**Computer Organization and Architecture (EET2211)**

**LAB I: Examine & Analyze Different Addressing Modes of 8086 Microprocessor**

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

|  |  |  |  |
| --- | --- | --- | --- |
| **Branch:** Computer Science and Engineering **Section:** ‘D’ | | | |
| **S. No.** | **Name** | **Registration No.** | **Signature** |
| 1 | Saswat Mohanty | 1941012407 | **D:\Pics and Sign\sign.jpg** |

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

**Remarks:**

**Teacher’s Signature**

**I. OBJECTIVE:**

1. Addition of two 16bit numbers using immediate addressing mode.
2. Addition of two 16bit numbers using direct addressing mode.
3. Addition of two 16bit numbers using indirect addressing mode.
4. Addition of two 16bit numbers using index addressing mode.
5. Addition of two 16bit numbers using base index addressing mode.

**II. PRE-LAB**

**For Obj. 1:**

1. **Explain immediate addressing mode briefly.**

The addressing mode in which the data operand is a part of the instruction itself is known as immediate addressing mode.

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

*MOV AX,2000H*

*MOV BX,9000H*

*ADD AX,BX*

Output = B000h

1. **Write the assembly code.**

|  |
| --- |
| **org 100h**  **MOV AX,2000H**  **MOV BX,9000H**  **ADD AX,BX**  **HLT**  **ret** |

**For Obj. 2:**

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

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 addition of two 16 bit numbers.**

*mov ax,[2000h]*

*mov bx,[9000h]*

*add ax,bx*

[2000h] = 1111h

[9000h] = 2222h

Output: [3004h] = 3333h

1. **Write the assembly code.**

|  |
| --- |
| **org 100h**  **MOV AX,0000H**  **MOV DS,AX**  **ADD AX,[2000H]**  **MOV BX ,[2100H]**  **ADD AX,BX**  **MOV [3004H],AX**  **hlt** |

**For Obj. 3:**

1. **Explain indirect addressing mode briefly.**

This addressing mode allows data to be addressed at any memory location through an offset address held in any of following registers BP, BX, DI and SI

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

*mov ax,[si]*

*mov bx,[si]*

*add ax,bx*

[20400h] = 1111h

[20402h] = 2222h

Output : [20404] = 3333h

1. **Write the assembly code.**

|  |
| --- |
| **org 100h**  **MOV AX,2000H**  **MOV DS,AX**  **MOV SI,0400H**  **MOV AX ,[SI]**  **INC SI**  **INC SI**  **MOV BX,[SI]**  **ADD AX,BX**  **INC SI**  **INC SI**  **MOV [SI],AX**  **hlt** |

**For Obj. 4:**

1. **Explain index addressing mode briefly.**

In this addressing mode, the operands offset address is found by adding the contents of SI or DI register and 8 bit/ 16 bit displacements

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

*mov ax,[si]*

*mov bx,[si+2]*

*add ax,bx*

[20700h] = 1111h

[20702h] = 2222h

Output: [20704] = 3333h

1. **Write the assembly code.**

|  |
| --- |
| **org 100h**  **MOV AX,2000H**  **MOV DS,AX**  **MOV SI,0700H**  **MOV AX,[SI+0]**  **MOV BX,[SI+2]**  **ADD AX,BX**  **MOV [SI+4],AX**  **HLT** |

**For Obj. 5:**

1. **Explain base index addressing mode briefly.**

In this addressing mode, the offset address of the operand is computed by summing the base register to the contents of an Index register.

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

*mov ax,[bx+si]*

*mov cx,[bx+si]*

*add ax,cx*

[0000h] = 1111h

[3500h] = 2222h

[3502h] = 3333h

Output: [3504] = 5555h

1. **Write the assembly code.**

|  |
| --- |
| **org 100h**  **MOV AX,0000H**  **MOV DS,AX**  **MOV BX,3000H**  **MOV SI,0500H**  **MOV CX,[BX+SI]**  **MOV DX,[BX+SI+02]**  **MOV AX,CX**  **ADD AX,DX**  **HLT** |

**III. LAB:**

**Assembly Program:**

**For Obj. 1**

|  |
| --- |
| **; SASWAT MOHANTY**  **; 1941012407**  **; Addition of two 16bit numbers using immediate addressing mode**  **org 100h**  **MOV AX,2000H**  **MOV BX,9000H**  **ADD AX,BX**  **HLT**  **ret** |

**For Obj. 2**

|  |
| --- |
| **; SASWAT MOHANTY**  **; 1941012407**  **; Addition of two 16bit numbers using direct addressing mode**  **org 100h**  **MOV AX,0000H**  **MOV DS,AX**  **ADD AX,[2000H] ; value stored at 2000 = 1111**  **MOV BX ,[2100H] ; value stored at 2100 = 2222**  **ADD AX,BX**  **MOV [3004H],AX**  **hlt**  **ret** |

**For Obj. 3**

|  |
| --- |
| **; SASWAT MOHANTY**  **; 1941012407**  **; Addition of two 16bit numbers using indirect addressing mode**  **org 100h**  **MOV AX,2000H**  **MOV DS,AX**  **MOV SI,0400H**  **MOV AX ,[SI] ; value stored at 20400 = 1111**  **INC SI ; value stored at 20402 = 2222**  **INC SI**  **MOV BX,[SI]**  **ADD AX,BX**  **INC SI**  **INC SI**  **MOV [SI],AX**  **hlt**  **ret** |

**For Obj. 4**

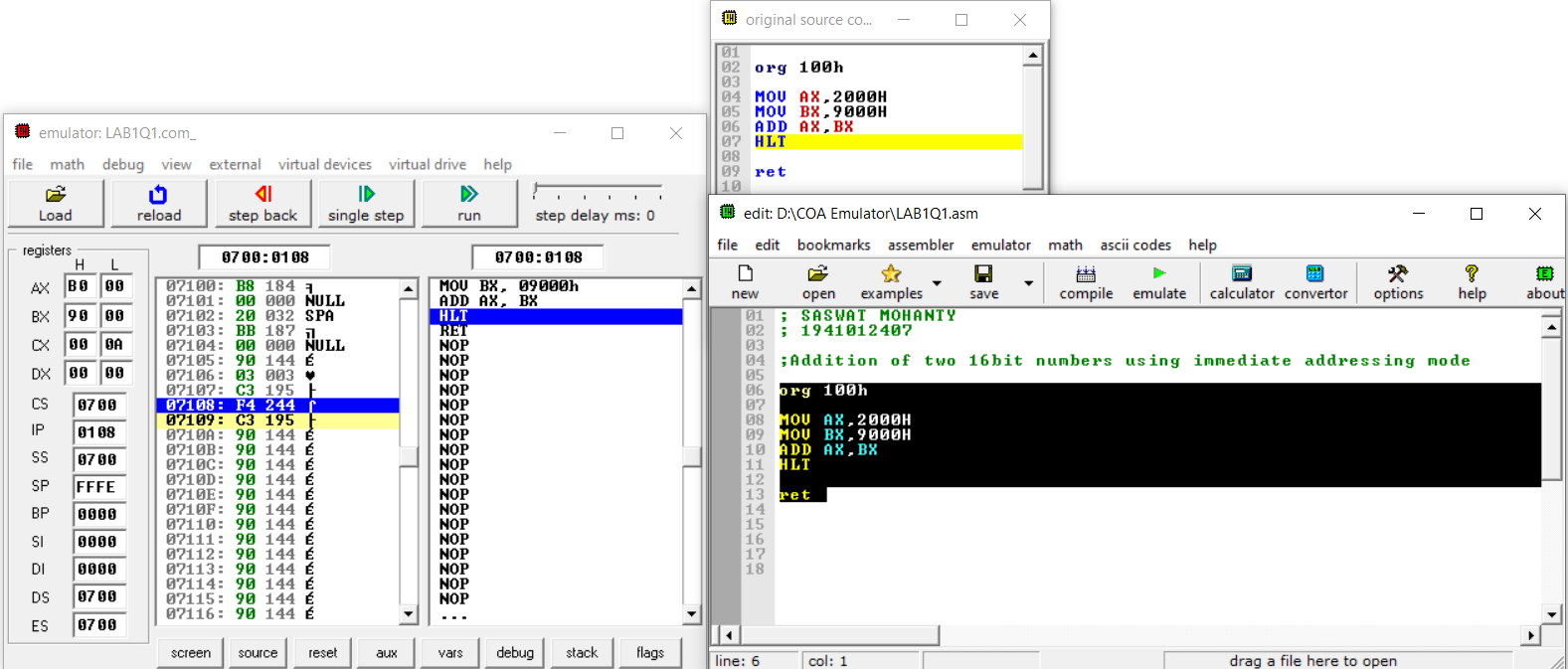
|  |
| --- |
| **; SASWAT MOHANTY**  **; 1941012407**  **; Addition of two 16bit numbers using index addressing mode**  **org 100h**  **MOV AX,2000H**  **MOV DS,AX**  **MOV SI,0700H ;VALUES STORED AT 20700 = 1111**  **MOV AX,[SI+0] ;VALUES STORED AT 20702 = 2222**  **MOV BX,[SI+2]**  **ADD AX,BX**  **MOV [SI+4],AX**  **HLT**  **ret** |

**For Obj. 5**

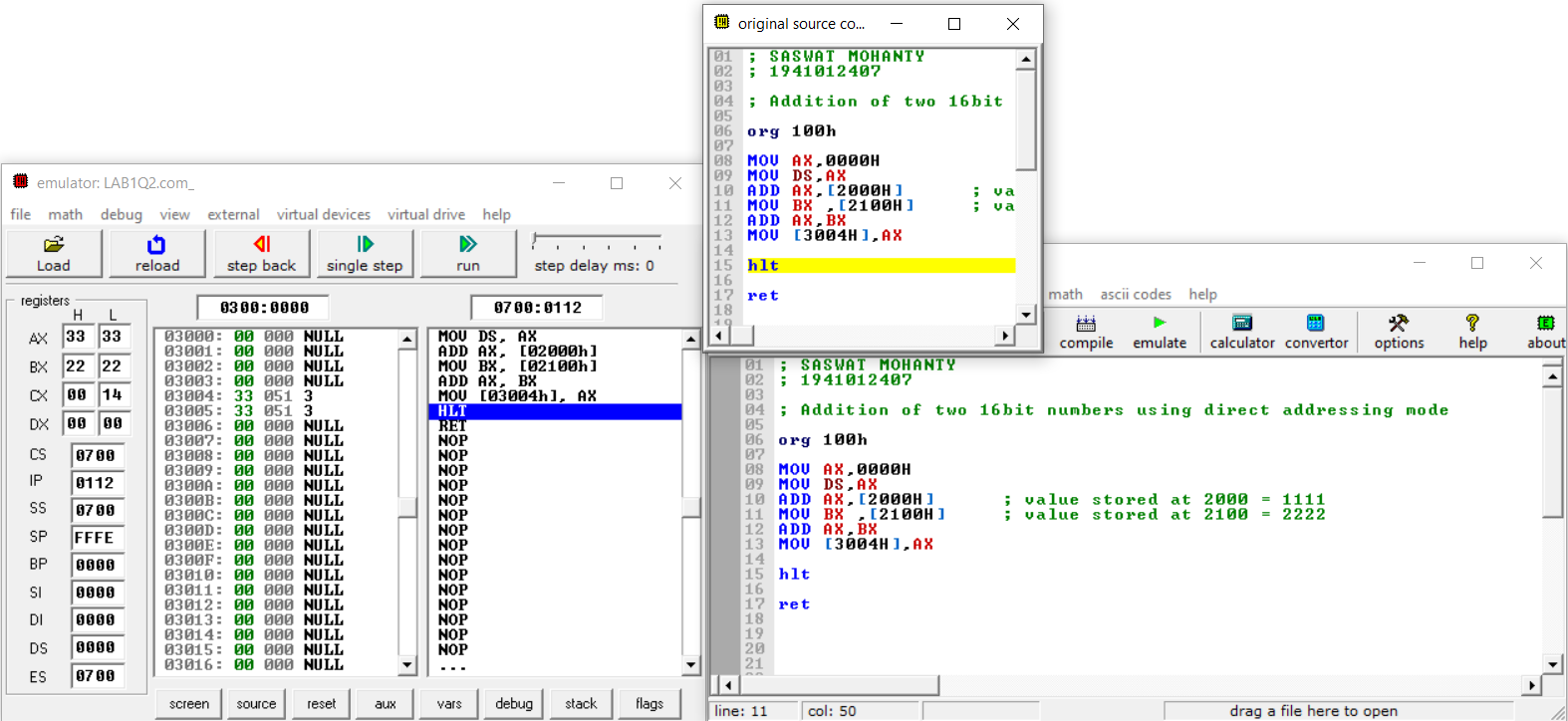
|  |
| --- |
| **; SASWAT MOHANTY**  **; 1941012407**  **; Addition of two 16bit numbers using base index addressing mode**  **org 100h**    **MOV AX,0000H ;value stored at 0000 = 1111**  **MOV DS,AX**  **MOV BX,3000H**  **MOV SI,0500H ;value stored at 3500 = 2222**  **MOV CX,[BX+SI] ;value stored at 3502 = 3333**  **MOV DX,[BX+SI+02]**  **MOV AX,CX**  **ADD AX,DX**  **HLT**  **ret** |

**Observations (with screen shots):**

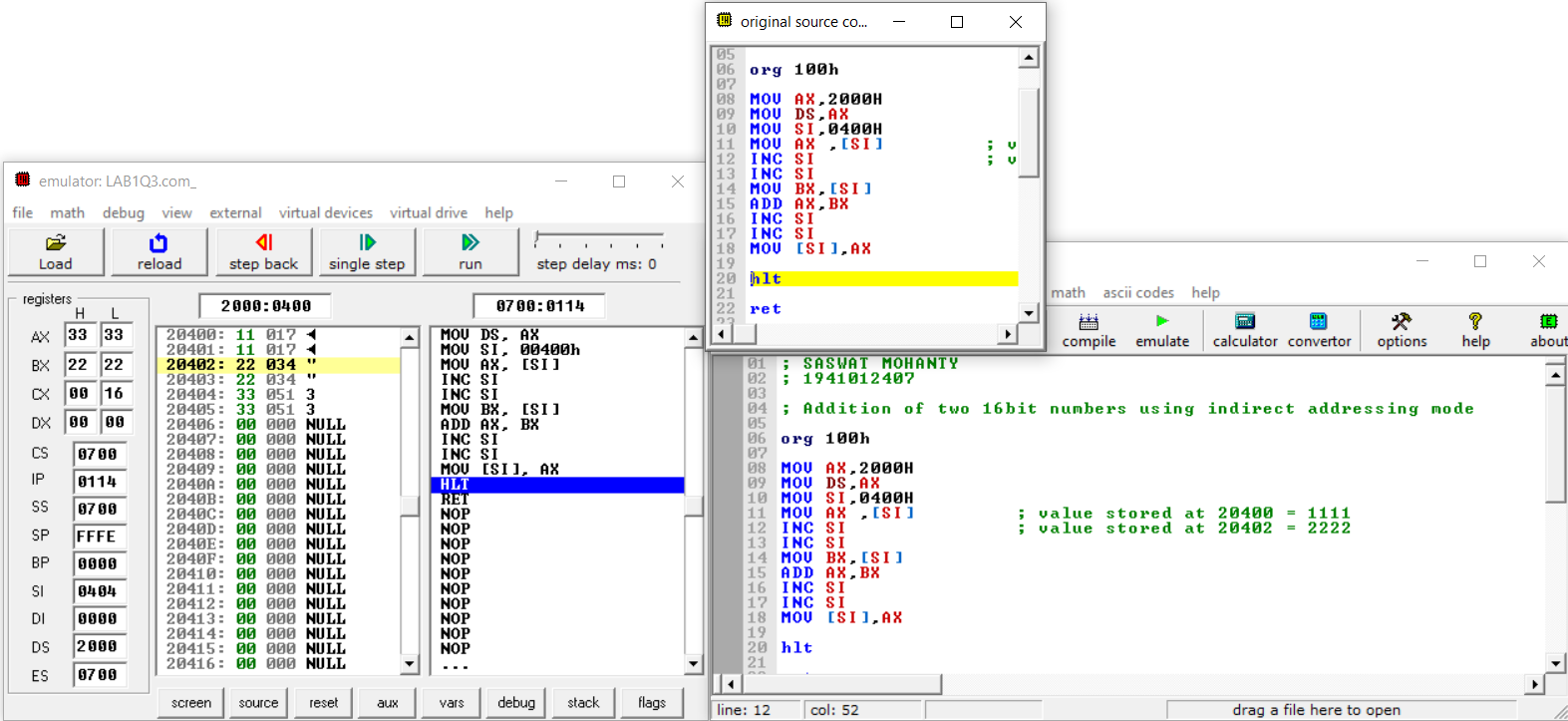
**For Obj. 1**

****

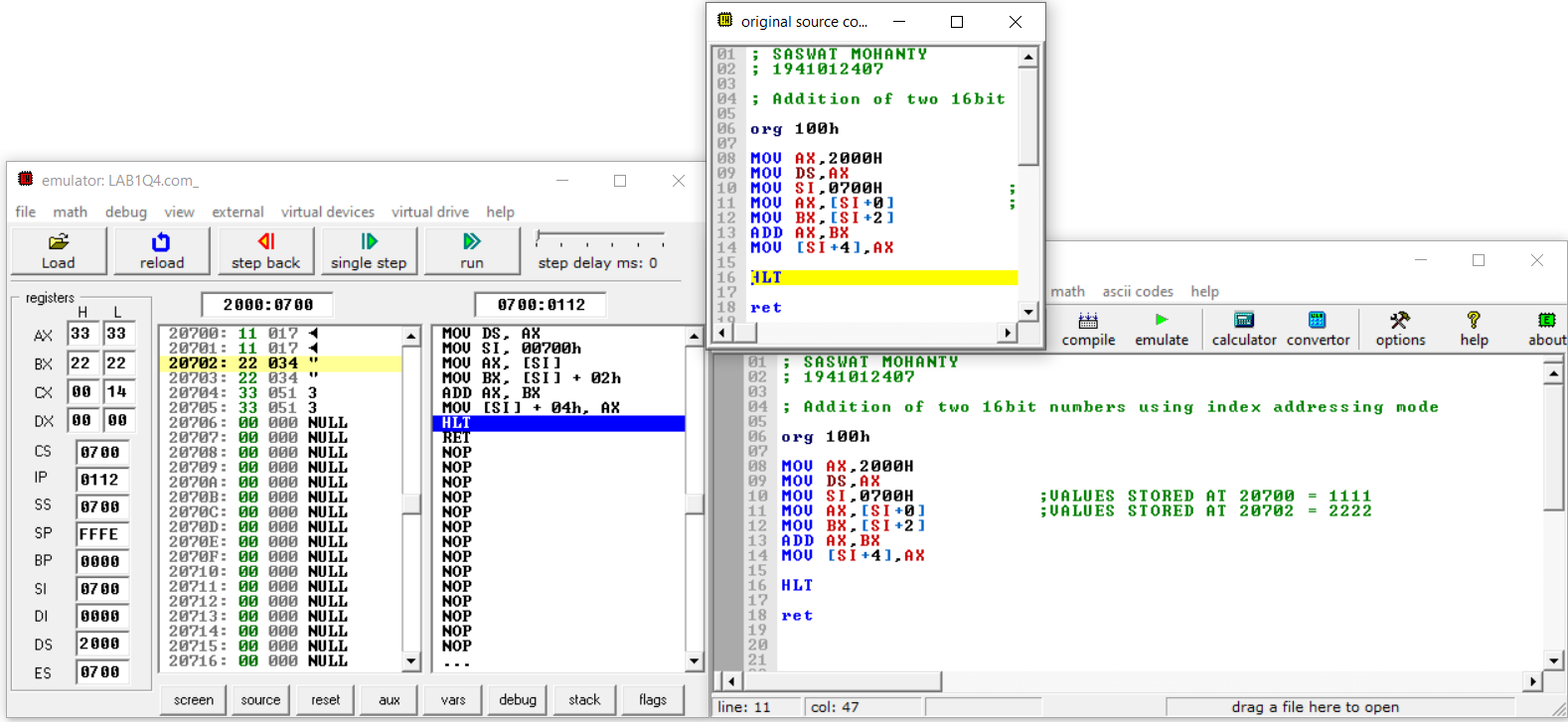
**For Obj. 2**

****

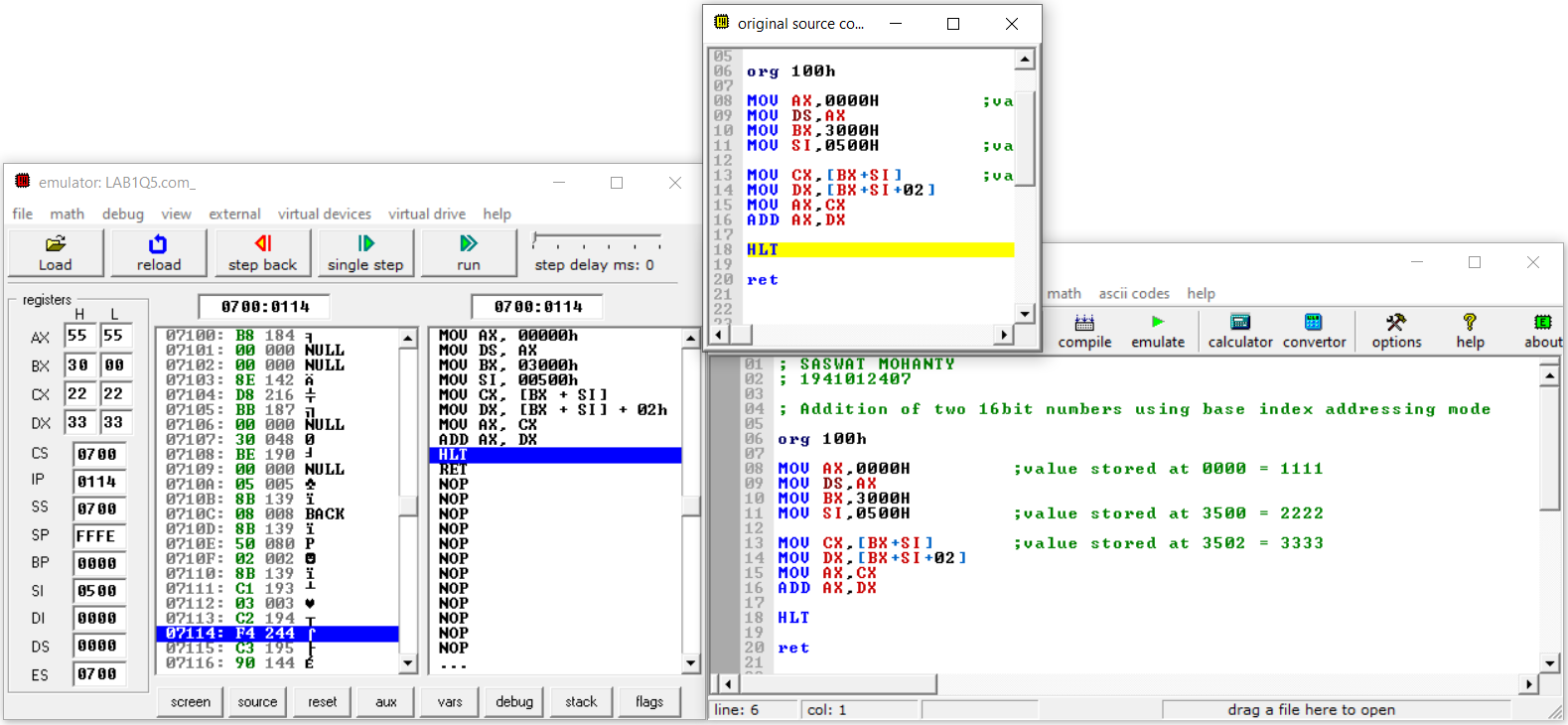
**For Obj. 3**

****

**For Obj. 4**

****

**For Obj. 5**

****

**Conclusion:**

**For Obj. 1:**

It can be concluded that for immediate addressing the operand is specified in the instruction itself.

**For Obj. 2:**

It can be concluded that for direct addressing the operands offset is given in the instruction as a 16-bit displacement element.

**For Obj. 3:**

It can be concluded that for indirect addressing the operands offset is placed SI register as specified in the instruction.

**For Obj. 4:**

It can be concluded that for index addressing the offset is the sum of the content of SI register and a 16-bit displacement element.

**For Obj. 5:**

It can be concluded that for base index addressing the offset is the sum of the content of BX and SI register.

**IV. POST LAB:**

**Discuss different general-purpose registers used in 8086 microprocessor.**

EU has 8 general purpose registers; two registers can also be combined to form 16-bit registers. The valid register pairs are

* **AX (AL, AH):** Word multiply, word divide, word I/O
* **BX (BL, BH):** Store address information
* **CX (CL, CH):** String operation, loops
* **DX (DL, DH):** Word multiply, word divide, indirect I/O (used to hold I/O address during I/O instructions If the result is more than 16 bits, the lower order 16 bits are stored in accumulator and higher order 6 bits are stored in DX register)

**Explain the concept of segmented memory. What are its advantages?**

Segmentation is the process in which the main memory of the computer is divided into different segments and each segment has its own base address. It is basically used to enhance the speed of execution of the computer system, so that processor is able to fetch and execute the data from the memory easily and fast.

The main advantages of segmentation memory are as follows:

1. It provides a powerful memory management mechanism.
2. Data related or stack related operations can be performed in different segments.
3. Code related operation can be done in separate code segments.
4. It allows to processes to easily share data.
5. It allows extending the address ability of the processor, i.e., segmentation allows the use of 16-bit registers to give an addressing capability of 1 Megabytes. Without segmentation, it would require 20-bit registers.
6. It is possible to enhance the memory size of code data or stack segments beyond 64 KB by allotting more than one segment for each area.

**Explain the physical address formation in 8086.**

Physical Address = Base Address \* 10H + Offset

**Write a program to add two 16 bit numbers 12H and 08H, and store the sum.**

*org 100h*

*mov ax,0012h*

*mov bx,0008h*

*add ax,bx*

*hlt*