BLG 351E – Microcomputer Laboratory

Experiment 2

Introduction

The main purpose of the experiment is to gain experience in assembly programming of MC 6802. To this end, you are required to write two distinct programs and test your code by using ITU-Training Kit.

Part 1 – Memory Test

As the first part of the experiment, you need to write a program which checks whether a memory address functions properly or not. To do so, firstly, assume that the address of the memory (*M*) which is going to be tested is being stored at $4000 (-$4001). Then, you should try to write a random data to address M and try to read data the result after write operation in order to decide whether memory slot M functions properly or not. Finally, in order to state the result of the test, you should write 0x01 (true) or 0x00 (false) to the memory address $4002.

The basic flow of the program is given below.

```
Read variable M from the Memory[$4000]
Write $AA to the Memory[M]
Read Memory[M]
Compare the read data with $AA
IF they do NOT match              //it means test fails
      Write 0x00 to the Memory[$4002]
      Conclude the test
ENDIF
Write $55 to the Memory[M]        //test continues
Read Memory[M]
Compare the read data with $55
IF they do NOT match              //it means test fails
      Write 0x00 to the Memory[$4002]
      Conclude the test
ENDIF
Write 0x01 to the Memory[$4002]
Conclude the test                 //Memory M functions properly
```

Part 2 – Sorting Algorithm

In this part, you are required to write an assembly code which implements bubble sort algorithm. Pseudo code of bubble sort algorithm for an array (**A**) sized ***n*** is given below.

```
for i=0 to n;
      for j=0 to n-1;
            if(A[j] > A[j+1])
                  Swap A[j], A[j+1]
            endif
      endfor
endfor
```

> Your program should start from the address $4000. Assume that, base address Array A is stored at $10 (- $11), and the size of the Array A is stored at $12 before your program starts.

Report

Your report should contain your program code (with explanations) for Part 1 and Part 2.