Assembler. 3 “Seyed Ali Maher” 9932113

Here is a summary of my way of designing an assembler:

At the beginning I read the inputs from a text file. In the following I split them line by line and also delete the “,”.

In the next I define Some functions. One of them that helps us to find which instruction (ADD, AND, SUB and OR) is called. And pass us to it’s exact function for the remaining calculation. And the others that help us to get the MOD and Direction Bit.

Then I define the Instructions. And in each instruction I check for the registers size (8, 16 or 32 bits) of the given operands. The calculation of each instructions is based on their exact opcodes and other buts and ifs :) At the end it will return the machine code of the given inputs.

Also, I defined the registers lists based on their size. And defining the registers values.

In calculation part, after considering the opcodes and assigning the registers values code, the string of integers will be converted to binary in the first level, and then then whole answer will be converted to the hexadecimal format.

At the end I consider a printing part that will print the machine code of each inputs in separate lines if the machine code is valid.

“Good Luck”

It is necessary to say that I specified different warnings for the different invalid inputs.

I add a new function with name “calculate_distance”. By this I calculate the amount of jump based on either it is a backward jump or forward jump. If we get a positive difference we just return that, but if we reach a negative value, then we the following formula to reach the correct value.

Amount = ff(255) – Number of Bytes – 1

With a variable I save the value of the counted bytes. And i updated after each instruction which used.

I also define a new instruction with name “JMP” which helps me to handle the jmp instruction opcode and machine code which had to be returned.

The major difference of this part with rest of the parts, is in the way that we return last machine code. Because we need to access the address of label and the address of the JMP at the same time in order to reach the correct value. In hence we return the whole converted code at the end in one line.

“It always seems impossible until it’s done.”

“Good Luck”