Our computer starts at 0 and works its way up.

To begin, we constructed a memory class that would serve as the simulator's memory.

This class oivy employed a basic array to function as a memory mechanism, but did not declare the size of this array directly in this class; the Backend class will do so later. This project also features a register class, in which we have implemented the register's load and save functionalities.

We basically did decoder here Opcode in the instruction class. This class's oprindes index and indirect oprindes index are calculated.

All simulation calculations are conducted in the Backend class, which is a large class. We'll quickly establish a new memory entity, as well as matching registry entities like gprs and wrs. have a pointer to the ISA, which is used to call actual instructions, and a pointer to the mainFrame, which is used to update mainFrame while in this class. First, we'll initialize all the registers and merisory, then bind the munFrame to BLI. We'll also have a function to emulate a "fetch" that uses MAR MBR, IR, and PC, as well as a function to emulate "execute" that uses a control unit and an instruction decoder on a physical chip. In this class, the run function will store the machine.

The project will calculate the effective address for the ISA class.

following the rationale outlined in the specification Determine the halt command, obtain LDR, and

ins STR, LDA, LDX, STX

The main class will be a driving class, and the curriculum will begin from there.