▶ The Floating Point Stack ◀

The 87 has its own register set, of 8 floating point numbers occupying 10 bytes each, plus 14 bytes of status and control information. Many of the 87's instructions cause the numbers to act like a stack, much like a Hewlett-Packard calculator. For this reason, the numbers are called the floating point stack.

The standard name for the top element of the floating point stack is ST0; the others are named ST1 through ST7. Thus, for example, the instruction to add stack element number 3 into the top stack element is usually coded FADD ST0,ST3.

▶ Floating Point Initializations ◀

In general, you use the 87 by loading numbers from 86 memory to the 87 stack (using FLD instructions), calculating on the 87 stack, and storing the results back to 86 memory (using FST and FSTP instructions). There are seven constant numbers built into the 87 instruction set: zero, one, Pi, and four logarithmic conversion constants. These can be loaded using the FLD0, FLD1, FLDPI, FLDL2T, FLDL2E, FLDLG2, and FLDLN2 instructions. All other constants must be declared in, then loaded from, 86 memory. Integer constant words and doublewords can be loaded via FILD. Non-integer constant doubleword, quadwords, and ten-byte numbers can be loaded via FLD.

▶ Floating Point Operand Types ◀

The list of floating point instructions contains a variety of operand types. Here is a brief explanation of those types:

0	stands for the top element of the floating point stack. A synonym for 0 is ST0.
i	stands for element number i of the floating point stack. i can range from 0 through 7. A synonym for i is STi.
mem10r	is a 10-byte memory quantity (typically declared with a DT directive) containing a full precision floating point number.
mem8r	is an 8-byte memory quantity (typically declared with a DQ directive) containing a double precision floating point number.
mem4r	is a 4-byte quantity (typically defined with a DD directive) containing a single precision floating point number.
mem10d	is a 10-byte quantity (also defined via DT) containing a special Binary Coded Decimal format recognized by the FBLD and FBSTP instructions. This format is useful for input and output of floating point numbers.
mem4i	is a 4-byte quantity representing a signed integer in two's-complement notation.
mem2i	is a 2-byte quantity representing a signed integer in two's-complement notation.
mem14	and mem94 are 14- and 94-byte buffers containing the 87 machine state.

▶ The 87 Instruction Set ◀

Following is the 87 instruction set. The "w" in the opcode field is the FWAIT opcode, hex 9B. Again, "0", "1", and "i" stand for the associated floating point stack registers, not constant numbers! Constant numbers in the descriptions are given with decimal points: 0.0, 1.0, 2.0, 10.0.

	Opcode	Instruction	Description
w	D9 F0	F2XM1	0 := (2.0 ** 0) - 1.0
w	DB F1	F4X4	IIT only: 4 by 4 matrix multiply
w	D9 E1	FABS	0 := |0|
w	DE C1	FADD	1 := 1 + 0, pop
w	D8 C0+i	FADD i	0 := i + 0
w	DC C0+i	FADD i,0	i := i + 0
w	D8 C0+i	FADD 0,i	0 := i + 0
w	D8 /0	FADD mem4r	0 := 0 + mem4r
w	DC /0	FADD mem8r	0 := 0 + mem8r
w	DE C0+i	FADDP i,0	i := i + 0, pop
w	DB E8	FBANK 0	IIT only: set bank pointer to default
w	DB EB	FBANK 1	IIT only: set bank pointer to bank 1
w	DB EA	FBANK 2	IIT only: set bank pointer to bank 2
w	DF /4	FBLD mem10d	push, 0 := mem10d
w	DF /6	FBSTP mem10d	mem10d := 0, pop
w	D9 E0	FCHS	0 := -0
9B	DB E2	FCLEX	clear exceptions
w	D8 D1	FCOM	compare 0 - 1
w	D8 D0+i	FCOM 0,i	compare 0 - i
w	D8 D0+i	FCOM i	compare 0 - i
w	D8 /2	FCOM mem4r	compare 0 - mem4r
w	DC /2	FCOM mem8r	compare 0 - mem8r
w	D8 D9	FCOMP	compare 0 - 1, pop
w	D8 D8+i	FCOMP 0,i	compare 0 - i, pop
w	D8 D8+i	FCOMP i	compare 0 - i, pop
w	D8 /3	FCOMP mem4r	compare 0 - mem4r, pop
w	DC /3	FCOMP mem8r	compare 0 - mem8r, pop
w	DE D9	FCOMPP	compare 0 - 1, pop both
w	D9 FF	FCOS	387 only: push, 1/0 := cosine(old 0)
w	D9 F6	FDECSTP	decrement stack pointer
w	DB E1	FDISI	disable interrupts (.287 ignore)
w	DE F9	FDIV	1 := 1 / 0, pop
w	D8 F0+i	FDIV i	0 := 0 / i
w	DC F8+i	FDIV i,0	i := i / 0
w	D8 F0+i	FDIV 0,i	0 := 0 / i
w	D8 /6	FDIV mem4r	0 := 0 / mem4r
w	DC /6	FDIV mem8r	0 := 0 / mem8r
w	DE F8+i	FDIVP i,0	i := i / 0, pop
w	DE F1	FDIVR	1 := 0 / 1, pop
w	D8 F8+i	FDIVR i	0 := i / 0
w	DC F0+i	FDIVR i,0	i := 0 / i
w	D8 F8+i	FDIVR 0,i	0 := i / 0
w	D8 /7	FDIVR mem4r	0 := mem4r / 0
w	DC /7	FDIVR mem8r	0 := mem8r / 0
w	DE F0+i	FDIVRP i,0	i := 0 / i, pop
w	DB E0	FENI	enable interrupts (.287 ignore)
w	DD C0+i	FFREE i	empty i
w	DE /0	FIADD mem2i	0 := 0 + mem4i
w	DA /0	FIADD mem4i	0 := 0 + mem2i
w	DE /2	FICOM mem2i	compare 0 - mem2i
w	DA /2	FICOM mem4i	compare 0 - mem4i
w	DE /3	FICOMP mem2i	compare 0 - mem2i, pop
w	DA /3	FICOMP mem4i	compare 0 - mem4i, pop
w	DE /6	FIDIV mem2i	0 := 0 / mem2i
w	DA /6	FIDIV mem4i	0 := 0 / mem4i
w	DE /7	FIDIVR mem2i	0 := mem2i / 0
w	DA /7	FIDIVR mem4i	0 := mem4i / 0
w	DF /0	FILD mem2i	push, 0 := mem2i
w	DB /0	FILD mem4i	push, 0 := mem4i
w	DF /5	FILD mem8i	push, 0 := mem8i
w	DE /1	FIMUL mem2i	0 := 0 * mem2i
w	DA /1	FIMUL mem4i	0 := 0 * mem4i
w	D9 F7	FINCSTP	increment stack pointer
9B	DB E3	FINIT	initialize 87
w	DF /2	FIST mem2i	mem2i := 0
w	DB /2	FIST mem4i	mem4i := 0
w	DF /3	FISTP mem2i	mem2i := 0, pop
w	DB /3	FISTP mem4i	mem4i := 0, pop
w	DF /7	FISTP mem8i	mem8i := 0, pop
w	DE /4	FISUB mem2i	0 := 0 - mem2i
w	DA /4	FISUB mem4i	0 := 0 - mem4i
w	DE /5	FISUBR mem2i	0 := mem2i - 0
w	DA /5	FISUBR mem4i	0 := mem4i - 0
w	D9 C0+i	FLD i	push, 0 := old i
w	DB /5	FLD mem10r	push, 0 := mem10r
w	D9 /0	FLD mem4r	push, 0 := mem4r
w	DD /0	FLD mem8r	push, 0 := mem8r
w	D9 E8	FLD1	push, 0 := 1.0
w	D9 /5	FLDCW mem2i	control word := mem2i
w	D9 /4	FLDENV mem14	environment := mem14
w	D9 EA	FLDL2E	push, 0 := log base 2.0 of e
w	D9 E9	FLDL2T	push, 0 := log base 2.0 of 10.0
w	D9 EC	FLDLG2	push, 0 := log base 10.0 of 2.0
w	D9 ED	FLDLN2	push, 0 := log base e of 2.0
w	D9 EB	FLDPI	push, 0 := Pi
w	D9 EE	FLDZ	push, 0 := +0.0
w	DE C9	FMUL	1 := 1 * 0, pop
w	D8 C8+i	FMUL i	0 := 0 * i
w	DC C8+i	FMUL i,0	i := i * 0
w	D8 C8+i	FMUL 0,i	0 := 0 * i
w	D8 /1	FMUL mem4r	0 := 0 * mem4r
w	DC /1	FMUL mem8r	0 := 0 * mem8r
w	DE C8+i	FMULP i,0	i := i * 0, pop
	DB E2	FNCLEX	nowait clear exceptions
	DB E1	FNDISI	disable interrupts (.287 ignore)
	DB E0	FNENI	enable interrupts (.287 ignore)
	DB E3	FNINIT	nowait initialize 87
w	D9 D0	FNOP	no operation
	DD /6	FNSAVE mem94	mem94 := 87 state
	D9 /7	FNSTCW mem2i	mem2i := control word
	D9 /6	FNSTENV mem14	mem14 := environment
	DF E0	FNSTSW AX	AX := status word
	DD /7	FNSTSW mem2i	mem2i := status word
w	D9 F3	FPATAN	0 := arctan(1/0), pop
w	D9 F8	FPREM	0 := REPEAT(0 - 1)
w	D9 F5	FPREM1	387 only: 0 := REPEAT(0 - 1) IEEE compat.
w	D9 F2	FPTAN	push, 1/0 := tan(old 0)
w	D9 FC	FRNDINT	0 := round(0)
w	DD /4	FRSTOR mem94	87 state := mem94
w	DD /6	FSAVE mem94	mem94 := 87 state
w	D9 FD	FSCALE	0 := 0 * 2.0 ** 1
9B	DB E4	FSETPM	set protection mode
w	D9 FE	FSIN	387 only: push, 1/0 := sine(old 0)
w	D9 FB	FSINCOS	387 only: push, 1 := sine, 0 := cos(old 0)
w	D9 FA	FSQRT	0 := square root of 0
w	DD D0+i	FST i	i := 0
w	D9 /2	FST mem4r	mem4r := 0
w	DD /2	FST mem8r	mem8r := 0
w	D9 /7	FSTCW mem2i	mem2i := control word
w	D9 /6	FSTENV mem14	mem14 := environment
w	DD D8+i	FSTP i	i := 0, pop
w	DB /7	FSTP mem10r	mem10r := 0, pop
w	D9 /3	FSTP mem4r	mem4r := 0, pop
w	DD /3	FSTP mem8r	mem8r := 0, pop
w	DF E0	FSTSW AX	AX := status word
w	DD /7	FSTSW mem2i	mem2i := status word
w	DE E9	FSUB	1 := 1 - 0, pop
w	D8 E0+i	FSUB i	0 := 0 - i
w	DC E8+i	FSUB i,0	i := i - 0
w	D8 E0+i	FSUB 0,i	0 := 0 - i
w	D8 /4	FSUB mem4r	0 := 0 - mem4r
w	DC /4	FSUB mem8r	0 := 0 - mem8r
w	DE E8+i	FSUBP i,0	i := i - 0, pop
w	DE E1	FSUBR	1 := 0 - 1, pop
w	D8 E8+i	FSUBR i	0 := i - 0
w	DC E0+i	FSUBR i,0	i := 0 - i
w	D8 E8+i	FSUBR 0,i	0 := i - 0
w	D8 /5	FSUBR mem4r	0 := mem4r - 0
w	DC /5	FSUBR mem8r	0 := mem8r - 0
w	DE E0+i	FSUBRP i,0	i := 0 - i, pop
w	D9 E4	FTST	compare 0 - 0.0
w	DD E0+i	FUCOM i	387 only: unordered compare 0 - i
w	DD E1	FUCOM	387 only: unordered compare 0 - 1
w	DD E8+i	FUCOMP i	387 only: unordered compare 0 - i, pop
w	DD E9	FUCOMP	387 only: unordered compare 0 - 1, pop
w	DA E9	FUCOMPP	387 only: unordered compare 0 - 1, pop both
9B		FWAIT	wait for 87 ready
w	D9 E5	FXAM	C3 -- C0 := type of 0
w	D9 C9	FXCH	exchange 0 and 1
w	D9 C8+i	FXCH 0,i	exchange 0 and i
w	D9 C8+i	FXCH i	exchange 0 and i
w	D9 C8+i	FXCH i,0	exchange 0 and i
w	D9 F4	FXTRACT	push, 1 := expo, 0 := sig
w	D9 F1	FYL2X	0 := 1 * log base 2.0 of 0, pop
w	D9 F9	FYL2XP1	0 := 1 * log base 2.0 of (0+1.0), pop