BLG 351E – Microcomputer Laboratory

Experiment 1

Introduction

This lab aims to introduce the ITU–Training kit, MC6802 microprocessor and Motorola 6800 assembly language.

ITU–Training Kit

Components of the ITU–Training kit are given in Figure 1 and Table 1.

TABLE 1: HARDWARE COMPONENTS

CPU	MC6802
Memory	24K x 8 Read/Write 16K x 8 Read Only
Address Decoder	
Control Unit	
Displays and Keypad	
Parallel Port	
Serial Port	
Programmable Counter	

MC6802 Microprocessor

MC6802 is an 8-bit microprocessor with 4 MHz of operating frequency. It has 128 Byte on-chip memory mapped between addresses $0000 and $007F. It also supports external memories. In ITU–Training kit, 24 KB Read/Write and 16 KB Read Only Memories are implemented. Memory mapping is given in Table 2.

TABLE 2: MEMORY MAP

Operating System	$FFFF	–	$E000
Read Only Memory	$DFFF	–	$C000
Empty	$BFFF	–	$A000
Empty	$9FFF	–	$8000
Empty	$7FFF	–	$6000
Read/Write Memory	$5FFF	–	$4000
Read/Write Memory	$3FFF	–	$2000
Read/Write Memory	$1FFF	–	$0000

Motorola 6800 Assembly Language

Instructions and machine codes are provided as a separate document. In addition, HyperVision 6800IDE can be used as an emulator and to generate machine codes.

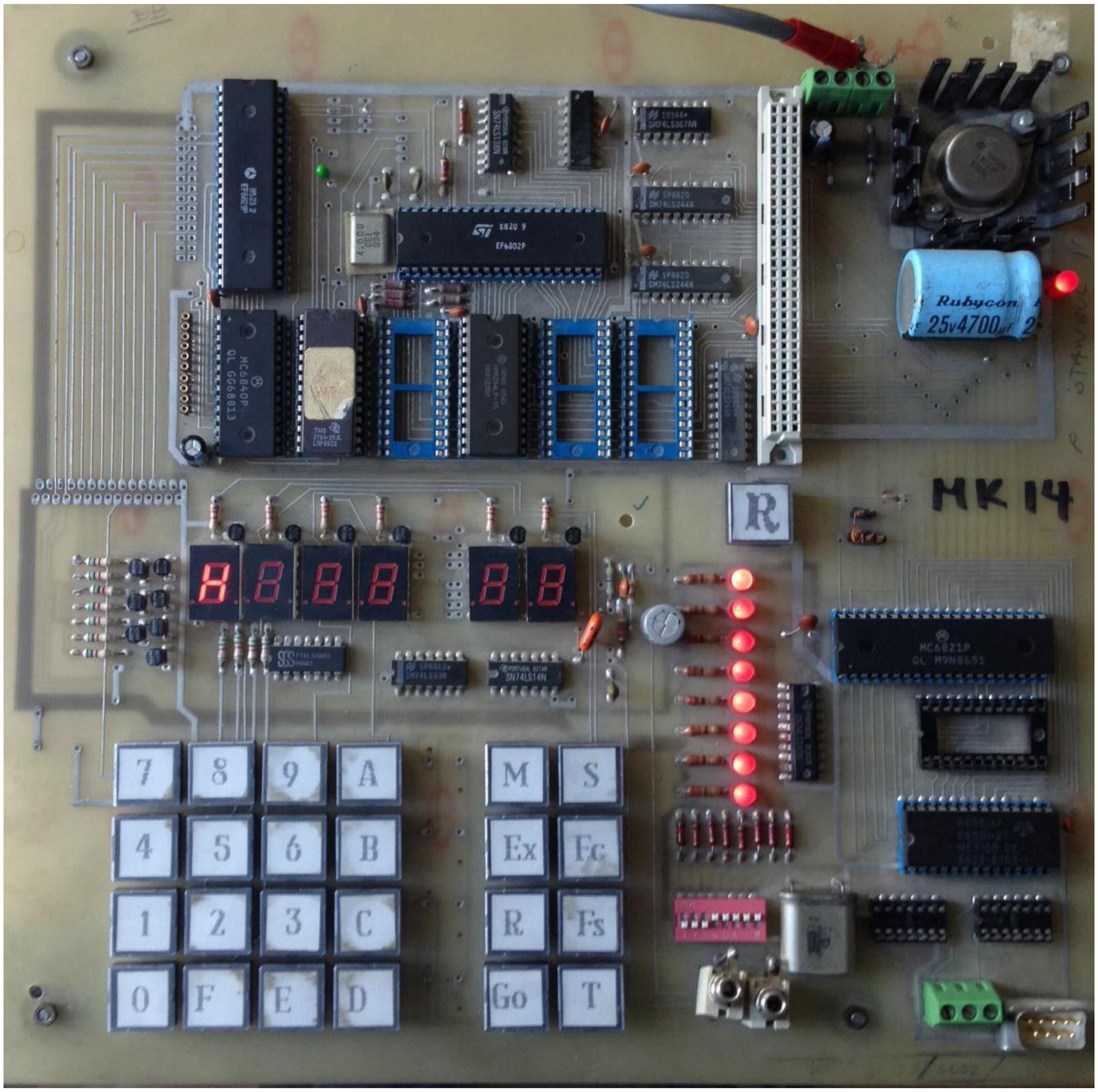

FIGURE 1: HARDWARE COMPONENTS

ITU–Training Kit Manual

Usage of the kit includes observing / editing memory, executing a program and debugging.

> Before each operation the kit should be in the ready state indicated by the letter **H** in the display. After each operation, press **Ex** key to return the ready state.

Observing and Editing Memory

To observe the contents of a memory address, enter the 16-bit address and press **M** key. The contents will be given in the 2-digit display. You can move to the next and previous addresses via **G** and **M** keys.

Program Execution

Simply enter the 16-bit address of the beginning of your program and press **G** key.

Debugging

Press **R** key to display Program Counter (PS) contents. Enter the beginning address and press **T** key to execute one instruction at a time. At each step you can observe the contents of all registers by pressing **M** and **G** keys.

MC6802 Addressing Modes

Inherent (Implied)

If the instruction is implied (e.g. INCA) then the contents of Accumulator A are incremented by 1 and the fetch-instruction action is complete. This instruction operates with the internal registers only.

Direct

If the instruction is direct (e.g. LDAA $42) then the next byte is fetched and 00 is placed as the High Byte of the address ((00)42) of the data to be acted upon. So it can only be used with addresses from $0000 to $00FF.

Extended

If the instruction is extended (e.g. LDAA $C300) then the next two bytes are fetched at the address ($C300) and the data to be acted upon are the contents of this address. So the contents of address ($C300) will be placed into Accumulator A.

Immediate

If the instruction is immediate (e.g. LDAA #$02) then the next byte is fetched and treated as data, in this example 2 is placed into the Accumulator A.

Indexed

If the instruction is indexed (e.g. ADDA 0,x) then the contents of the Index register are added to the Displacement byte ($00 to $FF) and this forms a pointer to the contents of the data to be acted upon.

Relative

If the instruction is relative then the next byte is fetched and treated as an offset to the PC during the following execution. This offset is a Signed Number and allows the program to branch forward up to +127 steps and backwards to a maximum of -128 steps.

Part 1

Run the following program on the kit. Explain the purpose of the program. Run the program step by step to examine register contents after each instruction.

Address	**Content**		**Instruction**
4000	4F		CLRA
4001	5F		CLRB
4002	CE 44 00		LDX #$4400
4005	AB 00	Label	ADDA 0,X
4007	08		INX
4008	5C		INCB
4009	C1 0A		CMPB #$0A
400B	2D F8		BLT Label
400D	B7 44 40		STAA $4440
4010	3F		SWI

Part 2

Write the assembly code for the algorithm below, convert it to machine code and execute on the kit.

> To test for evenness simply check the least significant bit. If it is equal to 0 then the number is even, otherwise odd.

```
j=0;
k=0;
for(i = 0; i < 10; i++){
      if(A[i] % 2 == 0)
            Even[j++] = A[i];
      else
            Odd[k++] = A[i];
```

> Your program should start from the address $4000. Array A starts from $4100. Array Even and Odd start from $4200 and $4300, respectively.

Report

Your report should contain your observations for Part 1 (as a table of register contents at each step) and your program code (with explanations) for Part 2.