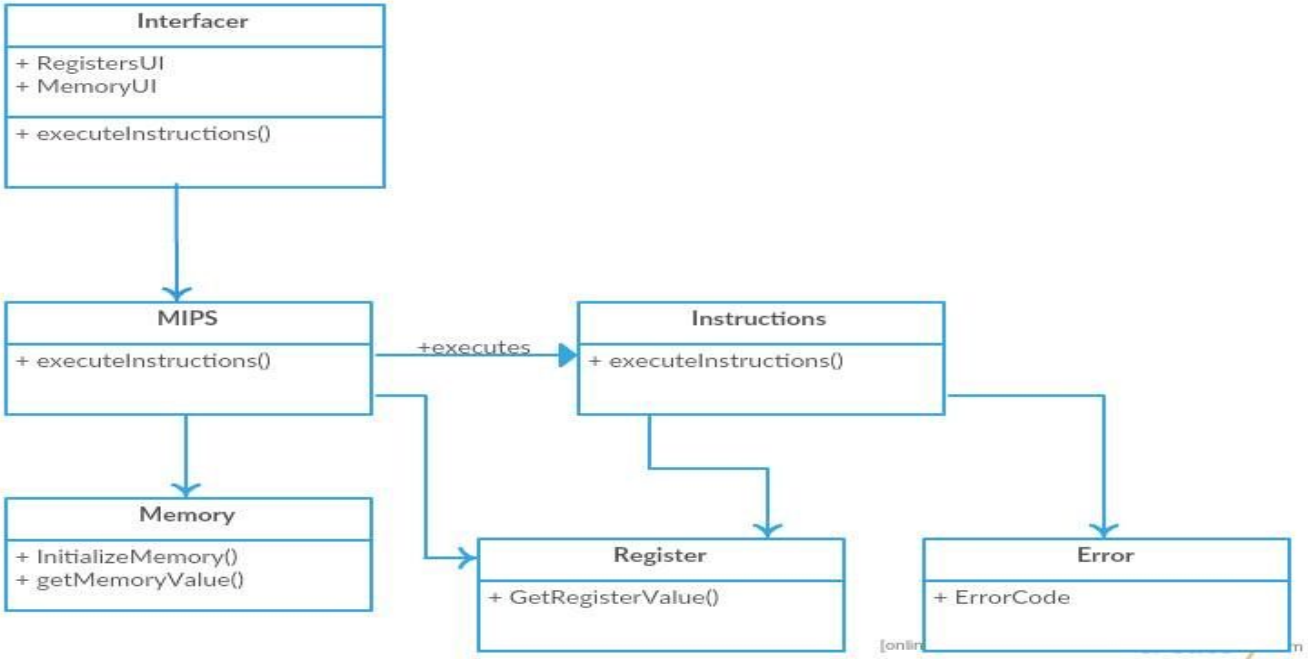
Software Engineering Project 1 MIPS Assembly Language Programming 2

Team 5
Ajitesh Gupta
Aabhas Majumdar
Abishek Kannan

Introduction

- MIPS stands for Microprocessor without Interlocked Pipeline Stages
- It is a reduced instruction set architecture (RISC) instruction set architecture (ISA)
- We are given a MIPS simulator which is written in Java
- Our project aims to convert the entire code base to Javascript and alter the internal structure without altering the external behaviour

Current Code Structure



Some Code Smells

Register.java

- Hard coding of values
  A lot of hexadecimals numbers are used like updateRegister("$sp", 0x10100000); and updatePcAbsolute(0x4000000);
- Very Long conditional blocks
  33 if else conditions
- Meaningless comments
  Like // error (-1 can be valid return value also)
- Indecent exposure
  Reg array is public

Some Code Smells

Error.java

- Indecent exposure
  Variables NO_ERROR, ASSEMBLE_TIME_ERROR are public
- No comments.

Code Smells (Contd.)

Memory.java

- Hard coding of addresses
- Lack of error handling
  Few else blocks are empty
- Indecent exposure
  All its data members are public
- Dead code
  Empty else blocks

Code Smells (Contd.)

Instruction.java

- Conditional complexity is very high
  132 line if-else block
- Indecent exposure
  variables startAddressOfInst, addressOfLevels are public
- Data Clamp
  10 data members
- Code duplication (180 duplicate lines)

Code Smells (Contd.)

MyLibrary.java

- Functional class (No data, only functions)
- Indecent exposure (The class and all functions are public)

Interface.java

- Heavy code duplication (156 duplicate lines)
- Hard coding of values ([`javax.swing.GroupLayout.DEFAULT_SIZE, 362`], [`javax.swing.GroupLayout.PREFERRED_SIZE, 124`])

Code Smells (Contd.)

MIPSProgram.java

- Blob
  10 data members and 25 functions
- Unnecessary Comments (Debugging code still present as comments)
  like //String fileName="/home/rishi/study/RAShip/MyTestCases/recursion.asm";
- Large Class
- Low Cohesion and High Coupling
  Data, Comment and Text section kept together

More Problems

- Absence of a readme file
- No documentation for the project
- Code Standard not Followed (Oracle Code Conventions)
  - Beginning Comments
  - Line Length (longer than 80 characters)
  - Wrapping Lines
  - Return Statements should not use parentheses. E.g return (ret << 16 >> 16).
  - Blank Lines and Spaces missing to make the code more readable. E.g currentDynamicDataAddress=startAddressOfDynamicData (space between =)

Proposed Code Structure