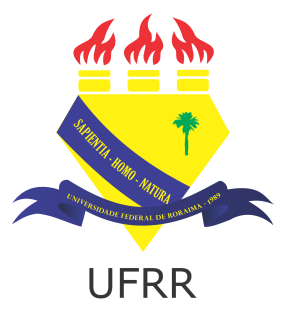
**PODER EXECUTIVO**

**MINISTÉRIO DA EDUCAÇÃO**

**UNIVERSIDADE FEDERAL DE RORAIMA**

**DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO**



**ARQUITETURA E ORGANIZAÇÃO DE COMPUTADORES**

**RELATÓRIO DO PROJETO: PROCESSADOR INTEL BAT**

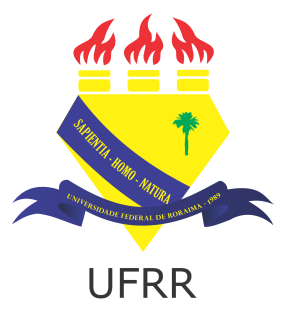
**ALUNOS:**

**Vitor Jordão Carneiro Briglia - 2021013087**

**Marcos Vinícius Tenacol Coêlho - 2021000759**

**Dezembro de 2022**

**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 INTEL BAT**

**Dezembro de 2022**

**Boa Vista/Roraima**

**Resumo**

Este trabalho aborda o projeto e implementação de um processador uni-ciclo de 8 bits com o que foi estudado na disciplina de arquitetura e organização de computadores. Tudo que foi utilizado e implementado durante os testes para seu funcionamento será descrito. O processador feito é capaz de executar 6 instruções: load, store, add, jump, beq e sub. Toda a implementação foi feita utilizando a linguagem Logsim, junto dos testes dos componentes. Usando instruções do próprio processador ele possui uma variedade de operações diferentes.

O processador **INTEL BAT** possui 4 registradores, podendo acessar até dois ao mesmo tempo: Ra e Rb, assim como 6 formatos de instruções de 8 bits cada, categorizadas pelos tipos **R, I e J**. As instruções do tipo R são as que acessam os registradores, enquanto que as do tipo I acessam um registrador e um valor imediato. Por fim, as instruções do tipo J alteram diretamente o valor do PC, condicionalmente ou não.

**Conteúdo**

[1 Especificação](#_heading=h.3znysh7) **7**

[1.1 Plataforma de desenvolvimento](#_heading=h.2et92p0) 7

[1.2 Conjunto de instruções](#_heading=h.3dy6vkm) 8

[1.3 Descrição do Hardware](#_heading=h.4d34og8) 10

[1.3.1 ALU ou ULA](#_heading=h.2s8eyo1) 10

[1.3.2 Banco de Registradores](#_heading=h.3rdcrjn) 10

[1.3.3 Clock](#_heading=h.26in1rg) 11

[1.3.4 Unidade de Controle](#_heading=h.lnxbz9) 11

[1.3.5 Memória de dados](#_heading=h.1ksv4uv) 12

[1.3.6 Memória de Instruções](#_heading=h.44sinio) 12

[1.3.7 Somador](#_heading=h.2jxsxqh) 13

[1.3.8 And](#_heading=h.z337ya) 13

[1.3.9 Mux\_2x1](#_heading=h.3j2qqm3) 13

[1.3.10 PC](#_heading=h.1y810tw) 13

[1.3.11 ZERO](#_heading=h.4i7ojhp) 13

[1.4 Datapath](#_heading=h.f3ki3ul8sfhm) 14

[2 Simulações e Testes](#_heading=h.1ci93xb) 15

[3 Considerações finais](#_heading=h.qsh70q) 16

**Lista de Figuras**

[Figura 1 - Especificações no logisim.](#_heading=h.tyjcwt) 7

[Figura 2 - Bloco simbólico do componente ALU gerado pelo logisim](#_heading=h.17dp8vu) . 10

Figura 3 - Bloco simbólico do componente [Banco de Registradores](#_heading=h.3rdcrjn) gerado pelo logisim. 11

[Figura 4 - clock. 1](#_heading=h.2bn6wsx)2

[Figura](#_heading=h.2bn6wsx) 5 - Unidade de controle. 12

Figura 6 - memória de dados. 13

Figura 7 - (Memória RAM com dois bits de endereço) 13

Figura 8 - memória de instruções . 14

Figura 9 - (Memória ROM com dois bits de endereço). 14

Figura 10 - Somador +1. 14

Figura 11 - Representação por circuito do Somador+1 de 8bits . 14

Figura 12 - Somador 1 bit 16  
Figura 13 - somador de 8 bits 16

Figura 14 - Representação do AND. 17

Figura 15 - Representação do circuito do mux 2x1. 17

Figura 16 - Registrador tipo D representando um PC. 18

Figura 17 - Representação do ZERO dentro da ULA. 18

Figura 18 - Representação do Datapath. 19

Figura 19 - Resultado na waveform. 20

**Lista de Tabelas**

[Tabela 1 – Tabela que mostra a lista de Opcodes utilizadas pelo processador](#_heading=h.1t3h5sf)

[Intel Bat. 7](#_heading=h.1t3h5sf)

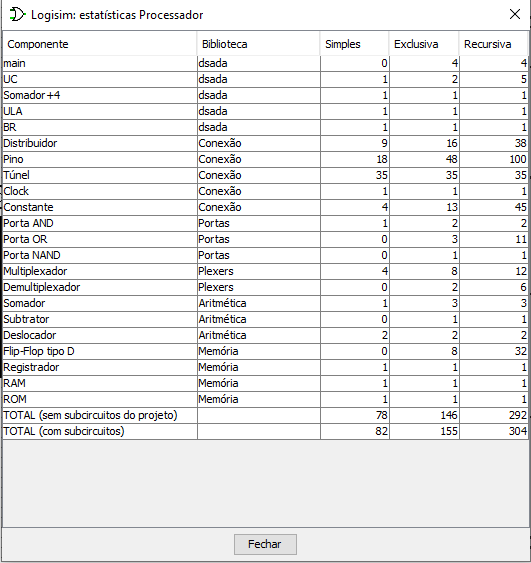
[Tabela 2 - Detalhes das flags de controle do processador. 9](#_heading=h.35nkun2)

# Especificação

Nesta seção é apresentado o conjunto de itens para o desenvolvimento do processador **INTEL BAT**, bem como a descrição detalhada de cada etapa da construção do processador.

## Plataforma de desenvolvimento

Para a implementação do processador **INTEL BAT** foi utilizado a IDE: logisim-win-2.7.1



**Figura 1 - Especificações no logisim**

## Conjunto de instruções

O processador **INTEL BAT** possui 4 registradores, podendo acessar até dois ao mesmo tempo: Ra e Rb. Assim como 6 formatos de instruções de 8 bits cada, Instruções do **tipo R, I e J** seguem algumas considerações sobre as estruturas contidas nas instruções:

* **Opcode**: a operação básica a ser executada pelo processador, tradicionalmente chamado de código de operação;
* **Reg1**: 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**: o registrador contendo o segundo operando fonte;
* **Imediato**: neste tipo de endereçamento, o operando é especificado diretamente no campo de endereço-base da instrução;
* **Endereço**: um valor de endereço.

Tipo de Instruções:

**- Formato do tipo R:** Este formato aborda instruções de Soma e Subtração (Instruções relacionadas a aritmética).

Formato para escrita de código na linguagem Quantum:

| OPCODE  3 | Reg2 | Reg2 | Shamt  X |
| --- | --- | --- | --- |

Formato para escrita em código binário:

| 3 bits | 2 bits | 2 bits | 1 Bit |
| --- | --- | --- | --- |
| 7-5 | 4-3 | 2-1 | 0 |
| Opcode  3 | Reg2 | Reg2 | Shamt  X |

**- Formato do tipo I:** Este formato aborda instruções de Load, Store e BEQ (Instruções relacionadas à imediato).

Formato para escrita de código na linguagem Quantum:

| OPCODE  3 | Reg2 | Reg2 | I  1 |
| --- | --- | --- | --- |

Formato para escrita em código binário:

| 3 bits | 2 bits | 2 bits | 1 Bit |
| --- | --- | --- | --- |
| 7-5 | 4-3 | 2-1 | 0 |
| Opcode  3 | Reg2 | Reg2 | I  1 |

**- Formato do tipo J:** Este formato aborda a instrução Jump.

Formato para escrita de código na linguagem Quantum:

| OPCODE  3 | Endereço  5 |
| --- | --- |

Formato para escrita em código binário:

| 3 bits | 5 bits |
| --- | --- |
| 7-5 | 4-0 |
| Opcode  3 | Endereço  5 |

**Visão geral das instruções do Processador INTEL BAT:**

O número de bits do campo **Opcode** das instruções é igual a três, sendo assim obtemos um total ( de 8 **Opcodes (0-7)** que são distribuídos entre as instruções, assim como é apresentado na Tabela 1.

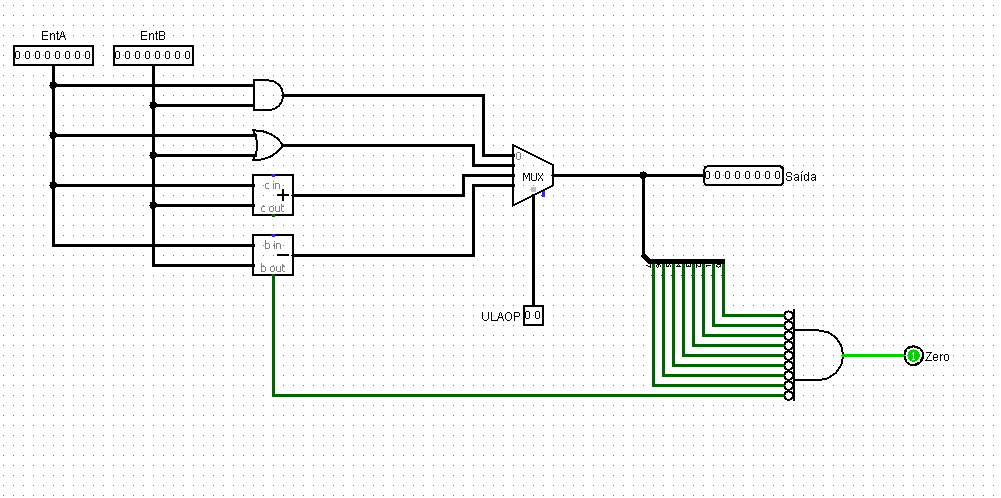
**Tabela 1 – Tabela que mostra a lista de Opcodes utilizadas pelo processador INTEL BAT.**

| **Opcode** | **Nome** | **Formato** | **Breve Descrição** | **Exemplo** |
| --- | --- | --- | --- | --- |
| 000 | LW | I | Load | **lw** $S0, Endereço memória |
| 001 | SW | R | Store | Sw $S0, Endereço memória |
| 010 | ADD | R | Soma | **add** $S0, $S1 ,ou seja, $S0 := $S0+$S1 |
| 011 | SUB | R | Subtração | **sub** $S0, $S1 ,ou seja, $S0 := $S0 - $S1 |
| 100 | BEQ | I | Branch if Equal | **beq** $S0, $S1, Imediato |
| 101 | JUMP | J | Salto Incondicional | **Jump** 0000 |
| 110 e 111 | Não implementados |  |  |  |

## 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.

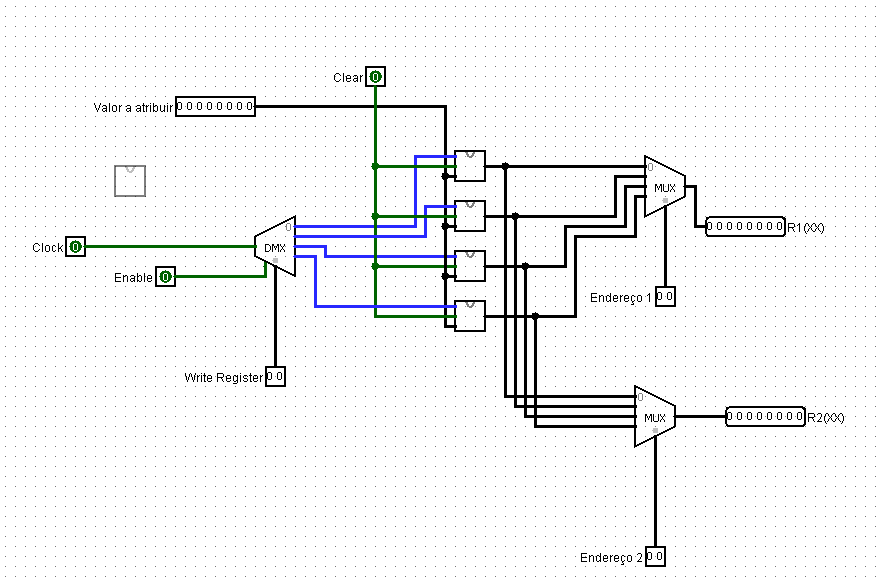
### ALU ou ULA

O componente ALU (Unidade Lógica Aritmética) tem como principal objetivo efetuar as principais operações aritméticas, dentre elas: soma e subtração. Adicionalmente o ALU efetua operações de comparação de valor igual, além das operações AND e OR. O componente ALU 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 2bits. O ALU também possui duas saídas: **zero** – identificador de resultado (2bit) para comparações (1 se verdade e 0 caso contrário); e **resultado** – saída com o resultado das operações aritméticas.

**Figura 2 - Bloco simbólico do componente ALU gerado pelo logsim**

### Banco de Registradores

Um banco de registradores é um componente digital composto por um conjunto de registradores que podem ser acessados de forma organizada. De uma maneira geral, podem ser executadas operações de leitura dos dados anteriormente gravados e de escrita de dados para modificar as informações internas. Este componente é um dos componentes mais importantes do fluxo de dados em um processador.

****

**Figura 3 - Bloco simbólico do componente** [**Banco de Registradores**](#_heading=h.3rdcrjn) **gerado pelo logisim**

### Clock

O clock tem apenas um pino, uma saída com largura de 1 bit, cujo valor representa o estado corrente do clock, os componentes que foram necessários para a entrada de clock estão conectados com um único clock.

****

**Figura 4 - Clock**

### Unidade de Controle

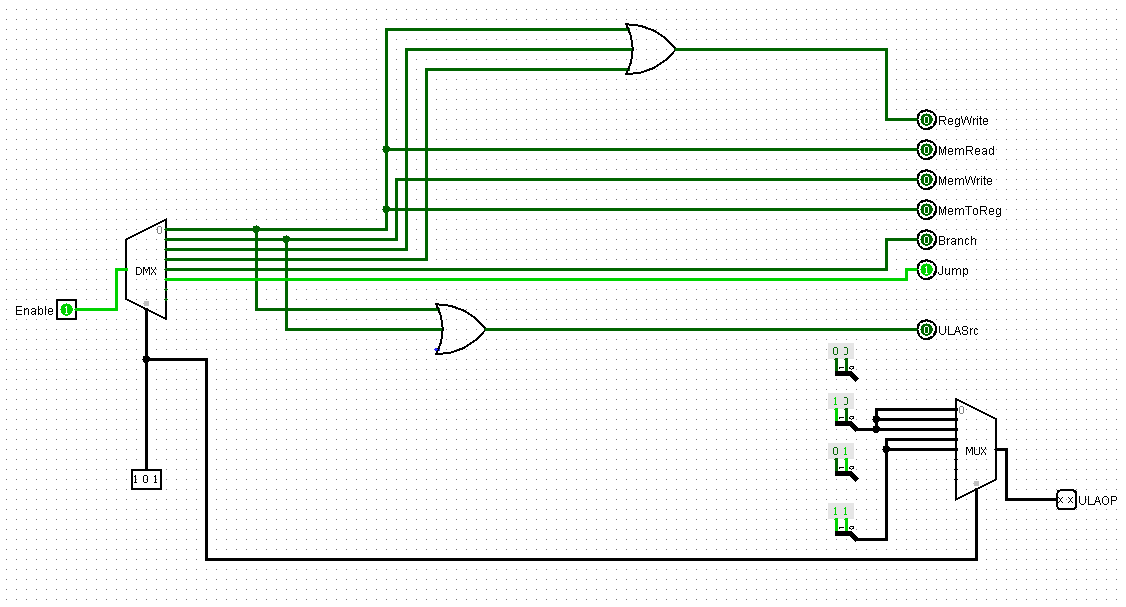
O componente Control tem como objetivo realizar o controle de todos os componentes do processador de acordo com o opcode ... Esse controle é feito através das flags de saída abaixo:

* **RegWrite**: 0,1.
* **MemRead**: 0,1.
* **MemWrite**: 0,1.
* **MemToReg**: 0,1.
* **Branch:** 0,1.
* **Jump**: 0,1.
* **ULASource:** 0,1.
* **ULAOP**: 00,01,10,11.

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.**

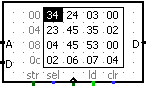
| **Comando** | RegWrite | MemRead | MemWrite | MemToReg | Branch | Jump | ULASrc | ULAOP |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| LW | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 10 |
| SW | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 10 |
| ADD | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 10 |
| SUB | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 11 |
| BEQ | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 11 |
| JUMP | 0 | 0 | 0 | 0 | 0 | 1 | 0 | XX |



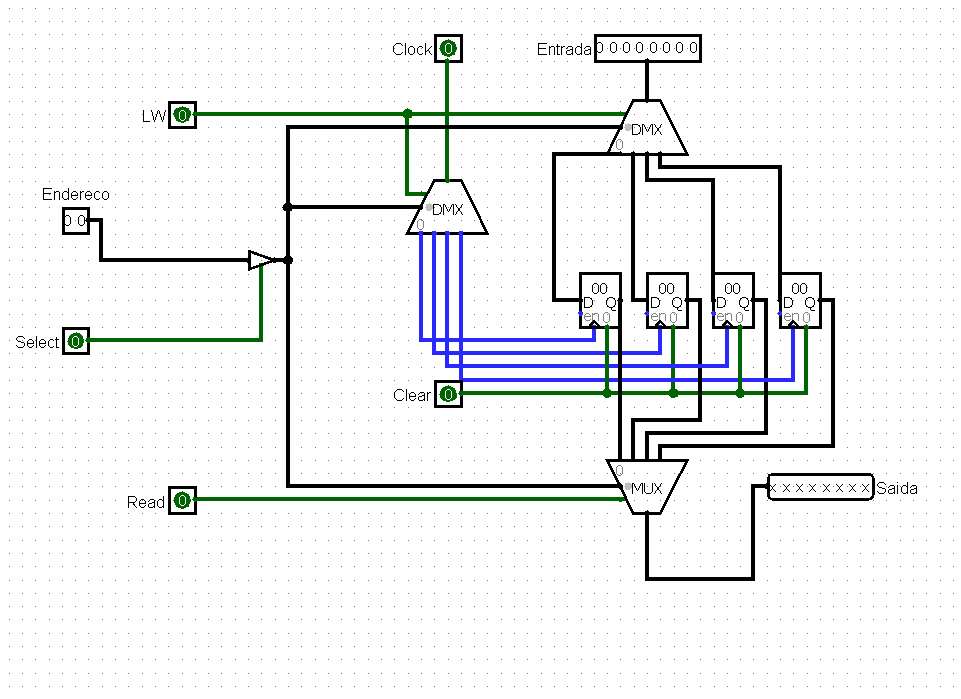
**Figura 5 - Unidade de controle**

### Memória de dados

O componente RAM, é o componente mais complexo nas bibliotecas predefinidas do Logisim. A memória RAM é uma memória de acesso rápido aleatório, que permite a leitura como a memória ROM e a escrita de arquivos. Diferentemente da memória de leitura, a RAM é volátil e tem suas informações perdidas com o desligamento da energia.’’

****

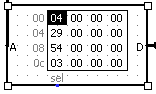
**Figura 6 - Memória de dados (Memória RAM, 8 bits de endereço)**

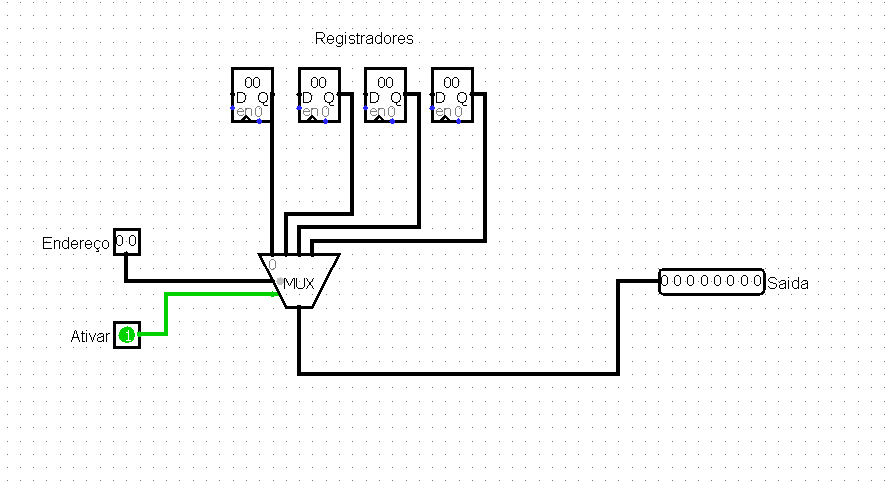
****

**Figura 7 - (Memória RAM com dois bits de endereço)**

### Memória de Instruções

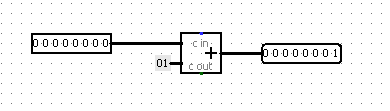
A memória de instruções, é o componente responsável por armazenar os passos e instruções relativas aos mesmos. Será a ROM que enviará para o divisor uma instrução associada ao passo recebido pelo PC. Este tipo de memória é não-volátil, ou seja, ela não perde os dados com o desligamento da energia.



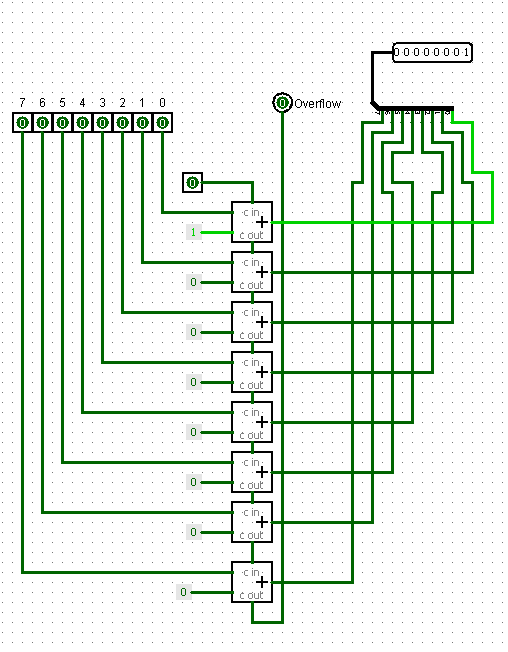
**Figura 8 - Memória de instruções (Memória ROM, 8 bits de endereço)  
  
Figura 9 - (Memória ROM com dois bits de endereço)**

### Somador

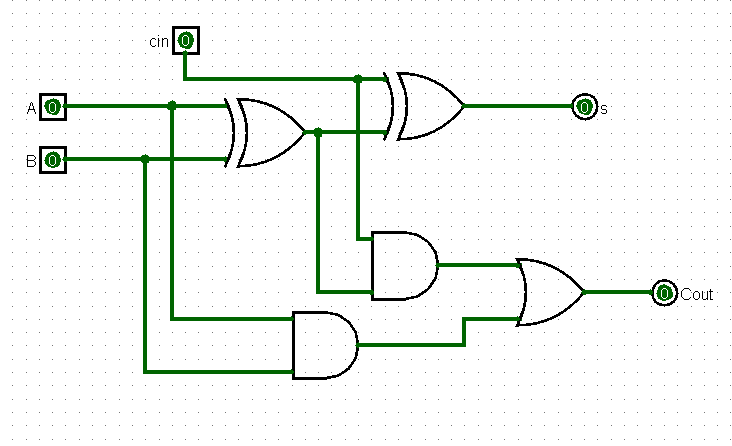
O circuito **somador completo** pode ser representado por três entradas, A, B e Carry de entrada, ou Carry In, que são somados e obtemos o resultado da soma, ou sinal S de saída, e Carry de Saída, ou Carry Out. A tabela verdade pode ser representada conforme a tabela a seguir para diferentes valores de entrada.

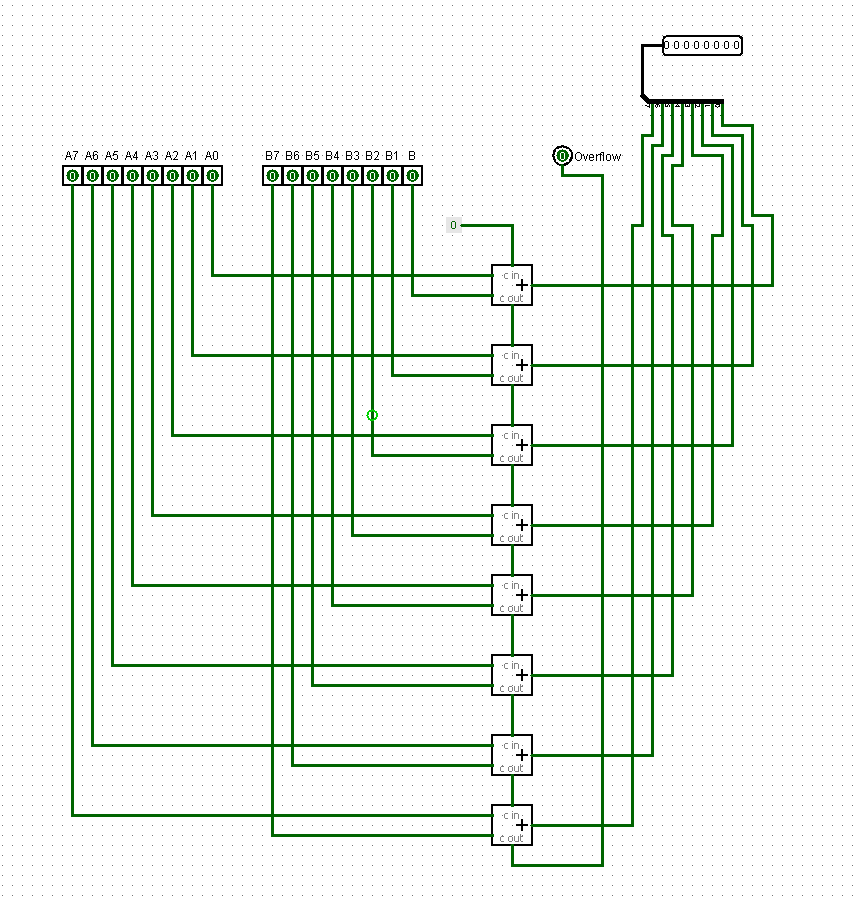


**Figura 10 - Somador + 1**

****

**Figura 11 - Representação por circuito do somador de 8 bits + 1**

****

**Figura 12 - Somador 1 bit  
**

**Figura 13 - somador de 8 bits**

### And

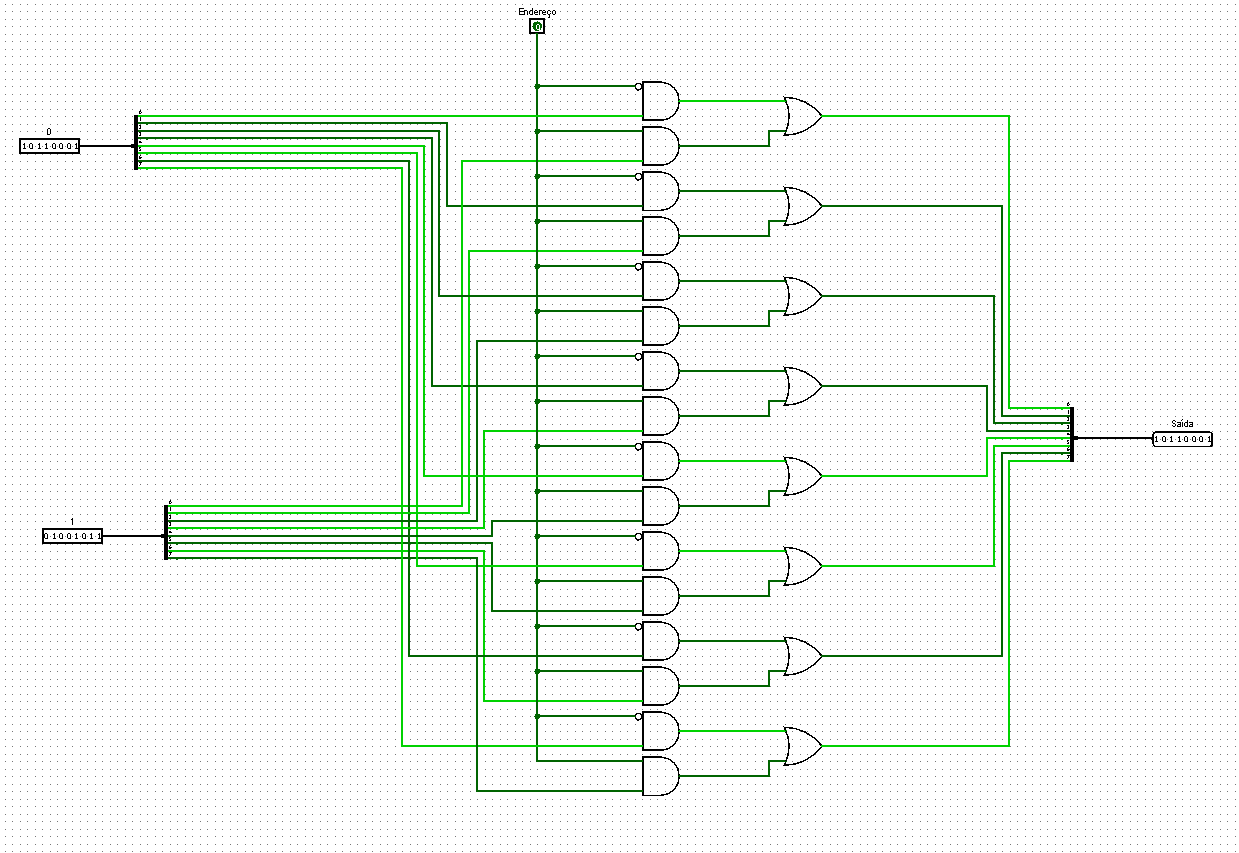
É uma porta lógica em dois operandos que resulta em um valor lógico verdadeiro somente se todos os operado tem um valor verdadeiro.



**Figura 14 - Representação do AND**

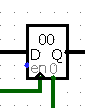
### Mux\_2x1

Os multiplexadores é um seletor utilizado para definir qual dos valores de entrada será o valor de saída, utilizando como endereço de seleção, flags da unidade de controle.



**Figura 15 - Representação do circuito do mux 2x1**

### PC

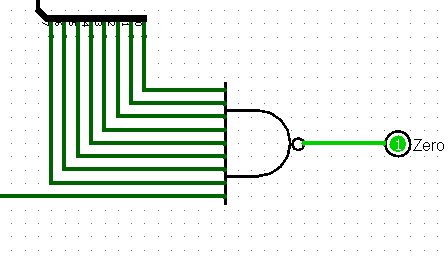
O PC é um registrador flip-flop tipo D que armazena o endereço da instrução que está sendo executada no momento.  


**Figura 16 - Registrador tipo D, representando um PC**

### ZERO

O Zero fica dentro da ULA, e é utilizado apenas no caso das operações comparativas. Sua função é apenas inicializar a flag necessária para realizar a comparação.

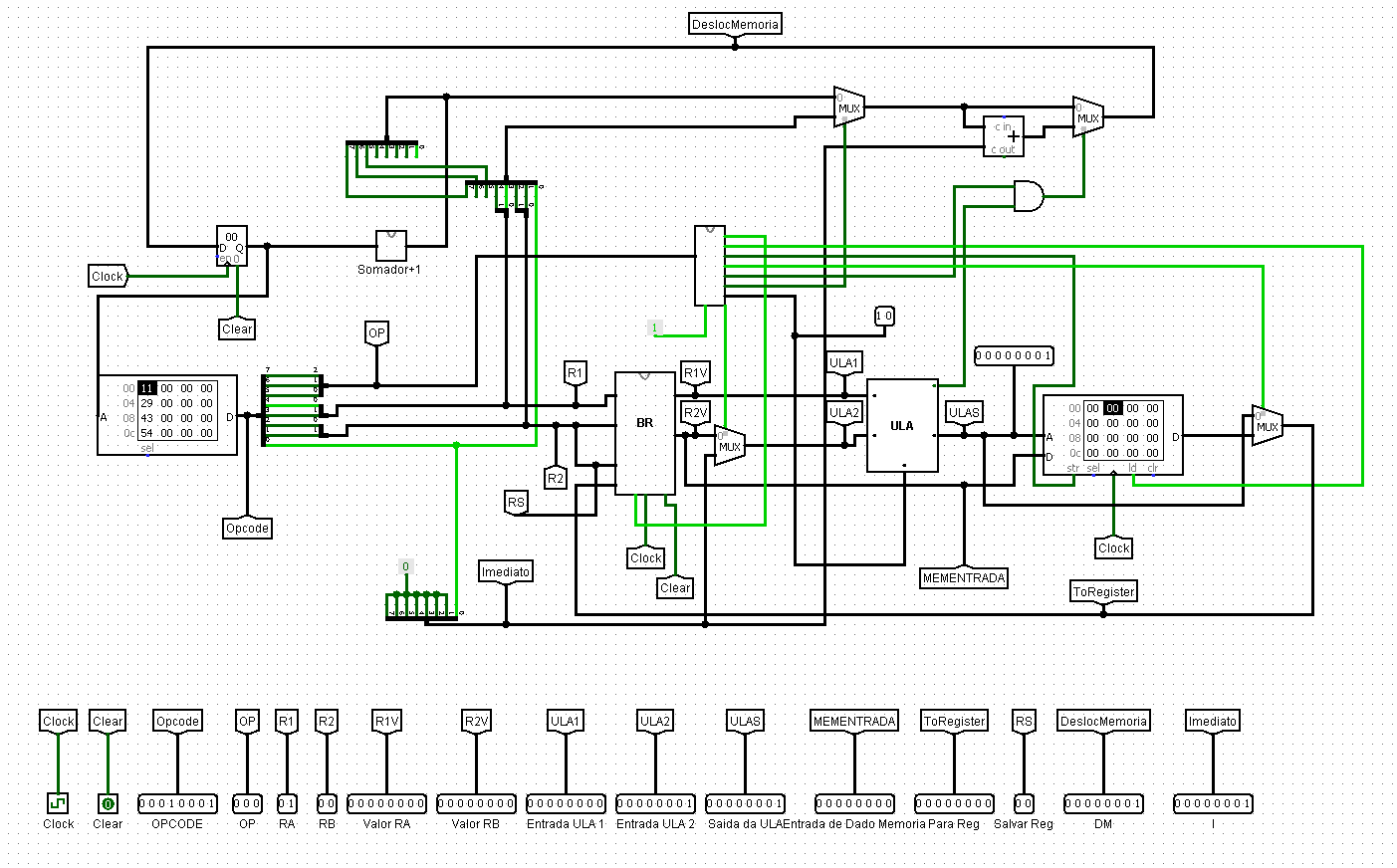
Ele recebe todos os valores de saída do cálculo e o overflow do subtrator, passando tudo por um AND negando as entradas, caso todos os valores forem 0, a saída Zero é positiva.



**Figura 17 - Representação do ZERO dentro da ULA**

## Datapath

É a conexão entre as unidades funcionais formando um único caminho de dados e adicionando uma unidade de controle responsável pelo gerenciamento das ações que serão realizadas para diferentes classes de instruções. O **Datapath** Representa um caminho para os dados em um repositório de armazenamento. O caminho representado pelo objeto **DataPath** pode apontar para um diretório ou um artefato de dados.

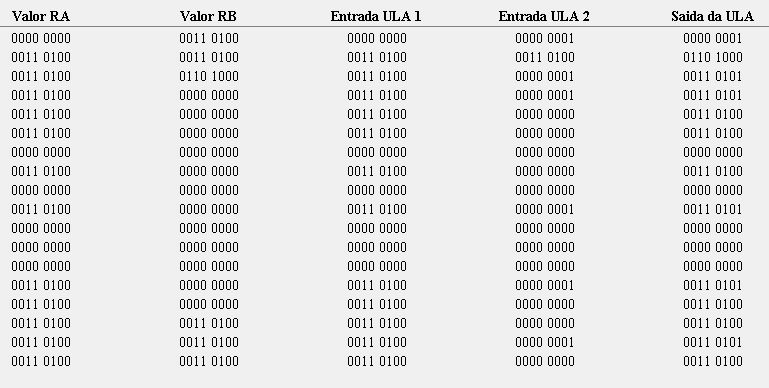
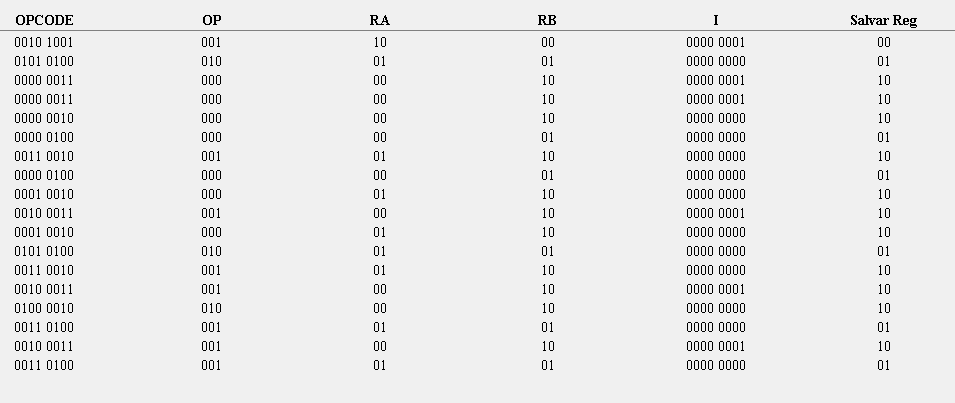


**Figura 18 - Representação do Datapath**

# Simulações e Testes

Objetivando analisar e verificar o funcionamento do processador, efetuamos alguns testes analisando cada componente do processador em específico, em seguida efetuamos testes de cada instrução que o processador implementa. Para demonstrar o funcionamento do processador **INTEL BAT.**

**Verificação dos resultados no relatório da simulação:** Após a compilação e execução da simulação, o seguinte relatório é exibido.

 **Figura 19 - Resultado na waveform.**

# Considerações finais

Este trabalho apresentou o projeto e implementação do processador de 8 bits denominado de **INTEL BAT**. Inicialmente o processador foi algo difícil de ser pensado , mas com o tempo as ideias de como executar foram surgindo depois de um longo esforço de pesquisas e testes. A atividade de Laboratório de Circuitos ajudou elaborada pelo professor Herbert Oliveira, foi de grande ajuda para entender como funcionam os componentes de um processador.

O processador possui uma limitação onde o Load e Store atua em uma instrução de 8 bits. O processador atua exatamente como um processador uniciclo deveria agir com suas respectivas instruções, com a capacidade de armazenar até 8 bits em seus registradores.

**4 Referências bibliográficas**

PATTERSON, D.; HENESSY, J. L. **Organização e projeto de computadores: a interface hardware/software.** 3ª Edição. São Paulo: Elsevier, 2005, 484 p