**İ.T.Ü.**

**Faculty of Computer and Informatics**

**Computer Engineering**



**MICROCOMPUTER LAB**

**REPORT**

|  |  |  |
| --- | --- | --- |
| **Lab No** | **:** 01 | |
| **Lab Date** | **:** 26.09.2013 | |
| **Group** | **:** B9 | |
| **Group Members** | **:**  040100014 Teoman TURAN  040100018 Mustafa DURMUŞ  040100117 Tuğrul YATAĞAN  040100124 Emre GÖKREM | |
|  |  | |
|  |  | |
|  |  | |
| **Research Assistant :** Hasan ÜNLÜ | | |  |
|  | |  |

**1. THE AIM/CONTENT of THE EXPERIMENT**

The purpose of doing this experiment is introducing us ITU-Training kit and receiving information about how to program MC6802 which is the microprocessor used in ITU-Training Kit. Also, the programming language the kit uses is Motorola 6800 Assembly language. Programs are written by machine codes and run via the kit.

**2. EQUIPMENT**

Only ITU-Training Kit has been used in the experiment. This kit consists of the following hardware components:

* CPU: MC6802
* Memory: 24K\*8 R/W + 16K\*8 Read Only
* Address decoder
* Control unit
* Display and Keypad
* Parallel Port
* Serial Port
* Programmable counter

**3. THE PROGRAM for THE EXPERIMENT**

The code below has been run on ITU-Training Kit.

4000 4F clra

4001 5F clrb

4002 CE 44 00 ldx #$4400

4005 AB 00 label adda 0, x

4007 08 inx

4009 C1 0A cmpb #$0A

400B 2D F8 blt label

400D B7 44 40 staa $4440

4010 3F swi

The left-most column shows Program Counter (PC). The one next to it contains machine codes for the instructions which are listed on the right-most column. The third column contains the word “label” pointing to the line being branched.

What these instructions mean can be found on the datasheet for Motorola MC6802 Microprocessor, provided along with the experiment page, or in the “Reference” tab of SDK6800 Simulator software of the microprocessor. However, here are the explanations for the instructions being used in the program.

**clra** Clears the content of Accumulator A

**clrb**  Clears the content of Accumulator B

**ldx #$4400** “ldx” loads a value to Index Register. Here, this instruction loads the address value of $4400 to Index Register.

**adda 0,x** “adda” adds the content of Accumulator A to that of another register and loads the result to Accumulator A. “0” is the offset value with respect to the address located in x.

**inx** Increases the value located in Index Register

**incb** Increases the value located in Accumulator B

**cmpb #$0A** “cmpb” compares the value located in Accumulator B with another value. Here, the content of Accumulator B is compared with $0A.

**blt label** According to the comparison; if the content of Accumulator B is equal to $0A, the program skips this line and keeps running. If not, it branches to the line tagged with label and continues from there. This loop ends when the result of that comparison becomes equality.

**staa $4440** “staa” stores the content of Accumulator A to a memory address. Here, it is stored to $4440.

**swi** Ends the program

**What this program does:** Starting from the address of $4400, this program sums 10 numbers from 1 to 10 and stores the result into the address of $4440.

**4000 4F clra**

// The content of Accumulator A (A) = 0

// Program counter (PC) = 4001

// Status Register (SR) = 000100

// Stack Pointer (SP) = F000

// The content of Accumulator B (B) does not change.

**4001 5F clrb**

// (B) = 0

// PC = 4002

// SR = 000100

// SP = F000 (There is no push-pop operation.)

// (A) = 0 (It does not change.)

**4002 CE 44 00 ldx #$4400**

// Index Register = $4400

// PC = 4005

// (A) = 0 (It does not change.)

// (B) = 0 (It does not change.)

// SR = 000000

// SP = F000 (There is no push-pop operation.)

**4005 AB 00 label adda 0, x**

// OFFSET CALCULATION: x + 0 = $4400 + x = $4400

// (A) = 1

// (B) = 0 (It does not change.)

// PC = 4007

// SR = 000000

// SP = F000 (There is no push-pop operation.)

**4007 08 inx**

// Index Register = $4401

// PC = 4008

// (A) = 1 (It does not change.)

// (B) = 0 (It does not change.)

// SR = 000000

// SP = F000 (There is no push-pop operation.)

**4008 5C incb**

// SR = 001001

// PC = 400B

// (A) = 1 (It does not change.)

// (B) = 0 (It does not change.)

// SP = F000 (There is no push-pop operation.)

**4009 C1 0A cmpb #$0A**

// PC = 4007

// SR = 001001

// (A) = 1 (It does not change.)

// (B) = 0 (It does not change.)

// SP = F000 (There is no push-pop operation.)

// Until (B) has been equal to $0A, this loop works.

**400B 2D F8 blt label**

// (A) = $37

// (B) = $0A

// PC = 400D

// SR = 100100

// SP = F000 (There is no push-pop operation.)

**400D B7 44 40 staa $4440**

// The content of $4440 = $37

// (A) = $37 (It does not change.)

// (B) = $0A (It does not change.)

// PC = 4010

// SR = 100000

// SP = F000 (There is no push-pop operation.)

**4010 3F swi**

// PC = 4000

// (A) = $37 (It does not change.)

// (B) = $0A (It does not change.)

// SR = 110000

// SP = EFF9

Here are two screenshots demonstrating the result from SDK6800 Emulator v1.08 simulation software of the microprocessor:



