

**PODER EXECUTIVO**

**MINISTÉRIO DA EDUCAÇÃO**

**UNIVERSIDADE FEDERAL DE RORAIMA**

**DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO**

**ARQUITETURA E ORGANIZAÇÃO DE COMPUTADORES**

**RELATÓRIO DO PROJETO: PROCESSADOR MK-IV**

**ALUNOS:**

**Hendrick Silva Ferreira 2020026830**

**Vitor Jordão Carneiro Briglia 2021013087**

**Março de 2025**

**Boa Vista/Roraima**





**PODER EXECUTIVO**

**MINISTÉRIO DA EDUCAÇÃO**

**UNIVERSIDADE FEDERAL DE RORAIMA**

**DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO**

**ARQUITETURA E ORGANIZAÇÃO DE COMPUTADORES**

**RELATÓRIO DO PROJETO: PROCESSADOR MK-IV**

**Março de 2025**

**Boa Vista/Roraima**

**Resumo**

Este relatório tem como objetivo apresentar a implementação de um processador uniciclo de 8 bits denominado MK-IV (Processador de 8 Bits). Serão apresentados seus principais componentes, bem como suas funcionalidades. O processador é capaz de executar instruções básicas como load, store, subtração, soma, beq e bne (saltos condicionais) e salto incondicional.

O componente é capaz de executar 3 tipos de instrução, que podem ser encontrados em um processador de arquitetura MIPS (O MIPS é uma arquitetura baseada em registrador, ou seja, a CPU utiliza apenas registradores para realizar suas operações aritméticas e lógicas). Instruções do tipo R (que utiliza os registradores como operandos), tipo I (que acessa um registrador com valor imediato), tipo J (instruções incondicionais, que irão mudar para outro endereço).

O processador é Uniciclo, ou seja, as instruções são executadas em apenas um ciclo de clock.

**Conteúdo**

[1 Especificação 7](#_Toc98084813)

[1.1 Plataforma de desenvolvimento 7](#_Toc98084814)

[1.2 Conjunto de instruções 7](#_Toc98084815)

[1.3 Descrição do Hardware 9](#_Toc98084816)

[1.3.1 ALU ou ULA 9](#_Toc98084817)

[1.3.2 Banco de registradores 10](#_Toc98084818)

[1.3.3 Clock 11](#_Toc98084819)

[1.3.4 Unidade de Controle 11](#_Toc98084820)

[1.3.5 Memória de dados 13](#_Toc98084821)

[1.3.6 Memória de Instruções 13](#_Toc98084822)

[1.3.7 PC Counter 14](#_Toc98084823)

[1.3.8 And 14](#_Toc98084824)

[1.3.9 Mux\_2x1 15](#_Toc98084825)

[1.4 PC 15](#_Toc98084826)

[1.5 ZERO 16](#_Toc98084827)

[1.6 Datapath 16](#_Toc98084828)

[2 Simulações e Testes 17](#_Toc98084829)

1.1 Descrição do programa 18

1.2 Waveform 18

3 Considerações Finais 20

4 Repositorio do projeto 20

**Lista de Figuras**

FIGURA 1 - ESPECIFICAÇÕES NO QUARTUS………………………………………7

FIGURA 2 - RTL DA ULA………………………………………………………………..10

FIGURA 3 - RTL DO BANCO DE REGISTRADORES……………………………….11

FIGURA 4 - RTL DA UNIDADE DE CONTROLE………………………………………12

FIGURA 5 - RTL DA MEMÓRIA DE DADOS…………………………………………..14

FIGURA 6 - RTL DA MEMÓRIA DE INSTRUÇÕES…………………………………...15

FIGURA 7 - RTL DO PC COUNTER…………………………………………………….15

FIGURA 8 - RTL DO AND………………………………………………………………...16

FIGURA 9 - RTL DO MUX\_2X1………………………………………………………….16

FIGURA 10 - RTL DO PC…………………………………………………………………17

FIGURA 11 - RTL DO ZERO …………………………………………………………….17

FIGURA 12 - RTL DO MK-IV…………………………………………………………….18

FIGURA 13 – WAVEFORM 1…………………………………………………………..20

FIGURA 14 – WAVEFORM 2…………………………………………………………..21

FIGURA 15 – WAVEFORM 3…………………………………………………………..22

**Lista de Tabelas**

TABELA 1 – TABELA QUE MOSTRA A LISTA DE OPCODES DE UTILIZADOS PELO PROCESSADOR MK-IV……………………………………………………..……9

TABELA 2 – DETALHES DAS FLAGS DE CONTROLE DOPROCESSADOR……13

TABELA 3 - CÓDIGO FIBONACCI PARA O PROCESSADOR MK-IV……………..19

# **1 Especificação**

Nesta seção é apresentado o conjunto de itens para o desenvolvimento do processador MK-IV, bem como a descrição detalhada de cada etapa da construção do processador.

## **1.1 Plataforma de desenvolvimento**

Para a implementação do processador MK-IV foi utilizado o software Quartus Prime Lite Edition 20.1, software da Intel. Ele possui IDE para codificação, gerador de waveforms, possui ainda um visualizador de RTL, entre outras funcionalidades.

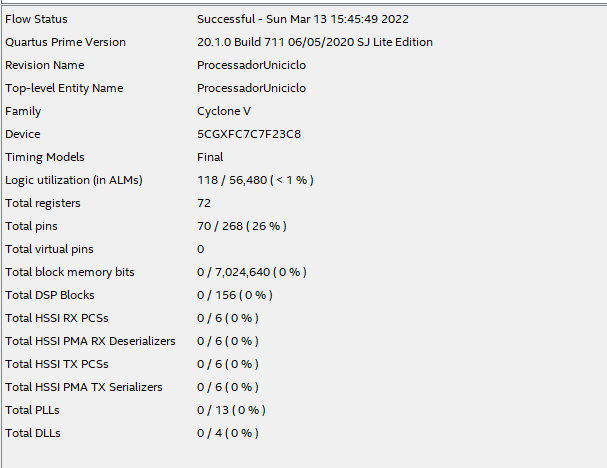


FIGURA 1- ESPECIFICAÇÕES NO QUARTU

## **1.2 Conjunto de instruções**

O processador MV-IV possui 4 registradores: S0, S1, S2, S3. Assim como 3 formatos de instruções de 8 bits cada, seguem algumas considerações sobre as estruturas contidas nas instruções:

Tipo R:

**Opcode (4 bits):** A operação básica a ser executada pelo processador, tradicionalmente chamado de código de operação;

R**eg1 (5° e 6°bits):** O registrador contendo o primeiro operando fonte e adicionalmente para alguns tipos de instruções (ex. instruções do tipo R) é o registrador de destino;

**Reg2 (7° e 8° bits):** O registrador contendo o segundo operando fonte;

Tipo I:

**Opcode (4 bits):** Define qual a operação será executada pelo processador;

**Reg1 (5° e 6°bits):** Endereço do registrador que será acessado;

**Imediato (5° e 6°bits)**

Tipo J:

**Opcode (4 bits):** Define qual operação será executada;

**Endereço (últimos 4 bits):** Endereço que será acessado pelo jump**;**

Tipo de Instruções:

**Tipo R:** Este formatado aborda instruções de Load (exceto *load Immediately*), Store e instruções baseadas em operações aritméticas.

Formato para escrita de código em linguagem de alto nível:

|  |  |  |
| --- | --- | --- |
| Tipo da Instrução | Reg1 | Reg2 |

Formato para escrita em código binário:

|  |  |  |
| --- | --- | --- |
| 4 bits | 2 bits | 2 bits |
| 7-4 | 3-2 | 1-0 |
| Opcode | Reg1 | Reg2 |

**Tipo I:** Este formato aborda instruções que usam um valor gerado no decorrer do programa;

Formato para escrita de código em linguagem de alto nível:

|  |  |  |
| --- | --- | --- |
| Tipo da Instrução | Reg1 | Reg2 |

Formato para escrita em código binário:

|  |  |  |
| --- | --- | --- |
| 4 bits | 2 bits | 2 bits |
| 7 - 4 | 3 - 2 | 1-0 |
| Opcode | Reg1 | Imediato |

**Tipo J:** Este formato está relacionado às instruções de salto incondicional (jump), e salto condicional (bne beq);

Formato para escrita de código em linguagem de alto nível:

|  |  |
| --- | --- |
| Tipo de Instrução | Endereço |

Formato para escrita em código binário:

|  |  |
| --- | --- |
| 4 bits | 2 bits |
| 7 - 4 | 3 - 0 |
| Opcode | Endereço |

**Visão geral das instruções do Processador P8B:**

O número de bits do campo **Opcode** das instruções é igual a quatro, sendo assim obtemos um total (bit(0e1) = 16) de 16 Opcodes **(0-15)** que são distribuídos entre as instruções, assim como é apresentado na Tabela 1.

**Tabela 1 – Tabela que mostra a lista de Opcodes utilizadas pelo processador MK-IV.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Opcode** | **Nome** | **Formato** | **Breve Descrição** | **Exemplo** |
| 0000 | LW | R | Load | lw S0, memória(00) |
| 0001 | SW | R | Store | sw S0, memória(00) |
| 0010 | ADD | R | Soma | add S0, S1, ou seja: S0 = S0 + S1 |
| 0011 | SUB | R | Subtração | sub S0, S1, ou seja: S0 = S0 - S1 |
| 0100 | ADDI | I | Adição imediata | addi S0, 11, ou seja: S0 = S0 + 3 |
| 0101 | SUBI | I | Subtração imediata | subi S0, 11, ou seja: S0 = S0 - 3 |
| 0110 | MOVE | R | Move | move S0 S1, ou seja: S0 = S1 |
| 0111 | LI | I | Load imediato | li S0, 11, ou seja: S0 = 3 |
| 1000 | BEQ | J | Branch if equal | beq 0000 |
| 1001 | BNE | J | Branch if not equal | bne 0000 |
| 1010 | CMP | R | Comparação | cmp S0, S1 |

## **1.3 Descrição do Hardware**

Nesta seção são descritos os componentes do hardware que compõem o processador Quantum, incluindo uma descrição de suas funcionalidades, valores de entrada e saída.

### **1.3.1 ALU ou ULA**

O componente ALU (Unidade Lógica Aritmética) tem como principal objetivo efetuar as principais operações aritméticas, dentre elas: soma, subtração, divisão (considerando apenas resultados inteiros) e multiplicação. Adicionalmente o QALU efetua operações de comparação de valor como maior ou igual, menor ou igual, somente maior, menor ou igual. O componente QALU recebe como entrada três valores: **A** – dado de 8bits para operação; **B** - dado de 8bits para operação e **OP** – identificador da operação que será realizada de 4bits. O QALU também possui três saídas: **zero** – identificador de resultado (2bit) para comparações (1 se verdade e 0 caso contrário); **overflow** – identificador de overflow caso a operação exceda os 8bits; e **result** – saída com o resultado das operações aritméticas;

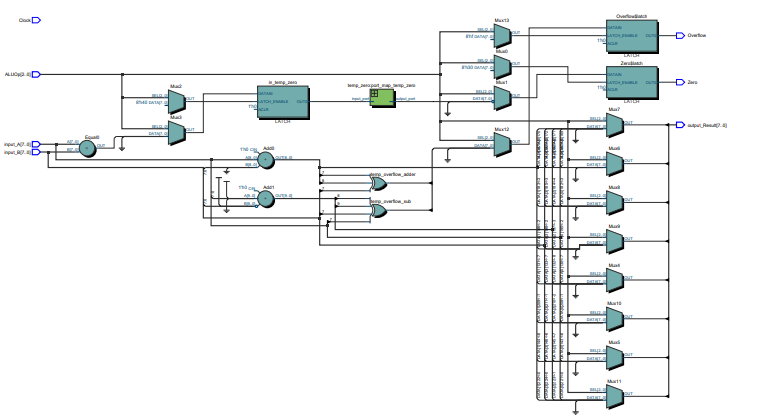


FIGURA 2 – RTL DA ULA

### **1.3.2 Banco de registradores**

O Banco de registradores são uma forma de armazenar uma quantidade bits de forma mais rápida, principalmente por dispensar o acesso a memória, que no geral é mais lenta de acessar. Sua utilidade no desenvolvimento de processadores se dá pelo fato de poder armazenar e transferir dados entre outros registradores de modo mais eficaz.

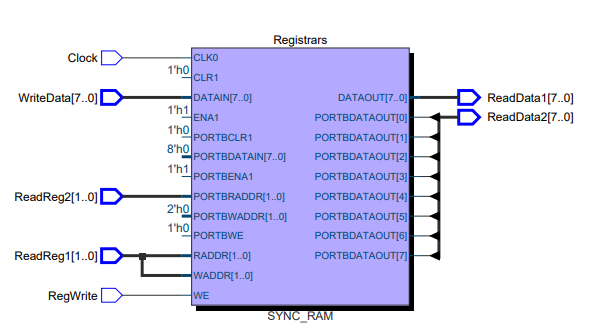


FIGURA 3 – RTL DO BANCO DE REGISTRADORES

### **1.3.3 Clock**

O clock não foi implementado, porém é bastante importante para o funcionamento do processador, pois é o responsável pelo controle de ciclos da unidade e também simular os clocks.

### **1.3.4 Unidade de Controle**

A unidade de controle é componente responsável por definir o caminho de dados de cada instrução, com base no opcode.

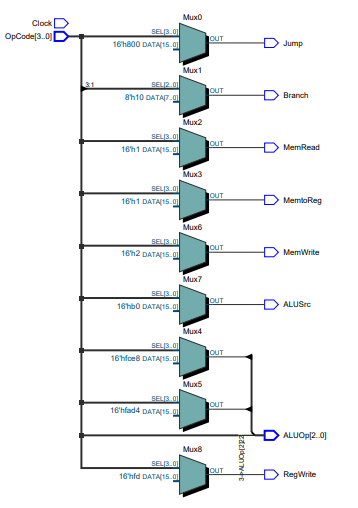


FIGURA 4 – RTL DA UNIDADE DE CONTROLE

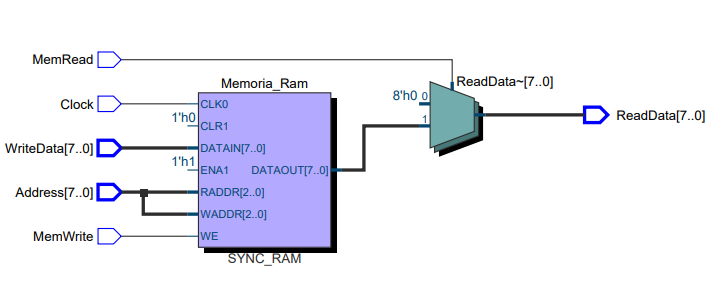
Abaixo segue a tabela, onde é feita a associação entre os opcodes e as flags de controle:

**Tabela 2 - Detalhes das flags de controle do processador.**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Intrução** | **jump** | **Branch** | **MemRead** | **MentoReg** | **ALUOp** | **MemWrite** | **ALUSre** | **RegWrite** |
| **lw** | 0 | 0 | 1 | 1 | 000 | 1 | 0 | 1 |
| **sw** | 0 | 0 | 0 | 0 | 000 | 0 | 0 | 0 |
| **add** | 0 | 0 | 0 | 0 | 001 | 0 | 0 | 1 |
| **sub** | 0 | 0 | 0 | 0 | 010 | 0 | 0 | 1 |
| **addi** | 0 | 0 | 0 | 0 | 001 | 0 | 1 | 1 |
| **subi** | 0 | 0 | 0 | 0 | 010 | 0 | 1 | 1 |
| **move** | 0 | 0 | 0 | 0 | 011 | 0 | 0 | 1 |
| **li** | 0 | 0 | 0 | 0 | 011 | 0 | 1 | 1 |
| **beq** | 0 | 1 | 0 | 0 | 100 | 0 | 0 | 0 |
| **bne** | 0 | 1 | 0 | 0 | 101 | 0 | 0 | 0 |
| **cmp** | 0 | 0 | 0 | 0 | 110 | 0 | 0 | 0 |
| **j** | 1 | 0 | 0 | 0 | 111 | 0 | 0 | 0 |

### **1.3.5 Memória de dados**

É a unidade responsável pelo armazenamento de dados, podendo ser permanentes ou temporários. a memória de dados tem uma capacidade de armazenamento maior que a do banco de registradores e acaba sendo útil em programas que exigem a utilização de todos os registradores do banco ou que não se deseja perder o resultado de uma operação.

FIGURA 5 – RTL DA MEMÓRIA DE DADOS

### **1.3.6 Memória de Instruções**

A memória de instruções é componente responsavél por armazenar os passos das instruções. Dado um endereço, apresentar com saída a instrução correspondente.

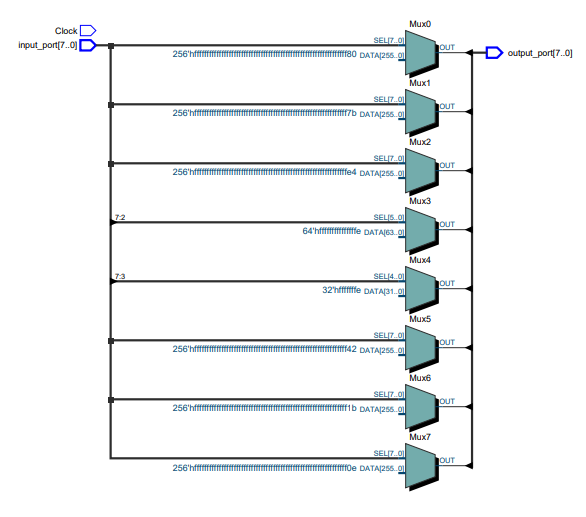


FIGURA 6 – RTL DA MEMÓRIA DE INSTRUÇÕES

### **1.3.7 PC Counter**

É um componente que responsável por adicionar 1 passo no PC, ele adiciona 1 ao endereço recebido do PC fazendo com ele seja atualizado.

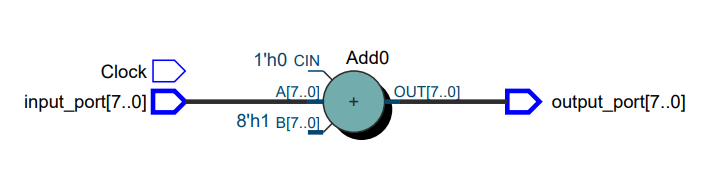


FIGURA 7 – RTL DO PC COUNTER

### **1.3.8 And**

É um subcomponente que recebi 2 bits, sendo positivo quando ambos os bits valem 1.

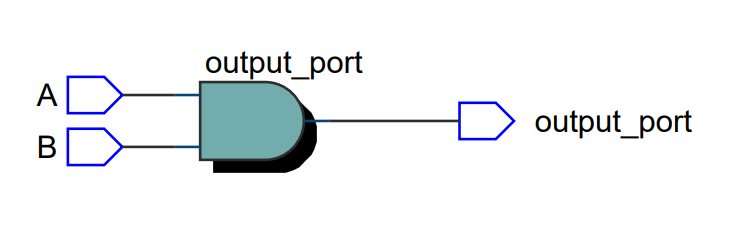


FIGURA 8 – RTL DO AND

### **1.3.9 Mux\_2x1**

O mutiplexador é um componente ultilizado para selecionar dentre 2 valores qual será usado no processador.

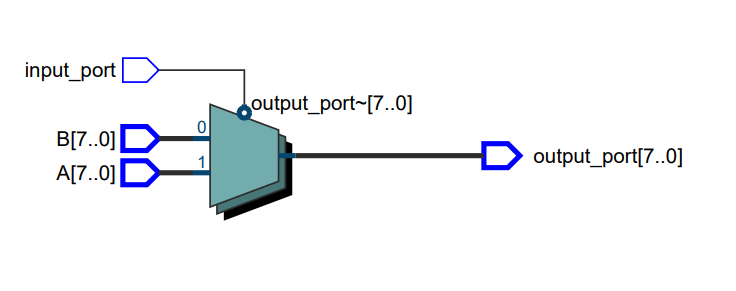


FIGURA 9 – RTL DO MUX\_2X1

### **1.4 PC**

O PC (Program Counter) é um componente importante pois é o responsável por sequenciar o código, ao receber um clock igual a 1, recebe o valor de 8 bits, que são referentes ao endereço de uma instrução, e manda para 2 componentes para o PC Counter e a Memória de Instruções.

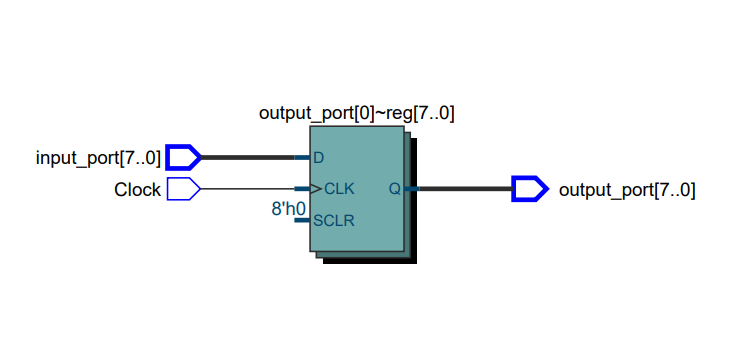


FIGURA 10 – RTL DO PC

### **1.5 ZERO**

O zero é um componente que fica dentro da ULA, e somente é iniciado quando uma operação de comparação é solicitada, sua função basicamente é iniciar a flag necessária para essa operação.

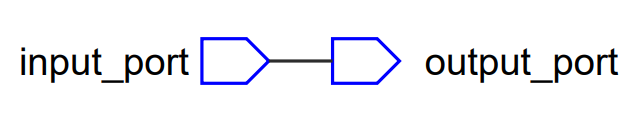
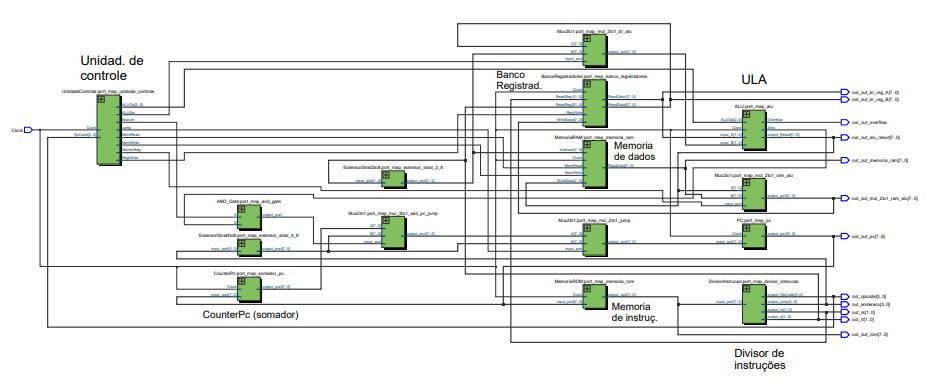


FIGURA 11 – RTL DO ZERO

## **1.6 Datapath**

É a conexão entre as unidades funcionais formando um único caminho de dados e acrescentando uma unidade de controle responsável pelo gerenciamento das ações que serão realizadas para diferentes classes de instruções.



1 FIGURA 2 – RTL DO DATAPATH

# **2 Simulações e Testes**

Objetivando analisar e verificar o funcionamento do processador, efetuamos alguns testes analisando cada componente do processador em especifico, em seguida efetuamos testes de cada instrução que o processador implementa. Para demonstrar o funcionamento do processador MK-IV utilizaremos como exemplo o código para calcular o número da sequência de Fibonacci.

**Tabela 3 - Código Fibonacci para o processador MK-IV**

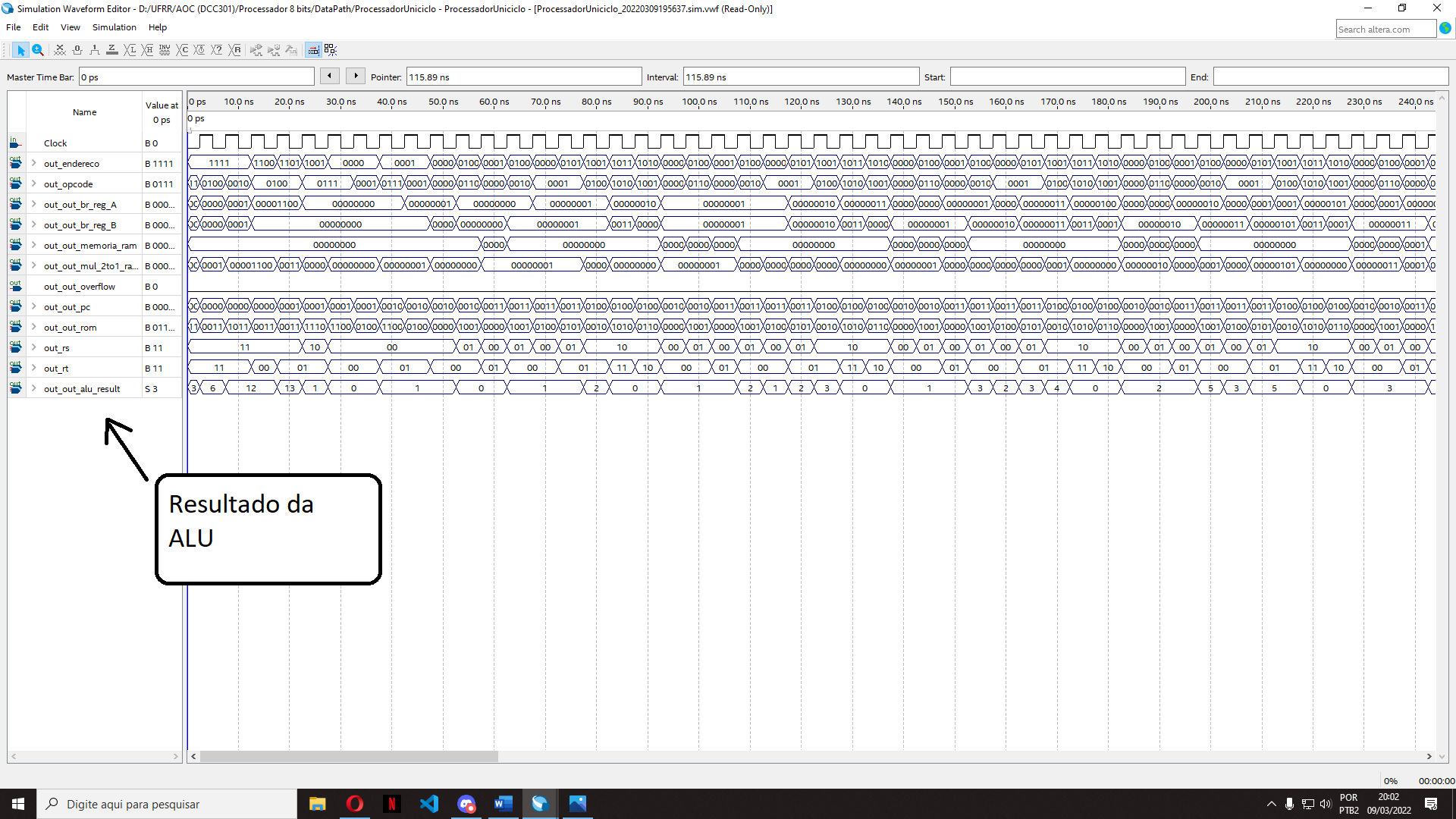
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Endereço** | **Instrução** | **Linguagem de Alto Nível** | **Binário** | | |
| Opcode | Reg2 | Reg1 |
| Endereço | |
| imediato | | |
| 0 | 01111111 | li S3, 3 | 0111 | 11 | 11 |
| 1 | 01001111 | addi S3, 3 | 0100 | 11 | 11 |
| 2 | 00101111 | add S3, S3 | 0010 | 11 | 11 |
| 3 | 01001100 | addi S3, 0 | 0100 | 11 | 10 |
| 4 | 01001101 | addi S3, 1 | 0100 | 11 | 01 |
| 5 | 01111001 | li S2, 1 | 0111 | 10 | 01 |
| 6 | 01110000 | S0, 0 | 0111 | 00 | 00 |
| 7 | 00010000 | sw S0, ram(00) | 0001 | 00 | 00 |
| 8 | 01110001 | li S0, 1 | 0111 | 00 | 01 |
| 9 | 00010001 | sw S0, ram(01) | 0001 | 00 | 01 |
| 10 | 00000000 | lw S0, ram(00) | 0000 | 00 | 00 |
| 11 | 01100100 | move S1, S0 | 0110 | 01 | 00 |
| 12 | 00000001 | lw S0, ram(01) | 0000 | 00 | 01 |
| 13 | 00100100 | add S1, S0 | 0010 | 01 | 00 |
| 14 | 00010000 | sw S0, ram(00) | 0001 | 00 | 00 |
| 15 | 00010101 | sw s1, ram(01) | 0001 | 01 | 01 |
| 16 | 01001001 | addi s2, 1 | 0100 | 10 | 01 |
| 17 | 10101011 | cmp S2, S3 | 1010 | 10 | 11 |
| 18 | 10011010 | bne 1010 | 1001 | 10 | 10 |
| 19 | 10011010 | li S0, 0 | 0111 | 00 | 00 |
| 20 | 01110100 | li S1, 0 | 0111 | 01 | 00 |
| 21 | 01111000 | 0 li S2, 0 | 0111 | 10 | 00 |
| 22 | 01111100 | li S3, 0 | 0111 | 11 | 00 |

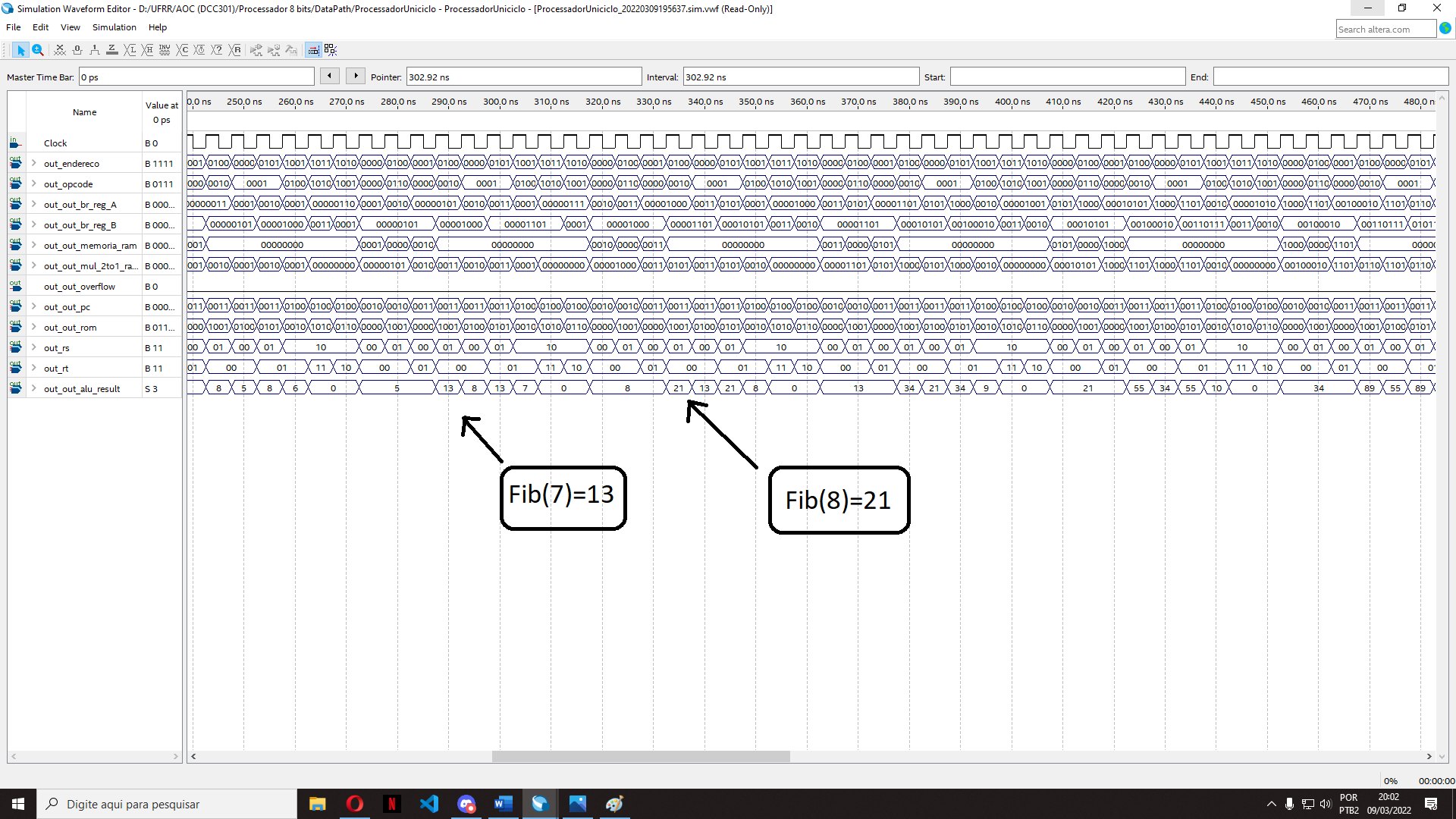
**2.1 Descrição do Programa**

O programa acima descrito representa o cálculo da sequência de Fibonacci. Os 4 registradores foram utilizados nesse programa, sendo que S0 foi utilizado para acessar os valores da RAM, S1 como auxiliar da soma, S2 como contador e, finalmente, S3 para o número Fibonacci objetivado. Enquanto o programa avança, são armazenados na memória RAM o último número da sequência e seu anterior.

**2.2 Waveform**

Verificando o resultado da simulação, após a simulação ser concluída temos a a seguinte waverform.

FIGURA 13 – WAVEFORM

FIGURA 14 – WAVEFORM

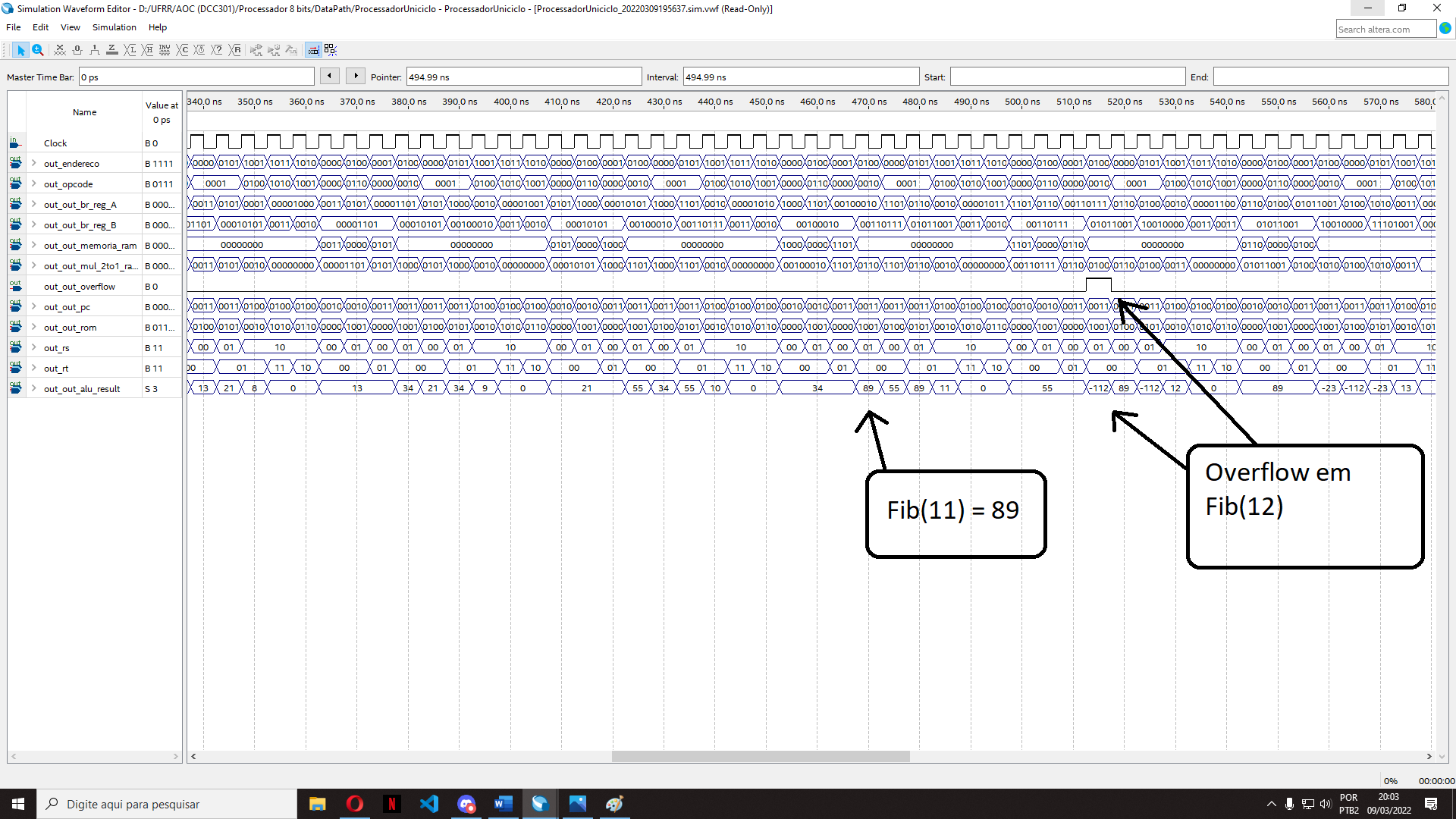


FIGURA 15 - WAVEFORM

**3 Considerações Finais**

Este relatório teve como objetivo apresentar o projeto e a implementação de um processador uniciclo de 8 bits denominado de MK-IV, bem como seus componentes principais e suas funcionalidades. Foi apresentado também o Datapath do processador com seus barramentos, esse processador pode executar os 3 tipos de instruções presentes em um processador de arquitetura MIPS, instruções do tipo R, tipo I, tipo J.

**4 Repositório do projeto**

[https://github.com/vitor1616/AOC\_VitorJordao\_HendrikSilva\_UFRR2024\_ProjetoFinal.git](https://github.com/Lucasx10/AOC_Lucas_Prado_Lucas_Ribeiro_UFRR_2022.git)