**Simulator documentation:**

The simulator is comprised of 6 different method types:

* Helpers: Methods called by other methods in the program, that have minimal functionality, and are mainly used to translate variables from various formats into other formats.
* Init: Inititalizing methods. They prepare the simulation environment for the program to run.
* Output: Methods to print to the output files. They format the strings according to the desired format for each file and print them to the relevant files.
* CPU Run: The methods that comprise the CPU-level commands that we simulate, as well as other necessary functions to assist in that.
* Peripherals: Methods to simulate the external hardware, especially the parts which run in parallel.
* Finalize: Methods to finish the program, call the relevant output methods and free the allocated memory.

Other than these 6 types, we have 2 “main” functions:

* **“THE”** Main function: The interface with the user. The function called when “./sim.exe” is run. Verifies the integrity of the files given, and not much more.
* Main Loop: run\_program(), The Loop that signifies each cycle run by the CPU.

**Helpers:**

* void hex2bin(char\* hexstr, char\* binstr) **–** Translates hexstr to binary string and save the value in binstr.
* int get\_bin\_value(char\* strval) – Gets hex string “strval” and translates to integer value under 2’s complement. **(<string.h>’s strtol had bugs under specific circumstances).**
* void sign\_extend\_immediate(int immediate, char\* immstr) – Sign-extends 3 Byte word to 8 Bytes.
* char\* get\_IO\_reg\_name(int regnum) – Translate IO Reg # to IO Reg Name.

**Init:**

* bool init\_files(char\* argv[]) – Opens all file pointers (input and output). Returns false iff file opening failed.
* bool init\_code() – Read Imemin.txt into local array.
* bool init\_mem() – Read Dmemin.txt into local array.
* bool init\_disk() – Read Diskin.txt into local array.
* bool init\_misc() – Initializes all local variables.
* bool init\_values() – Wrapping function, calls the other Init functions.

**Output:**

* Each function writes to a specific output files. Using internal variables, we made sure not to write redundant zeroes (for example, in dmemout, diskout, etc.)

**CPU Run**:

* void decode\_instruction(char\* instruction) – Reads the current instruction (saved from Imemin) and translates the instruction string into the relevant registers and other variables.
* void prepare\_instruction() – calls “sign\_extend\_immediate” on immediate 1 and 2.
* void handle\_input() – Handles reading from IO Regs.
* void handle\_output() – Handles writing to IO Regs.
* void run\_instruction() – Main functionality of CPU. Runs the relevant instruction on relevant Regs. Basically a big switch-case statement.

**Peripherals:**

* void write2disk() – Writes the contents of memory (from local mem array) to local disk array.
* void read\_from\_disk() – Writes the contents of disk (from local disk array) to local mem array.
* void update\_disk\_timer() – Checks if we need to update disk timer, and does it if needed.
* void update\_irq() – Checks if we need to raise any interrupt, does so, and updates the global irq flag.
* void handle\_irq() – Jumps to ISR if irq == true.
* void update\_timer() – Updates timercurrent if enabled.
* void get\_next\_irq2() – Reads the next line from irq2in.txt and saves the clock cycle value in a program variable.
* void save\_in\_monitor() – Handles interface with local monitor array.
* void handle\_parallel\_systems() – Wrapping function, calls other Peripheral funcs.

**Finalize:**

* Call write\_to\_output(), close file pointers and free allocated memory.