

Assembly Instruction Set

In this appendix, we show the binary encoding of a typical 8086 instruction and give a summary of the common 8086, 8087, 80286, and 80386 instructions.

Typical 8086 Instruction Format

A machine instruction for the 8086 occupies from one to six bytes. For most instructions, the first byte contains the opcode, the second byte contains the addressing modes of the operands, and the other bytes contain either address information or immediate data. A typical two-operand instruction has the format given in Figure F.1

In the first byte, we see a six-bit opcode that identifies the operation. The same opcode is used for both 8- and 16-bit operations. The size of the operands is given by the W bit: W = 0 means 8-bit data and W = 1 means 16-bit data.

For register-to-register, register-to-memory, and memory-to-register operations, the REG field in the second byte contains a register number and the D bit specifies whether the register in the REG field is a source or destination operand, D = 0 means source and D = 1 means destination. For other types of operations, the REG field contains a three-bit extension of the opcode.

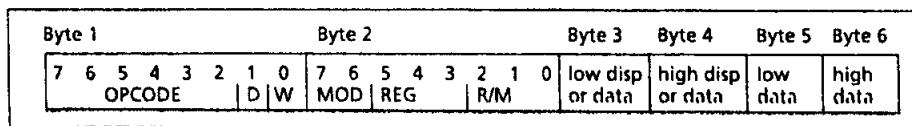


Figure F.1 Opcode Format

MOD=11			Effective Address Calculation			
R/M	W = 0	W = 1	R/M	MOD = 00	MOD = 01	MOD = 10
000	AL	AX	000	(BX) + (SI)	(BX) + (SI) + D8	(BX) + (SI) + D16
001	CL	CX	001	(BX) + (DI)	(BX) + (DI) + D8	(BX) + (DI) + D16
010	DL	DX	010	(BP) + (SI)	(BP) + (SI) + D8	(BP) + (SI) + D16
011	BL	BX	011	(BP) + (DI)	(BP) + (DI) + D8	(BP) + (DI) + D16
100	AH	SP	100	(SI)	(SI) + D8	(SI) + D16
101	CH	BP	101	(DI)	(DI) + D8	(DI) + D16
110	DH	SI	110	DIRECT ADDRESS	(BP) + D8	(BP) + D16
111	BH	DI	111	(BX)	(BX) + D8	(BX) + D16

MOD = 11 means register mode.

MOD = 00 means memory mode with no displacement, except when R/M = 110, then a 16-bit displacement follows.

MOD = 01 means memory, with 8-bit displacement following (D8).

MOD = 10 means memory mode with 16-bit displacement following (D16).

Figure F.2 MOD and R/M Fields

The combination of the W bit and the REG field can specify a total of 16 registers, see Table E.1

The second operand is specified by the MOD and R/M fields. Figure F.2 shows the various modes.

For segment registers, the field is indicated by SEG. Table F.2 shows the segment register encodings.

8086 Instructions

The following set of 8086 instructions appears in alphabetical order. In the set

- (register) stands for the contents of the register
- (EA) stands for the contents of the memory location given by the effective address EA
- flags affected means those flags that are modified by the instruction according to the result
- flags undefined means the values of those flags are unreliable
- disp means 8-bit displacement
- disp-low disp-hi means 16-bit displacement

Table F.1 Register Encoding

REG	W = 0	W = 1
000	AL	Ax
001	CL	cx
010	DL	DX
011	BL	BX
100	AH	SP
101	CH	BP
110	DH	SI
111	BH	DI

Table F.2 Segment Register Encoding

SEG	Register
00	ES
01	CS
10	SS
11	DS

AAA: ASCII Adjust for **Addition**

Corrects the result in AL of adding **two** unpacked BCD **digits** or two ASCII digits.

Format: **AAA**

Operation: If **the** lower **nibble** of AL is greater than 9 or if **AF** is set to 1, then AL **is incremented** by 6, AH **is** incremented by 1, and AF is set to 1. This **instruction** always clears the upper nibble of AL and copies AF to CF.

Flags: Affected-AF, CF
Undefined-OF, **PF**, SF, **ZF**

Encoding: 00110111
37

AAD: ASCII Adjust for Division

Adjusts **the** unpacked BCD dividend in AX in preparation for division.

Format: **AAD**

Operation: **The** unpacked **BCD** operand in AX is converted into binary and stored in AL. This is achieved by **multiplying** AH by 10 and **adding** the result to AL. AH is then cleared.

Flags: **Affected**—**PF**, SF, ZF
Undefined-AF, CF, OF

Encoding: 11010101 00001010
D5 0A

AAM: ASCII Adjust for **Multiplication**

Converts the result of multiplying two BCD digits into unpacked BCD format. Can be used in converting numbers lower than 100 into unpacked BCD format.

Format: **AAM**

Operation: The contents of AL are converted into two unpacked **BCD** digits and placed in AX. AL is divided by 10 and the quotient is placed in AH and the **remainder** in AL.

Flags: **Affected**—PF, SF, ZF
 Undefined—AF, CF, OF
Encoding: 11010100 00001010
 D4 OA

AAS: ASCII Adjust for Subtraction

Corrects the result in AL of subtracting two unpacked **BCD** numbers.

Format: **AAS**
Operation: If the lower nibble of AL is greater than 9 or If AF is set to 1, then AL is decremented by 6, AH is decremented by 1, and AF is set to 1. This instruction always clears the upper nibble of AL and copies AF to **CF**.
Flags: **Affected**—AF, CF
 Undefined—OF, PF, SF, ZF
Encoding: 00111111
 3F

ADC: Add with Carry

The carry flag is added to the sum of the source and destination.

Format: ADC destination, source
Operation: If CF = 1, then (dest) = (source) + (dest) + 1
 If CF = 0, then (dest) = (source) + (dest)
Flags: Affected—AC CF, OF, PF, SF, ZF
Encoding: Memory or register with register
 000100dw mod reg r/m
 Immediate to accumulator
 0001 010w data
 Immediate to memory or register
 100000sw mod 010 r/m data
 (s is set if a byte of data is added to **16-bit** memory or register.)

ADD: Addition

Format: ADD destination, source
Operation: (dest) = (source) + (dest)
Flags: Affected—AF, CF, OF, PF, SF, ZF
Encoding: Memory or register with register
 000000dw mod reg r/m
 Immediate to accumulator
 000001 0w data
 Immediate to memory or register
 100000sw mod 000 r/m data
 (s is set if a byte data is **added** to **16-bit** memory or register.)

AND: Logical AND

Format: AND destination, source
Operation: Each bit of the source is **ANDed** with the corresponding bit in the destination, with the result stored in the destination. CF and OF are cleared.
Flags: **Affected**—CF, OF) PF, SF, ZF
 Undefined: AF
Encoding: Memory or register with register
 001000dw mod reg r/m
 Immediate to accumulator
 0010010w data

Immediate to memory or **register**
1000000w mod 100 r/m data

CALL: Procedure Call

Format: CALL target

Operation: The offset address of the next **sequential** instruction is pushed onto the stack, and control is transferred to the target operand. The target address is computed as follows: (1) intrasegment direct, offset = IP + displacement, (2) intrasegment indirect, offset = (EA), (3) intersegment direct, **segment:offset** given in instruction, and (4) intersegment indirect, segment = (EA + 2), offset = (EA).

Flags: Affected—none

Encoding: Intra-segment Direct

11101000 disp-low disp-high

Intra-segment Indirect

11111111 mod 010 r/m

Intersegment Direct

10011010 offset-low offset-high seg-low seg-high

Intersegment Indirect

11111111 mod 011 r/m

CBW: Convert Byte to Word

Converts the signed **8-bit** number in AL into a signed M-bit number in AX.

Format: CBW

Operation: If bit 7 of AL is set, then AH gets FFh.

If bit 7 of AL is clear, then AH is cleared.

Flags: Affected—none

Encoding: 10011000

98

CLC: Clear Carry Flag

Format: CLC

Operation: Clears CF

Flags: Affected—CF

Encoding: 11111000

F8

CLD: Clear Direction Flag

Format: CLD

Operation: Clears DF

Flags: Affected—DF

Encoding: 11111100

FC

CLI: Clear Interrupt Flag

Disables **maskable** external interrupts.

Format: CLI

Operation: Clears IF

Flags: Affected—IF

Encoding: 11111010

FA

CMC: Complement Carry Flag

Format: CMC
Operation: Complements CF
Flags: Affected-CF
Encoding: 11110100
F5

CMP: Compare

Compares two operands by subtraction. The flags are affected, but the result is not stored.

Format: CMP destination, source
Operation: The source operand is subtracted from the destination and the flags are set according to the result. The operands are not affected.
Flags: Affected-U, CF, OF, PF, SF, ZF
Encoding: Memory or register with register
001110dw mod reg r/m
Immediate with accumulator
0011110w data
Immediate with memory or register
100000sw mod 111 r/m data

CMPS/CMPSB/CMPSW: Compare Byte or Word String

Compares two memory operands. If preceded by a REP prefix, strings of arbitrary size can be compared.

Format: CMPS source-string, dest-string
or
CMPSB
or
CMPSW
Operation: The dest-string indexed by ES:DI is subtracted from the source-string indexed by SI. The status flags are affected. If the control flag DF is 0, then SI and DI are incremented; otherwise, they are decremented. The increments are 1 for byte strings and 2 for word strings.
Flags: Affected-AR CF, OF, PF, SF, ZF
Encoding: 1010011"

CWD: Convert Word to Double Word

Converts the signed 16-bit number in AX into a signed 32-bit number in DX:AX.

Format: CWD
Operation: If bit 15 of AX is set, then DX gets FFFF.
If bit 15 of AX is clear, then DX is cleared.
Flags: Affected-none
Encoding: 10011001
99

DAA: Decimal Adjust for Addition

Corrects the result in AL of adding two packed BCD operands.

Format: DAA
Operation: If the lower nibble of AL is greater than 9 or if AF is set to 1, then AL is incremented by 6, and AF is set to 1. If AL is

greater than 9Fh or if the CF: is set, then 60h is added to AL and CF is set to 1.
 Flags: Affected—AF, CF, PF, SF, ZF
 Undefined—OF
 Encoding: 00100111
 27

DAS: Decimal Adjust for Subtraction

Corrects the result in AL of subtracting two packed BCD operands.
 Format: DAS
 Operation: If the lower nibble of AL is greater than 9 or if AF is set to 1, then 60h is subtracted from AL and CF is set to 1.
 Flags: Affected—AF, CF, PF, SF, ZF
 Encoding: 00101111
 2F

DEC: Decrement

Format: DEC destination
 Operation: Decrements the destination operand by 1
 Flags: Affected—AF, OF, PF, SF, ZF
 Encoding: Register (word)
 01001 reg
 Memory or register
 1111111w mod 001 r/m

DIV: Divide

Performs unsigned division.
 Format: DIV source
 Operation: The divisor is the source operand, which is either memory or register. For byte division (R-bit source) the dividend is AX, and for word division (16-bit source) the dividend is DX:AX. The quotient is returned to AL (AX for word division), and the remainder is returned to AH (DX for word division). If the quotient is greater than 8 bits (16 bits for word division), then an INT 0 is generated.
 Flags: Undefined—AK CF, OF, PF, SF, ZF
 Encoding: 1111011w mod 110 r/m

ESC: Escape

Allows other processors, such as the 8087 coprocessor, to access instructions. The 8086 processor performs no operation except to fetch a memory operand for the other processor.
 Format: ESC external-opcode, source
 Flags: none
 Encoding: 11011xxx mod xxx r/m
 (The xxx sequence indicates an opcode for the coprocessor.)

HLT: Halt

Causes the processor to enter its halt state to wait for an external interrupt.
 Format: HLT
 Flags: none
 Encoding: 11110100
 F4

IDIV: Integer Divide

Performs signed division.

Format: **IDIV** source

Operation: The divisor is the source operand, which is either memory or register. For byte division (8-bit source) the dividend is AX, and for word division (16-bit source) the dividend is DX:AX. The quotient is returned to AL (AX for word division), and the remainder is returned to AH (DX for word division). If the quotient is greater than 8 bits (16 bits for word division), then an INT 0 is generated.

Flags: Undefined-AF, CF, OF, PF, SF, ZF

Encoding: 1111011w mod 111 r/m

IMUL: integer Multiply

Performs signed multiplication.

Format: **IMUL** source

Operation: The multiplier is the source operand, which is either memory or register. For byte multiplication (8-bit source) the multiplicand is AL, and for word multiplication (16-bit source) the multiplicand is AX. The product is returned to AX (DX:AX for word multiplication). The flags CF and OF are set if the upper half of the product is not the sign-extension of the lower half.

Flags: Affected-U, OF
Undefined—AF, PF, SF, ZF

Encoding: 1111011w mod 101 r/m

IN: Input Byte or Word

Format: **IN** accumulator, port

Operation: The contents of the accumulator are replaced by the contents of the designated I/O port. The port operand is either a constant (for fixed port), or DX (for variable port).

Flags: Affected—none

Encoding: Fixed port
1110010w port
Variable port
1110110w

INC: Increment

Format: **INC** destination

Operation: increments the destination operand by 1.

Flags: Affected-AF, OF, PF, SF, ZF

Encoding: Register (word)
01000 reg
Memory or register
1111111w mod 000 r/m

INT: Interrupt

Transfers control to one of 256 interrupt routines.

Format: **INT** Interrupt-type

Operation: The FLAGS register is pushed onto the stack, then TF and IF are cleared, CS is pushed onto the stack and then filled by the high-order word of the interrupt vector, IP is pushed

onto the stack and then filled by the low-order word of the interrupt vector.

Flags: Affected—IF,TF
 Encoding: Type 3
 11001100
 Other types
 11001101 type

INTO: Interrupt if Overflow

Generates an INT 4 if OF is set.

Format: INTO
 Operation: If OF = 1, then same operation as INT 4. If OF = 0, then no operation takes place.
 Flags: If OF=1 then OF and TF are cleared.
 If OF=0 then no flags are affected.
 Encoding: 11001110
 CE

IRET: Interrupt Return

Provides a return from an interrupt routine.

Format: IRET
 Operation: Pops the stack into the registers IP, CS, and FLAGS.
 Flags: Affected—all
 Encoding: 11001111
 CF

J(condition): Jump Short, If Condition Is Met

Format: J(condition) short-label
 Operation: If the condition is true, then a short jump is made to the label. The label must be within -128 to +127 bytes of the next instruction.
 Flags: Affected—none

Instruction	Jump If	Condition	Encoding
JA	above	CF = 0 and ZF = 0	77 disp
JAE	above or equal	CF = 0	73 disp
JB	below	CF = 1	72 disp
JBE	below or equal	CF = 1 or ZF = 1	76 disp
JC	carry	CF = 0	72 disp
JCXZ	CX is 0	(CF or ZF) = 0	E3 disp
JE	equal	ZF = 1	74 disp
JG	greater	ZF = 0 and SF = OF	7F disp
JGE	greater or equal	ZF = OF	7D disp
JL	less	(SF xor OF) = 1	7C disp
JLE	less or equal	(SF xor OF) or ZF = 1	7E disp
JNA	not above	CF = 1 or ZF = 1	76 disp
JNAE	not above or equal	CF = 1	72 disp
JNB	not below	CF = 0	73 disp
JNBE	not below or equal	CF = 0 and ZF = 0	77 disp
JNC	not carry	CF = 0	73 disp
JNE	not equal	ZF = 0	75 disp

JNG	not greater	(SF xor OF) or ZF = 1	7E disp
JNGE	not greater nor equal (SF xor OF) = 1	ZF = 0	7C disp
JNL	not less	SF = OF	7D disp
JNLE	not less nor equal	ZF = 0 and SF = OF	7F disp
JNO	not overflow	OF = 0	71 disp
JNP	not parity	PF = 0	7B disp
JNS	not sign	SF = 0	79 disp
JNZ	not zero	ZF = 0	75 disp
JO	overflow	OF = 1	70 disp
JP	parity	PF = 1	7A disp
JPE	parity even	PF = 1	7A disp
JPO	parity odd	PF = 0	7B disp
JS	sign	SF = 1	79 disp
JZ	zero	ZF = 1	74 disp

JMP: Jump

Format: JMP target
 Operation: Control is transferred to target label.
 Flags: Affected—none
 Encoding: Intra-segment direct
 11101001 disp-low disp-hi
 Intra-segment direct short
 11101011 disp
 Inter-segment direct
 11101010
 Inter-segment indirect
 11111111 mod 101 r/m
 Intra-segment indirect
 11111111 mod 100 r/m

LAHF: Load AH from Flags

Format: LAHF
 Operation: The low eight bits of the FLAGS register are transferred to AH.
 Flags: Affected—none
 Encoding: 10011111
 9F

LDS: Load Data Segment Register

Loads the DS register with a segment address and a general register with an offset so that data at the segment:offset may be accessed.
 Format: LDS destination, source
 Operation: The source is a doubleword memory operand. The lower word is placed in the destination register, and the upper word is placed in DS.
 Flags: Affected—none
 Encoding: 11000101 mod reg r/m

LEA: Load Effective Address

Loads an offset memory address to a register.

Format: LEA destination, source
Operation: The offset address of the source memory operand is placed in the destination, which is a general register.
Flags: Affected—none
Encoding: 10001101 mod reg r/m

LES: Load Extra Segment Register

Loads the ES register with a segment address and a general register with an offset so that data at the segment:offset may be accessed.

Format: LES destination, source
Operation: The source is a doubleword memory operand. The lower word is placed in the destination register, and the upper word is placed in ES.
Flags: none
Encoding: 11000100 mod reg r / m

LOCK: Lock Bus

In a multiprocessor environment, locks the bus.

Format: LOCK
Operation: LOCK is used as a prefix that can precede any instruction. The bus is locked for the duration of the execution of the instruction to prevent other processors from accessing memory.
Flags: none
Encoding: 11110000
 F0

LODS/LODSB/LODSW: Load Byte or Word String

Transfers a memory byte or word indexed by SI to the accumulator.

Format: LODS source-string
 or
 LODSB
 or
 LODSW
Operation: The source byte (word) is loaded into AL (or AX). SI is incremented by 1 (or 2) if DF is clear; otherwise SI is decremented by 1 (or 2).
Flags: Affected—none
Encoding: 1010110w

LOOP

Loop until count is complete.

Format: LOOP short-label
Operation: CX is decremented by 1, and if the result is not zero then control is transferred to the labeled instruction; otherwise control flows to the next instruction.
Flags: Affected—none
Encoding: 11100010 disp
 E2

LOOPE/LOOPZ: Loop if Equal/Loop if zero

A loop is controlled by the counter and the ZF.

Format: LOOPE short-label
 or

LOOPZ short-label
 Operation: CX is decremented by 1, if the result is not zero and ZF = 1, then control is transferred to the labeled instruction; otherwise, control flows to the next instruction.
 Flags: Affected—none
 Encoding: 11100001 disp
 E1

LOOPNE/LOOPNZ: Loop If Not Equal/Loop If Not Zero

A loop is controlled by the counter and the ZF.

Format: LOOPNE short-label
 or
 LOOPNZ short-label
 Operation: CX is decremented by 1, if the result is not zero and ZF = 0, then control is transferred to the labeled instruction; otherwise, control flows to the next instruction.
 Flags: Affected—none
 Encoding: 11100000 disp
 E0

MOV: Move

Move data.

Format: MOV destination, source
 Operation: Copies the source operand to the destination operand.
 Flags: Affected—none
 Encoding: To memory from accumulator
 1010001w addr-low addr-high
 To accumulator from memory
 1010000w addr-low addr-high
 To segment register from memory or register
 10001110 mod 0 seg r/m
 To memory or register from segment register
 10001100 mod 0 seg r/m
 To register from memory or register/ To memory from reg
 100010d1 mod reg r/m (addr-low addr-high)
 To register from immediate-data
 1011w reg data (data-high)
 To memory or register from immediate-data
 1100011w mod 000 r/m data (data-high)

MOVS/MOVSb/MOVSW: Move Byte or Word String

Transfers memory data addressed by SI to memory location addressed by ES:DI. Multiple bytes (or words) can be transferred if the prefix REP is used.

Format: MOVS dest-string, source-string
 or
 MOVSb
 or
 MOVSW
 Operation: The source string byte (or word) is transferred to the destination operand. Both SI and DI are then incremented by 1 (or 2 for word strings) if DF = 0; otherwise, both are decremented by 1 (or 2 for word strings).
 Flags: Affected—none
 Encoding: 1010010w

MUL: Multiply

Unsigned multiplication.

Format: **MUL source**

Operation: The multiplier is the source operand which is either memory or register. For byte multiplication (8-bit source) the multiplicand is AL and for word multiplication (16-bit source) the multiplicand is AX. The product is returned to AX (DX:AX for word multiplication). The flags CF and OF are set if the upper half of the product is not zero.

Flags: Affected—CF, OF
Undefined—AF, PF, SF, ZF

Encoding: **1111011w mod 100 r/m**

NEG: Negate

Forms two's complement.

Format: **NEG destination**

Operation: The destination operand is subtracted from all 1's (0FFh for bytes and 0FFFh for words). Then a 1 is added and the result placed in the destination.

Flags: Affected—AF, CF, OF, PF, SF, ZF

Encoding: **1111011w mod 011 r/m**

NOP: No Operation

Format: **NOP**

Operation: No operation is performed

Flags: Affected—none

Encoding: **13010000**
30

NOT: Logical Not

Format: **NOT destination**

Operation: Forms the one's complement of the destination.

Flags: Affected—none

Encoding: **1111011w mod 010 r/m**

OR: Logical Inclusive Or

Format: **OR destination, source**

Operation: Performs logical OR operation on each bit position of the operands and places the result in the destination.

Flags: Affected—CF, OF, PF, SF, ZF
Undefined—AF

Encoding: **Memory or register with register**
000010dw mod reg r/m
immediate to accumulator
0000110w data
immediate to memory or register
1000000w mod 001 r/m

OUT: Output Byte or Word

Format: **OUT accumulator, port**

Operation: The contents of the designated I/O port are replaced by the contents of the accumulator. The port is either a constant (for fixed port) or DX (for variable port).

Flags: Affected—none
Encoding: Fixed Port
1110011% port
Variable port
1110111w

POP: Pop Word Off Stack to Destination

Format: POP destination
Operation: The contents of the destination are replaced by the word at the top of the stack. The stack pointer is incremented by 2.
Flags: Affected—none
Encoding: General register
01011 reg
Segment register
000 seg 1 11
Memory or register
10001111 mod 0 0 0 r/m

POPF: Pop Flags Off Stack

Format: POPF
Operation: Transfers flag bits from the top of the stack to the FLAGS register and then increments SP by 2.
Flags: Affected—all
Encoding: 10011101
9D

PUSH: Push Word onto Stack

Format: PUSH source
Operation: Decrements the SP register by 2 and then transfers a word from the source operand to the new top of stack.
Flags: none
Encoding: General register
01010reg
Segment register
000 seg 110
Memory or register
11111111 mod 110 r / m

PUSHF: Push Flags onto Stack

Format: PUSHF
Operation: Decrements SP by 2 and transfers flag bits to the top of the stack.
Flags: Affected—none
Encoding: 10011100
9C

RCL: Rotate Left Through Carry

Rotates destination left through the CF flag one or more times.

Format: RCL destination, 1
or

RCL destination, CL

Operation: The first format rotates the destination once through CF resulting in the msb being placed in CF and the old CF ended in the lsb. To rotate more than once, the count must be

placed in CL. When the count is 1 and the leftmost two bits of the old destination are equal, then OF is cleared; if they are unequal, OF is set to 1. When the count is not 1, then OF is undefined. CL is not changed.

Flags: Affected—CF,OF
 Encoding: 110100vw nod 010 r/m
 If v = 0, count = 1
 If v = 1, count = (CL)

RCR: Rotate Right Through **Carry**

Rotates destination right through the CF flag one or more times.

Format: RCR destination, 1
 or
 RCR destination, CL

Operation: The first format rotates the destination once through CF resulting in the lsb being placed in CF and the old CF ended in the msb. To rotate more than once, the count must be placed in CL. When the count is 1 and the leftmost two bits of the new destination are equal, then OF is cleared; if they are unequal, OF is set to 1. When the count is not 1, then OF is undefined. CL is not changed.

Flags: Affected—CF,OF
 Encoding: 110100vw mod 011 r/m
 If v = 0, count = 1
 if v = 1, count = (CL)

REP/REPZ/REPE/REPNE/REPZ: Repeat String Operation

The string operation that follows is repeated while (CX) is not zero.

Format: REP/REPZ/REPE/REPNE/REPZ string-instruction
 Operation: The string operation is carried out until (CX) is decremented to 0. For CMPS and SCAS operations, the ZF is also used in terminating the iteration. For REP/REPZ/REPE the CMPS and SCAS operations are repeated if (CX) is not zero and ZF is 1. For REPNE/REPZ, the CMPS and SCAS operations are repeated if (CX) is not zero and ZF is 0.
 Flags: See the associated string instruction.
 Encoding: REP/REPZ/REPE 11110011
 REPNE/REPZ 11110010

RET: Return from Procedure

Returns control after a called procedure has been executed.

Format: RET [pop-value]
 Operation: If RET is within a NEAR procedure, it is translated into an intrasegment return, which updates the IP by popping one word from the stack. If RET is within a FAR procedure, it is translated into an intersegment return that updates both the IP and CS. The optional pop value specifies a number of bytes in the stack to be discarded. These are parameters passed to the procedure.
 Flags: Affected—none
 Encoding: Intrasegment
 11000011
 Intrasegment with pop value
 11000010

Intersegment
 11001011
 Intersegment with pop value
 11001010

ROL: Rotate Left

Rotates destination left one or more times.

Format: ROL destination, 1
 OR
 ROL destination, CL

Operation: The first format rotates the destination once; CF also gets the msb. To rotate more than once, the count must be placed in CL. When the count is 1 and the new CF is not the same as the msb, then the OF is set, otherwise, OF is cleared. When the count is not 1, then OF is undefined. CL is not changed.

Flags: Affected-CF, OF

Encoding: 110100vw mod 000 r/m
 If v = 0, count = 1
 If v = 1, count = (CL)

ROR: Rotate Right

Rotates destination right one or more times.

Format: ROR destination, 1
 OR
 ROR destination, CL

Operation: The first format rotates the destination once; CF also gets the lsb. To rotate more than once, the count must be placed in CL. When the count is 1 and the leftmost two bits of the new destination are equal, then OF is cleared; if they are unequal, OF is set to 1. When the count is not 1, then OF is undefined. CL is not changed.

Flags: Affected-CF, OF

Encoding: 110100vw mod 001 r/m
 If v = 0, count = 1
 If v = 1, count = (CL)

SAHF: Store AH in FLAG5 Register

Format: SAHF

Operation: Stores five bits of AH into the lower byte of the FLAGS register. Only the bits corresponding to the flags are transferred. The flags in the lower byte of FLAGS register are SF = bit 7, ZF = bit 6, AF = bit 4, PF = bit 2, and CF = bit 0.

Flags: Affected-AF, CF, PF, SF, ZF

Encoding: 10011110
 7E

SAL/SHL: Shift Arithmetic Left/Shift Logical Left

Format: SAL/SHL destination, 1
 OR
 SAL/SHL destination, CL

Operation: The first format shifts the destination once; CF gets the msb and a 0 is shifted into the lsb. To shift more than once, the count must be placed in CL. When the count is 1 and the

new CF is not the same as the msb, then the OF is set; otherwise, OF is cleared. When the count is not 1, then OF is undefined. CL is not changed.

Flags: Affected—CF, OF, PF, SF, ZF
 Undefined—AF
 Encoding: 110100vw mod 100 r/m
 If v = 0, count = 1
 If v = 1, count = (CL)

SAR: Shift Arithmetic Right

Format: SAR destination, 1
 or
 SAR destination, CL
 Operation: The first format shifts the destination once; CF gets the lsb and the msb is repeated (sign is retained). To shift more than once, the count must be placed in CL. When the count is 1 OF is cleared. When the count is not 1, then OF is undefined. CL is not changed.
 Flags: Affected—CF, OF, PF, SF, ZF
 Undefined—AF
 Encoding: 110100vw mod 111 r/m
 If v = 0, count = 1
 If v = 1, count = (CL)

SBB: Subtract with Borrow

Format: SBR destination, source
 Operation: Subtracts source from destination; and if CF is 1 then subtract 1 from the result. The result is placed in the destination.
 Flags: Affected—AF, CF, OF, PF, SF, ZF
 Encoding: Memory or register with register
 000110dw mod reg r/m
 Immediate from accumulator
 0001110w data
 Immediate from memory or register
 100000sw mod 011 r/m data
 (s is set if an immediate-data-byte is subtracted from 16-bit memory or register.)

SCASISCASBISCASW: Scan Byte or Word String

Compares memory against the accumulator. Used with REP, it can scan multiple memory locations for a particular value.

Format: SCAS dest-string
 or
 SCASB
 or
 SCASW
 Operation: Subtracts the destination bytr (or word) addressed by DI from AL (or AX). The flags are affected but the result is not saved. DI is incremented (if DF = 1), or decremented (if DF = 0) by 1 (byte strings) or 2 (word strings).
 Flags: Affected—AF, CF, OF, PF, SF, ZF
 Encoding: 1010111w

4

SHR: Shift Logical Right

Format: SHR destination, 1
or
SHR destination, CL

Operation: The first format shifts the destination once; CF gets the lsb and a 0 is shifted into the msb. To shift more than once, the count must be placed in CL. When the count is 1 and the leftmost two bits are equal, then OF is cleared; otherwise, OF is set to 1. When the count is not 1, then OF is undefined. CL is not changed.

Flags: Affected—CF, OF, PF, SF, ZF
Undefined—AF

Encoding: 110100vw mod 101 r/m
If v = 0, count = 1
If v = 1, count = (CL)

STC: Set Carry Flag

Format: STC

Operation: CF is set to 1.

Flags: Affected—CF

Encoding: 11111001
F9

STD: Set Direction Flag

Format: STD

Operation: DF is set to 1

Flags: Affected—DF

Encoding: 11111101
FD

STI: Set Interrupt Flag

Format: STI

Operation: IF is set to 1, thus enabling external interrupts

Flags: Affected—IF

Encoding: 11111011
FB

STOS/STOSB/STOSW: Store Byte or Word String

Stores the accumulator into memory. When used with REP, it can store multiple memory locations with the same value.

Format: STOS dest-string
or
STOSB
or
STOSW

Operation: Stores AL (or AX) into the destination byte (or word) addressed by DI. DI is incremented (DF = 1), or decremented (DF = 0) by 1 (byte strings) or 2 (word strings).

Flags: Affected—none

Encoding: 1010101w

SUB: Subtract

Format: SUB destination, source
Operation: Subtracts source from destination. The result is placed in the destination.
Flags: Affected—AF, CF, OF, PF, SF, ZF
Encoding: Memory or register with register
 001010dw mod reg r / m
 Immediate from accumulator
 0010110w data
 Immediate from memory or register
 100000sw mod 101 r/m data
 (s is set if an immediate-data-byte is subtracted from 1-bit memory or register.)

TEST: Test (Logical Compare)

Format: TEST destination, source
Operation: The two operands are ANDed to affect the flags. The operands are not affected.
Flags: Affected—CF, OF, PF, SF, ZF
 Undefined—AF
Encoding: Memory or register with register
 1000010w mod reg r/m
 Immediate with accumulator
 1010100w data
 Immediate with memory or register
 1111011w mod 000 r/m data

WAIT

Format: WAIT
Operation: The processor is placed in a wait state until activated by an external interrupt.
Flags: Affected—none
Encoding: 10011011
 9B

XCHG: Exchange

Format: XCHG destination, source
Operation: The source operand and the destination operand are interchanged.
Flags: Affected—none
Encoding: Register with accumulator
 10010reg
 Memory or register with register
 1000011w mod reg r/m

XLAT: Translate

Performs a table lookup translation.
Format: XLAT source-table
Operation: BX must contain the offset address of the source table, which is at most 256 bytes. AL should contain the index of the table element. The operation replaces AL by the contents of the table element addressed by BX and AL.
Flags: Affected—none
Encoding: 11010111
 D7

XOR: Exclusive OR

Format: XOR destination, **source**
Operation: The exclusive OR operation is performed bit-wise with the source and destination operands; the result is stored in the destination. CF and **OF** are cleared.
Flags: Affected-CF, OF, **PF**, SF, **ZF**
Undefined: **AF**
Encoding: Memory or register with register
001100dw mod **reg** r/m
Immediate to accumulator
0011010w **data**
Immediate to memory or register
1000000w mod 110 r/m **data**

8087 instructions

The 8087 uses several data types, when transferring data to or from memory, the memory data definition determines the data type format. Table F.3 shows the association between the 8087 data types and the memory data definitions. In this section we only give 8087 instructions for simple arithmetic operations. Check the 8087 manual for other instructions.

FADD: Add Real

Format: FADD
OF
FADD **source**
or
FADD **destination, source**
Operation: Adds a source operand to the destination. For the first form, the source operand is the top of the stack and the destination is ST(1). The top of the stack is popped, and its value is added to the new top. For the second form, the source is either short real or long real in memory; the destination is the top of the stack. For the third form, one of the operands is the top of the stack and the other is another stack register; the stack is not popped.

Table F.3 8087 Data Types

Data Type	Size (bits)	Memory Definition	Pointer Type
Word integer	16	DW	WORD PTR
Short integer	32	DD	DWORD PTR
Long integer	64	DQ	QWORD PTR
Packed decimal	80	DT	TBYTE PTR
Short real	32	DD	DWORD PTR
Long real	64	DQ	QWORD PTR
Temporary real	80	DT	TBYTE PTR

FBLD: Packed **Decimal** Load

Format: FBLD source

Operation: Loads a packed decimal number to the top of the stack. The source operand is of type DT (10 bytes).

FBSTP: Packed BCD Store and Pop

Format: FBSTP destination

Operation: Converts the top of the stack to a packed BCD format and stores the result in the memory destination. Then the stack is popped.

FDIV: Divide Real

Format: FDIV

or
FDIV source
or

FDIV destination, source

Operation: Divides the destination by the source. For the first form, the source operand is the top of the stack and the destination is ST(1). The top of the stack is popped and its value is used to divide into the new top. For the second form, the source is either short real or long real in memory; the destination is the top of the stack. For the third form, one of the operands is the top of the stack and the other is another stack register; the stack is not popped.

FIADD: Integer Add

Format: FIADD source

Operation: Adds the source operand to the top of the stack. The source operand can be either a short integer or a word integer.

FIDIV: Integer Divide

Format: FIDIV source

Operation: Divides the top of the stack by the source. The source operand can be either a short integer or a word integer.

FILD: Integer Load

Format: FILD source

Operation: Loads a memory integer operand onto the top of the stack. The source operand is either word integer, short integer, or long integer.

FIMUL: Integer Multiply

Format: FIMUL source

Operation: Multiplies the source operand to the top of the stack. The source operand can be either a short integer or a word integer.

FIST: Integer Store

Format: FIST destination

Operation: Rounds the top of the stack to an integer value and stores to a memory location. The destination may be word integer or short integer. The stack is not popped.

FISTP: integer Store and Pop

Format: FISTP destination

Operation: Rounds the top of the stack to an integer value and stores to a memory location. Then the stack is popped. The destination may be word integer, short integer, or long integer.

FISUB: Integer Subtract

Format: FISUB source

Operation: Subtracts the source operand from the top of the stack. The source operand can be either a short integer or a word integer.

FLD: Load Real

Format: FLD source

Operation: Loads a real operand onto the top of the stack. The source may be a stack register ST(i), or a memory location. For a memory operand, the data type may be any of the real formats.

FMUL: Multiply Real

Format: FMUL

or

FMUL source

or

FMUL destination, source

Operation: Multiplies a source operand to the destination. For the first form, the source operand is the top of the stack and the destination is ST(1). The top of the stack is popped and its value is multiplied to the new top. For the second form, the source is either short real or long real in memory; the destination is the top of the stack. For the third form, one of the operands is the top of the stack and the other is another stack register; the stack is not popped.

FST: Store Real

Format: FST destination

Operation: Stores the top of the stack to a memory location or another stack register. The memory destination may be short real (doubleword) or long real (quadword). The stack is not popped.

FSTP: Store Real and Pop

Format: FSTP destination

Operation: Stores the top of the stack to a memory location or another stack register. Then the stack is popped. The memory destination may be short real (doubleword), long real (quadword), or temporary real (10 bytes).

FSUB: Subtract Real

Format: FSUB

or

FSUB source

or

FSUB destination, source

Operation: Subtracts a source operand from the destination. For the first form, the source operand is the top of the stack and the destination is ST(0). The top of the stack is popped and its value is subtracted from the new top. For the second form, the source is either short real or long real in memory; the destination is the top of the stack. For the third form, one of the operands is the top of the stack and the other is another stack register; the stack is not popped.

80286 Instructions

The real-mode 80286 instruction set includes all 8086 instructions plus the extended instruction set. The extended instruction set contains five groups of instructions, (1) multiply with immediate values (IMUL), (2) input and output strings (INS and OUTS), (3) stack operations (POPA, PUSH immediate, PUSHA), and (4) shifts and rotates with immediate count values, and (5) instructions for translating high-level language constructs (BOUND and ENTER). We only give the instructions in groups 1–4.

IMUL: integer Immediate Multiply

Format: IMUL destination, immediate
or

IMUL destination, source, immediate

Operation: For the first format, the immediate operand, which must be a byte, is multiplied with the destination, which must be a 16-bit register. The lower 16-bit of the result is stored in the register. For the second format, the 8- or 16-bit immediate operand is multiplied with the source operand, which may be a 16-bit register or a memory word. The lower 16-bit of the result is stored in the destination, which must be a 16-bit register. The flags CF and OF are set if the upper half of the product is not the sign-extension of the lower half.

Flags: Affected—CF, OF
Undefined—AF, PF, SF, ZF

Encoding: 011010s1 mod reg r/m d a t a [data if s=0]

INS/INSB/INSW: Input from Port to String

Transfers a byte or word string element from a port to memory. Multiple bytes or words can be transferred if the prefix REP is used.

Format: INS destination-string, port
or
INSB
or
INSW

Operation: A byte or word is transferred from the port designated by DX to the location ES:DI. DI is then incremented by 1 (or 2 for word strings) if DF = 0; otherwise, DI is decremented by 1 (or 2 for word strings).

Flags: Affected—none

Encoding: 0110110w

OUTS/OUTSB/OUTSW: Output String to Port

Transfers a byte or word string element from memory to a port. Multiple bytes or words **can** be transferred if the prefix REP is used.

Format: OUTS **destination-string**, port
 or
 OUTSB
 or
 OUTSW

Operation: A byte or word **is** transferred from **memory** located at **DS:SI** to the port **designated** by **DX**. **SI** is then **incremented** by 1 (or 2 for word **strings**) if **DF = 0**; otherwise, **SI** is decremented by 1 (or 2 for word strings).

Flags: Affected-none

Encoding: 0110111w

POPA: Pop All General Registers

Format: POPA

Operation: The registers are popped in the order **DI, SI, BP, SP, BX, DX, CX, and AX**.

Encoding: 01100001
 6l

PUSH: Push Immediate

Format: **PUSH data**

Operation: The data may be **8** or 16 bits. A data byte is **signed extended** into 16 bits before pushing onto the stack.

Flags: Affected-none

Encoding: 011010s0 data [data if s = 0]

PUSHA: Push All General Registers

Format: PUSHA

Operation: The registers are pushed in the order **AX, CX, DX, BX, original SP, BP, SI, and DI**.

Flags: Affected-none

Encoding: 01100000
 60

The general format of shifts and rotates with **immediate** count values is:

Opcode **destination,immediate**

where **opcode** is any one of RCL, RCR, ROL, ROR, SAL, SHL, SAR, and SHR. If the immediate value is 1, then the instruction is the same as an 8086 instruction. For an immediate **value of 2-31**, the instruction operates like an 8086 instruction in which CL contains the value. The 80286 does not allow a constant count value to be greater than 31.

The encodings for immediate values of 2-31 are

```
RCL
1100000w mod 010 r/m
RCR
1100000w mod 011 r/m
ROL
1100000w mod 000 r/m
ROR
1100000w m o d 001 r/m
```


SAL/SHL
 1100000w mod 100 r/m
SAR
 1100000w mod 111 r/m
SHR
 1100000w mod 101 r/m

80386 Instructions

The real-mode 80386 instruction set includes all real-mode 80286 instructions plus their 32-bit extensions, together with six groups of new instructions, (1) bit scans, (2) bit tests, (3) move with extensions, (4) set byte on condition, (5) double-precision shifts, and (6) move to or from special registers. We only give instructions in groups 1-S.

Bit Scan Instructions

The bit scan instructions are **BSF** (bit scan forward) and **BSR** (bit scan reverse). They are used to scan an operand to find the first set bit, and they differ only in the direction of the scan.

Formats: BSF destination, source
 or
 BSR destination, source

Operation: The destination must be a register, the source is either a register or a memory location. They must be both words or both doublewords. The source is scanned for the first set bit. If the bits are all 0, then **ZF** is cleared; otherwise, **ZF** is set and the destination register is loaded with the bit position of the first set bit. For **BSF** the scanning is from bit 0 to the msb, and for **BSR** the scanning is from the msb to bit 0.

Flags: Affected—**ZF**

Encoding: BSF
 00001111 10111101 mod reg r/m
 BSR
 00001111 10111101 mod reg r/m

Bit Test Instructions

The bit test instructions are **BT** (bit test), **BTC** (bit test and complement), **BTR** (bit test and reset), and **BTS** (bit test and set). They are used to copy a bit from the destination operand to the **CF** so that the bit can be tested by a **JC** or **JNC** instruction.

Format: BT destination, source
 or
 BTC destination, source
 or
 BTR destination, source
 or
 BTS destination, source

Operation: The source specifies a bit position in the destination to be copied to the **CF**. **BT** simply copies the bit to **CF**, **BTC** copies the bit and complements it in the destination, **BTR** copies the bit and resets it in the destination, and **BTS** copies the

bit and sets it in the destination. The source is either a **16-bit** register, **32-bit** register, or an R-bit constant. The destination may be a **16-bit** or **32-bit** register or memory. If the source is a register, then the source and destination must have the same size.

Flags: Affected-CF

Encoding: Source is U-bit immediate data:
BI
 00001111 10111010 mod 100 r/m
BTC
 00001111 10111010 mod 111 r/m
BTR
 00001111 10111010 mod 110 r/m
BTS
 00001111 10111010 mod 101 r/m
 Source is register:
BT
 00001111 10100011 mod reg, r/m
BTC
 00001111 10111011 mod reg r/m
BTR
 00001111 10110011 mod reg r/m
BTS
 00001111 10101011 mod reg r/m

Move with Extension Instructions

The move with extension instructions are MOV_{ESX} (move with sign-extend) and MOV_{ZX} (move with zero-extend). These instructions move a small source into a bigger destination and extend to the upper half with the sign or a zero.

Format: MOV_{ESX} destination, source
 or
 MOV_{ZX} destination, source

Operation: The destination must be a register, the source is either a register or memory. If the source is a byte (or word) the destination is a word (or doubleword). MOV_{ESX} copies and sign extends the source into the destination. MOV_{ZX} copies and zero extends the source into the destination.

Flags: Affected-none

Encoding: MOV_{ESX}
 00001111 1011111w mod reg r/m
 MOV_{ZX}
 00001111 1011011w mod reg r/m

Set Byte on Condition Instructions

The set byte on condition instructions set the destination byte to 1 if the condition is true and clear it if the condition is false.

Format: SET (condition) destination

Operation: The destination is either an **8-bit** register or memory. It is set to 1 if condition is true and to 0 if condition is false.

Flags: Affected-nom

Encoding: 00001111 opcode mod 000 r/m
 (the opcode byte is given in the following in hex)

Instruction	Set If	Condition	Opcode
SETA	above	CF = 0 and ZF = 0	97
SETAE	above or equal	CF = 0	93
SETB	below	CF = 1	92
SETBE	below or equal	CF = 1 or ZF = 1	96
SETC	carry	CF = 0	92
SCTE	equal	ZF = 1	94
SETGG	greater	ZF = 0 and SF = OF	9F
SETGE	greater or equal	ZF = OF	9D
SETL	less	(SF xor OF) = 1	9C
SETLE	less or equal	(SF xor OF) or ZF = 1	9E
SCTNA	not above	CF = 1 or ZF = 1	96
SETNAE	not above or equal	CF = 1	92
SETNB	not below	CF = 0	93
SETNBE	not below or equal	CF = 0 and ZF = 0	97
SETNC	not carry	CF = 0	93
SETNE	not equal	ZF = 0	95
SETNG	not greater	(SF xor OF) or ZF = 1	9E
SETNGE	not greater nor equal	(SF xor OF) = 1	9C
SETNL	not less	SF = OF	9D
SETNLE	not less nor equal	ZF = 0 and SF = OF	9F
SETNO	not overflow	OF = 0	91
SETNP	not parity	PF = 0	9B
SETNS	not sign	SF = 0	99
SETNZ	not zero	ZF = 0	95
SETO	overflow	OF = 1	90
SETP	parity	PF = 1	9A
SETPE	parity even	PF = 1	9A
SETPO	parity odd	PF = 0	9B
SETS	sign	SF = 1	98
SETZ	zero	ZF = 1	94

Double-Precision Shift Instructions

The **double-precision** shift instructions are SHLD (double-precision shift left) and SHRD (double-precision shift right).

Format: SHLD *destination*, *source*, *count*
or
SHRD *destination*, *source*, *count*

Operation: The destination is either register or memory, the source is a register, and both must **be** of the same size (either 16 or 32 bits). Count is **either** an 1-bit constant or CL. The count **specifies** the number of shifts for the destination. Instead of shifting in zeros as in the case of the single-precision shifts, **the** bits shifted into the **destination** are from the source. **However**, the source is not altered. **The** SF, ZF, and **PF** flags are set according to the result; CF is set to the last bit shifted out; **OF** and **AF** are undefined.

Flags: Affected—SF, ZF, PF, CF:
 Undefined—OF, AF
 Encoding: Count is immediate data:
 SHLD
 00001111 10100100 mod reg r/m [disp] data
 SHRD
 00001111 10101100 mod reg r/m [disp] data
 Count is Cl.:
 SHLD
 00001111 10100101 mod reg r/m [disp]
 SHRD
 00001111 10101101 mod reg r/m [disp]