido representa um valor maior ou igual a 40, o programa realiza a operação:

**POUT = PIN – 20**

* Esta situação encontra-se representada na simulação presente no Anexo A, ponto 7.4.
* Caso contrário, o PIN é um valor positivo menor que 40, o processador realiza a operação:

**POUT = 3 \* PIN**

* Esta situação encontra-se representada na simulação presente no Anexo A, ponto 7.5.

O programa encontra-se esquematizado através de um fluxograma no ponto 7.2 do Anexo A.

1. **Conclusão**

Para concluir, este projeto permitiu aos alunos ganhar uma melhor compreensão sobre o funcionamento interno do computador, em particular o processador. As simulações no ISim e os testes nas FPGA resultaram como era esperado, confirmando-se assim o sucesso durante a elaboração deste primeiro projeto.

1. **Bibliografia**

J. Delgado e C. Ribeiro, Arquitectura de Computadores, FCA - Editora de Informática, 2010.

1. **Anexo A**
   1. **Tabela de Instruções de teste**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Endereço** | **Instrução (Assembly)** | **Instrução (código máquina)** | | |
| **Opcode** | **SEL\_R** | **Constante** |
| **0** | 00000000 | LD RA, 3 | 00010 | 0 | 00000011 |
| **1** | 00000001 | ST [0], RA | 00100 | 0 | 00000000 |
| **2** | 00000010 | LD RA, 20 | 00010 | 0 | 00010100 |
| **3** | 00000011 | ST [1], RA | 00100 | 0 | 00000001 |
| **4** | 00000100 | LDP RA | 00000 | 0 | XXXXXXXX |
| **5** | 00000101 | JN 22 | 10001 | X | 00010110 |
| **6** | 00000110 | LD RB, 40 | 00010 | 1 | 00101000 |
| **7** | 00000111 | CMP RA, RB | 01010 | X | XXXXXXXX |
| **8** | 00001000 | JGE 27 | 01100 | X | 00011011 |
| **9** | 00001001 | ST [2], RA | 00100 | 0 | 00000010 |
| **10** | 00001010 | ST [3], RA | 00100 | 0 | 00000011 |
| **11** | 00001011 | LD RA, [0] | 00011 | 0 | 00000000 |
| **12** | 00001100 | LD RB, 1 | 00010 | 1 | 00000001 |
| **13** | 00001101 | SUB RA, RB | 00110 | 0 | XXXXXXXX |
| **14** | 00001110 | JZ 20 | 10010 | X | 00010100 |
| **15** | 00001111 | ST [0], RA | 00100 | 0 | 00000000 |
| **16** | 00010000 | LD RA, [3] | 00011 | 0 | 00000011 |
| **17** | 00010001 | LD RB, [2] | 00011 | 1 | 00000010 |
| **18** | 00010010 | ADD RA, RB | 00101 | 0 | XXXXXXXX |
| **19** | 00010011 | JMP 10 | 10000 | X | 00001010 |
| **20** | 00010100 | LD RA [3] | 00011 | 0 | 00000011 |
| **21** | 00010101 | JMP 29 | 10000 | X | 00011101 |
| **22** | 00010110 | LD RB, -1 | 00010 | 1 | 11111111 |
| **23** | 00010111 | XOR RA, RB | 01001 | 0 | XXXXXXXX |
| **24** | 00011000 | LD RB, 1 | 00010 | 1 | 00000001 |
| **25** | 00011001 | ADD RA, RB | 00101 | 0 | XXXXXXXX |
| **26** | 00011010 | JMP 29 | 10000 | X | 00011101 |
| **27** | 00011011 | LD RB, [1] | 00011 | 1 | 00000001 |
| **28** | 00011100 | SUB RA, RB | 00110 | 0 | XXXXXXXX |
| **29** | 00011101 | STP RA | 00001 | 0 | XXXXXXXX |
| **30** | 00011110 | JMP 30 | 10000 | X | 00011110 |

Tabela 6 Instruções de teste do projeto

* 1. **Fluxograma**

**Uma imagem com iPod

Descrição gerada automaticamente**

Figura Fluxograma do projeto

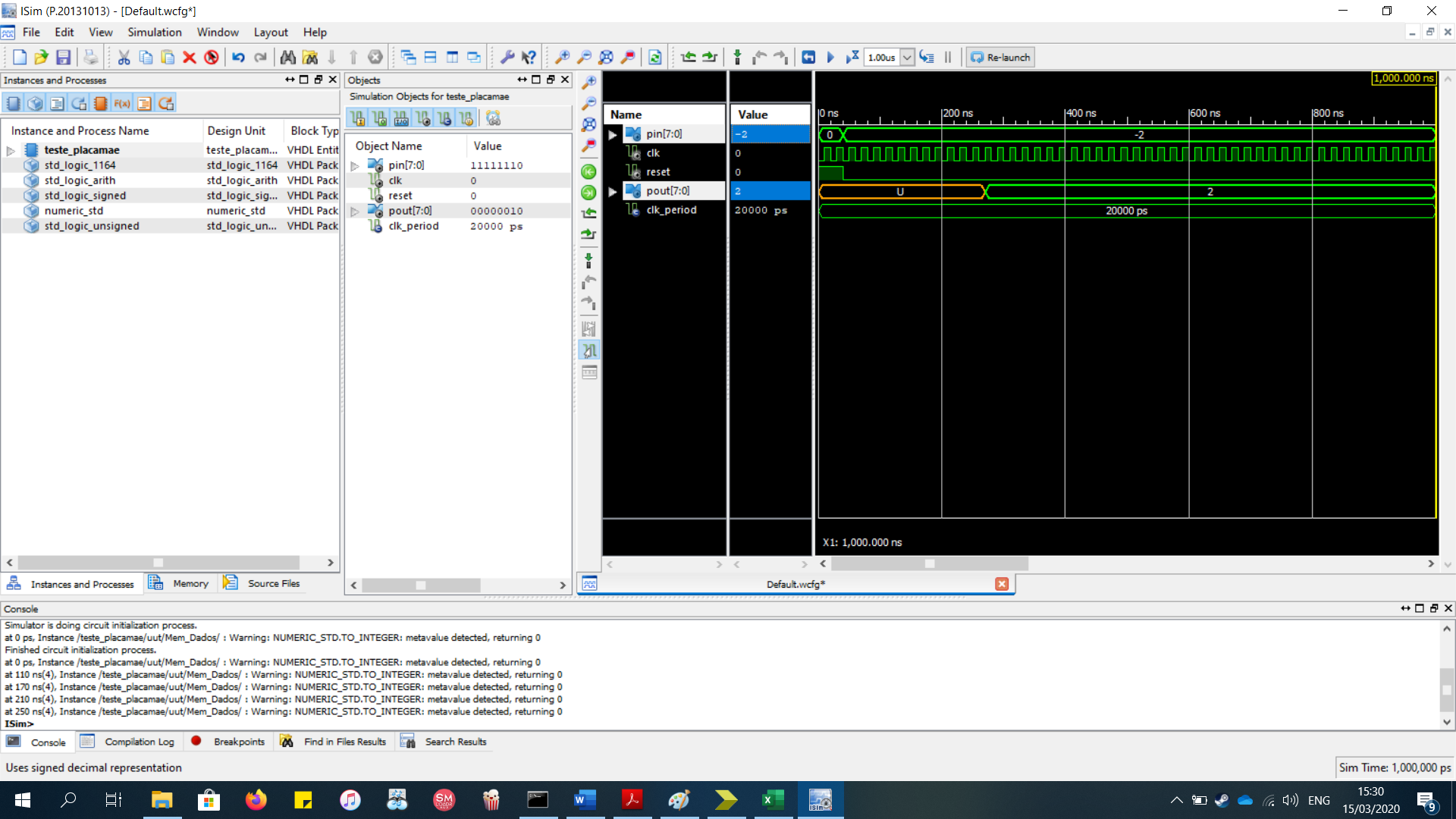
* 1. **Simulação com o PIN < 0**

Ilustração Simulação com o valor de PIN a -2

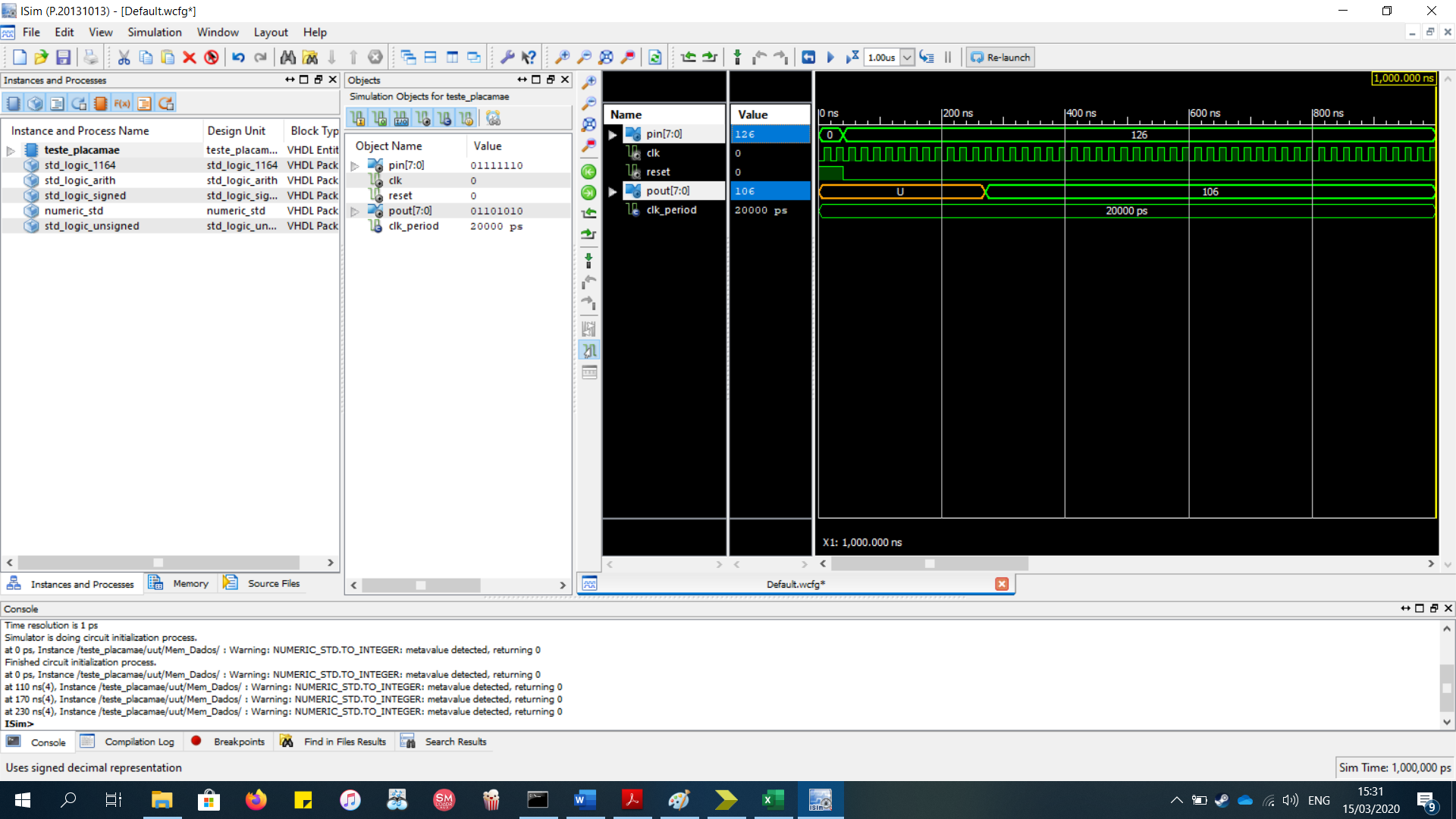
* 1. **Simulação com o PIN >= 40**

Ilustração Simulação com o valor de PIN a 126

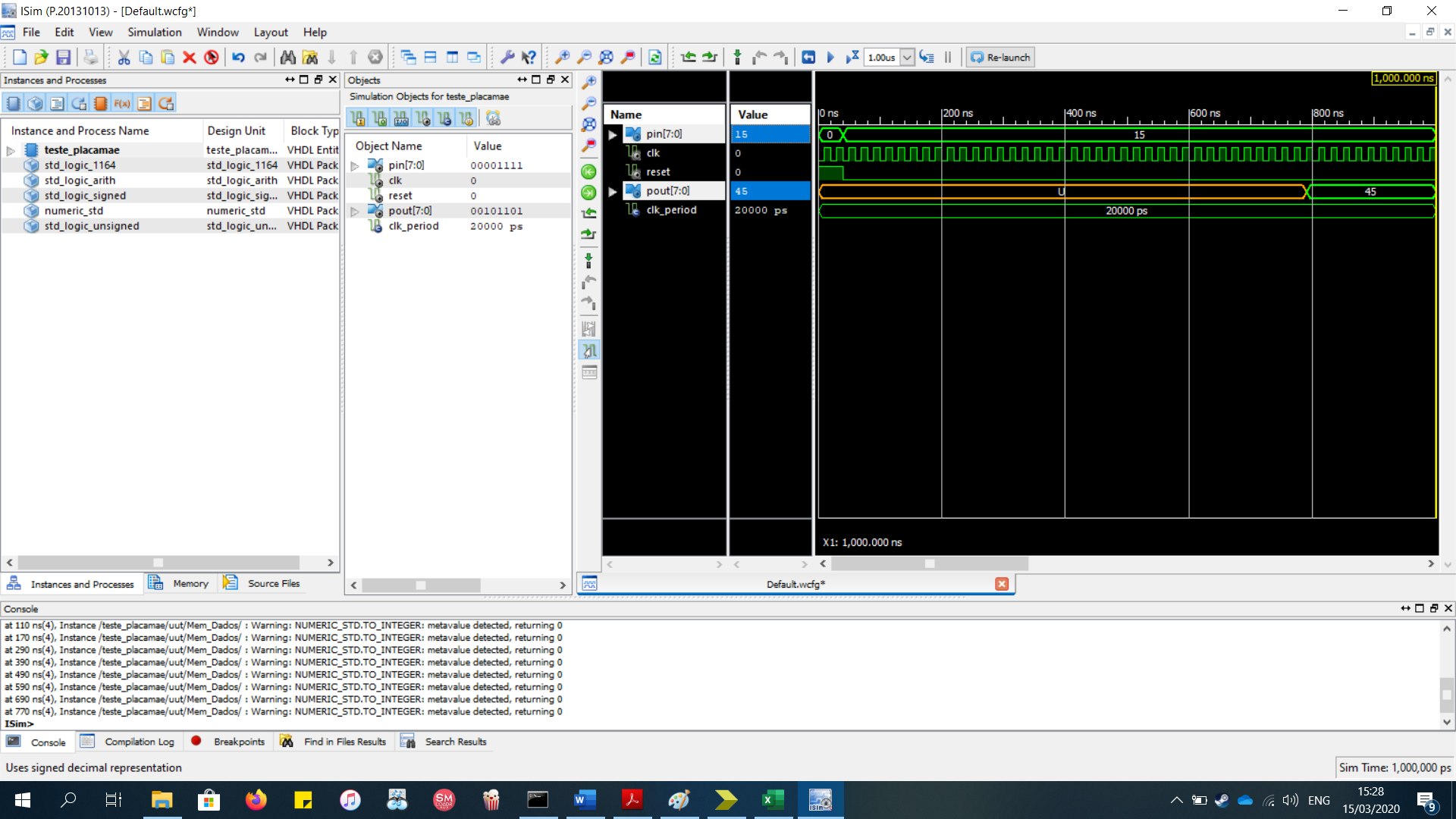
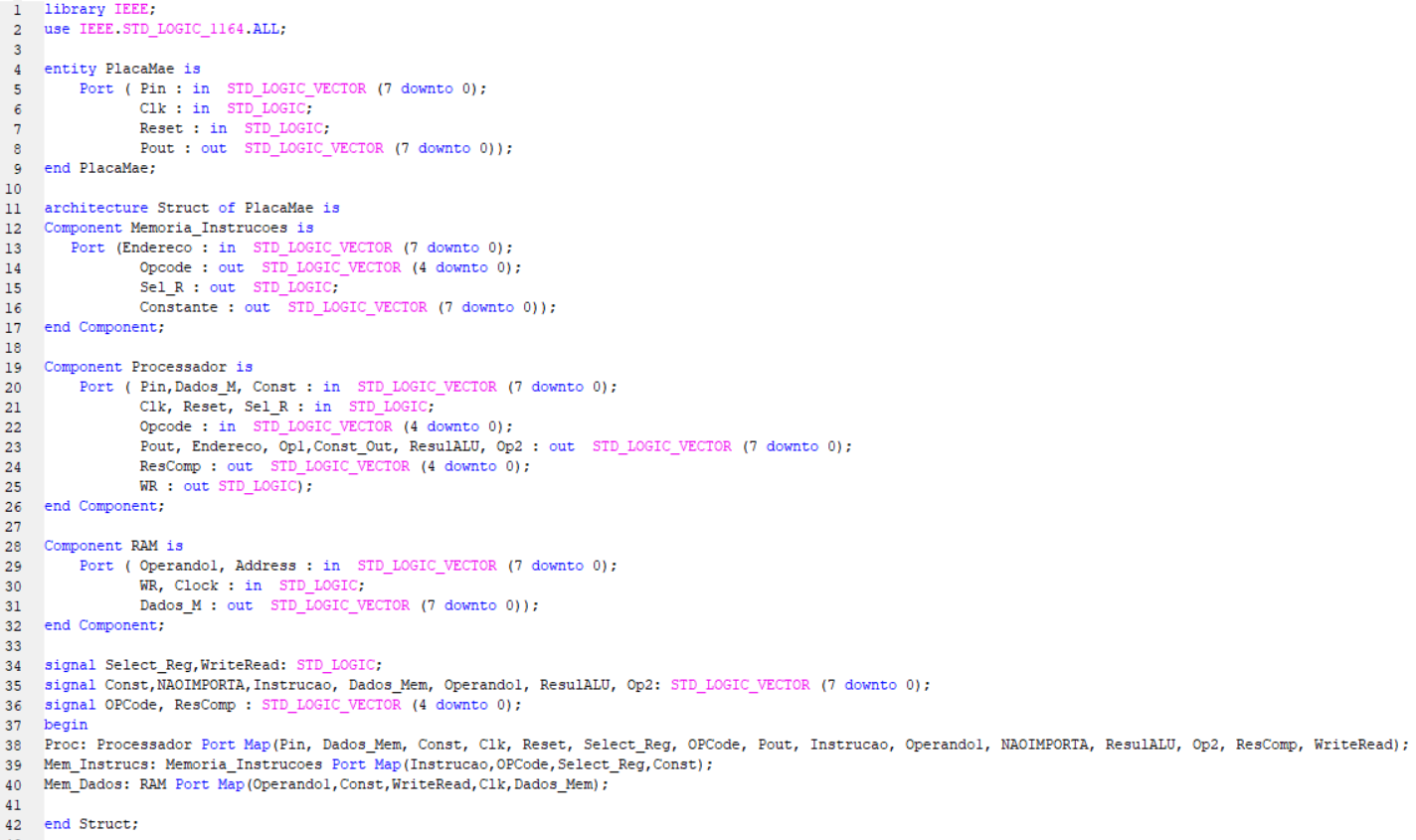
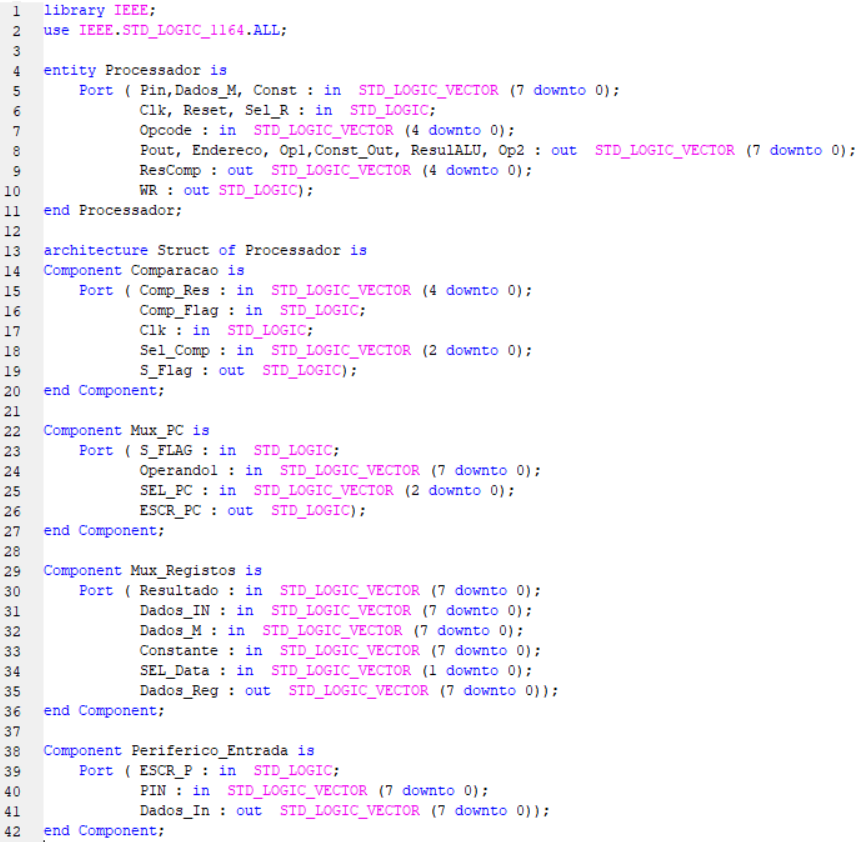
* 1. **Simulação com o 0<=PIN<40**

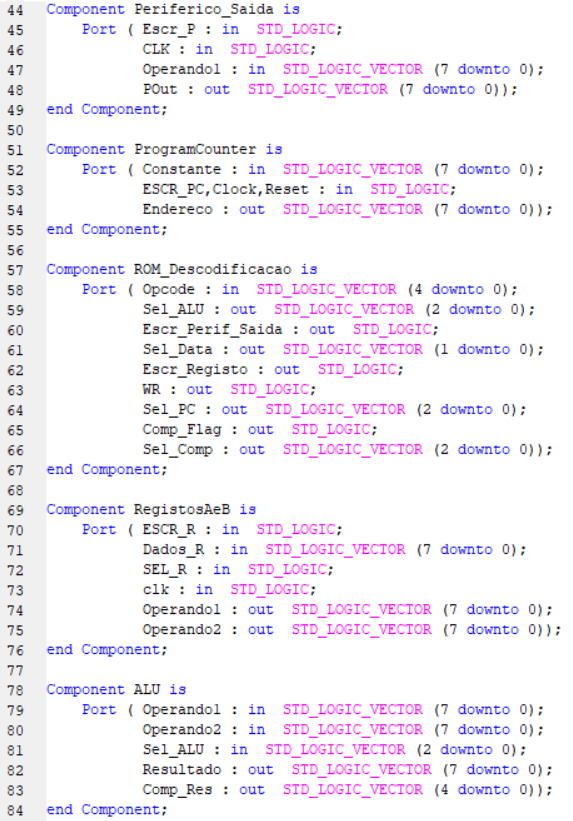
Ilustração Simulação com o valor de PIN a 15

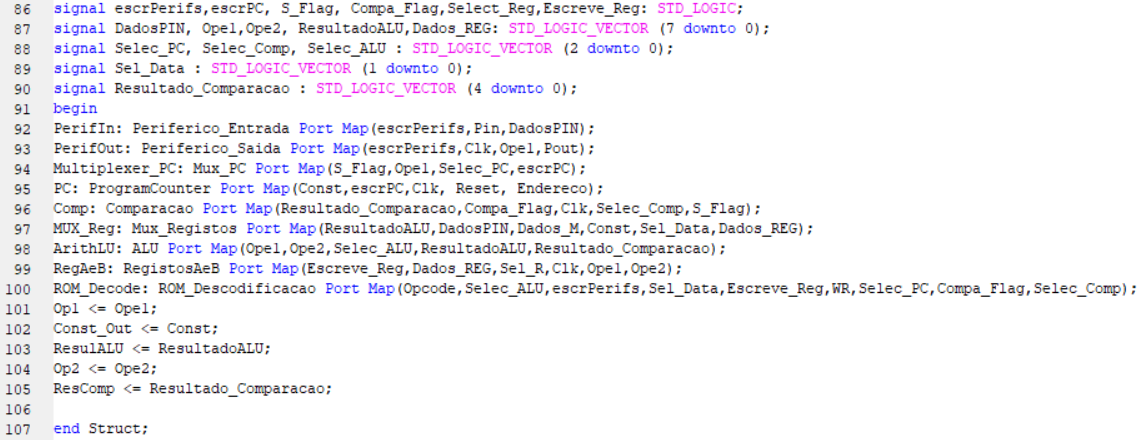
1. **Anexo B**
   1. **Placa-Mãe**



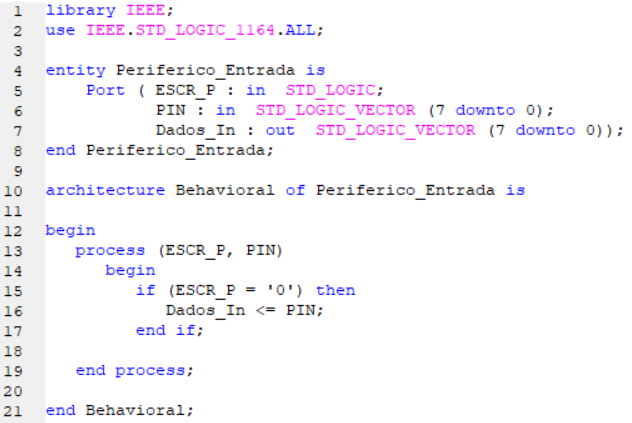
* + 1. **Processador**



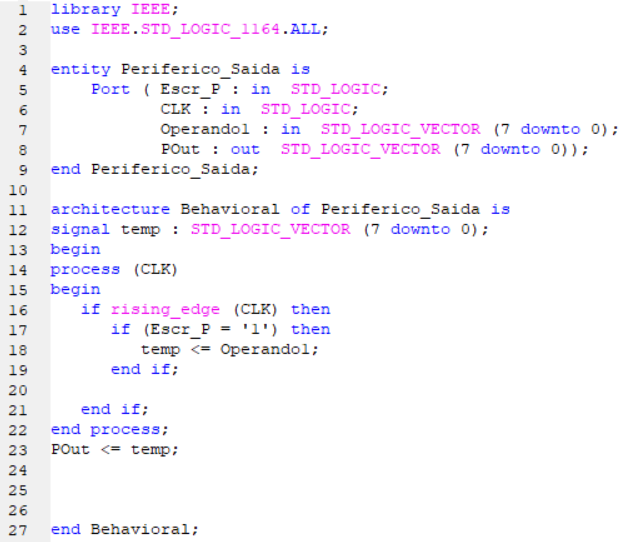




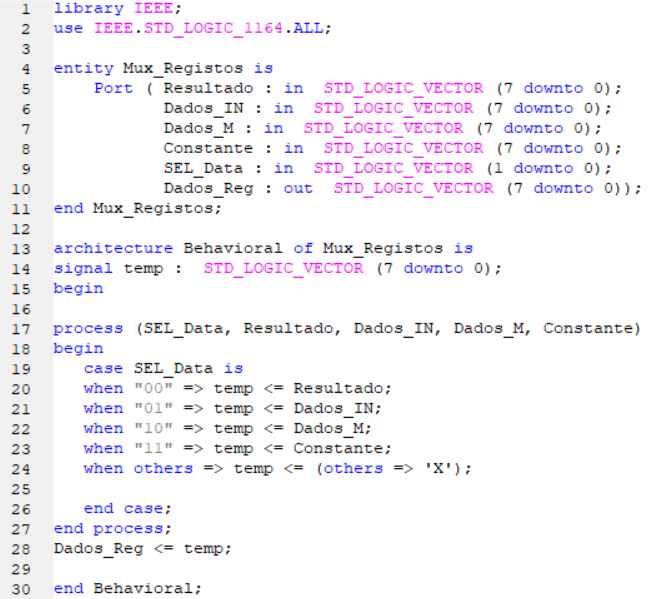
* + - 1. **Periférico de Entrada**



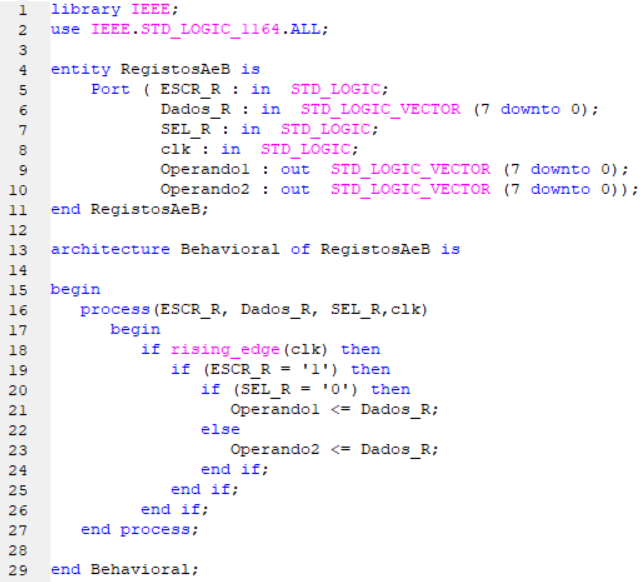
* + - 1. **Periférico de Saída**



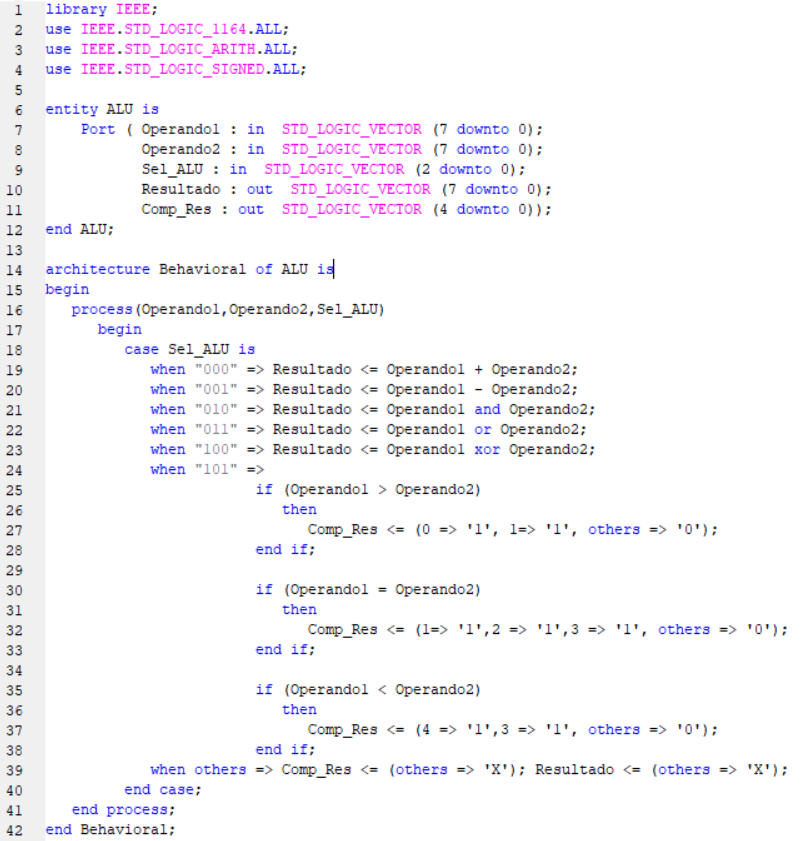
* + - 1. **Multiplexer dos Registos (Mux R)**



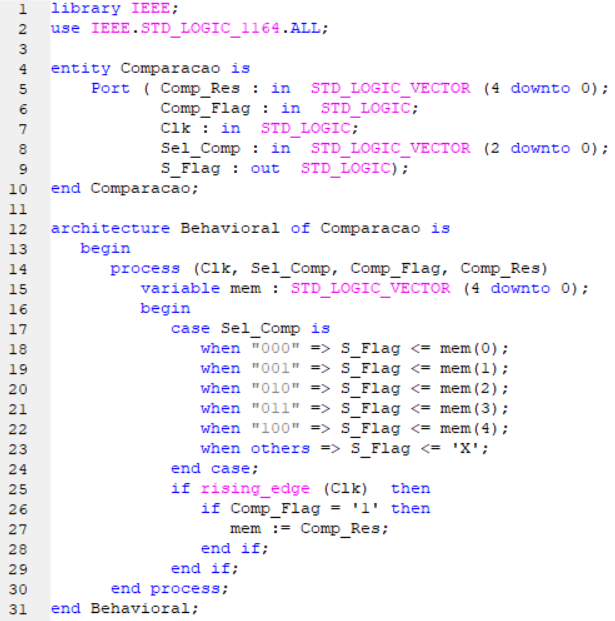
* + - 1. **Registos A e B**



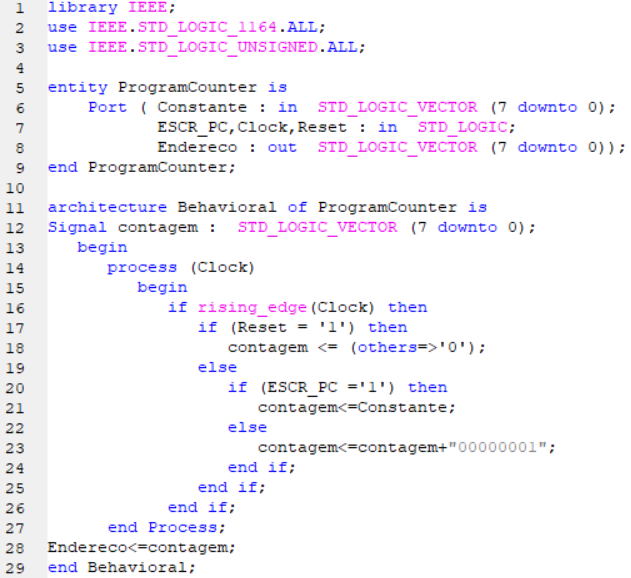
* + - 1. **Unidade Aritmética e Lógica (ALU)**



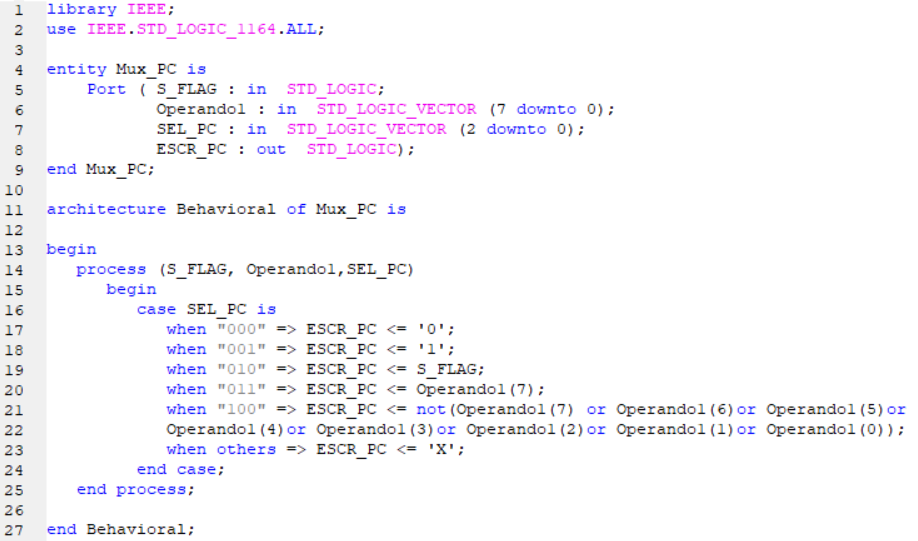
* + - 1. **Comparação**



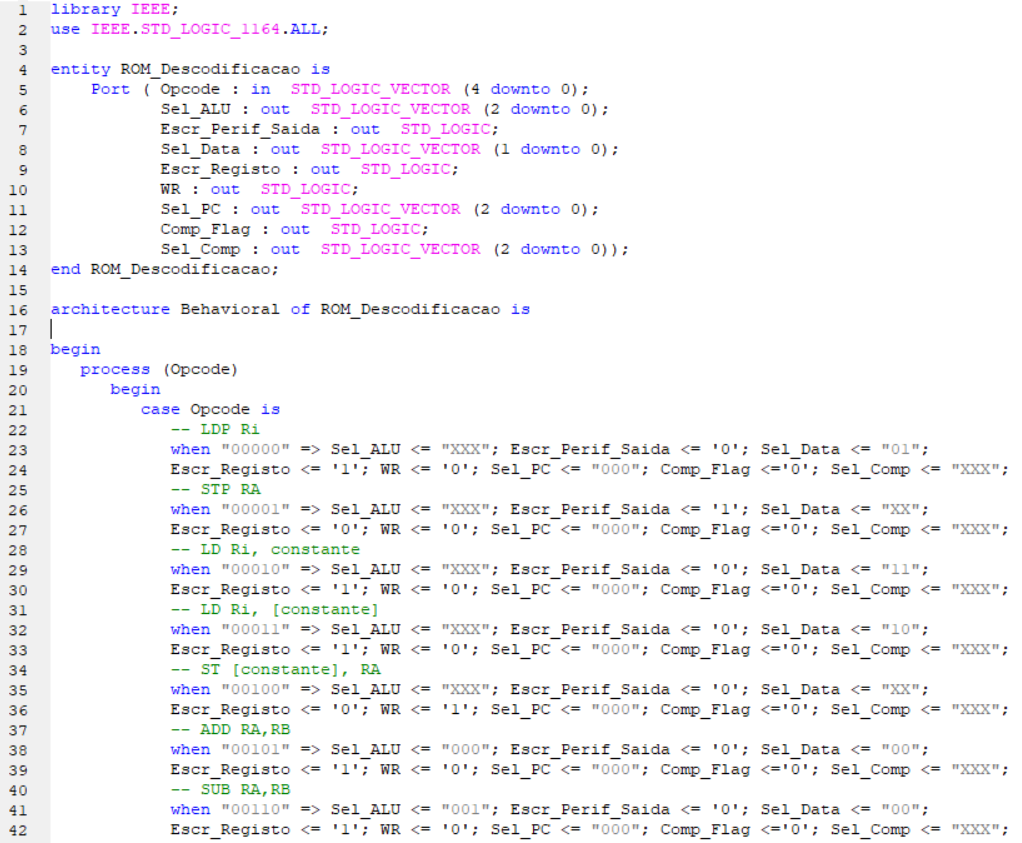
* + - 1. **Contador de programa (PC)**

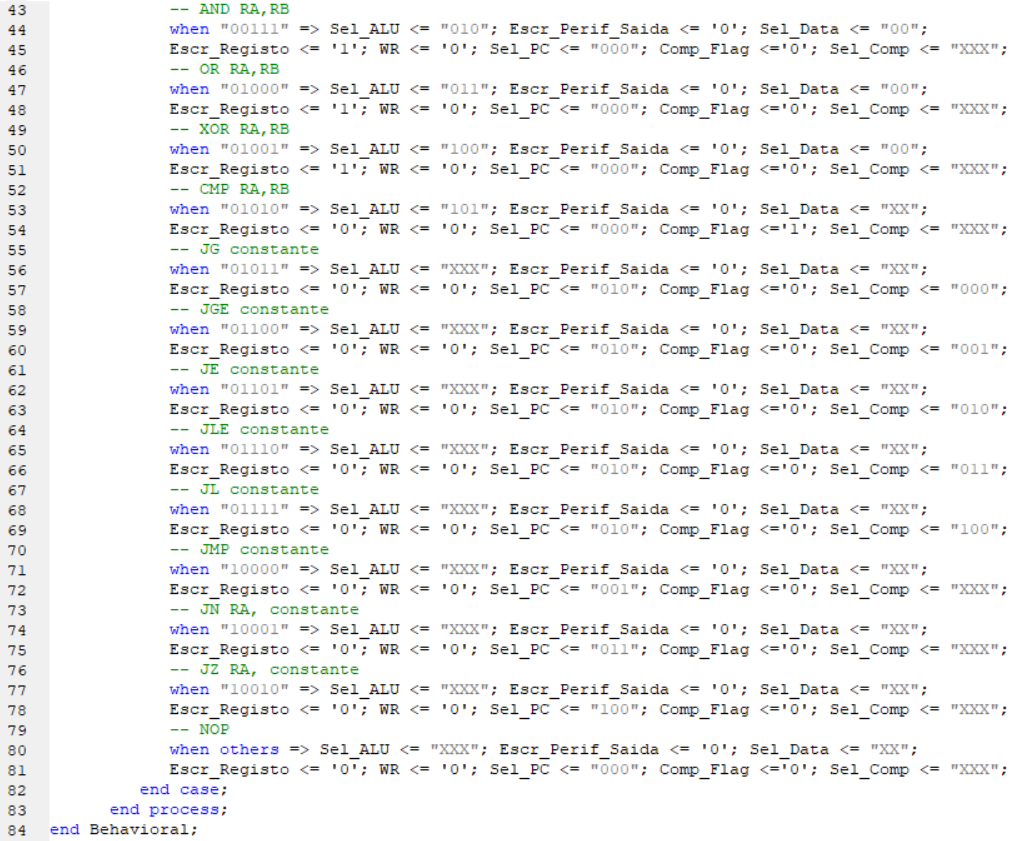


* + - 1. **Multiplexer do Program Counter (Mux\_PC)**



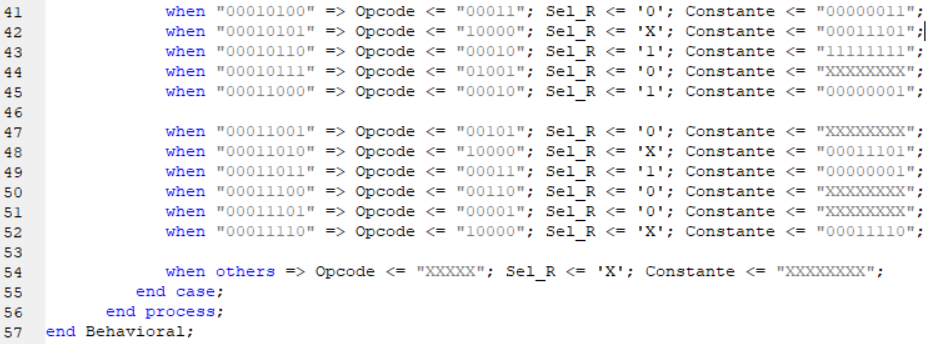
* + - 1. **ROM de descodificação (ROM)**





* + 1. **Memória de Instruções**





* + 1. **Memória de Dados (RAM)**

