

Projeto

ZeptoProcessador-II de 16 bits

Nathália Oliveira Pereira, 18042980

Kassio Gandara de Souza, 180140540

Grupo G36

¹Dep. Ciéncia da Computação – Universidade de Brasília (UnB)
CIC0231 - Laboratório de Circuitos Lógicos

180042980@aluno.unb.br, 180140540@aluno.unb.br

Abstract. This project consists in building a Zeptoprocessor capable of performing eight operations implemented using 32 bits instructions. Some basic components were used for that purpose, namely a PC address register, a instruction memory, a register file, an arithmetic logic unit, a comparator between two numbers of 16 bits and a control unit. A couple programs were tested including a counter from -16 to 16, an adder of odd numbers between 0 and 15, a multiplier between two numbers either signaled or unsignaled, a divider between two unsignaled numbers and a program which returns the rest of the division between two unsignaled numbers. Among the main results obtained, it was observed that the maximum clock frequency allowed so that the processor still operates normally is 384,62 MHz and that addition operations were more frequently used when writing programs to be run by the Zeptoprocessor.

Resumo. Esse projeto consiste na montagem de um Zeptoprocessador capaz de realizar oito operações implementadas por meio de instruções de 32 bits. Alguns componentes básicos foram utilizados para isso, como um registrador de endereço PC, uma memória de instruções, um banco de registradores, uma unidade lógico aritmética, um comparador entre dois números de 16 bits e uma unidade de controle. Foram testados alguns programas como um contador de -16 a 16, um somador de números ímpares de 0 a 15, um multiplicador de dois números tanto sem quanto com sinal, um divisor de dois números sem sinal e um programa que retorna o resto da divisão entre dois números sem sinal. Entre os principais resultados obtidos, observou-se que a frequência de relógio máxima permitida para que o processador ainda opere normalmente é 384,62 MHz e que operações de adição foram mais frequentemente utilizadas ao escrever programas a serem executados pelo Zeptoprocessador.

1. Introdução

O processador, também chamado de unidade central de processamento, CPU ou *Central Processing Unit*, é a parte de um sistema computacional que realiza as instruções aritméticas e lógicas de um programa de computador [Wikipedia 2020c]. Este projeto consiste na implementação de um processador simples capaz de realizar oito operações diferentes. Nos programas de teste, o circuito recebe uma série de instruções de 32 bits escritas em linguagem de máquina [Wikipedia 2020a], que contém um *opcode*, endereços para os registradores Rd, Ra e Rb e um número constante de 16 bits chamado de imediato, que é um número utilizado nas operações que não está armazenado em nenhum registrador. O *opcode* define qual das oito operações deverá ser realizada entre os dados dos registradores Ra e Rb e o imediato, sendo que o resultado será registrado em Rd. É importante notar que o imediato e os dados armazenados nos registradores são escritos em complemento de dois. Assim, números positivos são representados de forma idêntica ao sistema binário, enquanto números negativos tem seus bits invertidos e acrescidos de 1. A seguir, as diferentes partes do processador montado são explanadas.

Inicialmente, cada instrução do programa do usuário é escrita em uma linha diferente na memória de instruções. O registrador PC é então responsável por armazenar a informação de qual linha do programa está sendo executada no momento. O processador montado possui três instruções de salto (salto se igual sem considerar o sinal dos números, salto se menor ou igual sem considerar o sinal dos números e salto se menor ou igual considerando o sinal dos números), ou seja, instruções que comparam a informação armazenada em dois registradores e, se a comparação for verdadeira, salta para uma determinada linha do programa, definida pela adição entre o endereço PC atual e o imediato da instrução. Como existem instruções que comparam os números tanto com quanto sem sinal, é necessário utilizar um circuito capaz de comparar os números dessa maneira.

Ademais, o banco de registradores é um conjunto de registradores que podem ser acessados de forma organizada. Esse circuito é capaz de realizar operações de leitura de dados anteriormente gravados (acesso aos registradores Ra e Rb que serão utilizados para realizar operações) e de escrita de dados para modificar as informações previamente armazenadas (armazenamento do resultado da operação no registrador Rd) [EPUSP 2021]. Note que apesar de ser possível modificar os dados armazenados nos registradores, as instruções em si são permanentes depois de iniciada a simulação do circuito. Se fosse necessário implementar um processador capaz de realizar operações de *load* e *store*, ou seja, um circuito que pode alterar as instruções do programa, seria preciso implementar uma

memória de dados (círcuito que permite a escrita de dados e não só a leitura) utilizando-se memórias RAM. Como, instruções de escrita não fazem parte das oito instruções que o processador montado é capaz de realizar, apenas a memória de instruções, implementada com o uso de memórias somente de leitura [Wikipedia 2020b], foi utilizada.

Outro componente presente no processador é a Unidade Lógico Aritmética, responsável por realizar as operações de adição, subtração e or, and e xor bitwise, ou seja, qualquer operação que não seja de salto. Para esse projeto, será utilizada a ULA disponível no deeds [Deeds] que já realiza as operações considerando os números em complemento de dois. Dessa forma, utilizando os circuitos explicados, pode-se montar o Zeptoprocessador.

1.1. Objetivos

Esse projeto tem como objetivo utilizar os conhecimentos obtidos ao longo da disciplina para implementar um processador funcional simples capaz de realizar 8 operações.

Nome	Operação
Soma com imediato	$Rd = Ra + Rb + Imm$
Subtração com imediato	$Rd = Ra - Rb - Imm$
And bitwise com imediato Rb	$Rd = Ra$ Imm
Or bitwise com imediato	$Rd = Ra — Rb — Imm$
Xor bitwise com imediato	$Rd = Ra \oplus Rb \oplus Imm$
Salto se igual	$Ra == Rb ? PC = PC + Imm : PC = PC + 1$
Salto se menor ou igual sem sinal	$Ra <= Rb ? PC = PC + Imm : PC = PC + 1$
Salto se menor ou igual com sinal	$Ra <= Rb ? PC = PC + Imm : PC = PC + 1$

Tabela 1. Operações do Processador

Esse processador poderá ser utilizado para executar diversos programas, porém os programas a serem testados são os seguintes: um contador de -16 a 16, a soma dos números ímpares de 0 a 15, a divisão de dois números sem sinal e com sinal e a multiplicação de dois números sem sinal e com sinal. Ao montar um processador que execute com sucesso esses programas, será possível entender melhor como os conceitos estudados até agora são utilizados em um computador.

2. Metodologia

Os componentes explicados anteriormente foram implementados como blocos no simulador e então utilizados para montar o processador. Depois de escrever o programa de teste disponível no roteiro nas memórias ROM do circuito, foi utilizado inicialmente uma entrada em forma de botão no lugar do relógio usado como entrada em todos os registradores. Desse modo, controlando manualmente a borda de subida do relógio, observou-se o resultado de cada linha de instrução com o auxílio de LEDs mostrando o endereço de PC, o número da instrução e dos registradores Ra, Rb e Rd, o imediato e os dados armazenados em Ra e Rb. Assim é mais simples identificar eventuais problemas no circuito, por exemplo, se depois de uma instrução de salto o endereço de PC não se comportar da maneira esperada, o erro provavelmente estará no comparador.

Começa-se a montagem do circuito pelo registrador PC definido em 0000 com a ajuda da entrada CLEAR 0, que deverá ser alterada para 1 depois de iniciada a simulação para que o endereço PC possa ser alterado. O *enable* do registrador, representada por um botão de entrada, também deve ser ativado para que o PC possa funcionar. É revelante observar que o endereço PC só é alterado durante a borda de subida do relógio. Dois displays de segmento de dois dígitos hexadecimais foram usados para mostrar a entrada PC de 16 bits.

O banco de registradores lê o dado armazenado nos registradores Ra e Rb e assim obtém-se as saídas Sa e Sb que serão usadas junto com o imediato para realizar as operação pedida (lembrando que o banco de registradores só é utilizado se a instrução possuir uma operação a ser realizada, ou seja, não for do tipo salto). o resultado obtido da operação realizada na ULA usando os dados do registradores Ra, Rb e o imediato será armazenado no registrador Rd.

Observando a parte do circuito que determina o endereço PC, nota-se um multiplexador 2x1 cuja entrada 00 (usada quando a instrução não é de salto) é o endereço PC acrescentado de 1, ou seja, o programa passa para a próxima linha da instrução. Já se a instrução for do tipo salto, o circuito soma ou subtrai (dependendo se o imediato for positivo ou negativo) o imediato do endereço PC. Assim, o bit mais significativo do imediato, ou seja, o bit que define se o número em complemento de dois é positivo ou negativo é usado para definir qual será a operação realizada na ULA enquanto o imediato com o bit mais significativo igualado a zero é usado como entrada na ULA juntamente com o endereço PC.

2.1. Memória de Instruções

Como o tamanho da instrução do processador é 32 bits, foram utilizadas duas memórias ROM 4Kx16 do Deeds [Deeds] mostradas na figura 1 para montar a memória de instruções.

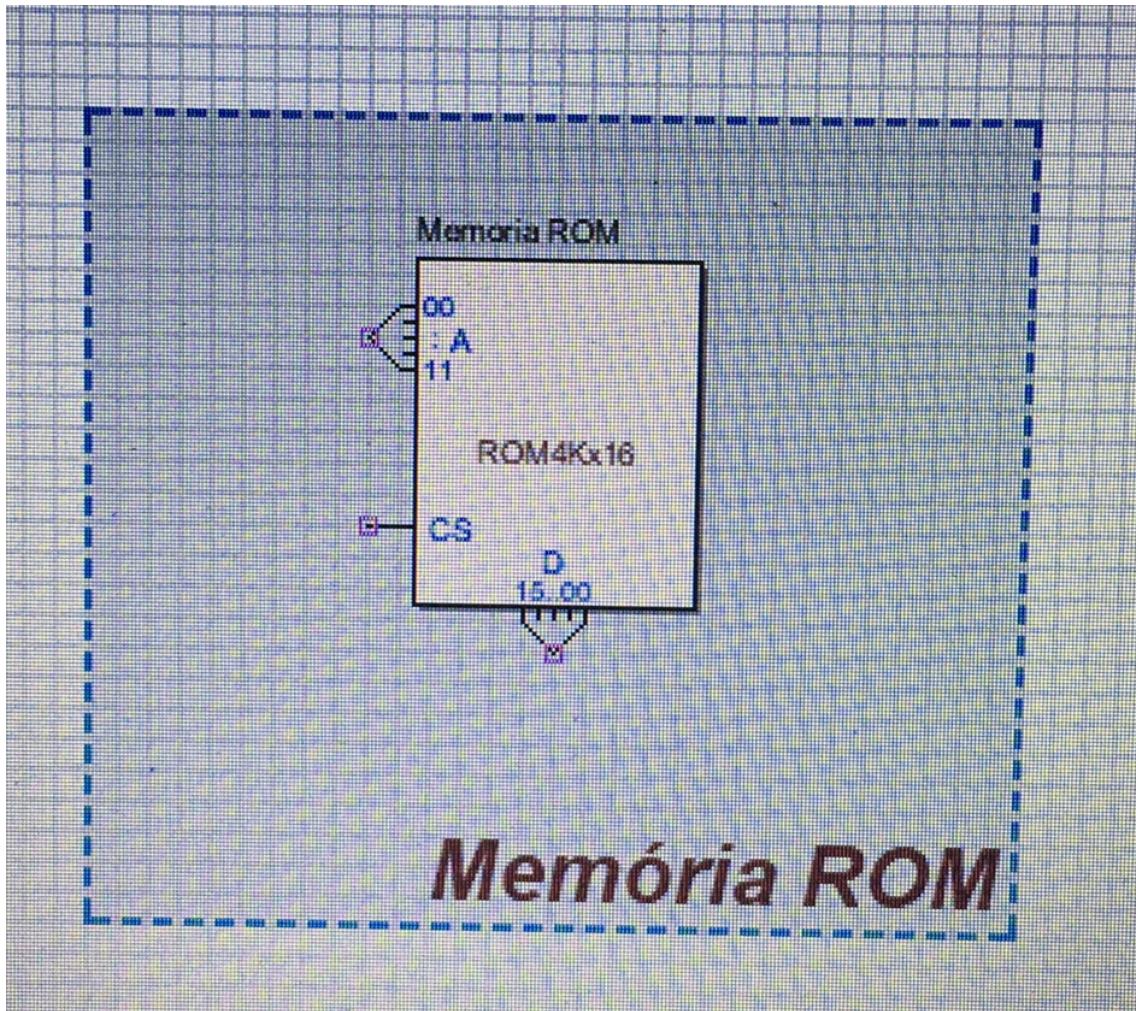


Figura 1. Memória ROM 4Kx16

Assim, o circuito recebe como entrada o endereço PC de 16 bits e usa os 11 bits menos significativos como entrada para ambas as memórias programadas com as instruções do programa do usuário. A primeira memória contém os 4 últimos bits de cada instrução em hexadecimal, escritos na linha do endereço PC correspondente. A saída dessa memória é então usada como os 16 bits menos significativos da instrução em binário, ou seja, armazena os registradores e o opcode. Já a segunda memória contém os primeiros 4 bits da instrução em hexadecimal e a saída é usada como os 16 bits mais significativos da instrução em binário, que armazenam o imediato, totalizando a instrução

de 32 bits. Com o uso de fios de barramento para separar cada bit da saída das memórias, pode-se conectar ambas as saídas em um único fio de barramento. É válido observar também que entradas de nível lógico alto 1 foram usadas como *Chip Selector* em ambas as memórias para que estas sempre estejam ativas durante o funcionamento do circuito.

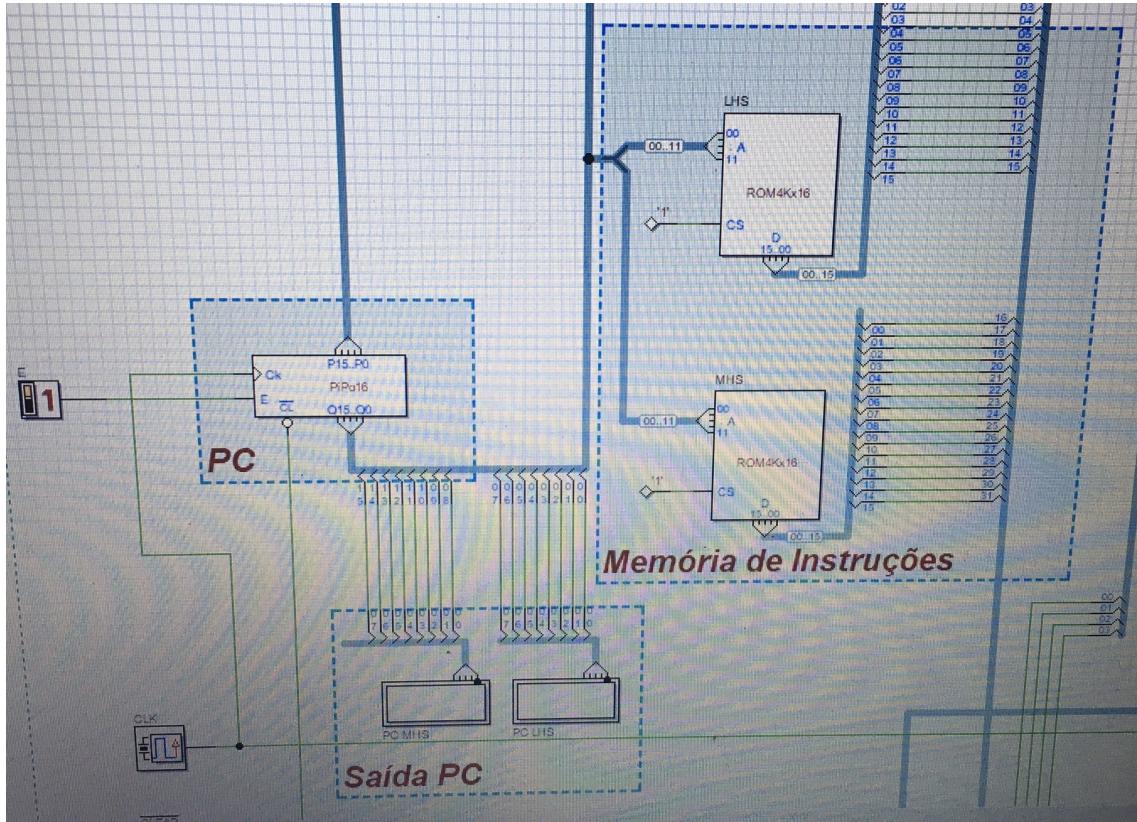


Figura 2. Memória de Instruções e registrador PC

Então, usando o barramento contendo a instrução de 32 bits, separa-se os bits de acordo com a função de cada um. Os quatro bits menos significativos são o opcode da instrução (mostrados com um display de 1 dígito hexadecimal) e serão usados como a entrada no bloco de controle. Os próximos doze bits definem os registradores Ra, Rb e Rd respectivamente, cada um definido por quatro bits, que serão usados como entrada no banco de registradores. Os demais 16 bits definem o imediato a ser usado nas operações.

2.2. Unidade de Controle

A Unidade de controle recebe como entrada o Opcode de 4 bits e retorna o Opcode da ULA correspondente conforme a tabela 2. Como o bit mais significativo do opcode da instrução é sempre 0, pode-se desconsiderá-lo e usar os bits restantes como entradas S0, S1 e S2 em um multiplexador 8x1 que relaciona cada opcode da instrução com um opcode da ULA usando entradas constantes de 3 bits. Note que essas entradas representam os 4 bits mais significativos do opcode da ULA, já que o bit menos significativo é sempre igual ao bit anterior para qualquer opcode da ULA conforme pode-se observar na tabela 2.

Opcode	Opcode da ULA
0000	11000
0001	11011
0010	01000
0011	01100
0100	10000
0101	00000
0110	00000
0111	00000

Tabela 2. Opcode da ULA

Além disso, a unidade de controle também retorna o Write Enable WE para definir se o resultado da operação deverá ser armazenado no banco de registradores, que só será 1 se a operação possuir um resultado, ou seja, se a operação não for do tipo salto. A saída SI para definir se o comparador deverá considerar os números com ou sem sinal só será 1 para a operação 0111 (salto se menor ou igual Ra e Rb considerados com sinal). A saída IG define se a operação é do tipo salto se igual e a saída MEI define se a operação é do tipo salto se menor ou igual. Assim, montando a tabela verdade 3 podemos obter as seguintes equações e implementá-las usando portas lógicas:

$$WE = \overline{S2} + S2.\overline{S1}.\overline{S0} \quad (1)$$

$$SI = S2.S1.S0 \quad (2)$$

$$IG = S2.\overline{S1}.S0 \quad (3)$$

$$MEI = S2.S1 \quad (4)$$

Entradas	Saídas				
Opcode (S3 S2 S1 S0)	Opcode da ULA	Write Enable	Signaled	Igual	Menor ou igual
0000	11000	1	0	0	0
0001	11011	1	0	0	0
0010	01000	1	0	0	0
0011	01100	1	0	0	0
0100	10000	1	0	0	0
0101	00000	0	0	1	0
0110	00000	0	0	0	1
0111	00000	0	1	0	1

Tabela 3. Bloco de Controle

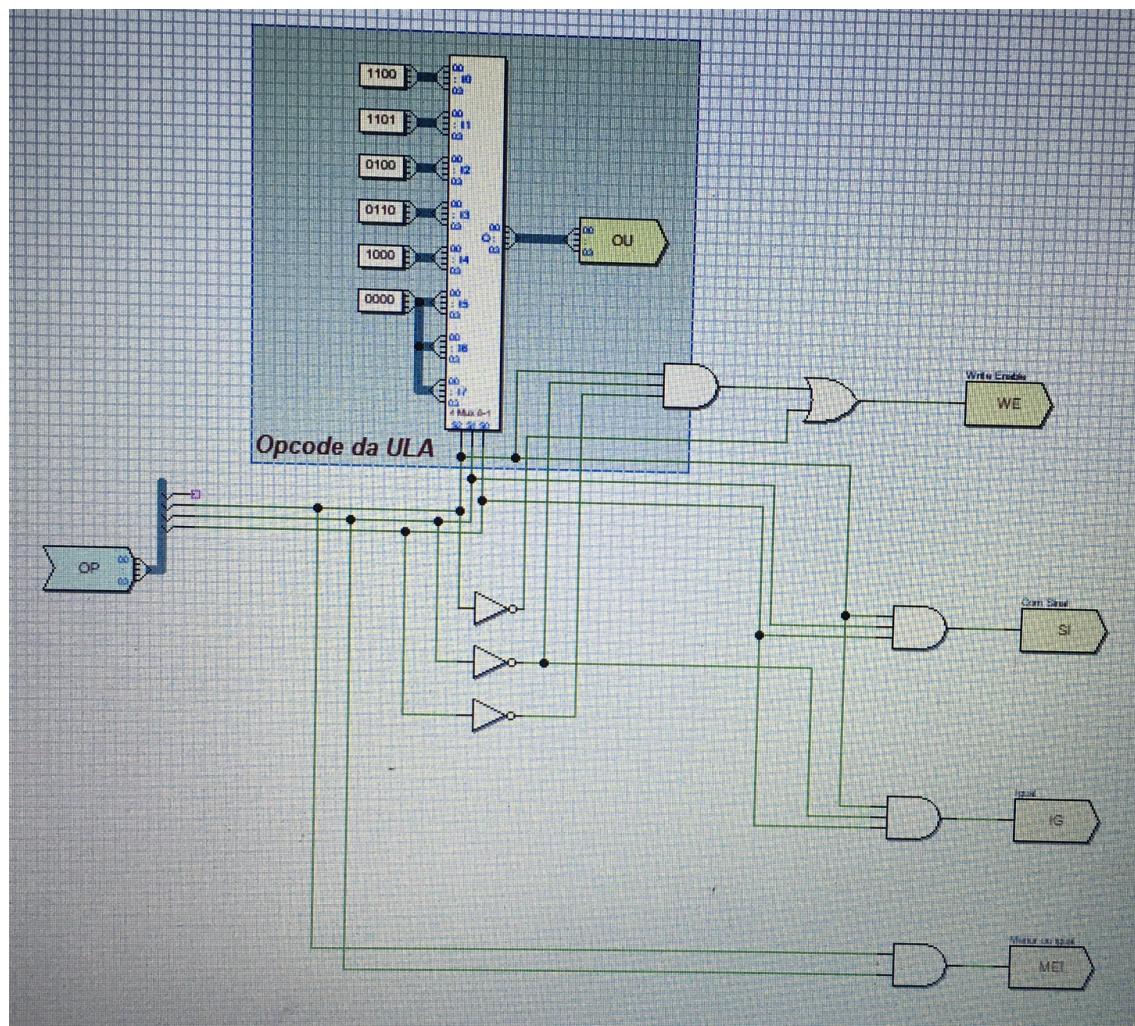


Figura 3. Unidade de Controle

2.3. Banco de Registradores

Para montar o banco de registradores de 16 bits [Khatib 2014], foram usados registradores PiPo16 disponíveis no Deeds mostrados na figura 10.

O banco de registradores recebe como entrada o endereço de 4 bits do registrador Rd no qual o resultado da operação realizada na ULA deverá ser registrado. Dessa maneira, o endereço Rd é usado como entrada de um decodificador 4x16 que definirá em qual dos 16 registradores do banco a operação deverá ser escrita. Assim, cada saída do decodificador será conectada a entrada *Enable* do registrador correspondente após passar por uma operação AND com a entrada WE (*Write Enable*). Note que a entrada WE só é ativada quando existe um resultado de operação a ser escrito em um registrador, ou seja, se a instrução não for do tipo salto. Note também que assim como o registrador PC, todos os 16 registradores possuem uma entrada CLEAR para que possam ser inicializados em 0 e uma entrada de relógio cuja borda de subida define a escrita ou leitura de uma instrução. A entrada W (*Write*) é o resultado da operação realizada pela ULA e representa o dado a ser registrado, logo é usada como entrada para os registradores, sendo que a instrução só será armazenada naquele registrador se este estiver ativo, ou seja, se este for registrador Rd.

Como o deeds não tem um multiplexador 16 x 1 com entradas de barramento, foram usados dois multiplexadores 8 x 1 e um multiplexador 2 x 1 com entradas de barramento de 16 bits [Youtube 2020]. Desse modo, as saídas dos oito primeiros registradores são usados como entradas para o primeiro multiplexador 8x1 e as saídas dos demais registradores no segundo multiplexador. A saída de cada multiplexador será então definida pelos 3 bits menos significativos do endereço Ra e o bit mais significativo será usado no multiplexador 2x1 para definir qual das duas saídas é correspondente ao dado armazenado no registrador Ra. Um procedimento análogo foi usado para definir a saída correspondente ao dado armazenado no registrador Rb.

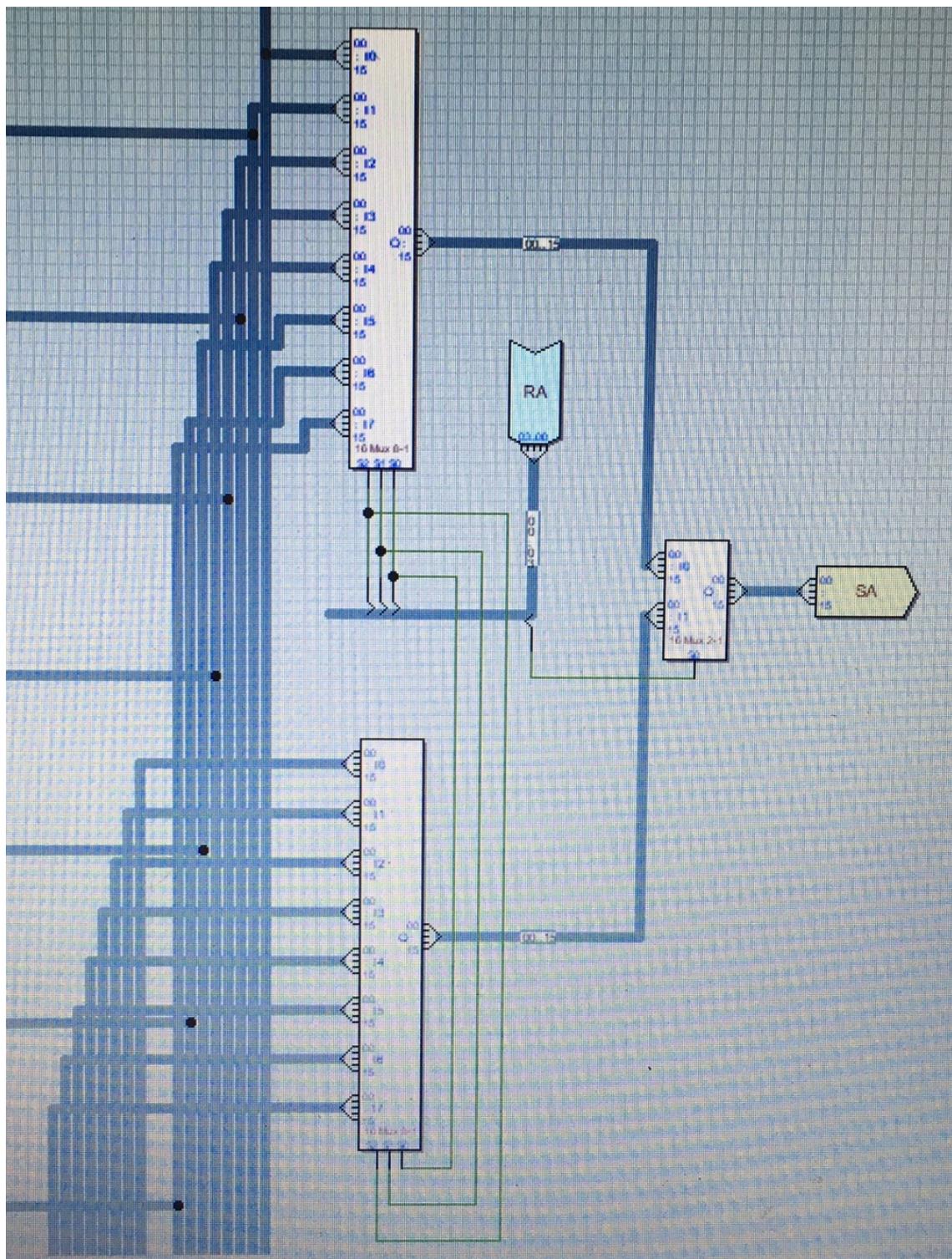


Figura 4. Multiplexador 16 x 1

Assim, foi montado um sistema de três multiplexadores, mostrado na figura 4, representando o multiplexador 16x1 para a saída SA (o multiplexador 16 x 1 para a saída SB funciona de maneira análoga). O primeiro multiplexador 8x1 recebe os dados dos 8 primeiros registradores e o segundo registrador recebe os dados dos 8 registradores restan-

tes. A saída de cada multiplexador será então definida pelos 3 bits menos significativos do endereço Ra e o bit mais significativo será usado no multiplexador 2x1 para definir qual das duas saídas é correspondente ao dado armazenado no registrador Ra.

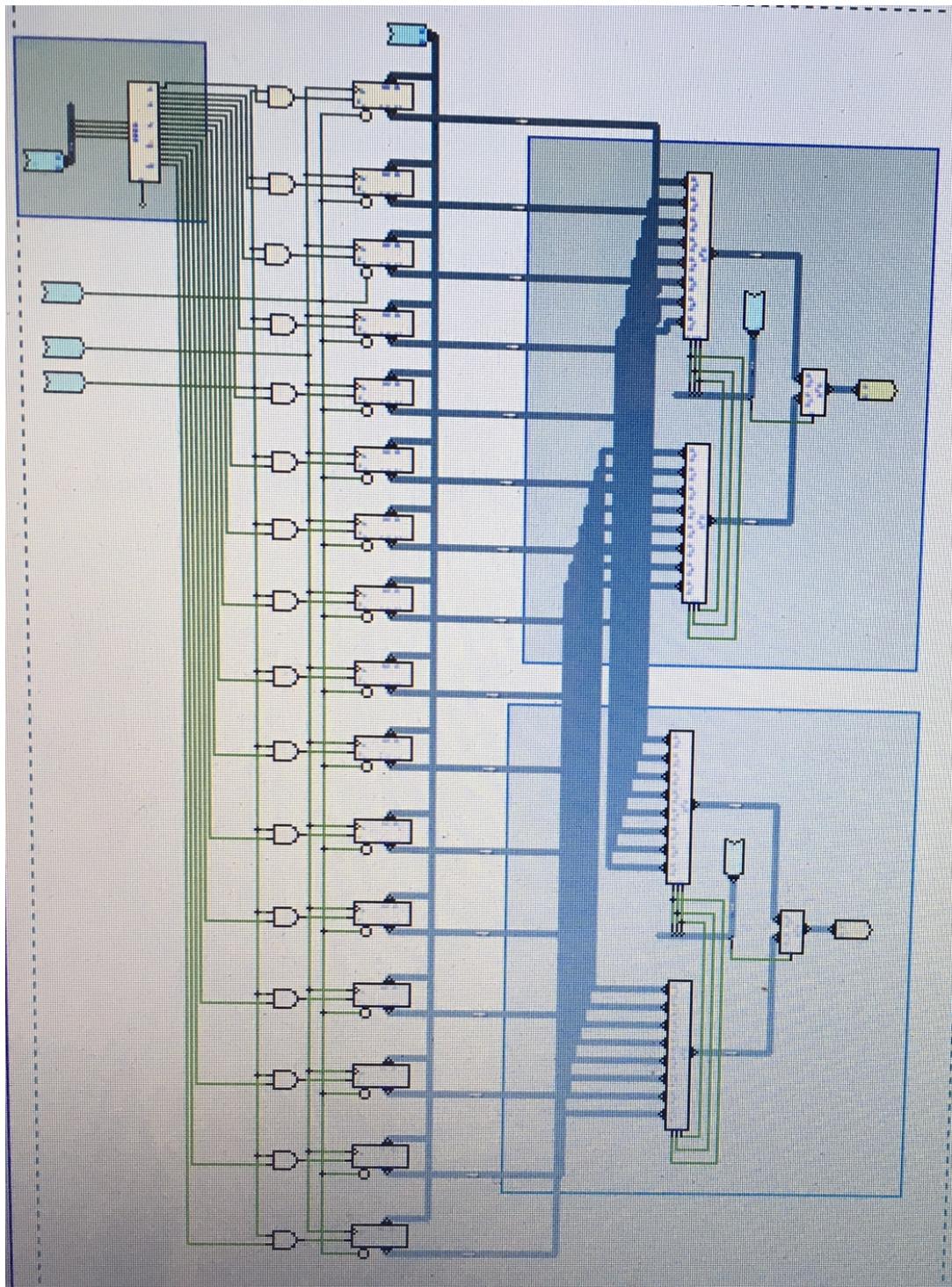


Figura 5. Banco de Registradores de 16 bits

2.4. ULA

Continuando a montagem dos circuitos pelos quais o processador é composto, a saídas de 16 bits SA e SB do banco de registradores serão usadas como entradas para a Unidade Lógico Aritmética, juntamente com o número imediato de 16 bits e o Opcode da ULA gerada pelo bloco de controle. Como é preciso realizar duas operações, uma entre os dados de Ra e Rb e outra entre o resultado dessa operação e o imediato, foram utilizadas duas Unidades Lógico Aritméticas ALU 16 disponíveis no deeds. O Opcode da ULA é conectado às entradas S4, S3, S2, S1, S0 de ambas as ULAs, do bit mais significativo ao menos significativo, lembrando que o bit menos significativo é usado tanto na entrada S1 quanto na entrada S0. As entradas Ci Carry In são sempre 0 e a saída S é um número de 16 bits representado o resultado da operação a ser escrito no banco de registradores.

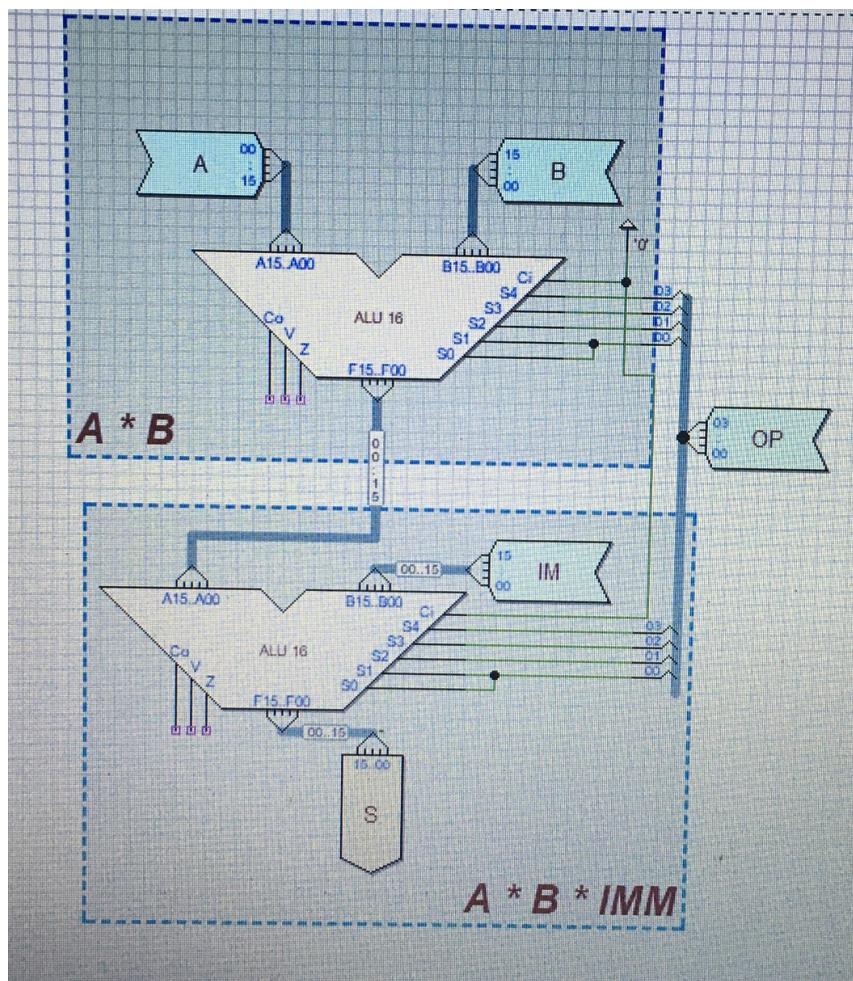


Figura 6. Unidade Lógico Aritmética

2.5. Comparador com sinal e sem sinal

O Deeds já possui um comparador para o módulo de dois números de 16 bits, assim foi preciso apenas implementar o circuito de maneira que os números escritos em complemento de dois possam ser comparados tanto com a inclusão do sinal quanto com a exclusão dele, dependendo do nível lógico da entrada Si. Para isso, é preciso primeiro converter as entradas A e B de complemento de 2 para binário. Se o número for positivo, não há conversão a ser feita, pois números positivos tem a mesma representação tanto em binário quanto em complemento de dois. Se o número for negativo, é preciso subtrair 1 com o auxílio de uma Unidade Lógico Aritmética com opcode constante de 11011 (operação de subtração) e depois inverter todos os bits utilizando fios de barramento e algumas portas NOT. Pode-se utilizar um multiplexador 2x1 que recebe na entrada S0 o bit mais significativo do número. A entrada 0 do multiplexador recebe o número sem alteração (o bit mais significativo igual 0 significa que o número é positivo) e a entrada 1 recebe o número subtraído de 1 com os bits invertidos (o bit mais significativo igual 1 significa que o número é negativo). Realizando esse procedimento tanto para a entrada A quanto para a entrada B, obtém-se o módulo de cada um dos números.

Para definir se A é menor ou igual a B considerando o sinal, um multiplexador 4x1 foi utilizado para lidar com quatro diferentes situações. Na primeira situação, ambos os números são positivos (*MSB Most Significative Bit*, ou seja, o bit mais significativo de ambos é igual a 0) e a comparação pode ser feita diretamente entre os módulos dos números (operação OR entre as saídas $A < B$ e $A = B$). Na segunda situação, A é positivo e B é negativo (o MSB de A é 0 e o de B é 1), assim A é sempre maior que B e a comparação $A \leq B$ é sempre falsa e a entrada 01 do multiplexador recebe sempre o nível lógico baixo. Outrossim, quando A é negativo e B é positivo, A é sempre menor ou igual a B e a entrada 10 do multiplexador é sempre 1. Por fim, quando ambos forem negativos, o menor número será aquele que tiver o menor módulo, assim podemos realizar a comparação $A \neq B$ entre os módulos para determinar se o número negativo A é menor ou igual ao número negativo B.

Depois disso, pode-se usar um multiplexador 2x1 com a entrada S0 conectada a Si. Assim, se Si for 0, ou seja, o sinal não deve ser considerado, a saída MEI que define se um número é menor ou igual ao outro será resultado direto da comparação entre módulos. Já caso a entrada Si for 1, ou seja, o sinal deve ser considerado, a saída MEI será igual a saída do multiplexador 4x1 discutido no parágrafo anterior.

Por fim, para obter a saída IG que define se A é B, basta utilizar um multiplexador

2×1 que retorna o resultado da comparação $A = B$ entre os módulos de A e B se $Si = 0$ ou entre as próprias entradas A e B se $Si = 1$.

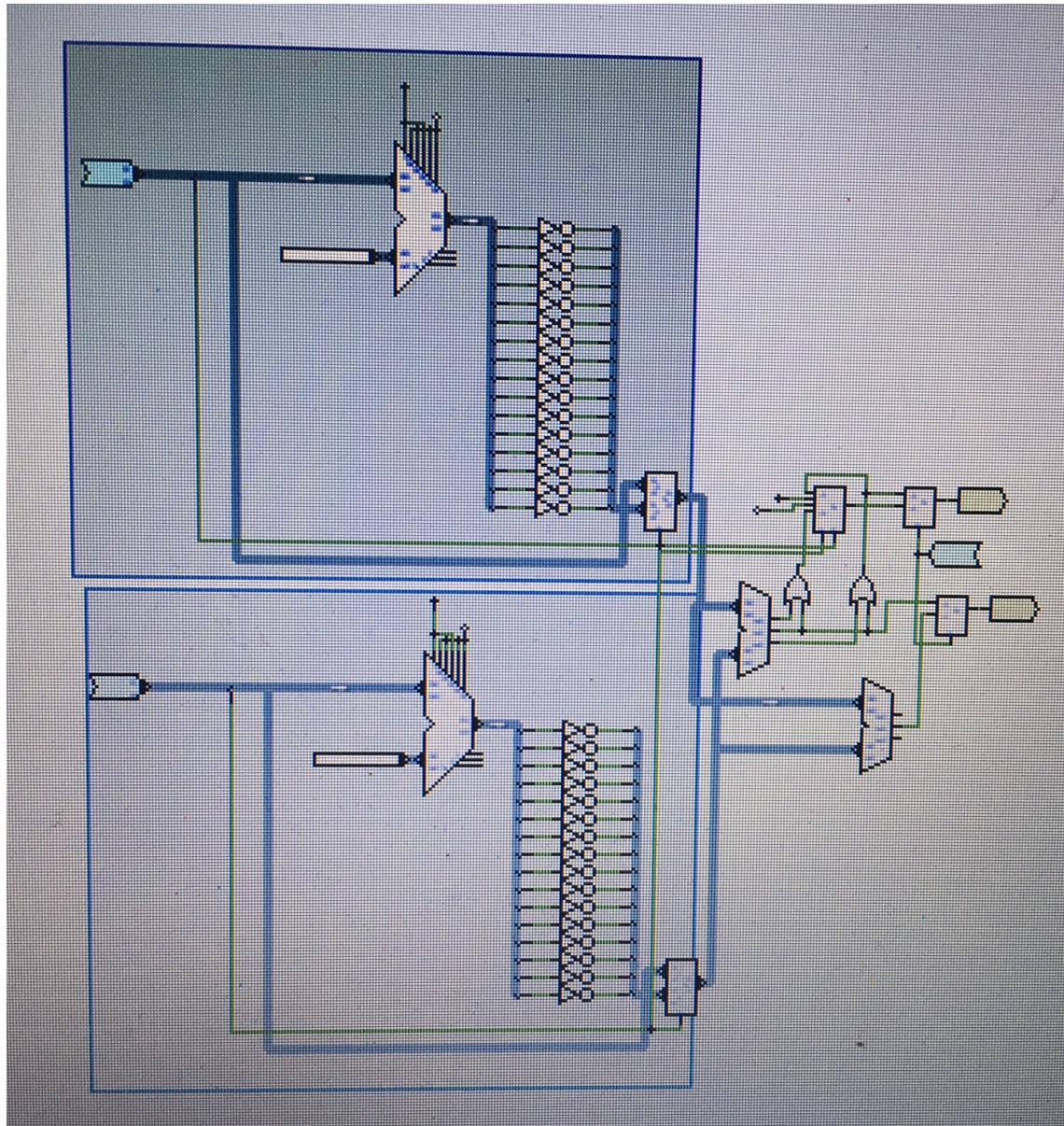


Figura 7. Comparador de números com sinal e sem sinal

Observando o circuito mais de perto, percebe-se a parte do circuito responsável por obter o módulo do número na figura e a parte responsável por fazer as comparações e retornar o resultado na figura

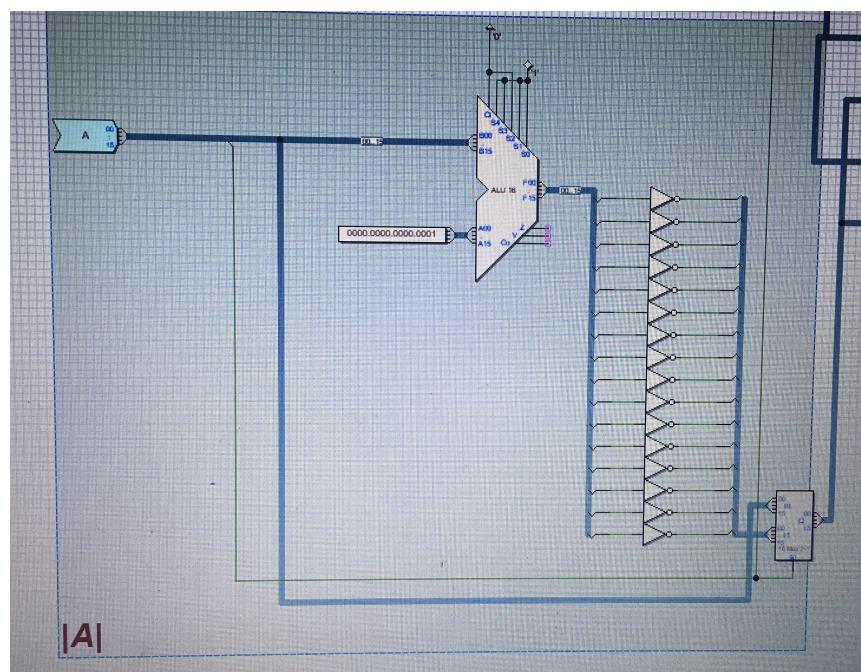


Figura 8. Obtenção do módulo de A

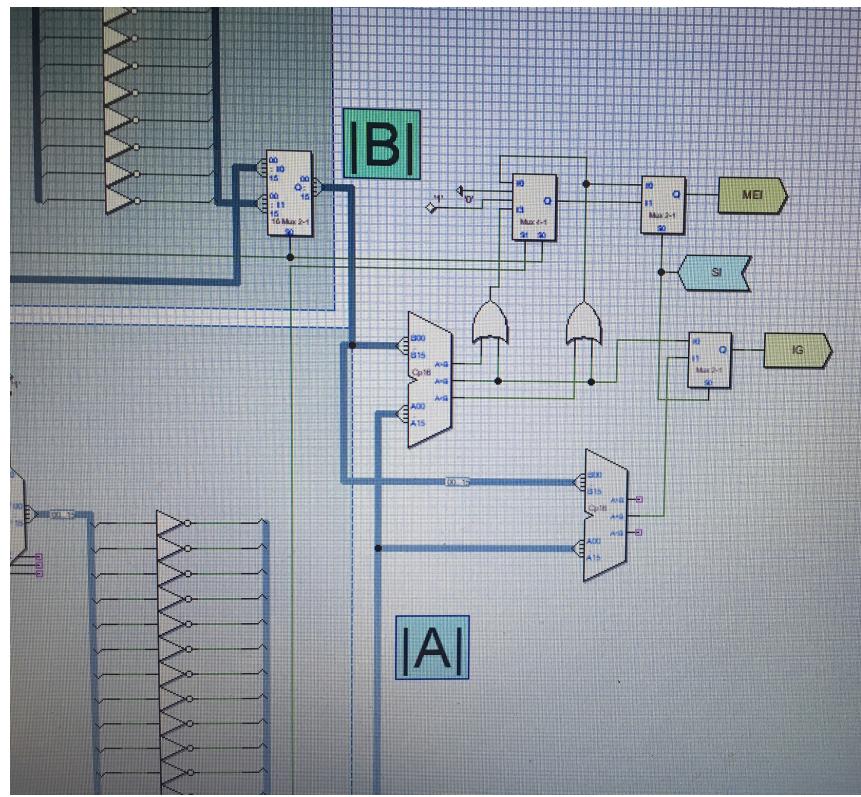


Figura 9. Comparaçao entre os dois números

2.6. Endereço PC

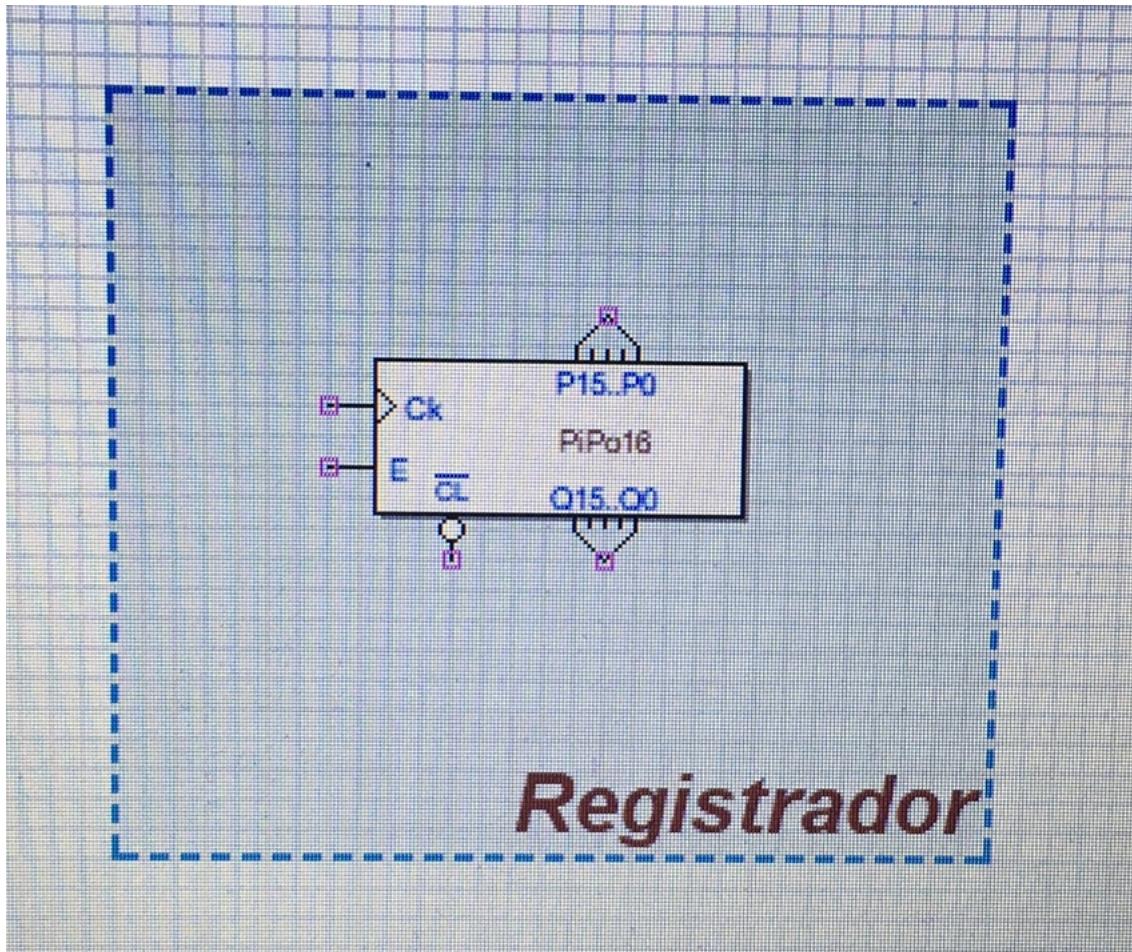


Figura 10. Registrador Pipo 16

Observando agora a parte do circuito que determina o endereço PC, o resultado da comparação entre A e B é usado para determinar se deve ocorrer um salto no endereço. Para isso, é preciso realizar uma operação AND entre a saída ME do bloco de controle e a saída MEI do comparador, que retornará 1 se a instrução for do tipo salto se menor ou igual e o dado de Ra for menor ou igual ao dado de Rb. Similarmente, realiza-se uma operação AND entre a saída IG do bloco de controle e a saída IG do comparador, que retornará 1 se a instrução for do tipo salto se igual e o dado de Ra for igual ao dado de Rb. Por fim, o resultado de uma operação OR entre essas duas saídas determinará se o salto deve ser realizado. Assim, o resultado obtido pode ser utilizado em um multiplexador 2x1 no qual a entrada 0 (usada quando a instrução não é de salto) é o endereço PC acrescentado de 1 (utilizou-se um somador do Deeds [Deeds] como na figura 15 e uma entrada constante de barramento de 16 bits definida em 1), ou seja, o programa passa para

a próxima linha da instrução. Já se a instrução for do tipo salto, o circuito soma o imediato ao endereço PC (também usou-se um somador do deeds para realizar essa operação).

O circuito final do processador é mostrado na figura 11. As figura 12, 13 e 14 mostram as partes do circuito com maior detalhe.

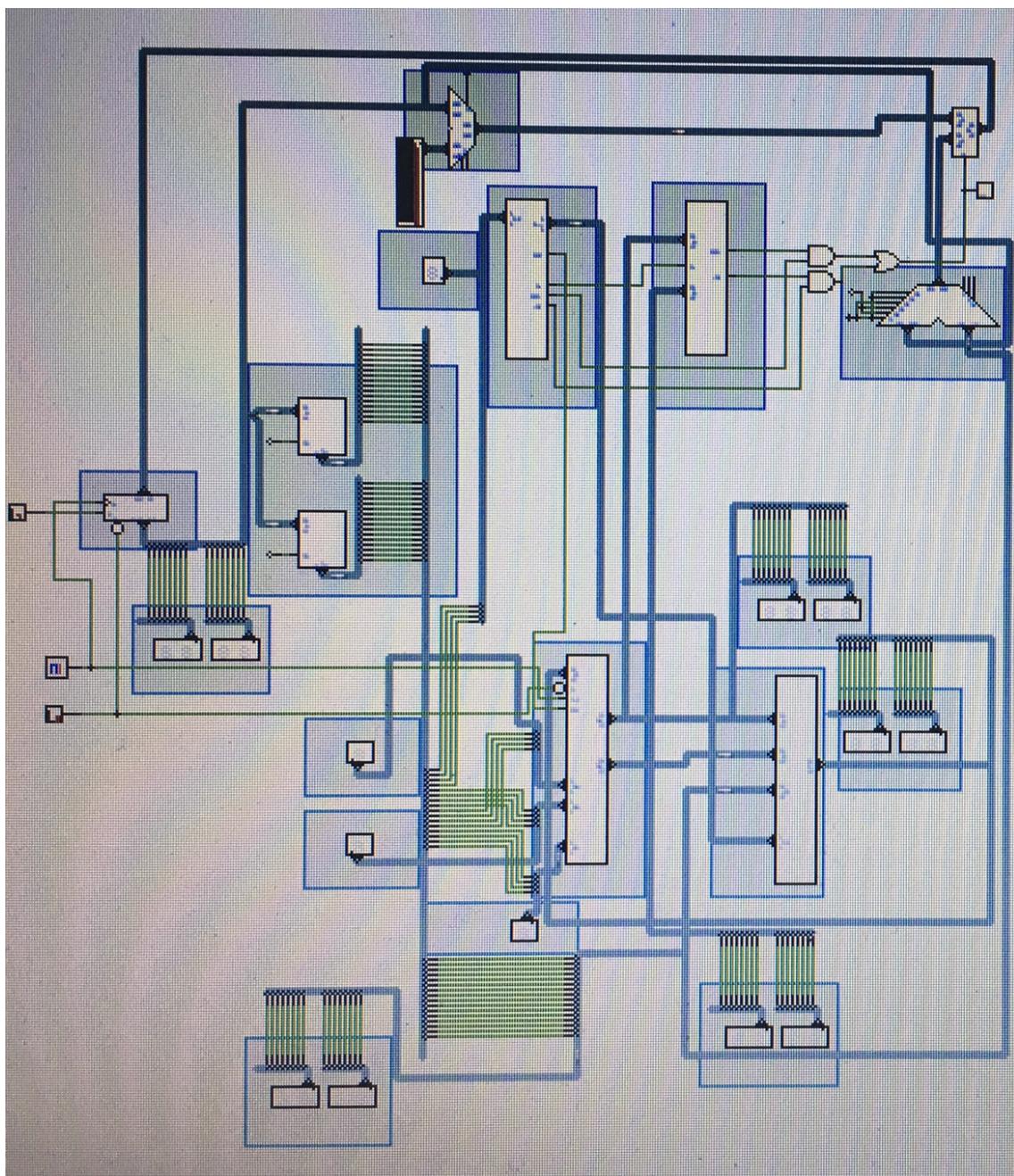


Figura 11. Zettoprocessador-II de 16 bits

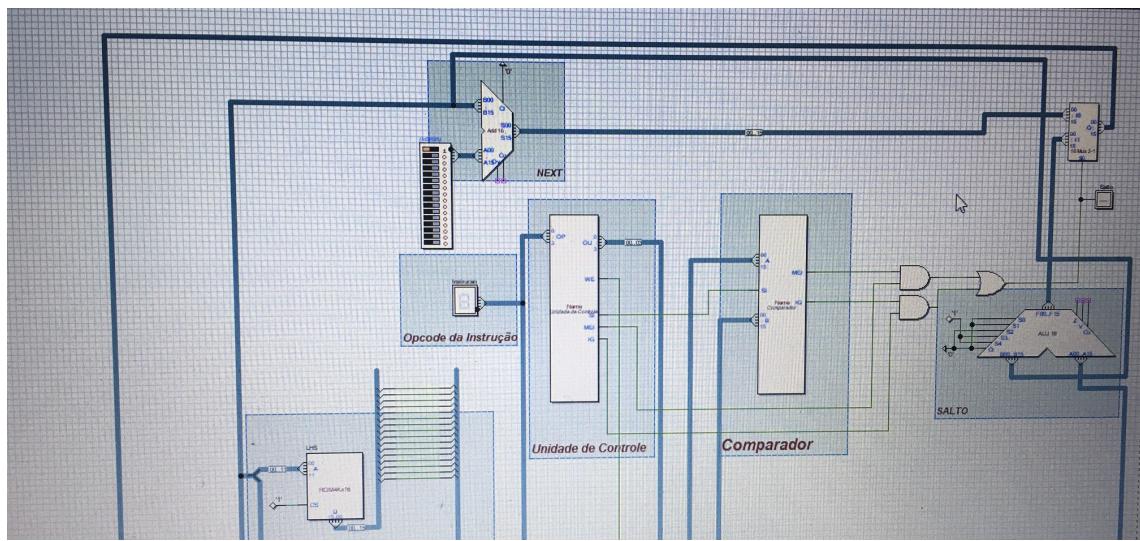


Figura 12. Zeprocessador-II de 16 bits

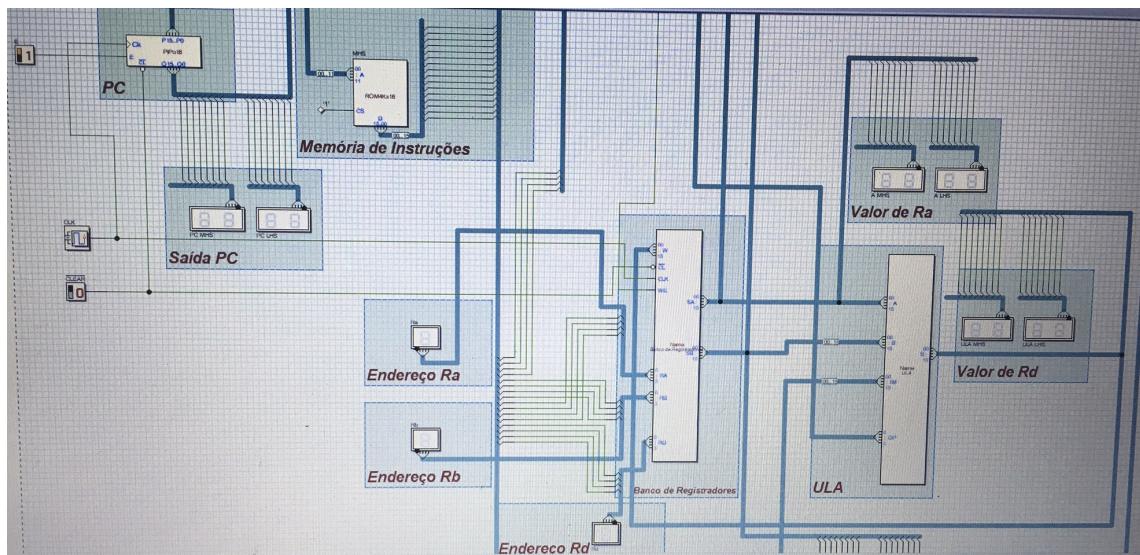


Figura 13. Zeprocessador-II de 16 bits

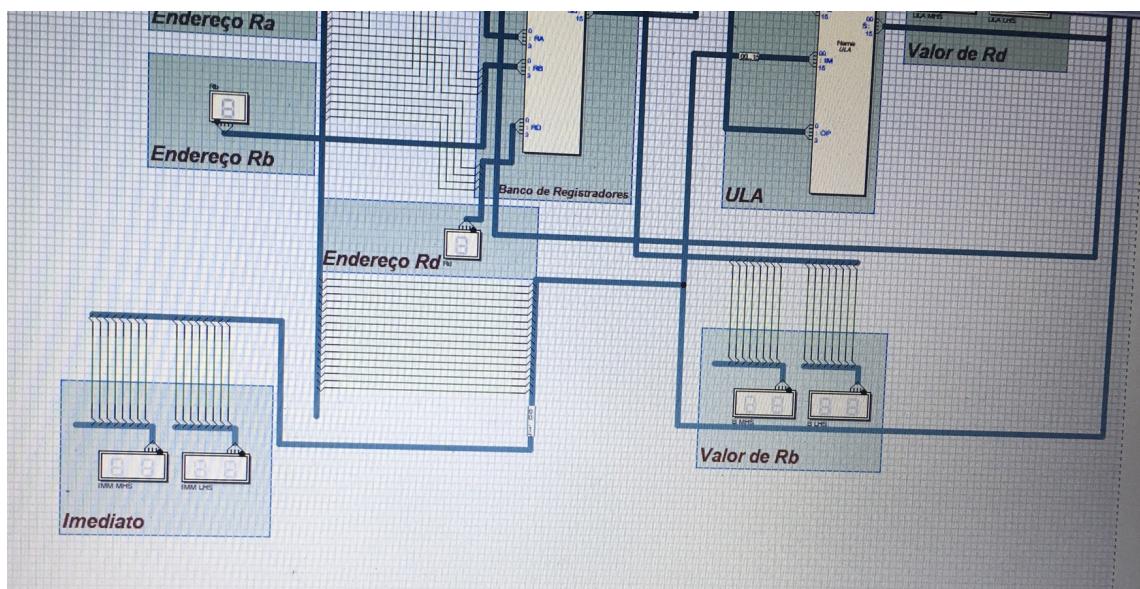


Figura 14. Zétoprocessador-II de 16 bits

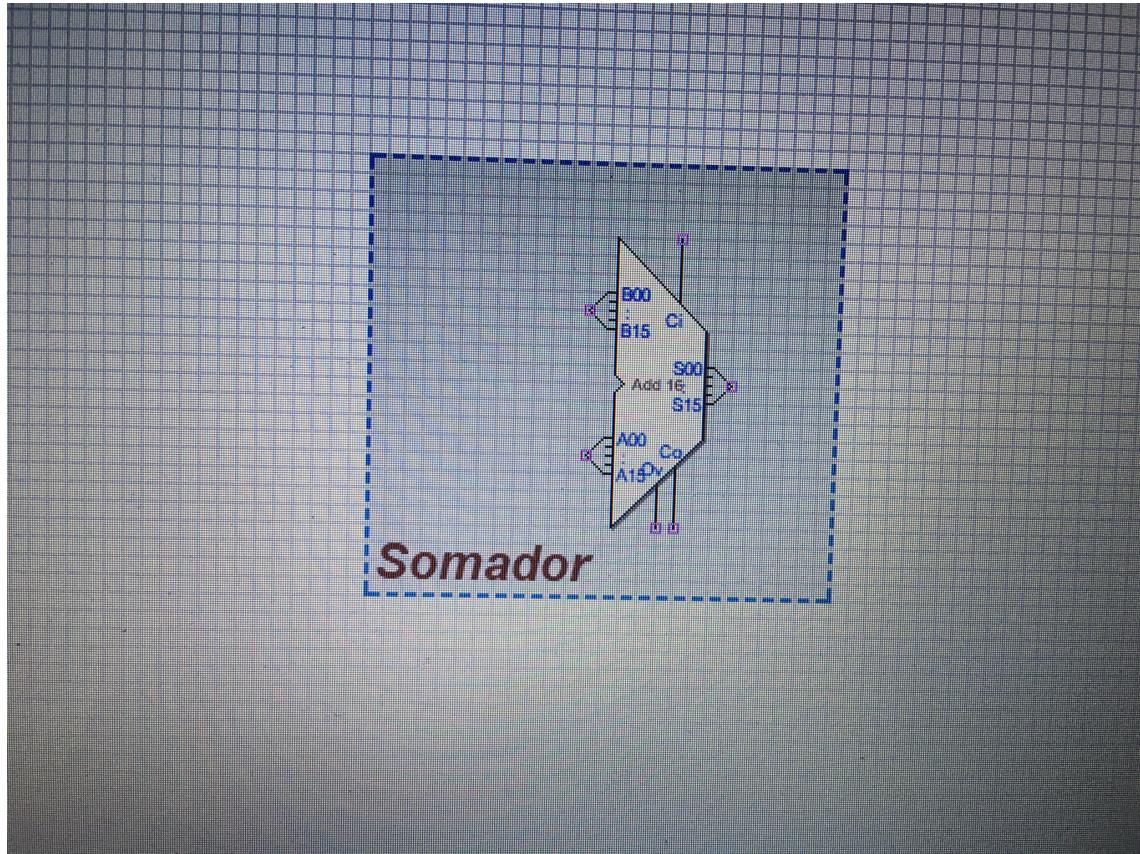


Figura 15. Somador do Deeds

3. Resultados Obtidos

3.1. Contador de -16 a 16

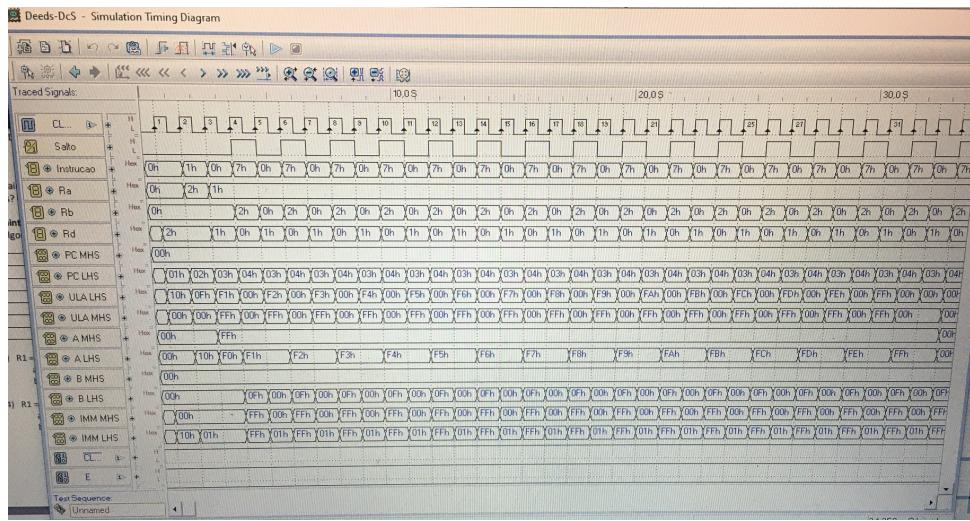


Figura 16. Parte 1 da simulação em forma de onda do contador de -16 a 16

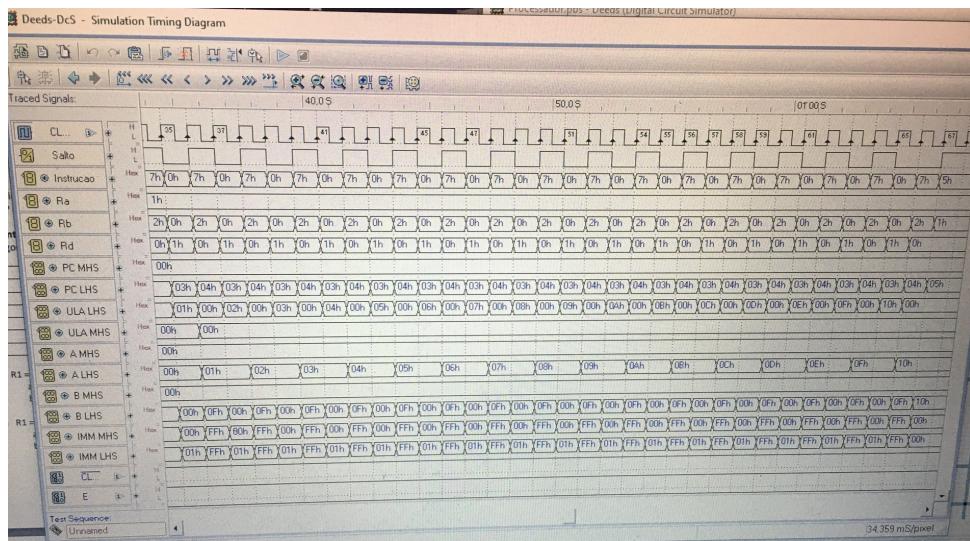


Figura 17. Parte 2 da simulação em forma de onda do contador de -16 a 16

O programa do contador de -16 a 16, disponível nas especificações do projeto, foi simulado nesse vídeo. Note que a instrução da primeira linha do programa, que era originalmente 0xFFFF0 1001 (subi R1,R0,R0,-16), foi corrigida para 0XFF0 1000 (addi R1,R0,R0,-16). A simulação em forma de onda é mostrada nas figuras 16 e 17.

3.2. Soma dos números ímpares de 0 a 15

O programa da soma de todos os ímpares de 0 a 15, disponível nas especificações do projeto, foi simulado nesse vídeo e a simulação em forma de onda é mostrada na figura 18. Observe que a frequência de relógio 384,62 MHz é a maior frequência possível de simulação do circuito para que esta funcione adequadamente. Isso porque, quando a frequência de relógio é muito alta, o tempo entre as bordas de subidas do relógio é maior que o tempo que o circuito demora para realizar as operações e registrar os resultados, ou seja, o tempo de propagação total do circuito.

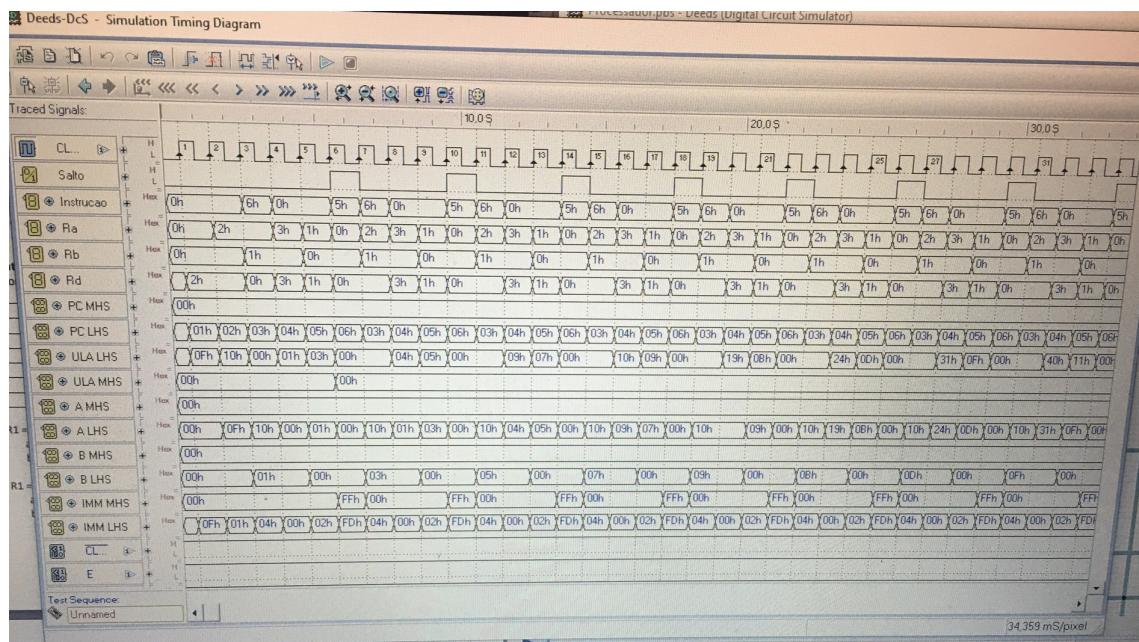


Figura 18. Simulação em forma de onda da soma dos números ímpares de 0 a 15

3.3. Multiplicação de dois números sem sinal

Endereço	Código hexadecimal	Instrução	Comentário
0x0000	0x0000 1000	addi R1,R0,R0,0	R1=0
0x0001	0xXXXX 2000	addi R2,R0,R0,X	R2=X
0x0002	0xYYYY 3000	addi R3,R0,R0,Y	R3=Y
0x0003	0x0001 5000	addi R5,R0,R0,1	R5=1 devido ao =
0x0004 Loop:	0x0001 5500	addi R5,R5,R0,1	R5=R5+1
0x0005	0x0000 1120	addi R1,R1,R2,0	R1=R1+R2
0X0006	0xFFFF 0536	bleu R0,R5,R3,-2	R5<=R3 (sem sinal) ? Loop : Fim
0x0007 Fim:	0x0000 0115	beq R0,R1,R1,0	J Fim

Tabela 4. Programa de multiplicação de dois números sem sinal

O programa de multiplicação de dois números sem sinal, apresentado na tabela 4, foi simulado nesse vídeo. De maneira geral, o programa adiciona o primeiro número ao resultado, somando 1 ao registrador auxiliar R5 a cada loop que deve ser executado N vezes, sendo que N é o valor do segundo número. A simulação em forma de onda é mostrada nas figura 19.

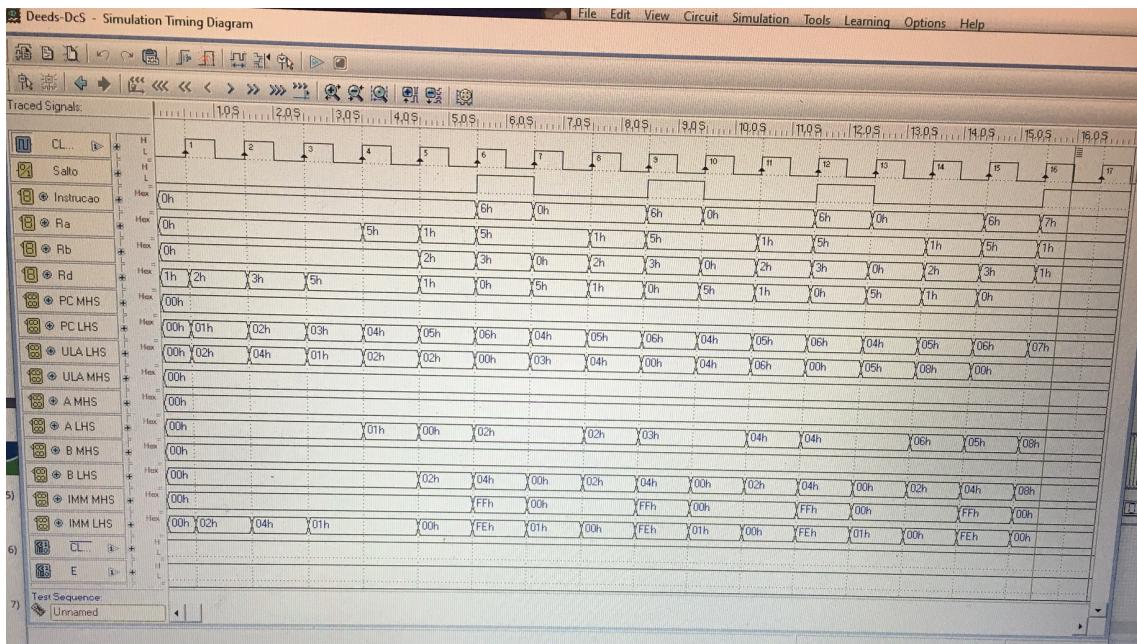


Figura 19. Simulação em forma de onda da multiplicação de dois números sem sinal

3.4. Multiplicação de dois números com sinal

Endereço	Código hexadecimal	Instrução	Comentário
0x0000	0x0000 1000	addi R1,R0,R0,0	R1=0
0x0001	0xXXXX 2000	addi R2,R0,R0,X	R2=X
0x0002	0xYYYY 3000	addi R3,R0,R0,Y	R3=Y
0x0003	0x0003 0027	bles R0,R0,R2,3	R0<=R3 (com sinal) ? Salto1 : Next
0x0004	0x0000 2021	subi R2,R0,R2,0	R2=-R2
0x0005	0x0001 6600	addi R6,R6,R0,1	R6=R6+1
0X0006 Salto1:	0x0003 0037	bles R0,R0,R3,3	R0<=R3 (com sinal) ? Salto2 : Next
0x0007	0x0000 3031	subi R3,R0,R3,0	R3=-R3
0x0008	0x0001 6600	addi R6,R6,R0,1	R6=R6+1
0x0009 Salto2:	0x0001 5500	addi R5,R5,R0,1	R5=R5+1 devido ao =
0x000A Loop:	0x0001 5500	addi R5,R5,R0,1	R5=R5+1
0x000B	0x0000 1120	addi R1,R1,R2,0	R1=R1+R2
0x000C	0xFFFFE 0536	bleu R0,R5,R3,-2	R5<=R3 (com sinal) ? Loop : Next
0x000D	0x0004 0605	beq R0,R6,R0,4	R6==R0 ? J Fim : Next
0x000E	0x0002 4000	addi R4,R0,R0,2	R4 = 2
0x000F	0x0002 0645	beq R0,R6,R4,2	R6==R4 ? J Fim : Next
0x0010	0x0000 1011	subi R1,R0,R1,0	R1=-R1
0x0011 Fim:	0x0000 0115	beq R0,R1,R1,0	J Fim

Tabela 5. Programa de multiplicação de dois números com sinal

O programa de multiplicação de dois números com sinal, disponível na tabela 5, foi simulado nesse vídeo. Esse programa funciona de modo similar ao programa da seção . Primeiramente, cada número é substituído pelo módulo dele, ou seja, multiplicado por -1 se for negativo. Depois, os módulos são multiplicados de acordo com o programa implementado na seção . Observe que o registrador R6 foi utilizado para armazenar a informação dos sinais dos números multiplicados. Assim, se R6=0 ambos os números são positivos e se R6=2 ambos os números são negativos. Em ambos os casos, a saída é positiva, ou seja, o resultado é a multiplicação entre os módulos. Caso R6 seja igual a 1, apenas um dos números é negativo e a saída deve ser negativa, ou seja, o resultado da multiplicação entre os módulos dever ser multiplicado por -1. A simulação em forma de onda é mostrada na figura 20.

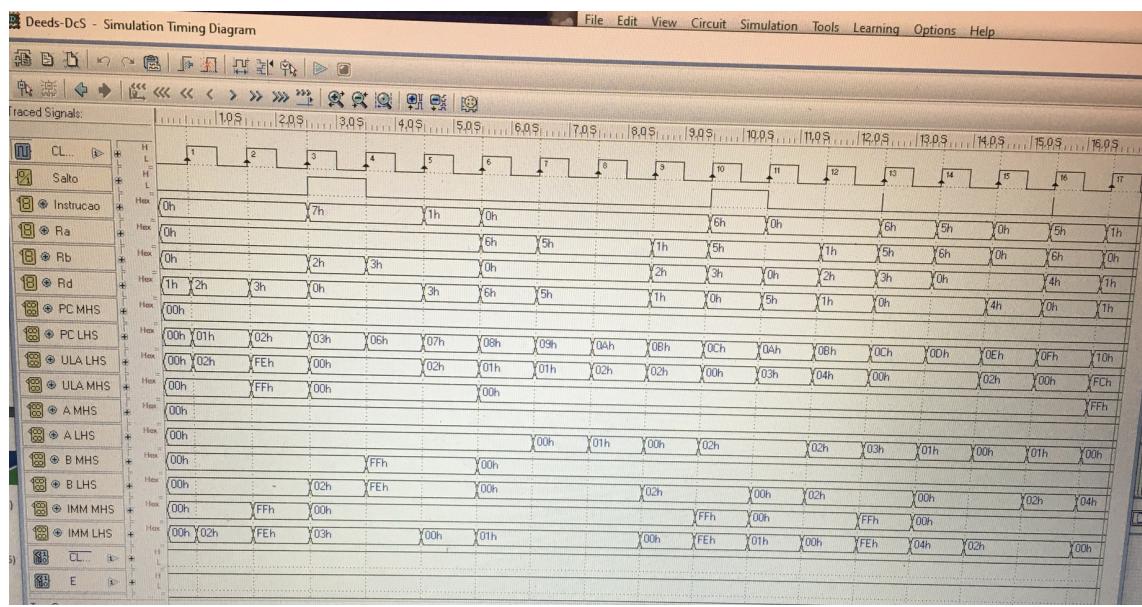


Figura 20. Simulação em forma de onda da multiplicação de dois números com sinal

3.5. Divisão inteira de dois números sem sinal

Endereço	Código hexadecimal	Instrução	Comentário
0x0000	0x0000 1000	addi R1,R0,R0,0	R1=0
0x0001	0xXXXX 2000	addi R2,R0,R0,X	R2=X
0x0002	0xYYYY 3000	addi R3,R0,R0,Y	R3=Y
0x0003 Loop:	0x0000 2231	subi R2,R2,R3,0	R2=R2-R3
0x0004	0x0001 1100	addi R1,R1,R0,1	R1=R1+1
0x0005	0xFFE 0326	bleu R0,R3,R2,-2	R3<=R2 (sem sinal) ? Loop : Fim
0x0006 Fim:	0x0000 0115	beq R0,R1,R1,0	J Fim

Tabela 6. Programa da divisão inteira de dois números sem sinal

O programa de divisão inteira entre dois números sem sinal, disponível na tabela 6, foi simulado nesse vídeo. O primeiro número é subtraído pelo segundo número, adicionando 1 ao resultado R1 a cada ciclo, até que o primeiro número seja menor ou igual ao segundo número. A simulação em forma de onda é mostrada na figura 21.

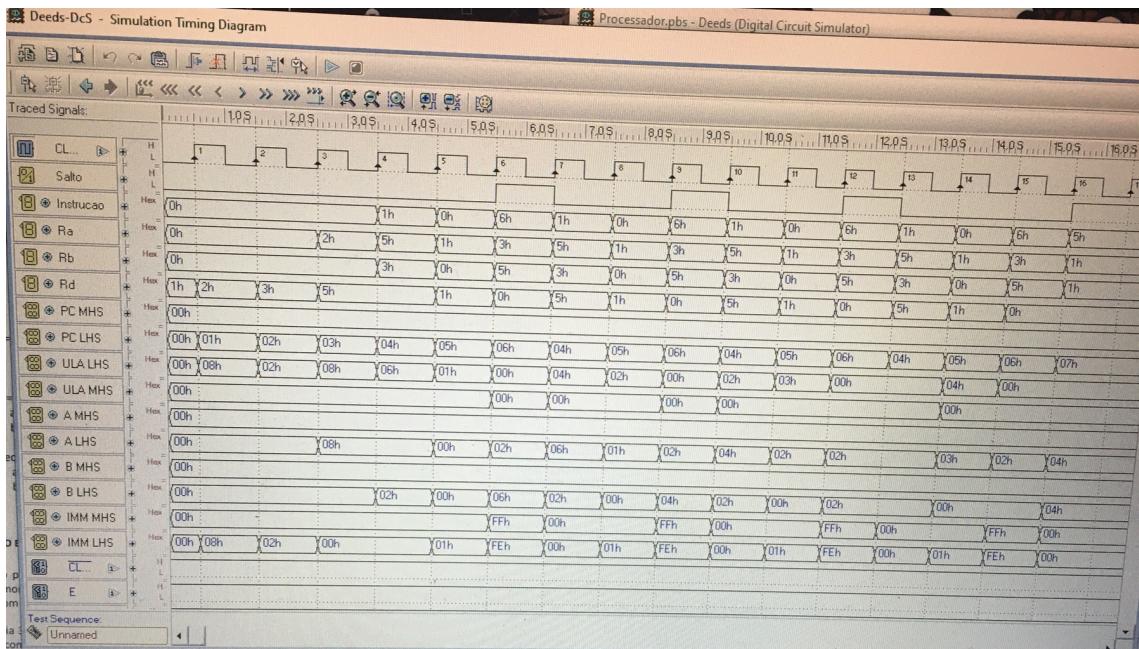


Figura 21. Simulação em forma de onda da divisão inteira de dois números sem sinal

3.6. Resto da divisão inteira de dois números sem sinal

Endereço	Código hexadecimal	Instrução	Comentário
0x0000	0x0000 1000	addi R1,R0,R0,0	R1=0
0x0001	0xXXXX 2000	addi R2,R0,R0,X	R2=X
0x0002	0xYYYY 3000	addi R3,R0,R0,Y	R3=Y
0x0003	0x0000 1200	addi R1,R2,R0,0	R1=R2
0x0004 Loop:	0x0000 1131	subi R1,R1,R3,0	R1=R1-R3
0x0005	0xFFFF 0316	bleu R0,R3,R1,-1	R3<=R1 (sem sinal) ? Loop : Next
0x0006 Fim:	0x0000 0115	beq R0,R1,R1,0	J Fim

Tabela 7. Programa do resto da divisão inteira de dois números sem sinal

O programa de resto da divisão inteira de dois números sem sinal, disponível na tabela 7, foi simulado nesse vídeo. O valor do primeiro número é armazenado no resultado e depois o segundo número é subtraído desse resultado até que o resultado seja menor ou igual ao segundo número. A simulação em forma de onda é mostrada na figura 22.

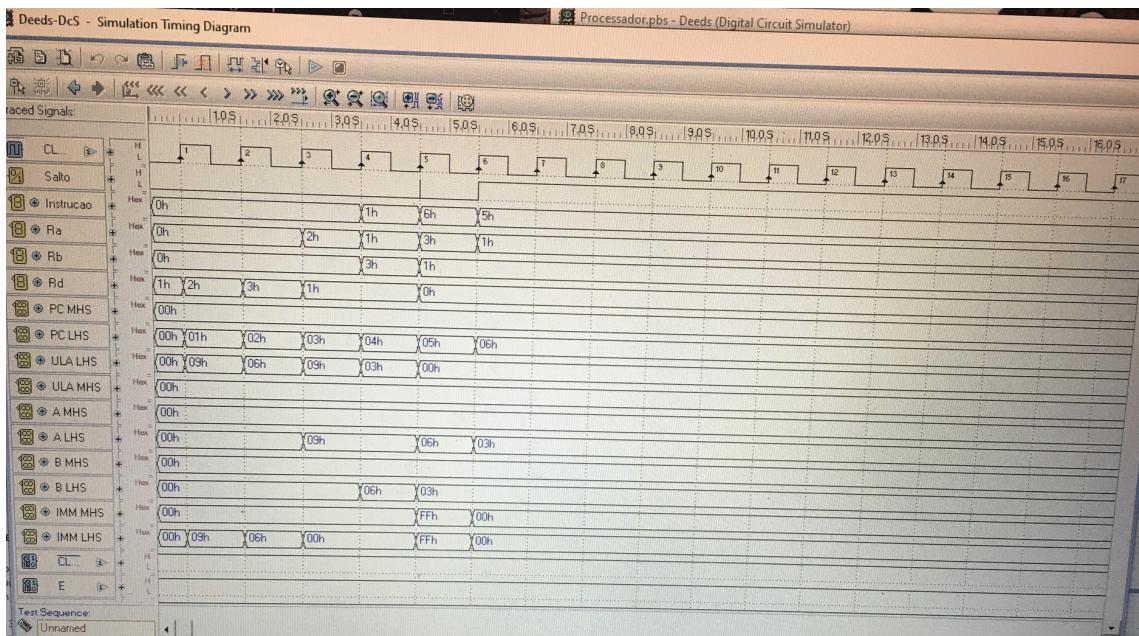


Figura 22. Simulação em forma de onda do resto da divisão inteira de dois números sem sinal

3.7. Programa do professor

O programa disponibilizado pelo professor no Aprender3 foi simulado nesse vídeo e a simulação em forma de onda é mostrada na figura 23.

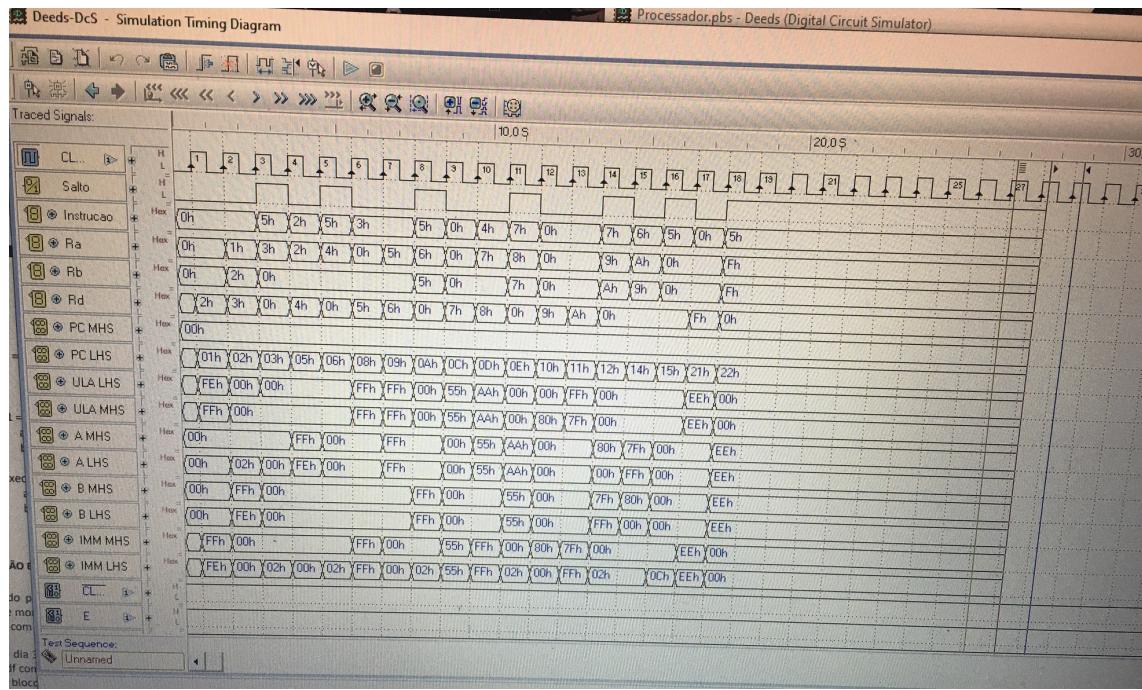


Figura 23. Simulação em forma de onda do programa do professor

4. Conclusão

Concluindo, foi implementado um processador capaz de receber um programa com instruções de 32 bits escritas em linguagem de máquina e armazenadas em uma memória ROM sendo que essas instruções envolvem operações entre valores armazenadas em um banco de registradores e um número imediato que não está armazenado em nenhum registrador. Esse processador pode realizar 8 operações distintas: adição, subtração, and bitwise, or bitwise e xor bitwise, que são calculadas com o auxílio de uma unidade lógico aritmética, além de operações de salto condicional com e sem sinal, que são definidas com o auxílio de um comparador de dois números de 16 bits.

Alguns programas foram escritos e testados nesse circuito, entre eles: um contador de -16 a 16, um somador de números ímpares entre 0 e 15, um multiplicador de dois números tanto sem quanto com sinal e programas que mostram o resto e o resultado da divisão inteira entre dois números. Observou-se que o circuito funciona corretamente somente se a frequência de relógio for no máximo 384,62 MHZ devido ao atraso do circuito e é válido notar também que algumas instruções como a adição são bem mais utilizadas na escrita dos programas do que operação como andi e ori.

Em suma, o processador implementado é relativamente simples e não possui instruções de escrita e leitura que exigiriam o uso de memórias RAM para a implementação de uma memória de dados. Mesmo assim, o circuito montado e os programas testados permitem o entendimento do funcionamento básico de um computador, pois foram usados componentes presentes em computador real para montagem dos circuitos, mesmo que só em uma simulação, e linguagem baixo nível para elaboração dos programas requisitados.

Referências

- [Deeds] Deeds. Deeds: Digital electronics education and design suite. <https://www.digitalelectronicsdeeds.com/deeds.html>.
- [EPUSP 2021] EPUSP (2021). Banco de registradores e ula. http://www2.pcs.usp.br/~labdig/pdffiles_2015/banco-registradores-ula.pdf. [Online; accessed 18-May-2021].
- [Khatib 2014] Khatib, M. (2014). *Aging Analysis of Datapath Sub-blocks Based on CET Map Model for Negative Bias Temperature Instability (NBTI)*. PhD thesis.
- [Wikipedia 2020a] Wikipedia (2020a). Machine code. https://en.wikipedia.org/wiki/Machine_code. [Online; accessed 18-May-2021].
- [Wikipedia 2020b] Wikipedia (2020b). Read only memory. https://en.wikipedia.org/wiki/Read-only_memory. [Online; accessed 18-May-2016].
- [Wikipedia 2020c] Wikipedia (2020c). Unidade central de processamento. https://pt.wikipedia.org/wiki/Unidade_central_de_processamento. [Online; accessed 18-May-2021].
- [Youtube 2020] Youtube (2020). Construct a 16*1 multiplexer with two 8*1 and one 2*1 multiplexers. <https://www.youtube.com/watch?v=1DKEaWzoUJU>. [Online; accessed 18-May-2021].