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

30 de setembro 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.

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.

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.

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

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.

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.

4 Avaliação

O trabalho deve ser realizado em grupo, conta para o processo de avaliação da unidade curricular e tem a duração de duas semanas.

A apresentação da solução proposta por cada grupo decorre em sessão de laboratório, em data a combinar com o docente responsável pela lecionação das aulas da respetiva turma.

Após esta apresentação, cada grupo deverá entregar o relatório do trabalho ao docente, no qual deve constar:

- Uma descrição sucinta da solução proposta, acompanhada dos esquemas de todos os circuitos e subcircuitos desenvolvidos;
- As conclusões.

5 Diagramas de blocos

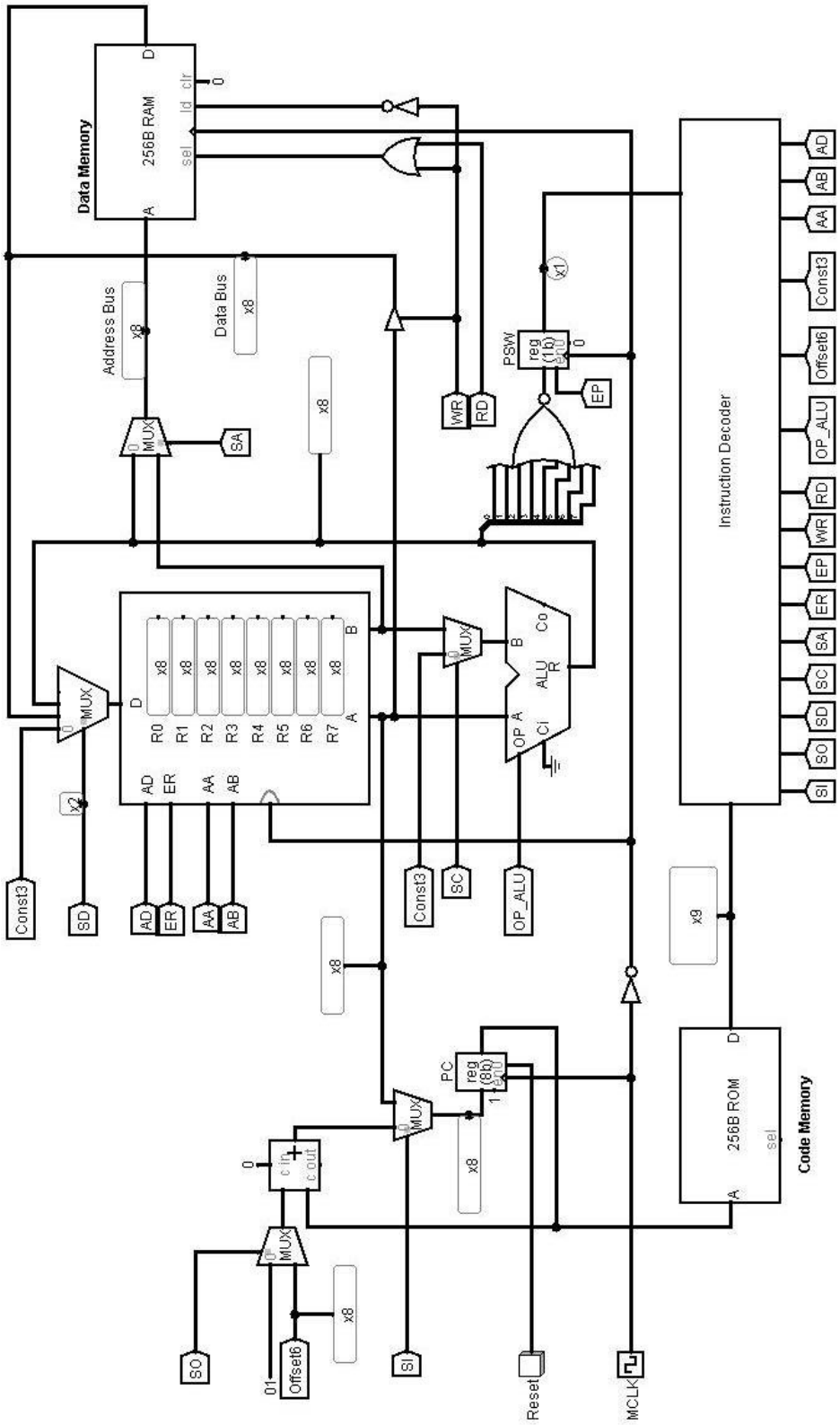

Figura 1 – Diagrama de blocos do processador.

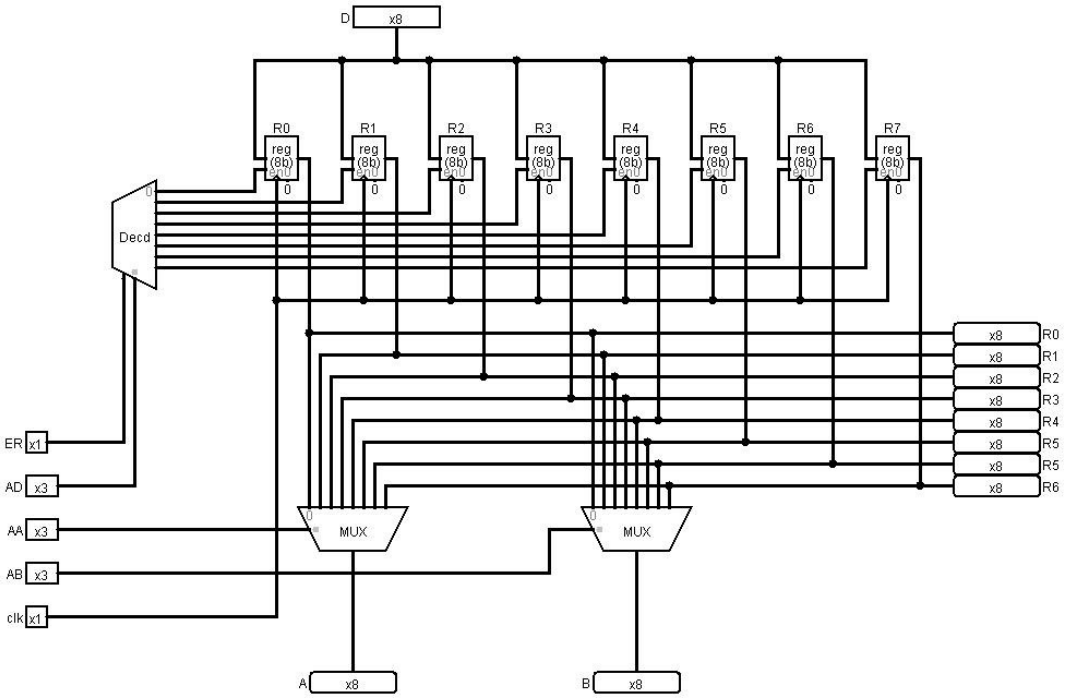

Figura 2 – Diagrama de blocos do banco de registos.

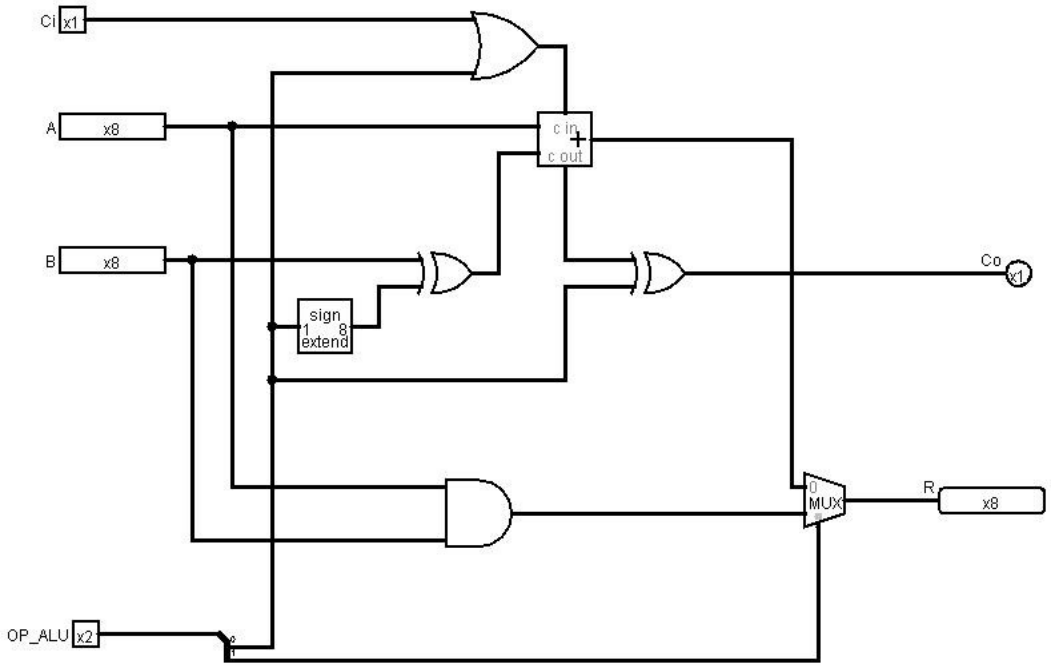

Figura 3 – Diagrama de blocos da ALU.