Overview of the Assembler

Authors: Haoyi Shi & Haoyuan Du

1. How does it Work?

Firstly, our assembler reads the certain .as file given from the args in console. It should be a file with assembly language in it. It will output the result to another file given by user or print it to the console.

Next, assembler is going to read all labels and save them. Also, we save the index of the label's line. If any labels are illegal (too long, doesn't start with a letter, etc.), the assembler will print the error and shut down the program immediately.

After reading all labels, assembler reads the file line by line again. Starts at the instructions, saves each non-whitespace character into a place (We call it "allLines") until it reaches the comments or the end of line.

Then, we are going to check the first character line by line in "allLines" (sometimes also need to check the second character) and determine which instruction it is. If the instructions' name are weird, again, the assembler will print error. Once we make sure which instruction it is, we change this line into machine code (using bit-wise etc.) and save it. (If has offset field, we change the label to its real offset by looking up our label's array and doing some math. Any offset more than 16 bits will be an error being catched.)

Finally, print the results to the console or to a .mc file in decimal numbers one instruction per line.

2. Any Difficulties?

1. Each time we create a new array and print it, there are some unreadable characters showing on the console. Luckily, we found "memset()" which is very helpful.

2. Think about catching all errors which could happen in the future is a tough work. Sometimes we need to modify our codes so that it could detect the errors. For instance, hundreds of kinds of illegal labels.

3. In C language, we don't have string type, instead we only have char arrays. It takes us some time to get comfortable with. 2D arrays are often used when we'd like to save several lines of strings. It's hard for for-loop.