Table 4-11. Key to Machine Instruction Encoding and Decoding (Cont'd.)

IDENTIFIER	EXPLANATION
SRC-STR8	Byte string addressed by SI.
DEST-STR16	Word string addressed by DI.
SRC-STR16	Word string addressed by SI.
SHORT-LABEL	Label within ±127 bytes of instruction.
NEAR-PROC	Procedure in current code segment.
FAR-PROC	Procedure in another code segment.
NEAR-LABEL	Label in current code segment but farther than −128 to +127 bytes from instruction.
FAR-LABEL	Label in another code segment.
SOURCE-TABLE	XLAT translation table addressed by BX.
OPCODE	ESC opcode operand.
SOURCE	ESC register or memory operand.

Table 4-12. 8086 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			

Table 4-12. 8086 Instruction Encoding (Cont'd.)

DATA TRANSFER (Cont'd.)

XCHG = Exchange:	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 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					

Table 4-12. 8086 Instruction Encoding (Cont'd.)

ARITHMETIC (Cont'd.)

SUB = Subtract:	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
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)

Table 4-12. 8086 Instruction Encoding (Cont'd.)

LOGIC (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
ROR Rotate right	1 1 0 1 0 0 v w	mod 0 0 1 r/m	(DISP-LO)	(DISP-HI)		
RCL Rotate through carry flag left	1 1 0 1 0 0 v w	mod 0 1 0 r/m	(DISP-LO)	(DISP-HI)		
RCR Rotate through carry right	1 1 0 1 0 0 v w	mod 0 1 1 r/m	(DISP-LO)	(DISP-HI)		
AND = And:						
Reg/memory with register to either	0 0 1 0 0 0 d w	mod reg r/m	(DISP-LO)	(DISP-HI)		
Immediate to register/memory	1 0 0 0 0 0 0 w	mod 1 0 0 r/m	(DISP-LO)	(DISP-HI)	data	data if w=1
Immediate to accumulator	0 0 1 0 0 1 0 w	data	data if w=1			
TEST = And function to flags no result:						
Register/memory and register	0 0 0 1 0 0 d w	mod reg r/m	(DISP-LO)	(DISP-HI)		
Immediate data and register/memory	1 1 1 1 0 1 1 w	mod 0 0 0 r/m	(DISP-LO)	(DISP-HI)	data	data if w=1
Immediate data and accumulator	1 0 1 0 1 0 0 w	data				
OR = Or:						
Reg/memory and register to either	0 0 0 0 1 0 d w	mod reg r/m	(DISP-LO)	(DISP-HI)		
Immediate to register/memory	1 0 0 0 0 0 0 w	mod 0 0 1 r/m	(DISP-LO)	(DISP-HI)	data	data if w=1
Immediate to accumulator	0 0 0 0 1 1 0 w	data	data if w=1			
XOR = Exclusive or:						
Reg/memory and register to either	0 0 1 1 0 0 d w	mod reg r/m	(DISP-LO)	(DISP-HI)		
Immediate to register/memory	0 0 1 1 0 1 0 w	data	(DISP-LO)	(DISP-HI)	data	data if w=1
Immediate to accumulator	0 0 1 1 0 1 0 w	data	data if w=1			
STRING MANIPULATION						
REP=Repeat	1 1 1 1 0 0 1 z					
MOVS=Move byte/word	1 0 1 0 0 1 0 w					
CMPS=Compare byte/word	1 0 1 0 0 1 1 w					
SCAS=Scan byte/word	1 0 1 0 1 1 1 w					
LODS=Load byte/wd to AL/AX	1 0 1 0 1 1 0 w					
STDS=Stor byte/wd from AL/A	1 0 1 0 1 0 1 w					

Table 4-12. 8086 Instruction Encoding (Cont'd.)

CONTROL TRANSFER

CALL = Call:	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
Direct within segment	1 1 1 0 1 0 0 0	IP-INC-LO	IP-INC-HI			
Indirect within segment	1 1 1 1 1 1 1 1	mod 0 1 0 r/m	(DISP-LO)	(DISP-HI)		
Direct intersegment	1 0 0 1 1 0 1 0	IP-lo	IP-hi			
		CS-lo	CS-hi			
Indirect intersegment	1 1 1 1 1 1 1 1	mod 0 1 1 r/m	(DISP-LO)	(DISP-HI)		
JMP = Unconditional Jump:						
Direct within segment	1 1 1 0 1 0 0 1	IP-INC-LO	IP-INC-HI			
Direct within segment-short	1 1 1 0 1 0 1 1	IP-INC8				
Indirect within segment	1 1 1 1 1 1 1 1	mod 1 0 0 r/m	(DISP-LO)	(DISP-HI)		
Direct intersegment	1 1 1 0 1 0 1 0	IP-lo	IP-hi			
		CS-lo	CS-hi			
Indirect intersegment	1 1 1 1 1 1 1 1	mod 1 0 1 r/m	(DISP-LO)	(DISP-HI)		
RET = Return from CALL:						
Within segment	1 1 0 0 0 0 1 1					
Within seg adding immed to SP	1 1 0 0 0 0 1 0	data-lo	data-hi			
Intersegment	1 1 0 0 1 0 1 1					
Intersegment adding immediate to SP	1 1 0 0 1 0 1 0	data-lo	data-hi			
JE/JZ = Jump on equal/zero	0 1 1 1 0 1 0 0	IP-INC8				
JL/JNGE = Jump on less/not greater or equal	0 1 1 1 1 1 0 0	IP-INC8				
JLE/JNG = Jump on less or equal/not greater	0 1 1 1 1 1 1 0	IP-INC8				
JB/JNAE = Jump on below/not above or equal	0 1 1 1 0 0 1 0	IP-INC8				
JBE/JNA = Jump on below or equal/not above	0 1 1 1 0 1 1 0	IP-INC8				
JP/JPE = Jump on parity/parity even	0 1 1 1 1 0 1 0	IP-INC8				
JO = Jump on overflow	0 1 1 1 0 0 0 0	IP-INC8				
JS = Jump on sign	0 1 1 1 1 0 0 0	IP-INC8				
JNE/JNZ = Jump on not equal/not zer0	0 1 1 1 0 1 0 1	IP-INC8				
JNL/JGE = Jump on not less/greater or equal	0 1 1 1 1 1 0 1	IP-INC8				
JNLE/JG = Jump on not less or equal/greater	0 1 1 1 1 1 1 1	IP-INC8				
JNB/JAE = Jump on not below/above or equal	0 1 1 1 0 0 1 1	IP-INC8				
JNBE/JA = Jump on not below or equal/above	0 1 1 1 0 1 1 1	IP-INC8				
JNP/JPO = Jump on not par/par odd	0 1 1 1 1 0 1 1	IP-INC8				
JNO = Jump on not overflow	0 1 1 1 0 0 0 1	IP-INC8				

Table 4-12. 8086 Instruction Encoding (Cont'd.)

CONTROL TRANSFER (Cont'd.)

RET = Return from CALL:	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
JNS = Jump on not sign	0 1 1 1 1 0 0 1	IP-INC8				
LOOP = Loop CX times	1 1 1 0 0 0 1 0	IP-INC8				
LOOPZ/LOOPE = Loop while zero/equal	1 1 1 0 0 0 0 1	IP-INC8				
LOOPNZ/LOOPNE = Loop while not zero/equal	1 1 1 0 0 0 0 0	IP-INC8				
JCXZ = Jump on CX zero	1 1 1 0 0 0 1 1	IP-INC8				
INT = Interrupt:						
Type specified	1 1 0 0 1 1 0 1	DATA-8				
Type 3	1 1 0 0 1 1 0 0					
INTO = Interrupt on overflow	1 1 0 0 1 1 1 0					
IRET = Interrupt return	1 1 0 0 1 1 1 1					
PROCESSOR CONTROL						
CLC = Clear carry	1 1 1 1 1 0 0 0					
CMC = Complement carry	1 1 1 1 0 1 0 1					
STC = Set carry	1 1 1 1 1 0 0 1					
CLD = Clear direction	1 1 1 1 1 1 0 0					
STD = Set direction	1 1 1 1 1 1 0 1					
CLI = Clear interrupt	1 1 1 1 1 0 1 0					
STI = Set interrupt	1 1 1 1 1 0 1 1					
HLT = Halt	1 1 1 1 0 1 0 0					
WAIT = Wait	1 0 0 1 1 0 1 1					
ESC = Escape (to external device)	1 1 0 1 1 x x x	m o d y y y r / m	(DISP-LO)	(DISP-HI)		
LOCK = Bus lock prefix	1 1 1 1 0 0 0 0					
SEGMENT = Override prefix	0 0 1 reg 1 1 0					

Table 4-13. Machine Instruction Decoding Guide

1ST BYTE		2ND BYTE	BYTES 3, 4, 5, 6	ASM-86 INSTRUCTION FORMAT	
HEX	BINARY				
00	0000 0000	MOD REG R/M	(DISP-LO),(DISP-HI)	ADD	REG8/MEM8,REG8
01	0000 0001	MOD REG R/M	(DISP-LO),(DISP-HI)	ADD	REG16/MEM16,REG16
02	0000 0010	MOD REG R/M	(DISP-LO),(DISP-HI)	ADD	REG8,REG8/MEM8
03	0000 0011	MOD REG R/M	(DISP-LO),(DISP-HI)	ADD	REG16,REG16/MEM16
04	0000 0100	DATA-8		ADD	AL,IMMED8
05	0000 0101	DATA-LO	DATA-HI	ADD	AX,IMMED16
06	0000 0110			PUSH	ES
07	0000 0111			POP	ES