Khalifa University of Science, Technology and Research

Electronic Engineering Department

Module Name: Microprocessor Systems Laboratory
Module Code: ELCE333

Pre-Laboratory Experiment No. 1

Microcontroller Assembly Program Development

Laboratory Partners

Mohammad Jassem 100037002

Khalid Alzarouni 100035499

Khalid Alhosani 100037003

Date pre-lab submitted: 28-Jan-2015

Laboratory Instructor: Dr. Mohammed Al Zaabi

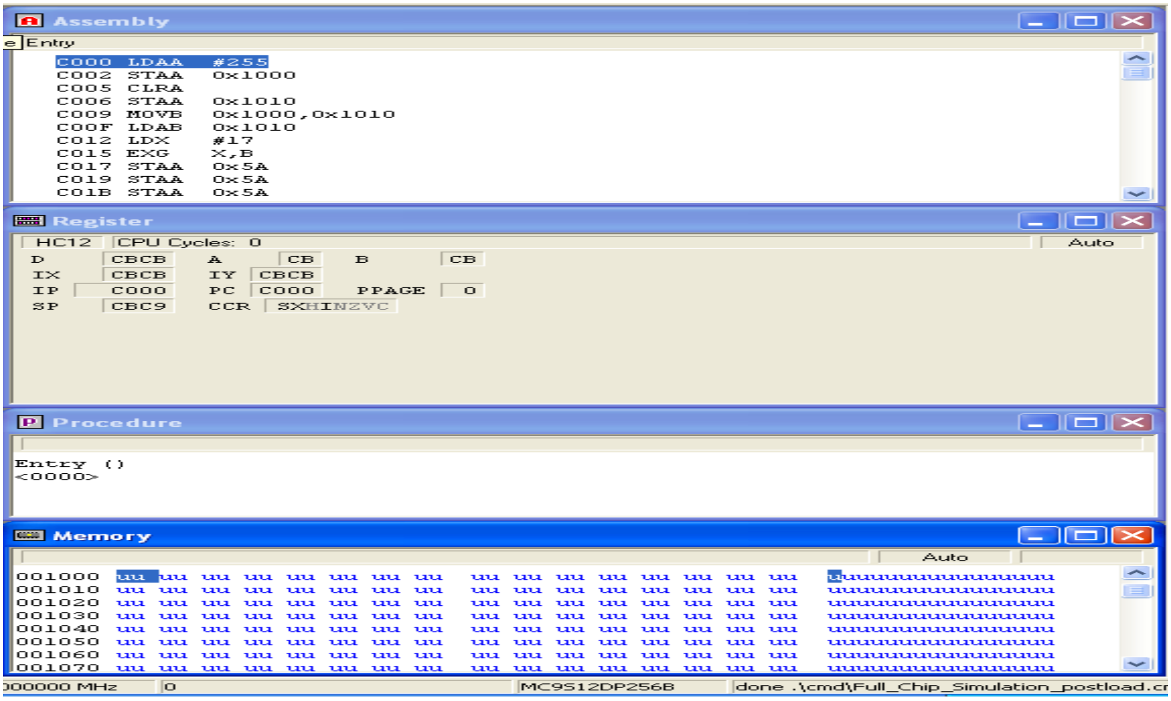
Dr. Mahmoud Khonji

Spring 2015

1.3)

2.

The figures bellow shows the process in steps as we move through each line. As shown in Figure 1 Acc. D contain the content of Acc. “A” and Acc. “B”, and the values are presented in hex. Also, there is a register called “IP” or “PC” which contain instruction pointer that hold the address of the next instruction to be executed, so in our case it is “C000”. At the beginning of the program, Acc. “A” and Acc. “B” have random values as shown in figure 1.



```
C000 LDAA   #255
C002 STAA   0x1000
C005 CLRA
C006 STAA   0x1010
C009 MOVB   0x1000,0x1010
C00F LDAB   0x1010
C012 LDX    #17
C015 EXG    X,B
C017 STAA   0x5A
C019 STAA   0x5A
C01B STAA   0x5A
```

```
HC12   CPU Cycles:  0
D      CBCB    A     CB     B       CB
IX     CBCB    IY    CBCB
IP     C000    PC    C000   PPAGE   0
SP     CBC9    CCR   SXHINZVC
```

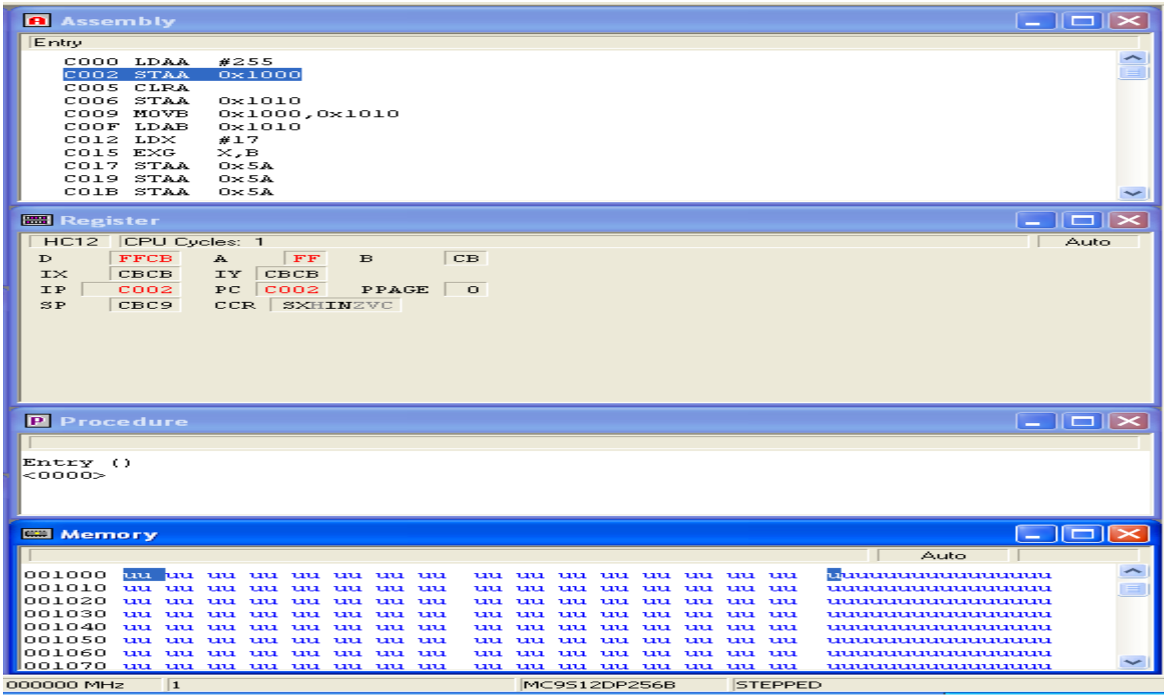
```
Entry ()
<0000>
```

```
001000 uu uu uu uu uu uu uu uu  uu uu uu uu uu uu uu uu  uuuuuuuuuuuuuuuu
001010 uu uu uu uu uu uu uu uu  uu uu uu uu uu uu uu uu  uuuuuuuuuuuuuuuu
001020 uu uu uu uu uu uu uu uu  uu uu uu uu uu uu uu uu  uuuuuuuuuuuuuuuu
001030 uu uu uu uu uu uu uu uu  uu uu uu uu uu uu uu uu  uuuuuuuuuuuuuuuu
001040 uu uu uu uu uu uu uu uu  uu uu uu uu uu uu uu uu  uuuuuuuuuuuuuuuu
001050 uu uu uu uu uu uu uu uu  uu uu uu uu uu uu uu uu  uuuuuuuuuuuuuuuu
001060 uu uu uu uu uu uu uu uu  uu uu uu uu uu uu uu uu  uuuuuuuuuuuuuuuu
001070 uu uu uu uu uu uu uu uu  uu uu uu uu uu uu uu uu  uuuuuuuuuuuuuuuu
```

```
000000 MHz   0   MC9S12DP256B   done .\cmd\Full_Chip_Simulation_postload.cr
```

Figure 1: Random values in the accumulator before starting the program

Once the program started and we moved to the first line, which is loading the value 0xFF to Acc. “A”. We noticed that Acc. “D” value is changed to “FFCB” since it contains the value of Acc. “A” and Acc. “B”. The IP changed to “C002”.



```
Entry
  C000 LDAA   #255
  C002 STAA   0x1000
  C005 CLRA
  C006 STAA   0x1010
  C009 MOVB   0x1000,0x1010
  C00F LDAB   0x1010
  C012 LDX    #17
  C015 EXG    X,B
  C017 STAA   0x5A
  C019 STAA   0x5A
  C01B STAA   0x5A
```

```
HC12   CPU Cycles: 1                                   Auto
D    FFCB     A    FF      B      CB
IX   CBCB     IY   CBCB
IP   C002     PC   C002    PPAGE  0
SP   CBC9     CCR  SXHINZVC
```

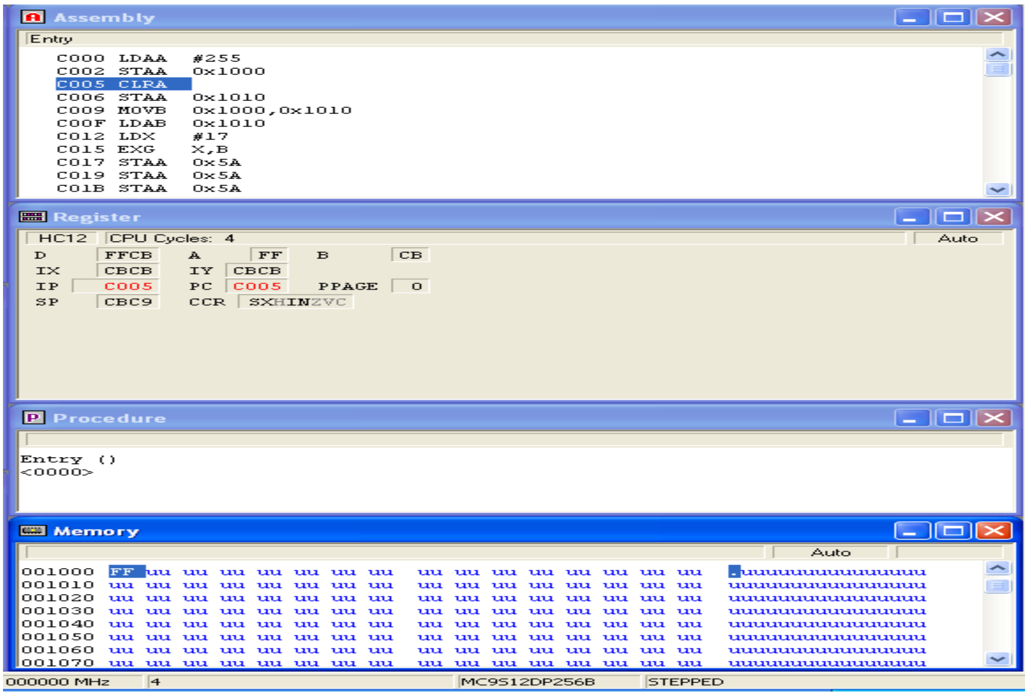
```
Entry ()
<0000>
```

```
                                                                  Auto
001000 uu uu uu uu uu uu uu uu   uu uu uu uu uu uu uu uu   uuuuuuuuuuuuuuuu
001010 uu uu uu uu uu uu uu uu   uu uu uu uu uu uu uu uu   uuuuuuuuuuuuuuuu
001020 uu uu uu uu uu uu uu uu   uu uu uu uu uu uu uu uu   uuuuuuuuuuuuuuuu
001030 uu uu uu uu uu uu uu uu   uu uu uu uu uu uu uu uu   uuuuuuuuuuuuuuuu
001040 uu uu uu uu uu uu uu uu   uu uu uu uu uu uu uu uu   uuuuuuuuuuuuuuuu
001050 uu uu uu uu uu uu uu uu   uu uu uu uu uu uu uu uu   uuuuuuuuuuuuuuuu
001060 uu uu uu uu uu uu uu uu   uu uu uu uu uu uu uu uu   uuuuuuuuuuuuuuuu
001070 uu uu uu uu uu uu uu uu   uu uu uu uu uu uu uu uu   uuuuuuuuuuuuuuuu
```

```
000000 MHz   1                MC9S12DP256B    STEPPED
```

Figure 2: Loading FF to A.

The next line was a store command in which we have two operand which is the address “0x1000” and the Acc. “A”. The role of this command is to save the value hold by Acc. “A” to the address “0x1000”. The IP is changed to “C005”. Also we noticed that the address “0x1000” in the memory has changed to FF as shown in Figure 3.



```
Entry
  C000 LDAA   #255
  C002 STAA   0x1000
  C005 CLRA
  C006 STAA   0x1010
  C009 MOVB   0x1000,0x1010
  C00F LDAB   0x1010
  C012 LDX    #17
  C015 EXG    X,B
  C017 STAA   0x5A
  C019 STAA   0x5A
  C01B STAA   0x5A
```

```
HC12   CPU Cycles: 4                                   Auto
D    FFCB     A    FF      B      CB
IX   CBCB     IY   CBCB
IP   C005     PC   C005    PPAGE  0
SP   CBC9     CCR  SXHINZVC
```

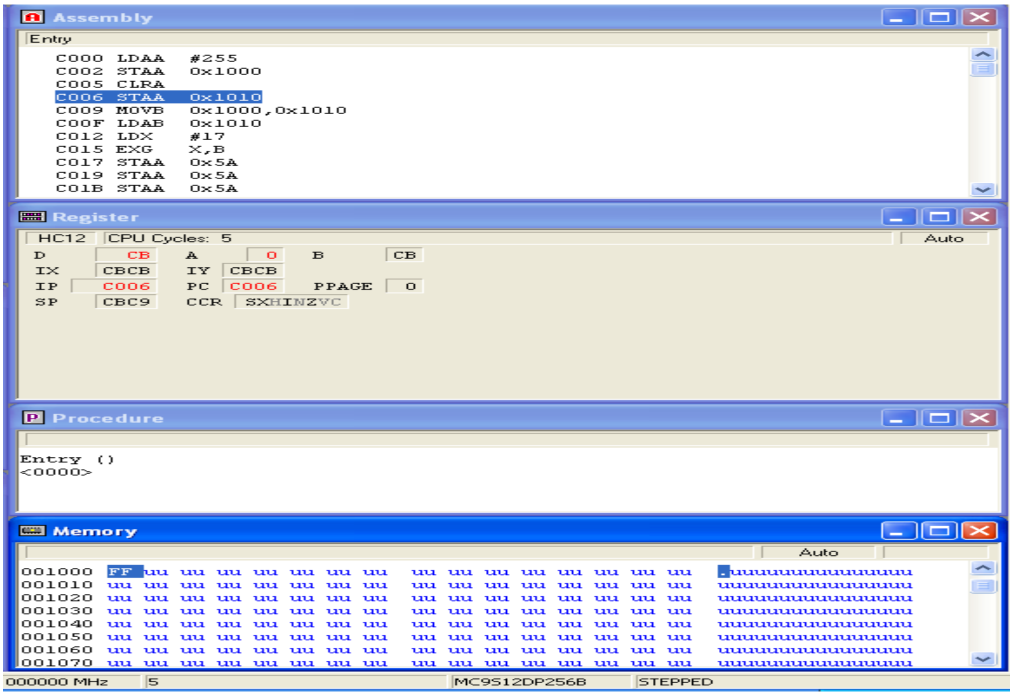
```
Entry ()
<0000>
```

```
                                                                  Auto
001000 FF uu uu uu uu uu uu uu   uu uu uu uu uu uu uu uu   .uuuuuuuuuuuuuuu
001010 uu uu uu uu uu uu uu uu   uu uu uu uu uu uu uu uu   uuuuuuuuuuuuuuuu
001020 uu uu uu uu uu uu uu uu   uu uu uu uu uu uu uu uu   uuuuuuuuuuuuuuuu
001030 uu uu uu uu uu uu uu uu   uu uu uu uu uu uu uu uu   uuuuuuuuuuuuuuuu
001040 uu uu uu uu uu uu uu uu   uu uu uu uu uu uu uu uu   uuuuuuuuuuuuuuuu
001050 uu uu uu uu uu uu uu uu   uu uu uu uu uu uu uu uu   uuuuuuuuuuuuuuuu
001060 uu uu uu uu uu uu uu uu   uu uu uu uu uu uu uu uu   uuuuuuuuuuuuuuuu
001070 uu uu uu uu uu uu uu uu   uu uu uu uu uu uu uu uu   uuuuuuuuuuuuuuuu
```

```
000000 MHz   4                MC9S12DP256B    STEPPED
```

Figure 3: Storing the accumulator A content in 0x1000

We used the clear command to clear the data that is hold by Acc. “A” to 0 as it is shown in Figure 4. We noticed that the value of D changed to “CB” which only the value of Acc. “B” since Acc. “A” is zero.



```
Entry
   C000  LDAA   #255
   C002  STAA   0x1000
   C005  CLRA
   C006  STAA   0x1010
   C009  MOVB   0x1000,0x1010
   C00F  LDAB   0x1010
   C012  LDX    #17
   C015  EXG    X,B
   C017  STAA   0x5A
   C019  STAA   0x5A
   C01B  STAA   0x5A
```

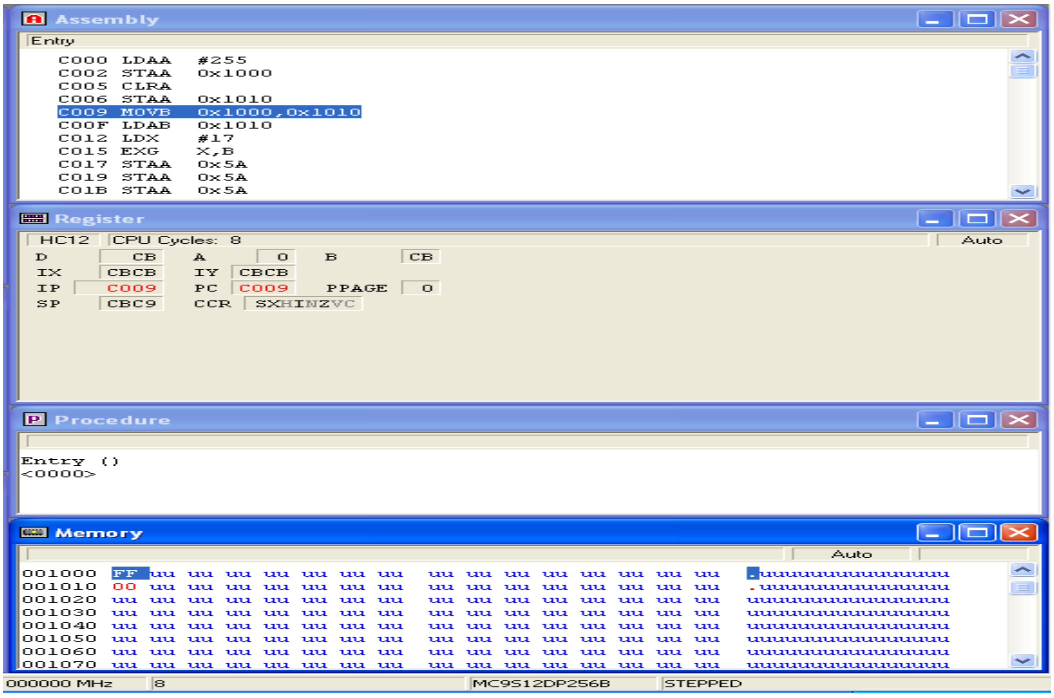
```
HC12   CPU Cycles: 5                                  Auto
D     CB      A     0       B       CB
IX    CBCB    IY    CBCB
IP    C006    PC    C006    PPAGE   0
SP    CBC9    CCR   SXHINZVC
```

```
Entry ()
<0000>
```

```
                                                              Auto
001000 FF uu uu uu uu uu uu uu  uu uu uu uu uu uu uu uu  .uuuuuuuuuuuuuuu
001010 uu uu uu uu uu uu uu uu  uu uu uu uu uu uu uu uu  uuuuuuuuuuuuuuuu
001020 uu uu uu uu uu uu uu uu  uu uu uu uu uu uu uu uu  uuuuuuuuuuuuuuuu
001030 uu uu uu uu uu uu uu uu  uu uu uu uu uu uu uu uu  uuuuuuuuuuuuuuuu
001040 uu uu uu uu uu uu uu uu  uu uu uu uu uu uu uu uu  uuuuuuuuuuuuuuuu
001050 uu uu uu uu uu uu uu uu  uu uu uu uu uu uu uu uu  uuuuuuuuuuuuuuuu
001060 uu uu uu uu uu uu uu uu  uu uu uu uu uu uu uu uu  uuuuuuuuuuuuuuuu
001070 uu uu uu uu uu uu uu uu  uu uu uu uu uu uu uu uu  uuuuuuuuuuuuuuuu
000000 MHz   5      MC9S12DP256B   STEPPED
```

Figure 4: Clearing the content of A

We store the new value that is hold by Acc. "A" to address "0x1010" by using the store command, as shown in figure 5 in the memory section.



```
Entry
   C000  LDAA   #255
   C002  STAA   0x1000
   C005  CLRA
   C006  STAA   0x1010
   C009  MOVB   0x1000,0x1010
   C00F  LDAB   0x1010
   C012  LDX    #17
   C015  EXG    X,B
   C017  STAA   0x5A
   C019  STAA   0x5A
   C01B  STAA   0x5A
```

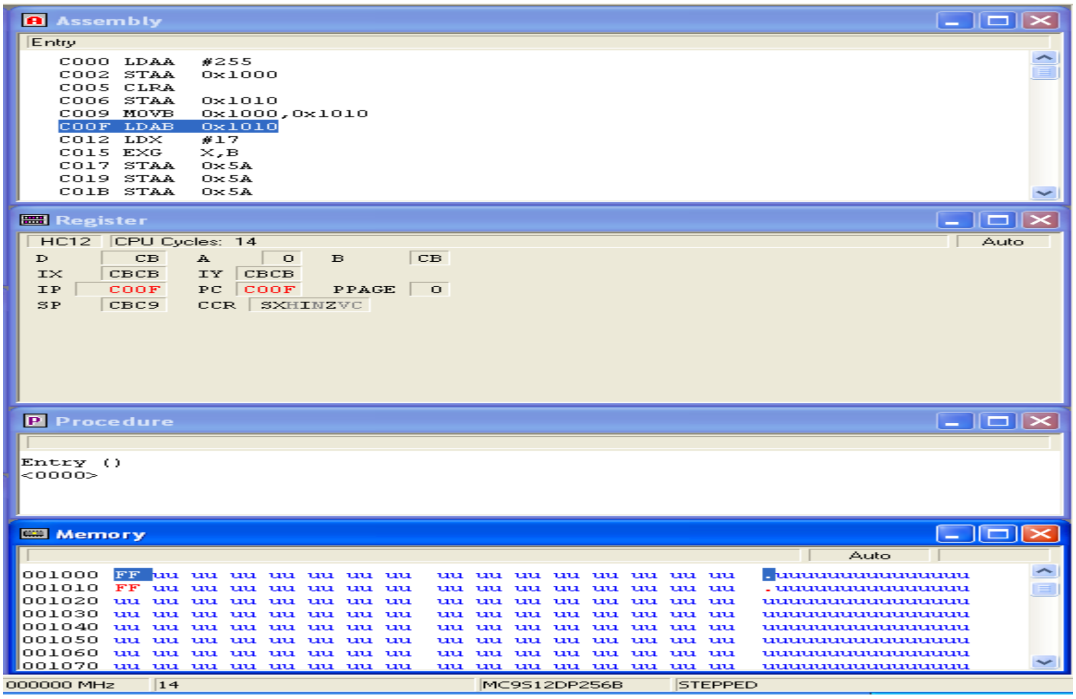
```
HC12   CPU Cycles: 8                                  Auto
D     CB      A     0       B       CB
IX    CBCB    IY    CBCB
IP    C009    PC    C009    PPAGE   0
SP    CBC9    CCR   SXHINZVC
```

```
Entry ()
<0000>
```

```
                                                              Auto
001000 FF uu uu uu uu uu uu uu  uu uu uu uu uu uu uu uu  .uuuuuuuuuuuuuuu
001010 00 uu uu uu uu uu uu uu  uu uu uu uu uu uu uu uu  .uuuuuuuuuuuuuuu
001020 uu uu uu uu uu uu uu uu  uu uu uu uu uu uu uu uu  uuuuuuuuuuuuuuuu
001030 uu uu uu uu uu uu uu uu  uu uu uu uu uu uu uu uu  uuuuuuuuuuuuuuuu
001040 uu uu uu uu uu uu uu uu  uu uu uu uu uu uu uu uu  uuuuuuuuuuuuuuuu
001050 uu uu uu uu uu uu uu uu  uu uu uu uu uu uu uu uu  uuuuuuuuuuuuuuuu
001060 uu uu uu uu uu uu uu uu  uu uu uu uu uu uu uu uu  uuuuuuuuuuuuuuuu
001070 uu uu uu uu uu uu uu uu  uu uu uu uu uu uu uu uu  uuuuuuuuuuuuuuuu
000000 MHz   8      MC9S12DP256B   STEPPED
```

Figure 5: Storing the new value of A in 0x1010

We used the move command "MOVEB" which will use Acc. B as a swap medium to save the data of address "0x1000" to address "0x1010". It is shown in Figure 6.



```
C000 LDAA   #255
C002 STAA   0x1000
C005 CLRA
C006 STAA   0x1010
C009 MOVB   0x1000,0x1010
C00F LDAB   0x1010
C012 LDX    #17
C015 EXG    X,B
C017 STAA   0x5A
C019 STAA   0x5A
C01B STAA   0x5A
```

```
HC12   CPU Cycles: 14
D    CB     A   0      B      CB
IX   CBCB   IY  CBCB
IP   C00F   PC  C00F   PPAGE  0
SP   CBC9   CCR SXHINZVC
```

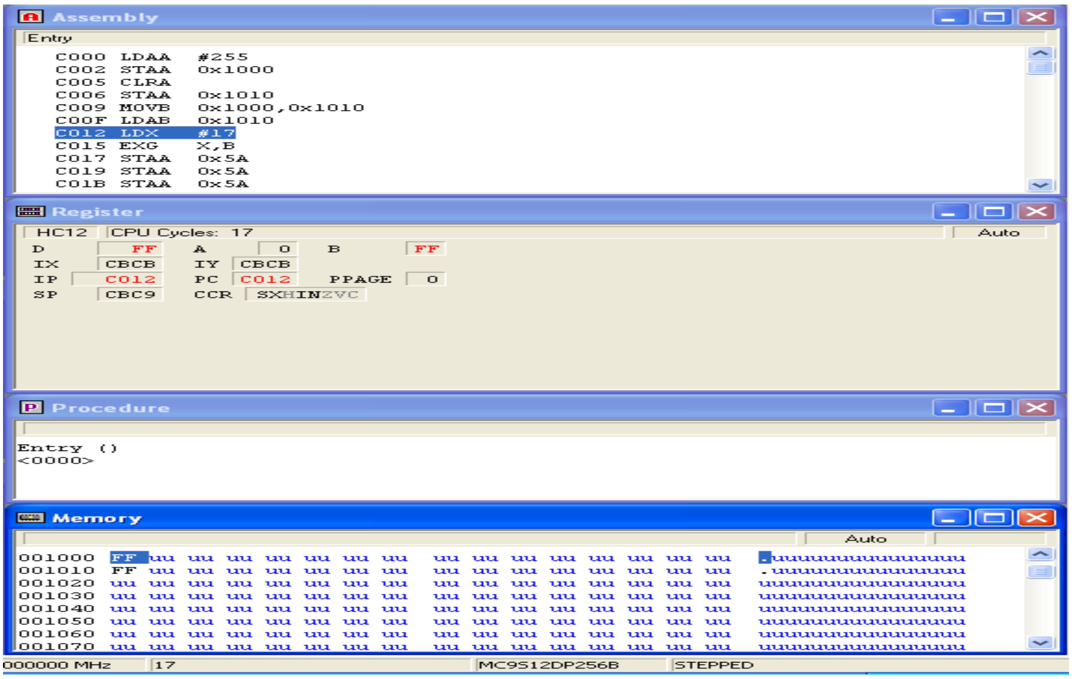
```
Entry ()
<0000>
```

```
001000 FF uu uu uu uu uu uu uu  uu uu uu uu uu uu uu uu  .uuuuuuuuuuuuuuu
001010 FF uu uu uu uu uu uu uu  uu uu uu uu uu uu uu uu  .uuuuuuuuuuuuuuu
001020 uu uu uu uu uu uu uu uu  uu uu uu uu uu uu uu uu  uuuuuuuuuuuuuuuu
001030 uu uu uu uu uu uu uu uu  uu uu uu uu uu uu uu uu  uuuuuuuuuuuuuuuu
001040 uu uu uu uu uu uu uu uu  uu uu uu uu uu uu uu uu  uuuuuuuuuuuuuuuu
001050 uu uu uu uu uu uu uu uu  uu uu uu uu uu uu uu uu  uuuuuuuuuuuuuuuu
001060 uu uu uu uu uu uu uu uu  uu uu uu uu uu uu uu uu  uuuuuuuuuuuuuuuu
001070 uu uu uu uu uu uu uu uu  uu uu uu uu uu uu uu uu  uuuuuuuuuuuuuuuu
```

```
000000 MHz   14   MC9S12DP256B   STEPPED
```

Figure 6: Moving a byte from memory location 0x1000 to 0x1010 using the MOVB command.

We used the load command to take the data from memory “0x1010” then putting it in Acc. B to hold the data.



```
C000 LDAA   #255
C002 STAA   0x1000
C005 CLRA
C006 STAA   0x1010
C009 MOVB   0x1000,0x1010
C00F LDAB   0x1010
C012 LDX    #17
C015 EXG    X,B
C017 STAA   0x5A
C019 STAA   0x5A
C01B STAA   0x5A
```

```
HC12   CPU Cycles: 17
D    FF     A   0      B      FF
IX   CBCB   IY  CBCB
IP   C012   PC  C012   PPAGE  0
SP   CBC9   CCR SXHINZVC
```

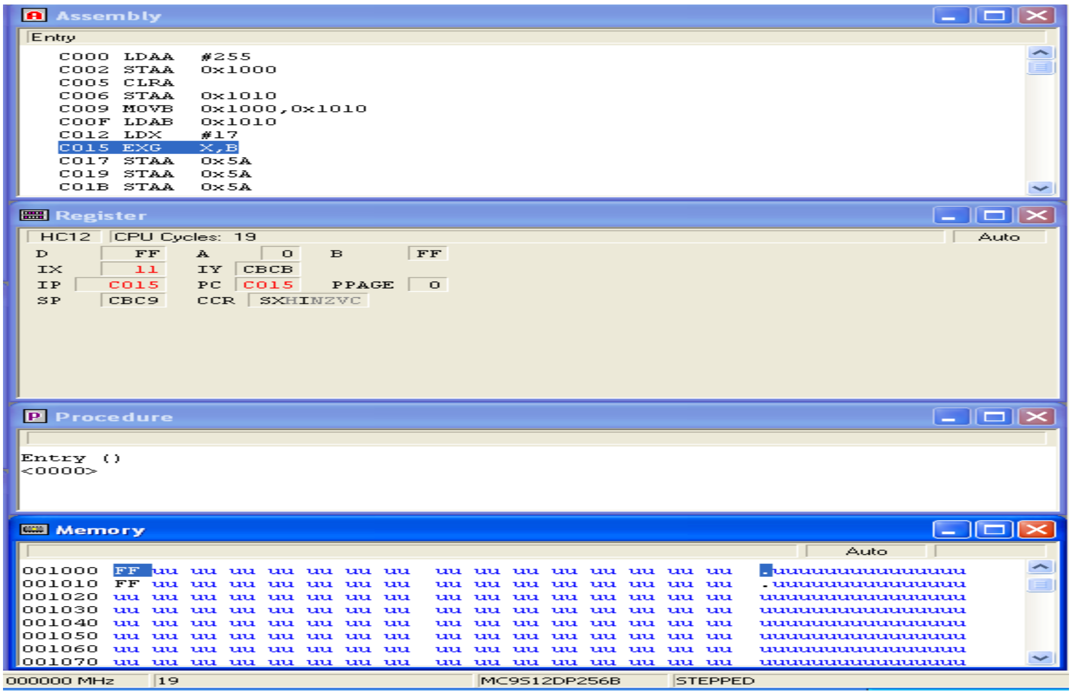
```
Entry ()
<0000>
```

```
001000 FF uu uu uu uu uu uu uu  uu uu uu uu uu uu uu uu  .uuuuuuuuuuuuuuu
001010 FF uu uu uu uu uu uu uu  uu uu uu uu uu uu uu uu  .uuuuuuuuuuuuuuu
001020 uu uu uu uu uu uu uu uu  uu uu uu uu uu uu uu uu  uuuuuuuuuuuuuuuu
001030 uu uu uu uu uu uu uu uu  uu uu uu uu uu uu uu uu  uuuuuuuuuuuuuuuu
001040 uu uu uu uu uu uu uu uu  uu uu uu uu uu uu uu uu  uuuuuuuuuuuuuuuu
001050 uu uu uu uu uu uu uu uu  uu uu uu uu uu uu uu uu  uuuuuuuuuuuuuuuu
001060 uu uu uu uu uu uu uu uu  uu uu uu uu uu uu uu uu  uuuuuuuuuuuuuuuu
001070 uu uu uu uu uu uu uu uu  uu uu uu uu uu uu uu uu  uuuuuuuuuuuuuuuu
```

```
000000 MHz   17   MC9S12DP256B   STEPPED
```

Figure 7: LDAB $1010 step

Load the data “0011” to Acc “X”.



```
Entry
    C000  LDAA   #255
    C002  STAA   0x1000
    C005  CLRA
    C006  STAA   0x1010
    C009  MOVB   0x1000,0x1010
    C00F  LDAB   0x1010
    C012  LDX    #17
    C015  EXG    X,B
    C017  STAA   0x5A
    C019  STAA   0x5A
    C01B  STAA   0x5A
```

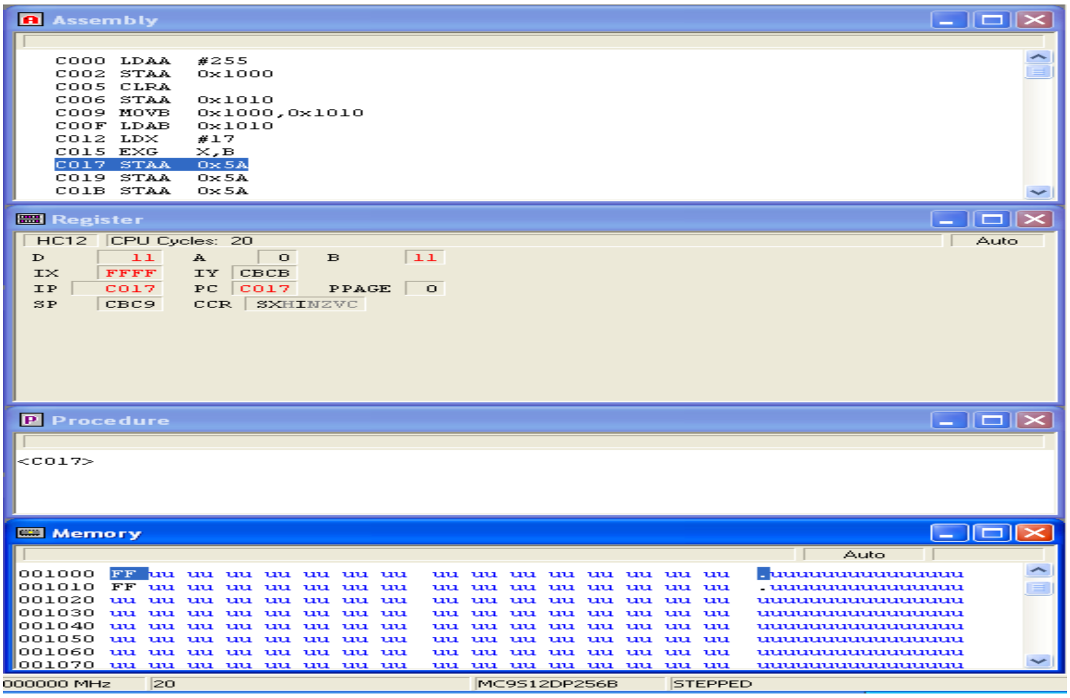
```
HC12  CPU Cycles: 19                                   Auto
D     FF      A     0      B      FF
IX    11      IY    CBCB
IP    C015    PC    C015   PPAGE  0
SP    CBC9    CCR   SXHINZVC
```

```
Entry ()
<0000>
```

```
                                                        Auto
001000 FF uu uu uu uu uu uu uu  uu uu uu uu uu uu uu uu  .uuuuuuuuuuuuuuu
001010 FF uu uu uu uu uu uu uu  uu uu uu uu uu uu uu uu  .uuuuuuuuuuuuuuu
001020 uu uu uu uu uu uu uu uu  uu uu uu uu uu uu uu uu  uuuuuuuuuuuuuuuu
001030 uu uu uu uu uu uu uu uu  uu uu uu uu uu uu uu uu  uuuuuuuuuuuuuuuu
001040 uu uu uu uu uu uu uu uu  uu uu uu uu uu uu uu uu  uuuuuuuuuuuuuuuu
001050 uu uu uu uu uu uu uu uu  uu uu uu uu uu uu uu uu  uuuuuuuuuuuuuuuu
001060 uu uu uu uu uu uu uu uu  uu uu uu uu uu uu uu uu  uuuuuuuuuuuuuuuu
001070 uu uu uu uu uu uu uu uu  uu uu uu uu uu uu uu uu  uuuuuuuuuuuuuuuu
000000 MHz   19          MC9S12DP256B   STEPPED
```

Figure 8: Code LDX #$0011 step

Exchange commad is used to swap the values of Acc. “X” and Acc. “B”



```
    C000  LDAA   #255
    C002  STAA   0x1000
    C005  CLRA
    C006  STAA   0x1010
    C009  MOVB   0x1000,0x1010
    C00F  LDAB   0x1010
    C012  LDX    #17
    C015  EXG    X,B
    C017  STAA   0x5A
    C019  STAA   0x5A
    C01B  STAA   0x5A
```

```
HC12  CPU Cycles: 20                                   Auto
D     11      A     0      B      11
IX    FFFF    IY    CBCB
IP    C017    PC    C017   PPAGE  0
SP    CBC9    CCR   SXHINZVC
```

```
<C017>
```

```
                                                        Auto
001000 FF uu uu uu uu uu uu uu  uu uu uu uu uu uu uu uu  .uuuuuuuuuuuuuuu
001010 FF uu uu uu uu uu uu uu  uu uu uu uu uu uu uu uu  .uuuuuuuuuuuuuuu
001020 uu uu uu uu uu uu uu uu  uu uu uu uu uu uu uu uu  uuuuuuuuuuuuuuuu
001030 uu uu uu uu uu uu uu uu  uu uu uu uu uu uu uu uu  uuuuuuuuuuuuuuuu
001040 uu uu uu uu uu uu uu uu  uu uu uu uu uu uu uu uu  uuuuuuuuuuuuuuuu
001050 uu uu uu uu uu uu uu uu  uu uu uu uu uu uu uu uu  uuuuuuuuuuuuuuuu
001060 uu uu uu uu uu uu uu uu  uu uu uu uu uu uu uu uu  uuuuuuuuuuuuuuuu
001070 uu uu uu uu uu uu uu uu  uu uu uu uu uu uu uu uu  uuuuuuuuuuuuuuuu
000000 MHz   20          MC9S12DP256B   STEPPED
```

Figure 9: Code EXG X,B step

Here, the content of X and B are exchanged.

3.

1. Immediate addressing mode
2. Inherent addressing mode
3. Direct addressing mode

Table1: instruction set execution results:

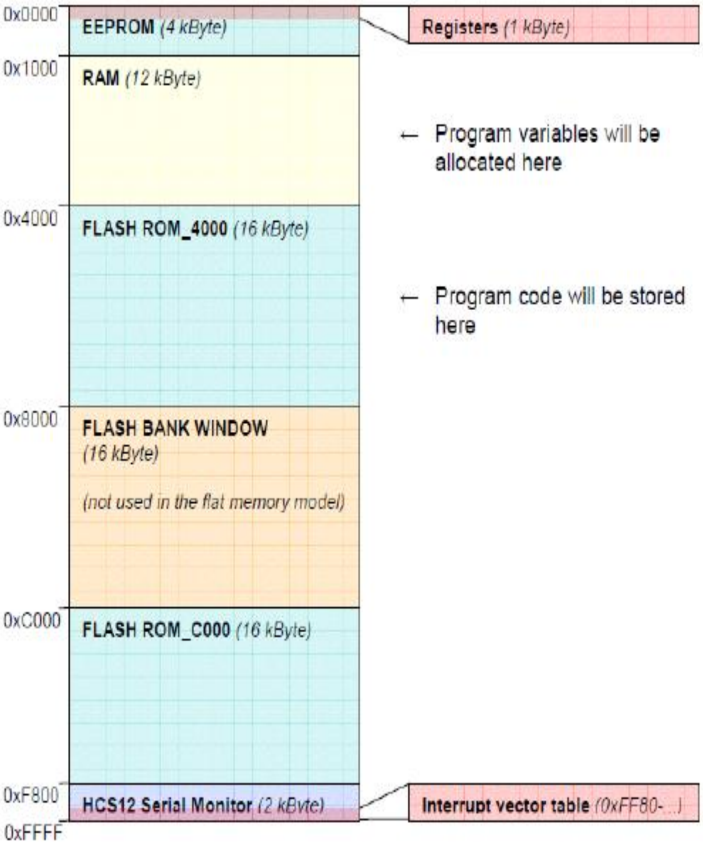
Instruction	Memory Location 0x1000	Memory Location 0x1010	Accumulator A	Accumulator B	Index Register X
LDAA #$FF	uu		FF		
STAA $1000	FF		FF		
CLRA	FF	uu	0		
STAA $1010	FF	00	0		
MOVB $1000,$1010	FF	FF	0	CB	
LDAB $1010	FF	FF	0	FF	
LDX #$0011	FF	FF	0	FF	0011
EXG X,B	FF	FF	0	11	FFFF

1.4)

1)

- 12 Kbyte RAM
- 4 Kbyte EEPROM
- 256 Kbyte flash
- It has registers

2)


0x0000
EEPROM (4 kByte)
Registers (1 kByte)
0x1000
RAM (12 kByte)
← Program variables will be allocated here
0x4000
FLASH ROM_4000 (16 kByte)
← Program code will be stored here
0x8000
FLASH BANK WINDOW (16 kByte)
(not used in the flat memory model)
0xC000
FLASH ROM_C000 (16 kByte)
0xF800
HCS12 Serial Monitor (2 kByte)
Interrupt vector table (0xFF80-...)
0xFFFF

Registers available in the micro controller.

- Accumulator A , Accumulator B combing them results in accumulator D
- Index Register X,y ,
- SP ,
- PC