***بسم الله الرحمن الرحيم***

**

**

***Computer Systems Engineering Department***

***ENCS 212***

***Digital Electronics And Computer Organization Lab***

***Report******For Experiment NO.10***

***Introduction to DEBUG Program***

***Student Name: Mutaz Abu-Awad ID#: 1091838***

In this lab we will be use some commands in order to be familiarized with DOS DEGUG program.

At first we will open the DOS DEGUG program from start 🡪 run --type “cmd”🡪Enter.

And then we will activate the DEBUG program by typing DEBUG at the DOS prompt.

After that we will make some activites :

***PART I : Immediate Operands***

**Activity 1.1:**

we will use the command A 100 at the DEBUG program to

enter the following program instructions in assembly code at the offset memory location 100h .

MOV AX, 2864

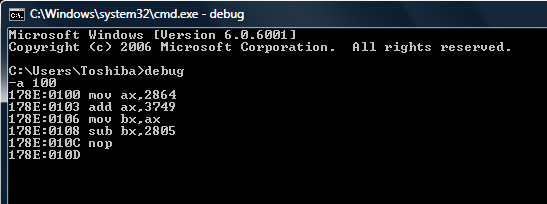
ADD AX, 3749

MOV BX, AX

SUB BX, 2805

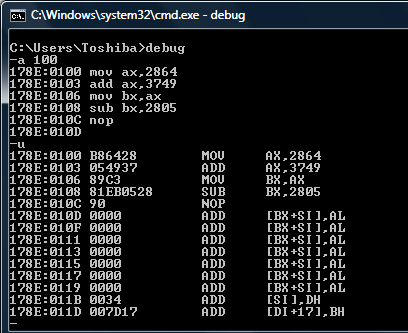
NOP

When we entered them we get :



**Activity 1.2:**

We will use DEBUG command U to unassembled the instructions(get the machine code of the instructions) in activity 1.1.



We get the following code after using command u :

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

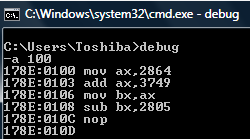
**Activity 1.3:**

We want to knowhow many bytes does it need to represent each instruction in binary .

|  |  |
| --- | --- |
| Assembly code | Machine code |
| MOV AX, 2864 | 3 |
| ADD AX, 3749 | 3 |
| MOV BX, AX | 2 |
| SUB BX, 2805 | 4 |
| NOP | 1 |

We know the previous information from the address of each instruction

MOV AX, 2864 stored in 178E:100 , and the next stored in 178E:103 , so the first one needed 3 bytes ,, and so on ,, as the following :



**Activity 1.4:**

We can store the (immediate) data 2864 at memory offset 101h using little Indian , so :

64 is stored at memory location 101h

28 is stored at memory location 102h

**Activity 1.5:**

We can use DEBUG command **R** to displaythe contents of **CS**, **IP**, **AX**, and **BX**.



|  |  |
| --- | --- |
| Register | Content |
| CS | 178E |
| IP | 0100 |
| AX | 0000 |
| BX | 0000 |

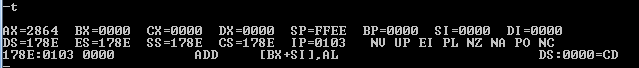
**Activity 1.6:**

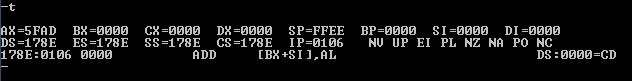
The contents of **CS**, **IP**, **AX**, and **BX** after execution of each instruction:

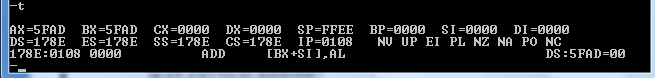
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Register | MOV AX, 2864 | ADD AX, 3749 | MOV BX, AX | SUB BX, 2805 |
| CS | **178E** | **178E** | **178E** | 178E |
| IP | **0103** | **0106** | **0108** | 010C |
| AX | **2864** | **5FAD** | **5FAD** | 5FAD |
| BX | 0000 | 0000 | 5FAD | 37A8 |

**Activity 1.7:**

We will use the T command to execute the program. Then we will determine the content of the above registers after executing each instruction.









We can summarize the previous in the following table :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Register | MOV AX, 2864 | ADD AX, 3749 | MOV BX, AX | SUB BX, 2805 |
| CS | **178E** | **178E** | **178E** | 178E |
| IP | **0103** | **0106** | **0108** | 010C |
| AX | **2864** | **5FAD** | **5FAD** | 5FAD |
| BX | 0000 | 0000 | 5FAD | 37A8 |

**Activity 1.8:**

The content of IP changes after each instruction is executed because IP holds the address of the next instruction ,, and it is changes according to the size of the instruction in bytes , words ,, etc ,,,,,

**Activity 1.9:**

The offset address of the second **MOV** instruction is 0106h.

It’s physical address can be obtained by multiplying the DS with 10h and then add the IP to it ,, so :

physical address = DS\*10h + IP

= 178E \*10h + 0106

= 179E6

***PART II : Memory Addressing***

**Activity 2.1:**

We use DEBUG command **E to**  enter the following data at the offset memory location **200h.**

**E DS: 200 1B 9F**

**E DS: 202 36 4A 00 00**

**E DS: 206 2A 2A 2A**

****

**Activity 2.2:**

We willuse DEBUG command **E to** enter the following program instructions in machine code at the offset memory location **100h.**

**E CS: 100 A1 00 02**

**E CS: 103 8B 1E 02 02**

**E CS: 107 01 C3**

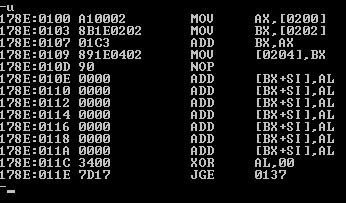
**E CS: 109 89 1E 04 02**

**E CS: 10D 90**

****

**Activity 2.3:**

The assembly code corresponding to each machine codeInstructions is :

****

We can summarize that in the following table :

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

**Activity 2.4:**

The 8-bit data value stored at DS: 0200 after the data in Activity 2.1has been entered is : 1B (1 byte).

**Activity 2.5:**

The 16-bit data value stored at DS: 0200 after the data in Activity 2.1 has been entered is : 9F 1B ( stored in reverse order “ little Indian” ).

**Activity 2.6:**

The data value stored at DS: 0204 AFTER the code in Activity 2.2 is executed is : 1E (1 byte).

**Activity 2.7:**

Execute the program, and then determine the content of **AX** after the instruction **A10002** is executed is 9F 1B (little Indian).

**Activity 2.8:**

The content (data value) of memory at offset address 0204 after each instruction is :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | A10002 | 8B1E0202 | 01C3 | 891E0402 |
| DS:204 | 0000 | 0000 | 0000 | 51E9 |

***PART II : Entering assembly code in DEBUG***

**Activity 3.1:**

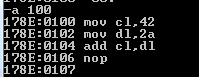
We will use the DEBUG command **A** to enter the following assembly code at CS offset address 100h:

**MOV CL, 42**

**MOV DL, 2A**

**ADD CL, DL**

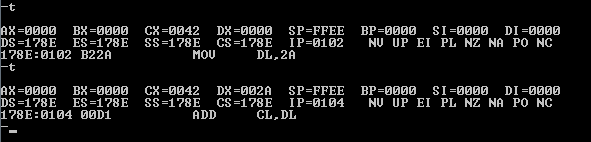
**NOP**

****

**Activity 3.2:**

We willexecute the program using the **T** command. and then we will determine the content of **CL**, **DL**, and **IP** after execution of each instruction

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



***Conclusion :***

In this experiment we learned :

1. how to use the debug program , also we learned the job of some commands in it ,, such that :

A : to enter some assembly instructions in a specific offset in the memory.

R : to view the content of the registers after execution of a program.

T : to execute some instructions and then view the content of registers after it.

U : to get the machine code of an assembly instruction.

E : toenter the data at the offset memory location .

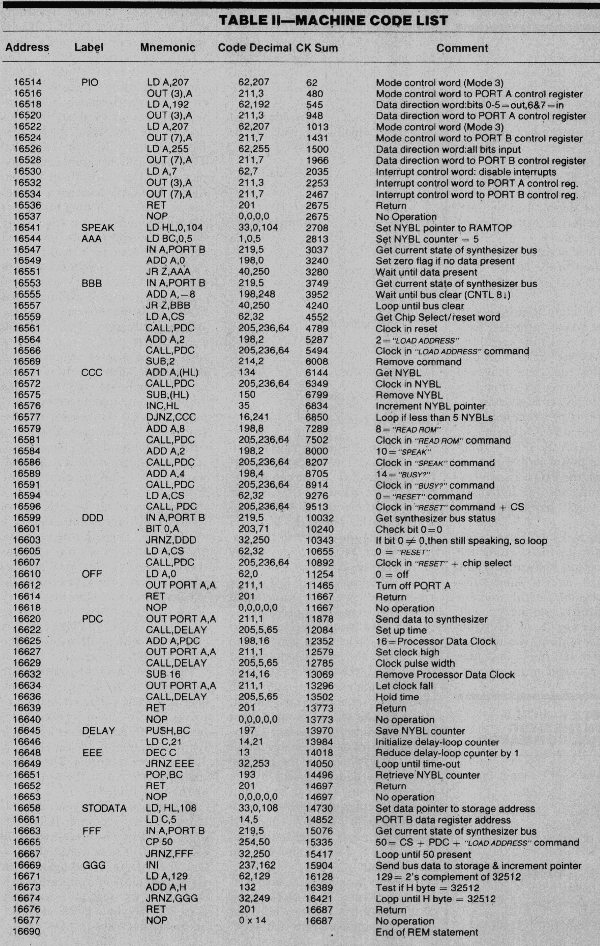
2. the difference between the assembly code and the machine code :

The machine code is the lowest-level representation of a [compiled](http://en.wikipedia.org/wiki/Compiled) and/or [assembled](http://en.wikipedia.org/wiki/Assembly_language) [computer program](http://en.wikipedia.org/wiki/Computer_program). And it is understood by the human ,, but the assembly code can be understood by the human.

Also A machine code instruction set may have all instructions of the same length, or it may have variable-length instructions. Most instructions have one or more [opcode](http://en.wikipedia.org/wiki/Opcode) fields which specifies the basic instruction type (such as arithmetic, logical, jump, etc) and the actual operation (such as add or compare).

A [utility program](http://en.wikipedia.org/wiki/Utility_program) called an **assembler** is used to translate assembly language statements into the target computer's machine code.

Attached a machine code list in a picture :



References :

<http://en.wikipedia.org/wiki/Assembly_language>

<http://en.wikipedia.org/wiki/Machine_code#Machine_code_instructions>

<http://werdav.tripod.com/spspz81c.gif>

<http://www.wisegeek.com/what-is-machine-code.htm>