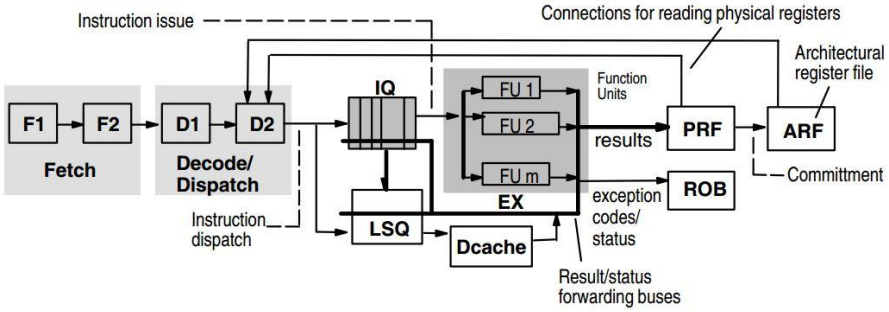
APEX Simulator Project

by Roman Kurbanov
Binghamton University. 2016

A Pipeline EXample(APEX) simulator with reorder buffer, physical register file, architectural register file, issue queue, load store queue and functional units. Pipeline consists of 2 fetch stages, 2 decode/dispatch stages and execution stage with several function units with different latencies. Memory is represented as instruction and data caches only.



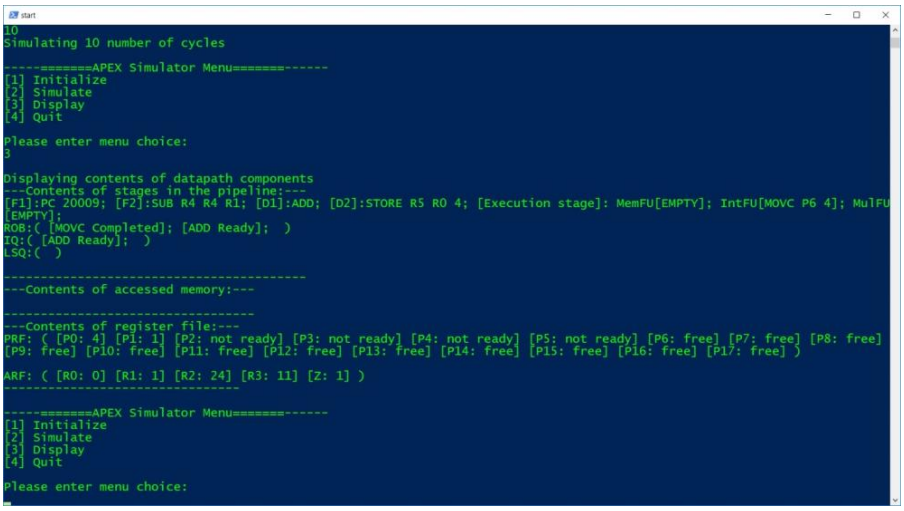
Functional units that are present: * Int FU (add, sub and logical instructions), 1 cycle latency * Multiplication FU, 4 cycles latency, non-pipelined * Memory FU (implements Load, Store instructions), 3 cycle latency, pipelined

Usage: java -jar APEX_Simulator.jar "inputfile"
inputfile - text file with assembler instructions (supported instructions are *ADD, SUB, MOVC, MUL, AND, OR, EX-OR, LOAD, STORE, BZ, BNZ, JUMP, BAL, HALT*)

Project: Developed in *Java 8* using *IntelliJ IDEA v2016.1* project format.

Implementation: Most of the project documentation can be extracted with *javadoc*. Classes are split into according packages. Software follows MVC (Model View Controller) ideology. Since this is a console application View is merged with Model for simplicity purposes, but can be extended further, since all components implement *Display* interface. Model is enclosed in *components* package and all its subpackages, Controller is solely in *main* package.
Memory(d-Cache, i-Cache) and RegisterFile(ARF, PRF, RAT) are implemented as singletons with double-checked locking(thread safe for future extensions).
All stages are implemented as discrete entities that interact between each other by means of producer and consumer roles with single slot buffer.
Execution stage is the biggest stage in terms of design since it encapsulates Issue Queue, Functional Units, Load/Store Queue, Retirement Unit and Reorder Buffer

Running: All operations are entered through numbers. If simulation reaches end of program it displays internal state and ends simulator. Sample run is displayed in the image below



Thorough documentation of classes starts at the next page. Documentation is grouped in packages for easier reference. Private members are not documented, only public API. Inherited members from language framework also not documented/ At the end of the document there is index with page numbers.

Package
apexsimulator.util

This package defines classes and interfaces that will form core utility and helper classes and functions

The FileInterface defines core API needed by software for file operations

The FileProcessor implements all the methods of the FileInterface

The InstructionsEnum defines current instruction set

The ArchRegisterEnum defines current ISA registers

The StringParser combines utility methods for parsing different text portions

The ErrorCodes predefined typical error codes

The InstructionStatus predefined statuses for instruction

apexsimulator.util
Class ArchRegisterEnum

```
java.lang.Object
   |-
      |-apexsimulator.util.ArchRegisterEnum
```

All Implemented Interfaces:
java.io.Serializable, java.lang.Comparable

---

public final class **ArchRegisterEnum**
extends java.lang.Enum

Enum for architectural registers
Author:
Roman Kurbanov

Field Summary

public static final	R0
public static final	R1
public static final	R2
public static final	R3
public static final	R4
public static final	R5
public static final	R6
public static final	R7
public static final	UNUSED
public static final	X
public static final	Z

Method Summary

static ArchRegisterEnum	valueOf(java.lang.String name)
static ArchRegisterEnum[]	values()

Fields

R0

```
public static final apexsimulator.util.ArchRegisterEnum R0
```

R1

```
public static final apexsimulator.util.ArchRegisterEnum R1
```

R2

```
public static final apexsimulator.util.ArchRegisterEnum R2
```

R3

```
public static final apexsimulator.util.ArchRegisterEnum R3
```

R4

```
public static final apexsimulator.util.ArchRegisterEnum R4
```

R5

```
public static final apexsimulator.util.ArchRegisterEnum R5
```

R6

```
public static final apexsimulator.util.ArchRegisterEnum R6
```

R7

```
public static final apexsimulator.util.ArchRegisterEnum R7
```

X

```
public static final apexsimulator.util.ArchRegisterEnum X
```

Z

```
public static final apexsimulator.util.ArchRegisterEnum Z
```

UNUSED

```
public static final apexsimulator.util.ArchRegisterEnum UNUSED
```

Methods

values

```
public static ArchRegisterEnum[] values()
```

valueOf

```
public static ArchRegisterEnum valueOf(java.lang.String name)
```

apexsimulator.util
Class ErrorCodes

```
java.lang.Object
   |
   +-apexsimulator.util.ErrorCodes
```

public class **ErrorCodes**
extends java.lang.Object

This class defines constants for typical errors during emulator runtime. List can be extended.
Author:
Roman Kurbanov

Field Summary

public static final	ACCESS_ERROR Value: **6**
public static final	CONSOLE_ERROR Value: **3**
public static final	DECODE_ERROR Value: **5**
public static final	FILE_ERROR Value: **2**
public static final	SEGFAULT_ERROR Value: **4**
public static final	USAGE_ERROR Value: **1**

Fields

USAGE_ERROR

```
public static final int USAGE_ERROR
```

Constant value: **1**

FILE_ERROR

```
public static final int FILE_ERROR
```

Constant value: **2**

CONSOLE_ERROR

```
public static final int CONSOLE_ERROR
```

Constant value: **3**

SEGFAULT_ERROR

```
public static final int SEGFAULT_ERROR
```

Constant value: **4**

DECODE_ERROR

```
public static final int DECODE_ERROR
```

Constant value: **5**

ACCESS_ERROR

```
public static final int ACCESS_ERROR
```

Constant value: **6**

apexsimulator.util
Interface FileInterface

All Known Implementing Classes:
FileProcessor

---

public interface **FileInterface**
extends

The FileInterface defines core API needed by software for file operations
Author:
Roman Kurbanov

Method Summary

abstract void	closeFile() Tries to close file streem
abstract void	openFile() Tries to open file stream
abstract java.lang.String	readLine() Reads one line at a time from the file

---

Methods

readLine

```
public abstract java.lang.String readLine()
```

Reads one line at a time from the file

Returns:
read out string or null if nothing to read

---

openFile

```
public abstract void openFile()
```

Tries to open file stream

---

closeFile

```
public abstract void closeFile()
```

Tries to close file streem

apexsimulator.util
Class FileProcessor

```
java.lang.Object
   |
   +-apexsimulator.util.FileProcessor
```

All Implemented Interfaces:
FileInterface

---

public class **FileProcessor**
extends java.lang.Object
implements FileInterface

This class implements methods to read lines from the file.

Implements FileInterface
Author:
Roman Kurbanov

Constructor Summary

public	`FileProcessor(java.lang.String inputFileIn)` initializes member variables.

Method Summary

void	`closeFile()` Tries to close file streem
void	`openFile()` Tries to open file stream
java.lang.String	`readLine()` Reads one line at a time from the file

Methods inherited from interface
`closeFile, openFile, readLine`

Constructors

FileProcessor

`public FileProcessor(java.lang.String inputFileIn)`

initializes member variables. Stream will be opened on demand only

Parameters:
`inputFileIn` - file to read

Methods

(continued from last page)

readLine

```
public java.lang.String readLine()
```

Reads one line at a time from the file

Returns:
read out string or null if nothing to read

---

openFile

```
public void openFile()
```

Tries to open file stream

---

closeFile

```
public void closeFile()
```

Tries to close file streem

apexsimulator.util
Class InstructionsEnum

```
java.lang.Object
   |-
      |-apexsimulator.util.InstructionsEnum
```

All Implemented Interfaces:
java.io.Serializable, java.lang.Comparable

---

public final class **InstructionsEnum**
extends java.lang.Enum

This enum defines all the supported instructions List can be extended without affecting existing functionality.
Author:
Roman Kurbanov

Field Summary

public static final	ADD
public static final	AND
public static final	BAL
public static final	BNZ
public static final	BZ
public static final	EX_OR
public static final	HALT
public static final	JUMP
public static final	LOAD
public static final	MOVC
public static final	MUL
public static final	NOP
public static final	OR
public static final	STORE

public static final	SUB
public static final	WRONG

Method Summary

static InstructionsEnum	valueOf(java.lang.String name)
static InstructionsEnum[]	values()

Fields

ADD

public static final apexsimulator.util.InstructionsEnum **ADD**

SUB

public static final apexsimulator.util.InstructionsEnum **SUB**

MOVC

public static final apexsimulator.util.InstructionsEnum **MOVC**

MUL

public static final apexsimulator.util.InstructionsEnum **MUL**

AND

public static final apexsimulator.util.InstructionsEnum **AND**

OR

public static final apexsimulator.util.InstructionsEnum **OR**

EX_OR

public static final apexsimulator.util.InstructionsEnum **EX_OR**

LOAD

public static final apexsimulator.util.InstructionsEnum **LOAD**

STORE

public static final apexsimulator.util.InstructionsEnum **STORE**

BZ

public static final apexsimulator.util.InstructionsEnum **BZ**

BNZ

public static final apexsimulator.util.InstructionsEnum **BNZ**

JUMP

public static final apexsimulator.util.InstructionsEnum **JUMP**

BAL

public static final apexsimulator.util.InstructionsEnum **BAL**

HALT

public static final apexsimulator.util.InstructionsEnum **HALT**

NOP

public static final apexsimulator.util.InstructionsEnum **NOP**

WRONG

public static final apexsimulator.util.InstructionsEnum **WRONG**

Methods

(continued from last page)

values

```
public static InstructionsEnum[] values()
```

---

valueOf

```
public static InstructionsEnum valueOf(java.lang.String name)
```

apexsimulator.util
Class InstructionStatus

```
java.lang.Object
   |-
      |-apexsimulator.util.InstructionStatus
```

All Implemented Interfaces:
java.io.Serializable, java.lang.Comparable

---

public final class **InstructionStatus**
extends java.lang.Enum

This enum for instruction statuses
Author:
Roman Kurbanov

Field Summary

public static final	Completed
public static final	Executing
public static final	Raw
public static final	Ready
public static final	Waiting

Method Summary

static InstructionStatus	valueOf(java.lang.String name)
static InstructionStatus[]	values()

Fields

Raw

public static final apexsimulator.util.InstructionStatus **Raw**

Waiting

public static final apexsimulator.util.InstructionStatus **Waiting**

Executing

```
public static final apexsimulator.util.InstructionStatus Executing
```

Ready

```
public static final apexsimulator.util.InstructionStatus Ready
```

Completed

```
public static final apexsimulator.util.InstructionStatus Completed
```

Methods

values

```
public static InstructionStatus[] values()
```

valueOf

```
public static InstructionStatus valueOf(java.lang.String name)
```

apexsimulator.util
Class StringParser

```
java.lang.Object
   |
   +-apexsimulator.util.StringParser
```

public class **StringParser**
extends java.lang.Object

Has some static helper functions to deal with parsing strings.
Author:
Roman Kurbanov

Method Summary

static InstructionsEnum	`getInstr(java.lang.String instr)` Converts text representation to Instruction Enumerator

Methods

getInstr

```
public static InstructionsEnum getInstr(java.lang.String instr)
```

Converts text representation to Instruction Enumerator

Parameters:
`instr` - text

Returns:
enumerator type

Package
apexsimulator.main

This package contains classes that are necessary to setup simulator and interact with user.

The Driver class is the entry point of the program

apexsimulator.main
Class Controller

```
java.lang.Object
   |
   +-apexsimulator.main.Controller
```

public class **Controller**
extends java.lang.Object

This class implements main controller logic of the simulator
Author:
Roman Kurbanov

Constructor Summary

public	`Controller()` Instantiates datapath and creates menu

Method Summary

void	`startSimulator()`

Constructors

Controller

`public Controller()`

Instantiates datapath and creates menu

Methods

startSimulator

`public void startSimulator()`

apexsimulator.main
Class Driver

```
java.lang.Object
   |
   +-apexsimulator.main.Driver
```

public class **Driver**
extends java.lang.Object

Entry point of the program. Does simple usage checking and starts the execution
Author:
Roman Kurbanov

Constructor Summary

public	Driver()

Method Summary

static void	main(java.lang.String[] args) Instantiates controller and controls commands sent to it

Constructors

Driver

```
public Driver()
```

Methods

main

```
public static void main(java.lang.String[] args)
```

Instantiates controller and controls commands sent to it

Parameters:
`args` - file name to load

apexsimulator.main
Class Menu

```
java.lang.Object
   |
   +-apexsimulator.main.Menu
```

All Implemented Interfaces:
MenuInterface

---

public class **Menu**
extends java.lang.Object
implements MenuInterface

This class implements a menu to interact with user
Author:
Roman Kurbanov

Constructor Summary

public	`Menu()` Sets buffered reader to console input

Method Summary

void	`displayMenu()` Display menu items to screen
MenuCodes	`readCode()` Get user input
int	`readCycles()` If simulation was chosen read number of cycles does all error checking

Methods inherited from interface
`displayMenu, readCode, readCycles`

Constructors

Menu

`public Menu()`

Sets buffered reader to console input

Methods

readCode

`public MenuCodes readCode()`

(continued from last page)

Get user input

Returns:
user selection

displayMenu

```
public void displayMenu()
```

Display menu items to screen

readCycles

```
public int readCycles()
```

If simulation was chosen read number of cycles does all error checking

Returns:
number of cycles to simulate

apexsimulator.main
Class MenuCodes

```
java.lang.Object
   |-
      |-apexsimulator.main.MenuCodes
```

All Implemented Interfaces:
java.io.Serializable, java.lang.Comparable

---

public final class **MenuCodes**
extends java.lang.Enum

This enum defines currently supported user commands
Author:
Roman Kurbanov

Field Summary

public static final	Display
public static final	Exit
public static final	Initialize
public static final	Simulate

Method Summary

static MenuCodes	valueOf(java.lang.String name)
static MenuCodes[]	values()

Fields

Initialize

public static final apexsimulator.main.MenuCodes **Initialize**

Simulate

public static final apexsimulator.main.MenuCodes **Simulate**

(continued from last page)

Display

```
public static final apexsimulator.main.MenuCodes Display
```

Exit

```
public static final apexsimulator.main.MenuCodes Exit
```

Methods

values

```
public static MenuCodes[] values()
```

---

valueOf

```
public static MenuCodes valueOf(java.lang.String name)
```

apexsimulator.main
Interface MenuInterface

All Known Implementing Classes:
Menu

---

public interface **MenuInterface**
extends

Interface for interacting with menu
Author:
Roman Kurbanov

Method Summary

`abstract void`	`displayMenu()` Display menu items to screen
`abstract MenuCodes`	`readCode()` Get user input
`abstract int`	`readCycles()` If simulation was chosen read number of cycles does all error checking

---

Methods

readCode

`public abstract MenuCodes readCode()`

Get user input

Returns:
user selection

---

displayMenu

`public abstract void displayMenu()`

Display menu items to screen

---

readCycles

`public abstract int readCycles()`

If simulation was chosen read number of cycles does all error checking

Returns:
number of cycles to simulate

Package
apexsimulator.components

Package with all necessary technical components for datapath of the APEX

apexsimulator.components
Class Datapath

```
java.lang.Object
   |
   +-apexsimulator.components.Datapath
```

All Implemented Interfaces:
DatapathInterface

---

public class **Datapath**
extends java.lang.Object
implements DatapathInterface

This class combines all the simulated components into a working CPU datapath
Author:
Roman Kurbanov

Constructor Summary

public	Datapath()

Method Summary

void	display() Displays the contents of each stage in the pipeline and the contents of the first 100 memory locations containing data, starting with address 0.
void	initialize() Initializes the simulator state, sets the PC of the fetch stage to point to the first instruction.
void	simulate(int n) simulates the number of cycles specified as "n" and waits.

Methods inherited from interface
display, initialize, simulate

Constructors

Datapath

```
public Datapath()
```

Methods

initialize

```
public void initialize()
```

(continued from last page)

Initializes the simulator state, sets the PC of the fetch stage to point to the first instruction. Reloads current execution.

simulate

```
public void simulate(int n)
```

simulates the number of cycles specified as "n" and waits. Simulation can stop earlier if a HALT instruction is encountered and when the HALT instruction is in the WB stage.

Parameters:

`n` - number of cycles

display

```
public void display()
```

Displays the contents of each stage in the pipeline and the contents of the first 100 memory locations containing data, starting with address 0. displays the contents of the architectural and physical registers, what is inside each FU, the rename table, the IQ and ROB, the free list of registers and all structures relevant to renaming, including the tag and result value broadcasted etc

apexsimulator.components
Interface DatapathInterface

All Known Implementing Classes:
Datapath

---

public interface **DatapathInterface**
extends

Interface for establishing communication between user and control logic
Author:
Roman Kurbanov

Method Summary

abstract void	`display()` Displays the contents of each stage in the pipeline and the contents of the first 100 memory locations containing data, starting with address 0.
abstract void	`initialize()` Initializes the simulator state, sets the PC of the fetch stage to point to the first instruction.
abstract void	`simulate(int n)` simulates the number of cycles specified as "n" and waits.

Methods

initialize

```
public abstract void initialize()
```

Initializes the simulator state, sets the PC of the fetch stage to point to the first instruction. Reloads current execution.

---

simulate

```
public abstract void simulate(int n)
```

simulates the number of cycles specified as "n" and waits. Simulation can stop earlier if a HALT instruction is encountered and when the HALT instruction is in the WB stage.

Parameters:
n - number of cycles

---

display

```
public abstract void display()
```

Displays the contents of each stage in the pipeline and the contents of the first 100 memory locations containing data, starting with address 0. displays the contents of the architectural and physical registers, what is inside each FU, the rename table, the IQ and ROB, the free list of registers and all structures relevant to renaming, including the tag and result value broadcasted etc

apexsimulator.components
Interface DisplayInterface

All Known Implementing Classes:
Memory, RegisterFile, Pipeline

public interface **DisplayInterface**
extends

Interface for all components that need to display some results at some point.
Author:
Roman Kurbanov

Method Summary

abstract void	display() Prints status to console

Methods

display

```
public abstract void display()
```

Prints status to console

apexsimulator.components
Class IssueQueue

```
java.lang.Object
   |
   +-apexsimulator.components.IssueQueue
```

public class **IssueQueue**
extends java.lang.Object

Author:
Roman Kurbanov

Field Summary

public	issueQueue
public	size

Method Summary

void	reload()

Fields

issueQueue

```
public apexsimulator.components.instructions.Instruction issueQueue
```

size

```
public int size
```

Methods

reload

```
public void reload()
```

apexsimulator.components
Interface LoaderInterface

All Known Implementing Classes:
Memory

public interface **LoaderInterface**
extends

This interface defines methods for the program loader. Different implementations can exist.
Author:
Roman Kurbanov

Method Summary

abstract void	loadProgram() Attempts to load instructions into memory

Methods

loadProgram

```
public abstract void loadProgram()
```

Attempts to load instructions into memory

apexsimulator.components
Class Pipeline

```
java.lang.Object
   |
   +-apexsimulator.components.Pipeline
```

All Implemented Interfaces:
DisplayInterface

---

public class **Pipeline**
extends java.lang.Object
implements DisplayInterface

This class contains all the stages, latches and common clock
Author:
Roman Kurbanov

Constructor Summary

public	Pipeline()

Method Summary

void	display() Prints status to console
void	nextCycle()
void	reload()

Methods inherited from interface
display

Constructors

Pipeline

```
public Pipeline()
```

Methods

reload

```
public void reload()
```

nextCycle

```
public void nextCycle()
```

display

```
public void display()
```

Prints status to console

apexsimulator.components
Class Predictor

```
java.lang.Object
   |
   +-apexsimulator.components.Predictor
```

public class **Predictor**
extends java.lang.Object

Predictor class
Author:
Roman Kurbanov

Constructor Summary

public	Predictor()

Method Summary

static boolean	predict(int offset) Determines prediction based on offset

Constructors

Predictor

```
public Predictor()
```

Methods

predict

```
public static boolean predict(int offset)
```

Determines prediction based on offset

Parameters:
`offset` - integer value

Returns:
prediction

apexsimulator.components
Class Queues

```
java.lang.Object
   |
   +-apexsimulator.components.Queues
```

public class **Queues**
extends java.lang.Object

Issue queue and load/store queue
Author:
Roman Kurbanov

Constructor Summary

public	Queues()

Method Summary

void	addInstruction(Instruction instrIn)
boolean	canAdd(Instruction instrIn)
void	display() output contents to the console
boolean	fullIq()
boolean	fullLsq()
Instruction	getVfu0Instruction()
Instruction	getVfu1Instruction()
Instruction	getVfu2Instruction()
Instruction	getVfu3Instruction()
void	reload() clears the queues

Constructors

Queues

public **Queues**()

Methods

fullIq

```
public boolean fullIq()
```

fullLsq

```
public boolean fullLsq()
```

canAdd

```
public boolean canAdd(Instruction instrIn)
```

addInstruction

```
public void addInstruction(Instruction instrIn)
```

reload

```
public void reload()
```

clears the queues

display

```
public void display()
```

output contents to the console

getVfu0Instruction

```
public Instruction getVfu0Instruction()
```

getVfu1Instruction

```
public Instruction getVfu1Instruction()
```

getVfu2Instruction

```
public Instruction getVfu2Instruction()
```

(continued from last page)

getVfu3Instruction

```
public Instruction getVfu3Instruction()
```

apexsimulator.components
Class ROB

```
java.lang.Object
   |
   +-apexsimulator.components.ROB
```

public class **ROB**
extends java.lang.Object

Reorder buffer
Author:
Roman Kurbanov

Constructor Summary

public	ROB()

Method Summary

void	addInstruction(Instruction instrIn)
void	display() output contents to the console
boolean	empty()
boolean	full()
void	reload() clears the rob
void	retire() Commit latest instruction at the head
void	revert(Instruction brIn) In case of misprediction or unconditional branching, remove wrongly dispatched items

Constructors

ROB

```
public ROB()
```

Methods

full

```
public boolean full()
```

(continued from last page)

empty

```
public boolean empty()
```

addInstruction

```
public void addInstruction(Instruction instrIn)
```

retire

```
public void retire()
```

Commit latest instruction at the head

revert

```
public void revert(Instruction brIn)
```

In case of misprediction or unconditional branching, remove wrongly dispatched items

Parameters:

`brIn` - branching instruction

reload

```
public void reload()
```

clears the rob

display

```
public void display()
```

output contents to the console

Package

apexsimulator.components.stages

apexsimulator.components.stages
Class Decode1

```
java.lang.Object
   |
   +-apexsimulator.components.stages.Decode1
```

All Implemented Interfaces:
StageInterface

---

public class **Decode1**
extends java.lang.Object
implements StageInterface

Decode/Dispatch stage 1 Determines the type of instruction and checks if name is correct and number of arguments is correct
Author:
Roman Kurbanov

Constructor Summary

public	Decode1()

Method Summary

void	clear() Clears stage
void	display() Prints status to console
void	nextCycle() Clock cycle received
void	setId(int id) Id of stage

Methods inherited from interface

clear, display, nextCycle, setId

Constructors

Decode1

public **Decode1**()

Methods

(continued from last page)

nextCycle

```
public void nextCycle()
```

Clock cycle received

---

clear

```
public void clear()
```

Clears stage

---

display

```
public void display()
```

Prints status to console

---

setId

```
public void setId(int id)
```

Id of stage

Parameters:

`id` - unique number of the stage

apexsimulator.components.stages
Class Decode2

```
java.lang.Object
   |
   +-apexsimulator.components.stages.Decode2
```

All Implemented Interfaces:
StageInterface

---

public class **Decode2**
extends java.lang.Object
implements StageInterface

Decode/Dispatch stage 2 Dispatches instruction if all renames were successfull otherwise does nothing and stalls pipeline. Execution stage will consume and add it to appropriate queue

Author:
Roman Kurbanov

Constructor Summary

public	Decode2()

Method Summary

void	clear() Clears stage
void	display() Displays stage
void	nextCycle() Clock cycle received
void	setId(int id) Id of stage

Methods inherited from interface

clear, display, nextCycle, setId

Constructors

Decode2

public **Decode2**()

Methods

(continued from last page)

nextCycle

```
public void nextCycle()
```

Clock cycle received

---

clear

```
public void clear()
```

Clears stage

---

display

```
public void display()
```

Displays stage

---

setId

```
public void setId(int id)
```

Id of stage

Parameters:

`id` - unique number of the stage

apexsimulator.components.stages
Class Execution

```
java.lang.Object
   |
   +-apexsimulator.components.stages.Execution
```

All Implemented Interfaces:
StageInterface

---

public class **Execution**
extends java.lang.Object
implements StageInterface

Execution stage. Responsible for manipulating issue and load/store queues as well ass reorder buffer
Author:
Roman Kurbanov

Constructor Summary

public	Execution()

Method Summary

void	clear() Clears stage
void	display() Displays stage
void	nextCycle() Clock cycle received
void	setId(int id) Id of stage

Methods inherited from interface

clear, display, nextCycle, setId

Constructors

Execution

public **Execution**()

Methods

(continued from last page)

nextCycle

```
public void nextCycle()
```

Clock cycle received

---

clear

```
public void clear()
```

Clears stage

---

display

```
public void display()
```

Displays stage

---

setId

```
public void setId(int id)
```

Id of stage

Parameters:

`id` - unique number of the stage

apexsimulator.components.stages
Class Fetch1

```
java.lang.Object
   |
   +-apexsimulator.components.stages.Fetch1
```

All Implemented Interfaces:
StageInterface

---

public class **Fetch1**
extends java.lang.Object
implements StageInterface

Fetch1 stage Sets PC for instruction and advances global fetch PC value
Author:
Roman Kurbanov

Constructor Summary

public	`Fetch1()` Gets instance of register file to figure out program counter

Method Summary

void	`clear()` Clears stage
void	`display()` Prints status to console
void	`nextCycle()` Fetches instruction from memory
void	`setId(int id)` Id of stage

Methods inherited from interface

`clear, display, nextCycle, setId`

Constructors

Fetch1

`public Fetch1()`

Gets instance of register file to figure out program counter

Methods

(continued from last page)

nextCycle

```
public void nextCycle()
```

Fetches instruction from memory

---

clear

```
public void clear()
```

Clears stage

---

display

```
public void display()
```

Prints status to console

---

setId

```
public void setId(int id)
```

Id of stage

Parameters:

`id` - unique number of the stage

apexsimulator.components.stages
Class Fetch2

```
java.lang.Object
   |
   +-apexsimulator.components.stages.Fetch2
```

All Implemented Interfaces:
StageInterface

---

public class **Fetch2**
extends java.lang.Object
implements StageInterface

Fetch2 stage fetches instruction from memory
Author:
Roman Kurbanov

Constructor Summary

public	`Fetch2()` Gets instance of memory

Method Summary

void	`clear()` Clears stage
void	`display()` Prints status to console
void	`nextCycle()` Fetches instruction from memory
void	`setId(int id)` Id of stage

Methods inherited from interface

`clear, display, nextCycle, setId`

Constructors

Fetch2

`public Fetch2()`

Gets instance of memory

Methods

(continued from last page)

nextCycle

```
public void nextCycle()
```

Fetches instruction from memory

---

clear

```
public void clear()
```

Clears stage

---

display

```
public void display()
```

Prints status to console

---

setId

```
public void setId(int id)
```

Id of stage

Parameters:

`id` - unique number of the stage

apexsimulator.components.stages
Interface StageInterface

All Known Implementing Classes:
Decode1, Decode2, Execution, Fetch1, Fetch2

---

public interface **StageInterface**
extends

Interface for a typical stage in the pipeline
Author:
Roman Kurbanov

Method Summary

abstract void	clear() Clears stage
abstract void	display() Displays stage
abstract void	nextCycle() Clock cycle received
abstract void	setId(int id) Id of stage

Methods

nextCycle

```
public abstract void nextCycle()
```

Clock cycle received

---

clear

```
public abstract void clear()
```

Clears stage

---

display

```
public abstract void display()
```

Displays stage

---

setId

```
public abstract void setId(int id)
```

Id of stage

(continued from last page)

Parameters:

`id` - unique number of the stage

Package

apexsimulator.components.functionunits

apexsimulator.components.functionunits
Class BlankFU

```
java.lang.Object
   |
   +-apexsimulator.components.functionunits.BlankFU
```

public class **BlankFU**
extends java.lang.Object

Function unit for NOPs and HALT
Author:
Roman Kurbanov

Constructor Summary

public	BlankFU(Queues queues)

Method Summary

void	nextCycle()

Constructors

BlankFU

```
public BlankFU(Queues queues)
```

Methods

nextCycle

```
public void nextCycle()
```

apexsimulator.components.functionunits
Interface FunctionUnitInterface

public interface **FunctionUnitInterface**
extends

This class defines public interface for all function units
Author:
Roman Kurbanov

Method Summary

abstract void	nextCycle()
abstract boolean	ready()

Methods

ready

```
public abstract boolean ready()
```

nextCycle

```
public abstract void nextCycle()
```

apexsimulator.components.functionunits
Class IntegerFU

```
java.lang.Object
   |
   +-apexsimulator.components.functionunits.IntegerFU
```

public class **IntegerFU**
extends java.lang.Object

Integer function unit for instructions ADD, SUB, MOVC, AND, OR, EX-OR BZ, BNZ, JUMP, BAL
Author:
Roman Kurbanov

Field Summary

public	available

Constructor Summary

public	IntegerFU(Queues queues, ROB rob)

Method Summary

void	display()
void	nextCycle()
void	reload()

Fields

available

```
public boolean available
```

Constructors

IntegerFU

```
public IntegerFU(Queues queues,
                 ROB rob)
```

Methods

(continued from last page)

nextCycle

```
public void nextCycle()
```

---

reload

```
public void reload()
```

---

display

```
public void display()
```

apexsimulator.components.functionunits
Class MemoryFU

```
java.lang.Object
   |
   +-apexsimulator.components.functionunits.MemoryFU
```

public class **MemoryFU**
extends java.lang.Object

Memory function unit
Author:
Roman Kurbanov

Field Summary

public	available

Constructor Summary

public	MemoryFU(Queues queues)

Method Summary

void	display()
void	nextCycle()
void	reload()

Fields

available

```
public boolean available
```

Constructors

MemoryFU

```
public MemoryFU(Queues queues)
```

Methods

(continued from last page)

nextCycle

```
public void nextCycle()
```

---

reload

```
public void reload()
```

---

display

```
public void display()
```

apexsimulator.components.functionunits
Class MultiplierFU

```
java.lang.Object
   |
   +-apexsimulator.components.functionunits.MultiplierFU
```

public class **MultiplierFU**
extends java.lang.Object

Multiplier Function Unit
Author:
Roman Kurbanov

Field Summary

public	available

Constructor Summary

public	MultiplierFU(Queues queues)

Method Summary

void	display()
void	nextCycle()
void	reload()

Fields

available

public boolean **available**

Constructors

MultiplierFU

public **MultiplierFU**(Queues queues)

Methods

(continued from last page)

nextCycle

```
public void nextCycle()
```

---

reload

```
public void reload()
```

---

display

```
public void display()
```

Package

apexsimulator.components.registerfile

apexsimulator.components.registerfile
Class GlobalVars

```
java.lang.Object
   |
   +-apexsimulator.components.registerfile.GlobalVars
```

public class **GlobalVars**
extends java.lang.Object

Class with static variables that serve as global constants and variable for datapath internal usage
Author:
Roman Kurbanov

Field Summary

public static	execution_completed
public static final	INSTR_START Value: **20000**
public static final	IQ_SIZE Value: **8**
public static final	LSQ_SIZE Value: **8**
public static final	PHYS_REG_COUNT Value: **16**
public static	pipeline_frozen
public static final	ROB_SIZE Value: **16**

Constructor Summary

public	GlobalVars()

Fields

INSTR_START

public static final int **INSTR_START**

Constant value: **20000**

execution_completed

```
public static boolean execution_completed
```

pipeline_frozen

```
public static boolean pipeline_frozen
```

PHYS_REG_COUNT

```
public static final int PHYS_REG_COUNT
```

Constant value: **16**

ROB_SIZE

```
public static final int ROB_SIZE
```

Constant value: **16**

IQ_SIZE

```
public static final int IQ_SIZE
```

Constant value: **8**

LSQ_SIZE

```
public static final int LSQ_SIZE
```

Constant value: **8**

Constructors

GlobalVars

```
public GlobalVars()
```

apexsimulator.components.registerfile
Class RAT

```
java.lang.Object
   |
   +-apexsimulator.components.registerfile.RAT
```

public class **RAT**
extends java.lang.Object

Rename table that will keep track of free list, allocating free register, committing register to ARF, restoring PRF on branch misprediction, and maintaining a register alias table.
Author:
Roman Kurbanov

Constructor Summary

public	`RAT()` Constructs empty RAT table.

Method Summary

Register	`assignFree(ArchRegisterEnum archRegIn)`
void	`commit(ArchRegisterEnum archRegIn, Register regIn)`
void	`deleteRename(ArchRegisterEnum archRegIn, Register reg)` Removes specific mapping from the RAT, useful for branches
void	`display()`
Register	`getLatest(ArchRegisterEnum archRegIn)` returns valid mapping
void	`reload()` resets internal structure

Constructors

RAT

`public RAT()`

Constructs empty RAT table.

Methods

reload

`public void reload()`

resets internal structure

getLatest

```
public Register getLatest(ArchRegisterEnum archRegIn)
```

returns valid mapping

Parameters:

`archRegIn` - name of register

Returns:

returns valid mapping or null if mapping doesn't exist

assignFree

```
public Register assignFree(ArchRegisterEnum archRegIn)
```

commit

```
public void commit(ArchRegisterEnum archRegIn,
         Register regIn)
```

display

```
public void display()
```

deleteRename

```
public void deleteRename(ArchRegisterEnum archRegIn,
         Register reg)
```

Removes specific mapping from the RAT, useful for branches

Parameters:

`archRegIn` - name of register

`reg` - Register to be deleted from rename

apexsimulator.components.registerfile
Class Register

```
java.lang.Object
   |
   +-apexsimulator.components.registerfile.Register
```

public class **Register**
extends java.lang.Object

This class packs all the fields needed for register

No exceptions will be generated in case of wrong access Also can be used as a flag
Author:
Roman Kurbanov

Field Summary

public	physId

Constructor Summary

public	Register() Creates default register which is not valid and not used beforehand

Method Summary

int	getValue()
boolean	isUsed()
boolean	isValid()
void	reload() Resets internal state to default
void	setUsed(boolean used)
void	setValid(boolean valid)
void	setValue(int value)

Fields

physId

public int **physId**

Constructors

Register

```
public Register()
```

Creates default register which is not valid and not used beforehand

Methods

reload

```
public void reload()
```

Resets internal state to default

---

isValid

```
public boolean isValid()
```

---

setValid

```
public void setValid(boolean valid)
```

---

getValue

```
public int getValue()
```

---

setValue

```
public void setValue(int value)
```

---

isUsed

```
public boolean isUsed()
```

---

setUsed

```
public void setUsed(boolean used)
```

apexsimulator.components.registerfile
Class RegisterFile

```
java.lang.Object
   |
   +-apexsimulator.components.registerfile.RegisterFile
```

All Implemented Interfaces:
DisplayInterface

---

public class **RegisterFile**
extends java.lang.Object
implements DisplayInterface

Register file implemented as singleton Has wide variety of responsibilities, but generally encapsulates everything that is needed for register operation. Also manipulates global var variables

Author:
Roman Kurbanov

Field Summary

public	forwardingAvailable
public	production
public	rat

Method Summary

void	clearLatches()
void	display() Prints contents of register file to console
int	getCommitedPC()
int	getFetchPC()
static RegisterFile	getInstance() creates or returns previously created unique instance of the class this method is thread safe
void	reload() resets register file to default value
void	setCommitedPC(int commitedPC)
void	setFetchPC(int fetchPC)

Methods inherited from interface

display

Fields

forwardingAvailable

```
public boolean forwardingAvailable
```

rat

```
public apexsimulator.components.registerfile.RAT rat
```

production

```
public apexsimulator.components.instructions.Instruction production
```

Methods

getInstance

```
public static RegisterFile getInstance()
```

creates or returns previously created unique instance of the class this method is thread safe

Returns:
unique instance of Memory class

reload

```
public void reload()
```

resets register file to default value

getFetchPC

```
public int getFetchPC()
```

setFetchPC

```
public void setFetchPC(int fetchPC)
```

getCommitedPC

```
public int getCommitedPC()
```

setCommitedPC

```
public void setCommitedPC(int commitedPC)
```

display

```
public void display()
```

Prints contents of register file to console

clearLatches

```
public void clearLatches()
```

Package
apexsimulator.components.memory

apexsimulator.components.memory
Class Memory

```
java.lang.Object
   |
   +-apexsimulator.components.memory.Memory
```

All Implemented Interfaces:
DisplayInterface, MemoryInterface, LoaderInterface

---

public class **Memory**
extends java.lang.Object
implements LoaderInterface, MemoryInterface, DisplayInterface

This class implements i-cache and d-cache Taking into account its uniqueness across the program this class is implemented as Singleton

Author:
Roman Kurbanov

Field Summary

public	iCacheSize

Method Summary

void	display() Display contents of first 100 memory locations
java.lang.String	fetch(int address) Returns an instruction at particular address halts the execution if wrong address is passed
static Memory	getInstance() creates or returns previously created unique instance of the class this method is thread safe
void	loadProgram() Attempts to load instructions into memory
int	readData(int address) Returns data read from memory
void	reload() Purges existing data in d-cache and reloads instructions in i-cache
void	setFileName(java.lang.String fileName) Sets filename to read instructions from
void	writeMem(int address, int val) Writes data to data cache

Methods inherited from interface

loadProgram

Methods inherited from interface

display, fetch, readData, reload, setFileName

Methods inherited from interface
display

Fields

iCacheSize

public int **iCacheSize**

Methods

getInstance

public static Memory **getInstance**()

creates or returns previously created unique instance of the class this method is thread safe

Returns:
unique instance of Memory class

loadProgram

public void **loadProgram**()

Attempts to load instructions into memory

fetch

public java.lang.String **fetch**(int address)

Returns an instruction at particular address halts the execution if wrong address is passed

Parameters:
address - addreess of instruction

Returns:
Instruction at a particular address

readData

public int **readData**(int address)

Returns data read from memory

Parameters:
address - address of memory area

Returns:
integer value stored in memory

writeMem

```
public void writeMem(int address,
         int val)
```

Writes data to data cache

Parameters:

`address` - address of memory cell
`val` - value to be written

reload

```
public void reload()
```

Purges existing data in d-cache and reloads instructions in i-cache

setFileName

```
public void setFileName(java.lang.String fileName)
```

Sets filename to read instructions from

Parameters:

`fileName` - filename with instructions

display

```
public void display()
```

Display contents of first 100 memory locations

apexsimulator.components.memory
Interface MemoryInterface

All Known Implementing Classes:
Memory

---

public interface **MemoryInterface**
extends

Interface that fetches instructions at particular address
Author:
Roman Kurbanov

Method Summary

`abstract void`	`display()` Display contents of first 100 memory locations
`abstract java.lang.String`	`fetch(int address)` Returns text representation of instruction at particular address halts the execution if wrong address is passed
`abstract int`	`readData(int address)` Returns data read from memory
`abstract void`	`reload()` Purges existing data in d-cache and reloads instructions in i-cache
`abstract void`	`setFileName(java.lang.String fileName)` Sets filename to read instructions from

Methods

fetch

`public abstract java.lang.String fetch(int address)`

Returns text representation of instruction at particular address halts the execution if wrong address is passed

Parameters:
`address` - addreess of instruction

Returns:
Instruction at a particular address

---

readData

`public abstract int readData(int address)`

Returns data read from memory

Parameters:
`address` - address of memory area

(continued from last page)

Returns:
integer value stored in memory

reload

```
public abstract void reload()
```

Purges existing data in d-cache and reloads instructions in i-cache

setFileName

```
public abstract void setFileName(java.lang.String fileName)
```

Sets filename to read instructions from

Parameters:
`fileName` - filename with instructions

display

```
public abstract void display()
```

Display contents of first 100 memory locations

Package

apexsimulator.components.instructions

apexsimulator.components.instructions
Class Instruction

```
java.lang.Object
   |
   +-apexsimulator.components.instructions.Instruction
```

public class **Instruction**
extends java.lang.Object

This class defines instruction type and its common characteristics Fields will be updated as the instruction moves in the pipeline
Author:
Roman Kurbanov

Field Summary

public	operands
public	prediction

Constructor Summary

public	Instruction() Minimal version.

Method Summary

InstructionsEnum	getInstr()
java.lang.String	getInstruction() Getter for instruction field
int	getPC()
InstructionStatus	getStatus()
void	setInstr(InstructionsEnum instr)
void	setInstruction(java.lang.String instruction)
void	setPC(int PC)
void	setStatus(InstructionStatus status)

Fields

(continued from last page)

operands

```
public apexsimulator.components.instructions.Operands operands
```

---

prediction

```
public boolean prediction
```

Constructors

Instruction

```
public Instruction()
```

Minimal version. Everything is initialized in stages

Methods

getInstr

```
public InstructionsEnum getInstr()
```

---

setInstr

```
public void setInstr(InstructionsEnum instr)
```

---

getStatus

```
public InstructionStatus getStatus()
```

---

setStatus

```
public void setStatus(InstructionStatus status)
```

---

getPC

```
public int getPC()
```

---

setPC

```
public void setPC(int PC)
```

getInstruction

```
public java.lang.String getInstruction()
```

Getter for instruction field

Returns:
string version of instruction

setInstruction

```
public void setInstruction(java.lang.String instruction)
```

apexsimulator.components.instructions
Class Operands

```
java.lang.Object
   |
   +-apexsimulator.components.instructions.Operands
```

public class **Operands**
extends java.lang.Object

Class that will hold tags, values and operands for an instruction Number of them will depend on instruction type
Author:
Roman Kurbanov

Field Summary

public	ops
public	output
public	regNames
public	waiting

Constructor Summary

public	Operands(java.lang.String[] tokens, InstructionsEnum type) Sets of overloaded constructors for different type of arguments

Method Summary

void	getRenames()
java.lang.String	opsToString() Prints values themselves not register names
void	populateOps()
boolean	readyToDispatch() Checks if all values were received and instruction is ready for execution
boolean	readyToIssue() Checks if instruction is ready to be added to issue queue
void	releasePhysReg()
void	reset() Releases all acquired resources and renames of squashed instruction
java.lang.String	toString()

Fields

ops

```
public int ops
```

regNames

```
public apexsimulator.util.ArchRegisterEnum regNames
```

output

```
public apexsimulator.components.registerfile.Register output
```

waiting

```
public apexsimulator.components.registerfile.Register waiting
```

Constructors

Operands

```
public Operands(java.lang.String[] tokens,
                InstructionsEnum type)
```

Sets of overloaded constructors for different type of arguments

Parameters:

`tokens` - string tokens
`type` - type of instruction

Methods

toString

```
public java.lang.String toString()
```

opsToString

```
public java.lang.String opsToString()
```

Prints values themselves not register names

Returns:

formatted string with data

populateOps

```
public void populateOps()
```

releasePhysReg

```
public void releasePhysReg()
```

readyToDispatch

```
public boolean readyToDispatch()
```

Checks if all values were received and instruction is ready for execution

Returns:
true is all operands are ready, false otherwise

getRenames

```
public void getRenames()
```

readyToIssue

```
public boolean readyToIssue()
```

Checks if instruction is ready to be added to issue queue

Returns:
false if renames weren't acquired

reset

```
public void reset()
```

Releases all acquired resources and renames of squashed instruction

Index

J

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z