x86 instruction listings

The x86 instruction set has been extended several times, introducing wider registers and datatypes and/or new functionality.

x86 integer instructions

This is the full 8086/8088 instruction set, but most, if not all of these instructions are available in 32-bit mode, they just operate on 32-bit registers (eax, ebx, etc) and values instead of their 16-bit (ax, bx, etc) counterparts. See also x86 assembly language for a quick tutorial for this processor family. The updated instruction set is also grouped according to architecture (i386, i486, i686) and more generally is referred to as x86_32 and x86_64 (also known as AMD64).

Original 8086/8088 instructions

Instruction	Meaning	Notes
AAA	**A**SCII **a**djust AL after **a**ddition	used with unpacked binary coded decimal
AAD	**A**SCII **a**djust AX before **d**ivision	8086/8088 datasheet documents only base 10 version of the AAD instruction (opcode 0xD5 0x0A), but any other base will work. Later Intel's documentation has the generic form too. NEC V20 and V30 (and possibly other NEC V-series CPUs) always use base 10, and ignore the argument, causing a number of incompatibilities
AAM	**A**SCII **a**djust AX after **m**ultiplication	Only base 10 version is documented, see notes for AAD
AAS	**A**SCII **a**djust AL after **s**ubtraction	
ADC	**Ad**d with **c**arry	destination := destination + source + carry_flag
ADD	Add	
AND	Logical AND	
CALL	Call procedure	
CBW	**C**onvert **b**yte to **w**ord	
CLC	**Cl**ear **c**arry flag	
CLD	**Cl**ear **d**irection flag	
CLI	**Cl**ear **i**nterrupt flag	
CMC	**Com**plement **c**arry flag	
CMP	**Comp**are operands	
CMPSB	**Comp**are bytes in memory	
CMPSW	**Comp**are words	
CWD	**C**onvert **w**ord to **d**oubleword	
DAA	**D**ecimal **a**djust AL after **a**ddition	(used with packed binary coded decimal)
DAS	**D**ecimal **a**djust AL after **s**ubtraction	
DEC	**Dec**rement by 1	
DIV	Unsigned **div**ide	
ESC	Used with floating-point unit	
HLT	Enter **halt** state	
IDIV	S**i**gned **div**ide	

IMUL	S**i**gned **mul**tiply	
IN	**In**put from port	
INC	**Inc**rement by 1	
INT	Call to **int**errupt	
INTO	Call to **int**errupt if **o**verflow	
IRET	**Ret**urn from **i**nterrupt	
Jxx	**J**ump if condition	(*JA, JAE, JB, JBE, JC, JCXZ, JE, JG, JGE, JL, JLE, JNA, JNAE, JNB, JNBE, JNC, JNE, JNG, JNGE, JNL, JNLE, JNO, JNP, JNS, JNZ, JO, JP, JPE, JPO, JS, JZ*)
JMP	**Jump**	
LAHF	**L**oad **f**lags into **AH** register	
LDS	**L**oad pointer using **DS**	
LEA	**L**oad **E**ffective **A**ddress	
LES	**L**oad **ES** with pointer	
LOCK	Assert BUS **LOCK#** signal	(for multiprocessing)
LODSB	**Lo**a**d** **s**igned **b**yte	
LODSW	**Lo**a**d** **s**igned **w**ord	
LOOP/LOOPx	**Loop** control	(*LOOPE, LOOPNE, LOOPNZ, LOOPZ*)
MOV	**Mov**e	
MOVSB	**Mov**e **b**yte from **s**tring to string	
MOVSW	**Mov**e word from **s**tring to **s**tring	
MUL	Unsigned **mul**tiply	
NEG	Two's complement **neg**ation	
NOP	**No** **op**eration	opcode (0x90) equivalent to XCHG EAX, EAX
NOT	Negate the operand, logical **NOT**	
OR	Logical **OR**	
OUT	**Out**put to port	
POP	**Pop** data from stack	POP CS (opcode 0x0F) works only on 8086/8088. Later CPUs use 0x0F as a prefix for newer instructions.
POPF	**Pop** data into **f**lags register	
PUSH	**Push** data onto stack	
PUSHF	**Push** **f**lags onto stack	
RCL	**R**otate **l**eft (with **c**arry)	
RCR	**R**otate **r**ight (with **c**arry)	
REPxx	**Rep**eat MOVS/STOS/CMPS/LODS/SCAS	(*REP, REPE, REPNE, REPNZ, REPZ*)
RET	**Ret**urn from procedure	
RETN	**Ret**urn from **n**ear procedure	
RETF	**Ret**urn from **f**ar procedure	
ROL	**Ro**tate **l**eft	
ROR	**Ro**tate **r**ight	

SAHF	**S**tore **AH** into **f**lags	
SAL	**S**hift **A**rithmetically **l**eft (signed shift left)	
SAR	**S**hift **A**rithmetically **r**ight (signed shift right)	
SBB	**S**u**b**traction with **b**orrow	
SCASB	**C**omp**a**re **b**yte **s**tring	
SCASW	**C**omp**a**re **w**ord **s**tring	
SHL	**Sh**ift **l**eft (unsigned shift left)	
SHR	**Sh**ift **r**ight (unsigned shift right)	
STC	**Set c**arry flag	
STD	**Set d**irection flag	
STI	**Set i**nterrupt flag	
STOSB	**Sto**re **b**yte in **s**tring	
STOSW	**Sto**re **w**ord in **s**tring	
SUB	**Sub**traction	
TEST	Logical compare (AND)	
WAIT	**Wait** until not busy	Waits until BUSY# pin is inactive (used with floating-point unit)
XCHG	E**xch**an**g**e data	
XLAT	Table look-up translation	
XOR	E**x**clusive **OR**	

Added in specific processors

Added with 80186/80188

Instruction	Meaning	Notes
BOUND	Check array index against **bound**s	raises software interrupt 5 if test fails

ENTER	**Enter** stack frame	equivalent to
		`PUSH BP`
		`MOV BP, SP`
		`SUB SP, n`

INS	**In**put from port to **s**tring	equivalent to
		`IN (E)AX, DX`
		`MOV ES:[(E)DI], (E)AX`
		`; adjust (E)DI according to operand size and DF`

LEAVE	**Leave** stack frame	equivalent to
		`MOV SP, BP`
		`POP BP`

OUTS	**Out**put **s**tring to port	equivalent to
		`MOV (E)AX, DS:[(E)SI]`
		`OUT DX, (E)AX`
		`; adjust (E)SI according to operand size and DF`

POPA	**Pop a**ll general purpose registers from stack	equivalent to `POP DI, SI, BP, SP, BX, DX, CX, AX`
PUSHA	**Push a**ll general purpose registers onto stack	equivalent to `PUSH AX, CX, DX, BX, SP, BP, SI, DI`

Added with 80286

Instruction	Meaning	Notes
ARPL	**A**djust **RPL** field of selector	
CLTS	**Cl**ear **t**ask-**s**witched flag in register CR0	
LAR	**L**oad **a**ccess **r**ights byte	
LGDT	**L**oad **g**lobal **d**escriptor **t**able	
LIDT	**L**oad **i**nterrupt **d**escriptor **t**able	
LLDT	**L**oad **l**ocal **d**escriptor **t**able	
LMSW	**L**oad **m**achine **s**tatus **w**ord	
LOADALL	**Load all** CPU registers, including internal ones such as GDT	Undocumented, (80)286 and 386 only
LSL	**L**oad **s**egment **l**imit	
LTR	**L**oad **t**ask **r**egister	
SGDT	**S**tore **g**lobal **d**escriptor **t**able	
SIDT	**S**tore **i**nterrupt **d**escriptor **t**able	
SLDT	**S**tore **l**ocal **d**escriptor **t**able	
SMSW	**S**tore **m**achine **s**tatus **w**ord	
STR	**S**tore **t**ask **r**egister	
VERR	**Ver**ify a segment for **r**eading	
VERW	**Ver**ify a segment for **w**riting	

Added with 80386

Instruction	Meaning	Notes
BSF	**B**it **s**can **f**orward	
BSR	**B**it **s**can **r**everse	
BT	**B**it **t**est	
BTC	**B**it **t**est and **c**omplement	
BTR	**B**it **t**est and **r**eset	
BTS	**B**it **t**est and **s**et	
CDQ	**C**onvert **d**ouble-word to **q**uad-word	Sign-extends EAX into EDX, forming the quad-word EDX:EAX. Since (I)DIV uses EDX:EAX as its input, CDQ must be called after setting EAX if EDX is not manually initialized (as in 64/32 division) before (I)DIV.
CMPSD	**C**ompare **s**tring **d**ouble-word	Compares ES:[(E)DI] with DS:[SI]

CWDE	**C**onvert **w**ord to **d**ouble-word	Unlike CWD, CWDE sign-extends AX to EAX instead of AX to DX:AX
INSB, INSW, INSD	**In**put from port to **s**tring with explicit size	same as INS
IRETx	**I**nterrupt **ret**urn; D suffix means 32-bit return, F suffix means do not generate epilogue code (i.e. LEAVE instruction)	Use IRETD rather than IRET in 32-bit situations
JCXZ, JECXZ	**J**ump if register (**E**)**CX** is **z**ero	
LFS, LGS	Load far pointer	
LSS	**L**oad **s**tack **s**egment	
LODSD	**Load** string	can be prefixed with REP
LOOPW, LOOPD	**Loop**	Loop; counter register is (E)CX
LOOPEW, LOOPED	**Loop** while **e**qual	
LOOPZW, LOOPZD	**Loop** while **z**ero	
LOOPNEW, LOOPNED	**Loop** while **n**ot **e**qual	
LOOPNZW, LOOPNZD	**Loop** while **n**ot **z**ero	
MOVSW, MOVSD	**Mov**e data from **s**tring to string	
MOVSX	**Mov**e with **s**ign-e**x**tend	
MOVZX	**Mov**e with **z**ero-e**x**tend	
POPAD	**Pop** **a**ll **d**ouble-word (32-bit) registers from stack	Does not pop register ESP off of stack
POPFD	**Pop** data into E**F**LAGS register	
PUSHAD	**Push** **a**ll **d**ouble-word (32-bit) registers onto stack	
PUSHFD	**Push** E**F**LAGS register onto stack	
SCASD	**Sca**n **s**tring data **d**ouble-word	
SETA, SETAE, SETB, SETBE, SETC, SETE, SETG, SETGE, SETL, SETLE, SETNA, SETNAE, SETNB, SETNBE, SETNC, SETNE, SETNG, SETNGE, SETNL, SETNLE, SETNO, SETNP, SETNS, SETNZ, SETO, SETP, SETPE, SETPO, SETS, SETZ	**Set** byte to one on condition	
SHLD	**Sh**ift **l**eft **d**ouble-word	
SHRD	**Sh**ift **r**ight **d**ouble-word	
STOSx	**Sto**re string	

Added with 80486

Instruction	Meaning	Notes
BSWAP	**B**yte **Swap**	Only works for 32 bit registers.
CMPXCHG	**C**o**MP**are and e**XCH**an**G**e	
INVD	**Inv**ali**d**ate Internal Caches	
INVLPG	Invalidate TLB Entry	
WBINVD	Write Back and Invalidate Cache	
XADD	Exchange and Add	

Added with Pentium

Instruction	Meaning	Notes
CPUID	**CPU ID**entification	This was also added to later 80486 processors.
CMPXCHG8B	**C**o**MP**are and e**XCH**an**G**e **8** **b**ytes	
RDMSR	**R**ea**D** from **M**odel-**S**pecific **R**egister	
RDTSC	**R**ea**D** **T**ime **S**tamp **C**ounter	
WRMSR	**WR**ite to **M**odel-**S**pecific **R**egister	
RSM [1]	**R**esume from **S**ystem **M**anagement Mode	This was introduced by the i386SL and later and is also in the i486SL and later. Resumes from System Management Mode (SMM)

Added with Pentium MMX

Instruction	Meaning	Notes
RDPMC	Read the PMC [Performance Monitoring Counter]	Specified in the ECX register into registers EDX:EAX

Also MMX registers and MMX support instructions were added. They are usable for both integer and floating point operations, see below.

Added with AMD K6-2

SYSCALL, SYSRET (functionally equivalent to SYSENTER and SYSEXIT)

Added with Pentium Pro

Conditional MOV: CMOVA, CMOVAE, CMOVB, CMOVBE, CMOVC, CMOVE, CMOVG, CMOVGE, CMOVL, CMOVLE, CMOVNA, CMOVNAE, CMOVNB, CMOVNBE, CMOVNC, CMOVNE, CMOVNG, CMOVNGE, CMOVNL, CMOVNLE, CMOVNO, CMOVNP, CMOVNS, CMOVNZ, CMOVO, CMOVP, CMOVPE, CMOVPO, CMOVS, CMOVZ, SYSENTER (SYStem call ENTER), SYSEXIT (SYStem call EXIT), UD2

Added with SSE

MASKMOVQ, MOVNTPS, MOVNTQ, PREFETCH0, PREFETCH1, PREFETCH2, PREFETCHNTA, SFENCE (for Cacheability and Memory Ordering)

Added with SSE2

CLFLUSH, LFENCE, MASKMOVDQU, MFENCE, MOVNTDQ, MOVNTI, MOVNTPD, PAUSE (for Cacheability)

Added with x86-64

CMPXCHG16B (CoMPare and eXCHanGe 16 Bytes), RDTSCP (ReaD Time Stamp Counter and Processor ID)

Added with SSE3

LDDQU (for Video Encoding)

MONITOR, MWAIT (for thread synchronization; only on processors supporting Hyper-threading and some dual-core processors like Core 2, Phenom and others)

Added with AMD-V

CLGI, SKINIT, STGI, VMLOAD, VMMCALL, VMRUN, VMSAVE (SVM instructions of AMD-V)

Added with Intel VT-x

VMPTRLD, VMPTRST, VMCLEAR, VMREAD, VMWRITE, VMCALL, VMLAUNCH, VMRESUME, VMXOFF, VMXON

Added with SSE4a

LZCNT, POPCNT (POPulation CouNT) - advanced bit manipulation

x87 floating-point instructions

Original 8087 instructions

Instruction	Meaning	Notes
F2XM1	2^x - 1	Can be computed faster than 2^x
FABS	Absolute value	
FADD	Add	
FADDP	Add and pop	
FBLD	Load BCD	
FBSTP	Store BCD and pop	
FCHS	Change sign	
FCLEX	Clear exceptions	
FCOM	Compare	
FCOMP	Compare and pop	
FCOMPP	Compare and pop twice	
FDECSTP	Decrement floating point stack pointer	
FDISI	Disable interrupts	8087 only, otherwise FNOP

FDIV	Divide	Pentium FDIV bug
FDIVP	Divide and pop	
FDIVR	Divide reversed	
FDIVRP	Divide reversed and pop	
FENI	Enable interrupts	8087 only, otherwise FNOP
FFREE	Free register	
FIADD	Integer add	
FICOM	Integer compare	
FICOMP	Integer compare and pop	
FIDIV	Integer divide	
FIDIVR	Integer divide reversed	
FILD	Load integer	
FIMUL	Integer multiply	
FINCSTP	Increment floating point stack pointer	
FINIT	Initialize floating point processor	
FIST	Store integer	
FISTP	Store integer and pop	
FISUB	Integer subtract	
FISUBR	Integer subtract reversed	
FLD	Floating point load	
FLD1	Load 1.0 onto stack	
FLDCW	Load control word	
FLDENV	Load environment state	
FLDENVW		
FLDL2E	Load $\log_2(e)$ onto stack	
FLDL2T	Load $\log_2(10)$ onto stack	
FLDLG2	Load $\log_{10}(2)$ onto stack	
FLDLN2	Load ln(2) onto stack	
FLDPI	Load π onto stack	
FLDZ	Load 0.0 onto stack	
FMUL	Multiply	
FMULP	Multiply and pop	
FNCLEX	Clear exceptions, no wait	
FNDISI	Disable interrupts, no wait	8087 only, otherwise FNOP
FNENI	Enable interrupts, no wait	8087 only, otherwise FNOP
FNINIT	Initialize floating point processor, no wait	
FNOP	No operation	
FNSAVE	Save FPU state, no wait, 8-bit	
FNSAVEW	Save FPU state, no wait, 16-bit	

FNSTCW	Store control word, no wait	
FNSTENV	Store FPU environment, no wait	
FNSTENVW	Store FPU environment, no wait, 16-bit	
FNSTSW	Store status word, no wait	
FPATAN	Partial arctangent	
FPREM	Partial remainder	
FPTAN	Partial tangent	
FRNDINT	Round to integer	
FRSTOR	Restore saved state	
FRSTORW	Restore saved state	Perhaps not actually available in 8087
FSAVE	Save FPU state	
FSAVEW	Save FPU state, 16-bit	
FSCALE	Scale by factor of 2	
FSQRT	Square root	
FST	Floating point store	
FSTCW	Store control word	
FSTENV	Store FPU environment	
FSTENVW	Store FPU environment, 16-bit	
FSTP	Store and pop	
FSTSW	Store status word	
FSUB	Subtract	
FSUBP	Subtract and pop	
FSUBR	Reverse subtract	
FSUBRP	Reverse subtract and pop	
FTST	Test for zero	
FWAIT	Wait while FPU is executing	
FXAM	Examine condition flags	
FXCH	Exchange registers	
FXTRACT	Extract exponent and significand	
FYL2X	y * log$_2$(x)	
FYL2XP1	y * log$_2$(x+1)	

Added in specific processors

Added with 80287

FSETPM

Added with 80387

FCOS, FLDENVD, FNSAVED, FNSTENVD, FPREM1, FRSTORD, FSAVED, FSIN, FSINCOS, FSTENVD, FUCOM, FUCOMP, FUCOMPP

Added with Pentium Pro

- FCMOV variants: FCMOVB, FCMOVBE, FCMOVE, FCMOVNB, FCMOVNBE, FCMOVNE, FCMOVNU, FCMOVU
- FCOMI variants: FCOMI, FCOMIP, FUCOMI, FUCOMIP

Added with SSE

- FXRSTOR*, FXSAVE*
- Also supported on later Pentium IIs, though they do not contain SSE support

Added with SSE3

FISTTP (x87 to integer conversion with truncation regardless of status word)

Undocumented instructions

FFREEP performs FFREE ST(i) and pop stack

SIMD instructions

MMX instructions

added with Pentium MMX EMMS, MOVD, MOVQ, PACKSSDW, PACKSSWB, PACKUSWB, PADDB, PADDD, PADDSB, PADDSW, PADDUSB, PADDUSW, PADDW, PAND, PANDN, PCMPEQB, PCMPEQD, PCMPEQW, PCMPGTB, PCMPGTD, PCMPGTW, PMADDWD, PMULHW, PMULLW, POR, PSLLD, PSLLQ, PSLLW, PSRAD, PSRAW, PSRLD, PSRLQ, PSRLW, PSUBB, PSUBD, PSUBSB, PSUBSW, PSUBUSB, PSUBUSW, PSUBW, PUNPCKHBW, PUNPCKHDQ, PUNPCKHWD, PUNPCKLBW, PUNPCKLDQ, PUNPCKLWD, PXOR

MMX+ instructions

added with Athlon

Same as the SSE SIMD Integer Instructions which operated on MMX registers.

EMMX instructions

EMMI instructions - added with 6x86MX from Cyrix, deprecated now

PAVEB, PADDSIW, PMAGW, PDISTIB, PSUBSIW, PMVZB, PMULHRW, PMVNZB, PMVLZB, PMVGEZB, PMULHRIW, PMACHRIW

3DNow! instructions

added with K6-2

FEMMS, PAVGUSB, PF2ID, PFACC, PFADD, PFCMPEQ, PFCMPGE, PFCMPGT, PFMAX, PFMIN, PFMUL, PFRCP, PFRCPIT1, PFRCPIT2, PFRSQIT1, PFRSQRT, PFSUB, PFSUBR, PI2FD, PMULHRW, PREFETCH, PREFETCHW

3DNow!+ instructions

added with Athlon

PF2IW, PFNACC, PFPNACC, PI2FW, PSWAPD

added with Geode GX

PFRSQRTV, PFRCPV

SSE instructions

added with Pentium III also see integer instruction added with Pentium III

SSE SIMD Floating-Point Instructions

ADDPS, ADDSS, CMPPS, CMPSS, COMISS, CVTPI2PS, CVTPS2PI, CVTSI2SS, CVTSS2SI, CVTTPS2PI, CVTTSS2SI, DIVPS, DIVSS, LDMXCSR, MAXPS, MAXSS, MINPS, MINSS, MOVAPS, MOVHLPS, MOVHPS, MOVLHPS, MOVLPS, MOVMSKPS, MOVNTPS, MOVSS, MOVUPS, MULPS, MULSS, RCPPS, RCPSS, RSQRTPS, RSQRTSS, SHUFPS, SQRTPS, SQRTSS, STMXCSR, SUBPS, SUBSS, UCOMISS, UNPCKHPS, UNPCKLPS

SSE SIMD Integer Instructions

ANDNPS, ANDPS, ORPS, PAVGB, PAVGW, PEXTRW, PINSRW, PMAXSW, PMAXUB, PMINSW, PMINUB, PMOVMSKB, PMULHUW, PSADBW, PSHUFW, XORPS

Instruction	Opcode	Meaning	Notes
MOVUPS xmm1, xmm2/m128	0F 10 /r	Move Unaligned Packed Single-Precision Floating-Point Values	
MOVSS xmm1, xmm2/m32	F3 0F 10 /r	Move Scalar Single-Precision Floating-Point Values	
MOVUPS xmm2/m128, xmm1	0F 11 /r	Move Unaligned Packed Single-Precision Floating-Point Values	
MOVSS xmm2/m32, xmm1	F3 0F 11 /r	Move Scalar Single-Precision Floating-Point Values	
MOVLPS xmm, m64	0F 12 /r	Move Low Packed Single-Precision Floating-Point Values	
MOVHLPS xmm1, xmm2	0F 12 /r	Move Packed Single-Precision Floating-Point Values High to Low	
MOVLPS m64, xmm	0F 13 /r	Move Low Packed Single-Precision Floating-Point Values	
UNPCKLPS xmm1, xmm2/m128	0F 14 /r	Unpack and Interleave Low Packed Single-Precision Floating-Point Values	
UNPCKHPS xmm1, xmm2/m128	0F 15 /r	Unpack and Interleave High Packed Single-Precision Floating-Point Values	
MOVHPS xmm, m64	0F 16 /r	Move High Packed Single-Precision Floating-Point Values	
MOVLHPS xmm1, xmm2	0F 16 /r	Move Packed Single-Precision Floating-Point Values Low to High	
MOVHPS m64, xmm	0F 17 /r	Move High Packed Single-Precision Floating-Point Values	
PREFETCHNTA	0F 18 /0	Prefetch Data Into Caches (non-temporal data with respect to all cache levels)	
PREFETCH0	0F 18 /1	Prefetch Data Into Caches (temporal data)	
PREFETCH1	0F 18 /2	Prefetch Data Into Caches (temporal data with respect to first level cache)	
PREFETCH2	0F 18 /3	Prefetch Data Into Caches (temporal data with respect to second level cache)	

NOP	0F 1F /0	No Operation	
MOVAPS xmm1, xmm2/m128	0F 28 /r	Move Aligned Packed Single-Precision Floating-Point Values	
MOVAPS xmm2/m128, xmm1	0F 29 /r	Move Aligned Packed Single-Precision Floating-Point Values	
CVTPI2PS xmm, mm/m64	0F 2A /r	Convert Packed Dword Integers to Packed Single-Precision FP Values	
CVTSI2SS xmm, r/m32	F3 0F 2A /r	Convert Dword Integer to Scalar Single-Precision FP Value	
MOVNTPS m128, xmm	0F 2B /r	Store Packed Single-Precision Floating-Point Values Using Non-Temporal Hint	
CVTTPS2PI mm, xmm/m64	0F 2C /r	Convert with Truncation Packed Single-Precision FP Values to Packed Dword Integers	
CVTTSS2SI r32, xmm/m32	F3 0F 2C /r	Convert with Truncation Scalar Single-Precision FP Value to Dword Integer	
CVTPS2PI mm, xmm/m64	0F 2D /r	Convert Packed Single-Precision FP Values to Packed Dword Integers	
CVTSS2SI r32, xmm/m32	F3 0F 2D /r	Convert Scalar Single-Precision FP Value to Dword Integer	
UCOMISS xmm1, xmm2/m32	0F 2E /r	Unordered Compare Scalar Single-Precision Floating-Point Values and Set EFLAGS	
COMISS xmm1, xmm2/m32	0F 2F /r	Compare Scalar Ordered Single-Precision Floating-Point Values and Set EFLAGS	
SQRTPS xmm1, xmm2/m128	0F 51 /r	Compute Square Roots of Packed Single-Precision Floating-Point Values	
SQRTSS xmm1, xmm2/m32	F3 0F 51 /r	Compute Square Root of Scalar Single-Precision Floating-Point Value	
RSQRTPS xmm1, xmm2/m128	0F 52 /r	Compute Reciprocal of Square Root of Packed Single-Precision Floating-Point Value	
RSQRTSS xmm1, xmm2/m32	F3 0F 52 /r	Compute Reciprocal of Square Root of Scalar Single-Precision Floating-Point Value	
RCPPS xmm1, xmm2/m128	0F 53 /r	Compute Reciprocal of Packed Single-Precision Floating-Point Values	
RCPSS xmm1, xmm2/m32	F3 0F 53 /r	Compute Reciprocal of Scalar Single-Precision Floating-Point Values	
ANDPS xmm1, xmm2/m128	0F 54 /r	Bitwise Logical AND of Packed Single-Precision Floating-Point Values	
ANDNPS xmm1, xmm2/m128	0F 55 /r	Bitwise Logical AND NOT of Packed Single-Precision Floating-Point Values	
ORPS xmm1, xmm2/m128	0F 56 /r	Bitwise Logical OR of Single-Precision Floating-Point Values	
XORPS xmm1, xmm2/m128	0F 57 /r	Bitwise Logical XOR for Single-Precision Floating-Point Values	
ADDPS xmm1, xmm2/m128	0F 58 /r	Add Packed Single-Precision Floating-Point Values	
ADDSS xmm1, xmm2/m32	F3 0F 58 /r	Add Scalar Single-Precision Floating-Point Values	
MULPS xmm1, xmm2/m128	0F 59 /r	Multiply Packed Single-Precision Floating-Point Values	
MULSS xmm1, xmm2/m32	F3 0F 59 /r	Multiply Scalar Single-Precision Floating-Point Values	
SUBPS xmm1, xmm2/m128	0F 5C /r	Subtract Packed Single-Precision Floating-Point Values	
SUBSS xmm1, xmm2/m32	F3 0F 5C /r	Subtract Scalar Single-Precision Floating-Point Values	
MINPS xmm1, xmm2/m128	0F 5D /r	Return Minimum Packed Single-Precision Floating-Point Values	
MINSS xmm1, xmm2/m32	F3 0F 5D /r	Return Minimum Scalar Single-Precision Floating-Point Values	
DIVPS xmm1, xmm2/m128	0F 5E /r	Divide Packed Single-Precision Floating-Point Values	
DIVSS xmm1, xmm2/m32	F3 0F 5E /r	Divide Scalar Single-Precision Floating-Point Values	
MAXPS xmm1, xmm2/m128	0F 5F /r	Return Maximum Packed Single-Precision Floating-Point Values	
MAXSS xmm1, xmm2/m32	F3 0F 5F /r	Return Maximum Scalar Single-Precision Floating-Point Values	
PSHUFW mm1, mm2/m64, imm8	0F 70 /r ib	Shuffle Packed Words	
LDMXCSR m32	0F AE /2	Load MXCSR Register State	
STMXCSR m32	0F AE /3	Store MXCSR Register State	

SFENCE	0F AE /7	Store Fence	
CMPPS xmm1, xmm2/m128, imm8	0F C2 /r ib	Compare Packed Single-Precision Floating-Point Values	
CMPSS xmm1, xmm2/m32, imm8	F3 0F C2 /r ib	Compare Scalar Single-Precision Floating-Point Values	
PINSRW mm, r32/m16, imm8	0F C4 /r	Insert Word	
PEXTRW r32, mm, imm8	0F C5 /r	Extract Word	
SHUFPS xmm1, xmm2/m128, imm8	0F C6 /r ib	Shuffle Packed Single-Precision Floating-Point Values	
PMOVMSKB r32, mm	0F D7 /r	Move Byte Mask	
PMINUB mm1, mm2/m64	0F DA /r	Minimum of Packed Unsigned Byte Integers	
PMAXUB mm1, mm2/m64	0F DE /r	Maximum of Packed Unsigned Byte Integers	
PAVGB mm1, mm2/m64	0F E0 /r	Average Packed Integers	
PAVGW mm1, mm2/m64	0F E3 /r	Average Packed Integers	
PMULHUW mm1, mm2/m64	0F E4 /r	Multiply Packed Unsigned Integers and Store High Result	
MOVNTQ m64, mm	0F E7 /r	Store of Quadword Using Non-Temporal Hint	
PMINSW mm1, mm2/m64	0F EA /r	Minimum of Packed Signed Word Integers	
PMAXSW mm1, mm2/m64	0F EE /r	Maximum of Packed Signed Word Integers	
PSADBW mm1, mm2/m64	0F F6 /r	Compute Sum of Absolute Differences	
MASKMOVQ mm1, mm2	0F F7 /r	Store Selected Bytes of Quadword	

SSE2 instructions

added with Pentium 4 also see integer instructions added with Pentium 4

SSE2 SIMD Floating-Point Instructions

ADDPD, ADDSD, ANDNPD, ANDPD, CMPPD, CMPSD*, COMISD, CVTDQ2PD, CVTDQ2PS, CVTPD2DQ, CVTPD2PI, CVTPD2PS, CVTPI2PD, CVTPS2DQ, CVTPS2PD, CVTSD2SI, CVTSD2SS, CVTSI2SD, CVTSS2SD, CVTTPD2DQ, CVTTPD2PI, CVTTPS2DQ, CVTTSD2SI, DIVPD, DIVSD, MAXPD, MAXSD, MINPD, MINSD, MOVAPD, MOVHPD, MOVLPD, MOVMSKPD, MOVSD*, MOVUPD, MULPD, MULSD, ORPD, SHUFPD, SQRTPD, SQRTSD, SUBPD, SUBSD, UCOMISD, UNPCKHPD, UNPCKLPD, XORPD

- CMPSD *and* MOVSD *have the same name as the string instruction mnemonics* CMPSD (CMPS) *and* MOVSD (MOVS)*, however, the former refer to scalar double-precision floating-points whereas the latters refer to doubleword strings.*

SSE2 SIMD Integer Instructions

MOVDQ2Q, MOVDQA, MOVDQU, MOVQ2DQ, PADDQ, PSUBQ, PMULUDQ, PSHUFHW, PSHUFLW, PSHUFD, PSLLDQ, PSRLDQ, PUNPCKHQDQ, PUNPCKLQDQ

SSE3 instructions

added with Pentium 4 supporting SSE3 also see integer and floating-point instructions added with Pentium 4 SSE3

SSE3 SIMD Floating-Point Instructions

- ADDSUBPD, ADDSUBPS (for Complex Arithmetic)
- HADDPD, HADDPS, HSUBPD, HSUBPS (for Graphics)
- MOVDDUP, MOVSHDUP, MOVSLDUP (for Complex Arithmetic)

SSSE3 instructions

added with Xeon 5100 series and initial Core 2

- PSIGNW, PSIGND, PSIGNB
- PSHUFB
- PMULHRSW, PMADDUBSW
- PHSUBW, PHSUBSW, PHSUBD
- PHADDW, PHADDSW, PHADDD
- PALIGNR
- PABSW, PABSD, PABSB

SSE4 instructions

SSE4.1

added with Core 2 manufactured in 45nm

- MPSADBW
- PHMINPOSUW
- PMULLD, PMULDQ
- DPPS, DPPD
- BLENDPS, BLENDPD, BLENDVPS, BLENDVPD, PBLENDVB, PBLENDW
- PMINSB, PMAXSB, PMINUW, PMAXUW, PMINUD, PMAXUD, PMINSD, PMAXSD
- ROUNDPS, ROUNDSS, ROUNDPD, ROUNDSD
- INSERTPS, PINSRB, PINSRD/PINSRQ, EXTRACTPS, PEXTRB, PEXTRW, PEXTRD/PEXTRQ
- PMOVSXBW, PMOVZXBW, PMOVSXBD, PMOVZXBD, PMOVSXBQ, PMOVZXBQ, PMOVSXWD, PMOVZXWD, PMOVSXWQ, PMOVZXWQ, PMOVSXDQ, *PMOVZXDQ
- PTEST
- PCMPEQQ
- PACKUSDW
- MOVNTDQA

SSE4a

added with Phenom processors

- LZCNT, POPCNT (POPulation CouNT) - advanced bit manipulation
- EXTRQ/INSERTQ
- MOVNTSD/MOVNTSS

SSE4.2

added with Nehalem processors

- CRC32
- PCMPESTRI
- PCMPESTRM
- PCMPISTRI
- PCMPISTRM
- PCMPGTQ

Intel AVX FMA instructions

Instruction	Opcode	Meaning	Notes
VFMADDPD xmm0, xmm1, xmm2, xmm3	C4E3 WvvvvL01 69 /r /is4	Fused Multiply-Add of Packed Double-Precision Floating-Point Values	
VFMADDPS xmm0, xmm1, xmm2, xmm3	C4E3 WvvvvL01 68 /r /is4	Fused Multiply-Add of Packed Single-Precision Floating-Point Values	
VFMADDSD xmm0, xmm1, xmm2, xmm3	C4E3 WvvvvL01 6B /r /is4	Fused Multiply-Add of Scalar Double-Precision Floating-Point Values	
VFMADDSS xmm0, xmm1, xmm2, xmm3	C4E3 WvvvvL01 6A /r /is4	Fused Multiply-Add of Scalar Single-Precision Floating-Point Values	
VFMADDSUBPD xmm0, xmm1, xmm2, xmm3	C4E3 WvvvvL01 5D /r /is4	Fused Multiply-Alternating Add/Subtract of Packed Double-Precision Floating-Point Values	
VFMADDSUBPS xmm0, xmm1, xmm2, xmm3	C4E3 WvvvvL01 5C /r /is4	Fused Multiply-Alternating Add/Subtract of Packed Single-Precision Floating-Point Values	
VFMSUBADDPD xmm0, xmm1, xmm2, xmm3	C4E3 WvvvvL01 5F /r /is4	Fused Multiply-Alternating Subtract/Add of Packed Double-Precision Floating-Point Values	
VFMSUBADDPS xmm0, xmm1, xmm2, xmm3	C4E3 WvvvvL01 5E /r /is4	Fused Multiply-Alternating Subtract/Add of Packed Single-Precision Floating-Point Values	
VFMSUBPD xmm0, xmm1, xmm2, xmm3	C4E3 WvvvvL01 6D /r /is4	Fused Multiply-Subtract of Packed Double-Precision Floating-Point Values	
VFMSUBPS xmm0, xmm1, xmm2, xmm3	C4E3 WvvvvL01 6C /r /is4	Fused Multiply-Subtract of Packed Single-Precision Floating-Point Values	
VFMSUBSD xmm0, xmm1, xmm2, xmm3	C4E3 WvvvvL01 6F /r /is4	Fused Multiply-Subtract of Scalar Double-Precision Floating-Point Values	
VFMSUBSS xmm0, xmm1, xmm2, xmm3	C4E3 WvvvvL01 6E /r /is4	Fused Multiply-Subtract of Scalar Single-Precision Floating-Point Values	
VFNMADDPD xmm0, xmm1, xmm2, xmm3	C4E3 WvvvvL01 79 /r /is4	Fused Negative Multiply-Add of Packed Double-Precision Floating-Point Values	
VFNMADDPS xmm0, xmm1, xmm2, xmm3	C4E3 WvvvvL01 78 /r /is4	Fused Negative Multiply-Add of Packed Single-Precision Floating-Point Values	

VFNMADDSD xmm0, xmm1, xmm2, xmm3	C4E3 WvvvvL01 7B /r /is4	Fused Negative Multiply-Add of Scalar Double-Precision Floating-Point Values	
VFNMADDSS xmm0, xmm1, xmm2, xmm3	C4E3 WvvvvL01 7A /r /is4	Fused Negative Multiply-Add of Scalar Single-Precision Floating-Point Values	
VFNMSUBPD xmm0, xmm1, xmm2, xmm3	C4E3 WvvvvL01 7D /r /is4	Fused Negative Multiply-Subtract of Packed Double-Precision Floating-Point Values	
VFNMSUBPS xmm0, xmm1, xmm2, xmm3	C4E3 WvvvvL01 7C /r /is4	Fused Negative Multiply-Subtract of Packed Single-Precision Floating-Point Values	
VFNMSUBSD xmm0, xmm1, xmm2, xmm3	C4E3 WvvvvL01 7F /r /is4	Fused Negative Multiply-Subtract of Scalar Double-Precision Floating-Point Values	
VFNMSUBSS xmm0, xmm1, xmm2, xmm3	C4E3 WvvvvL01 7E /r /is4	Fused Negative Multiply-Subtract of Scalar Single-Precision Floating-Point Values	

Intel AES instructions

6 new instructions.

Instruction	Description
AESENC	Perform one round of an AES encryption flow
AESENCLAST	Perform the last round of an AES encryption flow
AESDEC	Perform one round of an AES decryption flow
AESDECLAST	Perform the last round of an AES decryption flow
AESKEYGENASSIST	Assist in AES round key generation
AESIMC	Assist in AES Inverse Mix Columns

Undocumented instructions

The x86 CPUs contain undocumented instructions which are implemented on the chips but not listed in some official documents. They can be found in various sources across the Internet, such as Ralf Brown's Interrupt List and at http://sandpile.org.

mnemonic	opcode	description	undoc status
AAM imm8	D4 imm8	Divide AL by imm8, put the quotient in AH, and the remainder in AL	Available beginning with 8086, documented since Pentium (earlier documentation lists no arguments)
AAD imm8	D5 imm8	Multiplication counterpart of AAM	Available beginning with 8086, documented since Pentium (earlier documentation lists no arguments)
SALC	D6	Set AL depending on the value of the Carry Flag (a 1-byte alternative of SBB AL, AL)	Available beginning with 8086, but only documented since Pentium Pro.
HCF	F0 0F C7 C8	**H**alt and **C**atch **F**ire - Causes the CPU to lock, forcing the user to hard-reboot.	This was considered a bug by Intel and has been fixed in Pentium Pro step myB2 and later processors.[2]
UD1	0F B9	Intentionally undefined instruction, but unlike UD2 this was not published	
ICEBP	F1	Single byte single-step exception / Invoke ICE	Available beginning with 80386, documented (as INT1) since Pentium Pro
LOADALL	0F 05	Loads All Registers from Memory Address 0x000800H	Only available on 80286

Unknown opcode	0F 04	Exact purpose unknown, causes CPU hang. (the only way out is CPU reset)[3] In some implementations, emulated through BIOS as a halting sequence.[4]	Only available on 80286
LOADALLD	0F 07	Loads All Registers from Memory Address ES:EDI	Only available on 80386
POP CS	0F	Pop top of the stack into CS Segment register (causing a far jump)	Only available on earliest models of 8086. Beginning with 80286 this opcode is used as a prefix for 2-Byte-Instructions
MOV CS,r/m	8E/1	Moves a value from register/memory into CS Segment register (causing a far jump)	Only available on earliest models of 8086. Beginning with 80286 this opcode causes an invalid opcode exception
MOV ES,r/m	8E/4	Moves a value from register/memory into ES segment register	Only available on earliest models of 8086. On 80286 this opcode causes an invalid opcode exception. Beginning with 80386 the value is moved into the FS segment register.
MOV CS,r/m	8E/5	Pop top of the stack into CS Segment register (?)	Only available on earliest models of 8086. On 80286 this opcode causes an invalid opcode exception. Beginning with 80386 the value is moved into the GS segment register.
MOV SS,r/m	8E/6	Moves a value from register/memory into SS Segment register	Only available on earliest models of 8086. Beginning with 80286 this opcode causes an invalid opcode exception
MOV DS,r/m	8E/7	Moves a value from register/memory into DS Segment register	Only available on earliest models of 8086. Beginning with 80286 this opcode causes an invalid opcode exception

References

[1] http://www.softeng.rl.ac.uk/st/archive/SoftEng/SESP/html/SoftwareTools/vtune/users_guide/mergedProjects/analyzer_ec/mergedProjects/reference_olh/mergedProjects/instructions/instruct32_hh/vc279.htm

[2] (PDF) *Pentium Processor Specification Update* (http://www.biblio.deis.unibo.it/Testi_Liberi/Pentium/24248041.PDF). Intel Corporation. 1999-01. pp. 51–52. order number 242480-041. . Retrieved 2006-11-02.

[3] "Re: Undocumented opcodes (HINT_NOP)" (http://www.sandpile.org/post/msgs/20004129.htm). . Retrieved 2010-11-07.

[4] "Re: Also some undocumented 0Fh opcodes" (http://www.sandpile.org/post/msgs/20003986.htm). . Retrieved 2010-11-07.

- Intel Software Developer's Manuals (http://www.intel.com/products/processor/manuals/)

External links

- The 8086 / 80286 / 80386 / 80486 Instruction Set (http://home.comcast.net/~fbui/intel.html)
- Free IA-32 and x86-64 documentation (http://www.intel.com/products/processor/manuals/index.htm), provided by Intel
- Netwide Assembler Instruction List (http://www.nasm.us/doc/nasmdocb.html) (from Netwide Assembler)
- x86 Instruction Set Reference (http://siyobik.info/index.php?module=x86)
- X86 Opcode and Instruction Reference (http://ref.x86asm.net)

Article Sources and Contributors

x86 instruction listings *Source*: http://en.wikipedia.org/w/index.php?oldid=422983474 *Contributors*: A5b, Abdull, Ablonus, Abu badali, AlbertCahalan, Alinor, AlphaPyro, Anakin101, Andreas Rejbrand, Andreyvul, Android Mouse, Anon2, Arndbergmann, Balabiot, Bbbl67, Binrapt, C xong, Chicago god, Cubbi, Cumbyjoe, Cwolfsheep, DanniDK, Dapete, David.kaplan, Doktorspin, Dtrebbien, Einsidler, Elkman, FatalError, Feezo, Frap, Fresheneesz, Gerbrant, Gormunkul, Guy Harris, Honnza, Interiot, JVz, Jengelh, Jeword, JonHarder, Josephycc, Jsmethers, JulesH, Levin, Markhobley, MazeGen, Mconnew, Mellum, Mikeblas, Mrand, Mulad, Naelphin, Neitsa, Nickptar, Notlobbo, Offerman, Panfider, Petrb, Phatom87, Plaes, Qed, Quuxplusone, RCX, Rettetast, Richardcavell, Rik G., Rjwilmsi, RokerHRO, Shawnchin, Sikon, Skiselev, Suruena, TheProject, Theultramage, Thisara.d.m, Tim Shepard, Toby.E.Hawkins, Toobaz, Topsfield99, Tweenk, Uncle G, Versageek, WereSpielChequers, Widefox, Ybungalobill, Yuhong, Zuxy, 162 anonymous edits

License