PŘENOSOVÉ INSTRUKCE		
FLD	src mem32/64 st(i)	Load Float *src → ST(0)*
FILD	src mem16/32/64	Load Integer *src → ST(0)*
FST(P)	dest mem32/64 st(i)	Store Real *ST(0) → dst* *if *P then POP*
FIST(P) P	dst mem16/32 mem16/32/64	Store Integer *ST(0) → dst* *if *P then POP*
FXCH	dst st(i) none ⇒ ST(1)	Exchange Register Contents *ST(0) ⟷ st(i)*
KONSTANTY *const → ST(0)*		
FLD1		Load 1
FLDZ		Load 0
FLDPI		Load π
FLDL2T		Load $\log_2(10)$
FLDL2E		Load $\log_2(e)$
FLDLG2		Load $\log_{10}(2)$
FLDLN2		Load ln(2)
PUSH FLD	__FLOAT32__(fimm) DWORD [ESP]	Load any constant
ARITMETICKÉ INSTRUKCE (kombinace a principy v F(I)ADD(P) - jsou stejné)		
FADD	dst, src none ⇒ ST(0), mem32/64 st(i), ST(0) ST(0), st(i)	Addition *dst + src → dest*
FADDP	dst, src st(i), ST(0) none ⇒ ST(1), none ⇒ ST(0)	Addition (with pop) *dst + src → dest* *POP*
FIADD	src mem16/32	Integer Addition *ST(0) + src → ST(0)*
FSUB	dst, src none ⇒ ST(0), mem32/64 st(i), ST(0) ST(0), st(i)	Subtraction *dst - src → dest*
FSUBP	dst, src st(i), ST(0) none ⇒ ST(1), none ⇒ ST(0)	Addition (/w pop) *dst - src → dest* *POP*
FISUB	src mem16/32	Integer Subtraction *ST(0) + src → ST(0)*
FSUBR(P) FISUBR	dst, src src	Reverse Subtraction (same princ. and comb. as FSUB) *src - dst → dst*

FMUL(P) dst, src FIMUL src	Multiplication *dst * src → dst*
FDIV(P) dst, src FIDIV src	Division *dst / src → dst*
FDIVR(P) dst, src FIDIVR src	Reverse Division *src / dst → dst*
FABS	Absolute Value *|ST(0)| → ST(0)*
FCHS	Change Sign *-ST(0) → ST(0)*
FSQRT	Square Root *√ST(0) → ST(0)*
FRNDINT	Round to integer *Round(ST(0)) → ST(0)*
POROVNÁVACÍ INSTRUKCE	
FCOM(P) src mem32/64 st(i) none ⇒ ST(1)	Compare *ST(0) > src : 000 → C3,C2,C0* *ST(0) < src : 001 → C3,C2,C0* *ST(0) = src : 100 → C3,C2,C0*
FCOMPP src none ⇒ ST(1)	Compare and pop both *(same as FCOM, but pops both)*
FICOM(P) src mem16/32	Compare with integer *(same as FCOM)*
FCOMI(P) src st(i)	Compare and set EFLAGS *ST(0) > src : 000 → ZF,PF,CF* *ST(0) < src : 001 → ZF,PF,CF* *ST(0) = src : 100 → ZF,PF,CF*
FTST	Test *ST(0) > 0 : 000 → C3,C2,C0* *ST(0) < 0 : 001 → C3,C2,C0* *ST(0) = 0 : 100 → C3,C2,C0*
FXAM	Examine *unsupported : 000 → C3,C2,C0* *NaN : 001 → C3,C2,C0* *normal : 010 → C3,C2,C0* *infinity : 011 → C3,C2,C0* *zero : 100 → C3,C2,C0* *empty : 101 → C3,C2,C0* *denormal : 110 → C3,C2,C0*
TRANSCENDENTNÍ INSTRUKCE	
FSIN	Sine *if |ST(0)| < 2^{63} then 1 → C2* *else sin(ST(0)) → ST(0)*
FCOS	Cosine *if |ST(0)| < 2^{63} then 1 → C2* *else cos(ST(0)) → ST(0)*

FSINCOS	Sine and Cosine *if* $\lvert ST(0) \rvert < 2^{63}$ *then* 1 → C2 *else* sin(ST(0)) → ST(1) *and* cos(ST(0)) → ST(0)
FPTAN	Partial Tangent *if* $\lvert ST(0) \rvert < 2^{63}$ *then* 1 → C2 *else* tan(ST(0)) → ST(1) *and* 1 → ST(0)
FPATAN	Partial Arctangent *arctan(ST(1)/ST(0))* → *ST(1)* *POP*
F2XM1	Compute 2^X-1 *if ST(0) in* ‹-1,1› *then* $2^{ST(0)}$ - 1 → ST(0)
FYL2X	Compute y * $\log_2 X$ *if ST(0) > 0 then* *ST(1) * log2(ST(0))* → *ST(1)* *and POP*
ŘÍDÍCÍ INSTRUKCE (modifier N = nečeká na spracování FPU výjimek)	
~~F(N)INIT~~	~~Set FPU to initial state~~
F(N)STCW dst mem16	Store Control Word *FPUControlWord* → *dst*
FLCDW src mem16	Load Control Word *src* → *FPUControlWord*
F(N)STSW dst mem16 AX	Store Status Word *FPUStatusWord* → *dst*
F(N)STENV dst mem14B/28B	Store FPU Environment *FPUEnvironment* → *dst*
FLDENV src mem14B/28B	Load FPU Environment *src* → *FPUEnvironment*
F(N)SAVE dst mem94B/108B	Store FPU State *FPUState* → *dst* *FINIT*
FRSTOR src mem94B/108B	Restore FPU State src → FPUState
FFREE	Free FPU Registers
FWAIT	Wait (for sync. of CPU and FPU)
FNOP	No Operation
FINCSTP	Increment Stack-Top Pointer
FDECSTP	Decrement Stack-Top Pointer