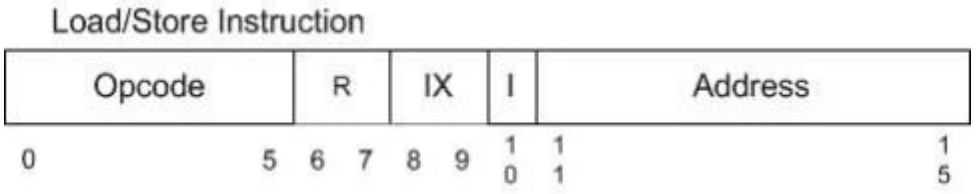
Team 9: Design Notes

- ✓ Sree Vital Chalamalasetty
- ✓ Lijadis Layew
- ✓ Shejal Shankar
- ✓ Aditi Vyas

The project's goal is to create an assembly language-based simulator of a simple traditional CISC computer. The following features will be incorporated in the initial phase of the project:

- 4 General Purpose Registers (GPRs) – each 16 bits in length **[R0, R1, R2, R3]**
- 3 Index Registers – 16 bits in length **[X1, X2, X3]**
- 16-bit words
- Memory of 2048 words, expandable to 4096 words
- Word addressable

Load/Store Instruction:



Opcode : 6 bits – Specifies one of 64 possible instructions; Phase 1 will have 5 Load/Store Opcodes.

R : 2 bits – R0(00), R1(01), R2(10), R3(11) General Purpose Registers.

IX : 2 bits – X1(01), X2(10), X3(11) Indexed Registers.

I : 1 bit – I=0 specifies indirect addressing, or otherwise no indirect addressing. Address : 5 bits – Specifies one of 32 locations.

The effective address for executing various Load/Store instructions will be determined as follows:

Effective Address (EA) = I

= 0:

IX = 00: content (address field)

IX = 01 or 10 or 11: content(IX) + content of address field I

= 1:

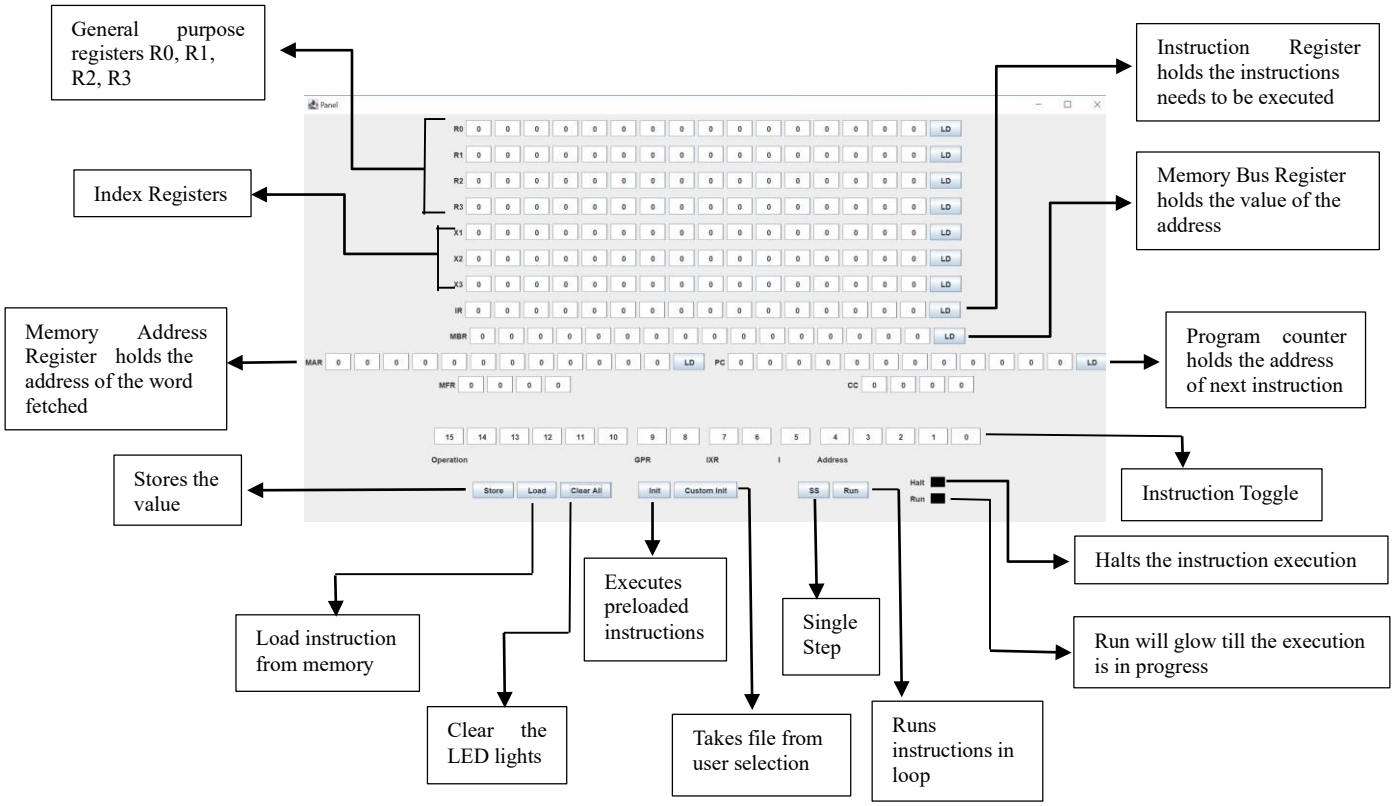
IX = 00: content (content (address field))

IX = 01 or 10 or 11: content(content(IX) + content of address field)

The Simulator's Load/Store instructions are as follows (I is optional):

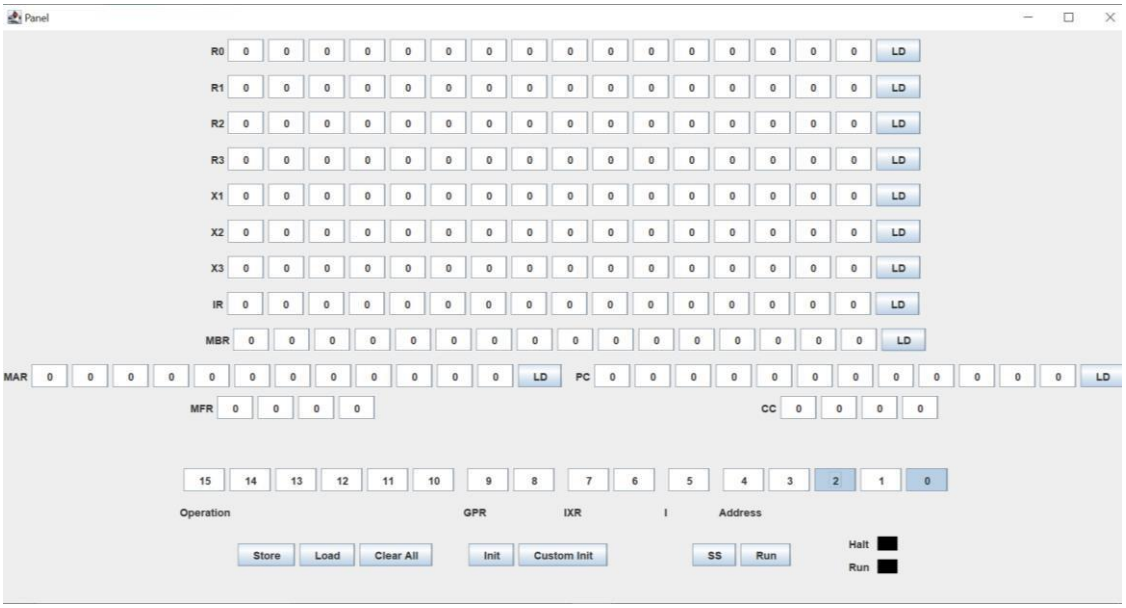
OpCode	Instruction	Description
01	LDR r x I address	Load Register From Memory, r = 0..3 r <- c(EA)
02	STR r x I address	Store Register To Memory, r = 0..3 Memory(EA) <- c(r)
03	LDA r x I address	Load Register with Address, r = 0..3 r <- EA
41	LDX x I address	Load Index Register from Memory, x = 1..3 Xx <- c(EA)
42	STX x I address	Store Index Register to Memory. X = 1..3 Memory(EA) <- c(Xx)

Simulator Design:

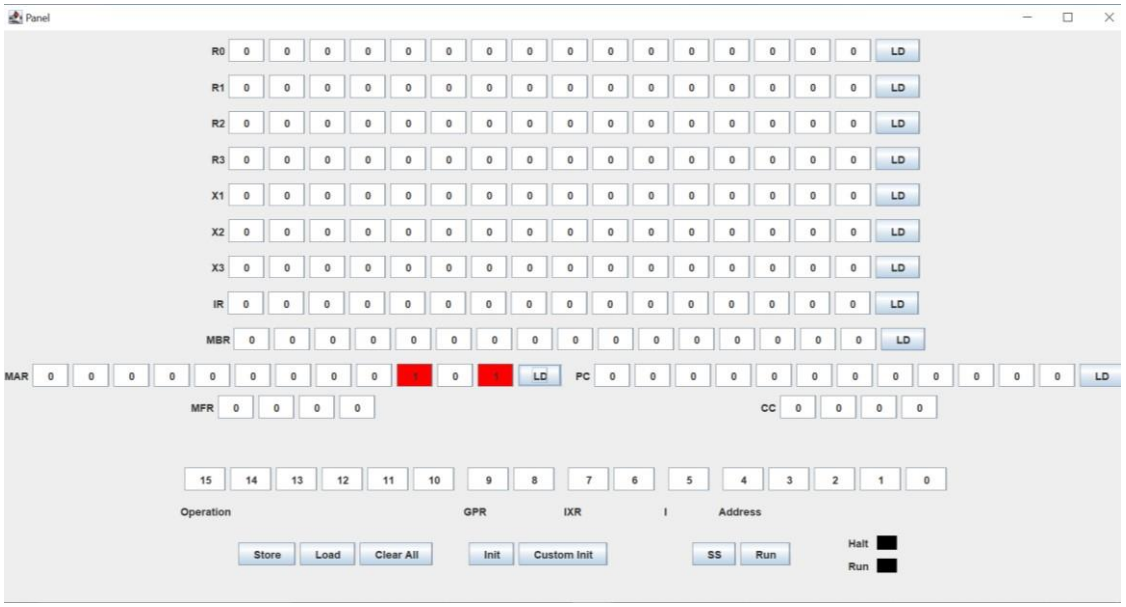


How to add instruction to memory:

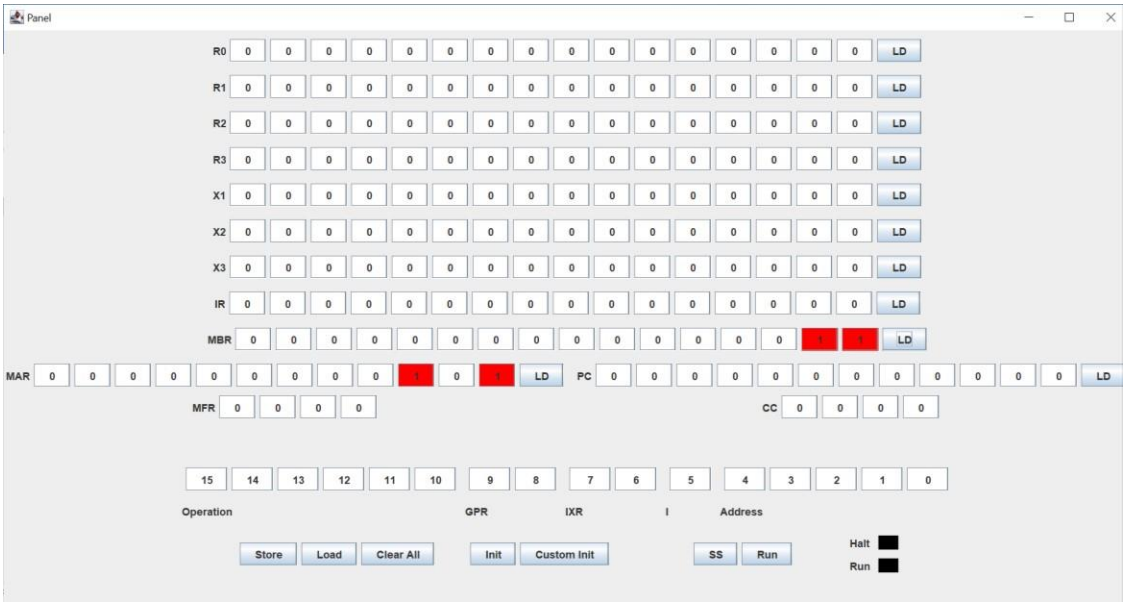
1. Input the MAR instruction into the Instruction Toggle (White LED is 0; Blue LED is 1), as illustrated below:



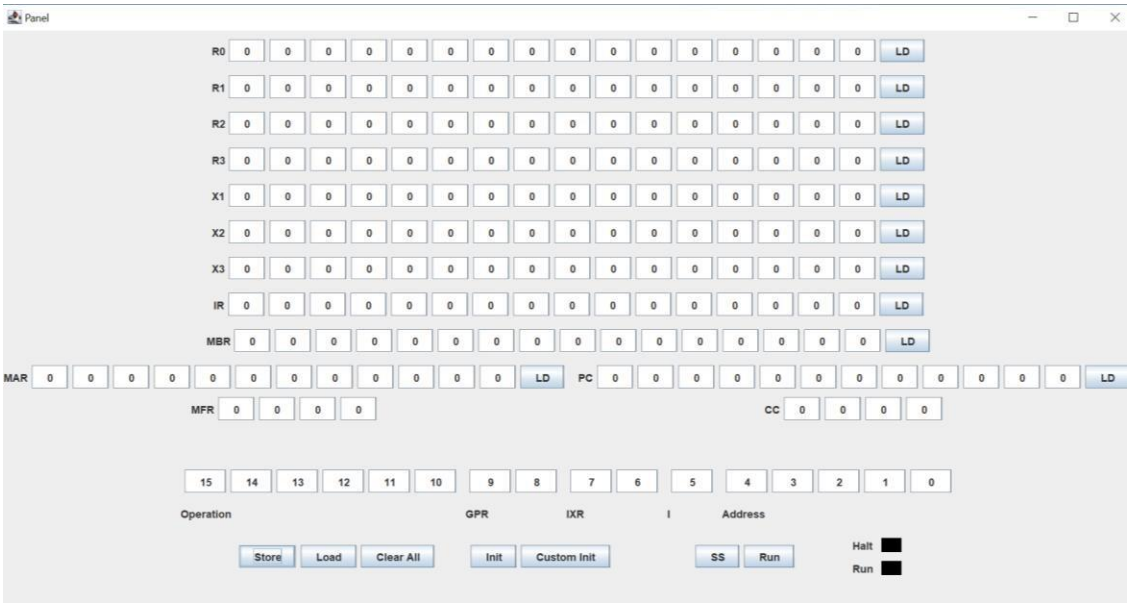
2. To load it into the MAR register, click the "LD" button next to MAR (the red LED represents 1 and the white LED represents 0):



3. Rep the previous steps for MBR as well:

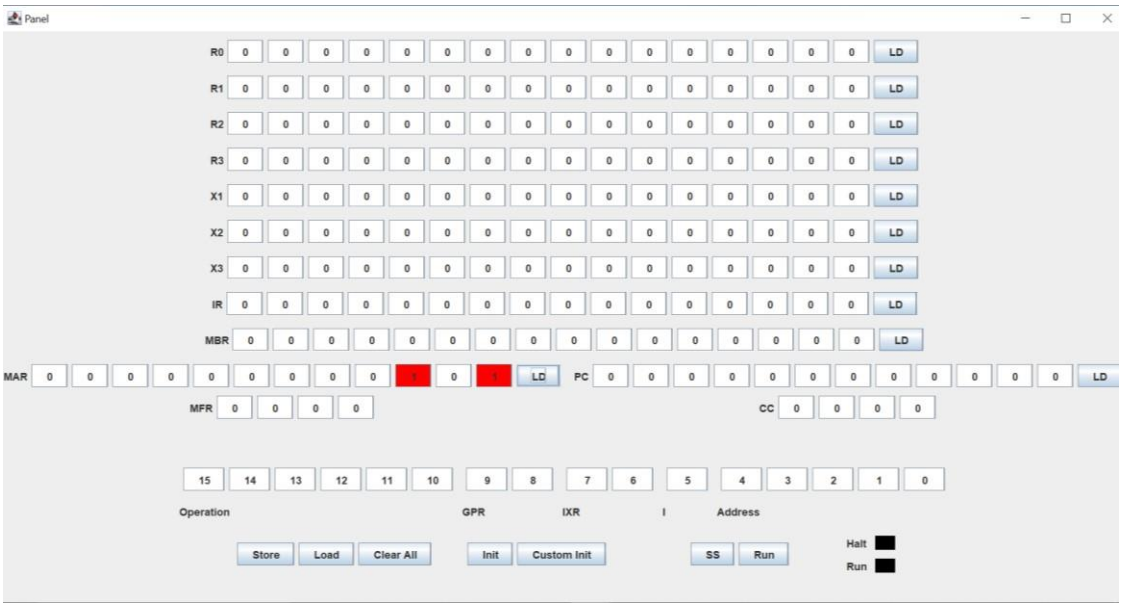


4. Click the "Store" button to save the MBR instruction to the MAR memory address:

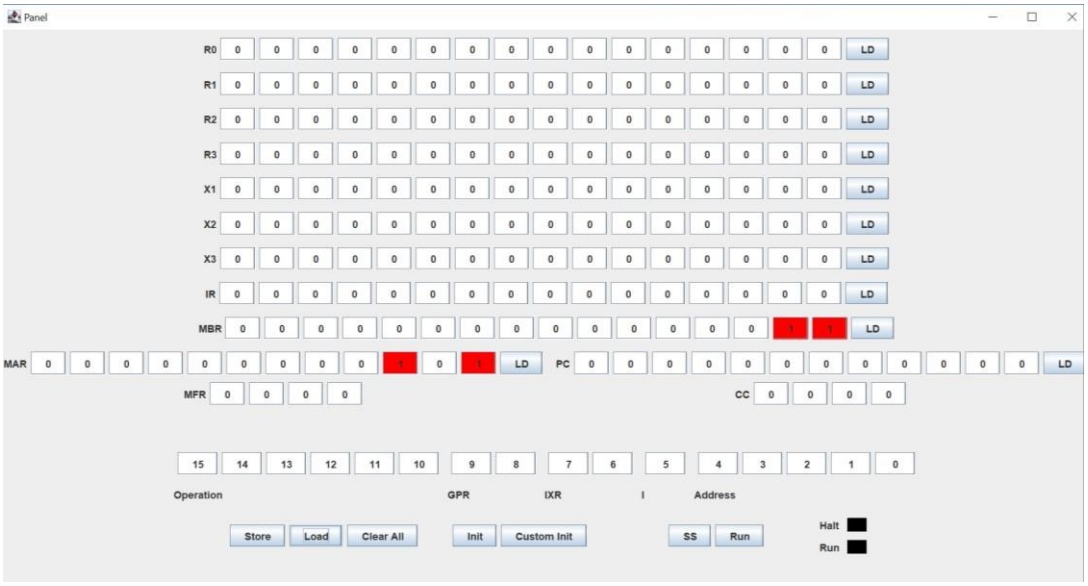


How to Load instruction from memory:

1. Select MAR and then "LD" in the Instruction toggle to load it into the MAR register.

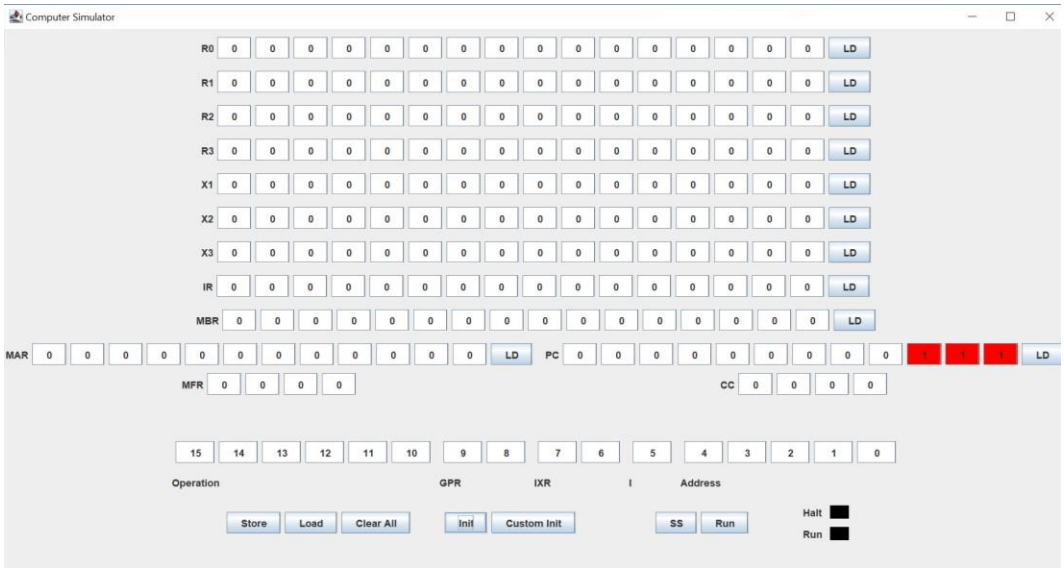


2. The instruction saved in MAR is written in the MBR register when you hit the "Load" button:

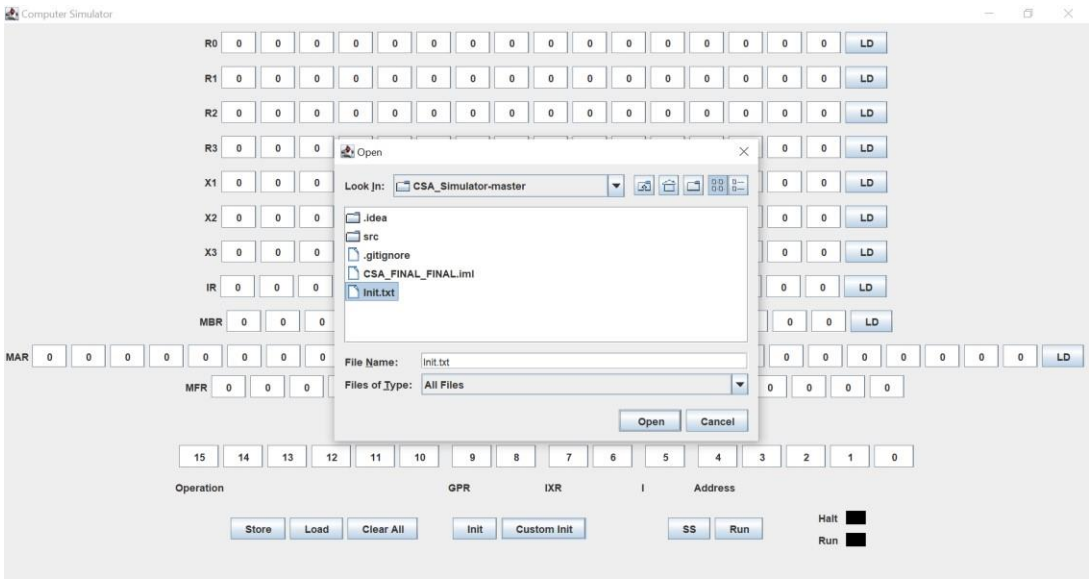


How to Load IPL file or Custom IPL file:

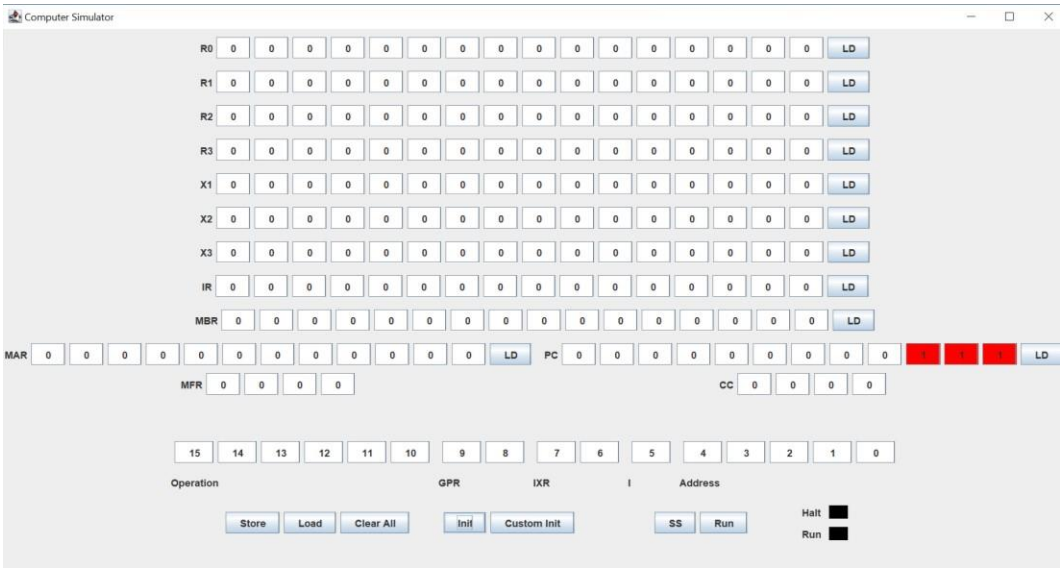
1. The contents of the default Init.txt file are loaded into memory when you hit the "Init" button.



2. Click the "Custom Init" button to load the contents of another txt file into memory:



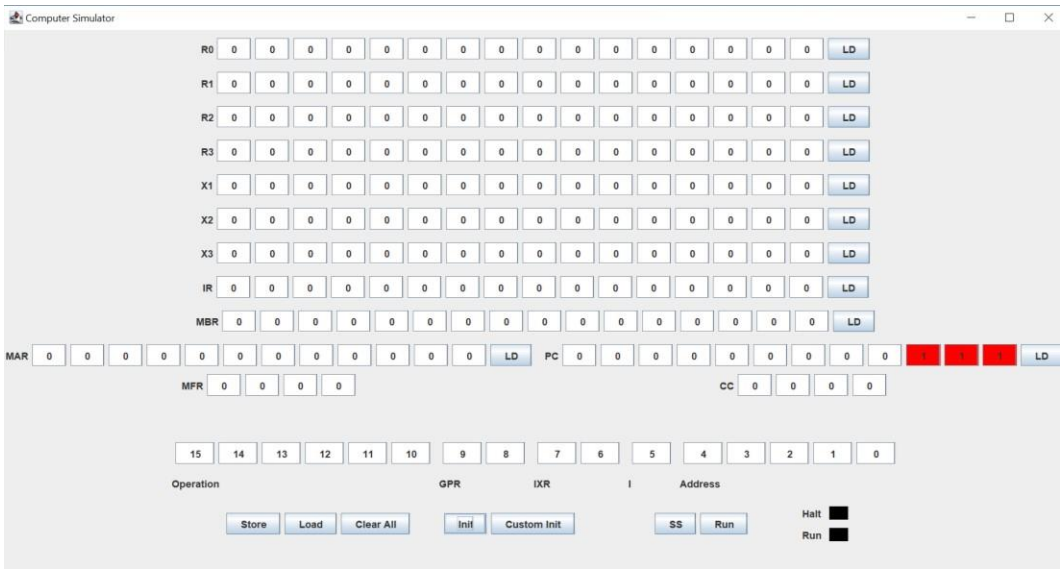
3. When you press the "Init" or "Custom Init" button, the PC loads the memory address of the first instruction:



How to Single Step:

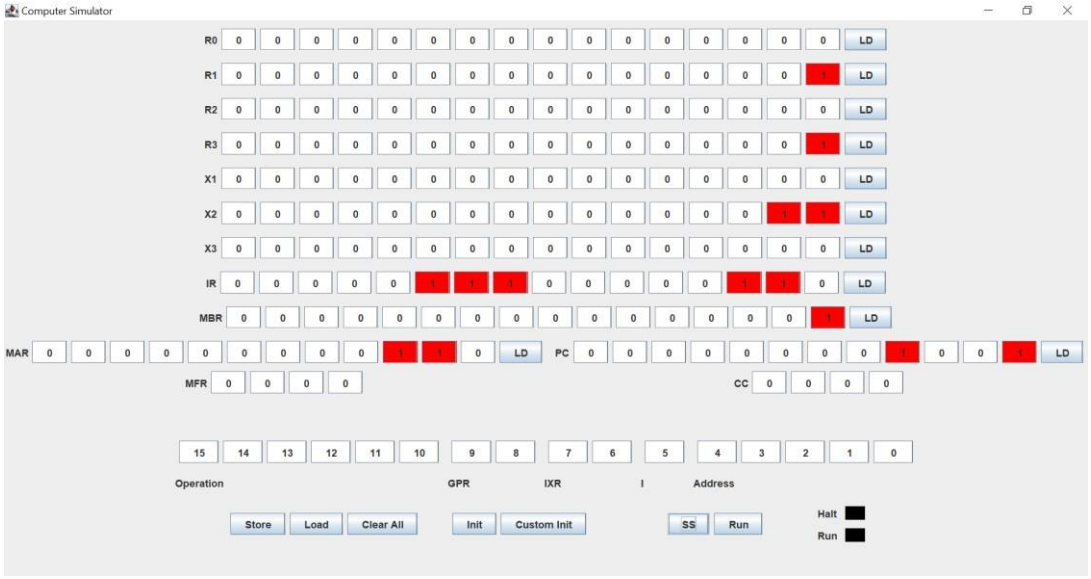
1. If you selected "Init" or "Custom Init", the PC would have loaded automatically; if you want to try a different PC value, click on the Instruction toggle buttons and pick "LD" to load the PC.

   Reading instruction “0007 0906” from the init.txt.

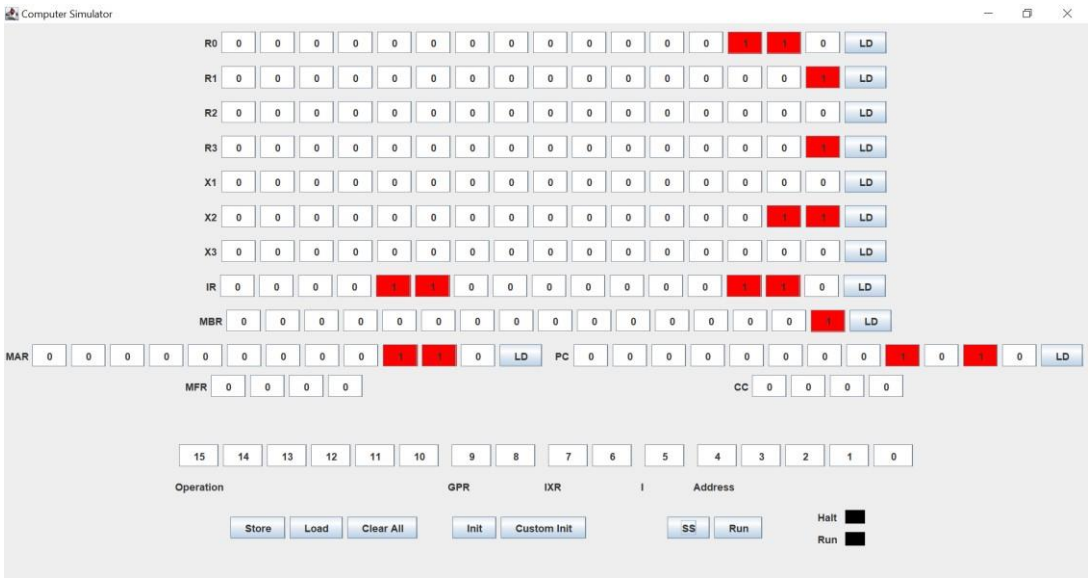


2. There will be a 1-second delay before the registers are loaded with their values when you click the "SS" button to Single step:
   Reading instruction “0008 0706” from the init.txt. PC gets incremented to the value “0008” and IR stores the value “0706”.

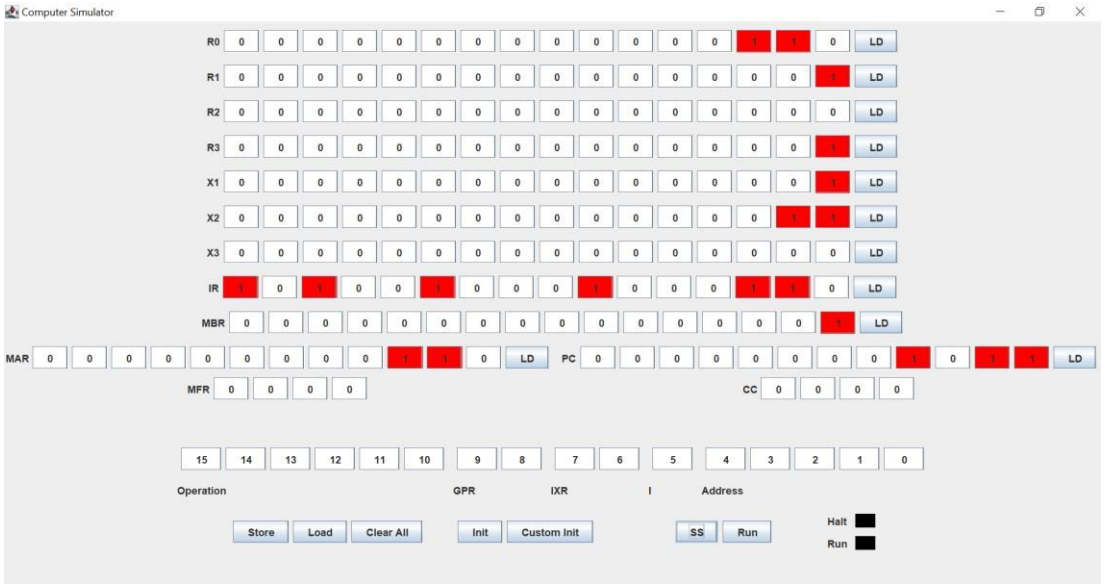
3. When the "SS" key is hit, the values of the registers are loaded in the following order: IR > MAR -> MBR -> GPRs or IXs -> PC=PC+1.
   Reading instruction “0009 0C06” from the init.txt. PC gets incremented to the value “0009” and IR stores the value “0C06”.



Reading instruction “000A A446” from the init.txt. PC gets incremented to the value “000A”.



Reading instruction “000B 4886” from the init.txt. PC gets incremented to the value “000B”.



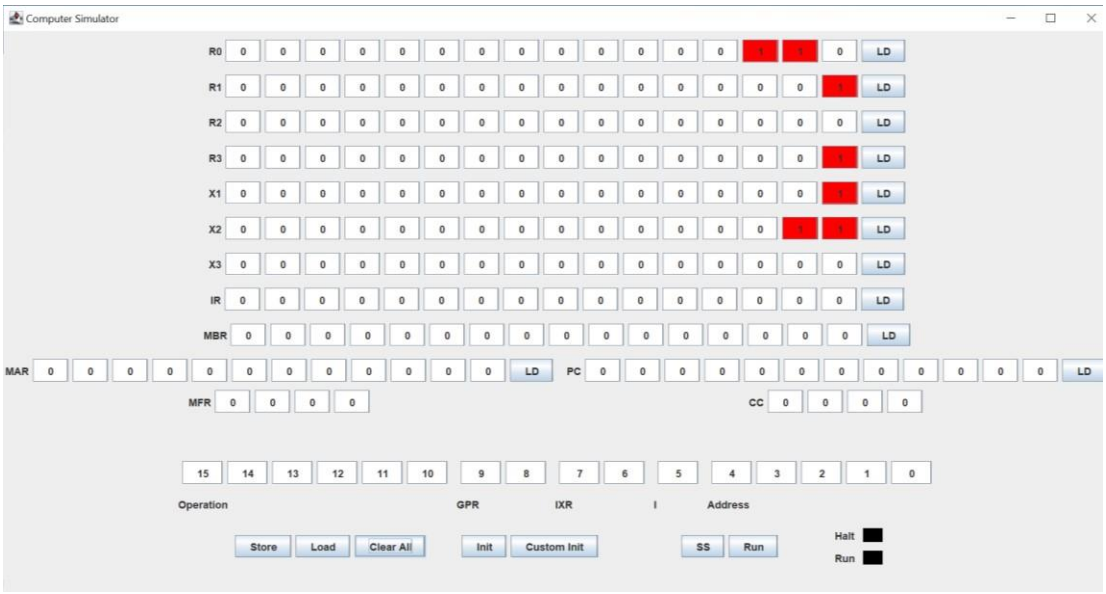
4. If the Halt condition is met, the HALT LED will illuminate:

How to Run:

1. Click the "RUN" button which will perform the SS in a loop until HALT is encountered; the RUN LED will glow red until the run is complete, then turn black when HALT is detected. The RUN is carried out by the instructions in the init.txt file:

Clear LEDs:

The “Clear All” button will clear the registers essential for the instruction execution (like IR, MBR, MAR, PC) LEDs:


Computer Simulator
R0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 LD
R1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 LD
R2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 LD
R3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 LD
X1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 LD
X2 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 LD
X3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 LD
IR 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 LD
MBR 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 LD
MAR 0 0 0 0 0 0 0 0 0 0 0 0 LD
PC 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 LD
MFR 0 0 0 0
CC 0 0 0 0
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Operation
GPR
IXR
I
Address
Store
Load
Clear All
Init
Custom Init
SS
Run
Halt
Run

End!