
Laboratory Exercise 3

8086 Assembly Programming (DOSBox)

Duration: Until 3:50 PM

Submission Format:

- Take screenshots of all your code (use `u` multiple times to scroll and capture the entire program).
- Take screenshots of the memory dump **before** and **after** execution.
- Zip these files and submit them to **local quanta**.

Note:

1. All programs must start execution from offset `0100H`.
2. **Instruction Limitation:** For this exercise, you are expected to use **String Operations**.
    - Allowed Instructions: `MOV`, `ADD`, `SUB`, `INC`, `DEC`, `JMP` (and conditional jumps), `CMP`, `INT`.
    - **String Instructions:** `MOVS` (Move String), `LODS` (Load String), `SCAS` (Scan String).
    - Usage of complex high-level directives is prohibited.

Q1. Substring Count and Replace

Objective: Practice string manipulation using specific x86 string instructions (`LODS`, `SCAS`, `MOVS`) to scan, compare, and modify text data in memory.

Problem Statement:

A sentence string is stored in memory starting at address `0200H` (terminated by `00H`).
A target 3-letter substring is stored at address `0300H`.
A replacement 3-letter substring is stored at address `0310H`.

Write an 8086 assembly program to:

1. Count the number of times the target substring (from `0300H`) appears in the sentence.
2. Store this count in the **CX** register.
3. Replace **every** occurrence of the target substring in the sentence with the replacement substring (from `0310H`).

Input Parameters

- **Memory at 0200H:** Source Sentence (Null-terminated string).
- **Memory at 0300H:** Target Substring (3 characters exactly).
- **Memory at 0310H:** Replacement Substring (3 characters exactly).

Output Parameter

- **Register CX:** Total count of occurrences found.
- **Memory at 0200H:** Modified sentence with replacements.

Constraints

- The search and replacement must be case-sensitive.
- You must use string manipulation instructions (`LODS`, `SCAS`, `MOVS`) where applicable.
- There will be no test cases which require overlap (Eg: ata etc).

Test Case

Input:

- **Sentence (at 0200H):** " catering to cats catching cold$" (terminated by **$**:24)
- **Target (at 0300H):** "cat"
- **Replacement (at 0310H):** "dog"

Memory Dump (Input):

Plaintext

```
0200: 20 63 61 74 65 72 69 6E-67 20 74 6F 20 63 61 74 catering
to cat 0210: 73 20 63 61 74 63 68 69-6E 67 20 63 6F 6C 64 24 s
catching cold$ 0300: 63 61 74 cat 0310: 64 6F 67 dog
```

```
0200: 20 63 61 74 65 72 69 6E-67 20 74 6F 20 63 61 74   catering to cat
0210: 73 20 63 61 74 63 68 69-6E 67 20 63 6F 6C 64 24   s catching cold$
0300: 63 61 74                                          cat
0310: 64 6F 67                                          dog
```

Output:

- **Register CX:** `0003` (Occurrences in "**cat**ering", "**cat**s", "**cat**ching")
- **Sentence (at 0200H):** " dogering to dogs dogching cold"

Memory Dump (Output):

Plaintext

```
0200: 20 64 6F 67 65 72 69 6E-67 20 74 6F 20 64 6F 67 dogering
to dog
0210: 73 20 64 6F 67 63 68 69-6E 67 20 63 6F 6C 64 24 s dogching
cold$
```

```
0200: 20 64 6F 67 65 72 69 6E-67 20 74 6F 20 64 6F 67   dogering to dog
0210: 73 20 64 6F 67 63 68 69-6E 67 20 63 6F 6C 64 24   s dogching cold$
```

Refer to Tutorial 3: I/O and String Handling for File input (n and l commands) to enter the string in memory instead of writing manually

Important ASCII (HEX) Values:

- Space ‘ ‘ : **20**
- Dollar’$’: **24**