ipPROCESS

Documento de Arquitetura

Core-MUSA

Universidade Estadual de Feira de Santana

Build 3.0

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
15/12/2014	Revisão da visão geral da arquitetura	di3goleite
15/12/2014	Correção dos OPCODES do JR e do SUBi	di3goleite
16/12/2014	Mudanças no datapath e descrição da arquitetura	di3goleite
17/12/2014	Finalização da descrição da arquitetura	di3goleite
17/12/2014	Revisão parcial do documento e das definições de entrada e saídas	gordinh

SUMÁRIO

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-MUSA e 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 dos seguintes capítulos:

- **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
UC	Unidade de Controle
OPCODE	*Operation Code*
RS	*Register Source*
RD	*Register Destination*
LW	*Load Word*
SW	*Store Word*

2 | Visão Geral da Arquitetura

1. Descrição dos Componentes

A unidade de processamento a ser desenvolvida é constituída pelos 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, guardando todos os dados com tamanho de 32 *bits*.
- **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*, por conta do tamanho da palavra de instrução, 32 *bits*.
- **ULA** – É responsável por todo o processamento de instruções aritméticas do processador.
- **Unidade de Controle** – A UC é o componente responsável pela decodificação das instruções e pela definição dos sinais de controle que ativam cada bloco funcional do processador.
- **Banco de Registradores** – Contém os 32 registradores de propósito geral do processador.
- **Pilha** – Memória destinada para armazenamento dos endereços de retorno das chamadas de funções. Possui 32 registradores de 18 *bits* e um contador responsável por apontar o topo da pilha.

2. Instruções

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

- **Tipo R** – Operações entre registradores.

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

Tabela 2.1: Instruções do tipo R.

- **Tipo I** – Operações com imediatos.

INSTRUÇÃO	DESCRIÇÃO	OPCODE	FUNCTION
ADDi	Soma de dois valores, sendo um destes imediato.	001000	-
SUBi	Subtração de dois valores, sendo um destes imediato.	001110	-
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	-
JR	Desvia o programa para um endereço de destino.	011000	-
BRFL	Desvia o programa para um endereço de destino, atendendo uma condição de *flag*.	010001	111111
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 e NOT :**

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

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

O conjunto de instruções do tipo R utiliza o código da operação (OPCODE), dois registradores fontes (RS e RT) de dados e um registrador de destino (RD), para auxiliar na realização das operações. O campo FUNCTION é utilizado como um segundo campo de código de operação, ampliando o leque de operações possíveis. O campo SHAMT não será utilizado no projeto deste processador.

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

OPCODE	RD	RS	IMMEDIATE
06	05	05	16

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

Além do OPCODE, este tipo de instrução utiliza: um registrador fonte (RS) e um registrador de destino (RD) para instruções de leitura (LW) e de escrita (SW). Também conta com o campo IMMEDIATE (I) (de 16 *bits*) que representa o deslocamento do registrador base.

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

OPCODE	TARGET
06	26

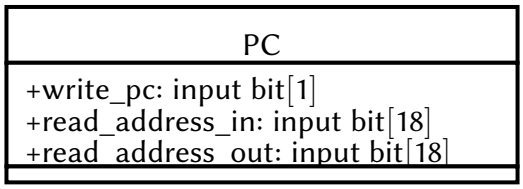
Tabela 2.6: *Layout* das Operações de Salto (Tipo J)

As instruções do tipo J contam com o OPCODE e um campo de 26 *bits*, representando uma posição de endereço de memória, estes campos são utilizados pelas instruções CALL, HALT e JPC, enquanto RET e NOP só utilizam o OPCODE da instrução.

3 | Descrição da Arquitetura

1. PC

1.1. Diagrama de bloco

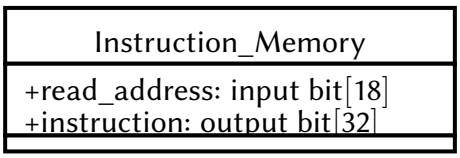


1.2. Definições de Entradas e Saídas

Nome	Tamanho	Direção	Descrição
write_pc	1	entrada	Entrada de controle que habilita a escrita do registrador do PC
read_address_in	18	entrada	Novo PC a ser armazenado
read_address_out	18	saída	Saída com o PC atualizado

2. Memória de Instrução

2.1. Diagrama de bloco

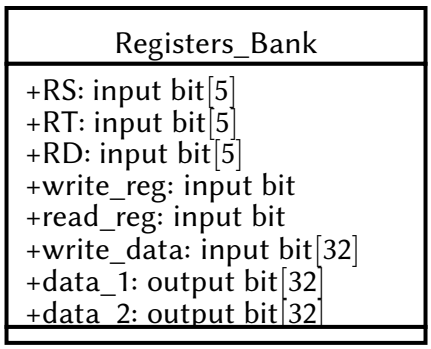


2.2. Definições de Entradas e Saídas

Nome	Tamanho	Direção	Descrição
read_address	18	entrada	Endereço que será lido
instruction	32	saída	Saída da instrução atual

3. Banco de Registradores

3.1. Diagrama de bloco

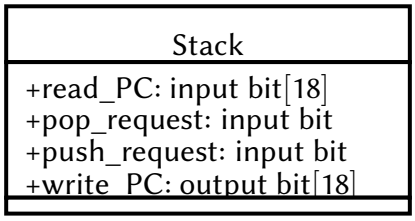


3.2. Definições de Entradas e Saídas

Nome	Tamanho	Direção	Descrição
RS	5	entrada	Registrador fonte
RT	5	entrada	Registrador fonte
RD	5	entrada	Registrador de destino
write_reg	1	entrada	Habilita a escrita do registrador destino em RD
read_reg	1	entrada	Habilita a leitura dos registradores fonte (RS e RT)
write_data	32	entrada	Dado a ser escrito no registrador de destino RD
data_1	32	saída	Dado 1
data_2	32	saída	Dado 2

4. Pilha

4.1. Diagrama de bloco

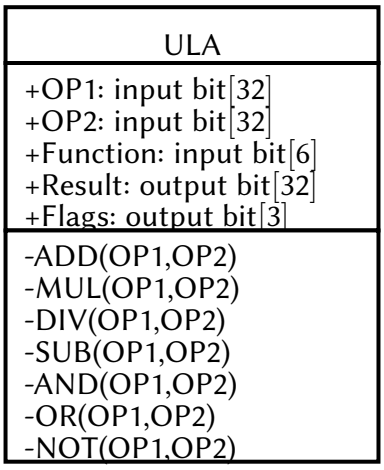


4.2. Definições de Entradas e Saídas

Nome	Tamanho	Direção	Descrição
read_PC	18	entrada	Endereço atual capturado do PC
pop_request	1	entrada	Sinal utilizado para remover o último endereço armazenado na pilha
push_request	1	entrada	Sinal utilizado para armazenar um endereço na pilha
write_PC	18	saída	Altera o PC pelo endereço retornado por um pop_request

5. ULA

5.1. Diagrama de bloco

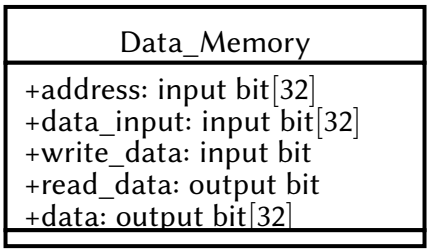


5.2. Definições de Entradas e Saídas

Nome	Tamanho	Direção	Descrição
OP1	32	entrada	Primeiro operando
OP2	32	entrada	Segundo operando
Function	6	entrada	Identificador da operação
Result	32	saída	Resultado da operação
Flags	3	saída	*Flags above*, *equals* e *overflow*

6. Acesso à memória

6.1. Diagrama de bloco



6.2. Definições de Entradas e Saídas

Nome	Tamanho	Direção	Descrição
address	32	entrada	Endereço da memória onde o dado será armazenado
data_input	32	entrada	Valor a ser armazenado na memória de dados
write_data	1	entrada	Sinal que habilita a escrita da memória de dados
read_data	1	entrada	Sinal que habilita a leitura da memória de dados
data	32	saída	Saída do dado requisitado

4 | DataPath Externo