Relatório do Trabalho Prático de Projetos Digitais e Microprocessadores - CI1210

Andrieli Luci Gonçalves

25 de novembro de 2024

1 Introdução

Este relatório descreve o processo de construção de um processador simplificado, baseado na Arquitetura MIPS. O objetivo foi implementar um processador capaz de buscar e executar instruções lógico-aritméticas e de controle de fluxo a cada ciclo. Para tal, foi utilizado o simulador *Logisim Evolution 3.8.0*.

As instruções implementadas foram criadas para calcular a soma dos termos de uma progressão aritmética (PA), demonstrando o funcionamento do processador em relação à Arquitetura de Conjunto de Instruções (ISA).

2 Microprocessador

O projeto do processador é composto por dois blocos principais: o circuito de dados, que inclui a Unidade Lógica e Aritmética (ULA) e o banco de registradores; e o circuito de controle, que abrange o contador de programa (PC), a memória de instruções e a memória de controle – além de circuitos auxiliares em ambos.

2.1 Banco de registradores

O componente foi construído com 16 registradores de 32 bits, formados pela união de oito registradores de quatro bits, cada um composto por quatro *flip-flops* tipo D com *Enable*.

A escrita no banco de registradores é controlada por um demultiplexador com 16 saídas. Ele possui uma entrada $WriteReg$ que habilita a escrita e um seletor $C_{0:3}$ para escolher o registrador onde a escrita acontecerá.

A leitura utiliza dois multiplexadores de 32 bits, cujos seletores de quatro bits, RA e RB, indicam os registradores a serem lidos.

2.2 Contador de programa (PC)

O contador de programa (PC) foi implementado de forma similar aos componentes do banco de registradores, exceto pela não utilização do *Enable* nos flip-flops. Devido à limitação de 24 bits nos endereços da memória de instruções no *Logisim Evolution*, o PC possui em sua saída um distribuidor que coleta apenas os seus 24 bits menos significativos.

Sua função é controlar o fluxo das instruções, indicando a posição atual de execução na sequência programada.

2.3 Memória de instrução

A memória de instrução foi implementada utilizando o componente ROM do *Logisim Evolution*, com larguras de 24 bits para endereços e 32 bits para dados. Na memória ROM, foram armazenadas as instruções do código em C que calcula a soma de uma progressão aritmética, convertidas previamente para Assembly, binário e hexadecimal, nesta ordem.

2.3.1 Conversão para Assembly

```
main:
    li r1, 10    ; N - numero de termos
    li r2, 7     ; a - termo inicial
    li r3, 18    ; d - razao
    li r4, 0     ; soma - soma da PA

    li r5, 0     ; contador
loop:
    branch r5, r1, fim_loop
    mult r6, r5, r3
    add r6, r6, r2
    add r4, r4, r6
    show r4

    addi r5, r5, 1
    jump loop
fim_loop:
    show r4
    halt
```

Para as conversões binárias, definiu-se um código para cada uma das instruções, iniciando em 0000 para o *nor* e indo até 1110 em *halt*. Além disso, cada um dos 16 registradores também foi contemplado com um código, seguindo a contagem padrão em binário. O mesmo foi realizado com os valores imediatos.

2.4 Memória de controle

Para a memória de controle, também foi utilizado o componente ROM da ferramenta; neste caso, a largura de endereço equivale a quatro bits e a largura de dados, nove bits. Desse modo, é possível armazenar os sinais de controle necessários para guiar a execução das instruções. Tais sinais foram definidos considerando a funcionalidade de cada instrução:

- $ALUSrc$: ULA deve receber um operando imediato ou proveniente de um registrador;
- $ALUOp$: operação a ser realizada pela ULA;
- $Branch$: salto condicional (com base em uma comparação entre dois valores);
- $Jump$: salto incondicional;
- $WriteReg$: controla se o resultado da operação será armazenado em um registrador;
- $ImmToReg$: controla se o dado a ser carregado em um registrador provém de um valor imediato ou da ULA;
- $EDisplay$: determina quando o resultado aparecerá em tela.

2.5 Unidade Lógica e Aritmética (ULA)

A Unidade Lógica e Aritmética é composta por sete instruções. A primeira (000) e a segunda (001), que representam respectivamente a soma e a subtração, são formadas por somadores de 32 bits (gerados a partir da combinação de somadores de 1 bit). Para multiplicação (010), AND (011), OR (100) e XOR (101), utilizou-se diretamente os componentes do *Logisim Evolution*.

A operação de deslocamento à esquerda (110) foi implementada com 32 multiplexadores. A entrada $A_{0:31}$ representa o dado a ser deslocado e $B_{0:31}$, a quantidade de bits de deslocamento, sendo considerados apenas os cinco bits menos significativos de B para a operação. No multiplexador correspondente à saída $S31$, todas as entradas recebem os bits de A. Para $S30$, a última entrada é substituída por 0. Esse padrão continua até $S0$, onde apenas a primeira entrada contém dados de A, com as demais em 0. Para deslocamentos superiores a 31 bits, foi criado um circuito combinacional apenas com a porta lógica OR que verifica se qualquer bit posterior ao quinto de B é 1.

Na região externa da ULA, um multiplexador estabelece a operação a ser realizada com base no seletor $Funct_{0:3}$, definindo a saída S. Além disso, há uma saída chamada $Zero$, que é gerada pela negação de todos os bits de S combinada por 11 portas lógicas AND – o propósito desta *flag* é indicar se a saída é igual (ou não) a zero.

2.6 Circuitos auxiliares

Os extensores foram moldados a partir de distribuidores: A entrada $A_{0:15}$ é expandida em 16 partes de 1 bit, ao passo que a saída $B_{0:31}$ é expandida em 32 partes de 1 bit. No caso do extensor de zero, os bits adicionais são preenchidos com 0. Já no extensor de sinal, o bit mais significativo de A é replicado para completar a saída B.

Para o funcionamento do processador, também foram utilizados dois somadores de 32 bits, ambos conectados ao PC. O primeiro somador tem como operandos $A_{0:31} = 1$ e $B_{0:31} =$ endereço atual do PC; dessa forma, seu objetivo é incrementar em uma unidade o PC para que a próxima instrução aconteça. O segundo somador tem como operandos $A_{0:31} =$ Const com sinal extendido e $B_{0:31} =$ endereço atual do PC; assim, sua funcionalidade é provocar saltos (condicionais ou incondicionais) ao decorrer da execução.

Também associados à atuação do PC e ao controle dos saltos, utilizou-se duas portas lógicas (AND e OR): a primeira é ligada quando $Branch$ está ativo e o resultado contido na ULA equivale a zero; a segunda é ligada quando a primeira porta ou a *flag* $Jump$ estão ativas.

Ademais, para exibir os resultados parciais e finais da soma da progressão aritmética, foi usado um componente de *display* do próprio *Logisim Evolution*. Para garantir que o resultado permanecesse visível durante toda a execução, houve a adição de um registrador para armazenar o estado anterior do *display*.

Por fim, trabalhou-se também com três multiplexadores, todos associados à seleção de conteúdos específicos com base nos sinais de controle do microprocessador.