Program #1 – Emulator

William Doering

Running the Program

To run the program, you must first compile it using the command given below while in the program's directory. It will take an optional object file formatted using the Intel HEX file format. It will load this file into the emulator's memory. After it is running it will have 3 separate functions along with an exit command.

```
$ python3.6 main.py [file.obj]
```

Read Single

To read the value in a specific memory space, type the address to that space in hex format. It will output the address you typed along with the value stored there.

```
> 200
200   h4
```

Read Multiple

To read the values in a range of memory, type the starting address followed by a period and then the ending address in hex. This will output the values in chunks of 8. There will be an address at the beginning of each chunk for ease of viewing.

```
> 200.20d
200   h4 7a 99 00 ab cd 74 e4
208   00 69 00 ba 04 20
```

Assign

To assign value(s) to memory, type in the starting address followed by a colon. Then type in the new value for each following address in order, separated by space.

```
> 300: aa ab ac ad ae af a1 a2 a3 a4
```

Exit

To exit, type exit and press enter. There needs to be a space on either side of the word or it needs to be on a new line with a space following it to work. The other two forms of exit are not implemented.

Testing

For testing, I first used the files given and the sample commands in Appendix A of the assignment. I then made a random file using the same format as the test files and typed in random commands to see that it works in general. There is no error checking on input so the commands need to be typed correctly.

Functions

Below is a list of the functions in the program with their descriptions.

main()

This starts the program. It calls the parse file and gets user input. It then passes it on to evaluate.

evaluate()
This evaluates the user input and determines what command the user is giving. It then calls the appropriate function for that command.

prog_run()
This “runs” the program. For now it simply prints out the opcode labels and the address the program would start at. Later it will be expanded to run the program and print out the commands line-by-line.

print_one()
This will print the value at the address given.

print_range()
This will print the values at the range of addresses given.

edit_mem()
This will edit the values at and after the address given.

parse_file()
Parses the incoming file and stores it in memory

scream_and_die()
Output closing message and exit the program.

class Memory
This holds all the memory and registers for the emulator.

Notes
So as of now the program stores memory as a string with the hex values. I made the program so I could easily change it to holding its decimal equivalent. I will do this when I get the emulator to run the program.