Some words about the ANC opcode

Written by FTC/HT.

Like many other illegal opcodes the ANC instruction performs “two operations in one”, and as is also often the case, one of these two operations is an AND operation. The ANC opcode only exist in the “immediate” version (ANC #$ff) and it works like this:

1. AND immediate value (#$xx) with A, and store result in A.
2. Put highest bit of the result in the carry.

When is ANC useful?

Here are some cases where I found this opcode to be useful. If you've encountered other cases, feel free to add those.

Case 1 - implicit enforcement of carry flag state

When using an AND instruction before an addition (or any other operation where you might want to know the state of the carry flag), you might save two cycles (not having to do CLC or SEC) by using ANC instead of AND. Since a cleared hibit in the value used with the ANC instruction always leads to a unset carry flag after this operation, you can take advantage of that. An example:

```
    lda value
    anc #$0f    ;Carry flag is always set to 0 after this op.
    adc value2    ;Add a value. CLC not needed!
    sta result
```

Another case like this is when you want to set the A register to #$00 specifically, and also happen to want to have the carry cleared:

```
    anc #0        :Carry always cleared after this op, and A register always
set to zero.
```

Case 2 - remembering a bit

You can use ANC to simply putting the highest bit of a byte into the carry flag without affecting the A register, by using ANC #$FF. This is sometimes useful since there are many instructions that do not destroy/modify the (C)arry flag and the (N)egative flag. The ones that DO change the state of carry/negative flag are mainly mathematic operations, shifting operations and comparison operations. This means that you can use ANC to “remember” the info about the carry flag during the execution of other code, such as some LDA/STA stuff.

Another command that can be used to achieve the same thing is CMP #$80 (as well as CPX and CPY), which puts the high bit of a register into Carry as well, without changing the register itself. See also

Changing Registers.

Using illegal opcodes in assemblers that do not support them

In case your assembler does not support illegal opcodes, remember that you can always use them anyway using a byte directive, like this:

```
    .byte $0b,$ff ;produces ANC #$FF. Note that $2b instead of $0b produces equivalent results.
```

Info on ANC on external sites

- Ninjas reference
- Grahams reference