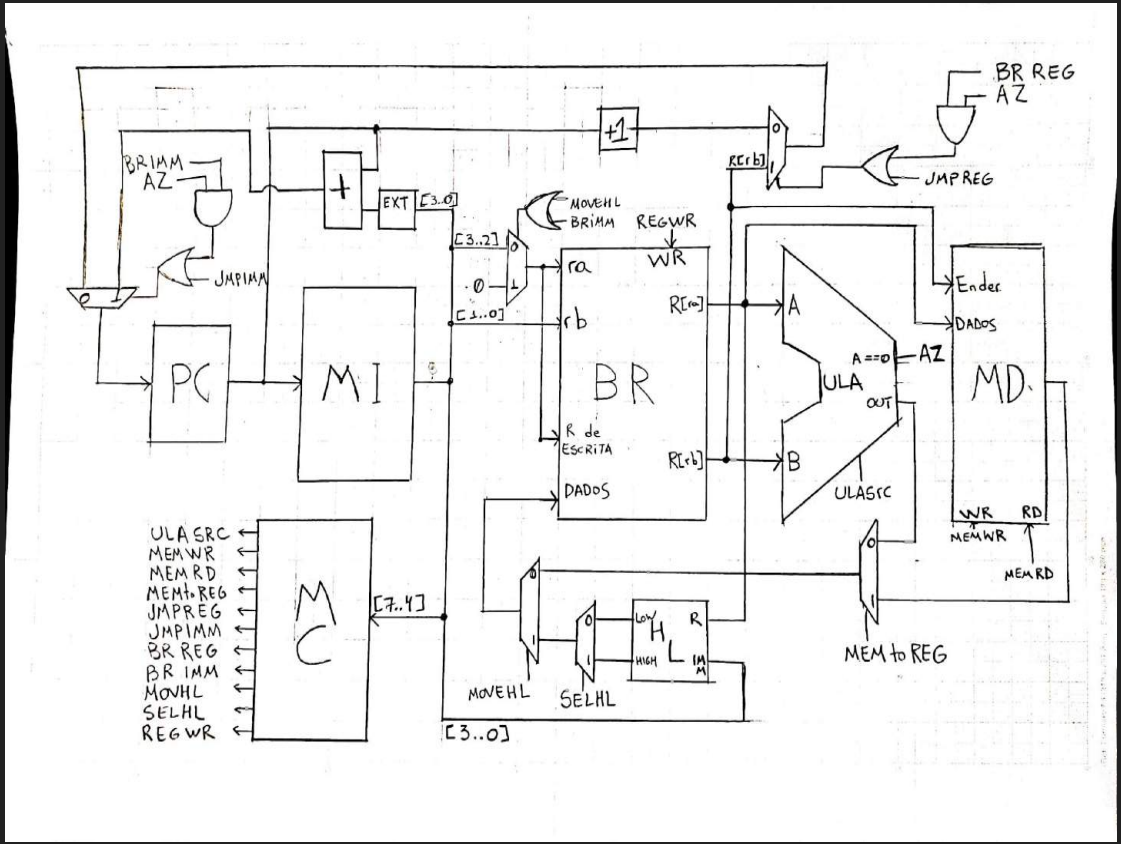
Sagui

CI1212 - Arquitetura de Computadores - ERE2

Jorge Lucas Vicilli Jabczenski

Projeto do Processador

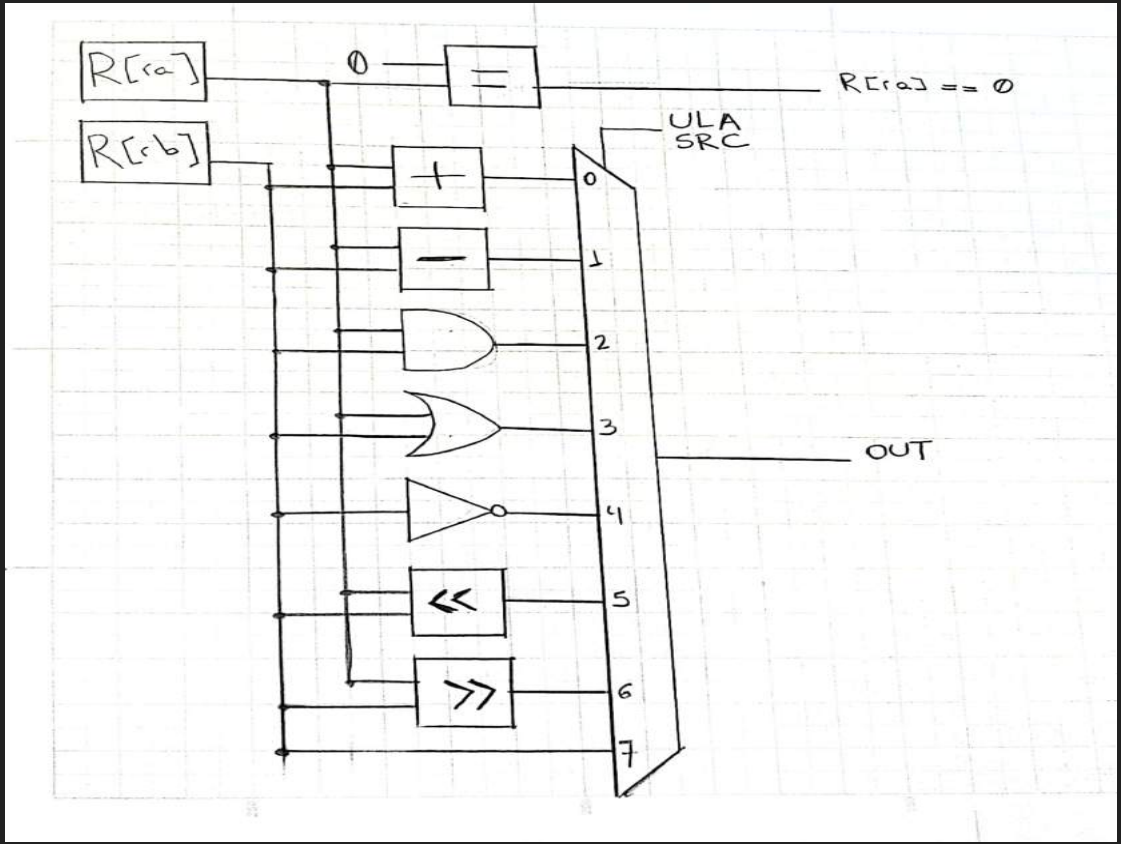


HL: Circuito responsável pelas operações *movh* e *movl*

AZ: Sinal quando a entrada A da ULA é zero. Usada nas operações de *Branch*.

EXT: Transforma os 4 bits do Immediato em 8 para poder realizar a soma

ULA

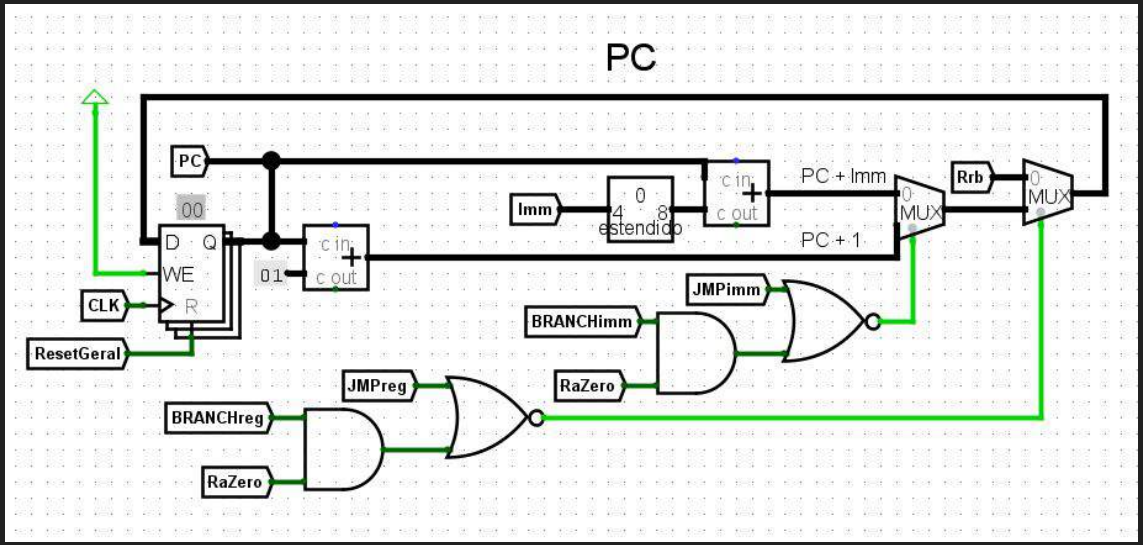


R[ra], R[rb] e saída da ULA possuem 8 bits, enquanto o sinal R[ra]==0 possui apenas 1

ULASRC é o selecionador de operação e vem diretamente da Memória de Controle. Possui 3 bits

A última entrada do MUX é apenas R[rb] pois é assim que a operação *move register* foi implementada

PC



Aqui temos o PC do circuito de uma forma mais compacta para melhor visualização.

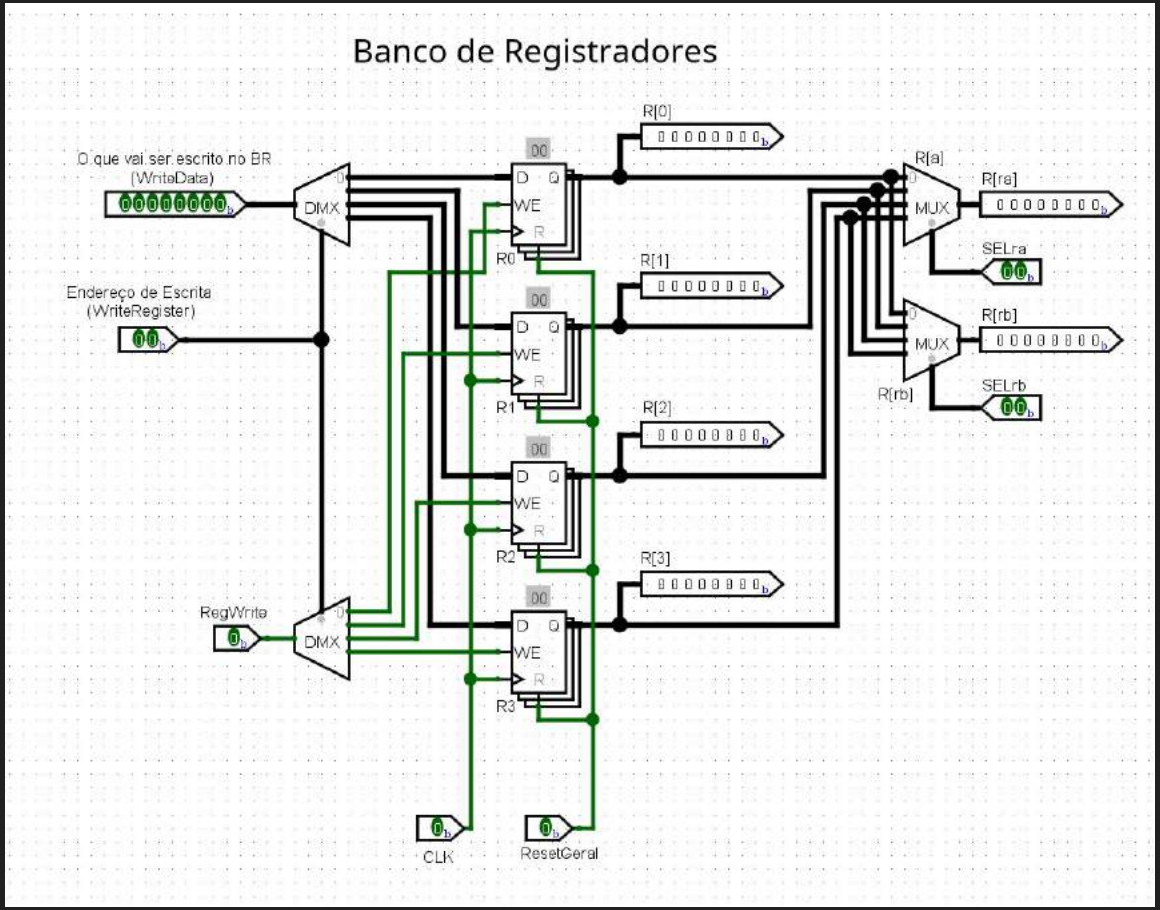
O PC tem 3 resultados possíveis:

PC = PC+1

PC = PC+Imm

PC = R[rb]

Banco de Registradores

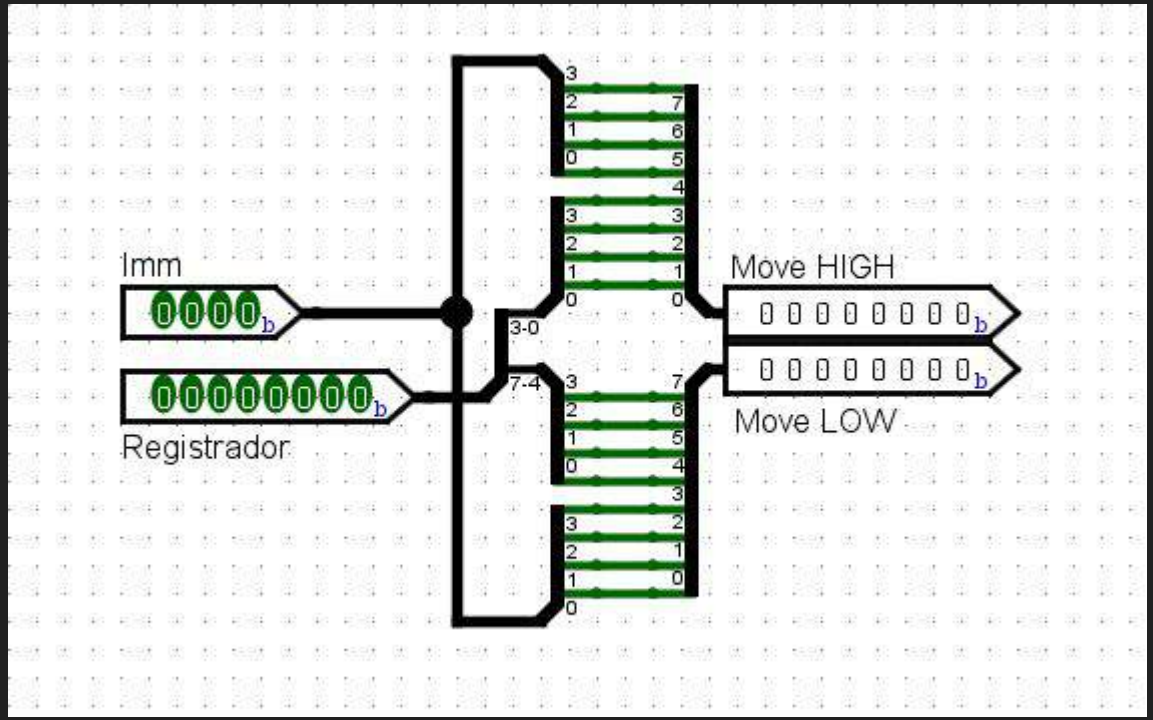


O banco de registradores foi implementado de modo que dois registradores podem ser acessados em paralelo para poder implementar as instruções requisitadas no projeto, tais como:

add R[ra], R[ra], R[rb]

M[R[rb]] = R[ra]

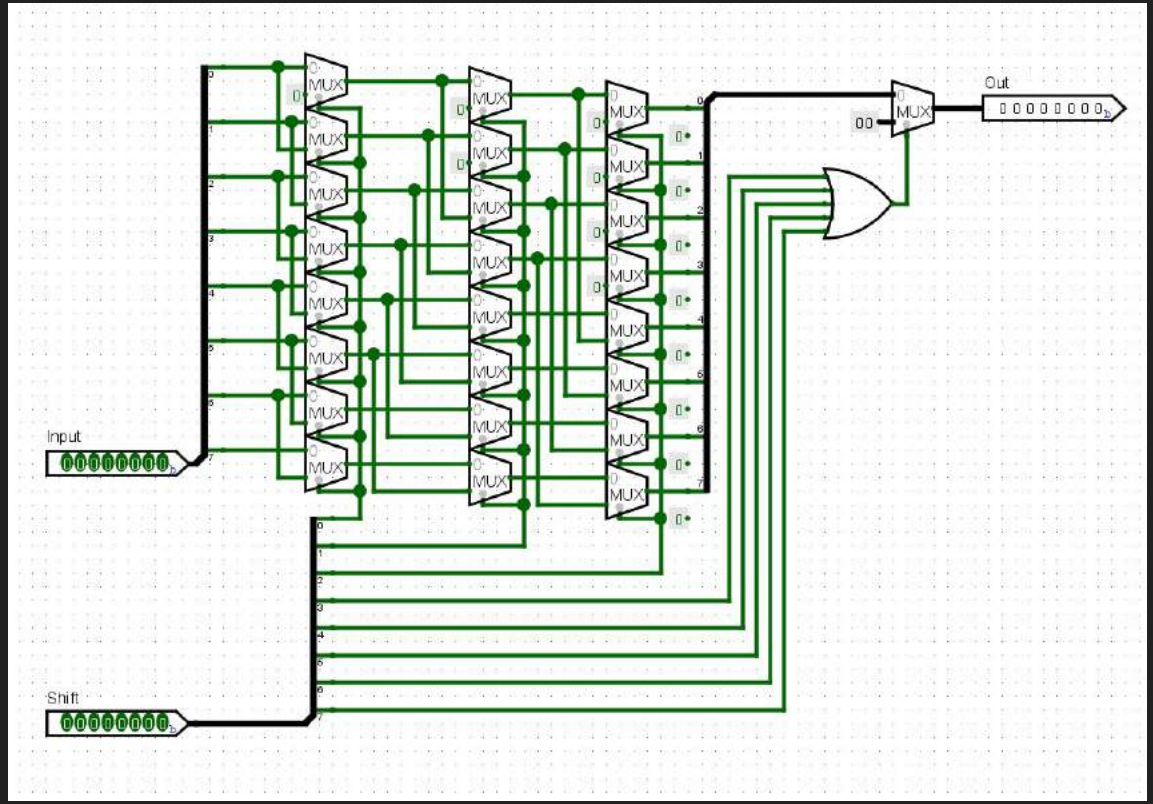
“Move High\Low”



Um circuito extremamente simples, apenas com o objetivo de implementar as duas instruções:

moveH e *moveL*

RBS & LBS
(right bit shifter & left bit shifter)

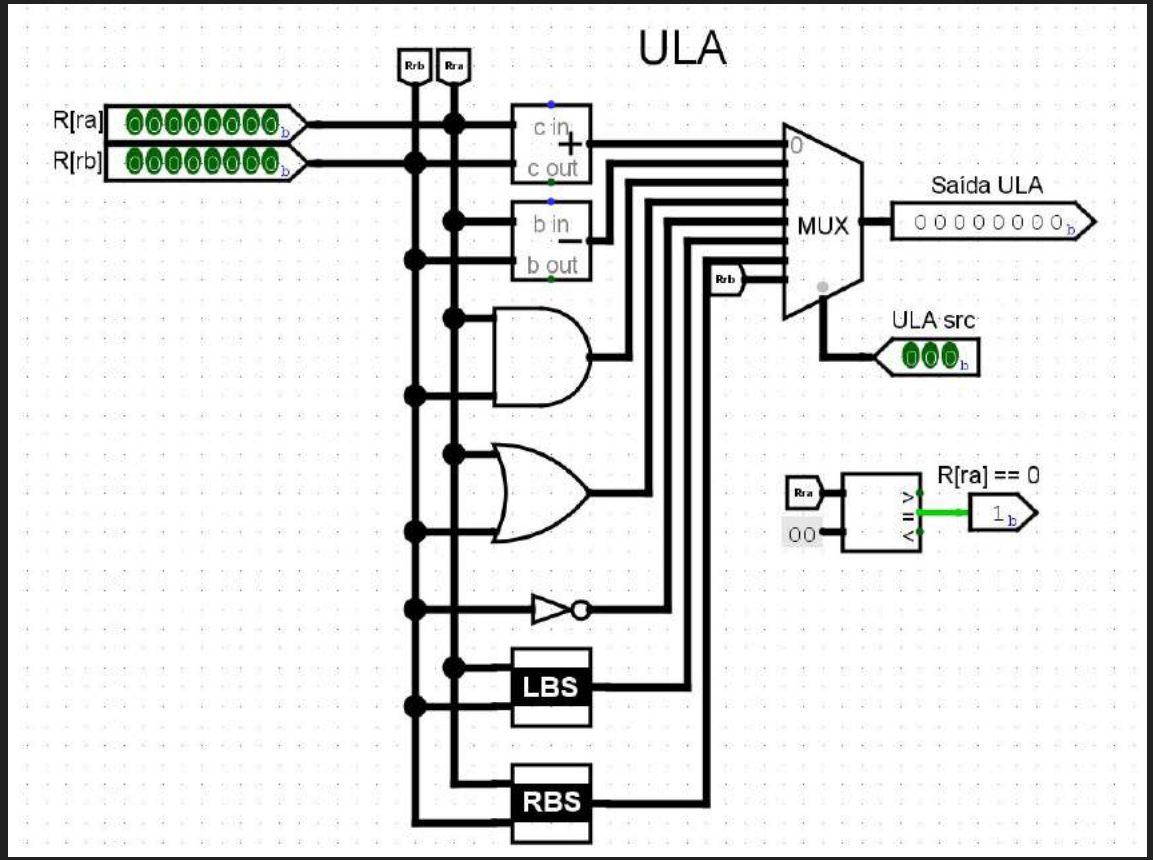


Para implementar as operações de SLR e SRR, foi necessária a construção de um shifter “personalizado” para evitar o desperdício de multiplexadores.

Então, quando o valor na entrada ***shift*** é maior ou igual a 8, automaticamente o sinal enviado é zero. Isso vale tanto para o LBS quanto para o RBS

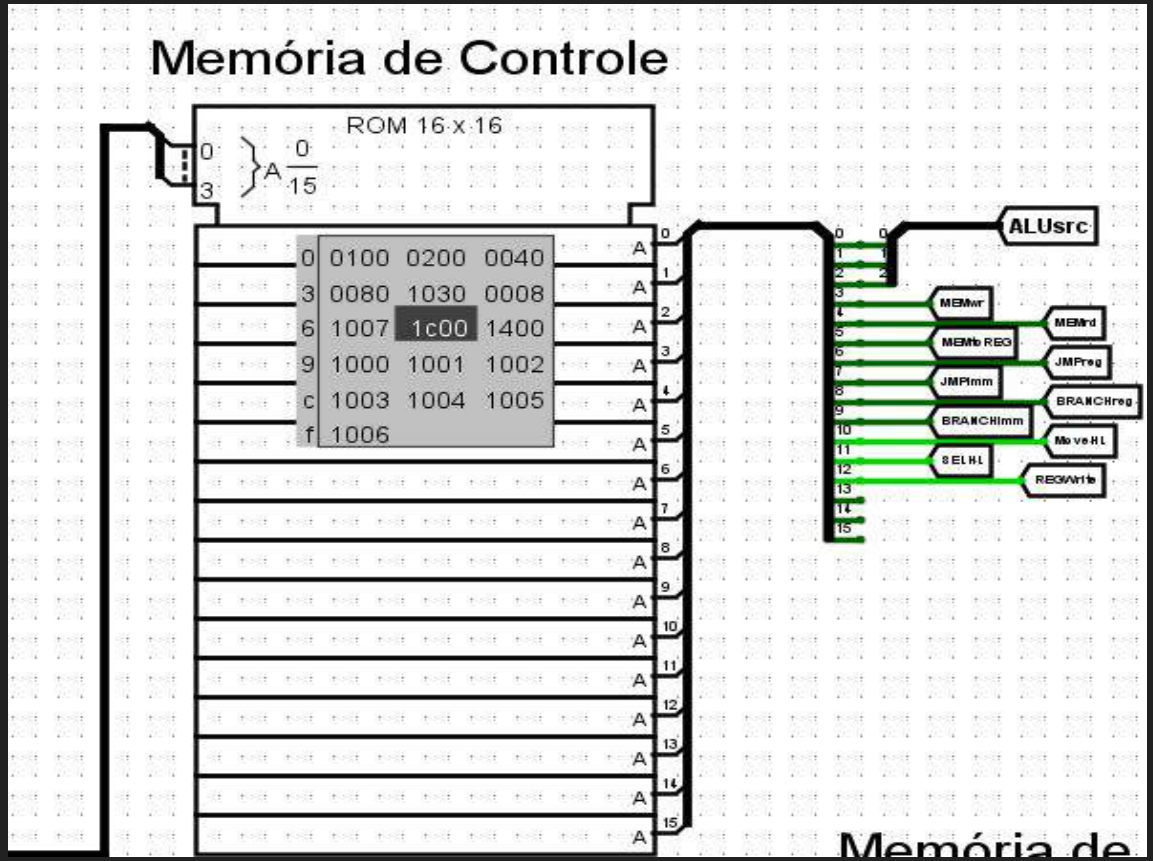
Na imagem ao lado temos um LBS

ULA



Aqui pode ser visto que a Ula implementada é extremamente parecida com a projetada, fazendo uso do LBS e RBS citados no slide anterior

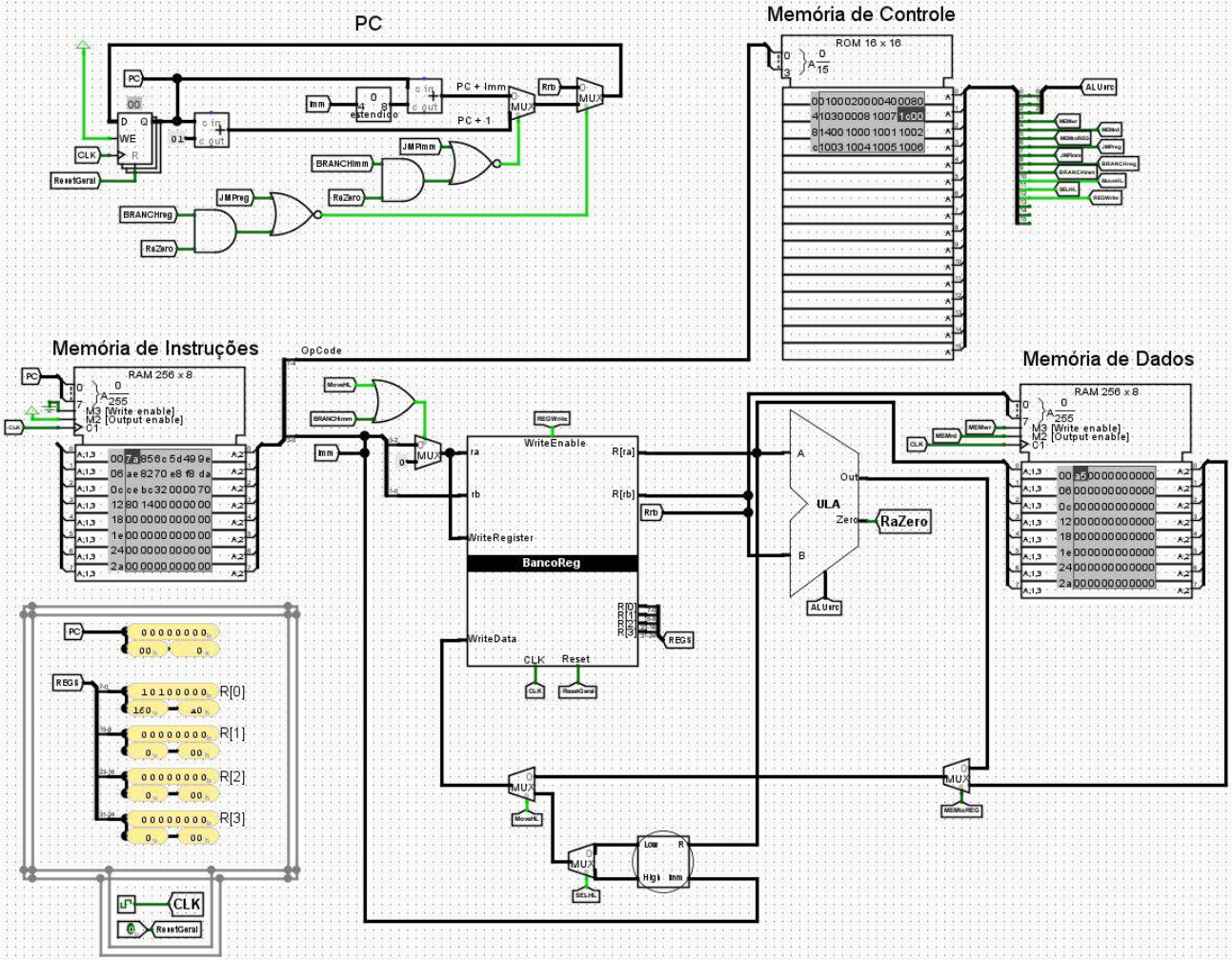
Memória de Controle



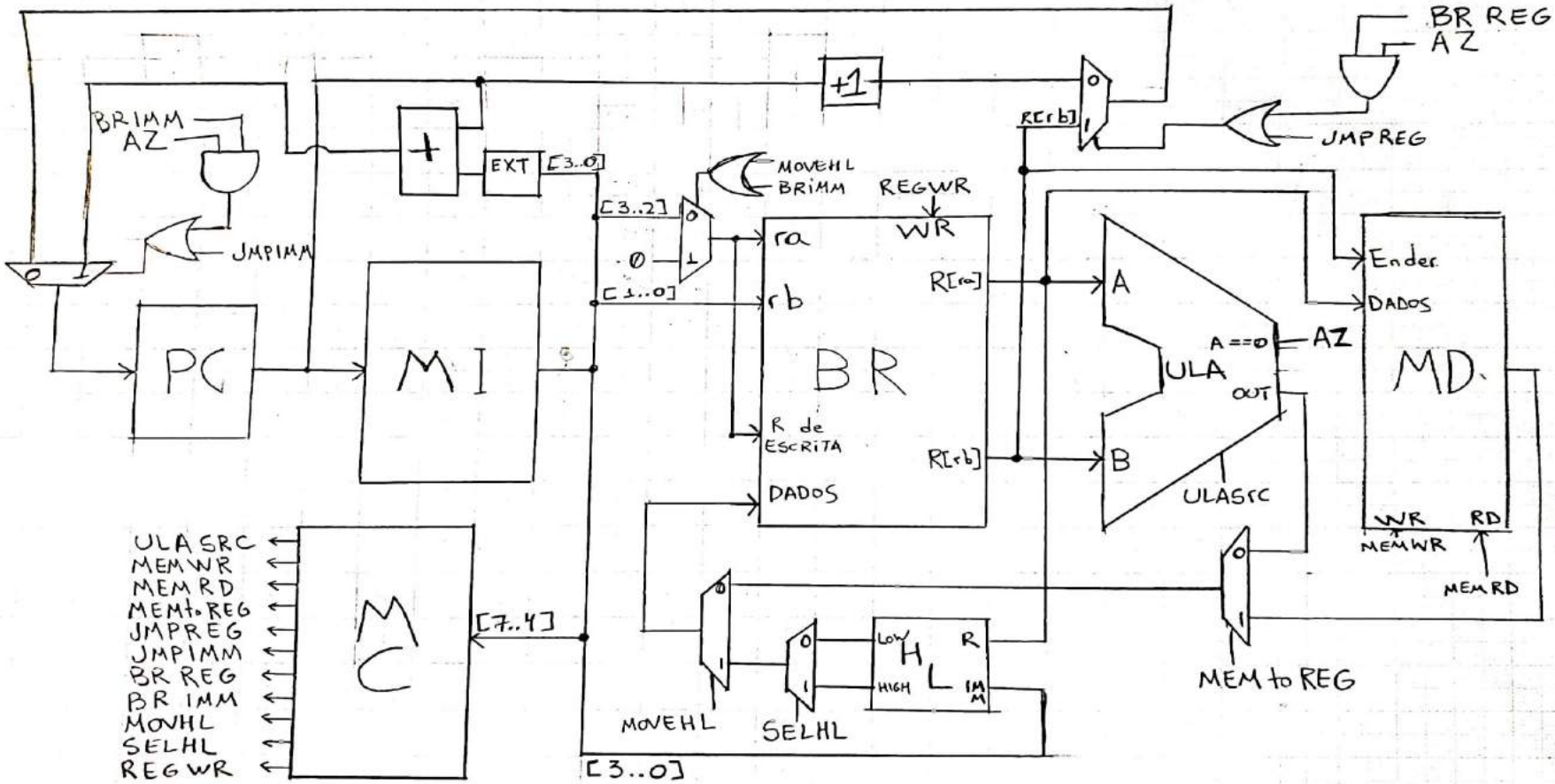
A MC foi implementada utilizando uma memória ROM 16x16.

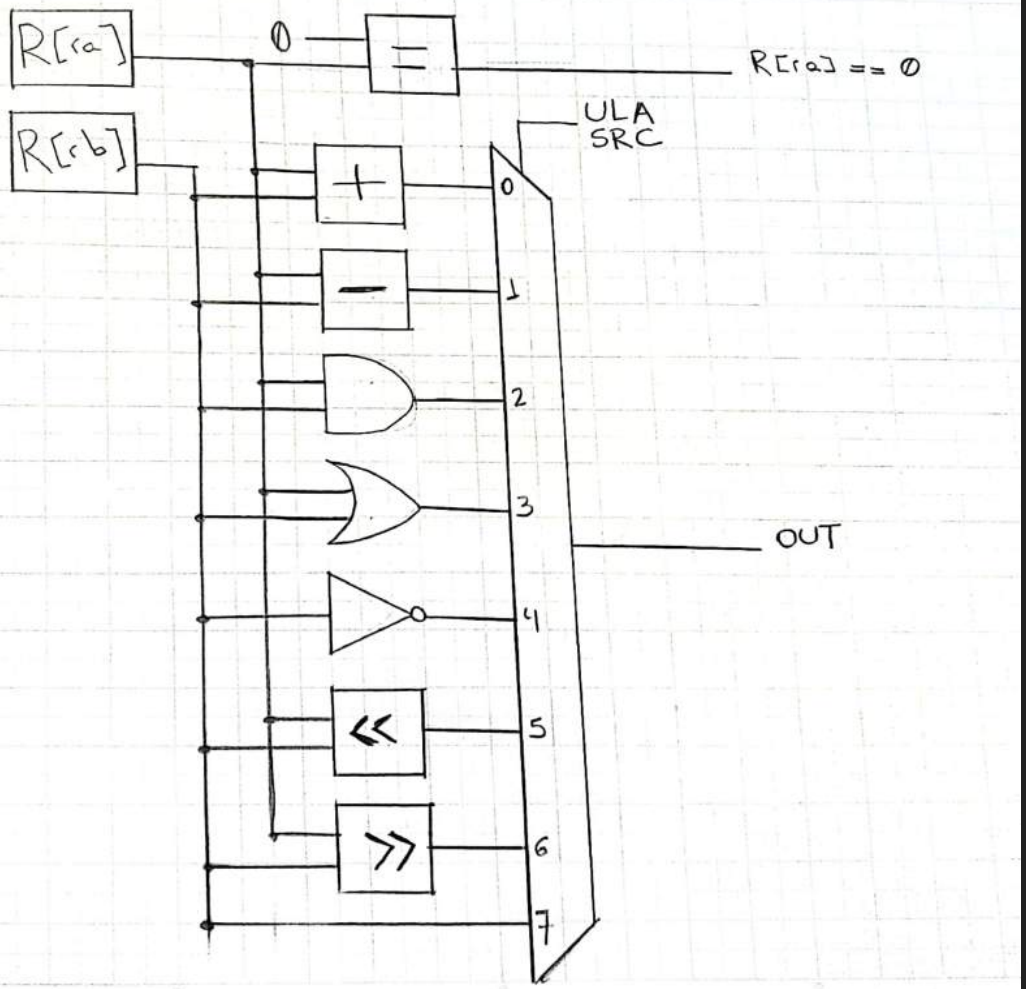
Memória de Controle

NOME	Função	Bits
ALUsrc	Controla a op da ula	3
MEMwr	Escrita na Memória	1
MEMrd	Ler da Memória	1
MEMtoREG	Armazenar da memória em um registrador	1
JMPReg	jr flag (Usado no controle do PC)	1
JMPImm	ji flag (Usado no controle do PC)	1
BranchReg	brzr flag (Usado no controle do PC)	1
BranchImm	brzi (Usado no controle do PC)	1
MoveHL	1 se é uma operação de *Move High* ou *Move Low*	1
SelHL	0 - *Move Low* 1 - *Move High*	1
REGwr	Habilita escrita no banco de registradores	1



Projeto final implementado no Logisim


BR REG
AZ
+1
R[rb]
JMPREG
BR IMM
AZ
JMPIMM
+
EXT
[3..0]
MOVEHL
BRIMM
REGWR
WR
[3..2]
0
[1..0]
ra
rb
R de
ESCRITA
DADOS
R[ra]
R[rb]
BR
PC
MI
A
B
ULA
A==0
AZ
OUT
ULASrc
Ender.
DADOS
MD.
WR
RD
MEMWR
MEMRD
MEM to REG
ULASRC
MEMWR
MEMRD
MEM.REG
JMPREG
JMPIMM
BR REG
BR IMM
MOVHL
SELHL
REGWR
MC
[7..4]
MOVEHL
SELHL
LOW
HIGH
H
L
R
IMM
[3..0]


R[ra]
R[rb]
0
=
R[ra] == 0
ULA SRC
+
−
<<
>>
0
1
2
3
4
5
6
7
OUT