UFRR

PODER EXECUTIVO
MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE RORAIMA
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

ARQUITETURA E ORGANIZAÇÃO DE COMPUTADORES

RELATÓRIO DO PROJETO: PROCESSADOR

ALUNOS:

Ana Júlia Vieira Pereira Andrade da Costa – 2021013069
Shelly da Costa Leal - 2020001671

Dezembro de 2022
Boa Vista/Roraima

UFRR

PODER EXECUTIVO
MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE RORAIMA
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

ARQUITETURA E ORGANIZAÇÃO DE COMPUTADORES

RELATÓRIO DO PROJETO: PROCESSADOR AJ.SL

Dezembro de 2022
Boa Vista/Roraima

Resumo

Este trabalho aborda o projeto e implementação de um processador de 8 bits baseado na arquitetura mips monociclo. O presente projeto foi feito para a apresentação da disciplina de Arquitetura e Organização de Computadores visando obtenção de nota parcial do semestre. No processador haverá um total de 12 instruções e 4 registradores, seu nome é AJ.SL por conta de serem a abreviação de nosso nome.

1 Especificação

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

1.1 Plataforma de desenvolvimento

Para a implementação do processador AJ.SL foi utilizado a IDE: Programa Logisim 2.7.1 simulação digital de circuitos lógicos. O Logisim é uma ferramenta educacional para a criação e a simulação digital de circuitos lógicos. Com uma interface simples e com ferramentas para simular circuitos à medida em que são construídos, é simples o bastante para facilitar a aprendizagem dos conceitos mais básicos relacionados aos circuitos lógicos. Com a capacidade de construir circuitos maiores a partir de subcircuitos menores, traçar conexões com um mero arrastar do mouse, o Logisim pode ser usado (e é usado) para projetar e simular CPUs completas para fins educacionais.

1.2 Conjunto de instruções

O processador AJ.SL possui 4 registradores. Assim como 12 formatos de instruções de 8 bits cada, 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;

Tipo de Instruções:

- Formato do tipo R: Este formato aborda instruções de Load (exceto *load Immediately*), Store e instruções baseadas em operações aritméticas.

Instruções do tipo R		
Opcode	Reg1	Reg2
4 bits	2 bits	2 bits

-Formato do tipo I: Este formato tem instruções de load word e store word.

Instruções do tipo I			
Opcode	rs	rt	imediative
4 bits	1 bits	1 bits	2 bits

-Formato do tipo J: Instruções de tipo jump.

Instruções do tipo J	
Opcode	Endereço
4 bits	4 bits

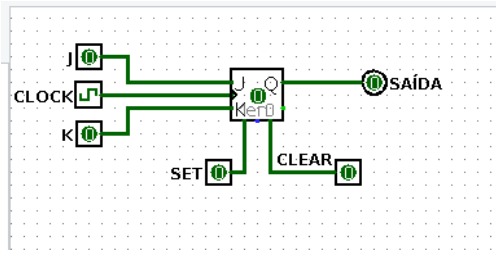
2. Componentes

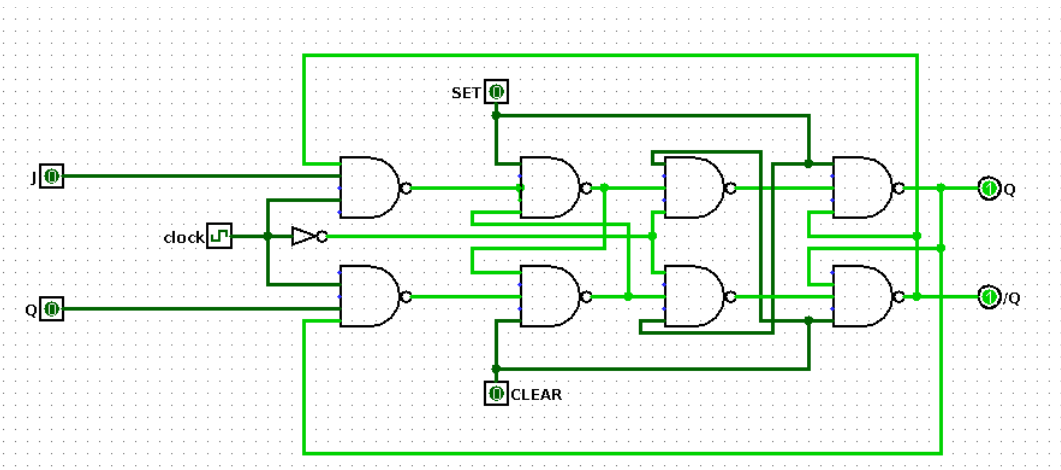
2.1. Flip Flop do Tipo D e JK

Flip-flops são basicamente circuitos digitais de memória. Cada flip-flop corresponde a uma memória de 1 bit. Em particular, haverá variação quando o ***clock***, varia de 0 para 1 (borda de subida). Existem diversos tipos de flip-flops, cada um deles com suas aplicações. A seguir veremos mais por dentro dos flip-flop´s de tipo JK e D.

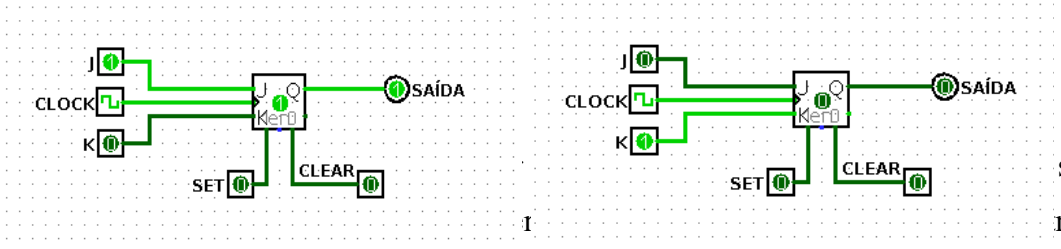
2.1.1. Flip Flop JK

O flip-flop de tipo JK, funciona como uma versão melhorada do tipo básico de flip-flop, o SR.





O Circuito do Flip-Flop JK, possui as entradas J e K, a negação de SET e a negação de CLEAR, as saídas são Q e Q negado, e o clock para alterar o valor dentro do Flip-Flop.

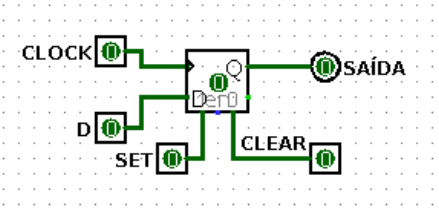


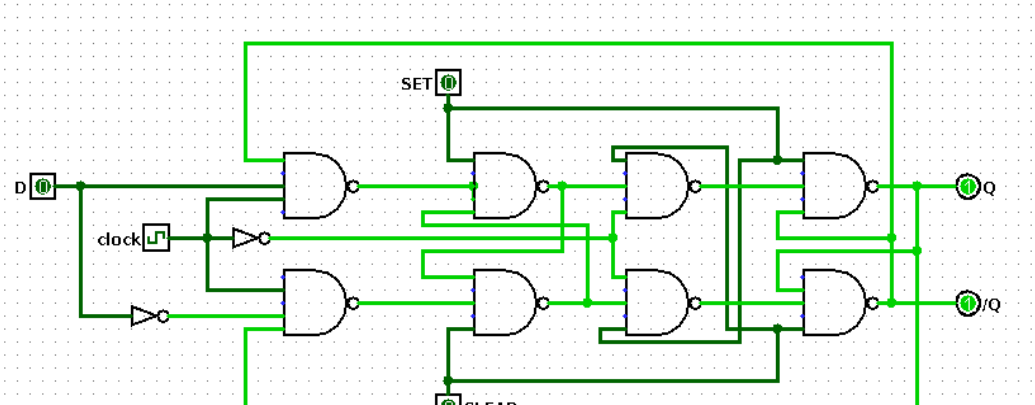
Quai se as entra radas forem diferentes, então o valor torna-se 1 se a entrada *J* for 1 e 0 se a entrada *K* for 1.

	CLK	J	K	$\overline{Preset}$	$\overline{Clear}$	Q_f
1	0	0	0	1	1	Q_a
2	0	0	1	1	1	0
3	0	1	0	1	1	1
4	0	1	1	1	1	$\overline{Q_a}$
5	X	X	X	0	1	1
6	X	X	X	1	0	0
7	X	X	X	0	0	

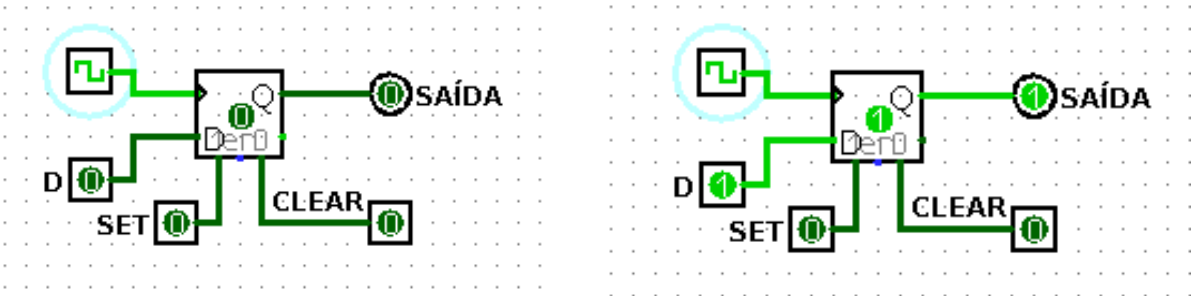
2.1.2. Flip Flop D

O Flip Flop D é baseado no Flip Flop JK, sendo mudado somente as entradas.





O circuito Flip Flop *D* possui, diferente do JK, uma entrada sendo D, uma entrada de *clock*, a negação de SET e a negação de CLEAR, as saídas são Q e Q negado.



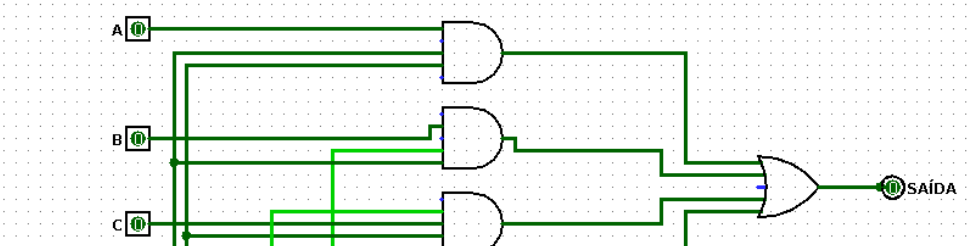
Nesse Flip Flop, o estado da saída de Q é igualado ao estado de D quando o clock varia entre 0 e 1.

TRUTH TABLE

INPUTS				OUTPUTS	
$\overline{PR}$	$\overline{CLR}$	CLK	D	Q	$\overline{Q}$
0	1	X	X	1	0
1	0	X	X	0	1
0	0	X	X	X	X
1	1	↑	1	1	0
1	1	↑	0	0	1
1	1	0	X	Q_0	$\overline{Q}_0$

2.2. Multiplexador de 4 entradas

O multiplexador é um dispositivo eletrônico que é utilizado para selecionar um sinal de entrada, para que esteja presente na saída. O multiplexador de 4 entradas é capaz de selecionar um dado entre os outros 3 que vai receber, por meio do endereço recebido.

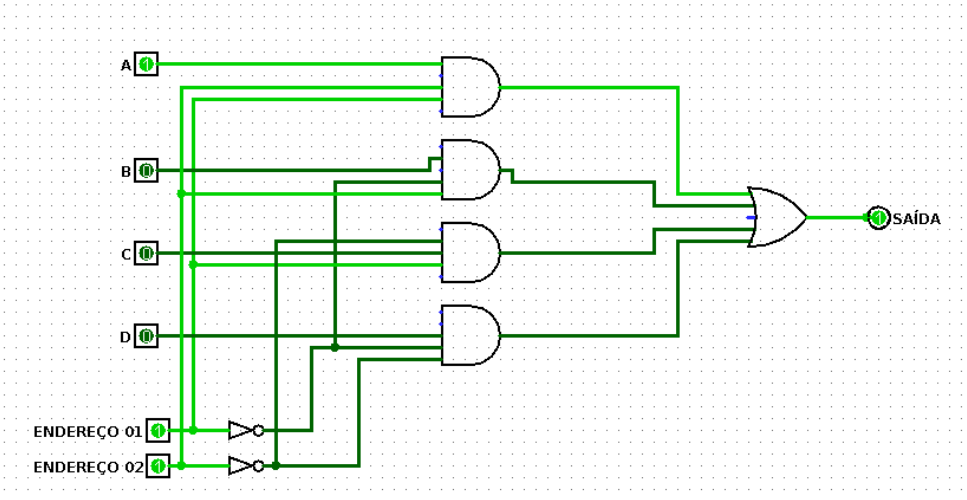


Nesse circuito estão presentes 4 entradas, sendo A ,B ,C e D. O endereço é formado pelas entradas endereço 01 e endereço 02, onde vai ser selecionado qual das 4 entradas vai passar para a saída.

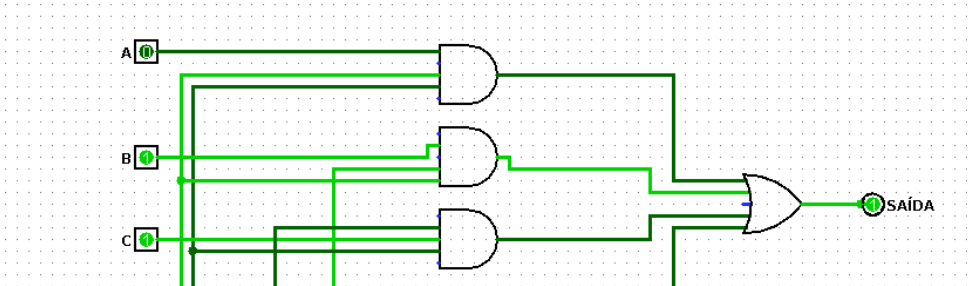
A	B	C	D	Endereço 01	Endereço 02	Saída
1	0	0	0	1	1	1
0	1	0	0	0	1	1
0	0	1	0	1	0	1
0	0	0	1	0	0	1

Saídas do multiplexador:

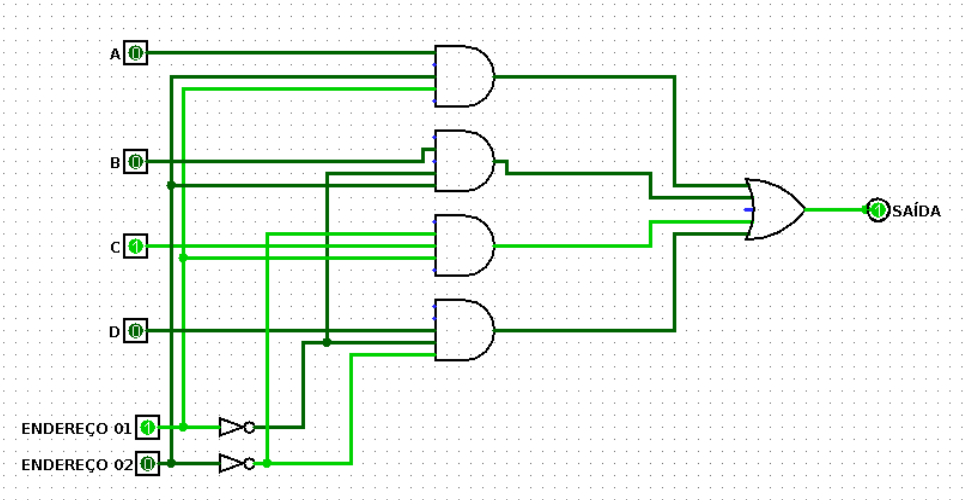
- Endereço 01 = 1 e Endereço 02 = 1, saída recebe A.



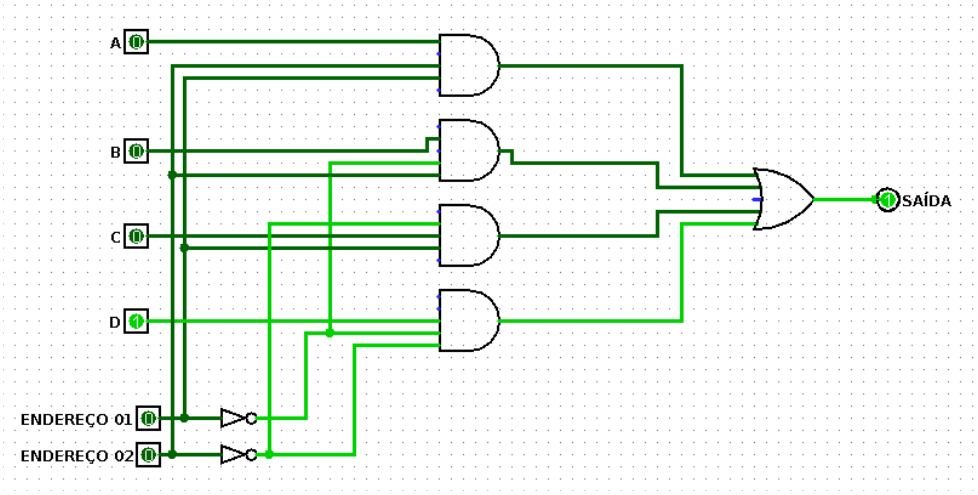
- Endereço 01 = 0 e Endereço 02 = 1, saída recebe B.



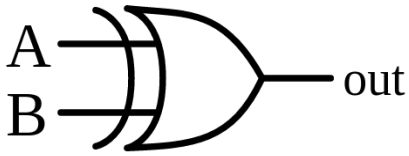
- Endereço 01 = 1 e Endereço 02 = 0, saída recebe C.



- Endereço 01 = 0 e Endereço 02 = 0, saída recebe D.



2.3. Porta lógica XOR usando port map com os componentes : AND, NOT e OR



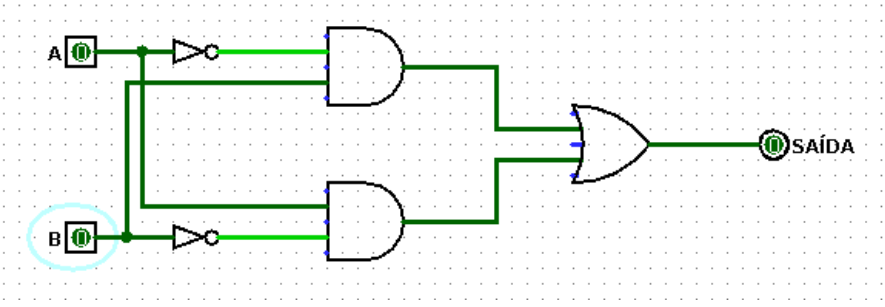
A porta lógica XOR, é uma **porta** de duas entradas que produz em sua saída o nível lógico 1 quando suas entradas tiverem valores diferentes entre si, e o nível lógico 0 quando as entradas forem iguais.

A	B	SAÍDA
0	0	0
0	1	1
1	0	1
1	1	0

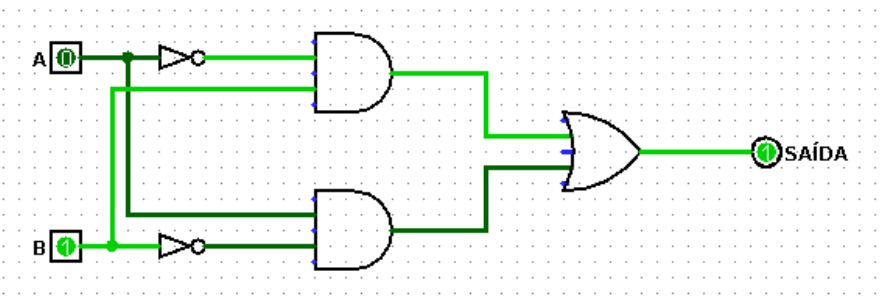
O circuito da porta lógica XOR possui 2 entradas e uma saída, com os componentes, NOT, AND e OR presentes.

Estados:

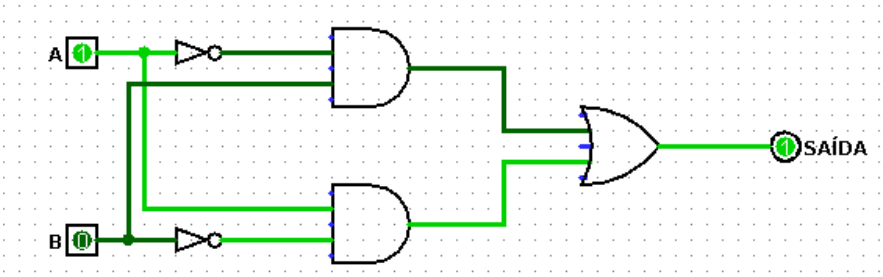
- A = 0 e B = 0, a saída é 0.



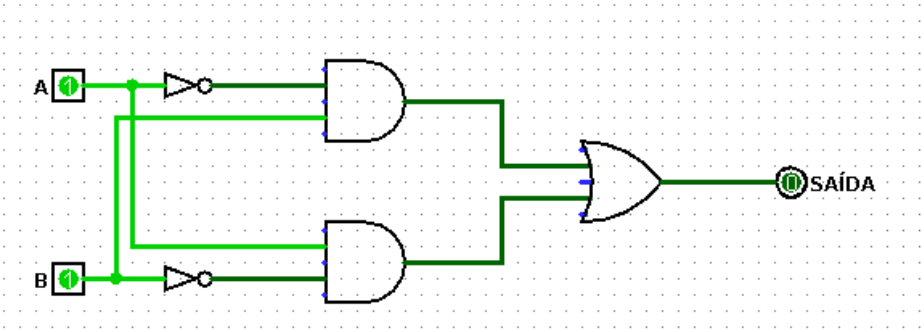
- A = 0 e B = 1, a saída é 1.



- A = 1 e B = 0, a saída é 1.



- A = 1 e B = 1, a saída é 0.

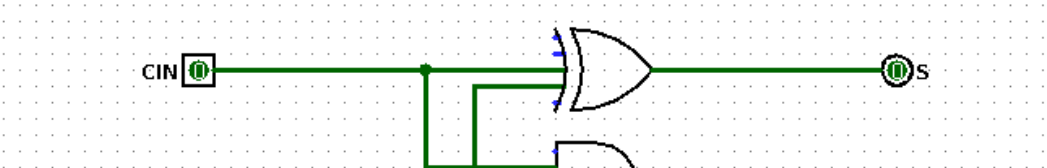


2.4. Somador de 8 bits

O somador de 8 bits é um componente de um circuito que consegue realizar equações de adição, este somador pode ser tanto feito por 8 somadores de 1-bit ou 2 somadores de 4 - bits, na criação deste somador foi realizado a construção de 1 somador de 1-bit, 1 somador de 4-bits e 1 somador de 8-bits. A seguir são apresentadas todas as etapas da construção até o componente final.

2.4.1. Somador de 1-bit

O somador de 1-bit é capaz de realizar soma por duas entradas de 1 bit, sendo as entradas A e B, esse somador pode receber um valor extra para a soma como CIN, as saídas são S (soma) e COUT.

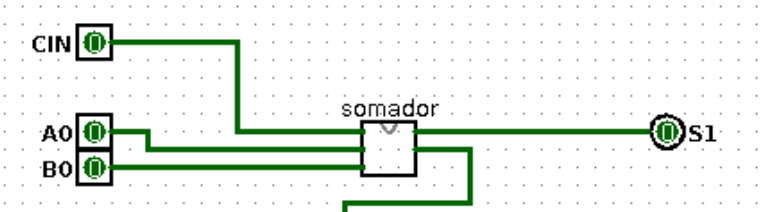


A soma é o resultado da operação de XOR entre A, B e CIN. COUT é o resultado da operação OR de três diferentes resultados: da operação de (A.B), da operação (A.Cin) e (B.Cin).

A	B	CIN	COUT	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

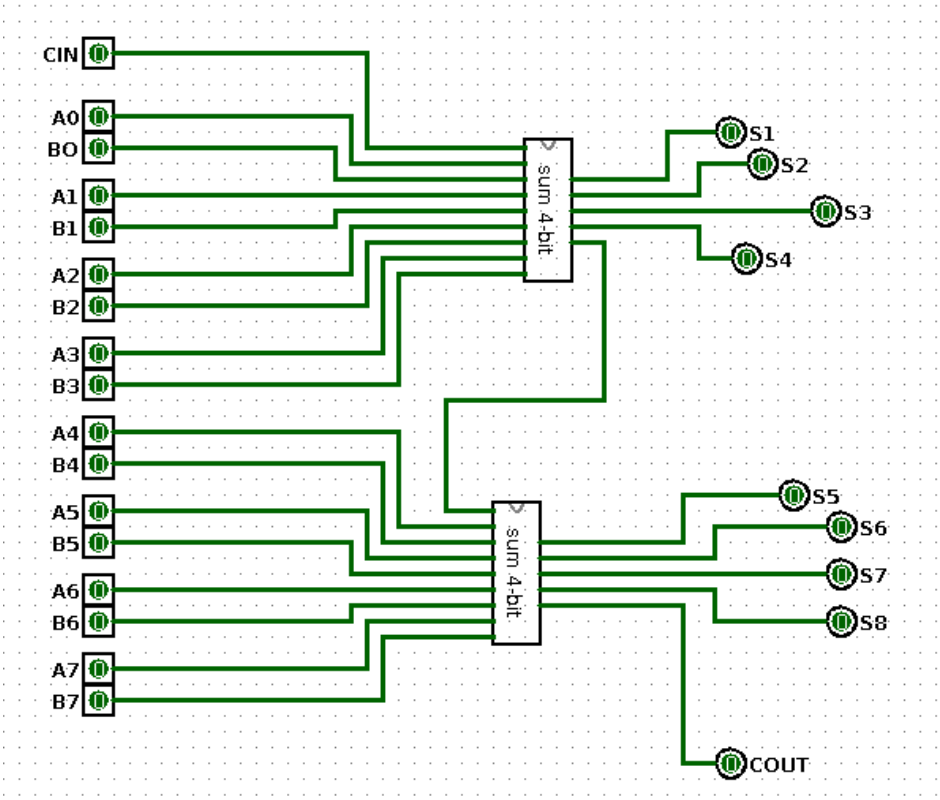
2.4.2. Somador de 4 - bit

O somador de 4 bits possui 4 somadores de 1-bit e 9 entradas e 5 saídas, sendo que as entradas são A0B0, A1B1, A2B2, A3B3 e o CIN. As saídas são 4 resultados de soma e o COUT. Sendo que o valor do COUT de um somador está conectado ao CIN do próximo somador.



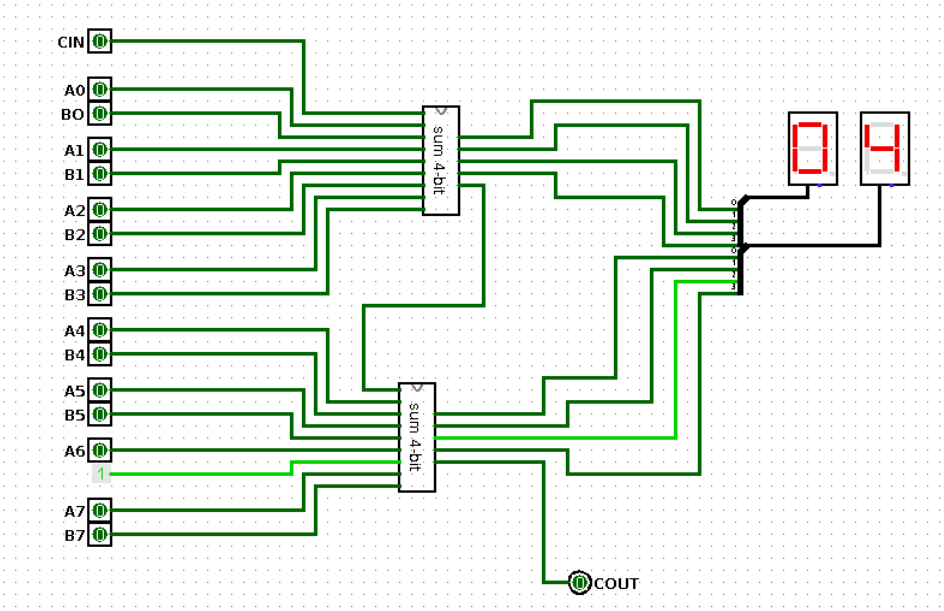
2.4.3. Circuito do somador de 8-bit completo

O somador de 8 bits é composto por 2 somadores de 4-bits, onde o CIN entra em um dos somadores e o COUT desse somador entra no CIN do outro somador, formando assim uma saída de 8 resultados de soma, que seria 8 bits estivessem juntas.



2.4.4. Somador de 8 bits que recebe um valor inteiro e soma com o valor 4.

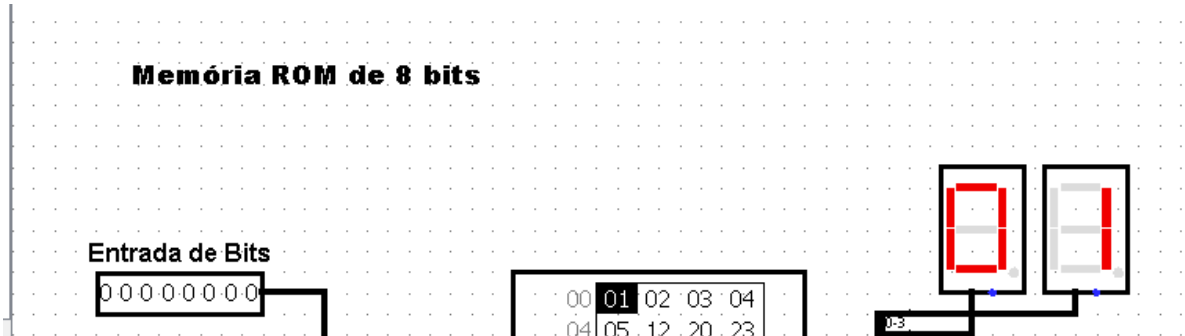
Modificação do circuito de 8 bits, contabilizando a soma de 8 bits no final. É utilizado uma constante no valor de 4, sendo no circuito como B6, para sempre seja somado valores com 4.



2.5. Memória ROM de 8 bits

A memória ROM é um dispositivo eletrônico, em que somente ocorre a leitura de dados, não sendo possível alterar ou apagar dados, somente acessá-los.

Os valores guardados na memória ROM são mostrados no componente. Dentro do componente cada valor é listado em hexadecimal.



No componente de Memória ROM com 8 bits, os bits de entrada vão ser setados pelo usuário dentro do campo de 8 bits distribuídos e o endereço digitado será localizado dentro da memória, exibindo nas placas de led o valor ali armazenado, utilizamos 5 componentes diferentes para demonstrar a funcionalidade do circuito.

Para este circuito, os componentes/pinos utilizados foram:

- 1x pino de entrada de dados: O pino utilizado tem suporte para 8 bits e receberá os dados do usuário para poder buscar o endereço solicitado.

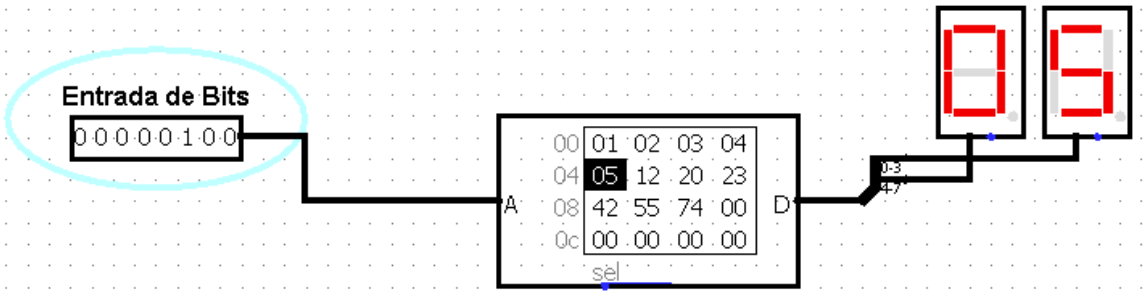
- 1x Memória ROM de 8 bits: A memória ROM vai armazenar diferentes valores para cada endereço de que ela pode armazenar (256 totais), a para cada endereço chamado pela entrada de dados, o valor guardado passará para o distribuidor.

- 1x Distribuidor: O distribuidor de dados de 2 bits vai enviar os dados da memória pelos seus caminhos por onde está ligado.

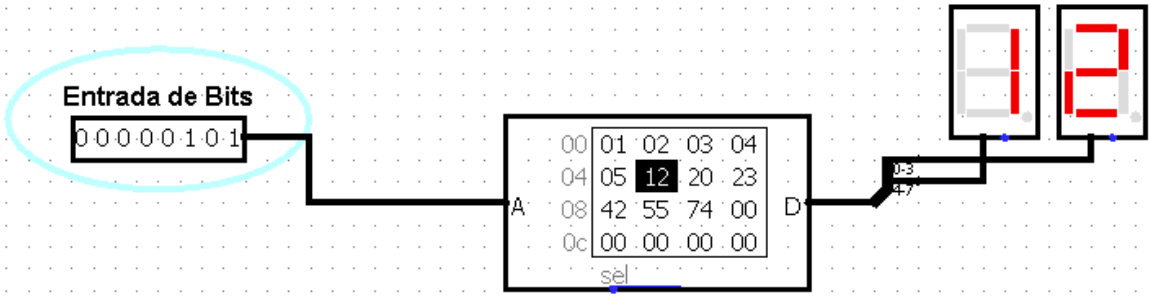
- 2x Displays Hexadecimais: Os displays irão funcionar como um telão de led, onde ao receber os dados do distribuidor, mostrarão o valor que está ali guardado naquele endereço de memória da qual o usuário fez a chamada.

Os testes funcionaram com exito, onde para cada endereço de 8 bits da memória, são apresentados nos displays hexadecimais os valores corretos com que estão guardados, contudo, apenas os 32 primeiros endereços da memória tiverem seus valores registrados e guardados para fins de testes, contudo, por ser um circuito de memória ROM de 8 bits, possui ao todo 256 endereços diferentes para serem trabalhados.

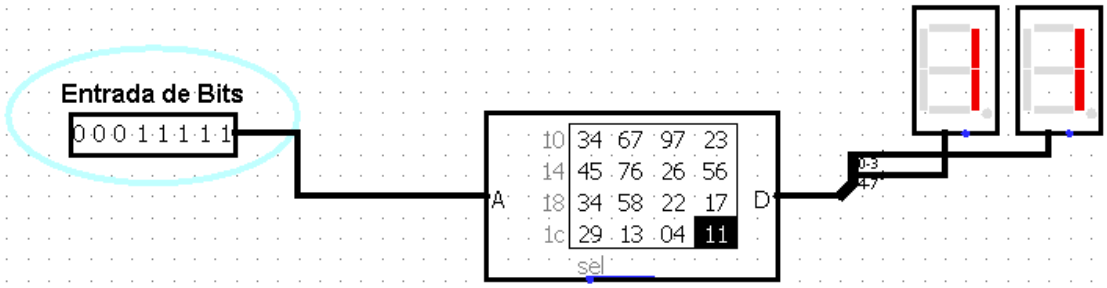
Memória ROM de 8 bits



Memória ROM de 8 bits

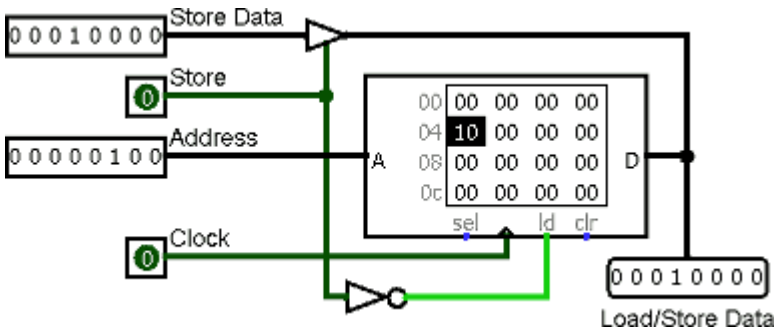


Memória ROM de 8 bits



2.6. Memória RAM de 8 bits

A memória RAM guarda temporariamente toda a informação que o computador precisa, seja para aquele momento ou para um futuro próximo.



O componente memória RAM possui :

- Uma porta para leitura/escrita síncrona (padrão);

O componente possui uma porta que servirá tanto para ler quanto para gravar dados. Qual a ação a ser executada dependerá da entrada marcada por *ld*: 1 (ou flutuante) indicará a leitura de dados do endereço informado pela face oeste, e 0 indicará a escrita dos dados entregues na porta. Para transmitir dados para dentro e para fora do componente, precisaremos usar um componente do tipo *Buffer* Controlado, como ilustrado no circuito.

- Uma porta para leitura/escrita assíncrona;

O mesmo descrito acima, exceto que não haverá participação do *clock*. O valor encontrado no barramento de dados será carregado na memória sempre que a entrada *ld* estiver em 0. Se, enquanto *ld* estiver em 0, o endereço ou os dados mudarem, então uma carga adicional ocorrerá. Essa opção é a que mais se aproxima de muitas das memórias de acesso aleatório comumente disponíveis.

- Portas separadas para leitura e escrita;

Duas portas estão disponíveis - uma na face oeste para a escrita de dados, e outra na face leste para leitura de dados. Essa opção retira a necessidade de se lidar com o *Buffer* Controlado e então ficará mais fácil de usar.

Seus pinos são:

A (entrada, com largura em bits de acordo com o atributo Bits de Dados): Serve para selecionar quais os valores estão sendo acessados atualmente pelo circuito.

D (entrada/saída, com largura em bits de acordo com o atributo Bits de Dados): Se essa entrada estará presente somente se "portas separadas para leitura e escrita" tiverem sido selecionadas pelo atributo Interface de Dados. Quando uma escrita for requisitada (via mudança do *clock* de 0 para 1 enquanto *sel* e *str* forem ambos iguais a 1 ou flutuante), o valor encontrado nessa porta será escrito na memória no endereço atualmente selecionado.

Store (entrada, com largura de 1 bit): essa entrada estará presente somente se "portas separadas para leitura e escrita" tiverem sido selecionadas pelo atributo Interface de Dados. Se for 1 ou flutuante, um pulso de *clock* resultará na escrita dos dados encontrados na face oeste na memória (se a entrada *sel* também for 1 ou flutuante).

Select (entrada, com largura de 1 bit): essa entrada habilita ou desabilita o módulo RAM por inteiro, caso o valor seja igual a 1, flutuante ou 0. Essa entrada destina-se primariamente para as situações em que houver múltiplas RAMs, mas somente uma delas deve estar habilitada em certo instante.

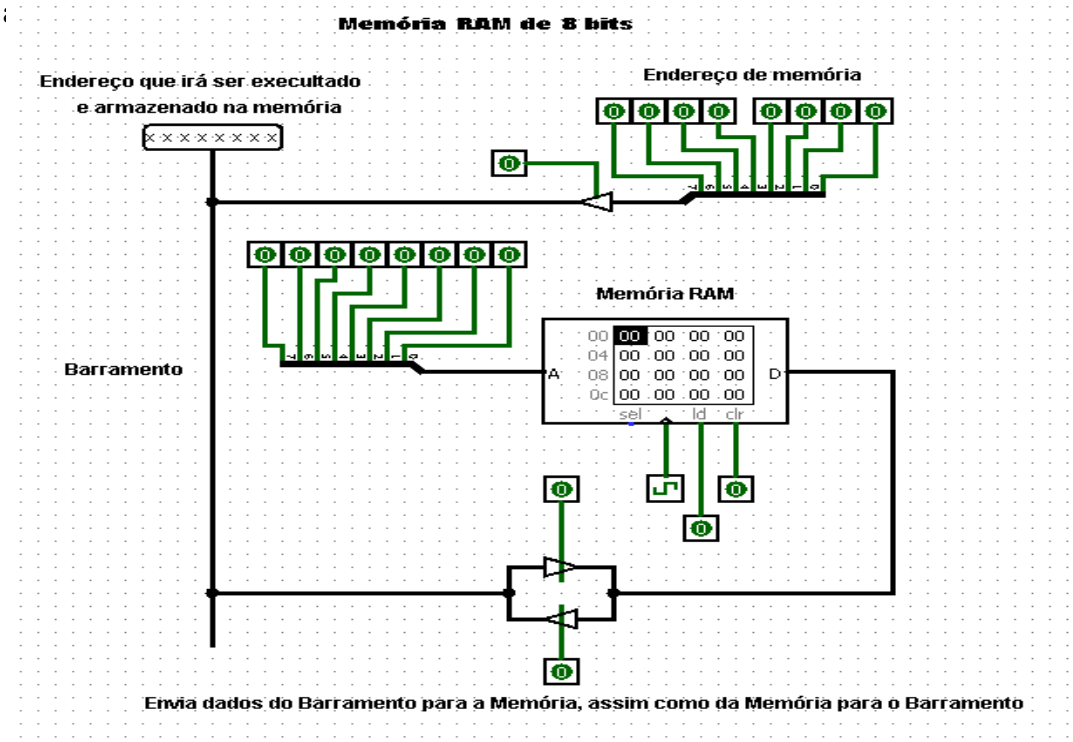
Clock (entrada, com largura de 1 bit): essa entrada estará ausente quanto o atributo Interface de Dados for "uma porta assíncrona para leitura/escrita". Em outras circunstâncias, quando *ld* for igual a 0, e essa entrada variar de 0 para 1 (e *sel* for igual a 1, indefinido e *clr* for 0), então o valor no endereço atualmente selecionado será alterado para o valor presente no pino *D*. Enquanto a entrada de *clock* permanecer em 0 ou 1, no entanto, o valor em *D* não será escrito na memória.

Load (entrada, com largura de 1 bit): serve para selecionar se a RAM deverá emitir (em *D*) o valor no endereço atual (*A*). O comportamento dessa saída estará habilitado se *out* estiver em 1 ou indefinido; se *out* for 0, então nenhum valor será colocado em *D* - mas se houver uma porta de leitura/escrita combinada, a escrita estará habilitada.

Clear (entrada, com largura de 1 bit): quando for igual a 1, todos os valores na memória ficarão iguais a 0, independente do que estiver nas outras portas.

Na elaboração do circuito para memória RAM de 8 bits, dividimos o circuito em 4 partes para seu funcionamento dentro da seguinte proposta, poder buscar e localizar endereços dentro da memória, podendo também alterar o valor pertencente a esses endereços se,

somente se, os Buffers presentes no circuito forem abertos para a memória RAM, liberando



Para esse circuito foram usados 29 pinos/componentes, que estão dispostos da seguinte maneira.

- 22x pinos de entrada de dados: Para estes casos, os pinos possuem duas finalidades para o circuito, a primeira será a de receber os endereços desejados para realizar a busca dentro da memória, a segundo será para permitir a passagem desses endereços percorrem o circuito (conexões com os buffers).

- 1x pino de saída de dados: Este pino setado com 8 bits será responsável por ler e interpretar o endereço solicitado, encontrando qual o valor nele está guardado e que alterações serão feitos (caso solicitadas), passando assim pelo barramento do circuito carregando essas informações diretamente para a memória.

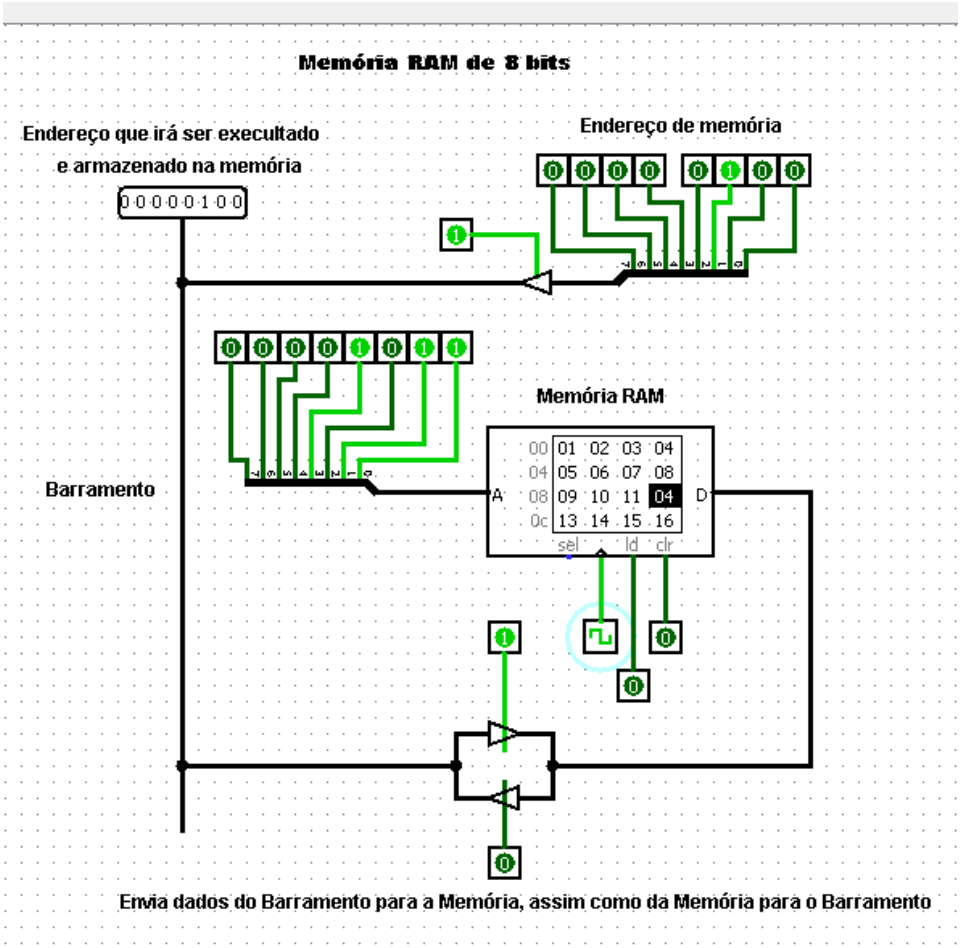
- 2x distribuidores: Neste caso, os distribuidores são atarefados com o propósito de buscar e localizar endereços na memória, sejam para verificar o valor ali armazenado ou alterar o valor pertencente, a depender claro do que está sendo realizado pelo circuito.

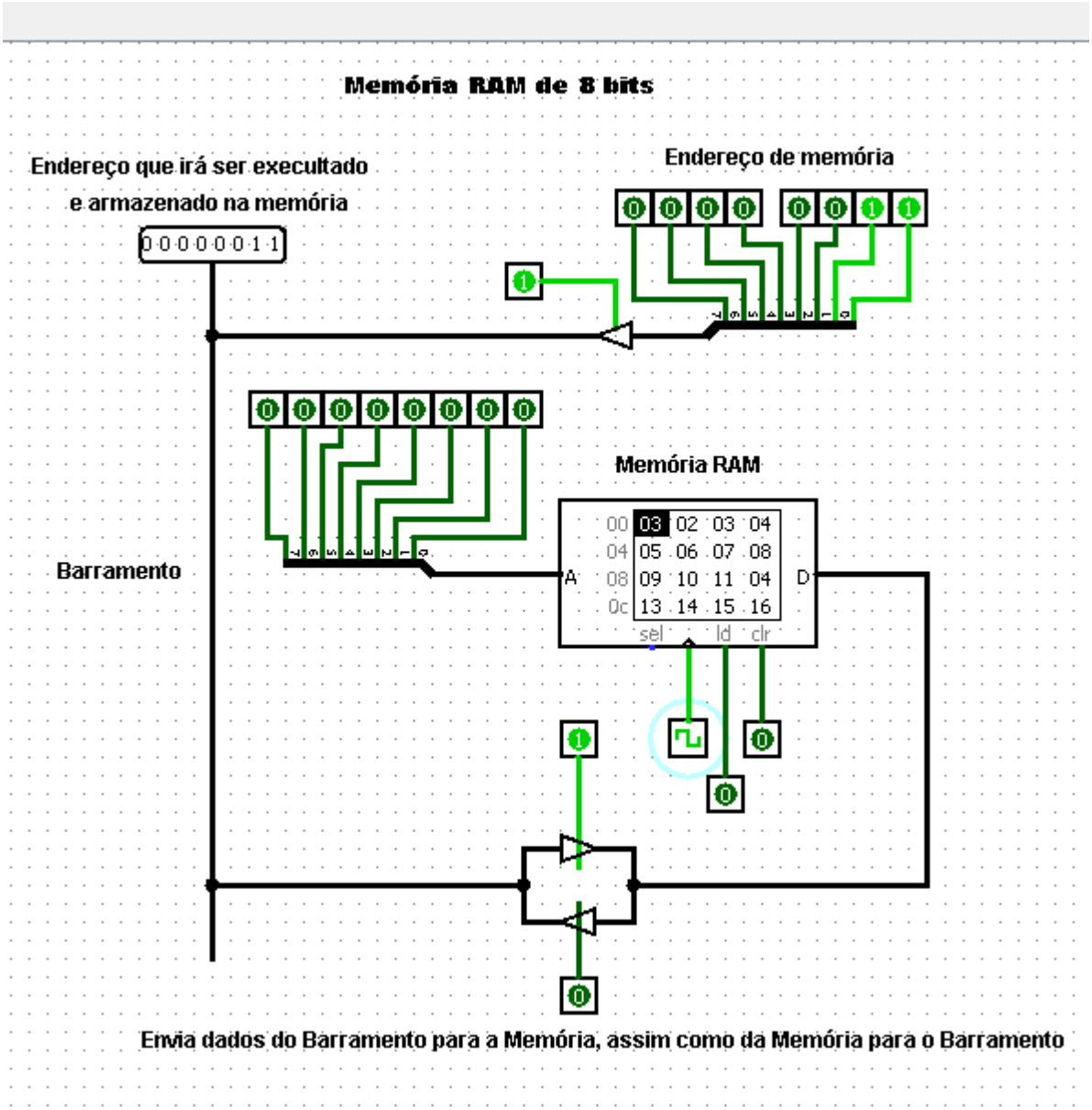
- 3x buffers: Os buffers para o circuito são encarregados de encaminhar os dados informados nas entradas de endereço, apenas se os pinos a eles conectados estiverem em valor lógico 1 durante o ciclo de clock, autorizando assim o fluxo de dados no circuito, caso o valor lógico dos pinos a eles ligados estejam em 0, mesmo durante um impulso de clock,

os dados não passarão pelo barramento do circuito, assim sendo, não serão passados os dados solicitados para buscar endereços ou alterar os valores conforme a chamada.

- 1x memória RAM: Por último, o principal componente do circuito, a memória RAM de 8 bits capaz de armazenar/guardar até 256 valores/dados diferentes no seu interior, permitindo que para cada um dos seus 256 espaços de locação, exista um endereço de memória único e distinto dos demais, de modo que cada um destes endereços pode ser chamado pela sua entrada A (entrada de dados da memória para localizar e buscar seus endereços) ou chamado pela sua entrada D (que lê e escreve valores e/ou dados a serem guardados para o endereço que receber a chamada do usuário antes de passar pelo barramento).

O endereço do distribuidor passa de (0000 0000) para (0000 1011), que representa o espaço 12 dentro da memória, enquanto que o endereço de memória chamado e passado ao barramento foi (0000 0100), na qual guarda o valor 04 dentro da memoria RAM, quando este valor foi lido e jogado para a memória após um impulso de clock, perceba que o valor 04 (pertencente ao endereço chamada 0000 0100) foi guardado na posição 12 (posição dentro da memória registrada com o endereço 0000 1011).



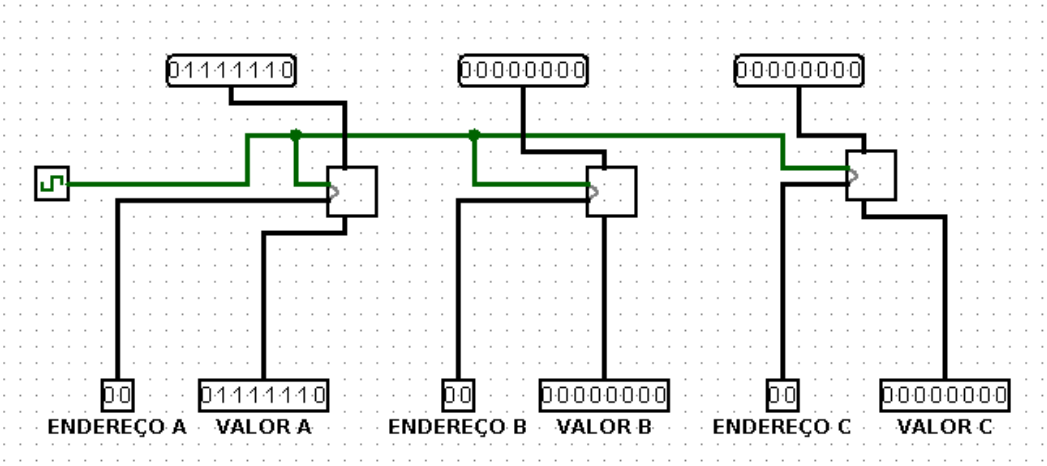


Foi setado um endereço de memória (0000 0011), onde o mesmo foi impulsionado para o barramento do circuito, lido e depois impulsionado para dentro da memória, onde o endereço foi localizado, o valor foi reconhecido (03) e foi guardado no primeiro espaço da memória, pois o distribuidor no qual está ligado mostra o primeiro endereço da memória (0000 0000).

2.7. Banco de Registradores de 8 bits

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

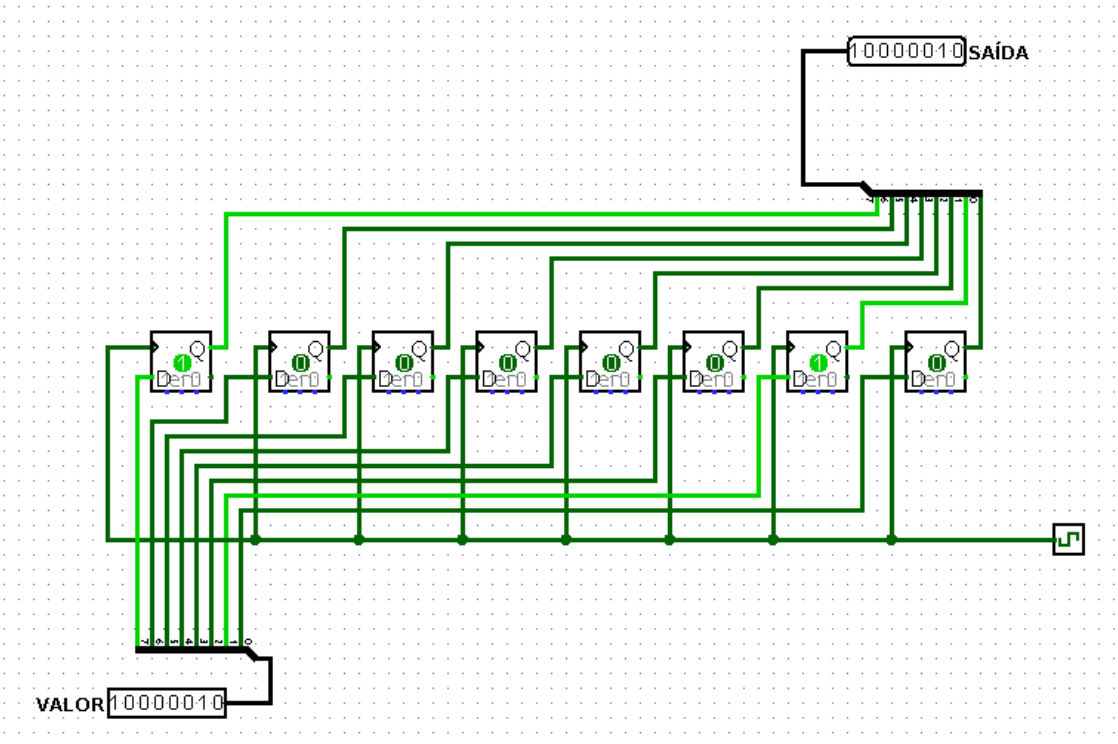
ser executadas operações de leitura dos dados anteriormente gravados e de escrita de dados para modificar as informações internas.



O banco de registradores é composto por 3 Registradores com 4 espaços de memória, dispositivo de entrada, chamada de Endereço, clock para alterar os dados do registrador e a entrada que diz se o valor do registrador será alterado. Para a realização deste circuito é preciso a construção de um registrador de 8-bits e um registrador com 4 espaços de memória.

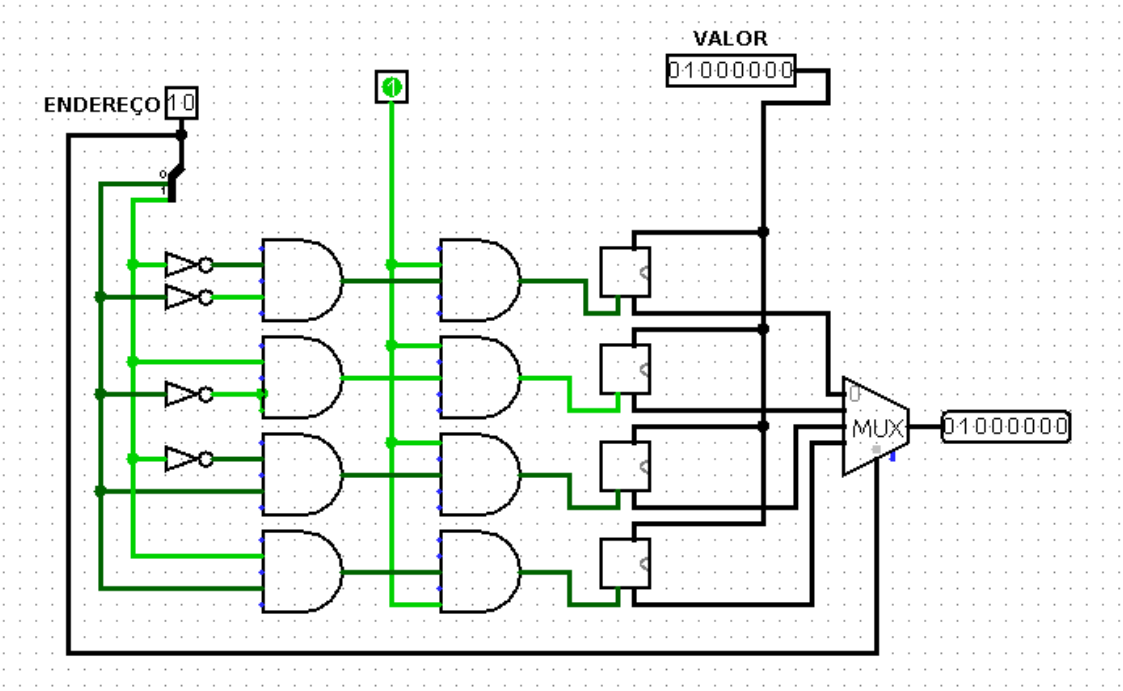
2.7.1. Registrador 8-bits

Nesse circuito foram utilizados 8 registradores de 1-bit com a estrutura de storage e load, onde em toda subida de clock para 1, o valor é salvo no espaço de memória.



2.7.2. Registrador com 4 espaços de memória

Para a construção de um registrador de 4 espaços de memória, foi usado o componente criado no circuito anterior. Para esse circuito a cada endereço pode ser atribuído um valor.



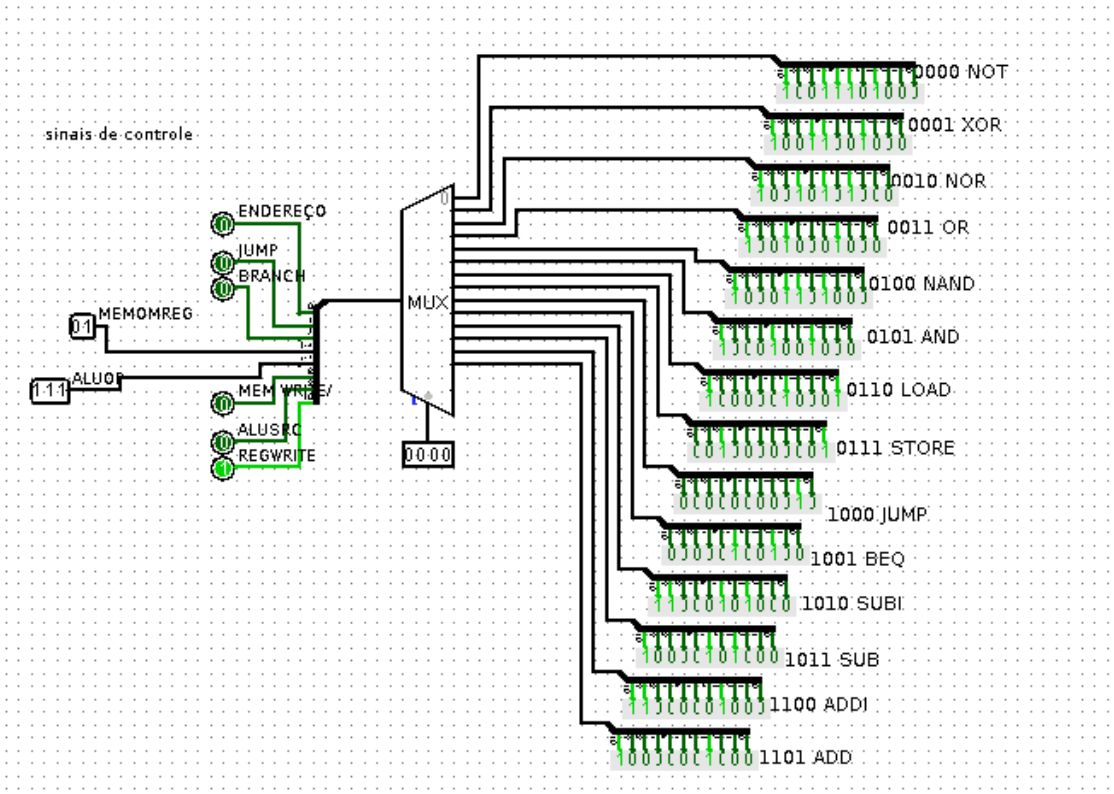
2.8. Unidade de Controle Uniciclo do MIPS de 16 bits

Unidade de controle é um componente de um processador uniciclo que recebe sinal de equivalente a instrução que será executada e através dessa informação, ativa os componentes que serão utilizados nessa execução.

Control signals									
Instruction	Reg Dst	ALU Src	Memto Reg	Reg Write	MemRead	Mem Write	Branch	ALUOp	Jump
R-type	1	0	0	1	0	0	0	00	0
LW	0	1	1	1	1	0	0	11	0
SW	0	1	0	0	0	1	0	11	0
addi	0	1	0	1	0	0	0	11	0
beq	0	0	0	0	0	0	1	01	0
j	0	0	0	0	0	0	0	00	1
jal	2	0	2	1	0	0	0	00	1
slti	0	1	0	1	0	0	0	10	0

(Tabela de Sinais da Unidade de Controle do MIPS 16 bits)

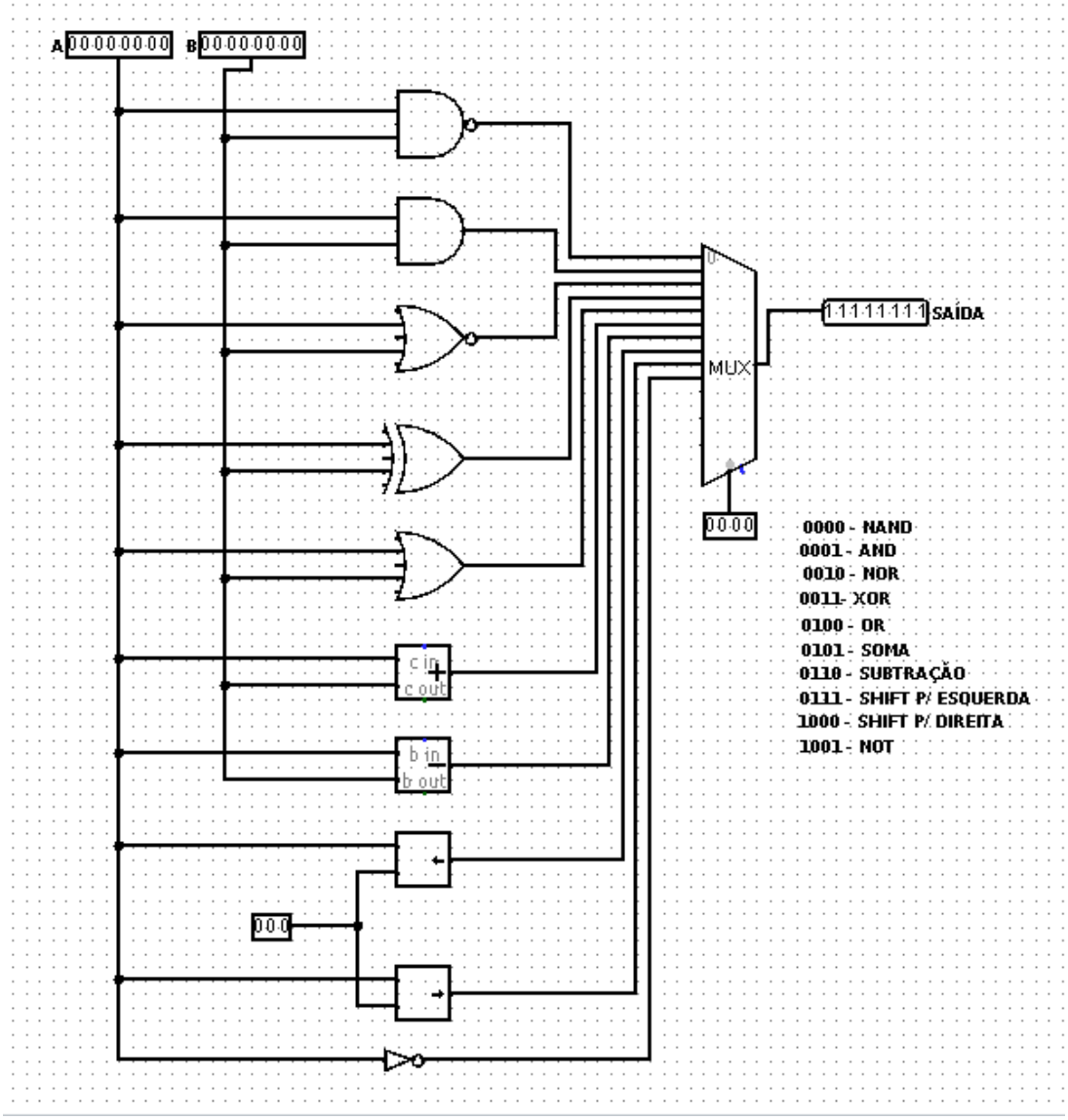
Esse é o circuito da Unidade de Controle (UC), construída a partir da tabela de sinais apresentada acima, a mesma possui os valores constantes relacionados ao OPCode, um seletor e as possíveis saídas que são acionadas com o código selecionado.



2.9. ULA de 8 bits

Uma Unidade Lógica e Aritmética (ULA) é um dispositivo que realiza operações lógicas e aritméticas sobre números representados em circuitos lógicos. Tipicamente, uma ULA recebe dois operandos como entrada, e uma entrada auxiliar de controle permite especificar qual operação deverá ser realizada.

Por esse motivo, a construção de uma ULA se baseia em dois fundamentos principais: o controle de fluxo de dados e a construção de circuitos que implementam operações.



No circuito acima é apresentado uma ULA de 8 bits, que possui as entradas A e B, a saída, um seletor para as equações e para o shift. A ULA apresenta os componentes AND, OR, NOT, NOR, NAND, XOR, SHIFT de 2 bits à esquerda, SHIFT de bits à direita, soma e subtração. No circuito a escolha do tipo de equação ocorre pelo seletor.

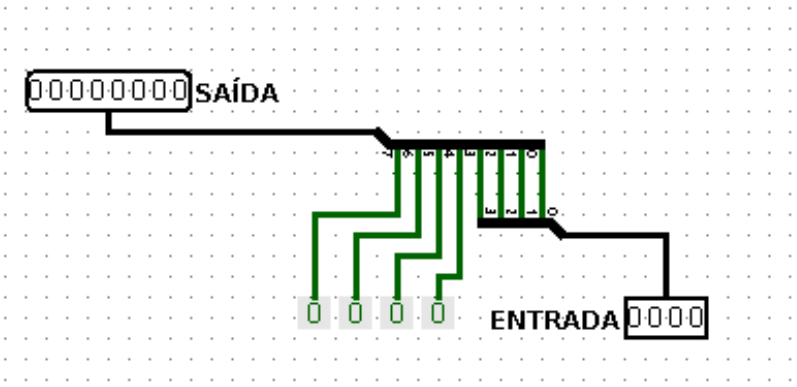
Saídas:

- Ao digitar 0000 no seletor, vai ser acionado a porta lógica NAND.
- Ao digitar 0001 no seletor, vai ser acionado a porta lógica AND.
- Ao digitar 0010 no seletor, vai ser acionado a porta lógica NOR.
- Ao digitar 0011 no seletor, vai ser acionado a porta lógica XOR.

- Ao digitar 0100 no seletor, vai ser acionado a porta lógica OR.
- Ao digitar 0101 no seletor, vai ser acionado a porta lógica SOMA.
- Ao digitar 0110 no seletor, vai ser acionado a porta lógica SUBTRAÇÃO.
- Ao digitar 0111 no seletor, vai ser acionado a porta lógica SHIFT P/ESQUERDA.
- Ao digitar 1000 no seletor, vai ser acionado a porta lógica SHIFT P/DIREITA.
- Ao digitar 1001 no seletor, vai ser acionado a porta lógica NOT.

2.10. Extensor de sinal de 4 bits para 8 bits.

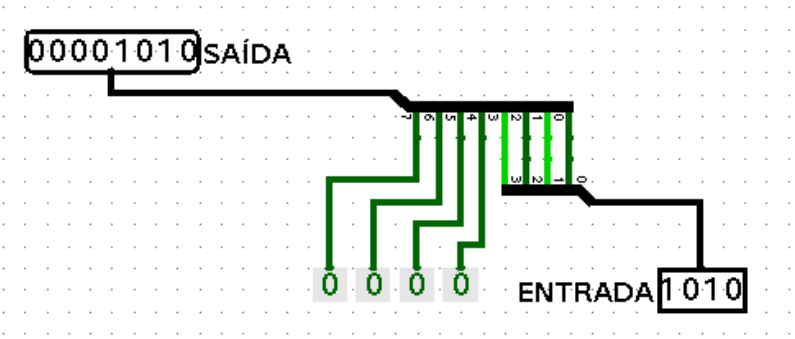
O extensor de sinal transformará um valor em outro de largura diferente. Se for para transformar para uma largura menor, os bits de mais baixa ordem serão simplesmente truncados. Se for para transformar para uma largura maior, os bits menos significativos serão os mesmos, e você terá uma escolha para os bits de mais alta ordem: eles poderão ser todos iguais a 0, ou todos iguais a 1, o concordarem com a entrada do bit de sinal (o mais significativo), ou ainda ter esse valor determinado por uma entrada adicional.



No circuito acima é feito um extensor de sinal de 4 bits, esse circuito possui uma entrada de 4 bits, uma saída de 8 bits, 2 distribuidores sendo um de 8 bits e outro de 4 bits, e 4 valor de constantes como 0 (poderia ser 1 também). Quando ocorre a entrada de 4 bits no distribuidor de 4 bits, é passado esses para os bits de mais baixa ordem do distribuidor de 8 bits e os de mais alta ordem recebem 0 para que ocorra a saída do circuito como 8 bits.

Saída:

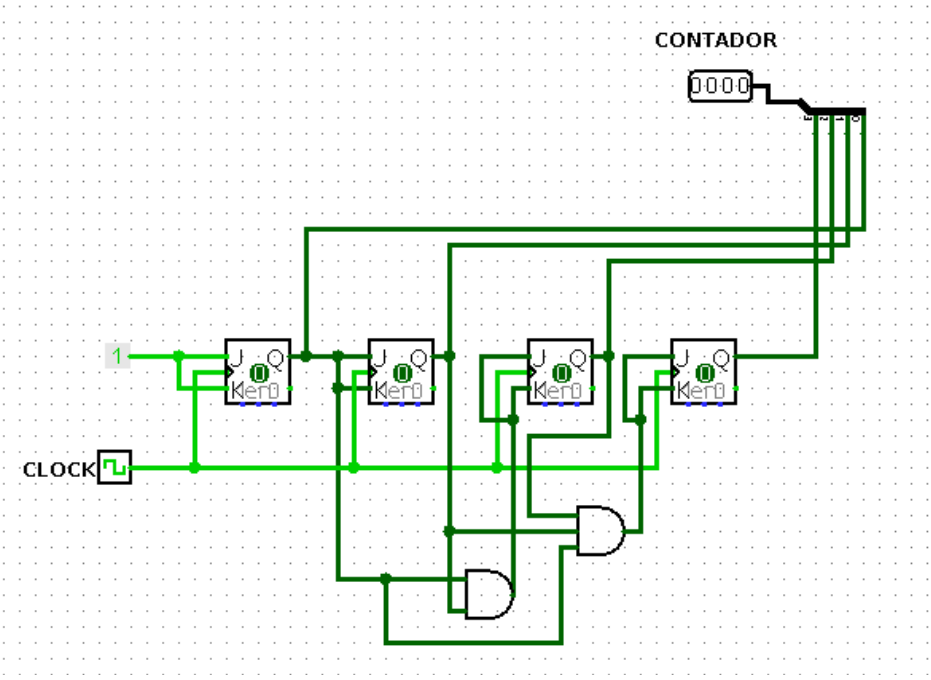
- Entrada de 1010 no circuito;



2.11. Máquina de Estados

2.12. Contador Síncrono

Contador síncrono é um circuito digital formado por flip-flops em paralelos, tal que todas as entradas clocks estejam conectadas na mesma fonte de clock.



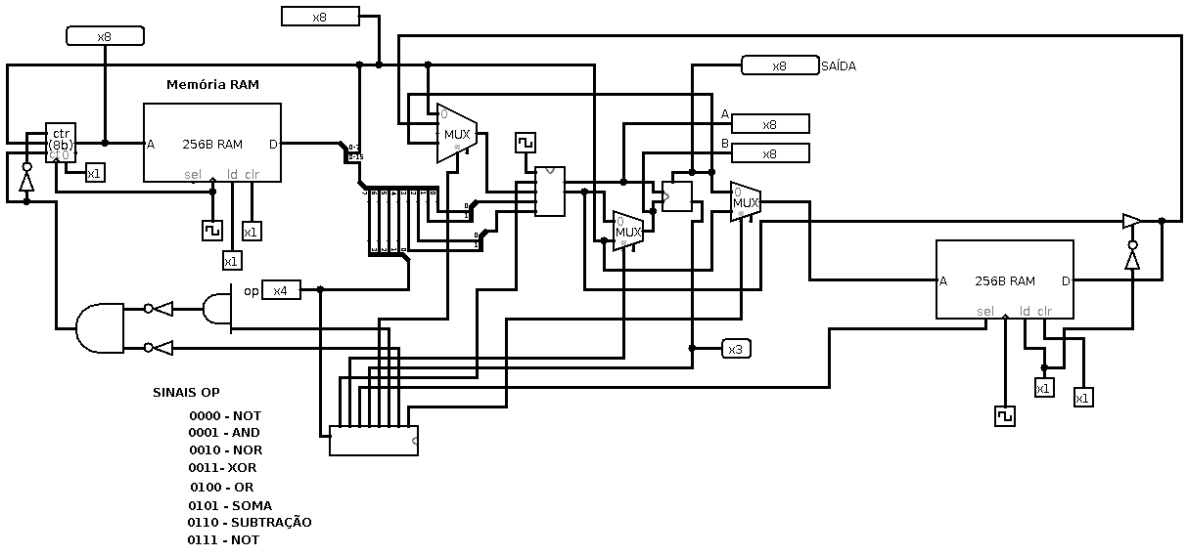
O contador possui uma sequência de 4 Flip-Flops de tipo JK, um pino de uma constante, o pulso de clock e a saída. A cada subida do clock para 1, é aumentado o número da saída. A seguir é apresentado todos os resultados de saída ao pulso do clock.

Pulso de clock	Saída de bits
1	0001
0	0001
1	0010
0	0010
1	0011
0	0011
1	0100
0	0100
1	0101
0	0101
1	0110
0	0110
1	0111
0	0111
1	1000
0	1000
1	1001
0	1001
1	1010
0	1010
1	1011
0	1011
1	1100
0	1100
1	1101

0	1101
1	1110
0	1110
1	1111
0	1111
1	0000

3. 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.



3. Considerações finais

Para com a construção e organização do processador final, foi perceptível a dificuldade para a realização do mesmo, tanto por motivos de entender a lógica por dentro de todos os componentes, como também a implementação dessa lógica em um circuito funcional.

Por meio de muitas pesquisas e incansáveis dias tentando implementar os componentes de uma maneira certa, conseguimos relacionar a lógica deles à criação dos circuitos em Logisim. O processador de 8 bits infelizmente não foi testado em simulações por motivos de tempo de realização dos componentes e a construção do mesmo.