8086 Instruction Set Opcodes

Operation	Operands	Opcode
ADC	see ADD	ADD opcode + $10, and xx010xxx (ModR/M byte) for $80-$83
ADD	r/m8, reg8	$00
ADD	r/m16, reg16	$01
ADD	reg8, r/m8	$02
ADD	reg16, r/m16	$03
ADD	AL, imm8	$04
ADD	AX, imm16	$05
ADD	r/m8, imm8	$80 xx000xxx (ModR/M byte)
ADD	r/m16, imm16	$81 xx000xxx (ModR/M byte)
ADD	r/m16, imm8	$83 xx000xxx (ModR/M byte)
AND	see ADD	ADD opcode + $20, and xx100xxx (ModR/M byte) for $80, $81,$83
CALL	32-bit displacement	$9A
CALL	16-bit displacement	$E8
CLD		$FC
CMP	See ADD	ADD opcode + $38, and xx111xxx (ModR/M byte) for $80, $81,$83
CMPSB	ES:[DI]==DS:[SI]	$A6
CMPW	ES:[DI]==DS:[SI]	$A7
DEC	r/m8	$FE, xx001xxx (ModR/M byte)
DEC	r/m16	$FF, xx001xxx (ModR/M byte)
DEC	reg16	$48 + reg16 code
DIV	r/m8	$F6, xx110xxx (ModR/M byte)
DIV	r/m16	$F7, xx110xxx (ModR/M byte)
HLT		$F4
IDIV	r/m8	$F6, xx111xxx (ModR/M byte)
IDIV	r/m16	$F7, xx111xxx (ModR/M byte)
IMUL	r/m8	$F6, xx101xxx (ModR/M byte)
IMUL	r/m16	$F7, xx101xxx (ModR/M byte)
IN	AL, addr8	$E4
IN	AX, addr8	$E5
IN	AL, port[DX]	$EC
IN	AX, port[DX]	$ED
INC	r/m8	$FE, xx000xxx (ModR/M byte)
INC	r/m16	$FF, xx000xxx (ModR/M byte)
INC	reg16	$40 + reg16 code
IRET	48-bit POP	$CF
JA	8-bit relative	$77
JAE	8-bit relative	$73
JB	8-bit relative	$72
JBE	8-bit relative	$76
JE	8-bit relative	$74
JG	8-bit relative	$7F
JGE	8-bit relative	$7D
JL	8-bit relative	$7C
JLE	8-bit relative	$7E
JMP	32-bit displacement	$EA
JNE	8-bit relative	$75
JZ	8-bit relative	$74
LDS	reg16, mem32	$C4
LES	reg16, mem32	$C5
LODSB	AL = DS:[SI]	$AC
LODSW	AX = DS:[SI]	$AD

LOOP	8-bit relative	$E2
MOV	r/m8, reg8	$88
MOV	r/m16, reg16	$89
MOV	AL, mem8	$A0
MOV	AX, mem16	$A1
MOV	mem8, AL	$A2
MOV	mem16, AX	$A3
MOV	reg8, imm8	$B0 + reg8 code
MOV	reg16,imm16	$B8 + reg16 code
MOV	r/m8, imm8	$C6, xx000xxx(ModR/M byte)
MOV	r/m16, imm16	$C7, xx000xxx(ModR/M byte)
MOV	r/m16,sreg	$8C, xx0 sreg xxx(ModR/M byte)
MOV	sreg, r/m16	$8E, xx0 sreg xxx(ModR/M byte)
MOVSB	ES:[DI] = DS:[SI]	$A4
MOVSW	ES:[DI] = DS:[SI]	$A5
MUL	r/m8	$F6, xx100xxx (ModR/M byte)
MUL	r/m16	$F7, xx100xxx (ModR/M byte)
NEG	r/m8	$F6, xx011xxx (ModR/M byte)
NEG	r/m16	$F7, xx011xxx (ModR/M byte)
NOT	r/m8	$F6, xx010xxx (ModR/M byte)
NOT	r/m16	$F7, xx010xxx (ModR/M byte)
OR	see ADD	ADD opcode + $08, and xx001xxx (ModR/M byte) for $80, $81,$83
OUT	addr8, AL	$E6
OUT	addr8, AX	$E7
OUT	port[DX], AL	$EE
OUT	port[DX], AX	$EF
POP	r/m16	$8F
POP	reg16	$58 + reg16 code
POP	sreg	$07 + ES = 0, CS = 8, SS = $10, DS = $18
PUSH	r/m16	$FF, xx110xxx (ModR/M byte)
PUSH	reg16	$50 + reg16 code
PUSH	sreg	$06 + ES = 0, CS = 8, SS = $10, DS = $18
REP		$F3
REPNE		$F2
RET	32-bit POP	$CA
RET	16-bit POP	$C2
SBB	see ADD	ADD opcode + $18, and xx011xxx (ModR/M byte) for $80, $81,$83
SCASB	ES:[DI] == AL	$AE
SCASW	ES:[DI] == AX	$AF
STD		$FD
STOSB	ES:[DI] = AL	$AA
STOSW	ES:[DI] = AX	$AB
SUB	see ADD	ADD opcode + $28, and xx101xxx (ModR/M byte) for $80, $81,$83
XOR	see ADD	ADD opcode + $30, and xx110xxx (ModR/M byte) for $80, $81,$83

addr8 = 8-bit address of I/O port
reg8 = AL = 0, CL = 1, DL = 2, BL = 3, AH =4, CH = 5, DH = 6, BH = 7
reg16 = AX = 0, CX =1, DX =2, BX =3, SP = 4, BP = 5, SI = 6, DI = 7
sreg = ES = 0, CS = 1, SS = 2, DS = 3
mem8 = memory byte (direct addressing only)
mem16 = memory word (direct addressing only)
r/m8 = reg8 or mem8
r/m16 = reg16 or mem16
imm8 = 8 bit immediate
imm16 = 16 bit immediate