Faculty of Computers and Artificial Intelligence

Cairo University

OOP

Dr. Mohamed EL_Ramly

Assignment 1 Task 4,5

Name:	IDS:	Emails:
Marwan Tamer Sayed Ali	20230380	marwantamer004@gmail.com
Mohamed Refaat Mohamed Awd	20230337	mrefaat8853@gmail.com
Abdulrahman Ahmed Kamal Kamel	20230202	bedoa973@gmail.com

Name:	Task 4	Task 5
Marwan Tamer Sayed Ali	We worked together on the code.	-
Mohamed Refaat Mohamed Awd	We worked together on the code.	-
Abdulrahman Ahmed Kamal Kamel	We worked together on the code.	-

Vole Machine Simulator

This document provides a detailed design and testing report for the Vole Machine Simulator project, The simulator interprets and executes instructions on a virtual machine with simulated registers and

memory.

Design Section

Class Overview

The main class in this project is VoleMachine, which handles all operations for simulating the Vole machine. The main

attributes include:

- registers: array of 16-bit registers R0 to R15
- memory: 16x16 matrix simulating memory
- programCounter: points to the current instruction
- instructionRegister: holds the current instruction being executed

Attributes and Methods

Attributes:

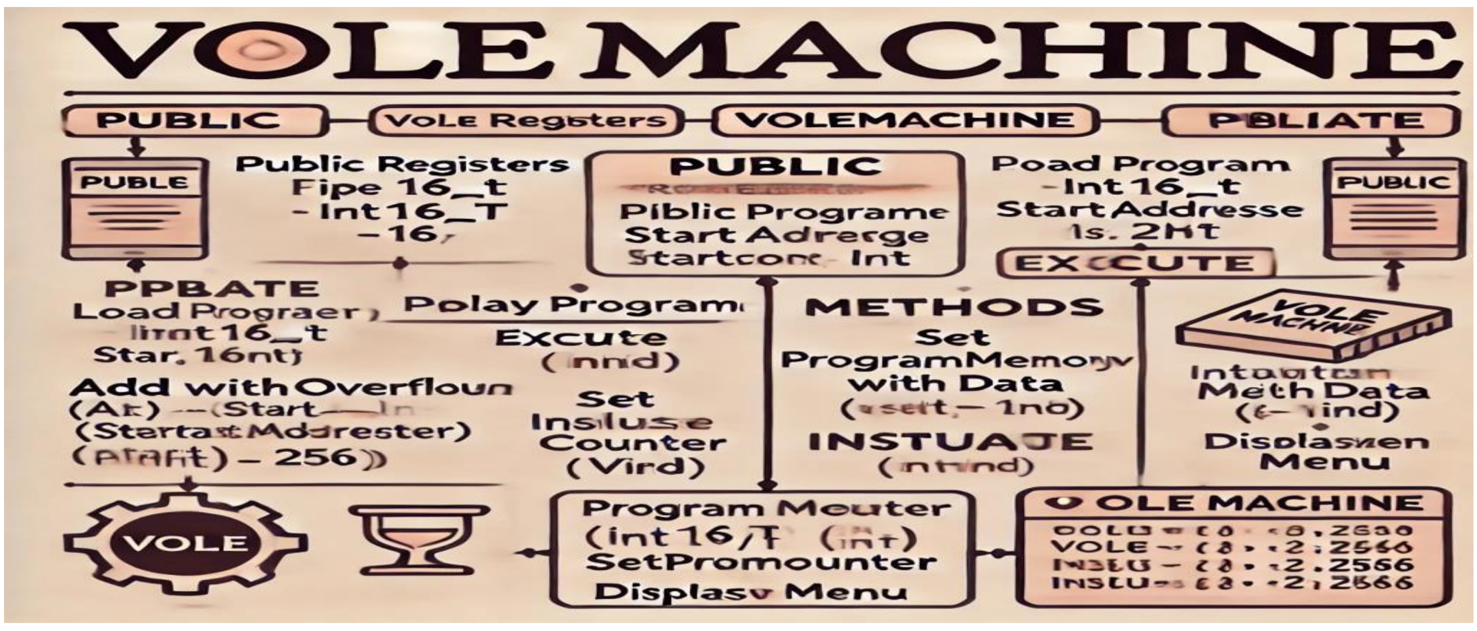
- registers: Array<int16_t, 16> - Represents 16 registers.
- memory: Array<int16_t, 256> - Simulates 256 memory cells.
- programCounter: int - Tracks the current instruction location.

Methods:

- loadProgram(filename, address): Loads a file into memory starting from a given address.
- execute(): Executes instructions one by one based on programCounter.
- displayStatus(): Shows registers, memory, and program counter status.

UML Overview

A simplified UML diagram representing the VoleMachine class and its primary interactions is included in this design. The class contains attributes for memory, registers, and program control, as well as methods for loading and executing instructions.



Opcode Logic

Each opcode is mapped to a specific operation. For instance, LOAD (opcode 1) transfers a memory value to a register, while STORE (opcode 3) places register data into memory. Other opcodes support arithmetic, logical, and control flow instructions. This logic enables effective simulation of program flow and arithmetic operations.

Testing Section

Test Plan

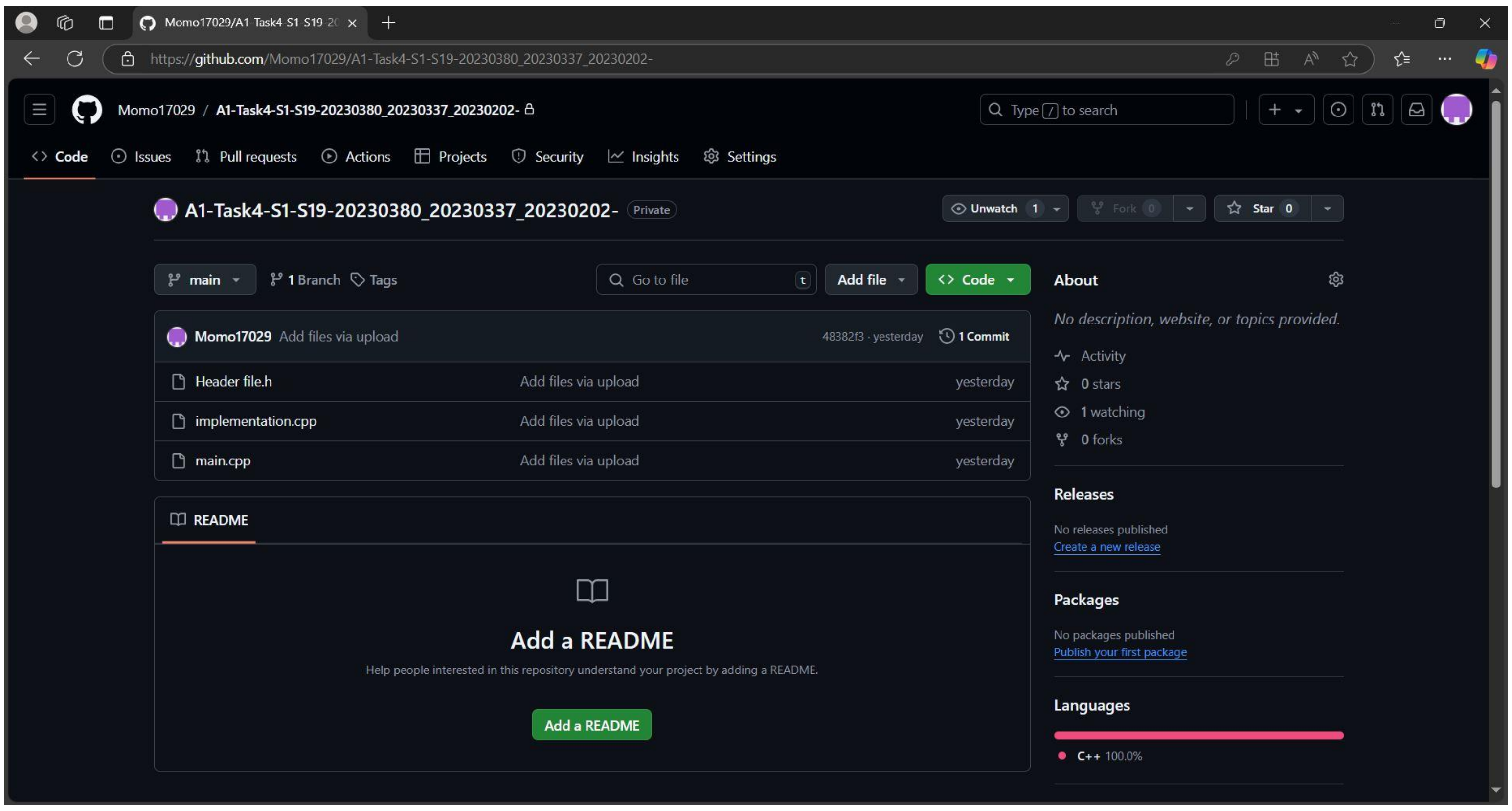
The testing plan includes loading, executing, and verifying memory, register, and program counter statuses. Specific cases are tested for each opcode to ensure accurate results.

Sample Test Cases

- Test Load Program: Verifies program loading into memory from a file.

- Test Execution: Confirms correct operation of each opcode.

- Test Invalid Opcode Handling: Tests that invalid instructions halt execution appropriately.


Momo17029/A1-Task4-S1-S19-2(
https://github.com/Momo17029/A1-Task4-S1-S19-20230380_20230337_20230202-
Momo17029 / A1-Task4-S1-S19-20230380_20230337_20230202-
Type / to search
Code
Issues
Pull requests
Actions
Projects
Security
Insights
Settings
A1-Task4-S1-S19-20230380_20230337_20230202-
Private
Unwatch 1
Fork 0
Star 0
main
1 Branch
Tags
Go to file
Add file
Code
About
No description, website, or topics provided.
Activity
0 stars
1 watching
0 forks
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package
Languages
C++ 100.0%
README
Add a README
Help people interested in this repository understand your project by adding a README.
Add a README

Momo17029 Add files via upload		48382f3 · yesterday · 1 Commit
Header file.h	Add files via upload	yesterday
implementation.cpp	Add files via upload	yesterday
main.cpp	Add files via upload	yesterday