We have three packages including components, coversion and gui. Components consists all kinds of components of CPU and the Memory. The conversion consists three static function that can convert binary, decimal and hex.

In components, the class CPU\_control control all the components' object and can decode the instruction and execute them. The class interact can be used by the GUI which consists an object CPU\_control that can set data, get data and execute instruction and return status of error or not.

**Classes of components:**

Specifically, in class General\_Purpose\_Registers, the class attributes include an

private array which include 4 integer. The function getregister can get the value of the registers and can checks if the index value is between 0 and 3 and then grabs it. The setregister function sets the register according to the new value being passed in and limit the index and the value.

In class Index\_Registers, the class attributes include an private array which include 3 integer. The function getregister fetches the index address if it is between 1 and 3. The function setregister can set the IXRs from GUI and limit the index and value.

In class Instruction\_Register, the class attributes include integers of instruction and all parts of the instruction. The function setinstruction sets all of the appropriate values according to the instruction (part of decode).

And the functions next can return each part of the instruciton.

In class Machine\_Fault\_Register, the class attribues include an integer of Faultindex which can be set to 0, 2, 3 now. The function getFault can return the Faultindex. The setFault can set the Faultindex if appropreate. And the resetMFR can reset the Faultindex to -1.

In class Memory\_Address\_Register, the class attribues include an integer of address of Memory that the CPU wants to access. The function getMemaddress gets the returns the current address in the MAR. The function setMemaddress Gets the new address, determines if it is appropriate, then sets.

In class Memory\_Buffer\_Register, the class attribues include an integer of Data. The function getData gets the returns the current data in the MBR. The function setData Gets the new Data, determines if it is appropriate, then sets.

In class ProgramCounter, the class attribues include an integer of PCaddress. The function ProgramCounter can reset the PC if appropriate. The function PCPlus increments the PC. The function getPCaddress gets the PC address. The function

setPCaddress re-assigns the PC address if required and fits the appropriate range of the machine.

In class Memory, the class attributes include an array of integers to store the memory data and an integer start which means the start location after the reserved location. The function Memory can reset the memory if the size needs to be bigger than 2048. The function readMem can can return the data of memory, it can determines if the memory address is acceptable or not, else throws and error.

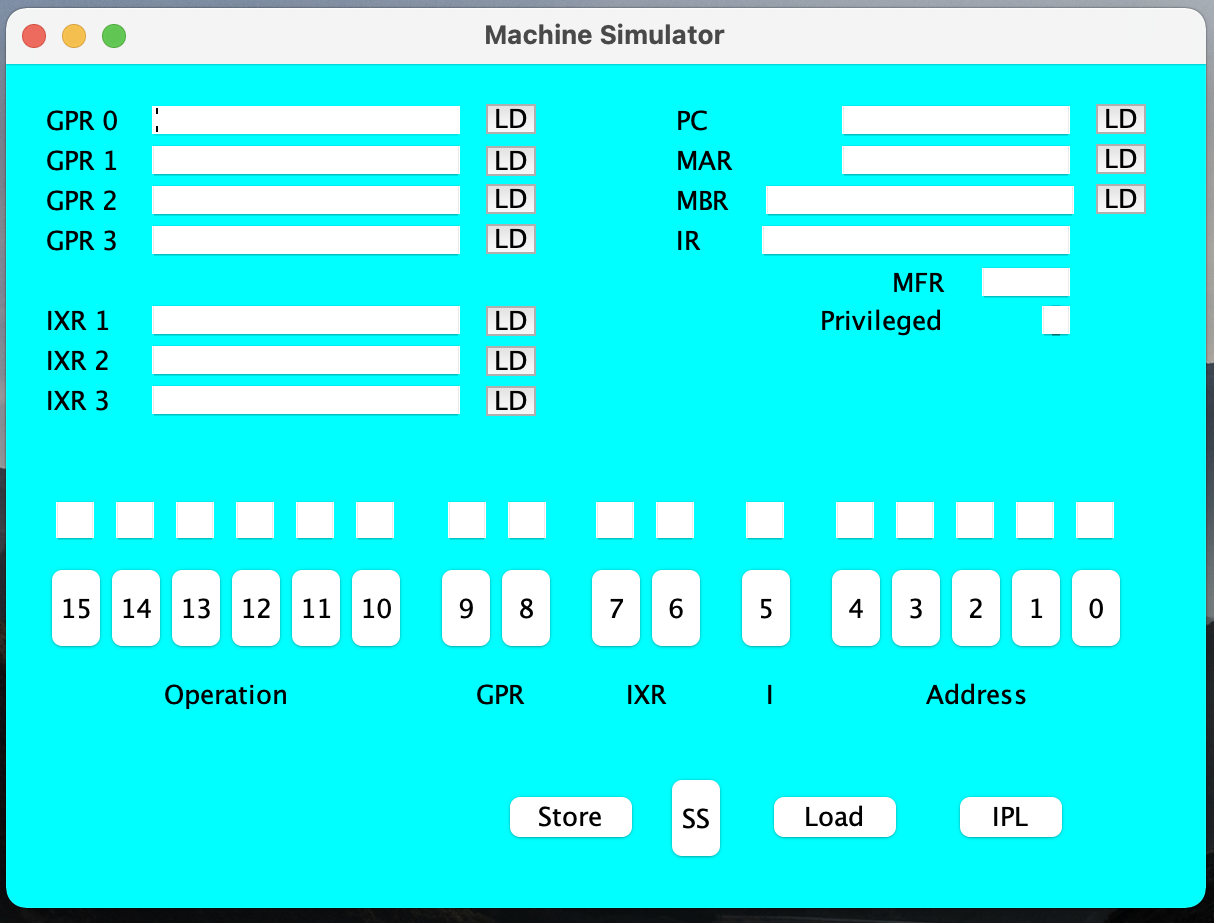
The function writeMem can writes to memory if the address is within the acceptable range and if the new data is within the limits of our bit-limits. The function CPUaccess and CPUwrite are used by CPU\_control to access the reserved location because the CPU needs to handle the machine fault.

In class CPU\_Control, the class attributes include all the components class above and an integer mfindex which is used to determine the machine fault and an integer halt which can be sent to the GUI and tell the GUI to halt and block the runsinglestep function and runinstruction function. The function initial sets the initial components of the machine which can start or restart the machine and reset halt and read IPX.txt and load it to the memory. The function runsinglestep can run the instrution where the PC points to one by one including fetching, decoding and executing. The function runinstruction can run the current instrution in IR without fetching and it is used to execute the load and store button. The function load acts as the load instruction. The function store acts as the store instruction. The function checkaddress can check the access of memory is write or not. If not, we go to set the MFR and get solution which is halt right now and will be called after each read and write to the memory. The function machinefault deal with the machine fault, it will find the solution's address which is 6 and get the solution instruction which is halt right now and run it.

The class interact is the class that GUI can use and all the components are in the object CPU. The class attributes include an object CPU and its class is CPU\_control. The function IPL\_button will call initial function to start or restart the machine and return halt or not, if hault is 1 then it means something went wrong, and the display of MFR will change. The SS\_button will call function runsinglestep and return halt or not, if hault is 1 then it means something went wrong, and the display of MFR will change. The function Load\_button will run the load instruction if the opcode of input is correct which is 000001 and return halt or not, if hault is 1 then it means something went wrong, and the display of MFR will change. The function Store\_button will run the store instruction if the opcode of input is correct which is 000010 and return halt or not, if hault is 1 then it means something went wrong, and the display of MFR will change. The function LD\_button can change the value of each components according to the order of the GUI using the input from GUI. The function get\_number can return the value of each components according to the order of the GUI and it will be used to send the value to GUI.

**GUI:**

The UI is built using the Java Swing Library, with individual UI components mounted on to a JFrame.The JFrame contains a JPanel which has the text fields representing all the registers on the UI.



The individual registers have a label, built using the component JLabel which is just a display name for the register and following that there’s a JText Field, which is a text field used to display the contents of that particular register. There are a few load buttons next to some registers, built using JButtons.

Down below on the Panel we have the space for entering individual instruction. The buttons with numbers 0-15 represent the 16 bit instruction set that can be entered bit by bit in the text boxes above. Each text box only takes 1 bit as input which can be either 0 or 1. No other input can be supplied

Down right on the palen, there are 4 buttons present. Load, Store, SS and IPL. The load button when pressed loads the instruction onto the specified register. Store is used to store the instruction to the memory. IPL button can be used to run the IPL.txt file and the SS button is used to run the instruction one after another as single steps.

The class PanelView.java contains a constructor PanelView which initializes the GUI. All the individual components have been added in this constructor. The class PanelView has a main method that initializes a frame and an instance of PanelView is called to show the UI on the screen.

The Load and Store button functions have been handled using event listeners defined in the code, that then trigger the backend and all the operations in the backend are performed. The text fields have also been limited to only having binary numbers as input and output using action listener(KeyPressed and KeyReleased) methods defined in the code.