**CSCI 6461 Fall Semester 2021**

**Project 1 (Team 5)**

**Computer Architecture Simulator**

**Lakshmi Kesava Reddy Indla**

**Indu Goel**

**Humaira Riaz**

**Chengzheng Ye**

**Basic Machine**

**Requirement analysis:**

Build a project to simulate the computer system. As in project 1 we need to design and implement the basic machine. This machine is going to perform several functions. These are load/store data to/from the memory.

**Goal:**

Goals of this project are as follows:

Load the data from the address in the memory

LDR load register from memory

SI will do the single step instruction.

**Basic Machine**

The purpose of this project is to:

* Design and implement the basic machine architecture.
* Implement a simple memory
* Execute Load and Store instructions
* Implement the Halt Instruction
* Build simulator front panel

**Design Notes:**

**Running the JAR File:**

The JAR file can be run in three ways:

1. For Windows double click the downloaded JAR file. You may have to allow access for this to execute  properly.
2. For Mac go to terminal and type.

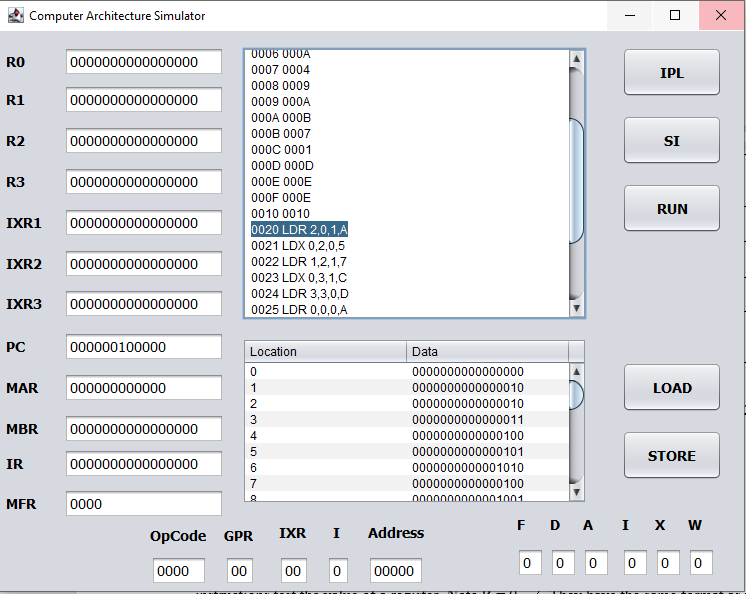
cd desktop

java -jar CASimulator.jar

1. For Linux, right-click the file, go to “Properties” then “Permissions”, and check the box “Allow executing file as program”.

Please note that the files were compiled with Java 16, so the JAR file may not execute properly if  using an older version of Java.

**GUI (Graphical User Interface):**



**Basic Buttons and their Functionality:**

1. IPL button loads program into memory .
2. SI button for the single instruction.
3. Run button starts executing instructions in memory
4. Halt button - used to manually halt the machine.(will add it later)
5. Load button will load the instruction.
6. Store is used to manually store a value in the memory.

**How to Operate:**

1. As computer architecture simulator starts you can see that nothing has been uploaded to memory.
2. When you press the IPL button. It will browse for the file. You have to select the ipl.txt file from the location.
3. After selecting the ipl.txt file the file is loaded and the PC changes to binary value 20.
4. Now you have two button options i.e. SI and Run.
5. When you click SI (the single instruction) button single instruction will get executed and the PC will increment by 1.
6. Pressing Run button will display no more instructions as all the instructions are executed.
7. In our implementation where there is no indirect addressing, we have explicitly mentioned 0, as four arguments are must otherwise it gives an error. So for LDX and STX need to mention 0 as r since we need four parameters. As can be seen in the table below.

|  |  |  |
| --- | --- | --- |
| **Op Code** | **Instruction** | **GPR,IXR,I,Address** |
| 0020 | LDR | 2,0,1,A |
| 0021 | LDX | 0,2,0,5 |
| 0022 | LDR | 1,2,1,7 |
| 0023 | LDX | 0,3,1,C |
| 0024 | LDR | 3,3,0,D |
| 0025 | LDR | 0,0,0,A |
| 0026 | STR | 2,0,1,C |
| 0027 | STX | 0,2,0,2 |
| 0028 | STR | 1,2,1,3 |
| 0029 | STX | 0,3,1,D |
| 002A | STR | 3,3,0,1 |
| 002B | STR | 0,0,0,8 |

5. We are calculating effective address using three parameters i.e. IX, I and address. Based on these three parameters effective address (EA) will be calculated. After calculating respective instruction is performed.

6. If values are loaded into the registers, then the decision would be to either LOAD or STORE the instruction. Make sure to specify which registers you want to use in the  instruction since they hold values, then you can RUN or SI through the  simulator.

7. If values ARE NOT loaded, then you can just press the RUN button to iterate through all the memory or press SI to step through memory one PC at a time.

8. The console will print the PC, value, and memory on each iteration.

9. Important to note that in our implementation if you use the IPL button second time memory gets reset. Start it fresh.

**Deliverables:**

|  |  |
| --- | --- |
| **Project Requirement** | **What we implement** |
| **You should be able to enter data into any of R0 – R3** | (Disabled) |
| **enter data into memory via switches:** | We are entering data in to memory via IPL document. |
| **enter the various Load and Store instructions into memory:** | We have taken care of Via IPL document. |
| **enter address into PC** | We have hard coded it to memory location 20 (in Hex) |
| **Press Single Step switch to execute the instruction at that address:** | Taken care of. |