Str_trim Procedure

The Str_trim procedure removes all occurrences of a specified trailing character from the end of a null-terminated string. For example, it can trim trailing spaces from a string.

It takes two parameters:

pString - Pointer to the null-terminated ASCII string to trim.
char - The ASCII character to trim from the end of the string It does not return anything, but modifies the string in-place by truncating it.

Logic
It handles several cases:

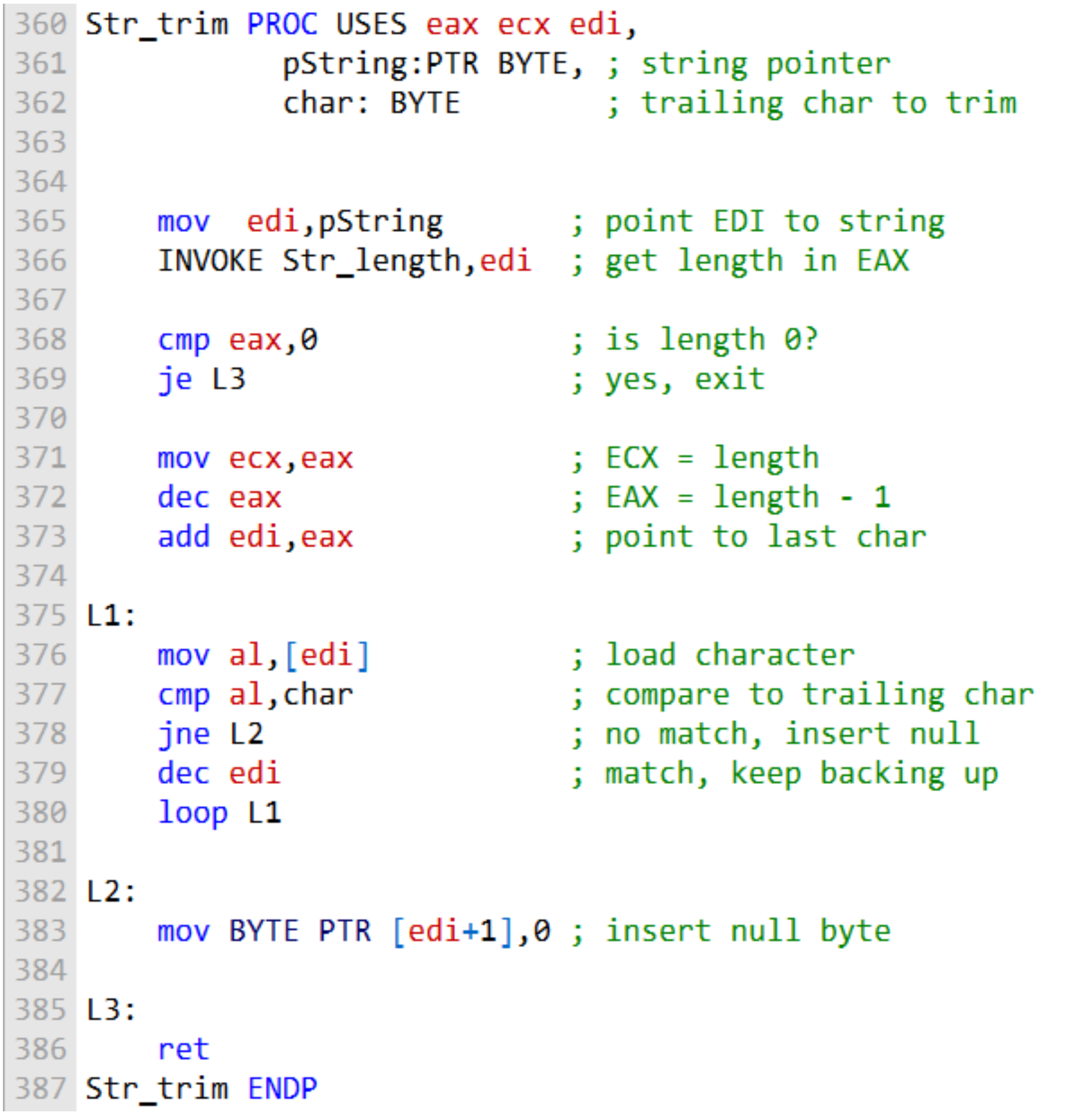
1. Empty string - nothing to do
2. String with trailing character(s) - remove them
3. String with only the trailing char - truncate to empty string
4. String without any trailing char - leave unchanged
5. String with trailing char(s) followed by other chars - remove only trailing char(s)

To trim the string, it inserts a null byte (\0) after the last character to keep. Any chars after the null become insignificant.

It uses these steps:

1. Get length of string
2. Check if length is 0 and exit if so (empty string case)
3. Initialize loop counter to string length
4. Point to last character
5. Loop backwards until beginning

- Check if current char matches trailing char to trim
- If yes, decrement counter to keep backing up
- If no, insert null byte after current char and exit loop
- Insert null byte truncates the string



```
Str_trim PROC USES eax ecx edi,
           pString:PTR BYTE, ; string pointer
           char: BYTE        ; trailing char to trim


    mov  edi,pString        ; point EDI to string
    INVOKE Str_length,edi  ; get length in EAX

    cmp eax,0               ; is length 0?
    je L3                   ; yes, exit

    mov ecx,eax             ; ECX = length
    dec eax                 ; EAX = length - 1
    add edi,eax             ; point to last char

L1:
    mov al,[edi]            ; load character
    cmp al,char             ; compare to trailing char
    jne L2                  ; no match, insert null
    dec edi                 ; match, keep backing up
    loop L1

L2:
    mov BYTE PTR [edi+1],0 ; insert null byte

L3:
    ret
Str_trim ENDP
```

Line 1: The Str_trim procedure pushes the EAX, ECX, and EDI registers onto the stack. This is necessary because the procedure uses these registers.

Line 2: The Str_trim procedure moves the pointer to the string to be trimmed into the EDI register.

Line 3: The Str_trim procedure calls the Str_length procedure to get the length of the string. The length of the string is stored in the EAX register.

Line 4: The Str_trim procedure compares the length of the string to 0. If the length of the string is equal to 0, then the string is empty and the procedure exits.

Line 6: The Str_trim procedure moves the length of the string to the ECX register. This will be used as the loop counter.

Line 7: The Str_trim procedure decrements the EAX register. This will be used to point to the last character in the string.

Line 8: The Str_trim procedure adds the EAX register to the EDI register. This will point the EDI register to the last character in the string.

Line 9: The Str_trim procedure enters a loop. On each iteration of the loop, the following steps are performed:

• The Str_trim procedure moves the byte at the memory location pointed to by the EDI register into the AL register.
• The Str_trim procedure compares the AL register to the character to be trimmed.
• If the two characters are equal, then the Str_trim procedure decrements the EDI register and continues to the next iteration of the loop.
• If the two characters are not equal, then the Str_trim procedure breaks out of the loop.

Line 14: The Str_trim procedure inserts a null byte at the memory location pointed to by the EDI register plus one. This will terminate the string at the last character that is not the character to be trimmed.

Line 15: The Str_trim procedure exits the loop.

Line 16: The Str_trim procedure pops the EAX, ECX, and EDI registers from the stack.

Line 17: The Str_trim procedure returns.

Here is an example of how to use the Str_trim procedure:

```
; Trim all trailing spaces from the string "Hello, world!    "
mov eax, OFFSET "Hello, world!    "
call Str_trim
; The EAX register will now contain a pointer to the string "Hello, world!"
```