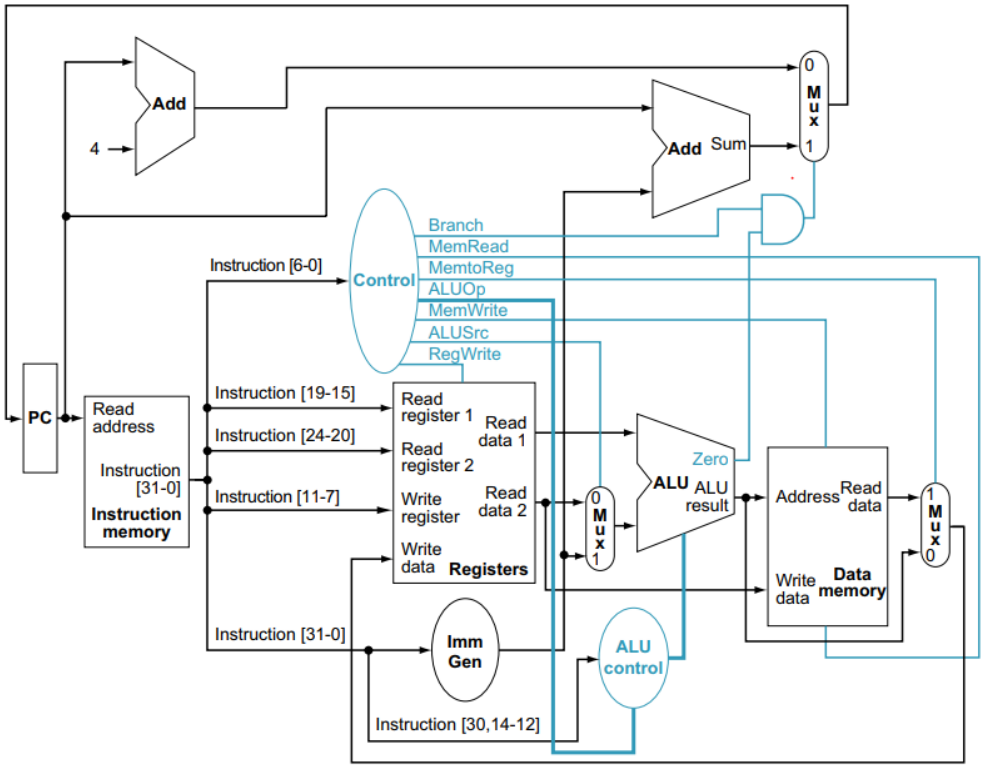
Universidade Federal de Viçosa

Trabalho Prático 02 – Caminho de Dados do RISC-V
CCF 252 – Organização de Computadores I
Ciência da Computação – Campus UFV-Florestal
Prof. José Augusto Miranda Nacif

O trabalho consiste na implementação de uma versão simplificada do caminho de dados 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 4.21 do livro RISC-V utilizado em classe 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 SystemVerilog/Verilog e arquivos de simulação através do GitHub, como no trabalho anterior.
- Projeto do Intel Quartus completo para implementação e execução na placa DE2-115 (Cyclone 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 aluno irá passar 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 do TP***:

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	lw	sw	sub	xor	addi	srl	beq
07	lb	sb	add	and	ori	sll	bne
08	lh	sh	sub	or	andi	srl	beq
09	lw	sw	add	xor	addi	sll	bne
10	lb	sb	sub	and	ori	srl	beq
11	lh	sh	add	or	andi	sll	bne
12	lw	sw	sub	xor	addi	srl	beq

13	lb	sb	add	and	ori	sll	bne
14	lh	sh	sub	or	andi	srl	beq
15	lw	sw	add	xor	addi	sll	bne
16	lb	sb	sub	and	ori	srl	beq
17	lh	sh	add	or	andi	sll	bne
18	lw	sw	sub	xor	addi	srl	beq
19	lb	sb	add	and	ori	sll	bne
20	lh	sh	sub	or	andi	srl	beq
21	lw	sw	add	xor	addi	sll	bne
22	lb	sb	sub	and	ori	srl	beq
23	lh	sh	add	or	andi	sll	bne
24	lw	sw	sub	xor	addi	srl	beq
25	lb	sb	add	and	ori	sll	bne
26	lh	sh	sub	or	andi	srl	beq
27	lw	sw	add	xor	addi	sll	bne
28	lb	sb	sub	and	ori	srl	beq
29	lh	sh	add	or	andi	sll	bne
30	lw	sw	sub	xor	addi	srl	beq
31	lb	sb	add	and	ori	sll	bne
32	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)
```

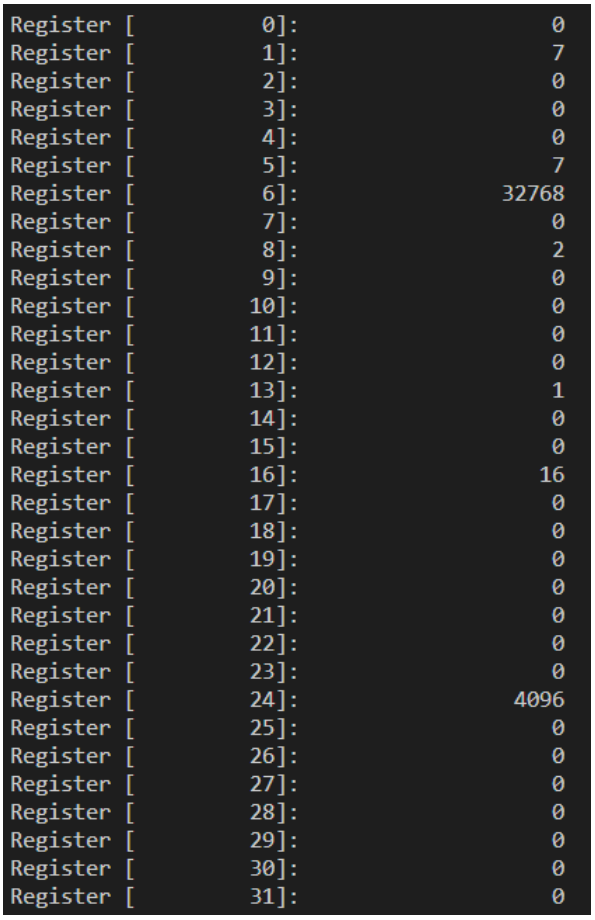
```
lw x1, 4(x0)
add x2, x1, x0

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.



Exemplo de saída no terminal após execução da simulação.

>>> Parte 2: ***Síntese:***

A criação da síntese para a implementação no FPGA DE2-115 (Cyclone IV, Altera), deve ser criado, conectando:

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

>>> Parte 3: ***Pontuação extra***

- Uma pontuação também será dada para a implementação de um registrador (não sendo o registrador x0) em uma fileira de LEDs ou display de 7 segmentos no FPGA.
- Pontuação para a implementação do program counter completo (0-31) com 2 displays de 7 segmentos.

>>> Parte 4: ***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.
- Caso haja partes extras implementadas (Parte 3), a documentação das mesmas mostrando o desenvolvimento é oque foi feito será necessário.

>>> Parte 5: ***Entrega do código.***

No código enviado pelo Github da organização além do código implementado em verilog e o projeto do QUARTUS , deverá conter 4 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 (visto que ADDI não é cobrado pelo caminho de dados implementado, necessitando ter os valores inseridos manualmente.)
- Um arquivo contendo o estado inicial da memória de dados (Data Memory).

>>> Parte 6: ***Entrevista:***

- Será marcado em uma data posterior.

Observação: podem ser cobrados outros testes parecidos com este na análise do trabalho. Utilize o seu montador (TP 01) para conversão para binário antes de executar no seu RISC-V.