
Section 7. 80286 and 80287 Microprocessor Instruction Sets

- 80286 Microprocessor Instruction Set 7-3
 - Data Transfer 7-3
 - Arithmetic 7-6
 - Logic 7-10
 - String Manipulation 7-12
 - Control Transfer 7-13
 - Processor Control 7-17
 - Protection Control 7-19
- 80287 Coprocessor Instruction Set 7-22
 - Data Transfer 7-22
 - Comparison 7-24
 - Constants 7-25
 - Arithmetic 7-26
 - Transcendental 7-27

Notes:

80286 Microprocessor Instruction Set

Data Transfer

MOV = move

Register to Register/Memory

1000100w	mod reg r/m
----------	-------------

Register/Memory to Register

1000101w	mod reg r/m
----------	-------------

Immediate to Register/Memory

1100011w	mod 000 r/m	data	data if w = 1
----------	-------------	------	---------------

Immediate to Register

1011wreg	data	data if w = 1
----------	------	---------------

Memory to Accumulator

1010000w	addr-low	addr-high
----------	----------	-----------

Accumulator to Memory

1010001w	addr-low	addr-high
----------	----------	-----------

Register/Memory to Segment Register

10001110	mod 0 reg r/m
----------	---------------

Segment Register to Register/Memory

10001100	mod 0 reg r/m
----------	---------------

PUSH = Push

Memory

11111111	mod 110 r/w
----------	-------------

Register

01010reg

Segment Register

000reg110

Immediate

011010s0	data	data if s = 0
----------	------	---------------

PUSHA = Push All

01100000

POP = Pop

Register/Memory

10001111	mod 000 r/m
----------	-------------

Register

01011reg

Segment Register

000reg111	reg \neq 01
-----------	---------------

POPA = Pop All

01100001

XCHG = Exchange

Register/Memory with Register

1000011w	mod reg r/m
----------	-------------

Register With Accumulator

10010reg

IN = Input From

Fixed Port

1110010w	port
----------	------

Variable Port

1110110w

OUT = Output To

Fixed Port

1110011w	port
----------	------

Variable Port

1110111w

XLAT = Translate Byte to AL

11010111

LEA = Load EA to Register

10001101	mod reg r/m
----------	-------------

LDS = Load Pointer to DS

11000101	mod reg r/m mod \neq 11
----------	---------------------------

LES = Load Pointer to ES

11000100	mod reg r/m mod \neq 11
----------	---------------------------

LAHF = Load AH with Flags

10011111

SAHF = Store AH with Flags

10011110

PUSHF = Push Flags

10011100

POPF = Pop Flags

10011101

Arithmetic

ADD = Add

Register/Memory with Register to Either

000000w	mod reg r/m
---------	-------------

Immediate to Register/Memory

100000sw	mod 000 r/m	data	data if sw = 01
----------	-------------	------	-----------------

Immediate to Accumulator

0000010w	data	data if w = 1
----------	------	---------------

ADC = Add with Carry

Register/Memory with Register to Either

0000100dw	mod reg r/m
-----------	-------------

Immediate to Register/Memory

100000sw	mod 000 r/m	data	data if sw = 01
----------	-------------	------	-----------------

Immediate to Accumulator

0001010w	data	data if w = 1
----------	------	---------------

INC = Increment

Register/Memory

1111111w	mod 000 r/m
----------	-------------

Register

01000reg

SUB = Subtract

Register/Memory with Register to Either

001010dw	mod reg r/m
----------	-------------

Immediate from Register/Memory

100000sw	mod 101 r/m	data	data if sw = 01
----------	-------------	------	-----------------

Immediate from Accumulator

0010110w	data	data if w = 1
----------	------	---------------

SBB = Subtract with Borrow

Register/Memory with Register to Either

000110dw	mod reg r/m
----------	-------------

Immediate to Register/Memory

100000sw	mod 011 r/m	data	data if sw = 01
----------	-------------	------	-----------------

Immediate to Accumulator

0001110w	data	data if w = 1
----------	------	---------------

DEC = Decrement

Register/Memory

1111111w	mod 001 r/m
----------	-------------

Register

01001reg

CMP = Compare

Register/Memory with Register

0011101w	mod reg r/m
----------	-------------

Register with Register/Memory

0011100w	mod reg r/m
----------	-------------

Immediate with Register/Memory

10000sw	mod 111 r/m	data	data if sw = 01
---------	-------------	------	-----------------

Immediate with Accumulator

0011110w	data	data if w = 1
----------	------	---------------

NEG = Change Sign

1111011w	mod 011 r/m
----------	-------------

AAA = ASCII Adjust for Add

00110111

DAA = Decimal Adjust for Add

00100111

AAS = ASCII Adjust for Subtract

00111111

DAS = Decimal Adjust for Subtract

00101111

MUL = Multiply (Unsigned)

1111011w	mod 100 r/m
----------	-------------

IMUL = Integer Multiply (Signed)

1111011w	mod 101 r/m
----------	-------------

IIMUL = Integer Immediate Multiply (Signed)

011010s1	mod reg r/m	data	data if s = 0
----------	-------------	------	---------------

DIV = Divide (Unsigned)

1111011w	mod 110 r/m
----------	-------------

IDIV = Integer Divide (Signed)

1111011w	mod 111 r/m
----------	-------------

AAM = ASCII Adjust for Multiply

11010100	00001010
----------	----------

AAD = ASCII Adjust for Divide

11010101	00001010
----------	----------

CBW = Convert Byte to Word

10011000

CWD = Convert Word to Doubleword

10011001

Logic

Shift/Rotate Instructions

Register/Memory by 1

1101000w	mod TTT r/m
----------	-------------

Register/Memory by CL

1101001w	mod TTT r/m
----------	-------------

Register/Memory by Count

1100000w	mod TTT r/m	count
----------	-------------	-------

TTT	Instruction
000	ROL
001	ROR
010	RCL
011	RCR
100	SHL/SAL
101	SHR
111	SAR

AND = And

Register/Memory and Register to Either

001000dw	mod reg r/m
----------	-------------

Immediate to Register/Memory

1000000w	mod 100 r/m	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

1000010w	mod reg r/m
----------	-------------

Immediate Data and Register/Memory

1111011w	mod 000 r/m	data	data if w = 1
----------	-------------	------	---------------

Immediate Data and Accumulator

1010100w	data	data if w = 1
----------	------	---------------

Or = Or

Register/Memory and Register to Either

000010dw	mod reg r/m
----------	-------------

Immediate to Register/Memory

1000000w	mod 001 r/m	data	data if w = 1
----------	-------------	------	---------------

Immediate to Accumulator

0000110w	data	data if w = 1
----------	------	---------------

XOR = Exclusive OR

Register/Memory and Register to Either

001100dw	mod reg r/m
----------	-------------

Immediate to Register/Memory

1000000w	mod 110 r/m	data	data if w = 1
----------	-------------	------	---------------

Immediate to Accumulator

0011010w	data	data if w = 1
----------	------	---------------

NOT = Invert Register/Memory

1111011w	mod 010 r/m
----------	-------------

String Manipulation

MOVS = Move Byte Word

1010010w

CMPS = Compare Byte Word

1010011w

SCAS = Scan Byte Word

1010111w

LODS = Load Byte Word to AL/AX

1010110w

STOS = Store Byte Word from AL/AX

1010101w

INS = Input Byte from DX Port

0110110w

OUTS = Output Byte Word to DX Port

0110111w

REP/REPNE, REPZ/REPNZ = Repeat String

Repeat Move String

11110011	1010010w
----------	----------

Repeat Compare String (z/Not z)

1111001z	1010011w
----------	----------

Repeat Scan String (z/Not z)

1111001z	1010111w
----------	----------

Repeat Load String

11110011	1010110w
----------	----------

Repeat Store String

11110011	1010101w
----------	----------

Repeat Input String

11110011	0110110w
----------	----------

Repeat Output String

11110011	0110111w
----------	----------

Control Transfer

CALL = Call

Direct Within Segment

11110100	disp-low	disp-high
----------	----------	-----------

Register/Memory Indirect Within Segment

11111111	mod 010 r/m
----------	-------------

Direct Intersegment

10011010	Segment Offset	Segment Selector
----------	----------------	------------------

Indirect Intersegment

11111111	mod 011 r/m (mod \neq 11)
----------	-----------------------------

JMP = Unconditional Jump

Short/Long

11101011	disp-low
----------	----------

Direct within Segment

11101001	disp-low	disp-high
----------	----------	-----------

Register/Memory Indirect Within Segment

11111111	mod 100 r/m
----------	-------------

Direct Intersegment

11101010	Segment Offset	Segment Selector
----------	----------------	------------------

Indirect Intersegment

11111111	mod 101 r/m (mod \neq 11)
----------	-----------------------------

RET = Return from Call

Within Segment

11000011

Within Segment Adding Immediate to SP

11000010	data-low	data-high
----------	----------	-----------

Intersegment

11001011

Intersegment Adding Immediate to SP

11001010	data-low	data-high
----------	----------	-----------

JE/JZ = Jump on Equal/Zero

01110100	disp
----------	------

JL/JNGE = Jump on Less/Not Greater, or Equal

01111100	disp
----------	------

JLE/JNG = Jump on Less, or Equal/Not Greater

01111110	disp
----------	------

JB/JNAE = Jump on Below/Not Above, or Equal

01110010	disp
----------	------

JBE/JNA = Jump on Below, or Equal/Not Above

01110110	disp
----------	------

JP/JPE = Jump on Parity/Parity Even

01111010	disp
----------	------

JO = Jump on Overflow

01110000	disp
----------	------

JS = Jump on Sign

01111000	disp
----------	------

JNE/JNZ = Jump on Not Equal/Not Zero

01110101	disp
----------	------

JNL/JGE = Jump on Not Less/Greater, or Equal

01111101	disp
----------	------

JNLE/JG = Jump on Not Less, or Equal/Greater

01111111	disp
----------	------

JNB/JAE = Jump on Not Below/Above, or Equal

01110011	disp
----------	------

JNBE/JA = Jump on Not Below, or Equal/Above

01110111	disp
----------	------

JNP/JPO = Jump on Not Parity/Parity Odd

01111011	disp
----------	------

JNO = Jump on Not Overflow

01110001	disp
----------	------

JNS = Jump on Not Sign

01111001	disp
----------	------

LOOP = Loop CX Times

11100010	disp
----------	------

LOOPZ/LOOPE = Loop while Zero/Equal

11100001	disp
----------	------

LOOPNZ/LOOPNE = Loop while Not Zero/Not Equal

11100000	disp
----------	------

JCXZ = Jump on CX Zero

11100011	disp
----------	------

ENTER = Enter Procedure

11001000	data-low	data-high	L
----------	----------	-----------	---

LEAVE = Leave Procedure

11001001

INT = Interrupt

Type Specified

11001101	type
----------	------

Type 3

11001100

INTO = Interrupt on Overflow

11001110

IRET = Interrupt Return

11001111

BOUND = Detect Value Out of Range

01100010	mod reg r/m
----------	-------------

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 Enable Flag

11111011

HLT = Halt

11110100

WAIT = Wait

10011011

LOCK = Bus Lock Prefix

11110000

CTS = Clear Task Switched Flag

00001111	00000110
----------	----------

ESC = Processor Extension Escape

11011TTT	mod LLL r/m
----------	-------------

Protection Control

LGDT = Load Global Descriptor Table Register

00001111	00000001	mod 010 r/m
----------	----------	-------------

SGDT = Store Global Descriptor Table Register

00001111	00000001	mod 000 r/m
----------	----------	-------------

LIDT = Load Interrupt Descriptor Table Register

00001111	00000001	mod 011 r/m
----------	----------	-------------

SIDT = Store Interrupt Descriptor Table Register

00001111	00000001	mod 001 r/m
----------	----------	-------------

LLDT = Load Local Descriptor Table Register from Register/Memory

00001111	00000000	mod 010 r/m
----------	----------	-------------

SLDT = Store Local Descriptor Table Register from Register/Memory

00001111	00000000	mod 000 r/m
----------	----------	-------------

LTR = Load Task Register from Register/Memory

00001111	00000000	mod 011 r/m
----------	----------	-------------

STR = Store Task Register to Register/Memory

00001111	00000000	mod 001 r/m
----------	----------	-------------

LMSW = Load Machine Status Word from Register/Memory

00001111	00000001	mod 110 r/m
----------	----------	-------------

SMSW = Store Machine Status Word

00001111	00000001	mod 100 r/m
----------	----------	-------------

LAR = Load Access Rights from Register/Memory

00001111	00000010	mod reg r/m
----------	----------	-------------

LSL = Load Segment Limit from Register/Memory

00001111	00000011	mod reg r/m
----------	----------	-------------

ARPL = Adjust Requested Privilege Level from Register/Memory

	01100011	mod reg r/m
--	----------	-------------

VERR = Verify Read Access; Register/Memory

00001111	00000000	mod 100 r/m
----------	----------	-------------

VERR = Verify Write Access

00001111	00000000	mod 101 r/m
----------	----------	-------------

The effective address (EA) of the memory operand is computed according to the mod and r/m fields:

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 the second byte of the instruction (before data if required).

Note: An exception to the above statements occurs when mod = 00 and r/m = 110, in which case EA = disp-high; disp-low.

Segment Override Prefix

001reg110

The 2-bit and 3-bit reg fields are defined as follows:

reg	Segment Register	reg	Segment Register
00	ES	10	SS
01	CS	11	DS

Figure 7-1. 2-Bit Register Field

16-bit (w = 1)	8-bit (w = 0)
000 AX	000 AL
001 CX	001 CL
010 DX	010 DL
011 BX	011 BL
100 SP	100 AH
101 BP	101 CH
110 SI	110 DH
111 DI	111 BH

Figure 7-2. 3-Bit Register Field

The physical addresses of all operands addressed by the BP register are computed using the SS segment register. The physical addresses of the destination operands of the string primitive operations (those addressed by the DI register) are computed using the ES segment, which may not be overridden.

80287 Coprocessor Instruction Set

The following is an instruction set summary for the 80287 math coprocessor.

The following figure shows abbreviations used in the summary.

Field	Description	Bit Information
escape	80286 Extension Escape	Bit Pattern = 11011
MF	Memory Format	00 = 32-bit Real 01 = 32-bit integer 10 = 64-bit Real 11 = 16-bit integer
ST(0)	Current Stack Top	
ST(i)	ith register below the stack top	
d	Destination	0 = Destination is ST(0) 1 = Destination is ST(i)
P	Pop	0 = No pop 1 = Pop ST(0)
R	Reverse*	0 = Destination (op) source 1 = Source (op) destination

* When d = 1, reverse the sense of R.

Figure 7-3. 80287 Encoding Field Summary

Data Transfer

FLD = Load

Integer/Real Memory to ST(0)

escape MF 1	mod 000 r/m
-------------	-------------

Long Integer Memory to ST(0)

escape 111	mod 101 r/m
------------	-------------

Temporary Real Memory to ST(0)

escape 011	mod 101 r/m
------------	-------------

BCD Memory to ST(0)

escape 111	mod 100 r/m
------------	-------------

ST(i) to ST(0)

escape 001	11000ST(i)
------------	------------

FST = Store

ST(0) to Integer/Real Memory

escape MF 1	mod 010 r/m
-------------	-------------

ST(0) to ST(i)

escape 101	11010 ST(i)
------------	-------------

FSTP = Store and Pop

ST(0) to Integer/Real Memory

escape MF 1	mod 011 r/m
-------------	-------------

ST(0) to Long Integer Memory

escape 111	mod 111 r/m
------------	-------------

ST(0) to Temporary Real Memory

escape 011	mod 111 r/m
------------	-------------

ST(0) to BCD Memory

escape 111	mod 110 r/m
------------	-------------

ST(0) to ST(i)

escape 101	11011 ST(i)
------------	-------------

FXCH = Exchange ST(i) and ST(0)

escape 001	11001 ST(i)
------------	-------------

Comparison

FCOM = Compare

Integer/Real Memory to ST(0)

escape MF 0	mod 010 r/m
-------------	-------------

ST(i) to ST(0)

escape 000	11010 ST(i)
------------	-------------

FCOMP = Compare and Pop

Integer/Real Memory to ST(0)

escape MF 0	mod 011 r/m
-------------	-------------

ST(i) to ST(0)

escape 000	11011 ST(i)
------------	-------------

FCOMPP = Compare ST(1) to ST(0) and Pop Twice

escape 110	11011001
------------	----------

FTST = Test ST(0)

escape 001	11100100
------------	----------

FXAM = Examine ST(0)

escape 001	11100101
------------	----------

Constants

FLDZ = Load + 0.0 into ST(0)

escape 001	11101110
------------	----------

FLD1 = Load + 1.0 into ST(0)

escape 001	11101000
------------	----------

FLDPI = Load π into ST(0)

escape 001	11101011
------------	----------

FLDL2T = Load $\log_2 10$ into ST(0)

escape 001	11101001
------------	----------

FLDL2E = Load $\log_2 e$ into ST(0)

escape 001	11101010
------------	----------

FLDLG2 = Load $\log_{10} 2$ into ST(0)

escape 001	11101100
------------	----------

FLDLN2 = Load $\log_e 2$ into ST(0)

escape 001	11101101
------------	----------

Arithmetic

FADD = Addition

Integer/Real Memory with ST(0)

escape MF 0	mod 000 r/m
-------------	-------------

ST(i) and ST(0)

escape dP0	11000 ST(i)
------------	-------------

FSUB = Subtraction

Integer/Real Memory with ST(0)

escape MF 0	mod 10R r/m
-------------	-------------

ST(i) and ST(0)

escape dP0	1110R r/m
------------	-----------

FMUL = Multiplication

Integer/Real Memory with ST(0)

escape MF 0	mod 001 r/m
-------------	-------------

ST(i) and ST(0)

escape dP0	11001 r/m
------------	-----------

FDIV = Division

Integer/Real Memory with ST(0)

escape MF 0	mod 11R r/m
-------------	-------------

ST(i) and ST(0)

escape dP0	1111R r/m
------------	-----------

FSQRT = Square Root of ST(0)

escape 001	11111010
------------	----------

FSCALE = Scale ST(0) by ST(1)

escape 001	11111101
------------	----------

FPREM = Partial Remainder of ST(0) ÷ ST(1)

escape 001	11111000
------------	----------

FRNDINT = Round ST(0) to Integer

escape 001	11111100
------------	----------

FEXTRACT = Extract Components of ST(0)

escape 001	11110100
------------	----------

FABS = Absolute Value of ST(0)

escape 001	11100001
------------	----------

FCHS = Change Sign of ST(0)

escape 001	11100000
------------	----------

Transcendental

FPTAN = Partial Tangent of ST(0)

escape 001	11110010
------------	----------

FPATAN = Partial Arctangent of ST(1) ÷ ST(0)

escape 001	11110011
------------	----------

F2XM1 = $2^{ST(0)} - 1$

escape 001	11110000
------------	----------

FYL2X = ST(1) x Log₂ [ST(0)]

escape 001	11110001
------------	----------

FYL2XP1 = ST(1) x Log₂ [ST(0) + 1]

escape 001	11111001
------------	----------

FINIT = Initialize NPX

escape 011	11100011
------------	----------

FSETPM = Enter Protected Mode

escape 011	11100100
------------	----------

FSTSW AX = Store Control Word

escape 111	11100000
------------	----------

FLDCW = Load Control Word

escape 001	mod 101 r/m
------------	-------------

FSTCW = Store Control Word

escape 001	mod 111 r/m
------------	-------------

FSTSW = Store Status Word

escape 101	mod 111 r/m
------------	-------------

FCLEX = Clear Exceptions

escape 011	11100010
------------	----------

FSTENV = Store Environment

escape 001	mod 110 r/m
------------	-------------

FLDENV = Load Environment

escape 001	mod 100 r/m
------------	-------------

FSAVE = Save State

escape 101	mod 110 r/m
------------	-------------

FRSTOR = Restore State

escape 101	mod 100 r/m
------------	-------------

FINCSTP = Increment Stack Pointer

escape 001	11110111
------------	----------

FDECSTP = Decrement Stack Pointer

escape 001	111100110
------------	-----------

FFREE = Free ST(i)

escape 101	11000ST(i)
------------	------------

FNOP = No Operation

escape 001	11010000
------------	----------

Notes: