Str_compare Procedure

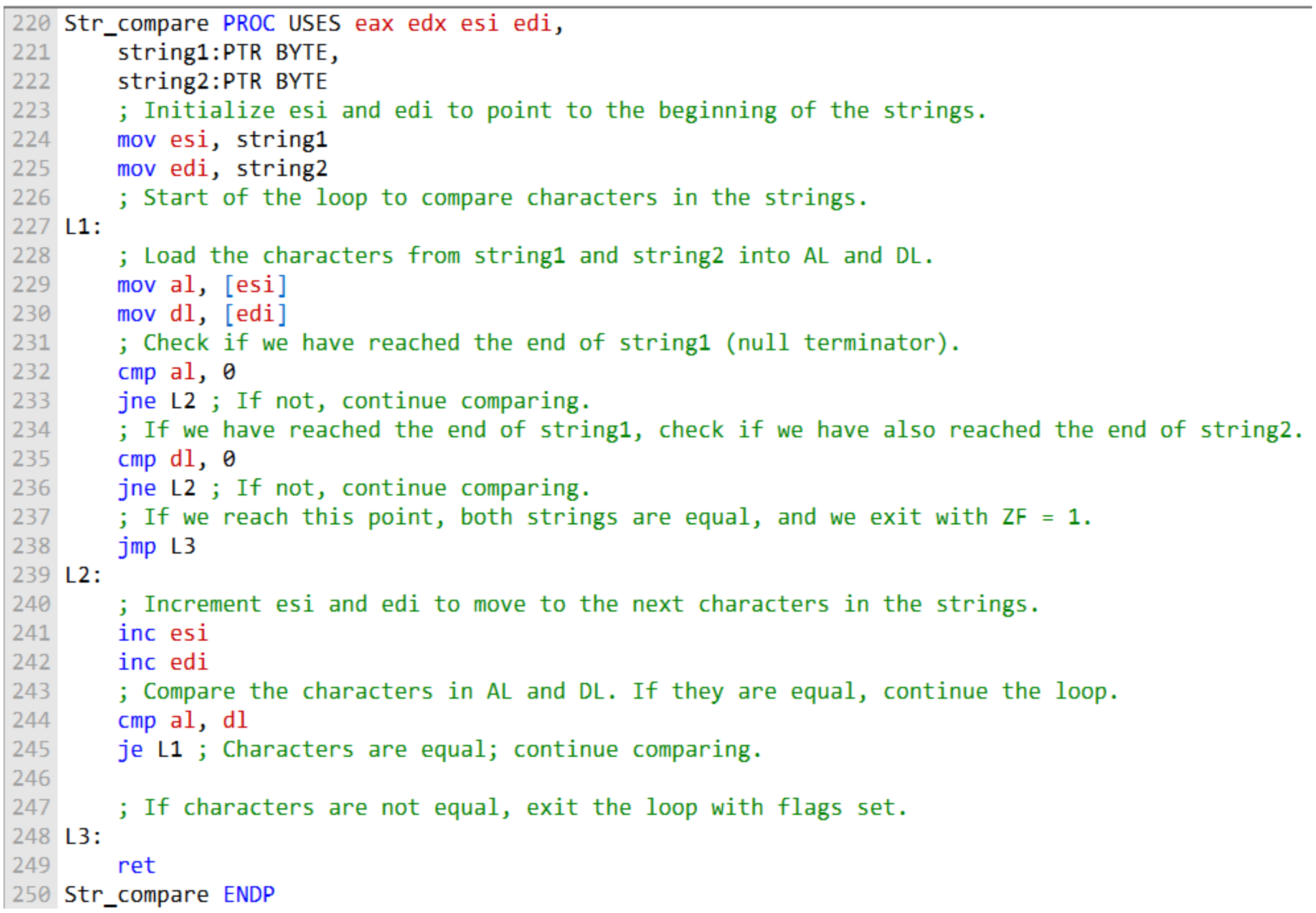
The Str_compare procedure compares two strings and sets the Carry and Zero flags in the same way as the CMP instruction. It takes two arguments: pointers to the two strings.

The Str_compare procedure works by comparing byte by byte of the two strings.

If a byte is found that is not equal in both strings, the Str_compare procedure sets the Carry flag and exits the loop.

If the end of both strings is reached, the Str_compare procedure sets the Zero flag and exits the loop.

The following is a more detailed explanation of the Str_compare procedure:



```
Str_compare PROC USES eax edx esi edi,
    string1:PTR BYTE,
    string2:PTR BYTE
    ; Initialize esi and edi to point to the beginning of the strings.
    mov esi, string1
    mov edi, string2
    ; Start of the loop to compare characters in the strings.
L1:
    ; Load the characters from string1 and string2 into AL and DL.
    mov al, [esi]
    mov dl, [edi]
    ; Check if we have reached the end of string1 (null terminator).
    cmp al, 0
    jne L2 ; If not, continue comparing.
    ; If we have reached the end of string1, check if we have also reached the end of string2.
    cmp dl, 0
    jne L2 ; If not, continue comparing.
    ; If we reach this point, both strings are equal, and we exit with ZF = 1.
    jmp L3
L2:
    ; Increment esi and edi to move to the next characters in the strings.
    inc esi
    inc edi
    ; Compare the characters in AL and DL. If they are equal, continue the loop.
    cmp al, dl
    je L1 ; Characters are equal; continue comparing.

    ; If characters are not equal, exit the loop with flags set.
L3:
    ret
Str_compare ENDP
```

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

Line 2: The Str_compare procedure moves the pointers to the two strings into the ESI and EDI registers, respectively.

Line 3: The Str_compare procedure enters a loop. On each iteration of the loop, the following steps are performed:

- The Str_compare procedure compares the bytes at the memory locations pointed to by the ESI and EDI registers.
- If the bytes are equal, the Str_compare procedure increments the ESI and EDI registers and continues to the next iteration of the loop.
- If the bytes are not equal, the Str_compare procedure sets the Carry flag and exits the loop.

Line 10: The Str_compare procedure checks if the end of the string pointed to by the ESI register has been reached. If the end of the string has been reached, the Str_compare procedure checks if the end of the string pointed to by the EDI register has also been reached.

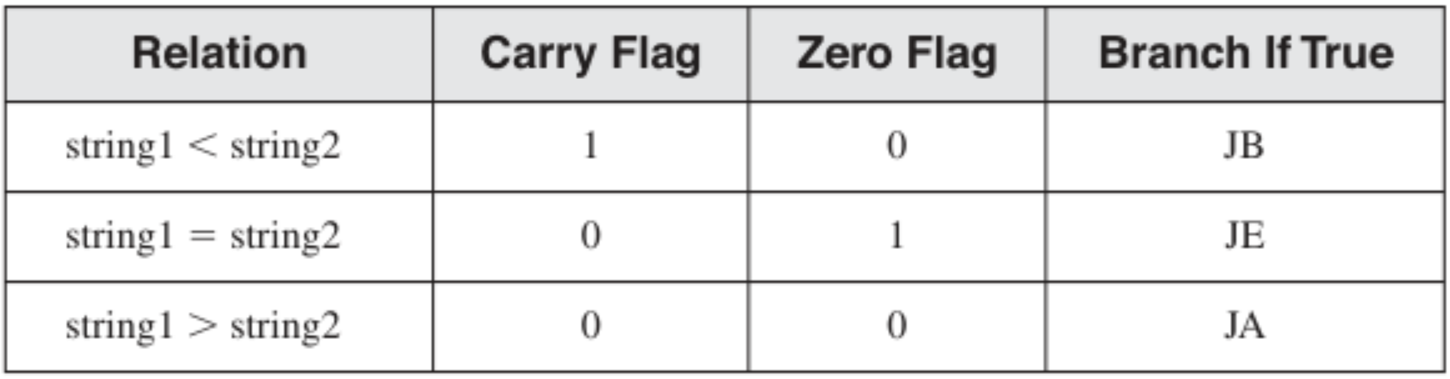
Line 12: If the end of both strings has been reached, the Str_compare procedure sets the Zero flag and exits the loop.

Line 14: The Str_compare procedure increments the ESI and EDI registers so that they point to the next byte in the respective strings.

Line 16: The Str_compare procedure compares the bytes at the memory locations pointed to by the ESI and EDI registers. If the bytes are equal, the Str_compare procedure continues to the next iteration of the loop. If the bytes are not equal, the Str_compare procedure sets the Carry flag and exits the loop.

Line 20: The Str_compare procedure exits the loop and returns to the caller.

The following is a table of the flags affected by the Str_compare procedure:



Relation	Carry Flag	Zero Flag	Branch If True
string1 < string2	1	0	JB
string1 = string2	0	1	JE
string1 > string2	0	0	JA

This code defines the Str_compare procedure, which compares two strings character by character. It uses esi and edi to traverse the strings and compares characters while checking for the end of the strings.

The procedure exits with the Zero and Carry flags set according to the result of the comparison. If both strings are equal, ZF is set to 1; otherwise, ZF is set to 0.

The Carry flag is set to indicate the relation between the strings: (JB for "string1 < string2," JA for "string1 > string2," JE for "string1 = string2")