Back-end:

Our team uses a list named memory to simulate the computer memory. The index of it is MAR and the value corresponding to each MAR is MBR. All the content in the memory will be initiated as 0000000000000000.

Clicking LD button, the program will load an instruction from 16 the binary value button on the bottom side into corresponding location.

Clicking STORE button, the program will get the MAR and MBR value from the GUI and store them into the memory. Memory[MAR] = MBR

Clicking St+ button, the program will execute the same step as STORE button does, and then increase the MAR by plus 1 and display it on GUI.

Clicking Load button, the program will get the value of MAR and MBR. Then using MAR value as address (index of the memory list) to get the binary value in it. Updated MBR display by using that binary value.

Clicking Init button, the program will open the IPL.txt file and read each line of it. Use the first column as address and use the second column as value in that address, and then store it into the memory.

Clicking SS button, the program will get the value of PC, and use it as the address to find the instruction in that address from the memory. Then store the first six binary values into Opcode, put seventh and eighth binary values into GPR, put the nineth and tenth binary values into IXR, put the Eleventh binary value into I, put the last five binary value into Address. After that, computing EA using I, Address and IXR and analyzing the Opcode to do LDR, STR, LDA, LDX, or STX.

Clicking RUN button, the program will go through all instructions until meet the condition of HALT.

GUI:

We used 12 label to simulate and display the binary in it. They are the four GPR, three IXR, PC, MAR, MBR, IR and MFR. Using 16 button displayed on the bottom of the window to let the user enter binary and load it in the GPR IXR registers, PC, MAR or MBR for future use. Put Store, St+, Load, Init, SS, and Run button on the right side