Curso de Engenharia de Computação

05434P – LABORATÓRIO DE ARQUITETURA DE COMPUTADORES

Projeto 3 - CPU Pipeline
Profº Dr. Ricardo Pannain

Equipe:

Henrique Sartori Siqueira	19240472
Jemis Dievas José Manhiça	19076272

1 - Descrição do projeto com a topologia da CPU

Este projeto foi realizado no sistema operacional Windows 10, com os softwares Quartus Prime Lite Edition 18.1 (testes e execução do código), Visio (desenho do datapath e diagrama de estados), Visual Studio Code e Github (edição e armazenamento do código), e Google Drive (confecção do relatório, tabela de estados e armazenamento dos demais arquivos).

Para este presente projeto foram registrados 64 endereços com 4 bits por endereço na memória, sendo que PC é atualizado em 4 posições, havendo 16 instruções executáveis com 16 bits cada uma. Com o datapath definido a seguir:

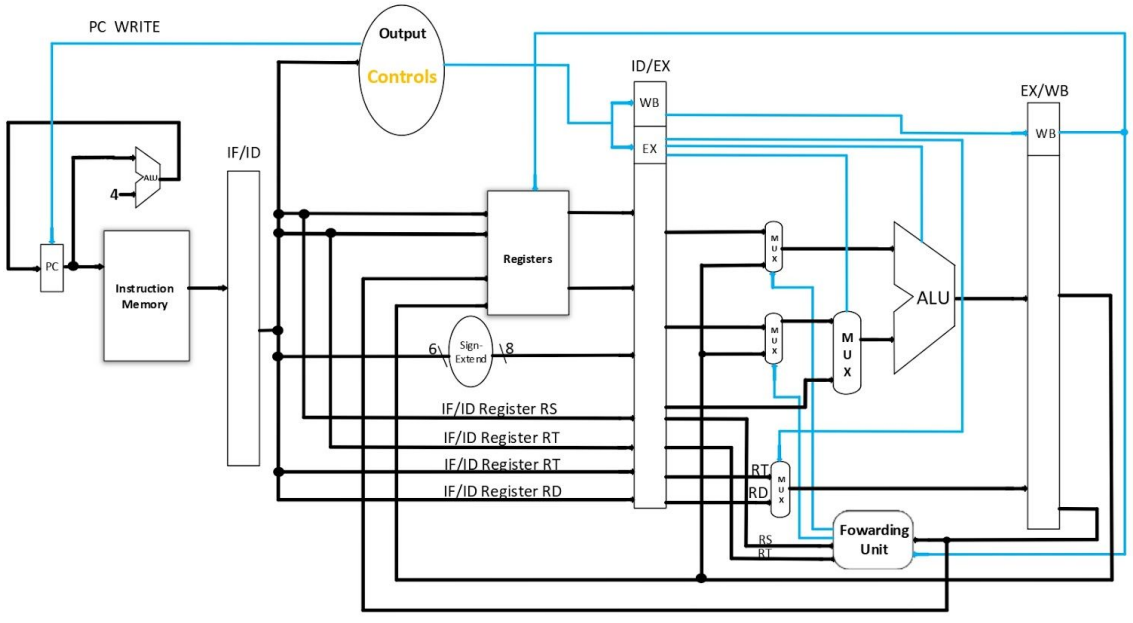

Imagem 1. Datapath.

2 - Especificação

2.1 - Registradores

Neste datapath, o banco de registradores possui oito registradores de 8 bits cada, com os respectivos endereços representados a seguir:

Registrador	Endereço
$0	000
$1	001
$2	010
$3	011
$4	100
$5	101
$6	110
$7	111

Todos os registradores, com exceção do registrador $0 que tem o valor constante de 0, podem ter seus valores modificados.

2.2 - Formato das instruções

As instruções possuem 16 bits, sendo divididas em dois formatos descritos a seguir:

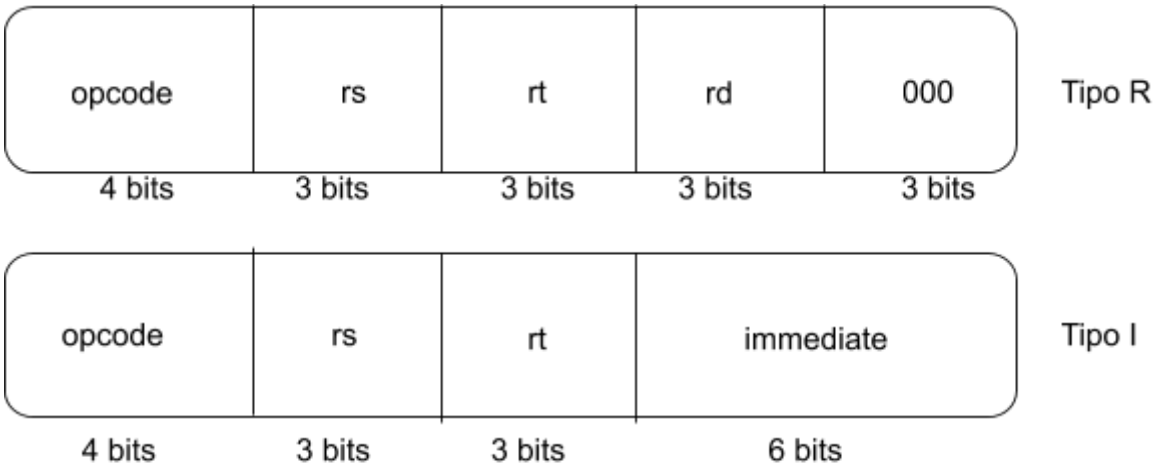

Imagem 2. Formato das instruções.

Opcode	Instrução	Significado	Descrição
0000	MOV Ri, Rj	Ri ← Rj	Move
0001	MOV Ri, Imediato	Ri ← Imediato	Move Imediato
0010	ADD Ri, Rj, Rk	Ri ← Rj + Rk	Adição
0011	ADDI Ri, Rj, Imediato	Ri ← Rj + Imediato	Adição Imediata
0100	SUB Ri, Rj, Rk	Ri ← Rj - Rk	Subtração
0101	SUBI Ri, Rj, Imediato	Ri ← Rj - Imediato	Subtração Imediata
0110	AND Ri, Rj, Rk	Ri ← Rj & Rk	And
0111	ANDI Ri, Rj, Imediato	Ri ← Rj & Imediato	And Imediato
1000	OR Ri, Rj, Rk	Ri ← Rj | Rk	Or
1001	ORI Ri, Rj, Imediato	Ri ← Rj | Imediato	Or Imediato

Instruções pares são do tipo R e instruções ímpares são do tipo I.

2.3 - Unidade de Controle: diagrama, tabela de estados, sinais e seus significados

Os sinais de controle necessários para a execução completa de uma instrução são denotados a seguir:

Sinal	Efeito quando ‘0’	Efeito quando ‘1’
PcWrite	Mantém valor de PC	Atualiza o valor de PC
RegWrite	Não realiza escrita em um registrador	Realiza escrita em um registrador
ALUSrcB	Usa o valor (atualizado) de um registrador	Usa o valor do imediato estendido
RegDst	Salva o resultado em RT (tipo I)	Salva o resultado em RD (tipo R)
FRWA	Utiliza o valor do	Utiliza o valor do dado do

	registrador presente no banco de registradores	registrador atualizado na ULA (adiantamento)
FRWB	Utiliza o valor do registrador presente no banco de registradores	Utiliza o valor do dado do registrador atualizado na ULA (adiantamento)

Sinal	Efeito quando “00”	Efeito quando “01”	Efeito quando “10”	Efeito quando “11”
ALUOp	Realiza AND	Realiza OR	Realiza adição	Realiza subtração

Há uma duração de quatro ciclos para cada instrução presente, sendo eles denotados na tabela a seguir:

Ciclo	Ação
Fetch (IF)	Busca da instrução na memória e atualização de PC
Decode (ID)	Decodifica a instrução, gerando os sinais de controle e coletando os dados para operação
Execution (EX)	Realiza operação na ULA
Write-back (WB)	Escreve o resultado no registrador correspondente da chamada da instrução no banco de registradores

A máquina de estados, possui 8 estados, denotados por dois (tipo R e I) para cada instrução, sendo a operação de mov constituída por uma adição do valor (do registrador ou imediato) com zero. O diagrama de estados é denotado a seguir (tabela em anexo):

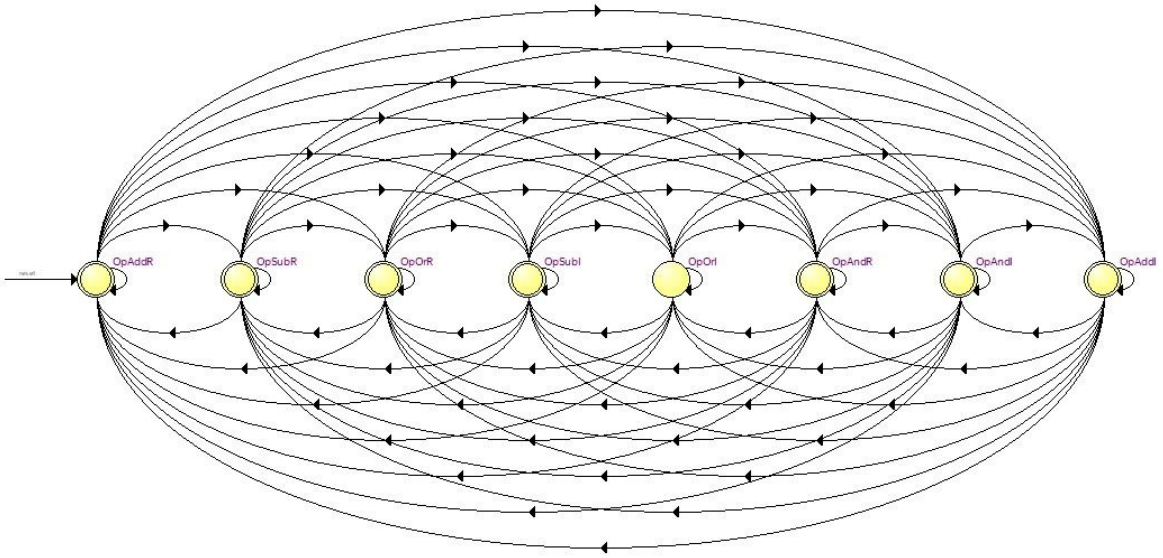

Imagem 3. Diagrama de estados.

3 - Resultados

3.1 - Descrição dos testes realizados

Testes realizados com o banco de registradores:

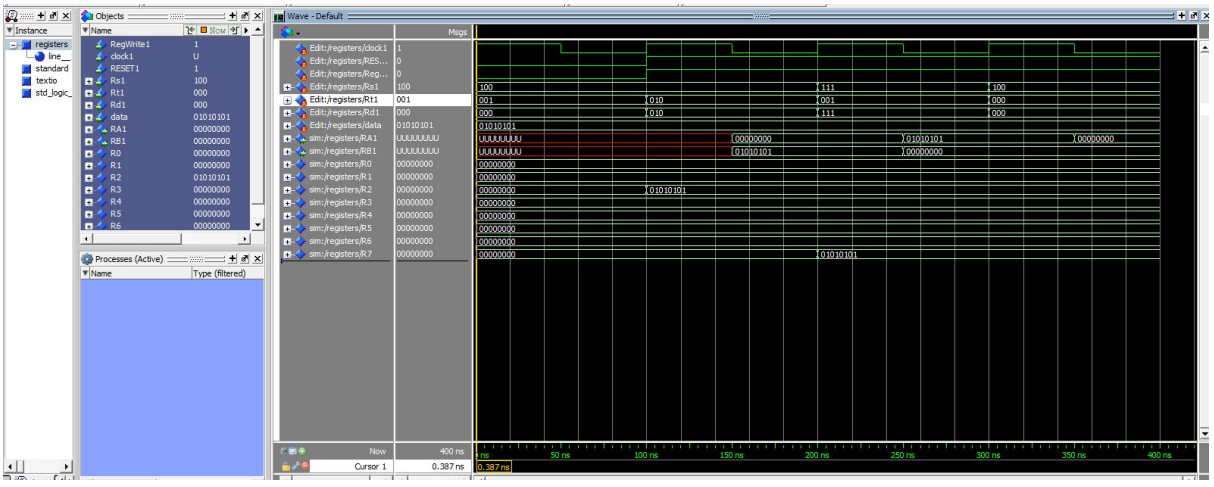

Imagem 4. Teste do banco de registradores.

Testes realizados com a unidade lógica e aritmética:

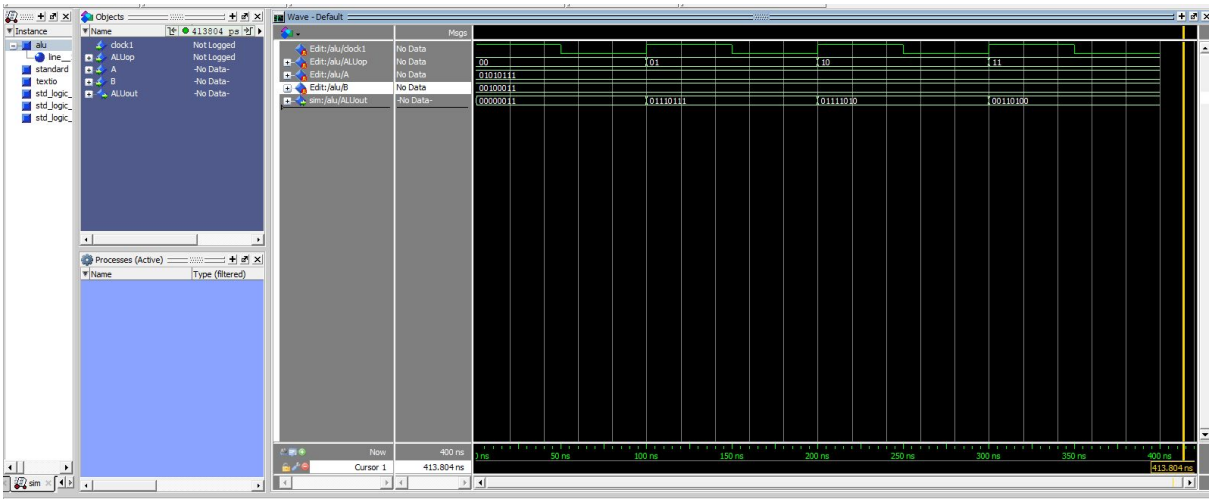

Imagem 5. Teste da ULA.

Testes realizados com a unidade lógica e aritmética de PC:

Imagem 6. Teste da ULA de PC.

Testes realizados com o controle (máquina de estados):

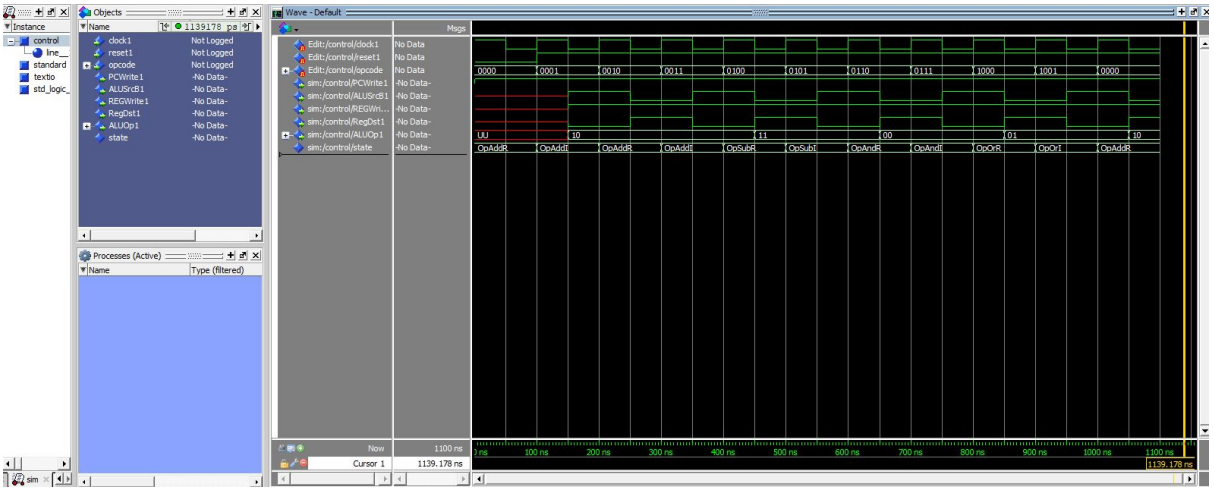

Imagem 7. Teste do controle de sinais.

Testes realizados com a unidade de adiantamento:

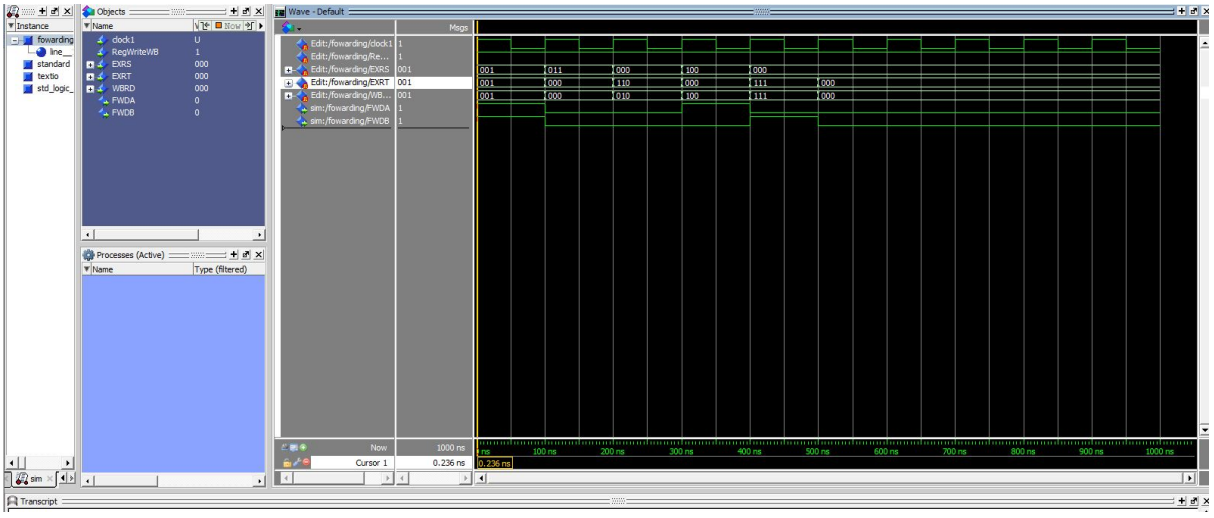

Imagem 8. Teste da unidade de adiantamento.

Testes realizados com a memória de instruções:

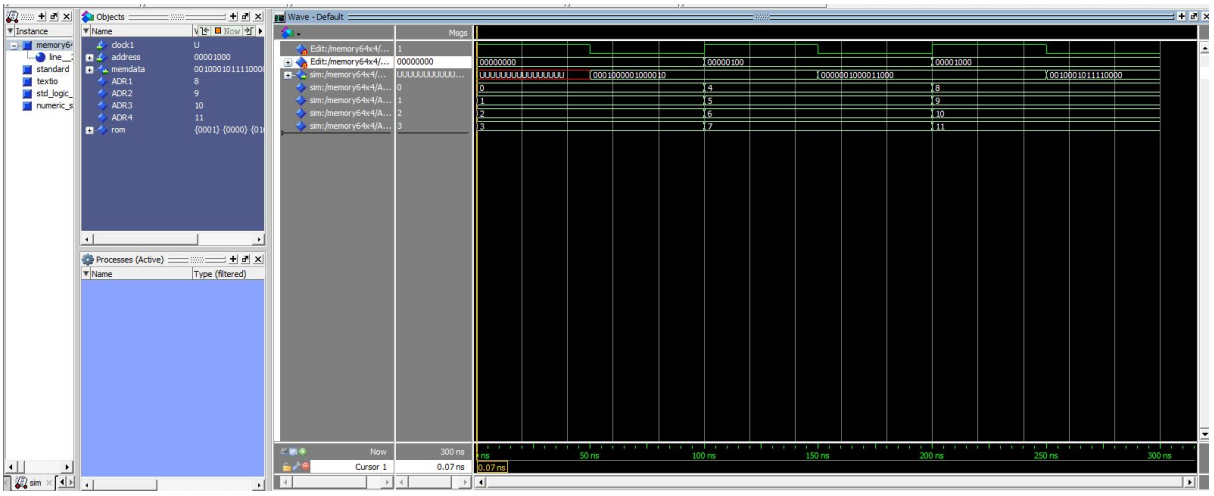

Imagem 9. Teste da memória (ROM) de instruções.

Testes realizados com o multiplexador de dados:

Imagem 10. Teste para o multiplexador de dados.

Testes realizados com o multiplexador de endereços:

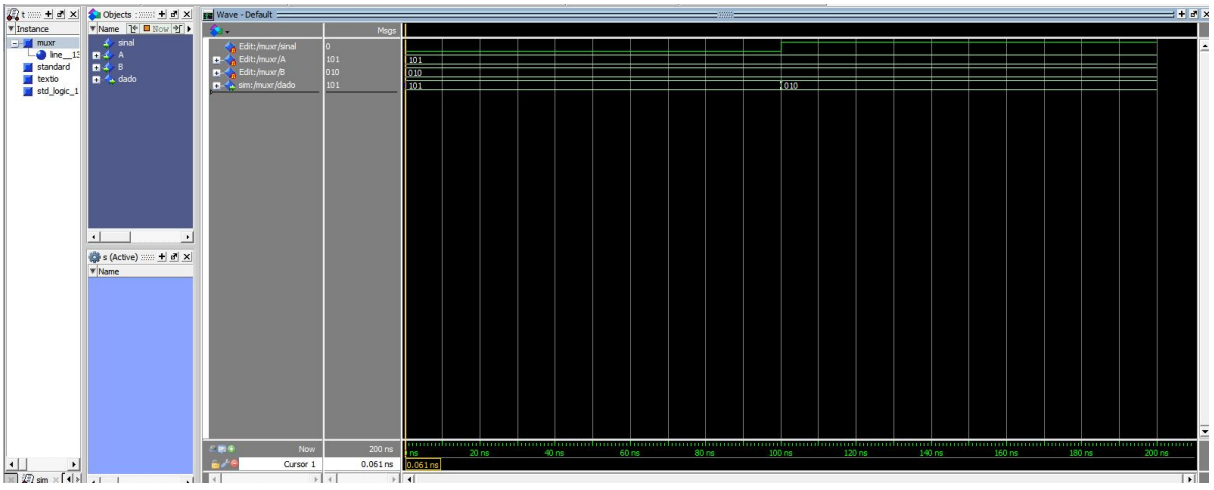

Imagem 11. Teste para o multiplexador de endereços.

Testes realizados com PC:

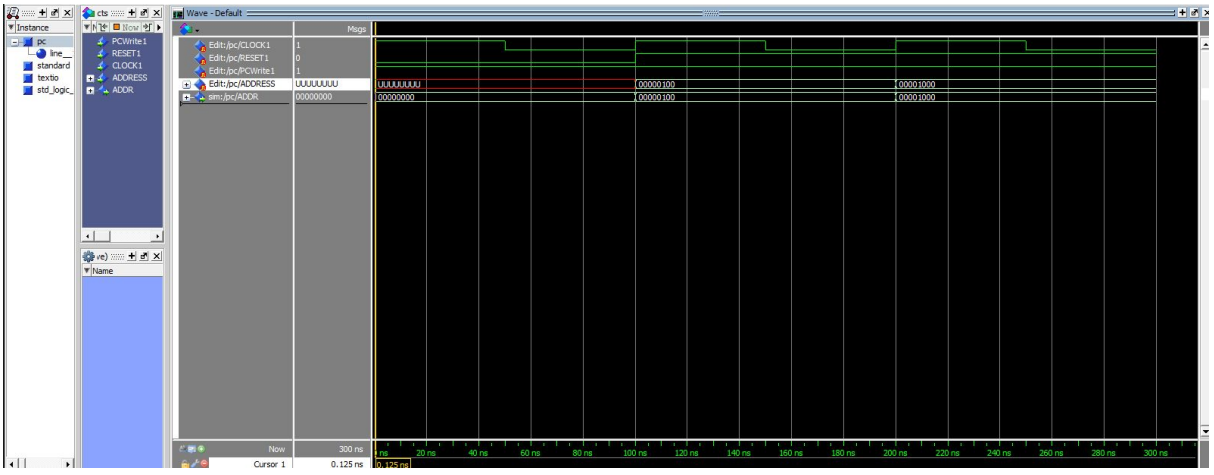

Imagem 12. Teste para o program counter (PC).

Testes realizados com o extensor de sinal:

Imagem 13. Teste com extensor de sinal.

Testes realizados com o pipe IFID:

Imagem 14. Teste com o pipe IFID.

Testes realizados com o pipe IDEX:

Imagem 15. Teste com o pipe IDEX.

Testes realizados com o pipe EXWB:

Imagem 16. Teste com o pipe EXWB.

3.2 - Resultados e discussões

Para a execução da CPU pipeline, foi criado um programa que realiza 16 instruções descritas a seguir:

```
MOV $1, 2          // 0001 0000 0100 0010
MOV $3, $1         // 0000 0010 0001 1000
ADD $6, $1, $3     // 0010 0010 1111 0000
ADDI $5, $1, 3     // 0011 0011 0100 0011
SUB $2, $5, $6     // 0100 1011 1001 0000
ORI $4, $1, 6      // 1001 0011 0000 0110
AND $7, $4, $5     // 0110 1001 0111 1000
ANDI $6, $2, 7     // 0111 0101 1000 0111
OR $6, $7, $6      // 1000 1111 1011 0000
SUBI $4, $1, 2     // 0101 0011 0000 0010
MOV $7, $4         // 0000 1000 0011 1000
ADD $4, $4, $6     // 0010 1001 1010 0000
SUBI $5, $6, 1     // 0101 1101 0100 0001
ORI $3, $3, 4      // 1001 0110 1100 0100
AND $2, $1, $2     // 0110 0010 1001 0000
MOV $4, 7          // 0001 0001 0000 0111
```

Imagem 17. Instruções para execução.

Para a realização da simulação foi utilizado um clock com período de 100 ns e duração de 2000 ns, com o mesmo iniciando com 1. Durante os testes, foram realizados vários ajustes para que houvesse a sincronia entre as componentes, no entanto, houve a necessidade de retirada de alguns sinais internos, relacionados ao PC e aos pipes, pois estes estavam demonstrando atraso em relação ao esperado, também o sinal de estado da unidade de controle teve que ser retirado, pois o mesmo era atribuído ao próximo estado, que não havia tal previsão e, ao mesmo tempo, havia atraso de 1 ciclo na geração dos sinais de controle.

Um dos erros mais pertinentes foi no caso do banco de registradores, em que havia a escrita e leitura em um mesmo ciclo de clock, fazendo com que primeiro fosse feita a leitura e posteriormente a escrita para que houvesse a sincronia de dados, inicialmente o teste apresentava erro por conta dos sinais internos descritos anteriormente, fazendo com que uma instrução demorasse o dobro do tempo esperado como denotado a seguir:

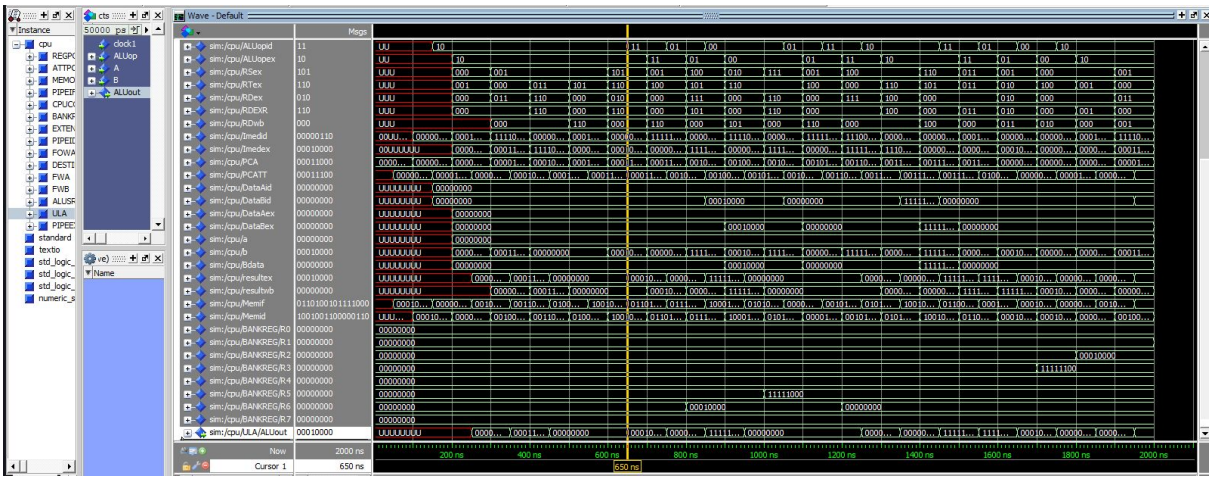

Imagem 18. Simulação da CPU pipeline com atraso.

No final, após resolver o erro, retirando a expressão de “clock’EVENT” na condição de leitura e escrita, a execução da simulação foi feita com sucesso e é denotada a seguir:

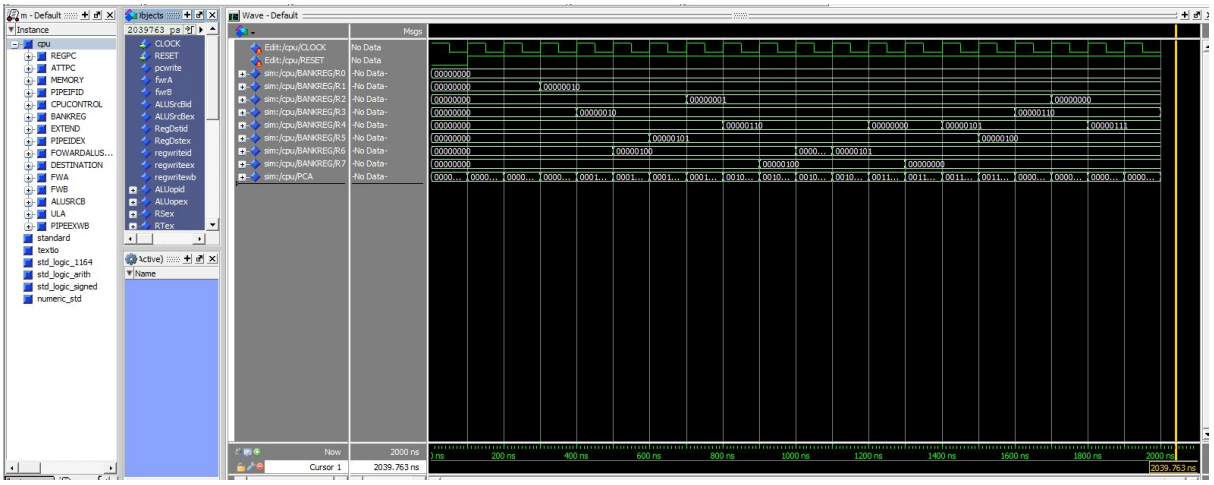

Imagem 19. Simulação da CPU pipeline.

4 - Bibliografia

http://people.sabanciuniv.edu/erkays/el310/MemoryModels.pdf

https://github.com/h-ssiqueira/CPU-multicycle