PODER EXECUTIVO

MINISTÉRIO DA EDUCAÇÃO

UNIVERSIDADE FEDERAL DE RORAIMA

DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

ARQUITETURA E ORGANIZAÇÃO DE COMPUTADORES

RELATÓRIO DO PROJETO: PROCESSADOR MK-IV

ALUNOS:

Hendrick Silva Ferreira - 202026830

Novembro de 2023

Boa Vista/Roraima

UFRR

PODER EXECUTIVO

MINISTÉRIO DA EDUCAÇÃO

UNIVERSIDADE FEDERAL DE RORAIMA

DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

ARQUITETURA E ORGANIZAÇÃO DE COMPUTADORES

RELATÓRIO DO PROJETO: PROCESSADOR MK-IV

Novembro de 2023

Boa Vista/Roraima

Resumo

Com os conhecimentos que foram desenvolvidos no decorrer da disciplina foi solicitado que os alunos, como representação da nota de projeto final da disciplina desenvolvessem um processador MIPS uniciclo de 16 bits. No trabalho serão demonstrados tanto as técnicas usadas para criar o mesmo tanto quanto o processador em funcionamento.

Conteúdo

1 Especificação....7
1.1 Plataforma de desenvolvimento....7
1.2 Conjunto de instruções....7
1.3 Descrição do Hardware....10
1.3.1 PC....10
1.3.2 Memoria de instruções....11
1.3.3 Banco de Registradores....12
1.3.4 ULA....13
1.3.5 Memória de Dados....13
1.3.6 Unidade de Controle....14
1.3.7 Somador PC....16
1.3.8 Extensor de Sinal....17
1.3.9 Shift à esquerda....17
1.3.10 Multiplexadores....18
1.3.11 Somador....19
1.3.12 Clock....20
1.4 Datapath....20
2 Simulações e Testes....22
3 Considerações finais....26

Lista de Figuras

Figura 1 - Especificações no Quartus 6

Figura 2 – PC RTL View 11

Figura 3 – Memoria de instrução RTL View 12

Figura 4 – Banco de Registradores RTL View.....12

Figura 5 – ULA RTL View.....13

Figura 6 – Memória de dados RTL View 14

Figura 7 – Unidade de Controle RTL View.....16

Figura 8 – Somador PC RTL View.....17

Figura 9 – Extensor de Sinal RTL View.....17

Figura 10 – Shifter de 2 bits a esquerda RTL View.....18

Figura 11 – Shifter de 2 bits a esquerda e concatena com PC RTL View.....18

Figura 12 – Multiplexador RTL View 19

Figura 13 – Somador RTL View 19

Figura 14 – Datapath RTL View……21

Figura 15 - Resultado na waveform exemplo 1 22

Figura 16 - Resultado na waveform exemplo 2 23

Figura 17 - Resultado na waveform exemplo 3 24

Figura 18 - Resultado na waveform exemplo 4.....25

Figura 19 - Resultado na waveform exemplo 5.....25

Lista de Tabelas

Tabela 1 – Tabela com a lista de OPCODES utilizadas pelo processador 9
Tabela 2 – Flags Unidade de Controle.....15
Tabela 3 – Exemplo 1 – multiplicação de dois valores.....22
Tabela 4 – Exemplo 2 – multiplicação de dois valores.....23
Tabela 5 – Exemplo 3 – multiplicação de dois valores.....23
Tabela 6 – Exemplo 4 – Load de um valor 24
Tabela 7 – Exemplo 5 – jump.....25

1 Especificação

O processador MK-IV foi desenvolvido por dois alunos do curso de ciência da computação, a linguagem utilizada foi a linguagem de síntese de circuitos digitais VHDL. O processador possui o conjunto de instruções de 16 bits e realiza operações do tipo R, I e J.

1.1 Plataforma de desenvolvimento

Para a implementação do processador MK-IV foi utilizado a IDE: Quartus Prime 18.0 Lite Edition

Revision Name	Processador_v1_uniciclo
Top-level Entity Name	DataPath
Family	Cyclone V
Device	5CGXFC7C7F23C8
Timing Models	Final
Logic utilization (in ALMs)	92 / 56,480 (< 1 %)
Total registers	16
Total pins	185 / 268 (69 %)
Total virtual pins	0
Total block memory bits	0 / 7,024,640 (0 %)
Total DSP Blocks	0 / 156 (0 %)
Total HSSI RX PCSs	0 / 6 (0 %)
Total HSSI PMA RX Deserializers	0 / 6 (0 %)
Total HSSI TX PCSs	0 / 6 (0 %)
Total HSSI PMA TX Serializers	0 / 6 (0 %)
Total PLLs	0 / 13 (0 %)
Total DLLs	0 / 4 (0 %)

Figura 1 - Especificações no Quartus

1.2 Conjunto de instruções

O processador MK-IV possui 8 registradores: S0, S1, S2, S3, S4, S5, S6, S7. Assim como 13 formatos de instruções de 16 bits cada, Instruções do tipo R (operações

aritméticas), tipo I (load, store, BEQ, BNE), tipo J (desvios), seguem algumas considerações sobre as estruturas contidas nas instruções:

- **OPCODE**: A operação básica a ser executada pelo processador, tradicionalmente chamado de código de operação;
- **RS**: O registrador contendo o primeiro operando fonte e adicionalmente para alguns tipos de instruções (ex. instruções do tipo R) é o registrador de destino;
- **RT**: O registrador contendo o segundo operando fonte;
- **RD**: Registrador onde será armazenado o resultado da operação

Tipo de Instruções:

- INSTRUÇÃO DO TIPO R:

Esse formato de instrução é caracterizado pelas operações aritméticas, as operações que são suportadas pelo MK-IV são soma, subtração, multiplicação, and e or.

Formato para escrita de código na linguagem :

Opcode	Reg1	Reg2	Reg3	Operação

Formato para escrita em código binário:

4 bits	3 bits	3 bits	3 bits	3 bits
15-12	11-9	8 - 6	5 - 3	2 - 0
Opcode	Reg3	Reg 1	Reg2	Funct

- INSTRUÇÃO DO TIPO I

São consideradas as instruções que compreendem loads e stores que são operações que trabalham diretamente com a memória e também engola as operações de desvios condicionais como BEQ e BNE.

Formato para escrita de código na linguagem :

Opcode	Reg1	Reg 2	Valor

Formato para escrita em código binário:

4 bits	3 bits	3 bits	6 bits
15-12	11-9	8 - 6	5 - 0
Opcode	Reg2	Reg1	Valor

- INSTRUÇÃO DO TIPO J

São consideradas as instruções que fazem os pulos, ou desvios, ou seja, são as operações de endereçamento de memória, muito utilizada para fazer laços de repetição, recursividade, entre outros.

Formato para escrita de código na linguagem :

Opcode	Valor do salto

Formato para escrita em código binário:

4 bits	12 bits
15-12	11 - 0
Opcode	Valor do salto

Visão geral das instruções do Processador MK-IV:

O número de bits do campo Opcode das instruções é igual a quatro, sendo assim obtemos um total ($Bit\{0\,e\,1\}^4 \therefore 2^4 = 16$) de 16 Opcodes **(0000 - 1111)** que são distribuídos entre as instruções, assim como é apresentado na Tabela 1.

Opcode	Nome	Formato	Breve Descrição	Exemplo
0000	Operação tipo R	R	Operações de soma que serão estabelecidas pelo campo funct	**add** $s2, $s0,$s1 **sub** $s2, $s0,$s1 **mult** $s2, $s0,$s1
0001	Load	I	Fazer a escrita de valores em um registrador	**lw** $s1, ($s2)30;
0010	Store	I	Armazenar valores na memória	sw $s2, ($s1)30;
0011	BEQ	I	Verificar dois valores e caso verdadeiro realizar um local de memória indicado	beq $s1,$s2,L;
0100	BNE	I	Verificar dois valores e caso falso realizar um local de memória indicado	bne $s1,$s2,L;
0101	Jump	J	Realizar um salto nas instruções	j 1000;

Tabela 1 – Tabela com a lista de OPCODES utilizadas pelo processador MK-IV

1.3 Descrição do Hardware

Nesta seção são descritos os componentes do hardware que compõem o processador MK-IV, incluindo uma descrição de suas funcionalidades, valores de entrada e saída.

1.3.1 PC

PC ou Program Counter é o componente responsável por armazenar o endereço da instrução que será realizada, no processador ele foi construído como sendo um registrador flip flop do tipo d, a necessidade do mesmo ser flip flop foi para ter certeza que o endereço da instrução não será perdido e assim ter um pouco mais de certeza que o processador poderá ler um código por completo sem haver perda de endereços de informação. O MK-IV possui a integração do pc conter com um contador síncrono de 3 bits para evitar problemas com lixo de memória.

No PC existem duas entradas, sendo uma para o clock, que irá definir o tempo de execução e a entrada do endereço de memória da instrução, também existe uma saída que será que irá entrar na memória de instrução relativo ao endereço de memória da instrução que será executada.

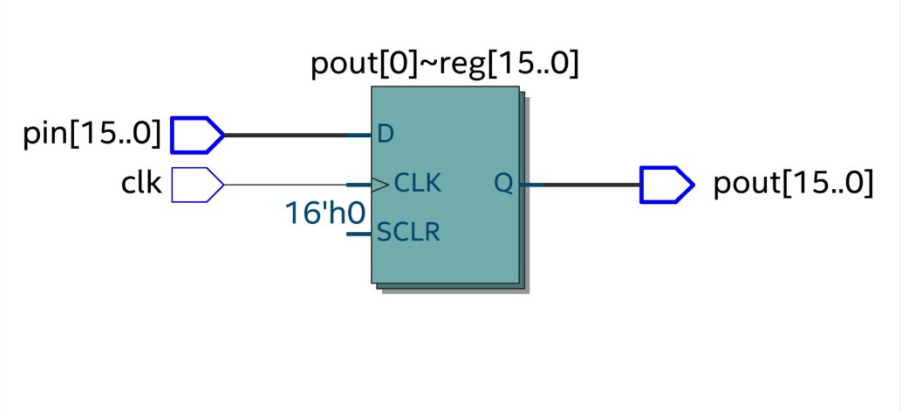

Figura 2 – PC RTL View

1.3.2 Memoria de instruções

A memória de instruções é o segundo passo da execução de uma instrução, na memória de instruções é onde está armazenada as instruções do programa que será executado, no MK-IV ela foi construída como uma memória ROM de 16 bits podendo endereçar até 65536 endereços ($2^{16} = 65536$, todos as instruções são armazenadas em uma matriz e o endereço que entra na mesma será o endereço da instrução a ser executada.

A memória de instrução conta com uma entrada que é o endereço de memória da instrução que será executada no momento, e 8 saidas: 1- clock: Clock do sistema que

ordena os ciclos de execução; 2 – Opcode: Opcode que irá para a unidade de controle para que se possam ser ativadas as respectivas flags necessárias da instrução; 3 – rd: Primeiro registrador que será utilizado na instrução; 4 – rt: Segundo registrador que será usado na instrução; 5 – rs: Registrador de destino onde será armazenado os valores depois da fase de execução da instrução; 6 – funct: Campo que indica para a ULA qual será a operação que será executada no caso de uma instrução do tipo R; 7 – tipoi: Campo exclusivo das instruções do tipo I, que será o endereço do desvio, ou valor a ser armazenado; 8 – jump: Campo exclusivo da instrução do tipo J, que se refere a qual será o próximo endereço de salto

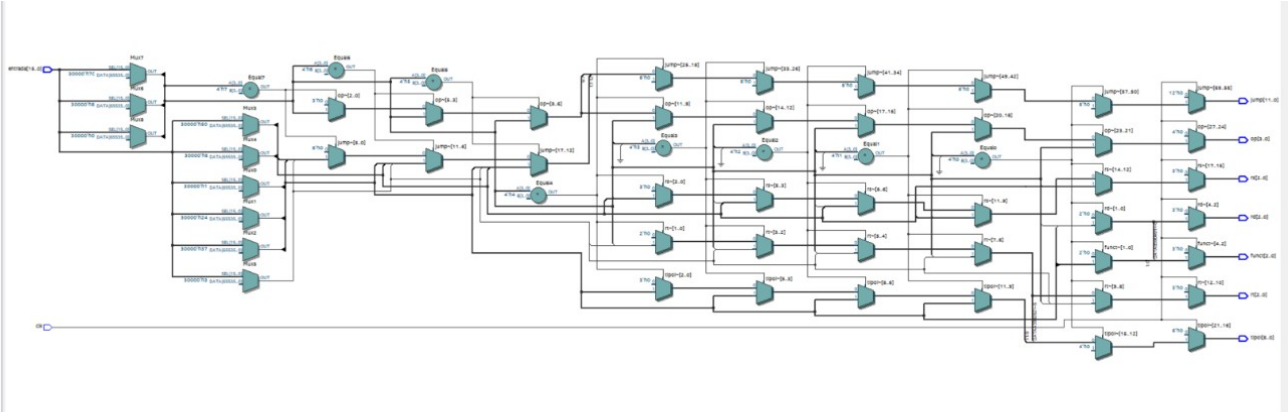

Figura 3 – Memoria de instrução RTL View

A figura da RTL View pode estar desorganizada, mas isso se deve ao fato da forma na qual o código foi descrito, o código usa a estrutura de decisão if para fazer a verificação da saída da memória, pois de acordo com o formato da instrução a saída será alterada, dessa forma durante o desenvolvimento foi necessário usar tal estrutura como o if, dessa forma se resulta com uma visualização não tão amigável, contudo tudo que se deseja saber pode ser extraído da mesma.

1.3.3 Banco de Registradores

Registradores são as estruturas auxiliares do processador, seu funcionamento pode ser comparado a de uma variável, no mesmo será utilizado como intermediador dos valores que serão utilizados no código.

No MK-IV foram utilizados 8 registradores, que foi definido segundo o formato da instrução do mesmo, o componente que irá armazenar todos esses registradores é conhecido como banco de registradores, seu comportamento será definido de acordo com as entradas que o mesmo receber, o banco de registradores foi desenvolvido com 4 entradas, sendo 3 delas valores binários relativos aos registradores que serão utilizados na instrução que está sendo executada, e uma entrada sendo uma flag da unidade de controle. O Banco também possui duas saídas que são referentes aos dados para serem enviados a ULA para serem operados;

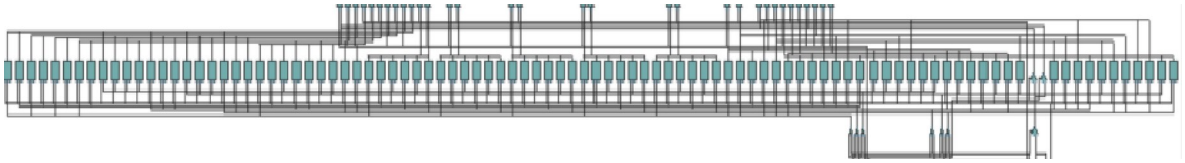

Figura 4 – Banco de Registradores RTL View

1.3.4 ULA

A Unidade Lógica e Aritmética é o componente que realiza as operações dentro do processador, sua função incorpora realizar somas, subtrações, multiplicações, calcular endereços de desvios, entre outras.

A ULA realiza as seguintes operações, soma, subtração, multiplicação, and e or, para realizar as operações soma e subtração foi utilizado a biblioteca arith do VHDL que permite que as operações sejam realizadas já tratando overflow, a multiplicação foi aplicada através do algoritmo de Booth, no qual realiza a operação com duas entradas de 8 bits resultando em uma saída de 16 bits, já que o processador não é provido dos registradores High e Low para tratamento de multiplicação.

A ULA possui 3 entradas e 3 saídas, sendo que as entradas são: 1 – Entrada1: Dado 1 saído do banco de registradores; 2 – Entrada2: Dado 2 saído do banco de registradores; 3 - Flag que indica a operação que ocorrerá na ULA, as saídas são: 1- Saída para dados: que é a saída para a memória de dados; 2- Saída de dados para mux:

Saida para o multiplexador que de acordo com a flag do mesmo irá definir o que será enviado para o write back; 3 – ZERO: Esse valor será usado para caso seja uma instrução de jump para fazer a escrita do PC.

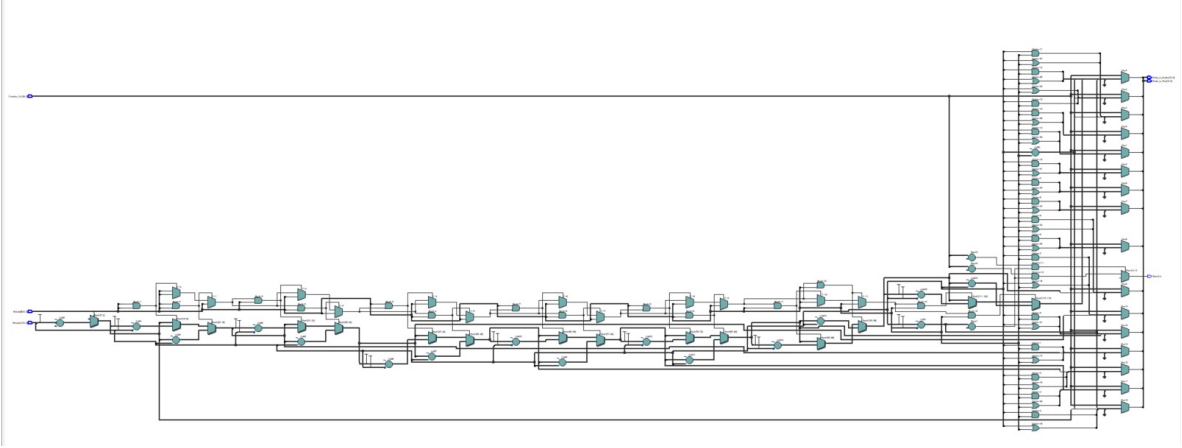

Figura 5 – ULA RTL View

1.3.5 Memória de Dados

A memória de dados é o componente que guarda alguns valores durante a execução do programa, seu comportamento pode ser definido como leitura e escrita, ela pode não só somente ler um valor que já foi armazenado e manda-lo para o resto do circuito como fazer com que dados sejam armazenados dentro da mesma.

Seguindo seu comportamento a memória de dados foi descrita como uma memória RAM, na qual só foi endereçado 16 espaços de memória, e não 65536 como esperado e isso se deve a limitações do ambiente de desenvolvimento Quartus, pois com todo processador unido já haviam sido endereçados 65536 endereços de memória e com mais 65536 endereços seria muita memória e para manter a saúde do computador no qual está sendo desenvolvido o Quartus veta esse tipo de declaração, logo definimos que apenas 16 endereços seriam mais do que o suficiente para o MK-IV, por isso foram definidos esses valores.

A Memória de Dados conta com 5 entradas e 1 saída, sendo as entradas: 1 – Endereço: É o valor que sai da ULA relativo ao endereço a ser acessado da memória; 2 – Clock: Clock do sistema que rege o funcionamento dos componentes; 3 – Entrada: É o dado 2 que sai do Banco de Registradores em caso de instruções do tipo load e store; 4 –

rd: Flag que indica se será feito uma escrita na memória; 5 – wd: Flag que indica se será feito uma leitura na memória, e a saída é dado que foi solicitado.

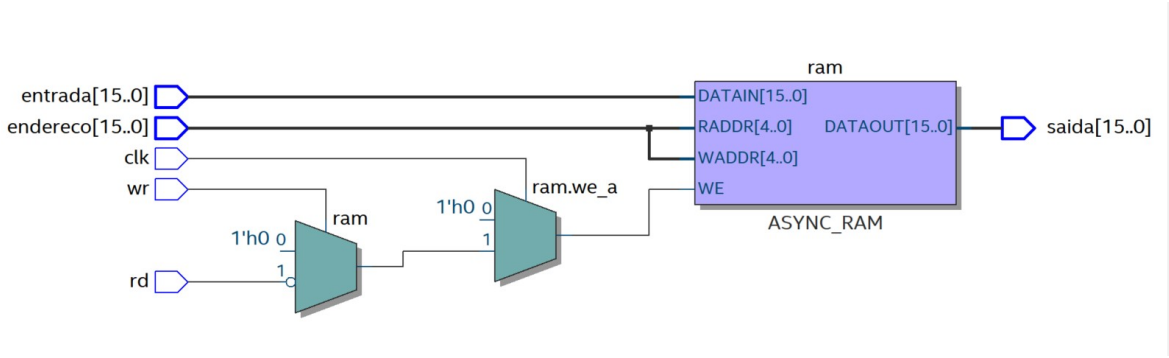

Figura 6 – Memória de dados RTL View

1.3.6 Unidade de Controle

A unidade de Controle é um dos componentes mais importantes do processador, pois a mesma orquestra como a instrução vai ser realizada, quais componentes devem ser ativados, quais não devem ser ativados.

No MK-IV a Unidade de Controle conta com 9 flags:

FLAG	FUNÇÃO
REGDEST	Essa flag indica que para um multiplexador que se encarrega de indicar se irá ter um registrador para fazer a escrita de dados (1 bit).
ORIGALU	Essa flag vai para a ULA indicando qual a operação que será realizada, ou seja, indicando o campo funct no caso de uma instrução do tipo R (4 bits).
MEMPARAREG	Essa flag vai para um multiplexador que irá definir qual o valor será devolvido para o registrador de destino, o resultado da ULA ou da Memória de Dados
ESCREVEREG	Essa flag indica que o registrador para escrita será utilizado.
LEMEM	Flag que indica para a Memória de Dados que será feito uma

	escrita na mesma.
BRANCH	Flag que indica para a porta AND e que é operado com a saída ZERO da ULA se a instrução será um branch;
ALUSRC	Flag que indica para um multiplexador qual a entrada que irá para a ULA, se será a saída do Banco de Registradores ou os bits 5:0 da instrução
JUMP	Flag que indica que um desvio será realizado.

Tabela 2 – Flags Unidade de Controle

Além dessas saídas a Unidade de Controle possui duas entradas sendo uma o clock do sistema e outra o opcode da instrução.

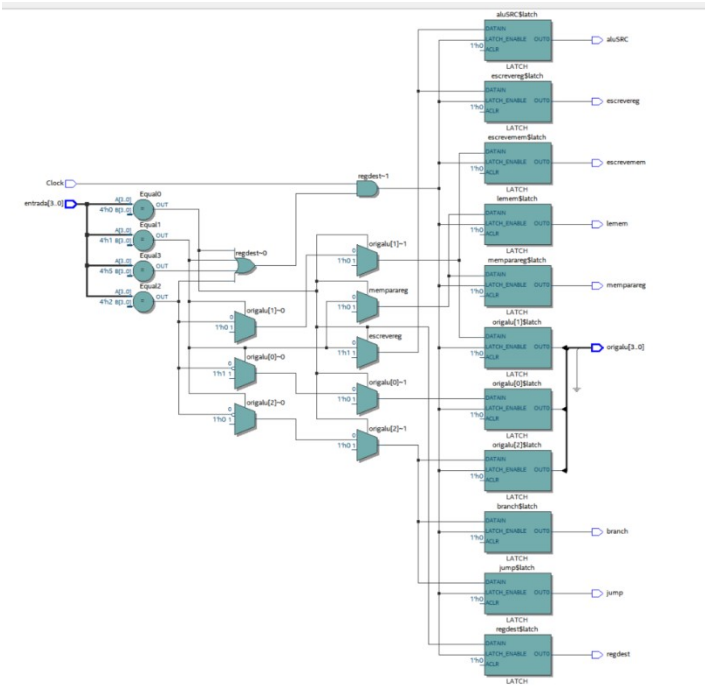

Figura 7 – Unidade de Controle RTL View

1.3.7 Somador PC

Componente que possui a responsabilidade de computar o endereço na Memória de Instruções da próxima instrução a ser utilizada, para o MIPS seria utilizado que uma word possui 2 bytes (16 bits), porém o funcionamento do MK-IV é diferente, nele as instruções estão armazenadas em uma matriz, logo o PC deve guardar a posição na matriz da instrução, então o somador irá se comportar somando mais 1 para cada execução que será o endereço da próxima instrução.

O Somador PC conta com uma entrada, o PC atual e tem como saída o PC + 1.

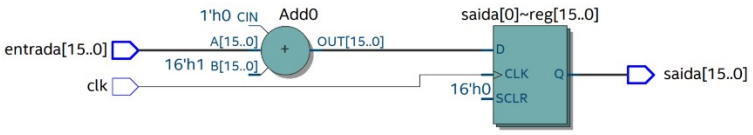

Figura 8 – Somador PC RTL View

1.3.8 Extensor de Sinal

Extensor de sinal é o componente no qual é relativo ao desvio no caso de uma operação do estilo branch,o processador compreende os bits 5:0 desse tipo de instrução, como todo o processador foi pensado para ser trabalhado com 16 bits, com 6 não teríamos resultados satisfatórios, por isso se pega os 6 bits e se concatena com 10 bits ‘0’ a frente, para que não haja alteração do valor proposto e assim pode se continuar a execução da instrução sem problemas.

No MK-IV o Extensor de sinal possui duas entradas que são os 6 bits e o clock do sistema, e uma saída que são os 16 bits para o salto.

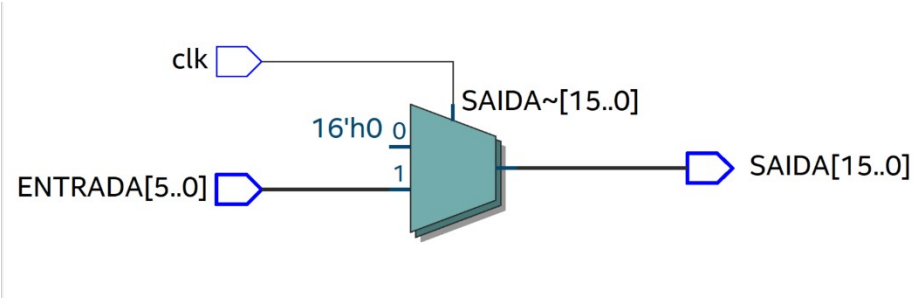

Figura 9 – Extensor de Sinal RTL View

1.3.9 Shift à esquerda

Esse componente basicamente trata a sua entrada e tira seus dois bits mais significativos e os repõe com 0 para que o número de bits seja o mesmo.

O processador possui dois tipos de Shifter, sendo um normal e um para jump que apenas concatena os bits a frente e depois concatena novamente com os 2 bits mais significativos do PC atual e isso seguirá para se calcular o endereço do jump.

Esses componentes contam com 2 entradas, sendo uma o clock do sistema e uma a entrada que será operada e contam com uma saída com o valor operado, o shift para jump possui mais uma entrada que é o PC para que seja concatenado.

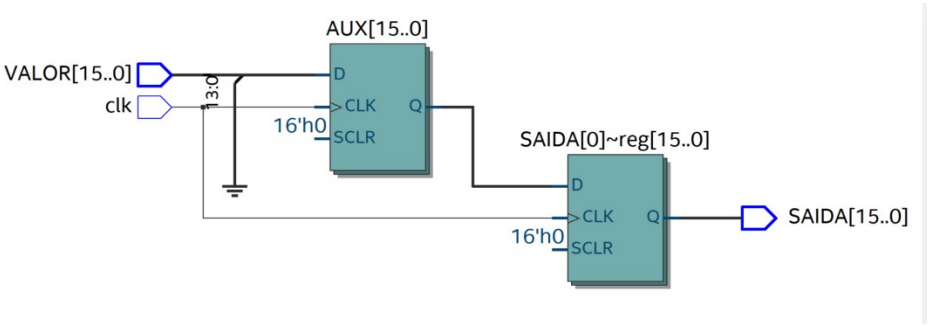

Figura 10 – Shifter de 2 bits a esquerda RTL View

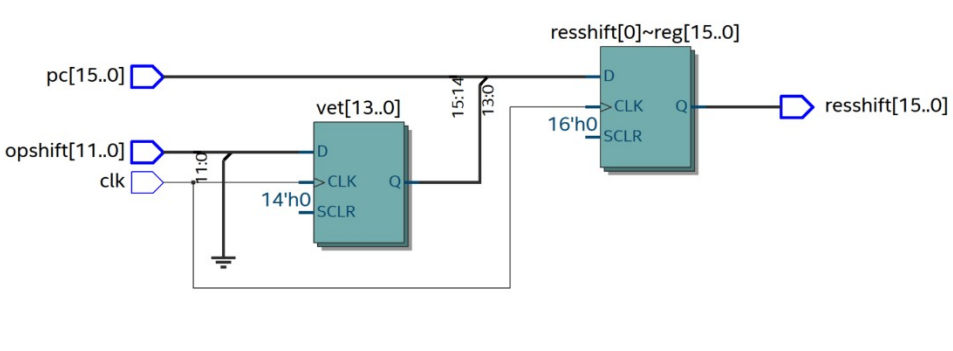

Figura 11 – Shifter de 2 bits a esquerda e concatena com PC RTL View

1.3.10 Multiplexadores

Multiplexadores são componentes que possuem a função de escolha de trilhas, dependendo das entradas e da flag que o mesmo receber a saída apropriada para a operação será obtida.

No processador são utilizados 5 multiplexadores:

1 – Multiplexador Memoria de Dados – Banco de Registradores: Esse multiplexador irá receber uma flag da Unidade de Controle que indicará qual o será o registrador de escrita de dados;

2 – Multiplexador Banco de Registradores – ULA: Esse multiplexador receberá uma flag da unidade de controle que irá indicar para a ULA qual vai ser a segunda entrada, pois a mesma irá variar para cada tipo de instrução;

3 – Multiplexador Memoria de Dados – Banco de Registradores: Esse multiplexador de acordo com a flag da Unidade de Controle irá definir qual será o valor que será inscrito no registrador de destino;

4 – Multiplexador Resultado da ULA – Multiplexador: Esse multiplexador está incumbido de indicar para o próximo multiplexador qual será utilizado o PC da próxima instrução ou se será o endereço calculado do branch;

5 – Multiplexador Multiplexador – PC: Esse multiplexador definirá qual será o próximo PC, se será o PC + 1, ou o endereço de desvio calculado.

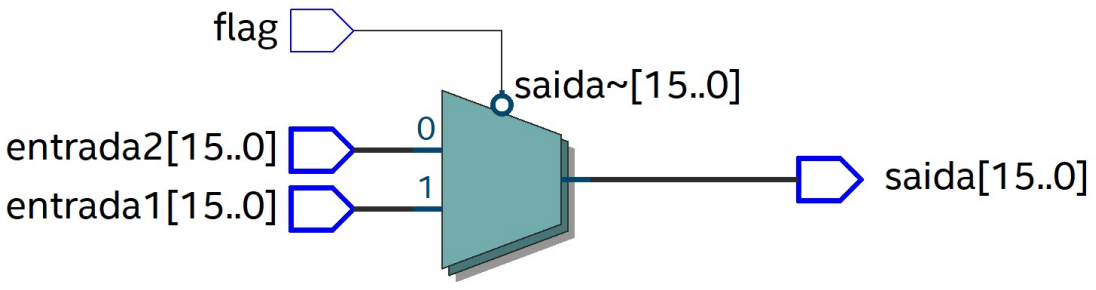

Figura 12 – Multiplexador RTL View

1.3.11 Somador

Esse componente é utilizado em caso de instruções do tipo branch, para calcular o valor do desvio, esse componente no recebe como entrada o PC + 1, e os 6 bits menos significativos da instrução estendido para 16 bits, e a saída será o valor do endereço de memória do branch.

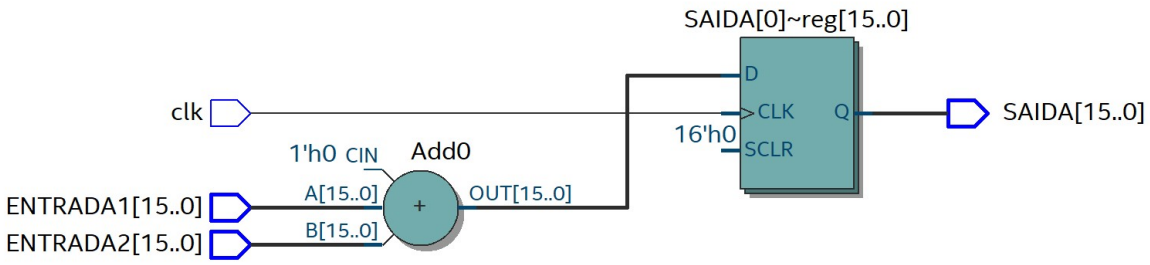

Figura 13 – Somador RTL View

1.3.12 Clock

O clock no é a estrutura que irá comandar as operações, como já pode ser observado a maioria dos componentes recebem uma entrada sendo clock e seu comportamento será regido pelo mesmo, esse controle é estabelecido por motivos de melhor controle de como as operações serão realizadas e também através do clock pode se aferir a eficiência do processador, assim calculando a necessidade de quantos ciclos para cada tipo de instrução.

1.4 Datapath

O Datapath pode ser traduzido como caminho de dados, que seria todos os componentes já citados sendo unidos para que se possa haver o processamento da instrução desejada.

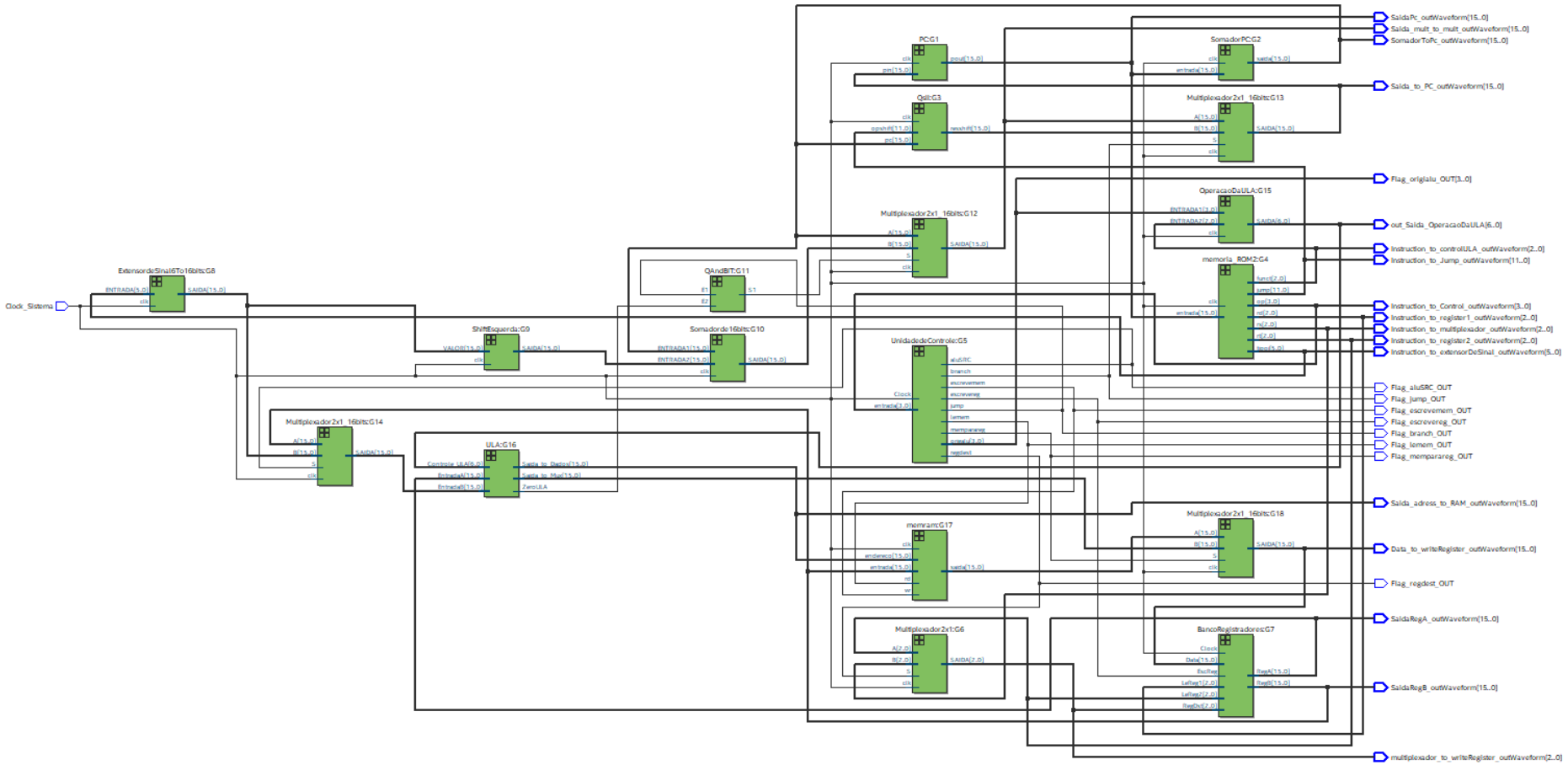

Figura 14 – MAGIC Datapath RTL View

2 Simulações e Testes

Objetivando analisar e verificar o funcionamento do processador, efetuamos alguns testes analisando cada componente do processador em específico, em seguida efetuamos testes de cada instrução que o processador implementa. Para demonstrar o funcionamento do processador utilizaremos como exemplo um código simples que realiza as operações básicas para verificar se todos os componentes funcionam juntos.

Endereço	Linguagem de Alto Nível	Binário				
0	**MULT** $s3, $s0, $s1	0000	011	000	001	0101

Tabela 2 – Exemplo 1 – multiplicação de dois valores.

Para essa instrução já havia sido setado o valor de 7 para o registrador $s0 e 5 para o registrador $s1, e quando se faz a multiplicação de ambos a saída da ULA é 35, e também o valor foi guardado no registrador $s3.

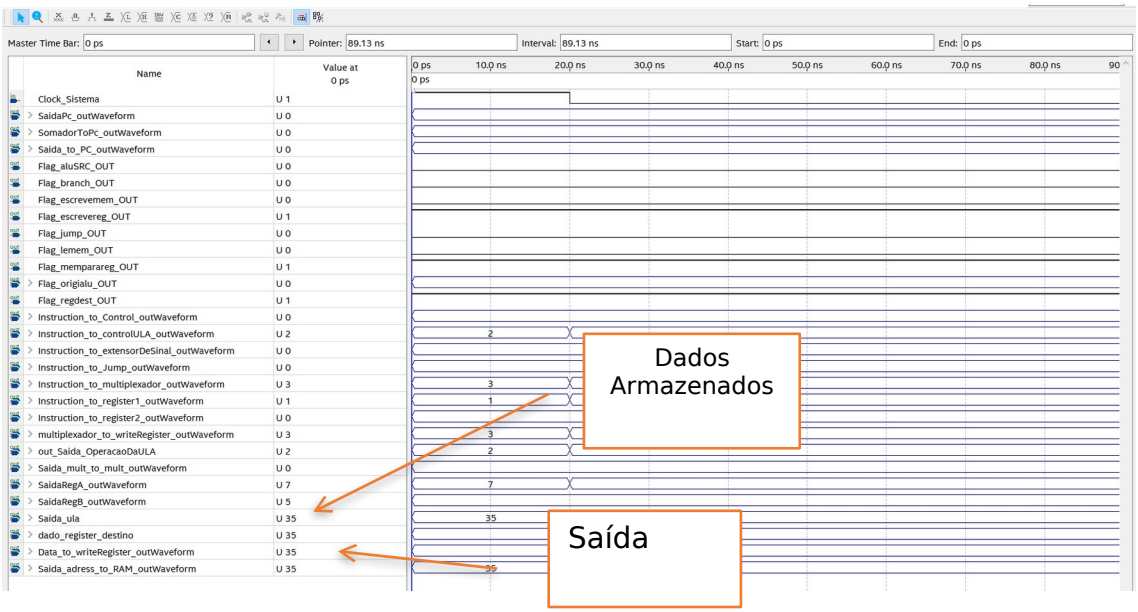

Figura 14 - Resultado na waveform exemplo 1

Endereço	Linguagem de Alto Nível	Binário				
0	**add** $s3, $s0, $s1	0000	100	000	101	0000

Tabela 4 – Exemplo 2 – soma de dois valores.

Para essa instrução já havia sido setado o valor de 13824 para o registrador $s0 e 11992 para o registrador $s3, e quando se faz a soma de ambos a saída da ULA é 25816, e também o valor foi guardado no registrador $s4.

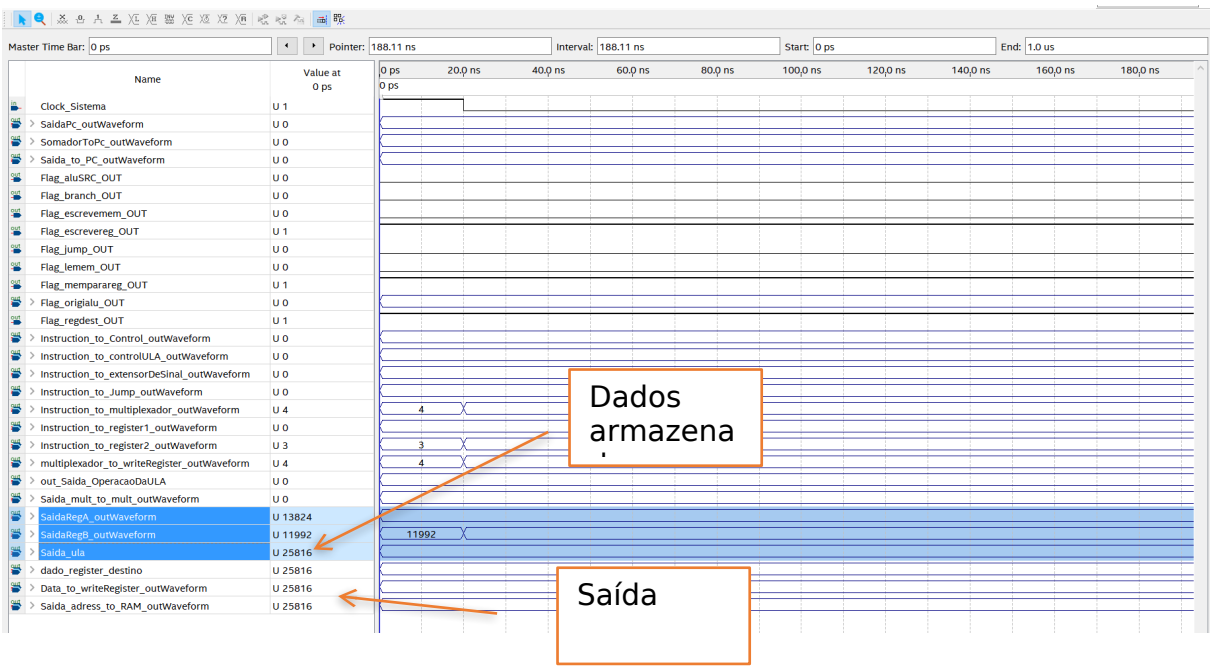

Figura 16 - Resultado na waveform exemplo 2

Endereço	Linguagem de Alto Nível	Binário				
0	**sub** $s5, $s2, $s3	0000	101	010	101	0001

Tabela 5 – Exemplo 3 – subtração de dois valores.

Para essa instrução já havia sido setado o valor de 43705 para o registrador $s2 e 65535 para o registrador $s3, e quando se faz a soma de ambos a saída da ULA é -21830, e também o valor foi guardado no registrador $s5.

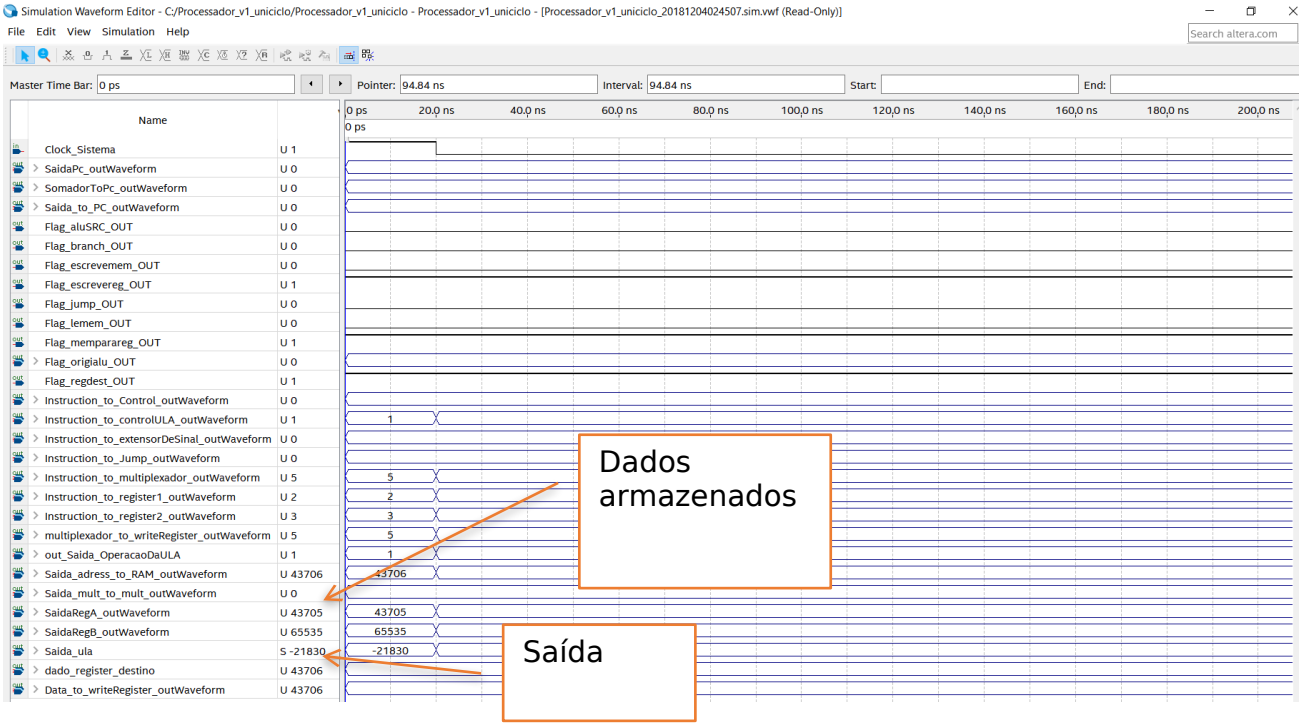

Figura 17 - Resultado na waveform exemplo 3

Endereço	Linguagem de Alto Nível	Binário			
0	**lw** $s0, (1) $s4	0001	100	000	000001

Tabela 6 – Exemplo 4 – Load de um valor.

Para essa instrução será realizado o load do valor de $s4 + 1 para que seja armazenado no registrador $s0.

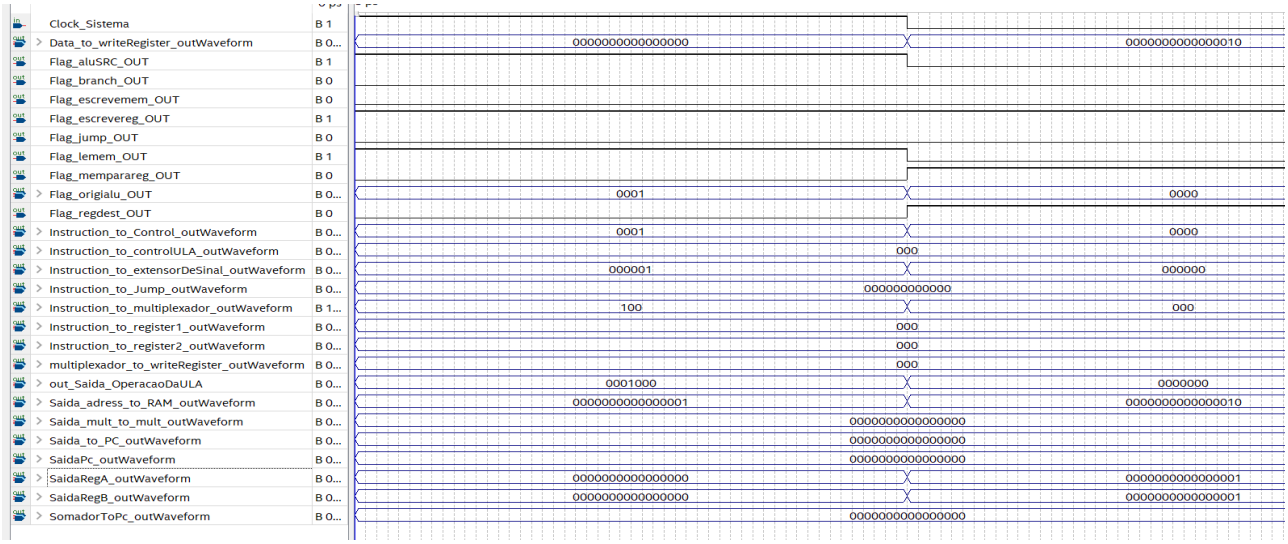

Figura 18 - Resultado na waveform exemplo 4

Endereço	Linguagem de Alto Nível	Binário	
0	**j** 1000	0101	001111101000

Tabela 7 – Exemplo 5 – jump.

Para essa instrução será realizado um jump para a posição 1000 na memória.

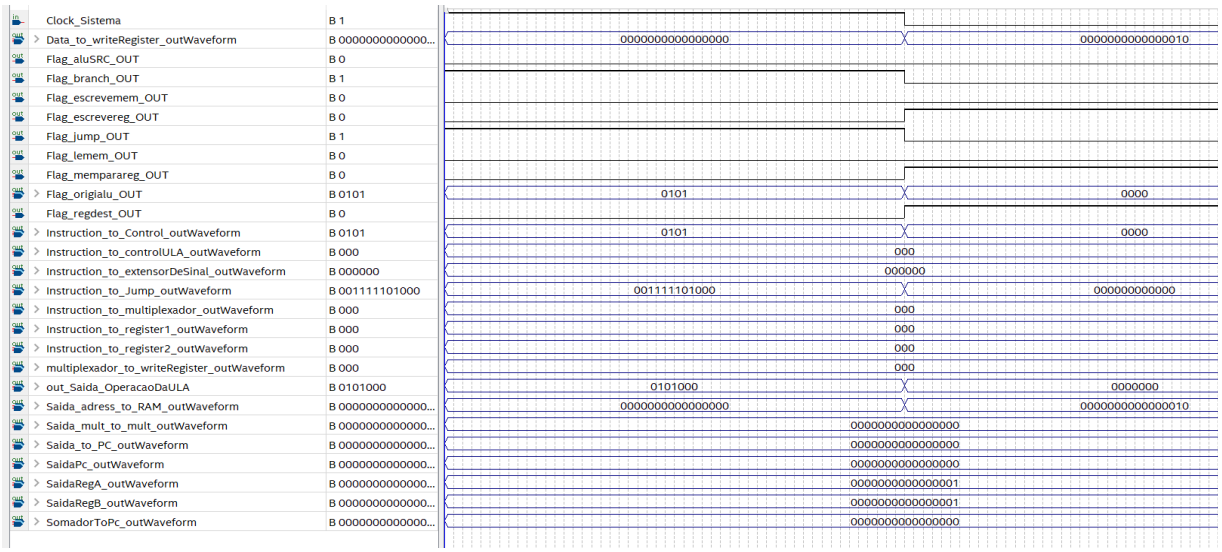

Figura 19 - Resultado na waveform exemplo 5

3 Considerações finais

O processador MK-IV comprovou a necessidade de se ter um bom conhecimento de hardware, para se saber como o computador irá se comportar para cada programa que for descrito, pois com um mal-uso, se terá um mal aproveitamento do componente, que não é interessante para nenhum arquiteto de hardware.