Trabalho Prático de Projetos Digitais e Microprocessadores - CI1210 - 2024/2

Prof. Giovanni Venâncio

1 Implementando um Processador

Sua tarefa é construir um processador simplificado baseado na arquitetura MIPS. A cada ciclo, o processador busca e executa uma instrução de lógica/aritmética, ou uma instrução de controle de fluxo (salto/desvio). Este trabalho consiste de duas etapas principais, descritas a seguir.

A Etapa 1 consiste no projeto do processador, no qual você deverá construir o bloco operativo do processador (circuito de dados e de controle) de forma que ele possa executar todo o conjunto de instruções (ISA - *Instruction Set Architecture*). Na Etapa 2, você deverá desenvolver um programa em *Assembly* e executá-lo na implementação do seu processador.

As Etapas 1 e 2 são descritas em detalhes a seguir.

1.1 Etapa 1: Projeto do Processador

Conjunto de Instruções:

O processador deve ser capaz de executar o conjunto de instruções definido na tabela abaixo:

Tabela 1: Conjunto de instruções do processador.

opcode	Instrução	Semântica	Comentário
0	*nop*	–	Sem operação
1	*li c, imm*	R(c) ← imm	Carrega imediato no registrador
2	*add c, a, b*	R(c) ← R(a) + R(b)	Soma entre dois registradores
3	*sub c, a, b*	R(c) ← R(a) - R(b)	Subtração entre dois registradores
4	*mult c, a, b*	R(c) ← R(a) * R(b)	Multiplicação entre dois registradores
5	*and c, a, b*	R(c) ← R(a) AND R(b)	Operador AND lógico
6	*or c, a, b*	R(c) ← R(a) OR R(b)	Operador OR lógico
7	*xor c, a, b*	R(c) ← R(a) XOR R(b)	Operador XOR lógico
8	*sll c, a, b*	R(c) ← R(a) « R(b)	Deslocamento lógico à esquerda
9	*addi c, a, imm*	R(c) ← R(a) + extZero(imm)	Soma com constante lógica
10	*subi c, a, imm*	R(c) ← R(a) - extZero(imm)	Subtração com constante lógica
11	*jump imm*	IP ← E	Salto incondicional. E ← IP + extSign(imm)
12	*branch a, b, imm*	if (R(a) == R(b)) {IP ← E}	Salto condicional. E ← IP + extSign(imm)
13	*show a*	Display R(a)	Exibe valor do registrador na saída
14	*halt*	IP ← IP + 0	Finaliza a execução

Na Tabela 1, *a*, *b*, *c* são nomes de registradores (endereços dos registradores), enquanto que R(a), R(b) e R(c) são os conteúdos dos respectivos registradores. Observe que as instruções não possuem formato.

Tabela 2: Operações da ULA.

Seletor	Operação
0	Somador: A + B
1	Subtrator: A - B
2	Multiplicador: A * B
3	AND: A AND B
4	OR: A OR B
5	XOR: A XOR B
6	Deslocamento à esquerda: A « B

Unidade Lógico Aritmética (ULA)

A ULA executa as operações de lógica e aritmética definidas para as instruções descritas acima. A ULA deverá ser capaz de processar as operações descritas na Tabela 2.

A ULA possui três entradas: A e B com 32 bits cada e um seletor, indicando a operação a ser realizada. A saída da ULA será o resultado com 32 bits, e um bit indicando se houve saída igual a zero em todos os bits (Saída ZERO).

Bloco Operativo

O bloco operativo do processador, composto pelo circuito de dados e o circuito de controle, é ilustrado na Figura 1.

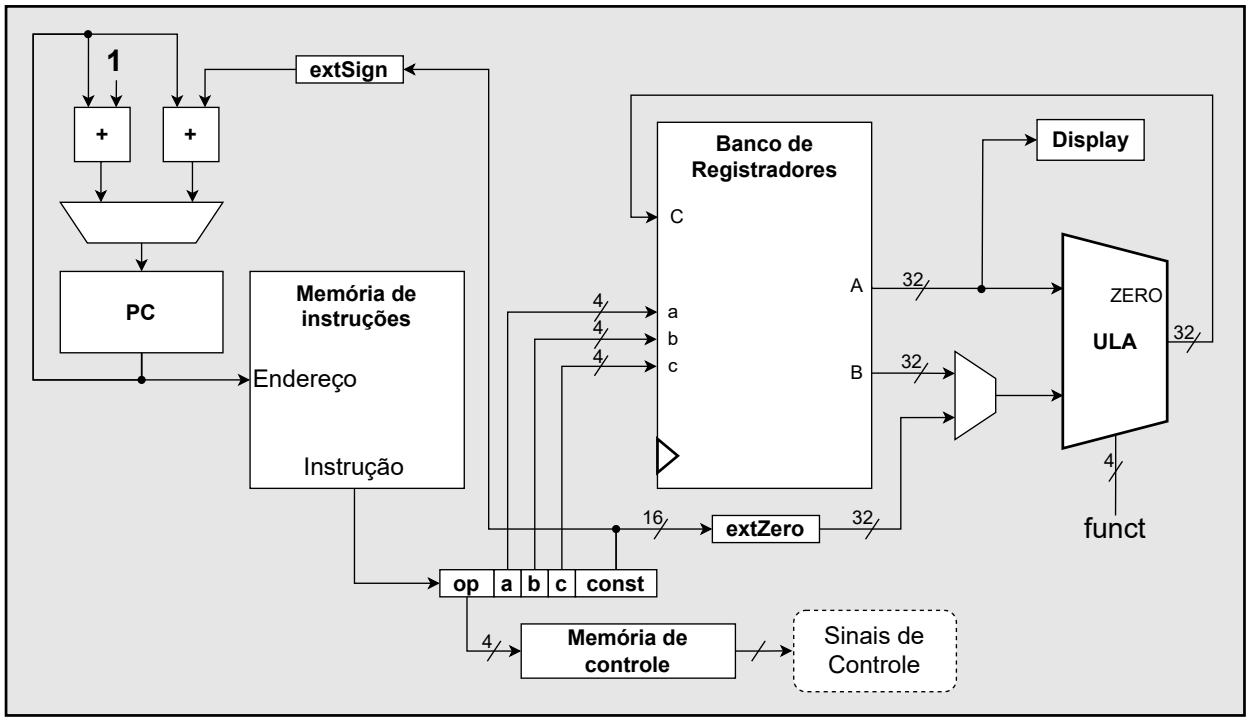

Figura 1: Bloco operativo: circuito de dados e de controle.

O circuito de dados consiste de uma ULA, um banco de registradores e de circuitos auxiliares (*e.g.,* somador, extensor de sinal, extensor de zero). Já o circuito de controle consiste de um apontador de instrução (registrador *Program Counter* - PC), memória de instruções, memória de controle e de circuitos auxiliares (*e.g.,* somador, comparador de igualdade).

O bloco de registradores deve conter 16 registradores de 32 bits, duas portas de leitura (A e B) e uma porta de escrita (C). O registrador 0 contém a constante zero (R(0) = 0), e escritas neste registrador não devem surtir nenhum efeito.

A memória de instruções contém as instruções (codificadas em binário) que o processador deverá executar. Já a memória de controle configura os sinais de controle (*e.g.,* seletores dos muxes, banco de registradores, etc.) de acordo com a operação a ser realizada. Para construir a memória de programa e memória de controle você deverá consultar os vídeos disponibilizados no Moodle (Trabalho - Vídeos auxiliares).

1.2 Etapa 2: Executando o Código Assembly

Uma vez que o processador esteja implementado, você deverá testar o seu funcionamento através da execução de um código teste. O programa de teste deverá imprimir todas as etapas de soma de uma Progressão Aritmética (PA), como definido no somatório a seguir:

$$S = \sum_{i=0}^{N-1} (a + i \cdot r)$$

Ou como definido na expressão expandida a seguir:

$$S = a + (a + r) + (a + 2r) + \ldots + (a + (N - 1)r)$$

Onde:

- N representa o número de termos;
- a representa o termo inicial;
- r representa a razão da PA.

Todas as etapas da soma da PA deverão ser impressas no *display* através da instrução *show*. Ao final, o programa deverá encerrar com a instrução *halt*.

Para esta etapa você deve:

1. Traduzir o programa em C (disponível no Moodle) para o *Assembly* do seu processador. Não serão permitidas alterações e/ou simplificações no código C fornecido, exceto a modificação dos valores de inicialização (*i.e.,* número de termos, termo inicial, etc.) para fins de teste;
2. Traduzir o *Assembly* para código binário e editar a memória de instruções.

Importante: O *Assembly* gerado deve ser referente ao seu processador, de acordo com as instruções definidas na Tabela 1.

2 Regras Gerais de Entrega, Apresentação e Avaliação

A implementação poderá ser feita no simulador *Digital* ou *Logisim-Evolution*. A implementação deve funcionar nos servidores do DInf. É permitido utilizar os seguintes componentes pré-existentes:

- Portas lógicas: AND, OR, NOT, etc.;
- Aritmética: Multiplicador 32 bits;

- Memórias: RAM, ROM, FF;
- Plexers: MUX, DEMUX, ENCODER, DECODER;
- Fios: túnel, *splitter*.

O trabalho é individual e a entrega deverá ser feita através do Moodle da disciplina.
A entrega consiste dos seguintes itens:

- Projeto do processador (arquivo .dig ou .circ);
- Relatório do trabalho.

O relatório deve ser entregue em formato PDF e conter no máximo 2 páginas A4 (frente e verso) com fonte tamanho mínimo 11 pontos. O relatório deve conter:

- Nome;
- Breve discussão sobre decisões de implementação (inclua descrições de *bugs*, caso exista);
- O código *Assembly* gerado a partir da tradução do código C.

Os trabalhos deverão ser apresentados de forma oral pelo aluno. A nota irá considerar domínio do tema, robustez da solução e rigorosidade da metodologia. A não apresentação do trabalho acarretará em nota igual a **Zero**.

A cópia do trabalho (plágio) não será tolerada e acarretará em nota igual a **Zero** para todos os envolvidos.