Intel x86 Assembler Instruction Set Opcode Table

ADD Eb Gb *00*	**ADD** Ev Gv *01*	**ADD** Gb Eb *02*	**ADD** Gv Ev *03*	**ADD** AL Ib *04*	**ADD** eAX Iv *05*	**PUSH** ES *06*	**POP** ES *07*	**OR** Eb Gb *08*	**OR** Ev Gv *09*	**OR** Gb Eb *0A*	**OR** Gv Ev *0B*	**OR** AL Ib *0C*	**OR** eAX Iv *0D*	**PUSH** CS *0E*	**TWOBYTE** *0F*
ADC Eb Gb *10*	**ADC** Ev Gv *11*	**ADC** Gb Eb *12*	**ADC** Gv Ev *13*	**ADC** AL Ib *14*	**ADC** eAX Iv *15*	**PUSH** SS *16*	**POP** SS *17*	**SBB** Eb Gb *18*	**SBB** Ev Gv *19*	**SBB** Gb Eb *1A*	**SBB** Gv Ev *1B*	**SBB** AL Ib *1C*	**SBB** eAX Iv *1D*	**PUSH** DS *1E*	**POP** DS *1F*
AND Eb Gb *20*	**AND** Ev Gv *21*	**AND** Gb Eb *22*	**AND** Gv Ev *23*	**AND** AL Ib *24*	**AND** eAX Iv *25*	**ES:** *26*	**DAA** *27*	**SUB** Eb Gb *28*	**SUB** Ev Gv *29*	**SUB** Gb Eb *2A*	**SUB** Gv Ev *2B*	**SUB** AL Ib *2C*	**SUB** eAX Iv *2D*	**CS:** *2E*	**DAS** *2F*
XOR Eb Gb *30*	**XOR** Ev Gv *31*	**XOR** Gb Eb *32*	**XOR** Gv Ev *33*	**XOR** AL Ib *34*	**XOR** eAX Iv *35*	**SS:** *36*	**AAA** *37*	**CMP** Eb Gb *38*	**CMP** Ev Gv *39*	**CMP** Gb Eb *3A*	**CMP** Gv Ev *3B*	**CMP** AL Ib *3C*	**CMP** eAX Iv *3D*	**DS:** *3E*	**AAS** *3F*
INC eAX *40*	**INC** eCX *41*	**INC** eDX *42*	**INC** eBX *43*	**INC** eSP *44*	**INC** eBP *45*	**INC** eSI *46*	**INC** eDI *47*	**DEC** eAX *48*	**DEC** eCX *49*	**DEC** eDX *4A*	**DEC** eBX *4B*	**DEC** eSP *4C*	**DEC** eBP *4D*	**DEC** eSI *4E*	**DEC** eDI *4F*
PUSH eAX *50*	**PUSH** eCX *51*	**PUSH** eDX *52*	**PUSH** eBX *53*	**PUSH** eSP *54*	**PUSH** eBP *55*	**PUSH** eSI *56*	**PUSH** eDI *57*	**POP** eAX *58*	**POP** eCX *59*	**POP** eDX *5A*	**POP** eBX *5B*	**POP** eSP *5C*	**POP** eBP *5D*	**POP** eSI *5E*	**POP** eDI *5F*
PUSHA *60*	**POPA** *61*	**BOUND** Gv Ma *62*	**ARPL** Ew Gw *63*	**FS:** *64*	**GS:** *65*	**OPSIZE:** *66*	**ADSIZE:** *67*	**PUSH** Iv *68*	**IMUL** Gv Ev Iv *69*	**PUSH** Ib *6A*	**IMUL** Gv Ev Ib *6B*	**INSB** Yb DX *6C*	**INSW** Yz DX *6D*	**OUTSB** DX Xb *6E*	**OUTSW** DX Xv *6F*
JO Jb *70*	**JNO** Jb *71*	**JB** Jb *72*	**JNB** Jb *73*	**JZ** Jb *74*	**JNZ** Jb *75*	**JBE** Jb *76*	**JA** Jb *77*	**JS** Jb *78*	**JNS** Jb *79*	**JP** Jb *7A*	**JNP** Jb *7B*	**JL** Jb *7C*	**JNL** Jb *7D*	**JLE** Jb *7E*	**JNLE** Jb *7F*
ADD Eb Ib *80*	**ADD** Ev Iv *81*	**SUB** Eb Ib *82*	**SUB** Ev Ib *83*	**TEST** Eb Gb *84*	**TEST** Ev Gv *85*	**XCHG** Eb Gb *86*	**XCHG** Ev Gv *87*	**MOV** Eb Gb *88*	**MOV** Ev Gv *89*	**MOV** Gb Eb *8A*	**MOV** Gv Ev *8B*	**MOV** Ew Sw *8C*	**LEA** Gv M *8D*	**MOV** Sw Ew *8E*	**POP** Ev *8F*
NOP *90*	**XCHG** eAX eCX *91*	**XCHG** eAX eDX *92*	**XCHG** eAX eBX *93*	**XCHG** eAX eSP *94*	**XCHG** eAX eBP *95*	**XCHG** eAX eSI *96*	**XCHG** eAX eDI *97*	**CBW** *98*	**CWD** *99*	**CALL** Ap *9A*	**WAIT** *9B*	**PUSHF** Fv *9C*	**POPF** Fv *9D*	**SAHF** *9E*	**LAHF** *9F*
MOV AL Ob *A0*	**MOV** eAX Ov *A1*	**MOV** Ob AL *A2*	**MOV** Ov eAX *A3*	**MOVSB** Xb Yb *A4*	**MOVSW** Xv Yv *A5*	**CMPSB** Xb Yb *A6*	**CMPSW** Xv Yv *A7*	**TEST** AL Ib *A8*	**TEST** eAX Iv *A9*	**STOSB** Yb AL *AA*	**STOSW** Yv eAX *AB*	**LODSB** AL Xb *AC*	**LODSW** eAX Xv *AD*	**SCASB** AL Yb *AE*	**SCASW** eAX Yv *AF*
MOV AL Ib *B0*	**MOV** CL Ib *B1*	**MOV** DL Ib *B2*	**MOV** BL Ib *B3*	**MOV** AH Ib *B4*	**MOV** CH Ib *B5*	**MOV** DH Ib *B6*	**MOV** BH Ib *B7*	**MOV** eAX Iv *B8*	**MOV** eCX Iv *B9*	**MOV** eDX Iv *BA*	**MOV** eBX Iv *BB*	**MOV** eSP Iv *BC*	**MOV** eBP Iv *BD*	**MOV** eSI Iv *BE*	**MOV** eDI Iv *BF*
#2 Eb Ib *C0*	**#2** Ev Ib *C1*	**RETN** Iw *C2*	**RETN** *C3*	**LES** Gv Mp *C4*	**LDS** Gv Mp *C5*	**MOV** Eb Ib *C6*	**MOV** Ev Iv *C7*	**ENTER** Iw Ib *C8*	**LEAVE** *C9*	**RETF** Iw *CA*	**RETF** *CB*	**INT3** *CC*	**INT** Ib *CD*	**INTO** *CE*	**IRET** *CF*
#2 Eb 1 *D0*	**#2** Ev 1 *D1*	**#2** Eb CL *D2*	**#2** Ev CL *D3*	**AAM** Ib *D4*	**AAD** Ib *D5*	**SALC** *D6*	**XLAT** *D7*	**ESC** 0 *D8*	**ESC** 1 *D9*	**ESC** 2 *DA*	**ESC** 3 *DB*	**ESC** 4 *DC*	**ESC** 5 *DD*	**ESC** 6 *DE*	**ESC** 7 *DF*
LOOPNZ Jb *E0*	**LOOPZ** Jb *E1*	**LOOP** Jb *E2*	**JCXZ** Jb *E3*	**IN** AL Ib *E4*	**IN** eAX Ib *E5*	**OUT** Ib AL *E6*	**OUT** Ib eAX *E7*	**CALL** Jz *E8*	**JMP** Jz *E9*	**JMP** Ap *EA*	**JMP** Jb *EB*	**IN** AL DX *EC*	**IN** eAX DX *ED*	**OUT** DX AL *EE*	**OUT** DX eAX *EF*
LOCK: *F0*	**INT1** *F1*	**REPNE:** *F2*	**REP:** *F3*	**HLT** *F4*	**CMC** *F5*	**#3** Eb *F6*	**#3** Ev *F7*	**CLC** *F8*	**STC** *F9*	**CLI** *FA*	**STI** *FB*	**CLD** *FC*	**STD** *FD*	**#4** INC/DEC *FE*	**#5** INC/DEC *FF*

Legend
HAS MOD R/M
LENGTH = 1
OTHER
UNDECODED

80386 Instruction Format

Prefix

INSTRUCTION PREFIX	ADDRESS SIZE PREFIX	OPERAND SIZE PREFIX	SEGMENT OVERRIDE
0 OR 1	0 OR 1	0 OR 1	0 OR 1

NUMBER OF BYTES

Required

OPCODE	MOD R/M	SIB	DISPLACEMENT	IMMEDIATE
1 OR 2	0 OR 1	0 OR 1	0,1,2 OR 4	0,1,2 OR 4
NUMBER OF BYTES				

MOD R/M BYTE

7	6	5	4	3	2	1	0
MOD		REG/OPCODE			R/M		

SIB BYTE

7	6	5	4	3	2	1	0
SCALE		INDEX			BASE		

MOD R/M 16

	0	1	2	3	4	5	6	7
0	[BX+SI] +1	[BX+DI] +1	[BP+SI] +1	[BP+DI] +1	[SI] +1	[DI] +1	[Iw] +3	[BX] +1
1	[BX+SI+Ib] +2	[BX+DI+Ib] +2	[BP+SI+Ib] +2	[BP+DI+Ib] +2	[SI+Ib] +2	[DI+Ib] +2	[BP+Ib] +2	[BX+Ib] +2
2	[BX+SI+Iw] +3	[BX+DI+Iw] +3	[BP+SI+Iw] +3	[BP+DI+Iw] +3	[SI+Iw] +3	[DI+Iw] +3	[BP+Iw] +3	[BX+Iw] +3
3	AX +1	CX +1	DX +1	BX +1	SP +1	BP +1	SI +1	DI +1

MOD R/M 32

	0	1	2	3	4	5	6	7
0	[eAX] +1	[eCX] +1	[eDX] +1	[eBX] +1	[SIB] +2	[Iv] +5	[eSI] +1	[eDI] +1
1	[eAX+Ib] +2	[eCX+Ib] +2	[eDX+Ib] +2	[eBX+Ib] +2	[SIB+Ib] +2	[eBP+Ib] +2	[eSI+Ib] +2	[eDI+Ib] +2
2	[eAX+Iv] +5	[eCX+Iv] +5	[eDX+Iv] +5	[eBX+Iv] +5	[SIB+Iv] +5	[eBP+Iv] +5	[eSI+Iv] +5	[eDI+Iv] +5
3	eAX +1	eCX +1	eDX +1	eBX +1	eSP +1	eBP +1	eSI +1	eDI +1

REGISTERS

	0	1	2	3	4	5	6	7
Reg 8	AL	CL	DL	BL	AH	CH	DH	BH
Reg 16	AX	CX	DX	BX	SP	BP	SI	DI
Reg 32	eAX	eCX	eDX	eBX	eSP	eBP	eSI	eDI
Segments	DS	ES	FS	GS	SS	CS	IP	

Addressing Method Codes

A	Direct address. The instruction has no ModR/M byte; the address of the operand is encoded in the instruction; and no base register, index register, or scaling factor can be applied (for example, far JMP (EA)).

C	The reg field of the ModR/M byte selects a control register (for example, MOV (0F20, 0F22)).
D	The reg field of the ModR/M byte selects a debug register (for example, MOV (0F21,0F23)).
E	A ModR/M byte follows the opcode and specifies the operand. The operand is either a general-purpose register or a memory address. If it is a memory address, the address is computed from a segment register and any of the following values: a base register, an index register, a scaling factor, a displacement.
F	EFLAGS Register.
G	The reg field of the ModR/M byte selects a general register (for example, AX (000)).
I	Immediate data. The operand value is encoded in subsequent bytes of the instruction.
J	The instruction contains a relative offset to be added to the instruction pointer register (for example, JMP (0E9), LOOP).
M	The ModR/M byte may refer only to memory (for example, BOUND, LES, LDS, LSS, LFS, LGS, CMPXCHG8B).
O	The instruction has no ModR/M byte; the offset of the operand is coded as a word or double word (depending on address size attribute) in the instruction. No base register, index register, or scaling factor can be applied (for example, MOV (A0–A3)).
P	The reg field of the ModR/M byte selects a packed quadword MMX™ technology register.
Q	A ModR/M byte follows the opcode and specifies the operand. The operand is either an MMX™ technology register or a memory address. If it is a memory address, the address is computed from a segment register and any of the following values: a base register,an index register, a scaling factor, and a displacement.
R	The mod field of the ModR/M byte may refer only to a general register (for example, MOV (0F20-0F24, 0F26)).
S	The reg field of the ModR/M byte selects a segment register (for example, MOV (8C,8E)).
T	The reg field of the ModR/M byte selects a test register (for example, MOV (0F24,0F26)).
V	The reg field of the ModR/M byte selects a packed SIMD floating-point register.
W	An ModR/M byte follows the opcode and specifies the operand. The operand is either a SIMD floating-point register or a memory address. If it is a memory address, the address is computed from a segment register and any of the following values: a base register, an index register, a scaling factor, and a displacement
X	Memory addressed by the DS:SI register pair (for example, MOVS, CMPS, OUTS, or LODS).
Y	Memory addressed by the ES:DI register pair (for example, MOVS, CMPS, INS, STOS, or SCAS).

Operand Type Codes

a	Two one-word operands in memory or two double-word operands in memory, depending on operand-size attribute (used only by the BOUND instruction).
b	Byte, regardless of operand-size attribute.
c	Byte or word, depending on operand-size attribute.
d	Doubleword, regardless of operand-size attribute
dq	Double-quadword, regardless of operand-size attribute.
p	32-bit or 48-bit pointer, depending on operand-size attribute.

pi	Quadword MMX™ technology register (e.g. mm0)
ps	128-bit packed FP single-precision data.
q	Quadword, regardless of operand-size attribute.
s	6-byte pseudo-descriptor.
ss	Scalar element of a 128-bit packed FP single-precision data.
si	Doubleword integer register (e.g., eax)
v	Word or doubleword, depending on operand-size attribute.
w	Word, regardless of operand-size attribute.