

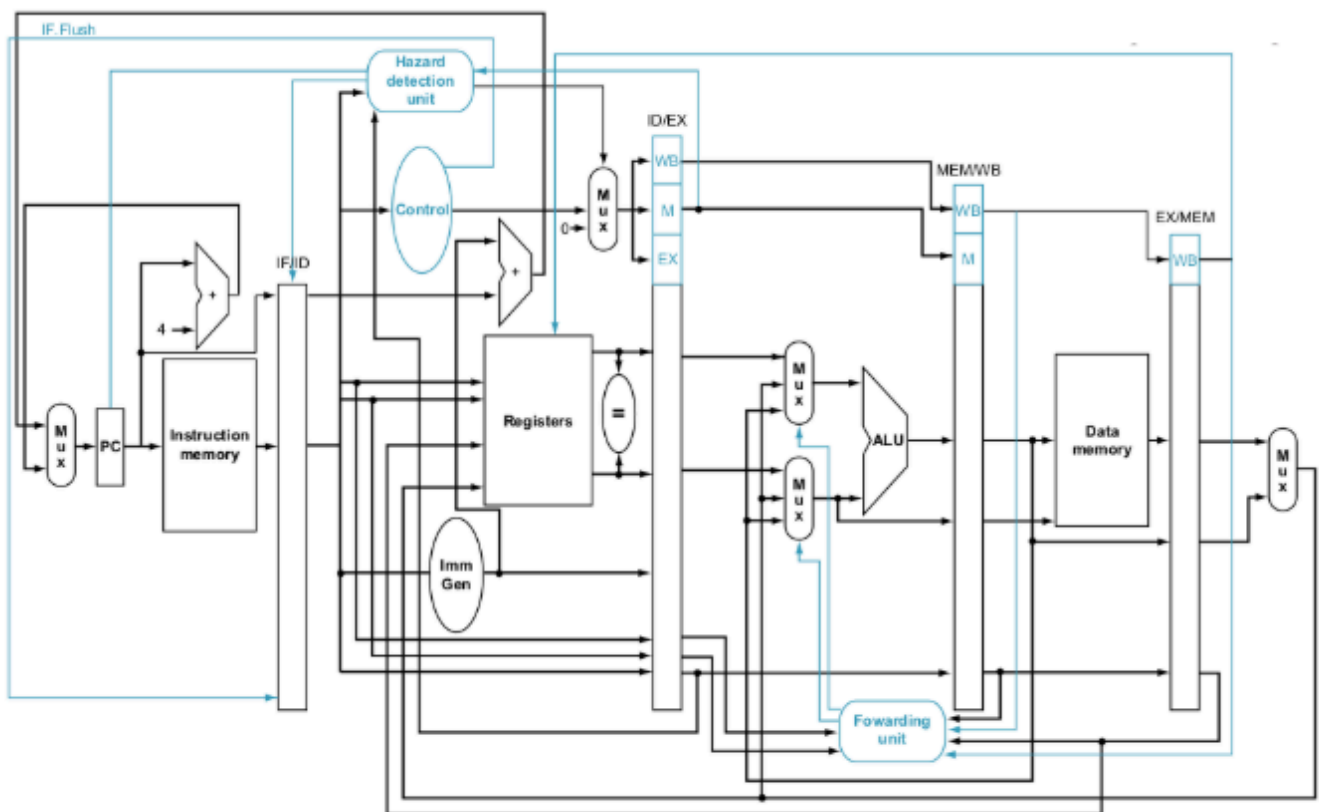


UFOP

Trabalho prático – RISC V
CSI509 – Organização e Arquitetura de computadores II
Engenharia da computação - Campus UFOP-João Monlevade
Prof. Racyus Delano Garcia Pacífico



O trabalho consiste na implementação de uma versão simplificada do caminho de dados e controle do RISC-V. Cada grupo deverá implementar um subconjunto diferente de instruções. O ponto de partida para todos os grupos é o caminho de dados apresentado abaixo. Trata-se da Figura 4.66 do livro de referência da disciplina e que implementa as instruções: ADD, SUB, AND, OR, LD, SD, BEQ.



Forma de entrega: Github para todos os arquivos.

O que deve ser entregue:

- Documentação do trabalho prático em formato SBC, com imagens da execução do código (simulado e implementado fisicamente).
- Arquivos fonte Verilog e arquivos de simulação através do GitHub.
- Projeto do Intel Quartus completo para implementação e execução na placa Mercurio IV.

Critérios de avaliação:

- A documentação deverá conter, pelo menos, introdução, desenvolvimento, resultados e considerações finais. Além disso, o texto da mesma deve ser justificado e com referências, todo no formato SBC, caso existam (utilizando LaTeX);
- O trabalho será apresentado dentro da documentação, com uma entrevista em uma data posterior.
- O caminho de dados do grupo passará por casos de testes previamente selecionados pelo professor, de modo que a nota irá sofrer alteração dependendo do resultado obtido;
- Cópias de trabalhos práticos serão exemplarmente punidas. A punição será a mesma para quem copiou e para quem forneceu o trabalho prático.
- Mais detalhes do critério de avaliação abaixo:

>>> Implementação:

Na simulação, pelo menos as primeiras 32 posições da memória devem ser exibidas na tela. A Tabela 1 apresenta as instruções a serem implementadas por cada grupo.

Tabela 01 - Subconjunto de instruções RISC-V a ser implementado por cada grupo.

Grupo	Instruções						
	01	02	03	04	05	06	07
01	lb	sb	add	and	ori	sll	bne
02	lh	sh	sub	or	andi	srl	beq
03	lw	sw	add	xor	addi	sll	bne
04	lb	sb	sub	and	ori	srl	beq
05	lh	sh	add	or	andi	sll	bne
06	lb	sb	add	and	ori	sll	beq
07	lb	sb	add	or	and	srl	beq
08	lw	sw	sub	xor	addi	srl	beq
09	lh	sh	sub	or	andi	srl	beq
10	lb	sb	sub	and	ori	srl	beq
11	lb	sb	add	and	ori	sll	bne
12	lh	sh	sub	or	andi	srl	beq
13	lh	sh	sub	or	andi	srl	beq

ADD, SUB, AND, OR, LD, SD, BEQ

Exemplo: Use o simulador RISC-V Interpreter:

<https://www.cs.cornell.edu/courses/cs3410/2019sp/riscv/interpreter/#>

(O simulador apenas suporta instruções para 1 palavra).

Código (**adaptar para o conjunto de instruções do grupo**):

```
# Colocando valor base na memória
# addi x2, x0, 7
# sw x2, 4(x0)

lw x1, 4(x0)
add x2, x1, x0

add x1, x1, x2
add x1, x1, x2
sub x1, x1, x2
sub x1, x1, x2

beq x1, x2, SAIDA

# Caso o fluxo venha para cá, seu processador está errado
add x1, x1, x1
sw x1, 0(x0)

SAIDA:
and x1, x1, x2
or x1, x1, x0
sw x1, 0(x0)
```

Memória:

Endereço 32 (4 bytes) = 7

Resultado previsto:

A posição x1 da memória deve ser preenchida com 7.

>>> Parte 1: Testbench:

Para essa primeira parte, um arquivo de TESTBENCH (ou seja, simulação) deverá ser criado para executar e imprimir no terminal do computador a memória de todos os 32 registradores durante ou ao fim da execução. Um clock automático do testbench poderá ser utilizado para essa parte.

```

Register [    0]:      0
Register [    1]:      7
Register [    2]:      0
Register [    3]:      0
Register [    4]:      0
Register [    5]:      7
Register [    6]:    32768
Register [    7]:      0
Register [    8]:      2
Register [    9]:      0
Register [   10]:      0
Register [   11]:      0
Register [   12]:      0
Register [   13]:      1
Register [   14]:      0
Register [   15]:      0
Register [   16]:     16
Register [   17]:      0
Register [   18]:      0
Register [   19]:      0
Register [   20]:      0
Register [   21]:      0
Register [   22]:      0
Register [   23]:      0
Register [   24]:    4096
Register [   25]:      0
Register [   26]:      0
Register [   27]:      0
Register [   28]:      0
Register [   29]:      0
Register [   30]:      0
Register [   31]:      0

```

Exemplo de saída no terminal após execução da simulação.

>>> Parte 2: Síntese

A criação da síntese para a implementação no FPGA Mercúrio IV, deve ser criado, conectando:

- O clock com um switch ou botão da placa FPGA.
- O reset com um switch ou botão da placa FPGA.
- 1 Display de 7 segmentos mostrando o Program Counter (0-9).

>>> Parte 3: Documentação

- A documentação deverá conter um resumo da implementação do caminho de dados, contendo pelo menos, introdução, desenvolvimento, resultados e considerações finais. O texto tem que ser justificado e com referências, utilizando formato SBC na ferramenta LaTeX.
- Na parte de resultados, fotos contendo a execução do testbench e da implementação física devem ser colocados, explicando também qual foi o código utilizado para o teste.

>>> Parte 4: Entrega do código

No link do código enviado do Github além do código implementado em verilog e o projeto do Quartus, deverá conter quatro arquivos de entrada:

- O código em ASM utilizado para o teste do trabalho.

- O código ASM traduzido em binário que foi implementado e utilizado dentro do processador RISC-V produzido para teste.
- Um arquivo contendo o estado inicial dos registradores do RISC-V.
- Um arquivo contendo o estado inicial da memória de dados (Data Memory).

Observações:

- Podem ser cobrados outros testes parecidos na análise do trabalho.
- Datas de entrega:
 - **Simulação:** 01/08/2025 (10 pontos)
 - **Síntese:** 22/08/2025 (10 pontos)