



Documento de Arquitetura

MUSA

Fazemos Qualquer Negócio Inc.

Compilação 2.1

Histórico de Revisões

| Date | Descrição | Autor(s) |
|------------|--|------------------------|
| 25/06/2014 | Concepção do documento | joaocarlos |
| 15/10/2014 | Adição da subseção de acesso à memória | weversongomes |
| 16/10/2014 | Adição da seção "Leitura da Instrução" com dados preliminares e modificação do nome do projeto no documento. | santana22 e gabri14el. |
| 19/10/2014 | Modificações na seção "Leitura da Instrução" | santana22 |
| 20/10/2014 | Correções na subseção de acesso à memória | weversongomes |
| 25/10/2014 | Adição das descrições da codificação | manuellemacedo |
| 29/10/2014 | Unificação das subseções de acesso à memória com a de write back | weversongomes |
| 29/10/2014 | Adição das descrições dos componentes | manuellemacedo |
| 30/10/2014 | Adição do Datapath (Instruction Fetch) | santana22 e gabri14el |
| 30/10/2014 | Alterações na subseção de acesso à memória com definições de número de bits | weversongomes |
| 03/11/2014 | Correções na subseção "Leitura da Instrução" | weversongomes |

SUMÁRIO

| | | |
|----------|---|-----------|
| 1 | Introdução | 4 |
| 1 | Propósito do Documento | 4 |
| 2 | Stakeholders | 4 |
| 3 | Visão Geral do Documento | 4 |
| 4 | Acrônimos e Abreviações | 5 |
| 2 | Visão Geral da Arquitetura | 6 |
| 1 | Codificação das instruções | 6 |
| 2 | Descrição dos Componentes | 9 |
| 3 | Descrição da Arquitetura | 11 |
| 1 | Leitura da Instrução | 11 |
| 1.1 | Diagrama de Classe | 11 |
| 1.2 | Definições de entrada e saída | 11 |
| 1.3 | Datapath Interno | 11 |
| 2 | Decodificação da Instrução | 12 |
| 2.1 | Diagrama de Classe | 12 |
| 2.2 | Definições de entrada e saída | 13 |
| 2.3 | Tabela de microinstruções | 13 |
| 2.4 | Datapath Interno | 14 |
| 3 | Estágio de execução | 15 |
| 3.1 | Diagrama de Classe | 15 |
| 3.2 | Definições de entrada e saída | 15 |
| 3.3 | Datapath Interno | 16 |
| 4 | Acesso à memória e write back | 17 |

| | | |
|-----|---|----|
| 4.1 | Diagrama de Classe | 17 |
| 4.2 | Definições de entrada e saída | 17 |
| 4.3 | Datapath Interno | 18 |

1 | Introdução

1. Propósito do Documento

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

2. Stakeholders

| Nome | Papel/Responsabilidades |
|--|-------------------------|
| Manuelle | Gerência |
| Manuelle, Patrick, Lucas, Mirela, Vinícius, Gabriel, Weverson, Anderson e Tarles | Análise |
| Manuelle, Patrick, Lucas, Mirela, Vinícius, Gabriel, Weverson, Anderson e Tarles | Desenvolvimento |
| Patrick, Lucas, Mirela, Vinícius, Gabriel, Weverson, Anderson e Tarles | Testes |
| Patrick, Lucas, Mirela, Vinícius, Gabriel, Weverson, Anderson e Tarles | Implementação |

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 descreve a arquitetura interna do IP a partir do detalhamento dos seus componentes, definição de portas de entrada e saída e especificação de caminho de dados.

4. Acrônimos e Abreviações

| Sigla | Descrição |
|--------|-----------------|
| PC | Program Counter |
| OPCODE | Operation Code |

2 | Visão Geral da Arquitetura

1. Codificação das instruções

Instrução é uma palavra da linguagem de máquina, sua codificação é de fundamental importância para o processamento das operações. Todas as instruções contêm 32 bits. Existem 4 formatos de instruções: R (R-type), I (I-type), Load/Store e Jump. Os OPCODES são os códigos de operação da instrução, neste documento ele é representação em números hexadecimais.

| Formato da instrução | Instrução | Descrição |
|----------------------|-----------|---|
| R-type | ADD | Soma dois valores |
| | SUB | Subtrai dois valores |
| | MUL | Multiplica dois valores |
| | DIV | Divide dois valores |
| | AND | AND lógico |
| | OR | OR lógico |
| | CMP | Compara dois valores |
| | NOT | NOT lógico |
| I-type | ADDI | Soma dois valores, um destes imediato. |
| | SUBI | Subtrai dois valores, um destes imediato. |
| | ANDI | AND lógico de dois valores, um destes imediato. |
| | ORI | OR lógico de dois valores, um destes imediato. |
| | LW | Leitura de um dado da memória de dados |
| | SW | Armazena um dado na memória de dados |
| Jump | JP | Desvia para um destino |
| | JPC | Desvia para um destino relativo ao PC |
| | BRFL | Desvia para um destino se RF==CST |

continua na próxima página

| continuação da página anterior | | |
|--------------------------------|-----------|----------------------|
| Formato da Instrução | Instrução | Descrição |
| | CALL | Chamada de subrotina |
| | RET | Retorno de Subrotina |
| | HALT | Parada do sistema |
| | NOPE | Refresh no módulo |

O formato R está relacionado as instruções lógicas e aritméticas.

| OPCODE | RS | RT | RD | SHAMT | FUNCT |
|--------|-------|-------|-------|-------|-------|
| 31:26 | 25:21 | 20:16 | 15:11 | 10:6 | 5:0 |

Figura 2.1: Formato R

Seus respectivos campos são:

- **OPCODE** - Código da operação básica da instrução.
- **RS** - Registrador do primeiro operando de origem.
- **RT** - Registrador do segundo operando de origem.
- **RD** - Registrador destino.
- **SHAMT** - *Shift amount*; Quantidade de deslocamento.
- **FUNCT** - Função; Esse campo seleciona a variante específica da operação no campo opcode, e as vezes, é chamado de código de função.

| OPCODE | INSTRUCTION | FUNCTION |
|--------|-------------|----------|
| 01 | ADD | 00 |
| 01 | SUB | 01 |
| 01 | MUL | 02 |
| 01 | DIV | 03 |
| 02 | AND | 04 |
| 02 | OR | 05 |
| 02 | CMP | 06 |
| 02 | NOT | 07 |

Tabela 2.2: Definição dos OPCODES do formato R

Um segundo tipo de formato de instrução é chamado de formato I, utilizado pelas instruções imediatas e de transferência de dados.

| OPCODE | RS | RT | ADDRESS OR IMMEDIATE |
|--------|-------|-------|----------------------|
| 31:26 | 25:21 | 20:16 | 15:0 |

Figura 2.2: Formato I

Seus respectivos campos são:

- **OPCODE** - Código da operação básica da instrução.
- **RS** - Registrador do operando de origem.
- **RT** - Registrador destino.
- **ADDRESS OR IMMEDIATE** - Endereço de memória ou constante numérica.

| OPCODE | INSTRUCTION |
|--------|-------------|
| 03 | ADDI |
| 04 | SUBI |
| 05 | ANDI |
| 06 | ORI |
| 07 | LW |
| 08 | SW |

Tabela 2.3: Definição dos OPCODES do formato I

O formato Jump servem para as instruções de desvio incondicional.

| OPCODE 31:26 | ADDRESS 25:0 |
|-----------------|-----------------|
|-----------------|-----------------|

Figura 2.3: Formato Jump

Seus respectivos campos são:

- **OPCODE** - Código da operação básica da instrução.
- **ADDRESS** - Endereço de memória ou constante numérica.

| OPCODE | INSTRUCTION |
|--------|-------------|
| 09 | JP |
| 0A | JPC |
| 0B | BRFL |
| 0C | CALL |
| 0D | RET |
| 0E | HALT |
| 00 | NOPE |

Tabela 2.4: Definição dos OPCODES do formato Jump

2. Descrição dos Componentes

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

- **Instruction Fetch** – Módulo responsável pela busca da instrução na memória de instrução.
- **Instruction Decode e Register Read** – Módulo responsável pela decodificação das instruções e leitura do banco de registradores.
- **Execute Operation or Calculate Address** – Módulo responsável pela execução das operações de carácter lógico/aritmético ou cálculos endereços.
- **Memory Access e Write Back** – Módulo responsável pelo acesso a memória de dados e escrita no banco de registradores.

3 | Descrição da Arquitetura

1. Leitura da Instrução

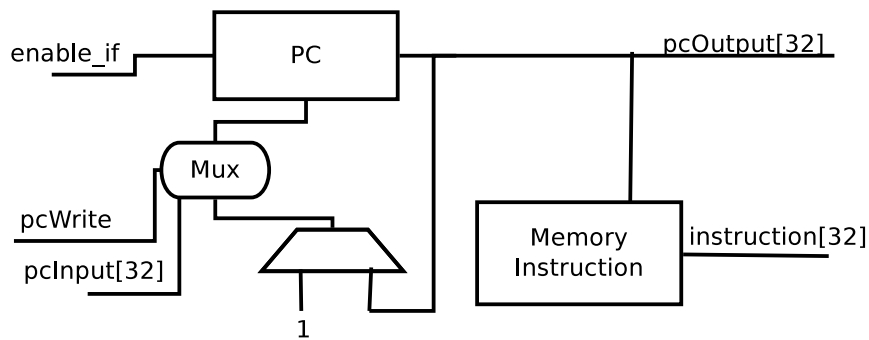
1.1. Diagrama de Classe

| Instruction Fetch |
|---|
| + clock : input bit + reset : input bit + pcInput : input bit[14] + pcWrite : input bit + pcOutput : output bit[14] + instruction : output bit[32] |
| - «comb» <u>search_Instruction()</u> - «comb» next_PC() |

1.2. Definições de entrada e saída

| Nome | Tamanho | Direção | Descrição |
|-------------|---------|---------|--|
| pcInput | 14 | entrada | Valor do PC atual. |
| pcWrite | 1 | entrada | Sinal que habilita a modificação do valor de PC. |
| pcOutput | 14 | saída | Valor do PC atual. |
| instruction | 32 | saída | Instrução a ser executada. |

1.3. Datapath Interno



2. Decodificação da Instrução

2.1. Diagrama de Classe

| Instruction Decode |
|--|
| <ul style="list-style-type: none"> + clock : input bit + reset : input bit + instruction : input bit[32] + regDst : input bit + writeData : input bit + writeRegister : input bit + word : input bit[16] + regDst : output bit + branch : output bit + memRead : output bit + memToReg : output bit + aluOp : output bit + memWrite : output bit + aluSrc : output bit + regWrite : output bit + jump : output bit + readData1 : output bit[32] + readData2 : output bit [32] + outputWord : output bit [32] + registers : reg bit[32] |
| <ul style="list-style-type: none"> - «comb» opcode_decoder() - «comb» search_register() - «comb» set_write_register() - «sequ» sign_extend() - «sequ» zero_extend() |

2.2. Definições de entrada e saída

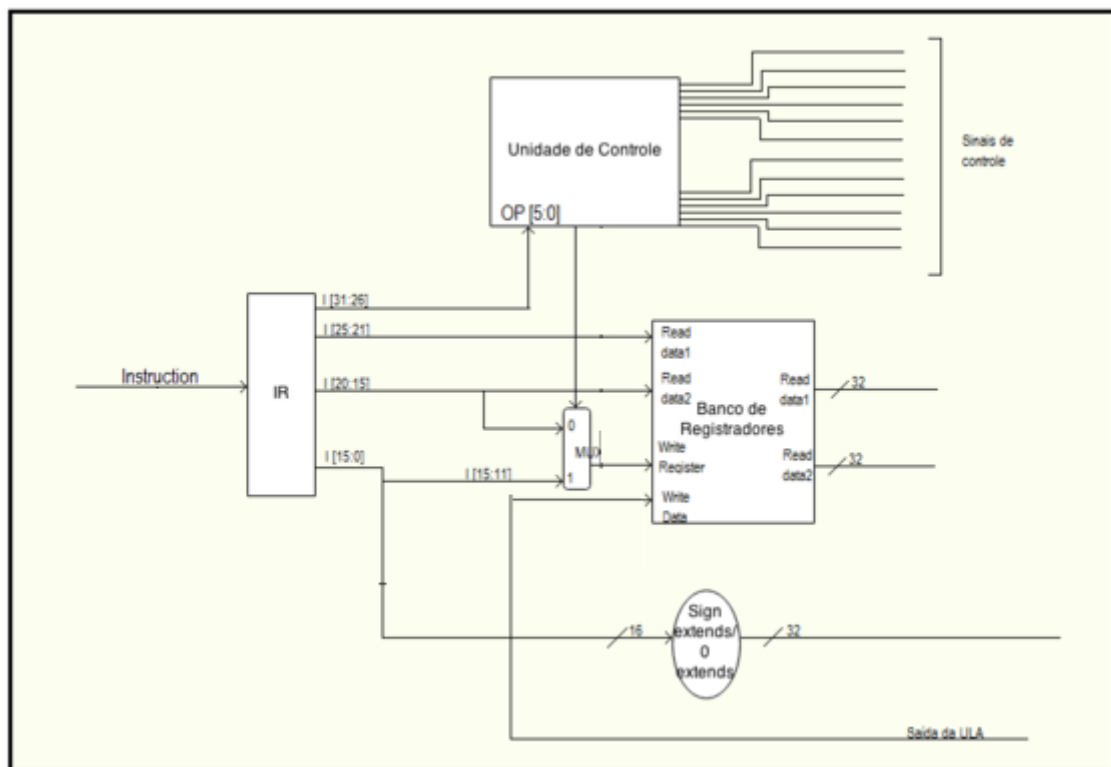
| Nome | Tamanho | Direção | Descrição |
|---------------|---------|---------|---|
| instruction | 32 | entrada | Instrução a ser executada. |
| writeData | 1 | entrada | Sinal de controle para escrita no registrador. |
| writeRegister | 1 | entrada | Endereço do registrador de destino do writeData. |
| word | 16 | entrada | é uma instrução. |
| branch | 1 | saída | Sinal que informa ao circuito se a instrução é de branch. |
| memRead | 1 | saída | Sinal de controle para realizar leitura da memória. |
| memToReg | 1 | saída | Sinal de controle que define se o dado deve vir da ULA ou da memória |
| aluOp | 2 | saída | Sinal de controle que define se o código funct da instrução deve ser levado em consideração ou não. |
| memWrite | 1 | saída | Sinal de controle para realizar escrita na memória. |
| aluSrc | 1 | saída | Sinal de controle que define qual entrada a ULA deve utilizar para realizar a operação. |
| regWrite | 1 | saída | Sinal de controle para realizar escrita no registrador. |
| jump | 1 | saída | Sinal que informa ao circuito se a operação é de jump. |

2.3. Tabela de microinstruções

| Tipo | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E | 00 |
|---------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| regDst | 1 | 1 | 0 | 0 | 0 | 0 | 1 | x | x | x | x | x | x | x | 0 |
| branch | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | x | x | 1 | x | x | x | 0 |
| memRead | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | x | x | 0 | x | x | x | 0 |

| Tipo | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E | 00 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| memToReg | 0 | 0 | 1 | 1 | 1 | 1 | 1 | x | x | x | x | x | x | x | 0 |
| aluOp | 10 | 10 | 10 | 10 | 10 | 10 | 00 | 00 | xx | xx | 01 | xx | xx | xx | 00 |
| memWrite | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | x | x | 0 | x | x | x | 0 |
| aluSrc | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | x | x | 0 | x | x | x | 0 |
| regWrite | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | x | x | 0 | x | x | x | 0 |
| jump | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | x | 0 |

2.4. Datapath Interno



Datapath do estágio 2.

3. Estágio de execução

3.1. Diagrama de Classe

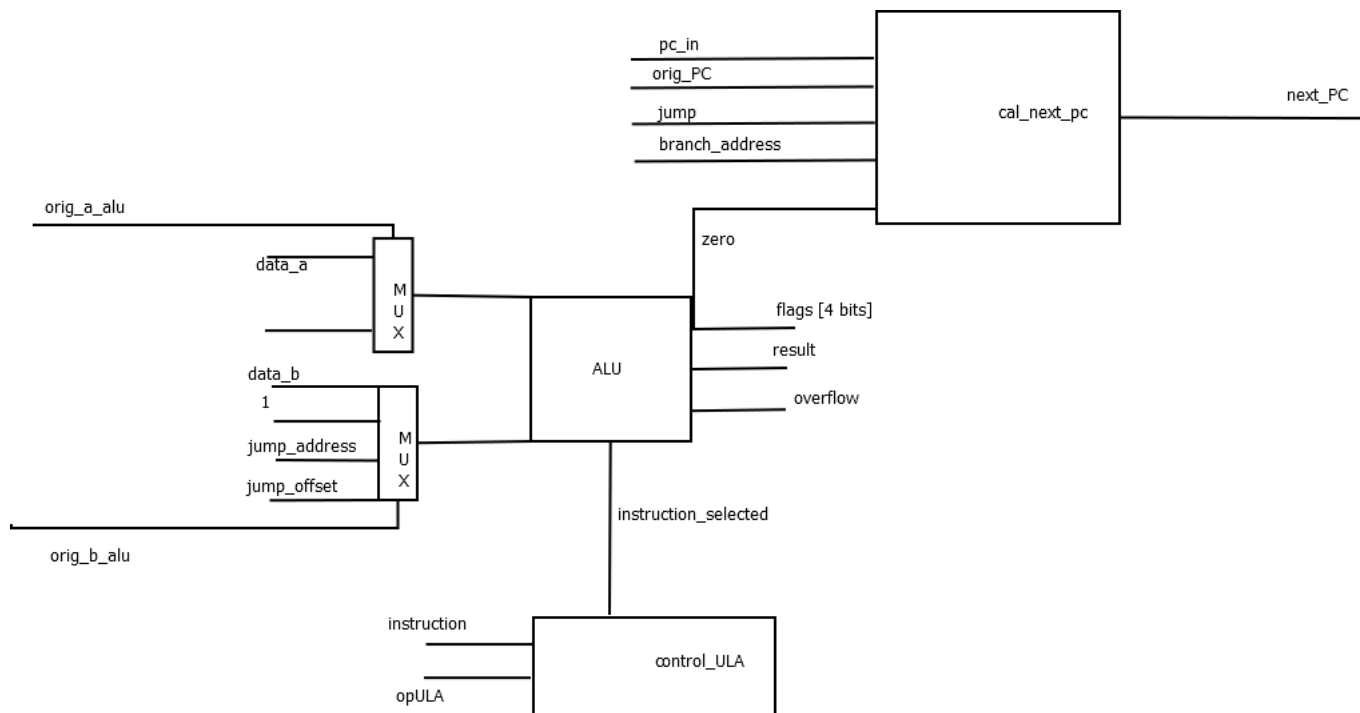
| Execute Operation |
|--|
| + clock : input bit + reset : input bit + data_a : input bit[32] + data_b : input bit[32] + pc_in : input bit[14] + orig_pc: input bit [2] + op_alu : input bit[5] + origin_aalu : input bit[1] + origin_balu : input bit[2] + instruction : output bit[32] + offset_inst : input [32] |
| +«comb» calc_next_pc() + «comb» cal_al_operation() |

3.2. Definições de entrada e saída

| Nome | Tamanho | Direção | Descrição |
|----------------|---------|---------|--|
| data_a | 32 | Entrada | Dado do primeiro operando. |
| data_b | 32 | Entrada | Dado do segundo operando. |
| pc_in | 14 | Entrada | Valor do PC atual. |
| rt_data_mem | 5 | Entrada | Dado vindo da memória. |
| opcode | 5 | Entrada | Código da operação. |
| control_signal | TBD | Entrada | Sinal de comando da unidade de controle. |
| alu_result | 32 | Saída | Representação do resultado da operação. |
| pc_out | 32 | Saída | Valor do PC atual. |
| write_reg_mem | 5 | Saída | Sinal proveniente da UC que habilita a escrita no registrador. |
| memory_address | 32 | Saída | Endereço da memória. |

| Nome | Tamanho | Direção | Descrição |
|---------------------|---------|---------|--|
| memory_write_enable | TBD | Saída | Sinal proveniente da UC que habilita a escrita na memória. |

3.3. Datapath Interno



Datapath interno do estágio de execução (EX).

4. Acesso à memória e write back

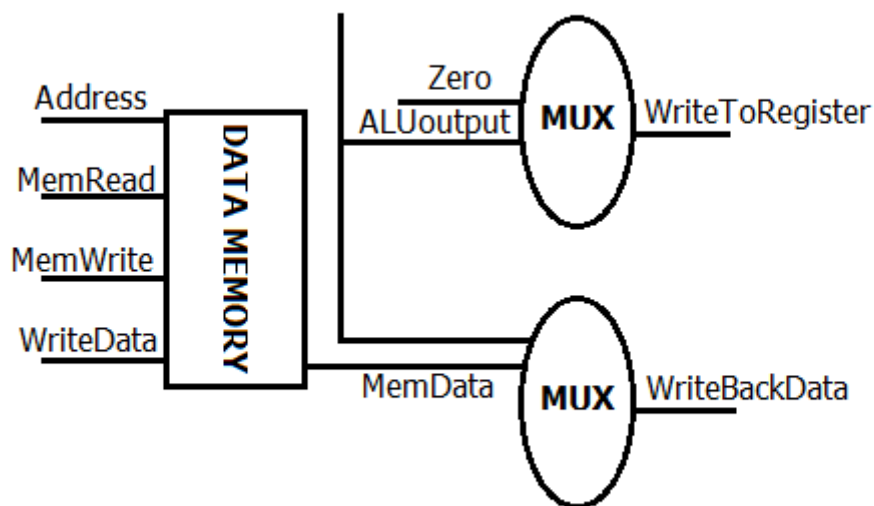
4.1. Diagrama de Classe

| Acess Memory |
|------------------------------------|
| + clock : input bit |
| + reset : input bit |
| + zero : input bit |
| + address : input bit |
| + writeData : input bit[13] |
| + memRead : input bit |
| + memWrite : input bit |
| - writeBack : output bit[14] |
| - writeToRegister : output bit[14] |

4.2. Definições de entrada e saída

| Nome | Tamanho | Direção | Descrição |
|-----------------|---------|---------|--|
| zero | 1 | entrada | Executa branch quando é zero. |
| address | 13 | entrada | Endereço no qual o dado deve ser escrito. |
| memRead | 1 | entrada | Sinal proveniente da UC que habilita leitura. |
| memWrite | 1 | entrada | Sinal proveniente da UC que habilita escrita. |
| writeData | 1 | entrada | Dado a ser escrito na memória. |
| writeBack | 14 | saída | Dado proveniente da ALU que será escrito no bloco de registradores |
| writeToRegister | 14 | saída | Dado do segundo operando. |

4.3. Datapath Interno



Datapath dos estágios 4 e 5.