ipPROCESS

Documento de Arquitetura

Core-MUSA

Universidade Estadual de Feira de Santana

Build 2.0a

Histórico de Revisões

Date	Descrição	Autor(s)
20/10/2014	Concepção do documento	fmbboaventura
23/10/2014	Revisão inicial	jadsonfirmo
29/10/2014	Foi adicionada uma breve descrição dos componentes	fmbboaventura
29/10/2014	*Stakeholders*	jadsonfirmo
30/10/2014	Ajustes estruturais	fmbboaventura e jadsonfirmo
30/10/2014	Detalhamento das instruções	jadsonfirmo, KelCarmo e Odivio
30/10/2014	Adição dos diagramas de classe	gordinh
06/11/2014	Mudanças nos *layouts* das instruções e no diagrama de classe da ULA	jadsonfirmo e KelCarmo
10/11/2014	Detalhamento dos opcodes	jadsonfirmo
13/11/2014	Alteração na tabela dos opcodes	jadsonfirmo
17/11/2014	Alteração no diagrama de classe da ULA e ajustes na tabela de opcodes	jadsonfirmo
18/11/2014	Ajuste dos opcodes	jadsonfirmo
24/11/2014	Refatoração do *Datapath*	Odivio Caio
07/12/2014	Opcodes de acordo com o *software* Vênus	jadsonfirmo
07/12/2014	Alteração no *layout* do JR e remoção de conflitos	jadsonfirmo
09/12/2014	Correções estruturais do documento	di3goleite
11/12/2014	Revisão	jadsonfirmo
continua na próxima página		

continuação da página anterior		
Date	**Descrição**	**Autor(s)**
12/12/2014	Correção de bug na tabela de Busca de Registradores	di3goleite
15/12/2014	Correção do datapath no documento e da tabela de histórico de revisões	di3goleite

SUMÁRIO

1 Introdução **5**

1 Propósito do Documento 5
2 Stakeholders 5
3 Visão Geral do Documento 5
4 Definições 6
5 Acrônimos e Abreviações 6

2 Visão Geral da Arquitetura **7**

1 Descrição dos Componentes 7
2 Intruções 7
3 Detalhamento das Instruções 9
4 DataPath interno 11

3 Descrição da Arquitetura **12**

1 ULA 12
1.1 Diagrama de Classe 12
1.2 Definições de Entrada e Saída 12
2 Busca de Instrução 13
2.1 Diagrama de Classe 13
2.2 Definições de Entrada e Saída 13
3 Pilha 13
3.1 Diagrama de Classe 13
3.2 Definições de Entrada e Saída 13
4 Acesso à memória 14

4.1 Diagrama de Classe 14
4.2 Definições de Entrada e Saída 14
5 Busca de Registradores 15
5.1 Diagrama de Classe 15
5.2 Definições de Entrada e Saída 15

1 | Introdução

1. Propósito do Documento

Este documento descreve a arquitetura do projeto Core-MUSA, incluindo as especificações dos circuitos internos, bem como suas devidas máquinas de estados. Também, serão apresentados diagramas de classes, de temporização e definições de entradas e saídas. O principal objetivo deste documento é definir as especificações de arquitetura do Core-MUSAe provê uma visão geral do projeto.

2. Stakeholders

Nome	Papel/Responsabilidades
Diego Leite e Lucas Morais	Gerência
Victor Figueiredo, Matheus Castro, Odivio Caio Santos e Kelvin Carmo	Desenvolvimento
Filipe Boaventura e Wagner Bittencourt	Implementação
Jadson Firmo	Análise e refatoração

3. Visão Geral do Documento

Este documento é dividido através das seguintes seções:

- **Capítulo 2** – Nesta seção será apresentada a visão geral da arquitetura do projeto.
- **Capítulo 3** – Neste capítulo encontram-se informações detalhadas sobre módulos e componentes do Core-MUSA, relacionados com sua arquitetura.

4. Definições

Termo	Descrição
Opcode	Código de operação da instrução
Function	Código de operação aplicado a operações que ocorrem dentro da Unidade Lógica e Aritmética (ULA)
Datapath	Caminho de dados percorrido para a execução de uma instrução

5. Acrônimos e Abreviações

Sigla	Descrição
PC	Contador de Programa (*Program Counter*)
ULA	Unidade Lógica e Aritmética
Bit	*Binary Digit* (Digito Binário)
Byte	*Binary Term* (Conjunto de 8 *bits*)

2 | Visão Geral da Arquitetura

1. Descrição dos Componentes

A unidade de processamento a ser desenvolvida é composta a partir dos seguintes componentes:

- **PC** – Registrador que guarda o endereço da próxima instrução a ser executada.
- **Memória de Dados** – A Memória de dados é endereçada com 33 *bits*, que comporta no máximo 2 elevado a 33 palavras de instrução no total, que guarda dados com tamanho de 32 *bits* que foram manipulados pelo programa ou processador.
- **Memória de Instrução** – A Memória de instrução é endereçada com 18 *bits*, que comporta no máximo 2 elevado a 18 palavras de instrução no total, tem tamanho de 1048576 *bytes*, justamente pelo fato da palavra de instrução ter 32 *bits*. Por fim é a memória que guarda o programa codificado em linguagem *Assembly*.
- **ULA** – É responsável por todo o processamento realizado no processador, pois esta unidade executa as instruções lógicas e aritméticas.
- **Unidade de Controle** – Esta unidade decodifica a instrução e define sinais de controle como, sinais de leitura, escrita de memória e de registradores de armazenamento temporário interno e sinais de liberação de barramentos para endereço e dados e unidades funcionais. As unidades funcionais internas do processador são controladas por esta unidade de temporização e controle. Os determinados sinais de controle são enviados para as demais unidades após a decodificação de uma determinada instrução que partem do registrador de instrução (IR).
- **Banco de Registradores** – Contém os 32 registradores de propósito geral do processador.
- **Pilha** – Memória destinada para armazenamento dos endereços de retorno de chamadas de funções. Possui 32 registradores de 18 *bits* e um contador responsável por apontar o topo da pilha.

2. Intruções

A unidade de processamento possui 21 instruções essenciais para o processamento das operações. Elas são desmembradas em três formatos: Tipo R (Registradores), Tipo I (Imediatas), e Tipo J (*Jump, ou Desvio*).

- **Tipo R** – Operações lógicas e aritméticas.

INSTRUÇÃO	DESCRIÇÃO	OPCODE	FUNCTION
ADD	Soma de dois valores.	000000	100000
SUB	Subtração de dois valores.	000000	100010
MUL	Multiplicação de dois valores.	000000	011000
DIV	Divisão de dois valores.	000000	011010
AND	Operação lógica AND entre dois valores.	000000	100100
OR	Operação lógica OR entre dois valores.	000000	100101
NOT	Operação lógica NOT.	000000	100111
CMP	Comparação de dois valores.	000000	011011
JR	Desvia o programa para um endereço de destino.	001000	-

Tabela 2.1: **Instruções do tipo R.**

- **Tipo I** – Operações imediatas.

INSTRUÇÃO	DESCRIÇÃO	OPCODE	FUNCTION
ADDi	Soma de dois valores, sendo um destes imediato.	001000	-
SUBi	Subtração de dois valores, sendo um destes imediato.	001001	-
ANDi	Operação lógica AND entre dois valores, sendo um destes imediato.	001100	-
ORi	Operação lógica OR entre dois valores, sendo um destes imediato.	001101	-
LW	Operação de leitura na memória de dados.	100011	-
SW	Operação de armazenamento na memória de dados.	101011	-

Tabela 2.2: **Instruções do tipo I.**

- **Tipo J** – Operações de desvio e branch.

INSTRUÇÃO	DESCRIÇÃO	OPCODE	FUNCTION
JPC	Desvia o programa para um endereço relativo ao PC.	001001	-
BRFL	Desvia o programa para um endereço de destino, atendendo uma condição de flag.	010001	-
CALL	Desvia um programa em execução para uma sub-rotina.	000011	-
RET	Retorna de uma sub-rotina.	000111	-
HALT	Para a execução de um programa.	000010	-
NOP	Não realiza operação.	000001	-

Tabela 2.3: Instruções do tipo J.

3. Detalhamento das Instruções

- **ADD, SUB, MUL, DIV, AND, OR, NOT e JR :**

OPCODE	RS	RT	RD	SHAMT	FUNCTION
06	05	05	05	05	06

Tabela 2.4: *Layout* das instruções do tipo R.

Esse conjunto de instruções utiliza dois registradores fontes (RS e RT) de dados e um registrador de destino (RD) para realizar as operações. O campo FUNCTION é utilizado como um segundo campo de código de operação, ampliando o leque de operações possíveis. O campo SHAMT é reservado para as operações imediatas.

- **LW e SW, ADDi, SUBi, ANDi, ORi e BRFL:**

OPCODE	RD	RS	IMMEDIATE
06	05	05	16

Tabela 2.5: *Layout* das Operações de Leitura/Escrita e operações imediatas

Esse conjunto de instruções utiliza, além do código de operação (OPCODE), um registrador fonte (RS), e um registrador destino (RD) para instruções de leitura

(LW) e de escrita (SW). Utiliza também do campo IMMEDIATE (I) (de 16 *bits*) que representa o deslocamento do registrador base.

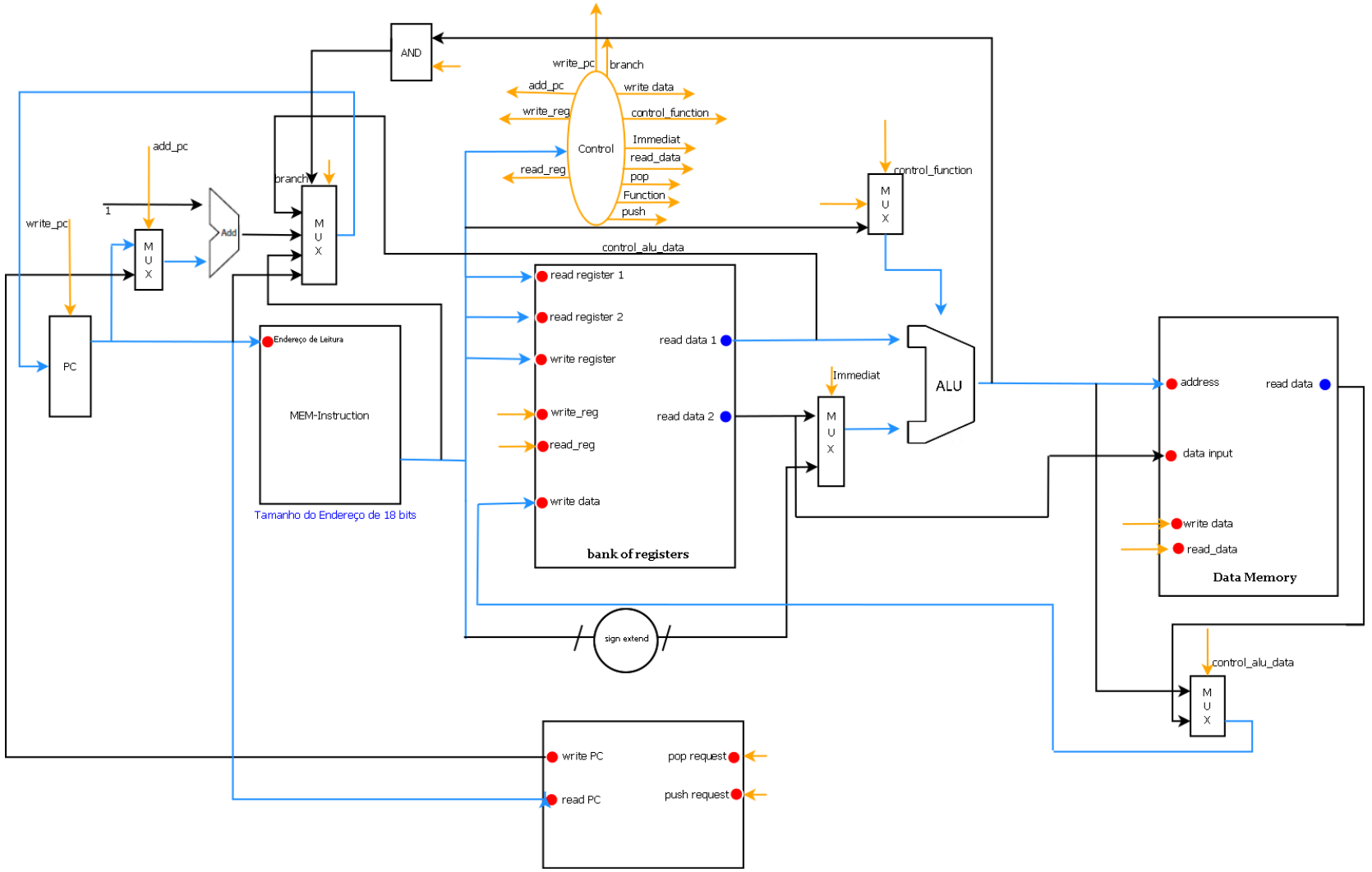
- **CALL, RET, HALT e NOP e JPC:**

OPCODE	TARGET
06	26

Tabela 2.6: Layout das Operações de Salto (Tipo J)

Instruções de salto possue, além do OPCODE, um campo de 26 *bits* representando uma posição de endereço de memória, para as instruções CALL, HALT e JPC. As instruções RET e NOP só utilizam o OPCODE da instrução.

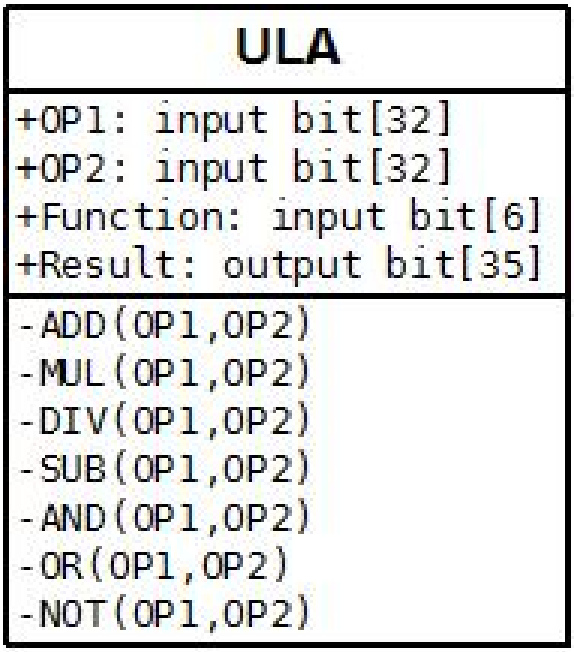
4. DataPath interno



3 | Descrição da Arquitetura

1. ULA

1.1. Diagrama de Classe



1.2. Definições de Entrada e Saída

Nome	Tamanho	Direção	Descrição
OP1	32	entrada	Primeiro operando.
OP2	32	entrada	Segundo operando.
Function	6	entrada	Identificador da operação.
Result	35	saída	Resultado da operação com as *flags*.

2. Busca de Instrução

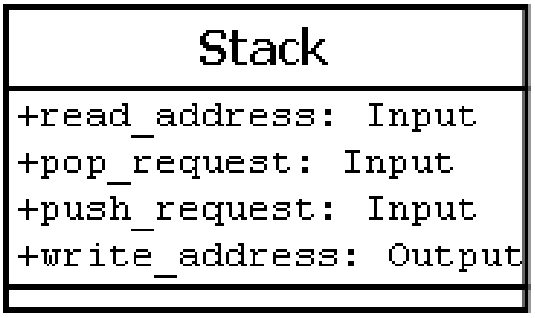
2.1. Diagrama de Classe

2.2. Definições de Entrada e Saída

Nome	Tamanho	Direção	Descrição
clock_in	1	entrada	Sinal de *clock* da fase.
entradaPC	18	entrada	Endereço do PC atual.
saidaInstrucao	32	saída	Instrução que sai da memória de instrução.

3. Pilha

3.1. Diagrama de Classe



3.2. Definições de Entrada e Saída

Nome	Tamanho	Direção	Descrição
clock_in	1	entrada	Sinal de *clock* da fase.
entradaDeDados	18	entrada	Endereço do PC que será armazenado.
popRequest	1	entrada	Sinal para tirar o ultimo endereço armmazenado da pilha.
continua na próxima página			

continuação da página anterior			
Nome	**Tamanho**	**Direção**	**Descrição**
pushRequest	1	entrada	Sinal para salvar o endereço do PC da pilha.
saidaDeDados	18	saída	Último endereço salvo na pilha para retorno do PC.

4. Acesso à memória

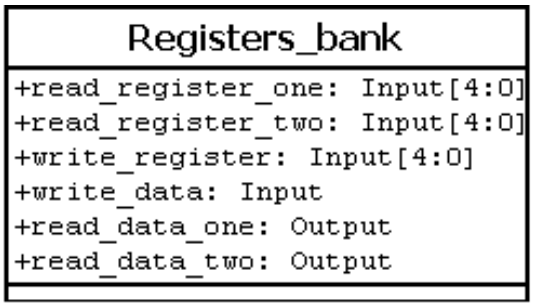
4.1. Diagrama de Classe

4.2. Definições de Entrada e Saída

Nome	Tamanho	Direção	Descrição
clock_in	1	entrada	Sinal de *clock* da fase.
entradaDeDados	32	entrada	Valor que será armazenado na memória de dados.
AtivaEscrita	1	entrada	Sinal para ativar a escrita na memória de dados.
AtivaLeitura	1	entrada	Sinal para ativar a leitura na memória de dados.
saidaDeDados	32	saída	Valor que irá sair da memória de dados.

5. Busca de Registradores

5.1. Diagrama de Classe



5.2. Definições de Entrada e Saída

Nome	Tamanho	Direção	Descrição
clock_in	1	entrada	Sinal de *clock* da fase.
write_data	32	entrada	Valor de escrita para o registrador de destino.
write_reg	1	entrada	Sinal para ativar a escrita no Banco de Registradores.
read_reg	1	entrada	Sinal para ativar a leitura no Banco de Registradores.
data1	32	saída	Saída do valor contido nos registradores.
data2	32	saída	Saída do valor contido nos registradores.
rs	5	Entrada	Endereço do registrador um.
rt	5	Entrada	Endereço do registrador dois.
rd	5	Entrada	Endereço do registrador destino.