Final Project Report

Simulator for M-Architecture

COE 403 – Semester 182

Bader Almazmumi
Mohammed Alhartomi
Ibrahim Hijan

For:

Dr. Muhamed F. Mudawar

Table of Contents

Introduction:

M-Simulator is a software developed to simulate the functionally of M-Architecture's instructions. The simulator covers all immediate type operations with exception to long immediate, registers type operation, jump type operation and branch type operations. Also, it includes the data segmentation and system calls. M-Simulator is designed to be reliable and convenient for users. The report covers the basic MSimulator Components, implemented instructions and the MSimulator features.

Objective:

Our aim is to do an interactive development environment (IDE) for an M-Architecture which is developed by Dr. Muhamed Mudawar.

The Simulator Components:

To do the simulator, we divide it mainly to 5 components which are Assembler, Instruction(IInstruction, BInstruction, JInstruction, RInstruction), ProgramCounter, RegisterFile and DataMemory.

The assembler's function is to take an array of string (an assembly code) in the text segment and the data segment and convert each string in the text segment to the corresponding instruction object format and add it to the InstructionList in the ProgramCounter.

Each instruction object type has its own execute method which executes different type of functions depending on the format of the instruction. This simulator can differentiate between different formats with the same function name (ADD R5 = R5, 5 and ADD R5 = R5, R5) and also it can understand the registers alias names. (t0,t1,s2, ….)

In case of there is an error in one of the instructions, the assembler will output a message with the instruction's line number to the I/O text pane.

After the code is assembled, the InstructionList will be filled with all instructions taken from the user, each instruction will have the index, the binary representation of it, the opcode, and all the parameters needed (a, b, offset, imm, …).

In this stage, the instructions are ready to be executed. So, we have two ways to execute them, either one by one (trace button) or execute them all in the same time with different speed frequency.

Each function executed will either get value of specific register from RegisterFile, set value of a specific register, modify the data memory in case of the load or store instructions, or change the program counter in the case of jump and branch instructions.

Instructions Implemented:

We implemented almost all the instructions in the M-Architecture, Alu, memory, branch and jump, loop as well as the first 9 system call Instructions. We didn't complete the Floating-Point Instructions although it's not hard to implement in our simulator. Also, we implement .data segment in order to initialize the memory with predefined bytes and add @lable to use it with all immediate instruction. We implemented .space and .byte.

Here are some shortcuts to test multiple functions, just type the shortcut and press enter. [Clear the text code area and enter the shortcut fallowed by space]

.scloop to test the loop function of the simulator

.scbranch to test the branch function

. scjumpandalu to test the different instruction of the ALU and jump

. scalu to test different functions of ALU

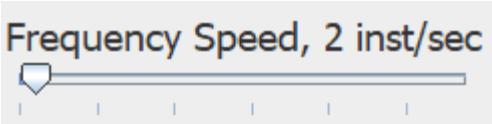
.scsort to test the bubble sort functions

The Simulation Features:

After we conclude this project, we have added in addition of designing the GUI, some features that will increase the value and the usability of the simulator. In this section we will mentions most of the features.

The Speed Frequency:

The user can change the value of the execution speed of the instructions by sliding the frequency slider in the bottom of the screen (instruction/sec).



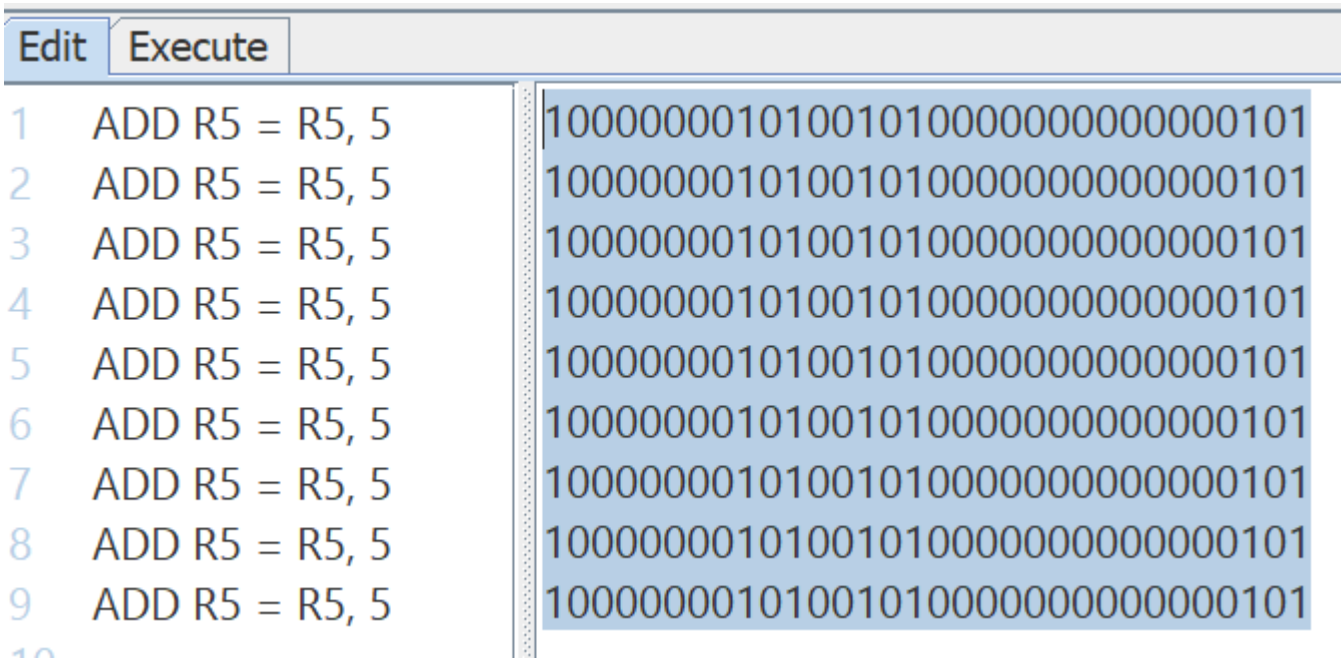
Change the Representation of Single/All Register Value:

The user can change the value representation of each single register to hex, decimal or binary representation. Also, you can change all the values to one representation by clicking on Edit -> Register Value -> chose one of them.

R Name	Register Number
R0	0
R1	55
R2	0x0000000000000021
R3	101100
R4	0
R5	0
R6	0
R7	0
R8	0
R9	0
R10(t0)	0
R11(t1)	0
R12(t2)	0

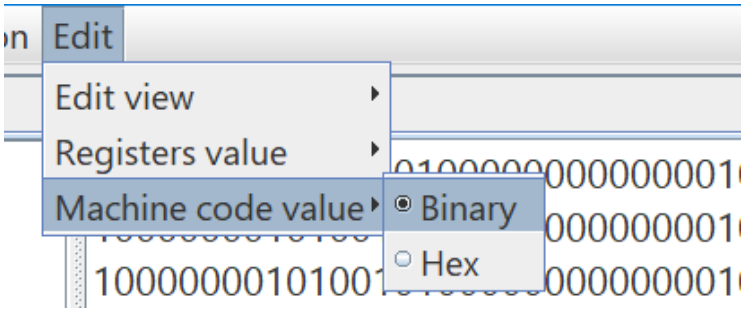
Extract the Machine Code Out from the Assembler In Different Representations:

The user can get the instructions in binary, or hex, format by going to edit tab and slide the split to left after the program is assembled.



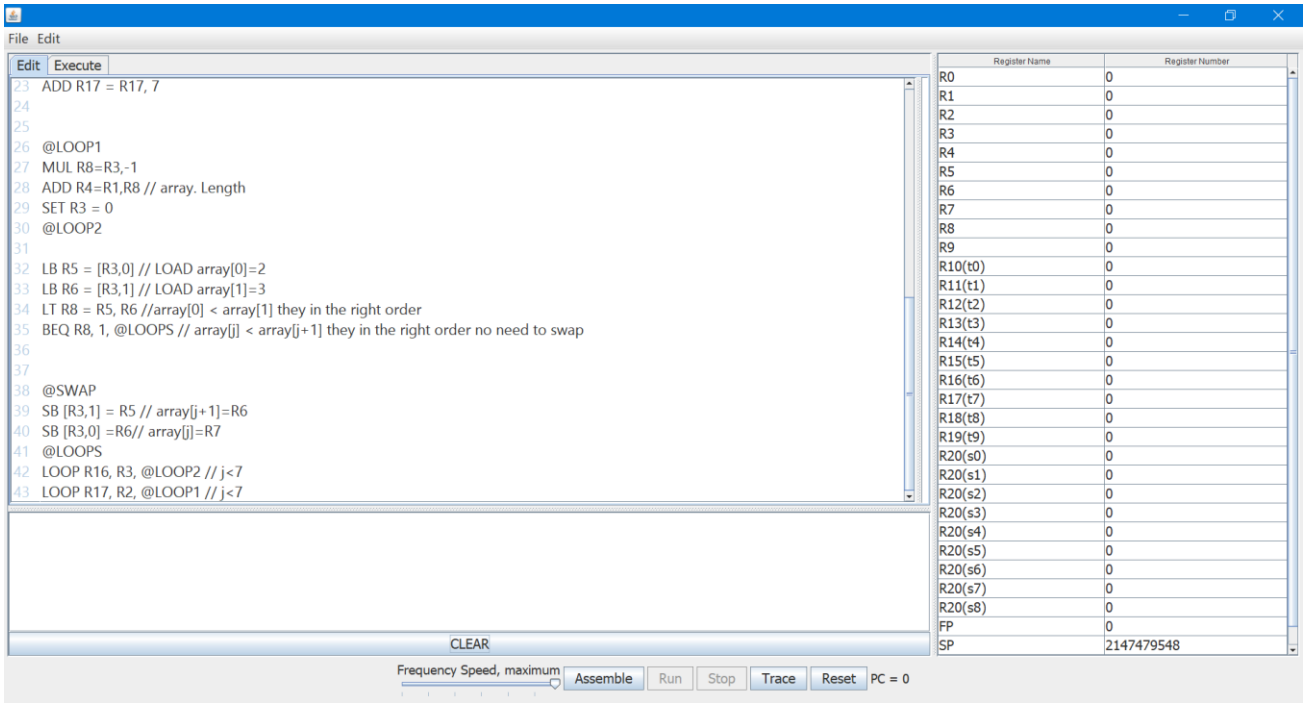
The format can be changed by going to Edit - > Machine Code Value.

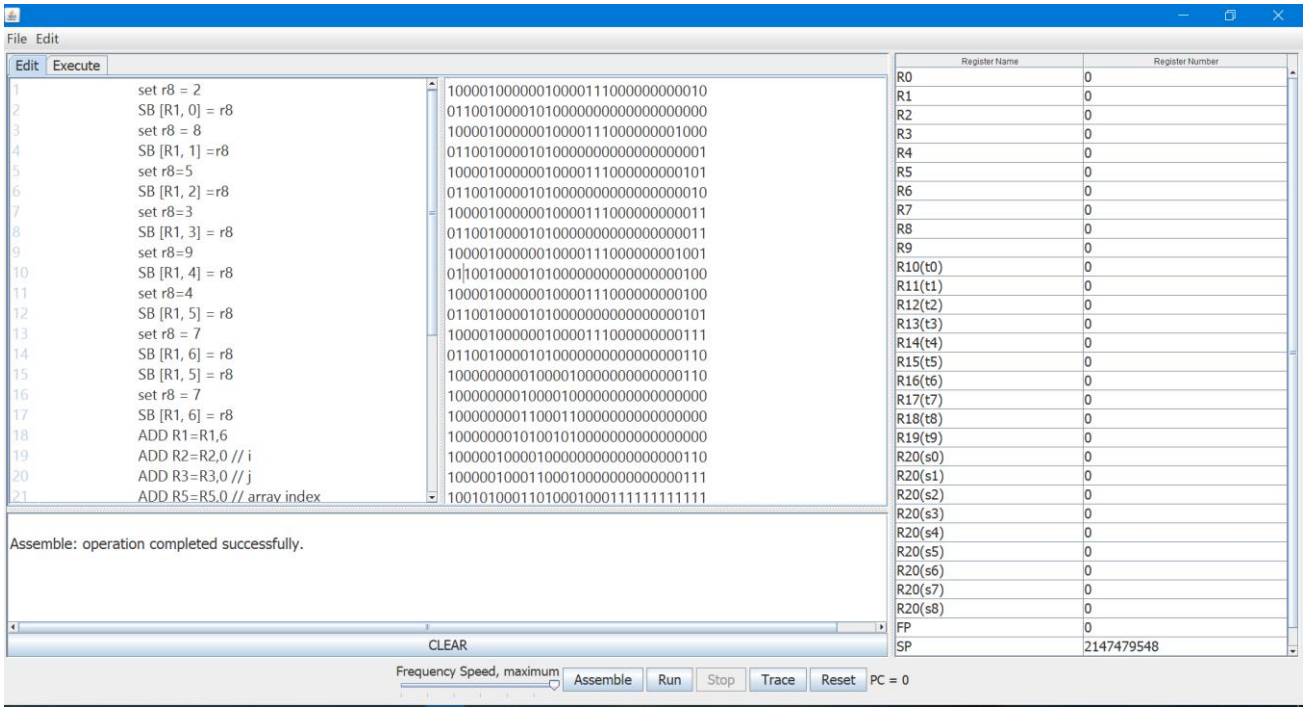
Also, you can hide this window by clicking on Edit -> Edit View -> Uncheck the machine code view.

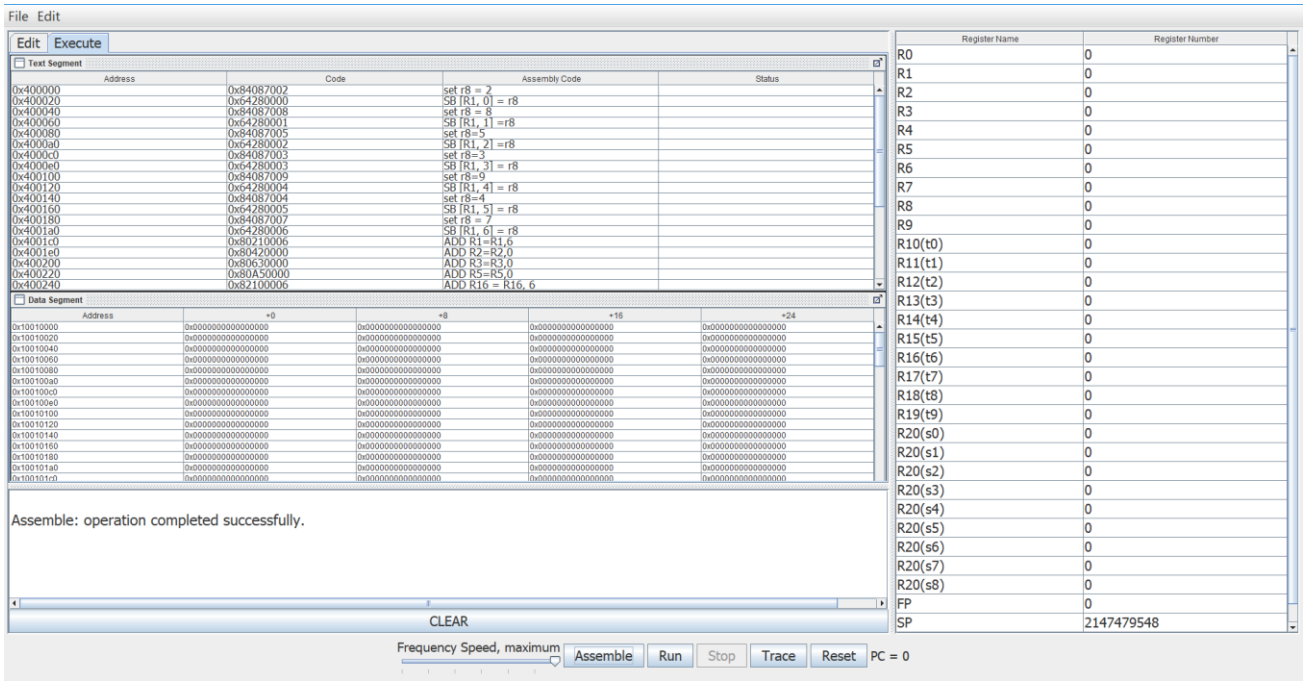


Screenshots:

Those are some screenshots of current state of MSimulator.







Conclusion:

Everything considered, our project has many functionalities, such as assembling and compiling an assembly code written in M-Architecture. As well as displaying the results in simple graphical interface with some features that make it more powerful.