SEZNAM INSTRUKCÍ JEDNOTLIVÝCH INSTRUKČNÍCH SAD

Tato příloha poskytuje seznam instrukcí přidaných v jednotlivých instrukčních sadách. Tabulky jsou v následujícím formátu:

Název instrukce	
Syntax instrukce.	Definice zdrojového a cílového operandu.
Operace provedena danou instrukcí.	Popis chování instrukce.

Instrukční sada SSE

Add Packed Single-Precision Floating-Point Values	
ADDPS c, z	Operandy: r128, r128/m128
c[0..31]=c[0..31]+z[0..31] c[32..63]=c[32..63]+z[32..63] c[64..95]=c[64..95]+z[64..95] c[96..127]=c[96..127]+z[96..127]	Instrukce provede vektorový součet čtyř FP hodnot s jednoduchou přesností ze zdrojového a cílového operandu a výsledek uloží do cílového operandu.

Add Scalar Single-Precision Floating-Point Values	
ADDSS c, z	Operandy: r128, r128/m32
c[0..31]=c[0..31]+z[0..31] c[32..127] zůstane nezměnen	Instrukce provede skalární součet dolních (32 bitů) FP hodnot s jednoduchou přesností ze zdrojového a cílového operandu a výsledek uloží do cílového operandu.

Bitwise Logical AND of Packed Single-Precision Floating-Point Values	
ANDPS c, z	Operandy: r128, r128/m128
c[0..127]=c[0..127] bitový AND z[0..127]	Instrukce provede bitový vektorový logický součin FP hodnot s jednoduchou přesností ze zdrojového a cílového operandu a výsledek uloží do cílového operandu.

Bitwise Logical AND NOT of Packed Single-Precision Floating-Point Values	
ANDNPS c, z	Operandy: r128, r128/m128
c[0..127]=c[0..127] bitový AND NOT z[0..127]	Instrukce provede negovaný bitový vektorový logický součin FP hodnot s jednoduchou přesností ze zdrojového a cílového operandu a výsledek uloží do cílového operandu.

Average Packed Word	
PAVGW c, z	Operandy: r64,r64/m64
	Instrukce provede vektorový průměr celočíselných hodnot datového typu Word ze zdrojového a cílového operandu a zaokrouhlený výsledek uloží do cílového operandu.

Average Packed Byte	
PAVGB c, z	Operandy: r64, r64/m64
	Instrukce provede vektorový průměr celočíselných hodnot datového typu Byte ze zdrojového a cílového operandu a zaokrouhlený výsledek uloží do cílového operandu.

Compare Packed Single-Precision Floating-Point Values	
CMPPS c, z, maska	Operandy: r128, r128/m128, imm8

CMP0=c[0..31] operand (imm8) z[0..31]
if (CMP0==true) c[0..31]=0xFFFFFFFF;
else c[0..31]=0;
...
CMP4=c[96..127] operand (imm8) z[96..127]
if (CMP4==true) c[96..127]=0xFFFFFFFF;
else c[96..127]=0;
	Instrukce provede vektorové porovnání hodnot, na základe hexadecimální hodnoty imm8 (větší, menší, rovná se atd), ze zdrojového a cílového operandu a výsledek uloží do cílového operandu.

Compare Scalar Single-Precision Floating-Point Values	
CMPSS c, z, maska	Operandy: r128, r128/m128, imm8

CMP0=c[0..31] operand (imm8) z[0..31]
if (CMP0==true) c[0..31]=0xFFFFFFFF;
else c[0..31]=0;
c[32..127] zůstane nezměnen
	Instrukce provede skalární porovnání dolních (32 bitů) hodnot, na základe hexadecimální hodnoty imm8 (větší, menší, rovná se atd), ze zdrojového a cílového operandu a výsledek uloží do cílového operandu.

Compare Scalar Ordered Single-Precision Floating- Point Values and Set EFLAGS	
COMISS z1, z2	Operandy: r128, r128/m32
Výsledek porovnání=neuspořádaný, ZF=1; PF=1; CF=1; Výsledek porovnání=větší, ZF=0; PF=0; CF=0; Výsledek porovnání=menší, ZF=0; PF=0; CF=1; Výsledek porovnání=rovná se, ZF=1; PF=0; CF=0; Značky OF, AF a SF se nastaví na 0.	Instrukce provede skalární porovnání dolních (32 bitů) hodnot ze zdrojových operandů. Podle výsledku porovnání (větší, menší, rovná se nebo neuspořádaný) nastaví značky registru EFLAGS.

Convert Packed Doubleword Integers to Packed Single-Precision Floating-Point Values	
CVTPI2PS c, z	Operandy: r128, r64/m32
c[0..31]=konverze na FP (z[0..31]) c[32..63]=konverze na FP (z[32..63]) c[64..127] zůstane nezměnen	Instrukce provede konverzi dvou celočíselných hodnot datového typu Doubleword se znaménkem (signed) ze zdrojového operandu na dvě FP hodnoty s jednoduchou přesností a výsledek uloží do spodné poloviny cílového operandu.

Convert Packed Single-Precision Floating-Point Values to Packed Doubleword Integers	
CVTPS2PI c, z	Operandy: r64, r128/m64
c[0..31]=konverze na Int (z[0..31]) c[32..63]=konverze na Int (z[32..63])	Instrukce provede konverzi dvou FP hodnot s jednoduchou přesností zdolní poloviny zdrojového operandu na dvě celočíselné hodnoty datového typu Doubleword se znaménkem (signed) a výsledek uloží cílového operandu.

Convert Doubleword Integer to Scalar Single- Precision Floating-Point Value	
CVTSI2SS c, z	Operandy: r128, r32/m32
c[0..31]=konverze na FP (z[0..31]) c[32..127] zůstane nezměnen	Instrukce provede konverzi celočíselný hodnoty datového typu Doubleword se znaménkem (signed) ze zdrojového operandu na FP hodnotu s jednoduchou přesností a výsledek uloží do prvních 32bitů cílového operandu.

Convert Scalar Single-Precision Floating-Point Value to Doubleword Integer	
CVTSS2SI c, z	Operandy: r32, r128/m32
c[0..31]=konverze na Int (z[0..31])	Instrukce provede konverzi FP hodnoty s jednoduchou přesností z prvních 32 bitů zdrojového operandu na celočíselnú hodnotu datového typu Doubleword se znaménkem (signed) a výsledek uloží do cílového operandu.

Divide Packed Single-Precision Floating-Point Values	
DIVPS c, z	Operandy: r128, r128/m128
c[0..31]=c[0..31] / z[0..31] c[32..63]=c[32..63] / z[32..63] c[64..95]=c[64..95] / z[64..95] c[96..127]=c[96..127] / z[96..127]	Instrukce provede vektorové delení FP hodnot s jednoduchou přesností ze zdrojového a cílového operandu a výsledek uloží do cílového operandu.

Divide Scalar Single-Precision Floating-Point Values	
DIVSS c, z	Operandy: r128, r128/m32

c[0..31]=c[0..31] / z[0..31] c[32..127] zůstane nezměnen	Instrukce provede saklární delení FP hodnot s jednoduchou přesností z dolních 32 bitů zdrojového a cílového operandu a výsledek uloží do dolních 32 bitů cílového operandu.

Store MXCSR Register State	
STMXCSR c	Operandy: m32
c[0..31]=obsah MXCSR	Instrukce uloží obsah MXCSR registru do paměte.

Move Aligned Packed Single-Precision Floating-Point Values	
MOVAPS c, z	Operandy: r128, r128/m128 nebo r128/m128, r128
c[0...127]=z[0..127]	Instrukce vektorově naplní cíloví operand FP hodnotami s jednoduchou přesností ze zdrojového operandu. Při použití paměte jako jeden z operandů, instrukce vyžaduje 16Byteové zarovnáni tohto operandu (aligned).

Move Scalar Single-Precision Floating-Point Values	
MOVSS c, z	Operandy: r128, r128/m32 nebo r128/m32, r128
Pokud je c XMM registr: c[0...31]=z[0..31] c[32..127] zůstane nezměnen Pokud je c 32bitová lokace paměti: c[0...31]=z[0..31]	Instrukce skalárne naplní cíloví operand dolní FP hodnotou s jednoduchou přesností ze zdrojového operandu.

Move High Packed Single-Precision Floating-Point Values	
MOVHPS c, z	Operandy: r128, m64 nebo m64, r128
Pokud je c XMM registr: c[64..127]=z c[0..63] zůstane nezměnen Pokud je c 64bitová lokace paměti: c=z[64..127]	Instrukce vektorově naplní cílový operand dvěma FP hodnotama s jednoduchou přesností ze zdrojového operandu. Instrukce pracuje s horní polovinou operandu (high).

Move Low Packed Single-Precision Floating-Point Values	
MOVLPS c, z	Operandy: r128, m64 nebo m64, r128
Pokud je c XMM registr: c[0..63]=z c[64..127] zůstane nezměnen Pokud je c 64bitová lokace paměti: c=z[0..63]	Instrukce vektorově naplní c dvěma FP hodnotama s jednoduchou přesností ze zdrojového operandu. Instrukce pracuje s dolní polovinou operandu (low).

Move Aligned Packed Single-Precision Floating-Point Values	
MOVUPS c, z	Operandy: r128, r128/m128 nebo r128/m128, r128
c[0...127]=z[0..127]	Instrukce vektorově naplní cíloví operand FP hodnotami s jednoduchou přesností ze zdrojového operandu. Instrukce nevyžaduje zarovnání (unaligned).

Maximum of Packed Signed Word Integers	
PMAXSW c, z	Operandy: r64, r64/m64 nebo r128, r128/m128
c[0..15]=větší z hodnot c[0..15] a z[0..15] c[16..31]=větší z hodnot c[16..31] a z[16..31] ... c[112..127]=větší: c[112..127] a z[112..127]	Instrukce provede vektorové porovnání celočíslených hodnot datového typu Word se znaménkem (signed). Po porovnání hodnot ze zdrojového a cílového operandu instrukce zapíše do cílového operandu větší z těchto hodnot.

Maximum of Packed Unsigned Byte Integers	
PMAXUB c, z	Operandy: r64, r64/m64 nebo r128, r128/m128
c[0..7]=větší z hodnot c[0..7] a z[0..7] c[8..15]=větší z hodnot c[8..15] a z[8..15] ... c[120..127]=větší: c[120..127] a z[120..127]	Instrukce provede vektorové porovnání celočíslených hodnot datového typu Byte bez znaménka (unsigned). Po porovnání hodnot ze zdrojového a cílového operandu instrukce zapíše do cílového operandu větší z těchto hodnot.

Minimum of Packed Signed Word Integers	
PMINSW c, z	Operandy: r64, r64/m64 nebo r128, r128/m128
c[0..15]=menší z hodnot c[0..15] a z[0..15] c[16..31]=menší z hodnot c[16..31] a z[16..31] ... c[112..127]=menší: c[112..127] a z[112..127]	Instrukce provede vektorové porovnání celočíslených hodnot datového typu Word se znaménkem (signed). Po porovnání hodnot ze zdrojového a cílového operandu instrukce zapíše do cílového operandu nižší z těchto hodnot.

Minimum of Packed Unsigned Byte Integers	
PMINUB c, z	Operandy: r64, r64/m64 nebo r128, r128/m128
c[0..7]=menší z hodnot c[0..7] a z[0..7] c[8..15]=menší z hodnot c[8..15] a z[8..15] ... c[120..127]=menší: c[120..127] a z[120..127]	Instrukce provede vektorové porovnání celočíslených hodnot datového typu Byte bez znaménka (unsigned). Po porovnání hodnot ze zdrojového a cílového operandu instrukce zapíše do cílového operandu nižší z těchto hodnot.

Move Packed Single-Precision Floating-Point Values High to Low	
MOVHLPS c, z	Operandy: r128, r128
c[0..63]=z[64..127] c[64..127] zůstane nezměnen	Instrukce vektorově naplní dolní polovinu cílového operandu dvěma FP hodnotama s jednoduchou přesností z horní poloviny zdrojového operandu.

Move Packed Single-Precision Floating-Point Values Low to High	
MOVLHPS c, z	Operandy: r128, r128
c[64..127]=z[0..63] c[0..63] zůstane nezměnen	Instrukce vektorově naplní horní polovinu cílového operandu dvěma FP hodnotama s jednoduchou přesností z dolní poloviny zdrojového operandu.

Extract Packed Single-Precision Floating-Point Sign Mask	
MOVMSKPS c, z	Operandy: r32, r128
c[0]=z[31] c[1]=z[63] c[2]=z[95] c[3]=z[127] c[4..31]=0	Instrukce extrahuje vektorové hodnoty ze zdrojového operandu, naformátuje je jako 4bitovou masku a tuto masku uloží do cílového operandu.

Multiply Packed Single-Precision Floating-Point Values	
MULPS c, z	Operandy: r128, r128/m128
c[0..31]=c[0..31] * z[0..31] c[32..63]=c[32..63] * z[32..63] c[64..95]=c[64..95] * z[64..95] c[96..127]=c[96..127] * z[96..127]	Instrukce provede vektorové násobení FP hodnot s jednoduchou přesností ze zdrojového a cílového operandu a výsledek uloží do cílového operandu.

Multiply Scalar Single-Precision Floating-Point Values	
MULSS c, z	Operandy: r128, r128/m32
c[0..31]=c[0..31] * z[0..31] c[32..127] zůstane nezměnen	Instrukce provede skalární násobení FP hodnoty s jednoduchou přesností z dolních 32 bitů zdrojového a cílového operandu a výsledek uloží do cílového operandu.

Multiply Packed Unsigned Integers and Store High Result	
PMULHUW c, z	Operandy: r64, r64/m64 nebo r128, r128/m128

MV=mezivýsledek: MV1[0..31]=c[0..15]*z[0..15] ... MV8[0..31]=c[112..127]*z[112..127] ---------- c[0..15]=MV1[16..31] ... c[112...127]=MV8[16..31]	Instrukce vektorově vynásobí celočíselné hodnoty datového typu Word bez znaménka (unsigned) ze zdrojového a cílového operandu. Do cílového operandu je uloženo horních 16 bitů mezivýsledku.

Bitwise Logical OR of Single-Precision Floating-Point Values	
ORPS c, z	Operandy: r128, r128/m128
c[0..127]=c[0..127] bitový OR z[0..127]	Instrukce provede bitový vektorový logický součet FP hodnot s jednoduchou přesností ze zdrojového a cílového operandu a výsledek uloží do cílového operandu.

Extract Word	
PEXTRW c, z, maska	Operandy: r16, r128, imm8
c[0..16]=výběr pomocí masky(z[0..127])	Instrukce skopíruje 16 bitů ze zdrojového operandu, zvolených pomocí masky, do dolních 16 cílového operandu. Horní hodnoty cílového operandu jsou vynulovány. Pracuje s datovým typem Word.

Insert Word	
PINSRW c, z, maska	Operandy:r64, r32/r16, imm8 nebo r128, r32/m16, imm8
c[výběr lokace pomocí masky]=z[0..15]	Instrukce skopíruje datový typ Word ze zdrojového operandu a vloží ho do lokace cílového operandu zvolené pomocí masky. Ostatní hodnoty v cílovém operandu zůstanou nezměneny.

Compute Sum of Absolute Differences	
PSADBW c, z	Operandy: r64, r64/m64 nebo r128, r128/m128
MV=mezivýsledek, AH=absolutní hodnota: MV1=AH z (c[0..7] - z[0..7]) ... MV16=AH z (c[120..127] - z[120..127]) ---------- c[0..15]=suma MV1 až MV8 c[16..63]=0 c[64..79]=suma MV9 až MV16 c[80..127]=0	Instrukce vektorově odčítá celočíselné hodnoty datového typu Byte ze zdrojového a cílového operandu a vytvoří z nich absolutní hodnotu. Následně se tyto mezivýsledky sečtou a výsledek je uložen do dolních 16 bitů každé poloviny cílového operandu. Ostatní hodnoty cílového operandu jsou vynulovány.

Shuffle Packed Words	
PSHUFW c, z, maska	Operandy: r64, r64/m64, imm8

c[0..15], [16..31], [32..47], [48..63]=na základě masky jedna hodnota z možností: z[0..15], z[16..31], z[32..47], z[48..63]	Instrukce vybere na základě masky jednu celočíselné hodnotu datového typu Word ze zdrojového operandu a tuto hodnotu uloží do všech datových typů Word v cílovém operandu.

Compute Reciprocals of Packed Single-Precision Floating-Point Values	
RCPPS c, z	Operandy: r128, r128/m128
c[0..31]≈(1,0 / z[0..31]) c[32..63]≈(1,0 / z[32..63]) c[64..95]≈(1,0 / z[64..95]) c[96..127]≈(1,0 / z[96..127])	Instrukce vektorově vypočítá přibližnou hodnotu po dělení hodnoty 1,0 hodnotami ze zdrojového operandu. Výsledek je uložen do cílového operandu. Pracuje s FP hodnotami s jednoduchou přesností.

Compute Reciprocal of Scalar Single-Precision Floating- Point Values	
RCPSS c, z	Operandy: r128, r128/m32
c[0..31]≈(1,0 / z[0..31]) c[32..127] zůstane nezměnen	Instrukce skalárne vypočítá přibližnou hodnotu po dělení hodnoty 1,0 hodnotou z dolních 32 bitů zdrojového operandu. Výsledek je uložen do dolních 32 bitů cílového operandu a ostatní hodnoty jsou ponechány. Pracuje s FP hodnotami s jednoduchou přesností.

Compute Reciprocals of Square Roots of Packed Single-Precision Floating-Point Values	
RSQRTPS c, z	Operandy: r128, r128/m128
c[0..31]≈(1,0 / √z[0..31]) c[32..63]≈(1,0 / √z[32..63]) c[64..95]≈(1,0 / √z[64..95]) c[96..127]≈(1,0 / √z[96..127])	Instrukce vektorově vypočítá přibližnou hodnotu po dělení hodnoty 1,0 hodnotami ze zdrojového operandu po jejich odmocnění. Výsledek je uložen do cílového operandu. Pracuje s FP hodnotami s jednoduchou přesností.

Compute Reciprocal of Square Root of Scalar Single-Precision Floating-Point Value	
RSQRTSS c, z	Operandy: r128, r128/m32
c[0..31]≈(1,0 / √z[0..31]) c[32..127] zůstane nezměnen	Instrukce skalárne vypočítá přibližnou hodnotu po dělení hodnoty 1,0 hodnotou z dolních 32 bitů zdrojového operandu. Výsledek je uložen do dolních 32 bitů cílového operandu a ostatní hodnoty jsou ponechány. Pracuje s FP hodnotami s jednoduchou přesností.

Load MXCSR Register	
LDMXCSR z	Oprand: m32
MSCSR=z[0..31]	Instrukce načte hodnotu z paměti a uloží ji do MXCSR registru.

Shuffle Packed Single-Precision Floating-Point Values	
SHUFPS c, z, maska	Operandy: r128, r128/m128, imm8

Příklad1: Příklad2: c[0..31]=c[96..127] | c[0..31]=c[0..31] c[32..63]=c[96..127] | c[32..63]=c[0..31] c[64..95]=z[32..63] | c[64..95]=z[0..31] c[96..127]=z[32..63] |c[96..127]=z[0..31]	Instrukce vektorově uloží dvě ze čtyř SP FP hodnot z cílového operandu do dolních 64 bitů cílového operandu a následně uloží dvě ze čtyř SP FP hodnot ze zdrojového operandu do horních 64 bitů cílového operandu. Maska definuje, které hodnoty budou vybrány.

Compute Square Roots of Packed Single-Precision Floating-Point Values	
SQRTPS c, z	Operandy: r128, r128/m128
c[0..31]=√z[0..31] c[32..63]=√z[32..63] c[64..95]=√z[64..95] c[96..127]=√z[96..127]	Instrukce provede vektorové odmocnění FP hodnot s jednoduchou přesností ve zdrojovém operandu a výsledek uloží do cílového operandu.

Compute Square Roots of Scalar Single-Precision Floating-Point Values	
SQRTSS c, z	Operandy: r128, r128/m32
c[0..31]=√z[0..31] c[32..127] zůstane nezměnen	Instrukce provede skalární odmocnění dolní FP hodnoty s jednoduchou přesností ve zdrojovém operandu a výsledek uloží do dolních 32 bitů cílového operandu.

Subtract Packed Single-Precision Floating-Point Values	
SUBPS c, z	Operandy: r128, r128/m128
c[0..31]=c[0..31]-z[0..31] c[32..63]=c[32..63]-z[32..63] c[64..95]=c[64..95]-z[64..95] c[96..127]=c[96..127]-z[96..127]	Instrukce provede vektorové odčítání čtyř FP hodnot s jednoduchou přesností ze zdrojového a cílového operandu a výsledek uloží do cílového operandu.

Subtract Scalar Single-Precision Floating-Point Values	
SUBSS c, z	Operandy: r128, r128/m32
c[0..31]=c[0..31]-z[0..31] c[32..127] zůstane nezměnen	Instrukce provede skalární odčítání dolní FP hodnoty s jednoduchou přesností ze zdrojového a cílového operandu a výsledek uloží do dolních 32 bitů cílového operandu.

Unordered Compare Scalar Single-Precision Floating- Point Values and Set EFLAGS	
UCOMISS z1, z2	Operandy: r128, r128/m32
Výsledek porovnání=neuspořádaný, ZF=1; PF=1; CF=1; Výsledek porovnání=větší, ZF=0; PF=0; CF=0; Výsledek porovnání=menší, ZF=0; PF=0; CF=1; Výsledek porovnání=rovná se, ZF=1; PF=0; CF=0; Značky OF, AF a SF se nastaví na 0.	Instrukce provede skalární porovnání dolní poloviny (64 bitů) hodnot ze zdrojových operandů. Podle výsledku porovnání (větší, menší, rovná se nebo neuspořádaný) nastaví značky registru EFLAGS.

Unpack and Interleave High Packed Single-Precision Floating-Point Values	
UNPCKHPS c, z	Operandy: r128, r128/m32
c[0..31]=c[64..95] c[32..63]=z[64..95] c[64..95]=c[96..127] c[96..127]=z[96..127]	Instrukce načítá dvě horní FP hodnoty s jednoduchou přesností z cílového a zdrojového operandu a uloží je přeloženě do cílového operandu. Viz operace.

Unpack and Interleave Low Packed Single-Precision Floating-Point Values	
UNPCKLPS c, z	Operandy: r128, r128/m32
c[0..31]=c[0..31] c[32..63]=z[0..31] c[64..95]=c[32..63] c[96..127]=z[32..63]	Instrukce načítá dvě dolní FP hodnoty s jednoduchou přesností z cílového a zdrojového operandu a uloží je přeloženě do cílového operandu. Viz operace.

Bitwise Logical XOR for Single-Precision Floating-Point Values	
XORPS c, z	Operandy: r128, r128/m32
c[0..127]=c[0..127] bitový XOR z[0..127]	Instrukce provede bitový vektorový exkluzivní logický součet FP hodnot s jednoduchou přesností ze zdrojového a cílového operandu a výsledek uloží do cílového operandu.

Instrukční sada SSE2

Add Packed Byte	
PADDB c, z	Operandy: r128, r128/m128 nebo r64, r64/m64
c[0..7]=c[0..7]+z[0..7] c[8..15]=c[8..15]+z[8..15] ... c[120..127]=c[120..127]+z[120..127]	Instrukce provede vektorový součet celočíslených hodnot datového typu Byte ze zdrojového a cílového operandu a výsledek uloží do cílového operandu.

Add Packed Word	
PADDW c, z	Operandy: r128, r128/m128 nebo r64, r64/m64
c[0..15]=c[0..15]+z[0..15] c[16..31]=c[16..31]+z[16..31] ... c[112..127]=c[112..127]+z[112..127]	Instrukce provede vektorový součet celočíslených hodnot datového typu Word ze zdrojového a cílového operandu a výsledek uloží do cílového operandu.

Add Packed Doubleword	
PADDD c, z	Operandy: r128, r128/m128 nebo r64, r64/m64
c[0..31]=c[0..31]+z[0..31] c[32..63]=c[32..63]+z[32..63] c[64..95]=c[64..95]+z[64..95] c[96..127]=c[96..127]+z[96..127]	Instrukce provede vektorový součet celočíslených hodnot datového typu Doubleword ze zdrojového a cílového operandu a výsledek uloží do cílového operandu.

Add Packed Quadword	
PADDQ c, z	Operandy: r128, r128/m128 nebo r64, r64/m64
c[0..63]=c[0..63]+z[0..63] c[64..127]=c[64..127]+z[64..127]	Instrukce provede vektorový součet celočíslených hodnot datového typu Quadword ze zdrojového a cílového operandu a výsledek uloží do cílového operandu.

Add Packed Double-Precision Floating-Point Values	
ADDPD c, z	Operandy: r128, r128/m128
c[0..63]=c[0..63]+z[0..63] c[64..127]=c[64..127]+z[64..127]	Instrukce provede vektorový součet dvou FP hodnot s dvojitou přesností ze zdrojového a cílového operandu a výsledek uloží do cílového operandu.

Add Scalar Double-Precision Floating-Point Values	
ADDSD c, z	Operandy: r128, r128/m32
c[0..63]=c[0..63]+z[0..63] c[64..127] zůstane nezměnen	Instrukce provede skalární součet dolních (64 bitů) FP hodnot s dvojitou přesnosí ze zdrojového a cílového operandu a výsledek uloží do cílového operandu.

Add Packed Signed Byte with Signed Saturation	
PADDSB c, z	Operandy: r128, r128/m128 nebo r64, r64/m64
c[0..7]=Saturace(c[0..7]+z[0..7]) c[8..15]=Saturace(c[8..15]+z[8..15]) ... c[120..127]=Satur.(c[120..127]+z[120..127])	Instrukce provede vektorový součet se saturací celočíslených hodnot datového typu Byte se znamánkem ze zdrojového a cílového operandu a výsledek uloží do cílového operandu.

Add Packed Signed Word with Signed Saturation	
PADDSW c, z	Operandy: r128, r128/m128 nebo r64, r64/m64
c[0..15]=Saturace(c[0..15]+z[0..15]) c[16..31]=Saturace(c[16..31]+z[16..31]) ... c[112..127]=Satur.(c[112..127]+z[112..127])	Instrukce provede vektorový součet se saturací celočíslených hodnot datového typu Word se znamánkem ze zdrojového a cílového operandu a výsledek uloží do cílového operandu.

Add Packed Unsigned Byte with Unsigned Saturation	
PADDUSB c, z	Operandy: r128, r128/m128 nebo r64, r64/m64
c[0..7]=Saturace(c[0..7]+z[0..7]) c[8..15]=Saturace(c[8..15]+z[8..15]) ... c[120..127]=Satur.(c[120..127]+z[120..127])	Instrukce provede vektorový součet se saturací celočíslených hodnot datového typu Byte bez znaménka ze zdrojového a cílového operandu a výsledek uloží do cílového operandu.

Add Packed Unsigned Word with Unsigned Saturation	
PADDUSW c, z	Operandy: r128, r128/m128 nebo r64, r64/m64
c[0..15]=Saturace(c[0..15]+z[0..15]) c[16..31]=Saturace(c[16..31]+z[16..31]) ... c[112..127]=Satur.(c[112..127]+z[112..127])	Instrukce provede vektorový součet se saturací celočíslených hodnot datového typu Word bez znaménka ze zdrojového a cílového operandu a výsledek uloží do cílového operandu.

Bitwise Logical AND of Packed Double-Precision Floating-Point Values	
ANDPD c, z	Operandy: r128, r128/m128
c[0..127]=c[0..127] bitový AND z[0..127]	Instrukce provede bitový vektorový logický součin FP hodnot s dvojitou přesností ze zdrojového a cílového operandu a výsledek uloží do cílového operandu.

Packed Logical AND	
PAND c, z	Operandy: r128, r128/m128 nebo r64, r64/m64
c[0..127]=c[0..127] bitový AND z[0..127]	Instrukce provede bitový vektorový logický součin FP hodnot s dvojitou přesností ze zdrojového a cílového operandu a výsledek uloží do cílového operandu.

Bitwise Logical AND NOT of Packed Double-Precision Floating-Point Values	
ANDNPD c, z	Operandy: r128, r128/m128
c[0..127]=c[0..127] bitový AND NOT z[0..127]	Instrukce provede bitový vektorový logický negovaný součin FP hodnot s dvojitou přesností ze zdrojového a cílového operandu a výsledek uloží do cílového operandu.

Packed Logical AND NOT	
PANDN c, z	Operandy: r128, r128/m128 nebo r64, r64/m64
c[0..127]=c[0..127] bitový AND NOT z[0..127] nebo c[0..64]=c[0..64] bitový AND NOT z[0..64]	Instrukce provede bitový vektorový logický negovaný součin FP hodnot s dvojitou přesností ze zdrojového a cílového operandu a výsledek uloží do cílového operandu.

Shift Double Quadword Left Logical	
PSLLDQ c, maska	Operandy: r128, imm8
c=posun vlevo na základě masky (c)	Instrukce provede logický posun cílového operandu vlevo na základě masky. Pokud je hodnota masky větší než 15, c je vynulován.

Shift Double Quadword Right Logical	
PSRLDQ c, maska	Operandy: r128, imm8
c=posun vpravo na základě masky (c)	Instrukce provede logický posun cílového operandu vpravo na základě masky. Pokud je hodnota masky větší než 15, c je vynulován.

Flush Cache Line	
CLFLUSH	
	Instrukce vyčistí řádek vyrovnávací paměti.

Compare Packed Data for Equal	
PCMPEQB c, z **PCMPEQW c, z** **PCMPEQD c, z**	Operandy: r128, r128/m128 nebo r64, r64/m64
pokud (c[x..y] == z[x..y]) c[x..y]=1 jinak c[x..y]=0	Instrukce provede porovnání celočíslených hodnot datového typu Byte/Word/Doubleword pro rovnost. Pokud se hodnoty liší, cílová hodnota je nastavena na 0. Pokus jsou hodnoty rovny, cílová hodnota je nastavena na 1.

Compare Packed Data for Greater Than	
PCMPGTB c, z **PCMPGTW c, z** **PCMPGTD c, z**	Operandy: r128, r128/m128 nebo r64, r64/m64
c[x..y]=1 pokud (c[x..y] > z[x..y]) jinak c[x..y]=0	Instrukce provede porovnání celočíslených hodnot datového typu Byte/Word/Doubleword pro rovnost. Pokud je hodnota cílového operandu větší než hodnota zdrojového operandu, pak je cílová hodnota nastavena na 1. Jinak je cílová hodnota nastavena na 0.

Compare Packed Double-Precision Floating-Point Values	
CMPPD c, z, imm8	Operandy: r128, r128/m128, imm8
c[x..y]=1 pokud (c[x..y]{==, >, < atd}z[x..y]) jinak c[x..y]=0	Instrukce provede vektorové porovnání FP hodnot s dvojitou přesností na základe hexadecimální hodnoty imm8 (větší, menší, rovná se atd), ze zdrojového a cílového operandu. Cílová hodnota je v případě splnění podmínky 1, jinak 0.

Compare Scalar Double-Precision Floating-Point Values	
CMPSD c, z, imm8	Operandy: r128, r128/m128, imm8
c[0..63]=1 pokud (c[63..0]{==, >, < atd}z[0..63]) jinak c[0..63]=0 c[64..127] zůstane nezměnen	Instrukce provede skalární porovnání dolních FP hodnot s dvojitou přesností, na základe hexadecimální hodnoty imm8 (větší, menší, rovná se atd), ze zdrojového a cílového operandu. Cílová hodnota je v případě splnění podmínky 1, jinak 0.

Compare Scalar Ordered Double-Precision Floating- Point Values and Set EFLAGS	
COMISD z1, z2	Operandy: r128, r128/m64
Výsledek porovnání=neuspořádaný, ZF=1; PF=1; CF=1; Výsledek porovnání=větší, ZF=0; PF=0; CF=0; Výsledek porovnání=menší, ZF=0; PF=0;CF=1; Výsledek porovnání=rovná se, ZF=1; PF=0; CF=0; Značky OF, AF a SF se nastaví na 0.	Instrukce provede skalární porovnání dolních (64 bitů) hodnot ze zdrojových operandů. Podle výsledku porovnání (větší, menší, rovná se nebo neuspořádaný) nastaví značky registru EFLAGS.

Convert Packed Doubleword Integers to Packed Double-Precision Floating-Point Values	
CVTDQ2PD c, z	Operandy: r128, r128/m64
c[0..63]=konverze na FP (z[0..31]) c[64..127]=konverze na FP (z[32..63])	Instrukce provede konverzi dvou celočíselných hodnot datového typu Doubleword ze zdrojového operandu na dvě FP hodnoty s dvojitou přesností a výsledek uloží do cílového operandu.

Convert Packed Doubleword Integers to Packed Double-Precision Floating-Point Values	
CVTPI2PD c, z	Operandy: r128, r64/m64
c[0..63]=konverze na FP (z[0..31]) c[64..127]=konverze na FP (z[32..63])	Instrukce provede konverzi dvou celočíselných hodnot datového typu Doubleword ze zdrojového operandu na dvě FP hodnoty s dvojitou přesností a výsledek uloží do cílového operandu.

Convert Packed Doubleword Integers to Packed Double-Precision Floating-Point Values	
CVTDQ2PS c, z	Operandy: r128, r128/m128
c[0..31]=konverze na FP (z[0..31]) c[32..63]=konverze na FP (z[32..63]) c[64..95]=konverze na FP (z[64..95]) c[96..127]=konverze na FP (z[96..127])	Instrukce provede konverzi čtyř celočíselných hodnot datového typu Doubleword ze zdrojového operandu na čtyři FP hodnoty s jednoduchou přesností a výsledek uloží do cílového operandu.

Convert Packed Double-Precision Floating-Point Values to Packed Doubleword Integers	
CVTPD2PI c, z	Operandy: r128, r128/m128

c[0..31]=konverze na Int (z[0..63]) c[32..63]=konverze na Int (z[64..127])	Instrukce provede konverzi dvou FP hodnot s dvojitou přesností ze zdrojového operandu na dvě celočíselné hodnoty datového typu Doubleword a výsledek uloží do cílového operandu.

Convert Packed Double-Precision Floating-Point Values to Packed Doubleword Integers	
CVTPD2DQ c, z	Operandy: r128, r128/m128
c[0..31]=konverze na Int (z[0..63]) c[32..63]=konverze na Int (z[64..127]) c[64..127]=0	Instrukce provede konverzi dvou FP hodnot s dvojitou přesností ze zdrojového operandu na dvě celočíselné hodnoty datového typu Doubleword a výsledek uloží do cílového operandu.

Convert Packed Double-Precision Floating-Point Values to Packed Single-Precision Floating-Point Values	
CVTPD2PS c, z	Operandy: r128, r128/m128
c[0..31]=konverze na SP FP (z[0..63]) c[32..63]=konverze na SP FP (z[64..127]) c[64..127]=0	Instrukce provede konverzi dvou FP hodnot s dvojitou přesností ze zdrojového operandu na dvě FP hodnoty s jednoduchou přesností a výsledek uloží do cílového operandu.

Convert Packed Single-Precision Floating-Point Values to Packed Doubleword Integers	
CVTPS2DQ c, z	Operandy: r128, r128/m128
c[0..31]=konverze na Int (z[0..63]) c[32..63]=konverze na Int (z[64..127]) c[64..95]=konverze na Int (z[64..95]) c[96..127]=konverze na Int (z[96..127])	Instrukce provede konverzi čtyř FP hodnot s jednoduchou přesností ze zdrojového operandu na čtyři celočíselné hodnoty datového typu Doubleword a výsledek uloží do cílového operandu.

Convert Packed Single-Precision Floating-Point Values to Packed Double-Precision Floating-Point Values	
CVTPS2PD c, z	Operandy: r128, r128/m128
c[0..63]=konverze na DP FP (z[0..31]) c[64..127]=konverze na DP FP (z[32..63])	Instrukce provede konverzi dvou FP hodnot s jednoduchou přesností ze zdrojového operandu na dvě FP hodnoty s dvojitou přesností a výsledek uloží do cílového operandu.

Convert Packed Doubleword Integers to Packed Double-Precision Floating-Point Values	
CVTSI2SD c, z	Operandy: r128, r32/m32
c[0..63]=konverze na FP (z[0..31]) c[64..127] zůstane nezměnen	Instrukce provede konverzi celočíselné hodnoty datového typu Doubleword ze zdrojového operandu na FP hodnotu s dvojitou přesností a výsledek uloží do cílového operandu.

Convert Packed Double-Precision Floating-Point Values to Packed Doubleword Integers	
CVTSD2SI c, z	Operandy: r32, r128/m64
c[0..31]=konverze na Int (z[0..63])	Instrukce provede konverzi FP hodnoty s dvojitou přesností ze zdrojového operandu na celočíselnou hodnotu datového typu Doubleword a výsledek uloží do cílového operandu.

Convert Scalar Double-Precision Floating-Point Value to Scalar Single-Precision Floating-Point Value	
CVTSD2SS c, z	Operandy: r128, r32/m64
c[0..31]=konverze na DP FP (z[0..63]) c[32..127] zůstane nezměnen	Instrukce provede konverzi FP hodnoty s dvojitou přesností ze zdrojového operandu na FP hodnotu s jednoduchou přesností a výsledek uloží do cílového operandu.

Convert Packed Single-Precision Floating-Point Values to Packed Double-Precision Floating-Point Values	
CVTSS2SD c, z	Operandy: r128, r32/m32
c[0..63]=konverze na DP FP (z[0..31]) c[64..127] zůstane nezměnen	Instrukce provede konverzi FP hodnoty s jednoduchou přesností ze zdrojového operandu na FP hodnotu s dvojitou přesností a výsledek uloží do cílového operandu.

Move Scalar Double-Precision Floating-Point Values	
MOVSD c, z	Operandy: r128, r128/m64 nebo r128/m64, r128
Pokud je c XMM registr: c[0...63]=z[0..63] c[64..127] zůstane nezměnen Pokud je c 64bitová lokace paměti: c[0...64]=z[0..64]	Instrukce skalárne naplní cíloví operand dolní FP hodnotou s dvojitou přesností ze zdrojového operandu.

Move Doubleword	
MOVD c, z	Operandy: z i c mohou být: r128, r64, r32, m32
c[0...31]=z[0..31] c[32..127]=0	Instrukce naplní cíloví operand celočíselné hodnotou datového typu Doubleword ze zdrojového operandu.

Move Quadword	
MOVQ c, z	Operandy: z i c mohou být: r128, r64, m64

c[0...63]=z[0..63] c[64..127]=0	Instrukce naplní cíloví operand celočíselné hodnotou datového typu Quadword ze zdrojového operandu.

Divide Packed Double-Precision Floating-Point Values	
DIVPD c, z	Operandy: r128, r128/m128
c[0..63]=c[0..63] / z[0..63] c[64..127]=c[64..1273] / z[64..127]	Instrukce provede vektorové delení FP hodnot s dvojitou přesností ze zdrojového a cílového operandu a výsledek uloží do cílového operandu.

Divide Packed Double-Precision Floating-Point Values	
DIVSD c, z	Operandy: r128, r128/m64
c[0..63]=c[0..63] / z[0..63] c[64..127] zůstane nezměnen	Instrukce provede saklární delení FP hodnot s dvojitou přesností z dolních 64 bitů zdrojového a cílového operandu a výsledek uloží do dolních 64 bitů cílového operandu.

Move Aligned Packed Double-Precision Floating-Point Values	
MOVAPD c, z	Operandy: r128, r128/m128 nebo r128/m128, r128
c[0...127]=z[0..127]	Instrukce vektorově naplní cíloví operand FP hodnotami s dvojitou přesností ze zdrojového operandu. Při použití paměte jako jeden z operandů, instrukce vyžaduje 16Byteové zarovnáni tohto operandu (aligned).

Move Aligned Double Quadword	
MOVDQA c, z	Operandy: r128, r128/m128 nebo r128/m128, r128
c[0...127]=z[0..127]	Instrukce naplní cíloví operand celočíselnýma hodnotama datového typu double quadwor ze zdrojového operandu. Při použití paměte jako jeden z operandů, instrukce vyžaduje 16Byteové zarovnáni tohto operandu (aligned).

Move High Packed Double-Precision Floating-Point Values	
MOVHPD c, z	Operandy: r128, r128/m64 nebo r128/m64, r128
Pokud je c XMM registr: c[64..127]=z c[0..63] zůstane nezměnen Pokud je c 64bitová lokace paměti: c=z[64..127]	Instrukce vektorově naplní c FP hodnotou s dvojitou přesností ze zdrojového operandu. Instrukce pracuje s horní polovinou operandu (high).

Move Low Packed Double-Precision Floating-Point Values	
MOVLPD c, z	Operandy: r128, r128/m64 nebo r128/m64, r128

Pokud je c XMM registr: c[0..63]=z c[64..127] zůstane nezměnen Pokud je c 64bitová lokace paměti: c=z[0..63]	Instrukce vektorově naplní c FP hodnotou s dvojitou přesností ze zdrojového operandu. Instrukce pracuje s dolní polovinou operandu (low).

Move Unaligned Packed Double-Precision Floating-Point Values	
MOVUPD c, z	Operandy: r128, r128/m128 nebo r128/m128, r128
c[0...127]=z[0..127]	Instrukce vektorově naplní cíloví operand FP hodnotami

Move Unaligned Double Quadword	
MOVDQU c, z	Operandy: r128, r128/m128 nebo r128/m128, r128
c[0...127]=z[0..127]	Instrukce naplní cíloví operand celočíselnýma hodnotama datového typu double quadword ze zdrojového operandu. Instrukce nevyžaduje zarovnání (unaligned).

Multiply and Add Packed Integers	
PMADDWD c, z	Operandy: r128, r128/m128 nebo r64, r64/m64
c[0..31]=(c[0..15]*z[0..15])+ +(c[16..31]*z[16..31]) ... c[96..127]=(c[96..111]*z[96..111])+ +(c[112..127]*z[112..127])	Instrukce provede vektorový součin dolních16 bitů zdrojového a cílového operandu. Následně provede vektorový součin druhých 16 bitů zdrojového a cílového operandu. Následně tyto mezivýsledky sečte a výsledek uloží do dolních 32 bitů cílového operandu. Instrukce pracuje s celýma číslama a datovým typem Doubleword a Word.

Return Maximum Packed Double-Precision Floating-Point Values	
MAXPD c, z	Operandy: r128, r128/m128
c[0..63]=větší z hodnot c[0..63] a z[0..63] c[64..127]=větší z hodnot c[64..127] a z [64..127]	Instrukce provede vektorové porovnání FP hodnot s dvojitou přesností. Po porovnání hodnot ze zdrojového a cílového operandu instrukce zapíše do cílového operandu větší z těchto hodnot.

Return Maximum Scalar Double-Precision Floating-Point Value	
MAXSD c, z	Operandy: r128, r128/m64
c[0..63]=větší z hodnot c[0..63] a z [0..63] c[64..127] zůstane nezměnen	Instrukce provede skalární porovnání dolních FP hodnot s dvojitou přesností. Po porovnání hodnot ze zdrojového a cílového operandu instrukce zapíše do cílového operandu větší z těchto hodnot.

Return Minimum Packed Double-Precision Floating-Point Values	
MINPD c, z	Operandy: r128, r128/m128
c[0..63]=nižší z hodnot c[0..63] a z[0..63] c[64..127]=nižší z hodnot c[64..127] a z[64..127]	Instrukce provede vektorové porovnání FP hodnot s dvojitou přesností. Po porovnání hodnot ze zdrojového a cílového operandu instrukce zapíše do cílového operandu menší z těchto hodnot.

Return Minimum Scalar Double-Precision Floating-Point Value	
MINSD c, z	Operandy: r128, r128/m64
c[0..63]=nižší z hodnot c[0..63] a z[0..63] c[64..127] zůstane nezměnen	Instrukce provede skalární porovnání dolních FP hodnot s dvojitou přesností. Po porovnání hodnot ze zdrojového a cílového operandu instrukce zapíše do cílového operandu menší z těchto hodnot.

Move Byte Mask	
PMOVMSKB c, z	Operandy: r32, r128/r64
c[0]=z[7] c[1]=z[15] c[2]=z[23] ... c[8..31]=0 (v případě MM registru) nebo c[16..31]=0 (v případě XMM registru)	Instrukce vytvoří masku na základě MSB (Most Significant Bit) každého byte-u zdrojového operandu a výsledek uloží do cílového operandu.

Extract Packed Double-Precision Floating-Point Sign Mask	
MOVMSKPS c, z	Operandy: r64, r128
c[0]=z[63] c[1]=z[127] c[2..31]=0	Instrukce extrahuje vektorové hodnoty ze zdrojového operandu, naformátuje je jako 2bitovou masku a tuto masku uloží do cílového operandu.

Move Quadword from XMM to MMX Technology Register	
MOVDQ2Q c, z	Operandy: r64, r128
c[0..63]=z[0..63]	Instrukce naplní c celočíselné hodnotou datového typu Quadword ze zdrojového operandu.

Move Quadword from XMM to MMX Technology Register	
MOVQ2DQ c, z	Operandy: r128, r64
c[0..63]=z[0..63] c[64..127]=0	Instrukce naplní c celočíselné hodnotou datového typu Quadword ze zdrojového operandu.

Multiply Packed Unsigned Doubleword Integers	
PMULUDQ c, z	Operandy: r128, r128/m128 nebo r64, r64/m64

c[0..63]=c[0..31]*z[0..31] c[64..127]=c[64..95]*z[64..95]	Instrukce provede vektorový součin celočíselnéch hodnot datového typu Doubleword ze zdrojového a cílového operandu. Výsledkem je datový typ Quadword a tento výsledek je uložen v cílovém operandu.

Multiply Packed Double-Precision Floating-Point Values	
MULPD c, z	Operandy: r128, r128/m128
c[0..63]=c[0..63] * z[0..63] c[64..127]=c[64..127] * z[64..127]	Instrukce provede vektorové násobení FP hodnot s dvojitou přesností ze zdrojového a cílového operandu a výsledek uloží do cílového operandu.

Multiply Scalar Double-Precision Floating-Point Values	
MULSD c, z	Operandy: r128, r128/m128
c[0..63]=c[0..63] * z[0..63] c[64..127] zůstane nezměnen	Instrukce provede skalární násobení dolních FP hodnot s dvojitou přesností ze zdrojového a cílového operandu a výsledek uloží do cílového operandu.

Multiply Packed Signed Integers and Store High Result	
PMULHW c, z	Operandy: r128, r128/m128 nebo r64, r64/m64
MV=mezivýsledek: MV1[0..31]=c[0..15]*z[0..15] ... MV8[0..31]=c[112..127]*z[112..127] ---------- c[0..15]=MV1[16..31] ... c[112...127]=MV8[16..31]	Instrukce vektorově vynásobí celočíselné hodnoty datového typu Word se znaménkem (signed) ze zdrojového a cílového operandu. Do cílového operandu je uloženo horních 16 bitů mezivýsledku.

Multiply Packed Signed Integers and Store High Result	
PMULLW c, z	Operandy: r128, r128/m128 nebo r64, r64/m64
MV=mezivýsledek: MV1[0..31]=c[0..15]*z[0..15] ... MV8[0..31]=c[112..127]*z[112..127] ---------- c[0..15]=MV1[0..15] ... c[112...127]=MV8[0..15]	Instrukce vektorově vynásobí celočíselné hodnoty datového typu Word se znaménkem (signed) ze zdrojového a cílového operandu. Do cílového operandu je uloženo dolních 16 bitů mezivýsledku.

Bitwise Logical OR of Double-Precision Floating-Point Values	
ORPD c, z	Operandy: r128, r128/m128

c[0..127]=c[0..127] bitový OR z[0..127]	Instrukce provede bitový vektorový logický součet FP hodnot s dvojitou přesností ze zdrojového a cílového operandu a výsledek uloží do cílového operandu.

Bitwise Logical XOR for Double-Precision Floating-Point Values	
XORPD c, z	Operandy: r128, r128/m128
c[0..127]=c[0..127] bitový XOR z[0..127]	Instrukce provede bitový vektorový exkluzivní logický součet FP hodnot s dvojitou přesností ze zdrojového a cílového operandu a výsledek uloží do cílového operandu.

Shuffle Packed Double-Precision Floating-Point Values	
SHUFPD c, z, maska	Operandy: r128, r128/m32, imm8
c[0..63]=c[0..63] c[64..127]=z[0..63] nebo c[0..63]=c[64..127] c[64..127]=z[64..127]	Instrukce vektorově uloží jednu ze dvou DP FP hodnot z cílového operandu do dolních 64 bitů cílového operandu a následně uloží jednu ze dvou DP FP hodnot ze zdrojového operandu do horních 64 bitů cílového operandu. Maska definuje, které hodnoty budou vybrány.

Shift Left Logical Integers	
PSLLW c, maska **PSLLD c, maska** **PSLLQ c, maska**	Operandy: r128, r128/m64, imm8
c=posun vlevo (c)	Instrukce provede logický posun cílového operandu vlevo na základě masky. Pokud je hodnota masky větší než 15 (pro Word), nebo 31 (pro Doubleword), nebo 63 (pro Quadword) c je vynulován.

Compute Square Roots of Packed Double-Precision Floating-Point Values	
SQRTPD c, z	Operandy: r128, r128/m128
c[0..63]=√z[0..63] c[64..127]=√z[64..127]	Instrukce provede vektorové odmocnění FP hodnot s dvojitou přesností ve zdrojovém operandu a výsledek uloží do cílového operandu.

Compute Square Roots of Scalar Double-Precision Floating-Point Values	
SQRTSS c, z	Operandy: r128, r128/m128
c[0..63]=√z[0..63] c[64..127] zůstane nezměnen	Instrukce provede skalární odmocnění dolní FP hodnoty s dvojitou přesností ve zdrojovém operandu a výsledek uloží do dolních 64 bitů cílového operandu.

Shift Right Logical Integers	
PSRLW c, maska **PSRLD c, maska** **PSRLQ c, maska**	Operandy: r128, r128/m64/imm8
c=posun vpravo (c)	Instrukce provede logický posun cílového operandu vpravo na základě masky. Pokud je hodnota masky větší než 15 (pro Word), nebo 31 (pro Doubleword), nebo 63 (pro Quadword) c je vynulován.

Store Packed Double-Precision Floating-Point Values Using Non-Temporal Hint	
MOVNTPD c, z	Operandy: r128/m128, r128
c=z	Instrukce překopíruje FP hodnoty s dvojitou přesností ze zdrojového do cílového operandu. Instrukce minimalizuje zahlcení vyrovnávací paměte pomocí tzv. non-temporal hint.

Store Double Quadword Using Non-Temporal Hint	
MOVNTDQ c, z	Operandy: r128/m128, r128
c=z	Instrukce překopíruje celočíselné hodnoty datového typu Quadword ze zdrojového do cílového operandu. Instrukce minimalizuje zahlcení vyrovnávací paměte pomocí tzv. non-temporal hint.

Store Doubleword Using Non-Temporal Hint	
MOVNTI c, z	Operandy: m32, r32
c=z	Instrukce vektorově překopíruje celočíselné hodnoty datového typu Doubleword ze zdrojo- vého do cílového operandu. Instrukce minimalizuje zahlcení vyrovnávací paměte pomocí tzv. non-temporal hint.

Subtract Packed Integers	
PSUBB c, z **PSUBW c, z** **PSUBD c, z**	Operandy: r128, r128/m128 nebo r64, r64/m64
c[x..y]=c[x..y]-z[x..y]	Instrukce provede vektorové odčítání celočíselných hodnot datových tupů Byte/Word/Doubleword ze zdrojového a cílového operandu a výsledek uloží do cílového operandu.

Subtract Packed Quadword Integers	
PSUBQ c, z	Operandy: r128, r128/m128 nebo r64, r64/m64

c[0..63]=c[0..63]-z[0..63] c[64..127]=c[64..127]-z[64..127]	Instrukce provede vektorové odčítání celočíselných hodnot datového typu Quadword ze zdrojového a cílového operandu a výsledek uloží do cílového operandu.

Subtract Packed Double-Precision Floating-Point Values	
SUBPD c, z	Operandy: r128, r128/m128
c[0..63]=c[0..63]-z[0..63] c[64..127]=c[64..127]-z[64..127]	Instrukce provede vektorové odčítání dvou FP hodnot s dvojitou přesností ze zdrojového a cílového operandu a výsledek uloží do cílového operandu.

Subtract Scalar Double-Precision Floating-Point Values	
SUBSD c, z	Operandy: r128, r128/m64
c[0..63]=c[0..63]-z[0..63] c[64..127] zůstane nezměnen	Instrukce provede skalární odčítání dolní FP hodnoty s dvojitou přesností ze zdrojového a cílového operandu a výsledek uloží do dolních 64 bitů cílového operandu.

Unordered Compare Scalar Single-Precision Floating- Point Values and Set EFLAGS	
UCOMISD z 1, z 2	Operandy: r128, r128/m64
Výsledek porovnání=neuspořádaný, ZF=1; PF=1; CF=1; Výsledek porovnání=větší, ZF=0; PF=0; CF=0; Výsledek porovnání=menší, ZF=0; PF=0; CF=1; Výsledek porovnání=rovná se, ZF=1; PF=0; CF=0; Značky OF, AF a SF se nastaví na 0.	Instrukce provede skalární porovnání dolní poloviny (64 bitů) hodnot ze zdrojových operandů. Podle výsledku porovnání (větší, menší, rovná se nebo neuspořádaný) nastaví značky registru EFLAGS.

Unpack High Data	
PUNPCKHBW c, z PUNPCKHWD c, z PUNPCKHDQ c, z PUNPCKHQDQ c, z	Operandy: r128, r128/m128 nebo r64, r64/m64
Př. - Doubleword: c[0..31]=c[64..95] c[32..63]=z[64..95] c[64..95]=c[96..127] c[96..127]=z[96..127]	Instrukce načítá dvě horní celočíselní hodnoty datových tupů Byte/Word/Doubleword/Quadword z cílového a zdrojového operandu a uloží je přeloženě do cílového operandu. Viz operace.

Unpack Low Data	
PUNPCKLBW c, z PUNPCKLWD c, z PUNPCKLDQ c, z PUNPCKLQDQ c, z	Operandy: r128, r128/m128 nebo r64, r64/m64

Př. - Doubleword: c[0..31]=c[0..31] c[32..63]=z[0..31] c[64..95]=c[32..63] c[96..127]=z[32..63]	Instrukce načítá dvě dolní celočíselní hodnoty datových tupů Byte/Word/Doubleword/Quadword z cílového a zdrojového operandu a uloží je přeloženě do cílového operandu. Viz operace.

Unpack and Interleave High Packed Double-Precision Floating-Point Values	
UNPCKHPD c, z	Operandy: r128, r128/m128
c[0..63]=c[64..127] c[64..127]=z[64..127]	Instrukce načítá dvě horní FP hodnoty s dvojitou přesností z cílového a zdrojového operandu a uloží je přeloženě do cílového operandu. Viz operace.

Unpack and Interleave Low Packed Double-Precision Floating-Point Values	
UNPCKLPD c, z	Operandy: r128, r128/m128
c[0..63]=c[0..63] c[64..127]=z[0..63]	Instrukce načítá dvě dolní FP hodnoty s dvojitou přesností z cílového a zdrojového operandu a uloží je přeloženě do cílového operandu. Viz operace.

Instrukční sada SSE3

Packed Single-FP Add/Subtract	
ADDSUBPS c, z	Operandy: r128, r128/m128
c[0..31]=c[0..31]-z[0..31] c[32..63]=c[32..63]+z[32..63] c[64..95]=c[64..95]-z[64..95] c[96..127]=c[96..127]+z[96..127]	Instrukce odečítá 2 FP hodnoty a sčítá 2 FP hodnoty s jednoduchou přesností z cílového a zdrojového operandu a výsledek uloží do cílového operandu. Viz operace.

Packed Double-FP Add/Subtract	
ADDSUBPD c, z	Operandy: r128, r128/m128
c[0..63]=c[0..63]-z[0..63] c[64..127]=c[64..127]+z[64..127]	Instrukce odečítá dolní poloviny cílového a zdrojového operandu a sčítá horní poloviny cílového a zdrojového operandu. Výsledek uloží do cílového operandu. Pracuje s dvojitou přesností.

Packed Single-FP Horizontal Add	
HADDPS c, z	Operandy: r128, r128/m128

c[0..31]=c[0..31]+c[32..63] c[32..63]=c[64..95]+c[96..127] c[64..95]=z[0..31]+z[32..63] c[96..127]=z[64..95]+z[96..127]	Instrukce sčítá 2 spodní hodnoty cílového operandu a výsledek uloží do nejspodnejší hodnoty cílového operandu. Pak sčítá 2 horní hodnoty cílového operandu a výsledek uloží do druhé nejspodnejší hodnoty cílového operandu. Postup je pro zdrojový operand stejný, ale výsledek se ukládá do horní poloviny cílového operandu. Viz operace.

Packed Double-FP Horizontal Add	
HADDPD c, z	Operandy: r128, r128/m128
c[0..63]=c[0..63]+c[64..127] c[64..127]=z[0..63]+z[64..127]	Instrukce sčítá 2 FP hodnoty s dvojitou přesností v cílovém operandu a výsledek uloží do dolní poloviny cílového operandu. Pak sčítá 2 FP hodnoty s dvojitou přesností v zdrojovém operandu a výsledek uloží do horní poloviny cílového operandu.

Packed Single-FP Horizontal Substract	
HSUBPS c, z	Operandy: r128, r128/m128
c[0..31]=c[0..31]-c[32..63] c[32..63]=c[64..95]-c[96..127] c[64..95]=z[0..31]-z[32..63] c[96..127]=z[64..95]-z[96..127]	Instrukce odečítá 2 spodní hodnoty cílového operandu a výsledek uloží do nejspodnejší hodnoty cílového operandu. Pak odečítá 2 horní hodnoty cílového operandu a výsledek uloží do druhé nejspodnejší hodnoty cílového operandu. Postup je pro zdrojový operand stejný, ale výsledek se ukládá do horní poloviny cílového operandu. Viz operace.

Packed Double-FP Horizontal Substract	
HSUBPD c, z	Operandy: r128, r128/m128
c[0..63]=c[0..63]-c[64..127] c[64..127]=z[0..63]-z[64..127]	Instrukce odečítá 2 FP hodnoty s dvojitou přesností v cílovém operandu a výsledek uloží do dolní poloviny cílového operandu. Pak odečítá 2 FP hodnoty s dvojitou přesností v zdrojovém operandu a výsledek uloží do horní poloviny cílového operandu.

Load Unaligned Integer 128 Bits	
LDDQU c, z	Operandy: r128, r128/m128
c[0..127]=z[0..127]	Instrukce načítá celočíselní hodnoty ze zdrojového operandu a skopíruje je do cílového operandu.

Move One Double-FP and Duplicate	
MOVDDUP c, z	Operandy: r128, r128/m128
c[0..63]=z[0..63] c[64..127]=z[0..63]	Instrukce načítá jednu FP hodnotu s dvojitou přesností ze zdrojového operandu a skopíruje ji do celého cílového operandu.

Move Packed Single-FP High and Duplicate	
MOVSHDUP c, z	Operandy: r128, r128/m128
c[0..31]=z[32..63] c[32..63]=z[32..63] c[64..95]=z[96..127] c[96..127]=z[96..127]	Instrukce načítá horní FP hodnotu s jednoduchou přesností z dolní poloviny zdrojového operandu a skopíruje ji do dolní poloviny cílového operandu. Pak načítá horní FP hodnotu s jednoduchou přesností z horní poloviny zdrojového operandu a uloží ji do horní poloviny cílového operandu.

Move Packed Single-FP Low and Duplicate	
MOVSLDUP c, z	Operandy: r128, r128/m128
c[0..31]=z[0..31] c[32..63]=z[0..31] c[64..95]=z[64..95] c[96..127]=z[64..95]	Instrukce načítá dolní FP hodnotu s jednoduchou přesností z dolní poloviny zdrojového operandu a skopíruje ji do dolní poloviny cílového operandu. Pak načítá dolní FP hodnotu s jednoduchou přesností z horní poloviny zdrojového operandu a uloží ji do horní poloviny cílového operandu.

Instrukční sada SSSE3

Packed Absolute Value	
PABSB c, z **PABSW c, z** **PABSD c, z**	Operandy: r128, r128/m128 nebo r64, r64/m64
c[0..127]= | z[0..127] |	Instrukce vypočítá absolutní hodnotu každého elementu zdrojového operandu a výsledek uloží do cílového operandu. Pracuje s datovýma typama Byte, Word, Doubleword.

Packed Horizontal Add	
PHADDW c, z **PHADDD c, z**	Operandy: r128, r128/m128 nebo r64, r64/m64
c[0..31]=c[0..31]+c[32..63] c[32..63]=c[64..95]+c[96..127] c[64..95]=z[0..31]+z[32..63] c[96..127]=z[64..95]+z[96..127]	Chování instrukce je téměř zhodné s instrukcí HADDPS, ovšem tahle pracuje s celýma číslama a datovýma typama Word a Doubleword. Viz operace.

Packed Horizontal Substract	
PHSUBW c, z **PHSUBD c, z**	Operandy: r128, r128/m128 nebo r64, r64/m64
c[0..31]=c[0..31]-c[32..63] c[32..63]=c[64..95]-c[96..127] c[64..95]=z[0..31]-z[32..63] c[96..127]=z[64..95]-z[96..127]	Chování instrukce je téměř zhodné s instrukcí HSUBPS, ovšem tahle pracuje s celýma číslama a datovýma typama Word a Doubleword. Viz operace.

Multiply and Add Packed Signed and Unsigned Bytes	
PMADDUBSW c, z	Operandy: r128, r128/m128 nebo r64, r64/m64
c[0..15]=c[0..7]*z[0..7]+c[8..15]*z[8..15] c[16..31]=c[16..23]*z[16..23]+ +c[24..31]*z[24..31] ..... c[112..127]=c[112..119]*z[112..119]+ +c[120..127]*z[120..127]	Instrukce vynásobí celočíselní hodnoty datového typu Byte ze zdrojového a cílového operandu, sčítá je a uloží jako datový typ Word do cílového operandu. Viz operace.

Packed Shuffle Bytes	
PSHUFB c, z	Operandy: r128, r128/m128 nebo r64, r64/m64
c[0..127]=výběr na základě masky(c[0..127])	Instrukce vybere z cílového operandu hodnoty na základě masky v zdrojovém operandu a uloží je do cílového operandu. Pracuje s datovým typem Byte.

Packed SIGN	
PSIGNB c, z **PSIGNW c, z** **PSIGND c, z**	Operandy: r128, r128/m128 nebo r64, r64/m64
c[0..127]=negace/ponechání/vynulování na základě hodnot v z[0..127]	Ak je hodnota zdrojového operandu <0, pak je hodnota cílového operandu znegovaná. Ak je hodnota zdrojového operandu >0, pak je hodnota cílového operandu nezměnena. Ak je hodnota zdrojového operandu =0, pak je hodnota cílového operandu vynulovaná. Pracuje s datovýma typama Byte, Word, Doubleword.

Instrukční sada SSE4.1

Blend Packed Words	
PBLENDW c, z, maska	Operandy: r128, r128/m128, imm8
c[0..15]=c[0..15] nebo z[0..15] c[16..31]=c[16..31] nebo z[16..31] ... c[112..127]=c[112..127] nebo z[112..127]	Instrukce pomíchá na základě masky hodnoty datového typu Word ze zdrojového a cílového operandu a vybranou hodnotu uloží do cílového operandu.

Blend Packed Double-FP Values	
PBLENDPD c, z, maska	Operandy: r128, r128/m128, imm8
c[0..63]=c[0..63] nebo z[0..63] c[63..127]=c[63..127] nebo z[63..127]	Instrukce pomíchá na základě masky FP hodnoty s dvojitou přesností ze zdrojového a cílového operandu a vybranou hodnotu uloží do cílového operandu.

Blend Packed Single-FP Values	
PBLENDPS c, z, maska	Operandy: r128, r128/m128, imm8
c[0..31]=c[0..31] nebo z[0..31] c[32..63]=c[32..63] nebo z[32..63] c[64..95]=c[64..95] nebo z[64..95] c[96..127]=c[96..127] nebo z[96..127]	Instrukce pomíchá na základě masky FP hodnoty s jednoduchou přesností ze zdrojového a cílového operandu a vybranou hodnotu uloží do cílového operandu.

Variable Blend Packed Bytes	
PBLENDB c, z, maska	Operandy: r128, r128/m128, imm8
c[0..7]=c[0..7] nebo z[0..7] c[8..15]=c[8..15] nebo z[8..15] ... c[120..127]=c[120..127] nebo z[120..127]	Instrukce pomíchá na základě masky hodnoty datového typu Byte ze zdrojového a cílového operandu a vybranou hodnotu uloží do cílového operandu.

Round Packed Single Precision Floating-Point Values	
ROUNDPS c, z, maska	Operandy: r128, r128/m128, imm8
c[0..31]=zaokrouhlení(z[0..31]) c[32..63]=zaokrouhlení(z[32..63]) c[64..95]=zaokrouhlení(z[64..95]) c[96..127]=zaokrouhlení(z[96..127])	Instrukce zaokrouhlí FP hodnoty s jednoduchou přesností ze zdrojového operandu na základě masky na celočíselní hodnotu a výsledek uloží jako FP hodnotu s jednoduchou přesností do cílového operandu.

Round Packed Double Precision Floating-Point Values	
ROUNDPS c, z, maska	Operandy: r128, r128/m128, imm8
c[0..63]=zaokrouhlení(z[0..63]) c[64..127]=zaokrouhlení(z[64..127])	Instrukce zaokrouhlí FP hodnoty s dvojitou přesností ze zdrojového operandu na základě masky na celočíselní hodnotu a výsledek uloží jako FP hodnotu sdvojitou přesností do cílového operandu.

Round Scalar Single Precision Floating-Point Values	
ROUNDSS c, z, maska	Operandy: r128, r128/m32, imm8
c[0..31]=zaokrouhlení(z[0..31]) c[32..127]=z[32..127]	Instrukce zaokrouhlí dolní FP hodnotu s jednoduchou přesností ze zdrojového operandu na základě masky na celočíselní hodnotu a výsledek uloží jako dolní FP hodnotu s jednoduchou přesností do cílového operandu.

Round Scalar Doubl Precision Floating-Point Values	
ROUNDSD c, z, maska	Operandy: r128, r128/m64, imm8
c[0..63]=zaokrouhlení(z[0..63]) c[64..127]=z[64..127]	Instrukce zaokrouhlí dolní FP hodnotu s dvojitou přesností ze zdrojového operandu na základě masky na celočíselní hodnotu a výsledek uloží jako dolní FP hodnotu s dvojitou přesností do cílového operandu.

Compare Packed Qword Data for Equal	
PCMPEQQ c, z	Operandy: r128, r128/m128

Pokud c[0..63]=z[0..63] c[0..63]=FFFFFFFFFFFFFFFFH jinak c[0..63]=0; Pokud c[64..127]=z[64..127] c[64..127]=FFFFFFFFFFFFFFFFH jinak c[64..127]=0;	Instrukce porovná celočíselní hodnoty datového typu Quadword. Pokud jsou hodnoty rovny, bity cílového operandu jsou nastaveny na 1, jinak na 0.

Dot Product of Packed Single Precision Floating-Point Values	
DPPS c, z, maska	Operandy: r128, r128/m128, imm8
Příklad: c[0..31]= buď c[0..31]*z[0..31] nebo c[0..31]	Instrukce na základě masky buď vynásobí FP hodnoty s jednoduchou přesností z cílového a zdrojového operandu, nebo ponechá hodnotu cílového operandu nezměnenou.

Dot Product of Packed Double Precision Floating-Point Values	
DPPD c, z, maska	Operandy: r128, r128/m128, imm8
Příklad: c[0..63]= buď c[0..63]*z[0..63] nebo c[0..63]	Instrukce na základě masky buď vynásobí FP hodnoty s dvojitou přesností z cílového a zdrojového operandu, nebo ponechá hodnotu cílového operandu nezměnenou.

Extract Dword	
PEXTRD c, z, maska	Operandy: r32/m32, r128, imm8
c[0..32]=výběr pomocí masky(z[0..127])	Instrukce skopíruje 32 bitů ze zdrojového operandu, zvolených pomocí masky, do dolních 32 cílového operandu. Horní hodnoty cílového operandu jsou vynulovány. Pracuje s datovým typem Doubleword.

Extract Quadword	
PEXTRQ c, z, maska	Operandy: r64/m64, r128, imm8
c[0..63]=výběr pomocí masky(z[0..127])	Instrukce skopíruje 64 bitů ze zdrojového operandu, zvolených pomocí masky, do dolních 64 cílového operandu. Horní hodnoty cílového operandu jsou vynulovány. Pracuje s datovým typem Quadword.

Extract Byte	
PEXTRQ c, z, maska	Operandy: r32/m8, r128, imm8
c[0..7]=výběr pomocí masky(z[0..127])	Instrukce skopíruje 8 bitů ze zdrojového operandu, zvolených pomocí masky, do dolních 8 cílového operandu. Horní hodnoty cílového operandu jsou vynulovány. Pracuje s datovým typem Quadword.

Extract Packed Single Precision Floating-Point Value	
EXTRACTPS c, z, maska	Operandy: r32/m32, r128, imm8

c[0..7]=výběr pomocí masky(z[0..127])	Instrukce skopíruje 32 bitů ze zdrojového operandu, zvolených pomocí masky, do dolních 32 cílového operandu. Horní hodnoty cílového operandu jsou vynulovány. Pracuje s jednoduchou přesností.

Insert Byte	
PINSRB c, z, maska	Operandy: r128, r32/m8, imm8
c[výběr lokace pomocí masky]=z[0..7]	Instrukce skopíruje datový typ Byte ze zdrojového operandu a vloží ho do lokace cílového operandu zvolené pomocí masky. Ostatní hodnoty v cílovém operandu zůstanou nezměneny.

Insert Doubleword	
PINSRD c, z, maska	Operandy: r128, r32/r64/r128/m32, imm8
c[výběr lokace pomocí masky]=z[0..31]	Instrukce skopíruje datový typ Doubleword ze zdrojového operandu a vloží ho do lokace cílového operandu zvolené pomocí masky. Ostatní hodnoty v cílovém operandu zůstanou nezměneny.

Insert Quadword	
PINSRQ c, z, maska	Operandy: r128, r64/r128/m64, imm8
c[výběr lokace pomocí masky]=z[0..63]	Instrukce skopíruje datový typ Quadword ze zdrojového operandu a vloží ho do lokace cílového operandu zvolené pomocí masky. Ostatní hodnoty v cílovém operandu zůstanou nezměneny.

Insert Doubleword	
INSERTPS c, z, maska	Operandy: r128, r128/m32, imm8
c[výběr lokace pomocí masky]=z[0..31]	Instrukce skopíruje FP hodnotu s jednoduchou přesností ze zdrojového operandu a vloží ho do lokace cílového operandu zvolené pomocí masky. Ostatní hodnoty v cílovém operandu zůstanou nezměneny.

Maximum of Packed Signed Byte Integers	
PMAXSB c, z	Operandy: r128, r128/m128
c[0..7]=větší z hodnot c[0..7] a z[0..7] c[8..15]=větší z hodnot c[8..15] a z[8..15] ... c[120..127]=větší: c[120..127] a z[120..127]	Instrukce provede vektorové porovnání celočíslených hodnot datového typu Byte se znaménkem (signed). Po porovnání hodnot ze zdrojového a cílového operandu instrukce zapíše do cílového operandu větší z těchto hodnot.

Maximum of Packed Signed Doubleword Integers	
PMAXSD c, z	Operandy: r128, r128/m128

c[0..31]=větší z hodnot c[0..31] a z[0..31] c[32..63]=větší z hodnot c[32..63] a z[32..63] ... c[96..127]=větší: c[96..127]a z[96..127]	Instrukce provede vektorové porovnání celočíslených hodnot datového typu Doubleword se znaménkem (signed). Po porovnání hodnot ze zdrojového a cílového operandu instrukce zapíše do cílového operandu větší z těchto hodnot.

Maximum of Packed Unsigned Word Integers	
PMAXUW c, z	Operandy: r128, r128/m128
c[0..15]=větší z hodnot c[0..15] a z[0..15] c[16..31]=větší z hodnot c[16..31] a z[16..31] ... c[112..127]=větší: c[112..127] a z[112..127]	Instrukce provede vektorové porovnání celočíslených hodnot datového typu Word bez znaménka (unsigned). Po porovnání hodnot ze zdrojového a cílového operandu instrukce zapíše do cílového operandu větší z těchto hodnot.

Maximum of Packed Unsigned Doubleword Integers	
PMAXUD c, z	Operandy: r128, r128/m128
c[0..31]=větší z hodnot c[0..31] a z[0..31] c[32..63]=větší z hodnot c[32..63] a z[32..63] ... c[96..127]=větší: c[96..127]a z[96..127]	Instrukce provede vektorové porovnání celočíslených hodnot datového typu Doubleword bez znaménka (unsigned). Po porovnání hodnot ze zdrojového a cílového operandu instrukce zapíše do cílového operandu větší z těchto hodnot.

Minimum of Packed Signed Byte Integers	
PMINSB c, z	Operandy: r128, r128/m128
c[0..7]=menší z hodnot c[0..7] a z[0..7] c[8..15]=menší z hodnot c[8..15] a z[8..15] ... c[120..127]=menší: c[120..127] a z[120..127]	Instrukce provede vektorové porovnání celočíslených hodnot datového typu Byte se znaménkem (signed). Po porovnání hodnot ze zdrojového a cílového operandu instrukce zapíše do cílového operandu menší z těchto hodnot.

Minimum of Packed Signed Doubleword Integers	
PMINSD c, z	Operandy: r128, r128/m128
c[0..31]=větší z hodnot c[0..31] a z[0..31] c[32..63]=větší z hodnot c[32..63] a z[32..63] ... c[96..127]=větší: c[96..127]a z[96..127]	Instrukce provede vektorové porovnání celočíslených hodnot datového typu Doubleword se znaménkem (signed). Po porovnání hodnot ze zdrojového a cílového operandu instrukce zapíše do cílového operandu větší z těchto hodnot.

Minimum of Packed Unsigned Word Integers	
PMINUW c, z	Operandy: r128, r128/m128
c[0..15]=menší z hodnot c[0..15] a z[0..15] c[16..31]=menší z hodnot c[16..31] a z[16..31] ... c[112..127]=menší: c[112..127] a z[112..127]	Instrukce provede vektorové porovnání celočíslených hodnot datového typu Word bez znaménka (unsigned). Po porovnání hodnot ze zdrojového a cílového operandu instrukce zapíše do cílového operandu menší z těchto hodnot.

Minimum of Packed Unsigned Doubleword Integers	
PMINUD c, z	Operandy: r128, r128/m128
c[0..31]=menší z hodnot c[0..31] a z[0..31] c[32..63]=menší z hodnot c[32..63] a z[32..63] ... c[96..127]=menší: c[96..127]a z[96..127]	Instrukce provede vektorové porovnání celočíslených hodnot datového typu Doubleword bez znaménka (unsigned). Po porovnání hodnot ze zdrojového a cílového operandu instrukce zapíše do cílového operandu menší z těchto hodnot.

Multiply Packed Signed Doubleword Integers	
PMULDQ c, z	Operandy: r128, r128/m128 nebo r64, r64/m64
c[0..63]=c[0..31]*z[0..31] c[64..127]=c[64..95]*z[64..95]	Instrukce provede vektorový součin celočíselnéch hodnot datového typu Doubleword ze zdrojového a cílového operandu. Výsledkem je datový typ Quadword a tento výsledek je uložen v cílovém operandu.

Pack with Unsigned Saturation	
PACKUSDW c, z	Operandy: r128, r128/m128
c[0..15]=kononverze(c[0..31]) c[16..31]=konverze(c[32..63]) ... c[96..111]=konverze(z[64..95]) c[112..127]=konverze(z[96..127])	Instrukce provede vektorovou konverzi 4 celočíselních hodnot datového typu Doubleword z cílového a zdrojového operandu na 8 celočíselních hodnot datového typu word a výsledek uloží do cílového operandu.

Load Double Quadword Non-Temporal Aligned Hint	
MOVNTDQA c, z	Operandy: r128, m128
c=z	Instrukce vektorově překopíruje celočíselné hodnoty datového typu Quadword ze zdrojového do cílového operandu. Instrukce minimalizuje zahlcení vyrovnávací paměte pomocí tzv. non-temporal hint.

Logical Compare	
PTEST c, z	Operandy: r128, r128/m128
Pokud c[0..127] AND z[0..127] = 0, ZF = 1 jinak ZF = 0. Pokud c[0..127] AND NOT z[0..127] = 0, CF = 1 jinak CF = 0.	Instrukce provede bitový logický součin zdrojového a cílového operandu a nastaví značky ZF. Pak provede negovaný logický součin a nastaví značky CF.

Instrukční sada SSE4.2

Packed Compare Explicit Length Strings, Return Index	
PCMPESTRI c, z, maska	Operandy: r128, r128/m128, imm8
	Instrukce má složité chování, viz [1], [2].

Compare Packed Data for Greater Than	
PCMPGTQ c, z	Operandy: r128, r128/m128
pokud (c[x..y] > z[x..y]) c[x..y]=1 jinak c[x..y]=0	Instrukce provede porovnání celočíslených hodnot datového typu Quadword pro rovnost. Pokud je hodnota cílového operandu větší než hodnota zdrojového operandu, pak je cílová hodnota nastavena na 1. Jinak je cílová hodnota nastavena na 0.

Accumulate CRC32 Value	
CRC32 c, z	Operandy: r32/r64, m8 až m64
	Inštrukce akumuluje hodnotu CRC32 pomocí hodnot ze zdrojového a cílového operandu. Viz [2].

Instrukční sada AVX

Vex Add Packed Single-Precision Floating-Point Values	
VADDPS c, z1, z2	Operandy: r256, r256, r256/m256
c[0..31]=z1[0..31]+z2[0..31] c[32..63]=z1[32..63]+z2[32..63] ... c[224..255]=z1[224..255]+z2[224..255]	Instrukce provede vektorový součet 8 FP hodnot s jednoduchou přesností ze zdrojových operandů a výsledek uloží do cílového operandu.

Vex Add Packed Double-Precision Floating-Point Values	
VADDPS c, z1, z2	Operandy: r256, r256, r256/m256
c[0..63]=z1[0..63]+z2[0..63] c[64..127]=z1[64..127]+z2[64..127] c[128..191]=z1[128..191]+z2[128..191] c[192..255]=z1[192..255]+z2[192..255]	Instrukce provede vektorový součet 4 FP hodnot s dvojitou přesností ze zdrojových operandů a výsledek uloží do cílového operandu.

Vex Packed Single-FP Add/Subtract	
VADDSUBPS c, z1, z2	Operandy: r256, r256, r256/m256
c[0..31]=c[0..31]-z[0..31] c[32..63]=c[32..63]+z[32..63] c[64..95]=c[64..95]-z[64..95] ... c[224..255]=c[224..255]+z[224..255]	Instrukce odečítá 4 FP hodnoty a sčítá 4 FP hodnoty s jednoduchou přesností ze zdrojových operandů a výsledek uloží do cílového operandu. Viz operace.

Vex Packed Double-FP Add/Subtract	
VADDSUBPD c, z1, z2	Operandy: r256, r256, r256/m256
c[0..63]=c[0..63]-z[0..63] c[64..127]=c[64..127]+z[64..127] c[128..191]=c[128..191]-z[128..191] c[192..255]=c[192..255]+z[192..255]	Instrukce odečítá 2 FP hodnoty a sčítá 2 FP hodnoty s dvojitou přesností ze zdrojových operandů a výsledek uloží do cílového operandu. Viz operace.

Vex Bitwise Logical AND of Packed Single-Precision Floating-Point Values	
VANDPS c, z1, z2	Operandy: r256, r256, r256/m256
c[0..255]=z1[0..255] bitový AND z2[0..255]	Instrukce provede bitový vektorový logický součin FP hodnot s jednoduchou přesností ze zdrojových operandů a výsledek uloží do cílového operandu.

Vex Bitwise Logical AND of Packed Double-Precision Floating-Point Values	
VANDPS c, z1, z2	Operandy: r256, r256, r256/m256
c[0..255]=z1[0..255] bitový AND z2[0..255]	Instrukce provede bitový vektorový logický součin FP hodnot s dvojitou přesností ze zdrojových operandů a výsledek uloží do cílového operandu.

Vex Bitwise Logical AND NOT of Packed Single-Precision Floating-Point Values	
VANDNPS c, z1, z2	Operandy: r256, r256, r256/m256
c[0..255]=z1[0..255] bit. AND NOT z2[0..255]	Instrukce provede negovaný bitový vektorový logický součin FP hodnot s jednoduchou přesností ze zdrojových operandů a výsledek uloží do cílového operandu.

Vex Bitwise Logical AND NOT of Packed Double-Precision Floating-Point Values	
VANDNPD c, z1, z2	Operandy: r256, r256, r256/m256
c[0..255]=z1[0..255] bit. AND NOT z2[0..255]	Instrukce provede negovaný bitový vektorový logický součin FP hodnot s dvojitou přesností ze zdrojových operandů a výsledek uloží do cílového operandu.

Vex Blend Packed Single-FP Values	
VPBLENDPS c, z1, z2, maska	Operandy: r256, r256, r256/m256, imm8
c[0..31]=c[0..31] nebo z[0..31] c[32..63]=c[32..63] nebo z[32..63] ... c[224..255]=c[224..255] nebo z[224..255]	Instrukce pomíchá na základě masky FP hodnoty s jednoduchou přesností ze zdrojového a cílového operandu a vybranou hodnotu uloží do cílového operandu.

Vex Blend Packed Double-FP Values	
VPBLENDPD c, z1, z2, maska	Operandy: r256, r256, r256/m256, imm8
c[0..63]=c[0..63] nebo z[0..63] c[64..127]=c[64..127] nebo z[64..127] c[128..191]=c[128..191] nebo z[128..191] c[192..255]=c[192..255] nebo z[192..255]	Instrukce pomíchá na základě masky FP hodnoty s dvojitou přesností ze zdrojového a cílového operandu a vybranou hodnotu uloží do cílového operandu.

Vex Broadcast Single-FP Data	
VBROADCASTSS c, z	Operandy: r128/r256, m32

c[0..31]=z c[32..63]=z ... c[224..255]=z	Instrukce nahrá 1 FP hodnotu s jednoduchou přesností ze zdrojového operandu do všech hodnot cílového operandu.

Vex Broadcast Double-FP Data	
VBROADCASTSD c, z	Operandy: r256, m64
c[0..63]=z c[64..127]=z c[128..191]=z c[192..255]=z	Instrukce nahrá 1 FP hodnotu s dvojitou přesností ze zdrojového operandu do všech hodnot cílového operandu.

Vex Broadcast Data	
VBROADCASTF128 c, z	Operandy: r256, m128
c[0..127]=z c[192..255]=z	Instrukce nahrá 1 FP hodnotu dané přesnosti ze zdrojového operandu do všech hodnot cílového operandu.

Vex Round Packed Single Precision Floating-Point Values	
VROUNDPS c, z, maska	Operandy: r256, r256/m256, imm8
c[0..31]=zaokrouhlení(z[0..31]) c[32..63]=zaokrouhlení(z[32..63]) ... c[224..255]=zaokrouhlení(z[224..255])	Instrukce zaokrouhlí FP hodnoty s jednoduchou přesností ze zdrojového operandu na základě masky na celočíselní hodnotu a výsledek uloží jako FP hodnotu s jednoduchou přesností do cílového operandu.

Vex Round Packed Double Precision Floating-Point Values	
VROUNDPD c, z, maska	Operandy: r256, r256/m256, imm8
c[0..63]=zaokrouhlení(z0..63]) c[64..127]=zaokrouhlení(z[64..127]) c[128..191]=zaokrouhlení(z[128..191]) c[192..255]=zaokrouhlení(z[192..255])	Instrukce zaokrouhlí FP hodnoty s dvojitou přesností ze zdrojového operandu na základě masky na celočíselní hodnotu a výsledek uloží jako FP hodnotu s dvojitou přesností do cílového operandu.

Vex Compare Packed Single-Precision Floating-Point Values	
VCMPPS c, z1, z2, maska	Operandy: r256, r256, r256/m256, imm8

CMP0=z1[0..31] operand (imm8) z2[0..31]
if (CMP0==true) c[0..31]=0xFFFFFFFF;
else c[0..31]=0;
...
CMP4=z1[224..255] operand (imm8) z2[224..255]
if (CMP4==true) c[224..255]=0xFFFFFFFF;
else c[224..255]=0;
	Instrukce provede vektorové porovnání hodnot, na základe hexadecimální hodnoty imm8 (větší, menší, rovná se atd), ze zdrojových operandů a výsledek uloží do cílového operandu.

Vex Compare Packed Double-Precision Floating-Point Values	
VCMPPD c, z1, z2, maska	Operandy: r256, r256, r256/m256, imm8

CMP0=z1[0..63] operand (imm8) z2[0..63]
if (CMP0==true) c[0..63]=0xFFFFFFFF;
else c[0..63]=0;
...
CMP4=z1[191..255] operand (imm8) z2[191..255]
if (CMP4==true) c[191..255]=0xFFFFFFFF;
else c[191..255]=0;
	Instrukce provede vektorové porovnání hodnot, na základe hexadecimální hodnoty imm8 (větší, menší, rovná se atd), ze zdrojových operandů a výsledek uloží do cílového operandu.

Vex Compare Scalar Single-Precision Floating-Point Values	
VCMPPS c, z1, z2, maska	Operandy: r256, r256, r256/m256, imm8

CMP0=z1[0..31] operand (imm8) z2[0..31]
if (CMP0==true) c[0..31]=0xFFFFFFFF;
else c[0..31]=0;
c[32..255] zůstane nezměnen
	Instrukce provede vektorové porovnání dolních hodnot, na základe hexadecimální hodnoty imm8 (větší, menší, rovná se atd), ze zdrojových operandů a výsledek uloží do cílového operandu.

Vex Compare Scalar Double-Precision Floating-Point Values	
VCMPPD c, z1, z2, maska	Operandy: r256, r256, r256/m256, imm8

CMP0=z1[0..63] operand (imm8) z2[0..631]
if (CMP0==true) c[0..63]=0xFFFFFFFF;
else c[0..63]=0;
c[64..255] zůstane nezměnen
	Instrukce provede vektorové porovnání dolních hodnot, na základe hexadecimální hodnoty imm8 (větší, menší, rovná se atd), ze zdrojových operandů a výsledek uloží do cílového operandu.

Vex Convert Packed Doubleword Integers to Packed Double-Precision Floating-Point Values	
VCVTDQ2PD c, z	Operandy: r256, r128/m128
c[0..63]=konverze na FP (z[0..31]) c[64..127]=konverze na FP (z[32..63]) c[128..191]=konverze na FP (z[64..95]) c[192..255]=konverze na FP (z[96..127])	Instrukce provede konverzi 4 celočíselných hodnot datového typu Doubleword ze zdrojového operandu na 4 FP hodnoty s dvojitou přesností a výsledek uloží do cílového operandu.

Vex Convert Packed Doubleword Integers to Packed Single-Precision Floating-Point Values	
VCVTDQ2PS c, z	Operandy: r256, r256/m256
c[0..31]=konverze na FP (z[0..31]) c[32..63]=konverze na FP (z[32..63]) ... c[192..255]=konverze na FP (z[192..255])	Instrukce provede konverzi 8 celočíselných hodnot datového typu Doubleword ze zdrojového operandu na 8 FP hodnoty s jednoduchou přesností a výsledek uloží do cílového operandu.

Vex Convert Packed Double-Precision Floating-Point Values to Packed Doubleword Integers	
VCVTPD2DQ c, z	Operandy: r128, r256/m256
c[0..31]=konverze na Int (z[0..63]) c[32..63]=konverze na Int (z[64..127]) c[64..95]=konverze na Int (z[128..191]) c[96..127]=konverze na Int (z[192..255])	Instrukce provede konverzi 4 FP hodnot s dvojitou přesností ze zdrojového operandu na 4 celočíselných hodnot datového typu Doubleword a výsledek uloží do cílového operandu.

Vex Convert Packed Single-Precision Floating-Point Values to Packed Doubleword Integers

VCVTPS2DQ c, z	Operandy: r256, r256/m256
c[0..31]=konverze na Int (z[0..31]) c[32..63]=konverze na Int (z[32..63]) ... c[224..255]=konverze na Int (z[224..255])	Instrukce provede konverzi 8 FP hodnot s jednoduchou přesností ze zdrojového operandu na 8 celočíselných hodnot datového typu Doubleword a výsledek uloží do cílového operandu.

Vex Convert Packed Double-Precision Floating-Point Values to Packed Single-Precision Floating-Point Values	
VCVTPD2PS c, z	Operandy: r128, r256/m256
c[0..31]=konverze na SP FP (z[0..63]) c[32..63]=konverze na SP FP (z[64..127]) c[64..95]=konverze na SP FP (z[128..191]) c[96..127]=konverze na SP FP (z[192..255])	Instrukce provede konverzi 4 FP hodnot s dvojitou přesností ze zdrojového operandu na 4 FP hodnoty s jednoduchou přesností a výsledek uloží do cílového operandu.

Vex Convert Packed Single-Precision Floating-Point Values to Packed Double-Precision Floating-Point Values	
VCVTPS2PD c, z	Operandy: r256, r128/m128
c[0..63]=konverze na DP FP (z[0..31]) c[64..127]=konverze na DP FP (z[32..63]) c[128..191]=konverze na DP FP (z[64..95]) c[192..255]=konverze na DP FP (z[96..127])	Instrukce provede konverzi 4 FP hodnot s jednoduchou přesností ze zdrojového operandu na 4 FP hodnoty s dvojitou přesností a výsledek uloží do cílového operandu.

Vex Divide Packed Single-Precision Floating-Point Values	
VDIVPS c, z1, z2	Operandy: r256, r256, r256/m256
c[0..31]=z1[0..31] / z2[0..31] c[32..63]=z1[32..63] / z2[32..63] ... c[224..255]=z1[224..255] / z2[224..255]	Instrukce provede vektorové delení FP hodnot s jednoduchou přesností ze zdrojových operandů a výsledek uloží do cílového operandu.

Vex Divide Packed Double-Precision Floating-Point Values	
VDIVPD c, z1, z2	Operandy: r256, r256, r256/m256
c[0..63]=z1[0..63] / z2[0..63] c[64..127]=z1[64..127] / z2[64..127] c[128..191]=z1[128..191] / z2[128..191] c[192..255]=z1[192..255] / z2[192..255]	Instrukce provede vektorové delení FP hodnot s dvojitou přesností ze zdrojových operandů a výsledek uloží do cílového operandu.

Vex Extract Packed Floating-Point Values	
VEXTRACTF128 c, z, maska	Operandy: r128/m128, r256, imm8
c[0..127]=z[0..127] nebo z[128..255]	Instrukce extrahuje FP hodnoty ze zdrojového operandu na základě masky a vybrané hodnoty uloží do cílového operandu.

Vex Round Packed Single-Precision Floating-Point Values	
VROUNDPS c, z, maska	Operandy: r256, r256/m256, imm8

c[0..31]=zaokrouhlení(z[0..31]) c[32..63]=zaokrouhlení(z[32..63]) ... c[224..255]=zaokrouhlení(z[224..255])	Instrukce zaokrouhlí FP hodnoty s jednoduchou přesností ze zdrojového operandu na základě masky na celočíselní hodnotu a výsledek uloží jako FP hodnotu s jednoduchou přesností do cílového operandu.

Vex Round Packed Double-Precision Floating-Point Values	
VROUNDPD c, z, maska	Operandy: r256, r256/m256, imm8
c[0..63]=zaokrouhlení(z[0..63]) c[64..127]=zaokrouhlení(z[64..127]) c[128..191]=zaokrouhlení(z[128..191]) c[192..255]=zaokrouhlení(z[192..255])	Instrukce zaokrouhlí FP hodnoty s dvojitou přesností ze zdrojového operandu na základě masky na celočíselní hodnotu a výsledek uloží jako FP hodnotu s dvojitou přesností do cílového operandu.

Vex Packed Single-FP Horizontal Add	
VHADDPS c, z1, z2	Operandy: r256, r256, r256/m256
c[0..31]=z1[0..31]+z1[32..63] c[32..63]=z1[64..95]+z1[96..127] ... c[95..127]=z1[224..255]+z1[224..255] c[128..159]=z2[0..31]+z2[32..63] c[160..191]=z2[64..95]+z2[96..127] ... c[224..255]=z2[224..255]+z2[224..255]	Instrukce sčítá 2 spodní hodnoty zdrojového operandu 1 a výsledek uloží do nejspodnejší hodnoty cílového operandu. Pak sčítá 2 další hodnoty zdrojového operandu 1 a výsledek uloží do druhé nejspodnejší hodnoty cílového operandu, atd. Postup je pro zdrojový operand 2 stejný, ale výsledek se ukládá do horní poloviny cílového operandu. Viz operace.

Vex Packed Double-FP Horizontal Add	
VHADDPD c, z1, z2	Operandy: r256, r256, r256/m256
c[0..63]=z1[0..63]+z1[64..127] c[64..127]=z2[0..63]+z2[64..127] c[128..191]=z1[128..191]+z1[192..255] c[192..255]=z2[128..191]+z2[192..255]	Instrukce sčítá 2 spodní hodnoty zdrojového operandu 1 a výsledek uloží do nejspodnejší hodnoty cílového operandu. Pak sčítá 2 horní hodnoty zdrojového operandu 1 a výsledek uloží do druhé nejspodnejší hodnoty cílového operandu. Postup je pro zdrojový operand 2 stejný, ale výsledek se ukládá do horní poloviny cílového operandu. Viz operace.

Vex Packed Single-FP Horizontal Substract	
VHSUBPS c, z1, z2	Operandy: r256, r256, r256/m256
c[0..31]=z1[0..31]-z1[32..63] c[32..63]=z1[64..95]-z1[96..127] ... c[95..127]=z1[224..255]-z1[224..255] c[128..159]=z2[0..31]-z2[32..63] c[160..191]=z2[64..95]-z2[96..127] ... c[224..255]=z2[224..255]-z2[224..255]	Instrukce odečítá 2 spodní hodnoty zdrojového operandu 1 a výsledek uloží do nejspodnejší hodnoty cílového operandu. Pak odečítá 2 další hodnoty zdrojového operandu 1 a výsledek uloží do druhé nejspodnejší hodnoty cílového operandu, atd. Postup je pro zdrojový operand 2 stejný, ale výsledek se ukládá do horní poloviny cílového operandu. Viz operace.

Vex Packed Double-FP Horizontal Substract	
VHSUBPD c, z1, z2	Operandy: r256, r256, r256/m256
c[0..63]=z1[0..63]-z1[[64..127] c[64..127]=z1[128..191]-z1[192..255] c[128..191]=z2[0..63]-z2[64..127] c[192..255]=z2[128..191]-z2[192..255]	Instrukce odečítá 2 spodní hodnoty zdrojového operandu 1 a výsledek uloží do nejspodnejší hodnoty cílového operandu. Pak odečítá 2 horní hodnoty zdrojového operandu 1 a výsledek uloží do druhé nejspodnejší hodnoty cílového operandu. Postup je pro zdrojový operand 2 stejný, ale výsledek se ukládá do horní poloviny cílového operandu. Viz operace.

Vex Load Unaligned Integer 256 Bits	
VLDDQU c, z	Operandy: r256, m256
c[0..255]=z[0..255]	Instrukce načítá celočíselní hodnoty ze zdrojového operandu a skopíruje je do cílového operandu.

Vex Move Aligned Packed Single-Precision Floating-Point Values	
VMOVAPS c, z	Operandy: r256, r256/m256 nebo r256/m256, r256
c[0...255]=z[0..255]	Instrukce vektorově naplní cíloví operand FP hodnotami s jednoduchou přesností ze zdrojového operandu. Vyžaduje zarovnání paměti.

Vex Move Aligned Packed Double-Precision Floating-Point Values	
VMOVAPD c, z	Operandy: r256, r256/m256 nebo r256/m256, r256
c[0...255]=z[0..255]	Instrukce vektorově naplní cíloví operand FP hodnotami s dvojitou přesností ze zdrojového operandu. Vyžaduje zarovnání paměti.

Vex Move Unaligned Packed Single-Precision Floating-Point Values	
VMOVUPS c, z	Operandy: r256, r256/m256 nebo r256/m256, r256
c[0...255]=z[0..255]	Instrukce vektorově naplní cíloví operand FP hodnotami s jednoduchou přesností ze zdrojového operandu. Nevyžaduje zarovnání paměti.

Vex Move Unaligned Packed Double-Precision Floating-Point Values	
VMOVUPD c, z	Operandy: r256, r256/m256 nebo r256/m256, r256
c[0...255]=z[0..255]	Instrukce vektorově naplní cíloví operand FP hodnotami s dvojitou přesností ze zdrojového operandu. Nevyžaduje zarovnání paměti

Vex Move Aligned Double Quadword	
VMOVDQA c, z	Operandy: r256, r256/m256 nebo r256/m256, r256

c[0...255]=z[0..255]	Instrukce naplní cíloví operand celočíselnýma hodnotama datového typu double quadword ze zdrojového operandu. Vyžaduje zarovnání.

Vex Move Unaligned Double Quadword	
VMOVDQU c, z	Operandy: r256, r256/m256 nebo r256/m256, r256
c[0...255]=z[0..255]	Instrukce naplní cíloví operand celočíselnýma hodnotama datového typu double quadword ze zdrojového operandu. Nevyžaduje zarovnání.

Vex Return Maximum Packed Single-Precision Floating-Point Values	
VMAXPS c, z1, z2	Operandy: r256, r256, r256/m256
c[0..31]=větší z hodnot z1[0..31] a z2[0..31] c[32..63]=větší z hodnot z1[32..63] a z2[32..63] ... c[224..255]=větší: z1[224..255] a z2[224..255]	Instrukce provede vektorové porovnání FP hodnot s jednoduchou přesností. Po porovnání hodnot ze zdrojových operandů instrukce zapíše do cílového operandu větší z těchto hodnot.

Vex Return Maximum Packed Double-Precision Floating-Point Values	
VMAXPD c, z1, z2	Operandy: r256, r256, r256/m256
c[0..63]=větší z hodnot z1[0..63] a z2[0..63] c[64..127]=větší: z1[64..127] a z2[64..127] c[128..191]=větší: z1[128..191] a z2[128..191] c[192..255]=větší: z1192..255] a z2[192..255]	Instrukce provede vektorové porovnání FP hodnot s dvojitou přesností. Po porovnání hodnot ze zdrojových operandů instrukce zapíše do cílového operandu větší z těchto hodnot.

Vex Return Minimum Packed Single-Precision Floating-Point Values	
VMAXPS c, z1, z2	Operandy: r256, r256, r256/m256
c[0..31]=menší z hodnot z1[0..31] a z2[0..31] c[32..63]=menší: z1[32..63] a z2[32..63] ... c[224..255]=menší: z1[224..255] z2[224..255]	Instrukce provede vektorové porovnání FP hodnot s jednoduchou přesností. Po porovnání hodnot ze zdrojových operandů instrukce zapíše do cílového operandu menší z těchto hodnot.

Vex Return Maximum Packed Double-Precision Floating-Point Values	
VMAXPD c, z1, z2	Operandy: r256, r256, r256/m256
c[0..63]=menší z hodnot z1[0..63] a z2[0..63] c[64..127]=menší: z1[64..127] a z2[64..127] c[128..191]=menší: z1[128..191] a z2[128..191] c[192..255]=menší: z1192..255] a z2[192..255]	Instrukce provede vektorové porovnání FP hodnot s dvojitou přesností. Po porovnání hodnot ze zdrojových operandů instrukce zapíše do cílového operandu menší z těchto hodnot.

Vex Move One Double-FP and Duplicate	
VMOVDDUP c, z	Operandy: r256, r256/m256

c[0..63]=z[0..63] c[64..127]=z[0..63] c[128..191]=z[128..191] c[192..255]=z[128..191]	Instrukce načítá dolní FP hodnotu s dvojitou přesností ze zdrojového operandu a skopíruje ji do spodní poloviny cílového operandu. Pak načítá dolní hodnotu s horní poloviny zdrojového operandu a uloží ji do horní poloviny cílového operandu

Vex Move Packed Single-FP High and Duplicate	
VMOVSHDUP c, z	Operandy: r256, r256/m256
c[0..31]=z[32..63] c[32..63]=z[32..63] c[64..95]=z[96..127] c[96..127]=z[96..127] ... c[192..223]=z[224..255] c[224..255]=z[224..255]	Instrukce načítá každou sudou FP hodnotu v jednoduché přesnosti ze zdrojového operandu a uloží jednu hodnotu do dvou elementů cílového operandu. Viz operace.

Vex Move Packed Single-FP High and Duplicate	
VMOVSLDUP c, z	Operandy: r256, r256/m256
c[0..31]=z[0..31] c[32..63]=z[0..31] c[64..95]=z[64..95] c[96..127]=z[64..95] ... c[192..223]=z[192..223] c[224..255]=z[192..223]	Instrukce načítá každou lichou FP hodnotu v jednoduché přesnosti ze zdrojového operandu a uloží jednu hodnotu do dvou elementů cílového operandu. Viz operace.

Vex Multiply Packed Single-Precision Floating-Point Values	
VMULPS c, z1, z2	Operandy: r256, r256, r256/m256
c[0..31]=z1[0..31] * z2[0..31] c[32..63]=z1[32..63] * z2[32..63] ... c[224..255]=z1[224..255] * z2[224..255]	Instrukce provede vektorové násobení FP hodnot s jednoduchou přesností ze zdrojových operandů a výsledek uloží do cílového operandu.

Vex Multiply Packed Double-Precision Floating-Point Values	
VMULPD c, z1, z2	Operandy: r256, r256, r256/m256
c[0..63]=z1[0..63] * z2[0..63] c[64..127]=z1[64..127] * z2[64..127] c[128..191]=z1[128..191] * z2[128..191] c[192..255]=z1[192..255] * z2[192..255]	Instrukce provede vektorové násobení FP hodnot s dvojitou přesností ze zdrojových operandů a výsledek uloží do cílového operandu.

Vex Bitwise Logical OR of Single-Precision Floating-Point Values	
VORPS c, z1, z2	Operandy: r256, r256, r256/m256

c[0..31]=z1[0..31] bitový OR z2[0..31] c[32..63]=z1[32..63] bitový OR z2[32..63] ... c[224..255]=z1[224..255] bit. OR z2[224..255]	Instrukce provede bitový vektorový logický součet FP hodnot s jednoduchou přesností ze zdrojových operandů a výsledek uloží do cílového operandu.

Vex Permute Single-Precision Floating-Point Values	
VPERMILPS c, z, maska	Operandy: r256, r256/m256, imm8
c[0..31]=jedna z hodnot (z[0..127]) c[32..63]=jedna z hodnot (z[0..127]) ... c[96..127]=jedna z hodnot (z[0..127]) c[128..159]=jedna z hodnot (z[128..255]) ... c[224..255]=jedna z hodnot (z[128..255])	Instrukce provede permutaci na základě masky. Do první poloviny cílového operandu se vybírá jedna ze 4 hodnot z dolní poloviny zdrojového operandu. Do horní poloviny cílového operandu se vybíra jedna z 4 hodnot z horní poloviny cílového operandu.

Vex Permute Double-Precision Floating-Point Values	
VPERMILPD c, z, maska	Operandy: r256, r256/m256, imm8
c[0..63]=jedna z hodnot (z[0..127]) c[64..127]=jedna z hodnot (z[0..127]) c[128..191]=jedna z hodnot (z[128..255]) c[192..255]=jedna z hodnot (z[128..255])	Instrukce provede permutaci na základě masky. Do první poloviny cílového operandu se vybírá jedna ze 2 hodnot z dolní poloviny zdrojového operandu. Do horní poloviny cílového operandu se vybíra jedna z 2 hodnot z horní poloviny cílového operandu.

Vex Permute Floating-Point Values	
VPERM2F128 c, z, maska	Operandy: r256, r256, r256/m256, imm8
c[0..127]=z1[0..127] nebo z1[128..255] nebo z2[0..127] nebo z2[128..255] c[128..255]=z1[0..127] nebo z1[128..255] nebo z2[0..127] nebo z2[128..255]	Instrukce provede permutaci na základě masky. Do dolní poloviny cílového operandu uloží buď horní nebo dolní polovinu zdrojového operandu 1 a nebo horní nebo dolní polovinu zdrojového operandu 2. Z těch samých hodnot vybíra i pro horní polovinu cílového operandu.

Vex Compute Reciprocals of Packed Single-Precision Floating-Point Values	
VRCPPS c, z	Operandy: r256, r256/m256
c[0..31]≈(1,0 / z[0..31]) c[32..63]≈(1,0 / z[32..63]) ... c[224..255]≈(1,0 / z[224..255])	Instrukce vektorově vypočítá přibližnou hodnotu po dělení hodnoty 1,0 hodnotami ze zdrojového operandu. Výsledek je uložen do cílového operandu. Pracuje s FP hodnotami s jednoduchou přesností.

Vex Compute Reciprocals of Square Roots of Packed Single-Precision Floating-Point Values	
VRSQRTPS c, z	Operandy: r256, r256/m256

c[0..31]≈(1,0 / √z[0..31]) c[32..63]≈(1,0 / √z[32..63]) ... c[224..255]≈(1,0 / √z[224..255])	Instrukce vektorově vypočítá přibližnou hodnotu po dělení hodnoty 1,0 hodnotami ze zdrojového operandu po jejich odmocnění. Výsledek je uložen do cílového operandu. Pracuje s FP hodnotami s jednoduchou přesností.

Vex Bitwise Logical XOR of Single-Precision Floating-Point Values	
VXORPS c, z1, z2	Operandy: r256, r256, r256/m256
c[0..31]=z1[0..31] bitový XOR z2[0..31] c[32..63]=z1[32..63] bitový XOR z2[32..63] ... c[224..255]=z1[224..255] bit. XOR z2[224..255]	Instrukce provede exkluzivní bitový vektorový logický součet FP hodnot s jednoduchou přesností ze zdrojových operandů a výsledek uloží do cílového operandu.

Vex Bitwise Logical XOR of Double-Precision Floating-Point Values	
VXORPD c, z1, z2	Operandy: r256, r256, r256/m256
c[0..63]=z1[0..63] bitový XOR z2[0..63] c[64..127]=z1[64..127] bitový XOR z2[64..127] c[128..191]=z1[128..191] bit. XOR z2[128..191]	Instrukce provede exkluzivní bitový vektorový logický součet FP hodnot s dvojitou přesností ze zdrojových operandů a výsledek uloží do cílového operandu.

Vex Shuffle Packed Single-Precision Floating-Point Values	
VSHUFPS c, z1, z2 maska	Operandy: r256, r256, r256/m256, imm8
c[0..31]=jedna z hodnot ve z1[0..127] c[32..63]=jedna z hodnot ve z1[0..127] c[64..95]=jedna z hodnot ve z2[0..127] c[96..127]=jedna z hodnot ve z2[0..127] c[128..159]=jedna z hodnot ve z1[128..255] c[160..191]=jedna z hodnot ve z1[128..255] c[192..224]=jedna z hodnot ve z2[128..255] c[224..255]=jedna z hodnot ve z2[128..255]	Instrukce uloží na základě masky FP hodnoty v jednoduché přesnosti ze zdrojových operandů do cílového operandu. Viz operace.

Vex Shuffle Packed Double-Precision Floating-Point Values	
VSHUFPD c, z1, z2 maska	Operandy: r256, r256, r256/m256, imm8
c[0..63]=jedna z hodnot ve z1[0..127] c[64..127]=jedna z hodnot ve z2[0..127] c[128..191]=jedna z hodnot ve z1[128..255] c[192..155]=jedna z hodnot ve z2[128..255]	Instrukce uloží na základě masky FP hodnoty v dvojité přesnosti ze zdrojových operandů do cílového operandu. Viz operace.

Vex Compute Square Roots of Packed Single-Precision Floating-Point Values	
VSQRTPS c, z	Operandy: r256, r256/m256

c[0..31]=√z[0..31] c[32..63]=√z[32..63] ... c[224..255]=√z[224..255]	Instrukce provede vektorové odmocnění FP hodnot s jednoduchou přesností ve zdrojovém operandu a výsledek uloží do cílového operandu.

Vex Compute Square Roots of Packed Double-Precision Floating-Point Values	
VSQRTPD c, z	Operandy: r256, r256/m256
c[0..63]=√z[0..63] c[64..127]=√z[64..127] c[128..191]=√z[128..191] c[192..255]=√z[192..255]	Instrukce provede vektorové odmocnění FP hodnot s dvojitou přesností ve zdrojovém operandu a výsledek uloží do cílového operandu.

Vex Store Packed Single-Precision Floating-Point Values Using Non-Temporal Hint	
VMOVNTPS c, z	Operandy: m256, r256
c=z	Instrukce překopíruje FP hodnoty s jednoduchou přesností ze zdrojového do cílového operandu. Instrukce minimalizuje zahlcení vyrovnávací paměte pomocí tzv. non-temporal hint.

Vex Store Packed Double-Precision Floating-Point Values Using Non-Temporal Hint	
VMOVNTPD c, z	Operandy: m256, r256
c=z	Instrukce překopíruje FP hodnoty s dvojitou přesností ze zdrojového do cílového operandu. Instrukce minimalizuje zahlcení vyrovnávací paměte pomocí tzv. non-temporal hint.

Vex Store Double Quadword Using Non-Temporal Hint	
VMOVNTDQ c, z	Operandy: r128/m128, r128
c=z	Instrukce překopíruje celočíselné hodnoty datového typu Quadword ze zdrojového do cílového operandu. Instrukce minimalizuje zahlcení vyrovnávací paměte pomocí tzv. non-temporal hint.

Vex Subtract Packed Single-Precision Floating-Point Values	
VSUBPS c, z1, z2	Operandy: r256, r256, r256/m256
c[0..31]=z1[0..31]-z2[0..31] c[32..63]=z1[32..63]-z2[32..63] ... c[224..255]=z1[224..255]-z2[224..255]	Instrukce provede vektorové odčítání FP hodnot s jednoduchou přesností ze zdrojového a cílového operandu a výsledek uloží do cílového operandu.

Vex Subtract Packed Double-Precision Floating-Point Values	
VSUBPD c, z1, z2	Operandy: r256, r256, r256/m256

c[0..63]=z1[0..63]-z2[0..63] c[64..127]=z1[64..127]-z2[64..127] c[128..191]=z1[128..191]-z2[128..191] c[192..255]=z1[192..255]-z2[192..255]	Instrukce provede vektorové odčítání FP hodnot s dvojitou přesností ze zdrojového a cílového operandu a výsledek uloží do cílového operandu.

Vex Unpack and Interleave High Packed Single-Precision Floating-Point Values	
VUNPCKHPS c, z1, z2	Operandy: r256, r256, r256/m256
c[0..31]=z1[64..95], c[32..63]=z2[64..95] c[64..95]=z1[96..127], c[96..127]=z2[96..127] ... c[192..223]=z1[224..255], c[224..255]=z2[224..255]	Instrukce načítá 2 horní FP hodnoty s jednoduchou přesností z horní i dolní poloviny zdrojového operandu 1, pak načítá 2 horní hodnoty z horní i dolní poloviny zdrojového operandu 2 a uloží je přeložene do cílového operandu. Viz operace.

Vex Unpack and Interleave Low Packed Single-Precision Floating-Point Values	
VUNPCKLPS c, z1, z2	Operandy: r256, r256, r256/m256
c[0..31]=z1[0..31], c[32..63]=z2[0..31] c[64..95]=z1[32..64], c[96..127]=z2[32..64] ... c[192..223]=z1[160..191], c[224..255]=z2[160..191]	Instrukce načítá 2 dolní FP hodnoty s jednoduchou přesností z horní i dolní poloviny zdrojového operandu 1, pak načítá 2 dolní hodnoty z horní i dolní poloviny zdrojového operandu 2 a uloží je přeložene do cílového operandu. Viz operace.

Vex Unpack and Interleave High Packed Double-Precision Floating-Point Values	
VUNPCKHPD c, z1, z2	Operandy: r256, r256, r256/m256
c[0..63]=z1[64..127] c[64..127]=z2[64..127] c[128..191]=z1[192..255] c[192..255]=z2[192..255]	Instrukce načítá horní FP hodnotu s dvojitou přesností z horní i dolní poloviny zdrojového operandu 1, pak načítá horní hodnotu z horní i dolní poloviny zdrojového operandu 2 a uloží je přeložene do cílového operandu. Viz operace.

Vex Unpack and Interleave Low Packed Double-Precision Floating-Point Values	
VUNPCKHPD c, z1, z2	Operandy: r256, r256, r256/m256
c[0..63]=z1[0..63] c[64..127]=z2[0..63] c[128..191]=z1[128..191] c[192..255]=z2[128..191]	Instrukce načítá horní FP hodnotu s dvojitou přesností z horní i dolní poloviny zdrojového operandu 1, pak načítá horní hodnotu z horní i dolní poloviny zdrojového operandu 2 a uloží je přeložene do cílového operandu. Viz operace.

Vex Zero Upper Bits of YMM Registers	
VZEROUPPER	Operandy: -
YMMx[128..255]=0	Instrukce vynuluje horní poloviny všech YMM registrů.

Zero All YMM Registers	
VZEROALL	Operandy: -
YMMx[0..255]=0	Instrukce vynuluje všechny YMM registry.

Instrukční sada AVX2

Vex Packed Absolute Value	
VPABSB c, z **VPABSW c, z** **VPABSD c, z**	Operandy: r256, r256/m256
c[0..255]= | z[0..255] |	Instrukce vypočítá absolutní hodnotu každého elementu zdrojového operandu a výsledek uloží do cílového operandu. Pracuje s datovýma typama Byte, Word, Doubleword.

Vex Add Packed Byte	
VPADDB c, z1, z2	Operandy: r256, r256, r256/m256
c[0..7]=z1[0..7]+z2[0..7] c[8..15]=z1[8..15]+z2[8..15] ... c[248..255]=z1[248..255]+z2[248..255]	Instrukce provede vektorový součet celočíslených hodnot datového typu Byte ze zdrojových operandů a výsledek uloží do cílového operandu.

Vex Add Packed Word	
VPADDW c, z1, z2	Operandy: r256, r256, r256/m256
c[0..15]=z1[0..15]+z2[0..15] c[16..31]=z1[16..31]+z2[16..31] ... c[240..255]=z1[240..255]+z2[240..255]	Instrukce provede vektorový součet celočíslených hodnot datového typu Word ze zdrojových operandů a výsledek uloží do cílového operandu.

Vex Add Packed Doubleword	
VPADDD c, z1, z2	Operandy: r256, r256, r256/m256
c[0..31]=z1[0..31]+z2[0..31] c[32..63]=z1[32..63]+z2[32..63] ... c[224..255]=z1[224..255]+z2[224..255]	Instrukce provede vektorový součet celočíslených hodnot datového typu Doubleword ze zdrojových operandů a výsledek uloží do cílového operandu.

Vex Add Packed Quadword	
VPADDQ c, z1, z2	Operandy: r256, r256, r256/m256
c[0..63]=z1[0..63]+z2[0..63] c[64..127]=z1[64..127]+z2[64..127] c[128..191]=z1[128..191]+z2[128..191] c[192..255]=z1[192..255]+z2[192..255]	Instrukce provede vektorový součet celočíslených hodnot datového typu Quadword ze zdrojových operandů a výsledek uloží do cílového operandu.

Vex Add Packed Signed Byte with Signed Saturation	
VPADDSB c, z1, z2	Operandy: r256, r256, r256/m256

c[0..7]=Saturace(z1[0..7]+z2[0..7]) c[8..15]=Saturace(z1[8..15]+z2[8..15]) ... c[248..255]=Sat.(z1[248..255]+z2[248..255])	Instrukce provede vektorový součet se saturací celočíslených hodnot datového typu Byte se znamánkem ze zdrojových operandů a výsledek uloží do cílového operandu.

Vex Add Packed Signed Word with Signed Saturation	
VPADDSW c, z1, z2	Operandy: r256, r256, r256/m256
c[0..15]=Saturace(z1[0..15]+z[0..15]) c[16..31]=Saturace(z1[16..31]+z[16..31]) ... c[240..255]=Sat.(z1[240..255]+z[240..255])	Instrukce provede vektorový součet se saturací celočíslených hodnot datového typu Word se znamánkem ze zdrojových operandů a výsledek uloží do cílového operandu.

Vex Add Packed Signed Word with Signed Saturation	
PADDSW c, z1, z2	Operandy: r256, r256, r256/m256
c[0..15]=Saturace(z1[0..15]+z[0..15]) c[16..31]=Saturace(z1[16..31]+z[16..31]) ... c[240..255]=Satur.(z1[240..255]+z[240..255])	Instrukce provede vektorový součet se saturací celočíslených hodnot datového typu Word se znamánkem ze zdrojových operandů a výsledek uloží do cílového operandu.

Vex Add Packed Unsigned Byte with Unsigned Saturation	
PADDUSB c, z1, z2	Operandy: r256, r256, r256/m256
c[0..7]=Saturace(z1[0..7]+z2[0..7]) c[8..15]=Saturace(z1[8..15]+z2[8..15]) ... c[248..255]=Sat.(z1[248..255]+z2[248..255])	Instrukce provede vektorový součet se saturací celočíslených hodnot datového typu Byte bez znaménka ze zdrojových operandů a výsledek uloží do cílového operandu.

Vex Packed Logical AND	
VPAND c, z1, z2	Operandy: r256, r256, r256/m256
c[0..255]=z1[0..255] bitový AND z2[0..255]	Instrukce provede bitový vektorový logický součin FP hodnot s dvojitou přesnostíze zdrojových operandů a výsledek uloží do cílového operandu.

Vex Packed Logical AND NOT	
VPANDN c, z1, z2	Operandy: r256, r256, r256/m256
c[0..255]=z1[0..255] bit. AND NOT z2[0..255]	Instrukce provede bitový vektorový negovaný logický součin FP hodnot s dvojitou přesností ze zdrojových operandů a výsledek uloží do cílového operandu.

Vex Average Packed Word	
PAVGW c, z1, z2	Operandy: r256, r256, r256/m256
	Instrukce provede vektorový průměr celočíselných hodnot datového typu Word ze zdrojových operandů a zaokrouhlený výsledek uloží do cílového operandu.

Vex Average Packed Byte	
PAVGB c, z1, z2	Operandy: r256, r256, r256/m256
	Instrukce provede vektorový průměr celočíselných hodnot datového typu Byte ze zdrojových operandů a zaokrouhlený výsledek uloží do cílového operandu.

Vex Blend Packed Words	
VPBLENDW c, z1, z2, maska	Operandy: r256, r256, r256/m256, imm8
c[0..15]=z1[0..15] nebo z2[0..15] c[16..31]=z1[16..31] nebo z2[16..31] ... c[240..255]=z1[240..255] nebo z2[240..255]	Instrukce pomíchá na základě masky hodnoty datového typu Word ze zdrojových operandů a vybranou hodnotu uloží do cílového operandu.

Vex Blend Packed Doublewords	
VPBLENDD c, z1, z2, maska	Operandy: r256, r256, r256/m256, imm8
c[0..31]=z1[0..31] nebo z2[0..31] c[32..63]=z1[32..63] nebo z2[32..63] ... c[224..255]=z1[224..255] nebo z2[224..255]	Instrukce pomíchá na základě masky hodnoty datového typu Doubleword ze zdrojových operandů a vybranou hodnotu uloží do cílového operandu.

Vex Broadcast Integer Data	
VPBROADCASTB c, z **VPBROADCASTD c, z** **VPBROADCASTQ c, z**	Operandy: r128/r256, r128/m8-m128
c[0..255]=z	Instrukce nahrá 1 spodní celočíselní hodnotu datového typu Byte/Doubleword/Quadword ze zdrojového operandu do všech hodnot cílového operandu.

Vex Broadcast Double-Precision FP Value	
VPBROADCASTSD c, z	Operandy: r256, r128/m64-m128
c[0..255]=z	Instrukce nahrá 1 spodní FP hodnotu v dvojité přesnosti ze zdrojového operandu do všech hodnot cílového operandu.

Vex Broadcast Integer	
VPBROADCASTI128 c, z	Operandy: r256, r128/m128
c[0..127]=z c[0..255]=z	Instrukce nahrá 1 spodní celočíselní hodnotu ze zdrojového operandu obou 128bitových částí cílového operandu.

Vex Broadcast Single-Precision FP Value	
VPBROADCASTSS c, z	Operandy: r256, r128/m32-m128

c[0..255]=z	Instrukce nahrá 1 spodní FP hodnotu v jednoduché přesnosti ze zdrojového operandu do všech hodnot cílového operandu.

Vex Broadcast Word	
VPBROADCASTW c, z	Operandy: r128/r256, r128/m16
c[0..255]=z	Instrukce nahrá 1 spodní celočíselní hodnotu datového typu Word ze zdrojového operandu do všech hodnot cílového operandu.

Vex Shift Double Quadword Left Logical	
VPSLLDQ c, maska	Operandy: r256, r256, imm8
c=posun vlevo na základě masky (z)	Instrukce provede logický posun cílového operandu vlevo na základě masky. Pokud je hodnota masky větší než 15, c je vynulován.

Vex Shift Double Quadword Right Logical	
VPSRLDQ c, maska	Operandy: r256, r256, imm8
c=posun vpravo na základě masky (z)	Instrukce provede logický posun cílového operandu vpravo na základě masky. Pokud je hodnota masky větší než 15, c je vynulován.

Vex Compare Packed Data for Equal	
VPCMPEQB c, z1, z2 **VPCMPEQW c, z1, z2** **VPCMPEQD c, z1, z2**	Operandy: r256, r256, r256/m256
pokud (z1[x..y] == z2[x..y]) c[x..y]=1 jinak c[x..y]=0	Instrukce provede porovnání celočíslených hodnot datového typu Byte/Word/Doubleword pro rovnost. Pokud se hodnoty liší, cílová hodnota je nastavena na 0. Pokus jsou hodnoty rovny, cílová hodnota je nastavena na 1.

Vex Compare Packed Qword Data for Equal	
VPCMPEQQ c, z1, z2	Operandy: r256, r256, r256/m256
pokud (z1[x..y] == z2[x..y]) c[x..y]=1 jinak c[x..y]=0	Instrukce provede porovnání celočíslených hodnot datového typu Quadword pro rovnost. Pokud se hodnoty liší, cílová hodnota je nastavena na 0. Pokus jsou hodnoty rovny, cílová hodnota je nastavena na 1.

Vex Compare Packed Data for Greater Than	
VPCMPGTB c, z1, z2 **VPCMPGTW c, z1, z2** **VPCMPGTD c, z1, z2**	Operandy: r256, r256, r256/m256

pokud (z1[x..y] > z2[x..y]) c[x..y]=1 jinak c[x..y]=0	Instrukce provede porovnání celočíslených hodnot datového typu Byte/Word/Doubleword pro rovnost. Pokud je hodnota zdrojového operandu 1 větší než hodnota zdrojového operandu 2, pak je cílová hodnota nastavena na 1. Jinak je cílová hodnota nastavena na 0.

Vex Compare Packed Qword Data for Greater Than	
VPCMPGTQ c, z1, z2	Operandy: r256, r256, r256/m256
pokud (z1[x..y] > z2[x..y]) c[x..y]=1 jinak c[x..y]=0	Instrukce provede porovnání celočíslených hodnot datového typu Quadword pro rovnost. Pokud je hodnota zdrojového operandu 1 větší než hodnota zdrojového operandu 2, pak je cílová hodnota nastavena na 1. Jinak je cílová hodnota nastavena na 0.

Vex Extract Packed Integer Values	
VEXTRACTFI128 c, z, maska	Operandy: r128/m128, r256, imm8
c[0..127]=z[0..127] nebo z[128..255]	Instrukce extrahuje celočíselní hodnoty ze zdrojového operandu na základě masky a vybrané hodnoty uloží do cílového operandu.

Vex Packed Horizontal Add	
VPHADDW c, z1, z2 **VPHADDD c, z1, z2**	Operandy: r256, r256, r256/m256
Příklad pro datový typ Doubleword: c[0..31]=z1[0..31]+z2[32..63] c[32..63]=z1[64..95]+z2[96..127] ... c[224..255]=z1[224..255]+z2[224..255]	Chování instrukce je zhodné s instrukcí PHADDD, ovšem tahle pracuje s dvounásobně větším objemem dat. Viz operace.

Vex Packed Horizontal Substract	
VPHSUBW c, z1, z2 **VPHSUBD c, z1, z2**	Operandy: r256, r256, r256/m256
Příklad pro datový typ Doubleword: c[0..31]=z1[0..31]-z2[32..63] c[32..63]=z1[64..95]-z2[96..127] ... c[224..255]=z1[224..255]-z2[224..255]	Chování instrukce je zhodné s instrukcí PHSUBD, ovšem tahle pracuje s dvounásobně větším objemem dat. Viz operace.

Vex Gather Packed Values Using Signed Indices	
VPGATHERDD c, vm, z **VPGATHERDQ c, vm, z** **VPGATHERQD c, vm, z** **VPGATHERQQ c, vm, z** **VPGATHERPS c, vm, z** **VPGATHERPD c, vm, z**	Operandy: r128/r256, vm32x/vm32y/vm64x/vm64y, r128/r256

	Instrukce má složité chování, viz [1], [2].

Vex Multiply and Add Packed Integers	
VPMADDWD c, z1, z2	Operandy: r256, r256, r256/m256
c[0..31]=(z1[0..15]*z2[0..15])+ +(z1[16..31]*z2[16..31]) ... c[224..255]=(z1[224..239]*z2[224..239])+ +(z1[240..255]*z2[240..255])	Instrukce provede vektorový součin dolních 16 bitů ze zdrojových operandů. Následně provede vektorový součin druhých 16 bitů ze zdrojových operandů. Následně tyto mezivýsledky sečte a výsledek uloží do dolních 32 bitů cílového operandu. Instrukce pracuje s celýma číslama a datovým typem Doubleword a Word.

Vex Maximum of Packed Signed Byte Integers	
VPMAXSB c, z1, z2	Operandy: r256, r256, r256/m256
c[0..7]=větší z hodnot z1[0..7] a z2[0..7] c[8..15]=větší z hodnot z1[8..15] a z2[8..15] ... c[248..255]=větší: z1[248..255] a z2[248..255]	Instrukce provede vektorové porovnání celočíslených hodnot datového typu Byte se znaménkem (signed). Po porovnání hodnot ze zdrojového a cílového operandu instrukce zapíše do cílového operandu větší z těchto hodnot.

Vex Maximum of Packed Signed Word Integers	
VPMAXSW c, z1, z2	Operandy: r256, r256, r256/m256
c[0..15]=větší z hodnot z1[0..15] a z2[0..15] c[16..31]=větší z hodnot z1[16..31] a z2[16..31] ... c[240..255]=větší: z1[240..255] a z2[240..255]	Instrukce provede vektorové porovnání celočíslených hodnot datového typu Word se znaménkem (signed). Po porovnání hodnot ze zdrojových operandů instrukce zapíše do cílového operandu větší z těchto hodnot.

Vex Maximum of Packed Signed Dword Integers	
VPMAXSD c, z1, z2	Operandy: r256, r256, r256/m256
c[0..31]=větší z hodnot z1[0..31] a z2[0..31] c[32..63]=větší z hodnot z1[32..63] a z2[32..63] ... c[224..255]=větší: z1[224..255] a z2[224..255]	Instrukce provede vektorové porovnání celočíslených hodnot datového typu Doubleword se znaménkem (signed). Po porovnání hodnot ze zdrojových operandů instrukce zapíše do cílového operandu větší z těchto hodnot.

Vex Maximum of Packed Unsigned Byte Integers	
VPMAXUB c, z1, z2	Operandy: r256, r256, r256/m256
c[0..7]=větší z hodnot z1[0..7] a z2[0..7] c[8..15]=větší z hodnot z1[8..15] a z2[8..15] ... c[248..255]=větší: z1[248..255] a z2[248..255]	Instrukce provede vektorové porovnání celočíslených hodnot datového typu Byte bez znaménka (unsigned). Po porovnání hodnot ze zdrojového a cílového operandu instrukce zapíše do cílového operandu větší z těchto hodnot.

Vex Maximum of Packed Unsigned Word Integers	
VPMAXUW c, z1, z2	Operandy: r256, r256, r256/m256
c[0..15]=větší z hodnot z1[0..15] a z2[0..15] c[16..31]=větší z hodnot z1[16..31] a z2[16..31] ... c[240..255]=větší: z1[240..255] a z2[240..255]	Instrukce provede vektorové porovnání celočíslených hodnot datového typu Word bez znaménka (unsigned). Po porovnání hodnot ze zdrojových operandů instrukce zapíše do cílového operandu větší z těchto hodnot.

Vex Maximum of Packed Unsigned Dword Integers	
VPMAXUD c, z1, z2	Operandy: r256, r256, r256/m256
c[0..31]=větší z hodnot z1[0..31] a z2[0..31] c[32..63]=větší z hodnot z1[32..63] a z2[32..63] ... c[224..255]=větší: z1[224..255] a z2[224..255]	Instrukce provede vektorové porovnání celočíslených hodnot datového typu Doubleword bez znaménka (unsigned). Po porovnání hodnot ze zdrojových operandů instrukce zapíše do cílového operandu větší z těchto hodnot.

Vex Minimum of Packed Signed Byte Integers	
VPMINSB c, z1, z2	Operandy: r256, r256, r256/m256
c[0..7]=menší z hodnot z1[0..7] a z2[0..7] c[8..15]=menší z hodnot z1[8..15] a z2[8..15] ... c[248..255]=menší: z1[248..255] a z2[248..255]	Instrukce provede vektorové porovnání celočíslených hodnot datového typu Byte se znaménkem (signed). Po porovnání hodnot ze zdrojového a cílového operandu instrukce zapíše do cílového operandu menší z těchto hodnot.

Vex Minimum of Packed Signed Word Integers	
VPMINSW c, z1, z2	Operandy: r256, r256, r256/m256
c[0..15]=větší z hodnot z1[0..15] a z2[0..15] c[16..31]=větší z hodnot z1[16..31] a z2[16..31] ... c[240..255]=větší: z1[240..255] a z2[240..255]	Instrukce provede vektorové porovnání celočíslených hodnot datového typu Word se znaménkem (signed). Po porovnání hodnot ze zdrojových operandů instrukce zapíše do cílového operandu menší z těchto hodnot.

Vex Minimum of Packed Signed Dword Integers	
VPMINSD c, z1, z2	Operandy: r256, r256, r256/m256
c[0..31]=menší z hodnot z1[0..31] a z2[0..31] c[32..63]=menší z hod. z1[32..63] a z2[32..63] ... c[224..255]=menší: z1[224..255] a z2[224..255]	Instrukce provede vektorové porovnání celočíslených hodnot datového typu Doubleword se znaménkem (signed). Po porovnání hodnot ze zdrojových operandů instrukce zapíše do cílového operandu menší z těchto hodnot.

Vex Minimum of Packed Unsigned Byte Integers	
VPMINUB c, z1, z2	Operandy: r256, r256, r256/m256

c[0..7]=menší z hodnot z1[0..7] a z2[0..7] c[8..15]=menší z hodnot z1[8..15] a z2[8..15] ... c[248..255]=menší: z1[248..255] a z2[248..255]	Instrukce provede vektorové porovnání celočíslených hodnot datového typu Byte bez znaménka (unsigned). Po porovnání hodnot ze zdrojového a cílového operandu instrukce zapíše do cílového operandu menší z těchto hodnot.

Vex Minimum of Packed Unsigned Word Integers	
VPMINUW c, z1, z2	Operandy: r256, r256, r256/m256
c[0..15]=větší z hodnot z1[0..15] a z2[0..15] c[16..31]=větší z hodnot z1[16..31] a z2[16..31] ... c[240..255]=větší: z1[240..255] a z2[240..255]	Instrukce provede vektorové porovnání celočíslených hodnot datového typu Word bez znaménka (unsigned). Po porovnání hodnot ze zdrojových operandů instrukce zapíše do cílového operandu menší z těchto hodnot.

Vex Minimum of Packed Unsigned Dword Integers	
VPMINUD c, z1, z2	Operandy: r256, r256, r256/m256
c[0..31]=menší z hodnot z1[0..31] a z2[0..31] c[32..63]=menší z hod. z1[32..63] a z2[32..63] ... c[224..255]=menší: z1[224..255] a z2[224..255]	Instrukce provede vektorové porovnání celočíslených hodnot datového typu Doubleword bez znaménka (unsigned). Po porovnání hodnot ze zdrojových operandů instrukce zapíše do cílového operandu menší z těchto hodnot.

Vex Multiply Packed Signed Doubleword Integers	
VPMULDQ c, z1, z2	Operandy: r256, r256, r256/m256
c[0..63]=z1[0..31]*z2[0..31] c[64..127]=z1[64..95]*z2[64..95] c[128..191]=z1[128..159]*z2[128..159] c[192..255]=z1[192..223]*z2[192..223]	Instrukce provede vektorový součin celočíselnéch hodnot datového typu Doubleword ze zdrojových operandů. Výsledkem je datový typ Quadword a tento výsledek je uložen v cílovém operandu.

Vex Multiply Packed Unsigned Doubleword Integers	
VPMULUDQ c, z1, z2	Operandy: r256, r256, r256/m256
c[0..63]=z1[0..31]*z2[0..31] c[64..127]=z1[64..95]*z2[64..95] c[128..191]=z1[128..159]*z2[128..159] c[192..255]=z1[192..223]*z2[192..223]	Instrukce provede vektorový součin celočíselnéch hodnot datového typu Doubleword ze zdrojových operandů. Výsledkem je datový typ Quadword a tento výsledek je uložen v cílovém operandu. Pracuje bez znaménka (unsigned).

Multiply Packed Integers and Store High Result	
VPMULHW c, z1, z2 **VPMULHD c, z1, z2**	Operandy: r256, r256, r256/m256

MV=mezivýsledek: MV1[0..31]=z1[0..15]*z2[0..15] ... MV8[0..31]=z1[112..127]*z2[112..127] ---------- c[0..15]=MV1[16..31] ... c[240..255]=MV8[16..31]	Instrukce vektorově vynásobí celočíselné hodnoty datového typu Doubleword/Word ze zdrojových operandů. Do cílového operandu je uloženo horních 16 bitů mezivýsledku.

Multiply Packed Integers and Store Low Result	
VPMULLW c, z1, z2 **VPMULLD c, z1, z2**	Operandy: r256, r256, r256/m256
MV=mezivýsledek: MV1[0..31]=z1[0..15]*z2[0..15] ... MV8[0..31]=z1[112..127]*z2[112..127] ---------- c[0..15]=MV1[0..15] ... c[240..255]=MV8[0..15]	Instrukce vektorově vynásobí celočíselné hodnoty datového typu Word/Doubleword ze zdrojových operandů. Do cílového operandu je uloženo dolních 16 bitů mezivýsledku.

Vex Permute Single-Precision Floating-Point Elements	
VPERMPS c, z1, z2	Operandy: r256, r256, r256/m256
Viz [1], [2].	Instrukce permutace používá indexovou hodnotu každých 32 bitů zdrojového operandu 1 k vybrání hodnoty ze zdrojového operandu 2. Vybraná hodnota je uložena v cílovém operandu na pozici vybrané indexovou hodnotou.

Vex Permute Double-Precision Floating-Point Elements	
VPERMPD c, z1, z2	Operandy: r256, r256, r256/m256
Viz [1], [2].	Instrukce permutace používá dvoubitovou indexovou hodnotu zdrojového operandu 1 k vybrání hodnoty ze zdrojového operandu 2. Vybraná hodnota je uložena v cílovém operandu na pozici vybrané indexovou hodnotou.

Vex Permute Integer Elements	
VPERMQ/VPERMD c, z1, z2	Operandy: r256, r256, r256/m256
Viz [1], [2].	Instrukce permutace používá dvoubitovou indexovou hodnotu zdrojového operandu 1 k vybrání hodnoty ze zdrojového operandu 2. Vybraná hodnota je uložena v cílovém operandu na pozici vybrané indexovou hodnotou.

Vex Shuffle Packed Words	
VPSHUFW c, z, maska	Operandy: r256, r256/m256
c[0..15], [16..31]...[240..255]=na základě masky jedna hodnota z možností: z[0..15], [16..31]...[240..255] atd.	Instrukce na základě masky vybere jednu hodnotu ze zdrojového operandu a výsledek uloží do dané části cílového operandu.

Vex Shuffle Packed Bytes	
VPSHUFW c, z, maska	Operandy: r256, r256/m256
c[0..7], [7..15]...[248..255]=na základě masky jedna hodnota z možností: z[0..7], [7..15]...[248..255] atd.	Instrukce na základě masky vybere jednu hodnotu ze zdrojového operandu a výsledek uloží do dané části cílového operandu.

Vex Packed SIGN	
VPSIGNB c, z1, z2 **VPSIGNW c, z1, z2** **VPSIGND c, z1, z2**	Operandy: r256, r256, r256/m256
c[0..255]= negace/ponechání/vynulování z2[0..255] na základě hodnot v z1[0..255]	Ak je hodnota zdrojového operandu 2 <0, pak je hodnota zdrojového operandu 1 znegovaná. Ak je hodnota zdrojového operandu 2>0, pak je hodnota zdrojového operandu 1 nezměnena. Ak je hodnota zdrojového operandu 2=0, pak je hodnota zdrojového operandu 1 vynulovaná. Výsledek je uložen do cílového operandu. Pracuje s datovýma typama Byte, Word, Doubleword.

Vex Subtract Packed Integers	
VPSUBB c, z1, z2 **VPSUBW c, z1, z2** **VPSUBD c, z1, z2**	Operandy: r256, r256, r256/m256
c[x..y]=c[x..y]-z[x..y]	Instrukce provede vektorové odčítání celočíselných hodnot datových tupů Byte/Word/Doubleword ze zdrojových operandů a výsledek uloží do cílového operandu.

Vex Unpack High Data	
VPUNPCKHBW c, z1, z2 **VPUNPCKHWD c, z1, z2** **VPUNPCKHDQ c, z1, z2** **VPUNPCKHQDQ c, z1, z2**	Operandy: r256, r256, r256/m256
Př. - Doubleword: c[0..31]=z1[64..95] c[32..63]=z2[64..95] c[64..95]=z1[96..127] c[96..127]=z2[96..127]	Instrukce načítá dvě horní celočíselní hodnoty datových tupů Byte/Word/Doubleword/Quadword ze zdrojových operandů a uloží je přeloženě do cílového operandu. Viz operace.

Vex Unpack Low Data	
VPUNPCKLBW c, z1, z2 **VPUNPCKLWD c, z1, z2** **VPUNPCKLDQ c, z1, z2** **VPUNPCKLQDQ c, z1, z2**	Operandy: r256, r256, r256/m256
Př. - Doubleword: c[0..31]=z1[0..31] c[32..63]=z2[0..31] c[64..95]=z1[32..63] c[96..127]=z2[32..63] atd.	Instrukce načítá dvě dolní celočíselní hodnoty datových tupů Byte/Word/Doubleword/Quadword ze zdrojových operandů a uloží je přeloženě do cílového operandu. Viz operace.

Vex Logical Exclusive OR	
VPXOR c, z1, z2	Operandy: r256, r256, r256/m256
c[0..255]=z1[0..255] bitový XOR z2[0..255]	Instrukce provede exkluzivní bitový vektorový logický součet hodnot ze zdrojových operandů a výsledek uloží do cílového operandu.

Instrukční sada FMA3

Fused Multiply-Add of Packed Single-Precision Floating-Point Values	
VFMADD132PS c, z1, z2 **VFMADD213PS c, z1, z2** **VFMADD231PS c, z1, z2**	Operandy: r128/r256, r128/r256, r128/m128/r256/m256
Příklad pro 231: c[0..31]=c[0..31]+(z1[0..31]*z2[0..31]) c[32..63]=c[32..63]+(z1[32..63]*z2[32..63]) ... c[224..255]=c[224..255]+(z1[224..255]* *z2[224....255])	Instrukce provede nejprve vektorové násobení hodnot dvou operandů a pak výsledek přičítá k třetímu operandu. Pořadí operandů je dáno samotnou instrukcí (132, 213, 231). Celkový výsledek je uložen do cílového operandu. Pracuje s jednoduchou přesností.

Fused Multiply-Add of Packed Double-Precision Floating-Point Values	
VFMADD132PD c, z1, z2 **VFMADD213PD c, z1, z2** **VFMADD231PD c, z1, z2**	Operandy: r128/r256, r128/r256, r128/m128/r256/m256
Příklad pro 231: c[0..63]=c[0..63]+(z1[0..63]*z2[0..63]) c[64..127]=c[64..127]+(z1[64..127]*z2[64..127]) ... c[192..255]=c[192..255]+(z1[192..255]* *z2[192....255])	Instrukce provede nejprve vektorové násobení hodnot dvou operandů a pak výsledek přičítá k třetímu operandu. Pořadí operandů je dáno samotnou instrukcí (132, 213, 231). Celkový výsledek je uložen do cílového operandu. Pracuje s dvojitou přesností.

Fused Multiply-Add of Scalar Single-Precision Floating-Point Values	
VFMADD132SS c, z1, z2 **VFMADD213SS c, z1, z2** **VFMADD231SS c, z1, z2**	Operandy: r128, r128, r128/m128
Příklad pro 231: c[0..31]=c[0..31]+(z1[0..31]*z2[0..31]) c[32..127] zůstane nezměnen	Instrukce provede nejprve skalární násobení dolních hodnot dvou operandů a pak výsledek přičítá k třetímu operandu. Pořadí operandů je dáno samotnou instrukcí (132, 213, 231). Celkový výsledek je uložen do cílového operandu. Pracuje s jednoduchou přesností.

Fused Multiply-Add of Scalar Double-Precision Floating-Point Values	
VFMADD132SD c, z1, z2 **VFMADD213SD c, z1, z2** **VFMADD231SD c, z1, z2**	Operandy: r128, r128, r128/m128
Příklad pro 231: c[0..63]=c[0..63]+(z1[0..63]*z2[0..63]) c[64..127] zůstane nezměnen	Instrukce provede nejprve skalární násobení dolních hodnot dvou operandů a pak výsledek přičítá k třetímu operandu. Pořadí operandů je dáno samotnou instrukcí (132, 213, 231). Celkový výsledek je uložen do cílového operandu. Pracuje s dvojitou přesností.

Fused Multiply-Alternating Add/Subtract of Packed Single-Precision Floating-Point Values	
VFMADDSUB132PS c, z1, z2 **VFMADDSUB213PS c, z1, z2** **VFMADDSUB231PS c, z1, z2**	Operandy: r128/r256, r128/r256, r128/m128/r256/m256
Příklad pro 231: c[0..31]=c[0..31]+(z1[0..31]*z2[0..31]) c[32..63]=c[32..63]-(z1[32..63]*z2[32..63]) ... c[224..255]=c[224..255]- -(z1[224..255]*z2[224..255])	Instrukce provede nejprve vektorové násobení hodnot dvou operandů a pak výsledek přičítá (liché prvky), resp. odečítá (sudé prvky) k třetímu operandu. Pořadí operandů je dáno samotnou instrukcí (132, 213, 231). Celkový výsledek je uložen do cílového operandu. Pracuje s jednoduchou přesností.

Fused Multiply-Alternating Add/Subtract of Packed Double-Precision Floating-Point Values	
VFMADDSUB132PD c, z1, z2 **VFMADDSUB213PD c, z1, z2** **VFMADDSUB231PD c, z1, z2**	Operandy: r128/r256, r128/r256, r128/m128/r256/m256
Příklad pro 231: c[0..63]=c[0..63]+(z1[0..63]*z2[0..63]) c[64..127]=c[64..127]- (z1[64..127]*z2[64..127]) ... c[191..255]=c[191..255]- -(z1[191..255]*z2[191..255])	Instrukce provede nejprve vektorové násobení hodnot dvou operandů a pak výsledek přičítá (liché prvky), resp. odečítá (sudé prvky) k třetímu operandu. Pořadí operandů je dáno samotnou instrukcí (132, 213, 231). Celkový výsledek je uložen do cílového operandu. Pracuje s dvojitou přesností.

Fused Multiply-Subtract of Packed Single-Precision Floating-Point Values	
VFMSUB132PS c, z1, z2 **VFMSUB213PS c, z1, z2** **VFMSUB231PS c, z1, z2**	Operandy: r128/r256, r128/r256, r128/m128/r256/m256
Příklad pro 231: c[0..31]=c[0..31]-(z1[0..31]*z2[0..31]) c[32..63]=c[32..63]-(z1[32..63]*z2[32..63]) ... c[224..255]=c[224..255]- (z1[224..255]*z2[224....255])	Instrukce provede nejprve vektorové násobení hodnot dvou operandů a pak výsledek odečítá od třetího operandu. Pořadí operandů je dáno samotnou instrukcí (132, 213, 231). Celkový výsledek je uložen do cílového operandu. Pracuje s jednoduchou přesností.

Fused Multiply-Substract of Packed Double-Precision Floating-Point Values	
VFMSUB132PD c, z1, z2 **VFMSUB213PD c, z1, z2** **VFMSUB231PD c, z1, z2**	Operandy: r128/r256, r128/r256, r128/m128/r256/m256
Příklad pro 231: c[0..63]=c[0..63]-(z1[0..63]*z2[0..63]) c[64..127]=c[64..127]- (z1[64..127]*z2[64..127]) ... c[192..255]=c[192..255]- -(z1[192..255]*z2[192....255])	Instrukce provede nejprve vektorové násobení hodnot dvou operandů a pak výsledek odečítá od třetího operandu. Pořadí operandů je dáno samotnou instrukcí (132, 213, 231). Celkový výsledek je uložen do cílového operandu. Pracuje s dvojitou přesností.

Fused Multiply-Substract of Scalar Single-Precision Floating-Point Values	
VFMSUB132SS c, z1, z2 **VFMSUB213SS c, z1, z2** **VFMSUB231SS c, z1, z2**	Operandy: r128, r128, r128/m128
Příklad pro 231: c[0..31]=c[0..31]-(z1[0..31]*z2[0..31]) c[32..127] zůstane nezměnen	Instrukce provede nejprve skalární násobení dolních hodnot dvou operandů a pak výsledek od třetího operandu. Pořadí operandů je dáno samotnou instrukcí (132, 213, 231). Celkový výsledek je uložen do cílového operandu. Pracuje s jednoduchou přesností.

Fused Multiply-Substract of Scalar Double-Precision Floating-Point Values	
VFMADD132SD c, z1, z2 **VFMADD213SD c, z1, z2** **VFMADD231SD c, z1, z2**	Operandy: r128, r128, r128/m128
Příklad pro 231: c[0..63]=c[0..63]-(z1[0..63]*z2[0..63]) c[64..127] zůstane nezměnen	Instrukce provede nejprve skalární násobení dolních hodnot dvou operandů a pak výsledek od třetího operandu. Pořadí operandů je dáno samotnou instrukcí (132, 213, 231). Celkový výsledek je uložen do cílového operandu. Pracuje s dvojitou přesností.

Fused Multiply-Alternating Add/Substract of Packed Single-Precision Floating-Point Values	
VFMSUBADD132PS c, z1, z2 **VFMSUBADD213PS c, z1, z2** **VFMSUBADD231PS c, z1, z2**	Operandy: r128/r256, r128/r256, r128/m128/r256/m256
Příklad pro 231: c[0..31]=c[0..31]-(z1[0..31]*z2[0..31]) c[32..63]=c[32..63]+(z1[32..63]*z2[32..63]) ... c[224..255]=c[224..255]+ +(z1[224..255]*z2[224..255])	Instrukce provede nejprve vektorové násobení hodnot dvou operandů a pak výsledek přičítá (sudé prvky), resp. odečítá (liché prvky) k třetímu operandu. Pořadí operandů je dáno samotnou instrukcí (132, 213, 231). Celkový výsledek je uložen do cílového operandu. Pracuje s jednoduchou přesností.

Fused Multiply-Alternating Add/Substract of Packed Double-Precision Floating-Point Values	
VFMSUBADD132PD c, z1, z2 **VFMSUBADD213PD c, z1, z2** **VFMSUBADD231PD c, z1, z2**	Operandy: r128/r256, r128/r256, r128/m128/r256/m256
Příklad pro 231: c[0..63]=c[0..63]-(z1[0..63]*z2[0..63]) c[64..127]=c[64..127]+(z1[64..127]*z2[64..127]) ... c[191..255]=c[191..255]+ +(z1[191..255]*z2[191..255])	Instrukce provede nejprve vektorové násobení hodnot dvou operandů a pak výsledek přičítá (sudé prvky), resp. odečítá (liché prvky) k třetímu operandu. Pořadí operandů je dáno samotnou instrukcí (132, 213, 231). Celkový výsledek je uložen do cílového operandu. Pracuje s dvojitou přesností.

Fused Negative Multiply-Add of Packed Single-Precision Floating-Point Values	
VFNMADD132PS c, z1, z2 **VFNMADD213PS c, z1, z2** **VFNMADD231PS c, z1, z2**	Operandy: r128/r256, r128/r256, r128/m128/r256/m256
Příklad pro 231: c[0..31]=c[0..31]+(negace(z1[0..31]*z2[0..31])) c[32..63]=c[32..63]+(negace(z1[32..63]*z2[32...63])) ... c[224..255]=c[224..255]+(negace(z1[224..255]*z2[224....255]))	Instrukce provede nejprve vektorové násobení hodnot dvou operandů a pak znegovaný výsledek přičítá k třetímu operandu. Pořadí operandů je dáno samotnou instrukcí (132, 213, 231). Celkový výsledek je uložen do cílového operandu. Pracuje s jednoduchou přesností.

Fused Negative Multiply-Add of Packed Double-Precision Floating-Point Values	
VFNMADD132PD c, z1, z2 **VFNMADD213PD c, z1, z2** **VFNMADD231PD c, z1, z2**	Operandy: r128/r256, r128/r256, r128/m128/r256/m256

Příklad pro 231: c[0..63]=c[0..63]+(negace(z1[0..63]*z2[0..63])) c[64..127]=c[64..127]+(negace(z1[64..127]* *z2[64..127])) ... c[192..255]=c[192..255]+(negace(z1[192..255] *z2[192..255]))	Instrukce provede nejprve vektorové násobení hodnot dvou operandů a pak znegovaný výsledek přičítá k třetímu operandu. Pořadí operandů je dáno samotnou instrukcí (132, 213, 231). Celkový výsledek je uložen do cílového operandu. Pracuje s dvojitou přesností.

Fused Negative Multiply-Add of Scalar Single-Precision Floating-Point Values	
VFNMADD132SS c, z1, z2 **VFNMADD213SS c, z1, z2** **VFNMADD231SS c, z1, z2**	Operandy: r128, r128, r128/m128
Příklad pro 231: c[0..31]=c[0..31]+(negace(z1[0..31]*z2[0..31]) c[32..127] zůstane nezměnen	Instrukce provede nejprve skalární násobení dolních hodnot dvou operandů a pak znegovaný výsledek přičítá k třetímu operandu. Pořadí operandů je dáno samotnou instrukcí (132, 213, 231). Celkový výsledek je uložen do cílového operandu. Pracuje s jednoduchou přesností.

Fused Multiply-Add of Scalar Double-Precision Floating-Point Values	
VFNMADD132SD c, z1, z2 **VFNMADD213SD c, z1, z2** **VFNMADD231SD c, z1, z2**	Operandy: r128, r128, r128/m128
Příklad pro 231: c[0..63]=c[0..63]+(negace(z1[0..63]*z2[0..63]) c[64..127] zůstane nezměnen	Instrukce provede nejprve skalární násobení dolních hodnot dvou operandů a pak znegovaný výsledek přičítá k třetímu operandu. Pořadí operandů je dáno samotnou instrukcí (132, 213, 231). Celkový výsledek je uložen do cílového operandu. Pracuje s dvojitou přesností.

Fused Negative Multiply-Substract of Packed Single-Precision Floating-Point Values	
VFNMSUB132PS c, z1, z2 **VFNMSUB213PS c, z1, z2** **VFNMSUB231PS c, z1, z2**	Operandy: r128/r256, r128/r256, r128/m128/r256/m256
Příklad pro 231: c[0..31]=c[0..31]-(negace(z1[0..31]*z2[0..31])) c[32..63]=c[32..63]-(negace(z1[32..63]*z2[32.. ..63])) ... c[224..255]=c[224..255]- -(negace(z1[224..255]*z2[224....255]))	Instrukce provede nejprve vektorové násobení hodnot dvou operandů a pak znegovaný výsledek odečítá od třetího operandu. Pořadí operandů je dáno samotnou instrukcí (132, 213, 231). Celkový výsledek je uložen do cílového operandu. Pracuje s jednoduchou přesností.

Fused Negative Multiply-Add of Packed Double-Precision Floating-Point Values	
VFNMSUB132PD c, z1, z2 **VFNMSUB213PD c, z1, z2** **VFNMSUB231PD c, z1, z2**	Operandy: r128/r256, r128/r256, r128/m128/r256/m256
Příklad pro 231: c[0..63]=c[0..63]-(negace(z1[0..63]*z2[0..63])) c[64..127]=c[64..127]-(negace(z1[64..127]* *z2[64..127])) ... c[192..255]=c[192..255]-(negace(z1[192..255]* *z2[192..255]))	Instrukce provede nejprve vektorové násobení hodnot dvou operandů a pak znegovaný výsledek odečítá od třetího operandu. Pořadí operandů je dáno samotnou instrukcí (132, 213, 231). Celkový výsledek je uložen do cílového operandu. Pracuje s dvojitou přesností.

Fused Negative Multiply-Add of Scalar Single-Precision Floating-Point Values	
VFNMSUB132SS c, z1, z2 **VFNMSUB213SS c, z1, z2** **VFNMSUB231SS c, z1, z2**	Operandy: r128, r128, r128/m128
Příklad pro 231: c[0..31]=c[0..31]-(negace(z1[0..31]*z2[0..31])) c[32..127] zůstane nezměnen	Instrukce provede nejprve skalární násobení dolních hodnot dvou operandů a pak znegovaný výsledek odečítá od třetího operandu. Pořadí operandů je dáno samotnou instrukcí (132, 213, 231). Celkový výsledek je uložen do cílového operandu. Pracuje s jednoduchou přesností.

Fused Multiply-Add of Scalar Double-Precision Floating-Point Values	
VFNMSUB132SD c, z1, z2 **VFNMSUB213SD c, z1, z2** **VFNMSUB231SD c, z1, z2**	Operandy: r128, r128, r128/m128
Příklad pro 231: c[0..63]=c[0..63]-(negace(z1[0..63]*z2[0..63])) c[64..127] zůstane nezměnen	Instrukce provede nejprve skalární násobení dolních hodnot dvou operandů a pak znegovaný výsledek odečítá od třetího operandu. Pořadí operandů je dáno samotnou instrukcí (132, 213, 231). Celkový výsledek je uložen do cílového operandu. Pracuje s dvojitou přesností.

Literatura

[1] Intel®. *Intrinsics Guide* [online].
Dostupní z URL: https://software.intel.com/sites/landingpage/IntrinsicsGuide/

[1] Cloutier, F. *x86 Instruction Set Reference* [online].
Dostupní z URL: http://www.felixcloutier.com/x86/