Computer Organization and Architecture
Programming Project 2

We have extended the simulator that we have developed for Project 1 with a renaming mechanism that uses a unified register file, a centralized IQ and a ROB.

The wakeup signal for a function unit is generated one cycle before the function unit completes to support back-to-back execution. Ties for selection of a specific function unit are broken using a FIFO policy that selects the instruction dispatched earlier. We have implemented the FIFO by adding an IQ entry field that holds the cycle in which the instruction was dispatched. Speculative execution is not supported.

The following function units are used and all function units, excepting the branch FU, has a writeback stage with a one cycle latency:

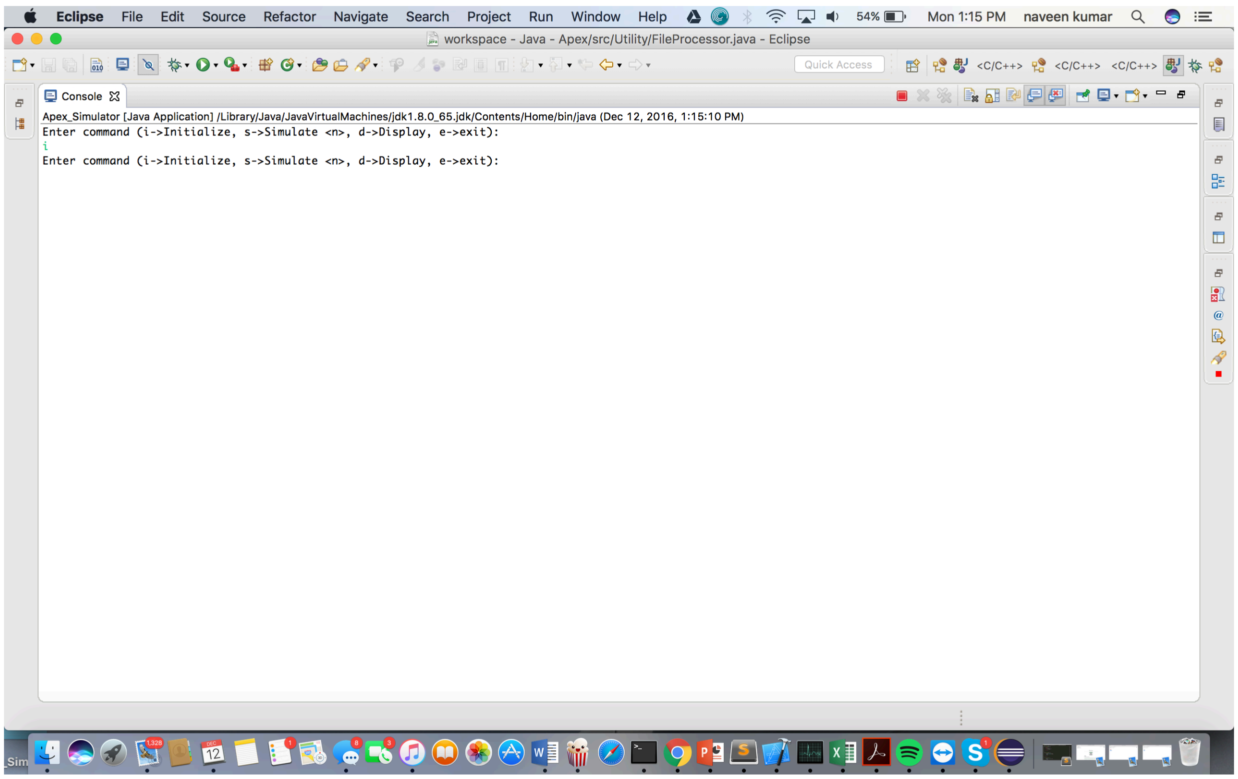
- A two-stage pipelined integer ALU (two stages, one cycle per stage) implementing all arithmetic instructions excepting a multiply. This function unit also implements the MOVC instruction by adding an implicit zero value to the literal in MOVC and writing it to the destination.
- A non-pipelined multiplication unit with a latency of 4 cycles that implements the multiply operation.
- A single cycle branch FU that computes the target address and decides whether to branch or not. This function unit also implements the JUMP and BAL instruction.
- A two-stage pipelined LSFU (one cycle per stage) implementing the LOAD and STORE instructions. LSFU generates memory address (stage 1 of LSFU), performs TLB lookup (LSFU 2nd stage) and then accesses cache when LOAD or STORE is at the head of the ROB. Assume Cache access performed by LSFU retrieves data in one cycle. There is no bypassing of earlier STORES by a later LOAD. When a LOAD completes, the result is written to the destination via the associated WB stage.

Usage: make (To compile the code)

make run (To execute the code with specified input file() .

Project: Developed in Java 8 and Eclipse(IDE).

Implementation: Most of the project documentation can be extracted with Javadoc. Classes are split into according packages. All stages are implemented as discrete entities that interact between each other Execution stage is the biggest stage in terms of design since it encapsulates Multi-Functional Units: All operations are entered through number options. If simulation reaches end of program it displays results. Sample run is displayed in the image below


Eclipse File Edit Source Refactor Navigate Search Project Run Window Help
Mon 1:15 PM naveen kumar
workspace - Java - Apex/src/Utility/FileProcessor.java - Eclipse
Quick Access
Console

```
Apex_Simulator [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_65.jdk/Contents/Home/bin/java (Dec 12, 2016, 1:15:10 PM)
Enter command (i->Initialize, s->Simulate <n>, d->Display, e->exit):
i
Enter command (i->Initialize, s->Simulate <n>, d->Display, e->exit):
```

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 Apex_Simulator

Interface Summary

ProcessListener

ProcessListener is the interface for process and pcValue methods in different stages.

Class Summary

Apex_Simulator

CycleListener

IQ

Memory

Processor

ROB

UnifiedRegisterFile

Apex_Simulator

Class Apex_Simulator

```
java.lang.Object
    |
    +--Apex_Simulator.Apex_Simulator
```

< Constructors > < Methods >

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

Constructors

Apex_Simulator

```
public  Apex_Simulator()
```

Methods

display

```
public static void display()
```

display method displays the status of last simulation(displays each stage,reg & mem informations)

formatDisp

```
public static java.lang.String formatDisp(Constants.Stage stage)
```

formatDisp method gets stage constants and format the relevant display information for given stage

Parameters:

stage - of type Constants.Stage

main

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

main method gets instruction file and initiate the apex simulator program and calls the process method,

Parameters:

args - instruction text file is the 1st argument

Apex_Simulator

Class CycleListener

```
java.lang.Object
    |
    +--Apex_Simulator.CycleListener
```

< Fields > < Constructors > < Methods >

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

Fields

cycle

```
public java.lang.Integer cycle
```

Constructors

CycleListener

```
public  CycleListener(Processor processor)
```

Constructor for CycleListener counts and keeps track of the cycle, instruction address, results of each stages.

Parameters:

processor - object of the processor.

Methods

ChangeCycle

```
public void ChangeCycle(CycleListener cL)
```

Writes temporary result of the different to the final result of different stages when cycle is changed.

Parameters:

cL - CycleListener object from processor.

---

read

```
public java.lang.Long read()
```

reads the final result just before incrementing the cycle.

---

temRread

```
public java.lang.Long temRread()
```

reads the temporary result of the different stages in middle of the cycle.

---

write

```
public void write(long result)
```

Writes the temporary result of the different stages in middle of the cycle.

Apex_Simulator

Class IQ

```
java.lang.Object
    |
    +--Apex_Simulator.IQ
```

< Fields > < Constructors > < Methods >

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

Fields

processor

```
public Processor processor
```

Constructors

IQ

```
public  IQ()
```

Constructor for IQ initializes the IQEntry.

Methods

flushIQEntry

```
public void flushIQEntry(int index)
```

readIQEntry

```
public Instruction readIQEntry(int index)
```

removeIQEntry

```
public void removeIQEntry(int index)
```

---

writeIQEntry

```
public boolean writeIQEntry(Instruction data)
```

Apex_Simulator

Class Memory

```
java.lang.Object
   |
   +--Apex_Simulator.Memory
```

< Constructors > < Methods >

---

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

Constructors

Memory

```
public  Memory(java.lang.String file)
```

Constructor for Memory initializes the Memory.

Parameters:

file - of string type to be processed and relevant results are stored in instruction array list in memory.

Methods

clearInstructions

```
public void clearInstructions()
```

clearInstructions method clears the instruction in the memory

---

getCachedData

`public long getCachedData(int memLoc)`

---

getInstruction

`public Instruction getInstruction(long index)`

getInstruction method calculates and return the instruction to be fetched along with the address

Parameters:

index - current instruction address to be fetched

Returns:

instruction contains instruction and instruction address

---

readCacheMem

`public long readCacheMem(int mem)`

---

readFirst100

`public java.util.List readFirst100()`

readFirst100 method reads the first 100 memory locations

Parameters:

stage - of type Constants.Stage

---

readMem

`public long readMem(int index)`

readMem method reads the value for the given memory index

Parameters:

index - of int type to specify memory location

---

readMemory

```
public java.util.List readMemory(int startIndex,
                                 int lastIndex)
```

readMemory method reads the memory locations from start index to the last index

Parameters:

startIndex - defines the start index from which the memory need to be read
lastIndex - defines the last index from which the memory need to be read

---

writeCacheMem

```
public void writeCacheMem(int mem,
                          int data)
```

---

writeMem

```
public long writeMem(int index,
                     long data)
```

writeMem method writes the data to the memory location

Parameters:

index - of type int which specify memory location
data - of type long that will be stored in given memory location

Returns:

of long type, contains written memory location

Apex_Simulator

Interface ProcessListener

< Methods >

---

public interface **ProcessListener**

ProcessListener is the interface for process and pcValue methods in different stages.

Methods

pcValue

```
public java.lang.Long pcValue()
```

---

process

```
public void process()
```

Apex_Simulator

Class Processor

```
java.lang.Object
    |
    +--Apex_Simulator.Processor
```

< Fields > < Constructors > < Methods >

---

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

Fields

INS_COUNT

```
public static int INS_COUNT
```

---

branchFU

```
public BranchFU branchFU
```

---

cL

```
public CycleListener cL
```

---

cycleListener

```
public java.util.List cycleListener
```

decode

```
public Decode decode
```

dispatch

```
public Dispatch dispatch
```

fALU1

```
public ALU1 fALU1
```

fALU2

```
public ALU2 fALU2
```

fetch

```
public Fetch fetch
```

iQ

```
public IQ iQ
```

isBranchZ

```
public boolean isBranchZ
```

isHalt

```
public boolean isHalt
```

isStalled

```
public boolean isStalled
```

isZero

`public boolean **isZero**`

---

lSFU1

`public LSFU1 lSFU1`

---

lSFU2

`public LSFU2 lSFU2`

---

memory

`public Memory memory`

---

mulResultFoundCheck

`public boolean mulResultFoundCheck`

---

multiplicationFU

`public MultiplicationFU multiplicationFU`

---

processListeners

`public java.util.List processListeners`

---

rOB

`public ROB rOB`

---

rOBCommit

`public ROBCommit rOBCommit`

---

register

`public UnifiedRegisterFile **register**`

---

writeBack

`public WriteBack writeBack`

Constructors

Processor

`public  Processor(java.lang.String file)`

Constructor for Processor initializes the Processor and also all the stages objects, memory, registers.

Parameters:

file - of string type to be processed and relevant results are stored in instruction array list in memory.

Methods

doProcess

`public void doProcess()`

doProcess method performs process for each stage, increments the cycle, sets the isSstallflag (based on stall check logic), and sets the src1Stall and src2Stall flags of the respective decode instruction. The stall check logic checks whether the src1 and src2 of the decode instruction is equal to the destination of the ALU1, ALU2, memory stage instructions.

Apex_Simulator

Class ROB

```
java.lang.Object
    |
    +--Apex_Simulator.ROB
```

< Constructors > < Methods >

---

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

Constructors

ROB

```
public  ROB()
```

Constructor for IQ initializes the IQEntry.

Methods

readROBEntry

```
public Utility.Instruction[] readROBEntry()
```

---

readROBEntry

```
public Instruction readROBEntry(int index)
```

---

removeROBEntry

```
public void removeROBEntry()
```

---

setBranchTaken

```
public void setBranchTaken(Constants.OpCode brnOpcode,
                           boolean isBrnTaken,
                           long brnTrgAdd)
```

---

writeROBEntry

```
public void writeROBEntry(Instruction data)
```

Apex_Simulator

Class UnifiedRegisterFile

```
java.lang.Object
    |
    +--Apex_Simulator.UnifiedRegisterFile
```

< Constructors > < Methods >

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

Constructors

UnifiedRegisterFile

```
public  UnifiedRegisterFile()
```

Constructor for URF initializes the physical registers.

Methods

getAllBackEntTable

```
public Utility.RAT[] getAllBackEntTable()
```

getBackEndPhyReg

```
public long getBackEndPhyReg(int index)
```

getFrontEndPhyReg

```
public long getFrontEndPhyReg(int index)
```

getIsRegValid

```
public boolean getIsRegValid(int index)
```

---

getRegAvailability

```
public boolean getRegAvailability(int index)
```

---

getReg_X

```
public long getReg_X()
```

getReg_X method returns the last register R16 reserved for X register

Returns:

register R16 reserved for register X

---

getZFlag

```
public int getZFlag(int phyReg)
```

---

getZReg

```
public long getZReg()
```

---

readReg

```
public long readReg(int index)
```

readReg method reads the register value from the given register

Parameters:

index - of type int, specifies the register (from R0 to R15) from which the value should be read

---

setAllFrontEntTable

```
public void setAllFrontEntTable(Utility.RAT[] newRAT)
```

---

setBackEndPhyReg

```
public void setBackEndPhyReg(int archReg,
                             int phyReg)
```

---

setFrontEndPhyReg

```
public long setFrontEndPhyReg(int index)
```

---

setIsRegValid

```
public void setIsRegValid(int index,
                          boolean data)
```

---

setRegAvailability

```
public void setRegAvailability(int index,
                               boolean data)
```

---

setReg_X

```
public void setReg_X(long reg_X)
```

setReg_X method sets the last register R16 reserved for X register with given value

Parameters:

reg_X - of type long, value of register R16 reserved for register X

---

setZFlag

```
public void setZFlag(int phyReg,
                     int data)
```

---

setZReg

```
public void setZReg(long data)
```

---

writeReg

```
public void writeReg(int index,
                     long data)
```

writeReg method writes the register value to the relevant register

Parameters:

index - of type int, specifies the register (from R0 to R15) for which the value should be written
data - of type long, contains data or value needed to be written to the given register

Package Stages

Class Summary

ALU1

ALU2

BranchFU

Decode

Delay

Dispatch

Fetch

LSFU1

LSFU2

MemoryStage

MultiplicationFU

ROBCommit

WriteBack

Stages

Class ALU1

```
java.lang.Object
    |
    +--Stages.ALU1
```

All Implemented Interfaces:
ProcessListener

< Fields > < Constructors > < Methods >

public class **ALU1**
extends java.lang.Object
implements ProcessListener

Fields

instruction

```
public Instruction instruction
```

---

pc

```
public CycleListener pc
```

---

processor

```
public Processor processor
```

---

result

```
public CycleListener result
```

Constructors

ALU1

```
public  ALU1(Processor processor)
```

Constructor for ALU1 stage initializes PC(instruction Address), result(like a latch which has results of the stage).

Parameters:

processor - a Processor object.

Methods

clearStage

```
public void clearStage()
```

clearStage method clears the ALU1 stage.

---

pcValue

```
public java.lang.Long pcValue()
```

pcValue method returns the pc Value(instruction address) of the ALU1 stage.

Returns:

long value of the pc Value(instruction address)

---

process

```
public void process()
```

process method performs the ALU1 processes such as stall implementation if stall based on isStall flag from processor, reads the source value from source register from decode stage seeking the required data for processing and implementing forwarding of register to register operations, Load and store operations. Register-to-register instructions: ADD, SUB, MOVC, MUL, AND, OR, EX-OR (all done on the Integer ALU in two cycles(1st cycle here)). You can assume that the result of multiplying two registers will fit into a single register.

---

toString

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

toString method returns the instruction currently in ALU1 as string if instruction is not null or returns the IDLE constants.

Returns:

String of the instruction or IDLE constants

Overrides:

toString in class java.lang.Object

Stages

Class ALU2

```
java.lang.Object
    |
    +--Stages.ALU2
```

All Implemented Interfaces:

ProcessListener

< Fields > < Constructors > < Methods >

---

public class **ALU2**
extends java.lang.Object
implements ProcessListener

Fields

instruction

`public Instruction instruction`

pc

`public CycleListener pc`

processor

`public Processor processor`

Constructors

ALU2

`public  ALU2(Processor processor)`

Constructor for ALU2 stage initializes PC(instruction Address), result(like a latch which has results of the stage).

Parameters:

processor - a Processor object.

Methods

clearStage

`public void clearStage()`

clearStage method clears the ALU2 stage.

pcValue

`public java.lang.Long pcValue()`

pcValue method returns the pc Value(instruction address) of the ALU2 stage.

Returns:

long value of the pc Value(instruction address)

---

process

`public void **process**()`

ALU2 process method performs relevant operations such as register-register (add, sub, .. etc), load, and store and writes the result to the destination register temporarily. Register-to-register instructions: ADD, SUB, MOVC, MUL, AND, OR, EX-OR (all done on the Integer ALU in two cycles(2nd cycle here)). You can assume that the result of multiplying two registers will fit into a single register.

---

toString

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

toString method returns the instruction currently in ALU2 as string if instruction is not null or returns the IDLE constants.

Returns:

String of the instruction or IDLE constants

Overrides:

toString in class java.lang.Object

Stages

Class BranchFU

```
java.lang.Object
    |
    +--Stages.BranchFU
```

All Implemented Interfaces:

ProcessListener

< Fields > < Constructors > < Methods >

---

public class **BranchFU**
extends java.lang.Object
implements ProcessListener

Fields

instruction

`public Instruction **instruction**`

---

pc

```
public CycleListener pc
```

processor

```
public Processor processor
```

Constructors

BranchFU

```
public  BranchFU(Processor processor)
```

Constructor for BranchFU stage initializes PC(instruction Address), result(like a latch which has results of the stage).

Parameters:

processor - a Processor object.

Methods

clearStage

```
public void clearStage()
```

clearStage method clears the BranchFU stage.

pcValue

```
public java.lang.Long pcValue()
```

pcValue method returns the pc Value(instruction address) of the BranchFU stage.

Returns:

long value of the pc Value(instruction address)

process

```
public void process()
```

BranchFU process method performs relevant control operations such as branching (BZ, BNZ, BAL, JUMP), and Halt Control flow instructions: BZ, BNZ, JUMP, BAL, HALT. Instructions following a BZ, BNZ, JUMP and BAL instruction in the pipeline should be flushed on a taken branch. The zero flag (Z) is set only by arithmetic instructions in ALU.

toString

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

toString method returns the instruction currently in BranchFU as string if instruction is not null or returns the IDLE constants.

Returns:

String of the instruction or IDLE constants

Overrides:

toString in class java.lang.Object

Stages

Class Decode

```
java.lang.Object
    |
    +--Stages.Decode
```

All Implemented Interfaces:

ProcessListener

< Fields > < Constructors > < Methods >

public class **Decode**
extends java.lang.Object
implements ProcessListener

Fields

instruction

```
public Instruction instruction
```

pc

```
public CycleListener pc
```

processor

```
public Processor processor
```

Constructors

Decode

```
public  Decode(Processor processor)
```

//false checkin ALU1 Constructor for Decode stage initializes PC(instruction Address), result(like a latch which has results of the stage).

Parameters:

processor - a Processor object.

Methods

clearStage

```
public void clearStage()
```

clearStage method clears the Decode stage.

---

pcValue

```
public java.lang.Long pcValue()
```

pcValue method returns the pc Value(instruction address) of the Decode stage.

Returns:

long value of the pc Value(instruction address)

---

process

```
public void process()
```

Decode process method performs relevant action for halt, stall and decodes the necessary instruction.

---

readSources

```
public void readSources()
```

ReadSources method reads the source registers of the instruction and fetches the same from register file.

---

toString

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

toString method returns the instruction currently in Decode as string if instruction is not null or returns the IDLE constants.

Returns:

String of the instruction or IDLE constants

Overrides:

toString in class java.lang.Object

Stages

Class Delay

```
java.lang.Object
    |
    +--Stages.Delay
```

All Implemented Interfaces:

ProcessListener

< Fields > < Constructors > < Methods >

---

public class **Delay**
extends java.lang.Object
implements ProcessListener

Fields

instruction

```
public Instruction instruction
```

---

pc

```
public CycleListener pc
```

---

processor

```
public Processor processor
```

Constructors

Delay

```
public  Delay(Processor processor)
```

Constructor for Delay stage initializes PC(instruction Address), result(like a latch which has results of the stage).

Parameters:

processor - a Processor object.

Methods

clearStage

```
public void clearStage()
```

clearStage method clears the Delay stage.

---

pcValue

```
public java.lang.Long pcValue()
```

pcValue method returns the pc Value(instruction address) of the Delay stage.

Returns:

long value of the pc Value(instruction address)

---

process

```
public void process()
```

Delay process method acts as an one cycle delay for the Branch FU stage.

---

toString

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

toString method returns the instruction currently in Delay as string if instruction is not null or returns the IDLE constants.

Returns:

String of the instruction or IDLE constants

Overrides:

toString in class java.lang.Object

Stages

Class Dispatch

```
java.lang.Object
   |
   +--Stages.Dispatch
```

All Implemented Interfaces:
ProcessListener

< Fields > < Constructors > < Methods >

public class **Dispatch**
extends java.lang.Object
implements ProcessListener

Fields

instruction

`public Instruction instruction`

pc

`public CycleListener pc`

processor

`public Processor processor`

Constructors

Dispatch

`public  Dispatch(Processor processor)`

Constructor for Decode stage initializes PC(instruction Address), result(like a latch which has results of the stage).

Parameters:

processor - a Processor object.

Methods

clearStage

`public void **clearStage**()`

clearStage method clears the Decode stage.

---

pcValue

`public java.lang.Long **pcValue**()`

pcValue method returns the pc Value(instruction address) of the Decode stage.

Returns:

long value of the pc Value(instruction address)

---

process

`public void **process**()`

//false checkin ALU1 Decode process method performs relevant action for halt, stall and decodes the necessary instruction.

---

readSources

`public void **readSources**()`

ReadSources method reads the source registers of the instruction and fetches the same from register file.

---

toString

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

toString method returns the instruction currently in Decode as string if instruction is not null or returns the IDLE constants.

Returns:

String of the instruction or IDLE constants

Overrides:

toString in class java.lang.Object

Stages

Class Fetch

```
java.lang.Object
    |
    +--Stages.Fetch
```

All Implemented Interfaces:

ProcessListener

< Fields > < Constructors > < Methods >

public class **Fetch**
extends java.lang.Object
implements ProcessListener

Fields

instruction

`public Instruction instruction`

pc

`public CycleListener pc`

processor

`public Processor processor`

Constructors

Fetch

`public  Fetch(Processor processor)`

Constructor for Fetch stage initializes PC(instruction Address), result(like a latch which has results of the stage).

Parameters:

processor - a Processor object.

Methods

clearStage

```
public void clearStage()
```

clearStage method clears the Fetch stage.

---

clearStage

```
public void clearStage(java.lang.Long newFetchAdd)
```

clearStage method gets the new instruction address from branchFU to fetch when the branch is taken.

Parameters:

newFetchAdd - of type long new fetch instruction address.

---

pcValue

```
public java.lang.Long pcValue()
```

pcValue method returns the pc Value(instruction address) of the Fetch stage.

Returns:

long value of the pc Value(instruction address)

---

process

```
public void process()
```

//false checkin ALU1 Fetch process method fetches the next instruction by instruction address from the instructions aray list - get instruction method which process the instruction array list .

---

toString

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

toString method returns the instruction currently in Fetch as string if instruction is not null or returns the IDLE constants.

Returns:

String of the instruction or IDLE constants

Overrides:

toString in class java.lang.Object

Stages

Class LSFU1

```
java.lang.Object
    |
    +--Stages.LSFU1
```

All Implemented Interfaces:

ProcessListener

< Fields > < Constructors > < Methods >

public class **LSFU1**
extends java.lang.Object
implements ProcessListener

Fields

getNextInstuction

`public static int getNextInstuction`

instruction

`public Instruction instruction`

pc

`public CycleListener pc`

processor

`public Processor processor`

Constructors

LSFU1

`public  LSFU1(Processor processor)`

Constructor for Memory stage initializes PC(instruction Address), result(like a latch which has results of the stage).

Parameters:

processor - a Processor object.

Methods

clearStage

`public void clearStage()`

clearStage method clears the Memory stage.

---

pcValue

`public java.lang.Long pcValue()`

pcValue method returns the pc Value(instruction address) of the Memory stage.

Returns:

long value of the pc Value(instruction address)

---

process

`public void process()`

MemoryStage process method performs the memory operations for LOAD and STORE. fetches data from memory for LOAD and writes data to memory for STORE.

---

toString

`public java.lang.String toString()`

toString method returns the instruction currently in Memory as string if instruction is not null or returns the IDLE constants.

Returns:

String of the instruction or IDLE constants

Overrides:

toString in class java.lang.Object

Stages

Class LSFU2

```
java.lang.Object
     |
     +--Stages.LSFU2
```

All Implemented Interfaces:
ProcessListener

< Fields > < Constructors > < Methods >

public class **LSFU2**
extends java.lang.Object
implements ProcessListener

Fields

instruction

`public Instruction instruction`

pc

`public CycleListener pc`

processor

`public Processor processor`

Constructors

LSFU2

`public  LSFU2(Processor processor)`

Constructor for Memory stage initializes PC(instruction Address), result(like a latch which has results of the stage).

Parameters:

processor - a Processor object.

Methods

clearStage

`public void clearStage()`

clearStage method clears the Memory stage.

---

pcValue

`public java.lang.Long pcValue()`

pcValue method returns the pc Value(instruction address) of the Memory stage.

Returns:

long value of the pc Value(instruction address)

---

process

`public void process()`

MemoryStage process method performs the memory operations for LOAD and STORE. fetches data from memory for LOAD and writes data to memory for STORE.

---

toString

`public java.lang.String toString()`

toString method returns the instruction currently in Memory as string if instruction is not null or returns the IDLE constants.

Returns:

String of the instruction or IDLE constants

Overrides:

toString in class java.lang.Object

Stages

Class MemoryStage

```
java.lang.Object
   |
   +--Stages.MemoryStage
```

All Implemented Interfaces:

ProcessListener

< Fields > < Constructors > < Methods >

public class **MemoryStage**
extends java.lang.Object
implements ProcessListener

Fields

instruction

`public Instruction instruction`

pc

`public CycleListener pc`

processor

`public Processor processor`

Constructors

MemoryStage

`public  MemoryStage(Processor processor)`

Constructor for Memory stage initializes PC(instruction Address), result(like a latch which has results of the stage).

Parameters:

processor - a Processor object.

Methods

clearStage

`public void clearStage()`

clearStage method clears the Memory stage.

pcValue

`public java.lang.Long **pcValue**()`

pcValue method returns the pc Value(instruction address) of the Memory stage.

Returns:

long value of the pc Value(instruction address)

---

process

`public void **process**()`

MemoryStage process method performs the memory operations for LOAD and STORE. fetches data from memory for LOAD and writes data to memory for STORE.

---

toString

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

toString method returns the instruction currently in Memory as string if instruction is not null or returns the IDLE constants.

Returns:

String of the instruction or IDLE constants

Overrides:

toString in class java.lang.Object

Stages

Class MultiplicationFU

```
java.lang.Object
   |
   +--Stages.MultiplicationFU
```

All Implemented Interfaces:

ProcessListener

< Fields > < Constructors > < Methods >

---

public class **MultiplicationFU**
extends java.lang.Object
implements ProcessListener

Fields

instruction

`public Instruction instruction`

---

mulCount

`public int mulCount`

---

pc

`public CycleListener pc`

---

processor

`public Processor processor`

Constructors

MultiplicationFU

`public  MultiplicationFU(Processor processor)`

Constructor for BranchFU stage initializes PC(instruction Address), result(like a latch which has results of the stage).

Parameters:

processor - a Processor object.

Methods

clearStage

`public void clearStage()`

clearStage method clears the BranchFU stage.

---

pcValue

`public java.lang.Long **pcValue**()`

pcValue method returns the pc Value(instruction address) of the BranchFU stage.

Returns:

long value of the pc Value(instruction address)

---

process

`public void **process**()`

BranchFU process method performs relevant control operations such as branching (BZ, BNZ, BAL, JUMP), and Halt Control flow instructions: BZ, BNZ, JUMP, BAL, HALT. Instructions following a BZ, BNZ, JUMP and BAL instruction in the pipeline should be flushed on a taken branch. The zero flag (Z) is set only by arithmetic instructions in ALU.

---

toString

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

toString method returns the instruction currently in BranchFU as string if instruction is not null or returns the IDLE constants.

Returns:

String of the instruction or IDLE constants

Overrides:

toString in class java.lang.Object

Stages

Class ROBCommit

```
java.lang.Object
  |
  +--Stages.ROBCommit
```

All Implemented Interfaces:

ProcessListener

< Fields > < Constructors > < Methods >

---

public class **ROBCommit**
extends java.lang.Object
implements ProcessListener

Fields

instruction

`public Instruction instruction`

---

pc

`public CycleListener pc`

---

processor

`public Processor processor`

Constructors

ROBCommit

`public  ROBCommit(Processor processor)`

Methods

clearStage

`public void clearStage()`

clearStage method clears the WriteBack stage.

---

pcValue

`public java.lang.Long pcValue()`

pcValue method returns the pc Value(instruction address) of the WriteBack stage.

Returns:

long value of the pc Value(instruction address)

---

process

```
public void process()
```

---

toString

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

toString method returns the instruction currently in WriteBack as string if instruction is not null or returns the IDLE constants.

Returns:

String of the instruction or IDLE constants

Overrides:

toString in class java.lang.Object

Stages

Class WriteBack

```
java.lang.Object
    |
    +--Stages.WriteBack
```

All Implemented Interfaces:

ProcessListener

< Fields > < Constructors > < Methods >

---

public class **WriteBack**
extends java.lang.Object
implements ProcessListener

Fields

instruction

```
public Instruction instruction
```

---

instructionList

```
public java.util.List instructionList
```

---

pc

```
public CycleListener pc
```

---

processor

```
public Processor processor
```

Constructors

WriteBack

```
public  WriteBack(Processor processor)
```

Constructor for Write Back stage initializes PC(instruction Address), result(like a latch which has results of the stage).

Parameters:

processor - a Processor object.

Methods

clearStage

```
public void clearStage()
```

clearStage method clears the WriteBack stage.

---

pcValue

```
public java.lang.Long pcValue()
```

pcValue method returns the pc Value(instruction address) of the WriteBack stage.

Returns:

long value of the pc Value(instruction address)

---

process

```
public void process()
```

WriteBack process method performs the register write operations. The registers are written when the instruction enters the write back stage in same cycle. Aborts the simulation when HALT instruction is encountered.

toString

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

toString method returns the instruction currently in WriteBack as string if instruction is not null or returns the IDLE constants.

Returns:

String of the instruction or IDLE constants

Overrides:

toString in class java.lang.Object

Package Utility

Class Summary

Constants

Constants.OpCode

OpCode enum contains operation codes of different instructions.

Constants.Stage

Stage enum contains Stage constants of different instructions.

FileProcessor

Instruction

PhysicalRegister

RAT

Utility

Class Constants

```
java.lang.Object
  |
  +--Utility.Constants
```

< Fields > < Constructors >

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

Fields

CACHE_SIZE

```
public static final int CACHE_SIZE
```

DISPLAY

```
public static final java.lang.String DISPLAY
```

INITIALIZE

`public static final java.lang.String` **`INITIALIZE`**

IQ_COUNT

`public static final int` **`IQ_COUNT`**

LITERAL_PREFIX

`public static final java.lang.String` **`LITERAL_PREFIX`**

MEM_SIZE

`public static final int` **`MEM_SIZE`**

RAT_COUNT

`public static final int` **`RAT_COUNT`**

REG_COUNT

`public static final int` **`REG_COUNT`**

REG_PREFIX

`public static final java.lang.String` **`REG_PREFIX`**

ROB_COUNT

`public static final int` **`ROB_COUNT`**

SEPARATOR1

`public static final java.lang.String` **`SEPARATOR1`**

SEPARATOR2

```
public static final java.lang.String SEPARATOR2
```

SIMULATE

```
public static final java.lang.String SIMULATE
```

START_ADDRESS

```
public static final long START_ADDRESS
```

Constructors

Constants

```
public  Constants()
```

Utility

Class Constants.OpCode

```
java.lang.Object
    |
    +--java.lang.Enum
        |
        +--Utility.Constants.OpCode
```

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

< Fields > < Methods >

public static final class **Constants.OpCode**
extends java.lang.Enum

OpCode enum contains operation codes of different instructions.

Fields

ADD

```
public static final Constants.OpCode ADD
```

AND

```
public static final Constants.OpCode AND
```

BAL

```
public static final Constants.OpCode BAL
```

BNZ

```
public static final Constants.OpCode BNZ
```

BZ

```
public static final Constants.OpCode BZ
```

EXOR

```
public static final Constants.OpCode EXOR
```

HALT

```
public static final Constants.OpCode HALT
```

IDLE

```
public static final Constants.OpCode IDLE
```

JUMP

```
public static final Constants.OpCode JUMP
```

LOAD

```
public static final Constants.OpCode LOAD
```

MOV

```
public static final Constants.OpCode MOV
```

MOVC

```
public static final Constants.OpCode MOVC
```

MUL

```
public static final Constants.OpCode MUL
```

OR

```
public static final Constants.OpCode OR
```

STORE

```
public static final Constants.OpCode STORE
```

SUB

```
public static final Constants.OpCode SUB
```

Methods

valueOf

```
public static Constants.OpCode valueOf(java.lang.String name)
```

values

```
public static Utility.Constants.OpCode[] values()
```

Utility

Class Constants.Stage

```
java.lang.Object
    |
    +--java.lang.Enum
        |
        +--Utility.Constants.Stage
```

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

< Fields > < Methods >

public static final class **Constants.Stage**
extends java.lang.Enum

Stage enum contains Stage constants of different instructions.

Fields

ALU1

`public static final Constants.Stage` **`ALU1`**

ALU2

`public static final Constants.Stage` **`ALU2`**

BRANCHFU

`public static final Constants.Stage` **`BRANCHFU`**

DECODE

`public static final Constants.Stage` **`DECODE`**

DISPATCH

`public static final Constants.Stage` **`DISPATCH`**

EMPTY

```
public static final Constants.Stage EMPTY
```

FETCH

```
public static final Constants.Stage FETCH
```

LSFU1

```
public static final Constants.Stage LSFU1
```

LSFU2

```
public static final Constants.Stage LSFU2
```

MULTIPLICATIONFU

```
public static final Constants.Stage MULTIPLICATIONFU
```

ROBCOMMIT

```
public static final Constants.Stage ROBCOMMIT
```

WRITEBACK

```
public static final Constants.Stage WRITEBACK
```

Methods

valueOf

```
public static Constants.Stage valueOf(java.lang.String name)
```

---

values

```
public static Utility.Constants.Stage[] values()
```

Utility

Class FileProcessor

```
java.lang.Object
    |
    +--Utility.FileProcessor
```

< Constructors > < Methods >

---

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

Constructors

FileProcessor

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

Constructor for FileProcessor initializes file object with relevant instruction file.

Parameters:

fileName - a instruction file set in string format.

Methods

fetchInstructions

```
public java.util.List fetchInstructions()
```

fetchInstructions method process the file object and stores the instructions in array format in a instruction array list.

Utility

Class Instruction

```
java.lang.Object
    |
    +--Utility.Instruction
```

< Fields > < Constructors > < Methods >

---

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

Fields

archdest

```
public java.lang.Long archdest
```

---

archsrc1Add

```
public java.lang.Long archsrc1Add
```

---

archsrc2Add

```
public java.lang.Long archsrc2Add
```

---

brnTrgAdd

```
public long brnTrgAdd
```

---

dest

```
public java.lang.Long dest
```

---

destVal

```
public java.lang.Long destVal
```

---

inExecution

```
public boolean inExecution
```

---

insPc

```
public long insPc
```

---

isBrnTaken

```
public boolean isBrnTaken
```

isLiteral

`public boolean **isLiteral**`

isROBCommit

`public boolean isROBCommit`

literal

`public java.lang.Long literal`

opCode

`public Constants.OpCode opCode`

src1

`public java.lang.Long src1`

src1Add

`public java.lang.Long src1Add`

src1FwdValIn

`public Constants.Stage src1FwdValIn`

src1Stall

`public boolean src1Stall`

src2

`public java.lang.Long src2`

src2Add

`public java.lang.Long **src2Add**`

---

src2FwdValIn

`public Constants.Stage **src2FwdValIn**`

---

src2Stall

`public boolean **src2Stall**`

---

stallIn

`public Constants.Stage **stallIn**`

Constructors

Instruction

`public  **Instruction**()`

Methods

getInstructionOpcode

`public static Constants.OpCode **getInstructionOpcode**(java.lang.String strOp)`

getInstructionOpcode which gets the opcode of an instruction and stores to result.

Parameters:

strOp - string opcode

---

toString

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

toString method returns the instruction format to the given opcode.

Returns:

String of the instruction or IDLE constants

Overrides:

toString in class java.lang.Object

Utility

Class PhysicalRegister

```
java.lang.Object
    |
    +--Utility.PhysicalRegister
```

< Constructors > < Methods >

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

Constructors

PhysicalRegister

```
public  PhysicalRegister()
```

Constructor for Register initializes the registers.

Methods

getAvailability

```
public boolean getAvailability()
```

getIsValid

```
public boolean getIsValid()
```

getRegValue

```
public long getRegValue()
```

getZFlag

```
public int getZFlag()
```

setAvailability

```
public void setAvailability(boolean availabilityVal)
```

setIsValid

```
public void setIsValid(boolean statusVal)
```

setRegValue

```
public void setRegValue(long reg_V)
```

setZFlag

```
public void setZFlag(int zFlagVal)
```

Utility

Class RAT

```
java.lang.Object
    |
    +--Utility.RAT
```

< Constructors > < Methods >

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

Constructors

RAT

```
public  RAT()
```

Constructor for RAT initializes the RAT.

Methods

getRATPhyReg

```
public long getRATPhyReg()
```

getRATStatus

```
public boolean getRATStatus()
```

setRATPhyReg

```
public void setRATPhyReg(long phy_R)
```

setRATStatus

```
public void setRATStatus(boolean statusVal)
```