

FPU instrukce

Načítání, ukládání a manipulace se zásobníkem FPU

FLD mem/ST(i)	- uloží číslo z paměti nebo ST(i) na vrchol zásobníku (pushne stack)
FST (P) mem/ST(i)	- uloží vrchol zásobníku do paměti nebo ST(i), FSTP provede i pop
FXCH ST(i)	- prohodí ST(0) a ST(i)
FCMOVcc ST(0), ST(i)	- pokud platí podmínka cc (dle EFLAGS), uloží ST(0) ← ST(i)
FILD mem	- načte z paměti int, zkonvertuje ho na double-extended a pushne na stack
FIST (P) mem	- uloží vrchol zásobníku zkonvertovaný do intu, FISTP navíc popne zásobník
FBLD mem	- pushne na stack 80bit BCD blok a převede ho na double-extended
FBSTP mem	- uloží ST(0) do paměti jako 18 ciferný BCD blok (80bit) a popne zásobník
FLDZ	- pushne na zásobník 0.0
FLD1	- pushne na zásobník 1.0
FLDPI	- pushne na zásobník π
FLDL2T	- pushne na zásobník $\log_2 10$
FLDL2E	- pushne na zásobník $\log_2 e$
FLDLG2	- pushne na zásobník $\log_{10} 2$
FLDLN2	- pushne na zásobník $\ln 2$

Základní operace

FADD mem/ST(0), ST(i)	- přičte hodnotu v paměti nebo v ST(i) k ST(0)
FADD (P) ST(i), ST(0)	- přičte ST(0) do ST(i), FADDP navíc popne stack
FIADD mem	- načte int z paměti, převede ho na double-extended a přičte k ST(0)
FSUB /FSUBP/FISUB	- analogie FADD , FADDP a FIADD pro odčítání
FSUBR /FSUBRP/FISUBR	- analogie FSUB , FSUBP a FISUB , ale čísla odečítá v opačném pořadí
FMUL /FMULP/FIMUL	- analogie FADD , FADDP a FIADD pro násobení
FDIV /FDIVP/FIDIV	- analogie FADD , FADDP a FIADD pro dělení
FDIVR /FDIVRP/FIDIVR	- analogie FDIV , FDIVP a FIDIV , ale prohodí dělence a dělitele
FABS	- ze ST(0) udělá absolutní hodnotu
FCHS	- změní znaménko ST(0)
FSQRT	- spočítá druhou odmocninu z ST(0)
FPREM (1)	- do ST(0) spočítá ST(0) modulo ST(1), PREM1 spočítá modulo dle IEEE
FRNDINT	- zaokrouhlí ST(0) na integer
FXTRACT	- rozloží ST(0) – exp. se uloží do ST(0) a následně se pushne mantisa na stack
FSIN	- spočítá sinus ze ST(0) (a uloží jej místo ST(0))
FCOS	- spočítá cosinus ze ST(0) (a uloží jej místo ST(0))
FSINCOS	- spočítá sinus a cosinus ze ST(0), sin uloží místo ST(0) a cos pushne na stack
FPTAN	- spočítá tangents na ST(0) a následně pushne 1.0 na stack
FPATAN	- do ST(1) spočítá arctan(ST(1)/ST(0)) a popne stack
FYL2X	- do ST(1) spočítá $ST(1) * \log_2 ST(0)$ a popne stack
FYL2XP1	- do ST(1) spočítá $ST(1) * \log_2 (ST(0) + 1.0)$ a popne stack
F2XM1	- spočítá $2^{ST(0)}$ (a výsledek uloží do ST(0))
FSCALE	- vezme ST(1), zaokrouhlí ho na int (směrem k 0) a přičte k exponentu ST(0)

Porovnávání

F(U)COMI (P) ST, ST(i)	- porovná ST(0) a ST(i) a nastaví EFLAGS, F(U)COMIP navíc popne stack
FXAM	- otestuje vlastnosti ST(0) (NaN, inf...) a nastaví C0-3 ve FPU status word

Řídící operace

F(N)INIT	- inicializace FPU, FNINIT nekontroluje visící exceptions
FLDCW mem	- načte z paměti 16bit řídicí slovo
F(N)STCW mem	- uloží do paměti 16bit řídicí slovo, FNSTCW nekontroluje exceptions
F(N)CLEX	- vymaže flagy exceptions, FCLEX předtím zkontroluje jestli žádná ex. neprobíhá
FINCSTP	- zvýší vrchol zásobníku (prázdný push)
FDECSTP	- sníží vrchol zásobníku (prázdný pop)
FRSTOR mem	- obnoví stav FPU z paměti
F(N)SAVE mem	- uloží stav FPU do paměti, FNSAVE nekontroluje probíhající exceptions
FFREE ST(i)	- označí ST(i) jako prázdný (ale nemodifikuje jeho obsah ani stack top)

MMX

Načítání a přesuny

<code>MOVD dest,src</code>	- přesun 32bit intů mezi běžnými registry nebo pamětí a MMX registry
<code>MOVQ dest,src</code>	- přesun 64bit intů mezi běžnými registry nebo pamětí a MMX registry
<code>PUNPCKHBW dest,src</code>	- proloží byty z src a dest z horních polovin mezi sebe (src jde dřív)
<code>PUNPCKHWD dest,src</code>	- proloží wordy z src a dest z horních polovin mezi sebe
<code>PUNPCKHDQ dest,src</code>	- proloží doublewordy z src a dest z horních polovin mezi sebe
<code>PUNPCKLBW dest,src</code>	- proloží byty z src a dest z dolních polovin mezi sebe (src jde dřív)
<code>PUNPCKLWD dest,src</code>	- proloží wordy z src a dest z dolních polovin mezi sebe
<code>PUNPCKLDQ dest,src</code>	- proloží doublewordy z src a dest z dolních polovin mezi sebe

Aritmetické a logické operace

<code>PADD (B W D Q) d,s</code>	- hromadné sčítání mezi dvěma MMX registry (nebo pamětí) s granularitou po Bytech , Wordech , Double-wordech nebo Quad-wordech (Qword asi jen s SSE)
<code>PADDQ (B W) d,s</code>	- saturované sčítání (ošetřuje přetečení) znamenkových bytů nebo wordů
<code>PADDUS (B W) d,s</code>	- saturované sčítání (ošetřuje přetečení) neznamenkových bytů nebo wordů
<code>PSUB (B W D Q) d,s</code>	- hromadné odčítání (obdoba PADDx)
<code>PSUBS (B W)</code>	- saturované odčítání znaménkové
<code>PSUBUS (B W)</code>	- saturované odčítání neznaménkové
<code>PMUL (L H)W d,s</code>	- přenásobí dva registry po 16bit slovech a uloží horní (H) nebo dolní (L) výsledek
<code>PMADDWD dest,src</code>	- přenásobí operandy po 16bit slovech a sousední dva horní/dolní sečte a vytvoří dvě 32bit slova, která uloží do dest.
<code>PAND dest,src</code>	- provede AND dvou MMX registrů
<code>PANDN dest,src</code>	- provede negaci dest a následně přiANDuje src
<code>POR dest,src</code>	- provede OR dvou MMX registrů
<code>PXOR dest,src</code>	- provede XOR dvou MMX registrů
<code>PSLL (W D Q) r,r/imm</code>	- provede SHL na každém wordu (doublu, quadu) (každý je shiftnutý o stejný kus)
<code>PSRL (W D Q) r,r/imm</code>	- analogicky k PSLLx , ale shiftuje se doprava
<code>PSRA (W D) r,r/imm</code>	- jako PSRLx , ale při shiftování zachovává aritmetické znaménko
<code>PCMPEQ (B W D) d,s</code>	- porovná po složkách oba operandy, pokud se složka rovná, uloží na příslušné místo v dest samé 1, jinak tam uloží 0
<code>PCMPGT (B W D) d,s</code>	- obdoba PCMPEQx , ale testuje, zda je dest > src
<code>PACKSSWB dest,src</code>	- „zpakuje“ wordy z src a dest do bytů v dest, přetečení řeší saturation
<code>PACKSSDW dest,src</code>	- „zpakuje“ doublewordy z src a dest do wordů v dest, přetečení řeší saturation
<code>PACKUSWB dest,src</code>	- „zpakuje“ znaménkové wordy z src a dest do bytů v dest, přetečení řeší saturation

Ostatní – porovnávání, pakování

<code>PCMPEQ (B W D) d,s</code>	- porovná po složkách oba operandy, pokud se složka rovná, uloží na příslušné místo v dest samé 1, jinak tam uloží 0
<code>PCMPGT (B W D) d,s</code>	- obdoba PCMPEQx , ale testuje, zda je dest > src
<code>PACKSSWB dest,src</code>	- „zpakuje“ wordy z src a dest do bytů v dest, přetečení řeší saturation
<code>PACKSSDW dest,src</code>	- „zpakuje“ doublewordy z src a dest do wordů v dest, přetečení řeší saturation
<code>PACKUSWB dest,src</code>	- „zpakuje“ znaménkové wordy z src a dest do bytů v dest, přetečení řeší saturation

SSE

Načítání, přesouvání, míchání...

<code>MOV (A U) PS dest,src</code>	- přesune 4 floaty z paměti do registru, z registru do paměti nebo mezi registry
<code>MOVSS dest,src</code>	- načte/uloží jeden float z/do paměti nebo registru; horní zbytek registru vyplní 0
<code>MOVLPS dest,src</code>	- načte/uloží dva floaty z/do dolní poloviny registru (druhá polovina se nemění)
<code>MOVHPS dest,src</code>	- načte/uloží dva floaty z/do horní poloviny registru (druhá polovina se nemění)
<code>MOVLHPS xmm1,xmm2</code>	- přesune dolní dva floaty z xmm2 do horních dvou floatů v xmm1
<code>MOVHLPS xmm1,xmm2</code>	- přesune horní dva floaty z xmm2 do dolních dvou floatů v xmm1
<code>MOVMSKPS reg,xmm</code>	- přečte znaménka všech čtyř floatů v xmm a uloží je do spodních 4 bitů registru
<code>SHUFPS x1,x2/m,imm</code>	- do x1 uloží do horní části dva floaty z x2 a do spodní části dva z x1 (dle imm)
<code>UNPCKHPS x1,x2/mem</code>	- do x1 proloží horní dva floaty z x1 a x2 v pořadí x2.3 x1.3 x2.2 x1.2
<code>UNPCKLPS x1,x2/mem</code>	- do x1 proloží dolní dva floaty z x1 a x2 v pořadí x2.1 x1.1 x2.0 x1.0

Aritmetické a logické operace s floaty

ADDPS <i>xmm1, xmm2/mem</i>	- sčítání čtyř floatů naráz (<i>xmm2</i> nebo paměť se přičte k <i>xmm1</i>)
SUBPS <i>xmm1, xmm2/mem</i>	- odčítání čtyř floatů naráz (<i>xmm2</i> nebo paměť se odečte od <i>xmm1</i>)
MULPS <i>xmm1, xmm2/mem</i>	- násobení čtyř floatů naráz (<i>xmm1</i> se přenásobí <i>xmm2</i> nebo paměť)
DIVPS <i>xmm1, xmm2/mem</i>	- dělení čtyř floatů naráz (<i>xmm1</i> se podělí <i>xmm2</i> nebo paměť)
RCPPS <i>xmm1, xmm2/mem</i>	- výpočet přibliž. převrácené hodnoty čtyř floatů ($xmm1 \leftarrow 1.0/xmm2$ nebo paměti)
SQRTPS <i>xmm1, xmm2/m</i>	- výpočet druhé odmocniny čtyř floatů ($xmm1 \leftarrow \sqrt{xmm2}$ nebo paměti)
RSQRTPS <i>xmm, xmm/m</i>	- kombinace SQRTPS a RCPPS (nejprve odmocnina a z ní převrácená hodnota)
MAXPS <i>xmm1, xmm2/mem</i>	- porovnání čtyř floatů, do <i>xmm1</i> se uloží max. z obou operandů (po floatech)
MINPS <i>xmm1, xmm2/mem</i>	- analogie MAXPS
ADDSS <i>xmm1, xmm2/mem</i>	- skalární sčítání na nejnižším floatu
SUBSS <i>xmm1, xmm2/mem</i>	- skalární odčítání na nejnižším floatu
MULSS <i>xmm1, xmm2/mem</i>	- skalární násobení nejnižších floatů
DIVSS <i>xmm1, xmm2/mem</i>	- skalární dělení na nejnižším floatu
RCPSS <i>xmm1, xmm2/mem</i>	- přibližný výpočet převrácené hodnoty na nejnižším floatu
SQRTSS <i>xmm1, xmm2/m</i>	- výpočet druhé odmocniny na nejnižším floatu
RSQRTSS <i>xmm, xmm/m</i>	- výpočet převrácené hodnoty z druhé odmocniny nejnižšího floatu
MAXSS <i>xmm1, xmm2/mem</i>	- porovná nejnižší floaty obou argumentů a větší z nich uloží do <i>xmm1</i>
MINSS <i>xmm1, xmm2/mem</i>	- analogie MAXSS
ANDPS <i>xmm1, xmm2/mem</i>	- provede logický AND na čtyřech floatech (výsledek je uložen do <i>xmm1</i>)
ANDNPS <i>xmm1, xmm2/m</i>	- bitově zneguje první <i>xmm1</i> a provede logický AND na čtyřech floatech
ORPS <i>xmm1, xmm2/mem</i>	- provede logický OR na čtyřech floatech (výsledek je uložen do <i>xmm1</i>)
XORPS <i>xmm1, xmm2/mem</i>	- provede logický XOR na čtyřech floatech (výsledek je uložen do <i>xmm1</i>)

Porovnávání a konverze

CMPPS <i>xmm, x/m, imm</i>	- provede porovnání na 4 floatech (jaké je dáno <i>imm</i>), do <i>dest</i> uloží 0 pokud je false, a samé 1 pokud je true (každý float je porovnán zvlášť)
CMPSS <i>xmm, x/m, imm</i>	- funguje jako CMPPS , ale porovnává jen nejnižší float
COMISS <i>xmm1, xmm2/m</i>	- porovná nejnižší floaty obou operandů a nastaví podle toho EFLAGS
UCOMISS <i>xmm1, xmm2/m</i>	- jako COMISS , ale poradí si s neporovnatelnými čísly (NaN...)
CVTPI2PS <i>xmm, mmx/m</i>	- zkonvertuje 2 doublewordy z <i>mmx</i> nebo paměti do dolních dvou floatů v <i>xmm</i>
CVTPS2PI <i>mmx/m, xmm</i>	- zkonvertuje 2 dolní floaty z <i>xmm</i> do dvou doublewordů v <i>mmx</i> nebo paměti
CVTTPS2PI <i>mmx/m, xmm</i>	- jako CVTPS2PI , ale zaokrouhluje směrem k 0
CVTSS2SS <i>xmm, r/m</i>	- zkonvertuje doubleword v registru nebo paměti do skalárního floatu v <i>xmm</i>
CVTSS2SI <i>r/m, xmm</i>	- zkonvertuje skalární float z <i>xmm</i> do doublewordu v registru nebo paměti

Celočíselné instrukce (rozšíření MMX)

PEXTRW <i>r, xmm, imm</i>	- extrahuje word z <i>xmm</i> a uloží jej do <i>r</i> , index wordu je definován v <i>imm</i>
PINSRW <i>xmm, r, imm</i>	- vloží word z registru na pozici <i>imm</i> v <i>xmm</i> registru
PMOVBK <i>r, xmm</i>	- vytáhne nejvyšší bit od každého bytu v <i>xmm</i> a vytvoří z něj bit. masku v <i>r</i>
PSHUFW <i>mmx, mmx/m, im</i>	- prohází wordy v 64bit MMX registru nebo paměti dle bitů v <i>imm</i>
PMULHUW <i>d, s</i>	- přenásobí dva registry neznaménkově po 16bit slovech a uloží horní výsledek
PAVG(B W) <i>dest, src</i>	- do <i>dest</i> uloží průměr (po bytech/wordech) spočítaný bezznaménkově
P(MAX MIN)UB <i>d, s</i>	- do <i>dest</i> uloží maximum/minimum po bytech (neznaménkově)
P(MAX MIN)SW <i>d, s</i>	- do <i>dest</i> uloží maximum/minimum po wordech (znaménkově)
PSADBW	- do <i>dest</i> uloží jediný word se součtem absolutních rozdílů <i>src</i> a <i>dest</i> po bytech

Ostatní

LDMXCSR <i>mem</i>	- načte MXCSR registr z paměti
STMXCSR	- uloží MXCSR registr do paměti
MOVNTQ <i>mem, mmx</i>	- přesune qword z <i>mmx</i> registru do paměti bez zanášení cache
MOVNTPS <i>mem, xmm</i>	- přesune 4 floaty z <i>xmm</i> registru do paměti bez zanášení cache
MASKMOVQ <i>mmx, mmx</i>	- přesune qword z <i>mmx</i> registru (používá masku druhého <i>mmx</i>) do [ES:EDI]
PREFETCH(n NTA) <i>mem</i>	- přesune data blíže procesoru (s hintem T0-T2, nebo NTA)
SFENCE	- vše co bylo zapsáno v dřívějších instrukcích se dozapíše do paměti

SSE2

Přesuny

- `MOV(A|U)PD dest,src` - přesune dva doubley mezi registrem a pamětí (nebo dvěma registry)
- `MOVSD dest,src` - přesune spodní double mezi registrem a pamětí (nebo dvěma registry)
- `MOVHPD dest,src` - načte/uloží horní double v xmm registru z/do paměti (spodní část zachová)
- `MOVLPD dest,src` - načte/uloží spodní double v xmm registru z/do paměti (horní část zachová)
- `SHUFPD xmm1,xmm2,imm` - do xmm1 složí jednu polovinu z xmm2 a jednu z xmm1 dle spodních 2 bitů imm
- `UNPCKHPD xmm1,xmm2` - do xmm1 vloží horní polovinu z xmm2 a za ní horní polovinu z xmm1
- `UNPCKLPD xmm1,xmm2` - do xmm1 vloží dolní polovinu z xmm2 a za ní ponechá polovinu z xmm1
- `MOVMSKPD r,xmm` - vytáhne znaménka obou doublů a uloží je do registru jako spodní dva bity

Float operace

- `MOVDQ(A|U) dest,src` - předouvá celočíselný quadword mezi pamětí a registrem (nebo mezi registry)
- `ADDPD, SUBPD, MULPD,`
`DIVPD, SQRTPD, MAXPD, MINPD` - operace analogické k `xxxPS`, ale pracují se dvěma doubley místo 4 floatů
- `ADDSD, SUBSD, MULSD,`
`DIVSD, SQRTSD, MAXSD, MINS`
`AND(N)PD, ORPD, XORPD` - operace analogické k `xxxSS`, ale pracují s nižším doublem (ne floatem)
- `AND(N)PD, ORPD, XORPD` - operace analogické k `xxxPS`, ale pracují se dvěma doubley místo 4 floatů

Celočíselné operace

- `PADDQ, PSUBQ d,s` - sčítání a odčítání quadwordů
- `PMULUDQ d,s` - přenásobí spodnější dva doublewordy z d,s a uloží je do dest jako quadwordy
- `PSHUFLW x,x/m,imm` - přehází wordy v dolní polovině xmm registru podle hodnoty imm
- `PSHUFLW x,x/m,imm` - přehází wordy v horní polovině xmm registru podle hodnoty imm
- `PSHUFD x,x/m,imm` - přehází doublewordy v celých 128bitech
- `PUNPCK(L|H)QDQ d,s` - proloží do dest dolní/horní quadwordy z src a dest
- `PSSLDQ xmm,imm` - shiftne celý xmm registr doleva o imm s granularitou jednotek bytů
- `MOVQ2DQ xmm,mmx` - přesune quadword z mmx do dolní poloviny xmm
- `MOVDQ2Q mmx,xmm` - přesune spodní quadword z xmm do mmx registru
- `MASKMOVDQU x1,x2` - zapíše obsah x1 do [DS:EDI] s použitím masky po bytech (jako maska se používá nejvyšší bit každého bytu v x2)

Ostatní

- `CMP(P|S)D, (U)COMISD` - operace porovnávání analogické se SSE operacemi, které pracují floaty
- `CLFLUSH mem` - flushne cache line obsahující danou paměť
- `MOVNTDQ mem,xmm` - uloží doublequadword z xmm do paměti netemporálně
- `MOVNTPD mem,xmm` - uloží dva doubley z xmm do paměti netemporálně
- `MOVNTI mem,r` - přesune doubleword z normálního registru do paměti netemporálně
- `LFENCE` - všechny předcházející instrukce čtení jsou dokončeny
- `MFENCE` - kombinace **LFENCE** a **SFENCE**

SSE3

- `FISTTP mem` - uloží ST(0) do paměti a zaokrouhlí jej (směrem k 0) na integer
- `LDDQU xmm,mem` - načte nezarovnaný doublequadword z paměti do xmm
- `MOVSHDUP xmm1,xmm2` - přesune floaty z x2 do x1 a zkopíruje vyšší floaty do nižších v každé polovině
- `MOVSLDUP xmm1,xmm2` - přesune floaty z x2 do x1 a zkopíruje nižší floaty do vyšších v každé polovině
- `MOVDDUP xmm,xmm/mem` - zkopíruje nižší double do obou v cílovém registru
- `ADDSD(P|D) d,s` - provede asymetrickou operaci - vyšší floaty/doubley se sčítají, nižší odčítají
- `HADDP(S|D) d,s` - počítá po dvou sousední floaty (doubley) horizontálně
- `HSUBP(S|D) d,s` - analogicky horizontální odčítání

Dodatek 1 – obrázky

