INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Licenciatura em Engenharia de Eletrónica e Telecomunicações e de Computadores

e

Licenciatura em Engenharia Informática e de Computadores

1.º Trabalho Prático

de

Arquitetura de Computadores

Estudo de um processador

Grupo 2

Paulo Rosa – 44873

Ricardo Pinto - 44808

10 de outubro de 2019

1 Objetivos

Este trabalho prático tem como principal objetivo o estudo do funcionamento de um processador. Neste contexto, são abordadas as problemáticas da codificação de um ISA, o projeto do descodificador de instruções para a unidade de controlo do processador e a codificação de programas usando a linguagem máquina.

2 Descrição da arquitetura

O processador considerado neste trabalho é de ciclo único e implementa uma arquitetura de Harvard a 8 bits, em que as memórias de dados e de código contêm, cada uma, 256 posições, conforme ilustrado na Figura 1.

A microarquitetura subjacente inclui oito registos de uso geral (`r0`, `r1`, ... e `r7`), uma Unidade Lógica e Aritmética (ALU) capaz de realizar três operações, conforme é ilustrado na Figura 3, e um registo de estado do processador (PSW) que disponibiliza o indicador de resultado igual a zero (Z).

A Tabela 1 apresenta o conjunto de instruções suportado pela arquitetura, codificadas com nove bits, em que:

- `rx` e `ry` representam um dos oito registos de uso geral do processador (`r0`, `r1`, `r2`, `r3`, `r4`, `r5`, `r6`, `r7`);
- `const3` simboliza o valor de uma constante, codificada sem sinal com 3 bits;
- `offset6` simboliza o valor de uma constante, codificada com 6 bits com sinal, que é usada como parte de menor peso na síntese do endereço relativo de memória (os bits de maior peso são estendidos com o bit de sinal).

Instrução	Descrição	
ldr rx, [ry]	Transfere para **rx** o conteúdo da posição de memória cujo endereço é definido pelo conteúdo de **ry**.	**rx** ← mem**[ry]**
str rx, [ry]	Transfere o conteúdo de **rx** para a posição de memória cujo endereço é definido pelo conteúdo de **ry**.	mem[**ry**] ← **rx**
mov rx, #const3	Carrega o valor da constante **const3** no registo **rx**.	**rx** ← **const3**
add rx, ry	Adiciona o conteúdo de **ry** ao conteúdo de **rx**, colocando o resultado em **rx** e atualizando o registo **PSW** com a informação da *flag* **Z** gerada na ALU.	**rx** ← **rx + ry** e atualiza **PSW**
sub rx, #const3	Subtrai o valor da constante **const3** ao conteúdo de **rx**, colocando o resultado em **rx** e atualizando o registo **PSW** com a informação da *flag* **Z** gerada na ALU.	**rx** ← **ry − const3** e atualiza **PSW**
and rx, ry	Realiza a operação lógica *and* entre os bits da mesma posição de **rx** e **ry**, colocando o resultado em **rx** e atualizando o registo **PSW** com a informação da *flag* **Z** gerada na ALU.	**rx** ← **ry & ry** e atualiza **PSW**
bzc offset6	Quando a *flag* **Z** apresenta o valor 0, muda a execução para o endereço resultante da adição ao **PC** do deslocamento representado por **offset6**.	**PC** ← (**Z** == 0) ? **PC** + **offset6** : **PC** + 1
b rx	Muda a execução para o endereço definido pelo conteúdo de **rx**.	**PC** ← **rx**

Tabela 1 – Conjunto de instruções do processador.

Na Tabela 2 apresentam-se os códigos incompletos das instruções do ISA (*opcodes*).

Instrução	*opcode*
ldr rx, [ry]	0??
str rx, [ry]	1??
mov rx, #const3	011
add rx, ry	0??
sub rx, #const3	0??
and rx, ry	1??
bzc offset6	100
b rx	111

Tabela 2 – Códigos incompletos das instruções do ISA.

3 Trabalho a realizar

Respeitando o ISA e a microarquitetura apresentados, pretende-se completar o projeto do processador proposto e utilizá-lo para executar um programa. Para tal, devem ser realizadas três tarefas.

3.1 Codificação das instruções do ISA

a) Complete os *opcodes* apresentados na Tabela 2, por forma a ser possível realizar todas as operações usando como ALU o circuito apresentado na Figura 3.

Instrução	*opcode*
`ldr rx, [ry]`	0**10**
`str rx, [ry]`	1**01**
`mov rx, #const3`	011
`add rx, ry`	0**00**
`sub rx, #const3`	0**01**
`and rx, ry`	1**10**
`bzc offset6`	100
`b rx`	111

b) Apresente o mapa de codificação das instruções, tendo em conta os *opcodes* referidos na alínea anterior e o diagrama de blocos do processador descrito na Figura 1.

Instrução	opcode	AD=(AA)	AB
`ldr rx, [ry]`	010	rx	ry
`str rx, [ry]`	101	rx	ry
`mov rx, #const3`	011	rx	CONST3
`add rx, ry`	000	rx	ry
`sub rx, #const3`	001	rx	CONST3
`and rx, ry`	110	rx	ry
`bzc offset6`	100	OFF	SET6
`b rx`	111	rx	-

3.2 Projeto do descodificador de instruções

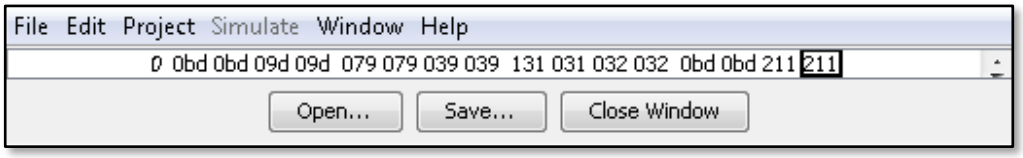
a) Apresente, numa tabela, o valor lógico das saídas do subcircuito `Instruction Decoder` do processador, descrito na Figura 1, para cada uma das instruções indicadas na Tabela 1. Explicite os casos de indiferença (*don't care*) e as saídas obtidas diretamente do código da instrução.

DC = "don't care"

Instrução	*Opcode*	Z	SI	SO	SD	SC	SA	ER	EP	WR	RD	HEX
`ldr rx, [ry]`	000	-	0	0	01	1(dc)	1	1	0	0	1	79
`str rx, [ry]`	001	-	0	0	00(dc)	1(dc)	1	0	0	1	0	32
`mov rx, #const3`	010	-	0	0	00	1(dc)	1(dc)	1	0	0	1	39
`add rx, ry`	011	-	0	0	10	1	1(dc)	1	1	0	1	BD
`sub rx, #const3`	100	-	0	0	10	0	1(dc)	1	1	0	1	9D
`and rx, ry`	101	-	0	0	10	1	1(dc)	1	1	0	1	BD
`bzc offset6`	110	0	0	1	00(dc)	1(dc)	1(dc)	0	0	0	1	131
`bzc offset6`	110	1	0	0	00(dc)	1(dc)	1(dc)	0	0	0	1	31
`b rx`	110	-	1	0(dc)	00(dc)	1(dc)	1(dc)	0	0	0	1	211

b) Determine o conteúdo da ROM utilizada na implementação do subcircuito `Instruction Decoder` no `Logisim`. Preencha a ROM com essa informação.

Foi preciso organizar por ordem binária(segundo o *opcode* e o *Z*),



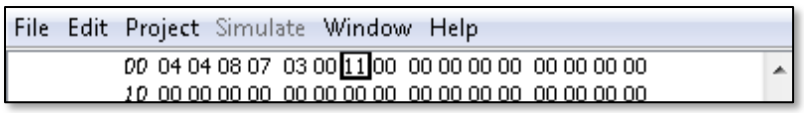
3.3 Teste da arquitetura

Considere a seguinte sequência de instruções, que deverá utilizar para testar o funcionamento do processador utilizando a aplicação `Logisim`.

```
mov  r0, #0
mov  r1, #5
sub  r1, #1
ldr  r2, [r1]
add  r0, r2
and  r1, r1
bzc  -4
mov  r4, #5
str  r0, [r4]
mov  r4, #6
ldr  r4, [r4]
b    r4
```

a) Codifique as instruções apresentadas e carregue-as na memória de código do processador no `Logisim`. Carregue também as primeiras cinco posições da memória de dados com o código dos dígitos que compõem o número de aluno de um dos elementos do grupo (o código de um dígito por posição de memória) e na posição 6 carregue o valor 11.

Instrução	Opcode	AD(=AA)	AB	HEX
Mov r0, #0	011	000	000	C0
Mov r1, #5	011	001	101	CD
Sub r1, #1	001	001	001	49
Ldr r2 [r1]	010	010	001	91
Add r0, r2	000	000	010	2
And r1, r1	110	001	001	189
Bzc -4	100	111	100	13C
Mov r4, #5	011	100	101	E5
Str r0, [r4]	101	000	100	144
Mov r4, #6	011	100	110	E6
Ldr r4, [r4]	010	100	100	A4
B r4	111	100	000(dc)	1E0



b) Execute o troço de código no `Logisim` e registe, para cada uma das instruções, as alterações ocorridas nos registos do processador (`r0-r7`, `PC` e `PSW`) e na memória de dados.

INSTRUÇÃO	PC	R0	R1	R2	R3	R4	PSW	MEM DADOS
mov r0, #0	0	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000	0	{4,4,8,7,3,0,11}
mov r1, #5	1	0000 0000	0000 0101	0000 0000	0000 0000	0000 0000	0	{4,4,8,7,3,0,11}
sub r1, #1	2	0000 0000	0000 0100	0000 0000	0000 0000	0000 0000	0	{4,4,8,7,3,0,11}
ldr r2, [r1]	3	0000 0000	0000 0100	0000 0011	0000 0000	0000 0000	0	{4,4,8,7,3,0,11}
add r0, r2	4	0000 0011	0000 0100	0000 0011	0000 0000	0000 0000	0	{4,4,8,7,3,0,11}
and r1, r1	5	0000 0011	0000 0100	0000 0011	0000 0000	0000 0000	0	{4,4,8,7,3,0,11}
bzc -4	6	0000 0011	0000 0100	0000 0011	0000 0000	0000 0000	0	{4,4,8,7,3,0,11}
sub r1, #1	**2**	0000 0011	0000 0011	0000 0011	0000 0000	0000 0000	0	{4,4,8,7,3,0,11}
ldr r2, [r1]	**3**	0000 0011	0000 0011	0000 0111	0000 0000	0000 0000	0	{4,4,8,7,3,0,11}
add r0, r2	**4**	0000 1010	0000 0011	0000 0111	0000 0000	0000 0000	0	{4,4,8,7,3,0,11}
and r1, r1	**5**	0000 1010	0000 0011	0000 0111	0000 0000	0000 0000	0	{4,4,8,7,3,0,11}
bzc -4	**6**	0000 1010	0000 0011	0000 0111	0000 0000	0000 0000	0	{4,4,8,7,3,0,11}
sub r1, #1	**2**	0000 1010	0000 0010	0000 0111	0000 0000	0000 0000	0	{4,4,8,7,3,0,11}
ldr r2, [r1]	**3**	0000 1010	0000 0010	0000 1000	0000 0000	0000 0000	0	{4,4,8,7,3,0,11}
add r0, r2	**4**	0001 0010	0000 0010	0000 1000	0000 0000	0000 0000	0	{4,4,8,7,3,0,11}
and r1, r1	**5**	0001 0010	0000 0010	0000 1000	0000 0000	0000 0000	0	{4,4,8,7,3,0,11}
bzc -4	**6**	0001 0010	0000 0010	0000 1000	0000 0000	0000 0000	0	{4,4,8,7,3,0,11}
sub r1, #1	**2**	0001 0010	0000 0001	0000 1000	0000 0000	0000 0000	0	{4,4,8,7,3,0,11}
ldr r2, [r1]	**3**	0001 0010	0000 0001	0000 0100	0000 0000	0000 0000	0	{4,4,8,7,3,0,11}
add r0, r2	**4**	0001 0110	0000 0001	0000 0100	0000 0000	0000 0000	0	{4,4,8,7,3,0,11}
and r1, r1	**5**	0001 0110	0000 0001	0000 0100	0000 0000	0000 0000	0	{4,4,8,7,3,0,11}
bzc -4	**6**	0001 0110	0000 0001	0000 0100	0000 0000	0000 0000	0	{4,4,8,7,3,0,11}
sub r1, #1	**2**	0001 0110	0000 0000	0000 0100	0000 0000	0000 0000	1	{4,4,8,7,3,0,11}
ldr r2, [r1]	**3**	0001 0110	0000 0000	0000 0100	0000 0000	0000 0000	1	{4,4,8,7,3,0,11}

add r0, r2	**4**	0001 1010	0000 0000	0000 0100	0000 0000	0000 0000	0	{4,4,8,7,3,0,11}
and r1, r1	**5**	0001 1010	0000 0000	0000 0100	0000 0000	0000 0000	1	{4,4,8,7,3,0,11}
bzc -4	**6**	0001 1010	0000 0000	0000 0100	0000 0000	0000 0000	1	{4,4,8,7,3,0,11}
mov r4, #5	7	0001 1010	0000 0000	0000 0100	0000 0000	0000 0101	1	{4,4,8,7,3,0,11}
str r0, [r4]	8	0001 1010	0000 0000	0000 0100	0000 0000	0000 0101	1	{4,4,8,7,3,1A,11}
mov r4, #6	9	0001 1010	0000 0000	0000 0100	0000 0000	0000 0110	1	{4,4,8,7,3,1A,11}
ldr r4, [r4]	A	0001 1010	0000 0000	0000 0100	0000 0000	0001 0001	1	{4,4,8,7,3,1A,11}
b r4	B	0001 1010	0000 0000	0000 0100	0000 0000	0001 0001	1	{4,4,8,7,3,1A,11}

Não houve alterações aos registos R5,R6,R7. Foram, ao longo da execução do código, iguais a 0000 0000.

4 Conclusão

Com a realização deste trabalho, pusemos em prática a matéria lecionada em aula. O que levou a uma melhor compreensão de como um básico processador funciona com o auxílio do programa Logisim. Foi muito interessante porque estudámos e fomos capazes de compreender um dos componentes mais importantes da arquitetura de um computador, nomeadamente um CPU.

5 Diagramas de bloco

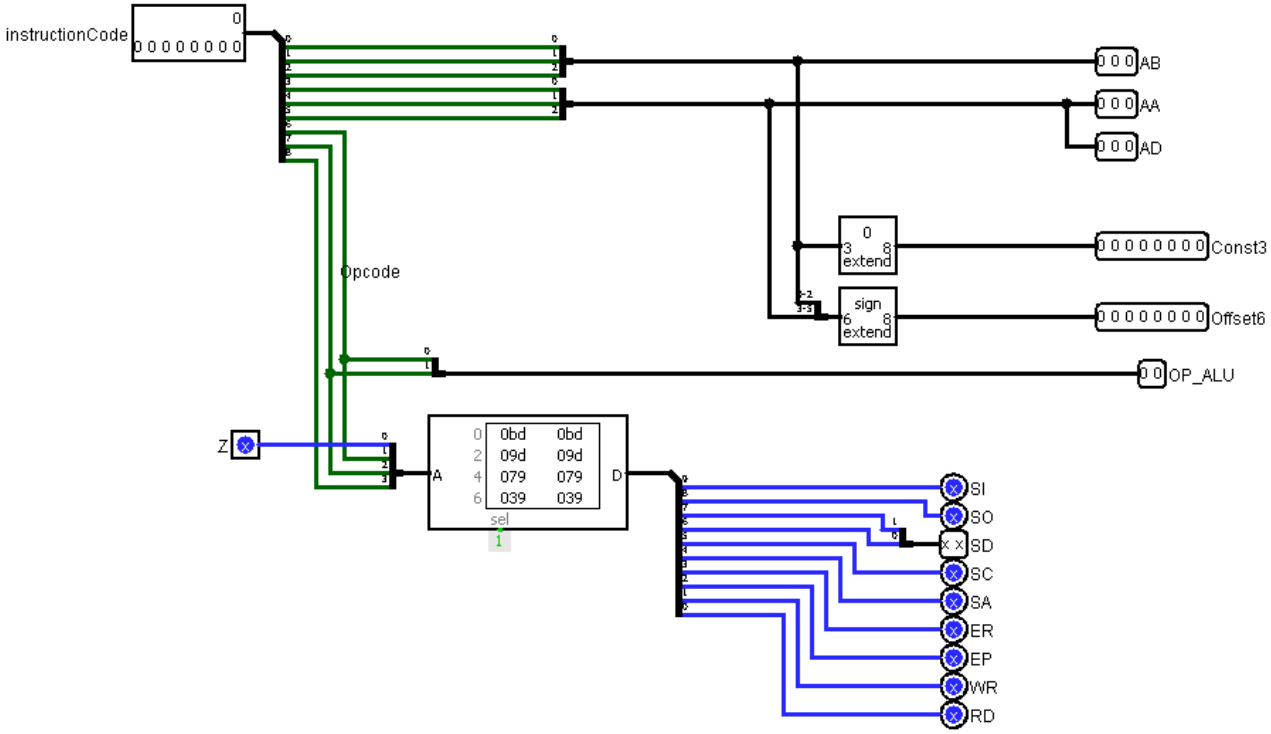


Figura 4 Instruction Decoder

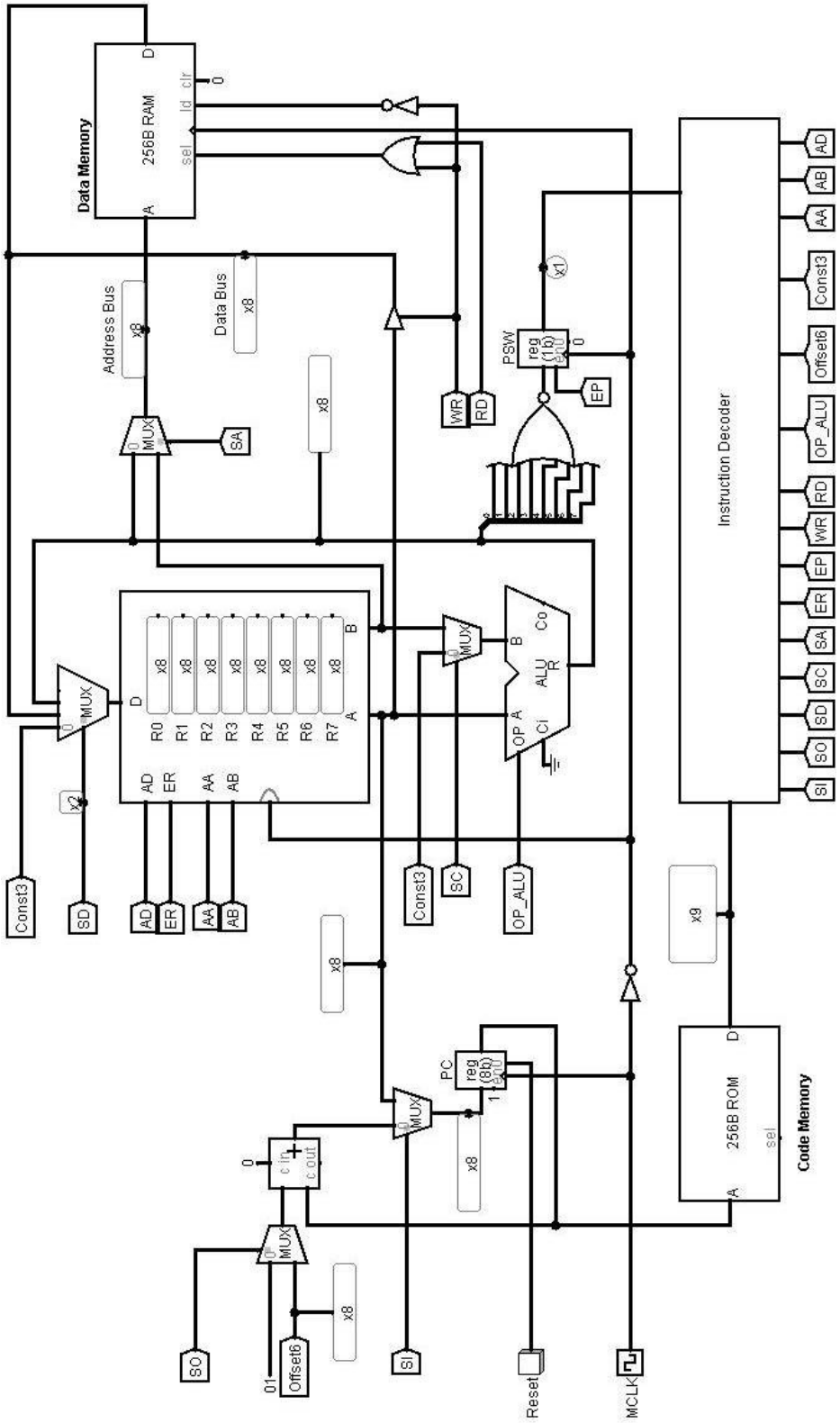


Figura 1 – Diagrama de blocos do processador.

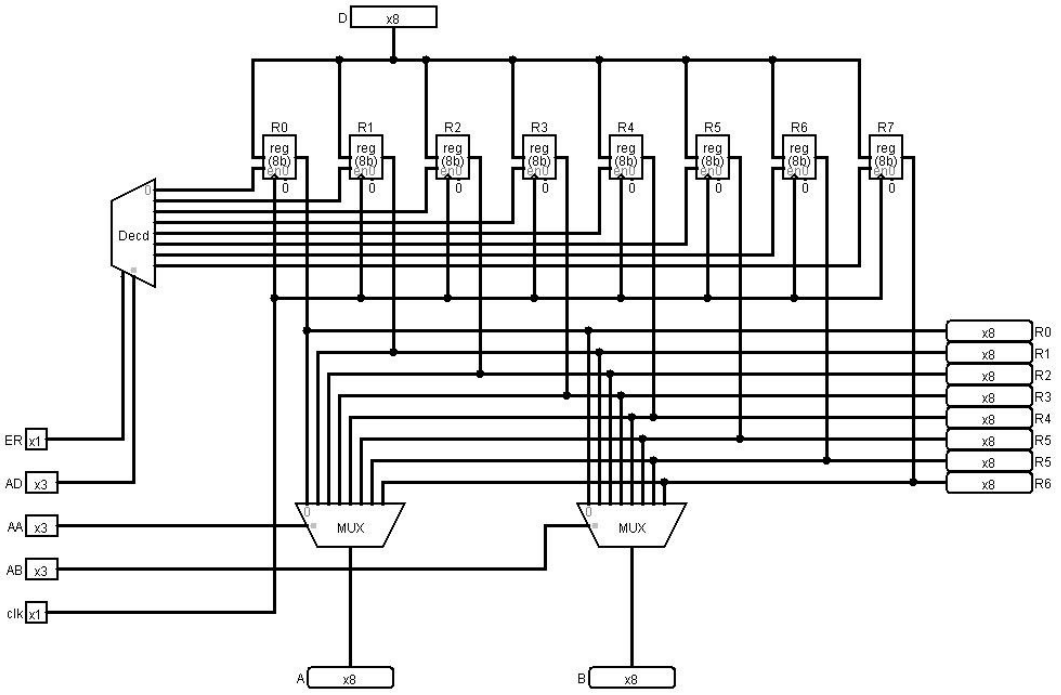


Figura 2 – Diagrama de blocos do banco de registos.

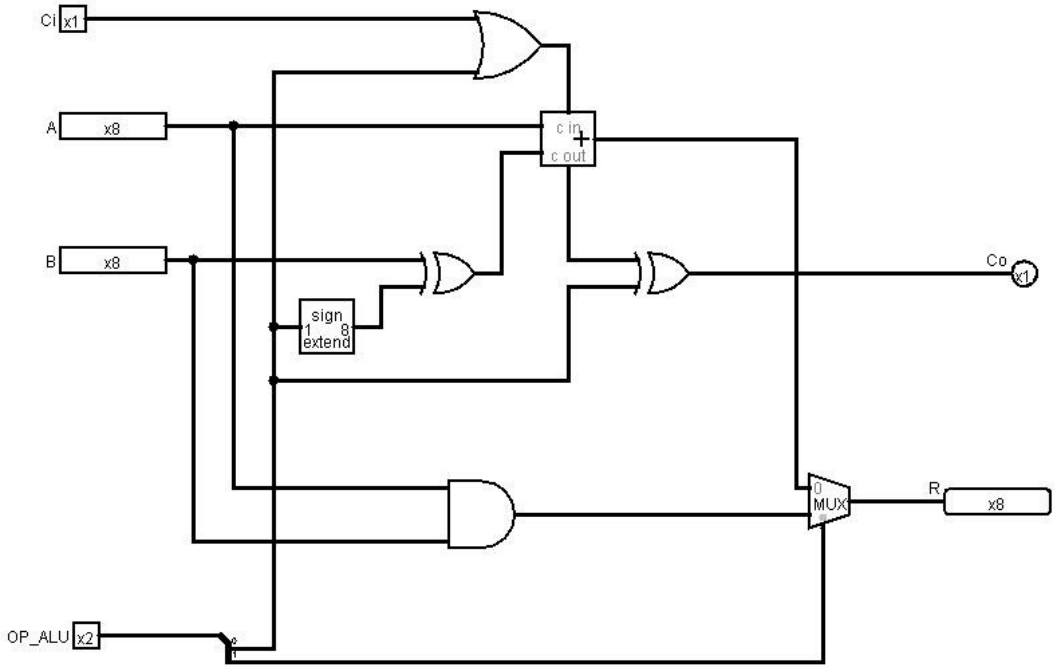


Figura 3 – Diagrama de blocos da ALU.