UNIVERSIDADE FEDERAL DA FRONTEIRA SUL
CURSO DE CIÊNCIA DA COMPUTAÇÃO
CAMPUS CHAPECÓ

ACÁCIA DOS CAMPOS DA TERRA
JOÃO PEDRO WINCKLER BERNARDI

RELATÓRIO DA IMPLEMENTAÇÃO DE UM SISTEMA EM VHDL QUE SIMULA UMA CPU- MONOCICLO- COM A ARQUITETURA MIPS

CHAPECÓ
2015

1. INTRODUÇÃO

O trabalho consiste na implementação de um sistema em VHDL que simule uma CPU – Monociclo (todas as etapas do ciclo de instrução devem ocorrer em um único ciclo de clock)– com a arquitetura MIPS. O sistema deverá conter 32 registradores, cada um com 32 bits, que estarão no banco de registradores; também deverá ter suporte às instruções “LW”, “SW”, “AND”, “SUB”, ADD”, “SLT”, “BEQ”, “J”, “ADDI”, “SLTI”, “BNE”, “JAL” e “JR”. A palavra de processamento no sistema é um vetor de 32 bits, assim como as instruções com que trabalha o processador.

Sendo assim, foram implementados módulos específicos e estabelecidos os sinais que o controle deve enviar ao datapath, de forma que satisfaçam as exigências para o correto funcionamento do datapath da CPU.

2. SOFTWARES UTILIZADOS

Para elaboração da CPU – Monociclo, foram utilizados os softwares MultiSim 13.0, para análise das formas de onda das instruções e ModelSim, para codificação em VHDL dos testbenchs e do datapath.

3. IMPLEMENTAÇÃO DOS MÓDULOS

A seguir serão apresentados os módulos presentes na arquitetura:

- **MULTIPLEXADOR 2X1**

A partir de um seletor, é selecionada uma das entradas e enviada à saída. Foi implementado um multiplexador com entradas de 5 bits e de 32 bits.

- **SOMADOR**

Efetua a soma entre duas entradas e envia o resultado à saída.

- **ULA**

A partir de um seletor, realiza determinada operação com as duas

entradas. O seletor funciona da seguinte forma:

Seletor	Operação
000	AND
001	OR
010	ADD
110	SUB
111	SLT

- **BANCO DE REGISTRADORES**

Contém 32 registradores, que serão utilizados durante a execução do programa. Foi implementado de forma algorítmica, usando 32 sinais de 32 bits.

- **EXTENSOR DE SINAL**

Transforma uma entrada de 16 bits em uma de 32 bits. Concatena a entrada com 16 bits '0';

- **CONTROLADOR DA ULA**

Define o que deve ser enviado como seletor à ULA para execução dos comandos. A partir do sinal ALUOp, selecionamos qual é a saída desejada.

- **CONTROLADOR DO DATAPATH**

Faz as atribuições de sinais de controle para cada comando de acordo com a instrução que recebe.

- **MEMÓRIA DE DADOS**

Local onde são armazenados dados que não estão em uso no momento. Foi implementado com um vetor de sinais de 32 bits. Com o endereço, transforma-se o valor em um inteiro e tem acesso àquela posição na memória.

- **CONTADOR DE PROGRAMA (PC)**

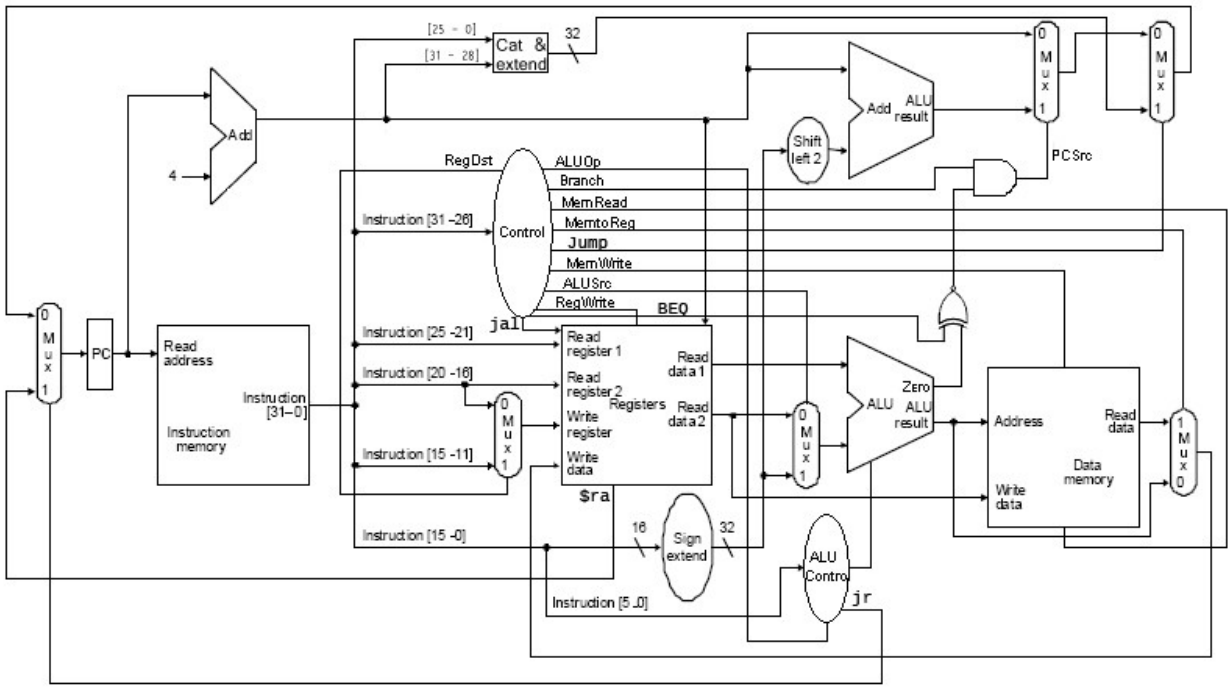
Registrador que contém o endereço da próxima instrução a ser executada.

- **DESLOCADOR DE BITS À ESQUERDA**

Desloca a informação dois bits para a esquerda concatenando com dois bits '0'.

4. DATAPATH DA CPU

A imagem a seguir mostra o datapath da CPU, ou seja, o caminho que os dados percorrem para se obter o resultado correto das operações ordenadas.



5. SINAIS DE CONTROLE

A seguir será apresentada a tabela com valores necessários aos seus respectivos sinais, necessários para o funcionamento correto do datapath apresentado acima. A tabela mostra também os valores dos mesmos em cada uma das instruções suportadas.

	R-format	LW	SW	BEQ	Jump	ADDI	SLTI	BNE	JAL
RegDst	1	0	X	X	X	0	0	X	X
Branch	0	0	0	1	0	0	0	1	0
Jump	0	0	0	0	1	0	0	0	1
MemRead	0	1	0	0	0	0	0	0	0
MemtoReg	0	1	X	X	X	0	0	X	X
ALUOp1	1	0	0	0	0	0	1	0	0
ALUOp2	1	0	0	1	0	0	0	1	0
MemWrite	0	0	1	0	0	0	0	0	0
ALUSrc	0	1	1	0	X	1	1	0	X
RegWrite	1	1	0	0	0	1	1	0	0
jr	0	0	0	0	0	0	0	0	0
jal	0	0	0	0	0	0	0	0	1
beq	X	X	X	1	X	X	X	0	X

- **Para instruções R-format:**

O sinal RegDst deverá estar em 1, pois instruções do tipo R têm como registrador de destino (onde serão armazenados os resultados das operações) o registrador que se encontra no campo rd (instrução[15-11]). Para que o datapath escreva no registrador correto, o mux deverá selecionar a segunda entrada.

O sinal branch deverá estar em 0, pois operações de formato R não farão cálculos para determinar se haverá um pulo de instruções ou não.

O sinal jump deverá estar em 0, pois queremos o resultado do mux anterior, onde o novo valor de PC é calculado (seja apenas para avançar uma instrução ou pular para alguma outra – como em branchs).

O sinal MemRead deverá estar em 0, pois não será lido nenhum valor da memória, porque em instruções do tipo R fazemos as operações de acordo com os valores contidos em registradores, apenas.

O sinal MemtoReg deverá estar em 0, para o mux selecionar diretamente a saída da ULA (onde a operação foi feita) e enviar o sinal para o banco de registrador, para então o resultado ser escrito no registrador de destino (selecionado em RegDst).

Os sinais ALUOp1 e ALUOp2 compõe um único seletor para o controlador da ULA, que é o módulo que determinará qual operação a ULA deverá fazer. Então, para instruções do tipo R, ALUOp1 deverá estar em 1 e ALUOp2 também deverá estar em 1. Dessa forma, o controlador da ULA saberá que é uma operação aritmética que deverá ser feita, dependendo apenas do campo funct

([5-0]) da instrução que determina qual das operações aritméticas deverá ser executada.

O sinal MemWrite deverá estar em 0, pois o resultado sempre será armazenado no registrador rd e não na memória quando estivermos fazendo operações do tipo R.

O sinal ALUSrc deverá estar em 0, para que o mux selecione o valor do registrador rt como segundo operando de entrada na ULA.

O sinal RegWrite deverá estar em 1, pois o resultado da ULA será armazenado no registrador rd (instrução [15-11]), sendo assim, haverá escrita em registrador.

O sinal jr deverá estar em 0, para que o valor de PC receba, ou o cálculo do branch (se for o caso) ou só o valor PC + 4 (para avançar instrução).

O sinal jal deverá estar em 0 também, pois não será feita a operação de desvio incondicional jal quando estivermos fazendo operações com formato R de instrução.

O valor do sinal beq não importa para nós, pois o valor do sinal branch já está em 0, então a porta and que determina o valor do seletor do mux não será 1. Sendo assim, não será feita nenhuma operação de branch

- **Para instruções LW:**

O sinal RegDst deverá estar em 0, pois instruções LW têm como registrador de destino (onde será armazenado o valor lido da memória) o registrador que se encontra no campo rt (instrução[20-16]). Para que o datapath escreva no registrador correto, o mux deverá selecionar a primeira entrada.

O sinal branch deverá estar em 0, pois nessas operações não serão feitos cálculos para determinar se haverá um pulo de instruções ou não.

O sinal jump deverá estar em 0, pois queremos o resultado do mux anterior, onde o novo valor de PC é calculado (seja apenas para avançar uma instrução ou pular para alguma outra – como em branchs).

O sinal MemRead deverá estar em 1, pois será lido um valor da memória, pois é exatamente isso que esta instrução está incubida de fazer.

O sinal MemtoReg deverá estar em 1, para o mux selecionar o valor lido da memória e enviar o sinal para o banco de registrador, para então o

resultado ser escrito no registrador de destino (selecionado em RegDst).

Os sinais ALUOp1 e ALUOp2 compõe um único seletor para o controlador da ULA, que é o módulo que determinará qual operação a ULA deverá fazer. Então, para instruções LW, ALUOp1 deverá estar em 0 e ALUOp2 também deverá estar em 0. Dessa forma, o controlador da ULA saberá que é uma operação de leitura ou escrita da memória, uma operação jump ou jal, ou uma operação addi que será feita.

O sinal MemWrite deverá estar em 0, pois estaremos lendo um dado da memória e não guardando nela.

O sinal ALUSrc deverá estar em 1, para que o mux selecione o valor contido no offset como segundo operando de entrada na ULA.

O sinal RegWrite deverá estar em 1, pois o valor da memória que foi lido será armazenado no registrador rt (instrução [20-16]), portanto haverá escrita em registrador.

O sinal jr deverá estar em 0, para que o valor de PC receba, ou o cálculo do branch (se for o caso) ou só o valor PC + 4 (para avançar instrução).

O sinal jal deverá estar em 0 também, pois não será feita a operação de desvio incondicional jal quando estivermos fazendo operações com leitura de memória.

O valor do sinal beq não importa para nós, pois o valor do sinal branch já está em 0, então a porta and que determina o valor do seletor do mux não será 1. Sendo assim, não será feita nenhuma operação de branch.

- **Para instruções SW:**

O valor em RegDst não importa, pois a instrução SW irá armazenar valores na memória e não em algum registrador, então não haverá um registrador de destino.

O sinal branch deverá estar em 0, pois nessas operações não serão feitos cálculos para determinar se haverá um pulo de instruções ou não.

O sinal jump deverá estar em 0, pois queremos o resultado do mux anterior, onde o novo valor de PC é calculado (seja apenas para avançar uma instrução ou pular para alguma outra – como em branchs).

O sinal MemRead deverá estar em 0, pois não será lido nenhum valor da memória.

O valor de MemtoReg não importa, pois nenhum dado será escrito em registrador, então não importa o que seja mandado para lá.

Os sinais ALUOp1 e ALUOp2 compõe um único seletor para o controlador da ULA, que é o módulo que determinará qual operação a ULA deverá fazer. Então, para instruções SW, ALUOp1 deverá estar em 0 e ALUOp2 também deverá estar em 0. Dessa forma, o controlador da ULA saberá que é uma operação de leitura ou escrita da memória, uma operação jump ou jal, ou uma operação addi que será feita.

O sinal MemWrite deverá estar em 1, pois estaremos armazenando um dado na memória.

O sinal ALUSrc deverá estar em 1, para que o mux selecione o valor contido no offset como segundo operando de entrada na ULA.

O sinal RegWrite deverá estar em 0, pois nada será escrito em registrador.

O sinal jr deverá estar em 0, para que o valor de PC receba, ou o cálculo do branch (se for o caso) ou só o valor PC + 4 (para avançar instrução).

O sinal jal deverá estar em 0 também, pois não será feita a operação de desvio incondicional jal quando estivermos fazendo operações com leitura de memória.

O valor do sinal beq não importa para nós, pois o valor do sinal branch já está em 0, então a porta and que determina o valor do seletor do mux não será 1. Sendo assim, não será feita nenhuma operação de branch.

- **Para instruções BEQ:**

O valor em RegDst não importa, pois essa instrução verificará se dois operandos são iguais e, se forem, efetuará um pulo, mas escreverá nada em registrador, não precisando selecionar nenhum de destino.

O sinal branch deverá estar em 1, pois caso o resultado da subtração entre os operandos na ULA resultar em 0, o pulo será feito e para isso, o sinal branch também deve estar em 1.

O sinal jump deverá estar em 0, pois queremos o resultado do mux anterior, onde o novo valor de PC é calculado (seja apenas para avançar uma instrução ou pular para alguma outra – como em branchs).

O sinal MemRead deverá estar em 0, pois não será lido nenhum valor da

memória.

O valor de MemtoReg não importa, pois nada será escrito em registrador.

Os sinais ALUOp1 e ALUOp2 compõe um único seletor para o controlador da ULA, que é o módulo que determinará qual operação a ULA deverá fazer. Então, para instruções BEQ, ALUOp1 deverá estar em 0 e ALUOp2 deverá estar em 1. Dessa forma, o controlador da ULA saberá que é uma operação de branch que será feita.

O sinal MemWrite deverá estar em 0, pois não será guardado nenhum dado na memória.

O sinal ALUSrc deverá estar em 0, para que o mux selecione o valor do registrador rt como segundo operando de entrada na ULA.

O sinal RegWrite deverá estar em 0, pois nada será escrito em registrador.

O sinal jr deverá estar em 0, para que o valor de PC receba, ou o cálculo do branch (se for o caso) ou só o valor PC + 4 (para avançar instrução).

O sinal jal deverá estar em 0 também, pois não será feita a operação de desvio incondicional jal quando estivermos fazendo operações com leitura de memória.

O valor do sinal beq deverá ser 1, pois o valor do sinal branch já está em 1, então a porta and que determina o valor do seletor do mux deverá resultar em 1 caso o branch seja verdadeiro, ou seja, ambos os operandos sejam iguais. Para isso, o sinal beq e zero (que sai da ULA) deverão entrar em uma porta nor e o resultado desta vai para a porta and.

- **Para instruções Jump:**

O valor em RegDst não importa, pois essa instrução efetuará um pulo nas instruções, mas escreverá nada em registrador, não precisando selecionar nenhum de destino.

O sinal branch deverá estar em 0, pois não queremos o resultado do cálculo de nenhum branch.

O sinal jump deverá estar em 1, queremos que o seletor da ULA pegue a segunda entrada, que é onde está calculado o novo valor de PC, de acordo com o offset da instrução concatenado e extendido.

O sinal MemRead deverá estar em 0, pois não será lido nenhum valor da memória.

O valor de MemtoReg não importa, pois nada será escrito em registrador.

Os sinais ALUOp1 e ALUOp2 compõe um único seletor para o controlador da ULA, que é o módulo que determinará qual operação a ULA deverá fazer. Então, para instruções Jump, ALUOp1 deverá estar em 0 e ALUOp2 também deverá estar em 0. Dessa forma, o controlador da ULA saberá que é uma operação de leitura ou escrita da memória, uma operação jump ou jal, ou uma operação addi que será feita.

O sinal MemWrite deverá estar em 0, pois não será guardado nenhum dado na memória.

O valor de ALUSrc não importa, pois não será feito nenhum cálculo na ULA, não precisando selecionar operando.

O sinal RegWrite deverá estar em 0, pois nada será escrito em registrador.

O sinal jr deverá estar em 0, para que o valor de PC receba, ou o cálculo do branch (se for o caso) ou só o valor PC + 4 (para avançar instrução).

O sinal jal deverá estar em 0 também, pois não será feita a operação de desvio incondicional jal quando estivermos fazendo operações com leitura de memória.

O valor do sinal beq não importa para nós, pois o valor do sinal branch já está em 0, então a porta and que determina o valor do seletor do mux não será 1. Sendo assim, não será feita nenhuma operação de branch.

- **Para instruções ADDI:**

O sinal RegDst deverá estar em 0, pois instrução ADDI têm como registrador de destino (onde serão armazenados os resultados das operações) o registrador que se encontra no campo rt (instrução[20-16]). Para que o datapath escreva no registrador correto, o mux deverá selecionar a primeira entrada.

O sinal branch deverá estar em 0, pois operações de formato R não farão cálculos para determinar se haverá um pulo de instruções ou não.

O sinal jump deverá estar em 0, pois queremos o resultado do mux

anterior, onde o novo valor de PC é calculado (seja apenas para avançar uma instrução ou pular para alguma outra – como em branchs).

O sinal MemRead deverá estar em 0, pois não será lido nenhum valor da memória, porque em instruções do tipo R fazemos as operações de acordo com os valores contidos em registradores, apenas.

O sinal MemtoReg deverá estar em 0, para o mux selecionar diretamente a saída da ULA (onde a operação foi feita) e enviar o sinal para o banco de registrador, para então o resultado ser escrito no registrador de destino (selecionado em RegDst).

Os sinais ALUOp1 e ALUOp2 compõe um único seletor para o controlador da ULA, que é o módulo que determinará qual operação a ULA deverá fazer. Então, para instruções do tipo R, ALUOp1 deverá estar em 0 e ALUOp2 também deverá estar em 0. Dessa forma, o controlador da ULA saberá que é uma operação de leitura ou escrita da memória, uma operação jump ou jal, ou uma operação addi que será feita.

O sinal MemWrite deverá estar em 0, pois o resultado sempre será armazenado no registrador rt e não na memória quando estivermos fazendo esta operação

O sinal ALUSrc deverá estar em 1, para que o mux selecione o valor do do offset como segundo operando de entrada na ULA.

O sinal RegWrite deverá estar em 1, pois o resultado da ULA será armazenado no registrador rt (instrução [20-16]), sendo assim, haverá escrita em registrador.

O sinal jr deverá estar em 0, para que o valor de PC receba, ou o cálculo do branch (se for o caso) ou só o valor PC + 4 (para avançar instrução).

O sinal jal deverá estar em 0 também, pois não será feita a operação de desvio incondicional jal quando estivermos fazendo operações com formato R de instrução.

O valor do sinal beq não importa para nós, pois o valor do sinal branch já está em 0, então a porta and que determina o valor do seletor do mux não será 1. Sendo assim, não será feita nenhuma operação de branch

- **Para instruções SLTI:**

O sinal RegDst deverá estar em 0, pois instrução SLTI têm como

registrador de destino (onde serão armazenados os resultados das operações) o registrador que se encontra no campo rd (instrução[20-16]). Para que o datapath escreva no registrador correto, o mux deverá selecionar a segunda entrada.

O sinal branch deverá estar em 0, pois operações SLTI não farão cálculos para determinar se haverá um pulo de instruções ou não.

O sinal jump deverá estar em 0, pois queremos o resultado do mux anterior, onde o novo valor de PC é calculado (seja apenas para avançar uma instrução ou pular para alguma outra – como em branchs).

O sinal MemRead deverá estar em 0, pois não será lido nenhum valor da memória, porque em instruções SLTI fazemos as operações de acordo com os valores contidos em registradores, apenas.

O sinal MemtoReg deverá estar em 0, para o mux selecionar diretamente a saída da ULA (onde a operação foi feita) e enviar o sinal para o banco de registrador, para então o resultado ser escrito no registrador de destino (selecionado em RegDst).

Os sinais ALUOp1 e ALUOp2 compõe um único seletor para o controlador da ULA, que é o módulo que determinará qual operação a ULA deverá fazer. Então, para instruções SLTI, ALUOp1 deverá estar em 1 e ALUOp2 deverá estar em 0. Dessa forma, o controlador da ULA saberá que é uma operação SLTI que será feita.

O sinal MemWrite deverá estar em 0, pois o resultado sempre será armazenado no registrador rd e não na memória quando estivermos fazendo esta operação

O sinal ALUSrc deverá estar em 1, para que o mux selecione o valor do do offset como segundo operando de entrada na ULA.

O sinal RegWrite deverá estar em 1, pois o resultado da ULA será armazenado no registrador rd (instrução [20-16]), sendo assim, haverá escrita em registrador.

O sinal jr deverá estar em 0, para que o valor de PC receba, ou o cálculo do branch (se for o caso) ou só o valor PC + 4 (para avançar instrução).

O sinal jal deverá estar em 0 também, pois não será feita a operação de desvio incondicional jal quando estivermos fazendo operações com formato R de instrução.

O valor do sinal beq não importa para nós, pois o valor do sinal branch já

está em 0, então a porta and que determina o valor do seletor do mux não será 1. Sendo assim, não será feita nenhuma operação de branch

- **Para instruções BNE:**

O valor em RegDst não importa, pois essa instrução verificará se dois operandos são iguais e, se forem, efetuará um pulo, mas escreverá nada em registrador, não precisando selecionar nenhum de destino.

O sinal branch deverá estar em 1, pois caso o resultado da subtração entre os operandos na ULA resultar em 0, o pulo será feito e para isso, o sinal branch também deve estar em 1.

O sinal jump deverá estar em 0, pois queremos o resultado do mux anterior, onde o novo valor de PC é calculado (seja apenas para avançar uma instrução ou pular para alguma outra – como em branchs).

O sinal MemRead deverá estar em 0, pois não será lido nenhum valor da memória.

O valor de MemtoReg não importa, pois nada será escrito em registrador.

Os sinais ALUOp1 e ALUOp2 compõe um único seletor para o controlador da ULA, que é o módulo que determinará qual operação a ULA deverá fazer. Então, para instruções BNE, ALUOp1 deverá estar em 0 e ALUOp2 deverá estar em 1. Dessa forma, o controlador da ULA saberá que é uma operação de branch que será feita.

O sinal MemWrite deverá estar em 0, pois não será guardado nenhum dado na memória.

O sinal ALUSrc deverá estar em 0, para que o mux selecione o valor do registrador rt como segundo operando de entrada na ULA.

O sinal RegWrite deverá estar em 0, pois nada será escrito em registrador.

O sinal jr deverá estar em 0, para que o valor de PC receba, ou o cálculo do branch (se for o caso) ou só o valor PC + 4 (para avançar instrução).

O sinal jal deverá estar em 0 também, pois não será feita a operação de desvio incondicional jal quando estivermos fazendo operações com leitura de memória.

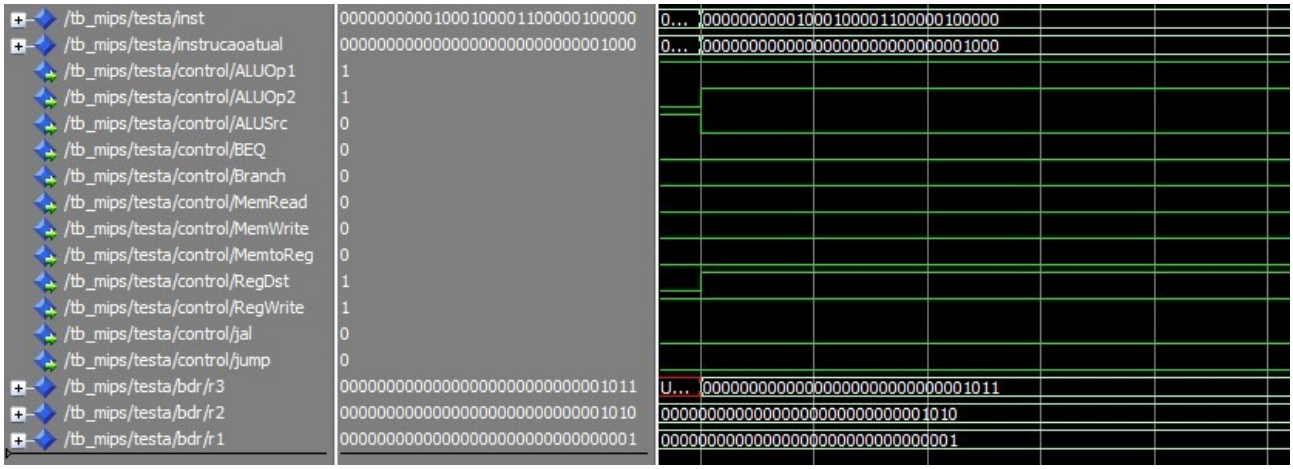
O valor do sinal beq deverá ser 0, pois o valor do sinal branch já está em

1, então a porta and que determina o valor do seletor do mux deverá resultar em 1 caso o branch seja verdadeiro, ou seja, ambos os operandos são diferentes. Para isso, o sinal beq e zero (que sai da ULA) deverão entrar em uma porta nor e o resultado desta vai para a porta and.

6. FORMAS DE ONDA

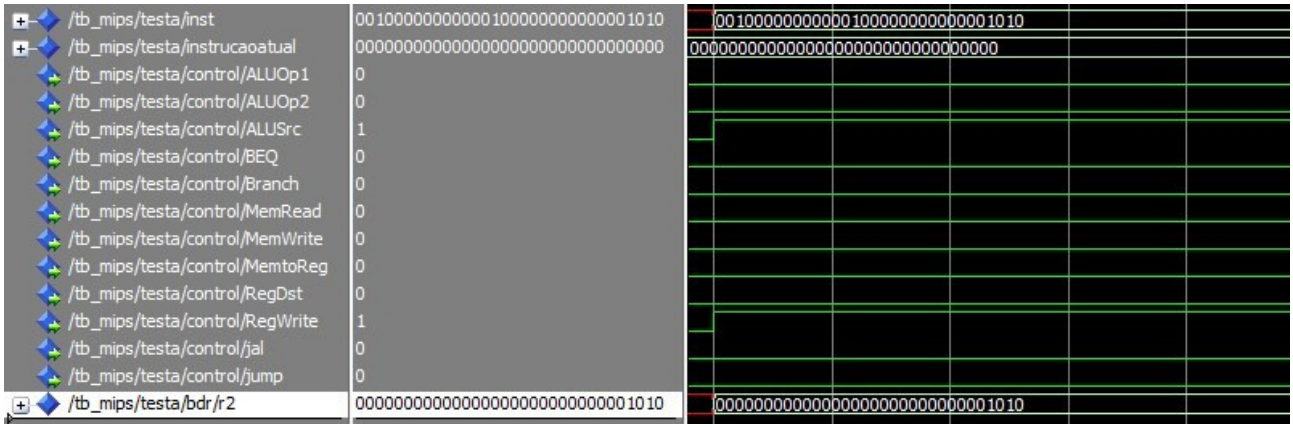
A seguir serão apresentadas formas de onda, uma de cada formato suportado pelo datapath, e suas descrições.

Ao executar "000000 00001 00010 00011 00000 100000" será feita uma adição entre os valores contigo no registrador 1(rs) e 2(rt) e o resultado será armazenado no registrador 3(rd).



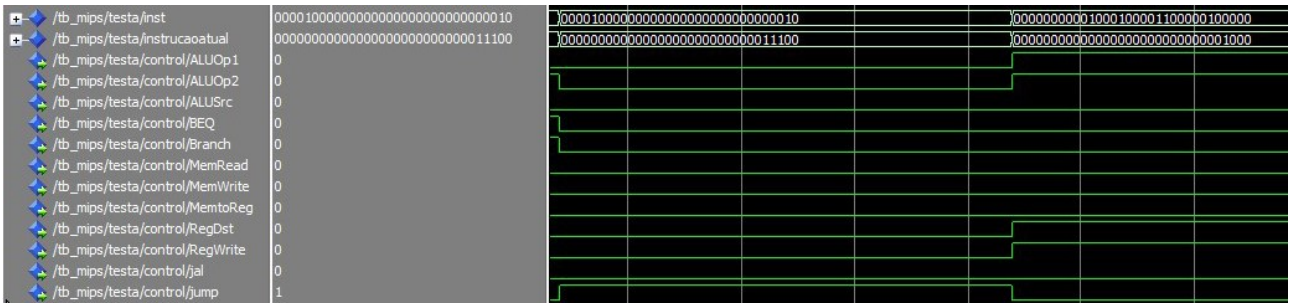
Instrução ADD (formato R) – soma os valores de rs e rt e armazena em rd.

Ao executar "001000 00000 00010 0000000000001010" será feita uma adição com constante, onde o valor do registrador 0 (rs) e o valor 10 (offset) serão somados e armazenados no registrador 2 (rt).



Instrução ADDI (formato I) – soma os valores de rs e offset e armazena em rt

Ao executar "000010 00000000000000000000000010" será feito o cálculo de forma que PC agora passe a apontar para a instrução 2 (informação presente no offset).



Instrução Jump (fomato J) – atualiza o valor de PC de acordo com o offset

7. CONCLUSÃO

A partir do desenho do datapath do MIPS – Monociclo, o mesmo foi implementado usando a linguagem VHDL, juntamente com o testbench que calcula todos os números da sequência Fibonacci menores que 500 e o testbench que faz a ordenação de um vetor na memória.

Há um problema com a sincronia na memória, por isso o testbench de ordenação pode não funcionar.