UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
INSTITUTO METRÓPOLE DIGITAL

Isaque Barbosa Martins

João Guilherme Lopes Alves da Costa

Processador RISC com pipeline

NATAL, RN
2023

Isaque Barbosa Martins

João Guilherme Lopes Alves da Costa

Processador RISC com pipeline

Relatório técnico apresentado na disciplina Organização de Computadores (DIM0129) para a primeira unidade do semestre letivo 2023.1 do curso Bacharelado em Tecnologia da Informação pela Universidade Federal do Rio Grande do Norte.

Orientador: Profº. Dr. Marcio Eduardo Kreutz

NATAL, RN
2023

Sumário

1 Decisões de Projeto

Este presente relatório tem como objetivo explicitar a abordagem adotada para implementar um Processador RISC com Pipeline. Esse trabalho foi produzido durante o período 2023.1 para a disciplina de Organização de Computadores pela Universidade Federal do Rio Grande do Norte (UFRN), aplicando os conceitos aprendidos em sala para implementar o processador.

1.1 Tamanho da palavra do processador

O tamanho da palavra do processador adotada foi de 32 bits, dada a necessidade do formato da palavra de instrução.

1.2 Formato da palavra de instrução

Por questões de padronização, construímos palavras de instrução de 32 Bits, onde tempos 3 tipos de palavras de instrução principais: Operações da ULA, Operações na Memória e Operações de Salto.

Para as Operações da ULA foi necessário o campo referente à Operação, um campo para a instrução a ser realizada pela ula, e 3 campos referentes ao endereço registrador fonte 1 e fonte 2 de onde virão os dados para operação, e o endereço do registrador destino.

Para as Operações de Memória foi necessário, também, o campo referente à Operação, e os campos de endereço fonte e destino.

Por fim, para as Operações de Salto foi necessário o campo referente à Operação, um campo de instrução utilizado nos Jumps que necessitam de comparação, e um campo de posição referente à posição para o qual o contador deve pular.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	OP			INSTRUÇÃO ULA			REGISTRADOR FONTE UM					REGISTRADOR FONTE 2					REGISTRADOR DESTINO																
AND	0	0	0	0	0	1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X											
OR	0	0	0	0	1	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X											
XOR	0	0	0	0	1	1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X											
NOT	0	0	0	1	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X											
CMP	0	0	0	1	0	1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X											
ADD	0	0	0	1	1	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X											
SUB	0	0	0	1	1	1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X											
	OP			INSTRUÇÃO ULA			REGISTRADOR FONTE					REGISTRADOR DESTINO																					
LD	0	0	1	X	X	X	X	X	X	X	X	X	X	X	X	X																
ST	0	1	0	X	X	X	X	X	X	X	X	X	X	X	X	X																
	OP			INSTRUÇÃO ULA			POSIÇÃO																										
J	0	1	1	X	X	X	X	X	X	X	X																					
JN	1	0	0	1	0	0	X	X	X	X	X																					
JZ	1	0	1	1	0	0	X	X	X	X	X																					

Entretanto nem todos os bits são usados para instrução, logo, segue abaixo a imagem com os bits significativos das instruções.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11
	OP			INSTRUÇÃO ULA			REGISTRADOR FONTE UM					REGISTRADOR FONTE 2					REGISTRADOR DESTINO				
AND	0	0	0	0	0	1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
OR	0	0	0	0	1	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
XOR	0	0	0	0	1	1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
NOT	0	0	0	1	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
CMP	0	0	0	1	0	1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
ADD	0	0	0	1	1	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
SUB	0	0	0	1	1	1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	OP			INSTRUÇÃO ULA			REGISTRADOR FONTE					REGISTRADOR DESTINO									
LD	0	0	1	X	X	X	X	X	X	X	X	X	X	X	X	X					
ST	0	1	0	X	X	X	X	X	X	X	X	X	X	X	X	X					
	OP			INSTRUÇÃO ULA			POSIÇÃO														
J	0	1	1	X	X	X	X	X	X	X	X										
JN	1	0	0	1	0	0	X	X	X	X	X										
JZ	1	0	1	1	0	0	X	X	X	X	X										

1.3 Modos de endereçamento de operandos

Foi utilizado o modo de endereçamento direto para todas as operações do processador.

1.4 Tamanho do banco de registradores

Foi adotado o tamanho de 32 registros de 32 bits para o banco de Registradores.

1.5 Tamanho das memórias de instruções e de dados

Foi adotado o tamanho de 32 registros de 32 bits para as memórias de instruções e dados.

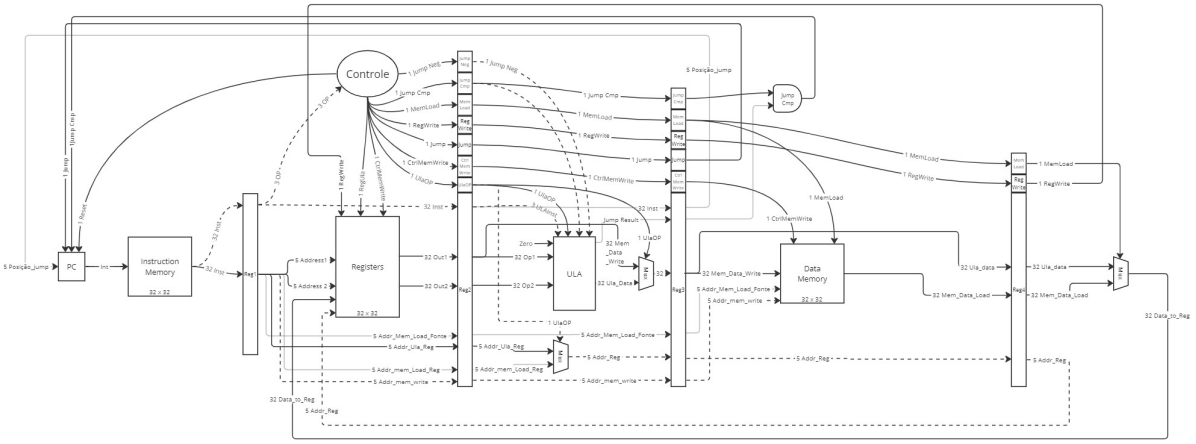
1.6 Número e tipos de barramentos (ou canais dedicados) da parte operativa

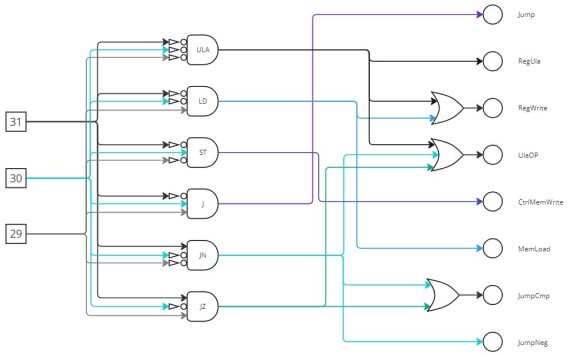
6 barramentos.

1.7 Organização do pipeline

O pipeline foi organizado em 5 estágios: Contador e seleção de instrução, movimentação do banco de registradores, execução da ULA, movimentação da memória e o envio de dados ao banco de registradores.

2 Diagramas





	Jump	RegUla	RegWrite	UlaOP	CtrlMem Write	MemLoad	JumpCmp	JumpNeg
AND	0	1	1	1	0	0	0	0
OR	0	1	1	1	0	0	0	0
XOR	0	1	1	1	0	0	0	0
NOT	0	1	1	1	0	0	0	0
CMP	0	1	1	1	0	0	0	0
ADD	0	1	1	1	0	0	0	0
SUB	0	1	1	1	0	0	0	0
LD	0	0	1	0	0	1	0	0
ST	0	0	0	0	1	0	0	0
J	1	0	0	0	0	0	0	0
JN	0	0	0	1	0	0	1	1
JZ	0	0	0	1	0	0	1	0

3 Implementação

3.1 Implementação do modelo da arquitetura em linguagem para descrição de hardware (VHDL ou SystemC)

O processador foi implementado na linguagem SystemC, para isto, nós implementamos separadamente os componentes: PC, memória de instrução, registradores de pipeline, banco de registradores, ULA, memória de dados, controlador e os MUX de 5 e 32 bits.

Para interligar os componentes, foi implementado um arquivo que instância os sinais que representam os fios entre os componentes.

3.2 Resultados de simulações da execução de instruções de pelo menos 3 algoritmos na arquitetura

Nas figuras 1, 2 e 3 é possível verificar o comportamento do processador implementado a partir de 3 algoritmos implementados por nós. Entretanto é possível notar nos algoritmos 2 e 3 que a execução não foi completa, tendo sido paradas no terceiro clock devido à erros de segmentation fault no qual não conseguimos encontrar a solução a tempo.

3.3 Relatório explicando e exemplificando a implementação da arquitetura e justificando as decisões de projeto acima elencadas.

Para a implementação da arquitetura, primeiro foi projetado o diagrama operativo de como seria o processador, tomando como base o processador MIPS inicialmente havíamos pensado em construir o processador com 4 fases de pipeline, entretanto, devido ao carregamento em registradores tanto da ula como da memória, optamos por 5 fases com uma fase separada para este carregamento.

Na memória do programa decidimos por fazer 2 sinais de controle sendo um para leitura e outro para escrita para melhor controle sobre as entradas.

Para a ula, temos a entrada de 3 sinais de controle, um que ditará se a ula será usada ou não, e os outros dois para coordenar as operações das instruções Jump se negativo e Jump se zero. Foi decidido 2 sinais para o jump, pois um ditará se está ocorrendo o jump ou não, e o outro ditará qual comparação deve ser feita. Ainda no mesmo estágio do pipeline temos 2 Mux, um de 32 bits que coordenará os dados de escrita em memória ou do resultado da ULA, e um de 5 bits que coordenará o endereço do registrador destino de operações da ULA ou o endereço de carregamento de dados da memória para o registrador.

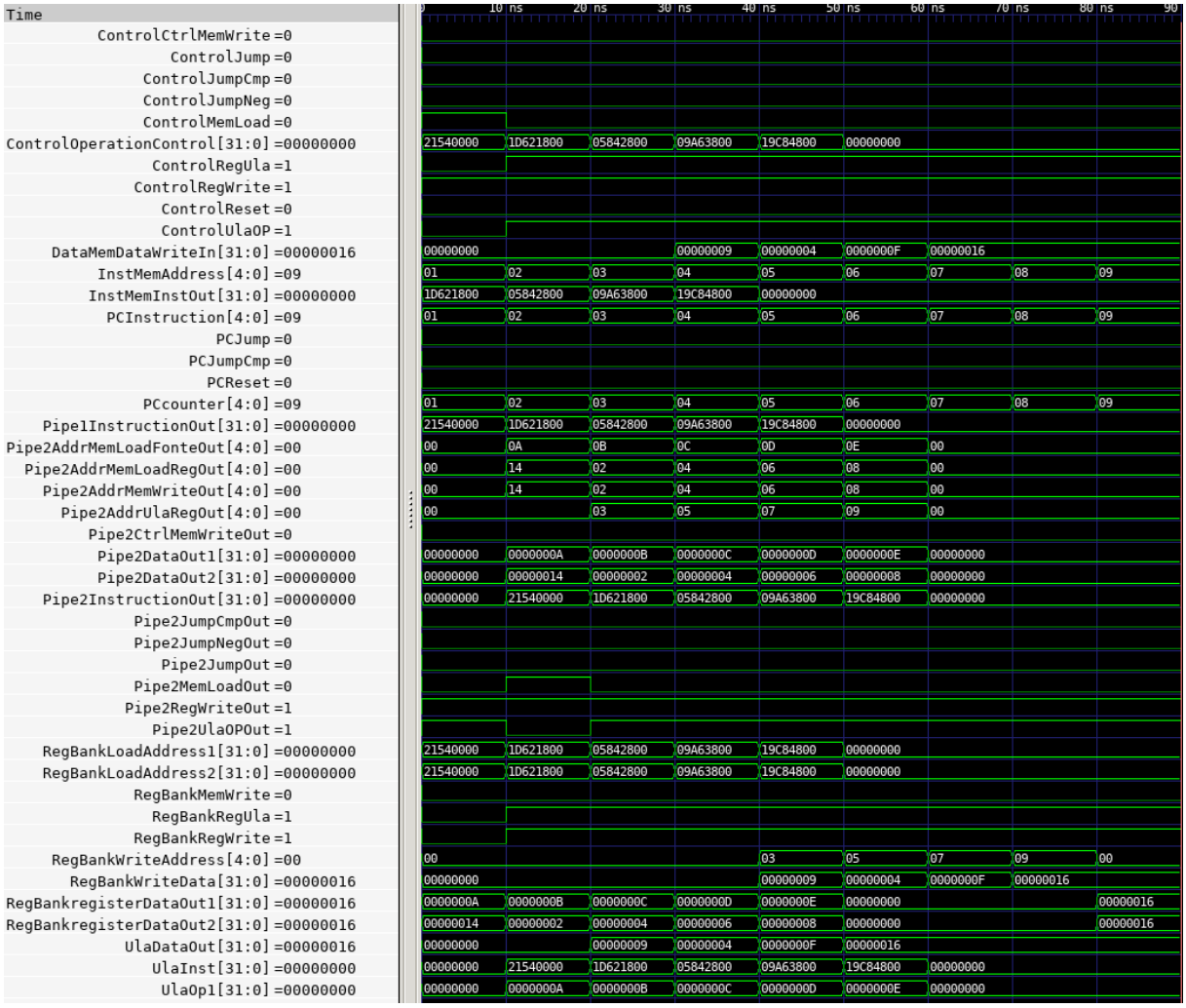


Figura 1 – Execução do primeiro algoritmo

Para o banco de registradores, há a entrada de 3 sinais de controle que determinam se irá escrever dados no registrador, se devem sair dados para a ULA ou se devem sair dados para a escrita na memória. E também há duas entradas de endereços apontando para os dados que devem sair.

Na memória de instruções há somente uma entrada que representa a posição da instrução que deve ser lida a seguir, e por fim, o contador do programa irá incrementar a cada clock a posição da instrução que deve ser lida, ou, quando há uma instrução de jump, ela pulará para a posição que recebeu da instrução.

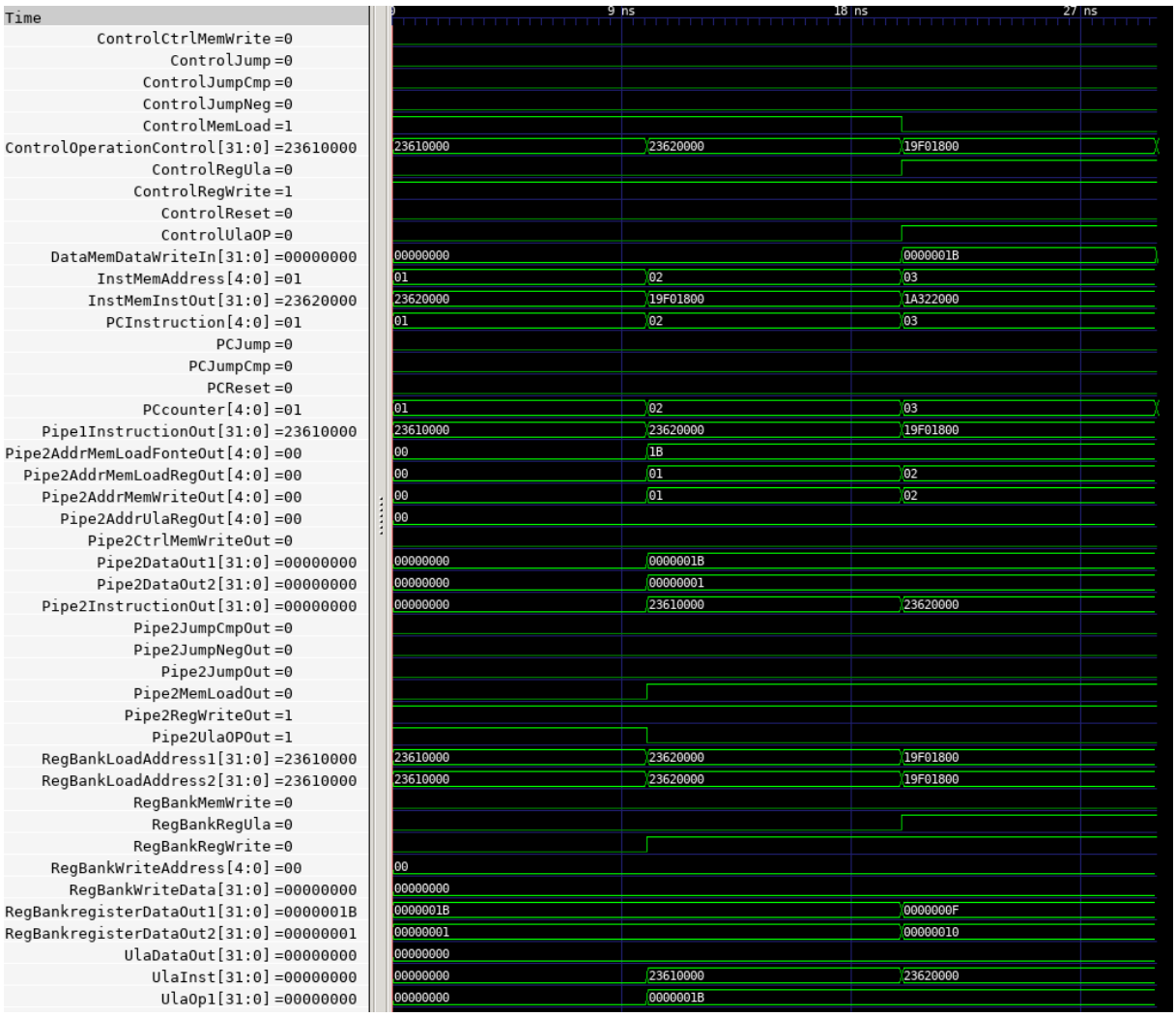


Figura 2 – Execução do segundo algoritmo

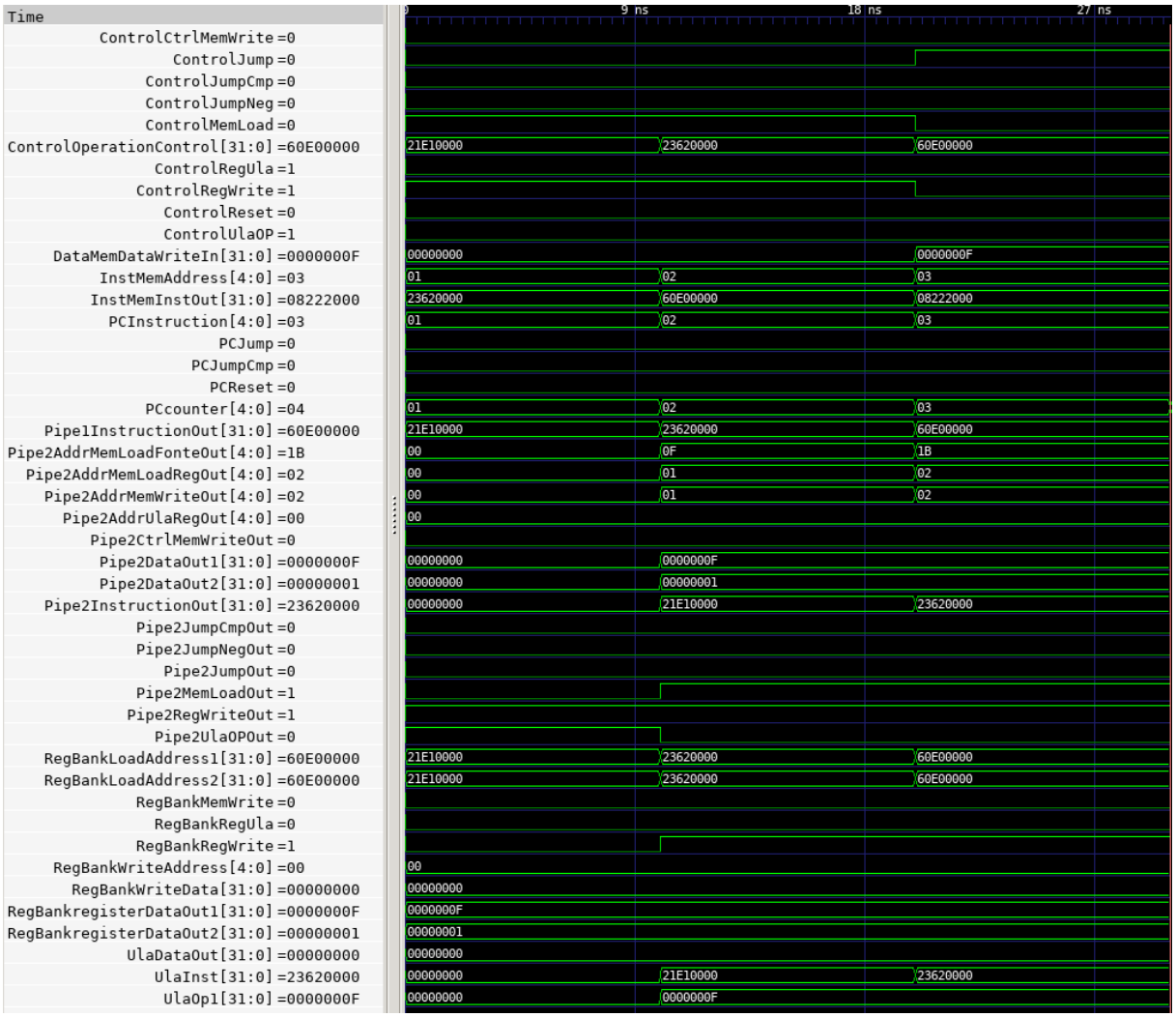


Figura 3 – Execução do terceiro algoritmo

4 Conclusão

Concluímos que foi possível construir um processador RISC com 5 etapas de pipeline, entretanto ainda há brechas no sistema que poderiam ser melhoradas para o completo funcionamento do processador.