# LAB03: STRING COMPARE

## Task

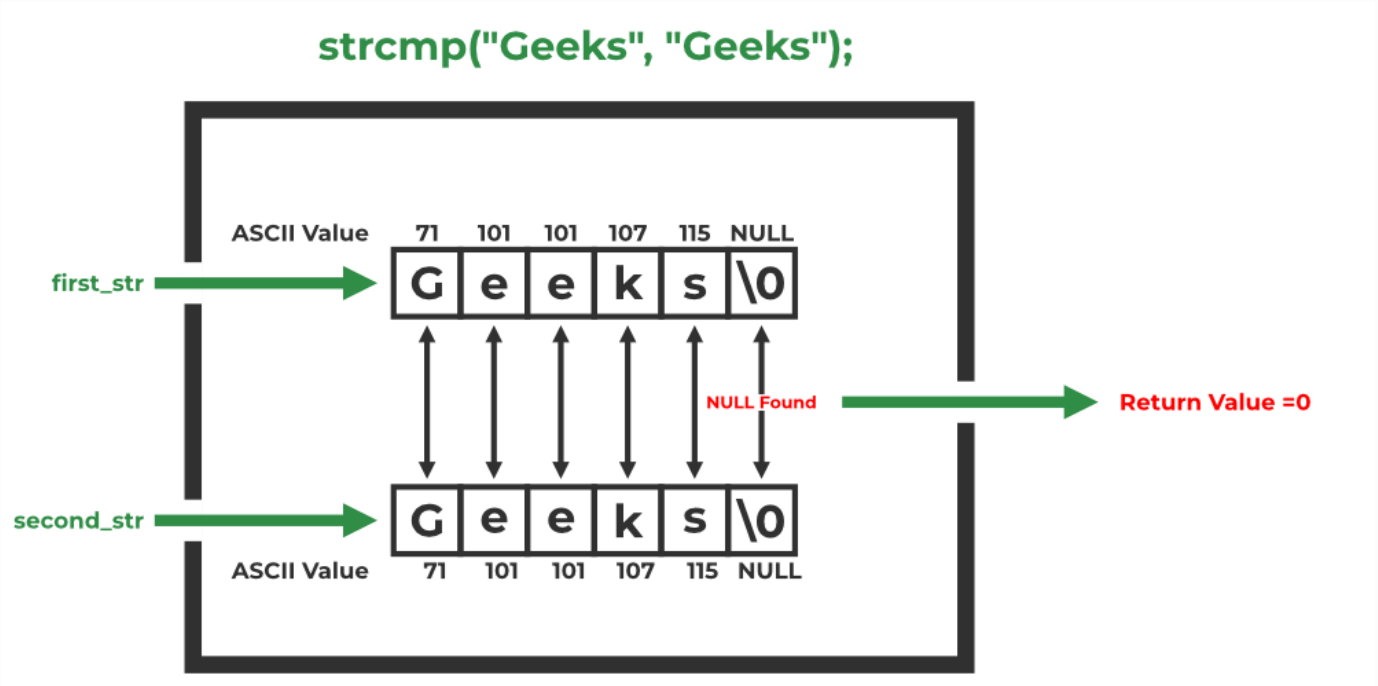
`strcmp()` is a built-in library function that is used for string comparison in C language. This function takes two strings (array of characters) as arguments, **compares these two strings lexicographically**. In this lab, you are required to use LC-3 assembly language to complete a simplified version of the `strcmp()` function.

The `strcmp()` function compares the ASCII value of each character till the non-matching value is found or the NULL character is found. The working of the C `strcmp()` function can be described as follows:

**1.** It starts with comparing the ASCII values of the first characters of both strings.

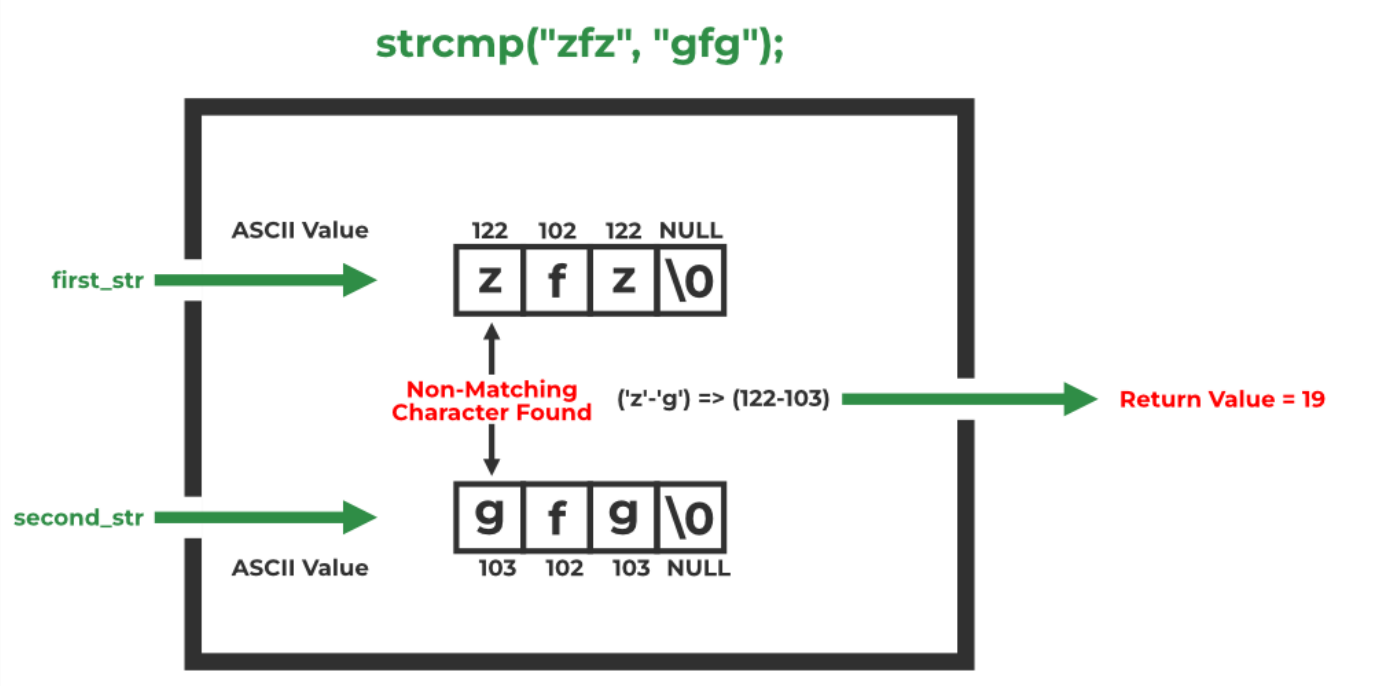
**2.** If the first characters in both strings are equal, then this function will check the second character, if they are also equal, then it will check the third, and so on till the first unmatched character is found or the **NULL** character is found.

- If a **NULL** character is found, the function **returns zero** as both strings will be the same.

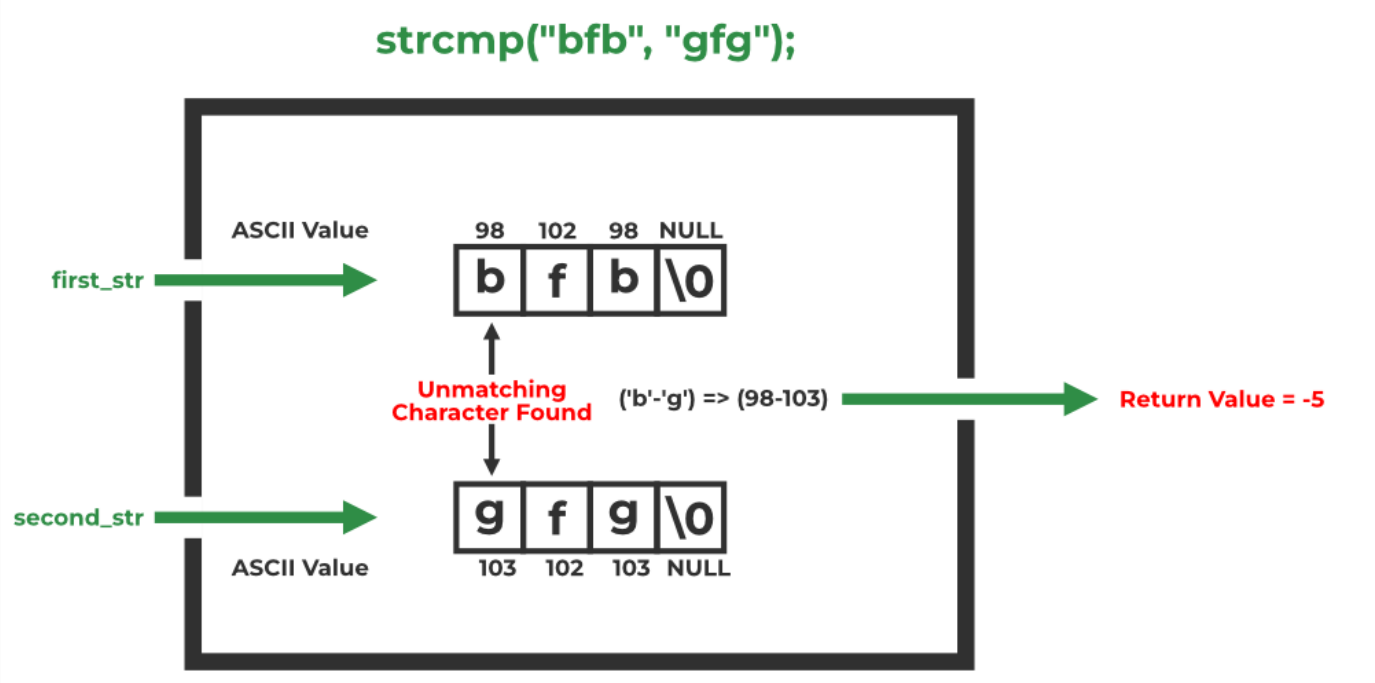


**3.** If a non-matching character is found,

- If the ASCII value of the character of the first string is greater than that of the second string, then the **positive difference** ( **> 0**) between their ASCII values is returned.




- If the ASCII value of the character of the first string is less than that of the second string, then the **negative difference** ( **< 0**) between their ASCII values is returned.




Given two strings S1 and S2, the starting addresses of the two strings are `x3100` and `x3200` respectively. Each character in the string is stored in successive memory location, and the string is

terminated with a NULL character. You can assume that S1 and S2 only contain characters from a-z, A-Z, and a NULL character as the terminator. Note that `0 < strlen(S1), strlen(S2) < 100`.

Your job: store the return value of strcmp() in `x3300`.

R0-R7 are set to zeroes at the beginning and your program should be start at `x3000`.

**Example 1:**

| **Address** | **x3100** | **x3101** | **x3102** | **x3103** | **x3104** | **x3105** |
|---|---|---|---|---|---|---|
| Value(Hex) | x0044 | x0073 | x0054 | x0041 | x0073 | x0000 |
| Character | D | s | T | A | s | NULL |

| **Address** | **x3200** | **x3201** | **x3202** | **x3203** | **x3204** | **x3205** |
|---|---|---|---|---|---|---|
| Value(Hex) | x0044 | x0073 | x0074 | x0041 | x0000 | - |
| Character | D | s | t | A | NULL | - |

| **Address** | **x3300** |
|---|---|
| Value(Hex) | xFFE0 |

**Example 2:**

| **Address** | **x3100** | **x3101** | **x3102** | **x3103** | **x3104** | **x3105** |
|---|---|---|---|---|---|---|
| Value(Hex) | x0044 | x0073 | x0054 | x0041 | x0073 | x0000 |
| Character | D | s | T | A | s | NULL |

| **Address** | **x3200** | **x3201** | **x3202** | **x3203** | **x3204** | **x3205** |
|---|---|---|---|---|---|---|
| Value(Hex) | x0044 | x0073 | x0074 | x0041 | x0000 | - |
| Character | D | s | T | A | NULL | - |

| Address | x3300 |
| --- | --- |
| Value(Hex) | x0073 |

For simplicity, your code can be written as follows.

```
        .ORIG     x3000

        ; Begin of your code
        ; ... ...
        ; ... ...
        ; End of your code

        STI       R2, RESULT  ; write back you result
        HALT

S1_ADDR .FILL     x3100
S2_ADDR .FILL     x3200
RESULT  .FILL     x3300
        .END

        .ORIG     x3100
S1      .STRINGZ  "FirstString"
        .END

        .ORIG     x3200
S2      .STRINGZ  "SecondString"
        .END
```

## Score

Your score will be split between correctness (50%) and the report (50%).

## Submission

For this lab, you are required to use **assembly code**. Please adhere to the following guidelines:

1. Your program should start with `.ORIG x3000`
2. Ensure your program ends with `.END`

3. Your last instruction must be `TRAP x25 (HALT)`
4. Use capitalized keywords and labels (e.g., "ADD" rather than "add").
5. Maintain spaces after commas for clarity.
6. Prefix decimal constants with `#` and hexadecimal constants with a lowercase `x` .
7. Include comments in your code where necessary for clarification.

Your submission should be structured as shown below:

```
PB********_Name.zip
├── PB********_Name_report.pdf
└── lab3.asm
```

## Reports

Your report should be structured into the following sections:

1. Purpose
2. Principles
3. Procedure (e.g. bugs or challenges you encountered and how to solve them)
4. Results