

EXPERIMENT--01-ALP-FOR-8086

Name : Sana Fathima H
Register no :212223240145
Date of experiment :19-08-2025

Aim: To Write and execute ALP on fundamental arithmetic and logical operations

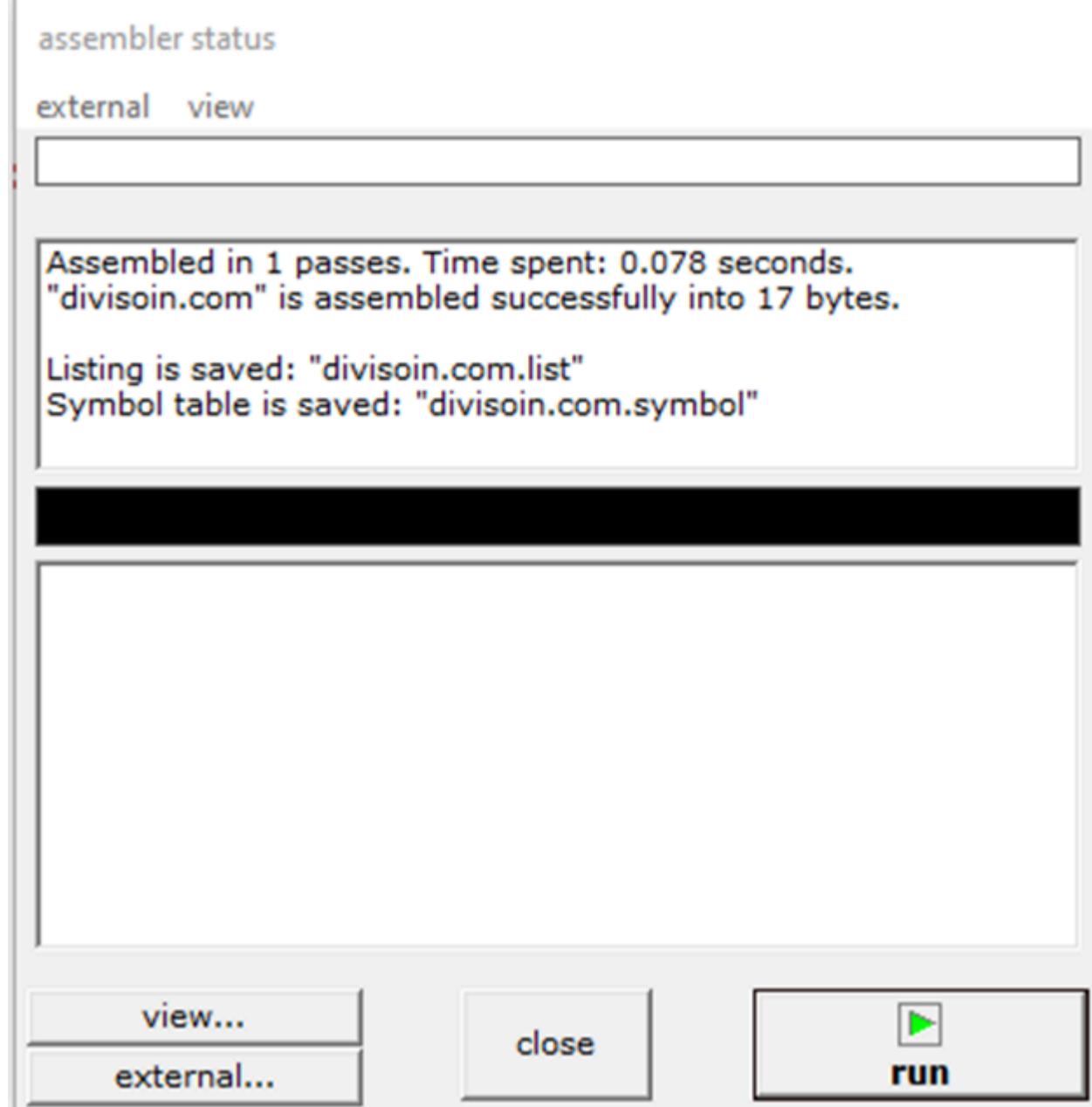
Components required: 8086 emulator

Theory

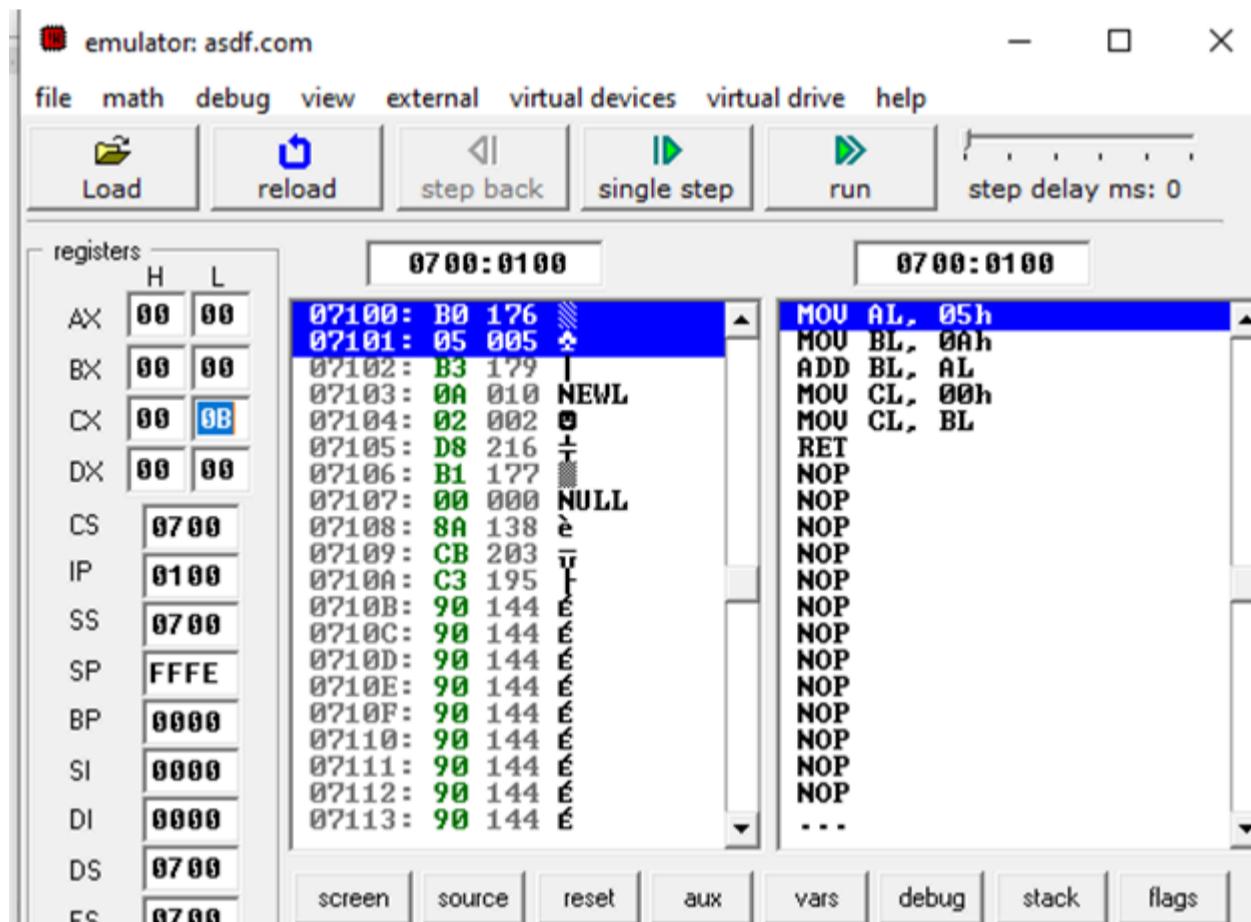
Running The Emulator (emu8086) Intro 8086 Microprocessor Emulator, also known as EMU8086, is an emulator of the program 8086 microprocessor. It is developed with a built-in 8086 assembler. This application is able to run programs on both PC desktops and laptops. This tool is primarily designed to copy or emulate hardware. These include the memory of a program, CPU, RAM, input and output devices, and even the display screen. There are instructions to follow when using this emulator. It can be executed into one of the two ways: backward or forward. There are also examples of assembly source code included. With this, it allows the programming of assembly language, reverse engineering, hardware architecture, and creating miniature operating system (OS). The user interface of 8086 Microprocessor Emulator is simple and easy to manage. There are five major buttons with icons and titles included. These are "Load", "Reload", "Step Back", "Single Step", and "Run". Above those buttons is the menu that includes "File", "View", "Virtual Devices", "Virtual Drive", and "Help". Below the buttons is a series of choices that are usually in numbers and codes. At the leftmost part is an area called "Registers" with an indication of either "H" or "L". The other side is divided into two, which enables users to manually reset, debug, flag, etc. What is 8086 emulator emu8086 is an emulator of Intel 8086 (AMD compatible) microprocessor with integrated 8086 assembler and tutorials for beginners. Emulator runs programs like the real microprocessor in step-by-step mode. it shows registers, memory, stack, variables and flags.

Running the Emulator :

1. Download and install emu8086 (www.emu8086.com) It is usually installed in C:\EMU8086 subfolder in the "Windows" directory
2. Run emu8086 icon (on the desktop or in the c:\EMU8086 folder of window) It has green color
3. write the code for the appropriate program for ADDITION,SUBTRACTION, MULTIPLICATION, DIVISION operations
4. Compile the program and check for the errors
5. Run (once there is no syntax error)
6. Click OK to see/view the output of your program on the Emulator screen.
7. After running the program, another menu screen will be displayed, where you have the option to "View" symbol table,
- 8.



9. Click on emulate to start emulation



10. If no errors are found click on run the program and check the status of various flags in the flags tab as shown below



Programs for arithmetic operations

Addition of 8 bit ALP

```
org 100h
mov ax,7db3h
mov bx,5adbh
add ax,bx
ret
```

Output

The screenshot shows the debugger interface with the following details:

- Assembly Window:** Displays the assembly code:


```
01 ; You may customize this
02 ; The location of this t
03
04 org 100h
05
06 mov ax,7db3h
07
08 mov bx,5adbh
09
10 add ax,bx
11
12 ret
```
- Registers Window:** Shows register values:

	H	L
AX	D8	8E
BX	5A	DB
CX	00	09
DX	00	00
CS	F400	
IP	0154	
SS	0700	
SP	FFFA	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	
- Memory Dump Window:** Shows memory dump starting at address F400:0154.
- Flags Window:** Shows the current state of CPU flags:

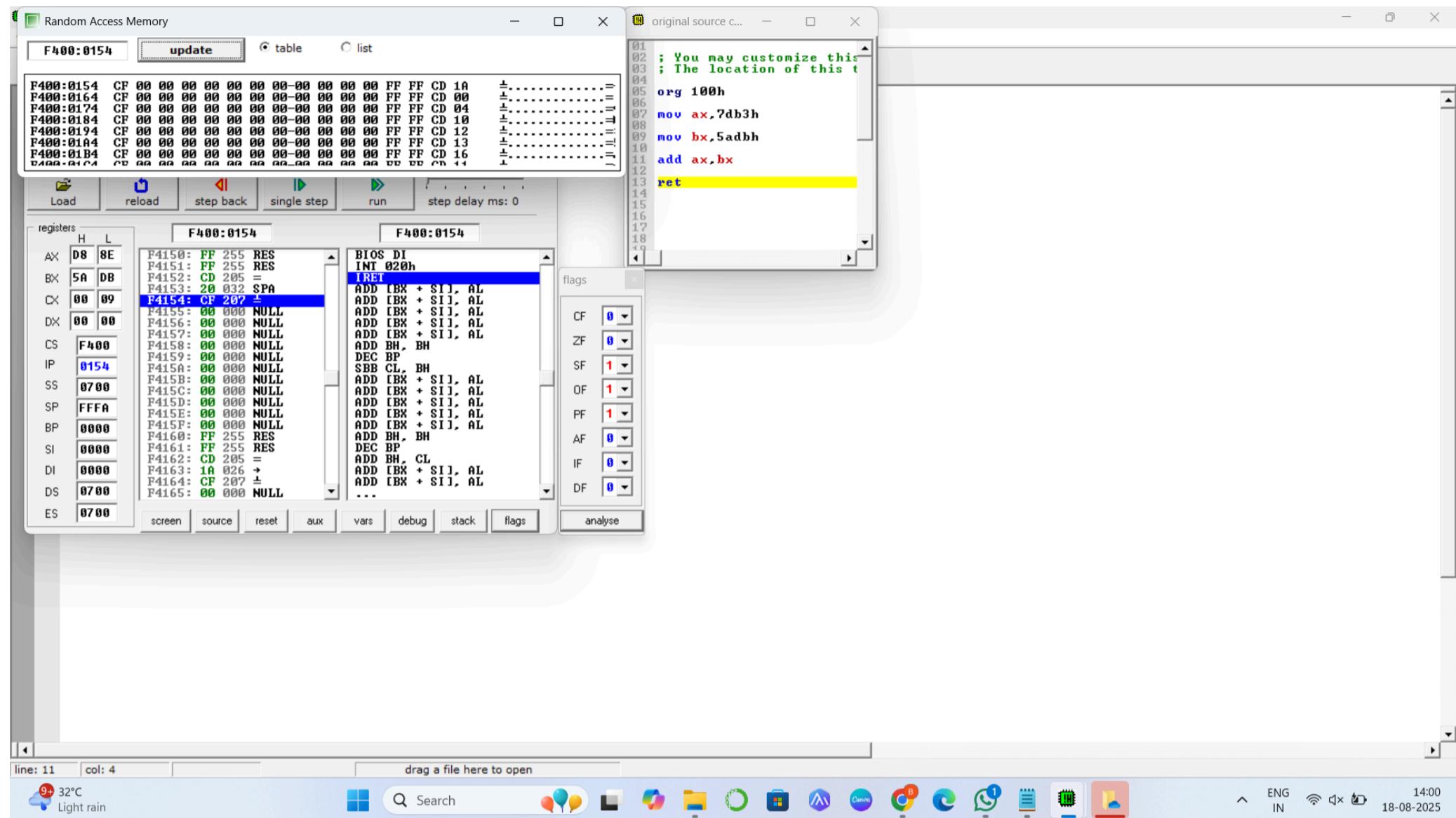
CF	0
ZF	0
SF	1
OF	1
PF	1
AF	0
IF	0
DF	0
- Status Bar:** Shows line: 11 col: 4, 9: 32°C Light rain, ENG IN, 14:00, 18-08-2025.

Subtraction of 8 bit numbers ALP

```
org 100h
mov ax,7db3h
```

```
mov bx,5adbh
sub ax,bx
ret
```

Output



Multiplication alp

```
org 100h
mov ax,93b4h
mov bx,5adbh
mul bx
ret
```

Output

The screenshot shows the MARS 8086 emulator interface. On the right, the assembly source code is displayed:

```

; You may customize this
; The location of this t
01 org 100h
02 mov ax,93b4h
03 mov bx,5adbh
04 mul bx
05 ret

```

The assembly code consists of five instructions: `org 100h`, `mov ax,93b4h`, `mov bx,5adbh`, `mul bx`, and `ret`. The `ret` instruction is highlighted in yellow.

On the left, the memory dump window shows the state of memory starting at address F400:0154. The AX register is set to 93b4h, and the BX register is set to 5adbh. The stack pointer (SP) is at F400:0154, and the stack grows downwards.

The registers window shows the following values:

Register	Value
AX	A2 FC
BX	5A DB
CX	88 09
DX	34 6B
CS	F400
IP	0154
SS	0700
SP	FFFFA
BP	0000
SI	0000
DI	0000
DS	0700
ES	0700

The flags window shows the following flag states:

- CF: 1
- ZF: 0
- SF: 0
- OF: 1
- PF: 0
- AF: 0
- IF: 0
- DF: 0

Division alp

```

org 100h
mov ax,71b4h
mov bx,43ebh
div ax
ret

```

Output

The screenshot shows the MARS 8086 emulator interface. On the right, the assembly source code is displayed:

```

; You may customize this
; The location of this t
01 org 100h
02 mov ax,71b4h
03 mov bx,43ebh
04 div ax
05 ret

```

The assembly code consists of five instructions: `org 100h`, `mov ax,71b4h`, `mov bx,43ebh`, `div ax`, and `ret`. The `ret` instruction is highlighted in yellow.

On the left, the memory dump window shows the state of memory starting at address F400:0154. The AX register is set to 71b4h, and the BX register is set to 43ebh. The stack pointer (SP) is at F400:0154, and the stack grows downwards.

The registers window shows the following values:

Register	Value
AX	00 01
BX	43 EB
CX	00 09
DX	00 00
CS	F400
IP	0154
SS	0700
SP	FFFFA
BP	0000
SI	0000
DI	0000
DS	0700
ES	0700

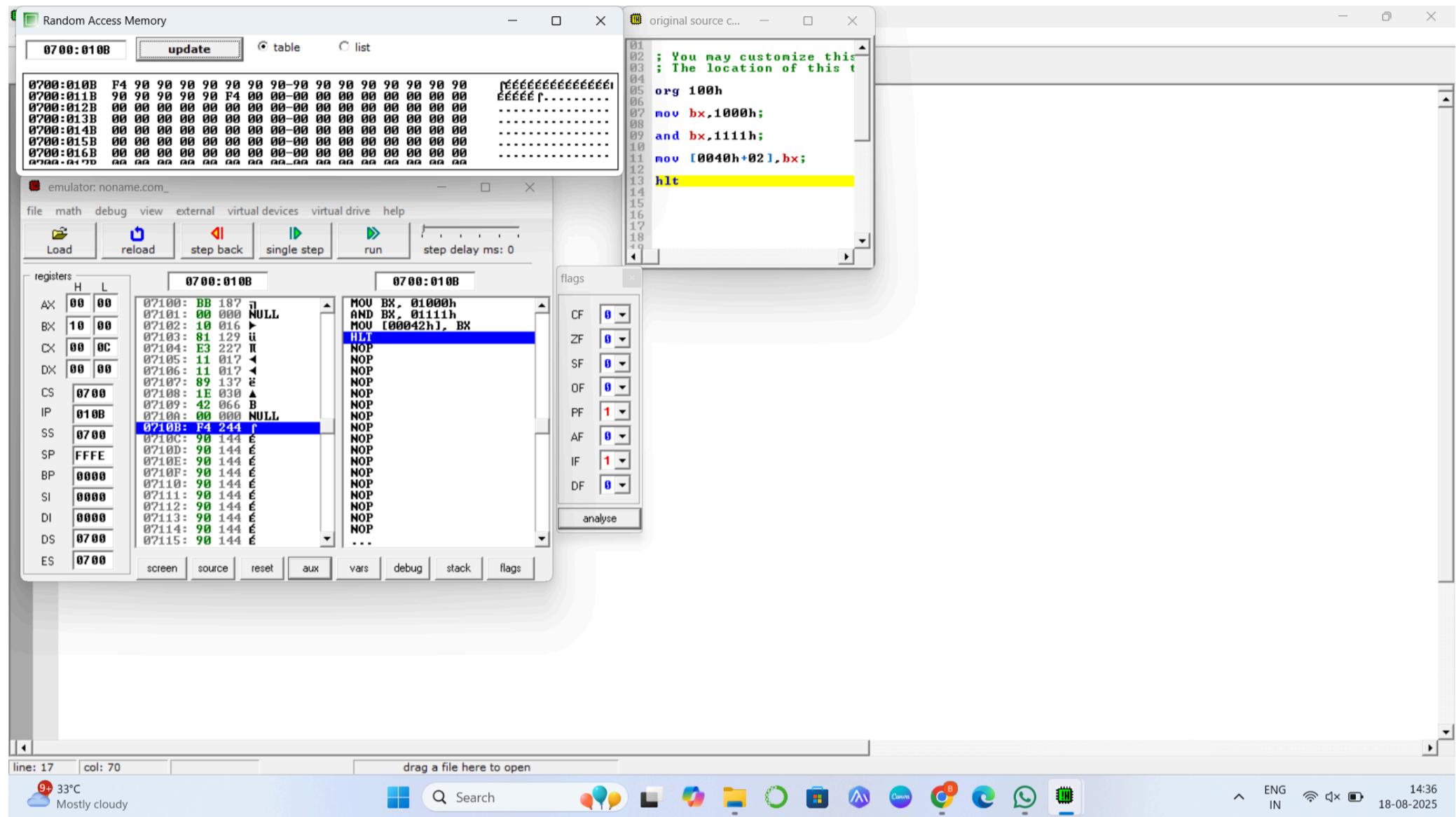
The flags window shows the following flag states:

- CF: 0
- ZF: 0
- SF: 0
- OF: 0
- PF: 0
- AF: 0
- IF: 0
- DF: 0

AND

```
org 100h
mov bx,1000h;
and bx,1111h;
mov [0040h+02],bx;
hlt
```

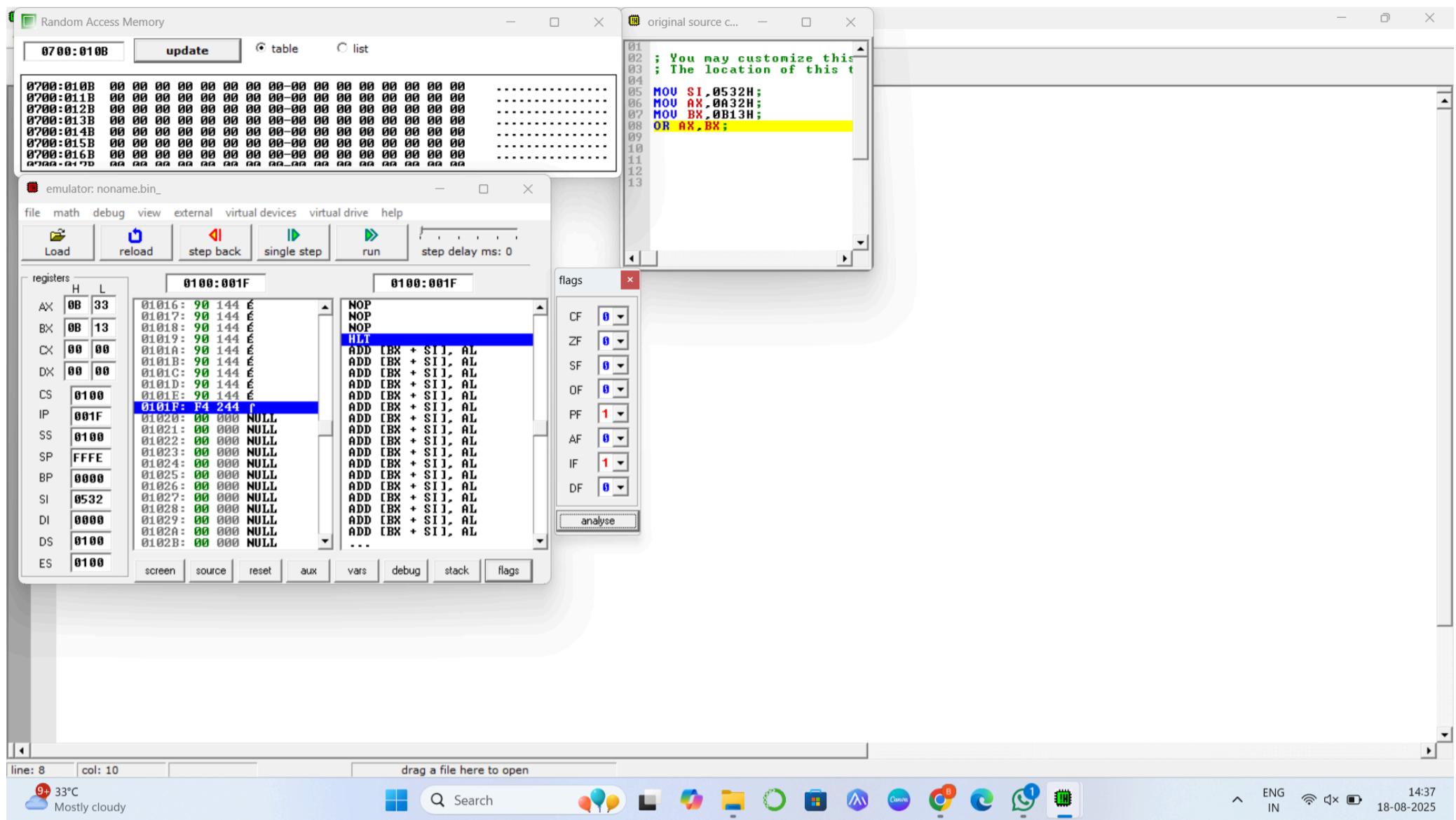
Output:



OR

```
MOV SI,0532H;
MOV AX,0A32H;
MOV BX,0B13H;
OR AX,BX;
```

Output:



NOT

```
org 100h
mov bx,0040h;
mov ax,[bx];
not al;
mov [0040h+04],ax;
hlt
```

Output:

The screenshot shows the MARS debugger interface. On the right, the assembly source code window displays:

```

01; You may customize this
02; The location of this t
03
04 org 100h
05 mov bx,0040h;
06 mov ax,[bx];
07 not al;
08 mov [0040h+04],ax;
09 hlt
10
11
12
13
14

```

The assembly code window has line 10 highlighted in yellow. On the left, the memory dump window shows the memory starting at address 0700:010A filled with FF (hex) values. The registers window shows the following values:

Register	Value
AX	00 FF
BX	00 40
CX	00 0B
DX	00 00
CS	0700
IP	010A
SS	0700
SP	FFFE
BP	0000
SI	0000
DI	0000
DS	0700
ES	0700

The instruction pointer (IP) is at 010A, and the current instruction is F4 244 f. The flags register shows:

- CF: 0
- ZF: 0
- SF: 0
- OF: 0
- PF: 0
- AF: 0
- IF: 1
- DF: 0

XOR

```

MOV [SI+2],AX;
MOV AX,0A32H;
MOV BX,0B13H;
XOR AX,BX;

```

Output:

The screenshot shows the MARS debugger interface. On the right, the assembly source code window displays:

```

01; You may customize this
02; The location of this t
03
04 MOV [SI+2],AX;
05 MOV AX,0A32H;
06 MOV BX,0B13H;
07 XOR AX,BX;
08
09

```

The assembly code window has line 07 highlighted in yellow. On the left, the memory dump window shows the memory starting at address 0100:001F filled with FF (hex) values. The registers window shows the following values:

Register	Value
AX	00 00
BX	00 00
CX	00 00
DX	00 00
CS	0100
IP	001F
SS	0100
SP	FFFE
BP	0000
SI	0000
DI	0000
DS	0100
ES	0100

The instruction pointer (IP) is at 001F, and the current instruction is F4 244 f. The flags register shows:

- CF: 0
- ZF: 1
- SF: 0
- OF: 0
- PF: 1
- AF: 0
- IF: 1
- DF: 0

Result :

Thus, to write and execute ALP on fundamental arithmetic operations and Logical operations is successful.