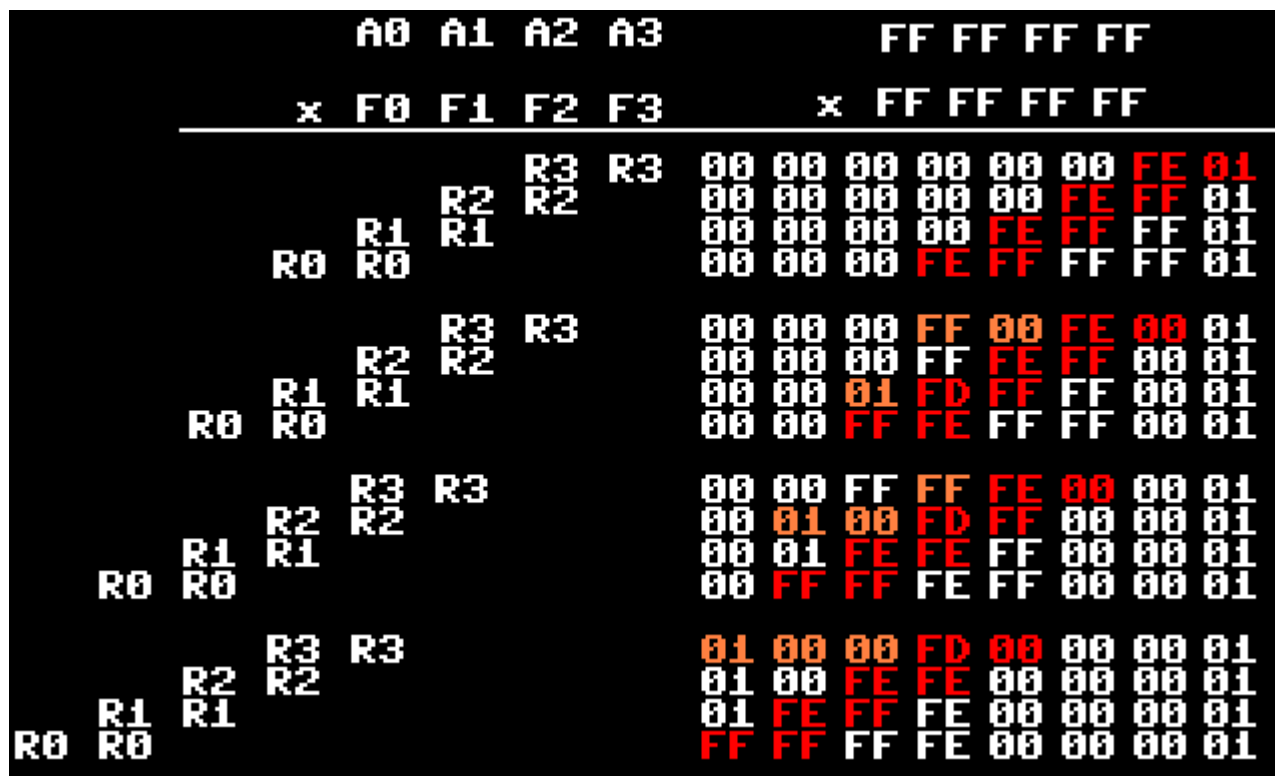


Fast Floating Point Multiply

This code uses the fast multiplication [found here](#) to quickly multiply the mantissas. Each multiplication is added to the product as you go.

In the procedure shown below, the red indicates where the two bytes have been added to the partial product, and the orange is where a carry has propagated.



```
;Fast Floating Point Multiplication
```

```
;ZP
```

```
RESULT = $26 ;-$2c
```

```
FACX = $61 ;exponent
```

```
FAC = $62 ;mantissa - $65
```

```
; $66 ;sign, in MSB
```

```
ARGX = $69 ;exponent
```

```
ARG = $6a ;mantissa - $6d
```

```
; $6e ;sign, in MSB
```

```
SGNCMP = $6f
```

```
res8 = $fb ;lobyte of product
```

```
;BASIC Subroutines
```

```
PRINTFAC = $aabc
```

```
VARtoFAC = $bba2 ;a/y, lo/hi
```

```
VARtoARG = $ba8c
```

```

!macro addressult .num, .carry {
    pha
    lda res8
    clc
    adc RESULT+.num
    sta RESULT+.num
    pla
    adc (RESULT-1)+.num
    sta (RESULT-1)+.num
    !if .carry != 0 {
        !set .num = .num - 1
        bcc +
        inc (RESULT-1)+.num
        !do while .carry > 1 {
            !set .num = .num - 1
            !set .carry = .carry - 1
            bne +
            inc (RESULT-1)+.num
        }
    }
}

+
}

!to "fastm.prg",cbm

*=$800          ;10 SYS 10240(3000)
!byte $00,$0E,$08,$0A,$00,$9E,$28,$31,$32,$32,$38,$38,$29,$00,$00,$00

*=$3000

ldx #$00
txa
!byte $c9      ;cmp #, clears carry, skips tya
lb1 tya
adc #$00
ml1 sta multabhi,x
tay
cmp #$40
txa
ror
ml9 adc #$00
sta ml9+1
inx
ml0 sta multablo,x
bne lb1
inc ml0+2
inc ml1+2
clc

```

```

    iny
    bne lb1

    ldx #$00
    ldy #$ff
-   lda multabhi+1,x
    sta multab2hi+$100,x
    lda multabhi,x
    sta multab2hi,y
    lda multablo+1,x
    sta multab2lo+$100,x
    lda multablo,x
    sta multab2lo,y
    dey
    inx
    bne -

    lda #<value
    ldy #>value
    jsr VARToFAC
    lda #<value
    ldy #>value
    jsr VARToARG
    jsr mul_FP
    jsr PRINTFAC

here    jmp here

mul_FP  bne +
        rts      ;0 in FAC returns 0
+   jsr $bab7    ;add exponents
    lda #0        ;clear RESULT
    sta $26
    sta $27
    sta $28
    sta $29
    sta $2a
    sta $2b
    sta $2c

    lda FAC+3
    ldx ARG+3
    jsr mul8x8
    sta RESULT+6
    ldx ARG+2
    jsr mulf
+address 6,0
    ldx ARG+1
    jsr mulf
+address 5,0
    ldx ARG

```

```
jsr mulf
+address 4,0

lda FAC+2
ldx ARG+3
jsr mul8x8
+address 6,2
ldx ARG+2
jsr mulf
+address 5,2
ldx ARG+1
jsr mulf
+address 4,2
ldx ARG
jsr mulf
+address 3,2

lda FAC+1
ldx ARG+3
jsr mul8x8
+address 5,2
ldx ARG+2
jsr mulf
+address 4,2
ldx ARG+1
jsr mulf
+address 3,2
ldx ARG
jsr mulf
+address 2,1

lda FAC
ldx ARG+3
jsr mul8x8
+address 4,3
lda RESULT+4
sta $70 ;rounding byte
ldx ARG+2
jsr mulf
+address 3,2
ldx ARG+1
jsr mulf
+address 2,1
ldx ARG
jsr mulf
+address 1,0

jmp $bb8f ;normal multiply exit.
;copies RESULT to FAC1
;and normalises.
```

```
mul8x8  sta sm1+1
        sta sm3+1
        eor #$ff
        sta sm2+1
        sta sm4+1
mulf    sec
sm1    lda multablo,x
sm2    sbc multab2lo,x
        sta res8
sm3    lda multabhi,x
sm4    sbc multab2hi,x
        rts

value   !byte $82,$49,$b2,$6c,$9c ;3.15151515

multablo = $c000
multabhi = $c200
multab2lo = $c400
multab2hi = $c600
```

From:
<https://codebase64.org/> - **Codebase 64 wiki**

Permanent link:
https://codebase64.org/doku.php?id=base:fast_floating_point_multiply

Last update: **2016-02-06 01:34**

