Relatório do Trabalho Prático

Thamiris Yamate Fischer

***Resumo*—O seguinte relatório tem como intuito descrever as decisões de implementação do processador simplificado inspirado na arquitetura MIPS, tanto no circuito quanto no código Assembly.**

I. Circuito

A. Main: Botão de Reset

O botão de reset no PC tem a finalidade de zerar o programa descrito na Memória de Instruções, assim facilitando a execução de testes.

B. Main: "Debug"

Os displays que permitem a visualização da instrução e seus componentes (opcode, a, b, c), assim como a posição atual do PC (SaidaPC), o conteúdo dos registradores (A, B , "SaidaULA") e os controles da Memória de Controle (ULAOp, ULASrc, WriteRG, enDisplay, PCSrc) são destinados à visualização do que está sendo processado no programa e se o comportamento está de acordo com o previsto (Tabela 1 e Tabela 2).

C. Main: Memória ROM

Tanto na Memória de Instruções quanto na Memória de Controle a codificação das Tabelas I e II foram transcritas para hexadecimal em um arquivo txt a parte para a leitura do Logisim

D. FullAdder: Si

Ao executar alguns testes, usar apenas uma porta XOR para o Si levava a resultados fora do comum para esse circuito, por isso, apesar de ser mais custoso o uso de duas portas, nessa implementação é usada duas portas XOR.

II. Memória de Instruções

A. Formato de Instrução

Para tornar a implementação mais simples, nesse processador é utilizado apenas um formato de instrução: OpCode(4) - Ra(4) - Rb(4) - Rc(4) - Imm(16). As instruções não utilizam todos os campos, por isso, por padrão, se o campo não utilizado é um registrador será utilizado o R0 já que esse não pode ser alterado, nem será usado. Já se o campo não utilizado é o imediato, esse será 0, visto que também não será usado.

III. Memória de Controle

A. Controles

1) ULAOP: Seleciona a operação da ULA conforme especificado no enunciado;

2) ULASrc: Seleciona se a operação da ULA irá considerar o registrador "B" ou o imediato da instrução. Sendo, 0 $\rightarrow$ registrador B e 1 $\rightarrow$ imediato.

3) WriteRG: Permite, ou não, a escrita no registrador de destino (rc). O resultado a ser escrito vem da ULA através da flag "SaidaULA" e é selecionado qual o registrador através da flag "c".

4) enDisplay: Permite, ou não, a visualização de ra no display. Sendo, 0 $\rightarrow$ não visualizar e 1 $\rightarrow$ visualizar.

5) PCSrc: Seleciona o desvio, PC+1 ou PC+imm (branch e jump). Para ambos os casos, é calculado na ULA a diferença entre ra e rb, se for 0, a flag "ZeroULA" será 1 e, assim, será possível o desvio.

- Obs1: É considerado PCSrc e ZeroULA para ser possível verificar essa diferença na ULA e evitar que qualquer operação que resulte em 0 ative o desvio.
- Obs2: A lógica do jump é semelhante ao branch, mesmo não necessitando, para aproveitar o mesmo formato de instrução. No jump, ra = rb = 0, então "ZeroULA" sempre será 1, então sempre o desvio será realizado.

B. Halt

Nas especificações do trabalho, no halt PC = PC+0, então o PCSrc e ULAOP são ativados, mesmo que para uma soma com 0.

IV. Resultados Esperados

Ao executar o programa em C disponibilizado para testes, os resultados esperados eram:
Soma (for): 7
Soma (for): 32
Soma (for): 75
Soma (for): 136
Soma (for): 215
Soma (for): 312
Soma (for): 427
Soma (for): 560
Soma (for): 711
Soma (for): 880
Soma: 880

Executando o programa em Assembly implementado na Memória de Instruções, os resultados mostrados no display estão corretos.

Tabela I
CÓDIGO DE TESTE - FORMATO DE INSTRUÇÃO

End	OpCode	Ra	Rb	Rc	Imm
0	0000	0000	0000	0000	0000000000000000
1	1001	0000	0000	0001	0000000000001010
2	1001	0000	0000	0010	0000000000000000
3	1001	0000	0000	0011	0000000000000111
4	1001	0000	0000	0100	0000000000010010
5	1001	0000	0000	0000	0000000000000000
6	1100	0010	0001	0000	0000000000000111
7	0100	0010	0100	0110	0000000000000000
8	0010	0110	0011	0110	0000000000000000
9	0010	0101	0110	0101	0000000000000000
10	1101	0101	0000	0000	0000000000000000
11	1001	0010	0000	0010	0000000000000001
12	1001	0000	0000	0000	1111111111111010
13	1101	0101	0000	0000	0000000000000000
14	1110	0000	0000	0000	0000000000000000

Tabela II
CÓDIGO DA MEMÓRIA DE CONTROLE

Inst	OpCode	ULAOp	ULASrc	WriteRG	EnDisplay	PCSrc
0(nop)	0000	000	0	0	0	0
1(li)	0001	000	1	1	0	0
2(add)	0010	000	0	1	0	0
3(sub)	0011	001	0	1	0	0
4(mult)	0100	010	0	1	0	0
5(and)	0101	011	0	1	0	0
6(or)	0110	100	0	1	0	0
7(xor)	0111	101	0	1	0	0
8(sll)	1000	110	0	1	0	0
9(addi)	1001	000	1	1	0	0
10(subi)	1010	001	1	1	0	0
11(jump)	1011	001	0	0	0	1
12(branch)	1100	001	0	0	0	1
13(show)	1101	000	0	0	1	0
14(halt)	1110	000	1	0	0	1

Tabela III
CÓDIGO DE TESTE - ASSEMBLY

End	Label	Assembly
0		nop //início do programa
1		addi R1, R0, 10 // n = 10
2		addi R2, R0, 0 // i = 0
3		addi R3, R0, 7 // a = 7
4		addi R4, R0, 18 // d = 18
5		addi R5, R0, 0 // soma = 0
	loop:	
6		branch R2, R1, 7 // 6+7 = 13(fim loop)
7		mult R6, R2, R4 // i*d
8		add R6, R3, R6 // a + i*d
9		add R5, R5, R6 // soma = soma + a + i*d
10		show R5
11		addi R2, R2, 1 // i ++
12		jump -6 // 12 - 6 = 6(loop)
	fim loop:	
13		show R5 // mostrar resultado final
14		halt // fim do programa