UNIVERSIDADE DE BRASÍLIA
INSTITUTO DE CIÊNCIAS EXATAS
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

116394 ORGANIZAÇÃO E ARQUITETURA DE COMPUTADORES

Projeto da Disciplina: RISC-V Uniciclo em VHDL

OBJETIVO

O projeto da disciplina consiste no desenvolvimento de uma versão do processador RV Uniciclo em VHDL. A ferramenta utilizada para o desenvolvimento do projeto deverá ser o ModelSim-Altera ou o EdaPlayground.

DESCRIÇÃO

O diagrama esquemático de referência para a arquitetura básica do RISC-V Uniciclo é apresentado na figura 1.

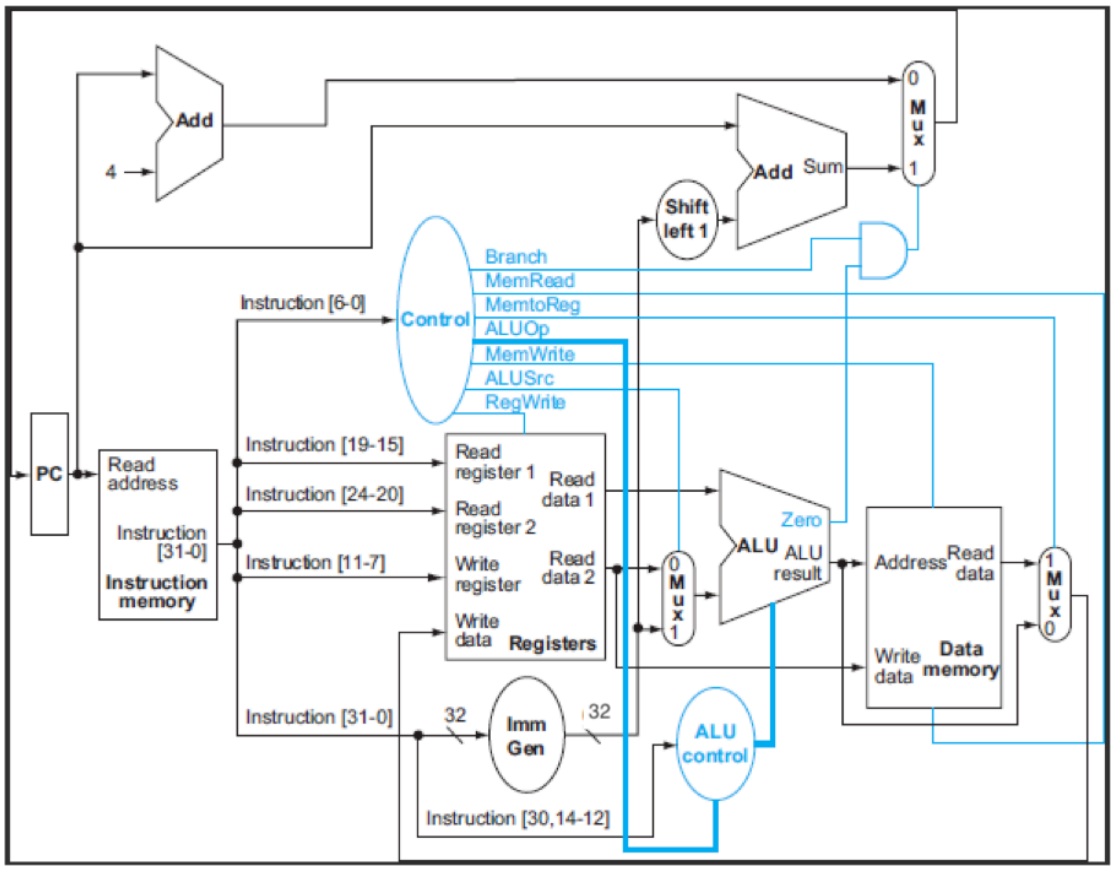

Figura 1. RISC-V Uniciclo.

Este diagrama não está completo. Por exemplo, não permite implementar as operações JAL e JALR, que requerem o salvamento de PC+4 em um registrador e ainda, no caso de JALR, o cálculo do endereço de salto.

A implementação VHDL consiste na descrição de cada módulo e sua interligação através de sinais.

A parte operativa do RISC-V é 32 bits, ou seja, os dados armazenados em memória, os registradores, a unidade lógico-aritmética, as instruções e as conexões utilizam 32 bits.

No caso das memórias, sugere-se utilizar apenas 12 bits de endereço, de forma a manter compatibilidade com o espaço de endereçamento da configuração compacta do RARS, onde o segmento de código começa no endereço zero e o segmento de dados começa no endereço 0x2000.

Módulos principais:

- **PC**: contador de programa. É um registrador de 32 bits. Entretanto, pelas restrições de memória adotadas, apenas o número de bits necessário deve ser enviado à memória de instruções, sendo o restante ignorado; é carregado a cada transição de subida do relógio;
- **Memória de Instruções (MI)**: armazena o código a ser executado. As instruções são de 32 bits. O espaço de endereçamento é reduzido (ex: 12 bits). Cada endereço da MI armazena uma instrução de 32 bits. Idealmente, a MI deve funcionar como um bloco combinacional neste projeto, ou seja, necessita-se apenas do endereço para ler a instrução, sem sinais adicionais de controle. Essa memória não permite o endereçamento a byte. Desta forma, se forem utilizados 12 bits de endereço de *byte* (PC(11:0)), deve-se utilizar os bits 2 a 11 do PC como endereço de instrução (1k word endereçável).
- **Banco de Registradores (XREG)**: é constituído por 32 registradores de 32 bits. O registrador índice zero, XREG[0], é uma constante. Sua leitura retorna sempre zero, e não pode ser escrito. O XREG tem duas entradas de endereços, permitindo a leitura de 2 registradores de forma simultânea. Uma terceira entrada de endereço é utilizada para selecionar um registrador para escrita de dados. A escrita de um dado em registrador ocorre na transição de subida do relógio. A escrita de um dado no XREG é controlada pelo sinal RegWrite: RegWrite = '1' habilita a escrita.
- **Unidade Lógico-Aritmética (ULA)**: opera sobre dados de 32 bits. Provê o resultado em 32 bits, juntamente com o sinal **ZERO**, que indica que o resultado da operação realizada é zero.
  - Operações implementadas na ULA são aquelas definidas no respectivo trabalho de aula.
    - Obs: para as instruções de deslocamento, sugere-se utilizar as funções disponíveis no pacote *numeric_std* : *shift_left* e *shift_right*, com o tipo apropriado de dados para o deslocamento lógico/ aritmético.
- **Memória de Dados (MD)**: armazena os dados do programa. Pode ser lida ou escrita. Neste projeto, a MD pode ser acessada em nível de ***byte*** ou de ***word.*** No caso de ***byte***, a leitura pode ser com extensão de sinal ou não. Considerando que a MD é reduzida, deve-se selecionar apenas o número necessário de bits de endereço (14 bits para endereçar 16KB). Os bits de

endereço selecionados para acessar a MD devem permitir a leitura do segmento de dados conforme o modelo de memória compacto do RARS, onde o endereço base é 0x00002000. A memória é escrita na subida do relógio, quando o sinal de controle MemWrite estiver acionado. O sinal de leitura LerMem não é necessário na implementação uniciclo;

- **Multiplexadores 2 para 1**: são utilizados 4 multiplexadores com 2 entradas de 32 bits e uma saída de 32 bits;
- **Somadores**: são utilizados 2 somadores de 32 bits para operar com endereços;

Tabela 1: Operações da ULA

Operação	Significado	OpCode
ADD A, B	Z recebe a soma das entradas A, B incluindo o vem-um	0000
SUB A, B	Z recebe A - B	0001
AND A, B	Z recebe a operação lógica A and B, bit a bit	0010
OR A, B	Z recebe a operação lógica A or B, bit a bit	0011
XOR A, B	Z recebe a operação lógica A xor B, bit a bit	0100
SLL A, B	Z recebe a entrada A deslocada B bits à esquerda	0101
SRL A, B	Z recebe a entrada A deslocada B bits à direita sem sinal	0110
SRA A, B	Z recebe a entrada A deslocada B bits à direita com sinal	0111
SLT A, B	Z = 1 se A < B, com sinal	1000
SLTU A, B	Z = 1 se A < B, sem sinal	1001
SGE A, B	Z = 1 se A ≥ B, com sinal	1010
SGEU A, B	Z = 1 se A ≥ B, sem sinal	1011
SEQ A, B	Z = 1 se A == B	1100
SNE A, B	Z = 1 se A != B	1101

INSTRUÇÕES A SEREM IMPLEMENTADAS

- Geração de constantes: AUIPC, LUI (K)
- Aritméticas: ADD, SUB (A)
- Aritméticas com imediato: ADDi (Ai)
- Lógicas: AND, SLT, OR, XOR (L)
- Lógicas com imediato: ANDi, ORi, XORi (Li)
- Shift: SLL, SRL, SRA (S)
- Shift com imediato: SLLi, SRLi, SRAi (Si)
- Comparação: SLT, SLTu (C)
- Comparação com imediato: SLTi, SLTUi (Ci)
- Subrotinas: JAL, JALR (J)

- Saltos 1: BEQ, BNE (S1)
- Saltos 2: BLT, BGE (S2)
- Saltos 3: BGEU, BLTU (S3)
- Memória: LW, LB, LBU, SW, SB (M)

VERIFICAÇÃO DO RISC-V UNICICLO

O processador implementado deve ser capaz de executar um código gerado pelo RARS, utilizando o modelo de memória compacto.

ENTREGA

- Código VHDL do projeto
- Telas de simulação do ModelSim/EdaPlayground, demonstrando a execução correta das instruções
- Relatório descrevendo a implementação e testes realizados

Entregar no Moodle em um arquivo compactado.