We created our MIPS simulator to be GUI based, so that it can show the instruction memory, data memory, and register file, all at once.

To run the program, please use “testProgDump.txt” to load instruction, and “testDataDump.txt” to load data memory. After correctly loading the program, clicking the “run one step” button will execute instructions step-by-step, highlighting which data/register/instruction is being used. Also, there is functionality that allows us to modify register content or data content directly by double-clicking the cell and entering a “decimal” value to modify the content. There is also a checkbox that contains the functionality of presenting a bit string in decimal form.

Our example instruction file doesn’t terminate because there is an instruction at the end which jumps back to the first instruction. However, if you create a new .txt file that does terminate, then a “You have reached the end of the file!” message will pop up when the end of the file has been reached.

MARS/Mips automatically loads instructions at 0x400000, but we made our program to load instructions at address 0x0 in the instruction memory. Therefore, the PC must initially be 0.

Just like MIPS’ instruction memory, we implemented our instruction memory to be byte-addressable in order to make following work smoothly: PC+4, shifting left 2 bits for the beq operation and J operation, and lw and sw offsets are multiples of 4. Therefore, the array size of the instruction memory is 800, which can hold up to 200 instructions.

However, since we are not implementing the instructions to access half word or byte (lh, lb), we thought that byte addressability within data memory was not necessary, so we implemented the data memory to be 32-bit addressable. Therefore, the array size of the data memory is 500, which can hold up to 500 pieces of data. One vital aspect was to make our simulator as close as possible to the actual MIPS implementation, therefore we assigned each data’s address to be a multiple of 4.