Berfin Şimşek 2018400009
Çisel Zümbül 2018400093

Our problem was to implement hypothetical CPU. To do this we also implemented an assembler and an execution simulator. Before starting to project, we tried to understand how the CPU works. We executed the example given in the description by hand and observed the mechanism behind. We studied on addressing modes and operation encoding.

First of all, we created dictionaries for instructions and registers matching with their codes. Then, we read the file to detect labels. We put them into another dictionary with their addresses which are calculated during reading process. After that, we read the file line by line once more. We tokenized lines and decided their operation codes according to instruction, their addressing modes according to operand. If the operand is a character we took ascii value of it as an operand. If the operand is enclosed by "[]" we treated inside as an address and decide its address accordingly. If the inside of the parentheses is a register, we took its value from register dictionary as the operand. If not, we took it directly. We converted the hex operation code, addressing mode, operand to binary, formatted them and then we concatenated them all and converted hex again. This way, we created instruction codes and wrote them into .bin file.

At the second part of the project, firstly we created a list whose length is 65536 referring to our 64KB memory. Then, we created a dictionary for the registers and their contents. Also we initialized sign, zero and carry flags as false. After that, we divide our hex instruction codes to three parts so that they can fit into memory, each instruction occupied three index since an instruction is 24 bits. To execute instructions we got from the memory, we constructed a while loop. In this while loop, we got three consecutive values from the memory and we decided their operation codes, address modes and operand of each instruction. Then, for each operation code, we formed an if statement to do necessary operations. After we decided our operation, inside these if statements we checked the addressing mode of it. According to address mode either we took operand directly as immediate data or we treated it as an address and got the data from the memory. In addition, if the operand is a register we took the data from register dictionary. We had some functions in order to avoid repetition. For example, "getdatafrommemo" function takes the address as the parameter and returns the data from the memory, "putdatatomemo" function takes the address and the data as parameters then it puts the data to the given address of the memory. Additionally, there is a function called "sum" which performs addition and subtraction operations then sets the flags accordingly. Finally, by considering some potential invalid inputs, we added print statements that prints the reason of the error. For example if there is an error in the input such as invalid label name, we printed "Syntax error".

We completed the first part of the project easily because it was straightforward. However, at the second part it took time to understand what is needed to be done. We looked over the lectures one more time. Since we started to code after convincing ourselves, we continued smoothly. Until more test cases arrived, we couldn't progress much. We wrote the parts that are not tested but we weren't sure whether they're true or not. After test cases were given, we realized we misinterpreted some of the jump operations. Also some flag

settings at the arithmetic operations were unclear so we handled them accordingly. We had some troubles when there are empty lines in the input, because we underestimated the first part and wrote it without considering the small details. Since we didn't know potential errors that may occur and the format of printing them we couldn't create extensive error handling mechanism.

On the other hand, we finished project in short time because we detected our mistakes and fixed them quickly. Python was easy to learn for us, we improved our Python skills thanks to this project. Both of us done her part in time responsibly, so we had a great job as a team. It was fun to do the project together because it is our only way to socialize. We understand how CPU works, we are now better computer engineers. This project was made with love.