

Range Checking a Byte

Approach by White Flame

In checking for the range $[x,y)$, instead of performing 2 comparisons the idea is to subtract x to align the range to $[0,y-x)$. This then allows us to perform a single unsigned comparison to check both ends of the range.

Any numbers lower than the original range will have wrapped the byte into the high values $\Leftarrow 255$, and any number higher than the original range will still be too large.

```
; Check .A for the range [10,100)
sec
sbc #10 ; start of the range
cmp #90 ; length of the range
bcs fail ; result needs to be 0-89 to pass the original 10-99 check
... ; .A is in range here, and Carry is clear
```

Approach by Lee Davison

For all of these we assume that the byte to be tested is in A and that the start and end values, n and m , are already defined. Also that $0 < n < m < \$FF$.

If you don't need to preserve the byte in A then testing the byte can be done in five bytes and only six cycles. This sets the carry if A is in the range n to m .

```
CLC ; clear carry for add
ADC # $\$FF-m$ ; make  $m = \$FF$ 
ADC # $m-n+1$ ; carry set if in range  $n$  to  $m$ 
```

Obviously, if the state of the carry flag is always known before executing this code, you can skip the CLC as well and adjust the routine accordingly.

From:

<https://codebase64.org/> - **Codebase 64 wiki**

Permanent link:

https://codebase64.org/doku.php?id=base:8-bit_ranged_comparison

Last update: **2015-04-22 20:15**

