6502 Program Example 02

Here is a program which is designed to add together two sixteen bits numbers. Unfortunately, the accumulator register is only 8 bits long, so we have to do it in two stages. After we have got the numbers stored in memory, we have to add together the least significant bytes of each number, which may or may not generate a carry, store that part of the result, then add together the most significant bytes of each number, along with the carry, and store that part of the result.

```
; Program to add two 16 bit numbers
; The numbers being added are $0194 and $01BA
; Add1601.65s

      .ORG $0200  ; Store machine code starting here

      LDA #$94    ; Store least significant end of number 1 in
      STA olo     ;   byte labelled olo
      LDA #$01    ; Store most significant end of number 1 in byte
      STA ohi     ;   labelled ohi

      LDA #$BA    ; Store little end of number 2 in
      STA tlo     ;   byte labelled tlo
      LDA #$01    ; Store big end of number 2 in byte
      STA thi     ;   labelled thi

      CLC         ; Clear the carry flag
      LDA olo     ; Load little end of number 1 into accumulator register
      ADC tlo     ; Add with carry the little end of number 2
      STA rlo     ; Store the little end of the result in byte labelled rlo
      LDA ohi     ; Load big end of number 1 into accumulator
      ADC thi     ; Add with carry the big end of number 2
      STA rhi     ; Store the big end of the result in the byte
                  ;     labelled rhi

      BRK         ; Stop running the program

olo:  .DB $00     ; These two bytes are going to be used to store the first
ohi:  .DB $00     ; number (one low and one high)

tlo:  .DB $00     ; These two bytes are going to be used to store the
thi:  .DB $00     ; second number (two low and two high)

rlo:  .DB $00     ; These two bytes are going to be used to store the
rhi:  .DB $00     ; result (result low and result high)
```

Exercises

Write out the machine code instructions that the assembler would produce for this program. Check your answer against the simulator.

Trace through the program and watch the processor do the work.

You might have once written a C program, which involved a loop, and put a very high number in for it to count to. At a certain point the numbers might have suddenly turned negative. What happened?

How could we make this program more robust: for example, what would happen if we added $7FFF to $7FFF?

What issues are involved with the output of the result?