RISC-V-Playground

0.1

Gerado por Doxygen 1.9.1

Capítulo 1

Algorítimo para Simulador RISC-V

1.1 Objetivo

Aplicativo tem como objetivo simular o comportamento de uma máquina RISC-V.

Considerando um conjunto limitado de instruções:

- ADDI
- ADD
- SUB
- AND
- OR
- LW
- SW
- BEQ
- BNE

1.2 Descrevendo a experiência

Escrever algo aqui

1.2.1 Pontos positivos

- Alguma coisa

1.2.2 Pontos negativos

- Alguma coisa

1.3 Descrição da dinâmica de trabalho do grupo

Dentro da documentação de cada método e classe possui o nome do author

Capítulo 2

Índice Hierárquico

2.1 Hierarquia de Classes

Esta lista de hierarquias está parcialmente ordenada (ordem alfabética):

Capítulo 3

Índice dos Componentes

3.1 Lista de Classes

Aqui estão as classes, estruturas, uniões e interfaces e suas respectivas descrições:

Capítulo 4

Índice dos Arquivos

4.1 Lista de Arquivos

Esta é a lista de todos os arquivos documentados e suas respectivas descrições:

Capítulo 5

Classes

5.1 Referência da Classe ArithmeticLogicUnit

A classe ArithmeticLogicUnit:

`#include <arithmeticlogicunit.h>`

5.1.1 Descrição detalhada

A classe ArithmeticLogicUnit:

Definição na linha 14 do arquivo arithmeticlogicunit.h.

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

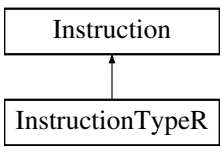
- arithmeticlogicunit.h
- arithmeticlogicunit.cpp

5.2 Referência da Classe Instruction

A classe Instruction: Classe abstrata a qual é base para todos os modos de instrução.

`#include <instruction.h>`

Diagrama de hierarquia para Instruction:



Membros Públicos

- Instruction ()

  Inicializa todos os campos com zero.
- Instruction (int op, int rd, int f3, int rs1)
- int getOpcode ()

  Obtêm o valor do opcode para alguém fora da classe.
- int getRegDest ()

  Obtêm o valor do registrador destino para alguém fora da classe.
- int getFunc3 ()

  Obtêm o valor do Funct 3 para alguém fora da classe.
- int getRegSrc1 ()

  Obtêm o valor do registrador sorce 1 para alguém fora da classe.
- virtual void **printInfo** ()=0
- virtual bool * **instructionToBin** ()=0

5.2.1 Descrição detalhada

A classe Instruction: Classe abstrata a qual é base para todos os modos de instrução.

Definição na linha 32 do arquivo instruction.h.

5.2.2 Construtores e Destrutores

5.2.2.1 Instruction() [1/2]

```
Instruction::Instruction ( )
```

Inicializa todos os campos com zero.

Vamos colocar algo mais detalhado

Definição na linha 15 do arquivo instruction.cpp.

5.2.2.2 Instruction() [2/2]

```
Instruction::Instruction (
            int op,
            int rd,
            int f3,
            int rs1 )
```

Esse é um método provido por conveniência. Ele difere do método acima apenas na lista de argumentos que devem ser utilizados.

Parâmetros

op	valor do opcode
rd	valor do registrador destino
f3	valor do funct 3
rs1	valor do registrador fonte 1

Definição na linha 31 do arquivo instruction.cpp.

5.2.3 Funções membros

5.2.3.1 getFunc3()

```
int Instruction::getFunc3 ( )
```

Obtêm o valor do Funct 3 para alguém fora da classe.

Retorna

o valor do funct3

Definição na linha 63 do arquivo instruction.cpp.

5.2.3.2 getOpcode()

```
int Instruction::getOpcode ( )
```

Obtêm o valor do opcode para alguém fora da classe.

Retorna

o valor do opcode

Definição na linha 43 do arquivo instruction.cpp.

5.2.3.3 getRegDest()

```
int Instruction::getRegDest ( )
```

Obtêm o valor do registrador destino para alguém fora da classe.

Retorna

o valor do registrador de destino

Definição na linha 53 do arquivo instruction.cpp.

5.2.3.4 getRegSrc1()

```
int Instruction::getRegSrc1 ( )
```

Obtêm o valor do registrador sorce 1 para alguém fora da classe.

Retorna

valor do registrador fonte 1

Definição na linha 72 do arquivo instruction.cpp.

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

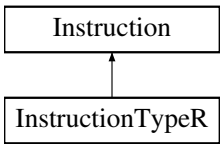
- instruction.h
- instruction.cpp

5.3 Referência da Classe InstructionTypeR

A classe InstructionTypeR:

```
#include <instructiontyper.h>
```

Diagrama de hierarquia para InstructionTypeR:



Membros Públicos

- InstructionTypeR ()
  Inicializa todos os valores com zero.
- InstructionTypeR (int op, int rd, int f3, int rs1, int rs2, int f7)
- bool * instructionToBin ()
  Cria um vetor contendo o valor binário da instrução.
- void printInfo ()
  Imprime na tela todas as informações dos membros da classe.

5.3.1 Descrição detalhada

A classe InstructionTypeR:

Definição na linha 19 do arquivo instructiontyper.h.

5.3.2 Construtores e Destrutores

5.3.2.1 InstructionTypeR() [1/2]

```
InstructionTypeR::InstructionTypeR ( )
```

Inicializa todos os valores com zero.

Explicar mais (Se remover isso quebra)

Definição na linha 15 do arquivo instructiontyper.cpp.

5.3.2.2 InstructionTypeR() [2/2]

```
InstructionTypeR::InstructionTypeR (
            int op,
            int rd,
            int f3,
            int rs1,
            int rs2,
            int f7 )
```

Esse é um método provido por conveniência. Ele difere do método acima apenas na lista de argumentos que devem ser utilizados.

Parâmetros

op	valor do opcode
rd	valor do registrador destino
f3	valor do funct 3
rs1	valor do registrador fonte 1
rs2	valor do registrador fonte 1
f7	valor do funct 7

Definição na linha 31 do arquivo instructiontyper.cpp.

5.3.3 Funções membros

5.3.3.1 instructionToBin()

```
bool * InstructionTypeR::instructionToBin ( )  [virtual]
```

Cria um vetor contendo o valor binário da instrução.

Dá para escrever mais aqui

Aviso

o vetor dinâmico de valores booleanos deve ser desalocado por quem chamou.

Pré-condição

Todos os membros da classe devem estar inicializados.

Pós-condição

cria o vetor dinâmico de valores booleanos, contendo o valor binário da instrução.

Retorna

vetor de valores booleanos, contendo o valor binário da instrução.

Implementa Instruction.

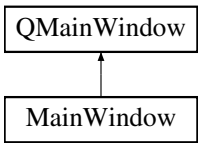
Definição na linha 46 do arquivo instructiontyper.cpp.

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- instructiontyper.h
- instructiontyper.cpp

5.4 Referência da Classe MainWindow

Diagrama de hierarquia para MainWindow:



Membros Públicos

- **MainWindow** (QWidget ∗parent=nullptr)

5.4.1 Descrição detalhada

Definição na linha 10 do arquivo mainwindow.h.

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- mainwindow.h
- mainwindow.cpp

5.5 Referência da Classe Memory

A classe Memory:

```
#include <memory.h>
```

Membros Públicos

- Memory ()

  Inicializa todas as células de memória com zero.
- int get (int position)

  Lê o valor contido na memória.
- void set (int position, int value)

  Escreve o valor passado por referência na memória.

5.5.1 Descrição detalhada

A classe Memory:

Definição na linha 16 do arquivo memory.h.

5.5.2 Funções membros

5.5.2.1 get()

```
int Memory::get (
            int position )
```

Lê o valor contido na memória.

Parâmetros

position	indica o registrador a ser lido

Retorna

o valor contido na memória na posição[position]

Definição na linha 25 do arquivo memory.cpp.

5.5.2.2 set()

```
void Memory::set (
            int position,
            int value )
```

Escreve o valor passado por referência na memória.

Parâmetros

position	indica o registrador a ser lido
value	valor a ser escrito

Definição na linha 35 do arquivo memory.cpp.

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- memory.h
- memory.cpp

5.6 Referência da Classe ProgramCounter

A classe ProgramCounter:

```
#include <programcounter.h>
```

5.6.1 Descrição detalhada

A classe ProgramCounter:

Definição na linha 14 do arquivo programcounter.h.

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- programcounter.h
- programcounter.cpp

5.7 Referência da Classe Registers

A classe Registers:

```
#include <registers.h>
```

Membros Públicos

- Registers ()

  Inicializa todos os registradores com zero.
- int get (int position)

  Lê o valor contido em um registrador.
- void set (int position, int value)

  Escreve o valor passado por referência em um registrador.

5.7.1 Descrição detalhada

A classe Registers:

Definição na linha 16 do arquivo registers.h.

5.7.2 Funções membros

5.7.2.1 get()

```
int Registers::get (
            int position )
```

Lê o valor contido em um registrador.

Parâmetros

position	indica o registrador a ser lido

Retorna

o valor contido no registrador x[position]

Definição na linha 25 do arquivo registers.cpp.

5.7.2.2 set()

```
void Registers::set (
            int position,
            int value )
```

Escreve o valor passado por referência em um registrador.

Parâmetros

position	indica o registrador a ser lido
value	valor a ser escrito

Definição na linha 35 do arquivo registers.cpp.

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- registers.h
- registers.cpp

5.8 Referência da Classe Simulator

A classe Simulator:

```
#include <simulator.h>
```

5.8.1 Descrição detalhada

A classe Simulator:

Definição na linha 21 do arquivo simulator.h.

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- simulator.h
- simulator.cpp

Capítulo 6

Arquivos

6.1 Referência do Arquivo arithmeticlogicunit.cpp

Arquivo que implementa os métodos da classe ArithmeticLogicUnit.

```
#include "arithmeticlogicunit.h"
```

6.1.1 Descrição detalhada

Arquivo que implementa os métodos da classe ArithmeticLogicUnit.

Autor

Victor Emanuel Almeida

Versão

0.1

6.2 Referência do Arquivo arithmeticlogicunit.h

Arquivo que define a classe ArithmeticLogicUnit.

Componentes

- class ArithmeticLogicUnit

  A classe ArithmeticLogicUnit:

6.2.1 Descrição detalhada

Arquivo que define a classe ArithmeticLogicUnit.

Autor

Victor Emanuel Almeida

Versão

0.1

6.3 Referência do Arquivo instruction.cpp

Arquivo que implementa os métodos da classe Instruction.

```
#include "instruction.h"
```

6.3.1 Descrição detalhada

Arquivo que implementa os métodos da classe Instruction.

Autor

Victor Emanuel Almeida

Versão

0.1

6.4 Referência do Arquivo instruction.h

Arquivo que define a classe Instruction.

```
#include <iostream>
#include <cstdlib>
```

Componentes

- class Instruction

  A classe Instruction: Classe abstrata a qual é base para todos os modos de instrução.

Definições e Macros

- #define **OP_SIZE** 7
- #define **RD_SIZE** 5
- #define **F3_SIZE** 3
- #define **RS1_SIZE** 5
- #define **OP_BEGIN** 0
- #define **RD_BEGIN** 7
- #define **F3_BEGIN** 12
- #define **RS1_BEGIN** 15
- #define **INSTRUCTION_SIZE** 32

6.4.1 Descrição detalhada

Arquivo que define a classe Instruction.

Autor

Victor Emanuel Almeida

Versão

0.1

6.5 Referência do Arquivo instructiontyper.cpp

Arquivo que implementa os métodos da classe InstructionTypeR.

```
#include "instructiontyper.h"
```

6.5.1 Descrição detalhada

Arquivo que implementa os métodos da classe InstructionTypeR.

Autor

Victor Emanuel Almeida

Versão

0.1

6.6 Referência do Arquivo instructiontyper.h

Arquivo que define a classe InstructionTypeR.

```
#include "instruction.h"
```

Componentes

- class InstructionTypeR

  A classe InstructionTypeR:

Definições e Macros

- #define **RS2_BEGIN** 20
- #define **F7_BEGIN** 25

6.6.1 Descrição detalhada

Arquivo que define a classe InstructionTypeR.

Autor

Victor Emanuel Almeida

Versão

0.1

6.7 Referência do Arquivo main.cpp

Arquivo principal.

```
#include "mainwindow.h"
#include "instruction.h"
#include "instructiontyper.h"
#include <QApplication>
#include <QFile>
```

Funções

- int **main** (int argc, char ∗argv[ ])

6.7.1 Descrição detalhada

Arquivo principal.

Autor

Victor Emanuel Almeida

Versão

0.1

6.8 Referência do Arquivo memory.cpp

Arquivo que implementa os métodos da classe Memory.

```
#include "memory.h"
```

6.8.1 Descrição detalhada

Arquivo que implementa os métodos da classe Memory.

Autor

Victor Emanuel Almeida

Versão

0.1

6.9 Referência do Arquivo memory.h

Arquivo que define a classe Memory.

Componentes

- class Memory

  A classe Memory:

Definições e Macros

- #define **MEMORY_SIZE** 1024

6.9.1 Descrição detalhada

Arquivo que define a classe Memory.

Autor

Victor Emanuel Almeida

Versão

0.1

6.10 Referência do Arquivo programcounter.cpp

Arquivo que implementa os métodos da classe ProgramCounter.

```
#include "programcounter.h"
```

6.10.1 Descrição detalhada

Arquivo que implementa os métodos da classe ProgramCounter.

Autor

Victor Emanuel Almeida

Versão

0.1

6.11 Referência do Arquivo programcounter.h

Arquivo que define a classe ProgramCounter.

Componentes

- class ProgramCounter
  A classe ProgramCounter:

6.11.1 Descrição detalhada

Arquivo que define a classe ProgramCounter.

Autor

Victor Emanuel Almeida

Versão

0.1

6.12 Referência do Arquivo registers.cpp

Arquivo que implementa os métodos da classe Registers.

```
#include "registers.h"
```

6.12.1 Descrição detalhada

Arquivo que implementa os métodos da classe Registers.

Autor

Victor Emanuel Almeida

Versão

0.1

6.13 Referência do Arquivo registers.h

Arquivo que define a classe Registers.

Componentes

- class Registers

  A classe Registers:

Definições e Macros

- #define **QUANTITY_REGISTERS** 32

6.13.1 Descrição detalhada

Arquivo que define a classe Registers.

Autor

Victor Emanuel Almeida

Versão

0.1

6.14 Referência do Arquivo simulator.cpp

Arquivo que implementa os métodos da classe Simulator.

```
#include "simulator.h"
```

6.14.1 Descrição detalhada

Arquivo que implementa os métodos da classe Simulator.

Autor

Victor Emanuel Almeida

Versão

0.1

6.15 Referência do Arquivo simulator.h

Arquivo que define a classe Simulator.

```
#include "arithmeticlogicunit.h"
#include "instruction.h"
#include "instructiontyper.h"
#include "memory.h"
#include "programcounter.h"
#include "registers.h"
```

Componentes

- class Simulator

  A classe Simulator:

6.15.1 Descrição detalhada

Arquivo que define a classe Simulator.

Autor

Victor Emanuel Almeida

Versão

0.1

Índice Remissivo