1 Multiplying numbers with *MUL* and *IMUL*

- **MUL** → Multiplies unsigned numbers.
- **IMUL** → Multiplies signed numbers.

1.1 Using the MUL instruction

Assembly program to multiply two numbers using **MUL**:

```
section .data

section .text
        global _start
        _start:
                MOV al, 2
                MOV bl, 3
                MUL bl

                MOV eax, 1
                INT 80h
```

The **MUL** instruction requires only one register. The reason is that the register **eax** is automatically used by multiplication and division instructions meaning that we don't need to specify it explicitly.

Here is an image showing the value in **al** before and after multiplication:

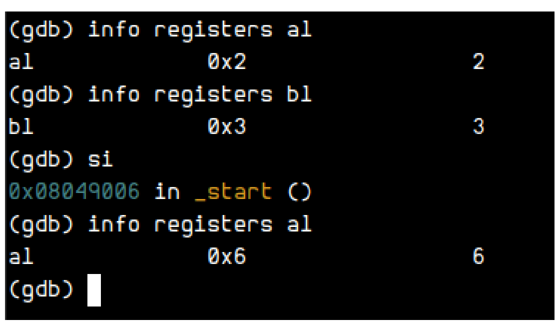

Figure 1: Values of **al**

What happens if the result of the multiplication is larger than the register?

Program:

```
section .data

section .text
        global _start
        _start:
                MOV al, 0xFF
                MOV bl, 2
                MUL bl

                MOV eax, 1
                INT 80h
```

0xFF in 8-bit representation is -1 and in 16-bit representation is 255. Now, we store this value in the register **al**.

When the multiplication operations is performed then we get the following result:

Figure 2: Negative numbers

So after performing multiplication we see that -2 is stored in **al**. We know that -2 can be represented in 8-bit binary as 11111110.

Now, if we check what is stored in the higher-bit register **ah**, then we will get the following:

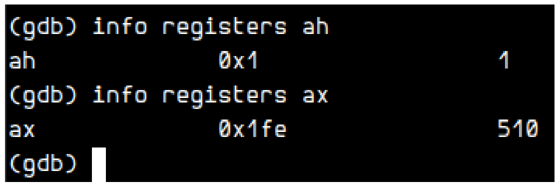

Figure 3: Value in **ah** and 16-bit representation of -2

Now, after that we check what is the value stored in 16-bits that is register **ax**. We get 510.

Now the binary representation of 510 in 16-bits is:

$$00000001\ 11111110$$

The lower 8-bits are in **al** and higher 8-bits are in the register **ah**.

1.2 Using the IMUL instruction

Program:

```
section .data

section .text
        global _start

        _start:
                MOV al, 0xFF
                MOV bl, 2
                IMUL bl

                MOV eax, 1
                INT 80h
```

When we multiply two n-bit numbers, the product is $2 * n$-bits long. Keeping that in mind, when 2 is multiplied to -1 then we get the product as -2. This product is stored in the register **al** and also in **ah** because the product is 16(i.e $2 * 8$)-bits long.

This is shown in the image below:

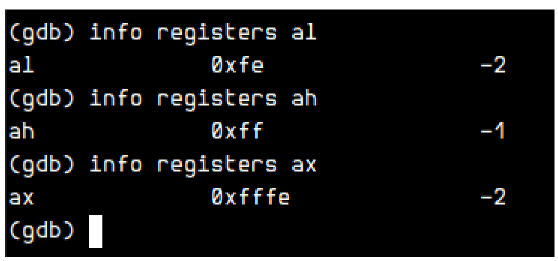

Figure 4: Registers: **al**, **ah** and **ax**

Basically this is what is stored in **ax**:

11111111 11111110

The lower 8-bits are stored in **al** and the higher 8-bits are stored in **ah**.