MToy

1.0

Generated by Doxygen 1.8.17

Chapter 1

MIPS Simulator

A toy GUI-aided simulator for MIPS 2000 assembly language.

1.1 Introduction

This is a toy simulator of MIPS 2000 assembly environment for CSC3050 assignment 2. An assembler is integrated into the simulator. The simulator will first invoke the assembler to transform the assembly language into binary code. After that, all binary code will be decoded and transform into predefined structs in `c++`.

When simulation starts, the simulator will maintain the status in the registers and memory (stack/static/heap). All IO operations will be handled by GUI events.

1.2 Language Support

MIPS 2000 language without floating point instructions, coprocessor instructions and pesudo instructions. The data part supports:

- word (array)
- halfword (array)
- byte (array)
- space (array)
- ascii
- asciiz

1.3 Syscall List

Name	Code
PRINT_INT	1
PRINT_STRING	4
READ_INT	5
READ_STRING	8
MMAP	9
EXIT	10
OPEN	13
READ	14
WRITE	15
CLOSE	16
EXIT2	17
FAST_COPY	10000
FILE_OPEN_DIALOG	10001
MUNMAP	10002

1.4 Build

1.4.1 Build Environment

- Linux (64bit, GNU environment, Kernel > 3.0)
- GNU Toolchain (GCC and binary tools, Clang >= 6/GCC >= 7, GCC 9 is recommended)
- CMake (3.5 and above)
- Qt5

1.4.2 Preparation

On Ubuntu 16.04, you can install the toolchains with the following code:

```
sudo apt update
sudo apt install clang-6.0 libomp-dev libomp5
sudo apt-get update
sudo apt-get -y install clang-6.0 clang-6.0 libomp-dev libomp5 qt5-default qtbase5-dev
```

1.4.3 Compilation

On Ubuntu 16.04,

```
mkdir build && cd build
env CC=clang-6.0 CXX=clang-6.0 cmake .. -DCMAKE_BUILD_TYPE=Release
make -j $(nproc)
```

1.4.4 Special Notice

Older GCC on Ubuntu has a bug on `thread_local` linkage, hence it is recommended to use `llvm` toolchain on old distribution.

Chapter 2

The Design Brochure

2.1 The Procedure of Simulation

After all translation is done, the process goes like the following:

- Run the code at the current `PC`
  - Arithmetic Operations are done directly
  - Load/Store Operations are moved to corresponding memory handlers
  - Syscall Operations are moved to corresponding functions
- Update GUI if needed
- If exceptions/faults/exit occurs
- Advance PC if no special jumps happened

2.2 The design of MMAP/MUNMAP Simulation

We use raw `mmap` syscall with `MMAP_32BIT` flags to directly handle the syscall. To maintain the GUI part, we use a tagged statistic red-black tree from GNU's `PBDS` library to record the order of the memory begin positions. Hence, the size and memory blocks can be visualized in a list widget of GUI

2.3 SigFault Capturing

We directly use the signal capture funtions of low-level `c` library to handle the faults on memory. However, if the target operations is too destructive, the simulator will quit directly without popping a dialog.

2.4 Code Deduplication

We use a lot of macros to generate the code for us. In fact, we have designed a simple DSL to help us declare and implement an instruction.

2.5 Store the Instructions

To store the compiled instructions, we use the polymorphic classes. The binary code is "compiled" into unique pointer to the implementation classes. Hence, we do not need to do a lot of branches and dispatches in the running time.

Chapter 3

Todo List

Member JImpl::exec () override

: CHECK WHETHER THIS IS REQUIRED

Chapter 4

Namespace Index

4.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 5

Hierarchical Index

5.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 6

Class Index

6.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 7

File Index

7.1 File List

Here is a list of all files with brief descriptions:

Chapter 8

Namespace Documentation

8.1 _SIM Namespace Reference

Classes

- struct InstrDeleter

Typedefs

- using InstrPtr = std::unique_ptr< InstructionImpl, InstrDeleter >

Functions

- template<typename T , typename ... Args>
  InstrPtr make_unique (Args &&...args)

8.1.1 Detailed Description

This namespace contains some workarounds to provide back support

8.1.2 Typedef Documentation

8.1.2.1 InstrPtr

```
using _SIM::InstrPtr = typedef std::unique_ptr<InstructionImpl, InstrDeleter>
```

Alisa of the unique_ptr of instruction implementations

8.1.3 Function Documentation

8.1.3.1 make_unique()

```
template<typename T , typename ...  Args>
InstrPtr _SIM::make_unique (
             Args &&...  args )
```

C++ 11 do not have make_unique. This functions works as an subtitution.

Template Parameters

T	data type
Args	initialization args type

Parameters

args	initialization args

Returns

the unique ptr

8.2 Ui Namespace Reference

8.2.1 Detailed Description

The Qt specified namespace for UI components

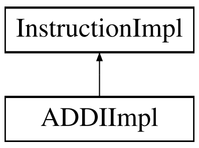
Chapter 9

Class Documentation

9.1 ADDIImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for ADDIImpl:



Public Member Functions

- ADDIImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.1.1 Constructor & Destructor Documentation

9.1.1.1 ADDIImpl()

```
ADDIImpl::ADDIImpl (
            Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.1.2 Member Function Documentation

9.1.2.1 exec()

```
void ADDIImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

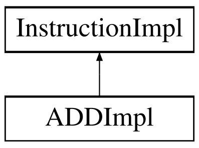
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.2 ADDImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for ADDImpl:



Public Member Functions

- ADDImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.2.1 Constructor & Destructor Documentation

9.2.1.1 ADDImpl()

```
ADDImpl::ADDImpl (
            Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.2.2 Member Function Documentation

9.2.2.1 exec()

```
void ADDImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

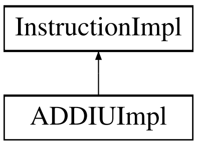
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.3 ADDIUImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for ADDIUImpl:



Public Member Functions

- ADDIUImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.3.1 Constructor & Destructor Documentation

9.3.1.1 ADDIUImpl()

```
ADDIUImpl::ADDIUImpl (
                Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.3.2 Member Function Documentation

9.3.2.1 exec()

```
void ADDIUImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

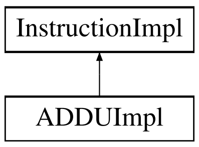
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.4 ADDUImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for ADDUImpl:



Public Member Functions

- ADDUImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.4.1 Constructor & Destructor Documentation

9.4.1.1 ADDUImpl()

```
ADDUImpl::ADDUImpl (
            Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.4.2 Member Function Documentation

9.4.2.1 exec()

```
void ADDUImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

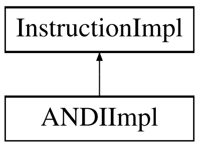
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.5 ANDIImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for ANDIImpl:



Public Member Functions

- ANDIImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.5.1 Constructor & Destructor Documentation

9.5.1.1 ANDIImpl()

```
ANDIImpl::ANDIImpl (
             Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.5.2 Member Function Documentation

9.5.2.1 exec()

```
void ANDIImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

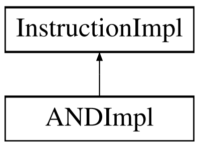
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.6 ANDImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for ANDImpl:



Public Member Functions

- ANDImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.6.1 Constructor & Destructor Documentation

9.6.1.1 ANDImpl()

```
ANDImpl::ANDImpl (
            Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.6.2 Member Function Documentation

9.6.2.1 exec()

```
void ANDImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

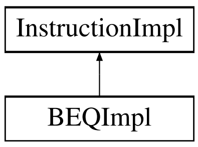
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.7 BEQImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for BEQImpl:



Public Member Functions

- BEQImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.7.1 Constructor & Destructor Documentation

9.7.1.1 BEQImpl()

```
BEQImpl::BEQImpl (
             Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.7.2 Member Function Documentation

9.7.2.1 exec()

```
void BEQImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

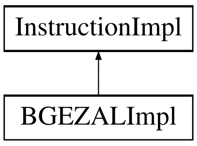
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.8 BGEZALImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for BGEZALImpl:



Public Member Functions

- BGEZALImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.8.1 Constructor & Destructor Documentation

9.8.1.1 BGEZALImpl()

```
BGEZALImpl::BGEZALImpl (
             Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.8.2 Member Function Documentation

9.8.2.1 exec()

```
void BGEZALImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

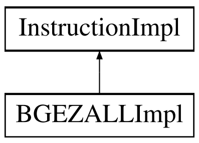
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.9 BGEZALLImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for BGEZALLImpl:



Public Member Functions

- BGEZALLImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.9.1 Constructor & Destructor Documentation

9.9.1.1 BGEZALLImpl()

```
BGEZALLImpl::BGEZALLImpl (
            Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.9.2 Member Function Documentation

9.9.2.1 exec()

```
void BGEZALLImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

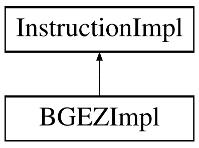
The documentation for this struct was generated from the following file:

- src/instruction_impl.h

9.10 BGEZImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for BGEZImpl:



Public Member Functions

- BGEZImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.10.1 Constructor & Destructor Documentation

9.10.1.1 BGEZImpl()

```
BGEZImpl::BGEZImpl (
            Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.10.2 Member Function Documentation

9.10.2.1 exec()

```
void BGEZImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

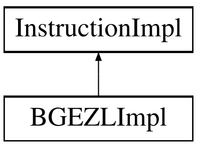
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.11 BGEZLImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for BGEZLImpl:



Public Member Functions

- BGEZLImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.11.1 Constructor & Destructor Documentation

9.11.1.1 BGEZLImpl()

```
BGEZLImpl::BGEZLImpl (
                Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.11.2 Member Function Documentation

9.11.2.1 exec()

```
void BGEZLImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

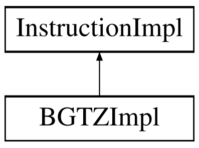
The documentation for this struct was generated from the following file:

- src/instruction_impl.h

9.12 BGTZImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for BGTZImpl:



Public Member Functions

- BGTZImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.12.1 Constructor & Destructor Documentation

9.12.1.1 BGTZImpl()

```
BGTZImpl::BGTZImpl (
            Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.12.2 Member Function Documentation

9.12.2.1 exec()

```
void BGTZImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

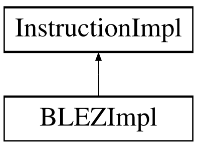
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.13 BLEZImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for BLEZImpl:



Public Member Functions

- BLEZImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.13.1 Constructor & Destructor Documentation

9.13.1.1 BLEZImpl()

```
BLEZImpl::BLEZImpl (
                Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.13.2 Member Function Documentation

9.13.2.1 exec()

```
void BLEZImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

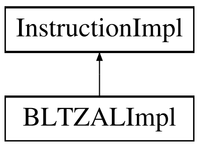
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.14 BLTZALImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for BLTZALImpl:



Public Member Functions

- BLTZALImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.14.1 Constructor & Destructor Documentation

9.14.1.1 BLTZALImpl()

```
BLTZALImpl::BLTZALImpl (
             Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.14.2 Member Function Documentation

9.14.2.1 exec()

```
void BLTZALImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

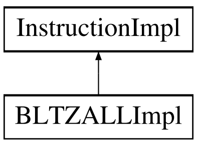
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.15 BLTZALLImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for BLTZALLImpl:



Public Member Functions

- BLTZALLImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.15.1 Constructor & Destructor Documentation

9.15.1.1 BLTZALLImpl()

```
BLTZALLImpl::BLTZALLImpl (
             Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.15.2 Member Function Documentation

9.15.2.1 exec()

```
void BLTZALLImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

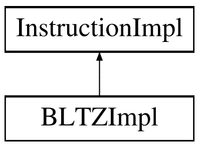
The documentation for this struct was generated from the following file:

- src/instruction_impl.h

9.16 BLTZImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for BLTZImpl:



Public Member Functions

- BLTZImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.16.1 Constructor & Destructor Documentation

9.16.1.1 BLTZImpl()

```
BLTZImpl::BLTZImpl (
            Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.16.2 Member Function Documentation

9.16.2.1 exec()

```
void BLTZImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

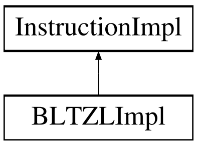
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.17 BLTZLImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for BLTZLImpl:



Public Member Functions

- BLTZLImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.17.1 Constructor & Destructor Documentation

9.17.1.1 BLTZLImpl()

```
BLTZLImpl::BLTZLImpl (
             Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.17.2 Member Function Documentation

9.17.2.1 exec()

```
void BLTZLImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

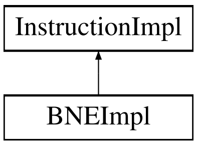
The documentation for this struct was generated from the following file:

- src/instruction_impl.h

9.18 BNEImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for BNEImpl:



Public Member Functions

- BNEImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.18.1 Constructor & Destructor Documentation

9.18.1.1 BNEImpl()

```
BNEImpl::BNEImpl (
            Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.18.2 Member Function Documentation

9.18.2.1 exec()

```
void BNEImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

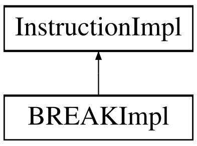
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.19 BREAKImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for BREAKImpl:



Public Member Functions

- BREAKImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.19.1 Constructor & Destructor Documentation

9.19.1.1 BREAKImpl()

```
BREAKImpl::BREAKImpl (
            Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.19.2 Member Function Documentation

9.19.2.1 exec()

```
void BREAKImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

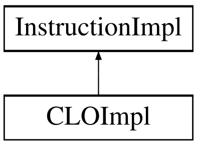
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.20 CLOImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for CLOImpl:



Public Member Functions

- CLOImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.20.1 Constructor & Destructor Documentation

9.20.1.1 CLOImpl()

```
CLOImpl::CLOImpl (
            Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.20.2 Member Function Documentation

9.20.2.1 exec()

```
void CLOImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

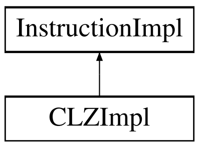
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.21 CLZImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for CLZImpl:



Public Member Functions

- CLZImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.21.1 Constructor & Destructor Documentation

9.21.1.1 CLZImpl()

```
CLZImpl::CLZImpl (
            Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.21.2 Member Function Documentation

9.21.2.1 exec()

```
void CLZImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

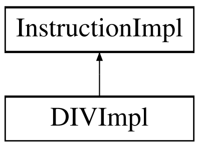
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.22 DIVImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for DIVImpl:



Public Member Functions

- DIVImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.22.1 Constructor & Destructor Documentation

9.22.1.1 DIVImpl()

```
DIVImpl::DIVImpl (
            Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.22.2 Member Function Documentation

9.22.2.1 exec()

```
void DIVImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

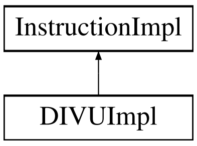
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.23 DIVUImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for DIVUImpl:



Public Member Functions

- DIVUImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.23.1 Constructor & Destructor Documentation

9.23.1.1 DIVUImpl()

```
DIVUImpl::DIVUImpl (
            Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.23.2 Member Function Documentation

9.23.2.1 exec()

```
void DIVUImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

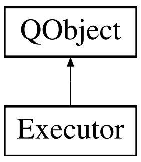
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.24 Executor Class Reference

```
#include <executor.h>
```

Inheritance diagram for Executor:



Public Slots

- void next ()
- void exit (int code=0)

Signals

- void finished ()

Public Attributes

- std::vector< _SIM::InstrPtr > impls

    a set of pointers to the translated result of the machine code.

Static Public Attributes

- static MainWindow ∗ mainW = nullptr

  a pointer back to the MainWindow, provides APIs on the UI components.

9.24.1 Detailed Description

Executor is a helper class for us to store compiled instructions and enable slots for automatic execution.

9.24.2 Member Function Documentation

9.24.2.1 exit

```
void Executor::exit (
            int code = 0 )  [slot]
```

Exit the execution with the given code. (Information is shown) It will also emit a finish signal on the closing of the information dialog.

Parameters

code	return code

9.24.2.2 finished

```
void Executor::finished ( )  [signal]
```

This signal will be emitted when the execution is finished.

9.24.2.3 next

```
void Executor::next ( )  [slot]
```

When this slot is invoked, it will execute the instruction at the current PC position. When PC touches the bottom line, it will emit a finish signal.

Attention

`std::runtime_error` and `SIGSEGV` will be caught within the range.

9.24.3 Member Data Documentation

9.24.3.1 impls

```
std::vector<_SIM::InstrPtr> Executor::impls
```

a set of pointers to the translated result of the machine code.

9.24.3.2 mainW

```
MainWindow * Executor::mainW = nullptr  [static]
```

a pointer back to the MainWindow, provides APIs on the UI components.

The documentation for this class was generated from the following files:

- src/executor.h
- src/executor.cpp
- src/mainwindow.cpp

9.25 Heap Class Reference

```
#include <heap.h>
```

Public Member Functions

- void clear ()
- ∼Heap ()
- uint32_t alloc (size_t n)
- void dealloc (uint32_t addr)
- size_t order (uint32_t addr)

Public Attributes

- size_t size = 0

  The size of current heap.

Private Attributes

- TreeSet< uint32_t, size_t > mapping

  Used to store the block size of each allocation.

9.25.1 Detailed Description

Heap help us to record and manage heap storage

9.25.2 Constructor & Destructor Documentation

9.25.2.1 ∼Heap()

```
Heap::∼Heap ( )
```

Deconstruct the heap, will do the same thing as clear();

9.25.3 Member Function Documentation

9.25.3.1 alloc()

```
uint32_t Heap::alloc (
             size_t n )
```

Alloc a new memory block.

Parameters

n	memory block size

Returns

the address of the newly allocated memory block

9.25.3.2 clear()

```
void Heap::clear ( )
```

Clear all heap content

9.25.3.3 dealloc()

```
void Heap::dealloc (
             uint32_t addr )
```

Dealloc a memory block

Parameters

addr	the start address of the memory block.

9.25.3.4 order()

```
size_t Heap::order (
            uint32_t addr )
```

Check the order of an address. This is used in UI display to check which block to remove from the list.

Parameters

addr	the address to check

Returns

the order of the block

9.25.4 Member Data Documentation

9.25.4.1 mapping

```
TreeSet<uint32_t, size_t> Heap::mapping  [private]
```

Used to store the block size of each allocation.

9.25.4.2 size

```
size_t Heap::size = 0
```

The size of current heap.

The documentation for this class was generated from the following files:

- src/heap.h
- src/heap.cpp

9.26 _SIM::InstrDeleter Struct Reference

```
#include <global.h>
```

Public Member Functions

- void operator() (InstructionImpl ∗t)

9.26.1 Detailed Description

The deleter for the unique_ptr of instruction implementations

9.26.2 Member Function Documentation

9.26.2.1 operator()()

```
void _SIM::InstrDeleter::operator() (
             InstructionImpl ∗ t )  [inline]
```

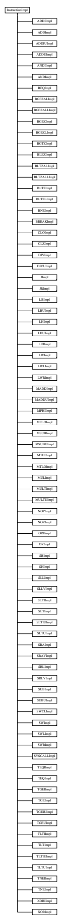
The documentation for this struct was generated from the following file:

- src/global.h

9.27 InstructionImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for InstructionImpl:



Public Member Functions

- virtual void exec ()=0
- InstructionImpl (Instruction instr)

Public Attributes

- Instruction instr

  the instruction value.

Static Public Attributes

- static MainWindow $*$ mainW = nullptr

9.27.1 Detailed Description

The base class of the instruction implementations. All subclasses must implement the `exec()` function, and this function decides the behavior of each construction.

9.27.2 Constructor & Destructor Documentation

9.27.2.1 InstructionImpl()

```
InstructionImpl::InstructionImpl (
            Instruction instr )  [explicit]
```

Construct the base class: InstructionImpl.

Attention

This function will set the instruction and should be invoked by all subclasses in some way.

Parameters

instr	the instruction value

9.27.3 Member Function Documentation

9.27.3.1 exec()

```
virtual void InstructionImpl::exec ( )  [pure virtual]
```

The purely virtual function `exec()` is to be overwritten in the subclasses.

Implemented in LLImpl, SCImpl, LWRImpl, LWLImpl, SWRImpl, SWLImpl, TLTUImpl, TLTImpl, TGEUImpl, TGEImpl, TNEImpl, TEQImpl, MSUBUImpl, MSUBImpl, MADDUImpl, MADDImpl, MULImpl, CLZImpl, CLOImpl, BGEZALLImpl, BLTZALLImpl, BGEZALImpl, BLTZALImpl, TNEIImpl, TEQIImpl, TLTIUImpl, TLTIImpl, TGEIUImpl, TGEIImpl, BGEZLImpl, BLTZLImpl, XORImpl, SYSCALLImpl, SUBUImpl, SUBImpl, SRLVImpl, SRLImpl, SRAVImpl, SRAImpl, SLTUImpl, SLTImpl, SLLVImpl, SLLImpl, ORImpl, NORImpl, MULTUImpl, MULTImpl, MTLOImpl, MTHIImpl, MFLOImpl, MFHIImpl, JALRImpl, JRImpl, DIVUImpl, DIVImpl, BREAKImpl, ANDImpl, ADDUImpl, ADDImpl, XORIImpl, SWCLImpl, SWImpl, SHImpl, SLTIUImpl, SLTIImpl, SBImpl, ORIImpl, LWImpl, LUIImpl, LHUImpl, LHImpl, LBUImpl, LBImpl, BNEImpl, BLTZImpl, BLEZImpl, BGTZImpl, BGEZImpl, BEQImpl, ANDIImpl, ADDIUImpl, ADDIImpl, JALImpl, JImpl, and NOPImpl.

9.27.4 Member Data Documentation

9.27.4.1 instr

```
Instruction InstructionImpl::instr
```

the instruction value.

9.27.4.2 mainW

```
MainWindow * InstructionImpl::mainW = nullptr  [static]
```

Static pointer back to the main windows. This pointer allow us to interact with the ui components.

Attention

This pointer should never be modified in any situation except for the setting happens during the initialization MainWindow

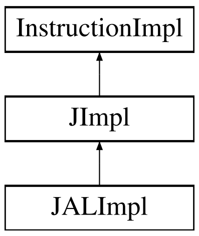
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.28 JALImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for JALImpl:



Public Member Functions

- JALImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.28.1 Constructor & Destructor Documentation

9.28.1.1 JALImpl()

```
JALImpl::JALImpl (
            Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.28.2 Member Function Documentation

9.28.2.1 exec()

```
void JALImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Reimplemented from JImpl.

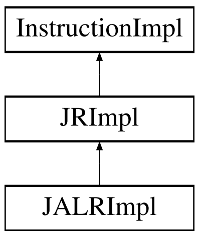
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.29 JALRImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for JALRImpl:



Public Member Functions

- JALRImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.29.1 Constructor & Destructor Documentation

9.29.1.1 JALRImpl()

```
JALRImpl::JALRImpl (
            Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.29.2 Member Function Documentation

9.29.2.1 exec()

```
void JALRImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Reimplemented from JRImpl.

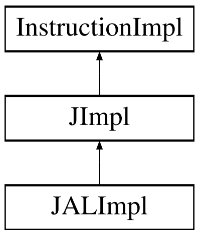
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.30 JImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for JImpl:



Public Member Functions

- JImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.30.1 Constructor & Destructor Documentation

9.30.1.1 JImpl()

```
JImpl::JImpl (
             Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.30.2 Member Function Documentation

9.30.2.1 exec()

```
void JImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Todo : CHECK WHETHER THIS IS REQUIRED

Implements InstructionImpl.

Reimplemented in JALImpl.

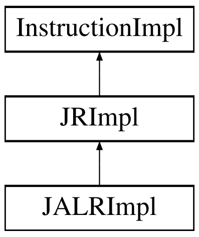
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.31 JRImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for JRImpl:



Public Member Functions

- JRImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.31.1 Constructor & Destructor Documentation

9.31.1.1 JRImpl()

```
JRImpl::JRImpl (
             Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.31.2 Member Function Documentation

9.31.2.1 exec()

```
void JRImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

Reimplemented in JALRImpl.

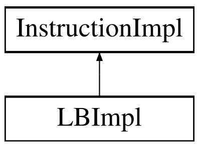
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.32 LBImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for LBImpl:



Public Member Functions

- LBImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.32.1 Constructor & Destructor Documentation

9.32.1.1 LBImpl()

```
LBImpl::LBImpl (
            Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.32.2 Member Function Documentation

9.32.2.1 exec()

```
void LBImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

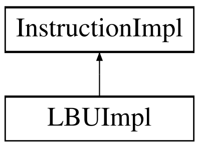
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.33 LBUImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for LBUImpl:



Public Member Functions

- LBUImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.33.1 Constructor & Destructor Documentation

9.33.1.1 LBUImpl()

```
LBUImpl::LBUImpl (
            Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.33.2 Member Function Documentation

9.33.2.1 exec()

```
void LBUImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

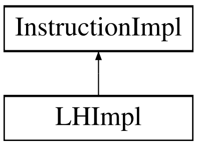
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.34 LHImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for LHImpl:



Public Member Functions

- LHImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.34.1 Constructor & Destructor Documentation

9.34.1.1 LHImpl()

```
LHImpl::LHImpl (
            Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.34.2 Member Function Documentation

9.34.2.1 exec()

```
void LHImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

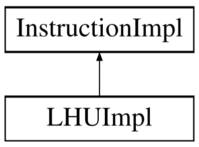
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.35 LHUImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for LHUImpl:



Public Member Functions

- LHUImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.35.1 Constructor & Destructor Documentation

9.35.1.1 LHUImpl()

```
LHUImpl::LHUImpl (
            Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.35.2 Member Function Documentation

9.35.2.1 exec()

```
void LHUImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

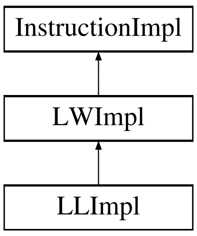
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.36 LLImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for LLImpl:



Public Member Functions

- LLImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.36.1 Constructor & Destructor Documentation

9.36.1.1 LLImpl()

```
LLImpl::LLImpl (
            Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.36.2 Member Function Documentation

9.36.2.1 exec()

```
void LLImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Attention

we have no difference between atomic ones and unatomic ones.

Reimplemented from LWImpl.

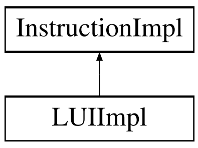
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.37 LUIImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for LUIImpl:



Public Member Functions

- LUIImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.37.1 Constructor & Destructor Documentation

9.37.1.1 LUIImpl()

```
LUIImpl::LUIImpl (
            Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.37.2 Member Function Documentation

9.37.2.1 exec()

```
void LUIImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

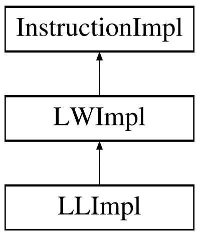
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.38 LWImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for LWImpl:



Public Member Functions

- LWImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.38.1 Constructor & Destructor Documentation

9.38.1.1 LWImpl()

```
LWImpl::LWImpl (
                Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.38.2 Member Function Documentation

9.38.2.1 exec()

```
void LWImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

Reimplemented in LLImpl.

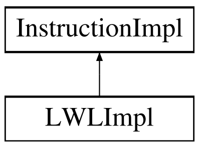
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.39 LWLImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for LWLImpl:



Public Member Functions

- LWLImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.39.1 Constructor & Destructor Documentation

9.39.1.1 LWLImpl()

```
LWLImpl::LWLImpl (
            Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.39.2 Member Function Documentation

9.39.2.1 exec()

```
void LWLImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

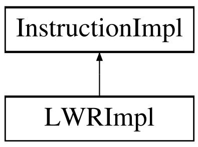
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.40 LWRImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for LWRImpl:



Public Member Functions

- LWRImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.40.1 Constructor & Destructor Documentation

9.40.1.1 LWRImpl()

```
LWRImpl::LWRImpl (
            Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.40.2 Member Function Documentation

9.40.2.1 exec()

```
void LWRImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

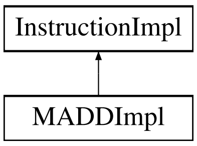
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.41 MADDImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for MADDImpl:



Public Member Functions

- MADDImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.41.1 Constructor & Destructor Documentation

9.41.1.1 MADDImpl()

```
MADDImpl::MADDImpl (
             Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.41.2 Member Function Documentation

9.41.2.1 exec()

```
void MADDImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

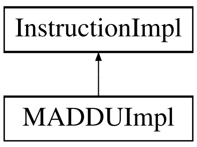
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.42 MADDUImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for MADDUImpl:



Public Member Functions

- MADDUImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.42.1 Constructor & Destructor Documentation

9.42.1.1 MADDUImpl()

```
MADDUImpl::MADDUImpl (
            Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.42.2 Member Function Documentation

9.42.2.1 exec()

```
void MADDUImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

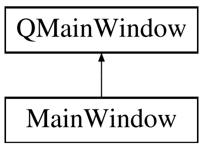
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.43 MainWindow Class Reference

```
#include <mainwindow.h>
```

Inheritance diagram for MainWindow:



Public Member Functions

- MainWindow (QWidget ∗parent=nullptr)
- ∼MainWindow () override
- void showWarning (QString str)
- void updateRegValue (int no, uint32_t value, const QBrush &brush=QBrush("red"), bool init=false)
- void updateProgramCounter (size_t value)
- uint32_t allocHeap (size_t size)
- void deallocHeap (size_t addr)
- template<class T >
  T fetchHeap (uint32_t addr)
- template<class T >
  void editHeap (uint32_t addr, T value)

- template< class T >
  void edit (uint32_t addr, T value)
- void increaseStack (size_t n)
- void decreaseStack (size_t n)
- template< class T >
  T & fetchStack (uint32_t addr)
- template< class T >
  void editStack (uint32_t addr, T value)
- MemoryType memoryType (uint32_t addr)
- void updateStack (uint32_t addr, size_t size)
- void updateLow (uint32_t value)
- void updateHigh (uint32_t value)
- void updateAcc (uint64_t value)
- void translateAll ()
- void resetAll ()
- template< class T >
  T ∗ getRealAddr (uint32_t addr)
- void handleSyscall ()

Public Attributes

- union {
  uint64_t all
  stands for the whole accumulator
  struct **LOW_HIGH** {
  uint32_t low
  stands for the low part of the accumulator
  uint32_t high
  stands for the high part of the accumulator
  } part
  } ACC

- std::array< char,(1024 ∗KiB) > frame
- Ui::MainWindow ∗ ui

  The pointer to the Qt UI components.
- Executor ∗ executor = nullptr

  The pointer to the executor.
- bool advanceCounter = true
- QTimer timer

  A timer to ignite periodical signals on execution.
- std::vector< Instruction > instructions {}

  The vector for instruction storage.
- Stack stack

  The stack operation simulator.
- Heap heap

  The heap operation simulator.
- uint32_t REGS [32] = {}

  The array for all registers.
- size_t PC = 0

  The program counter.

Private Slots

- void on_aboutButton_clicked ()
- void on_openButton_clicked ()
- void on_translateButton_clicked ()
- void on_executeButton_clicked ()
- void on_stepButton_clicked ()
- void on_resetButton_clicked ()
- void on_stopButton_clicked ()
- void on_pushButton_clicked ()

9.43.1 Detailed Description

MainWindow of the GUI program. It also contains the following component:

- a heap to simulate heap allocations
- a stack to simulate stack operations
- an executor to store and run simulation steps

  Attention

  we keep all fields in public to reduce the complexity of development

9.43.2 Constructor & Destructor Documentation

9.43.2.1 MainWindow()

```
MainWindow::MainWindow (
            QWidget * parent = nullptr )  [explicit]
```

construct the MainWindow

Parameters

parent	the parent widget, set as nullptr by default.

On initialization, it will:

- adjust the scale factor of the tables and lists.
- disable execution related buttons
- set initial values of the registers and other simulation storage
- capture all `SIGSEGV` signals and throw errors on caught

9.43.2.2 ∼MainWindow()

```
MainWindow::∼MainWindow ( ) [override]
```

deconstruct the MainWindows, this will:

- delete all ui components in the tree
- delete the executor
- the stack and heap space are freed automatically (since they are normal data members)

9.43.3 Member Function Documentation

9.43.3.1 allocHeap()

```
uint32_t MainWindow::allocHeap (
            size_t size )
```

Allocate some new memory on heap. This operation will update the UI components in the same time.

Attention

this will truly invoke the `mmap` syscall in your bare metal machine. For x64_64 linux target, we are using `MA↩ P_32BIT` as the flag to force the system give an available address in the first 2GiB area. In the simulation, we are actually storing the size in an associate set. The allocation and deallocation operations are much slower than the real world, but as it is just a simulator with heavy UI animations, the cost is bearable.

Parameters

size	the required size of the allocation

Returns

the pointer to the beginning of the newly allocated memory.

9.43.3.2 deallocHeap()

```
void MainWindow::deallocHeap (
            size_t addr )
```

Deallocate some memory on heap. This operation will update the UI components in the same time.

Parameters

addr	the address to dealloc

Attention

In our simulator, the following things are required:

- the pointer should point to the start position of the memory block
- the memory is managed by the heap

9.43.3.3 decreaseStack()

```
void MainWindow::decreaseStack (
            size_t n )
```

Decrease the stack size by the given amount. This will update the UI at the same time.

Parameters

n	the amount to decrease

9.43.3.4 edit()

```
template<class T >
void MainWindow::edit (
            uint32_t addr,
            T value )
```

9.43.3.5 editHeap()

```
template<class T >
void MainWindow::editHeap (
            uint32_t addr,
            T value )
```

Modify the target word cell on heap.

Template Parameters

T	T primitive word type

Parameters

addr	the target address
value	the new value

9.43.3.6 editStack()

```
template<class T >
void MainWindow::editStack (
             uint32_t addr,
             T value )
```

Edit the stack value and update the ui display

Template Parameters

T	primitive type

Parameters

addr	virtual address on stack
value	the value to be set

9.43.3.7 fetchHeap()

```
template<class T >
T MainWindow::fetchHeap (
             uint32_t addr )
```

Read a target word from heap.

Template Parameters

T	primitive word type

Parameters

addr	the target address

Returns

the required word value

9.43.3.8 fetchStack()

```
template<class T >
T & MainWindow::fetchStack (
             uint32_t addr )
```

get the reference of the T value at an address on stack

Template Parameters

T	primitive type

Parameters

addr	virtual address on stack

Returns

the reference to the target data

9.43.3.9 getRealAddr()

```
template<class T >
T * MainWindow::getRealAddr (
             uint32_t addr )
```

get the real address of the memory

Template Parameters

T	data type

Parameters

addr	simulated memory

Returns

the real address

9.43.3.10 handleSyscall()

```
void MainWindow::handleSyscall ( )
```

handle syscall based on the value in the target registers

9.43.3.11 increaseStack()

```
void MainWindow::increaseStack (
             size_t n )
```

Increase the stack size by the given amount. This will update the UI at the same time.

Parameters

n	the amount to increase

9.43.3.12 memoryType()

```
MemoryType MainWindow::memoryType (
             uint32_t addr )
```

Check whether a address is currently within the stack range.

Parameters

addr	the address to check

Returns

the checking result

9.43.3.13 on_aboutButton_clicked

```
void MainWindow::on_aboutButton_clicked ( )  [private], [slot]
```

The slot for the `aboutButton`, on click, it will pop up a about window with icon and basic information.

9.43.3.14 on_executeButton_clicked

```
void MainWindow::on_executeButton_clicked ( )  [private], [slot]
```

The slot for the `executionButton`, on click, it will start the execution. It will also:

- set the time interval set in the `delay`
- disable buttons and inputs that will change the execution behavior
- enable the `stopButton`

9.43.3.15 on_openButton_clicked

```
void MainWindow::on_openButton_clicked ( )  [private], [slot]
```

The slot for the `openButton`, on click, it will open a file chooser for the user to choose the compiled asm file.

Attention

Notice that if the file format is not correct, an error will be raised will a warning dialog.

9.43.3.16 on_pushButton_clicked

```
void MainWindow::on_pushButton_clicked ( )  [private], [slot]
```

9.43.3.17 on_resetButton_clicked

```
void MainWindow::on_resetButton_clicked ( )  [private], [slot]
```

The slot for the `stepButton`, on click, it will reset all state.

Attention

Noitce that `reset` will also clear all instructions, you will need to open your file again

9.43.3.18 on_stepButton_clicked

```
void MainWindow::on_stepButton_clicked ( )  [private], [slot]
```

The slot for the `stepButton`, on click, it will do a single line execution.

9.43.3.19 on_stopButton_clicked

```
void MainWindow::on_stopButton_clicked ( )  [private], [slot]
```

The slot for the `stepButton`, on click, it will stop the execution. The `PC` will remain unchanged and the execution can be resumed.

9.43.3.20 on_translateButton_clicked

```
void MainWindow::on_translateButton_clicked ( )  [private], [slot]
```

The slot for the `translateButton`, on click, it will start the parallel translation. After translation, it will enable those execution related buttons. It will free the current executor and try to create a new one.

Attention

Notice that if a binary string is not recognized, the translation procedure will not stop immediately. It will store the error and give a whole summary after all lines are checked and translated.

9.43.3.21 resetAll()

```
void MainWindow::resetAll ( )
```

Clean all data

9.43.3.22 showWarning()

```
void MainWindow::showWarning (
             QString str )
```

Pop up a warning dialog.

Parameters

str	the warning message

9.43.3.23 translateAll()

```
void MainWindow::translateAll ( )
```

Translate all instructions

9.43.3.24 updateAcc()

```
void MainWindow::updateAcc (
             uint64_t value )
```

Update the entire accumulator

Parameters

value	new value

9.43.3.25 updateHigh()

```
void MainWindow::updateHigh (
             uint32_t value )
```

Update the higher bits in the accumulator

Parameters

value	new value

9.43.3.26 updateLow()

```
void MainWindow::updateLow (
             uint32_t value )
```

Update the lower bits in the accumulator

Parameters

value	new value

9.43.3.27 updateProgramCounter()

```
void MainWindow::updateProgramCounter (
             size_t value )
```

Update the program counter.

Parameters

value	new value

9.43.3.28 updateRegValue()

```
void MainWindow::updateRegValue (
             int no,
             uint32_t value,
             const QBrush & brush = QBrush("red"),
             bool init = false )
```

Update the target register value.

Parameters

no	register number
value	new value
brush	the texture brush (to indicate there is a modification on this register)
init	whether this is a initialization (may lead to different behavior when updating the stack pointer)

9.43.3.29 updateStack()

```
void MainWindow::updateStack (
             uint32_t addr,
             size_t size )
```

Update the value in the target address in GUI.

Parameters

addr	address to update
size	number of bytes to update

9.43.4 Member Data Documentation

9.43.4.1 ACC

```
union { ...  } MainWindow::ACC
```

The accumulator for multiplication related operations

9.43.4.2 advanceCounter

```
bool MainWindow::advanceCounter = true
```

An indicator to check whether we should advance the counter or not. Used for jumping related instructions.

9.43.4.3 all

```
uint64_t MainWindow::all
```

stands for the whole accumulator

9.43.4.4 executor

```
Executor* MainWindow::executor = nullptr
```

The pointer to the executor.

9.43.4.5 frame

```
std::array<char, ( 1024 * KiB) > MainWindow::frame
```

9.43.4.6 heap

```
Heap MainWindow::heap
```

The heap operation simulator.

9.43.4.7 high

```
uint32_t MainWindow::high
```

stands for the high part of the accumulator

9.43.4.8 instructions

```
std::vector<Instruction> MainWindow::instructions {}
```

The vector for instruction storage.

9.43.4.9 low

```
uint32_t MainWindow::low
```

stands for the low part of the accumulator

9.43.4.10 part

```
struct { ...  } ::LOW_HIGH MainWindow::part
```

9.43.4.11 PC

```
size_t MainWindow::PC = 0
```

The program counter.

9.43.4.12 REGS

```
uint32_t MainWindow::REGS[32] = {}
```

The array for all registers.

9.43.4.13 stack

```
Stack MainWindow::stack
```

The stack operation simulator.

9.43.4.14 timer

```
QTimer MainWindow::timer
```

A timer to ignite periodical signals on execution.

9.43.4.15 ui

```
Ui::MainWindow* MainWindow::ui
```

The pointer to the Qt UI components.

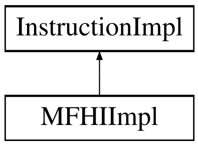
The documentation for this class was generated from the following files:

- src/mainwindow.h
- src/mainwindow.cpp
- src/mainwindow.ipp

9.44 MFHIImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for MFHIImpl:



Public Member Functions

- MFHIImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.44.1 Constructor & Destructor Documentation

9.44.1.1 MFHIImpl()

```
MFHIImpl::MFHIImpl (
            Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.44.2 Member Function Documentation

9.44.2.1 exec()

```
void MFHIImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

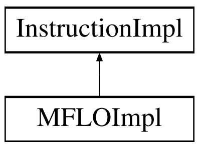
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.45 MFLOImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for MFLOImpl:



Public Member Functions

- MFLOImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.45.1 Constructor & Destructor Documentation

9.45.1.1 MFLOImpl()

```
MFLOImpl::MFLOImpl (
            Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.45.2 Member Function Documentation

9.45.2.1 exec()

```
void MFLOImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

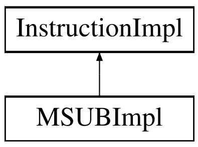
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.46 MSUBImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for MSUBImpl:



Public Member Functions

- MSUBImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.46.1 Constructor & Destructor Documentation

9.46.1.1 MSUBImpl()

```
MSUBImpl::MSUBImpl (
            Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.46.2 Member Function Documentation

9.46.2.1 exec()

```
void MSUBImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

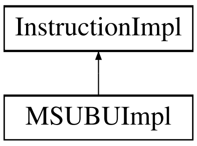
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.47 MSUBUImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for MSUBUImpl:



Public Member Functions

- MSUBUImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.47.1 Constructor & Destructor Documentation

9.47.1.1 MSUBUImpl()

```
MSUBUImpl::MSUBUImpl (
            Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.47.2 Member Function Documentation

9.47.2.1 exec()

```
void MSUBUImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

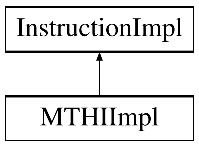
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.48 MTHIImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for MTHIImpl:



Public Member Functions

- MTHIImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.48.1 Constructor & Destructor Documentation

9.48.1.1 MTHIImpl()

```
MTHIImpl::MTHIImpl (
             Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.48.2 Member Function Documentation

9.48.2.1 exec()

```
void MTHIImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

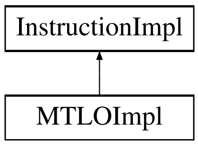
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.49 MTLOImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for MTLOImpl:



Public Member Functions

- MTLOImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.49.1 Constructor & Destructor Documentation

9.49.1.1 MTLOImpl()

```
MTLOImpl::MTLOImpl (
            Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.49.2 Member Function Documentation

9.49.2.1 exec()

```
void MTLOImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

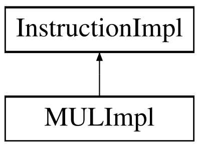
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.50 MULImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for MULImpl:



Public Member Functions

- MULImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.50.1 Constructor & Destructor Documentation

9.50.1.1 MULImpl()

```
MULImpl::MULImpl (
        Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.50.2 Member Function Documentation

9.50.2.1 exec()

```
void MULImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

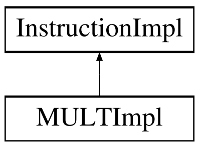
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.51 MULTImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for MULTImpl:



Public Member Functions

- MULTImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.51.1 Constructor & Destructor Documentation

9.51.1.1 MULTImpl()

```
MULTImpl::MULTImpl (
             Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.51.2 Member Function Documentation

9.51.2.1 exec()

```
void MULTImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

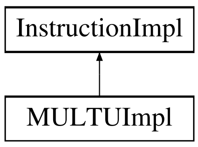
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.52 MULTUImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for MULTUImpl:



Public Member Functions

- MULTUImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.52.1 Constructor & Destructor Documentation

9.52.1.1 MULTUImpl()

```
MULTUImpl::MULTUImpl (
            Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.52.2 Member Function Documentation

9.52.2.1 exec()

```
void MULTUImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

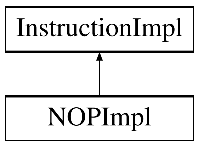
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.53 NOPImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for NOPImpl:



Public Member Functions

- NOPImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.53.1 Constructor & Destructor Documentation

9.53.1.1 NOPImpl()

```
NOPImpl::NOPImpl (
            Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.53.2 Member Function Documentation

9.53.2.1 exec()

```
void NOPImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

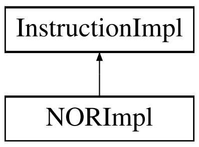
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.54 NORImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for NORImpl:



Public Member Functions

- NORImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.54.1 Constructor & Destructor Documentation

9.54.1.1 NORImpl()

```
NORImpl::NORImpl (
            Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.54.2 Member Function Documentation

9.54.2.1 exec()

```
void NORImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

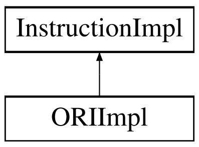
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.55 ORIImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for ORIImpl:



Public Member Functions

- ORIImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.55.1 Constructor & Destructor Documentation

9.55.1.1 ORIImpl()

```
ORIImpl::ORIImpl (
            Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.55.2 Member Function Documentation

9.55.2.1 exec()

```
void ORIImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

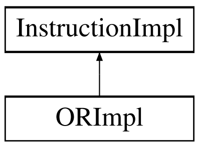
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.56 ORImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for ORImpl:



Public Member Functions

- ORImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.56.1 Constructor & Destructor Documentation

9.56.1.1 ORImpl()

```
ORImpl::ORImpl (
            Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.56.2 Member Function Documentation

9.56.2.1 exec()

```
void ORImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

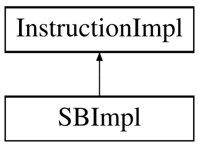
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.57 SBImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for SBImpl:



Public Member Functions

- SBImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.57.1 Constructor & Destructor Documentation

9.57.1.1 SBImpl()

```
SBImpl::SBImpl (
            Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.57.2 Member Function Documentation

9.57.2.1 exec()

```
void SBImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

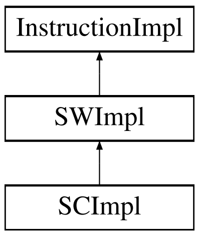
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.58 SCImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for SCImpl:



Public Member Functions

- SCImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.58.1 Constructor & Destructor Documentation

9.58.1.1 SCImpl()

```
SCImpl::SCImpl (
            Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.58.2 Member Function Documentation

9.58.2.1 exec()

```
void SCImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Attention

we have no difference between atomic ones and unatomic ones.

Reimplemented from SWImpl.

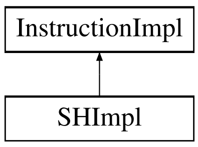
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.59 SHImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for SHImpl:



Public Member Functions

- SHImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.59.1 Constructor & Destructor Documentation

9.59.1.1 SHImpl()

```
SHImpl::SHImpl (
                Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.59.2 Member Function Documentation

9.59.2.1 exec()

```
void SHImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

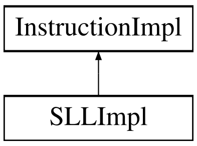
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.60 SLLImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for SLLImpl:



Public Member Functions

- SLLImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.60.1 Constructor & Destructor Documentation

9.60.1.1 SLLImpl()

```
SLLImpl::SLLImpl (
            Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.60.2 Member Function Documentation

9.60.2.1 exec()

```
void SLLImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

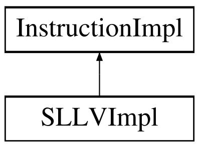
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.61 SLLVImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for SLLVImpl:



Public Member Functions

- SLLVImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.61.1 Constructor & Destructor Documentation

9.61.1.1 SLLVImpl()

```
SLLVImpl::SLLVImpl (
            Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.61.2 Member Function Documentation

9.61.2.1 exec()

```
void SLLVImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

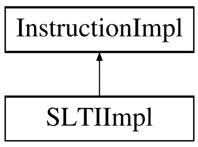
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.62 SLTIImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for SLTIImpl:



Public Member Functions

- SLTIImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.62.1 Constructor & Destructor Documentation

9.62.1.1 SLTIImpl()

```
SLTIImpl::SLTIImpl (
            Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.62.2 Member Function Documentation

9.62.2.1 exec()

```
void SLTIImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

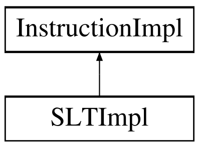
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.63 SLTImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for SLTImpl:



Public Member Functions

- SLTImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.63.1 Constructor & Destructor Documentation

9.63.1.1 SLTImpl()

```
SLTImpl::SLTImpl (
            Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.63.2 Member Function Documentation

9.63.2.1 exec()

```
void SLTImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

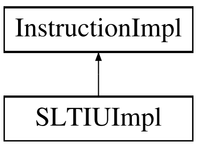
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.64 SLTIUImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for SLTIUImpl:



Public Member Functions

- SLTIUImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.64.1 Constructor & Destructor Documentation

9.64.1.1 SLTIUImpl()

```
SLTIUImpl::SLTIUImpl (
            Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.64.2 Member Function Documentation

9.64.2.1 exec()

```
void SLTIUImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

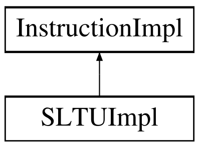
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.65 SLTUImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for SLTUImpl:



Public Member Functions

- SLTUImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.65.1 Constructor & Destructor Documentation

9.65.1.1 SLTUImpl()

```
SLTUImpl::SLTUImpl (
            Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.65.2 Member Function Documentation

9.65.2.1 exec()

```
void SLTUImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

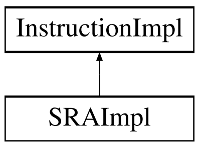
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.66 SRAImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for SRAImpl:



Public Member Functions

- SRAImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.66.1 Constructor & Destructor Documentation

9.66.1.1 SRAImpl()

```
SRAImpl::SRAImpl (
            Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.66.2 Member Function Documentation

9.66.2.1 exec()

```
void SRAImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

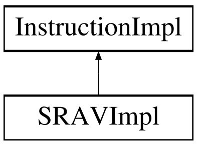
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.67 SRAVImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for SRAVImpl:



Public Member Functions

- SRAVImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.67.1 Constructor & Destructor Documentation

9.67.1.1 SRAVImpl()

```
SRAVImpl::SRAVImpl (
            Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.67.2 Member Function Documentation

9.67.2.1 exec()

```
void SRAVImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

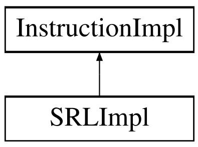
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.68 SRLImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for SRLImpl:



Public Member Functions

- SRLImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.68.1 Constructor & Destructor Documentation

9.68.1.1 SRLImpl()

```
SRLImpl::SRLImpl (
            Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.68.2 Member Function Documentation

9.68.2.1 exec()

```
void SRLImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

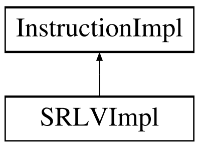
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.69 SRLVImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for SRLVImpl:



Public Member Functions

- SRLVImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.69.1 Constructor & Destructor Documentation

9.69.1.1 SRLVImpl()

```
SRLVImpl::SRLVImpl (
            Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.69.2 Member Function Documentation

9.69.2.1 exec()

```
void SRLVImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.70 Stack Struct Reference

```
#include <stack.h>
```

Public Member Functions

- Stack ()
- ~Stack ()
- void grow (size_t scale)
- bool isEnoughFor (size_t target)
- size_t size () const
- template<class T >
  T ∗ get (uint32_t addr)
- void enlarge (size_t n)
- void decrease (size_t n)
- long order (uint32_t addr)
- void clear ()

Public Attributes

- size_t capacity

  current capacity
- char ∗ highest

  memory area ending
- char ∗ current

  memory available address

9.70.1 Detailed Description

The stack simulator

9.70.2 Constructor & Destructor Documentation

9.70.2.1 Stack()

```
Stack::Stack ( )
```

Construct the stack. This will pre-allocate some memory and initialize the pointers

9.70.2.2 ∼Stack()

```
Stack::∼Stack ( )
```

Destruct the stack. This will free the holding memory

9.70.3 Member Function Documentation

9.70.3.1 clear()

```
void Stack::clear ( )
```

Clear the stack. The behavior is quite similar to initialization

9.70.3.2 decrease()

```
void Stack::decrease (
             size_t n )
```

Decrease the stack size

Parameters

n	The size to decrease by

9.70.3.3 enlarge()

```
void Stack::enlarge (
            size_t n )
```

Enlarge the stack size

Parameters

n	The size to enlarge by

9.70.3.4 get()

```
template<class T >
T * Stack::get (
            uint32_t addr )
```

Get the address at the simulated memory index

Template Parameters

T	data type

Parameters

addr	target index

Returns

real address

9.70.3.5 grow()

```
void Stack::grow (
            size_t scale )
```

Enlarge the stack capacity

Parameters

scale	The capacity to enlarge to

9.70.3.6 isEnoughFor()

```
bool Stack::isEnoughFor (
             size_t target )
```

Check whether the current capacity is enough

Parameters

target	target size

Returns

the checking result

9.70.3.7 order()

```
long Stack::order (
             uint32_t addr )
```

Get the order of an address

Parameters

addr	target address

Returns

order

9.70.3.8 size()

```
size_t Stack::size ( ) const
```

The current size

Returns

The current size

9.70.4 Member Data Documentation

9.70.4.1 capacity

```
size_t Stack::capacity
```

current capacity

9.70.4.2 current

```
char* Stack::current
```

memory available address

9.70.4.3 highest

```
char* Stack::highest
```

memory area ending

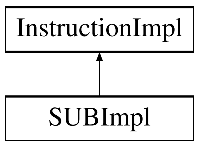
The documentation for this struct was generated from the following files:

- src/stack.h
- src/stack.cpp

9.71 SUBImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for SUBImpl:



Public Member Functions

- SUBImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.71.1 Constructor & Destructor Documentation

9.71.1.1 SUBImpl()

```
SUBImpl::SUBImpl (
            Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.71.2 Member Function Documentation

9.71.2.1 exec()

```
void SUBImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

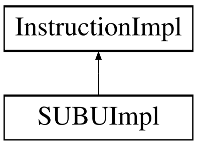
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.72 SUBUImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for SUBUImpl:



Public Member Functions

- SUBUImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.72.1 Constructor & Destructor Documentation

9.72.1.1 SUBUImpl()

```
SUBUImpl::SUBUImpl (
            Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.72.2 Member Function Documentation

9.72.2.1 exec()

```
void SUBUImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

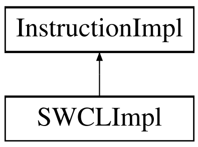
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.73 SWCLImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for SWCLImpl:



Public Member Functions

- SWCLImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.73.1 Constructor & Destructor Documentation

9.73.1.1 SWCLImpl()

```
SWCLImpl::SWCLImpl (
            Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.73.2 Member Function Documentation

9.73.2.1 exec()

```
void SWCLImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

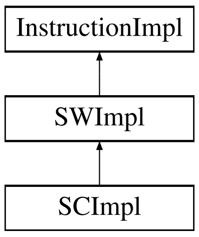
The documentation for this struct was generated from the following file:

- src/instruction_impl.h

9.74 SWImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for SWImpl:



Public Member Functions

- SWImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.74.1 Constructor & Destructor Documentation

9.74.1.1 SWImpl()

```
SWImpl::SWImpl (
            Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.74.2 Member Function Documentation

9.74.2.1 exec()

```
void SWImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

Reimplemented in SCImpl.

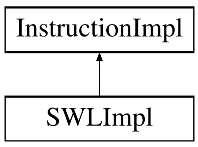
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.75 SWLImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for SWLImpl:



Public Member Functions

- SWLImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.75.1 Constructor & Destructor Documentation

9.75.1.1 SWLImpl()

```
SWLImpl::SWLImpl (
             Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.75.2 Member Function Documentation

9.75.2.1 exec()

```
void SWLImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

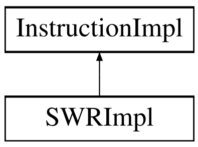
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.76 SWRImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for SWRImpl:



Public Member Functions

- SWRImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.76.1 Constructor & Destructor Documentation

9.76.1.1 SWRImpl()

```
SWRImpl::SWRImpl (
            Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.76.2 Member Function Documentation

9.76.2.1 exec()

```
void SWRImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

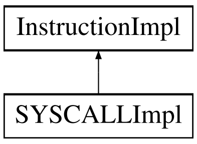
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.77 SYSCALLImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for SYSCALLImpl:



Public Member Functions

- SYSCALLImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.77.1 Constructor & Destructor Documentation

9.77.1.1 SYSCALLImpl()

```
SYSCALLImpl::SYSCALLImpl (
            Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.77.2 Member Function Documentation

9.77.2.1 exec()

```
void SYSCALLImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

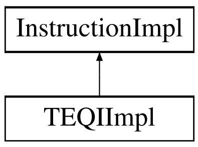
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.78 TEQIImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for TEQIImpl:



Public Member Functions

- TEQIImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.78.1 Constructor & Destructor Documentation

9.78.1.1 TEQIImpl()

```
TEQIImpl::TEQIImpl (
            Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.78.2 Member Function Documentation

9.78.2.1 exec()

```
void TEQIImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

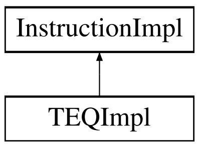
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.79 TEQImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for TEQImpl:



Public Member Functions

- TEQImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.79.1 Constructor & Destructor Documentation

9.79.1.1 TEQImpl()

```
TEQImpl::TEQImpl (
                Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.79.2 Member Function Documentation

9.79.2.1 exec()

```
void TEQImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

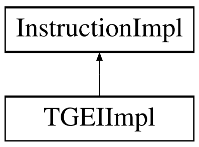
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.80 TGEIImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for TGEIImpl:



Public Member Functions

- TGEIImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.80.1 Constructor & Destructor Documentation

9.80.1.1 TGEIImpl()

```
TGEIImpl::TGEIImpl (
            Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.80.2 Member Function Documentation

9.80.2.1 exec()

```
void TGEIImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

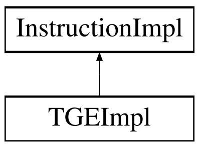
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.81 TGEImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for TGEImpl:



Public Member Functions

- TGEImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.81.1 Constructor & Destructor Documentation

9.81.1.1 TGEImpl()

```
TGEImpl::TGEImpl (
            Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.81.2 Member Function Documentation

9.81.2.1 exec()

```
void TGEImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

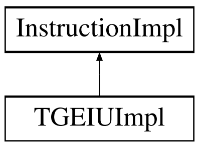
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.82 TGEIUImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for TGEIUImpl:



Public Member Functions

- TGEIUImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.82.1 Constructor & Destructor Documentation

9.82.1.1 TGEIUImpl()

```
TGEIUImpl::TGEIUImpl (
            Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.82.2 Member Function Documentation

9.82.2.1 exec()

```
void TGEIUImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

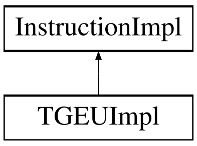
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.83 TGEUImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for TGEUImpl:



Public Member Functions

- TGEUImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.83.1 Constructor & Destructor Documentation

9.83.1.1 TGEUImpl()

```
TGEUImpl::TGEUImpl (
            Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.83.2 Member Function Documentation

9.83.2.1 exec()

```
void TGEUImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

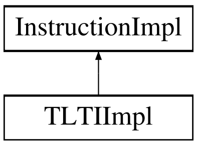
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.84 TLTIImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for TLTIImpl:



Public Member Functions

- TLTIImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.84.1 Constructor & Destructor Documentation

9.84.1.1 TLTIImpl()

```
TLTIImpl::TLTIImpl (
                Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.84.2 Member Function Documentation

9.84.2.1 exec()

```
void TLTIImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

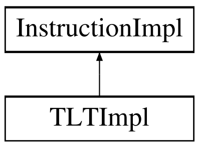
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.85 TLTImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for TLTImpl:



Public Member Functions

- TLTImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.85.1 Constructor & Destructor Documentation

9.85.1.1 TLTImpl()

```
TLTImpl::TLTImpl (
            Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.85.2 Member Function Documentation

9.85.2.1 exec()

```
void TLTImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

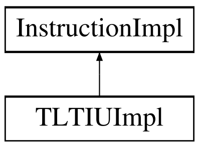
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.86 TLTIUImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for TLTIUImpl:



Public Member Functions

- TLTIUImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.86.1 Constructor & Destructor Documentation

9.86.1.1 TLTIUImpl()

```
TLTIUImpl::TLTIUImpl (
                Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.86.2 Member Function Documentation

9.86.2.1 exec()

```
void TLTIUImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

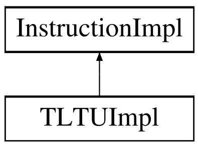
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.87 TLTUImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for TLTUImpl:



Public Member Functions

- TLTUImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.87.1 Constructor & Destructor Documentation

9.87.1.1 TLTUImpl()

```
TLTUImpl::TLTUImpl (
            Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.87.2 Member Function Documentation

9.87.2.1 exec()

```
void TLTUImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

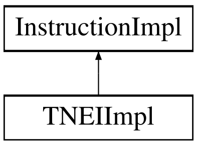
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.88 TNEIImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for TNEIImpl:



Public Member Functions

- TNEIImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.88.1 Constructor & Destructor Documentation

9.88.1.1 TNEIImpl()

```
TNEIImpl::TNEIImpl (
            Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.88.2 Member Function Documentation

9.88.2.1 exec()

```
void TNEIImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

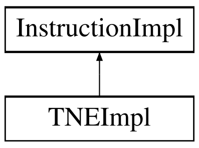
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.89 TNEImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for TNEImpl:



Public Member Functions

- TNEImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.89.1 Constructor & Destructor Documentation

9.89.1.1 TNEImpl()

```
TNEImpl::TNEImpl (
            Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.89.2 Member Function Documentation

9.89.2.1 exec()

```
void TNEImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

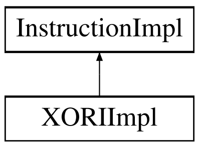
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.90 XORIImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for XORIImpl:



Public Member Functions

- XORIImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.90.1 Constructor & Destructor Documentation

9.90.1.1 XORIImpl()

```
XORIImpl::XORIImpl (
            Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.90.2 Member Function Documentation

9.90.2.1 exec()

```
void XORIImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

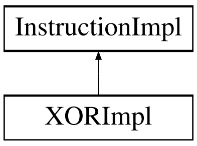
The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

9.91 XORImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for XORImpl:



Public Member Functions

- XORImpl (Instruction instr)
- void exec () override

Additional Inherited Members

9.91.1 Constructor & Destructor Documentation

9.91.1.1 XORImpl()

```
XORImpl::XORImpl (
            Instruction instr )  [explicit]
```

construct the target instruction implementation

Parameters

instr	the instruction value

9.91.2 Member Function Documentation

9.91.2.1 exec()

```
void XORImpl::exec ( )  [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements InstructionImpl.

The documentation for this struct was generated from the following files:

- src/instruction_impl.h
- src/instruction_impl.cpp

Chapter 10

File Documentation

10.1 README.md File Reference

10.2 REPORT.md File Reference

10.3 src/executor.cpp File Reference

```
#include "executor.h"
#include <QDialog>
#include <QVBoxLayout>
#include <QLabel>
#include <QPushButton>
```

10.4 src/executor.h File Reference

```
#include "instruction_impl.h"
```

Classes

- class Executor

10.5 src/fs.h File Reference

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
```

10.6 src/global.cpp File Reference

```
#include "global.h"
```

Variables

- size_t STATIC_HIGH = 0
- const char ∗ REG_NAME [32]

  a mapping between register no to register name

10.6.1 Variable Documentation

10.6.1.1 REG_NAME

```
const char* REG_NAME[32]
```

Initial value:

```
= {
        "$zero", "$at", "$v0", "$v1", "$a0", "$a1", "$a2", "$a3", "$t0", "$t1", "$t2", "$t3", "$t4",
        "$t5", "$t6", "$t7", "$s0", "$s1", "$s2", "$s3", "$s4", "$s5", "$s6", "$s7", "$t8", "$t9",
        "$k0", "$k1", "$gp", "$sp", "$fp", "$ra"
}
```

a mapping between register no to register name

10.6.1.2 STATIC_HIGH

```
size_t STATIC_HIGH = 0
```

10.7 src/global.h File Reference

```
#include <cstddef>
#include <cstdint>
#include <memory>
#include <mimalloc.h>
```

Classes

- struct _SIM::InstrDeleter

Namespaces

- _SIM

Macros

- #define BASE_ADDR 0x000000u

  start address of the MIPS program
- #define STATIC_LOW 0x500000u
- #define LIKELY(x) (x)
- #define UNLIKELY(x) (x)

Typedefs

- using _SIM::InstrPtr = std::unique_ptr< InstructionImpl, InstrDeleter >

Functions

- template<typename T , typename ... Args>
  InstrPtr _SIM::make_unique (Args &&...args)

Variables

- size_t STATIC_HIGH
- const char ∗ REG_NAME [32]

  a mapping between register no to register name

10.7.1 Macro Definition Documentation

10.7.1.1 BASE_ADDR

```
#define BASE_ADDR 0x000000u
```

start address of the MIPS program

10.7.1.2 LIKELY

```
#define LIKELY(
             x ) (x)
```

10.7.1.3 STATIC_LOW

```
#define STATIC_LOW 0x500000u
```

10.7.1.4 UNLIKELY

```
#define UNLIKELY(
              x ) (x)
```

10.7.2 Variable Documentation

10.7.2.1 REG_NAME

```
const char* REG_NAME[32]
```

a mapping between register no to register name

10.7.2.2 STATIC_HIGH

```
size_t STATIC_HIGH
```

10.8 src/heap.cpp File Reference

```
#include "heap.h"
#include "global.h"
#include <sys/mman.h>
#include <sstream>
```

Functions

- void segfault_sigaction (int signal, siginfo_t ∗si, void ∗arg)
- void bind_sigsegv ()

10.8.1 Function Documentation

10.8.1.1 bind_sigsegv()

```
void bind_sigsegv ( )
```

Bind the signal `SIGSEGV` to our own handler.

10.8.1.2 segfault_sigaction()

```
void segfault_sigaction (
             int signal,
             siginfo_t * si,
             void * arg )
```

A segfault handler that is used to handle the invalid memory access

Parameters

signal	signal value
si	signal information
arg	arguments list

10.9 src/heap.h File Reference

```
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
#include <list>
#include <queue>
#include <mimalloc.h>
#include <csignal>
#include <cstring>
```

Classes

- class Heap

Typedefs

- template<class Key , class Value >
  using TreeSet = tree< Key, Value, std::less< Key >, rb_tree_tag, tree_order_statistics_node_update >

Functions

- void segfault_sigaction (int signal, siginfo_t ∗si, void ∗arg)
- void bind_sigsegv ()

10.9.1 Typedef Documentation

10.9.1.1 TreeSet

```
template<class Key , class Value >
using TreeSet = tree<Key, Value, std::less<Key>, rb_tree_tag, tree_order_statistics_node_↩
update>
```

A Policy Based Statistics Tree to help us keep the order of the allocated chunks

10.9.2 Function Documentation

10.9.2.1 bind_sigsegv()

```
void bind_sigsegv ( )
```

Bind the signal `SIGSEGV` to our own handler.

10.9.2.2 segfault_sigaction()

```
void segfault_sigaction (
             int signal,
             siginfo_t * si,
             void * arg )
```

A segfault handler that is used to handle the invalid memory access

Parameters

signal	signal value
si	signal information
arg	arguments list

10.10 src/instruction.cpp File Reference

```
#include "instruction.h"
#include <stdexcept>
```

Functions

- TYPE resolv_type (Instruction inst)

10.10.1 Function Documentation

10.10.1.1 resolv_type()

```
TYPE resolv_type (
            Instruction inst )
```

Detect the type of an raw instruction

Parameters

inst	the instruction to detect

Returns

the type of the instruction

10.11 src/instruction.h File Reference

```
#include "global.h"
#include <instructions_types.h>
```

Macros

- #define OPC_J 0b000010
- #define OPC_JAL 0b000011
- #define OPC_ADDI 0b001000
- #define OPC_ADDIU 0b001001
- #define OPC_ANDI 0b001100
- #define OPC_BEQ 0b000100
- #define OPC_BGTZ 0b000111
- #define OPC_BLEZ 0b000110
- #define OPC_BNE 0b000101
- #define OPC_LB 0b100000
- #define OPC_LBU 0b100100
- #define OPC_LH 0b100001
- #define OPC_LHU 0b100101
- #define OPC_LUI 0b001111
- #define OPC_LW 0b100011
- #define OPC_ORI 0b001101
- #define OPC_SB 0b101000
- #define OPC_SLTI 0b001010
- #define OPC_SLTIU 0b001011
- #define OPC_SH 0b101001
- #define OPC_SW 0b101011
- #define OPC_SWCL 0b111001

- #define OPC_XORI 0b001110
- #define OPC_LWL 0b100010
- #define OPC_LWR 0b100110
- #define OPC_SWL 0b101010
- #define OPC_SWR 0b101110
- #define OPC_SC 0b111000
- #define OPC_LL 0b110000
- #define FCR_ADD 0b100000
- #define FCR_ADDU 0b100001
- #define FCR_AND 0b100100
- #define FCR_BREAK 0b001101
- #define FCR_DIV 0b011010
- #define FCR_DIVU 0b011011
- #define FCR_JALR 0b001001
- #define FCR_JR 0b001000
- #define FCR_MFHI 0b010000
- #define FCR_MFLO 0b010010
- #define FCR_MTHI 0b010001
- #define FCR_MTLO 0b010011
- #define FCR_MULT 0b011000
- #define FCR_MULTU 0b011001
- #define FCR_NOR 0b100111
- #define FCR_OR 0b100101
- #define FCR_SLL 0b000000
- #define FCR_SLLV 0b000100
- #define FCR_SLT 0b101010
- #define FCR_SLTU 0b101011
- #define FCR_SRA 0b000011
- #define FCR_SRAV 0b000111
- #define FCR_SRL 0b000010
- #define FCR_SRLV 0b000110
- #define FCR_SUB 0b100010
- #define FCR_SUBU 0b100011
- #define FCR_SYSCALL 0b001100
- #define FCR_XOR 0b100110
- #define FCR_TEQ 0b110100
- #define FCR_TNE 0b110110
- #define FCR_TGE 0b110000
- #define FCR_TGEU 0b110001
- #define FCR_TLT 0b110010
- #define FCR_TLTU 0b110011
- #define RI_BLTZ 0b00000
- #define RI_BGEZ 0b00001
- #define RI_TGEI 0b01000
- #define RI_TGEIU 0b01001
- #define RI_TLTI 0b01010
- #define RI_TLTIU 0b01011
- #define RI_TEQI 0b01100
- #define RI_TNEI 0b01110
- #define RI_BLTZAL 0b10000
- #define RI_BGEZAL 0b10001
- #define RLIKE_CLO 0b100001
- #define RLIKE_CLZ 0b100000
- #define RLIKE_MUL 0b000010
- #define RLIKE_MADD 0b000000
- #define RLIKE_MADDU 0b000001
- #define RLIKE_MSUB 0b000100
- #define RLIKE_MSUBU 0b000101

Enumerations

- enum TYPE : uint8_t {
  R, I, J, RI,
  RLIKE }

Functions

- TYPE resolv_type (Instruction inst)

10.11.1 Macro Definition Documentation

10.11.1.1 FCR_ADD

```
#define FCR_ADD 0b100000
```

10.11.1.2 FCR_ADDU

```
#define FCR_ADDU 0b100001
```

10.11.1.3 FCR_AND

```
#define FCR_AND 0b100100
```

10.11.1.4 FCR_BREAK

```
#define FCR_BREAK 0b001101
```

10.11.1.5 FCR_DIV

```
#define FCR_DIV 0b011010
```

10.11.1.6 FCR_DIVU

```
#define FCR_DIVU 0b011011
```

10.11.1.7 FCR_JALR

```
#define FCR_JALR 0b001001
```

10.11.1.8 FCR_JR

```
#define FCR_JR 0b001000
```

10.11.1.9 FCR_MFHI

```
#define FCR_MFHI 0b010000
```

10.11.1.10 FCR_MFLO

```
#define FCR_MFLO 0b010010
```

10.11.1.11 FCR_MTHI

```
#define FCR_MTHI 0b010001
```

10.11.1.12 FCR_MTLO

```
#define FCR_MTLO 0b010011
```

10.11.1.13 FCR_MULT

```
#define FCR_MULT 0b011000
```

10.11.1.14 FCR_MULTU

```
#define FCR_MULTU 0b011001
```

10.11.1.15 FCR_NOR

```
#define FCR_NOR 0b100111
```

10.11.1.16 FCR_OR

```
#define FCR_OR 0b100101
```

10.11.1.17 FCR_SLL

```
#define FCR_SLL 0b000000
```

10.11.1.18 FCR_SLLV

```
#define FCR_SLLV 0b000100
```

10.11.1.19 FCR_SLT

```
#define FCR_SLT 0b101010
```

10.11.1.20 FCR_SLTU

```
#define FCR_SLTU 0b101011
```

10.11.1.21 FCR_SRA

```
#define FCR_SRA 0b000011
```

10.11.1.22 FCR_SRAV

```
#define FCR_SRAV 0b000111
```

10.11.1.23 FCR_SRL

```
#define FCR_SRL 0b000010
```

10.11.1.24 FCR_SRLV

```
#define FCR_SRLV 0b000110
```

10.11.1.25 FCR_SUB

```
#define FCR_SUB 0b100010
```

10.11.1.26 FCR_SUBU

```
#define FCR_SUBU 0b100011
```

10.11.1.27 FCR_SYSCALL

```
#define FCR_SYSCALL 0b001100
```

10.11.1.28 FCR_TEQ

```
#define FCR_TEQ 0b110100
```

10.11.1.29 FCR_TGE

```
#define FCR_TGE 0b110000
```

10.11.1.30 FCR_TGEU

```
#define FCR_TGEU 0b110001
```

10.11.1.31 FCR_TLT

```
#define FCR_TLT 0b110010
```

10.11.1.32 FCR_TLTU

```
#define FCR_TLTU 0b110011
```

10.11.1.33 FCR_TNE

```
#define FCR_TNE 0b110110
```

10.11.1.34 FCR_XOR

```
#define FCR_XOR 0b100110
```

10.11.1.35 OPC_ADDI

```
#define OPC_ADDI 0b001000
```

10.11.1.36 OPC_ADDIU

```
#define OPC_ADDIU 0b001001
```

10.11.1.37 OPC_ANDI

```
#define OPC_ANDI 0b001100
```

10.11.1.38 OPC_BEQ

```
#define OPC_BEQ 0b000100
```

10.11.1.39 OPC_BGTZ

```
#define OPC_BGTZ 0b000111
```

10.11.1.40 OPC_BLEZ

```
#define OPC_BLEZ 0b000110
```

10.11.1.41 OPC_BNE

```
#define OPC_BNE 0b000101
```

10.11.1.42 OPC_J

```
#define OPC_J 0b000010
```

10.11.1.43 OPC_JAL

```
#define OPC_JAL 0b000011
```

10.11.1.44 OPC_LB

```
#define OPC_LB 0b100000
```

10.11.1.45 OPC_LBU

```
#define OPC_LBU 0b100100
```

10.11.1.46 OPC_LH

```
#define OPC_LH 0b100001
```

10.11.1.47 OPC_LHU

```
#define OPC_LHU 0b100101
```

10.11.1.48 OPC_LL

```
#define OPC_LL 0b110000
```

10.11.1.49 OPC_LUI

```
#define OPC_LUI 0b001111
```

10.11.1.50 OPC_LW

```
#define OPC_LW 0b100011
```

10.11.1.51 OPC_LWL

```
#define OPC_LWL 0b100010
```

10.11.1.52 OPC_LWR

```
#define OPC_LWR 0b100110
```

10.11.1.53 OPC_ORI

```
#define OPC_ORI 0b001101
```

10.11.1.54 OPC_SB

```
#define OPC_SB 0b101000
```

10.11.1.55 OPC_SC

```
#define OPC_SC 0b111000
```

10.11.1.56 OPC_SH

```
#define OPC_SH 0b101001
```

10.11.1.57 OPC_SLTI

```
#define OPC_SLTI 0b001010
```

10.11.1.58 OPC_SLTIU

```
#define OPC_SLTIU 0b001011
```

10.11.1.59 OPC_SW

```
#define OPC_SW 0b101011
```

10.11.1.60 OPC_SWCL

```
#define OPC_SWCL 0b111001
```

10.11.1.61 OPC_SWL

```
#define OPC_SWL 0b101010
```

10.11.1.62 OPC_SWR

```
#define OPC_SWR 0b101110
```

10.11.1.63 OPC_XORI

```
#define OPC_XORI 0b001110
```

10.11.1.64 RI_BGEZ

```
#define RI_BGEZ 0b00001
```

10.11.1.65 RI_BGEZAL

```
#define RI_BGEZAL 0b10001
```

10.11.1.66 RI_BLTZ

```
#define RI_BLTZ 0b00000
```

10.11.1.67 RI_BLTZAL

```
#define RI_BLTZAL 0b10000
```

10.11.1.68 RI_TEQI

```
#define RI_TEQI 0b01100
```

10.11.1.69 RI_TGEI

```
#define RI_TGEI 0b01000
```

10.11.1.70 RI_TGEIU

```
#define RI_TGEIU 0b01001
```

10.11.1.71 RI_TLTI

```
#define RI_TLTI 0b01010
```

10.11.1.72 RI_TLTIU

```
#define RI_TLTIU 0b01011
```

10.11.1.73 RI_TNEI

```
#define RI_TNEI 0b01110
```

10.11.1.74 RLIKE_CLO

```
#define RLIKE_CLO 0b100001
```

10.11.1.75 RLIKE_CLZ

```
#define RLIKE_CLZ 0b100000
```

10.11.1.76 RLIKE_MADD

```
#define RLIKE_MADD 0b000000
```

10.11.1.77 RLIKE_MADDU

```
#define RLIKE_MADDU 0b000001
```

10.11.1.78 RLIKE_MSUB

```
#define RLIKE_MSUB 0b000100
```

10.11.1.79 RLIKE_MSUBU

```
#define RLIKE_MSUBU 0b000101
```

10.11.1.80 RLIKE_MUL

```
#define RLIKE_MUL 0b000010
```

10.11.2 Enumeration Type Documentation

10.11.2.1 TYPE

```
enum TYPE :  uint8_t
```

An enumeration for different category of instructions

Enumerator

R	R-Type MIPS Instructions.
I	I-Type MIPS Instructions.
J	J-Type MIPS Instructions.
RI	RI-Type MIPS Instructions.
RLIKE	RLIKE-Type MIPS Instructions.

10.11.3 Function Documentation

10.11.3.1 resolv_type()

```
TYPE resolv_type (
             Instruction inst )
```

Detect the type of an raw instruction

Parameters

inst	the instruction to detect

Returns

the type of the instruction

10.12 src/instruction_impl.cpp File Reference

```
#include "instruction_impl.h"
#include "mainwindow.ipp"
#include "syscall.h"
```

10.13 src/instruction_impl.h File Reference

```
#include "mainwindow.h"
#include "./ui_mainwindow.h"
```

Classes

- struct InstructionImpl
- struct NOPImpl
- struct JImpl
- struct JALImpl
- struct ADDIImpl
- struct ADDIUImpl
- struct ANDIImpl
- struct BEQImpl
- struct BGEZImpl
- struct BGTZImpl
- struct BLEZImpl
- struct BLTZImpl
- struct BNEImpl
- struct LBImpl
- struct LBUImpl
- struct LHImpl
- struct LHUImpl
- struct LUIImpl
- struct LWImpl
- struct ORIImpl
- struct SBImpl
- struct SLTIImpl
- struct SLTIUImpl
- struct SHImpl
- struct SWImpl
- struct SWCLImpl

- struct XORIImpl
- struct ADDImpl
- struct ADDUImpl
- struct ANDImpl
- struct BREAKImpl
- struct DIVImpl
- struct DIVUImpl
- struct JRImpl
- struct JALRImpl
- struct MFHIImpl
- struct MFLOImpl
- struct MTHIImpl
- struct MTLOImpl
- struct MULTImpl
- struct MULTUImpl
- struct NORImpl
- struct ORImpl
- struct SLLImpl
- struct SLLVImpl
- struct SLTImpl
- struct SLTUImpl
- struct SRAImpl
- struct SRAVImpl
- struct SRLImpl
- struct SRLVImpl
- struct SUBImpl
- struct SUBUImpl
- struct SYSCALLImpl
- struct XORImpl
- struct BLTZLImpl
- struct BGEZLImpl
- struct TGEIImpl
- struct TGEIUImpl
- struct TLTIImpl
- struct TLTIUImpl
- struct TEQIImpl
- struct TNEIImpl
- struct BLTZALImpl
- struct BGEZALImpl
- struct BLTZALLImpl
- struct BGEZALLImpl
- struct CLOImpl
- struct CLZImpl
- struct MULImpl
- struct MADDImpl
- struct MADDUImpl
- struct MSUBImpl
- struct MSUBUImpl
- struct TEQImpl
- struct TNEImpl
- struct TGEImpl
- struct TGEUImpl
- struct TLTImpl
- struct TLTUImpl
- struct SWLImpl

- struct SWRImpl
- struct LWLImpl
- struct LWRImpl
- struct SCImpl
- struct LLImpl

Macros

- #define DEFAULT_INIT(NAME, FATHER) NAME##Impl::NAME##Impl(Instruction instr) : FATH↩ ER##Impl(instr) {}
- #define ComDef(CLASS, FATHER)
- #define SimDef(CLASS) ComDef(CLASS, Instruction)
- #define ComImplDef(CLASS, FATHER, BLOCK)
- #define SimImplDef(CLASS, BLOCK) ComImplDef(CLASS, Instruction, BLOCK)
- #define OP_AMONG_REGS(NAME, A, B, op, C)
- #define OP_AMONG_REGS_OVERFLOW(NAME, A, B, op, C)
- #define SHIFT_REAL(NAME, A, B, op, C, TYPE)
- #define SHIFT_IMM(NAME, A, B, op, C, TYPE)
- #define TRAP_R(NAME, op, TYPE)
- #define TRAP_RI(NAME, op, TYPE)
- #define BRANCH_IF(NAME, COND)
- #define BRANCH_IF_SAVE(NAME, COND)

10.13.1 Macro Definition Documentation

10.13.1.1 BRANCH_IF

```
#define BRANCH_IF(
                NAME,
                COND )
```

Value:

```
SimImplDef(NAME, {\
    if (COND) {\
        mainW->updateProgramCounter(mainW->PC + (int16_t) instr.INST_I.C * int32_t(4));\
    }\
})
```

Generate the real definition of Branch instructions

10.13.1.2 BRANCH_IF_SAVE

```
#define BRANCH_IF_SAVE(
                NAME,
                COND )
```

Value:

```
SimImplDef(NAME, {\
    if (COND) {\
        mainW->updateRegValue(31, mainW->PC + 4);\
        mainW->updateProgramCounter(mainW->PC + (int16_t) instr.INST_I.C * int32_t(4));\
    }\
})
```

Generate the real definition of Branch instructions with save operation

10.13.1.3 ComDef

```
#define ComDef(
              CLASS,
              FATHER )
```

Value:

```
    struct CLASS##Impl : public FATHER##Impl {\ \
        explicit CLASS##Impl( Instruction instr);\ \
        void exec() override;\
    };
```

Complex declaration of an instruction implementation class.

Parameters

NAME	name of the instruction
FATHER	direct base of the instruction

10.13.1.4 ComImplDef

```
#define ComImplDef(
              CLASS,
              FATHER,
              BLOCK )
```

Value:

```
    void CLASS##Impl::exec() BLOCK\
    DEFAULT_INIT(CLASS, FATHER)
```

Complex real definition of instruction behaviors (custom direct base)

Attention

must be used together with declarations

10.13.1.5 DEFAULT_INIT

```
#define DEFAULT_INIT(
              NAME,
              FATHER ) NAME##Impl::NAME##Impl(Instruction instr) :  FATHER##Impl(instr) {}
```

A handy macro to create a default constructor based on `NAME` and `FATHER`.

Parameters

NAME	name of the instruction
FATHER	direct base of the instruction

10.13.1.6 OP_AMONG_REGS

```
#define OP_AMONG_REGS(
              NAME,
              A,
              B,
              op,
              C )
```

Value:
```
SimImplDef(NAME, {\
    mainW->updateRegValue(instr.INST_R.A, mainW->REGS[instr.INST_R.B] op mainW->REGS[instr.INST_R.C]);\
})
```

Generate the real definition of the operations among registers

10.13.1.7 OP_AMONG_REGS_OVERFLOW

```
#define OP_AMONG_REGS_OVERFLOW(
              NAME,
              A,
              B,
              op,
              C )
```

Value:
```
SimImplDef(NAME, {\
    int32_t a = mainW->REGS[instr.INST_R.B];\
    int32_t b = mainW->REGS[instr.INST_R.C];\
    int32_t c = 0;\
    if(__builtin_##op##_overflow(a, b, &c)) throw std::runtime_error {"overflow"};\
    mainW->updateRegValue(instr.INST_R.A, c);\
})
```

Generate the real definition of the operations among registers with overflow checking

10.13.1.8 SHIFT_IMM

```
#define SHIFT_IMM(
              NAME,
              A,
              B,
              op,
              C,
              TYPE )
```

Value:
```
SimImplDef(NAME, {\
    mainW->updateRegValue(instr.INST_R.A,\
    static_cast<TYPE>(mainW->REGS[instr.INST_R.B]) op static_cast<TYPE>(instr.INST_R.C));\
})
```

Generate the real definition of shifting instructions whose shift among is stored in immediate value

10.13.1.9 SHIFT_REAL

```
#define SHIFT_REAL(
              NAME,
              A,
              B,
              op,
              C,
              TYPE )
```

Value:

```
SimImplDef(NAME, {\
    mainW->updateRegValue(instr.INST_R.A,\
    static_cast<TYPE>(mainW->REGS[instr.INST_R.B]) op static_cast<TYPE>(0b11111 &
       mainW->REGS[instr.INST_R.C]));\
})
```

Generate the real definition of shifting instructions whose shift among is stored in register

10.13.1.10 SimDef

```
#define SimDef(
              CLASS ) ComDef(CLASS, Instruction)
```

Simple declaration for those instruction implementations with a direct base of `InstructionImpl`

Parameters

CLASS	the name of the instruction

10.13.1.11 SimImplDef

```
#define SimImplDef(
              CLASS,
              BLOCK ) ComImplDef(CLASS, Instruction, BLOCK)
```

Simple real definition of instruction behaviors (default direct base)

Attention

must be used together with declarations

10.13.1.12 TRAP_R

```
#define TRAP_R(
               NAME,
               op,
               TYPE )
```

Value:

```
SimImplDef(NAME, {\
    TYPE a = mainW->REGS[instr.INST_R.s];\
    TYPE b = mainW->REGS[instr.INST_R.t];\
    if (a op b) throw std::runtime_error {"conditionally trapped!"};\
})
```

Generate the real definition of R-Type Trap instructions

10.13.1.13 TRAP_RI

```
#define TRAP_RI(
               NAME,
               op,
               TYPE )
```

Value:

```
SimImplDef(NAME, {\
    TYPE a = mainW->REGS[instr.INST_I.s];\
    TYPE b = instr.INST_I.C;\
    if (a op b) throw std::runtime_error {"conditionally trapped!"};\
})
```

Generate the real definition of RI-Type Trap instructions

10.14 src/main.cpp File Reference

```
#include "mainwindow.h"
#include <QApplication>
```

Functions

- int main (int argc, char ∗argv[ ])

10.14.1 Function Documentation

10.14.1.1 main()

```
int main (
            int argc,
            char ∗ argv[ ] )
```

10.15 src/mainwindow.cpp File Reference

```
#include "mainwindow.h"
#include "./ui_mainwindow.h"
#include "global.h"
#include <QMessageBox>
#include <QFileDialog>
#include <iostream>
#include <memory>
#include "instruction_impl.h"
#include "mainwindow.ipp"
#include "instruction.h"
#include <atomic>
#include "executor.h"
#include <cstring>
#include <QInputDialog>
#include "syscall.h"
#include "fs.h"
#include <QGraphicsView>
#include <QPainter>
```

10.16 src/mainwindow.h File Reference

```
#include <QMainWindow>
#include <QVector>
#include <QListWidget>
#include <memory>
#include <QThread>
#include <QTimer>
#include "instruction.h"
#include "heap.h"
#include "stack.h"
#include <assembler/include/api.h>
```

Classes

- class MainWindow

Namespaces

- Ui

Macros

- #define KiB 1024
- #define MiB (KiB ∗ KiB)
- #define FRAME_SIZE MiB
- #define CASE(NAME, TYPE)
- #define RCASE(NAME) CASE(NAME, FCR)
- #define IJCASE(NAME) CASE(NAME, OPC)
- #define RICASE(NAME) CASE(NAME, RI)
- #define RLCASE(NAME) CASE(NAME, RLIKE)
- #define HANDLE(NAME, BLOCK)

Enumerations

- enum MemoryType { STATIC, HEAP, STACK }

10.16.1 Macro Definition Documentation

10.16.1.1 CASE

```
#define CASE(
              NAME,
              TYPE )
```

Value:

```
case TYPE##_##NAME:\
executor->impls[i] = _SIM::make_unique<NAME##Impl> (instr);\
break;
```

Generate a branch case for instruction handling

10.16.1.2 FRAME_SIZE

```
#define FRAME_SIZE MiB
```

10.16.1.3 HANDLE

```
#define HANDLE(
              NAME,
              BLOCK )
```

Value:

```
case SYSCALL_##NAME:\
    BLOCK\
    break;
```

Generate a branch case for syscall handling

10.16.1.4 IJCASE

```
#define IJCASE(
              NAME ) CASE(NAME, OPC)
```

Generate a branch case for I/J-Type instruction handling

10.16.1.5 KiB

```
#define KiB 1024
```

10.16.1.6 MiB

```
#define MiB (KiB * KiB)
```

10.16.1.7 RCASE

```
#define RCASE(
              NAME ) CASE(NAME, FCR)
```

Generate a branch case for R-Type instruction handling

10.16.1.8 RICASE

```
#define RICASE(
              NAME ) CASE(NAME, RI)
```

Generate a branch case for RI-Type instruction handling

10.16.1.9 RLCASE

```
#define RLCASE(
              NAME ) CASE(NAME, RLIKE)
```

Generate a branch case for RLIKE-Type instruction handling

10.16.2 Enumeration Type Documentation

10.16.2.1 MemoryType

```
enum MemoryType
```

Different type of memories

Enumerator

STATIC	
HEAP	
STACK	

10.17 src/mainwindow.ipp File Reference

```
#include "./ui_mainwindow.h"
#include "mainwindow.h"
```

Macros

- #define MAINWINDOW_IPP

10.17.1 Macro Definition Documentation

10.17.1.1 MAINWINDOW_IPP

```
#define MAINWINDOW_IPP
```

10.18 src/stack.cpp File Reference

```
#include "stack.h"
```

Functions

- uint32_t nextPowerOfTwo (uint32_t n)

10.18.1 Function Documentation

10.18.1.1 nextPowerOfTwo()

```
uint32_t nextPowerOfTwo (
            uint32_t n )
```

10.19 src/stack.h File Reference

```
#include <cstddef>
#include <mimalloc.h>
#include <cstring>
#include <stdexcept>
```

Classes

- struct Stack

Macros

- #define DEFAULT_SIZE 1u
- #define STACK_HIGH 0x7fffffffu

Functions

- uint32_t nextPowerOfTwo (uint32_t n)

10.19.1 Macro Definition Documentation

10.19.1.1 DEFAULT_SIZE

```
#define DEFAULT_SIZE 1u
```

10.19.1.2 STACK_HIGH

```
#define STACK_HIGH 0x7fffffffu
```

10.19.2 Function Documentation

10.19.2.1 nextPowerOfTwo()

```
uint32_t nextPowerOfTwo (
            uint32_t n )
```

10.20 src/syscall.h File Reference

Macros

- #define SYSCALL_PRINT_CHAR 11
- #define SYSCALL_PRINT_INT 1
- #define SYSCALL_PRINT_STRING 4
- #define SYSCALL_READ_CHAR 12
- #define SYSCALL_READ_INT 5
- #define SYSCALL_READ_STRING 8
- #define SYSCALL_MMAP 9
- #define SYSCALL_EXIT 10
- #define SYSCALL_OPEN 13
- #define SYSCALL_READ 14
- #define SYSCALL_WRITE 15
- #define SYSCALL_CLOSE 16
- #define SYSCALL_EXIT2 17
- #define SYSCALL_FAST_COPY 10000
- #define SYSCALL_UI_OPEN_FILE 10001
- #define SYSCALL_MUNMAP 10002

10.20.1 Macro Definition Documentation

10.20.1.1 SYSCALL_CLOSE

```
#define SYSCALL_CLOSE 16
```

10.20.1.2 SYSCALL_EXIT

```
#define SYSCALL_EXIT 10
```

10.20.1.3 SYSCALL_EXIT2

```
#define SYSCALL_EXIT2 17
```

10.20.1.4 SYSCALL_FAST_COPY

```
#define SYSCALL_FAST_COPY 10000
```

10.20.1.5 SYSCALL_MMAP

```
#define SYSCALL_MMAP 9
```

10.20.1.6 SYSCALL_MUNMAP

```
#define SYSCALL_MUNMAP 10002
```

10.20.1.7 SYSCALL_OPEN

```
#define SYSCALL_OPEN 13
```

10.20.1.8 SYSCALL_PRINT_CHAR

```
#define SYSCALL_PRINT_CHAR 11
```

10.20.1.9 SYSCALL_PRINT_INT

```
#define SYSCALL_PRINT_INT 1
```

10.20.1.10 SYSCALL_PRINT_STRING

```
#define SYSCALL_PRINT_STRING 4
```

10.20.1.11 SYSCALL_READ

```
#define SYSCALL_READ 14
```

10.20.1.12 SYSCALL_READ_CHAR

```
#define SYSCALL_READ_CHAR 12
```

10.20.1.13 SYSCALL_READ_INT

```
#define SYSCALL_READ_INT 5
```

10.20.1.14 SYSCALL_READ_STRING

```
#define SYSCALL_READ_STRING 8
```

10.20.1.15 SYSCALL_UI_OPEN_FILE

```
#define SYSCALL_UI_OPEN_FILE 10001
```

10.20.1.16 SYSCALL_WRITE

```
#define SYSCALL_WRITE 15
```

Index