Relatório – Trabalho

Projetos Digitais e Microprocessadores

Processador RISC-V

Aluna:

Eloiza Sthefanny Rocha da Silva Cardoso

Professor:

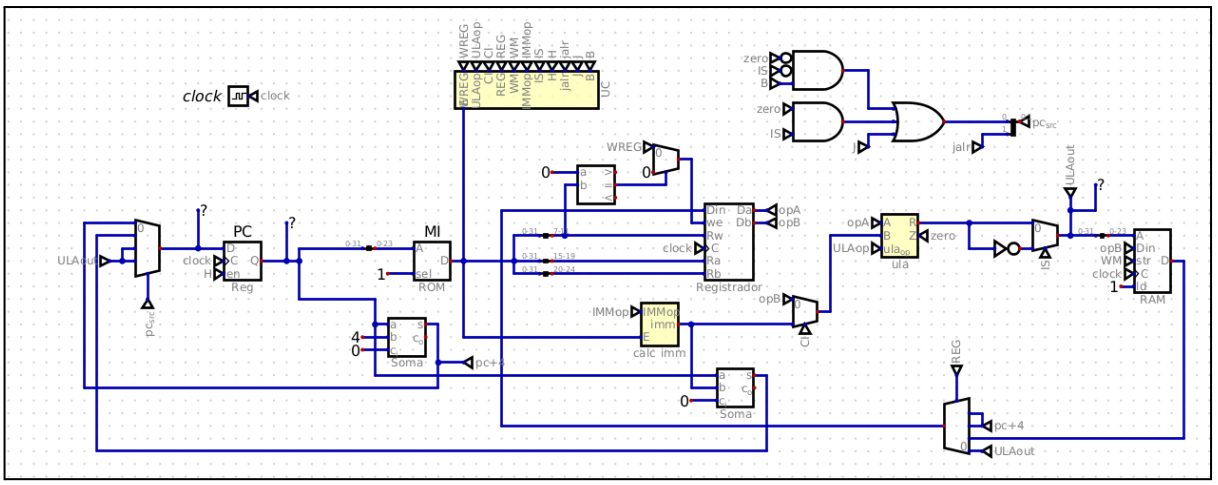
Daniel Oliveira

• INTRODUÇÃO

O objetivo do trabalho é implementar uma versão monociclo (single cycle) de 32 bits para o subconjunto da ISA RISC-V que realiza instruções e operações do sistema. O máximo de registadores que cada instrução pode utilizar são três, e a manipulação de imediatos (constantes). São baseados em seis formatos de instrução, do tipo R, operação com os registradores, do tipo I, operações com registradores e imediatos, do tipo S, salvar na memória, do tipo B, que são os condicionais e do tipo J, instruções de salto.

• PRIMEIRA ETAPA

O primeiro passo iniciado no trabalho foi implementar a base do processador, cálculo do PC tanto com 4 quanto com o endereço com a soma do imediato. Logo após coloquei a saída do PC, que é um registrador, na memória de instrução, que vai pegar o conteúdo que está contido no endereço que o PC forneceu. O conteúdo da memória ROM vai para a UC (unidade de controle), para separar em outras operações e controlar todos os processos do processador. Usei o Banco de Registradores disponibilizado pelo digital, onde o Rw é o rd, o registrador que é escrito, onde o Ra é o rs1 usados nas instruções e o Rb o rs2 com a mesma funcionalidade do rs1, cada um desses registradores com 5 bits.
Além disso, usei um comparador para verificar se o Rw irá receber o registrador que não pode ser escrito, portanto a entrada que controla se irá escrever ou não no registrador estará desligada. No projeto geral do processador, também há um calculador de imediato, conforme cada tipo de comando, uma Unidade Lógica Aritmética (ULA), um mux que controla se vai usar imediato na ULA ou não e outro mux se a saída da ula vai ser invertida ou não, outros mux's importantes são do seletor REG que controla qual será o resultado armazenado no registrador e do PC_src, que controla qual endereço será usado conforme cada instrução.



Visão geral do processador RISC-V.

• UNIDADE DE CONTROLE

A unidade de controle tem como entrada 32 bits que são posteriormente separados para obter o opcode do comando, o funct3 e funct7 que são importantes para identificar cada necessidade de utilização do processador. Além disso, possui onze saídas, sendo elas: WREG com 1 bit que indica se irá escrever ou não no registrador, ULAop com 3 bits que decide qual operação a ula irá realizar, CI que controla se irá ser usado imediato ou valor de registrador na ULA, REG com 2 bits que decide qual resultado irá gravar no registrador, se busca da memória, o salto de instrução ou o valor da saída da ULA, WM com 1 bit que indica se o valor será guardado ou não na memória, IMMop com 2 bits que indica qual tipo de imediato será utilizado, IS com 1 bit e decide se o sinal será invertido ou não e H que será o Halt, o identificador para parar o programa, ou seja, o comando de parada vai ter o bit mais significativo como 1 e o restante como 0, como o bit menos significativo de todos os opcodes são 1's, será parado o programa quando chegar no bit menos significativo igual a 0. Outros são B, J e jalr que são comandos que serão usados no projeto.

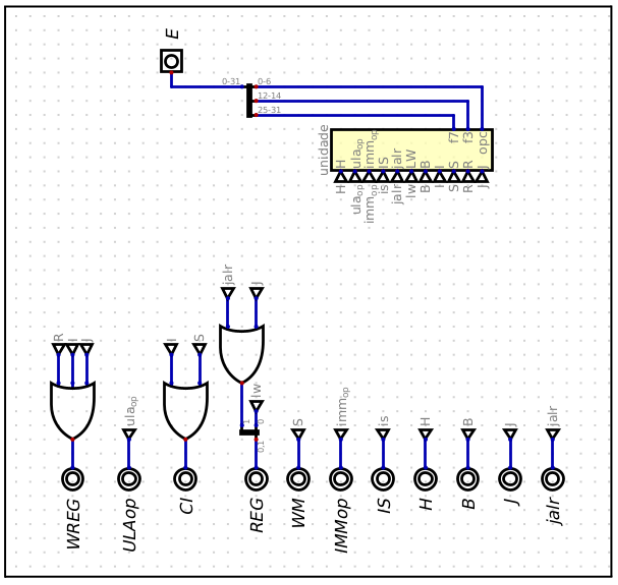


Tabela de saídas conforme a instrução

	Inst.	WREG	ULA op	CI	REG	WM	IMM op	IS	H	B	J	Jalr
R	ADD	1	add	0	00	0	XX	0	1	0	0	0
	SUB	1	subt	0	00	0	XX	0	1	0	0	0
	XOR	1	xor	0	00	0	XX	0	1	0	0	0
	OR	1	or	0	00	0	XX	0	1	0	0	0
	AND	1	and	0	00	0	XX	0	1	0	0	0
	SLL	1	sll	0	00	0	XX	0	1	0	0	0
	SLT	1	comp <	0	00	0	XX	0	1	0	0	0
I	ADDI	1	add	1	00	0	00	0	1	0	0	0
	XORI	1	xor	1	00	0	00	0	1	0	0	0
	ORI	1	or	1	00	0	00	0	1	0	0	0
	ANDI	1	and	1	00	0	00	0	1	0	0	0
	SLLI	1	sll	1	00	0	00	0	1	0	0	0
	SLTI	1	comp <	1	00	0	00	0	1	0	0	0
	LW	1	add	1	01	0	00	0	1	0	0	0
S	SW	0	add	1	XX	1	01	0	1	0	0	0
B	BEQ	0	comp =	0	XX	0	10	0	1	1	0	0
	BNE	0	comp =	0	XX	0	10	1	1	1	0	0
	BLT	0	comp <	0	XX	0	10	0	1	1	0	0
	BGE	0	comp <	0	XX	0	10	1	1	1	0	0
J	JAL	1	X	X	10	0	11	0	1	0	1	0
I	JALR	1	add	1	10	0	00	0	1	0	0	1
H	HALT	X	X	X	X	X	X	X	0	X	X	X

operações	ula op
add	000
sall	001
comp =	010
comp <	011
xor	100
subt	101
or	110
and	111

↳ Saída REG

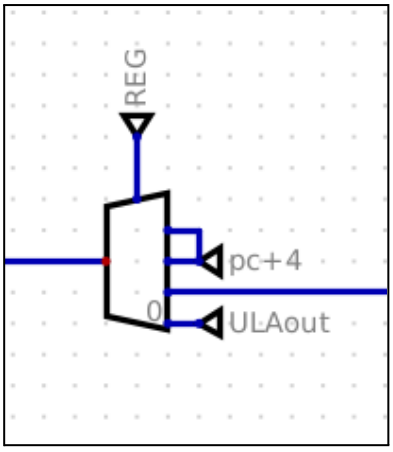
R + Isol ⇒ 00

J + Jalr ⇒ 10

LW ⇒ 01

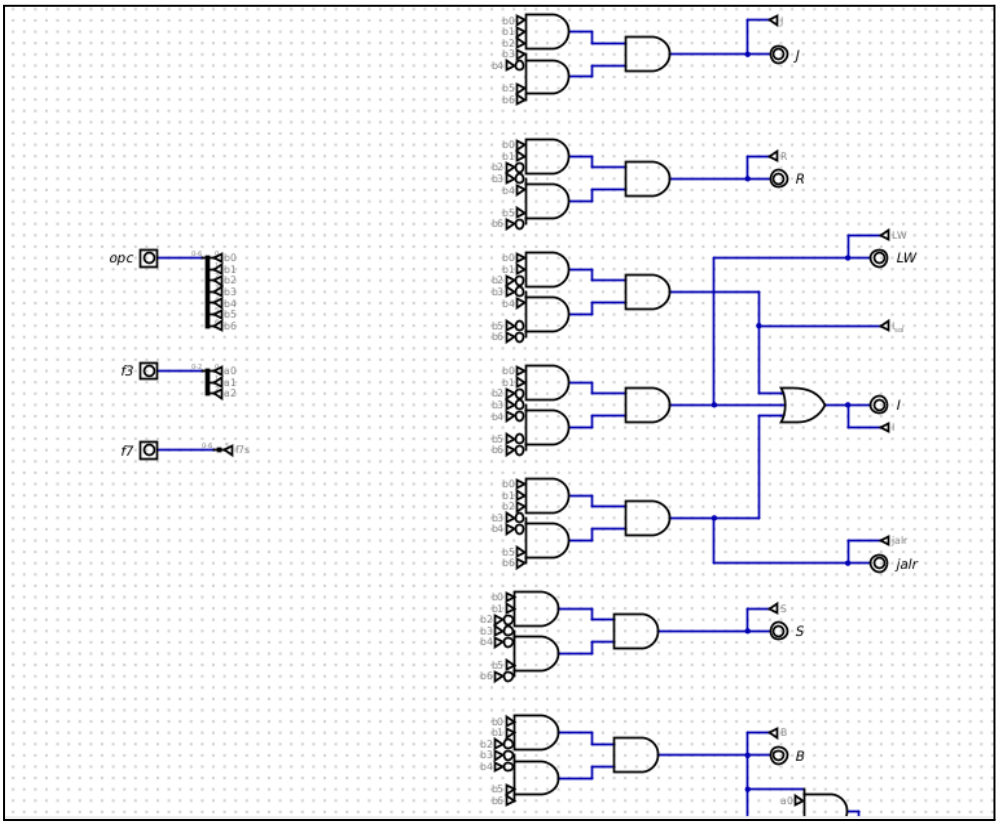
↳ Colocando na tabela verdade

R / Isol	J / jalr	LW	REG
1	0	0	00
0	0	1	01
0	1	0	10

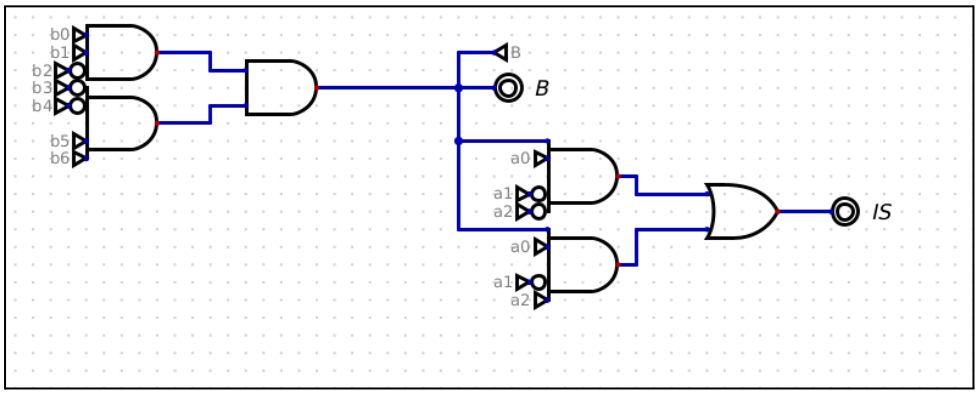


- Unidade

Dentro da Unidade de controle têm outro conjunto de funções onde foi identificado cada tipo de comando e realizado a sintetização de cada instrução. Foi separado os 7 bits do opcode bit a bit, de 3 bits do funct e apenas o bit que muda no funct 7, que é o 5, pois todos os restantes são zeros para todos. Com isso foi separado o tipo R, I, J, jalr LW, S e B, tudo feito por identificação de sinal.



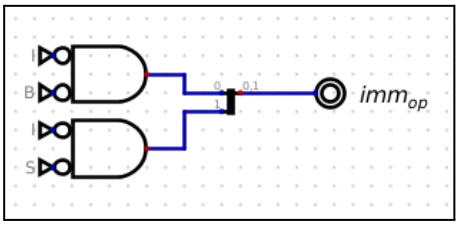
Também foi usado o identificador de sinal para identificar o tipo de instrução que invertia o valor, pois tanto para BEQ quanto para BNE foi usado o comparador de igual na ula, mas o bne é para saber se é diferente, portanto é necessário a inversão. Portanto, logo após identificar o tipo B, foi aplicado outra identificação de sinal para saber se era igual a determinado funct 3, assim seria aplicado a inversão e o IS valerá 1.



Logo após usar cada identificação, foi feita a saída que possibilita selecionar qual tipo de imediato vai ser utilizado, conforme a tabela e o circuito fornecido.

Tabela saída IMM_{op}

I	S	B	J	IMM_{op}
1	0	0	0	00
0	1	0	0	01
0	0	1	0	10
0	0	0	1	11



Para a saída da ULAop foi usada a estratégia de separar cada tipo, ou seja, fazer um seletor de mux que direciona cada operação. Se for do tipo S, LW, J ou Jalr a saída que irá aparecer será o 000, pois é a operação de adição da ula, depois for separado em R + Isol (tipo I dos imediatos) e em B.

A = S + JALR + J + LW
B = A + Isol
C = B

A	B	C	S_1 S_0
1	0	0	0 0
0	1	0	0 1
0	0	1	1 0

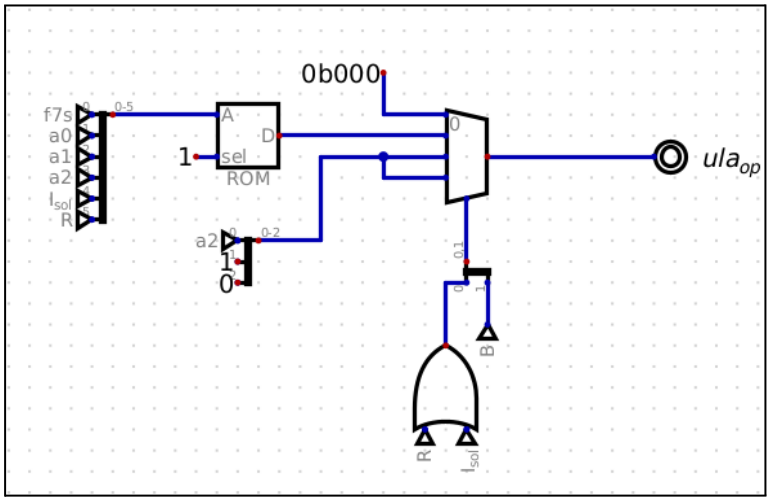
Quando o seletor for 00, a saída será 000, quando for 01, usará um memória ROM que o número binário da tabela foi transformado em endereços de 5 bits, em hexadecimal, que neles foram armazenados a saída da ula de 3 bits, também em hexadecimal.

	R	Isol	F3	F7	ULA op
ADD	1	0	000	0	0 0 0
SUB	1	0	000	1	1 0 0
XOR	1	0	100	0	1 0 0
OR	1	0	110	0	1 1 0
AND	1	0	111	0	1 1 1
SLL	1	0	001	0	0 0 1
SLT	1	0	010	0	0 1 1
ADDI	0	1	000	0	0 0 0
XORI	0	1	100	0	1 0 0
ORI	0	1	110	0	1 1 0
ANDI	0	1	111	0	1 1 1
SLLI	0	1	001	0	0 0 1
SLTI	0	1	010	0	0 0 1

No seletor 10 foi utilizado para o tipo B uma tabela verdade e identificado que a mudança dependia do bit mais significativo, ou seja, as outras saídas são constantes para todos.

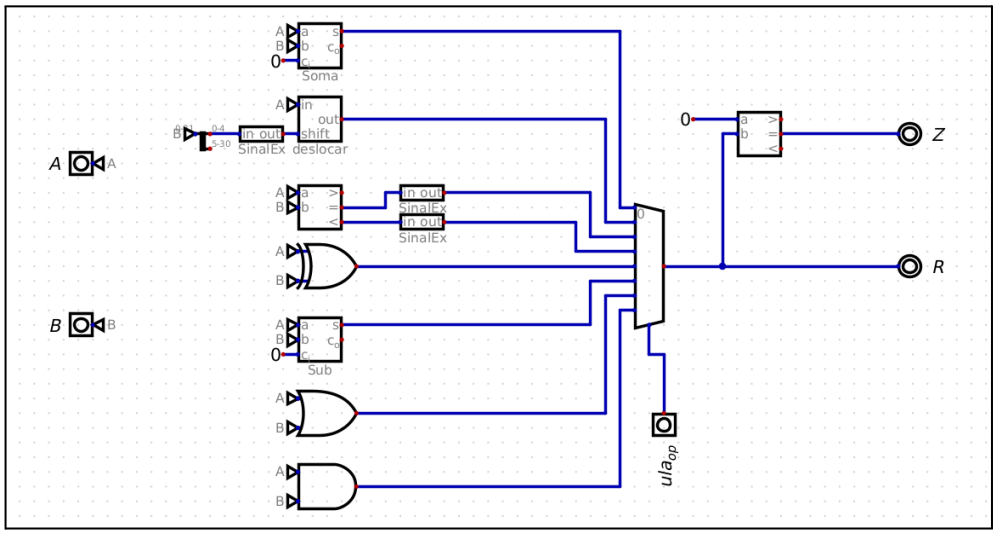
	F 3	ULA_OP
BEQ	0 0 0	0 1 0
BNE	0 0 1	0 1 0
BLT	1 0 0	0 1 1
BGE	1 0 1	0 1 1

Logo após, foi conectado nas entradas do mux obtendo assim cada saída necessária.



• UNIDADE LÓGICA E ARITMÉTICA

Para a ULA foram utilizadas 8 entradas, sendo elas: Soma, shift, comparação de igual, menor, xor, subtração, and e or, e duas saídas, o zero com 1 bit e o R que possui 32 bits . O seletor de cada função foi escolhido conforme o funct3 para melhor simplificação do circuito.

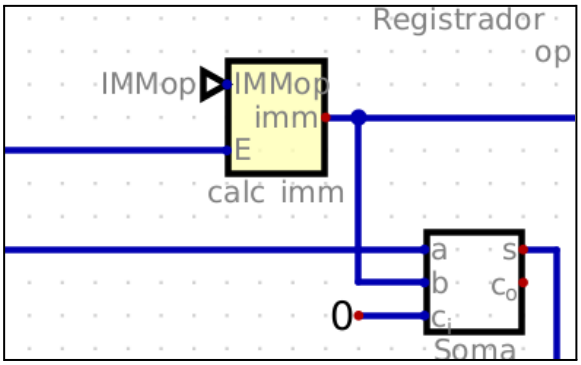


operações	ula op
add	000
sall	001
comp =	010
comp <	011
xor	100
srall	101
or	110
and	111

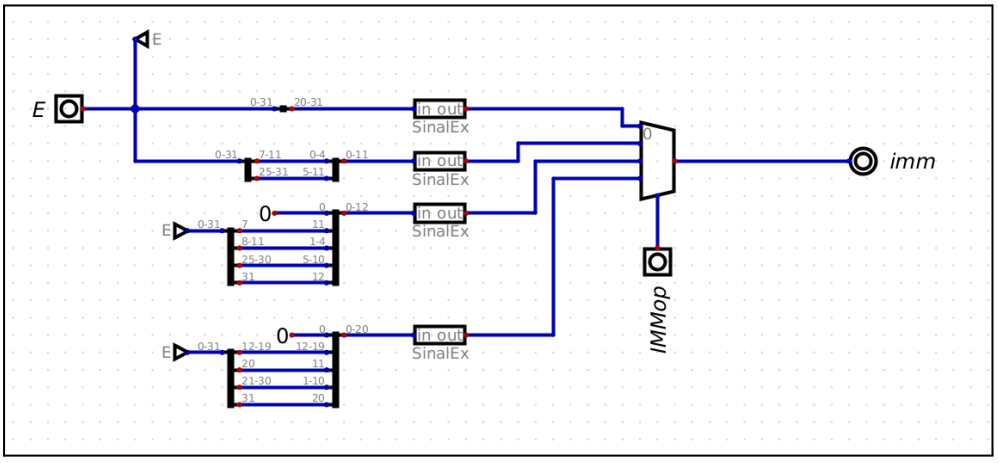
• CALCULADOR DE IMEDIATO

Outra parte do processador é o calculador de imediato que recebe como entrada os 32 bit de comando e o IMMop que seleciona qual tipo de imediato será utilizado.

31 27	26 25	24 20	19 15	14 12	11 7	6 0	
funct7		rs2	rs1	funct3	rd	opcode	R-type
imm[11:0]			rs1	funct3	rd	opcode	I-type
imm[11:5]		rs2	rs1	funct3	imm[4:0]	opcode	S-type
imm[12|10:5]		rs2	rs1	funct3	imm[4:1|11]	opcode	B-type
imm[20|10:1|11|19:12]					rd	opcode	J-type

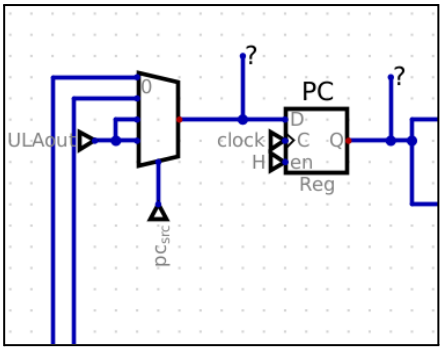


Os imediatos são usados pegando cada bit indicado na tabela de formatos de instrução dos tipos de imediatos, para isso foi usado distribuidores e extensão de sinal, pois os imediatos possuem de 12 a 13 bits.



• ENTRADAS DO PC

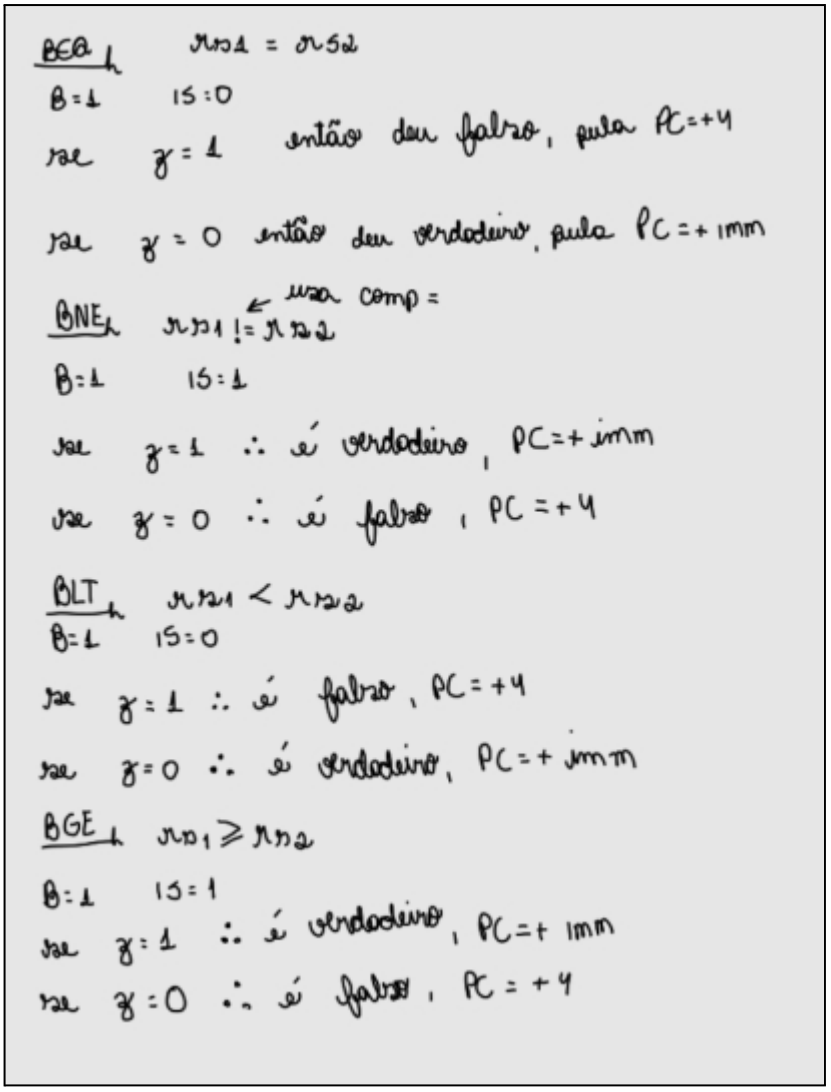
Quando a instrução é do tipo branch, jal e jalr, é necessário que o endereço pule algumas linhas diferente das convencionais de pular sempre para o endereço depois, por isso é necessário fazer um seletor que decida para qual instrução o próximo endereço vai realizar.



Quando o seletor PCsrc for 00, é porque a função irá pular normalmente para as operações seguintes, ou seja, o PC + 4, agora quando for 01 irá pular o PC + imm, pois estará realizando a função de salto do tipo J, ou do tipo B que é o condicional. Mas quando o seletor for 10 realizará o salto com a soma do rs1 + imm, ou seja, o resultado da ula da operação do jalr.

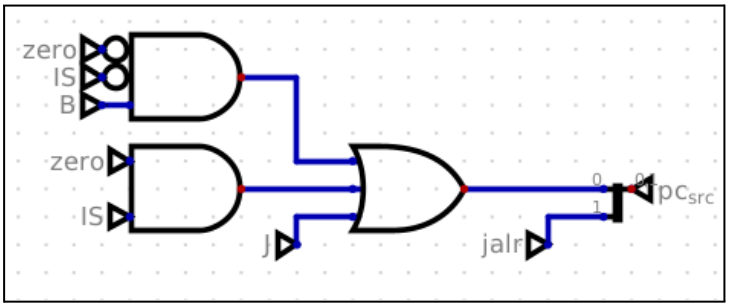
					PC_src	
B	IS	Z	J	JALR	S_1	S_0
0	0	0	0	0	0	0
0	0	0	0	1	1	0
0	0	1	0	1	1	0
0	0	1	0	0	0	0
0	0	1	1	0	0	1
0	0	0	1	0	0	1
1	0	0	0	0	0	1
1	1	1	0	0	0	1
1	0	1	0	0	0	0
1	1	0	0	0	0	0

Ao avaliar, especialmente, os casos de branch, é preciso ter atenção para a decisão do seletor.



Ou seja, quando IS = 0 e Z = 0, usa o PC + imm, mas quando o Z = 1, deu falso, então continua para instrução seguinte, ou seja, PC + 4.

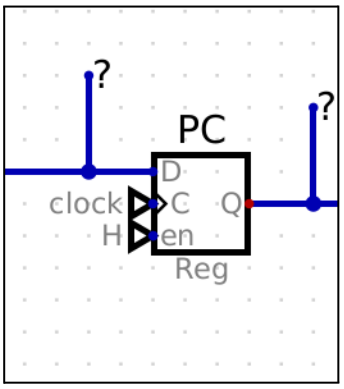
Quando IS = 1 e Z = 1, quer dizer que deu verdadeiro, pois o sinal da ula inverte e o endereço vai ser PC + imm, mas quando o Z = 0 quer dizer que deu falso então vai para PC + 4.



Circuito simplificado do PCsrc.

• **HALT**

Como pontuado anteriormente o halt tem a função de parar o programa, para isso o valor de sua instrução escolhida foi 1 no bit mais significativo e o restante zero, então foi ligado no enable do PC, pois quando o H for 0 nenhuma função será mais executada.



• **CONCLUSÃO**

Concluído que todos os requisitos do trabalho foram cumpridos e que o processador RISC-V o relatório foi elaborado com os principais aspectos detalhados da execução de cada etapa do trabalho.