Trabalho Prático 2

Estudo do funcionamento de um processador

Arquitetura de Computadores

Departamento de Engenharia Eletrónica e de
Telecomunicações e Computadores

24 de Outubro de 2022

1 Objetivos

Este trabalho tem como principal objetivo a compreensão do funcionamento de um processador, sendo abordados os seguintes tópicos: codificação de um conjunto de instruções, funcionamento de uma microarquitetura e codificação de programas em linguagem máquina. O ambiente de simulação `Logisim` é utilizado para consolidação do estudo.

2 Especificação do Exercício

Pretende-se completar o projeto de um processador, respeitando o seu modelo de programação e microarquitetura de ciclo único, bem como fazer a validação do projeto usando o ambiente de simulação `Logisim` e um pequeno programa de teste.

O processador considerado é de 8 bits e tem o seguinte modelo de programação:

- Espaço de endereçamento para código com 1024 endereços;
- Espaço de endereçamento para dados com 256 endereços;
- Oito registos de uso geral, denominados `r0`, `r1`, ... e `r7`;
- Um registo de estado, denominado `CPSR` (do inglês *Current Program Status Register*), que disponibiliza um bit que indica que a última operação realizada deu resultado zero (*flag* `Z`);
- O conjunto de instruções apresentado na Tabela 1.

Nas instruções apresentadas na Tabela 1, `rd`, `rm` e `rn` representam um dos registos de uso geral do processador, `imm`n simboliza um número natural codificado com n bits, `label` identifica um endereço na vizinhança de ± 256 endereços da instrução de salto em causa e `PC` referencia o registo *program counter*.

A microarquitetura do processador está descrita no formato `Logisim` no ficheiro "ac_2223i_tp2.circ", disponível na página de meta disciplina de AC na plataforma Moodle.

Instrução	Descrição	
`add rd, rn, rm`	Adiciona o conteúdo de `rm` ao conteúdo de `rn`, colocando o resultado em `rd` e atualizando o registo `CPSR` com a informação do bit `Z` gerada na ALU.	`rd` ← `rn` + `rm` `rd` == 0 ? `CPSR.Z` ← 1 : `CPSR.Z` ← 0
`b label`	Muda a execução para o endereço associado ao símbolo `label`.	`PC` ← `label`
`bne rn`	Muda a execução para o endereço definido pelo conteúdo de `rn` quando a *flag* `Z` do registo `CPSR` apresenta o valor zero.	`CPSR.Z` == 0 ? `PC` ← `label` : `PC` ← `PC` + 1
`cmp rn, rm`	Subtrai o conteúdo de `rm` ao conteúdo de `rm`, atualizando o registo `CPSR` com a informação do bit `Z` gerada na ALU.	`rn` − `rm` == 0 ? `CPSR.Z` ← 1 : `CPSR.Z` ← 0
`ldr rd, [rn, #imm3]`	Copia para `rd` o conteúdo da posição da memória de dados cujo endereço resulta da adição do valor da constante `imm3` ao conteúdo de `rn`.	`rd` ← M[`rn` + `imm3`]
`lsr rd, rn, #imm3`	Desloca o conteúdo de `rn` para a direita de `imm3` bits, colocando o resultado em `rd` e atualizando o registo `CPSR` com a informação do bit `Z` gerada na ALU.	`rd` ← `rn` ≫ `imm3`; `rd` == 0 ? `CPSR.Z` ← 1 : `CPSR.Z` ← 0
`mov rd, #imm6`	Estabelece em `rd` o valor da constante `imm6`.	`rd` ← `imm6`
`str rd, [rn]`	Copia o conteúdo de `rd` para a posição da memória de dados com o endereço definido pelo conteúdo de `rn`.	M[`rn`] ← `rd`

Tabela 1: Conjunto de instruções do processador

3 Trabalho a Realizar

3.1 Análise da microarquitectura

Considere a descrição da microarquitectura do processador disponibilizada no ficheiro "ac_2223i_tp2.circ".

1. Comente a seguinte afirmação: *"A microarquitetura do processador é do tipo von Neumann."*
2. Indique a funcionalidade realizada pelos blocos `Ext`, `LExt` e `RExt`. Justifique a sua resposta com base no funcionamento das instruções `b`, `bne`, `ldr`, `lsr` e `mov`.
3. Identifique as operações realizadas pela Unidade Lógica e Aritmética (ALU) do processador e indique o valor do sinal `OP` associado a cada operação.

3.2 Codificação das instruções

Considere a utilização de um código de comprimento fixo e um esquema de codificação uniforme para a codificação do conjunto de instruções apresentado.

1. Apresente um mapa de codificação para o conjunto de instruções, tendo em conta o formato de codificação da instrução `ldr` apresentado na Figura 1.

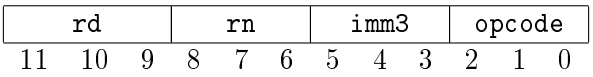

Figura 1: Formato de codificação da instrução `ldr`

2. Sabendo que o valor `001` corresponde ao código de operação (`opcode`) da instrução `ldr`, indique os valores deste campo para as restantes instruções. Justifique a sua resposta tendo em conta também o funcionamento da ALU.

3.3 Projeto do descodificador de instruções

Considere o subcircuito `Instruction Decoder` disponibilizado no ficheiro "ac_2223i_tp2.circ", que implementa o descodificador de instruções da microarquitectura proposta para o processador.

1. Usando uma tabela, apresente os valores das saídas deste bloco, em função dos seus sinais de entrada, para o conjunto de instruções apresentado. Explicite os casos de indiferença (*don't care*) e, se aplicável, as saídas obtidas diretamente do código da instrução.
2. Considerando uma implementação baseada, exclusivamente, numa ROM, determine o conteúdo dessa memória. Indique os valores dos endereços e conteúdos em notação hexadecimal.
3. Indique, em bits, a capacidade da memória ROM considerada no ponto 2.

3.4 Codificação de programas em linguagem máquina

Considere o troço de código apresentado na Listagem 1, escrito na linguagem *assembly* do processador.

1. Indique a funcionalidade do troço de código.
2. Traduza o troço de código para código máquina.

3. Pondera-se substituir a instrução `str rd, [rn]` pela instrução `str rd, [rn, rm]`, que realiza a operação `M[rn + rm] = rd`.
   (a) Indique, justificando, se é possível fazer esta alteração ao conjunto de instruções mantendo a organização da microarquitetura apresentada.
   (b) Discuta as vantagens e desvantagens desta alteração quanto aos seguintes aspetos: *i*) o endereçamento da memória de dados, *ii*) a densidade do código e *iii*) o impacto no desenho da microarquitetura.

```
        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]
loop:
        b     loop
```

Listagem 1: Programa de teste.

3.5 Implementação e simulação no Logisim

Considere a descrição da microarquitectura do processador disponibilizada no ficheiro "ac_2223i_tp2.circ".

1. No `Logisim`, implemente os subcircuitos `Ext`, `LExt` e `RExt` com base na análise apresentada no ponto 2 da secção 3.1.
2. No `Logisim`, implemente o subcircuito `Interconnections` tendo em conta o resultado indicado no ponto 1 da secção 3.2.
3. No `Logisim`, implemente o subcircuito `Instruction Decoder` com base no projeto apresentado no ponto 2 da secção 3.3.
4. No `Logisim`, programe a memória de código com o código máquina produzido no ponto 2 da secção 3.4. Programe também as primeiras quatro posições da memória de dados com valores à sua escolha.
5. No `Logisim`, execute o troço de código e, para cada uma das instruções, registe as alterações ocorridas nos registos do processador (`r0` a `r7`, `PC` e `CPSR`) e na memória de dados.

4 Avaliação

O trabalho deve ser realizado em grupo e conta para o processo de avaliação da unidade curricular. Na entrega do trabalho, cada grupo deverá submeter na plataforma Moodle os seguintes elementos:

- Um relatório do trabalho realizado, com as respostas às perguntas formuladas no enunciado e as conclusões retiradas do trabalho realizado;
- Nova versão do ficheiro "ac_2223i_tp2.circ" com a implementação de todos os subcircuitos desenvolvidos;
- Ficheiros com o conteúdo das memórias de programa e dados para o programa de teste implementado.

A data limite para a entrega dos trabalhos é sete de novembro de 2022. Após esta entrega, o docente responsável pela lecionação das aulas teórico-práticas combinará com cada grupo de alunos uma data e hora para a realização da apresentação do trabalho.