ASSIGNMENT 2

Tuan Hung Nguyen

COS10004

I. Description of my Assembly code

In this assignment, a guessing code will be created with the assembly language.

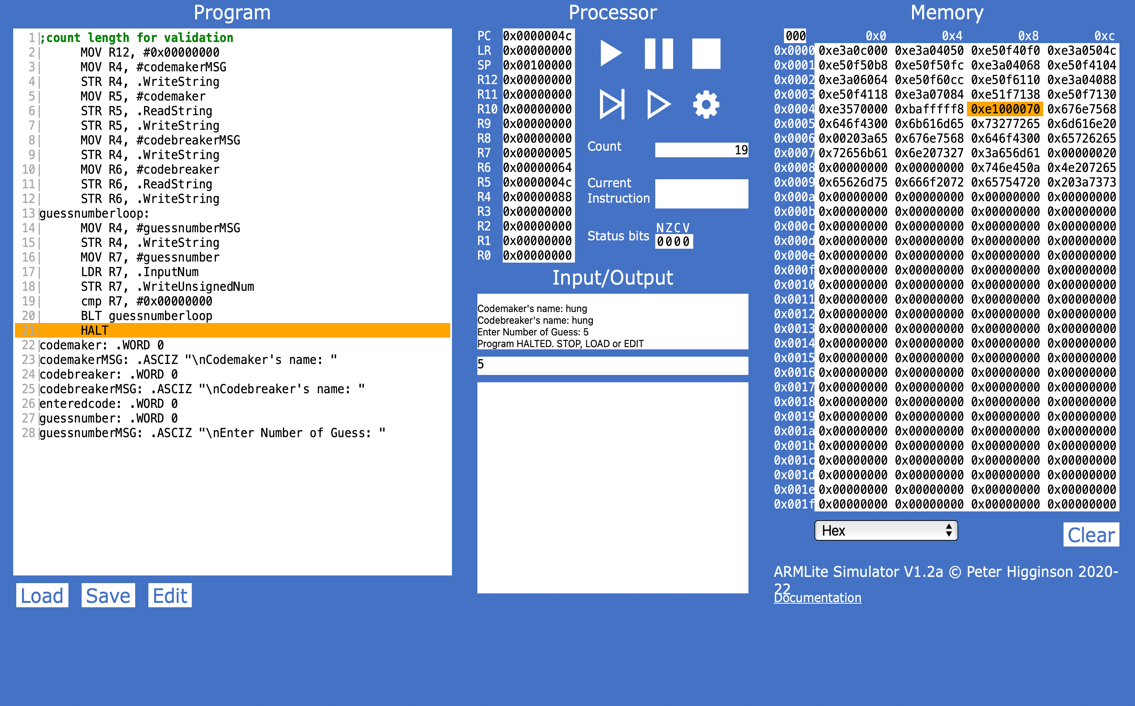
The code maker will make a code with 4 characters and a number of time to guess for code breaker to guess. When code breaker could guess the code, code breaker will win and in opposite case, code maker will win.

II. Design outline

1. General outline

The basic logic of my code bases on function, procedure, if else statement, loop, and branching. The code basically satisfies the requirement of assignment 2 from stage 1 to stage 5 without the bonus stage.

1. Specific outline

Technically, this assignment has 5 main stages. Firstly, in stage 1, several interfaces message will be displayed including the message for code breaker and code maker. The codebreaker’s name will be stored in register R6 and codemaker’s name will be stored in register R5. With the message that duplicated many times, I use register R4.

2.2.1. Code breaker and code maker message

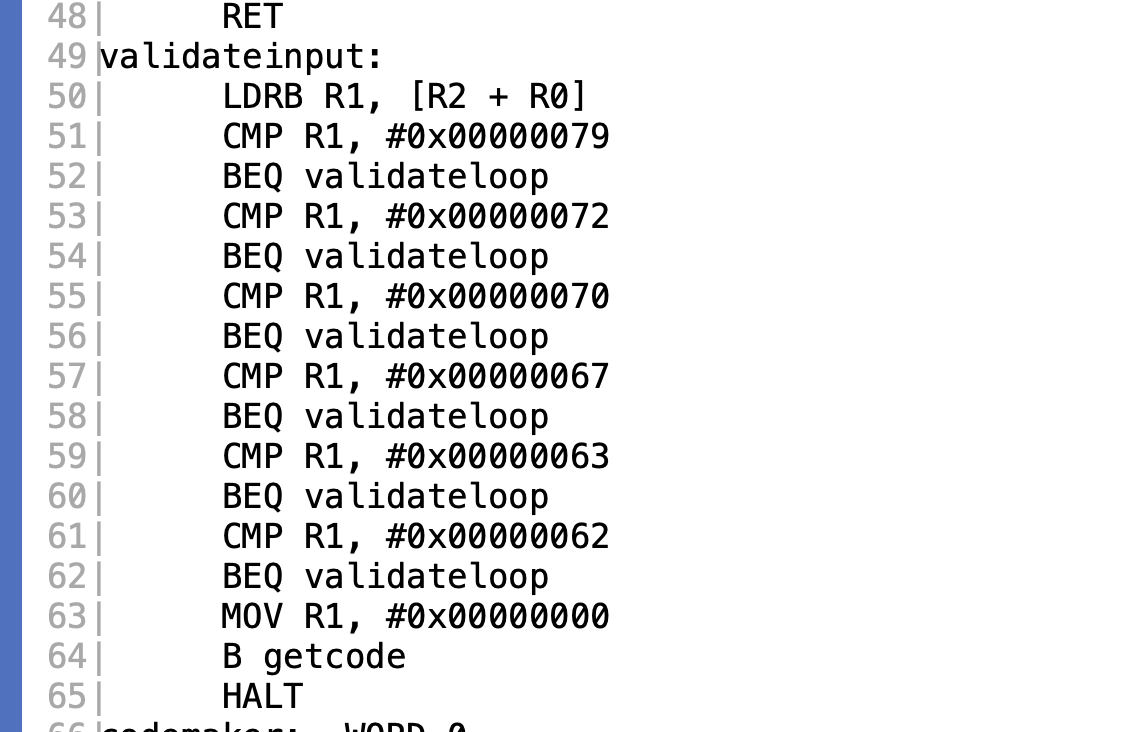
The next stage is stage 2 and stage 3. In stage 2, the game will only allow codebreaker enter 4 characters code. I check if it is a 3 characters code or not by take the LSR to if the character number 4 is 0 or not, the code will branch to get code again if the character number 4 is 0. Checking if the code have 5 character is almost similar with checking 3 characters, the different is checking the character number 5 in the code is 0 or not. This function will take R2 as a parameter to export the code.

Text, letter

Description automatically generated

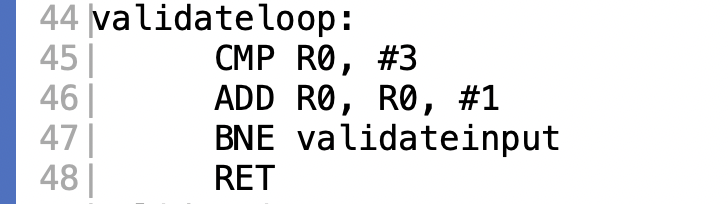
2.2.2. Checking the number of character in the code

With fixed characters allowed, to compare each character in the code, I use LDRB to load each character into register R1 to compare each character with the ascii code of the character.



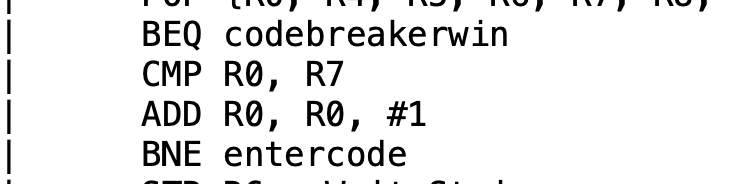
2.2.3. Validating the code

The validation of each character will do it 3 times by the validate loop.

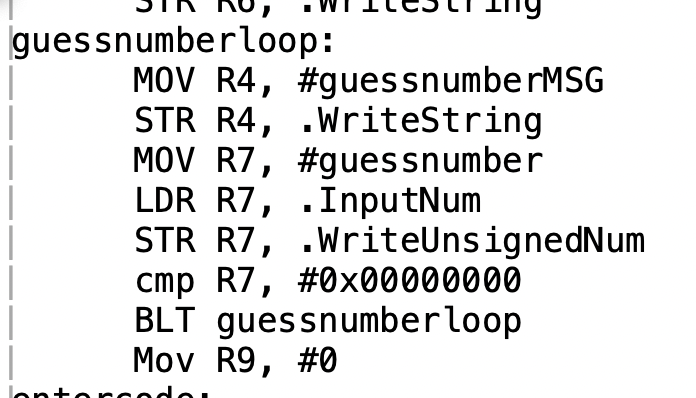


2.2.4. Validating loop

While stage 2 requires create a get code function, I do it with stage 3 after storing the number of guessing to register R7 going with the message “please enter a 4-character code” in register R4. R0 will be compared with R7, if R0 equal R7, the loop will be terminated. R7 will be resubmit if R7 is a negative number.

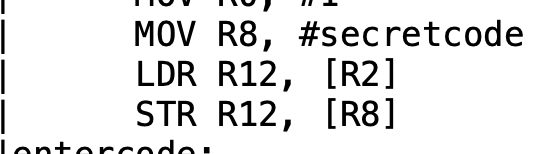


2.2.5. Get code loop for code breaker.



2.2.6. validate if input is negative number.

On the other hand, R8 is defined as a character array secret code to hold the secret code entered.

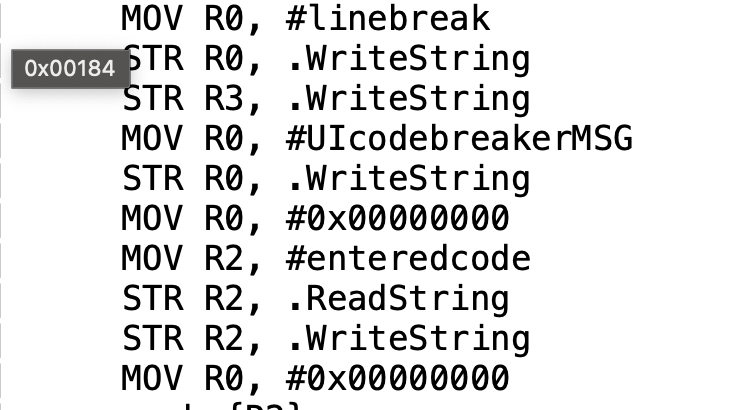


2.2.7. Define character array.

In stage 4, a message to indicate how many times code breaker has entered the code by using R4. Simultaneously, get code function run along with the above message. In case the code is not accepted, it will not have the message to indicate how many times of guessing left then allow code breaker reenter the code with “please enter a 4-character code” message. Text, letter

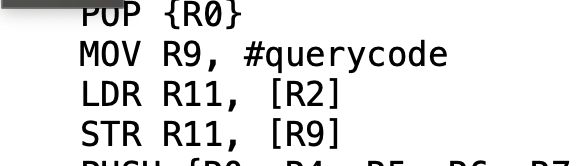
Description automatically generated

2.2.8. Display message “This is guess number N”



2.2.9. display message “Please enter a 4-character code”

R9 will be defined as a variable for an array called querycode to use for later purpose.



2.2.10. R9 hold querycode array.

In stage 5, the basic logic is the loop in loop. I created a function to compare 2 code. Each character in querycode will be compared with secret code. For the first case, if character with the same position in two different code is the same, R0 will plus 1.

Text

Description automatically generated

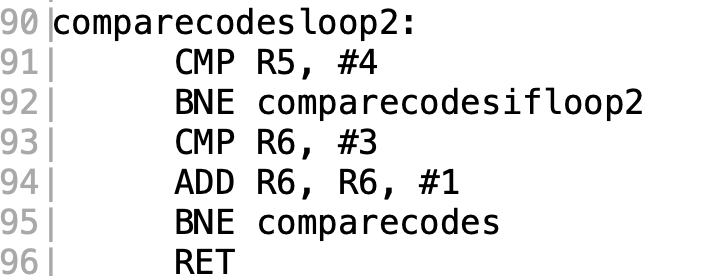
2.2.11. Compare 2 character with the same position.

In the next step, I will compare the character with the position of loop 1 and the character with the position of loop 2. When they are equal, R1 will plus 1. In this function, R6, R5 will take the role of loop 1 and loop 2count variable. While R6 do not need to reset after finishing the loop, each time loop 2 is finished, R5 will be reset to 4.

Text

Description automatically generated

2.2.12. Compare 2 character with different position.



2.2.13. Count variable R6

Text, schematic

Description automatically generated

2.2.14. Count variable R5

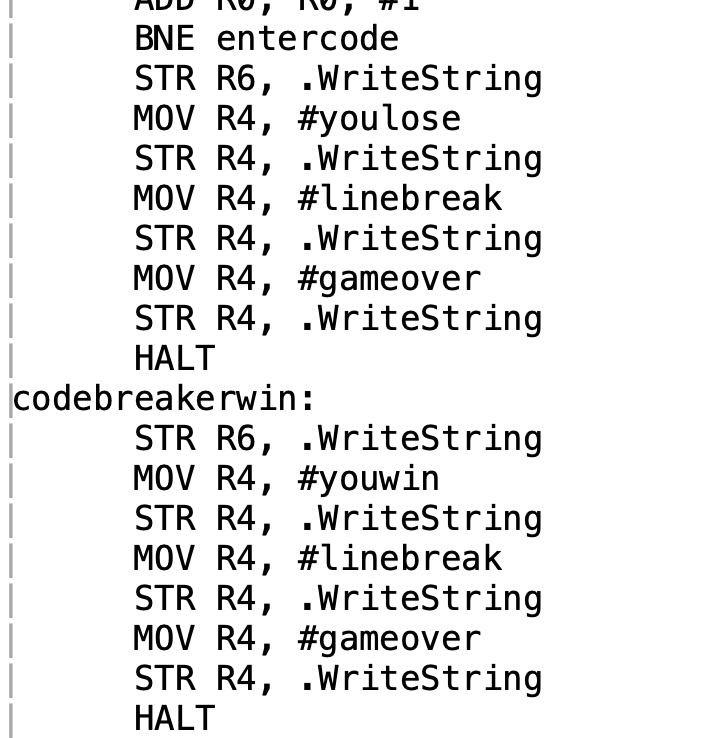
One of the problems in here that, when comparing the case 2 character in different character, case 1 also caculated to case 2, that means the right case 2 will be case 2 – case 1.

Text

Description automatically generated

2.2.15. Caculate the right case 2.

After finishing the loop and the number of entering the querycode, if R0 equal 4, that means all the characters are matched and vice versa. Then the notification to announce which one win the game will be displayed.



2.2.16. display winning notification

III. Assumption

During this project, I built up game without any assumption. All the logic in this can be found in the previous lectures.

IV. Unsolved problems

No problems are found in this assignment.