8bit Divide - 8bit Result

Normal binary division

...with shifting in loop. (If I remember right - submitted by Graham at CSDb forum)

```
;normal binary division
        ASL $FD
        LDA #$00
        ROL

        LDX #$08
.loop1
        CMP $FC
        BCC *+4
        SBC $FC
        ROL $FD
        ROL
        DEX
        BNE .loop1

        LDX #$08
.loop2
        CMP $FC
        BCC *+4
        SBC $FC
        ROL $FE
        ASL
        DEX
        BNE .loop2
```

Divides the value in $FD by the value in $FC, 8 bit integer result in $FD, the first 8 fraction bits are in $FE.

Ofcourse both loops should be unrolled :) I didn't want to write down the unrolled code here.

doynax: The remainder (in the accumulator) in the fraction loop seems to overflow for divisors above $80. A BCS jumping directly from the top of the loop to the SBC and forcibly setting carry afterwards seems to work. Is there a cleaner solution?

Smaller version

```
; 8bit/8bit division
; by White Flame
;
; Input: num, denom in zeropage
; Output: num = quotient, .A = remainder
```

```
 lda #$00
 ldx #$07
 clc
: rol num
  rol
  cmp denom
  bcc :+
   sbc denom
: dex
 bpl :--
 rol num

; 19 bytes
;
;  Best case  = 154 cycles
;  Worst case = 170 cycles
;
; With immediate denom:
;  Best case  = 146 cycles
;  Worst case = 162 cycles
;
; Unrolled with variable denom:
;  Best case  = 106 cycles
;  Worst case = 127 cycles
;
; Unrolled with immediate denom:
;  Best case  =  98 cycles
;  Worst case = 111 cycles
```

If you don't understand what :, :–, :+ means. : is an anonymous label. bpl :–, for example, goes back two labels in the code.

Division using tables

Comes from CSDb forum (Fix Me! source by... ???)

;This will divide two 8-bit numbers in some 90-150 cycles. ;The code can easily be extended to handle larger dividends.

```
_divu_8
    lda div_b
    cmp #2
    bcs + ; >= 2

    lda div_a
    rts

+   ldx #8
```

```
-   dex
    asl
    bcc -

    bne +

    lda div_a
-   lsr
    dex
    bne -
    rts

+   tay
    lda r0_table,y
    ldy div_a

    sta zp8_1
    sta zp8_2
        eor #$ff
    sta zp8_3
    sta zp8_4

        sec
    lda (zp8_1),y
    sbc (zp8_3),y
    lda (zp8_2),y
    sbc (zp8_4),y

    clc
    adc div_a

    ror
-   lsr
    dex
    bne -
    rts

div_a
    .byte $0
div_b
    .byte $0
r0_table
    .byte $01,$00,$fd,$00,$f9,$00,$f5,$00,$f1,$00,$ed,$00,$ea,$00,$e6,$00
    .byte $e2,$00,$df,$00,$db,$00,$d8,$00,$d5,$00,$d1,$00,$ce,$00,$cb,$00
    .byte $c8,$00,$c4,$00,$c1,$00,$be,$00,$bb,$00,$b8,$00,$b5,$00,$b3,$00
    .byte $b0,$00,$ad,$00,$aa,$00,$a7,$00,$a5,$00,$a2,$00,$9f,$00,$9d,$00
    .byte $9a,$00,$98,$00,$95,$00,$93,$00,$90,$00,$8e,$00,$8b,$00,$89,$00
    .byte $87,$00,$84,$00,$82,$00,$80,$00,$7e,$00,$7b,$00,$79,$00,$77,$00
    .byte $75,$00,$73,$00,$71,$00,$6f,$00,$6d,$00,$6b,$00,$69,$00,$67,$00
    .byte $65,$00,$63,$00,$61,$00,$5f,$00,$5d,$00,$5b,$00,$59,$00,$58,$00
    .byte $56,$00,$54,$00,$52,$00,$51,$00,$4f,$00,$4d,$00,$4b,$00,$4a,$00
```

```
    .byte $48,$00,$47,$00,$45,$00,$43,$00,$42,$00,$40,$00,$3f,$00,$3d,$00
    .byte $3c,$00,$3a,$00,$39,$00,$37,$00,$36,$00,$34,$00,$33,$00,$31,$00
    .byte $30,$00,$2f,$00,$2d,$00,$2c,$00,$2a,$00,$29,$00,$28,$00,$26,$00
    .byte $25,$00,$24,$00,$22,$00,$21,$00,$20,$00,$1f,$00,$1d,$00,$1c,$00
    .byte $1b,$00,$1a,$00,$19,$00,$17,$00,$16,$00,$15,$00,$14,$00,$13,$00
    .byte $12,$00,$10,$00,$0f,$00,$0e,$00,$0d,$00,$0c,$00,$0b,$00,$0a,$00
    .byte $09,$00,$08,$00,$07,$00,$06,$00,$05,$00,$04,$00,$03,$00,$02,$00
```

The same routine again, slightly optimized

```
Let me bore you with an optimized version:

    ; divide acc by y, result in acc
_divu_8
    ldx t0_table,y
    stx b1+1
    ldx t1_table,y
    beq +

    ldy r0_table,x

    sta zp8_1
    sta zp8_2
        eor #$ff
    sta zp8_3
    sta zp8_4

        sec
    lda (zp8_1),y
    sbc (zp8_3),y
    lda (zp8_2),y
    sbc (zp8_4),y

    clc
    adc zp8_1
    ror

+   sec
b1  bcs b1
    lsr
    lsr
    lsr
    lsr
    lsr
    lsr
    lsr

    rts

    .align $100
```

```
r0_table
    .byte $01,$00,$fd,$00,$f9,$00,$f5,$00,$f1,$00,$ed,$00,$ea,$00,$e6,$00
    .byte $e2,$00,$df,$00,$db,$00,$d8,$00,$d5,$00,$d1,$00,$ce,$00,$cb,$00
    .byte $c8,$00,$c4,$00,$c1,$00,$be,$00,$bb,$00,$b8,$00,$b5,$00,$b3,$00
    .byte $b0,$00,$ad,$00,$aa,$00,$a7,$00,$a5,$00,$a2,$00,$9f,$00,$9d,$00
    .byte $9a,$00,$98,$00,$95,$00,$93,$00,$90,$00,$8e,$00,$8b,$00,$89,$00
    .byte $87,$00,$84,$00,$82,$00,$80,$00,$7e,$00,$7b,$00,$79,$00,$77,$00
    .byte $75,$00,$73,$00,$71,$00,$6f,$00,$6d,$00,$6b,$00,$69,$00,$67,$00
    .byte $65,$00,$63,$00,$61,$00,$5f,$00,$5d,$00,$5b,$00,$59,$00,$58,$00
    .byte $56,$00,$54,$00,$52,$00,$51,$00,$4f,$00,$4d,$00,$4b,$00,$4a,$00
    .byte $48,$00,$47,$00,$45,$00,$43,$00,$42,$00,$40,$00,$3f,$00,$3d,$00
    .byte $3c,$00,$3a,$00,$39,$00,$37,$00,$36,$00,$34,$00,$33,$00,$31,$00
    .byte $30,$00,$2f,$00,$2d,$00,$2c,$00,$2a,$00,$29,$00,$28,$00,$26,$00
    .byte $25,$00,$24,$00,$22,$00,$21,$00,$20,$00,$1f,$00,$1d,$00,$1c,$00
    .byte $1b,$00,$1a,$00,$19,$00,$17,$00,$16,$00,$15,$00,$14,$00,$13,$00
    .byte $12,$00,$10,$00,$0f,$00,$0e,$00,$0d,$00,$0c,$00,$0b,$00,$0a,$00
    .byte $09,$00,$08,$00,$07,$00,$06,$00,$05,$00,$04,$00,$03,$00,$02,$00
t0_table
    .fill $100,0
t1_table
    .fill $100,0

_divu_8_setup
    ldy #1
next
    tya
    ldx #$ff
-   inx
    asl
    bcc -
    sta t1_table,y
    txa
    sta t0_table,y
    iny
    bne next
    rts
```

The init optimized (well, it packs better)

```
r0_table
    .byte $01,$fd,$f9,$f5,$f1,$ed,$ea,$e6
    .byte $e2,$df,$db,$d8,$d5,$d1,$ce,$cb
    .byte $c8,$c4,$c1,$be,$bb,$b8,$b5,$b3
    .byte $b0,$ad,$aa,$a7,$a5,$a2,$9f,$9d
    .byte $9a,$98,$95,$93,$90,$8e,$8b,$89
    .byte $87,$84,$82,$80,$7e,$7b,$79,$77
    .byte $75,$73,$71,$6f,$6d,$6b,$69,$67
    .byte $65,$63,$61,$5f,$5d,$5b,$59,$58
    .byte $56,$54,$52,$51,$4f,$4d,$4b,$4a
    .byte $48,$47,$45,$43,$42,$40,$3f,$3d
```

```
    .byte $3c,$3a,$39,$37,$36,$34,$33,$31
    .byte $30,$2f,$2d,$2c,$2a,$29,$28,$26
    .byte $25,$24,$22,$21,$20,$1f,$1d,$1c
    .byte $1b,$1a,$19,$17,$16,$15,$14,$13
    .byte $12,$10,$0f,$0e,$0d,$0c,$0b,$0a
    .byte $09,$08,$07,$06,$05,$04,$03,$02
    .fill $80,0
t0_table
    .fill $100,0
t1_table
    .fill $100,0

_divu_8_setup
    ldx #$7f
    ldy #$ff
-
    lda #0
    sta r0_table,y
    dey
    lda r0_table,x
    sta r0_table,y
    dey
    dex
    bpl -
    ldy #1
next
    tya
    ldx #$ff
-   inx
    asl
    bcc -
    sta t1_table,y
    txa
    sta t0_table,y
    iny
    bne next
    rts
```