Decimal to hexadecimal conversion

By Mace

This routine converts two bytes decimal to two bytes hexadecimal. 'hiInput' contains hundreds, 'loInput' contains tens and ones, so $0365 equals #365 decimal. It will be converted to 'hiResult' and 'loResult' (in this case $016d, the hexadecimal of 365).

```
    .pc = $0810

        lda #$00    ; Init result bytes
        sta loResult
        sta hiResult
        lda loInput    ; Fetch ones and tens
        tax        ; Save to X
        and #$0f    ; Strip the ones
        tay        ; Save to Y
        txa        ; Get original ones and tens
        lsr
        lsr
        lsr
        lsr        ; Divide by 16 to get tens only
        tax        ; Save to X
        tya        ; Put the ones in A
        cpx #$00    ; No tens? Then skip
        beq PROCEED
        clc
TENS:       adc #$0a    ; Add as many tens as value of X
        dex
        bne TENS
PROCEED:    ldx hiInput    ; Fetch the hundreds
        beq END        ; No hundreds? Then go to end
HUND:       clc
        adc #$64    ; Add as many hundreds as value of X
        bcc !+
        inc hiResult    ; Increase high byte every passed $FF
!:      dex
        bne HUND
END:        sta loResult    ; Store low byte
        rts

hiInput:    .byte $00
loInput:    .byte $01
hiResult:   .byte $00
loResult:   .byte $00
```

By Verz:

I made a small improvement, the idea being that a 10x multiplication can be computed as a sum of a

(8x + 2x); since we're shifting the high nibble to the right, we get the 8x and 2x values in the process. (Note that using a loResult in page zero there would be 5 bytes/cycles of improvement)

```
bcdbin:
            lda #$00        ; Init result bytes
            ;sta loResult    ; useless
            sta hiResult
            lda loInput     ; Fetch ones and tens
            tay             ; Save to Y
            and #$f0        ; this two instructions, or use the undocumented
ALR #$f0 = (and #$f0) + (lsr)
            lsr
            ;alr #$f0
            sta loResult
            lsr
            lsr
            adc loResult
            sta loResult
            tya
            and #$0f        ; Strip the ones
            adc loResult

            ldx hiInput     ; Fetch the hundreds
            beq END         ; No hundreds? Then go to end
HUND:       clc
            adc #$64        ; Add as many hundreds as value of X
            bcc loop
            inc hiResult    ; Increase high byte every passed $FF
loop:       dex
            bne HUND
END:        sta loResult    ; Store low byte
            rts

hiInput:    .byte $00
loInput:    .byte $01
hiResult:   .byte $00
loResult:   .byte $00
```

Here a routine to convert a single BCD byte (0-99) in A, to a binary in A:

```
BCDBIN8
          tay
          and            #%11110000
          lsr
          ;alr            #%11110000  ;  =(And #$f0)+(Lsr), to replace the
two previous instructions
          sta            pzTmp
          lsr
          lsr
          adc            pzTmp
```

```
          sta             pzTmp
          tya
          and             #%1111
          adc             pzTmp
          rts
pzTmp .equ $02    ; $02 is a unused zeropage memory location in C64
```