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

26 de março de 2020

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.

1 Descrição da arquitetura

O processador considerado neste trabalho é de ciclo único e implementa uma arquitetura Harvard a 8 bits, em que as memórias de dados e de código contêm, 256 e 1024 posições, respetivamente, conforme apresentado na Figura 1.

A microarquitetura subjacente inclui oito registos de uso geral (`r0`, `r1`, ... e `r7`), uma Unidade Lógica e Aritmética (em inglês, *Arithmetic and Logic Unit* - ALU) capaz de realizar quatro operações, conforme se ilustra na Figura 3, e um registo de estado do processador (em inglês, *Processor Status Register* - `PSR`) que disponibiliza o indicador de resultado igual a zero (`Z`).

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

- `rx`, `ry` e `rz` representam um dos registos de uso geral do processador (`r0`, `r1`, `r2`, `r3`, `r4`, `r5`, `r6`, `r7`);
- `imm3` simboliza o valor de um número natural, codificado com 3 bits;
- `imm6` simboliza o valor de um número natural, codificado com 6 bits;
- `offset8` simboliza o valor de um número inteiro, codificado com 8 bits, que é usada como parte de menor peso na síntese do endereço relativo de memória (os outros bits correspondem ao bit de sinal).

Instrução	Descrição	
`ldr rx, [ry, imm3]`	Transfere para **`rx`** o conteúdo da posição de memória cujo endereço é definido pela soma do conteúdo de **`ry`** com a constante **`imm3`**.	**`rx`** ← mem**`[ry + imm3]`**
`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, imm6`	Carrega o valor da constante **`imm6`** no registo **`rx`**.	**`rx ← imm6`**
`add rx, ry, rz`	Adiciona o conteúdo de **`rz`** ao conteúdo de **`ry`**, colocando o resultado em **`rx`** e atualiza o registo **`PSR`** com a informação da *flag* **`Z`** gerada na ALU.	**`rx ← ry + rz`** e atualiza **`PSW`**
`cmp rx, ry`	Subtrai o conteúdo de **`ry`** ao conteúdo de **`rx`** e atualiza o registo **`PSR`** com a informação da *flag* **`Z`** gerada na ALU.	**`rx − ry`** e atualiza **`PSW`**
`lsr rx, ry, imm3`	Desloca o conteúdo de **`ry`** para a direita de **`imm3`** bits e guarda o resultado em **`rx`**. Atualiza o registo **`PSR`** com a informação da *flag* **`Z`** gerada na ALU.	**`rx ← ry >> imm3`** e atualiza **`PSW`**
`b offset8`	Muda a execução para o endereço resultante da adição ao PC da constante **`offset8`**.	**`PC ← PC + offset8`**
`bne rx`	Quando a *flag* **`Z`** apresenta o valor 0, muda a execução para o endereço definido pelo conteúdo de **`rx`**.	**`PC`** ← (**`Z`** == 0) ? **`rx`** : **`PC`** + 1

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, imm3]`	1??
`str rx, [ry]`	1??
`mov rx, imm6`	011
`add rx, ry, rz`	0??
`cmp rx, ry`	0??
`lsr rx, ry, imm3`	0??
`b offset8`	110
`bne rx`	101

Tabela 2 – Códigos incompletos das instruções do ISA.

2 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.

2.1 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* definidos para a alínea anterior e o diagrama de blocos do processador apresentado na Figura 1.

2.2 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 para o `Logisim` do subcircuito `Instruction Decoder`. Preencha a ROM com essa informação.

2.3 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, 0
mov  r2, 4
mov  r4, 1
ldr  r3, [r0, 0]
add  r1, r1, r3
add  r0, r0, r4
cmp  r0, r2
bne  r2
lsr  r1, r1, 2
str  r1, [r2]
b    0
```

a) Codifique as instruções apresentadas e, no `Logisim`, carregue-as na memória de código do processador. Carregue também as primeiras quatro posições da memória de dados com valores à sua escolha.

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 `PSR`) e na memória de dados.

3 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 será feita 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.

4 Diagramas de blocos

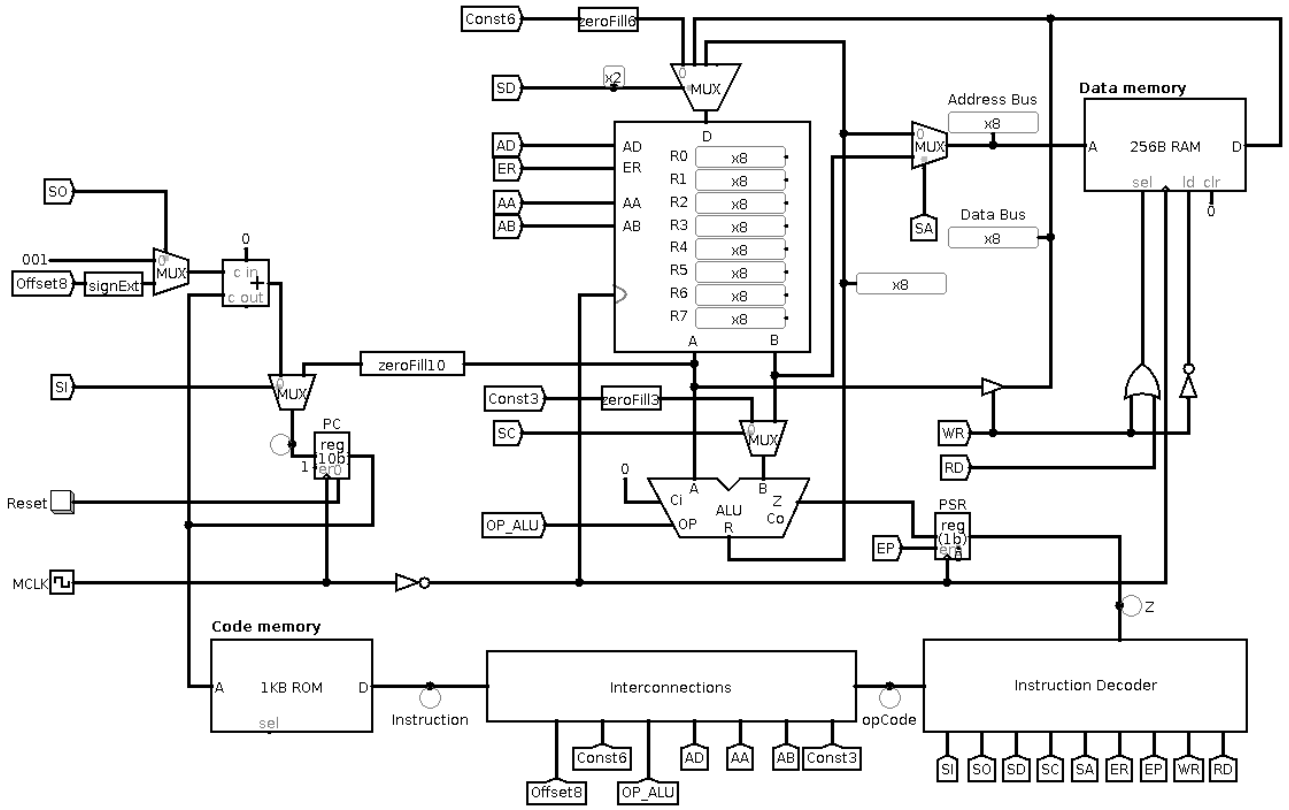

Figura 1 – Diagrama de blocos do processador.

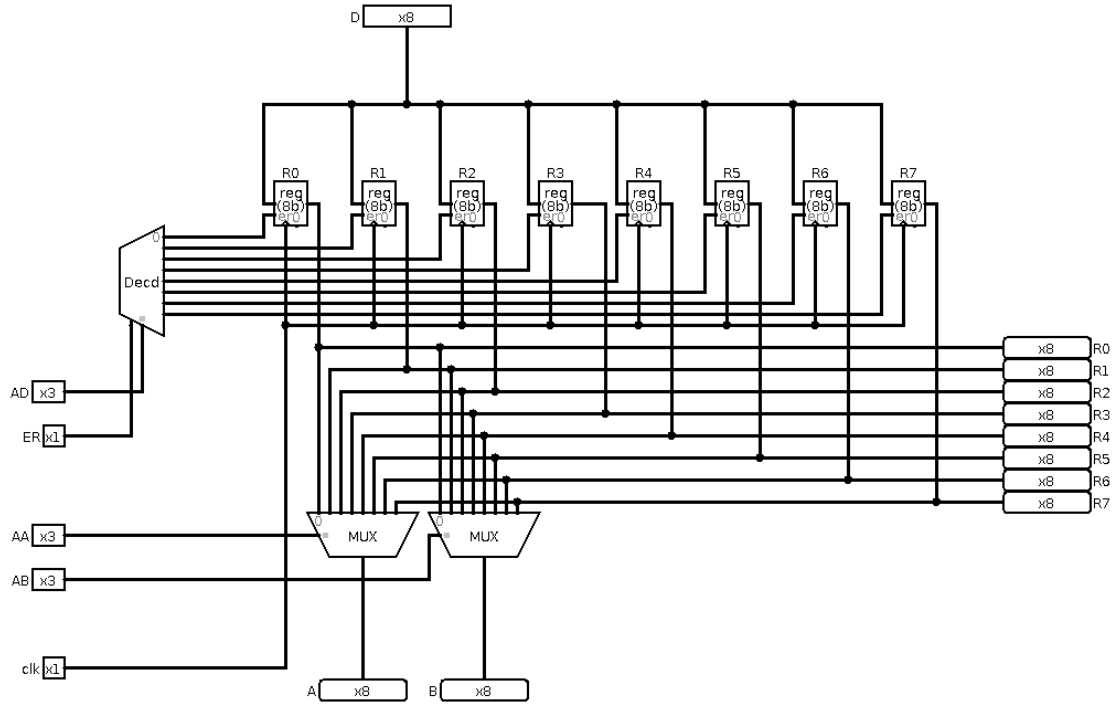

Figura 2 – Diagrama de blocos do banco de registos.

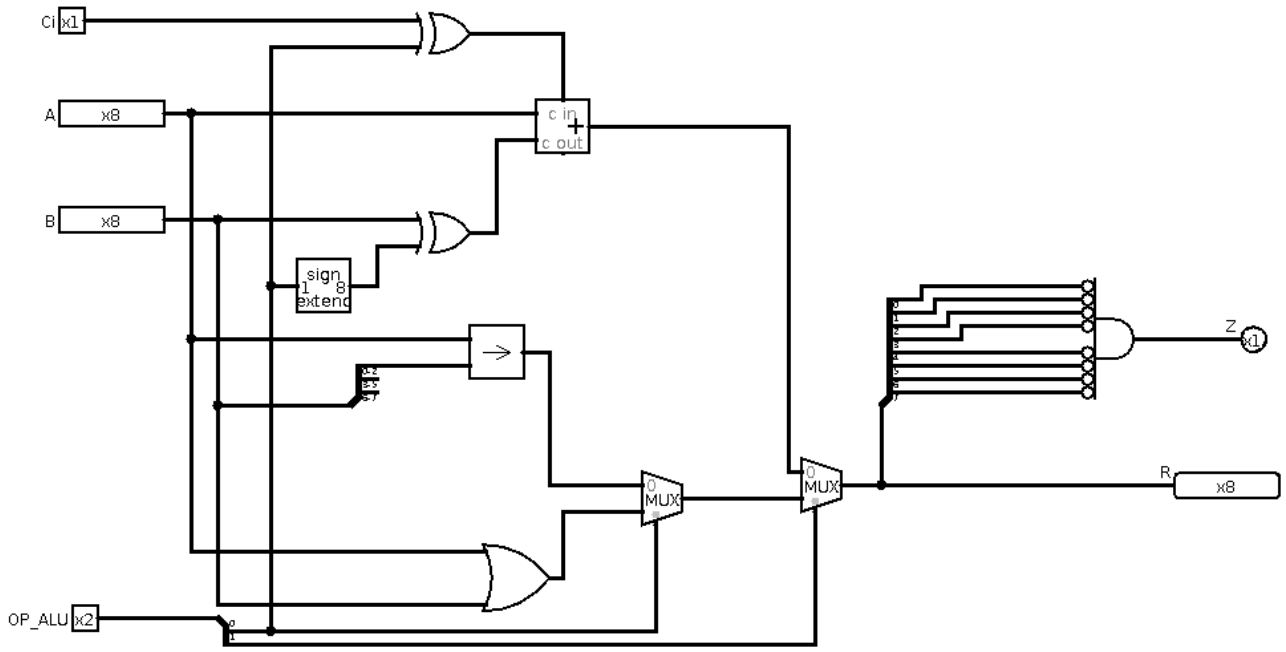

Figura 3 – Diagrama de blocos da ALU.