Universidade de Brasília

DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO - CIC

GRUPO 5

LABORATÓRIO 4 DE OAC

CPU MIPS MULTICICLO

Nome do Estudante	Matrícula
1. Iago Lobo Ribeiro de Moraes	14/0082921
2. Cristiano Krug Brust	15/0008058
3. José Marcos da Silva Leite	15/0038810
4. Yan Victor dos Santos	14/0033599
5. André Luiz de Moura Ramos Bittencourt	14/0130225

Professor(a):

Marcus Vinícius Lamar

Data : 16/11/2017

1 Processador Multiciclo

Objetivos:

- Treinar o aluno com a linguagem de descrição de hardware Verilog;
- Familiarizar o aluno com a plataforma de desenvolvimento FPGA DE2 da Altera e o software QUARTUS II;
- Desenvolver a capacidade de análise e síntese de sistemas digitais usando uma Linguagem de Descrição de Hardware;
- Apresentar ao aluno a implementação de uma CPU MIPS Multiciclo;

1) (0.0) Abra e compile o projeto do processador MIPS PUM v.5.3 com o Processador Multiciclo

a. Carregue o programa testeWAVEFORM2.s ;

b. Faça a análise do resultado da simulação por forma de onda gerada pelo DE2.vwf;

2) (2.0) Analise o processador Multiciclo desenhando o diagrama de blocos do Caminho de Dados usando a estrutura base vista em aula e a máquina de estados do Bloco Controlador.

3) (1.0) Usando seu programa teste.s, verifique o correto funcionamento de TODAS as instruções da ISA implementada, teste usando simulação por forma de onda e pela implementação na DE2 (filme a execução).

4) (1.0) Execute no processador em FPGA o seu programa de simulação de lançamento de bola de canhão desenvolvido no Laboratório 1 (Dica: defina os parâmetros no seu programa, sem usar syscall 6). Grave vídeos demonstrativos e disponibilize no YouTube com links no relatório.

5) (3.0) Verifique o correto funcionamento do Syscall 49 (leitura do cartão SD). Defina no PC e grave em um cartão SD os 12 cenários do jogo Street Fighter. Faça um programa que leia sequencialmente as telas e as apresente no monitor VGA. Faça comentários sobre as limitações e máxima taxa de quadros atingida. Filme o experimento com links no relatório.

6) (3.0) Implemente as instruções abaixo em conformidade com a ISA MIPS (livro See MIPS Run e Manual do MIPS):

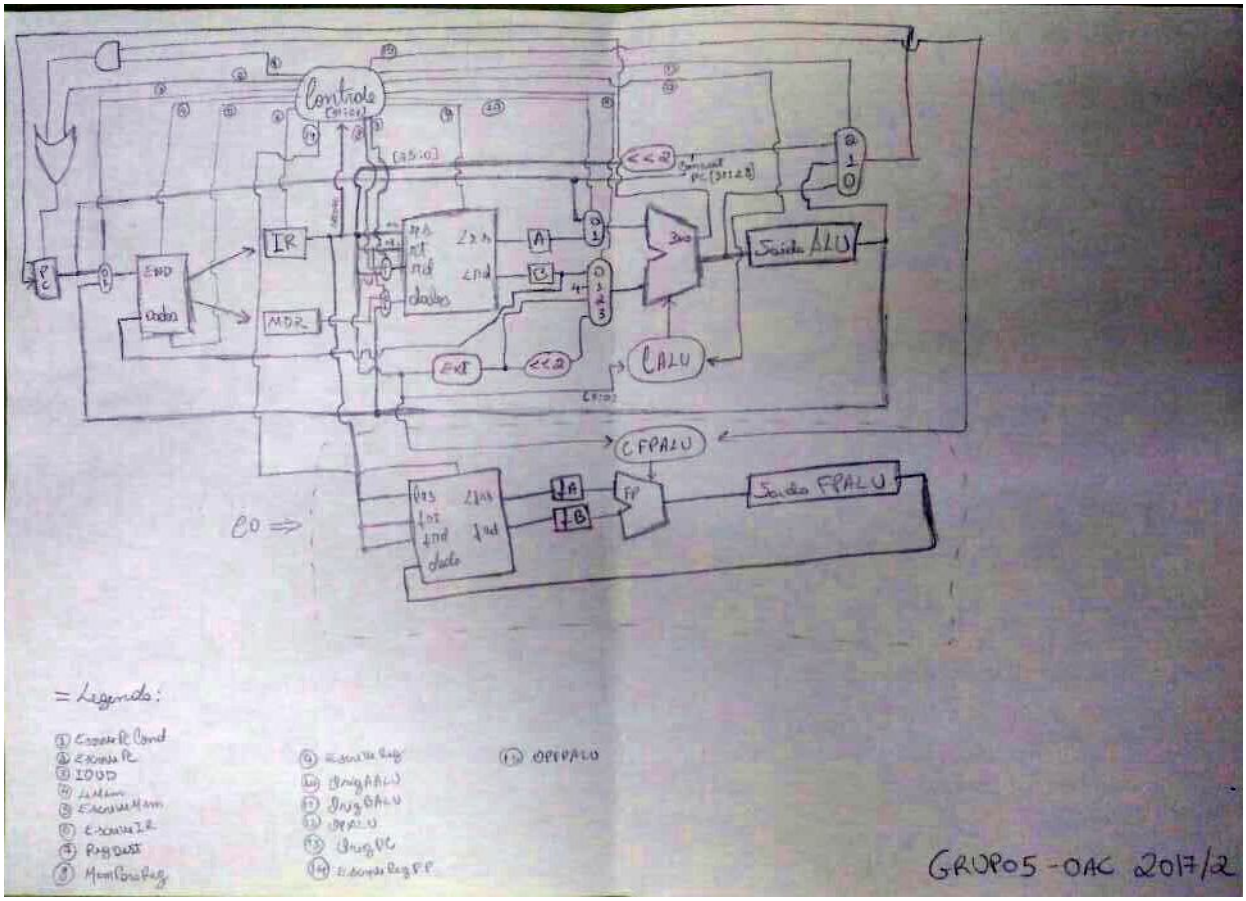
- mul $t1,$t2,$t3 Multiplication without overflow : Set HI to high-order 32 bits, LO and $t1 to low-order 32 bits of the product of $t2 and $t3
- jalr $t1 Jump and link register : Set $ra to Program Counter (return address) then jump to statement whose address is in $t1
- jalr $t1,$t2 Jump and link register : Set $t1 to Program Counter (return address) then jump to statement whose address is in $t2

a. (1.0) Indique as modificações necessárias no caminho de dados

b. (1.0) Indique as modificações necessárias no bloco de controle

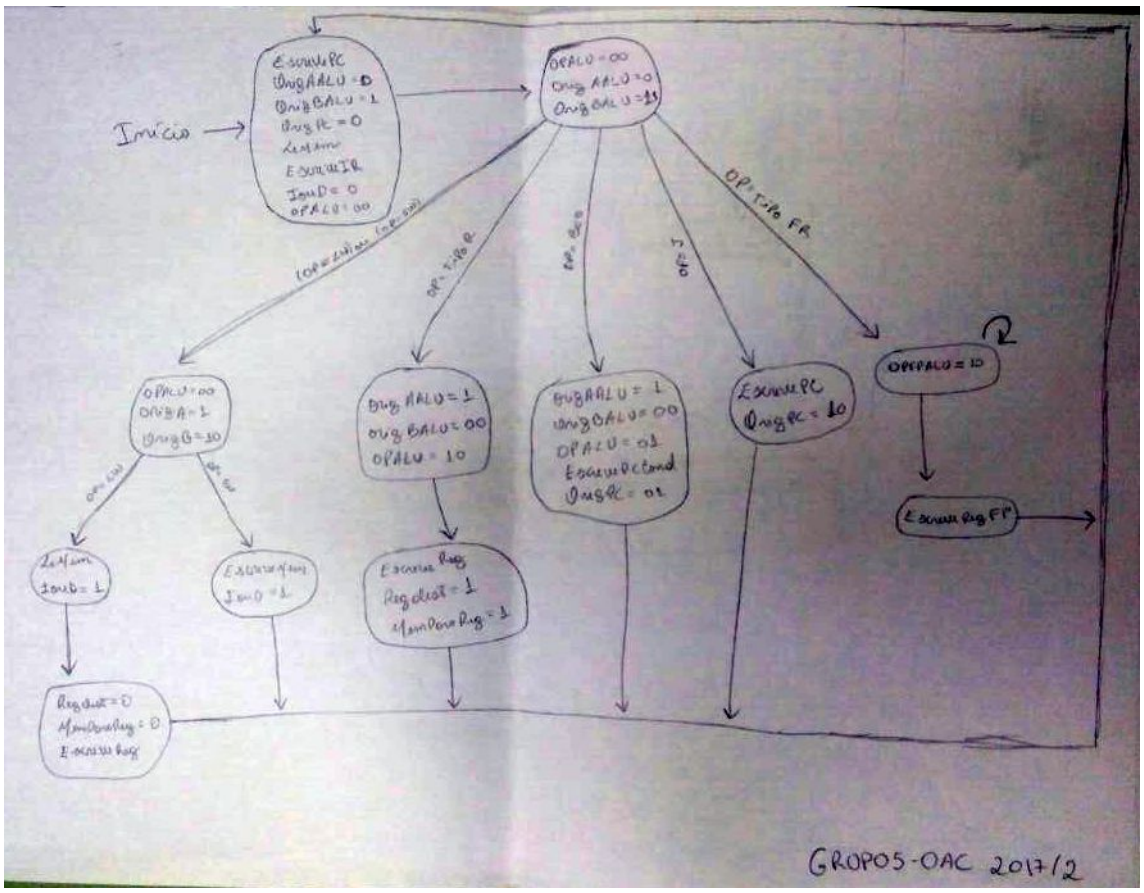
c. (1.0) Crie um programa teste que comprove o correto funcionamento das novas instruções. Faça a simulação em forma de onda e sintetize na DE2.

Respostas:
2) Imagem completa:



Baseando-se na estrutura base do diagrama de blocos visto em aula, foi projetado o caminho de dados das principais intruções da MIPS-PUM v5.3, visto que a complexidade do projeto define uma estrutura complexa. A compreensão do controle, e do projeto, tornou possível a implementação das novas instruções da questão 6 no Multiciclo. Portanto, mesmo que o diagrama não represente fielmente o MIPS-PUM v5.3, abstraimos o fluxo apresentado no diagrama da imagem para alterar de fato o caminho de dados dentro do código do projeto. Como a estrutura base não suporta operações em ponto flutuante, foi necessário inserir esta unidade conforme o projeto do PUM.

A máquina de estados que descreve o controle do diagrama de blocos, facilitou a compreensão do controle descrito em código no QUARTUS II, para a questão 6. Quando uma instrução precisa usar FPALU, é necessário uma grande quantidade de ciclos para completá-la. Portanto, é necessário inserir um *loop* que permite fazer a operação em vários ciclos até que ela encerre. Obs.: As duas abstrações não incluem as instruções criadas na questão 6, uma vez que não ficou claro se deveria ou não estar.



3) Vídeo: https://youtu.be/wY5pPZ5jDgU

Todas as instruções testadas no vídeo comprovam o correto funcionamento da MIPS-PUM v5.3.

4) Vídeo: https://youtu.be/ybW_eWm4-LI

O vídeo acima comprova o funcionamento do programa de simulação da bola de canhão na FPGA.

5)

6) Vídeo para instruções jalr $t1 e jalr $t1,$t2: https://youtu.be/srF56F_eVKw

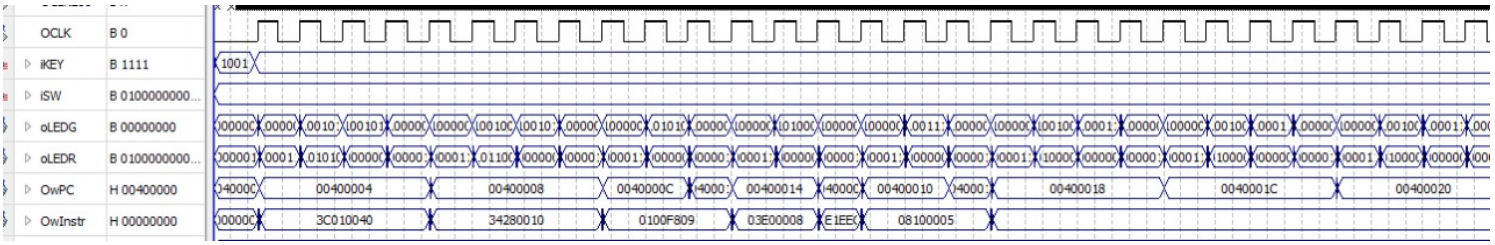
Vídeo para instrução mul $t1,$t2,$t3: https://youtu.be/BJYnl_ZXlYg

O vídeo do link acima comprova o correto funcionamento das instruções *jalr* $t1 e *jalr* $t1, $t2. O programa de teste insere as duas instruções a serem testadas.

a) Para o caminho de dados, utilizamos a implementação já feita das instruções *jal* e *jr*, onde a instrução *jal* coloca o *PC+4* dentro do registrador $ra, e a instrução *jr* (TIPO-R) torna *PC=R[rs]*. Para a instrução *mul*, usamos a instrução *mult* para pegar o valor de *lo* que era passado para o registrador da saída da ULA. Portanto, mais um estado foi criado para gerenciar o controle da escrita dos dados desta saída para o registrador *rd* definido pela nova instrução *mul*.

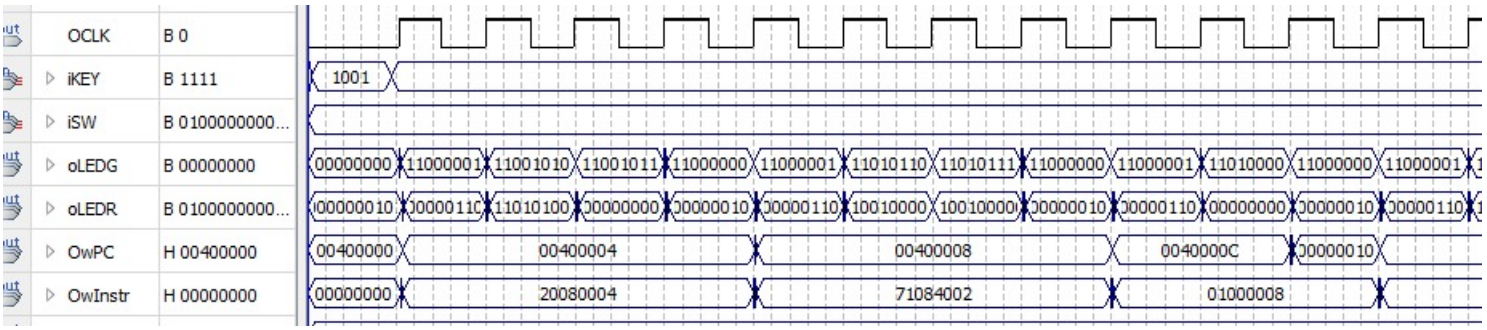
b) Para o bloco de controle, foi necessário habilitar a escrita no banco de registradores para o registrador *rd*, que seleciona se deverá escrever *PC+4* em $ra ou no registrador definido pela instrução *jalr* $t1, $t2. Portanto, o controle *RegDest* ativa a escrita para *rd* e o controle do *MemParaReg* ativa os dados de *PC+4*. Os outros controles permanecem semelhantes às instruções *jr* e *jal*, cujo *PC* é atualizado com o valor de *R[rs]*. Para a instrução *mul*, foi necessário criar mais um estado, definindo o controle para escrever no banco de registradores dentro do registrador *rd*, cujos dados vêm do registrador da saída da ULA. Os outros controles são semelhantes às instruções TIPO-R.

c) Formas de onda jal:



Assim como no vídeo, a forma de onda da imagem acima descrevem a execução do programa teste, definindo a ordem de execução que varia de acordo com as novas instruções implementadas com sucesso.

Forma de onda da instrução mul $t1,$t2,$t3:



A forma de onda acima demonstra o valor registrado em *rd*, que foi passado para o registrador *PC* pela instrução *jr*. A multiplicação feita foi 4x4=16, que em hexadecimal é 0x00000010.