CMP Instruction

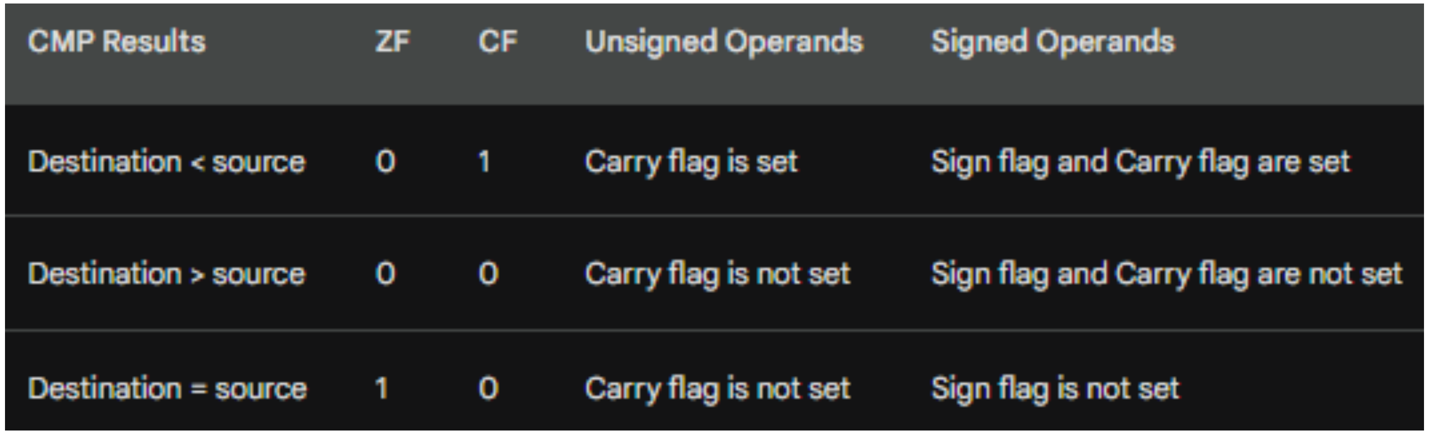
The CMP (compare) instruction is indeed an essential part of x86 assembly language when it comes to comparing integers, including character codes.

It's used to perform an implied subtraction of a source operand from a destination operand, without modifying either operand.

The result of this operation affects various flags in the CPU, which can then be used for conditional branching, making it a crucial component in creating conditional logic structures.

The **CMP instruction** performs an implied subtraction of the source operand from the destination operand. However, the actual subtraction is not performed. Instead, the status flags are set according to the result of the subtraction.

Here's a breakdown of how the CMP instruction affects flags based on the comparison results:



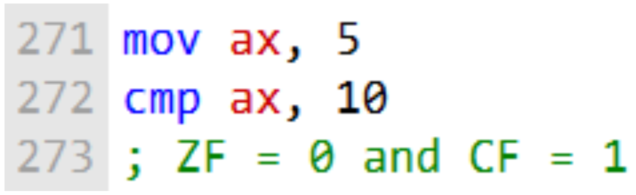
CMP Results	ZF	CF	Unsigned Operands	Signed Operands
Destination < source	0	1	Carry flag is set	Sign flag and Carry flag are set
Destination > source	0	0	Carry flag is not set	Sign flag and Carry flag are not set
Destination = source	1	0	Carry flag is not set	Sign flag is not set

Unsigned Operands. When comparing two unsigned operands:

If the destination is less than the source, the Zero Flag (ZF) is set to 0, and the Carry Flag (CF) is set to 1. If the destination is greater than the source, ZF is set to 0, and CF is set to 0. If the destination equals the source, ZF is set to 1, and CF is set to 0.

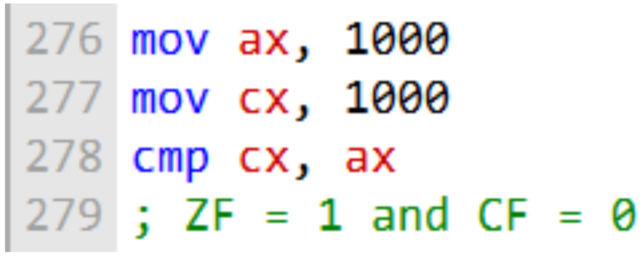
Signed Operands. When comparing two signed operands:

If the destination is less than the source, the Sign Flag (SF) is not equal to the Overflow Flag (OF). If the destination is greater than the source, SF is equal to OF. If the destination equals the source, the Zero Flag (ZF) is set to 1.



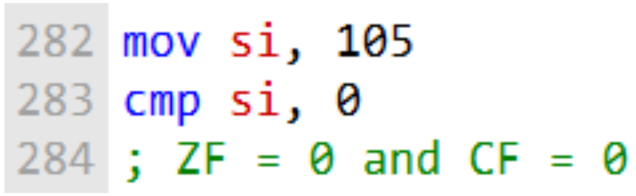
```
mov ax, 5
cmp ax, 10
; ZF = 0 and CF = 1
```

In this example, when AX (with a value of 5) is compared to 10, the CMP instruction sets the Zero Flag (ZF) to 0 because 5 is not equal to 10. The Carry Flag (CF) is set to 1 because subtracting 10 from 5 would require a borrow. Example 2:



```
mov ax, 1000
mov cx, 1000
cmp cx, ax
; ZF = 1 and CF = 0
```

Here, when AX and CX both contain 1000, the CMP instruction sets the Zero Flag (ZF) to 1 because subtracting one 1000 from the other results in zero. The Carry Flag (CF) is set to 0 because no borrow is required. Example 3:



```
mov si, 105
cmp si, 0
; ZF = 0 and CF = 0
```

In this case, when SI (with a value of 105) is compared to 0, the CMP instruction sets both the Zero Flag (ZF) and the Carry Flag (CF) to 0 because subtracting 0 from 105 generates a positive, nonzero value.

CMP, when used in conjunction with conditional jump instructions, allows you to create conditional logic structures, akin to high-level programming languages' IF statements, in assembly language. It's a powerful tool for controlling the flow of your programs based on comparisons between values in registers or memory locations.