BAI-4A COAL Task 5

Tazmeen Afroz
Roll No: 22P-9252
Section: BAI-4A

March 12, 2024

1 Introduction

This document explores the setting of flags in assembly programming, focusing on the Carry Flag (CF), Parity Flag (PF), Zero Flag (ZF), Auxiliary Carry Flag (AF), Overflow Flag(OF) and Sign Flag (SF), through a practical example.

2 Code Section

```
[org 0x0100]

    mov  ax, [num1]         ; load first number in ax

    mov  bx, [num2]
    add  ax, bx
    mov  ax, 0x4c00
    int  0x21


num1: dw   65535
num2: dw   1
```

3 State of Flags Before Code Execution

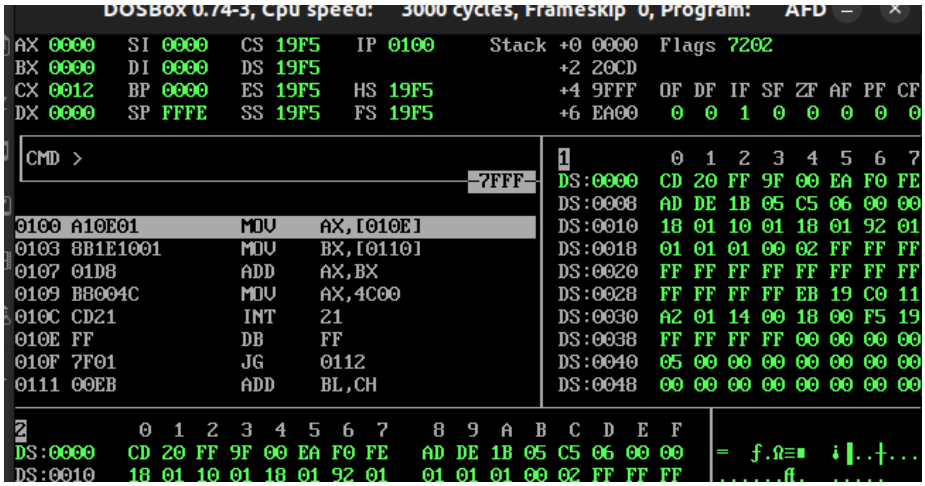


Figure 1: State of Flags Before Code Execution

4 State of Flags After Code Execution

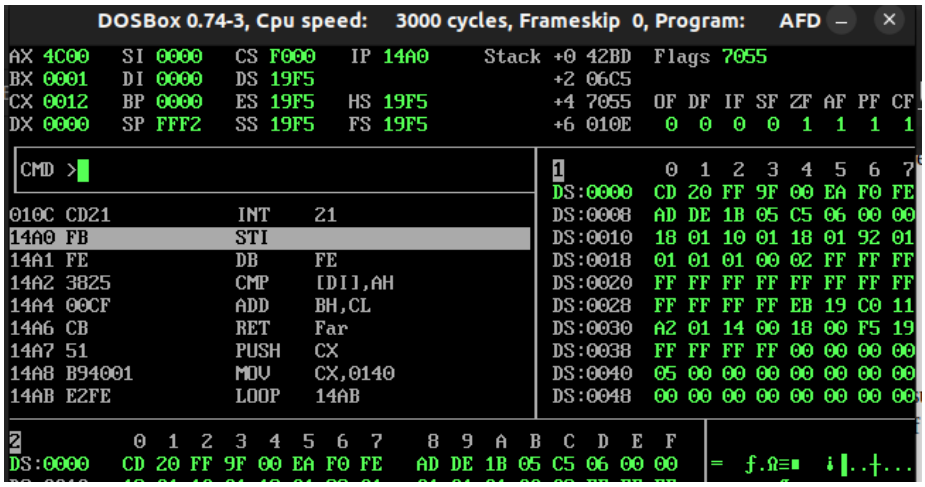


Figure 2: State of Flags After Code Execution

5 Reasons

5.1 Carry Flag (CF)

This flag is set to 1 when there is a carry out from the most significant bit (MSB) during addition or subtraction. Since num1 is 65535 (the maximum value a 16-bit number can hold) and num2 is 1, adding them together results in a value that requires 17 bits to represent. Since AX can only hold 16 bits, the extra bit is placed in the carry flag.

5.2 Parity Flag (PF)

This flag indicates the parity of the result of the last operation. Parity is either odd or even, based on the number of 1-bits in the binary representation of the result. Since the result of the addition is 0 which in binary is 0000, the parity flag is set to 1, indicating an even number of 1-bits. (0 is even number)

5.3 Zero Flag (ZF)

This flag is set if the result of the last operation is zero. Since the addition of num1 and num2 results in 0, the zero flag is set to 1.

5.4 Auxiliary Carry Flag (AF)

num1 (65535) in binary is 1111111111111111. num2 (1) in binary is 0000000000000001. The lower nibble of num1 is 1111 (15 in decimal), and the lower nibble of num2 is 0001 (1 in decimal). When we add these together:

```
  1111 (num1 lower nibble)
+ 0001 (num2 lower nibble)
------
  10000 (Result)
```

The result is 10000, which is 16 in decimal. Notice that the result requires 5 bits, but we only have 4 bits available for the lower nibble. This means there is a carry of 1 to the higher nibble.

The Auxiliary Carry Flag (AF) is set to 1 because there was a carry from the lower nibble to the higher nibble during the addition. The addition of the lower nibbles of num1 and num2 resulted in a carry. This process is what triggers the AF to be set to 1.

6 Code Section

```
[org 0x0100]

    mov  ax, [num1]         ; load first number in ax
```

```
    mov  bx, [num2]
    add  ax, bx
    mov  ax, 0x4c00
    int  0x21


num1: dw   32767
num2: dw   1
```

7 State of Flags Before Code Execution

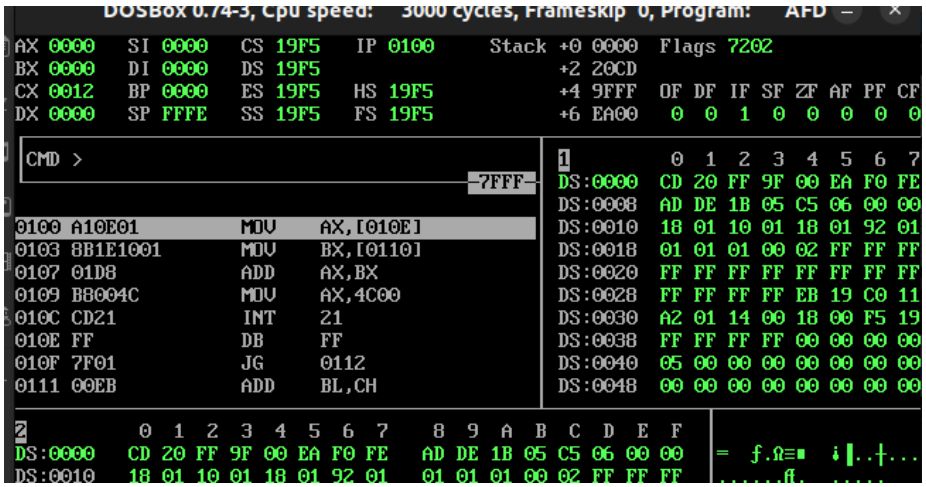


Figure 3: State of Flags Before Code Execution

8 State of Flags After Code Execution

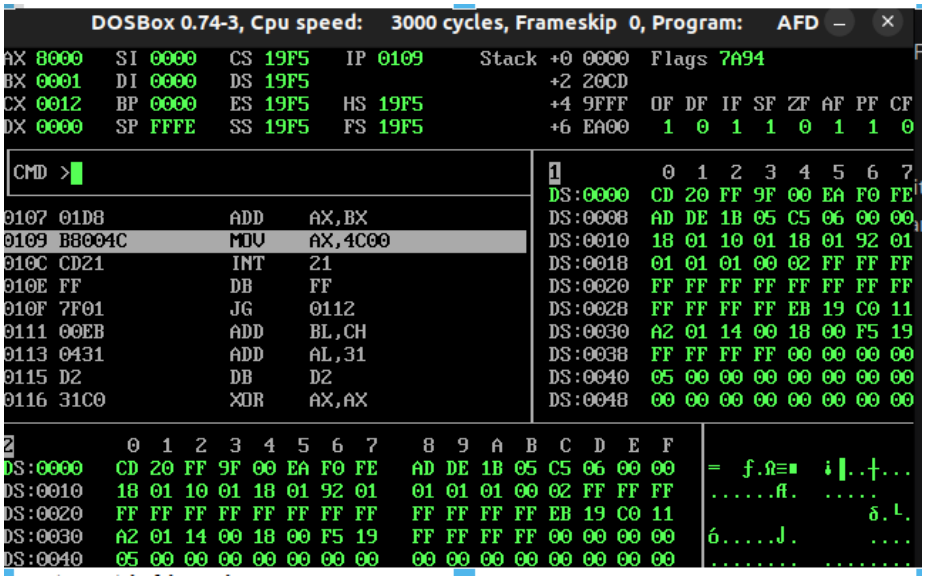


Figure 4: State of Flags After Code Execution

9 Reasons

9.1 Sign Flag (SF)

num1 (32767) in binary is 0111111111111111. num2 (1) in binary is 0000000000000001. Addition Operation When we add num1 and num2:

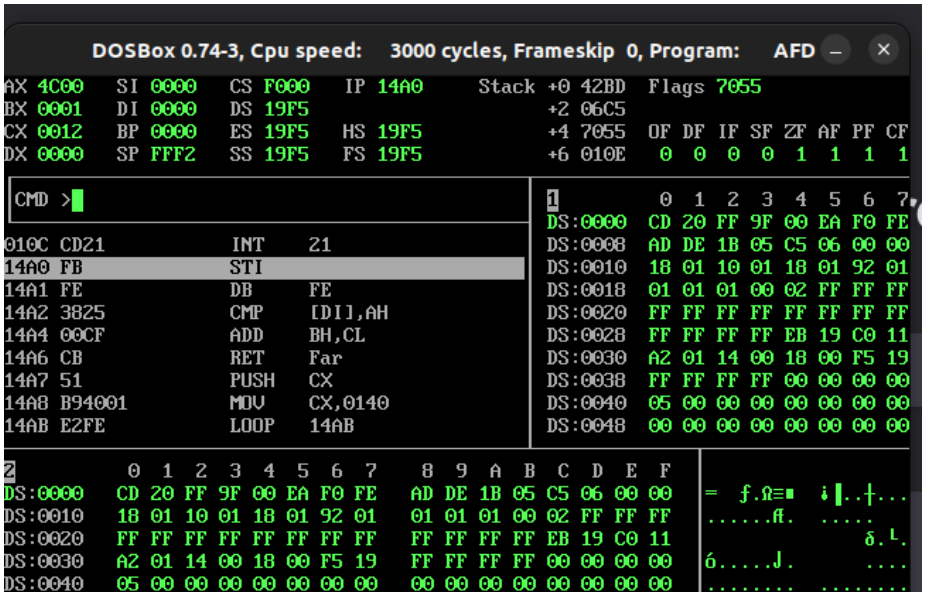
```
  0111111111111111 (num1)
+ 0000000000000001 (num2)
------------------
  1000000000000000 (Result)
```

Sign Flag (SF): The most significant bit (MSB) of the result is 1, indicating a negative number in two's complement representation. Therefore, the SF is set to 1.

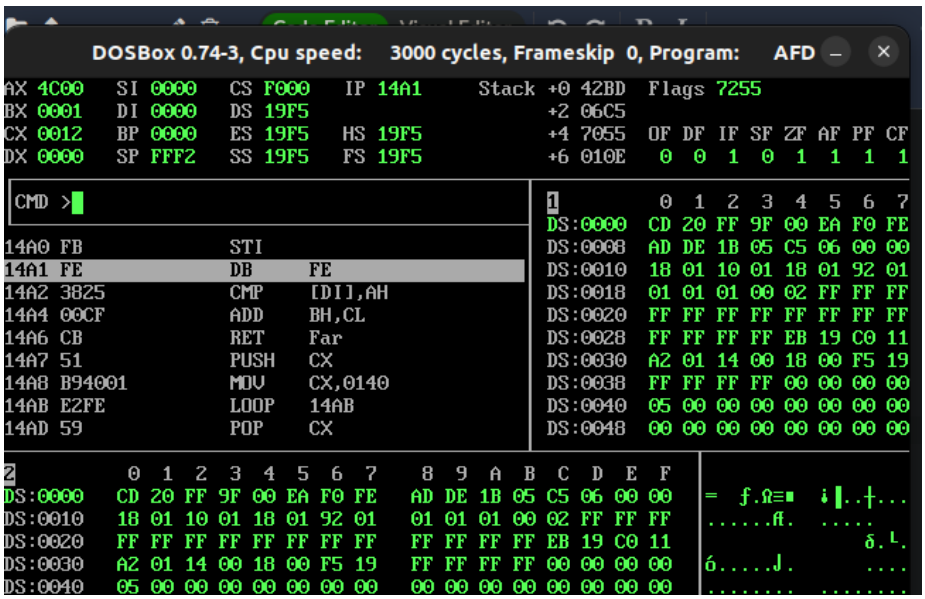
9.2 Overflow Flag (OF)

The overflow flag is set during signed arithmetic when the sign of the destination changes unexpectedly.The addition of num1 and num2 results in a negative number, which is unexpected since both num1 and num2 are positive. The carry into the MSB is 1 (from the addition), and the carry out of the MSB is 0 (since the result is negative). Therefore, the OF is set to 1.

9.3 Interrupt Flag (IF)



After 'int 0x21', the IF is cleared (set to 0) by the interrupt handler. To re-enable interrupts, 'sti' sets IF back to 1, allowing the processor to be interrupted again.



9.4 Direction Flag (DF)

The Direction Flag (DF) in x86 assembly is used to control the direction of string operations