[Return to Main Page] 6502 Compare Instructions from 6502 Software Design, Updated by Bruce Clark
[Up to Tutorials and Aids]

The compare instructions set or clear three of the status flags (Carry, Zero, and Negative) that can be tested with branch instructions, *without altering the contents of the operand*. There are three types of compare instructions:

The Compare Instructions

Instruction	Description
CMP	Compare Memory and Accumulator
CPX	Compare Memory and Index X
CPY	Compare Memory and Index Y

The CMP instruction supports eight different addressing modes, the same ones supported by the ADC and SBC instructions. Since the X and Y registers function primarily as counters and indexes, the CPX and CPY instructions do not require this elaborate addressing capability and operate with just three addressing modes (immediate, absolute, and zero page).

The compare instructions subtract (without carry) an immediate value or the contents of a memory location from the addressed register, but do not save the result in the register. The only indications of the results are the states of the three status flags: Negative (N), Zero (Z), and Carry (C). The combination of these three flags indicate whether the register contents are *less than, equal to* (the same as), or *greater than* the operand "data" (the immediate value or contents of the addressed memory location. The table below summarizes the result indicators for the compare instructions.

Compare Instruction Results

Compare Result	N	Z	C
A, X, or Y < Memory	*	0	0
A, X, or Y = Memory	0	1	1
A, X, or Y > Memory	*	0	1

** The N flag will be bit 7 of A, X, or Y - Memory*

The compare instructions serve only one purpose; they provide information that can be tested by a subsequent branch instruction. For example, to branch if the contents of a register are less than an immediate or memory value, you would follow the compare instruction with a Branch on Carry Clear (BCC) instruction, as shown by the following:

Example: Comparing Memory to the Accumulator

```
        CMP  $20     ;Accumulator less than location $20?
        BCC  THERE
HERE                 ;No, continue execution here.
        .
        .
        .
THERE                ;Execute this if Accumulator is less than location $20.
```

```
        .
        .
        .
```

The table below lists the branch instruction(s) that should follow the compare instruction, for each register/data relationship. In this table, THERE represents the label of the instruction executed if the branch test succeeds and HERE represents the label of te instruction executed if the branch test does not succeed. Besides comparing a memory location and a register, the compare instructions are handy for comparing one memory location with another, by loading one into a register (A, X, or Y).

Use of Branch Instructions with Compare

To Branch If	Follow compare instruction with	
	For unsigned numbers	For signed numbers
Register is less than data	BCC THERE	BMI THERE
Register is equal to data	BEQ THERE	BEQ THERE
Register is greater than data	BEQ HERE BCS THERE	BEQ HERE BPL THERE
Register is less than or equal to data	BCC THERE BEQ THERE	BMI THERE BEQ THERE
Register is greater than or equal to data	BCS THERE	BPL THERE

The next example contains a routine that tests whether the contents of two memory locations are identical, and sets a flag in memory to so indicate.

Example: Testing Two Locations for Equality

```
;This routine sets memory location $22 to "One" if the contents
;of locations $20 and $21 are equal, and to "Zero" if otherwise.

        LDX  #00     ;Initialize flag to zero
        LDA  $20     ;Get first value
        CMP  $21     ;Is second value identical?
        BNE  DONE
        INX          ;Yes, set flag to one
DONE    STX  $22     ;Store flag
```

Below is another memory-to-memory comparison routine. It stores the larger of two values in the higher memory location.

Example: Arranging Two Numbers in Order of Value

```
;This routine arranges two numbers in locations $20 and $21 in
;order of value, with the lower-valued number in location $20.
```

```
        LDA  $21     ;Load second number into accumulator
        CMP  $20     ;Compare the numbers
        BCS  DONE    ;DONE if first is less than or equal to second.
        LDX  $20
        STA  $20     ;Otherwise swap them.
        STX  $21
DONE     .
         .
         .
```

The next example contains a register-to-constant comparison routine in which two branch instructions are used with one compare instruction, so that the "less than", "equal to", and "greater than" conditions are tested. Since the branch instructions do not afect any flags in the processor status register, BNE GT3 is basing its branch decision on the same comparison that the BCS EQGT3 based its branch decision on!

Example: A Three-Way Decision Routine

```
;This routine stores the contents of the accumulator into location
;$20, $21, or $22, depending upon whether the accumulator holds a
;value les than 3, equal to 3, or grather than 3, respectively.

        CMP  #03     ;Compare accumulator to 3
        BCS  EQGT3
        STA  $20     ;Accumulator less than 3
        JMP  DONE
EQGT3   BNE  GT3
        STA  $21     ;Accumulator equal to 3
        JMP  DONE
GT3     STA  $22     ;Accumulator greater than 3
DONE     .
         .
         .
```

Thus far, our discussion has concentrated on the CMP instruction, which compares the accumulator with memory. Of what value are the other compare instructions (CPX and CPY)? Their primary value is to monitor the contents of X or Y when these registers are being employed as count-up counters. In these cases, CPX or CPY is used to compare the count against a maximum value of 255 (decimal) in memory. The next example is a routine that will move up to 256 consecutive memory bytes, with the CPX instruction doing the "all bytes moved" check each time a byte is moved.

Example: A Multiple-Byte Move Routine

```
;This routine moves up to 256 bytes of memory, starting at
;location $20, to another portion of memory, starting at location
;$0320.  The byte count is contained in location $1F.

        LDX  #00     ;Index = 0
NXTBYT  LDA  $20,X   ;Load next byte
        STA  $0320,X ;Store next byte
        INX          ;Increment index
        CPX  $1F     ;All bytes moved?
        BNE  NXTBYT  ;If not, move next byte
```

The compare instructions compare two "entire" 8-bit values. There are situations, however, when you will need to test one or more individual bits in a memory location. This can be done by masking out the unwanted bits with an AND instruction. However, in the process of masking-out the unwanted bits, the AND instruction destroys the mask contained in the accumulator. Certainly, the mask could be reloaded, but this requires additional processor time and additional instructions. The same job can be done, without altering the accumulator, by executing a BIT instruction. A tutorial on the BIT instruction can also be found in the Tutorials section of 6502.org.

Last Updated December 1, 2002.