

Floating Point Assembly Language

The floating point unit (FPU) was a separate chip through the 80386+80387. It is now located on-chip, but the programming model still requires most data to be transferred through memory, not between FPU and general purpose registers.

Data types

MASM	C	Format	Range
byte, db	char	8 bits, 2's complement	-128 to 127
word, dw	short	16 bits, 2's complement, little endian	-32768 to 32767
dword, dd	int, long	32 bits, 2's complement, little endian	-2147483648 to 2147483647
qword, dq	long long	64 bits, 2's complement, little endian	+/- 9223372036854775808
real4	float	sign, 8 bit exponent, 23 bit mantissa, implied 1	+/- 1.7e38, 6 significant digits
real8	double	sign, 11 bit exp, 52 bit mant, implied 1	+/- 1e308, 14 significant digits
tbyte, dt	long double	sign, 15 bit exp, 64 bit, explicit leading 1	+/- 1e4932, 18 significant digits

Floating point values can also be +infinity, -infinity, or NaN (not a number).

Registers

			Status word																													
+-----+			+-----+																													
ST	<-- top of stack		B C3 TOS C2 C1 C0 ES SF PE UE OE ZE DE IE																													
+-----+			+-----+																													
ST(1)			15 14 13-11 10 9 8 7 6 5 4 3 2 1 0																													
+-----+			+-----+																													
...			+-----+																													
+-----+			X X X X RC PC X X PM UM OM ZM DM IM																													
+-----+			+-----+																													
			15-12 11-10 9-8 7 6 5 4 3 2 1 0																													
			Control word																													

Stack of 80-bit (tbyte) values
Registers are marked as used, empty,
or invalid.

Status word:

B = busy (not used for on-chip FPU)
C3 = compare equal (zero flag)
TOS = top of stack (000-111)
C2 = comparison undefined (parity flag)
C1 = negative
C0 = compare less than (carry flag)
ES = error summary (one or more bits below are set)
SF = stack overflow/underflow
PE = precision
UE = underflow
OE = overflow
ZE = zero divide
DE = denormalized
IE = invalid

Control word:

X = reserved, do not use
RC = round control: 00 = nearest (even if tie), 01 = down, 10 = up, 11 = to zero
PC = precision control: 00 = real4, 10 = real8, 11 = tbyte (default)
PM, UM, OM, ZM, DM, IM = don't interrupt if corresponding fault occurs

Instructions

All operations are between the top of the stack, ST (also called ST(0)) and either ST(1-7) (ST0-ST7 in NASM) or memory. There are NO general purpose register operands (eax, etc) or immediate operands. ST(n) means n positions below the top of the stack. Thus, a push moves ST to ST(1), ST(1) to ST(2), etc. Pushing a full stack or popping an empty stack is a run time error.

Move

```
fld x      ; push real4, real8, tbyte, convert to tbyte
fild x     ; push word, dword, qword, convert to tbyte
fst x      ; convert ST and copy to real4, real8, tbyte
fst x      ; convert ST and copy to word, dword, qword
```

```

fstp x          ; convert to real and pop
fistp x         ; convert to integer and pop

fych st(n)      ; swap with st(0)

```

Qword integer operands are only valid for load and store, not arithmetic.

Arithmetic

Operands can be signed integers (word or dword), or floating point (real4, real8 or tbyte). All arithmetic is tbyte (80 bits) internally.

```

fadd            ; add st(0) to st(1) and pop (result now in st(0))
fadd st, st(n)  ; add st(1)-st(7) to st(0)
fadd st(n), st   ; add st(0) to st(1-7)
faddp st(n), st ; add to st(n) and pop
fadd x          ; add real x to st
fiadd x         ; add integer word or dword x to st

```

```

; operands are like fadd, faddp, fiadd
fsub, fisub     ; subtract real, integer
fsubr, fisubr   ; subtract in reverse: st(1) = st-st(1), pop
fmul, fimul     ; multiply
fdiv, fidiv     ; divide
fdivr, fidivr   ; divide in reverse
fsubp, fsubrp, fmulp, fdivp, fdivrp ; pop like faddp

```

```

; No explicit operands, result is put in st
; push constants

```

```

fldz           ; push 0
fld1           ; push 1
fldpi          ; push pi
fldl2e         ; push log2(e)
fldl2t         ; push log2(10)
fldlg2         ; push log10(2)
fldln2         ; push ln(2)

```

```

; replace st with result
fabs           ; st = abs(st)
fchs           ; st = -st
frndint        ; round to integer (depends on rounding mode)
fsqrt          ; square root
fcos           ; cosine (radians)
fsin           ; sine
fsincos        ; sine, then push cosine
fptan          ; tangent
fpatan         ; st(1) = arctan(st(1)/st), pop

```

```

; Can be combined to compute exponents
fextract       ; pop st, push exponent, mantissa parts
fscale         ; st *= pow(2, (int)st(1)) (inverse of fextract)
f2xm1         ; pow(2, st) - 1, -1 <= st <= 1
fyl2x          ; st(1) *= log2(st), pop
fyl2xp1        ; st(1) = st(1) * log2(st) + 1, pop

```

Compare

On Pentium 2 and higher, comparison sets carry and zero flags, and parity for undefined (NaN) comparisons. On older processors, use FNSTSW and SAHF to transfer FPU flags to carry, zero, and parity flags.

```

fcom x          ; compare (operands like fadd), set flags C0-C3
fnstsw ax       ; copy flags to AX
sahf            ; copy AH to flags
ja, je, jne, jb ; test CF, ZF flags as if for unsigned int compare

```

```

fcomp, fcompp   ; compare and pop once, twice
ficom x         ; compare with int
ficomp          ; compare with int, pop

```

```

fcomi           ; compare, setting CF, ZF directly (.686)
fcomip          ; compare direct and pop (.686)

```

```

fucom, fucomp, fucompp ; compare allowing unordered (NaN) without interrupt
fucomi, fucomip        ; compare setting CF, ZF, PF directly (.686)
jp                    ; test for unordered compare (parity flag)

```

```
ftst                ; compare st with 0
```

Rounding control

```
fninit              ; empty stack, mask exceptions, set default rounding to nearest
fnstcw x            ; store control word in word x
mov ax, x            ; default value is 037fh
or ax, c00h          ; set rounding control to round to 0 (as in C)
mov x, ax
fldcw x              ; load control word
```

References

[Art of Assembly](#) chapter 11

[NASM Instruction Set](#) (all instructions starting with F)

[Floating point unit](#), J. Loomis.

Material prepared for [CSE 3101](#) by Matt Mahoney, Oct. 25, 2004.