Figure 3-71 Machine code for the 8086/8088 instructions. (Reprinted by permission of Intel Corporation. Copyright 1979.)

AL = 8-bit accumulator
AX = 16-bit accumulator
CX = Count register
DS = Data segment
ES = Extra segment
Above/below refers to unsigned value
Greater = more positive;
Less = less positive (more negative) signed values
if d = 1 then "to" reg; if d = 0 then "from" reg
if w = 1 then word instruction; if w = 0 then byte instruction

if s:w = 01 then 16 bits of immediate data form the operand.
if s:w = 11 then an immediate data byte is sign extended to form the 16-bit operand.
if v = 0 then "count" = 1; if v = 1 then "count" in (CL)
x = don't care
z is used for string primitives for comparison with Z-F FLAG

SEGMENT OVERRIDE PREFIX

0 0 1 reg 1 1 0

if mod = 11 then r/m is treated as a REG field
if mod = 00 then DISP = 0*, disp-low and disp-high are absent
if mod = 01 then DISP = disp-low sign-extended to 16-bits, disp-high is absent
if mod = 10 then DISP = disp-high: disp-low

if r/m = 000 then EA = (BX) + (SI) + DISP
if r/m = 001 then EA = (BX) + (DI) + DISP
if r/m = 010 then EA = (BP) + (SI) + DISP
if r/m = 011 then EA = (BP) + (DI) + DISP
if r/m = 100 then EA = (SI) + DISP
if r/m = 101 then EA = (DI) + DISP
if r/m = 110 then EA = (BP) + DISP*
if r/m = 111 then EA = (BX) + DISP

DISP follows 2nd byte of instruction (before data if required)

*except if mod = 00 and r/m = 110 then EA = disp-high: disp-low.

REG is assigned according to the following table:

16-Bit (w = 1)	8-Bit (w = 0)	Segment
000 AX	000 AL	00 ES
001 CX	001 CL	01 CS
010 DX	010 DL	10 SS
011 BX	011 BL	11 DS
100 SP	100 AH	
101 BP	101 CH	
110 SI	110 DH	
111 DI	111 BH	

Instructions which reference the flag register file as a 16-bit object use the symbol FLAGS to represent the file:

FLAGS = X:X:X:X:(OF):(DF):(IF):(TF):(SF):(ZF):X:(AF):X:(PF):X:(CF)

8086/8088 Instruction Encoding

DATA TRANSFER

MOV = Move:

	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
Register/memory to/from register	1 0 0 0 1 0 d w	mod reg r/m	(DISP-LO)	(DISP-HI)		
Immediate to register/memory	1 1 0 0 0 1 1 w	mod 0 0 0 r/m	(DISP-LO)	(DISP-HI)	data	data if w = 1
Immediate to register	1 0 1 1 w reg	data	data if w = 1			
Memory to accumulator	1 0 1 0 0 0 0 w	addr-lo	addr-hi			
Accumulator to memory	1 0 1 0 0 0 1 w	addr-lo	addr-hi			
Register/memory to segment register	1 0 0 0 1 1 1 0	mod 0 SR r/m	(DISP-LO)	(DISP-HI)		
Segment register to register/memory	1 0 0 0 1 1 0 0	mod 0 SR r/m	(DISP-LO)	(DISP-HI)		

PUSH = Push:

Register/memory	1 1 1 1 1 1 1 1	mod 1 1 0 r/m	(DISP-LO)	(DISP-HI)
Register	0 1 0 1 0 reg			
Segment register	0 0 0 reg 1 1 0			

POP = Pop:

Register/memory	1 0 0 0 1 1 1 1	mod 0 0 0 r/m	(DISP-LO)	(DISP-HI)
Register	0 1 0 1 1 reg			
Segment register	0 0 0 reg 1 1 1			

DATA TRANSFER (Cont'd.)

	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
XCHG = Exchange:						
Register/memory with register	1 0 0 0 0 1 1 w	mod reg r/m	(DISP-LO)	(DISP-HI)		
Register with accumulator	1 0 0 1 0 reg					
IN = Input from:						
Fixed port	1 1 1 0 0 1 0 w	DATA-8				
Variable port	1 1 1 0 1 1 0 w					
OUT = Output to:						
Fixed port	1 1 1 0 0 1 1 w	DATA-8				
Variable port	1 1 1 0 1 1 1 w					
XLAT = Translate byte to AL	1 1 0 1 0 1 1 1					
LEA = Load EA to register	1 0 0 0 1 1 0 1	mod reg r/m	(DISP-LO)	(DISP-HI)		
LDS = Load pointer to DS	1 1 0 0 0 1 0 1	mod reg r/m	(DISP-LO)	(DISP-HI)		
LES = Load pointer to ES	1 1 0 0 0 1 0 0	mod reg r/m	(DISP-LO)	(DISP-HI)		
LAHF = Load AH with flags	1 0 0 1 1 1 1 1					
SAHF = Store AH into flags	1 0 0 1 1 1 1 0					
PUSHF = Push flags	1 0 0 1 1 1 0 0					
POPF = Pop flags	1 0 0 1 1 1 0 1					

ARITHMETIC

ADD = Add:						
Reg/memory with register to either	0 0 0 0 0 0 d w	mod reg r/m	(DISP-LO)	(DISP-HI)		
Immediate to register/memory	1 0 0 0 0 0 s w	mod 0 0 0 r/m	(DISP-LO)	(DISP-HI)	data	data if s: w=01
Immediate to accumulator	0 0 0 0 0 1 0 w	data	data if w=1			
ADC = Add with carry:						
Reg/memory with register to either	0 0 0 1 0 0 d w	mod reg r/m	(DISP-LO)	(DISP-HI)		
Immediate to register/memory	1 0 0 0 0 0 s w	mod 0 1 0 r/m	(DISP-LO)	(DISP-HI)	data	data if s: w=01
Immediate to accumulator	0 0 0 1 0 1 0 w	data	data if w=1			
INC = Increment:						
Register/memory	1 1 1 1 1 1 1 w	mod 0 0 0 r/m	(DISP-LO)	(DISP-HI)		
Register	0 1 0 0 0 reg					
AAA = ASCII adjust for add	0 0 1 1 0 1 1 1					
DAA = Decimal adjust for add	0 0 1 0 0 1 1 1					

ARITHMETIC (Cont'd.)

	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
SUB = Subtract:						
Reg/memory and register to either	0 0 1 0 1 0 d w	mod reg r/m	(DISP-LO)	(DISP-HI)		
Immediate from register/memory	1 0 0 0 0 0 s w	mod 1 0 1 r/m	(DISP-LO)	(DISP-HI)	data	data if s: w=01
Immediate from accumulator	0 0 1 0 1 1 0 w	data	data if w=1			
SBB = Subtract with borrow:						
Reg/memory and register to either	0 0 0 1 1 0 d w	mod reg r/m	(DISP-LO)	(DISP-HI)		
Immediate from register/memory	1 0 0 0 0 0 s w	mod 0 1 1 r/m	(DISP-LO)	(DISP-HI)	data	data if s: w=01
Immediate from accumulator	0 0 0 1 1 1 0 w	data	data if w=1			
DEC Decrement:						
Register/memory	1 1 1 1 1 1 1 w	mod 0 0 1 r/m	(DISP-LO)	(DISP-HI)		
Register	0 1 0 0 1 reg					
NEG Change sign	1 1 1 1 0 1 1 w	mod 0 1 1 r/m	(DISP-LO)	(DISP-HI)		
CMP = Compare:						
Register/memory and register	0 0 1 1 1 0 d w	mod reg r/m	(DISP-LO)	(DISP-HI)		
Immediate with register/memory	1 0 0 0 0 0 s w	mod 1 1 1 r/m	(DISP-LO)	(DISP-HI)	data	data if s: w=1
Immediate with accumulator	0 0 1 1 1 1 0 w	data				
AAS ASCII adjust for subtract	0 0 1 1 1 1 1 1					
DAS Decimal adjust for subtract	0 0 1 0 1 1 1 1					
MUL Multiply (unsigned)	1 1 1 1 0 1 1 w	mod 1 0 0 r/m	(DISP-LO)	(DISP-HI)		
IMUL Integer multiply (signed)	1 1 1 1 0 1 1 w	mod 1 0 1 r/m	(DISP-LO)	(DISP-HI)		
AAM ASCII adjust for multiply	1 1 0 1 0 1 0 0	0 0 0 0 1 0 1 0	(DISP-LO)	(DISP-HI)		
DIV Divide (unsigned)	1 1 1 1 0 1 1 w	mod 1 1 0 r/m	(DISP-LO)	(DISP-HI)		
IDIV Integer divide (signed)	1 1 1 1 0 1 1 w	mod 1 1 1 r/m	(DISP-LO)	(DISP-HI)		
AAD ASCII adjust for divide	1 1 0 1 0 1 0 1	0 0 0 0 1 0 1 0	(DISP-LO)	(DISP-HI)		
CBW Convert byte to word	1 0 0 1 1 0 0 0					
CWD Convert word to double word	1 0 0 1 1 0 0 1					

LOGIC

NOT Invert	1 1 1 1 0 1 1 w	mod 0 1 0 r/m	(DISP-LO)	(DISP-HI)		
SHL/SAL Shift logical/arithmetic left	1 1 0 1 0 0 v w	mod 1 0 0 r/m	(DISP-LO)	(DISP-HI)		
SHR Shift logical right	1 1 0 1 0 0 v w	mod 1 0 1 r/m	(DISP-LO)	(DISP-HI)		
SAR Shift arithmetic right	1 1 0 1 0 0 v w	mod 1 1 1 r/m	(DISP-LO)	(DISP-HI)		
ROL Rotate left	1 1 0 1 0 0 v w	mod 0 0 0 r/m	(DISP-LO)	(DISP-HI)		

LOGIC (Cont'd.)

	76543210	76543210	76543210	76543210	76543210	76543210
ROR Rotate right	110100vw	mod 001 r/m	(DISP-LO)	(DISP-HI)		
RCL Rotate through carry flag left	110100vw	mod 010 r/m	(DISP-LO)	(DISP-HI)		
RCR Rotate through carry right	110100vw	mod 011 r/m	(DISP-LO)	(DISP-HI)		

AND = And:

Reg/memory with register to either	001000dw	mod reg r/m	(DISP-LO)	(DISP-HI)		
Immediate to register/memory	1000000w	mod 100 r/m	(DISP-LO)	(DISP-HI)	data	data if w=1
Immediate to accumulator	0010010w	data	data if w=1			

TEST = And function to flags no result:

Register/memory and register	000100dw	mod reg r/m	(DISP-LO)	(DISP-HI)		
Immediate data and register/memory	1111011w	mod 000 r/m	(DISP-LO)	(DISP-HI)	data	data if w=1
Immediate data and accumulator	1010100w	data				

OR = Or:

Reg/memory and register to either	000010dw	mod reg r/m	(DISP-LO)	(DISP-HI)		
Immediate to register/memory	1000000w	mod 001 r/m	(DISP-LO)	(DISP-HI)	data	data if w=1
Immediate to accumulator	0000110w	data	data if w=1			

XOR = Exclusive or:

Reg/memory and register to either	001100dw	mod reg r/m	(DISP-LO)	(DISP-HI)		
Immediate to register/memory	0011010w	[illegible]	(DISP-LO)	(DISP-HI)	data	data if w=1
Immediate to accumulator	0011010w	data	data if w=1			

1000000W

mod (XXX) r/m
110

STRING MANIPULATION

REP = Repeat	1111001z
MOVS = Move byte/word	1010010w
CMPS = Compare byte/word	1010011w
SCAS = Scan byte/word	1010111w
LODS = Load byte/wd to AL/AX	1010110w
STOS = Stor byte/wd from AL/A	1010101w

CONTROL TRANSFER

CALL = Call:

	76543210	76543210	76543210	76543210	76543210	76543210
Direct within segment	11101000	IP-INC-LO	IP-INC-HI			
Indirect within segment	11111111	mod 0 1 0 r/m	(DISP-LO)	(DISP-HI)		
Direct intersegment	10011010	IP-lo	IP-hi			
		CS-lo	CS-hi			
Indirect intersegment	11111111	mod 0 1 1 r/m	(DISP-LO)	(DISP-HI)		

JMP = Unconditional Jump:

Direct within segment	11101001	IP-INC-LO	IP-INC-HI	
Direct within segment-short	11101011	IP-INC8		
Indirect within segment	11111111	mod 1 0 0 r/m	(DISP-LO)	(DISP-HI)
Direct intersegment	11101010	IP-lo	IP-hi	
		CS-lo	CS-hi	
Indirect intersegment	11111111	mod 1 0 1 r/m	(DISP-LO)	(DISP-HI)

RET = Return from CALL:

Within segment	11000011		
Within seg adding immed to SP	11000010	data-lo	data-hi
Intersegment	11001011		
Intersegment adding immediate to SP	11001010	data-lo	data-hi
JE/JZ = Jump on equal/zero	01110100	IP-INC8	
JL/JNGE = Jump on less/not greater or equal	01111100	IP-INC8	
JLE/JNG = Jump on less or equal/not greater	01111110	IP-INC8	
JB/JNAE = Jump on below/not above or equal	01110010	IP-INC8	
JBE/JNA = Jump on below or equal/not above	01110110	IP-INC8	
JP/JPE = Jump on parity/parity even	01111010	IP-INC8	
JO = Jump on overflow	01110000	IP-INC8	
JS = Jump on sign	01111000	IP-INC8	
JNE/JNZ = Jump on not equal/not zero	01111001	IP-INC8	
JNL/JGE = Jump on not less/greater or equal	01111101	IP-INC8	
JNLE/JG = Jump on not less or equal/greater	01111111	IP-INC8	
JNB/JAE = Jump on not below/above or equal	01110011	IP-INC8	
JNBE/JA = Jump on not below or equal/above	01110111	IP-INC8	
JNP/JPO = Jump on not par/par odd	01111011	IP-INC8	
JNO = Jump on not overflow	01110001	IP-INC8	

CONTROL TRANSFER (Cont'd.)

	76543210	76543210	76543210	76543210	76543210	76543210
RET = Return from CALL:						
JNS = Jump on not sign	01111001	IP-INC8				
LOOP = Loop CX times	11100010	IP-INC8				
LOOPZ/LOOPE = Loop while zero/equal	11100001	IP-INC8				
LOOPNZ/LOOPNE = Loop while not zero/equal	11100000	IP-INC8				
JCXZ = Jump on CX zero	11100011	IP-INC8				

INT = Interrupt:					
Type specified	11001101	DATA-8			
Type 3	11001100				
INTO = Interrupt on overflow	11001110				
IRET = Interrupt return	11001111				

PROCESSOR CONTROL

CLC = Clear carry	11111000			
CMC = Complement carry	11110101			
STC = Set carry	11111001			
CLD = Clear direction	11111100			
STD = Set direction	11111101			
CLI = Clear interrupt	11111010			
STI = Set interrupt	11111011			
HLT = Halt	11110100			
WAIT = Wait	10011011			
ESC = Escape (to external device)	11011xxx	mod yyy r/m	(DISP-LO)	(DISP-HI)
LOCK = Bus lock prefix	11110000			
SEGMENT = Override prefix	001reg110			

would be assembled as

00111101	10100010	00000101

instead of

10000001	11111000	10100010	00000101