**İ.T.Ü.**

**Faculty of Computer and Informatics**

**Computer Engineering**



**MICROCOMPUTER LAB**

**REPORT**

|  |  |  |
| --- | --- | --- |
| **Lab No** | **:** 02 | |
| **Lab Date** | **:** 03.10.2013 | |
| **Group** | **:** B9 | |
| **Group Members** | **:** 040100014 Teoman TURAN  040100018 Mustafa DURMUŞ  040100117 Tuğrul YATAĞAN  040100124 Emre GÖKREM | |
|  |  | |
|  |  | |
|  |  | |
| **Research Assistant :** Salih Serdar GÜÇLÜ | | |  |
| **1. THE AIM/CONTENT of THE EXPERIMENT** | |  |

The purpose of doing this experiment is to examine addressing modes (methods) of MC6802 within a program written in Motorola 6800 Assembly programming language: immediate, direct, inherent, extended, relative.

**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

In addition to this kit, for observing the simulation results during the experiment, one of our group members has used his own computer to run the simulation software of the microprocessor.

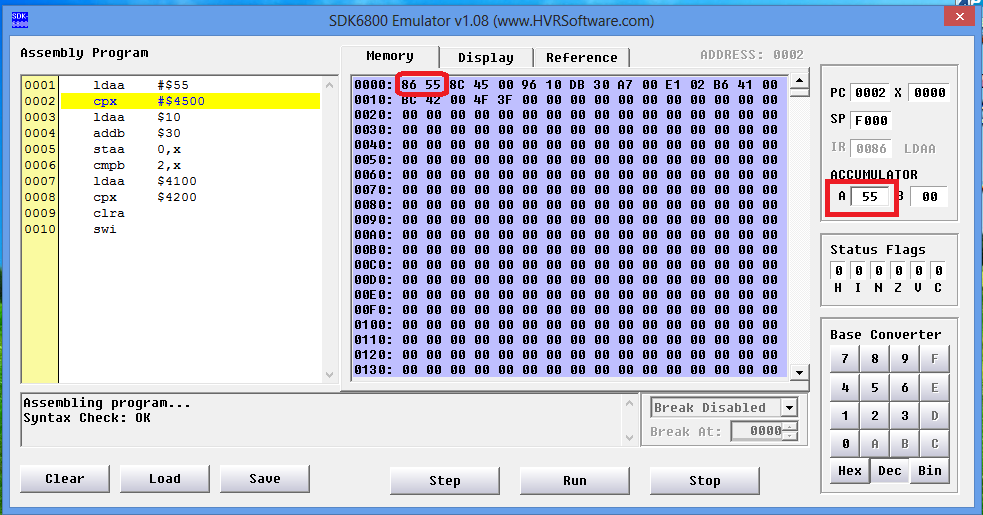
**3. THE PROGRAMS for THE EXPERIMENT**

**3.1. SECTION 2: ADDRESSING MODES**

The machine codes of the instructions in the code fragment below and their addressing modes had been determined before coming to the lab. Also, that code fragment had been written and run on the simulator software of the microprocessor. In the lab, the code has been run on ITU-Training Kit. Here are the machine codes of the instructions, their addressing modes, what they do and screenshots for each of them from the simulation software of the microprocessor:

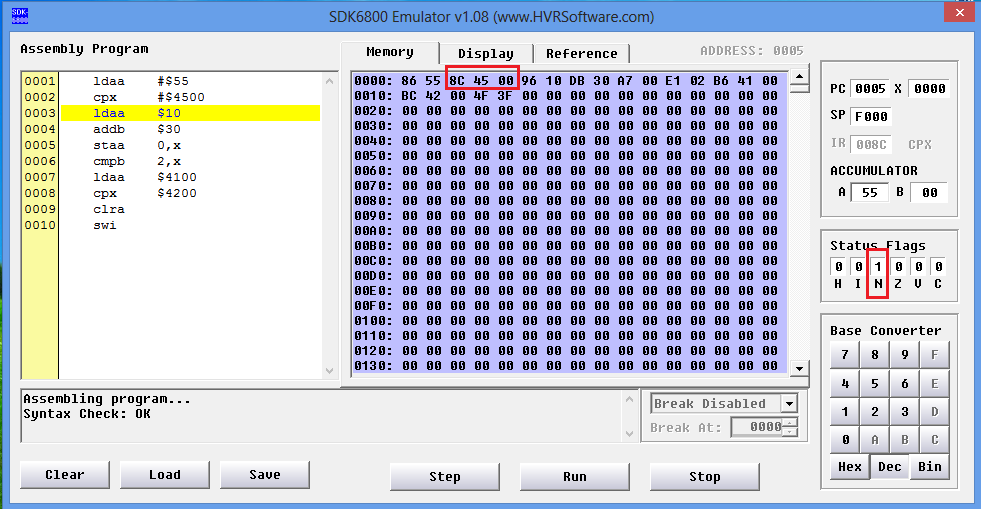
**LDAA #$55 86 55 Immediate**

// It immediately loads direct data $55 to Accumulator A.



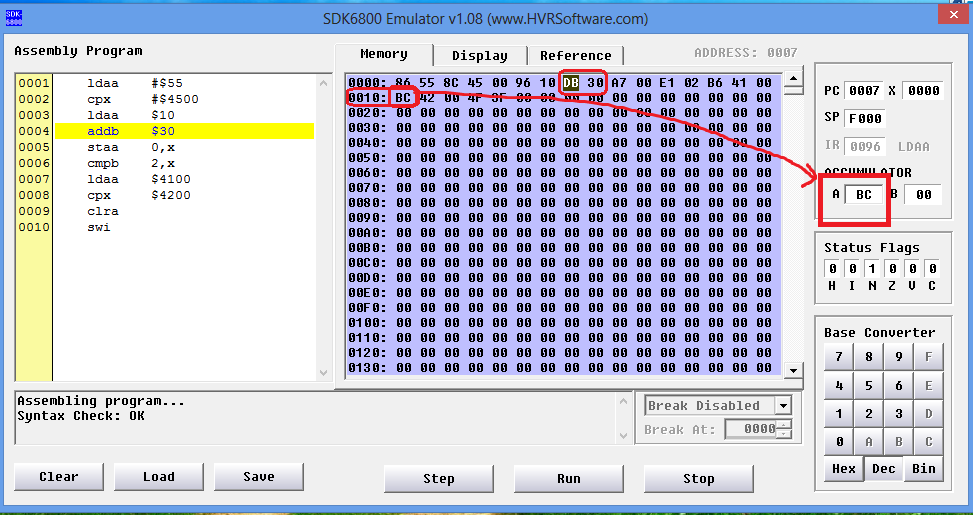
**CPX #$4500 8C 45 00 Immediate**

// It compares the value in the memory address kept in Index Register with another direct data, $4500.



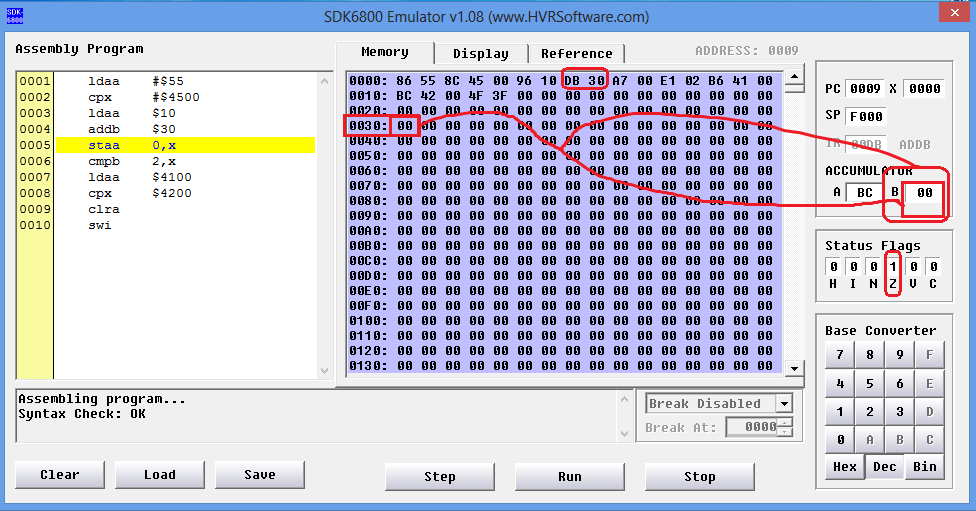
**LDAA $10 96 10 Direct**

// It loads the value in memory address $10 to the Accumulator A.



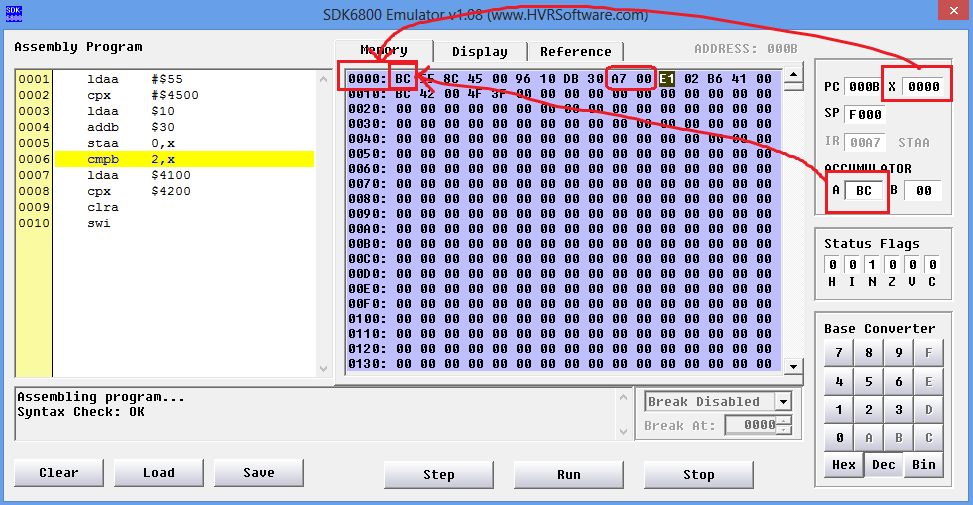
**ADDB $30 DB 30 Direct**

// It sums the value in memory address $30 and the one in Accumulator B, then stores the result in Accumulator B.



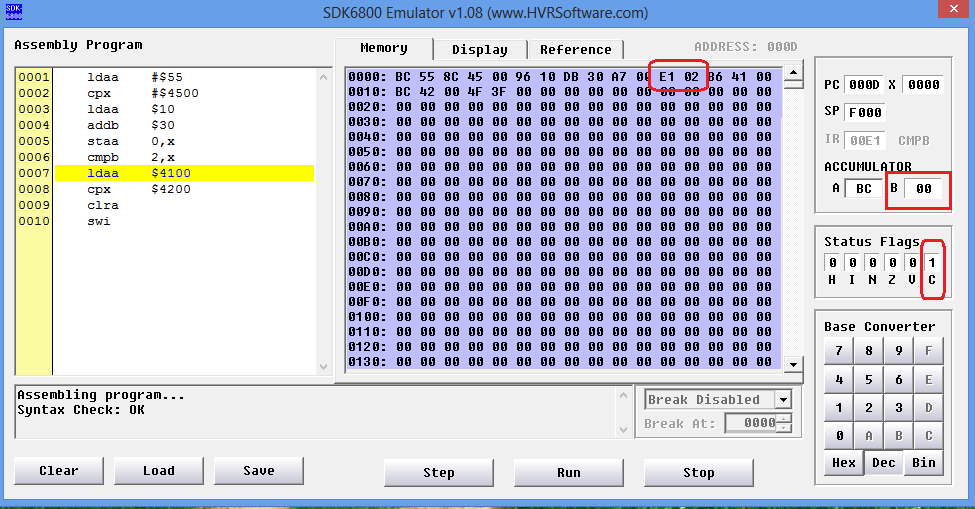
**STAA 0,x A7 00 Indexed**

// It stores the value in Accumulator A to the address kept in Index Register with the index of 0. So, there is no offset change.



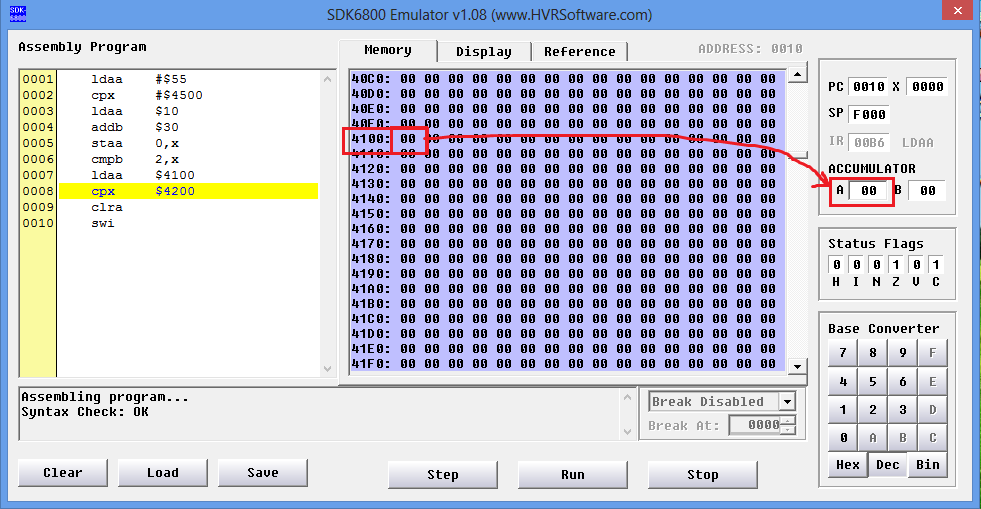
**CMPB 2,x E1 02 Indexed**

// It compares the value in Accumulator B with the value in the address with the index of 2 with respect to the one kept in Index Register.



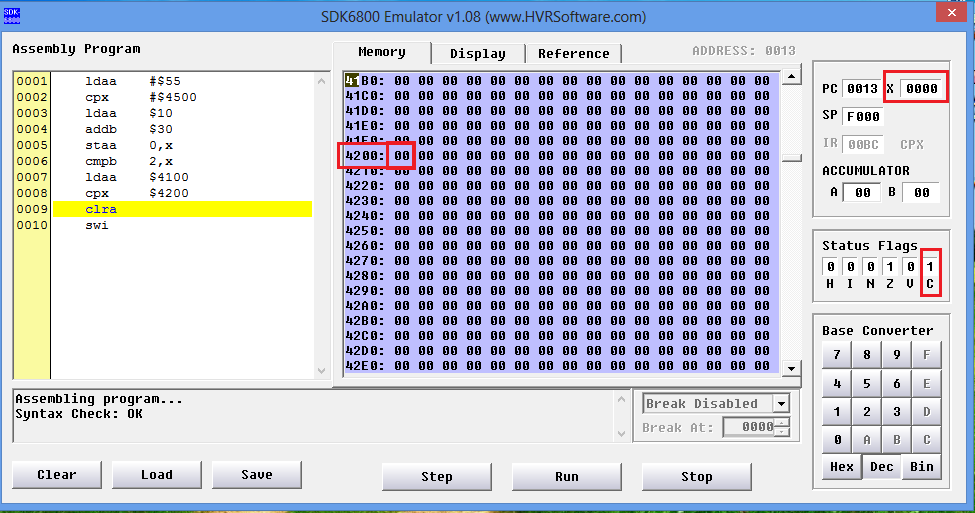
**LDAA $4100 B6 41 00 Extended**

// It loads the value in memory address $4100 to Accumulator A.



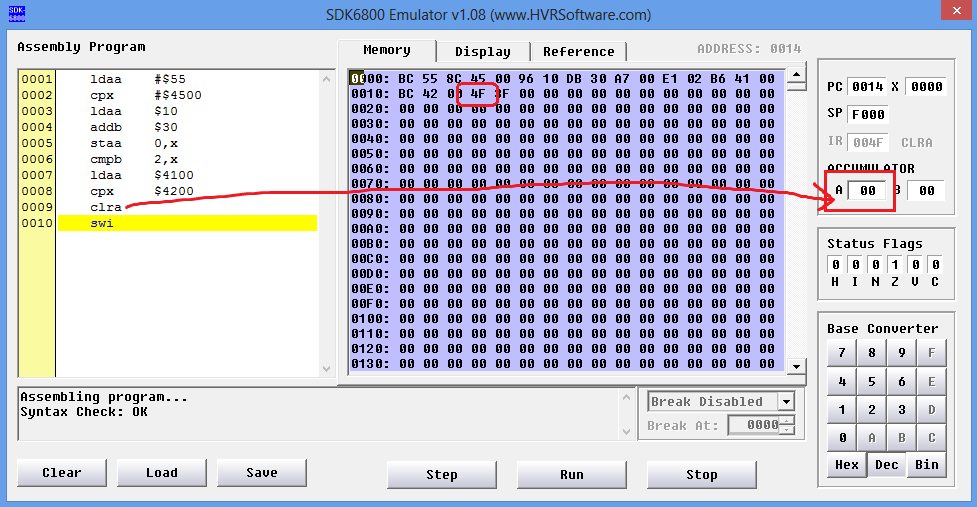
**CPX $4200 BC 42 00 Extended**

// It compares the value in memory address $4200 with the value in the memory address kept in Index Register.



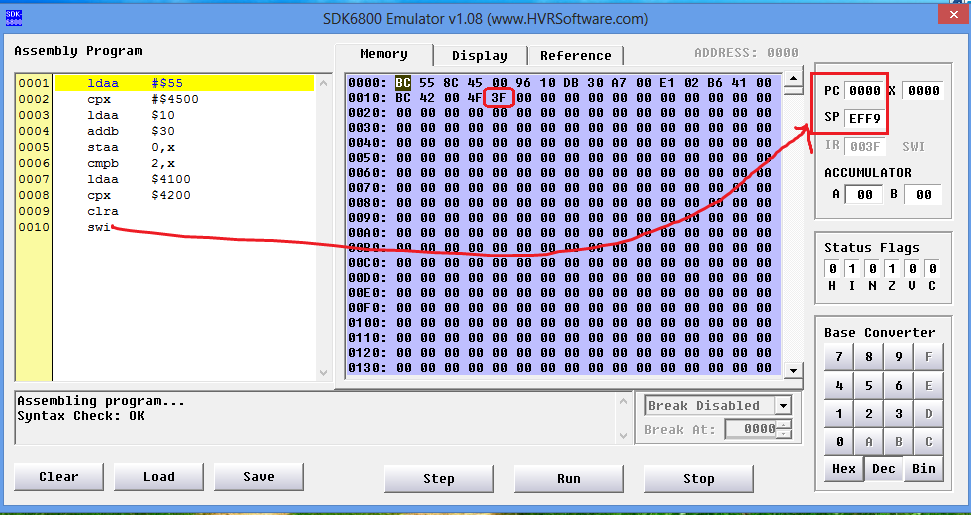
**CLRA 4F Inherent**

// This is an instruction taking no operand. It clears Accumulator A, in other words, makes the content of Accumulator A is zero.



**SWI 3F Inherent**

// This is an instruction taking no operand too. It terminates the program.



**3.2. SECTION 3: EXPERIMENT**

Here, a program which separates odd and even numbers of Array A that is the main array has been written. This program starts from the address $4000 and applies the following algorithm:

Beginning address of Array A: $4100

Beginning address of the new array for even numbers: $4200

Beginning address of the new array for odd numbers: $4300

j=0;

k=0;

for(i=0; i<n; i++)

{

if(A[i] % 2 == 0)

Even[j++] = A[i];

else

Odd[k++] = A[i];

}

If least significant bit of a binary number is 1, it is odd. Otherwise, it is even. The program has been written by applying the algorithm for 10 numbers. It has been converted into machine codes and run on ITU-Training Kit. Here is the code of the program in Motorola 6800 Assembly programming language:

4F CLRA

5F CLRB

CE 42 00 LDX #$4200

FF 44 00 STX $4400

CE 43 00 LDX #$4300

FF 45 00 STX $4500

CE 41 00 LDX #$4100

FF 46 00 STX $4600

A6 00 LABEL LDAA 0,x

08 INX

FF 46 00 STX $4600

46 RORA

25 11 BCS TEK

49 ROLA

FE 44 00 LDX $4400

A7 00 STAA 0,x

7C 44 INC $4401

5C INCB

FE 46 00 LDX $4600

C1 0A CMPB #$0A

2D E6 BLT LABEL

49 TEK ROLA

FE 45 00 LDX $4500

A7 00 STAA 0,x

7C 45 01 INC $4501

45 INCB

FE 46 00 LDX $4600

C1 0A CMPB #$0A

2D D5 BLT LABEL

.ORG $4100

.BYTE 1,2,3,4,5,6,7,8,9,10

The left-most column contains the machine codes for the instructions that are contained on the right-most column. The middle column contains the label worlds “TEK” and “LABEL” pointing to the lines being branched.

Here are screenshots demonstrating the result from the simulation software of the microprocessor:

