1 Subtraction and sign flags

1.1 Subtraction

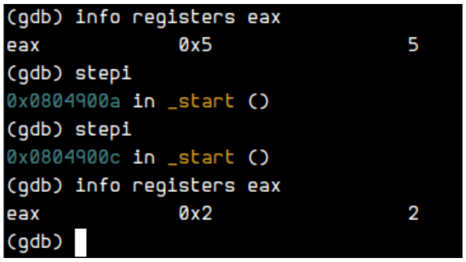
Assembly program:

```
section .data

section .text
        MOV eax, 5
        MOV ebx, 3
        SUB eax, ebx
        MOV eax, 1
        INT 80h
```

The instruction **SUB eax, ebx** means $eax = eax - ebx$. This is normal subtraction where we are subtracting a smaller number from a larger number.

We can see the values of the register **eax** before and after subtraction.



```
(gdb) info registers eax
eax            0x5                 5
(gdb) stepi
0x0804900a in _start ()
(gdb) stepi
0x0804900c in _start ()
(gdb) info registers eax
eax            0x2                 2
(gdb) 
```

Figure 1: Values in **eax**

This was simple subtraction.

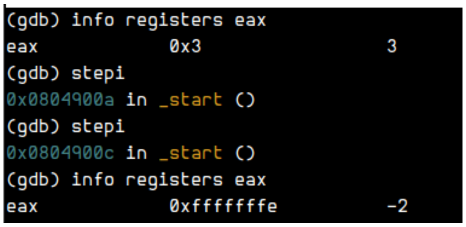
Now, we will execute the following assembly program using GDB:

```
section .data

section .text
        MOV eax, 3
        MOV ebx, 5
        SUB eax, ebx
        MOV eax, 1
        INT 80h
```

Here also, we are performing the subtraction operation, but this time we are subtracting the larger number from the smaller number(which will result in a negative number).

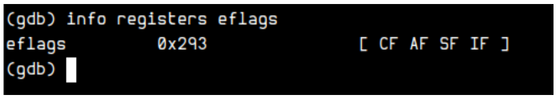
We can see that **eax** stores -2.



```
(gdb) info registers eax
eax            0x3                 3
(gdb) stepi
0x0804900a in _start ()
(gdb) stepi
0x0804900c in _start ()
(gdb) info registers eax
eax            0xfffffffe          -2
```

Figure 2: Negative value in **eax**

We can also view which flags were set during this operation.



```
(gdb) info registers eflags
eflags         0x293               [ CF AF SF IF ]
(gdb) 
```

Figure 3: The flags that were set

The **CF** serves two functions in x86:

- it represents a carry
- it also represents a **borrow**

Following are some rules for binary subtraction:

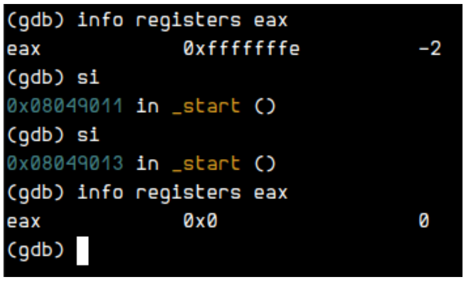
$$\begin{aligned}
0 - 0 &= 0\\
1 - 0 &= 1\\
0 - 1 &= 1 (with\ borrow\ 1)\\
1 - 1 &= 0
\end{aligned}$$

SF means the **S**ign **F**lag. When this flag is set to 1, it indicates that the operation produced a negative output(in our case it's -2).

Let's say that we added the following lines to the assembly program above:

```
MOV ebx, 2
ADD eax, ebx
```

Now, what we basically did was this: −2 was stored in the register **eax**. Now we changed the value of **ebx** to 2 and then performed addition operation on **eax** and **ebx**. Since we are adding −2(in **eax**) and 2(in **ebx**) we get a 0 which is stored in register **eax**.



```
(gdb) info registers eax
eax            0xfffffffe          -2
(gdb) si
0x08049011 in _start ()
(gdb) si
0x08049013 in _start ()
(gdb) info registers eax
eax            0x0                 0
(gdb) 
```

Figure 4: Values of **eax** before and after addition