Lab1 Report Name：杜峯 ID：109550096

1. **Fibonacci：**
   a. **Instructions:110**

      At first, I use two instructions to call “Fibonacci function” in the main function.

      Then, there are five instructions before we call “nFibonacci function” in the “Fibonacci function”. And there are six times of recursion. Including the first time we get in the “Fibonacci function”, there are seven times these five instructions, equally thirty-five instructions. At the last time, since the condition of t0 is not greater or equal to zero, it goes to the last two instructions.

      In the “nFibonacci function”, we have called this function for six times in total. We have nine instructions for a single round of this function, so we will have six times nine equal fifty-four instructions in this function.

      And now we go back to the “main function” for “printResult function”. In the “main function”, I use three instructions to call “printResult function”.

      There are ten instructions in the “printResult function”, and we only call this function for one time. So, at here is twelve instructions.

      In the end, go back to the “main function” for “exit”, it cost two instructions.

      Totally, there are one hundred and ten instructions in “fibonacci.s”

   b. **Stack:14**

      In the “Fibonacci function”, it will append two variables for each time we call this function. It has been called for seven times in the recursions. So there are seven times two equal fourteen variables in the maximum.

2. **GCD：**
   a. **Instructions:69**

      In the “main function”, I use three instructions to call “gcd function”.

      When going to the “gcd function”, there are four instructions worked with “stack” and two instructions for going to “ngcd function”. With running this code, we have two recursions and the first time we get in the “gcd function” for three rounds in total. So here are three times six equal eighteen instructions. At the last round, t3 is equal to zero and does not continues to recursive anymore. Go on the last two instructions in “gcd function”.

      We have two rounds of recursion on “ngcd function”. There are two times the number of instructions in the “ngcd function”, equal two times nine equal eighteen instructions.

      After recursion, we go back to “main function” to call “printResult function” with four instructions.

Similarly to the “fibonacci.s”, we only have single round for “printResult function”. There are twenty-two instructions.

In the end, there are two instructions for exit the program.

Totally, there are sixty-nine instructions in “gcd.s”.

b. **Stack:9**

There will append three variables for each time we call “gcd function”. And we call three times on this function. As the result, there are three times three equal nine variables in maximum.

3. **Bubble sort:**

a. **Array=[5,3,6,7]   size=4**

b. **Instructions:139**

At first, we print str1 with three instructions in the “main function”. And call “printarray function” with one instruction.

In the “printarray function”, I use three instructions to load data. And using loops to print the number in the array. Each loop cost eight instructions to print a single number. We have four numbers, so it totally costs four times eight equal thirty-two instructions. Then the return cost one instruction.

Afterward, go on the bubble sort with two instructions in the “main function”.

In the “bubblesort function”, using three instructions to load data. Then, go on the “loops_i” and “loops_j”. For each loops_i, I will detect if i(s1) is less than s0(size=4). If it is true, go to the loops_j, and start to determine whether the numbers need to be swapped or not.

In the “swap function”, I use two instructions to find the address which needs to be swapped. And use two “lw” and two “sw” to refresh the variable in the array. Than return. It totally cost seven instructions.

Go back to the loops, with the condition in the array, it totally cost four rounds of “loops_i”, and three rounds of “loops_j”. The number of instructions in these two loops is forty-four. And it cost one instruction to return to the “main function”.

Then, I used three instructions to print str2.

And used one instruction to call “printarray function”. Same as the second paragraph. It also cost thirty-five instructions in “printarray function” and one instruction to return.

In the end, it cost two instructions to exit the program.

Totally, there are one hundred and thirty-nine instructions in this program.