12. Soubor instrukcí FPU.

Instrukční soubor FPU procesorů Pentium:

- přenosové instrukce
- aritmetické instrukce
- porovnávací instrukce
- transcendentní instrukce
- řídící instrukce

Zásady pro názvy instrukcí FPU:

- Všechny instrukce FPU začínají písmenem **F**.
- Je-li druhým písmenem instrukce písmeno **I**, pak instrukce pracuje s celočíselným (Integer) operandem.
- Je-li druhým písmenem instrukce písmeno **B**, pak instrukce pracuje s binárně kódovaným dekadickým (BCD) operandem.
- Je-li druhým písmenem instrukce písmeno **N**, pak procesor nečeká na dokončení této instrukce a pracuje paralelně s FPU.
- Je-li posledním písmenem instrukce písmeno **P**, pak se po provedení této instrukce odstraní vrchní položka ze zásobníku FPU.

Přenosové instrukce

- **FLD**
- **FILD**
- FBLD
- **FST**
- **FIST**
- **FSTP**
- **FISTP**
- FBSTP
- **FXCH**
- FCMOVcc
- **FLD1**
- FLDL2T
- FLDL2E
- **FLDPI**
- FLDLG2
- FLDLN2
- **FLDZ**

FLD, FILD, FBLD

FLD src (Load Real)

D9 /0	FLD m32real
DD /0	FLD m64real
DB /5	FLD m80real
D9 C0+i	FLD ST(i)

```
if src is ST(i) then ST(i) → temp
top – 1 → top            {if old top is 0 then new top will be 7}
if src is ST(i)
    then temp → ST(0)
    else ExtendedReal(src) → ST(0)

if stack overflow occurred then 1 → C1 else 0 → C1
C3, C2, C0 are not defined
```

FILD src (Load Integer)

DF /0	FILD m16int
DB /0	FILD m32int
DF /5	FILD m64int

top – 1 $\rightarrow$ top {if old top is 0 then new top will be 7}
ExtendedReal(src) $\rightarrow$ ST(0)

if stack overflow occurred then 1 $\rightarrow$ C1 else 0 $\rightarrow$ C1
C3, C2, C0 are not defined

FBLD src (Load BCD)

DF /4	FBLD m80bcd

top – 1 $\rightarrow$ top {if old top is 0 then new top will be 7}
ExtendedReal(src) $\rightarrow$ ST(0)

if stack overflow occurred then 1 $\rightarrow$ C1 else 0 $\rightarrow$ C1
C3, C2, C0 are not defined

FST, FIST, FSTP, FISTP, FBSTP

FST dest (Store Real)

D9 /2	FST m32real
DD /2	FST m64real
DD D0+i	FST ST(i)

ST(0) $\rightarrow$ dest

Při ukládání do paměti je hodnota konvertována na Single Real, resp. Double Real formát.

```
if stack underflow occurred then 0 → C1
                            else C1 indicates rounding direction
C3, C2, C0 are not defined
```

FIST dest (Store Integer)

DF /2	FIST m16int
DB /2	FIST m32int

ST(0) $\rightarrow$ dest

Hodnota je konvertována na Word Integer, resp. Short Integer.

```
if stack underflow occurred then 0 → C1
                            else C1 indicates rounding direction
C3, C2, C0 are not defined
```

FSTP dest (Store Real and Pop)

D9 /3	FSTP m32real
DD /3	FSTP m64real
DB /7	FSTP m80real
DD D8+i	FSTP ST(i)

```
ST(0) → dest
PopRegisterStack          {if old top is 7 then new top will be 0}
```

Při ukládání do paměti je hodnota konvertována na Single Real, resp. Double Real formát.

```
if stack underflow occurred then 0 → C1,
                            else C1 indicates rounding direction
C3, C2, C0 are not defined
```

FISTP dest (Store Integer and Pop)

DF /3	FISTP m16int
DB /3	FISTP m32int
DF /7	FISTP m64int

```
ST(0) → dest
PopRegisterStack                {if old top is 7 then new top will be 0}
```

Hodnota je konvertována na Word Integer, Short Integer, resp. Long Integer formát.

```
if stack underflow occurred then 0 → C1
                               else C1 indicates rounding direction
C3, C2, C0 are not defined
```

FBSTP dest (Store BCD and Pop)

DF /6 FBSTP m80bcd

```
ST(0) → dest
PopRegisterStack                {if old top is 7 then new top will be 0}
```

Hodnota je konvertována na Packed BCD formát.

```
if stack underflow occurred then 0 → C1
                                 else C1 indicates rounding direction
C3, C2, C0 are not defined
```

FXCH

FXCH [dest] (Exchange Register Contents)

D9 C8+i	FXCH ST(i)	
D9 C9	FXCH	implicitním operandem je ST(1)

$ST(0) \rightarrow temp$
$ST(i) \rightarrow ST(0)$
$temp \rightarrow ST(i)$

if stack underflow occurred then $0 \rightarrow C1$ else $0 \rightarrow C1$
C3, C2, C0 are not defined

FCMOVcc

FCMOVcc ST(0),ST(i) (Float Point Conditional Move Pentium® Pro)

DA C0+i	FCMOVB ST(0),ST(i)	Move if below (CF=1)
DA C8+i	FCMOVE ST(0),ST(i)	Move if equal (ZF=1)
DA D0+i	FCMOVBE ST(0),ST(i)	Move if below or equal (CF=1 or ZF=1)
DA D8+i	FCMOVU ST(0),ST(i)	Move if unordered (PF=1)
DB C0+i	FCMOVNB ST(0),ST(i)	Move if not below (CF=0)
DB C8+i	FCMOVNE ST(0),ST(i)	Move if not equal (ZF=0)
DB D0+i	FCMOVNBE ST(0),ST(i)	Move if not below or equal (CF=0 and ZF=0)
DB D8+i	FCMOVNU ST(0),ST(i)	Move if not unordered (PF=0)

if cc then ST(i) $\rightarrow$ ST(0)

if stack underflow occurred then 0 $\rightarrow$ C1,
C3, C2, C0 are not defined

FLD1, FLDL2T, FLDL2E, **FLDPI**, FLDLG2, FLDLN2, **FLDZ**

FLD1/FLDL2T/FLDL2E/FLDPI/FLDLG2/FLDLN2/FLDZ (Load Constant)

D9 E8	FLD1	Constant = 1.0
D9 E9	FLDL2T	Constant = $\log_2 10$
D9 EA	FLDL2E	Constant = $\log_2 e$
D9 EB	FLDPI	Constant = π
D9 EC	FLDLG2	Constant = $\log_{10} 2$
D9 ED	FLDLN2	Constant = $\log_e 2$
D9 EE	FLDZ	Constant = 0.0

top – 1 → top {if old top is 0 then new top will be 7}
Constant → ST(0)

if stack overflow occurred then 1 → C1 else 0 → C1,
C3, C2, C0 are not defined

Aritmetické instrukce

- **FADD**
- **FADDP**
- **FIADD**
- **FSUB**
- **FSUBP**
- **FISUB**
- **FSUBR**
- **FSUBRP**
- **FISUBR**
- **FMUL**
- **FMULP**
- **FIMUL**
- **FDIV**
- **FDIVP**
- **FIDIV**
- **FDIVR**
- **FDIVRP**
- **FIDIVR**
- FPREM
- FPREM1
- **FABS**
- **FCHS**
- **FSQRT**
- FRNDINT
- FSCALE
- FXTRACT

FADD, FADDP, FIADD, FSUB, FSUBP, FISUB, FSUBR, FSUBRP, FISUBR, FMUL, FMULP, FIMUL, FDIV, FDIVP, FIDIV, FDIVR, FDIVRP, FIDIVR

FADD/FADDP/FIADD [[dest,]src] (Add)

D8 /0	FADD m32real	$dest = \mathrm{ST}(0)$
DC /0	FADD m64real	$dest = \mathrm{ST}(0)$
D8 C0+i	FADD ST(0),ST(i)	
DC C0+i	FADD ST(i),ST(0)	
DE C0+i	FADDP ST(i),ST(0)	
DE C1	FADDP	$dest = \mathrm{ST}(1), src = \mathrm{ST}(0)$
DA /0	FIADD m32int	$dest = \mathrm{ST}(0)$
DE /0	FIADD m16int	$dest = \mathrm{ST}(0)$

```
dest + ExtendedReal(src) $\rightarrow$ dest
if FADDP then PopRegisterStack
                        {if old top is 7 then new top will be 0}
if stack underflow occurred then 0 $\rightarrow$ C1,
                        else C1 indicates rounding direction
C3, C2, C0 are not defined
```

FSUB/FSUBP/FISUB [[dest,]src] (Subtract)

D8 /4	FSUB m32real	$dest = \mathrm{ST}(0)$
DC /4	FSUB m64real	$dest = \mathrm{ST}(0)$
D8 E0+i	FSUB ST(0),ST(i)	
DC E8+i	FSUB ST(i),ST(0)	
DE E8+i	FSUBP ST(i),ST(0)	
DE E9	FSUBP	$dest = \mathrm{ST}(1), src = \mathrm{ST}(0)$
DA /4	FISUB m32int	$dest = \mathrm{ST}(0)$
DE /4	FISUB m16int	$dest = \mathrm{ST}(0)$

```
dest – ExtendedReal(src) → dest
if FSUBP then PopRegisterStack
                                        {if old top is 7 then new top will be 0}
if stack underflow occurred then 0 → C1
                                     else C1 indicates rounding direction
C3, C2, C0 are not defined
```

FSUBR/FSUBRP/FISUBR [[dest,]src] (Reverse Subtract)

D8 /5	FSUBR m32real	$dest = \mathrm{ST}(0)$
DC /5	FSUBR m64real	$dest = \mathrm{ST}(0)$
D8 E8+i	FSUBR ST(0),ST(i)	
DC E0+i	FSUBR ST(i),ST(0)	
DE E0+i	FSUBRP ST(i),ST(0)	
DE E1	FSUBRP	$dest = \mathrm{ST}(1), src = \mathrm{ST}(0)$
DA /5	FISUBR m32int	$dest = \mathrm{ST}(0)$
DE /5	FISUBR m16int	$dest = \mathrm{ST}(0)$

```
ExtendedReal(src) – dest → dest
if FSUBRP then PopRegisterStack
                              {if old top is 7 then new top will be 0}
if stack underflow occurred then 0 → C1,
                           else C1 indicates rounding direction
C3, C2, C0 are not defined
```

FMUL/FMULP/FIMUL [[dest,]src] (Multiply)

D8 /1	FMUL m32real	$dest = \mathrm{ST}(0)$
DC /1	FMUL m64real	$dest = \mathrm{ST}(0)$
D8 C8+i	FMUL ST(0),ST(i)	
DC C8+i	FMUL ST(i),ST(0)	
DE C8+i	FMULP ST(i),ST(0)	
DE C9	FMULP	$dest = \mathrm{ST}(1), src = \mathrm{ST}(0)$
DA /1	FIMUL m32int	$dest = \mathrm{ST}(0)$
DE /1	FIMUL m16int	$dest = \mathrm{ST}(0)$

```
dest $*$ ExtendedReal(src) $\rightarrow$ dest
if FMULP then PopRegisterStack
                        {if old top is 7 then new top will be 0}
if stack underflow occurred then 0 $\rightarrow$ C1,
                        else C1 indicates rounding direction
C3, C2, C0 are not defined
```

FDIV/FDIVP/FIDIV [[dest,]src] (Divide)

D8 /6	FDIV m32real	$dest = \text{ST}(0)$
DC /6	FDIV m64real	$dest = \text{ST}(0)$
D8 F0+i	FDIV ST(0),ST(i)	
DC F8+i	FDIV ST(i),ST(0)	
DE F8+i	FDIVP ST(i),ST(0)	
DE F9	FDIVP	$dest = \text{ST}(1), src = \text{ST}(0)$
DA /6	FIDIV m32int	$dest = \text{ST}(0)$
DE /6	FIDIV m16int	$dest = \text{ST}(0)$

```
dest / ExtendedReal(src) $\rightarrow$ dest
if FDIVP then PopRegisterStack
                                {if old top is 7 then new top will be 0}
if stack underflow occurred then 0 $\rightarrow$ C1,
                             else C1 indicates rounding direction
C3, C2, C0 are not defined
```

FDIVR/FDIVRP/FIDIVR [[dest,]src]   (Reverse Divide)

D8 /7	FDIVR m32real	*dest* = ST(0)
DC /7	FDIVR m64real	*dest* = ST(0)
D8 F8+i	FDIVR ST(0),ST(i)	
DC F0+i	FDIVR ST(i),ST(0)	
DE F0+i	FDIVRP ST(i),ST(0)	
DE F1	FDIVRP	*dest* = ST(1), *src* = ST(0)
DA /7	FIDIVR m32int	*dest* = ST(0)
DE /7	FIDIVR m16int	*dest* = ST(0)

```
ExtendedReal(src) / dest $\rightarrow$ dest
if FDIVRP then PopRegisterStack
                    {if old top is 7 then new top will be 0}
if stack underflow occurred then 0 $\rightarrow$ C1,
                    else C1 indicates rounding direction
C3, C2, C0 are not defined
```

FPREM, FPREM1

FPREM (Partial Remainder)

D9 F8 FPREM

```
exponent(ST(0)) – exponent(ST(1)) $\rightarrow$ D
if D < 64 then begin
   Integer(TruncateTowardZero(ST(0) / ST(1))) $\rightarrow$ Q
   ST(0) – (ST(1) * Q) $\rightarrow$ ST(0)
   0 $\rightarrow$ C2
   LeastSignificantBits(Q): Q2, Q1, Q0 $\rightarrow$ C0, C3, C1
end
else begin
   1 $\rightarrow$ C2
   an implementation-dependent number between 32 and 63 $\rightarrow$ N
   Integer(TruncateTowardZero((ST(0) / ST(1)) / $2^{(D - N)}$ )) $\rightarrow$ QQ
   ST(0) – (ST(1) * QQ * $2^{(D - N)}$ ) $\rightarrow$ ST(0)
end;
if stack underflow occurred then 0 $\rightarrow$ C1
```

FPREM1 (Partial Remainder, 387)

D9 F5 FPREM1

```
exponent(ST(0)) – exponent(ST(1)) → D
if D < 64 then begin
   Integer(RoundTowardNearestInteger(ST(0) / ST(1))) → Q
   ST(0) – (ST(1) * Q) → ST(0)
   0 → C2
   LeastSignificantBits(Q): Q2, Q1, Q0 → C0, C3, C1
end
else begin
   1 → C2
   an implementation-dependent number between 32 and 63 → N
   Integer(TruncateTowardZero((ST(0) / ST(1)) / $2^{(D - N)}$)) → QQ
   ST(0) – (ST(1) * QQ * $2^{(D - N)}$ ) → ST(0)
end;
if stack underflow occurred then 0 → C1
```

FABS, FCHS, FSQRT

FABS (Absolute Value)

D9 E1 FABS

$|ST(0)| \rightarrow ST(0)$

if stack underflow occurred then $0 \rightarrow C1$ else $0 \rightarrow C1$
C3, C2, C0 are not defined

FCHS (Change Sign)

D9 E0 FCHS

$- ST(0) \rightarrow ST(0)$

if stack underflow occurred then $0 \rightarrow C1$ else $0 \rightarrow C1$
C3, C2, C0 are not defined

FSQRT (Square Root)

D9 FA FSQRT

$\sqrt{\mathrm{ST(0)}} \rightarrow \mathrm{ST(0)}$

```
if stack underflow occurred then 0 → C1
                            else C1 indicates rounding direction
```

C3, C2, C0 are not defined

FRNDINT, FSCALE, FXTRACT

FRNDINT (Round to Integer)

D9 FC FRNDINT

Integer(RoundTowardNearestInteger(ST(0))) → ST(0)

if stack underflow occurred then 0 → C1
else C1 indicates rounding direction

C3, C2, C0 are not defined

FSCALE (Scale)

D9 FD FSCALE

$ST(0) * 2^{ST(1)} \rightarrow ST(0)$

if stack underflow occurred then $0 \rightarrow C1$
else C1 indicates rounding direction

C3, C2, C0 are not defined

FXTRACT (Extract Exponent and Significant)

D9 F4 FXTRACT

Significand(ST(0)) $\rightarrow$ temp
Exponent(ST(0)) $\rightarrow$ ST(0)
top – 1 $\rightarrow$ top
temp $\rightarrow$ ST(0)

if stack underflow occurred then 0 $\rightarrow$ C1
if stack overflow occurred then 1 $\rightarrow$ C1
C3, C2, C0 are not defined

Porovnávací instrukce

- **FCOM**
- **FCOMP**
- **FCOMPP**
- **FICOM**
- **FICOMP**
- FUCOM
- FUCOMP
- FUCOMPP
- FCOMI
- FCOMIP
- FUCOMI
- FUCOMIP
- FTST
- FXAM

FCOM, FCOMP, FCOMPP

FCOM/FCOMP/FCOMPP [src] (Compare Real)

D8 /2	FCOM m32real	
DC /2	FCOM m64real	
D8 D0+i	FCOM ST(i)	
D8 D1	FCOM	*src* = ST(1)
D8 /3	FCOMP m32real	
DC /3	FCOMP m64real	
D8 D8+i	FCOMP ST(i)	
D8 D9	FCOMP	*src* = ST(1)
DE D9	FCOMPP	*src* = ST(1).

FCOM/FCOMP/FCOMPP [src] Continuation

```
case (relation of operands) of
  ST(0) > src : 000 → C3, C2, C0
  ST(0) < src : 001 → C3, C2, C0
  ST(0) = scr : 100 → C3, C2, C0
end
if (any operand is NaN or unsupported format) and (IM=1)
then  111 → C3, C2, C0                    {IM … viz řídící registr FPU}
if FCOMP or FCOMPP
then PopRegisterStack             {if old top is 7 then new top will be 0}
if FCOMPP
then PopRegisterStack             {if old top is 7 then new top will be 0}

if stack underflow occurred then 0 → C1 else 0 → C1
```

SW:

15	14	13	12	11	10	9	8
B	C3	TOP			C2	C1	C0

FLAGS:

7	6	5	4	3	2	1	0
SF	ZF	0	AF	0	PF	1	CF

ST(0) > src : 000 $\rightarrow$ C3, C2, C0

ST(0) < src : 001 $\rightarrow$ C3, C2, C0 (**CF**)

ST(0) = scr : 100 $\rightarrow$ C3, C2, C0 (**ZF**)

unordered 111 $\rightarrow$ C3, C2, C0 (ZF, **PF**, CF)

Skoky: JA, JB, JE a jejich negace a kombinace (přestože se v FPU porovnávají čísla se znaménky) !!!!

FICOM, FICOMP

FICOM/FICOMP [src] (Compare Integer)

DE /2	FICOM m16int
DA /2	FICOM m32int
DE /3	FICOMP m16int
DA /3	FICOMP m32int

FICOM/FICOMP [src]   Continuation

```
case (relation of operands) of
  ST(0) > src : 000 → C3, C2, C0
  ST(0) < src : 001 → C3, C2, C0
  ST(0) = scr : 100 → C3, C2, C0
end
if (any operand is NaN or unsupported format) and (IM=1)
then  111 → C3, C2, C0                    {IM … viz řídící registr FPU}
if FICOMP
then PopRegisterStack            {if old top is 7 then new top will be 0}

if stack underflow occurred then 0 → C1 else 0 → C1
```

FUCOM, FUCOMP, FUCOMPP

FUCOM/FUCOMP/FUCOMPP [src]
(Unordered Compare Real)

DD E0+i	FUCOM ST(i)	
DD E1	FUCOM	*src* = ST(1)
DD E8+i	FUCOMP ST(i)	
DD E9	FUCOMP	*src* = ST(1)
DA E9	FUCOMPP	*src* = ST(1)

Instrukce FUCOM/FUCOMP/FUCOMPP pracují stejně jako instrukce FCOM/FCOMP/FCOMPP; jediným rozdílem je, že je-li libovolným operandem tzv. Q-NaN (výklad přesahuje rámec předmětu) negenerují výjimku IE (neplatná operace, viz stavový registr FPU).

FUCOM/FUCOMP/FUCOMPP [src] Continuation

```
case (relation of operands) of
  ST(0) > src :    000 → C3, C2, C0
  ST(0) < src :    001 → C3, C2, C0
  ST(0) = scr :    100 → C3, C2, C0
end
if (any operand is Q-NaN) or
((any operand is NaN or unsupported format) and (IM=1))
then  111 → C3, C2, C0                      {IM … viz řídící registr FPU}
if FUCOMP or FUCOMPP
then PopRegisterStack             {if old top is 7 then new top will be 0}
if FUCOMPP
then PopRegisterStack             {if old top is 7 then new top will be 0}

if stack underflow occurred then 0 → C1
```

FCOMI, FCOMIP, FUCOMI, FUCOMI

FCOMI/FCOMIP/FUCOMI/FUCOMIP ST(i)
(Compare Real and Set EFLAGS, Pentium® Pro)

DB F0+i	FCOMI ST(i)
DF F0+i	FCOMIP ST(i)
DB E8+i	FUCOMI ST(i)
DF E8+i	FUCOMIP ST(i)

Instrukce FCOMI/FCOMIP, resp. FUCOMI/FUCOMIP pracují stejně jako instrukce FCOM/FCOMP, resp. FUCOM/FUCOMP, místo podmínkového kódu ve stavovém registru FPU však nastavují přímo příznaky ZF, PF a CF v registru (E)FLAGS.

FCOMI/FCOMIP/FUCOMI/FUCOMIP ST(i) (Continuation)

```
case (relation of operands) of
  ST(0) > ST(i) :  000 → ZF, PF, CF
  ST(0) < ST(i) :  001 → ZF, PF, CF
  ST(0) = ST(i) :  100 → ZF, PF, CF
  unordered     :  111 → ZF, PF, CF
end
if FCOMIP or FUCOMIP then PopRegisterStack
                        {if old top is 7 then new top will be 0}

if stack underflow occurred then 0 → C1 else 0 → C1
```

FTST

FTST (Test)

D9 E4 FTST

```
case (relation of operands) of
  ST(0) > 0.0 :  000 → C3, C2, C0
  ST(0) < 0.0 :  001 → C3, C2, C0
  ST(0) = 0.0 :  100 → C3, C2, C0
   unordered  :  111 → C3, C2, C0
end

if stack underflow occurred then 0 → C1 else 0 → C1
```

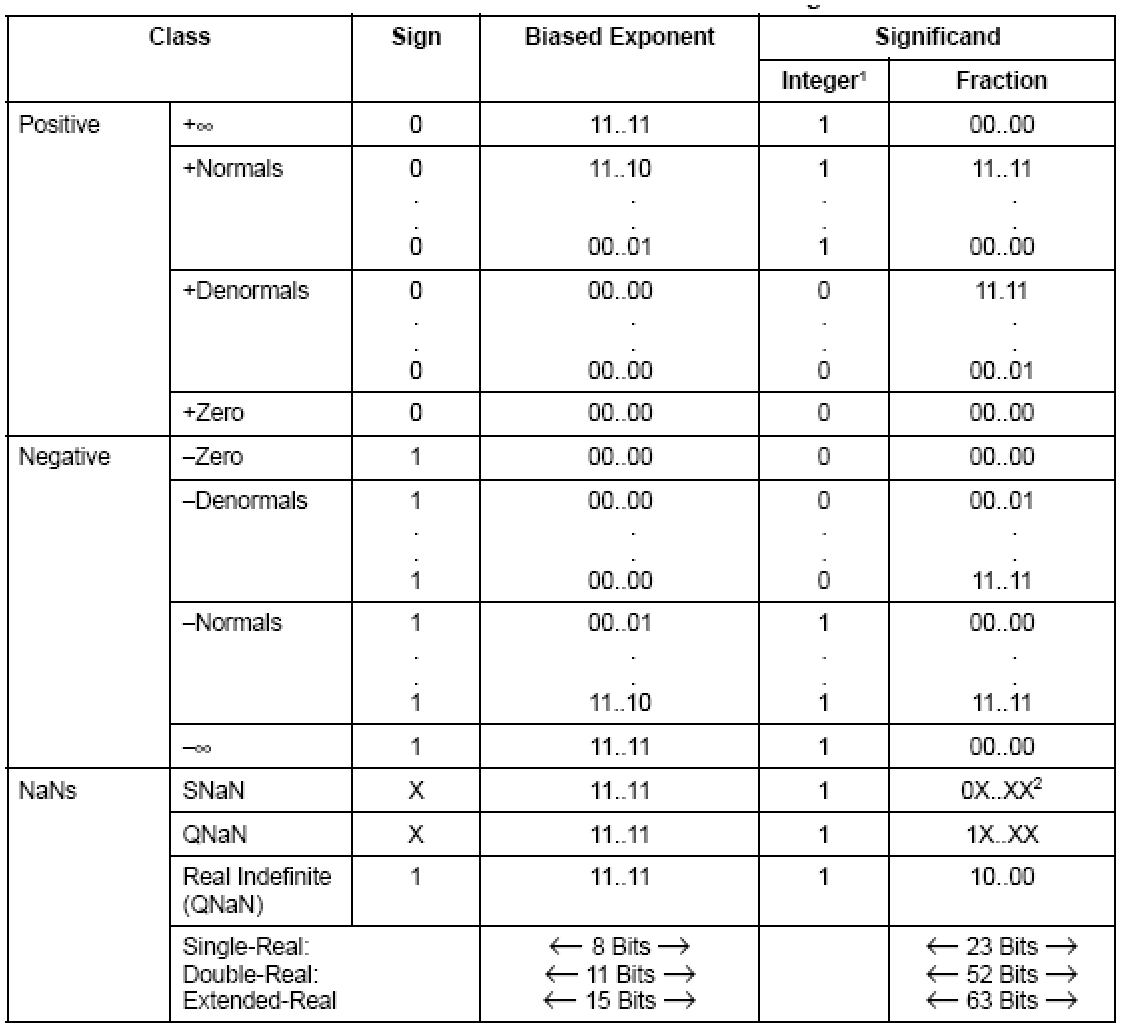
FXAM

FXAM (Examine)

D9 E5 FXAM

```
Sign bit of ST(0) → C1
case (class of value or number in ST(0)) of
  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
end
```

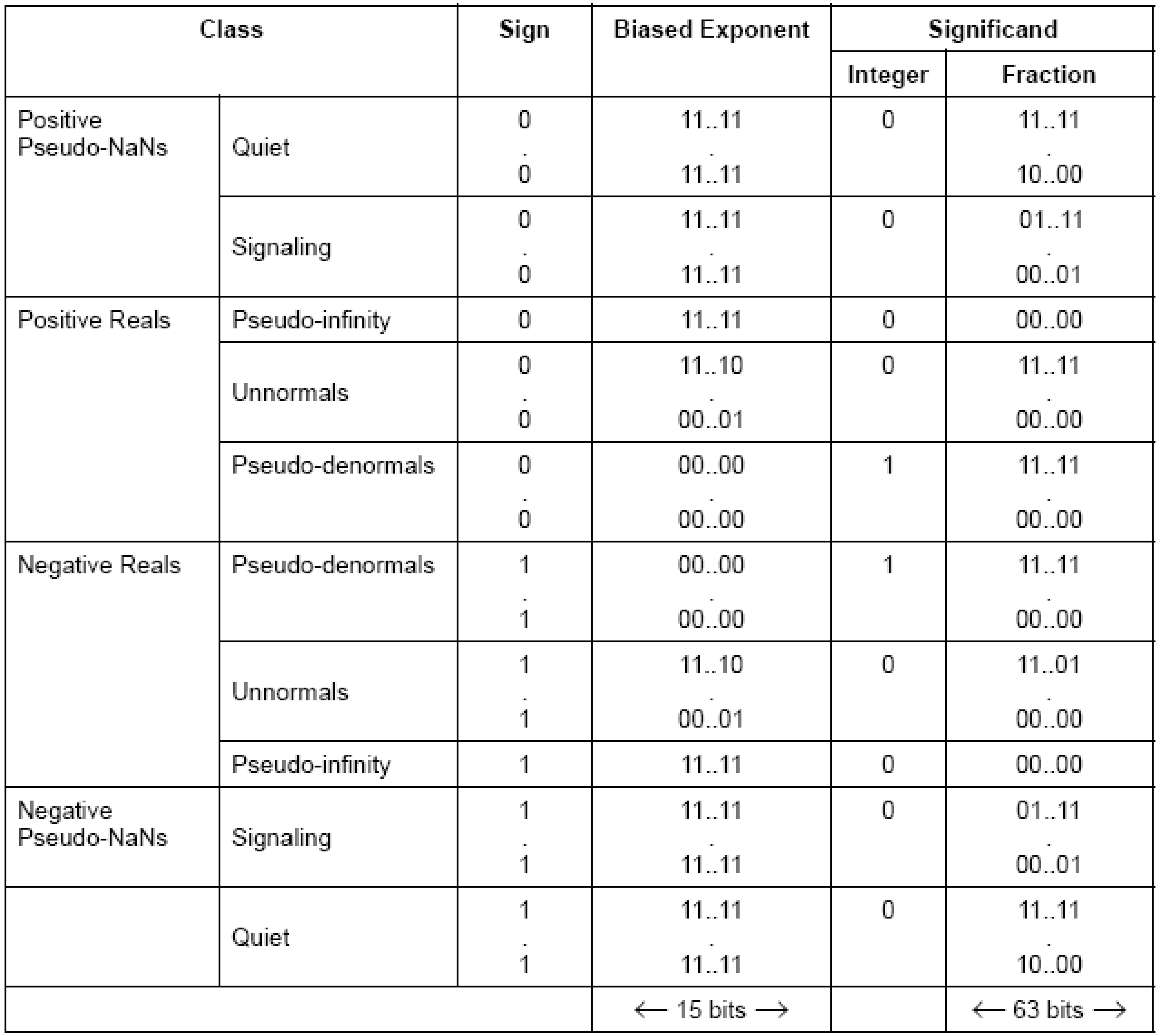
Real Number and NaN Encodings



Class		Sign	Biased Exponent	Significand	
				Integer[1]	Fraction
Positive	+∞	0	11..11	1	00..00
	+Normals	0	11..10	1	11..11
		.	.	.	.
		.	.	.	.
		0	00..01	1	00..00
	+Denormals	0	00..00	0	11.11
		.	.	.	.
		.	.	.	.
		0	00..00	0	00..01
	+Zero	0	00..00	0	00..00
Negative	−Zero	1	00..00	0	00..00
	−Denormals	1	00..00	0	00..01
		.	.	.	.
		.	.	.	.
		1	00..00	0	11..11
	−Normals	1	00..01	1	00..00
		.	.	.	.
		.	.	.	.
		1	11..10	1	11..11
	−∞	1	11..11	1	00..00
NaNs	SNaN	X	11..11	1	0X..XX[2]
	QNaN	X	11..11	1	1X..XX
	Real Indefinite (QNaN)	1	11..11	1	10..00
	Single-Real:		← 8 Bits →		← 23 Bits →
	Double-Real:		← 11 Bits →		← 52 Bits →
	Extended-Real		← 15 Bits →		← 63 Bits →

1. Integer bit is implied and not stored for single-real and double-real formats.
2. The fraction for SNaN encodings must be non-zero.

Unsupported formats:



Class		Sign	Biased Exponent	Significand	
				Integer	Fraction
Positive Pseudo-NaNs	Quiet	0 . 0	11..11 . 11..11	0	11..11 . 10..00
	Signaling	0 . 0	11..11 . 11..11	0	01..11 . 00..01
Positive Reals	Pseudo-infinity	0	11..11	0	00..00
	Unnormals	0 . 0	11..10 . 00..01	0	11..11 . 00..00
	Pseudo-denormals	0 . 0	00..00 . 00..00	1	11..11 . 00..00
Negative Reals	Pseudo-denormals	1 . 1	00..00 . 00..00	1	11..11 . 00..00
	Unnormals	1 . 1	11..10 . 00..01	0	11..01 . 00..00
	Pseudo-infinity	1	11..11	0	00..00
Negative Pseudo-NaNs	Signaling	1 . 1	11..11 . 11..11	0	01..11 . 00..01
	Quiet	1 . 1	11..11 . 11..11	0	11..11 . 10..00
			← 15 bits →		← 63 bits →

Transcendentní instrukce

- FSIN
- FCOS
- FSINCOS
- FPTAN
- FPATAN
- F2XM1
- FYL2X
- FYL2XP1

FSIN, FCOS, FSICOS, FPTAN, FPATAN

FSIN (Sine)

D9 FE FSIN

```
if $|ST(0)| < 2^{63}$ then begin
  $0 \rightarrow C2$
  $\sin(ST(0)) \rightarrow ST(0)$
end
else $1 \rightarrow C2$              {source operand is out of range}

if stack underflow occurred then $0 \rightarrow C1$
                    else C1 indicates rounding direction
C3, C0 are not defined
```

FCOS (Cosine)

D9 FF FCOS

```
if $|ST(0)| < 2^{63}$ then begin
  $0 \rightarrow C2$
  $\cos(ST(0)) \rightarrow ST(0)$
end
else $1 \rightarrow C2$            {source operand is out of range}

if stack underflow occurred then $0 \rightarrow C1$
                   else C1 indicates rounding direction
C3, C0 are not defined
```

FSINCOS (Sine and Cosine)

D9 FB FSINCOS

```
if $|ST(0)| < 2^{63}$ then begin
  $0 \rightarrow C2$
  $\cos(ST(0)) \rightarrow temp$
  $\sin(ST(0)) \rightarrow ST(0)$
  $top - 1 \rightarrow top$
  $temp \rightarrow ST(0)$
end
else $1 \rightarrow C2$        {source operand is out of range}
if stack underflow occurred then $0 \rightarrow C1$
if stack overflow occurred then $1 \rightarrow C1$
if stack is OK then C1 indicates rounding direction
C3, C0 are not defined
```

FPTAN (Partial Tangent)

D9 F2 FPTAN

```
if $|ST(0)| < 2^{63}$ then begin
  $0 \rightarrow C2$
  $\tan(ST(0)) \rightarrow ST(0)$
  $top - 1 \rightarrow top$
  $1.0 \rightarrow ST(0)$
end
else $1 \rightarrow C2$          {source operand is out of range}

if stack underflow occurred then $0 \rightarrow C1$
if stack overflow occurred then $1 \rightarrow C1$
if stack is OK then C1 indicates rounding direction
C3, C0 are not defined
```

FPATAN (Partial Arctangent)

D9 F3 FPATAN

```
arctan(ST(1) / ST(0)) → ST(1)
PopRegisterStack

if stack underflow occurred then 0 → C1
                else C1 indicates rounding direction
C3, C0 are not defined
```

Pozn.: Hodnoty operandů pro dřívější koprocesory (do 80287) byly omezeny podmínkou

$$0 \leq |ST(1)| < |ST(0)| < +\infty$$

F2XM1, FYL2X, FYL2XP1

F2XM1 (Compute $2^x - 1$)

D9 F0 F2XM1

{musí být splněna výchozí podmínka: $-1 \leq ST(0) \leq 1$}

$(2^{ST(0)} - 1) \rightarrow ST(0)$

if stack underflow occurred then $0 \rightarrow$ C1
else C1 indicates rounding direction
C3, C0 are not defined

FYL2X (Compute $y * \log_2 x$)

D9 F0 FYL2X

{musí být splněna výchozí podmínka: ST(0) > 0}

$ST(1) * \log_2 ST(0) \rightarrow ST(1)$
PopRegisterStack

if stack underflow occurred then $0 \rightarrow C1$
else C1 indicates rounding direction
C3, C0 are not defined

FYL2XP1 (Compute y $*$ $\log_2(x+1)$)

D9 F9 FYL2XP1

{musí být splněna výchozí podmínka: $-\left(1-\frac{\sqrt{2}}{2}\right) \leq ST(0) \leq \left(1-\frac{\sqrt{2}}{2}\right)$}

$ST(1) * \log_2(ST(0) + 1) \rightarrow ST(1)$
PopRegisterStack

if stack underflow occurred then $0 \rightarrow C1$
else C1 indicates rounding direction
C3, C0 are not defined

Řídící instrukce

- **FINIT**/FNINIT
- FCLEX/FNCLEX
- FSTCW/FNSTCW
- FLDCW
- **FSTSW**/FNSTSW
- FSTENV/FNSTENV
- FLDENV
- FSAVE/FNSAVE
- FRSTOR
- FINCSTP
- FDECSTP
- FFREE
- FWAIT/WAIT
- FNOP

FINIT, FNINIT

FINIT/FNINIT (Initialize FPU)

9B DB E3	FINIT
DB E3	FNINIT

037FH $\rightarrow$ FPU ControlWord
0 $\rightarrow$ FPU StatusWord
FFFFH $\rightarrow$ FPU TagWord
0 $\rightarrow$ FPU DataPointer
0 $\rightarrow$ FPU InstructionPointer
0 $\rightarrow$ FPU LastInstructionOpcode
0 $\rightarrow$ C3, C2, C1, C0

Instrukce FINIT čeká na zpracování všech čekajících FPU výjimek.

037FH → FPU ControlWord

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
			IC	RC		PC				PM	UM	OM	ZM	DM	IM
0	0	0	0	0	0	1	1	0	1	1	1	1	1	1	1

RC	= 0	zaokrouhlování k nejbližšímu číslu
PC	= 3	64 bitová přesnost mantisy

FCLEX, FNCLEX

FCLEX/FNCLEX (Clear Exceptions)

9B DB E2	FCLEX
DB E2	FNCLEX

$0 \rightarrow$ FPUStatusWord[7..0]
$0 \rightarrow$ FPUStatusWord[15]

C3, C2, C1, C0 are not defined

Instrukce FCLEX čeká na zpracování všech čekajících FPU výjimek.

FSTCW, FNSTCW

FSTCW/FNSTCW dest (Store Control Word)

9B D9 /7	FSTCW m16
D9 /7	FNSTCW m16

FPUControlWord $\rightarrow$ dest

C3, C2, C1, C0 are not defined

Instrukce FSTCW čeká na zpracování všech čekajících FPU výjimek.

FLDCW

FLDCW src (Load Control Word)

D9 /5 FLDCW m16

src $\rightarrow$ FPUControlWord

C3, C2, C1, C0 are not defined

FSTSW, FNSTSW

FSTSW/FNSTSW dest (Store Status Word)

9B DD /7	FSTSW m16
9B DF E0	FSTSW AX
DD /7	FNSTSW m16
DF E0	FNSTSW AX

FPUStatusWord $\rightarrow$ dest

C3, C2, C1, C0 are not defined

Instrukce FSTSW čeká na zpracování všech čekajících FPU výjimek.

FSTENV, FNSTENV

FSTENV/FNSTENV dest (Store FPU Environment)

9B D9 /6	FSTENV m14/28byte
D9 /6	FNSTENV m14/28byte

FPU_Environment → dest

C3, C2, C1, C0 are not defined

Instrukce FSTENV čeká na zpracování všech čekajících FPU výjimek.

FPU environment:

31	16	15		0		
0 0 0 0	OPER. POINTER 31…..16		0	0 0 0 0 0 0 0 0 0 0 0	+12 … +13	+24 … +27
RESERVED		OPERAND POINTER 15..0			+10 … +11	+20 … +23
0 0 0 0	INSTR. POINTER 31…..16		0	OPCODE 10..0	+8 … +9	+16 … +19
RESERVED		INSTRUCTION POINTER 15..0			+6 … +7	+12 … +15
RESERVED		TAG WORD			+4 … +5	+8 … +11
RESERVED		STATUS WORD			+2 … +3	+4 … +7
RESERVED		CONTROL WORD			0 … +1	0 … +3

FLDENV

FLDENV src (Load FPU Environment)

D9 /4 FLDENV m14/28byte

src → FPU_Environment

31	16	15	0		
0 0 0 0	OPER. POINTER 31…..16	0	0 0 0 0 0 0 0 0 0 0 0	+12 … +13	+24 … +27
RESERVED		OPERAND POINTER 15..0		+10 … +11	+20 … +23
0 0 0 0	INSTR. POINTER 31…..16	0	OPCODE 10..0	+8 … +9	+16 … +19
RESERVED		INSTRUCTION POINTER 15..0		+6 … +7	+12 … +15
RESERVED		TAG WORD		+4 … +5	+8 … +11
RESERVED		STATUS WORD		+2 … +3	+4 … +7
RESERVED		CONTROL WORD		0 … +1	0 … +3

FSAVE, FNSAVE

FSAVE/FNSAVE dest (Store FPU State)

9B DD /6	FSAVE m94/108byte
DD /6	FNSAVE m94/108byte

FPU_State $\rightarrow$ dest
Initialize FPU {see FINIT}

Instrukce FSAVE čeká na zpracování všech čekajících FPU výjimek.

FPU State:

15 0		
ST(7)	+84 … +93	+98 … +107
ST(6)	+74 … +83	+88 … +97
ST(5)	+64 … +73	+78 … +87
ST(4)	+54 … +63	+68 … +77
ST(3)	+44 … +53	+58 … +67
ST(2)	+34 … +43	+48 … +57
ST(1)	+24 … +33	+38 … +47
ST(0)	+14 … +23	+28 … +37
Environment	0 … +13	0 … +27

FRSTOR

FRSTOR src (Restore FPU State)

DD /4 FRSTOR m94/108byte

src $\rightarrow$ FPU_State

FINCSTP, FDECSTP, FFREE

FINCSTP (Increment Stack-Top Pointer)

D9 F7 FINCSTP

if top = 7 then 0 $\rightarrow$ top else top + 1 $\rightarrow$ top

0 $\rightarrow$ C1,
C3, C2, C0 are not defined

FDECSTP (Decrement Stack-Top Pointer)

D9 F6 FDECSTP

if top = 0 then 7 $\rightarrow$ top else top – 1 $\rightarrow$ top

0 $\rightarrow$ C1,
C3, C2, C0 are not defined

FFREE (Free Floating Point Register)

DD CO +i FFREE ST(i)

$11 \rightarrow TAG(i)$

C3, C2, C1, C0 are not defined

FWAIT/WAIT

FWAIT/WAIT (Wait for FPU)

B9 FWAIT/WAIT

C3, C2, C1, C0 are not defined

Slouží k synchronizaci práce procesoru a koprocesoru (FPU).
Asembler ji vkládá automaticky na potřebná místa.

FNOP

FNOP (No Operation)

D9 DO FNOP

C3, C2, C1, C0 are not defined