Fastest 16x16 unsigned multiplication

By Repose

Requires tables or a generator routine such as table_generator_routine_for_fast_8_bit_mul_table

Jack Asser's: 233 cycles ref: seriously_fast_multiplication
Chris Jam's: 204.5 ref: http://csdb.dk/forums/?roomid=11&topicid=91766
Mine: 196 zp variation: 192
Times above need to add 12 for jsr/rts

```
;World's fastest 16x16 unsigned mult for 6502
;you can go faster, but not without more code and/or data
;and being less elegant and harder to follow.
;by Repose 2017
;table generator by Graham
;addition improvement suggested by JackAsser

;data: 2044 bytes
;zero page ram required: minimum 8 bytes, ideally 14
;do_add: 30 bytes in zp, if used
;time: 196 cycles, option for 192 if you use 30 more zp bytes for do_add
;measurement method: branches/page boundary crossings were averaged

;How to use:
;put numbers in x/y and result is Y reg, X reg, z1, z0

;tables of squares
;sqr(x)=x^2/4
;negsqr(x)=(255-x)^2/4
sqrlo=$c000;511 bytes
sqrhi=$c200;511 bytes
negsqrlo=$c400;511 bytes
negsqrhi=$c600;511 bytes

;pointers to square tables above
p_sqr_lo=$8b;2 bytes
p_sqr_hi=$8d;2 bytes
p_invsqr_lo=$8f;2 bytes
p_invsqr_hi=$91;2 bytes

;the inputs and outputs
x0=$fb;multiplier, 2 bytes
x1=$fc
y0=$fd;multiplicand, 2 bytes
y1=$fe
z0=$80;product, 2 bytes
z1=$81
;z2=$82 returned in X reg
```

```
;z3=$83 returned in Y reg

;Example showing use
lda #$ff
sta x0
sta x1
sta y0
sta y1
jsr maketables
jsr umult16
;result should be $fffe0001, e.g. as viewed with a typical m 0080 monitor
command:
;0080 01 00 fe ff

makesqrtables:
;init zp square tables pointers
lda #>sqrlo
sta p_sqr_lo+1
lda #>sqrhi
sta p_sqr_hi+1
lda #>negsqrlo
sta p_invsqr_lo+1
lda #>negsqrhi
sta p_invsqr_hi+1

;generate sqr(x)=x^2/4
      ldx #$00
      txa
      !by $c9   ; CMP #immediate - skip TYA and clear carry flag
lb1:  tya
      adc #$00
ml1:  sta sqrhi,x
      tay
      cmp #$40
      txa
      ror
ml9:  adc #$00
      sta ml9+1
      inx
ml0:  sta sqrlo,x
      bne lb1
      inc ml0+2
      inc ml1+2
      clc
      iny
      bne lb1

;generate negsqr(x)=(255-x)^2/4
      ldx #$00
      ldy #$ff
mt1:
```

```
      lda sqrhi+1,x
      sta negsqrhi+$100,x
      lda sqrhi,x
      sta negsqrhi,y
      lda sqrlo+1,x
      sta negsqrlo+$100,x
      lda sqrlo,x
      sta negsqrlo,y
      dey
      inx
      bne mt1
      rts

umult16:
;set multiplier as x0
lda x0
sta p_sqr_lo
sta p_sqr_hi
eor #$ff
sta p_invsqr_lo
sta p_invsqr_hi;17

ldy y0
sec
lda (p_sqr_lo),y
sbc (p_invsqr_lo),y;note these two lines taken as 11 total
sta z0;x0*y0l
lda (p_sqr_hi),y
sbc (p_invsqr_hi),y
sta c1a+1;x0*y0h;31
;c1a means column 1, row a (partial product to be added later)

ldy y1
;sec  ;notice that the high byte of sub above is always +ve
lda (p_sqr_lo),y
sbc (p_invsqr_lo),y
sta c1b+1;x0*y1l
lda (p_sqr_hi),y
sbc (p_invsqr_hi),y
sta c2a+1;x0*y1h;31

;set multiplier as x1
lda x1
sta p_sqr_lo
sta p_sqr_hi
eor #$ff
sta p_invsqr_lo
sta p_invsqr_hi;17

ldy y0
;sec
```

```
lda (p_sqr_lo),y
sbc (p_invsqr_lo),y
sta c1c+1;x1*y0l
lda (p_sqr_hi),y
sbc (p_invsqr_hi),y
sta c2b+1;x1*y1h;31

ldy y1
;sec
lda (p_sqr_lo),y
sbc (p_invsqr_lo),y
sta c2c+1;x1*y1l
lda (p_sqr_hi),y
sbc (p_invsqr_hi),y
tay;x1*y1h;Y=z3, 30 cycles
;17+33+31+17+31+30=159 cycles for main multiply part

;jmp do_adds; can put do_adds in zp for a slight speed increase
do_adds:
;-add the first two numbers of column 1
    clc
c1a:    lda #0
c1b:    adc #0
    sta z1;9

;-continue to first two numbers of column 2
c2a:    lda #0
c2b:    adc #0
    tax;X=z2, 6 cycles
    bcc c1c;3/6 avg 4.5
    iny;z3++
    clc

;-add last number of column 1
c1c:    lda #0
    adc z1
    sta z1;8

;-add last number of column 2
    txa;A=z2
c2c:    adc #0
    tax;X=z2, 6
    bcc fin;3/4 avg 3.5
    iny;z3++

;Y=z3, X=z2
;9+6+4.5+8+6+3.5=37
fin:    rts

Diagram of the additions
                y1    y0
```

```
             x  x1    x0
                --------
             x0y0h x0y0l
+      x0y1h x0y1l
+      x1y0h x1y0l
+x1y1h x1y1l
------------------------
    z3    z2    z1    z0
```