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	Adcionada 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

SUMÁRIO

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 Morais	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

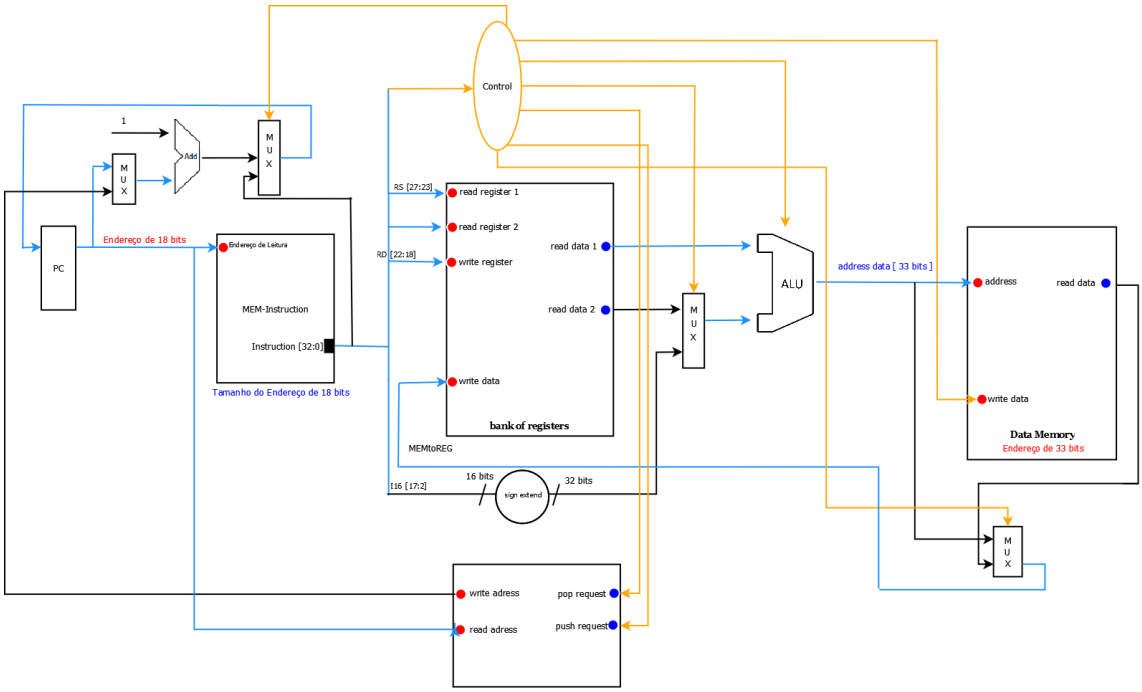
Termo	Descrição

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 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.

3. Intruções

A unidade de processamento possui 21 intruções 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.
  - **ADD:** Soma de dois valores.
  - **SUB:** Subtração de dois valores.
  - **MUL:** Multiplicação de dois valores.
  - **DIV:** Divisão de dois valores.
  - **AND:** Operação lógica AND entre dois valores.
  - **OR:** Operação lógica OR entre dois valores.
  - **NOT:** Operação lógica NOT.
  - **CMP:** Comparação de dois valores.
- **I-type** – Operações imediatas.
  - **ADDi:** Soma de dois valores, sendo um destes imediato.
  - **SUBi:** Subtração de dois valores, sendo um destes imediato.
  - **ANDi:** Operação lógica AND entre dois valores, sendo um destes imediato.

  - **ORi:** Operação lógica OR entre dois valores, sendo um destes imediato.

- **Load/Store** – Operações de carregamento e armazenamento.

  - **LW:** Operação de leitura na memória de dados.
  - **SW:** Operação de armazenamento na memória de dados.

- **Jump** – Operações de desvio.

  - **JR:** Desvia o programa para um endereço de destino.
  - **JPC:** Desvia o programa para um endereço relativo ao PC.
  - **BRFL:** Desvia o programa para um endereço de destino, atendendo uma condição de flag.
  - **CALL:** Desvia um programa em execução para uma sub-rotina.
  - **RET:** Retorna de uma sub-rotina.
  - **HALT:** Para a execução de um programa.
  - **NOP:** Não realiza operação.

4. Detalhamento das Intruções

- **ADD, SUB, MUL, DIV, AND, OR, NOT, CMP, ADDi, SUBi, ANDi e ORi:**

OPCODE	FUNCTION	RD	RS	RT	IM
04	04	05	05	05	09

Tabela 2.1: Layout das Operações Aritméticas, Lógicas e Imediatas

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:**

OPCODE	RD	RS	I	
04	04	05	16	02

Tabela 2.2: Layout das Operações de Leitura e Escrita

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.

- **JR, BRFL, CALL, RET, HALT e NOP:**

OPCODE	RF	CST	
04	05	05	18

Tabela 2.3: 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.

- **JPC:**

OPCODE	I
04	28

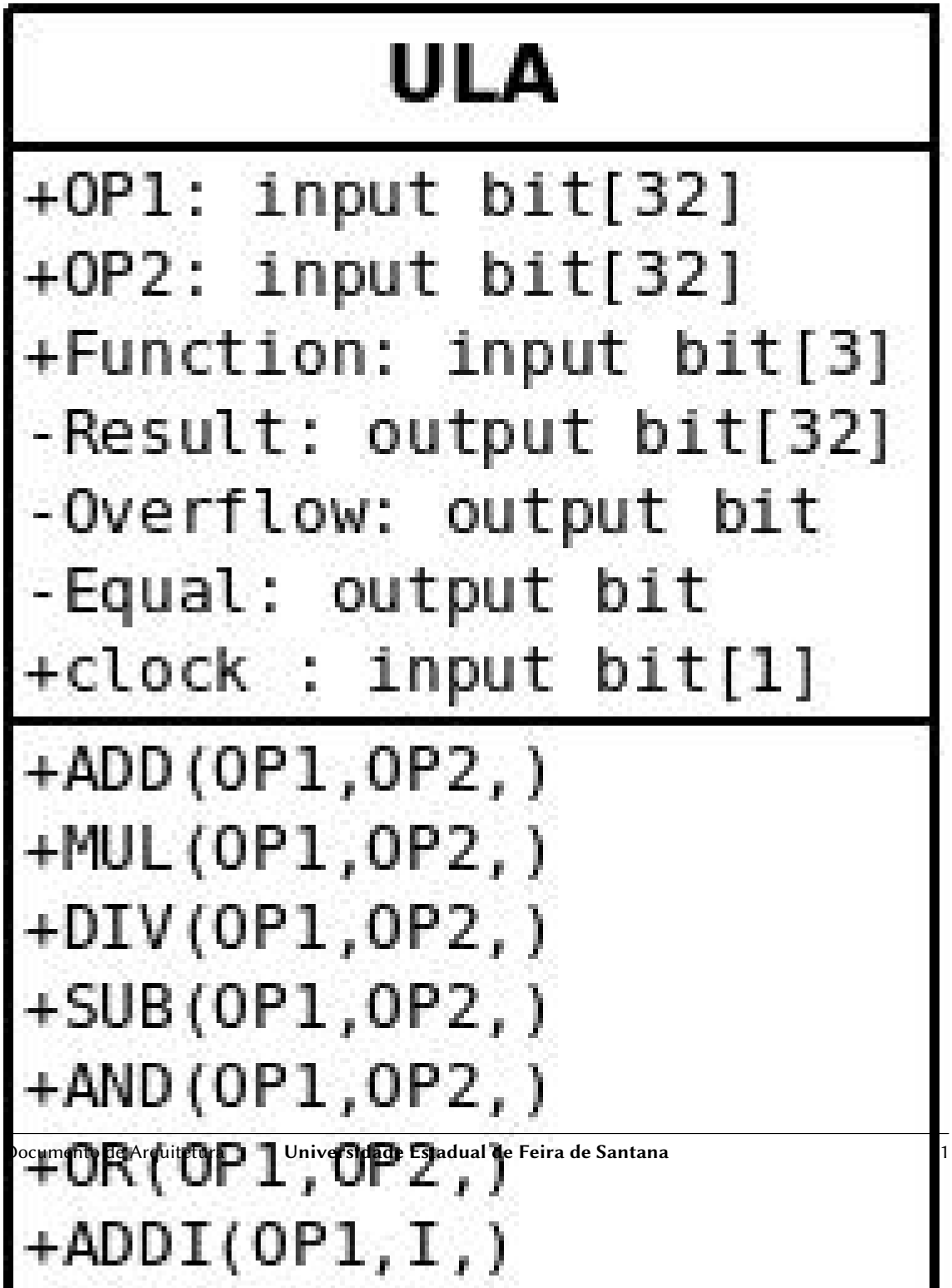
Tabela 2.4: Layout da Operação JPC

Essa instrução que representa um desvio relativo ao PC contém além do código de operação (OPCODE) um valor de 28 bits que representa o deslocamento o qual pode ser tomado.

3 | Descrição da Arquitetura

1. ULA

1.1. Diagrama de Classe

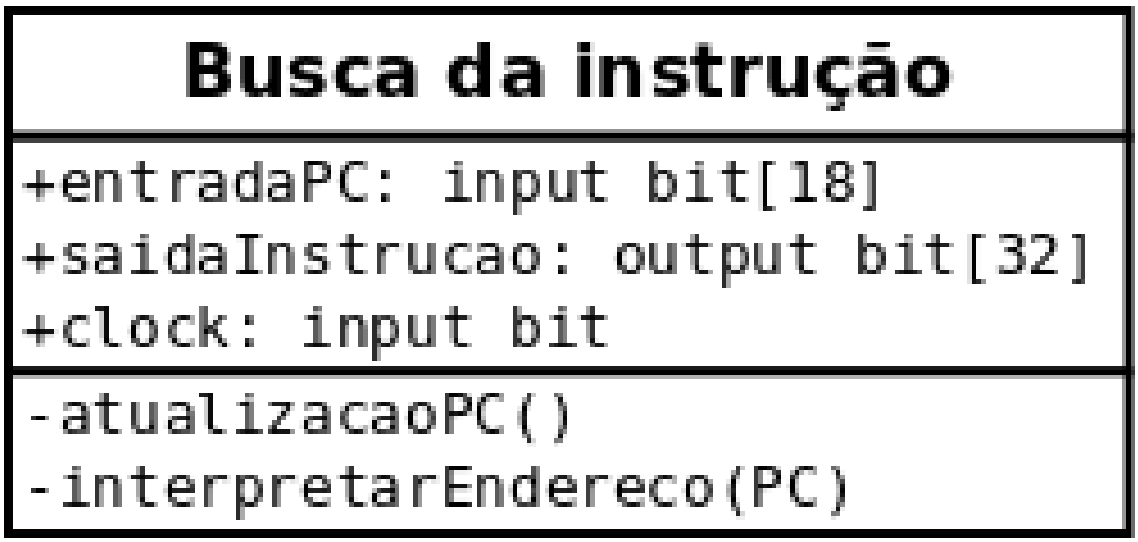


1.2. Definições de Entrada e Saída

Nome	Tamanho	Direção	Descrição
clock	1	entrada	Sinal de clock da fase.
OP1	32	entrada	Valor do primeiro operando.
OP2	32	entrada	Valor do segundo operando.
Function	3	entrada	Identificador da operação.
Result	32	saída	Valor do resultado da operação realizada.
Overflow	1	saída	Sinal de overflow, para quando ocorrer overflow durante a operação.
Equal	1	entrada	Sinal de igualdade, para quando ocorrer uma comparação entre dois valores iguais.

2. Busca de Instrução

2.1. Diagrama de Classe

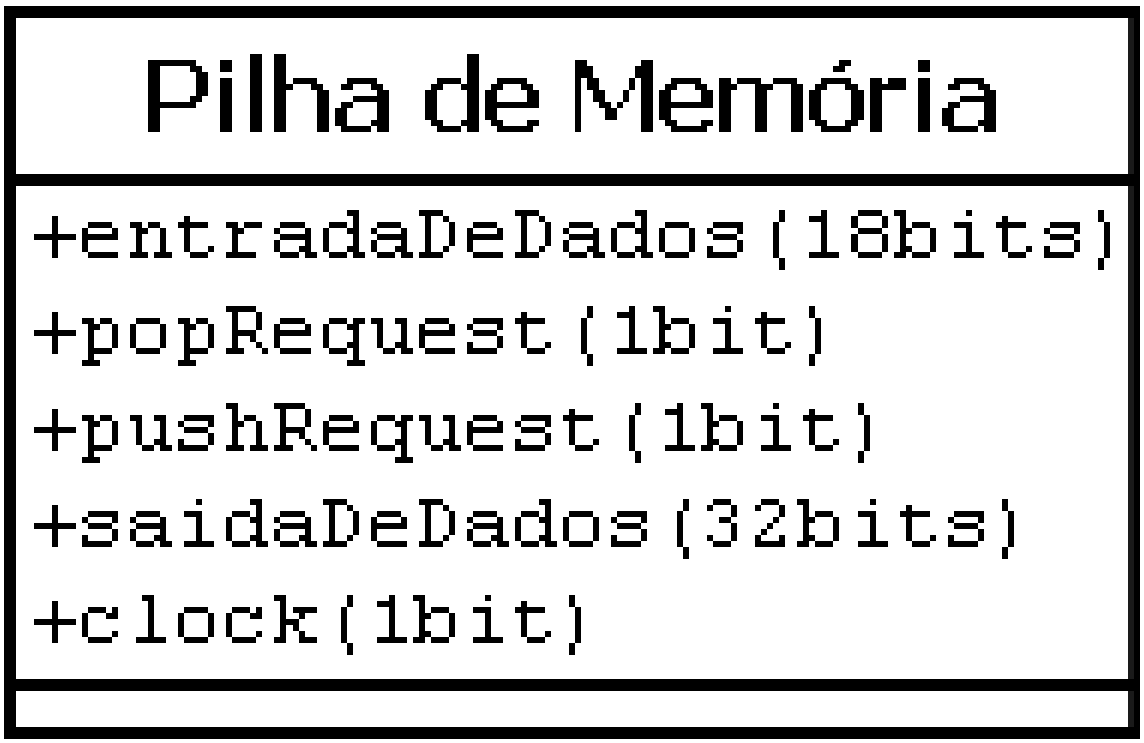


2.2. Definições de Entrada e Saída

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

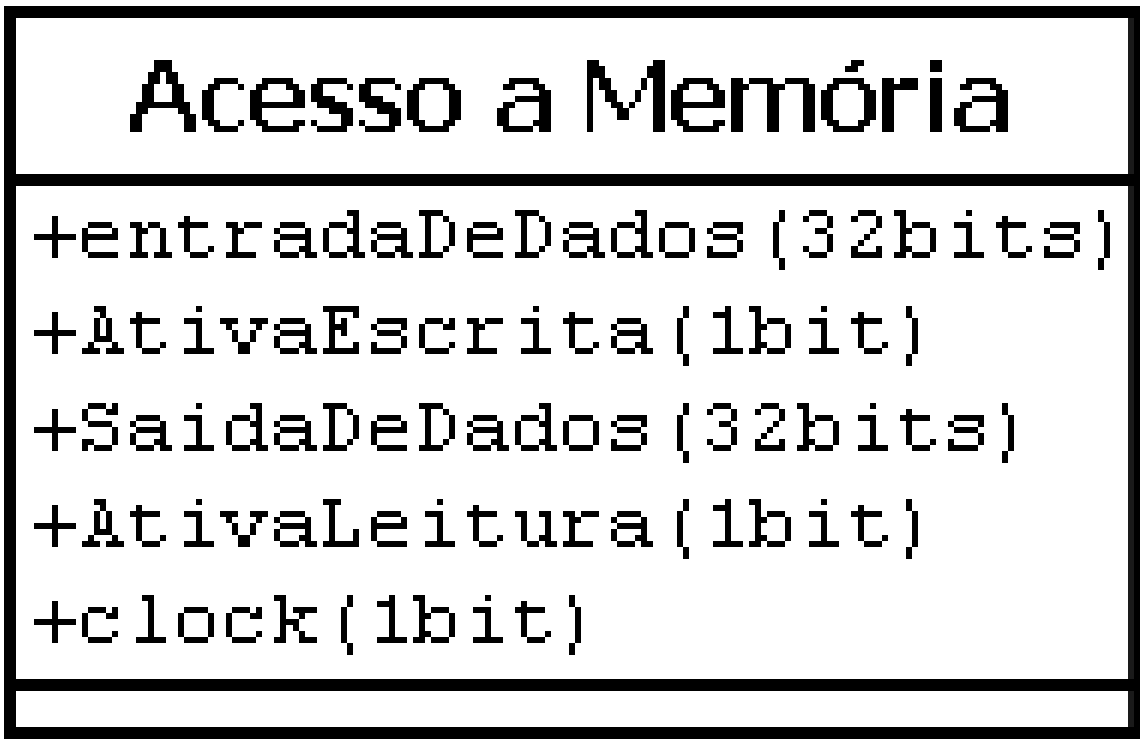
Nome	Tamanho	Direção	Descrição
clock_in	1	entrada	Sinal de clock da fase.
entradaDeDados	18	entrada	Endereço do PC que será armazenado.

continua na próxima página

continuação da página anterior			
Nome	**Tamanho**	**Direção**	**Descrição**
popRequest	1	entrada	Sinal para tirar o ultimo endereço armmazenado 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.