* Procedure And discussion :

To run the DEBUG program on windows 8 , I typed on search box “cmd” then pressed enter ,after that I typed debug command.

1-immediate operands:

**Activities 1.1 and 1.2:** first of all we entered the address of the offset “100h” then we entered the specific instruction we need -in this case ”U” to un-assemble that instruction into machine code , the tables below have the results.

|  |  |
| --- | --- |
| 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:** Here we can calculate the number if bytes each instruction need.

|  |  |
| --- | --- |
| **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 the101h offset as 64 in 101h and 28 in 102h

**Activity 1.5:** The contents of the registers are shown using the command ‘R’. See the figure and table below.

|  |  |
| --- | --- |
| 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, with each instruction showing us the contents of the registers.

See the figure and table below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 pointer, 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- Memory Addressing:**

**“This part I did it at home ☺ “**

**Activities 2.1 & 2.2:** I entered the data specified at the offset memory location 200h using the command ‘E’, thenI 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** |

****

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:** I 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



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



Figure 7

**Activity 2.8:** Again I used 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 |



Figure 8

**3- Entering Assembly code in DEBUG:**

**“Also this part I did it at home ☺ “**

**Activities 3.1 & 3.2:** I entered the specified assembly code using the command ‘A’ at the CS offset address 100h. Then using the command ‘T’ I executed the program step by step.

See figure 9 next page.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **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.

.

* References:
* Some pictures from other reports.
* Theory from our Pdf experiment sheet.