Birzeit University

Faculty of Information Technology
Computer Systems Dept.

Digital Electronics And Computer Organization Lab
ENCS 211

Experiment No. 10

Introduction to The DEBUG Program

Submitted by:

Elias Hazboun **No.** 1081518

Iyad Mousa **No.** 1081234

Instructor's Name: Mr. Abdulsalam Sayyad

Section: 4

Date: 7 / 4 / 2010

❖ Procedure and Discussion :

To run the DEBUG program on a vista machine, we clicked on start, then run, and we typed "cmd" and pressed enter. A console window appeared and we entered "DEBUG".

1. Immediate Operands

Activities 1.1 & 1.2: We first entered the address of the offset which is 100h, then we entered the specified instructions, and pressed 'U' for the program to un-assemble the instructions into machine code. See table below for results and figure 1 for illustration.

Assembly code	Machine Code
MOV AX, 2864	B86428
ADD AX, 3749	054937
MOV BX, AX	89C3
SUB BX, 2805	81EB0528
NOP	90

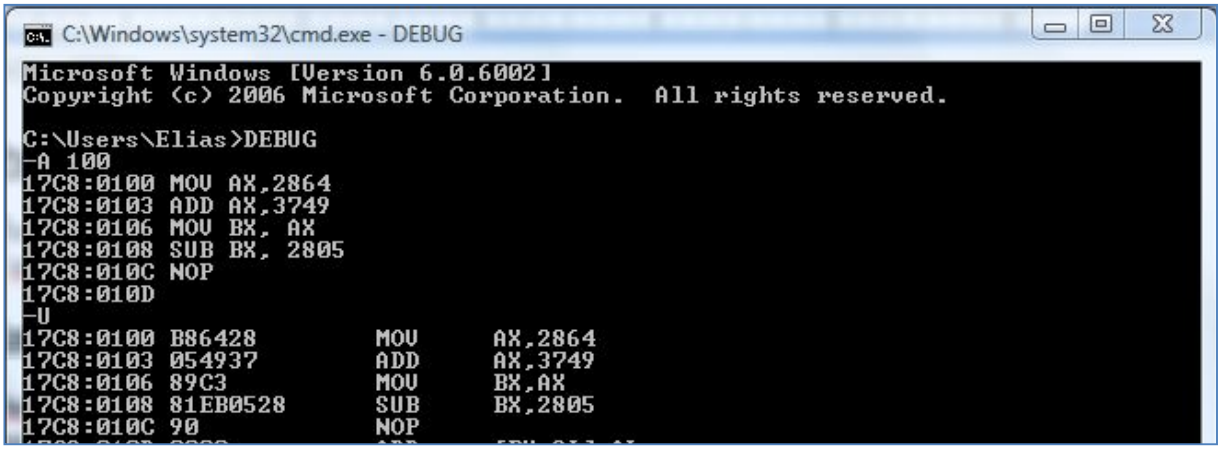

Figure 1

Activity 1.3: We can calculate how many bytes each instruction needs from its length:

Assembly code	Number of bytes
MOV AX, 2864	3
ADD AX, 3749	3
MOV BX, AX	2
SUB BX, 2805	4
NOP	1

Activity 1.4: Intel's x86 architecture uses little endian to store data. That is 2864 is stored at the 101h offset as 64 in 101h and 28 in 102h

Activity 1.5: The contents of the registers are shown using the command 'R'. See figure 2.

Register	Content
CS	0B0C
IP	0100
AX	0000
BX	0000

Figure 2

Activity 1.6:

Register	MOV AX, 2864	ADD AX, 3749	MOV BX, AX	SUB BX, 2805
CS	0B0C	0B0C	0B0C	0B0C
IP	0103	0106	0108	010C
AX	2864	5FAD	5FAD	5FAD
BX	0000	0000	5FAD	37A8

Activity 1.7: To execute the program we use the command 'T', which runs the program step by step- instruction by instruction, with each instruction showing us the contents of the registers. See figure 3.

Register	MOV AX, 2864	ADD AX, 3749	MOV BX, AX	SUB BX, 2805
CS	0B0C	0B0C	0B0C	0B0C
IP	0103	0106	0108	010C
AX	2864	5FAD	5FAD	5FAD
BX	0000	0000	5FAD	37A8

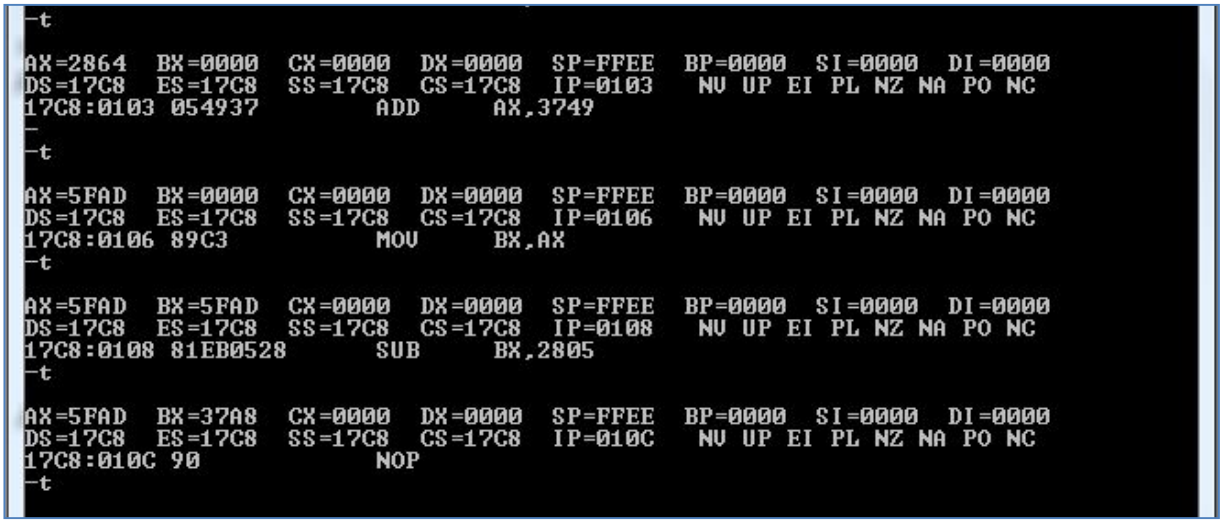

Figure 3

Activity 1.8: Since the IP stands for the **I**nstruction **P**ointer, it is only fair that its contents are the instruction currently in execution; hence its contents should change with each instruction.

Activity 1.9: The offset is **0103** (refer to figure 1), while the physical address = **IP + CS*10 = 0103 + B0C0 = B1C3**

2. Immediate Operands

Activities 2.1 & 2.2: We entered the data specified at the offset memory location 200h using the command 'E', then we entered the program instructions at the offset memory location 100h using also the command 'E'. See figure 4.

Figure 4

Activity 2.3: to find the assembly code we use the command ‘U’. See figure 5.

Assembly code	Machine Code
MOV AX, [0200]	A10002
MOV BX, [0202]	8B1E0202
ADD BX, AX	01C3
MOV [0204], BX	891E0402
NOP	90

```
-u
17C8:0100 A10002        MOV     AX,[0200]
17C8:0103 8B1E0202      MOV     BX,[0202]
17C8:0107 01C3          ADD     BX,AX
17C8:0109 891E0402      MOV     [0204],BX
17C8:010D 90            NOP
17C8:010E 0000          ADD     [BX+SI],AL
17C8:0110 0000          ADD     [BX+SI],AL
17C8:0112 0000          ADD     [BX+SI],AL
17C8:0114 0000          ADD     [BX+SI],AL
17C8:0116 0000          ADD     [BX+SI],AL
17C8:0118 0000          ADD     [BX+SI],AL
17C8:011A 0000          ADD     [BX+SI],AL
17C8:011C 3400          XOR     AL,00
17C8:011E B717          MOV     BH,17
-
```

Figure 5

Activity 2.4: From figure 4, we can clearly see that the first 8 bits are ‘1’.

Activity 2.5: From figure 4, we can clearly see that the first 16 bits ar “1B”.

Activity 2.6: We first move after tracing the code. We can see that the contents of 204 are the addition of [0200] & [0202] which are: “**E951**”. See figure 6

```
AX=9F1B  BX=E951  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=17C8  ES=17C8  SS=17C8  CS=17C8  IP=0109   NV UP EI NG NZ AC PO NC
17C8:0109 891E0402      MOV     [0204],BX                       DS:0204=0000
-t

AX=9F1B  BX=E951  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=17C8  ES=17C8  SS=17C8  CS=17C8  IP=010D   NV UP EI NG NZ AC PO NC
17C8:010D 90            NOP
```

Figure 6

Activity 2.7: The contents of AX can be investigated using the command ‘T’, which shows them to be: 9F1B. See figure 7

```
-t

AX=9F1B  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=17C8  ES=17C8  SS=17C8  CS=17C8  IP=0103   NV UP EI PL NZ NA PO NC
17C8:0103 8B1E0202      MOV     BX,[0202]                       DS:0202=4A36
```

Figure 7

Activity 2.8: Again we use the command 'T' to execute the program and inspect the contents after each instruction. See figure 8.

	A10002	8B1E0202	01C3	891E0402
DS:204	00	00	00	6C88

```
-t

AX=9F1B  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=17C8  ES=17C8  SS=17C8  CS=17C8  IP=0103   NV UP EI PL NZ NA PO NC
17C8:0103 8B1E0202      MOV     BX,[0202]                      DS:0202=4A36
-t

AX=9F1B  BX=4A36  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=17C8  ES=17C8  SS=17C8  CS=17C8  IP=0107   NV UP EI PL NZ NA PO NC
17C8:0107 01C3          ADD     BX,AX
-t

AX=9F1B  BX=E951  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=17C8  ES=17C8  SS=17C8  CS=17C8  IP=0109   NV UP EI NG NZ AC PO NC
17C8:0109 891E0402      MOV     [0204],BX                      DS:0204=0000
-t

AX=9F1B  BX=E951  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=17C8  ES=17C8  SS=17C8  CS=17C8  IP=010D   NV UP EI NG NZ AC PO NC
17C8:010D 90            NOP
-t

AX=9F1B  BX=E951  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=17C8  ES=17C8  SS=17C8  CS=17C8  IP=010E   NV UP EI NG NZ AC PO NC
17C8:010E 0000          ADD     [BX+SI],AL                     DS:E951=00
```

Figure 8

3. Entering assembly code in DEBUG

Activities 3.1 & 3.2: we enter the specified assembly code using the command 'A' at the CS offset address 100h. Then using the command 'T' we executed the program step by step. See figure 9.

	MOV CL,42	MOV DL,2A	ADD CL,DL
CL	42	42	6C
DL	00	2A	2A
IP	0102	0104	0106

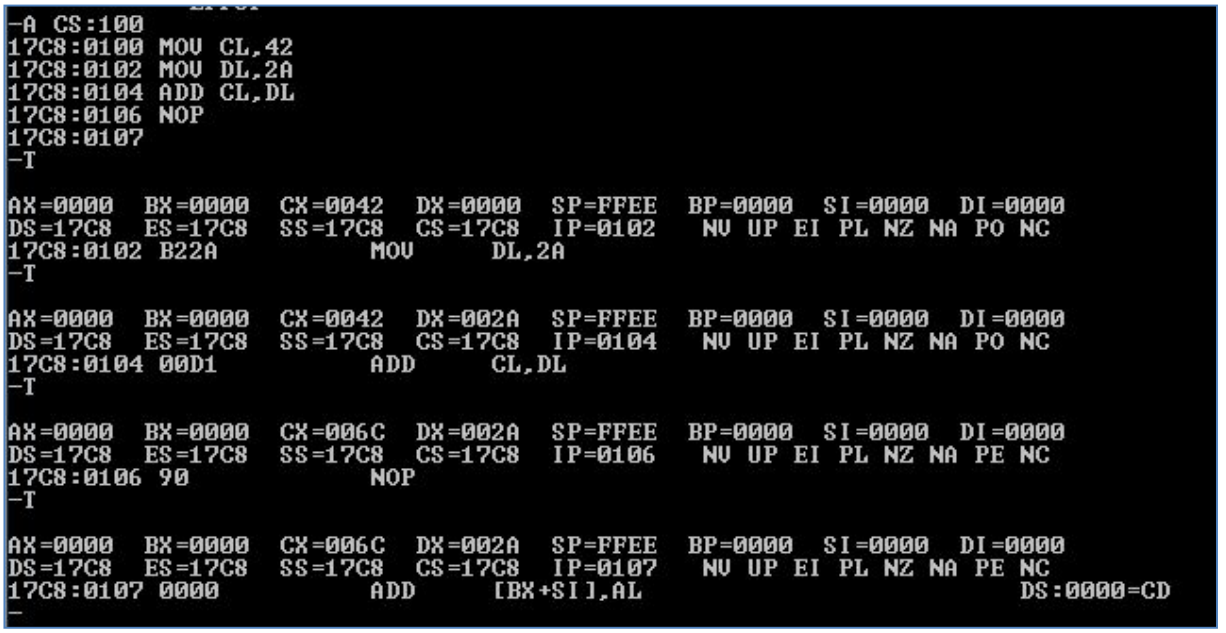

Figure 9

❖ Conclusion :

This experiment took us away from what we used to do in the digital lab; this experiment was aimed at solidifying our understanding of how programs are executed on a modern computer. We –for the first time- interacted with registers and RAM directly, with commands such as ‘T’, ‘U’, ‘R’ and ‘A’, each of which has a specific purpose in the DEBUG program. We were able to enter data and code to the data and code segments respectively and see the results after executing the program. And we were able to convert from assembly to machine code and vice-versa.

- Note that the machines we used a combination of windows operating system and Intel x86 processors.