**Computer Organization and Architecture (EET2211)**

**LAB III: Evaluate Different Logical operations on two 16 bit Data**

**Siksha ‘O’ Anusandhan Deemed to be University, Bhubaneswar**

|  |  |  |  |
| --- | --- | --- | --- |
| **Branch:** Computer Science and Engineering **Section:** ‘D’ | | | |
| **S. No.** | **Name** | **Registration No.** | **Signature** |
| 52 | Saswat Mohanty | 1941012407 | **D:\Pics and Sign\sign.jpg** |

**Marks: \_\_\_\_\_\_/10**

**Remarks:**

**Teacher’s Signature**

1. **OBJECTIVE:**
2. AND two 16 bit numbers using direct addressing mode.
3. OR two 16 bit numbers using direct addressing mode.
4. NOT of a 16 bit number using direct addressing mode.
5. XOR of two 16 bit numbers using direct addressing mode.

**II. PRE-LAB**

**For Obj. 1:**

1. **Explain direct addressing mode briefly.**

It is the addressing mode in which the effective address of the memory location is written directly in the instruction.

1. **Examine & analyze the output obtained from AND of two 16 bit numbers.**

*mov ax,[1000h]*

*mov bx,[1002h]*

*and ax,bx*

[1000h] = 1234h

[1002h] = 4321h

Output: 220h

1. **Write the assembly code.**

|  |
| --- |
| **org 100h**  **mov ax,0000h**  **mov ds,ax**  **mov ax,[3000h]**  **mov bx,[3002h]**  **and ax,bx**  **mov [3004h],ax**  **hlt**  **ret** |

**For Obj. 2:**

1. **Examine & analyze the output obtained from OR of two 16 bit numbers.**

*mov ax,[1000h]*

*mov bx,[1002h]*

*or ax,bx*

[1000h] = 1234h

[1002h] = 4321h

Output: 5335h

1. **Write the assembly code.**

|  |
| --- |
| **org 100h**  **mov ax,0000h**  **mov ds,ax**  **mov ax,[3000h]**  **mov bx,[3002h]**  **or ax,bx**  **mov [3004h],ax**  **hlt**  **ret** |

**For Obj. 3:**

1. **Examine & analyze the output obtained from NOT of a 16 bit number.**

*mov ax,1234h*

*not ax*

Output: EDCBh

1. **Write the assembly code.**

|  |
| --- |
| **org 100h**  **mov ax,0000h**  **mov ds,ax**  **mov ax,[3000h]**  **not ax**  **mov [3002h],ax**  **hlt**  **ret** |

**For Obj. 4:**

1. **Examine & analyze the output obtained from XOR of two 16 bit numbers.**

*mov ax,[1000h]*

*mov bx,[1002h]*

*xor ax,bx*

[1000h] = 1234h

[1002h] = 4321h

Output: 5115h

1. **Write the assembly code.**

|  |
| --- |
| **org 100h**  **mov ax,0000h**  **mov ds,ax**  **mov ax,[3000h]**  **mov bx,[3002h]**  **xor ax,bx**  **mov [3004h],ax**  **hlt**  **ret** |

1. **LAB:**

**Assembly Program:**

**For Obj. 1**

|  |
| --- |
| **; SASWAT MOHANTY**  **; 1941012407**  **; AND two 16 bit numbers using direct addressing mode**  **org 100h**  **mov ax,0000h**  **mov ds,ax**  **mov ax,[3000h] ; Value stored at 3000 = 0202 -> 0000 0010 0000 0010**  **mov bx,[3002h] ; Value stored at 3002 = 0202 -> 0000 0010 0000 0010**  **and ax,bx ; ----------------------------------------------------**  **mov [3004h],ax ; AND -> 0000 0010 0000 0010 = 0202**    **hlt**  **ret** |

**For Obj. 2**

|  |
| --- |
| **; SASWAT MOHANTY**  **; 1941012407**  **; OR two 16 bit numbers using direct addressing mode**  **org 100h**  **mov ax,0000h**  **mov ds,ax**  **mov ax,[3000h] ; Value stored at 3000 = 0202 -> 0000 0010 0000 0010**  **mov bx,[3002h] ; Value stored at 3002 = 0303 -> 0000 0011 0000 0011**  **or ax,bx ; -----------------------------------------------**  **mov [3004h],ax ; OR -> 0000 0011 0000 0011 = 0303**    **hlt**  **ret** |

**For Obj. 3**

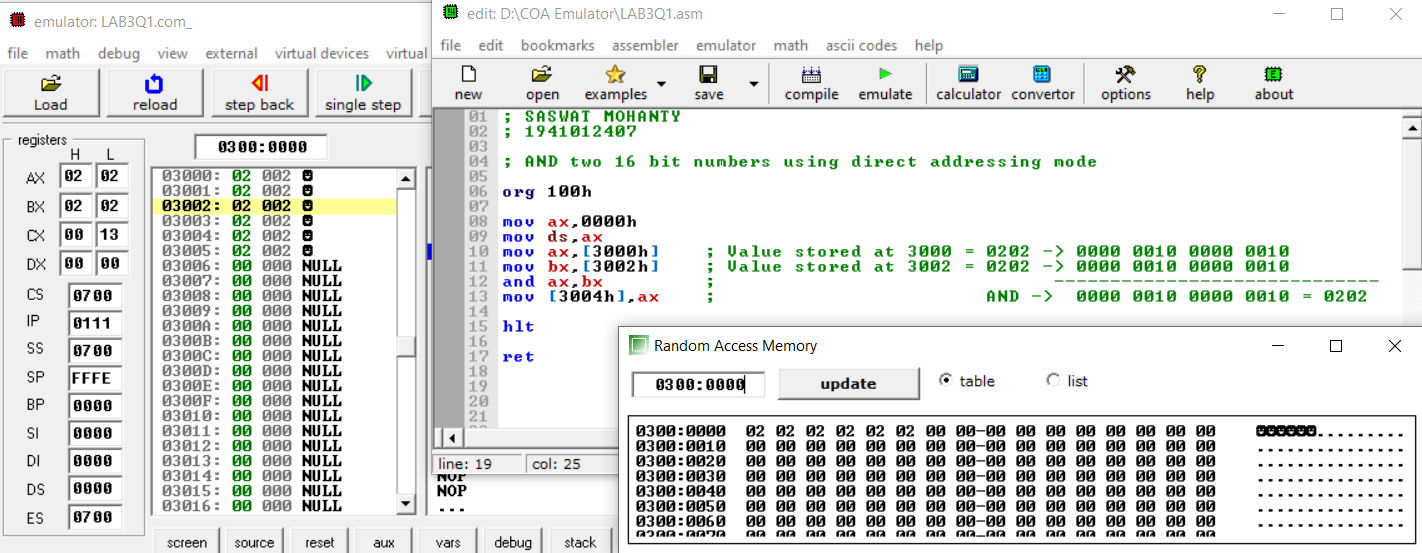
|  |
| --- |
| **; SASWAT MOHANTY**  **; 1941012407**  **; NOT of a 16 bit number using direct addressing mode**  **org 100h**  **mov ax,0000h**  **mov ds,ax**  **mov ax,[3000h] ; Value stored at 3000 = 0202 -> 0000 0010 0000 0010**  **not ax ; ----------------------------------------------------**  **mov [3002h],ax ; NOT -> 1111 1101 1111 1101 = FDFD**    **hlt**  **ret** |

**For Obj. 4**

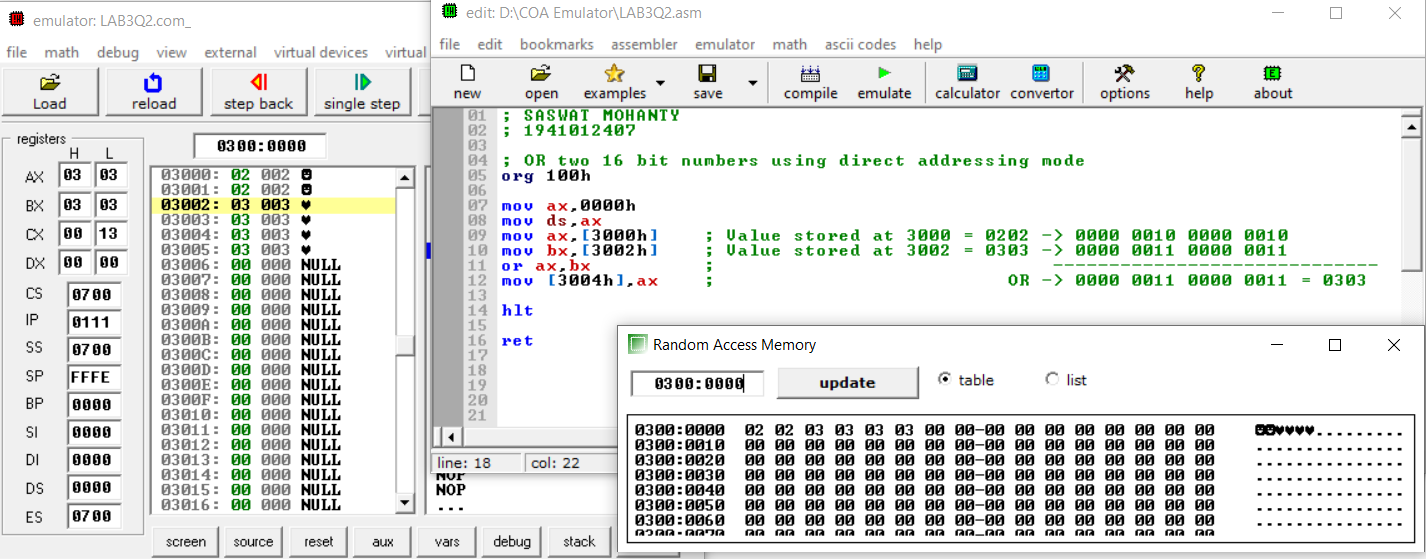
|  |
| --- |
| **; SASWAT MOHANTY**  **; 1941012407**  **; XOR of two 16 bit numbers using direct addressing mode**  **org 100h**  **mov ax,0000h**  **mov ds,ax**  **mov ax,[3000h] ; Value stored at 3000 = 0202 -> 0000 0010 0000 0010**  **mov bx,[3002h] ; Value stored at 3002 = 0303 -> 0000 0011 0000 0011**  **xor ax,bx ; -------------------------------------------------**  **mov [3004h],ax ; XOR -> 0000 0001 0000 0001 = 0101**    **hlt**  **ret** |

**Observations (with screen shots):**

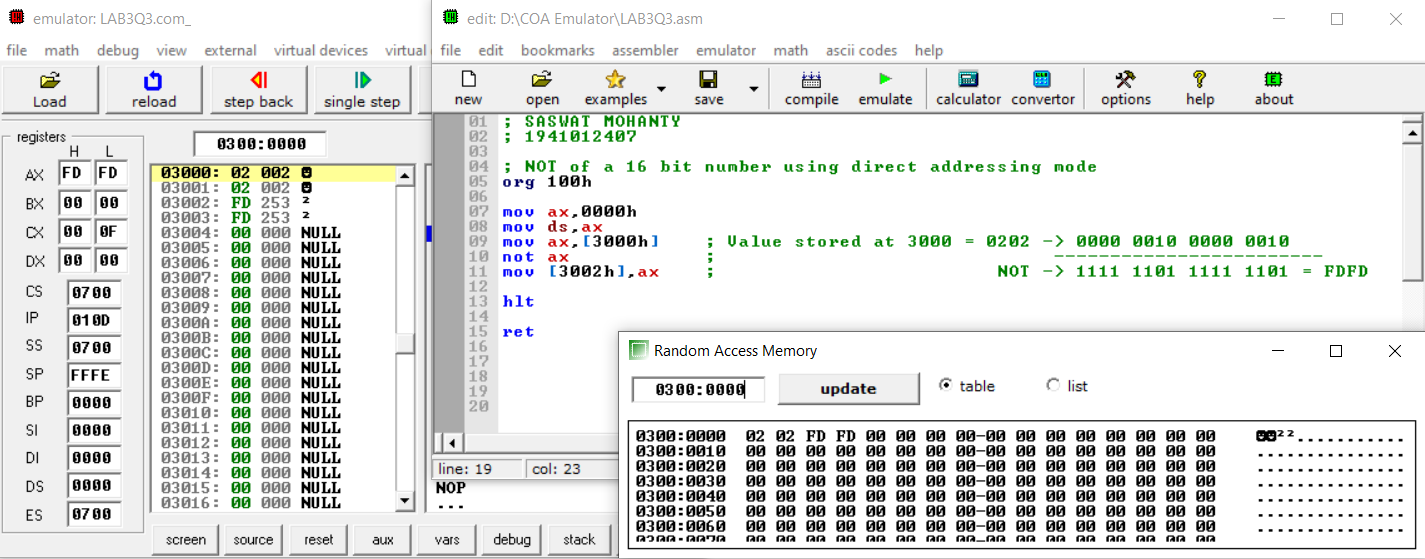
**For Obj. 1**

****

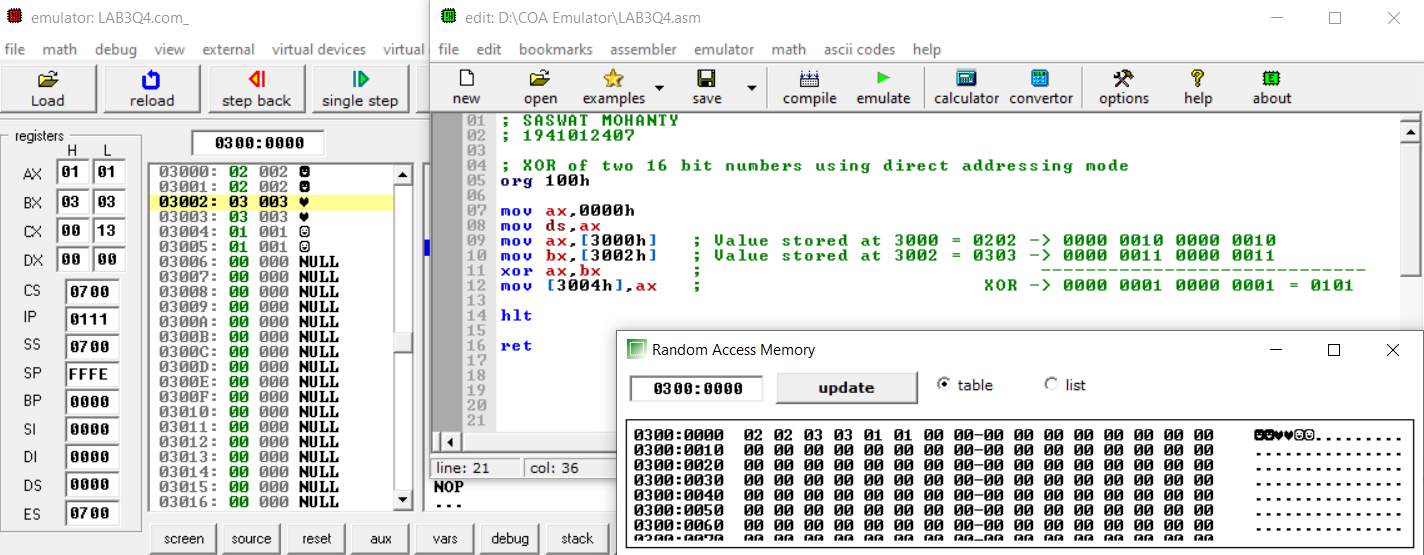
**For Obj. 2**

****

**For Obj. 3**

****

**For Obj. 4**

****

**Conclusion:**

**For Obj. 1:**

It can be concluded that the ‘and’ operation of numbers when dry run and executed in system found to be same. Thus, the program to and two 16-bit numbers was executed.

**For Obj. 2:**

It can be concluded that the ‘or’ operation of numbers when dry run and executed in system found to be same. Thus, the program to or two 16-bit numbers was executed.

**For Obj. 3:**

It can be concluded that the ‘not’ operation of numbers when dry run and executed in system found to be same. Thus, the program to not a 16-bit number was executed.

**For Obj. 4:**

It can be concluded that the ‘xor’ operation of numbers when dry run and executed in system found to be same. Thus, the program to xor two 16-bit numbers was executed.

1. **POST LAB:**

**Enlist the advantages of assembly language programming over machine language.**

* It allows complex jobs to run in a simpler way.
* It is memory efficient, as it requires less memory.
* It is faster in speed, as its execution time is less.
* It is mainly hardware-oriented.
* It requires less instruction to get the result.
* It is used for critical jobs.
* It is not required to keep track of memory locations.
* It is a low-level embedded system.

**Write the function of the following arithmetic instructions**

1. **ADC b) INC c) DEC d) SBB e) DAA**
2. **ADC: -** Used to add with carry.
3. **INC: -** Used to increment the provided byte/word by 1.
4. **DEC: -** Used to decrement the provided byte/word by 1.
5. **SBB: -** Used to perform subtraction with borrow.
6. **DAA: -** Used to adjust the decimal after the addition/subtraction operation.

**Write the function of the following logical instructions**

1. **SHL/SAL b) SHR c) SAR d) ROR e) ROL**
2. **SHL/SAL: -** Used to shift bits of a byte/word towards left and put zero(S) in LSBs.
3. **SHR: -** Used to shift bits of a byte/word towards the right and put zero(S) in MSBs.
4. **SAR: -** Used to shift bits of a byte/word towards the right and copy the old MSB into the new MSB.
5. **ROR: -** Used to rotate bits of byte/word towards the right, i.e., LSB to MSB and to Carry Flag [CF].
6. **ROL: -** Used to rotate bits of byte/word towards the left, i.e. MSB to LSB and to Carry Flag [CF].