Lab2 ARM Assembly I

0516076 施威綸

# Lab objectives 實驗目的

Familiar with basic ARMv7 assembly language.

In this Lab, we will learn topics below.

● How to use conditional branch to finish the loop.

● How to use logic and arithmetic instructions.

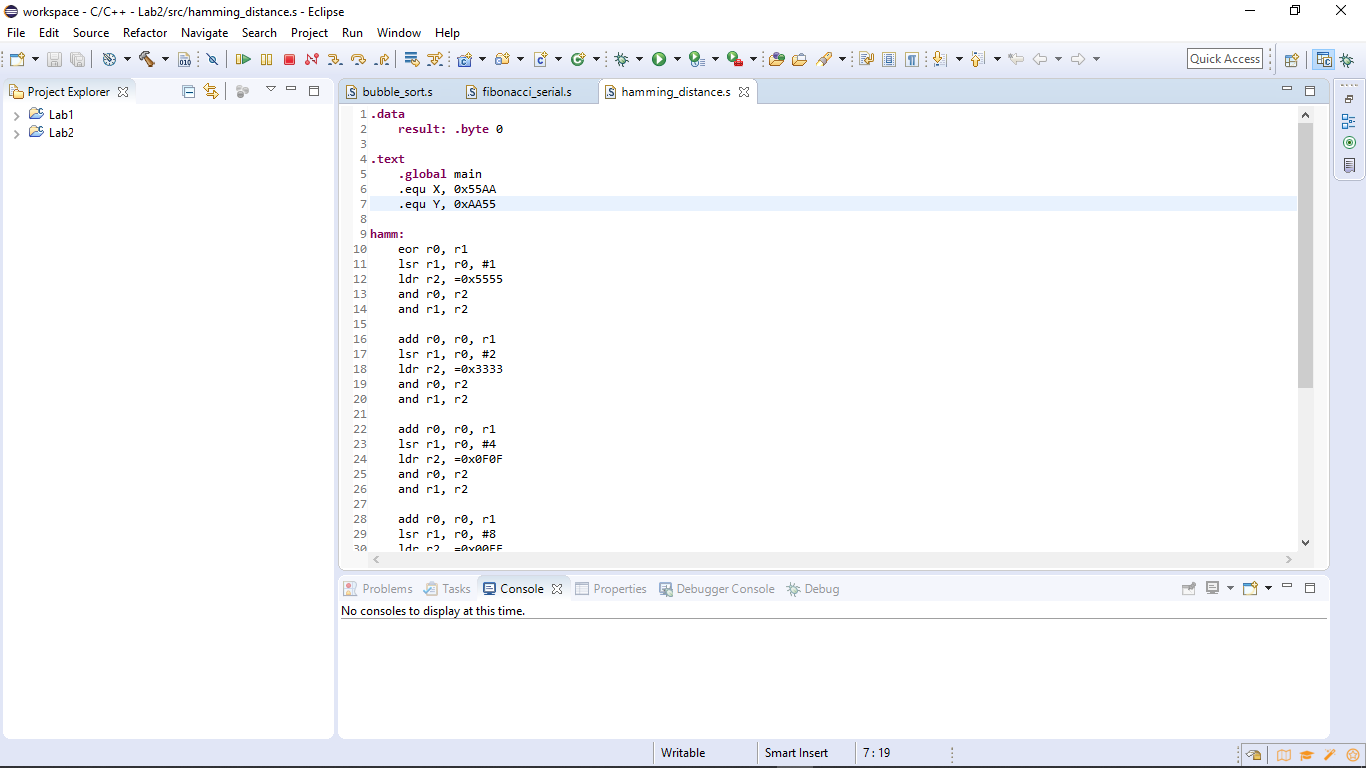
● How to use registers and basic function parameter passing.

● How to access memory and array.

# Steps 實驗步驟

* 1. **Hamming distance**

Calculate the Hamming distance of 2 half-word (2 bytes) numbers, and store the result into the variable “result”

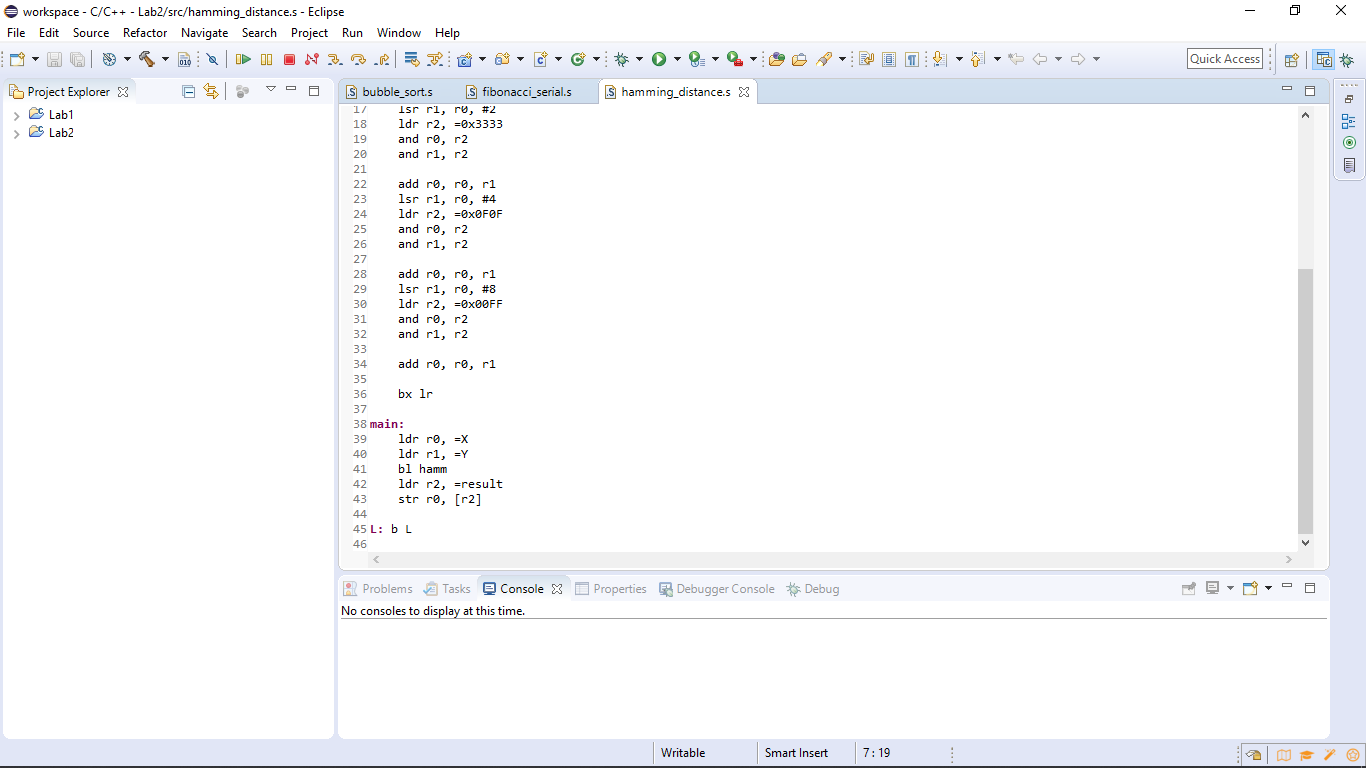


*10: we xor 2 strings to figure out the difference*

*12: load (0101 0101 0101 0101)2 into r2*

*18: load (0011 0011 0011 0011)2 into r2*

*24: load (0000 1111 0000 1111)2 into r2*



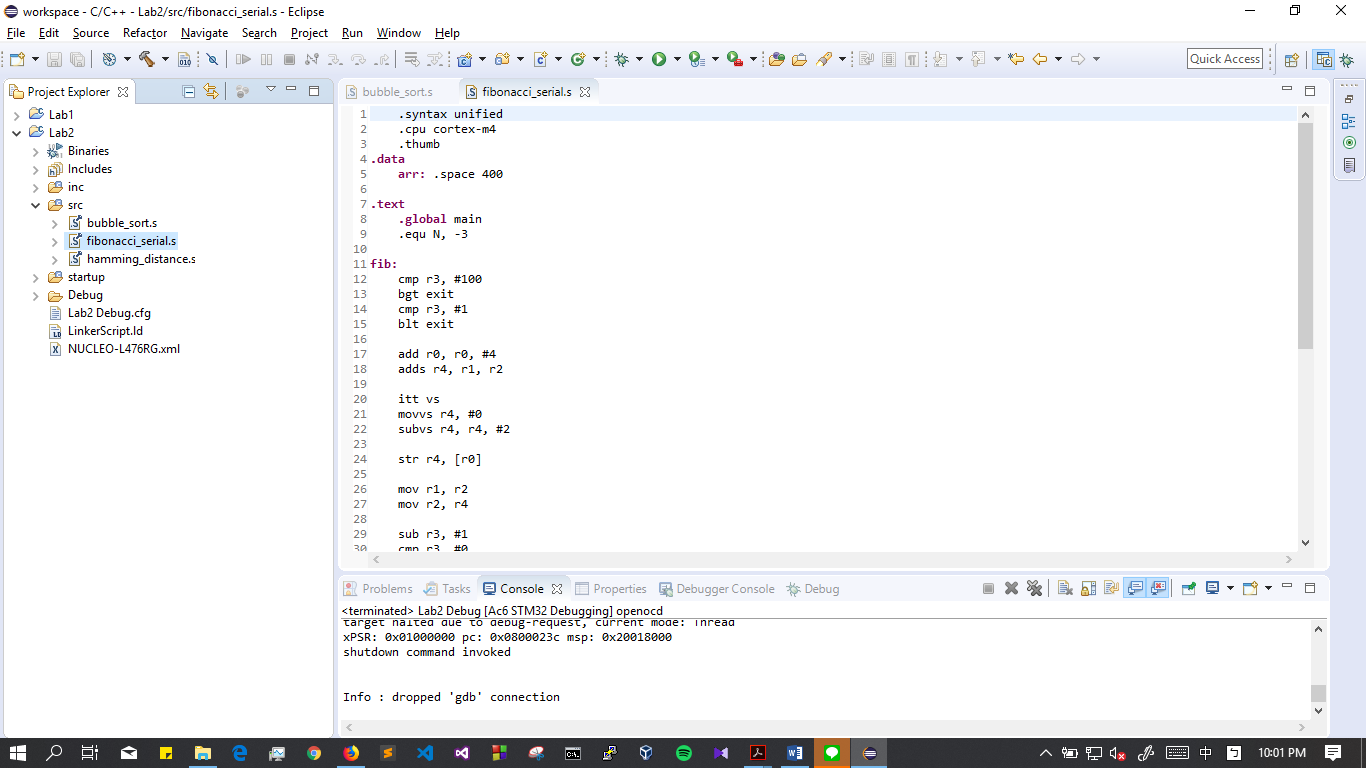
*30: load (0000 0000 1111 1111)2 into r2*

*36: branch to caller*

Hamming distance is basically using XOR to figure out the different “bits” of 2 numbers. The next step is to count the hamming weight, as well as the number of different bits with the result of XOR. By this algorithm, we easily came up with the number using AND & Right Shift to add counts in a tree pattern.

* 1. **Fibonacci serial**

Declare a number N (1 ≤ N ≤100) and calculate the Fibonacci serial Fib(N). Store the result into register R4.



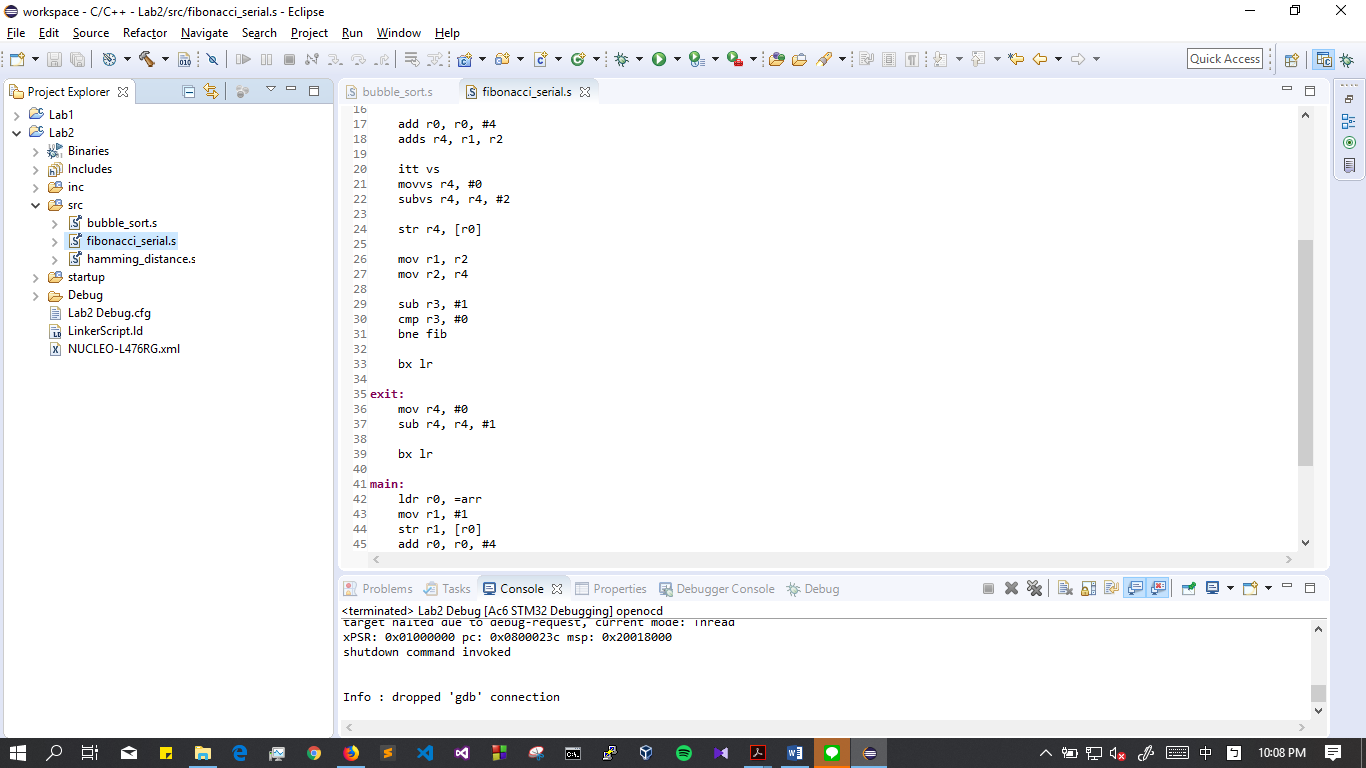
*9: in this case, N = -3, which is invalid*

*13: when N > 100, branch to exit*

*15: when N < 1, branch to exit*

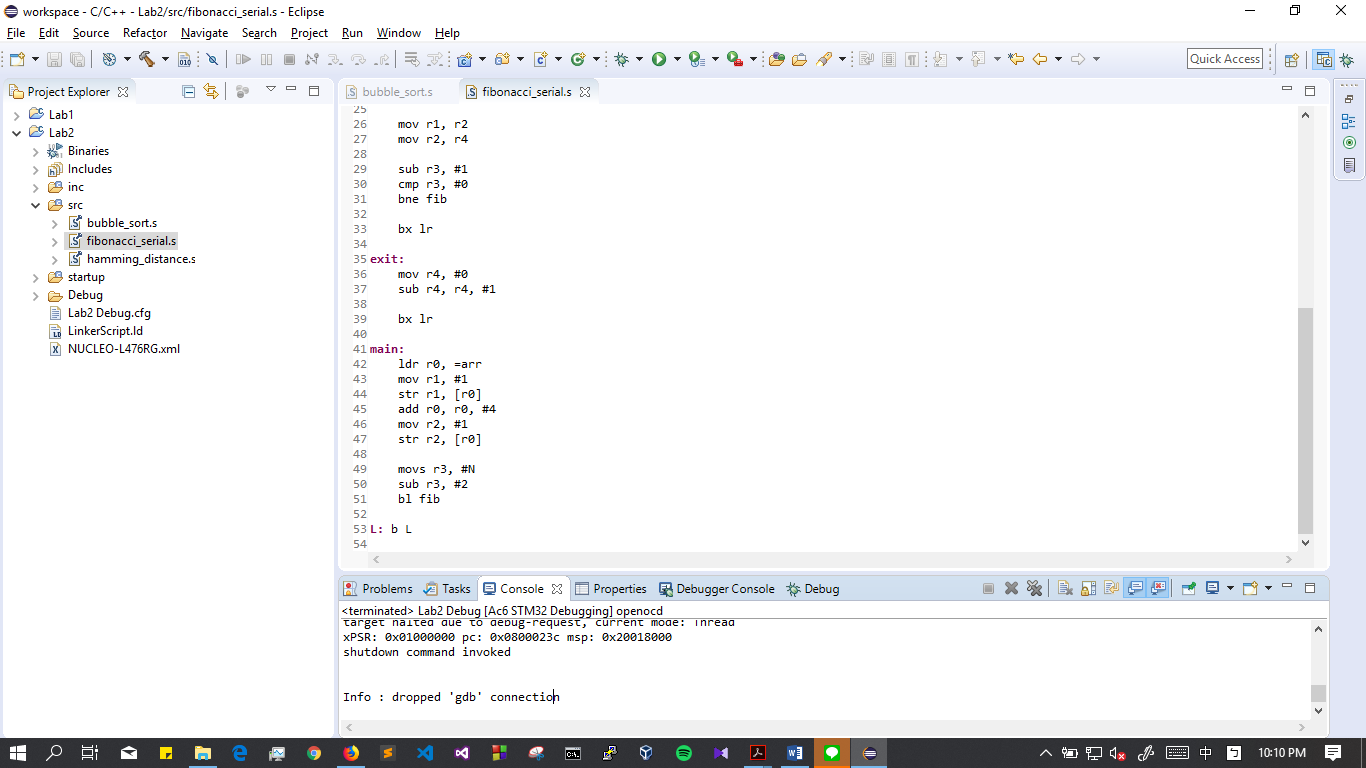
*18: r4 = fib(n), r1 = fib(n-2), r2 = fib(n-1)*

*20: IT block, return -2 when the result is overflow*



*26: move fib(n-1) to r1, fib(n) to r2, prepare for the next loop*

*35: return -1 when N > 100 or N < 1*

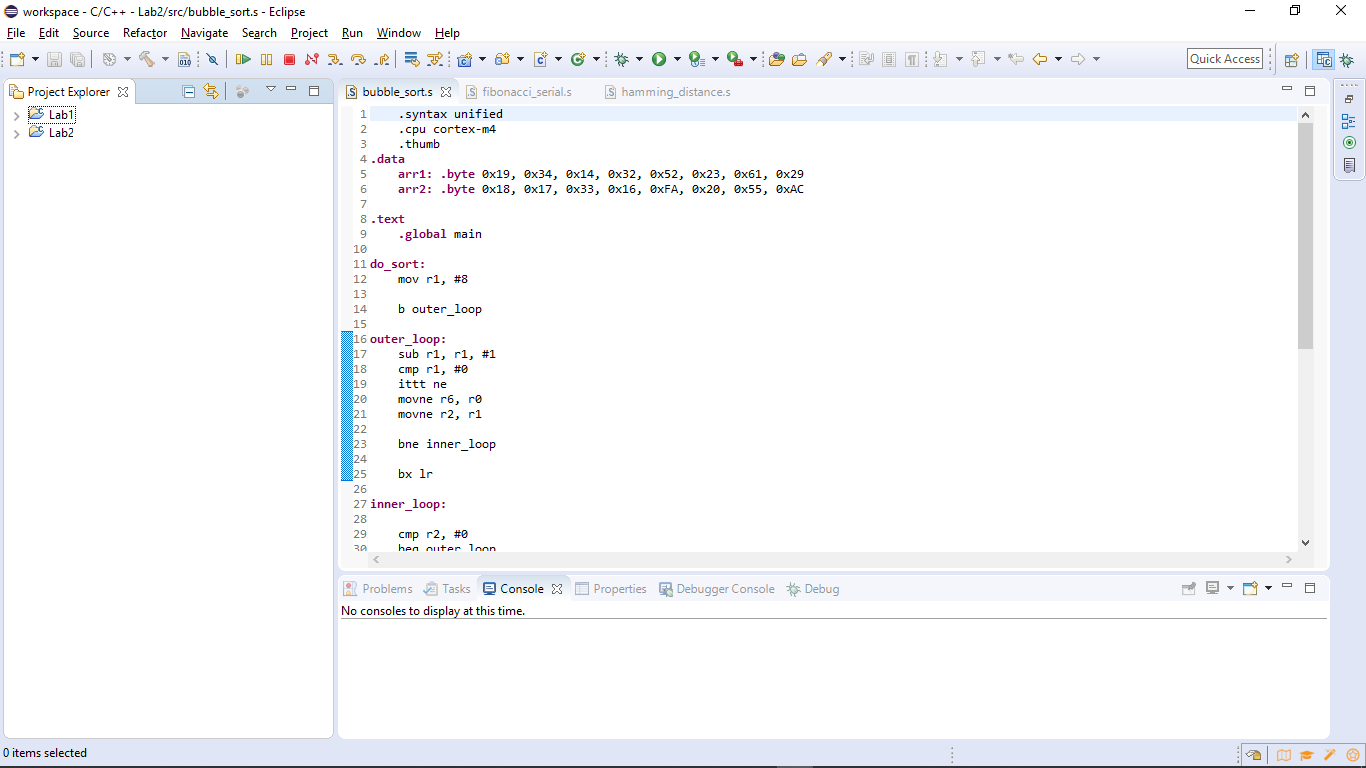


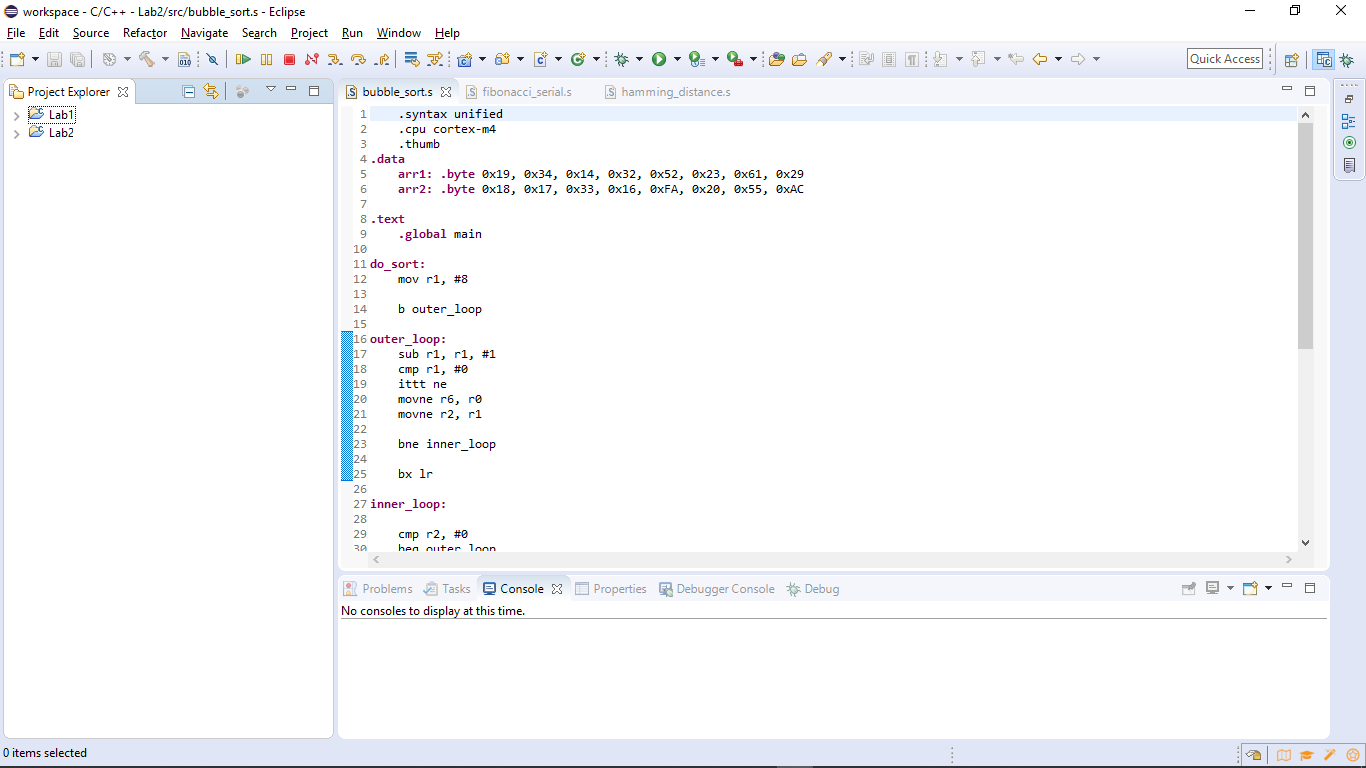
The function fib is actually a for loop which i = N. We added branches of returning -1 and returning -2 at the head of it.

The return value is in r4.

* 1. **Bubble sort**

Implement the Bubble sort algorithm for the 8-byte data array with each element in 8 bits by assembly.





*12: move the size of array into r1*

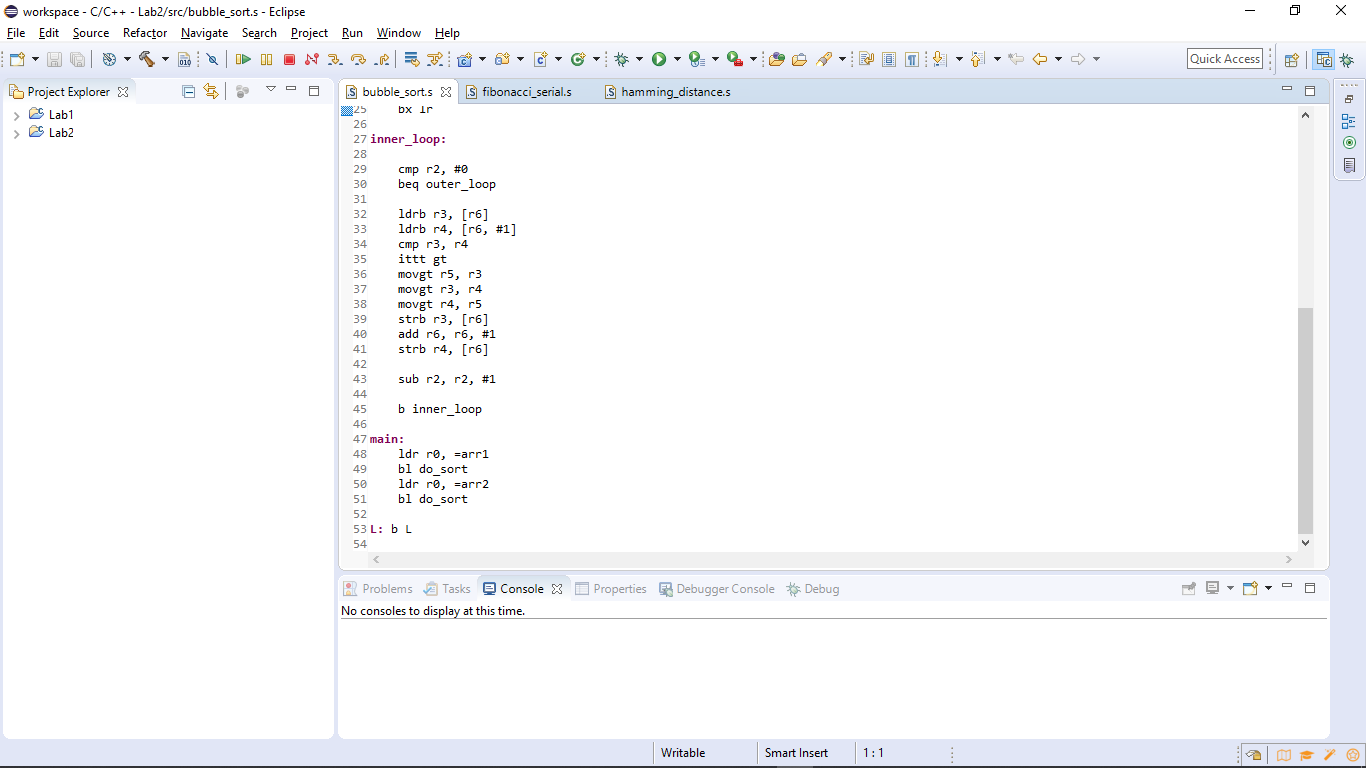
*19: IT block, run inner loop if r1 != 0*

*20: r6 temporarily store the address of the array*

*25: directly return to caller when outer loop is done*

*35: IT block, switch contents when arr[n] > arr[n+1]*

*41: store final r3, r4 to memory*

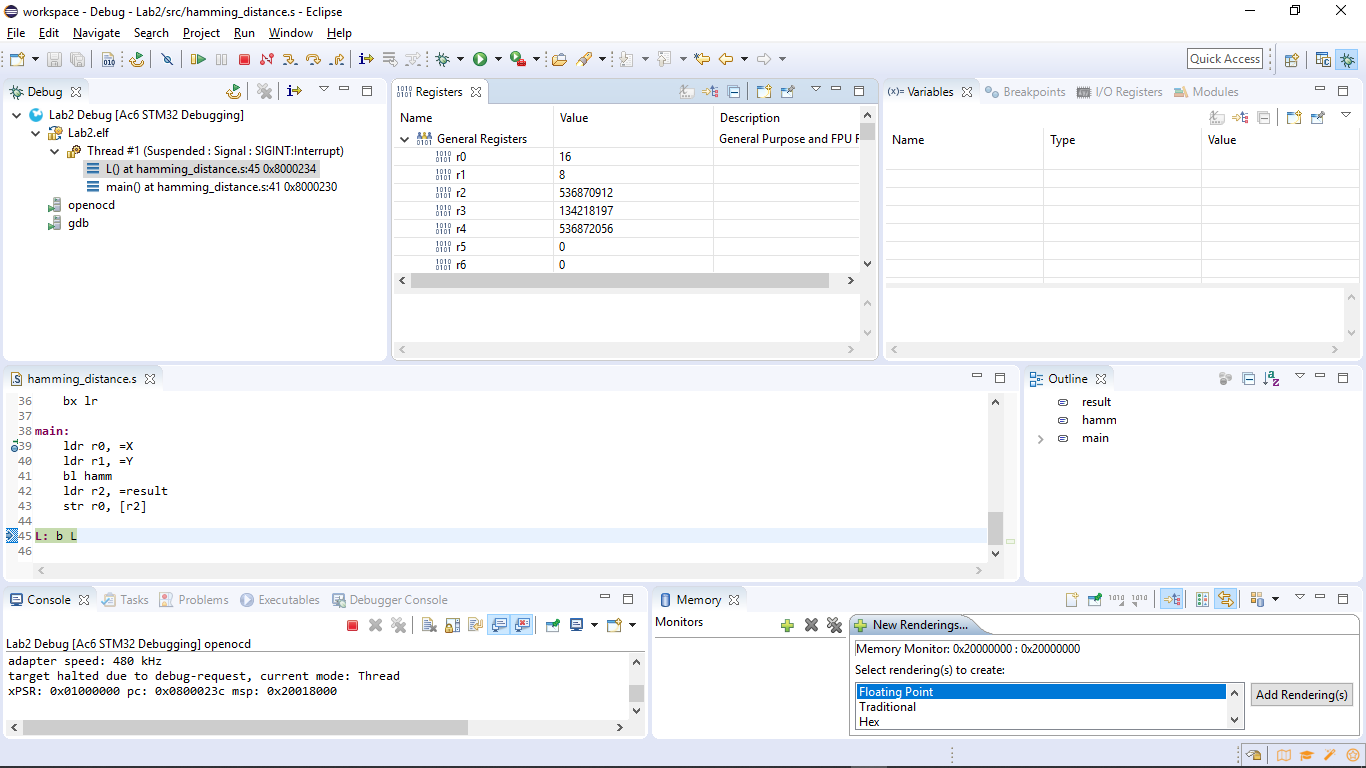


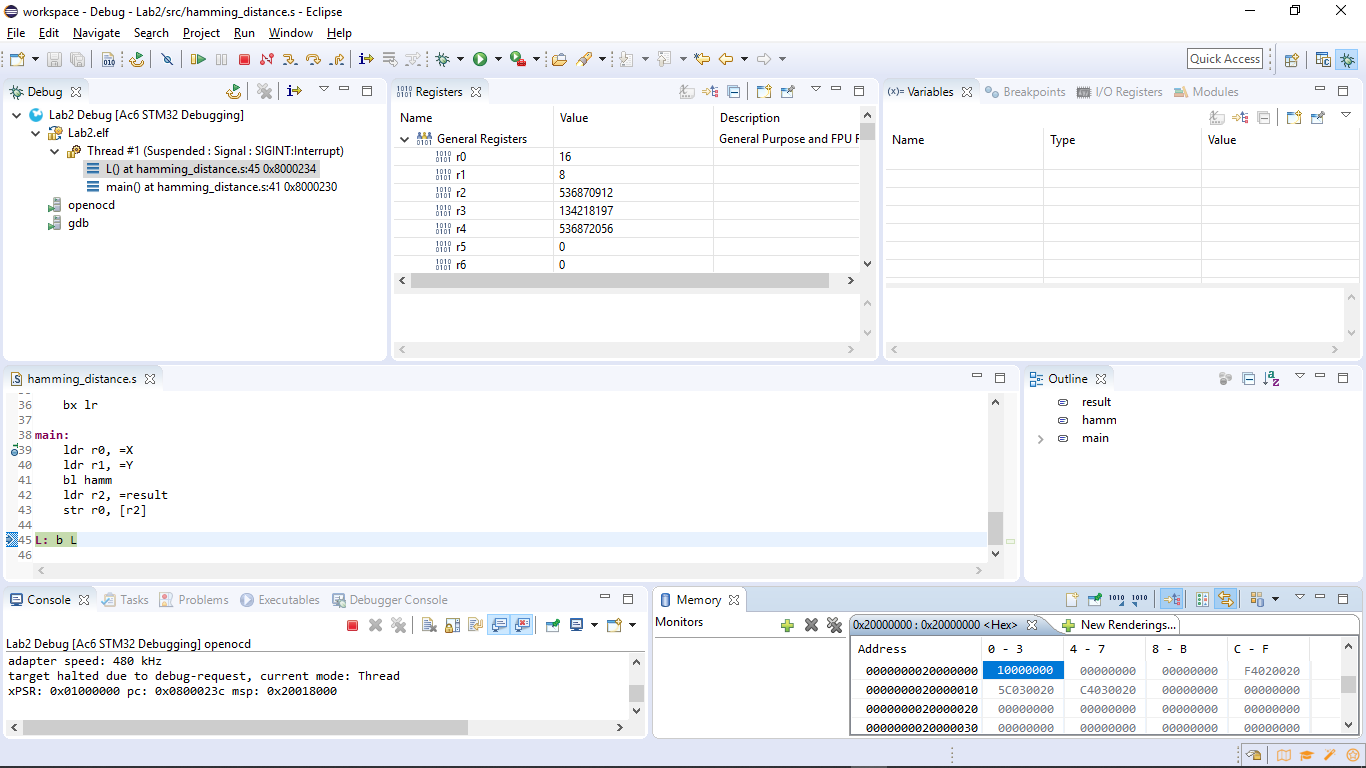
Bubble sort is an O(n2) algorithm with doubly-nested loop.

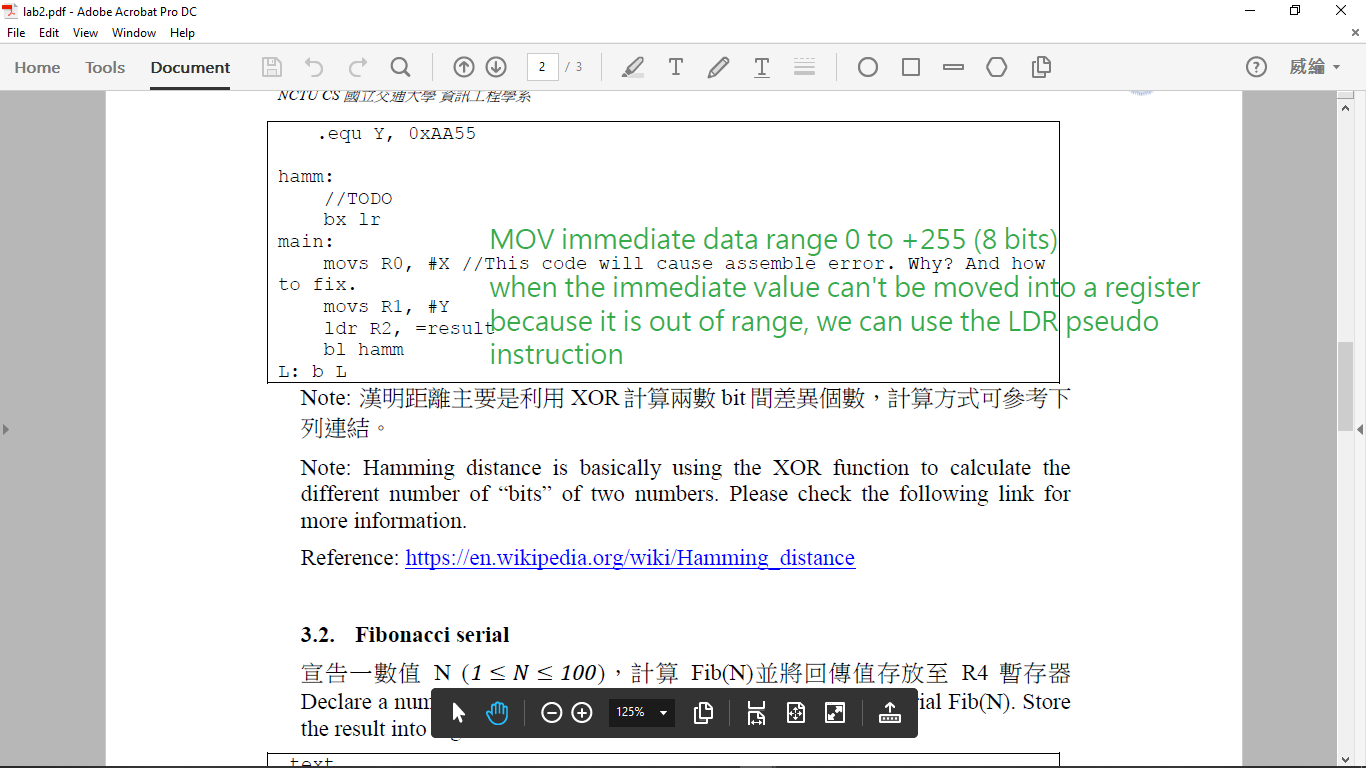
# Results and analysis 實驗結果與分析

* 1. **Hamming distance**

The result was stored from r0 to *result* in memory (0x20000000).

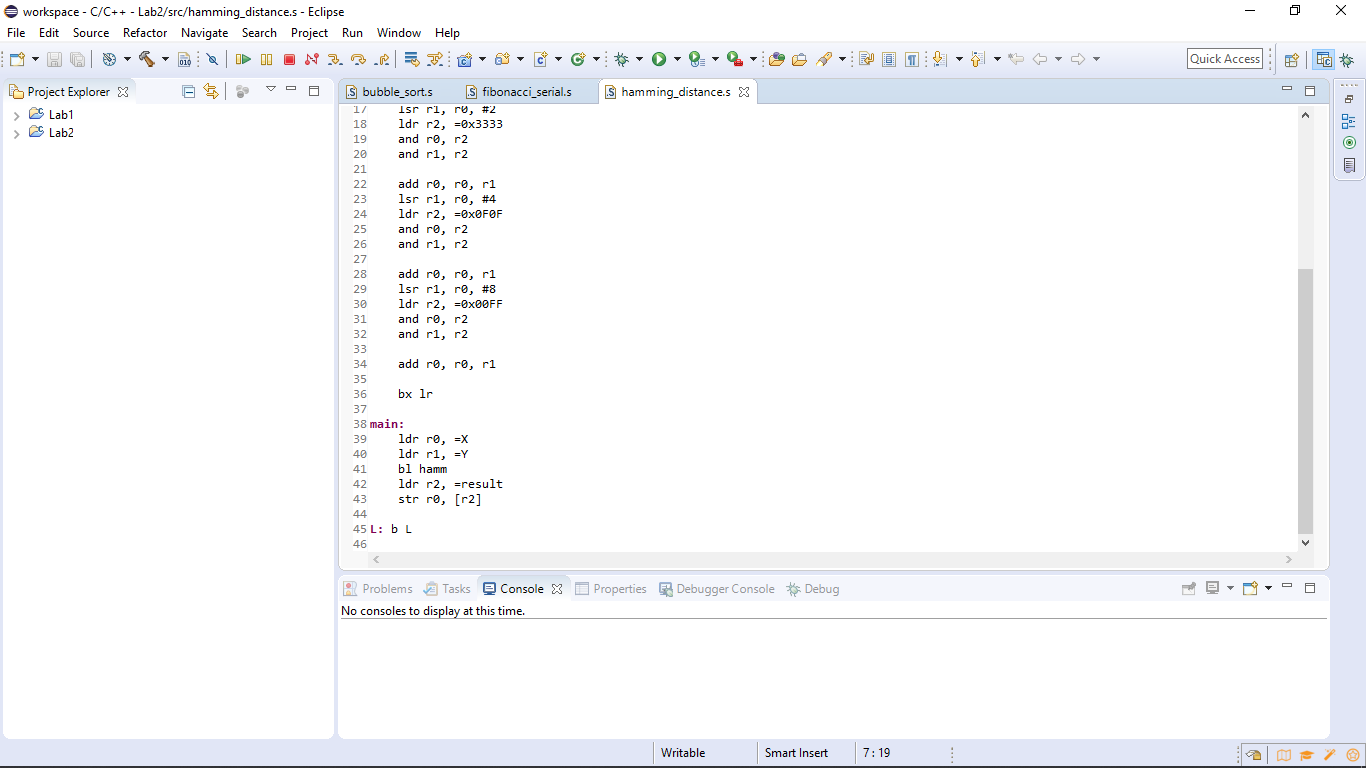






This code will cause assembler error because MOV immediate data range from 0 to +255 (8 bits).

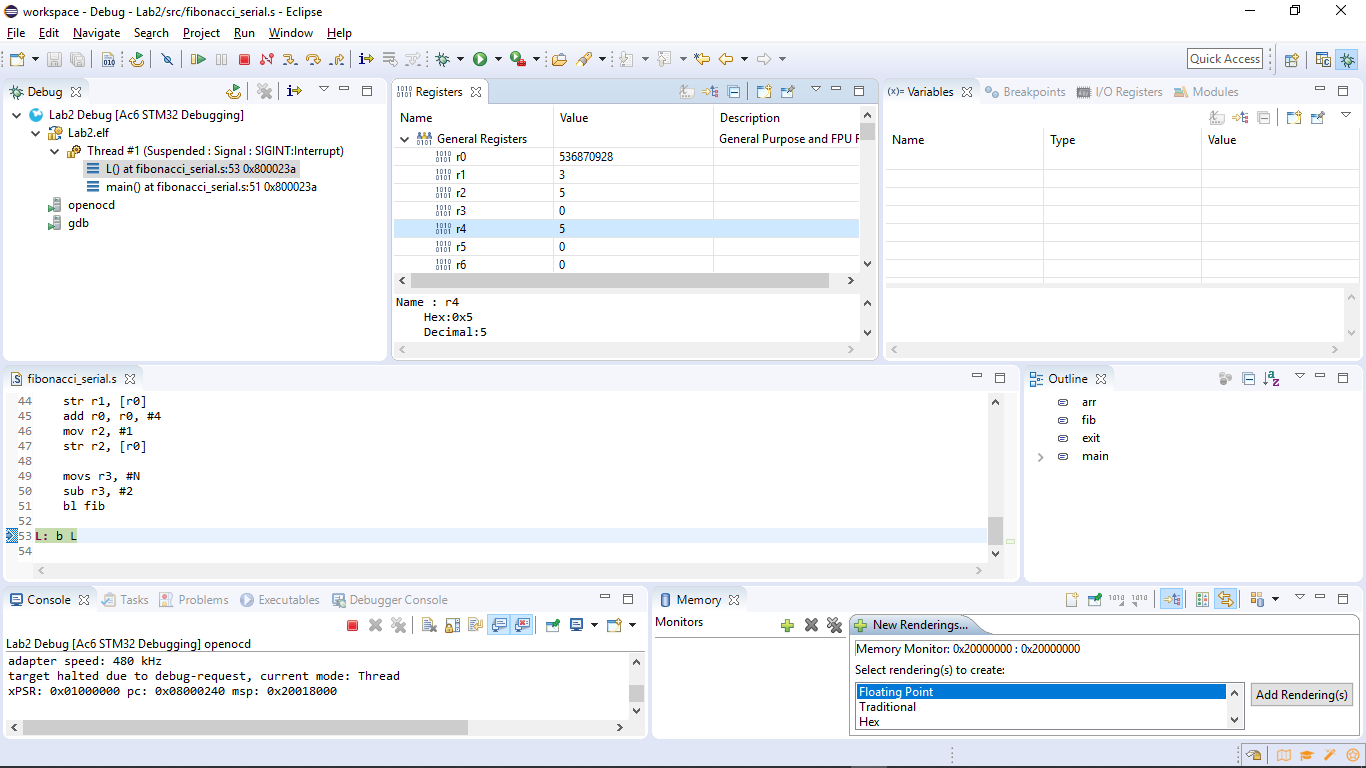
So we modified to this:



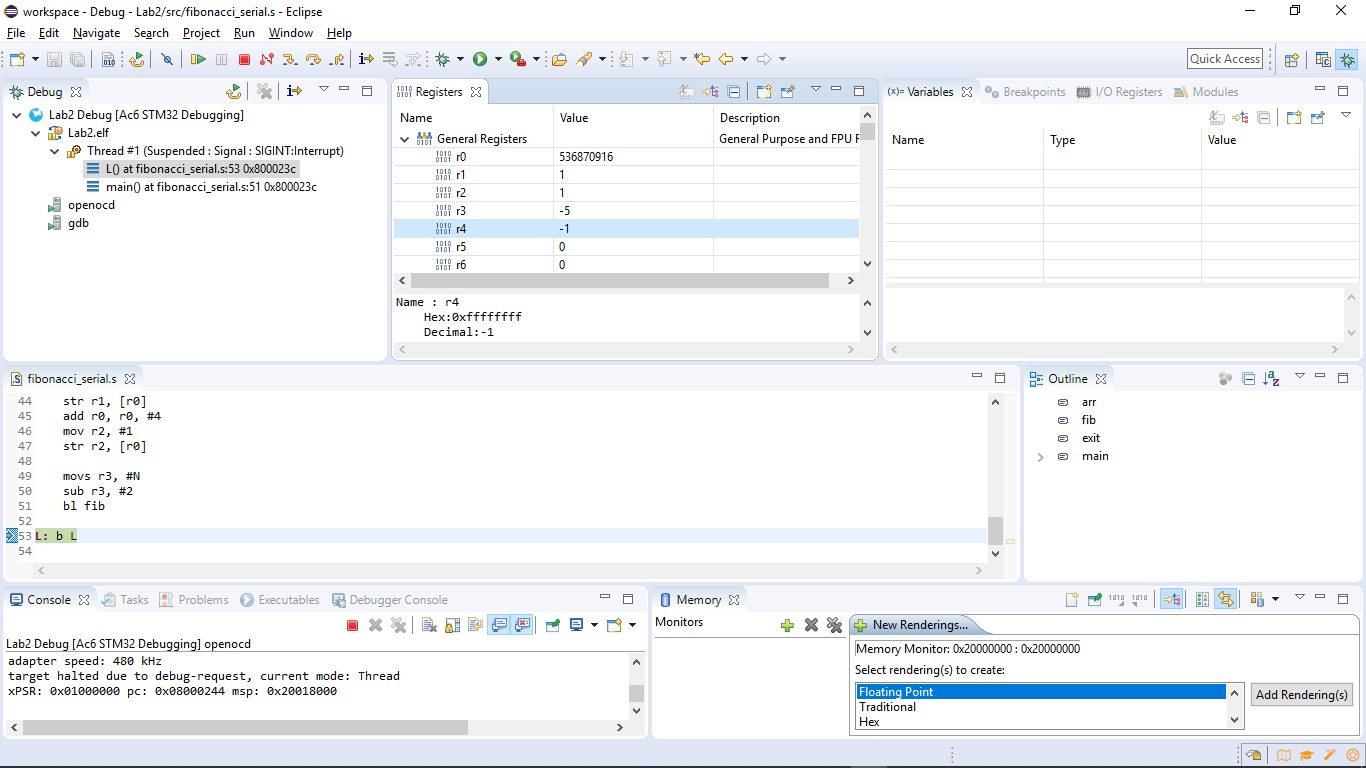
When the immediate value is too big, such as address, to be moved into a register, we can use the LDR pseudo instruction.

* 1. **Fibonacci serial**

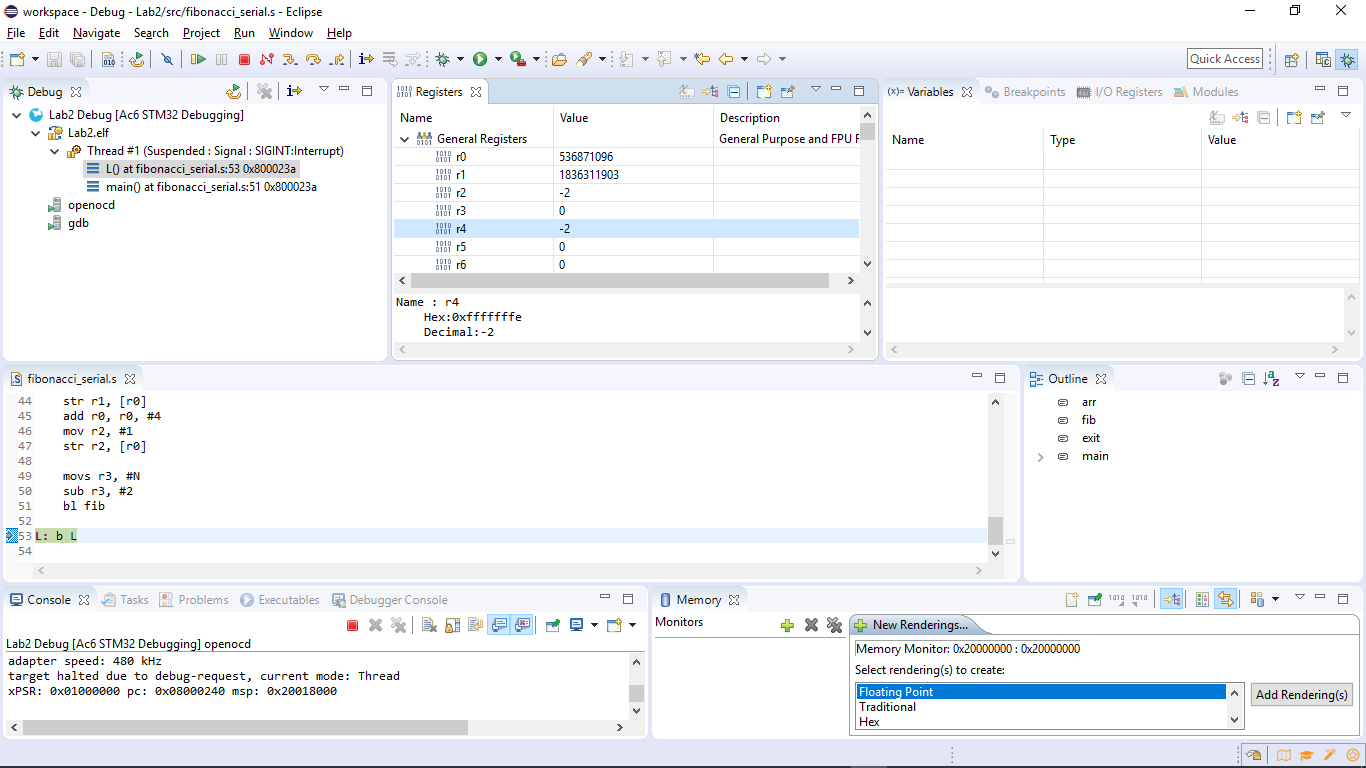
In the case of N = 5, the result was stored at r4 = 5.



N = -3, invalid number

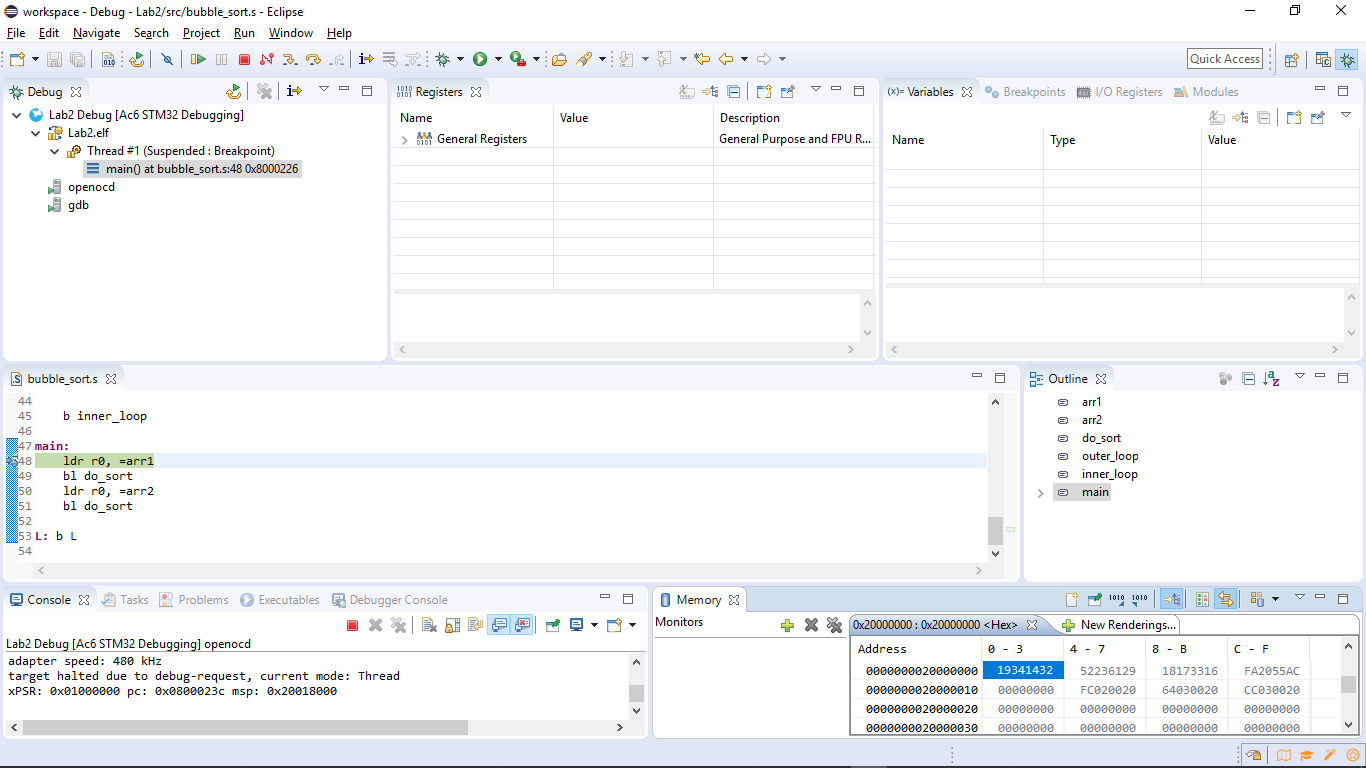


N = 47, overflow

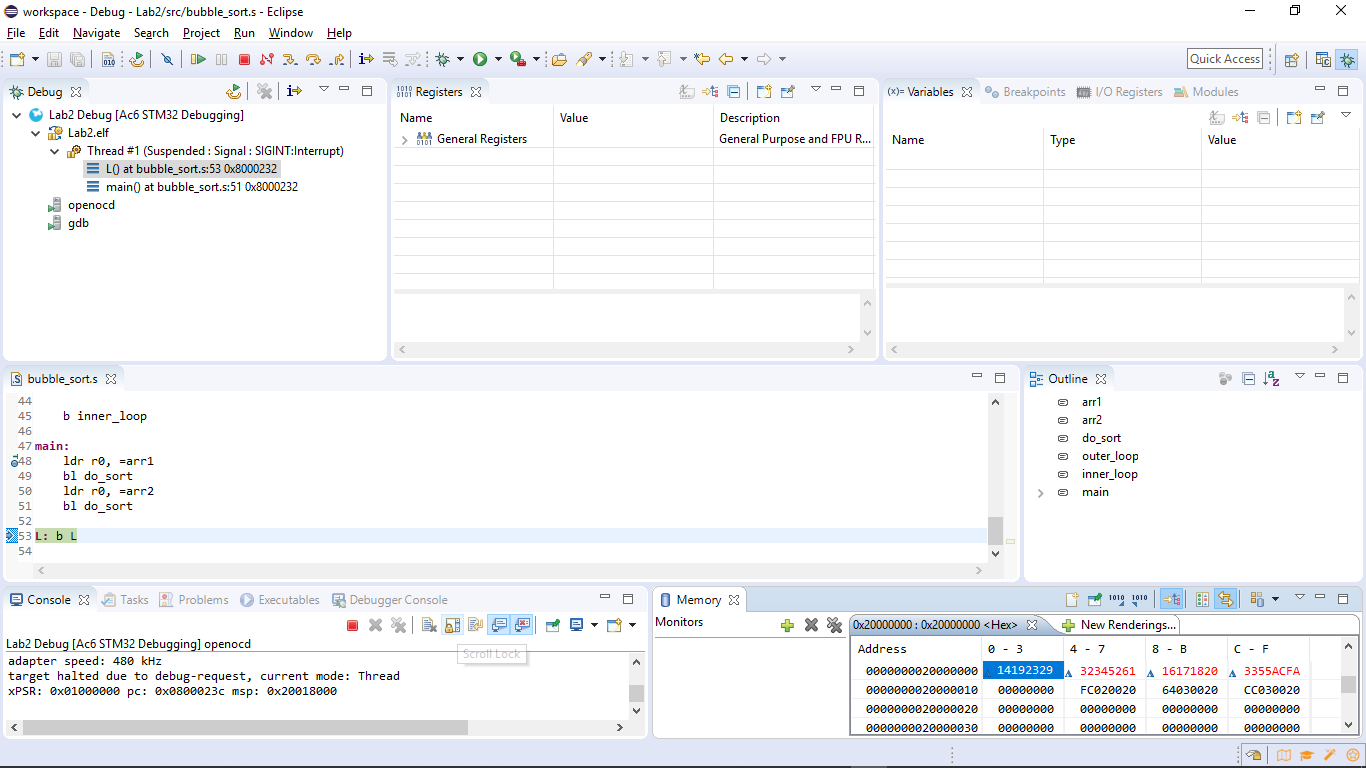


* 1. **Bubble sort**

Before:



After:



The ARM structure store value in little-endian way, so we use the <Hex> rendering way

1. **Conclusions and ideas 心得討論與應用聯想**

In last semester, I have once written MIPS for the CO assignment. It literally took me a week to finish 3 codes. With the experience, I have more basic concept of assembly language now. Although there are subtle differences between MIPS and ARM, especially xPSR, the assignment was still extremely helpful.