# Blocksembler Exercises

HTL Leonding WS2025

## Exercise1: Addition

Write a program that reads two numbers and stores them in registers `r1` and `r2`. Then, add the two registers and store the result in register `r3`. Finally, output the value of register `r3`.

**Example:**

| Input | Output |
|---|---|
| `r1 = 10, r2= 15` | `25` |

## Exercise 2: Countdown

Write a program that reads a number and stores it in register `r1`. Then, repeat the following operations until the value in register `r1` is `0`:

1. Decrease the value in register `r1` by one.
2. Output the value of register `r1`.

**Hint:** You will need a `branch equal zero` or `branch greater zero` instruction for this.

| Input | Output |
|---|---|
| `r1 = 5` | `4`<br>`3`<br>`2`<br>`1`<br>`0` |

## Exercise 3: Multiplication

Write a program that reads one number into **register `r1`** and another into **register `r2`**. Then, **multiply the two numbers**.

**Example:**

| Input | Output |
|---|---|
| `r1 = 5, r2 = 3` | `15` |

**Hint:** In the architecture we are using, there is **no dedicated multiplication instruction**. Therefore, you can implement multiplication using **repeated additions**.

**Hint:** There is a **more efficient way** to compute the result — think about how you would multiply two **multi-digit numbers** on paper!

## Exercise 4: Fibonacci Numbers

Fibonacci numbers form a sequence where each number is the sum of the two preceding ones, starting with 0 and 1. Thus, the sequence begins with 0, 1, 1, 2, 3, 5, 8, 13, and so on. This sequence was named after the Italian mathematician Leonardo Fibonacci, who lived in the 13th century and described it in his work **Liber Abaci**.

$$\begin{aligned}
&F(0) = 0\\
&F(1) = 1\\
&F(2) = 1\\
&F(3) = F(1) + F(2) = 2\\
&F(4) = F(2) + F(3) = 3\\
&F(5) = F(3) + F(4) = 5\\
&\dots
\end{aligned}$$

Write a program that calculates the Fibonacci number at the *n-th* position. The number **n** should be entered by the user at the beginning.

**Example:**

| Input | Output |
|---|---|
| `r1 = 7` | `13` |

## Exercise 5: Fill Operation

Write a program that performs the following steps:

1) Read two values from the user and store them in registers r1 and r2.
   `r1`: the number of memory locations to fill
   `r2`: the value to be stored in each location
2) Fill the last n (= the value in r1) memory locations with the value in r2.

**Example:**

If the user inputs `3` for `r1` and `4` for `r2`, the last memory locations look like this:

```
...
0xFFFA: 0 (0000 0000 0000 0000)
0xFFFB: 0 (0000 0000 0000 0000)
0xFFFC: 0 (0000 0000 0000 0000)
0xFFFD: 4 (0000 0000 0000 0100)
0xFFFE: 4 (0000 0000 0000 0100)
0xFFFF: 4 (0000 0000 0000 0100)
```