Relatório
Laboratório 5: Mips Pipeline

Grupo 3
Gabriel Alves Castro - 17/0033813
Henrique Mendes de Freitas Mariano - 17/0012280
Luthiery Costa Cavalcante - 17/0040631
Matheus Breder Branquinho Nogueira - 17/0018997

Universidade de Brasília - Dep. Ciência da Computação
Disciplina CIC 116394 - Organização e Arquitetura de Computadores - Turma A

Introdução

O projeto desse labortório é apresentar e analisar o processador MIPS Pipeline testando um simples código em assembly MIPS e a partir desse código analisar os resultados e comparar o desempenho com os processadores uniciclo e multiciclo ,e analisar também se, testando esse código, é possível indicar possíveis hazards no processador MIPS Pipeline.

Parte A: Apresentação do processador MIPS Pipeline

O vídeo da execução do arquivo testeSIMPLESPIPE.s no MARS encontra-se neste link.

1. a) Uniciclo

O uniciclo age de acordo com o esperado da execução do arquivo testeSIMPLESPIPE.s no MARS, a execução do uniciclo na DE1-SoC pode ser obtida através deste link. O waveform do uniciclo está de acordo com a execução no MARS. O tempo de execução do uniciclo é de um pouco mais de 1.0 μs. E o divisor de clock máximo para a execução no uniciclo é de 3, ou seja, a frequência máxima de execução é de aproximadamente 16,67 MHz, o vídeo da execução pode ser encontrado nesse link.

Waveform Uniciclo

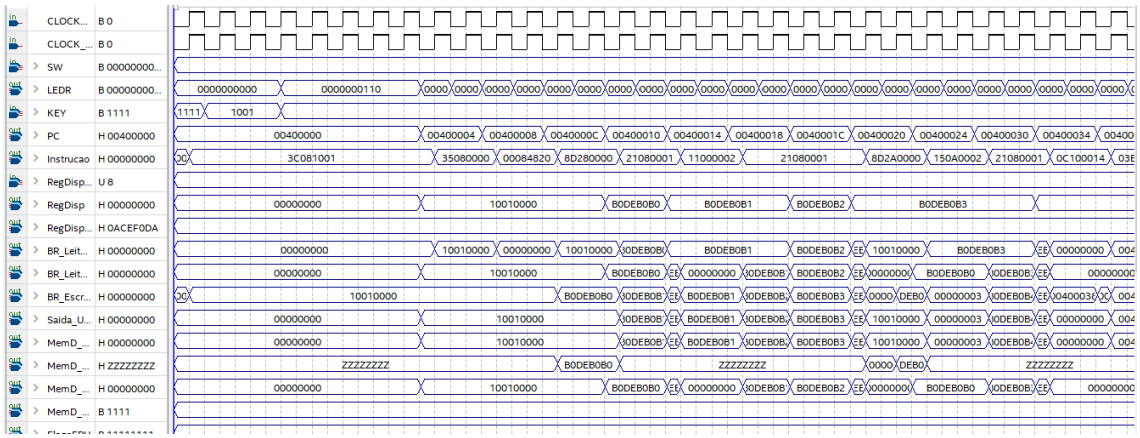


Figura 1: Waveform Uniciclo Parte 1.

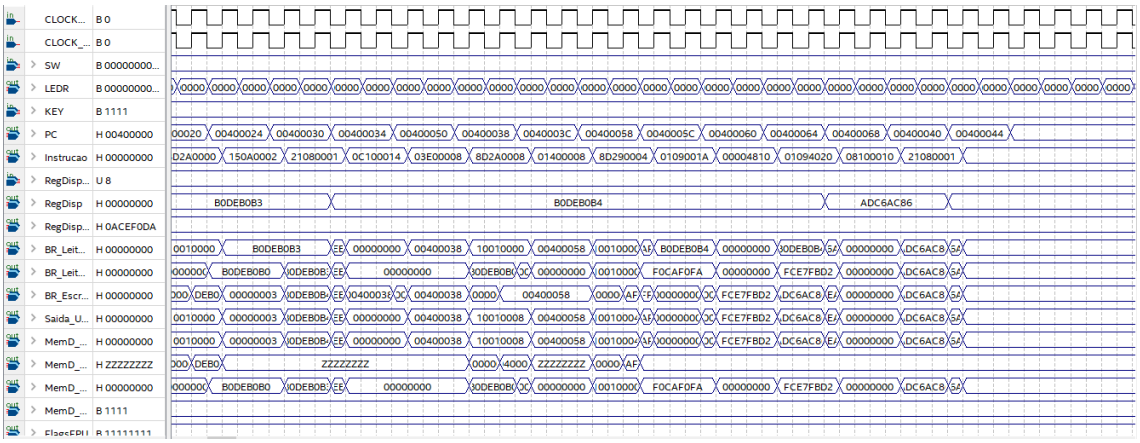


Figura 2: Waveform Uniciclo Parte 2.

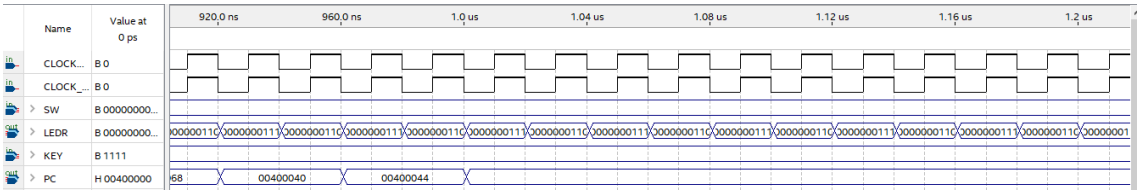


Figura 3: Waveform Uniciclo Tempo de execução.

1. b) Multiciclo

O processador MIPS multiciclo obtém o resultado esperado do arquivo de teste testeSIMPLESPIPE.s, porém ao analisar o tempo de execução do programa, ele demonstra ser menos eficiente que o processador MIPS uniciclo e o pipeline, já que demora mais para finalizar a execução do programa, sendo três vezes mais lento de acordo com os resultados do waveform, terminando a execução em um pouco mais de 3.5μs. A execução do programa encontra - se nesse link.

O menor divisor de clock utilizado na De1-SoC para que o programa fosse executado corretamente foi de 2, ou seja, a frequência máxima que podemos utilizar na placa é 25MHz. O vídeo da execução da frequência máxima pode ser encontrado neste link.

Waveform Multiciclo

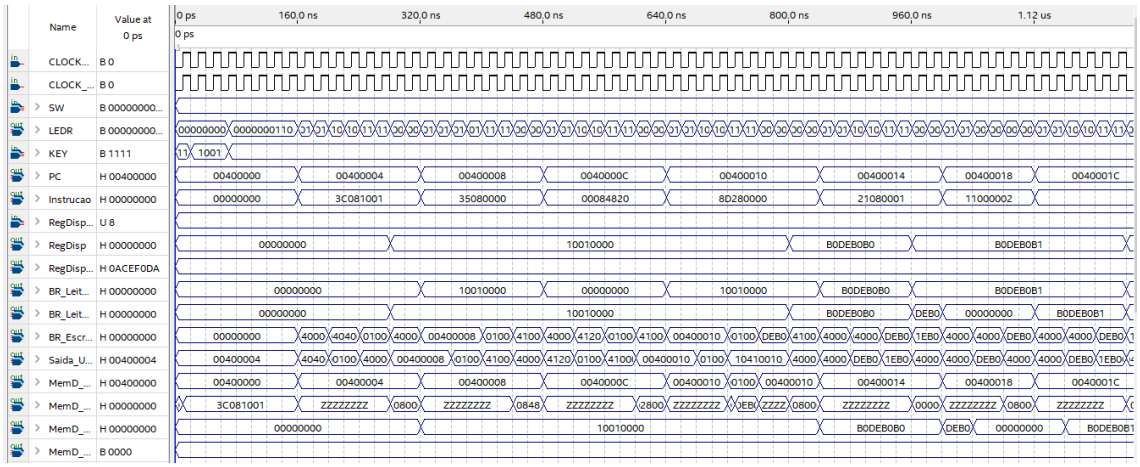


Figura 4: Waveform Multiciclo Parte 1.

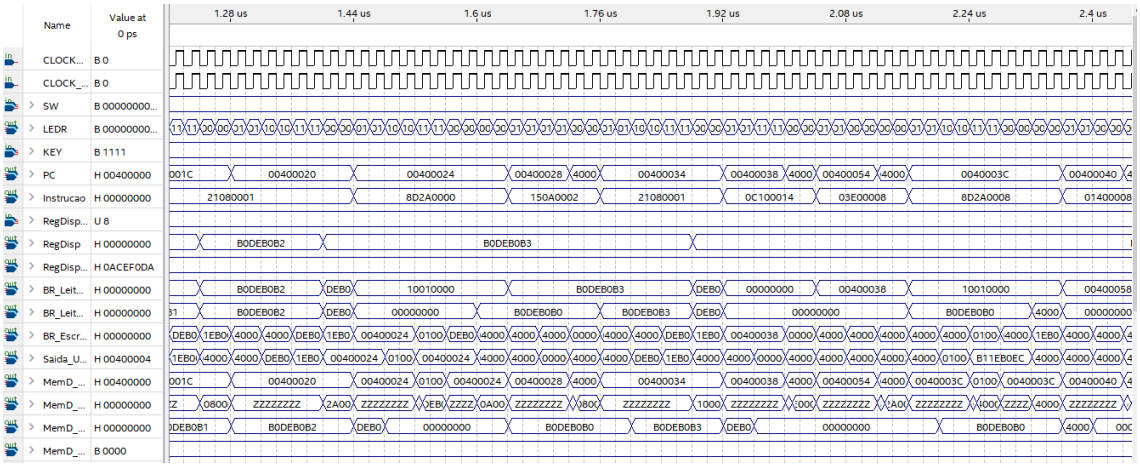


Figura 5: Waveform Multiciclo Parte 2.

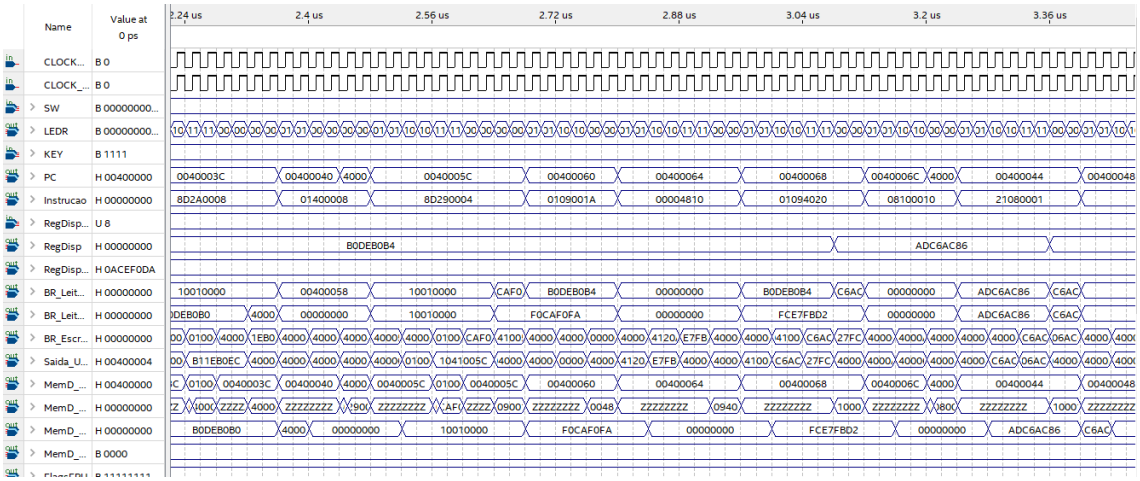


Figura 6: Waveform Multiciclo Parte 3.

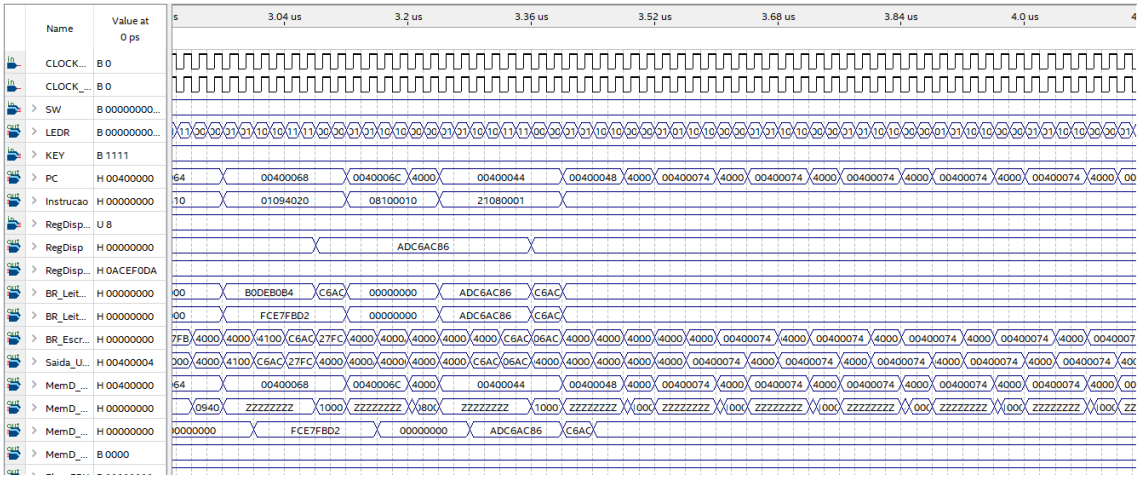


Figura 7: Waveform Multiciclo Parte 4 com tempo de execução.

1. c) Pipeline

Inicialmente o arquivo testeSIMPLESPIPE.s apresentava Hazards quando executado pelo processador MIPS pipeline de tal modo que o resultado final era diferente da simulação no Mars. Então após corrigir as instruções com bolhas como requisitado no laborátorio, o programa foi executado corretamente e apresentou o melhor desempenho no tempo de execução comparado com o uniciclo e o multiciclo como apresentado no waveform. O tempo de execução foi de 960 ns.

O menor divisor de clock que pode ser colocado na De1-SoC durante a execução do programa teste foi de 2, de tal modo que a máxima frequência possível sem que o programa desse erro foi de 25MHz. O vídeo da execução do programa no processador pipeline pode ser encontrado neste link.

Waveform Pipeline

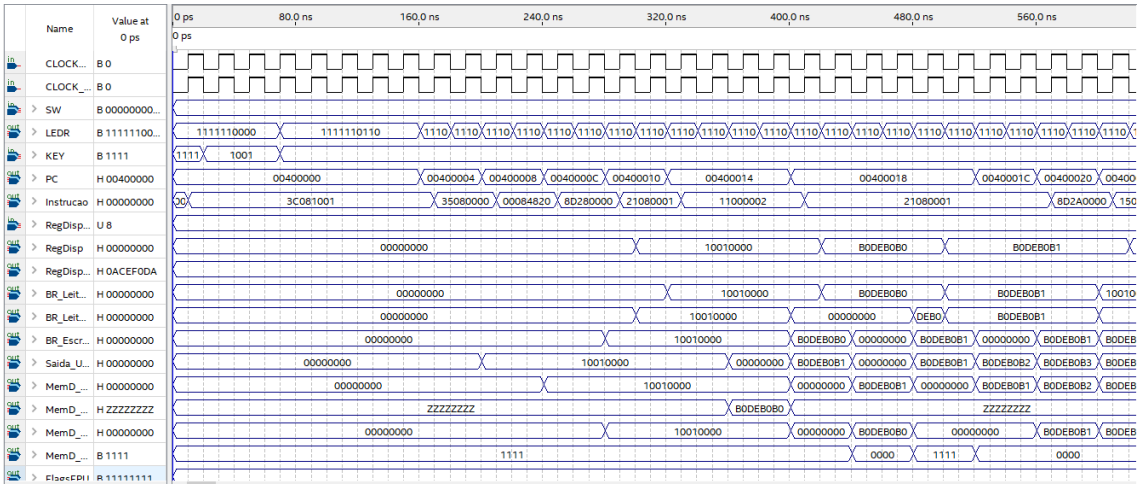


Figura 8: Waveform Pipeline Parte 1.

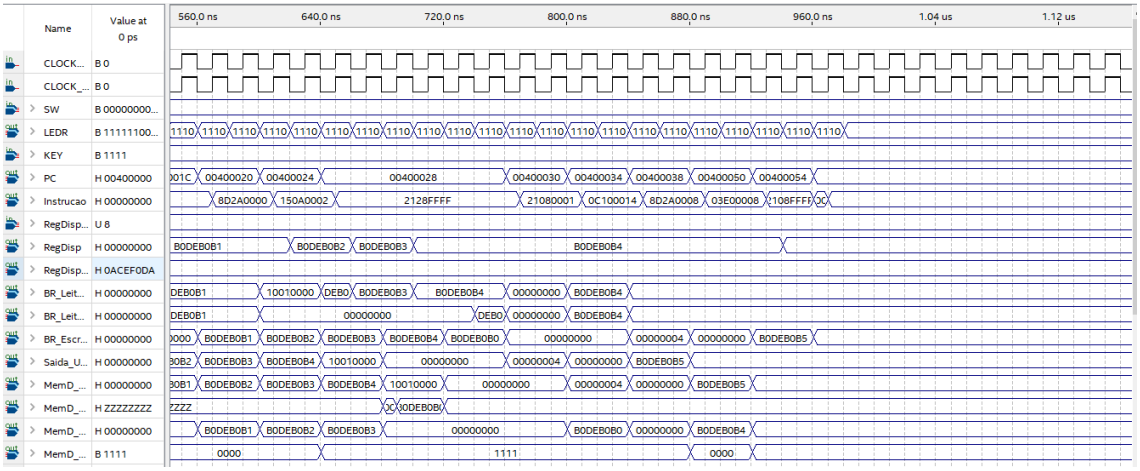


Figura 9: Waveform Pipeline Parte 2 com tempo de execução.

Requerimentos Físicos e Temporais

Uniciclo

Físicos

Logic utilization (in ALMs)	7,088 / 32,070 (22 %)
Total registers	4588
Total pins	241 / 457 (53 %)
Total virtual pins	0
Total block memory bits	934,912 / 4,065,280 (23 %)
Total DSP Blocks	18 / 87 (21 %)

Figura 10: ALMs, Registradores, Memória e número de blocos DSP.

Temporais

```
Worst-case setup slack is 12.723
Worst-case hold slack is 0.146
Worst-case recovery slack is 30.895
Worst-case removal slack is 0.316
Worst-case minimum pulse width slack is 15.029
```

Figura 11: Slacks.

Clock to Output Times

	Data Port	Clock Port	Rise	Fall	Clock Edge	Clock Reference
1	altera_reserved_tdo	altera...ed_tck	5.808	5.792	Fall	altera_reserved_tck

Figura 12: TCO.

Hold Times

	Data Port	Clock Port	Rise	Fall	Clock Edge	Clock Reference
1	altera_r...rved_tdi	altera...ed_tck	0.791	0.668	Rise	altera_reserved_tck
2	altera_...ved_tms	altera...ed_tck	0.662	0.450	Rise	altera_reserved_tck

Figura 13: TH.

Propagation Delay

	Input Port	Output Port	RR	RF	FR	FF
1	SW[7]	VGA_G[1]	21.392	23.566	21.747	23.919

Figura 14: TPD.

Setup Times

	Data Port	Clock Port	Rise	Fall	Clock Edge	Clock Reference
1	altera_r...rved_tdi	altera...ed_tck	2.657	3.846	Rise	altera_reserved_tck
2	altera_...ved_tms	altera...ed_tck	3.018	3.745	Rise	altera_reserved_tck

Figura 15: TSU.

Multiciclo

Físicos

Logic utilization (in ALMs)	6,841 / 32,070 (21 %)
Total registers	4801
Total pins	241 / 457 (53 %)
Total virtual pins	0
Total block memory bits	934,912 / 4,065,280 (23 %)
Total DSP Blocks	18 / 87 (21 %)

Figura 16: ALMs, Registradores, Memória e número de blocos DSP.

Temporais

```
Worst-case setup slack is 13.280
Worst-case hold slack is 0.135
Worst-case recovery slack is 30.703
Worst-case removal slack is 0.290
Worst-case minimum pulse width slack is 15.057
```

Figura 17: Slacks.

Clock to Output Times

	Data Port	Clock Port	Rise	Fall	Clock Edge	Clock Reference
	HEX0[*]	clk	25.516	25.590	Rise	clk

Figura 18: TCO.

Hold Times

	Data Port	Clock Port	Rise	Fall	Clock Edge	Clock Reference
1	SW[*]	clk	3.862	3.707	Rise	clk

Figura 19: TH.

Propagation Delay

Input Port	Output Port	RR	RF	FR	FF
SW[7]	VGA_G[2]	19.383	21.375	19.656	21.648

Figura 20: TPD.

Setup Times

	Data Port	Clock Port	Rise	Fall	Clock Edge	Clock Reference
1	KEY[*]	clk	12.620	11.766	Rise	clk

Figura 21: TSU.

Pipeline

Físicos

Logic utilization (in ALMs)	6,836 / 32,070 (21 %)
Total registers	4892
Total pins	241 / 457 (53 %)
Total virtual pins	0
Total block memory bits	934,912 / 4,065,280 (23 %)
Total DSP Blocks	18 / 87 (21 %)

Figura 22: ALMs, Registradores, Memória e número de blocos DSP.

Temporais

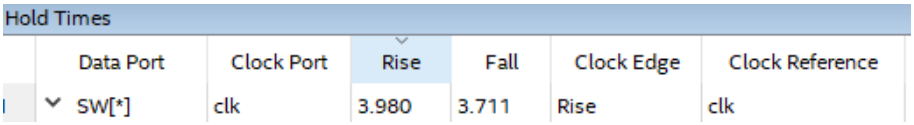
```
Worst-case setup slack is 13.477
Worst-case hold slack is 0.161
Worst-case recovery slack is 31.049
Worst-case removal slack is 0.317
Worst-case minimum pulse width slack is 15.058
```

Figura 23: Slacks.

Clock to Output Times

Data Port	Clock Port	Rise	Fall	Clock Edge	Clock Reference
HEX0[*]	clk	22.094	23.347	Rise	clk

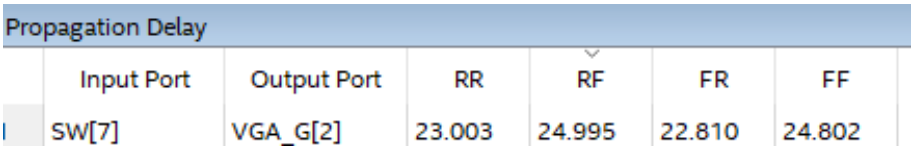
Figura 24: TCO.



Hold Times

Data Port	Clock Port	Rise	Fall	Clock Edge	Clock Reference
SW[*]	clk	3.980	3.711	Rise	clk

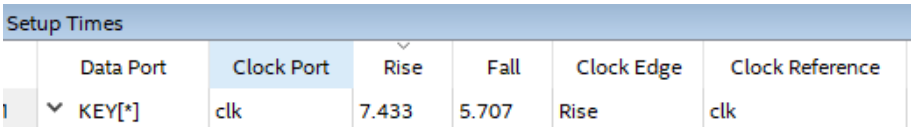
Figura 25: TH.



Propagation Delay

Input Port	Output Port	RR	RF	FR	FF
SW[7]	VGA_G[2]	23.003	24.995	22.810	24.802

Figura 26: TPD.



Setup Times

Data Port	Clock Port	Rise	Fall	Clock Edge	Clock Reference
KEY[*]	clk	7.433	5.707	Rise	clk

Figura 27: TSU.

Parte B: Análise do processador MIPS Pipeline

testeSIMPLESPIPE.s

O arquivo de teste testeSIMPLESPIPE.s fornecido para o teste do processador MIPS pipeline estava com muitos Hazards de dados, ou seja, o processador, ao realizar as instruções do programa, estava utilizando dados errados ou ainda não atualizados que faziam o programa não ser executado da maneira correta. Portanto, o item d) da questão 2) da parte B do laboratório pedia para que corrigíssimos o código utilizando bolhas para as instruções não causarem mais os resultados errôneos.

A partir disso, analisamos que as instruções do tipo R e tipo I não necssitavam de bolhas pois o próprio processador pipeline é capaz de buscar o dado antes de estar disponível na memória. As instruções do tipo Branch sempre são necessárias duas bolhas após ela, já que a comparação é feita na terceira etapa do datapath, então antes de começar a executar a próxima instrução, é necessário duas bolhas para que as instruções seguintes ao branch não sejam executadas caso ocorra o desvio. As instruções do tipo Jump, precisavam de uma bolha após ela, já que na linguagem MIPS as instruções realizam o desvio após a segunda etapa do datapath, que é quando o imediato é calculado e o banco de registradores verificado, então antes de começar a próxima instrução ,o PC precisa ser atualizado, e isso ocorre no final da segunda etapa. Nas instruções do tipo Acesso à Memória, mais especificamente do tipo Load, os dados ficam prontos na quarta etapa do datapath,

em que depois de calculado o endereço, o processador acessa a memória e já tem acesso ao dado, então é necessária uma bolha para que o precessador MIPS pipeline seja capaz de acessar ao dado necessitado e utilize corretamente nas próximas instruções que utilizam o mesmo dado.

Depois de ter realizado as mudanças necessárias no código para corrigir os Hazards existentes, o processador MIPS pipeline foi capaz de executar da maneira esperada com os resultados esperados, que são os da simulação no Mars, em que não são analizados os hazards.

testeHAZARDPipe.s

Um hazard que foi descoberto durante a inserção de bolhas no código de teste testeSIMPLESPIPE.s foi na instrução jr(Jump Register), pois teoricamente, se o registrador utilizado pela instrução jr for utilizado logo anteriomente, o processador deveria ser capaz de analisar a sua mudança e utilizar o dado correto logo após a terceira etapa do datapath, em que a ULA é utilizada, e não apenas no banco de registradores, em que o dado ficaria pronto para ser utilizado apenas após a quinta etapa do datapath, em que se escreve no banco de registradores.

Desse modo, corrigir o Hazard da intrução jr(Jump Register) tornaria o pipeline ainda mais eficiente, de modo que a instrução poderia utilizar um dado ainda não escrito na memória e não precisaria esperar mais dois ciclos para poder utilizar o dado corretamente. O vídeo da execução do arquivo encontra-se neste link.