Instruções SSE para operações com dados em vírgula flutuante

Tabela 1: Instruções para dados simples[(1)]

Nome	Utilização	Operação/Descrição
Transferência		
MOVSS	MOVSS xmm1, xmm2/m32 MOVSS m32, xmm	Copiar SPFP; xmm1 ← xmm2/m32 m32 ← xmm
MOVSD	MOVSD xmm1, xmm2/m64 MOVSD m64, xmm	Copiar DPFP; xmm1 ← xmm2/m64 m64 ← xmm
Aritmética		
ADDSS	ADDSS xmm1, xmm2/m32	Somar 2 SPFP; xmm1 ← xmm1 + xmm2/m32
ADDSD	ADDSD xmm1, xmm2/m64	Somar 2 DPFP; xmm1 ← xmm1 + xmm2/m64
SUBSS	SUBSS xmm1, xmm2/m32	Subtrair 2 SPFP; xmm1 ← xmm1 − xmm2/m32
SUBSD	SUBSD xmm1, xmm2/m64	Subtrair 2 DPFP; xmm1 ← xmm1 − xmm2/m64
MULSS	MULSS xmm1, xmm2/m32	Multiplicar 2 SPFP; xmm1 ← xmm1 × xmm2/m32
MULSD	MULSD xmm1, xmm2/m64	Multiplicar 2 DPFP; xmm1 ← xmm1 × xmm2/m64
DIVSS	DIVSS xmm1, xmm2/m32	Dividir 2 SPFP; xmm1 ← xmm1 ÷ xmm2/m32
DIVSD	DIVSD xmm1, xmm2/m64	Dividir 2 DPFP; xmm1 ← xmm1 ÷ xmm2/m64
MINSS	MINSS xmm1, xmm2/m32	Calcular mínimo de 2 SPFP; xmm1 ← min(xmm1, xmm2/m32)
MINSD	MINSD xmm1, xmm2/m64	Calcular mínimo de 2 DPFP; xmm1 ← min(xmm1, xmm2/m64)
MAXSS	MAXSS xmm1, xmm2/m32	Calcular máximo de 2 SPFP; xmm1 ← max(xmm1, xmm2/m32)
MAXSD	MAXSD xmm1, xmm2/m64	Calcular máximo de 2 DPFP; xmm1 ← max(xmm1, xmm2/m64)
SQRTSS	SQRTSS xmm1, xmm2/m32	Calcular raiz quadrada de SPFP; xmm1 ← $\sqrt{\text{xmm2/m32}}$
SQRTSD	SQRTSD xmm1, xmm2/m64	Calcular raiz quadrada de DPFP; xmm1 ← $\sqrt{\text{xmm2/m64}}$
RCPSS	RCPSS xmm1, xmm2/m32	Calcular recíproco de SPFP; xmm1 ← 1 ÷ xmm2/m32
RSQRTSS	RSQRTSS xmm1, xmm2/m32	Calcular recíproco da raiz quadrada de SPFP; xmm1 ← 1 ÷ $\sqrt{\text{xmm2/m32}}$
ROUNDSS	ROUNDSS xmm1, xmm2/m32, imm8	Arredondar SPFP pelo modo definido em imm8 [(2)]
ROUNDSD	ROUNDSD xmm1, xmm2/m64, imm8	Arredondar DPFP pelo modo definido em imm8 [(2)]
Comparação		
CMPSS	CMPSS xmm1, xmm2/m32, imm8	Comparar 2 SPFP; xmm1[31:0] ← 1...11 se predicado de comparação verdadeiro ou 0...00 se falso (ver tabela 2)
CMPSD	CMPSD xmm1, xmm2/m64, imm8	Comparar 2 DPFP; xmm1[63:0] ← 1...11 se predicado de comparação verdadeiro ou 0...00 se falso (ver tabela 2)
COMISS	COMISS xmm1, xmm2/m32	Comparar 2 SPFP *ordered* e atualizar ZF, PF e CF[(3)]
COMISD	COMISD xmm1, xmm2/m64	Comparar 2 DPFP *ordered* e atualizar ZF, PF e CF[(3)]
UCOMISS	UCOMISS xmm1, xmm2/m32	Comparar 2 SPFP *unordered* e atualizar ZF, PF e CF[(3)]
UCOMISD	UCOMISD xmm1, xmm2/m64	Comparar 2 DPFP *unordered* e atualizar ZF, PF e CF[(3)]
Conversão		
CVTSS2SD	CVTSS2SD xmm, r/m32	Converter SPFP para DPFP
CVTSD2SS	CVTSD2SS xmm, r/m32	Converter DPFP para SPFP
CVTSI2SS	CVTSI2SS xmm, r/m32	Converter SDWORD para SPFP
CVTSI2SD	CVTSI2SD xmm, r/m64	Converter SDWORD para DPFP
CVTSS2SI	CVTSS2SI r32, xmm/m32	Converter SPFP para SDWORD
CVTSD2SI	CVTSD2SI r32, xmm/m64	Converter DPFP para SDWORD
CVTTSS2SI	CVTTSS2SI r32, xmm/m32	Converter SPFP para SDWORD com truncatura
CVTTSD2SI	CVTTSD2SI r32, xmm/m64	Converter DPFP para SDWORD com truncatura

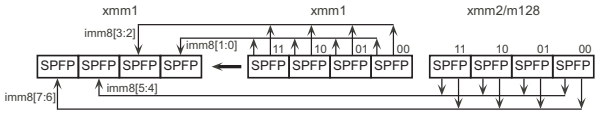
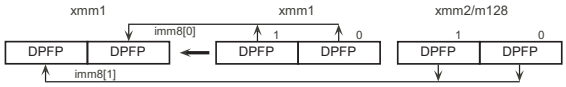
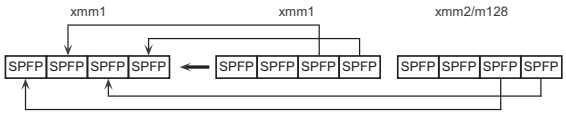
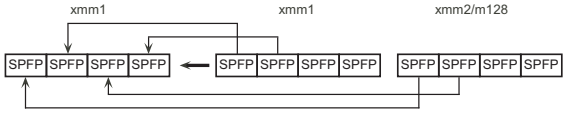
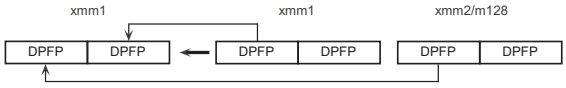
Tabela 2: Predicados de comparação e pseudo-instruções para dados simples e dados empacotados (também existem para dados DPFP e terminam em D)

Predicado	imm8	Pseudo-instrução	
		Dados simples	Dados empacotados
EQ (equal), ==	0	CMPEQSS xmm1, xmm2/m32	CMPEQPS xmm1, xmm2/m128
LT (less-than), <	1	CMPLTSS xmm1, xmm2/m32	CMPLTPS xmm1, xmm2/m128
LE (less-than-or-equal), ≤	2	CMPLESS xmm1, xmm2/m32	CMPLEPS xmm1, xmm2/m128
UNORD (unordered)	3	CMPUNORDSS xmm1, xmm2/m32	CMPUNORDPS xmm1, xmm2/m128
NEQ (not-equal), ≠	4	CMPNEQSS xmm1, xmm2/m32	CMPNEQPS xmm1, xmm2/m128
NLT (not-less-than), ≥	5	CMPNLTSS xmm1, xmm2/m32	CMPNLTPS xmm1, xmm2/m128
NLE (not-less-than-or-equal), >	6	CMPNLESS xmm1, xmm2/m32	CMPNLEPS xmm1, xmm2/m128
ORD (ordered)	7	CMPORDSS xmm1, xmm2/m32	CMPORDPS xmm1, xmm2/m128
		Obs.: m64 para DPFP	

Tabela 3: Instruções para dados empacotados – Parte 1

Nome	Utilização	Operação/Descrição
Transferência		
MOVAPS	MOVAPS xmm1, xmm2/m128 MOVAPS m128, xmm1	Copiar 4 SPFP alinhados; xmm1 ← xmm2/m128 m128 ← xmm1
MOVAPD	MOVAPD xmm1, xmm2/m128 MOVAPD m128, xmm1	Copiar 2 DPFP alinhados; xmm1 ← xmm2/m128 m128 ← xmm1
MOVUPS MOVUPD	MOVUPS xmm1, xmm2/m128 MOVUPS m128, xmm1 MOVUPD xmm1, xmm2/m128 MOVAPD m128, xmm1	Copiar 4 SPFP não alinhados; xmm1 ← xmm2/m128 m128 ← xmm1 Copiar 2 DPFP não alinhados; xmm1 ← xmm2/m128 m128 ← xmm1
MOVHPS	MOVHPS xmm1, m64 MOVHPS m64, xmm1	Copiar 2 SPFP para QWORD mais significativa; xmm1[127:64] ← m64 Copiar QWORD mais significativa (2 SPFP); m64 ← xmm1[127:64]
MOVHPD	MOVHPS xmm1, m64 MOVHPS m64, xmm1	Copiar DPFP para QWORD mais significativa; xmm1[127:64] ← m64 Copiar QWORD mais significativa (DPFP); m64 ← xmm1[127:64]
MOVLHPS	MOVLHPS xmm1, xmm2	xmm1[127:64] ← xmm2[63:0]
MOVHLPS	MOVHLPS xmm1, xmm2	xmm1[63:0] ← xmm2[127:64]
MOVMSKPS	MOVMSKPS r32, xmm	Copiar bits de sinal dos 4 SPFP; r32[31:4] ← 0, r32[3] ← xmm[127], r32[2] ← xmm[95], r32[1] ← xmm[63], r32[0] ← xmm[31]
MOVMSKPD	MOVMSKPD r32, xmm	Copiar bits de sinal dos 2 DPFP; r32[31:2] ← 0, r32[1] ← xmm[127], r32[0] ← xmm[63]
Aritmética		
ADDPS	ADDPS xmm1, xmm2/m128	Somar 4 SPFP; xmm1 ← xmm1 + xmm2/m128
ADDPD	ADDPD xmm1, xmm2/m128	Somar 2 DPFP; xmm1 ← xmm1 + xmm2/m128
SUBPS	SUBPS xmm1, xmm2/m128	Subtrair 4 SPFP; xmm1 ← xmm1 − xmm2/m128
SUBPD	SUBPD xmm1, xmm2/m128	Subtrair 2 DPFP; xmm1 ← xmm1 − xmm2/m128
MULPS	MULPS xmm1, xmm2/m128	Multiplicar 4 SPFP; xmm1 ← xmm1 × xmm2/m128
MULPD	MULPD xmm1, xmm2/m128	Multiplicar 2 DPFP; xmm1 ← xmm1 × xmm2/m128
DIVPS	DIVPS xmm1, xmm2/m128	Dividir 4 SPFP; xmm1 ← xmm1 ÷ xmm2/m128
DIVPD	DIVPD xmm1, xmm2/m128	Dividir 2 DPFP; xmm1 ← xmm1 ÷ xmm2/m128
HADDPS	HADDPS xmm1, xmm2/m128	Somar 4 SPFP na horizontal; xmm1[127:96] ← xmm2/m128[95:64] + xmm2/m128[127:96], xmm1[95:64] ← xmm2/m128[31:0] + xmm2/m128[63:32], xmm1[63:32] ← xmm1[95:64] + xmm1[127:96], xmm1[31:0] ← xmm1[31:0] + xmm1[63:32]
HADDPD	HADDPD xmm1, xmm2/m128	Somar 2 DPFP na horizontal; xmm1[127:64] ← xmm2/m128[63:0] + xmm2/m128[127:64], xmm1[63:0] ← xmm1[63:0] + xmm1[127:64]
HSUBPS	HSUBPS xmm1, xmm2/m128	Subtrair 4 SPFP na horizontal; xmm1[127:96] ← xmm2/m128[95:64] − xmm2/m128[127:96], xmm1[95:64] ← xmm2/m128[31:0] − xmm2/m128[63:32], xmm1[63:32] ← xmm1[95:64] − xmm1[127:96], xmm1[31:0] ← xmm1[31:0] − xmm1[63:32]
HSUBPD	HSUBPD xmm1, xmm2/m128	Subtrair 2 DPFP na horizontal; xmm1[127:64] ← xmm2/m128[63:0] − xmm2/m128[127:64], xmm1[63:0] ← xmm1[63:0] − xmm1[127:64]
ADDSUBPS	ADDSUBPS xmm1, xmm2/m128	Somar/subtrair 4 SPFP; xmm1[127:96] ← xmm1[127:96] + xmm2/m128[127:96], xmm1[95:64] ← xmm1[95:64] − xmm2/m128[95:64], xmm1[63:32] ← xmm1[63:32] + xmm1[63:32], xmm1[31:0] ← xmm1[31:0] − xmm1[31:0]
ADDSUBPD	ADDSUBPD xmm1, xmm2/m128	Somar/subtrair 2 DPFP; xmm1[127:64] ← xmm1[127:64] + xmm2/m128[127:64], xmm1[63:0] ← xmm1[63:0] − xmm1[63:0]
MINPS	MINPS xmm1, xmm2/m128	Calcular mínimo de cada 4 pares SPFP; xmm1 ← min(xmm1, xmm2/m128)
MINPD	MINPD xmm1, xmm2/m128	Calcular mínimo de cada 2 pares DPFP; xmm1 ← min(xmm1, xmm2/m128)
MAXPS	MAXPS xmm1, xmm2/m128	Calcular máximo de cada 4 pares SPFP; xmm1 ← max(xmm1, xmm2/m128)
MAXPD	MAXPD xmm1, xmm2/m128	Calcular máximo de cada 2 pares DPFP; xmm1 ← max(xmm1, xmm2/m128)
SQRTPS	SQRTPS xmm1, xmm2/m128	Calcular raiz quadrada de 4 SPFP; xmm1 ← $\sqrt{\text{xmm2/m128}}$
SQRTPD	SQRTPD xmm1, xmm2/m128	Calcular raiz quadrada de 2 DPFP; xmm1 ← $\sqrt{\text{xmm2/m128}}$
RCPPS	RCPPS xmm1, xmm2/m128	Calcular recíproco de 4 SPFP; xmm1 ← 1 ÷ xmm2/m128
RSQRTPS	RSQRTPS xmm1, xmm2/m128	Calcular recíproco da raiz quadrada de 4 SPFP; xmm1 ← 1 ÷ $\sqrt{\text{xmm2/m128}}$
ROUNDPS	ROUNDPS xmm1, xmm2/m128, imm8	Arredondar 4 SPFP pelo modo definido em imm8 (2)
ROUNDPD	ROUNDPD xmm1, xmm2/m128, imm8	Arredondar 2 DPFP pelo modo definido em imm8 (2)

Tabela 4: Instruções para dados empacotados – Parte 2

Nome	Utilização	Operação/Descrição
Comparação		
CMPPS	CMPPS xmm1, xmm2/m128, imm8	Comparar 4 SPFP; para cada par de operandos comparados resulta 1···11 (32 bits) se predicado de comparação verdadeiro ou 0···00 se falso (ver tabela 2)
CMPPD	CMPPD xmm1, xmm2/m128, imm8	Comparar 2 DPFP; para cada par de operandos comparados resulta 1···11 (64 bits) se predicado de comparação verdadeiro ou 0···00 se falso (ver tabela 2)
Conversão		
CVTPS2PD	CVTPS2PD xmm1, xmm2/m64	Converter 2 SPFP (xmm2/m64[63:0]) para 2 DPFP
CVTPD2PS	CVTPD2PS xmm1, xmm2/m128	Converter 2 DPFP para 2 SPFP; xmm1[127:64] ← 0
CVTPI2PS	CVTPI2PS xmm, m64	Converter 2 SDWORD para 2 SPFP (xmm[63:0])
CVTPI2PD	CVTPI2PD xmm, m64	Converter 2 SDWORD para 2 DPFP
CVTDQ2PS	CVTDQ2PS xmm1, xmm2/m128	Converter 4 SDWORD para 4 SPFP
CVTDQ2PD	CVTDQ2PD xmm1, xmm2/m64	Converter 2 SDWORD para 2 DPFP
CVTPS2DQ	CVTPS2DQ xmm1, xmm2/m128	Converter 4 SPFP para 4 SDWORD
CVTPD2DQ	CVTPD2DQ xmm1, xmm2/m128	Converter 2 DPFP para 2 SDWORD; xmm1[127:64] ← 0
CVTTPS2DQ	CVTTPS2DQ xmm1, xmm2/m128	Converter 4 SPFP para 4 SDWORD com truncatura
CVTTPD2DQ	CVTTPD2DQ xmm1, xmm2/m128	Converter 2 DPFP para 2 SDWORD com truncatura ; xmm1[127:64] ← 0
Rearranjo		
SHUFPS	SHUFPS xmm1, xmm2/m128, imm8	Copiar 2 SPFP de cada operando selecionados por imm8; 
SHUFPD	SHUFPD xmm1, xmm2/m128, imm8	Copiar DPFP de cada operando selecionados por imm8[1:0] (imm8[7:2] deve ser colocado a 0); 
UNPCKLPS	UNPCKLPS xmm1, xmm2/m128	Copiar 2 SPFP da parte menos significativa de cada operando; 
UNPCKLPD	UNPCKLPD xmm1, xmm2/m128	Copiar DPFP da parte menos significativa de cada operando; 
UNPCKHPS	UNPCKHPS xmm1, xmm2/m128	Copiar 2 SPFP da parte mais significativa de cada operando; 
UNPCKHPD	UNPCKHPD xmm1, xmm2/m128	Copiar DPFP da parte mais significativa de cada operando; 
INSERTPS	INSERTPS xmm1, xmm2/m32, imm8	Copiar 1 SPFP e, opcionalmente, colocar dados a zero; imm8[7:6] é a posição de origem (é 0 no caso de m32), imm8[5:4] é a posição de destino, imm8[3:0] indica o(s) SPFP a colocar a 0 (p. ex., imm8[3:0]=0011 coloca 2 SPFP menos signif. a 0)
EXTRACTPS	EXTRACTPS reg/m32, xmm, imm8	Copiar SPFP para memória ou registo de uso geral; se imm8[1:0] igual a: 00 copiar xmm[31:0], 01 copiar xmm[63:31], 10 copiar xmm[95:64], 11 copiar xmm[127:96]
Lógica		
ANDPS	ANDPS xmm1, xmm2/m128	AND bit-a-bit entre operandos com 4 SPFP; xmm1 ← xmm1 AND xmm2/m128
ANDPD	ANDPD xmm1, xmm2/m128	AND bit-a-bit entre operandos com 2 DPFP; xmm1 ← xmm1 AND xmm2/m128
ANDNPS	ANDNPS xmm1, xmm2/m128	NOT do destino seguido de AND bit-a-bit com 4 SPFP; xmm1 ← NOT(xmm1) AND xmm2/m128
ANDNPD	ANDNPD xmm1, xmm2/m128	NOT do destino seguido de AND bit-a-bit com 2 DPFP; xmm1 ← NOT(xmm1) AND xmm2/m128
ORPS	ORPS xmm1, xmm2/m128	OR bit-a-bit entre operandos com 4 SPFP; xmm1 ← xmm1 OR xmm2/m128
ORPD	ORPD xmm1, xmm2/m128	OR bit-a-bit entre operandos com 2 DPFP; xmm1 ← xmm1 OR xmm2/m128
XORPS	XORPS xmm1, xmm2/m128	XOR bit-a-bit entre operandos com 4 SPFP; xmm1 ← xmm1 XOR xmm2/m128
XORPD	XORPD xmm1, xmm2/m128	XOR bit-a-bit entre operandos com 2 DPFP; xmm1 ← xmm1 XOR xmm2/m128

Notas:
(1) SPFP: valor em vírgula flutuante com precisão simples; DPFP: valor em vírgula flutuante com precisão dupla. Nos registos XMM, um SPFP e um DPFP ocupam os 32 e os 64 bits menos significativos, respetivamente.
(2) Se imm8[2:0] igual a: 000 arredondar para o par mais próximo, 001 arredondar para baixo, 010 arredondar para cima, 011 truncar.
Para mais detalhes, consultar "Intel 64 and IA-32 Architectures Software Developer's Manual, Vol. 2B: Instruction Set Reference".
(3) Resultado da comparação: se $>$ ZF=0, PF=0 e CF=0, se $<$ ZF=0, PF=0 e CF=1, se $=$ ZF=1, PF=0 e CF=0, se unordered ZF=1, PF=1 e CF=1.