



**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	<i>Stakeholders</i>	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 classes	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 classes da ULA e ajustes na tabela dos Opcodes	jadsonfirmo
18/11/2014	Ajustes dos Opcodes	jadsonfirmo
24/11/2014	Refatoração do DataPath	Odivio Caio
07/12/2014	Opcodes de acordo com o Vênus	jadsonfirmo
07/12/2014	Alteração Layout do JR e ajustes de conflitos	jadsonfirmo
08/12/2014	Revisão documento de arquitetura	jadsonfirmo

## SUMÁRIO

<b>1</b>	<b>Introdução</b>	<b>4</b>
1	Propósito do Documento . . . . .	4
2	Stakeholders . . . . .	4
3	Visão Geral do Documento . . . . .	4
4	Definições . . . . .	4
5	Acrônimos e Abreviações . . . . .	5
<b>2</b>	<b>Visão Geral da Arquitetura</b>	<b>6</b>
1	Arquitetura geral MUSA . . . . .	6
2	Descrição dos Componentes . . . . .	6
3	Intruções . . . . .	7
4	Detalhamento das Intruções . . . . .	9
<b>3</b>	<b>Descrição da Arquitetura</b>	<b>11</b>
1	ULA . . . . .	11
1.1	Diagrama de Classe . . . . .	11
1.2	Definições de Entrada e Saída . . . . .	11
2	Busca de Instrução . . . . .	12
2.1	Diagrama de Classe . . . . .	12
2.2	Definições de Entrada e Saída . . . . .	12
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 . . . . .	14
5.1	Diagrama de Classe . . . . .	14
5.2	Definições de Entrada e Saída . . . . .	15

# 1 | Introdução

## 1. Propósito do Documento

Este documento descreve a arquitetura do projeto Core-MUSA, incluindo especificações dos circuitos internos e máquinas de estados de cada componente. Ele também apresenta diagramas de classe, definições de entrada e saída e diagramas de temporização. O principal objetivo deste documento é definir as especificações do projeto Core-MUSA e prover uma visão geral completa do mesmo.

## 2. Stakeholders

Nome	Papel/Responsabilidades
Diego Leite e Lucas Moraes	Gerencia
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

O presente documento é apresentado como segue:

- **Capítulo 2** – Este capítulo apresenta uma visão geral da arquitetura, com foco em entrada e saída do sistema e arquitetura geral do mesmo.
- **Capítulo 3** – Este capítulo apresenta a descrição detalhada da arquitetura bem como seus módulos e componentes.

## 4. Definições

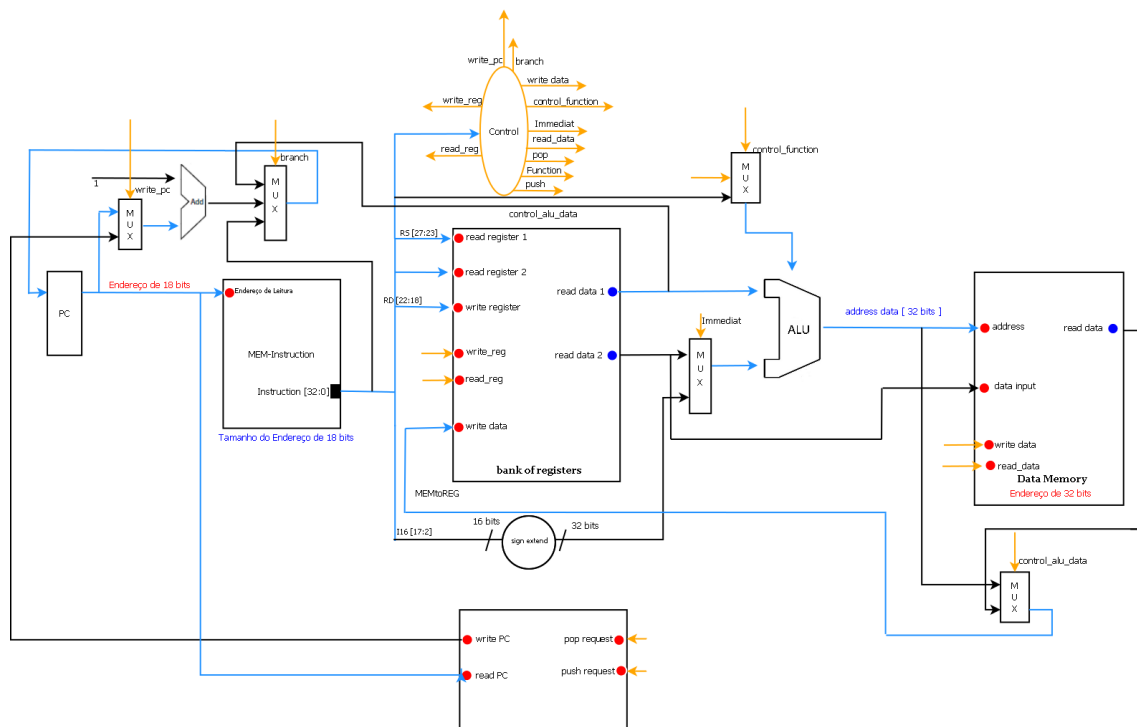
Termo	Descrição

## 5. Acrônimos e Abreviações

Sigla	Descrição
PC	Contador de Programa (Program Counter)
ULA	Unidade Lógica e Aritmética
OPCODE	Código da Operação

## 2 | Visão Geral da Arquitetura

### 1. Arquitetura geral MUSA



### 2. 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  $2^33$  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  $2^{18}$  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.

### 3. Instruções

A unidade de processamento possui 21 instruções essenciais para o processamento das operações. Elas são desmembradas em quatro formatos: R-type, I-type, Load/Store e Jump.

- **R-type** – 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.**



- **I-type** – 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	-

**Tabela 2.2: Instruções do tipo I.**

- **Load/Store** – Operações de carregamento e armazenamento (Instruções de tipo I).

INSTRUÇÃO	DESCRIÇÃO	OPCODE	FUNCTION
LW	Operação de leitura na memória de dados.	100011	-
SW	Operação de armazenamento na memória de dados.	101011	-

**Tabela 2.3: Instruções de Load/Store.**

- **Jump** – 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 2.4: Instruções de Load/Store.**

#### 4. 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 2.5: 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 IM é reservado para as operações imediatas.

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

OPCODE	RD	RS	IMMEDIATE
06	05	05	16

**Tabela 2.6: 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 I (de 16 bits) que representa

o deslocamento do registrador base. Os dois bits restantes serão sempre ignorados.

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

OPCODE	TARGET
06	26

**Tabela 2.7: Layout das Operações de Salto**

Instruções de salto, que especificam um registrador RF e uma CST (FLAG), para a instrução BRFL, que realiza um salto caso condição de comparação com a flag for verdadeira. Utilizará também 18 bits utilizará 18 bits na instrução que respresenta uma posição de endereço de memória, para as instruções JR, CALL e HALT. As instruções RET e NOP só utilizam o OPCODE da instrução.

## 3 | Descrição da Arquitetura

### 1. ULA

#### 1.1. Diagrama de Classe

ULA
+OP1: input bit[32] +OP2: input bit[32] +Function: input bit[3] +Result: output bit[35]
- ADD(OP1,OP2) - MUL(OP1,OP2) - DIV(OP1,OP2) - SUB(OP1,OP2) - AND(OP1,OP2) - OR(OP1,OP2) - NOT(OP1,OP2)

#### 1.2. Definições de Entrada e Saída

Nome	Tamanho	Direção	Descrição
OP1	32	entrada	Primeiro operando.
OP2	32	entrada	Segundo operando.
Function	3	entrada	Identificador da operação.
continua na próxima página			

continuação da página anterior			
Nome	Tamanho	Direção	Descrição
Result	35	saída	Resultado da operação com as <i>flags</i> .

## 2. Busca de Instrução

### 2.1. Diagrama de Classe

Busca da instrução
+entradaPC: input bit[18]
+saidaInstrucao: output bit[32]
+clock: input bit
-atualizacaoPC()
-interpretarEndereco(PC)

### 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

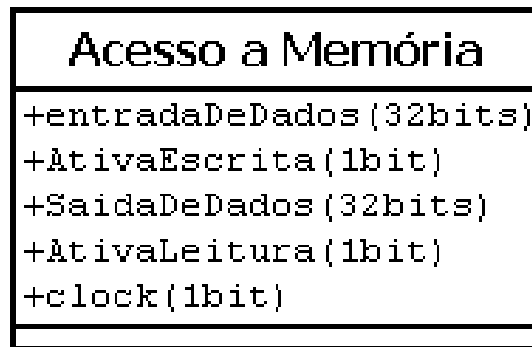


#### 3.2. Definições de Entrada e Saída

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 armazenado da pilha.
pushRequest	1	entrada	Sinal para salvar o endereço do PC da pilha.
saidaDeDados	32	saída	Ultimo endereço salvo na pilha para retorno do PC.

## 4. Acesso à memória

### 4.1. Diagrama de Classe

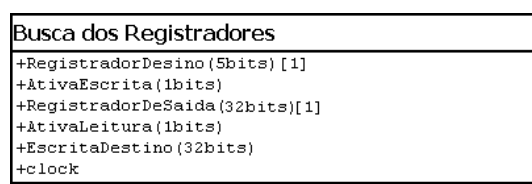


### 4.2. Definições de Entrada e Saída

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.
EscritaDestino	32	entrada	Escrita do valor nos registradores de destino.
AtivaEscrita	1	entrada	Sinal para ativar a escrita no Banco de Registradores.
AtivaLeitura	1	entrada	Sinal para ativar a leitura no Banco de Registradores.
RegistradorDeSaida	32	saída	Saída do valor contido nos registradores.
RegistradorDestino	5	Entrada	Endereço do registrador destino.