FSqrt.bas

fSqrt

ZX BASIC uses the ZX Spectrum ROM routine to calculate many of the floating point math functions. Unfortunately, some of the functions are notoriously slow. Square root being one of them - there wasn't enough space in the original ROM design to do a good routine, so they cheated. They calculate $x^{0.5}$ instead, and roll straight into the exponent routine. It turns out that though this works, it's exceptionally slow.

This function uses the Newton Raphson method. It produces exactly the same full floating point result (it even uses the ROM calculator), but it does it about six times faster. If you have a program that's calculating lots of square roots, this will make a big difference. However, note that integer square roots are faster still, and if you are games programming, accuracy might not be so critical. See iSqrt.bas for details.

```
REM Fast floating Point Square Root Function
REM Adapted and modified for Boriel's ZX BASIC
REM By Britlion

FUNCTION FASTCALL fSqrt (radicand as FLOAT) as FLOAT
ASM
        ; FLOAT value arrives in A ED CB
        ; A is the exponent.
          AND   A               ; Test for zero argument
          RET   Z               ; Return with zero.

          ;Strictly we should test the number for being negative and quit if it is.
          ;But let's assume we like imaginary numbers, hmm?
          ; If you'd rather break it change to a jump to an error below.
          ;BIT   7,E          ; Test the bit.
          ;JR    NZ,REPORT       ; back to REPORT_A
                                ; 'Invalid argument'
          RES 7,E               ; Now it's a positive number, no matter what.

          call __FPSTACK_PUSH   ; Okay, We put it on the calc stack. Stack contains ABS(x)

          ;   Halve the exponent to achieve a good guess.(accurate with .25 16 64 etc.)

                                ; Remember, A is the exponent.
          XOR   $80             ; toggle sign of exponent
          SRA   A               ; shift right, bit 7 unchanged.
          INC   A               ;
          JR    Z,ASIS          ; forward with say .25 -> .5
          JP    P,ASIS          ; leave increment if value > .5
          DEC   A               ; restore to shift only.

ASIS:     XOR   $80             ; restore sign.
          call __FPSTACK_PUSH   ; Okay, NOW we put the guess on the stack
          rst  28h   ; ROM CALC    ;;guess,x
          DEFB $C3              ;;st-mem-3
          DEFB $02              ;;delete

SQRLOOP:  DEFB  $31             ;;duplicate
          DEFB  $E3             ;;get-mem-3
          DEFB  $C4             ;;st-mem-4
          DEFB  $05             ;;div
          DEFB  $E3             ;;get-mem-3
          DEFB  $0F             ;;addition
          DEFB  $A2             ;;stk-half
          DEFB  $04             ;;multiply
          DEFB  $C3             ;;st-mem-3
          DEFB  $E4             ;;get-mem-4
          DEFB  $03             ;;subtract
          DEFB  $2A             ;;abs
          DEFB  $37             ;;greater-0
          DEFB  $00             ;;jump-true

          DEFB  SQRLOOP - $     ;;to sqrloop

          DEFB  $02             ;;delete
          DEFB  $E3             ;;get-mem-3
          DEFB  $38             ;;end-calc              sqr x.

          jp __FPSTACK_POP

END ASM
END FUNCTION
```