Universidade de Brasília

DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO - CIC

GRUPO 5

LABORATÓRIO 5 DE OAC

CPU MIPS Pipeline

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 : 27/11/2017

1 Processador Pipeline

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;
- Estudar e implementar uma CPU MIPS com ISA mínima e Pipeline de 5 estágios;

1) (0.0) Abra e compile o projeto do processador MIPS PUM v.6.0 com o Processador Pipeline:

a. Carregue o programa testePIPE.s ;

b. Simule no Mars e por forma de onda comparando os resultados;

c. Identifique e corrija o que está causando o resultado errôneo.

2) (1.0) Analise o processador MIPS Pipeline fornecido. Desenhe o Diagrama de Blocos do Caminho de Dados completo incluindo os registradores de pipeline e especifique a tabela verdade dos sinais de controle por estágio do pipeline;

3) (1.0) Analise as unidades de Hazard e Forward e com base na ISA especifique, através de exemplos, quais riscos de dados e de controle são detectados e tratados.

4) (1.0) Use o seu programa teste.s e verifique o correto funcionamento de TODAS as instruções da ISA implementada, teste usando simulação por forma de onda e pela implementação na DE2.

5) (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 usando o syscall 6). Grave vídeos demonstrativos e disponibilize no YouTube com links no relatório.

6) (2.0) 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.

7) (4.0) 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

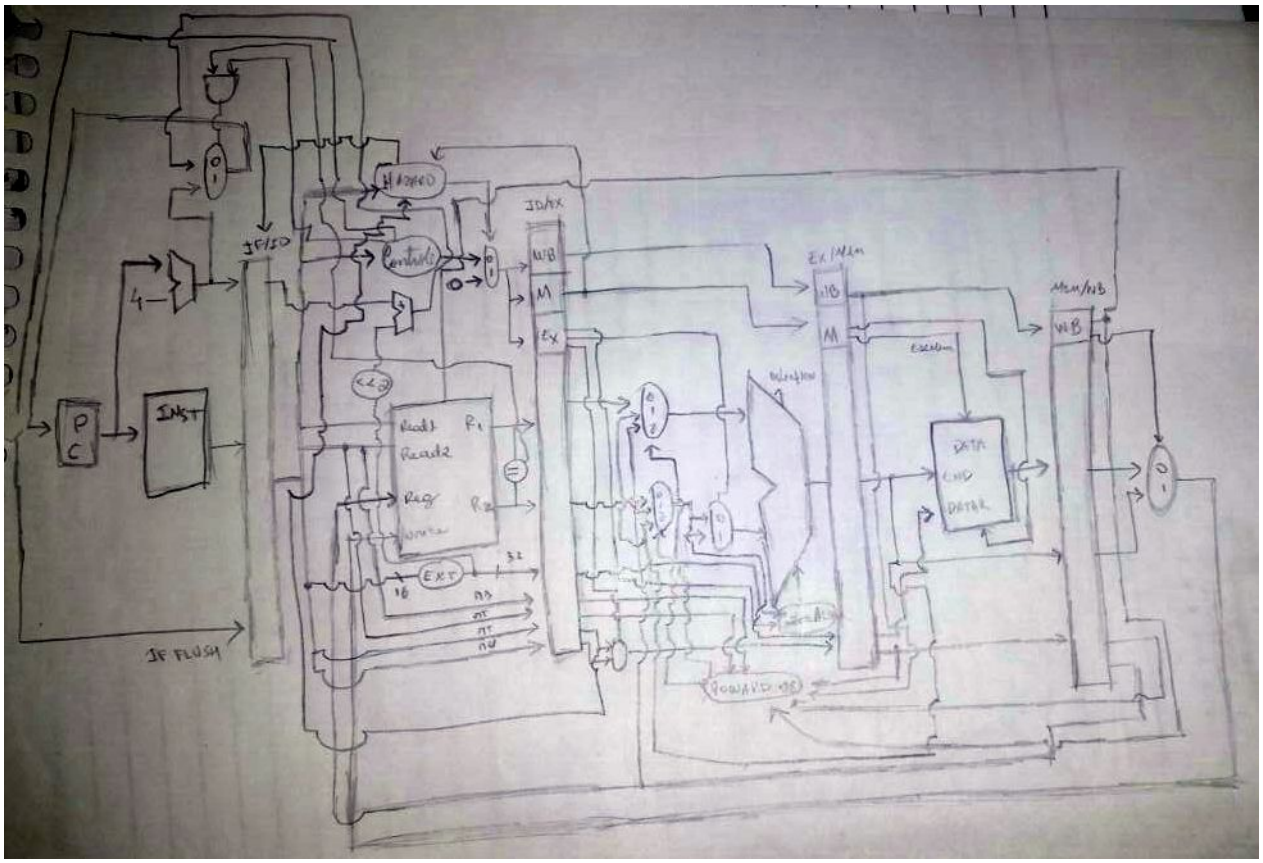
b. (1.0) Indique as modificações necessárias no bloco de controle

c. (1.0) Indique as modificações necessárias nas unidades de detecção e tratamento de Hazards

d. (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) Caminho de dados:



O caminho de dados representa o projeto de maneira simplória, dada a complexidade existente. Como a unidade de ponto flutuante não está devidamente implementada no *PIPELINE*, a mesma não está incluída na descrição em desenho da imagem.

Para a tabela verdade, acrescentamos um sinal de controle para a instrução *JALR* no estágio *IF/ID*.

3) Com base na implementação do projeto, a detecção de *hazards* implementada resolve os problemas por meio de *fowarding*, verificando se o registrador de escrita de uma instruçao é igual aos registradores de leitura *rs* e *rt*, juntamente com o sinal de *EscreveReg* ativado em seu respectivo estágio. Caso aconteça, a unidade indica o sinal para entrada da ULA como sendo o fio de *fowarding*, para resolver o *hazard*. Como por exemplo:

add $t0, $t1,$t2

sub $t3, $t0, $t1

Nas instruções acima, um *hazard* será detectado da seguinte forma:

"Se (((RegEscrita == *rsMEM*) *OU* (RegEscrita == *rtMEM*)) *E* EscreveRegMEM) *OU* (((RegEscrita == *rsWB*) *OU* (RegEscrita == *rtWB*)) *E* EscreveRegWB)".

A verificação para (RegEscrita == *rsMEM*) e *EscreveRegMEM* indicará que aconteceu um hazard. Portanto, o resultado da ULA dentro registrador *EX/MEM* será passado para a entrada da ULA para que a instrução *sub* possa ser executada normalmente.

Para hazard de *Load*, a verificação é feita anterioremente para evitar problemas de leitura e escrita. Exemplo:

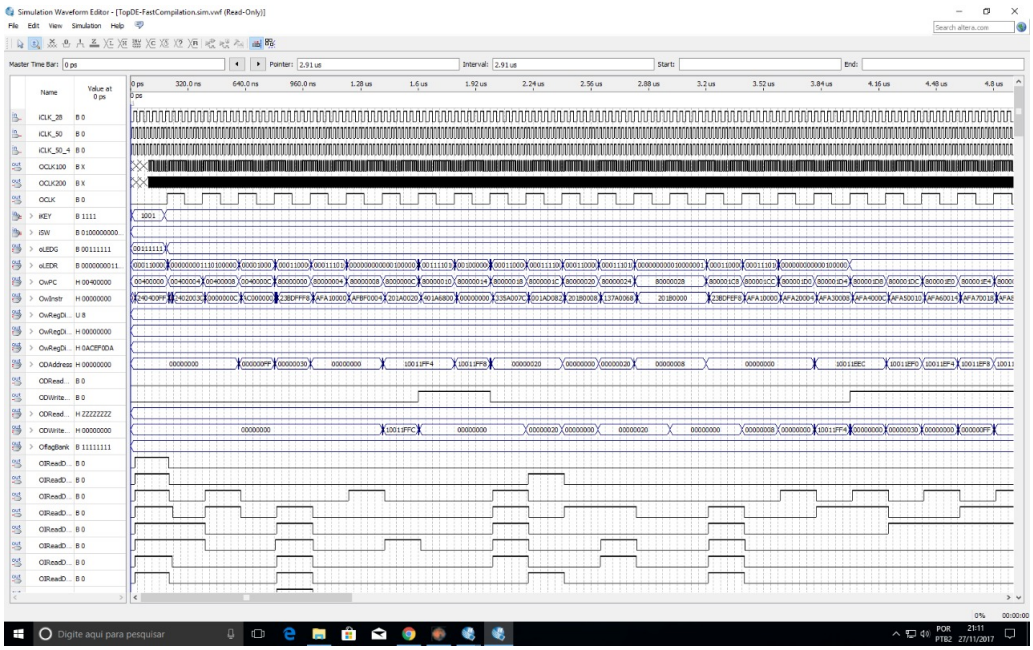
lw $t0, 4($t1)
sub $t3, $t0, $t1

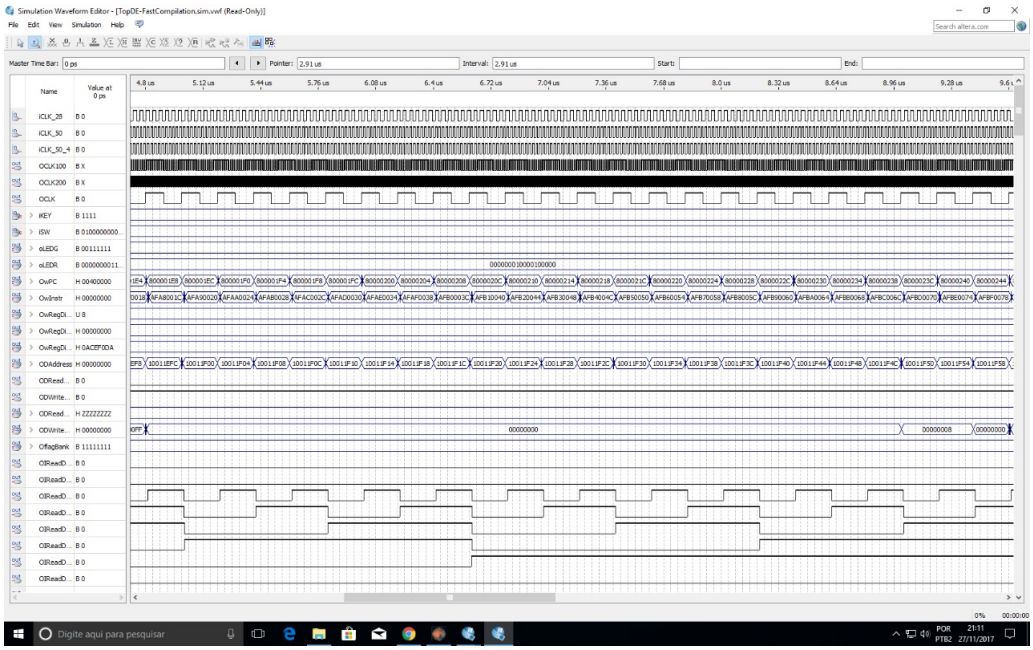
A detecção do *Hazard* será feita verificando o registrador *rt* de escrita da instrução *load*, com os registradores *rs* e *rt* da instrução *sub*. Além destes sinais, é feita a verificação do sinal de leitura da memória de dados para a instrução *load*. Neste caso, a detecção obrigará o surgimento de uma bolha, fazendo com que a instrução *sub* tenha que esperar mais um ciclo para que finalmente possa ser executada usando *fowarding* mais à frente.

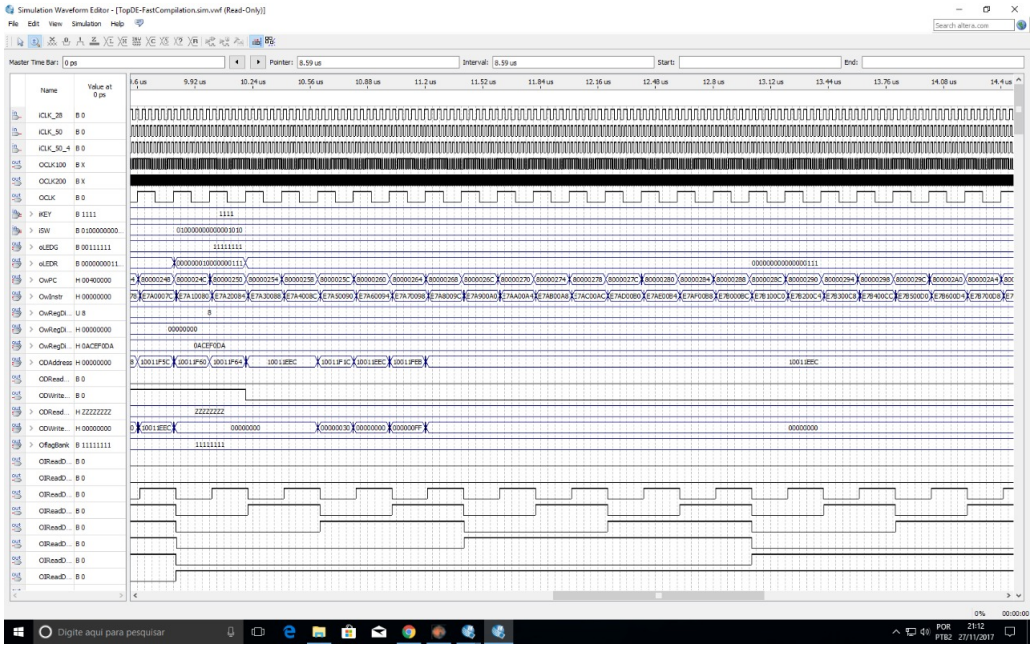
Para instruções do tipo *beq* e que fazem *jump*, a instrução indicada por *PC* sofre um *flush* após avaliação do segundo estágio, caso necessário.

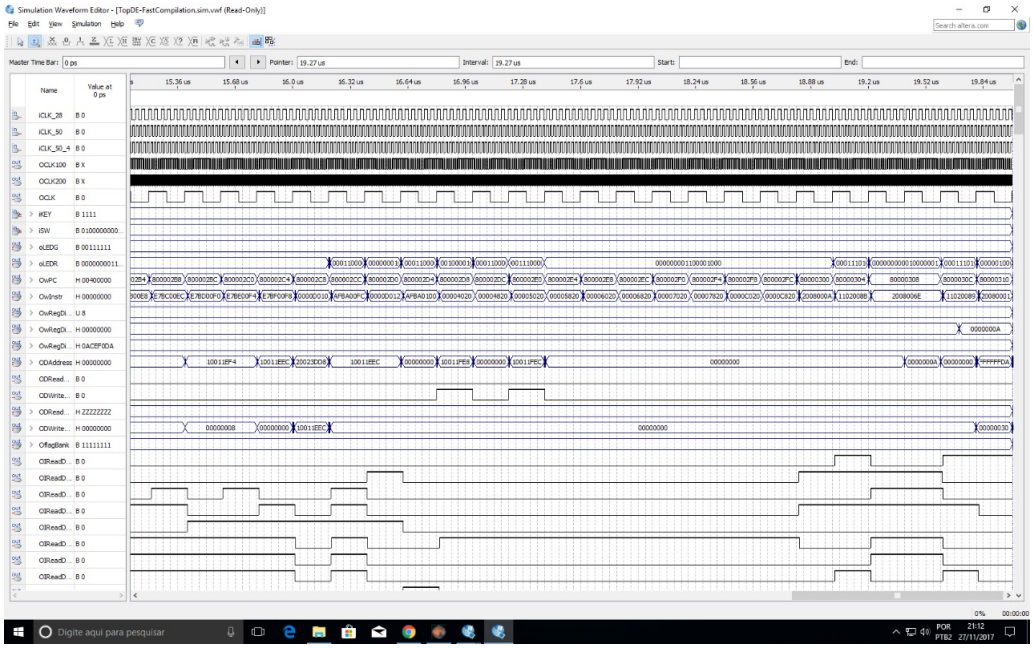
4) Vídeo: https://youtu.be/mb7nSMGHjpQ

Segue as formas de onda do código teste:




Simulation Waveform Editor - [TopDE-FastCompilation.sim.vwf (Read-Only)]
File Edit View Simulation Help
Master Time Bar: 0 ps
Pointer: 2.91 us
Interval: 2.91 us
Start:
End:
Name
Value at 0 ps
iCLK_28
iCLK_50
iCLK_50_4
OCLK100
OCLK200
OCLK
iKEY
iSW
oLEDG
oLEDR
OwPC
OwInstr
OwRegDi...
OwRegDi...
CDAddress
CDRead...
CDWrite...
CDRead...
CDWrite...
OflagBank
OIReadD...
Digite aqui para pesquisar
POR
PTB2
21:11
27/11/2017


Simulation Waveform Editor - [TopDE-FastCompilation.sim.vwf (Read-Only)]
File Edit View Simulation Help
Master Time Bar: 0 ps
Pointer: 6.59 us
Interval: 6.59 us
Start:
End:
Name
Value at 0 ps
iCLK_28
iCLK_50
iCLK_50_4
OCLK100
OCLK200
OCLK
iKEY
iSW
oLEDG
oLEDR
OwPC
OwInstr
OwRegDi...
OwRegDi...
CDAddress
CDRead...
CDWrite...
CDRead...
CDWrite...
OflagBank
OIReadD...
Digite aqui para pesquisar
POR
PTB2
21:12
27/11/2017



5) Vídeo: https://youtu.be/k9OVXqy7qjM

6)

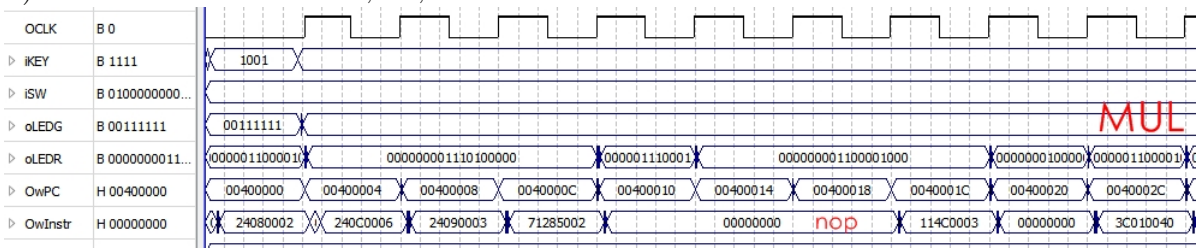
7) Os testes foram realizados com o arquivo "Teste_6_LAB5.asm", compactado junto à este relatório.
O link do vídeo que comprova seu funcionamento: https://youtu.be/i_IQeC43eoc

a) Assim como no laboratório 4, 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, foi necessário usar os dados armazenados em *WB* pela saída da ULA para armazenar no banco de registradores em *rd*, assim como definido pela nova instrução *mul*. A escrita em {*Hi,Lo*} continuou acontecendo normalmente, assim como na instrução *mult*.

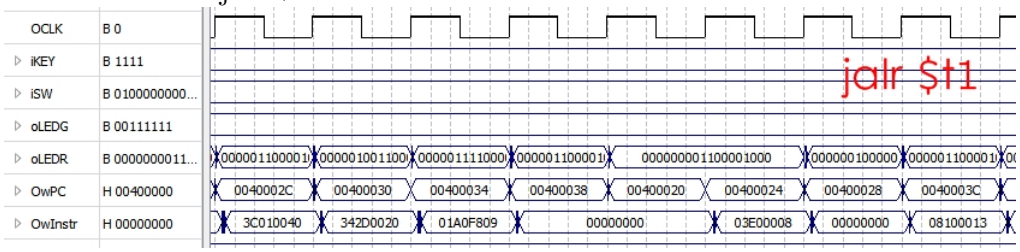
b) Para o bloco de controle, assim como no laboratório 4, foi necessário habilitar a escrita no banco de registradores em *WB*, para o registrador *rd*, que seleciona se deverá escrever *PC+4* em $ra ou no registrador definido pela instrução *jalr* $t1, $t2. Portanto, o controle *RegDest* ativa a escrita 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 definido o controle para escrever no banco de registradores dentro do registrador *rd*, cujos dados vêm do registrador *WB*, do fio da saída da ULA. Os outros controles são semelhantes às instruções TIPO-R.

c) Assim como na instrução *JR*, implementou-se o tratamento de hazard para verificar se a instrução é *JALR*, devendo assim não executar a instrução de *PC+4*, e ir para o endereço ao qual a instrução indica. O tratamento para *mul* continuou o mesmo para instruções que são do tipo R.

d) Forma de onda mul $t1,$t2,$t3:



Forma de onda jalr $t1:



Forma de onda jalr $t1,$t2: