Table 5-3. 8086 Memory Addressing Options Identified by the EA Abbreviations in Tables 5-4, 5-5, and 5-6

Memory Reference	Segment Register	Base Register	Index Register	Possible Displacements			Assembly Language Operand Mnemonic
				16-Bit Unsigned	8-Bit High-order Bit Extended	None	
Normal Data Memory Reference	DS (Alternate* CS, SS or ES)	None	SI	X	X	X	
			DI	X	X	X	
		BX	SI	X	X	X	
			DI	X	X	X	
			None	X	X	X	
	DS	None	None	X			
	SS (Alternate* CS, DS or ES)	BP	SI	X	X	X	
			DI	X	X	X	
			None	X	X		
Stack	SS	SP	None				
String Data	DS	None	SI				
	ES	None	DI				
Instruction Fetch	CS	PC	None				
Branch	CS	PC	None		X		
I/O Data	DS	DX	None				
		These columns contribute to OEA.					This column to be provided
		These columns contribute to EA.					

Shaded rows apply to EA and DADDR.

Shaded row applies to EA and LABEL.

* The segment override allows DS or SS to be replaced by one of the other segment registers

X These are displacements that can be used to compute memory addresses.

The following abbreviations are used in Tables 5-4 and 5-5:

AH	Accumulator, high-order byte
AL	Accumulator, low-order byte
AL7	The value of register AL high-order bit (0 or 1) extended to a byte (00_{16} or FF_{16})
AX	Accumulator, both bytes
AX15	The value of register AH high-order bit (0 or 1) extended to a 16-bit word (0000_{16} or $FFFF_{16}$)
BD	The destination is a byte operand (used only by the Assembler)
BH	B register, high-order byte
BL	B register, low-order byte
BRANCH	Program memory direct address, used in Branch addressing option shown in Tables 5-1 and 5-2
BS	The source is a byte operand (used only by the Assembler)
BX	B register, both bytes
C	Carry status
CH	C register, high-order byte
CL	C register, low-order byte
CS	Code Segment register
CX	C register, both bytes
DADDR	Data memory address operands identified in Table 5-3
DATA8	Eight bits of immediate data
DATA16	16 bits of immediate data
DH	D register, high-order byte
DI	Destination Index register
DISP	An 8-bit or 16-bit signed displacement
DISP8	An 8-bit signed displacement
DL	D register, low-order byte
DS	Data Segment register
DX	D register, both bytes
EA	Effective data memory address using any of the memory addressing options identified in Table 5-2
ES	Extra Segment register
I	Status flag set to 1
I/D	Increment/decrement selector for string operations; increment if D is 0, decrement if D is 1
LABEL	Direct data memory address, as identified in Table 5-2
N	A number between 0 and 7
O	Status flag reset to 0
OEA	Offset data memory address used to compute EA: EA =OEA + [DS] • 16
PC	Program Counter
PDX	I/O port addressed by DX register contents; port number can range from 0 through 65,536
PORT	A label identifying an I/O port number in the range 0 through 255_{10}
RB	Any one of the eight byte registers: AH, AL, BH, BL, CH, CL, DH, or DL
RBD	Any RB register as a destination
RBS	Any RB register as a source
RW	Any one of the eight 16-bit registers: AX, BX, CX, DX, SP, BP, SI, or DI
RWD	Any RW register as a destination
RWS	Any RW register as a source
SEGM	Label identifying a 16-bit value loaded into the CS Segment register to execute a segment jump
SFR	Status Flags register
SI	Source Index register
SP	Stack Pointer
SR	Any one of the Segment registers CS, DS, ES, or SS
SS	Stack Segment register

U	Status flag modified, but undefined
V	Any number in the range 0 through 255_{10}
X	Status flag modified to reflect result
WD	The destination is a word operand (used only by the Assembler)
WS	The source is a word operand (used only by the Assembler)
[[ ]]	Contents of the memory location addressed by the contents of the location enclosed in the double brackets
[ ]	The contents of the location enclosed in the brackets
←	Data on the right-hand side of the arrow is moved to the location on the left-hand side of the arrow
←→	Contents of locations on each side of ←→ are exchanged
—	The twos complement of the value under the —
≠	Not equal to

INSTRUCTION EXECUTION TIMES AND CODES

Table 5-5 lists instructions in alphabetical order, showing object codes and execution times, for the 8086 and the 8088, expressed in whole clock cycles. Execution time is the time required from beginning execution of an instruction that is in the queue to beginning execution of the next instruction in the queue. The time required to place an instruction from memory into the queue (instruction fetch time) is not shown in the table; because of queuing, instruction fetch time occurs concurrently with instruction execution time and thus has no effect on overall timing, except as specifically noted in the table.

Instruction object codes are represented as two hexadecimal digits for instruction bytes without variations.

Instruction object codes are represented as eight binary digits for instruction bytes with variations for the instruction.

The following notation is used in Tables 5-4 and 5-5:

[ ]	indicate an optional object code byte
a	one bit choosing length: in bit position 0 a=0 specifies 1 data byte; a=1 specifies 2 data bytes in bit position 1 a=0 specifies 2 data bytes: a=1 specifies 1 data byte
aa	two bits choosing address length: no DISP = 00 one DISP byte = 01 two DISP bytes = 10, or 00 with bbb = 110 11 causes bbb to select a register, using the 3-bit code given below for reg.
bbb	three bits choosing addressing mode: 000 EA = (BX) + (SI) + DISP 001 EA = (BX) + (DI) + DISP 010 EA = (BP) + (SI) + DISP 011 EA = (BP) + (DI) + DISP 100 EA = (SI) + DISP 101 EA = (DI) + DISP 110 EA = (BP) + DISP 111 EA = (BX) + DISP
DISP	represents two hexadecimal digit memory displacement
ddd	represents three binary digits identifying a destination register (see reg.)
rr	two binary digits identifying a segment register: 00 = ES 01 = CS 10 = SS 11 = DS
reg	three binary digits identifying a register:

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

sss	represents three binary digits identifying a source register (see reg)
PPQQ	represents four hexadecimal digit memory address
v	one bit choosing shift length: 0 count = 1 1 count = (CL)
x	"don't care" bit
YY	represents two hexadecimal data digits
YYYY	represents four hexadecimal data digits
z	one bit where z XOR (ZF) = 1 terminates loop
*	Execution time is less than or equal to instruction fetch time.
**	Includes up to eight clock cycles of overhead on each transfer due to queue maintenance. For conditional jumps, the lesser figure is when the test fails (no jump taken).

Effective Address calculation and extra clock cycles:

Extra Clock Periods			
bbb	EA	8086(1)	8088(2)
000	(BX) + (SI)	7	7
000	(BX) + (SI) + DISP8	11	11
000	(BX) + (SI) + DISP16	11	15
001	(BX) + (DI)	8	8
001	(BX) + (DI) + DISP8	12	12
001	(BX) + (DI) + DISP16	12	16
010	(BP) + (SI)	8	8
010	(BP) + (SI) + DISP8	12	12
010	(BP) + (SI) + DISP16	12	16
011	(BP) + (DI)	7	7
011	(BP) + (DI) + DISP8	11	11
011	(BP) + (DI) + DISP16	11	15
100	(SI) ir (DI) or (BD)	5	5
101	or (BX)		
110	+ DISP8	9	9
111	+ DISP16	9	13
	8-bit immediate	6	6
	16-bit immediate	6	10

(1) Add another 4 clock cycles for each 16-bit operand or an odd address boundary.

(2) Add anoter 4 clock cycles for each 16-bit operand.

Substitute the clock cycles shown above wherever EA appears in Tables 5-4 and 5-5.

Table 5-4. A Summary of 8086 and 8088 Instructions

Type	Mnemonic	Operand(s)	Object Code	Clock Cycles	Statuses								Operation Performed
					O	D	I	T	S	Z	A	P	C	
I/O	IN	AL,PORT	E4 YY	10										[AL] ← [PORT] Load one byte of data from I/O port PORT into AL
	IN	AL,[DX]	EC 1	8										[AL] ← [PDX] Load into AL one byte of data from I/O port whose address is held in the DX register
	IN	AX,PORT	E5 YY	10										[AL] ← [PORT], [AH] ← [PORT+1] Load 16 bits of data into AX, AL receives data from I/O port PORT, AH receives data from I/O port PORT+1
	IN	AX,[DX]	ED	8										[AL] ← [PDX], [AH] ← [PDX+1] Load 16 bits of data into AX, AL receives data from I/O port whose address is held in the DX register. AH receives data from the I/O port whose address is one higher
	OUT	AL,PORT	E6 YY	10										[PORT] ← [AL] Output one byte of data from register AL to I/O port PORT
	OUT	AL,[DX]	EE 1	8										[PDX] ← [AL] Output one byte of data from register AL to the I/O port whose address is held in the DX register
	OUT	AX,PORT	E7 YY	10										[PORT] ← [AL], [PORT+1] ← [AH] Output 16 bits of data. The AL register contents are output to I/O port PORT. The AH register contents are output to I/O port PORT+1
	OUT	AX,[DX]	EF	8										[PORT] ← [PDX], [PORT+1] ← [PDX+1] Output 16 bits of data. The AL register contents are output to the I/O port whose address is held in the DX register. The AH register contents is output to the I/O port whose address is one higher
Primary Memory Reference	LDS	RW,DADDR	C5 aasssbbb [DISP][DISP]	16+EA										[RW] ← [EA], [DS] ← [EA+2] Load 16 bits of data from the memory word addressed by DADDR into register RW. Load 16 bits of data from the next sequential memory word into the DS register
	LEA	RW,DADDR	8D aasssbbb [DISP][DISP]	2+EA										[RW] ← OEA Load into RW the 16-bit address displacement which, when added to the segment register contents, creates the effective data memory address
	LES	RW,DADDR	C4 aasssbbb [DISP][DISP]	16+EA										[RW] ← [EA], [ES] ← [EA+2] Load 16 bits of data from the memory word addressed by DADDR into register RW. Load 16 bits of data from the next sequential memory word into the ES register
	MOV	RB,DADDR	8A aedddbbb [DISP][DISP]	8+EA										[RB] ← [EA] Load one byte of data from the data memory location addressed by DADDR to register RB

Table 5-4. A Summary of 8086 and 8088 Instructions (Continued)

Type	Mnemonic	Operand(s)	Object Code	Clock Cycles	Statuses								Operation Performed
					O	D	I	T	S	Z	A	P	C	
Primary Memory Reference (Continued)	MOV	RW,DADDR	8B aedddbbb [DISP][DISP]	8+EA										[RW] ← [EA] Load 16 bits of data from the data memory word addressed by DADDR to register RW
	MOV	DADDR,RB	88 aasssbbb [DISP][DISP]	9+EA										[EA] ← [RB] Store the data byte from register RB in the memory byte addressed by DADDR
	MOV	DADDR,RW	89 aasssbbb [DISP][DISP]	9+EA										[EA] ← [RW] Store the 16-bit data word from register RW in the memory word addressed by DADDR
	MOV	AL,LABEL	A0 PPQQ	10										[AL] ← [EA] Load the data memory byte directly addressed by LABEL into register AL
	MOV	AX,LABEL	A1 PPQQ	10										[AX] ← [EA] Load the 16-bit data memory word directly addressed by LABEL into register AX
	MOV	LABEL,AL	A2 PPQQ	10										[EA] ← [AL] Store the 8-bit contents of register AL into the data memory byte directly addressed by LABEL
	MOV	LABEL,AX	A3 PPQQ	10										[EA] ← [AX] Store the 16-bit contents of register AX into the data memory word directly addressed by LABEL
	MOV	SR,DADDR	8E aaOrrbbb [DISP][DISP]	8+EA										[SR] ← [EA] Load into Segment register SR the contents of the 16-bit memory word addressed by DADDR
	MOV	DADDR,SR	8C aaOrrbbb [DISP][DISP]	9+EA										[EA] ← [SR] Store the contents of Segment register SR in the 16-bit memory location addresed by DADDR
	XCHG	RB,DADDR	86 aaregbbb [DISP][DISP]	17+EA										[RB] ←→ [EA] Exchange a byte of data between register RB and the data memory location addressed by DADDR
	XCHG	RW,DADDR	87 aaregbbb [DISP][DISP]	17+EA										[RW] ←→ [EA] Exchange 16 bits of data between register RW and the data memory location addressed by DADDR
	XLAT		D7	11										[AL] ← [[AL] + [BX]] Load into AL the data byte stored in the memory location addressed by summing initial AL contents with BX contents

Table 5-4. A Summary of 8086 and 8088 Instructions (Continued)

Type	Mnemonic	Operand(s)	Object Code	Clock Cycles	Statuses								Operation Performed
					O	D	I	T	S	Z	A	P	C	
Secondary Memory Reference (Memory Operate)	ADC	RB,DADDR	12 aadddbbb [DISP][DISP]	9+EA	X				X	X	X	X	X	[RB] ← [EA] + [RB] + [C] Add the contents of the data byte addressed by DADDR, plus the Carry status, to register RB
	ADC	RW,DADDR	13 aadddbbb [DISP][DISP]	9+EA	X				X	X	X	X	X	[RW] ← [EA] + [RW] + [C] Add the contents of the 16-bit data word addressed by DADDR, plus the Carry status, to register RW
	ADC	DADDR,RB	10 aasssbbb [DISP][DISP]	16+EA	X				X	X	X	X	X	[EA] ← [EA] + [RB] + [C] Add the 8-bit contents of register RB, plus the Carry status, to the data memory byte addressed by DADDR
	ADC	DADDR,RW	11 aasssbbb [DISP][DISP]	16+EA	X				X	X	X	X	X	[EA] ← [EA] + [RW] + [C] Add the 16-bit contents of register RW, plus the Carry status, to the data word addressed by DADDR
	ADD	RB,DADDR	02 aadddbbb [DISP][DISP]	9+EA	X				X	X	X	X	X	[RB] ← [EA] + [RB] Add the contents of the data byte addressed by DADDR to register RB
	ADD	RW,DADDR	03 aadddbbb [DISP][DISP]	9+EA	X				X	X	X	X	X	[RW] ← [EA] + [RW] Add the contents of the 16-bit word addressed by DADDR to register RW
	ADD	DADDR,RB	00 aasssbbb [DISP][DISP]	16+EA	X				X	X	X	X	X	[EA] ← [EA] + [RB] Add the 8-bit contents of register RB to the data memory byte addressed by DADDR
	ADD	DADDR,RW	01 aasssbbb [DISP][DISP]	16+EA	X				X	X	X	X	X	[EA] ← [EA] + [RW] Add the 16-bit contents of register RW to the data memory word addressed by DADDR
	AND	RB,DADDR	22 aadddbbb [DISP][DISP]	9+EA	0				X	X	U	X	0	[RB] ← [EA] AND [RB] AND the 8-bit contents of register RB with the data memory byte addressed by DADDR. Store the result in RB
	AND	RW,DADDR	23 aadddbbb [DISP][DISP]	9+EA	0				X	X	U	X	0	[RW] ← [EA] AND [RW] AND the 16-bit contents of register RW with the data memory word addressed by DADDR. Store the result in RW
	AND	DADDR,RB	20 aasssbbb [DISP][DISP]	16+EA	0				X	X	U	X	0	[EA] ← [EA] AND [RB] AND the 8-bit contents of register RB with the data memory byte addressed by DADDR. Store the result in the addressed data memory byte
	AND	DADDR,RW	21 aasssbbb [DISP][DISP]	16+EA	0				X	X	U	X	0	[EA] ← [EA] AND [RW] AND the 16-bit contents of register RW with the data memory word addressed by DADDR. Store the result in the addressed data memory word
	CMP	RB,DADDR	3A aadddbbb [DISP][DISP]	9+EA	X				X	X	X	X	X	[RB] − [EA] Subtract the contents of the data memory byte addressed by DADDR from the contents of register RB. Discard the result, but adjust status flags
	CMP	RW,DADDR	3B aadddbbb [DISP][DISP]	9+EA	X				X	X	X	X	X	[RW] - [EA] Subtract the 16-bit contents of the data memory word addressed by DADDR from the contents of register RW. Discard the result, but adjust status flags

Table 5-4. A Summary of 8086 and 8088 Instructions (Continued)

Type	Mnemonic	Operand(s)	Object Code	Clock Cycles	Statuses								Operation Performed
					O	D	I	T	S	Z	A	P	C	
Secondary Memory Reference (Memory Operate) (Continued)	CMP	DADDR,RB	38 aasssbbb [DISP][DISP]	9+EA	X				X	X	X	X	X	[EA] − [RB] Subtract the 8-bit contents of register RB from the data memory byte addressed by DADDR. Discard the result, but adjust status flags
	CMP	DADDR,RW	39 aasssbbb [DISP][DISP]	9+EA	X				X	X	X	X	X	[EA] − [RW] Subtract the 16-bit contents of register RW from the data memory word addressed by DADDR. Discard the result, but adjust status flags
	DEC	DADDR	1111111a aa001bbb [DISP][DISP]	15+EA	X				X	X	X	X		[EA] ← [EA] − 1 Decrement the contents of the memory location addressed by DADDR. Depending on the prior definition of DADDR, an 8-bit or a 16-bit memory location may be decremented
	DIV	AX,DADDR	F6 aa110bbb [DISP][DISP]	(86-96)+EA	U				U	U	U	U	U	[AX] ← [AX]/[EA] Divide the 16-bit contents of register AX by the 8-bit contents of the memory byte addressed by DADDR. Store the integer quotient in AL and the remainder in AH. If the quotient is greater than FF_{16}, execute a "divide by 0" interrupt
	DIV	DX,DADDR	F7 aa110bbb [DISP][DISP]	(150-168)+EA	U				U	U	U	U	U	[DX] [AX] ← [DX] [AX]/[EA] Divide the 32-bit contents of registers DX (high-order) and AX (low-order) by the 16-bit contents of the memory word addressed by DADDR. Store the integer quotient in AX and the remainder in DX. If the quotient is greater than $FFFF_{16}$, execute a "divide by 0" interrupt
	IDIV	AX,DADDR	F6 aa111bbb [DISP][DISP]	(107-118)8+EA	U				U	U	U	U	U	[AX] ← [AX]/[EA] Divide the 16-bit contents of register AX by the 8-bit contents of the memory byte addressed by DADDR, treating both contents as signed binary numbers. Store the quotient, as a signed binary number, in AL. Store the remainder, as an unsigned binary number, in AH. If the quotient is greater than $7F_{16}$, or less than -80_{16}, execute a "divide by 0" interrupt
	IDIV	DX,DADDR	F7 aa111bbb [DISP][DISP]	(171)-190)+EA	U				U	U	U	U	U	[DX] [AX] ← [DX] [AX]/[EA] Divide the 32-bit contents of register DX (high-order) and AX (low-order) by the 16-bit contents of the memory word addressed by DADDR. Treat both contents as signed binary numbers. Store the quotient, as a signed binary number, in AX. Store the remainder, as an unsigned binary number, in AH. If the quotient is greater than $7FFF_{16}$, or less than -8000_{16}, execute a "divide by 0" interrupt
	IMUL	AL,DADDR	F6 aa101bbb [DISP][DISP]	(86-104)+EA	X				U	U	U	U	X	[AX] ← [AL] • [EA] Multiply the 8-bit contents of register AL by the contents of the memory byte addressed by DADDR. Treat both numbers as signed binary numbers. Store the 16-bit product in AX
	IMUL	AX,DADDR	F7 aa101bbb [DISP][DISP]	(134-160)+EA	X				U	U	U	U	X	[DX] [AX] ← [AX] • [EA] Multiply the 16-bit contents of register AX by the 16-bit contents of the memory word addressed by DADDR. Treat both numbers as signed binary numbers. Store the 32-bit product in DX (high-order word) and AX (low-order word)

Table 5-4. A Summary of 8086 and 8088 Instructions (Continued)

Type	Mnemonic	Operand(s)	Object Code	Clock Cycles	Statuses O	D	I	T	S	Z	A	P	C	Operation Performed
Secondary Memory Reference (Memory Operate) (Continued)	INC	DADDR	1111111a aa000bb [DISP][DISP]	15+EA	X				X	X	X	X		[EA] ← [EA] + 1 Increment the contents of the memory location addressed by DADDR. Depending on the prior definition of DADDR, an 8-bit or a 16-bit memory location may be incremented
	MUL	AL,DADDR	F6 aa100bbb [DISP][DISP]	(76-83)+EA	X				U	U	U	U	X	[AX] ← [AL] * [EA] Multiply the 8-bit contents of register AL by the contents of the memory byte addressed by DADDR. Treat both numbers as unsigned binary numbers. Store the 16-bit product in AX
	MUL	F7	F7 aa100bbb [DISP][DISP]	(124-139)+EA	X				U	U	U	U	X	[DX] [AX] ← [AX] * [EA] Multiply the 16-bit contents of register AX by the 16-bit contents of the memory word addressed by DADDR. Treat both numbers as unsigned binary numbers. Store the 32-bit product in DX (high-order word) and AX (low-order word)
	NEG	DADDR	1111011a aa011bb [DISP][DISP]	16+EA	X				X	X	X	X	X	[EA] ← $\overline{[EA]}$ Twos complement the contents of the addressed memory location. Depending on the prior definition of DADDR, an 8-bit or 16-bit memory location may be twos complemented
	NOT	DADDR	1111011a aa010bbb [DISP][DISP]	16+EA										[EA] ← NOT [EA] Ones complement the contents of the addressed memory location. Depending on the prior definition of DADDR, an 8-bit or 16-bit memory location may be ones complemented
	OR	RB,DADDR	0A aadddbbb [DISP][DISP]	9+EA	X				X	X	U	X	X	[RB] ← [EA] OR [RB] OR the 8-bit contents of register RB with the data memory byte addressed by DADDR. Store the result in RB
	OR	RW,DADDR	0B aadddbbb [DISP][DISP]	9+EA	X				X	X	U	X	X	[RW] ← [EA] OR [RW] OR the 16-bit contents of register RW with the data memory word addressed by DADDR. Store the result in RW
	OR	DADDR,RB	08 aassbbb [DISP][DISP]	16+EA	X				X	X	U	X	X	[EA] ← [EA] OR [RB] OR the 8-bit contents of register RB with the data memory byte addressed by DADDR. Store the result in the data memory byte
	OR	DADDR,RW	09 aassbbb [DISP][DISP]	16+EA	X				X	X	U	X	X	[EA] ← [EA] OR [RW] OR the 16-bit contents of register RW with the data memory word addressed by DADDR. Store the result in the data memory word

Table 5-4. A Summary of 8086 and 8088 Instructions (Continued)

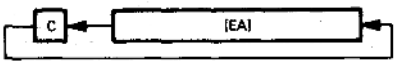
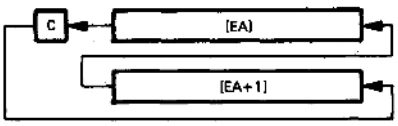
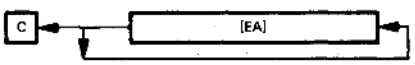
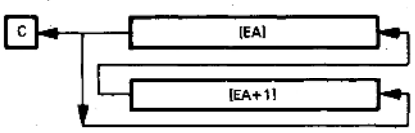
Type	Mnemonic	Operand(s)	Object Code	Clock Cycles	Statuses O	D	I	T	S	Z	A	P	C	Operation Performed
Secondary Memory Reference (Memory Operate) (Continued)	RCL	DADDR,N	110100va aa011bbb [DISP][DISP]	N=1 16+EA; N>1 4N+20+EA	X								X	Rotate the contents of the data memory location addressed by DADDR left through the Carry status. If N = 1, then rotate one bit position. If N = CL, then register CL contents provide the number of bit positions. Depending on prior definition, DADDR may address a byte:  or DADDR may address a word: 
	ROL	DADDR,N	110100va aa000bbb											
	RCR	DADDR,N	110100va aa001bbb [DISP][DISP]	N=1 15+EA	X								X	As RCL, but rotate right
	ROL	DADDR,N	110100va aa000bbb [DISP][DISP]	N>1 4N+20+EA	X								X	Rotate the contents of the data memory location addressed by DADDR left. Move the left most bit into the Carry status. If N = 1, then rotate one bit position. If N = CL, then register CL contents provides the number of bit positions. Depending on prior definition, DADDR may address a byte:  or DADDR may address a word: 

Table 5-4. A Summary of 8086 and 8088 Instructions (Continued)

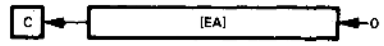
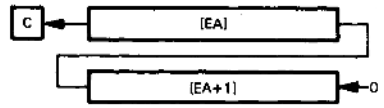
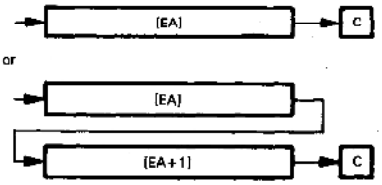
Type	Mnemonic	Operand(s)	Object Code	Clock Cycles	Statuses O	D	I	T	S	Z	A	P	C	Operation Performed
Secondary Memory Reference (Memory Operate) (Continued)		DADDR,N	110100va aa001bbb [DISP][DISP]	N=1 15+EA	X								X	As ROL, but rotate right
	SAL													Shift the contents of the data memory location addressed by DADDR left. Move the left most bit into the Carry status. If N = 1, then shift one bit position. If N = CL, then register CL contents provides the number of bit positions. Depending on prior definition, DADDR may address a byte:  or DADDR may address a word: 
	SAR	DADDR,N	110100va aa111bbb [DISP][DISP]	N=1 15+EA; N>1 4N+20+EA	X				X	X	U	X	X	As SAL, but shift right and propagate sign: 
	SBB	RB,DADDR	1A aaddd bbb [DISP][DISP]	9+EA	X				X	X	X	X	X	[RB] ← [RB] − [EA] − [C] Subtract the contents of the data byte addressed by DADDR from the contents of 8-bit register RB, using twos complement arithmetic. Decrement the result in RB if the Carry status was initially set
	SBB	RW,DADDR	1B aadddbbb [DISP][DISP]	9+EA	X				X	X	X	X	X	[RW] ← [RW] − [EA] − [C] Subtract the contents of the 16-bit data word addressed by DADDR from the contents of the 16-bit register RW, using twos complement arithmetic. Decrement the result in RW if the Carry status was initially set

Table 5-4. A Summary of 8086 and 8088 Instructions (Continued)

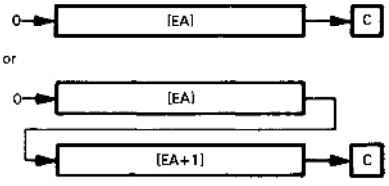
Type	Mnemonic	Operand(s)	Object Code	Clock Cycles	Statuses O	D	I	T	S	Z	A	P	C	Operation Performed
Secondary Memory Reference (Memory Operate) (Continued)	SBB	DADDR,RB	18 aasssbbb [DISP][DISP]	16+EA	X				X	X	X	X	X	[EA] ← [EA] − [RB] − [C] Subtract the contents of 8-bit register RB from the data byte addressed by DADDR, using twos complement arithmetic. Decrement the result in data memory if the Carry status was initially set
	SBB	DADDR,RW	19 aasssbbb [DISP][DISP]	16+EA	X				X	X	X	X	X	[EA] ← [EA] − [RW] − [C] Subtract the contents of 16-bit register RW from the 16-bit data word addressed by DADDR, using twos complement arithmetic. Decrement the result in data memory if the Carry status was initially set
	SHL	DADDR,N			X				X	X	U	X	X	This is an alternate mnemonic for SAL
	SHR	DADDR,N	110100va aa101bb [DISP][DISP]	N=1 15+EA; N>1 4N+20+EA	X				X	X	U	X	X	As SAL, but shift right: 
	SUB	RB,DADDR	2A aadddbbb [DISP][DISP]	9+EA	X				X	X	X	X	X	[RB] ← [RB] − [EA] Subtract the contents of the data memory byte addressed by DADDR from the contents of 8-bit register RB, using twos complement arithmetic
	SUB	RW,DADDR	2B aadddbbb [DISP][DISP]	9+EA	X				X	X	X	X	X	[RW] ← [RW] − [EA] Subtract the contents of the 16-bit data memory word addressed by DADDR from the contents of 16-bit register RW, using twos complement arithmetic
	SUB	DADDR,RB	28 aasssbbb [DISP][DISP]	16+EA	X				X	X	X	X	X	[EA] ← [EA] [RB] Subtract the contents of 8-bit register RB from the data memory byte addressed by DADDR, using twos complement arithmetic
	SUB	DADDR,RW	29 aasssbbb [DISP][DISP]	16+EA	X				X	X	X	X	X	[EA] ← [EA] − [RW] Subtract the contents of 16-bit register RW from the 16-bit data memory word addressed by DADDR, using twos complement arithmetic
	TEST	DADDR,RB	84 aaregbbb [DISP][DISP]	9+EA	0				X	X	U	X	0	[EA] AND [RB] AND the 8-bit contents of the data memory location addressed by DADDR with the contents of 8-bit register RB. Discard the result, but adjust status flags appropriately

Table 5-4. A Summary of 8086 and 8088 Instructions (Continued)

Type	Mnemonic	Operand(s)	Object Code	Clock Cycles	Statuses O	D	I	T	S	Z	A	P	C	Operation Performed
Secondary Memory Reference (Memory Operate) (Continued)	TEST	DADDR,RW	85 aareg bbb [DISP][DISP]	9+EA	0				X	X	U	X	0	(EA) AND (RW) AND the 16-bit contents of the data memory word addressed by DADDR with the contents of 16-bit register RW. Discard the result, but adjust status flags appropriately
	XOR	RB,DADDR	32 aadddbbb [DISP][DISP]	9+EA	0				X	X	U	X	0	(RB) ← (RB) XOR (EA) Exclusive OR the 8-bit contents of register RB with the data memory byte addressed by DADDR. Store the result in RB
	XOR	RW,DADDR	33 aadddbbb [DISP][DISP]	9+EA	0				X	X	U	X	0	(RW) ← (RW) XOR (EA) Exclusive OR the 16-bit contents of register RW with the 16-bit data memory word addressed by DADDR. Store the result in RW
	XOR	DADDR,RB	30 aaaaabbb [DISP][DISP]	16+EA	0				X	X	U	X	0	(EA) ← (RB) XOR (EA) Exclusive OR the 8-bit contents of register RB with the data memory byte addressed by DADDR. Store the result in the addressed data memory byte
	XOR	DADDR,RW	31 aaaaabbb [DISP][DISP]	16+EA	0				X	X	U	X	0	(EA) ← (RW) XOR (EA) Exclusive OR the 16-bit contents of register RW with the data memory word addressed by DADDR. Store the result in the addressed data memory word
Immediate	MOV	DADDR, DATA8	C6 aa000bbb [DISP][DISP] YY	10+EA										(EA) ← DATA8 Load the immediate data byte DATA8 into the data memory byte addressed by DADDR
	MOV	DADDR, DATA16	C7 aa000bbb [DISP][DISP] YYYY	10+EA										(EA) ← DATA16 Load the immediate 16-bit data word DATA16 into the data memory word addressed by DADDR
	MOV	RB,DATA8	10110ddd YY	4*										(RB) ← DATA8 Load the immediate data byte DATA8 into 8-bit register RB
	MOV	RW,DATA16	10111ddd YYYY	4*										(RW) ← DATA16 Load the immediate 16-bit data word DATA16 into 16-bit register RW
Jump	JMP	BRANCH	111010a1 DISP [DISP]	15**										(PC) ← (PC) + DISP Jump direct to program memory location identified by label BRANCH. The displacement DISP which must be added to the Program Counter will be computed as an 8-bit or 16-bit signed binary number, as needed, by the assembler
	JMP	BRANCH, SEGM	EA PPQQ PPQQ	15**										(PC) ← DATA16, (CS) ← DATA16 Jump direct into a new segment. BRANCH is a label which becomes a 16-bit unsigned data value which is loaded into PC. SEGM is a label which becomes another 16-bit unsigned data value that is loaded into the CS segment register
	JMP	DADDR	FF aa100bbb [DISP][DISP]	18+EA**										(PC) ← (EA) Jump indirect in current segment. The 16-bit contents of the data memory word addressed by DADDR is loaded into PC

Table 5-4. A Summary of 8086 and 8088 Instructions (Continued)

Type	Mnemonic	Operand(s)	Object Code	Clock Cycles	Statuses O	D	I	T	S	Z	A	P	C	Operation Performed
Jump (Cont.)	JMP	DADDR,CS	FF aa101bbb [DISP][DISP]	24+EA**										(PC) ← (EA), (CS) ← (EA+2) Jump indirect into a new segment. The 16-bit contents of the data memory word addressed by DADDR is loaded into PC. The next sequential 16-bit data memory word's contents is loaded into the CS segment register
	JMP	RW	FF 11100reg	11										(PC) ← (RW) Jump to memory location whose address is contained in register RW.
Subroutine Call and Return	CALL	BRANCH	E8 DISP DISP	19**										((SP)) ← (PC), (SP) ← (SP) −2, (PC) ← (PC) + DISP Call a subroutine in the current program segment using direct addressing
	CALL	BRANCH, SEGM	9A PPQQ PPQQ	28**										((SP)) ← (CS), (SP) ← (SP) - 2, ((SP)) ← (PC), (SP) ← (SP) −2, (PC) ← DATA16, (CS) ← DATA 16 Call a subroutine in another program segment using direct addressing. BRANCH and SEGM are labels that become different 16-bit data words; they are loaded into PC and CS, respectively
	CALL	DADDR	FF aa010bbb [DISP][DISP]	21+EA**										((SP)) ← (PC), (SP) ← (SP) −2, (PC) ← (EA) Call a subroutine in the current program segment using indirect addressing. The address of the subroutine called is stored in the 16-bit data memory word addressed by DADDR
	CALL	DADDR,CS	FF aa011bbb [DISP][DISP]	37+EA**										((SP)) ← (CS), (SP) ← (S2) −2, ((SP)) ← (PC), (SP) ← (SP) −2, (PC) ← (EA), (CS) ← (EA+2) Call a subroutine in a different program segment using indirect addressing. The address of the subroutine called is stored in the 16-bit data memory word addressed by DADDR. The new CS register contents is stored in the next sequential program memory word
	CALL	RW	FF 11010reg	16**										(SP) ← (PC),(SP) ← (SP−2), (PC) ← (RW) Call a subroutine whose address is contained in register RW.
	RET		C3	8**										(PC) ← ((SP)), (SP) ← (SP) + 2 Return from a subroutine in the current segment
	RET	CS	CB	12**										(PC) ← ((SP)), (SP) ← (SP) +2, (CS) ← ((SP)), (SP) ← (SP) +2 Return from a subroutine in another segment
	RET	DATA16	C2 YYYY	17**										(PC) ← ((SP)), (SP) ← (SP) +2 +DATA16 Return from a subroutine in the current segment and add an immediate displacement to SP
	RET	CS,DATA16	CA YYYY	18**										(PC) ← ((SP)), (SP) ← (SP) +2, (CS) ← ((SP)), (SP) ← (SP) +2 +DATA16 Return from a subroutine in another segment and add an immediate displacement to SP

Table 5-4. A Summary of 8086 and 8088 Instructions (Continued)

Type	Mnemonic	Operand(s)	Object Code	Clock Cycles	Statuses									Operation Performed
					O	D	I	T	S	Z	A	P	C	
Immediate Operate	ADD	AL,DATA8	04 YY	4*	X				X	X	X	X	X	[AL] ← [AL] + DATA8 Add 8-bit immediate data to the AL register
	ADD	AX,DATA16	05 YYYY	4*	X				X	X	X	X	X	[AX] ← [AX] + DATA16 Add 16-bit immediate data to the AX register
	ADD	RB,DATA8	80 11000ddd YY	4*	X				X	X	X	X	X	[RB] ← [RB] + DATA8 Add 8-bit immediate data to the RB register
	ADD	RW,DATA16	81 11000ddd YYYY	4*	X				X	X	X	X	X	[RW] ← [RW] + DATA16 Add 16-bit immediate data to the RW register
	ADD	DADDR, DATA8	80 aa000bbb [DISP][DISP] YY	17+EA	X				X	X	X	X	X	[EA] ← [EA] + DATA8 Add 8-bit immediate data to the data memory byte addressed by DADDR
	ADD	DADDR, DATA16	81 aa000bbb [DISP][DISP] YYYY	17+EA	X				X	X	X	X	X	[EA] ← [EA] + DATA16 Add 16-bit immediate data to the data memory word addressed by DADDR
	ADC	AL,DATA8	14 YY	4*	X				X	X	X	X	X	[AL] ← [AL] + DATA8 + [C] Add 8-bit immediate data, plus carry, to the AL register
	ADC	AX,DATA16	15 YYYY	4*	X				X	X	X	X	X	[AX] ← [AX] + DATA16 + [C] Add 16-bit immediate data, plus carry, to the AX register
	ADC	B,DATA8	80 11010ddd YY	4*	X				X	X	X	X	X	[RB] ← [RB] + DATA8 + [C] Add 8-bit immediate data, plus carry, to the RB register
	ADC	RW,DATA16	81 11010ddd YYYY	4*	X				X	X	X	X	X	[RW] ← [RW] + DATA16 + [C] Add 16-bit immediate data, plus carry, to the RW register
	ADC	DADDR, DATA8	80 aa010bbb [DISP][DISP] YY	17+EA	X				X	X	X	X	X	[EA] ← [EA] + DATA8 + [C] Add 8-bit immediate data, plus carry, to the data memory byte addressed by DADDR
	ADC	DADDR, DATA16	81 aa010bbb [DISP][DISP] YYYY	17+EA	X				X		X	X	X	[EA] ← [EA] + DATA16 + [C] Add 16-bit immediate data, plus carry, to the data memory word addressed by DADDR
	AND	AL,DATA8	24 YY	4*	0				X	X	U	X	0	[AL] ← [AL] AND DATA8 AND 8-bit immediate data with AL register contents
	AND	AX,DATA16	25 YYYY	4*	0				X	X	U	X	0	[AX] ← [AX] AND DATA16 AND 16-bit immediate data with AX register contents
	AND	RB,DATA8	80 11100ddd YY	4*	0				X	X	U	X	0	[RB] ← [RB] AND DATA8 AND 8-bit immediate data with RB register contents
	AND	RW,DATA16	81 11100ddd YYYY	4*	0				X	X	U	X	0	[RW] ← [RW] AND DATA16 AND 16-bit immediate data with RW register contents
	AND	DADDR,8	80 aa100bbb [DISP][DISP] YY	17+EA	0				X	X	U	X	0	[EA] ← [EA] AND DATA8 AND 8-bit immediate data with contents of data memory byte addressed by DADDR
	AND	DADDR, DATA16	81 aa100bbb [DISP][DISP] YYYY	17+EA	0				X	X	U	X	0	[EA] ← [EA] AND DATA16 AND 16-bit immediate data with contents of 16-bit data memory word addressed by DADDR

Table 5-4. A Summary of 8086 and 8088 Instructions (Continued)

Type	Mnemonic	Operand(s)	Object Code	Clock Cycles	Statuses									Operation Performed
					O	D	I	T	S	Z	A	P	C	
Immediate Operate (Continued)	CMP	AL,DATA8	3C YY	4*	X				X	X	X	X	X	[AL] − DATA8 Subtract 8-bit immediate data from AL register contents. Discard result, but adjust status flags
	CMP	AX,DATA16	3D YYYY	4*	X				X	X	X	X	X	[AX] − DATA16 Subtract 16-bit immediate data from AX register contents. Discard result, but adjust status flags
	CMP	RB,DATA8	80 11111ddd YY	4*	X				X	X	X	X	X	[RB] − DATA8 Subtract 8-bit immediate data from RB register contents. Discard result, but adjust status flags
	CMP	RW,DATA16	100000a1 1111ddd YY [YY]	4*	X				X	X	X	X	X	[RW] − DATA16 Subtract 16-bit immediate data from RW register contents. Discard result, but adjust status flags
	CMP	DADDR, DATA8	80 aa111bbb [DISP][DISP] YY	10+EA	X				X	X	X	X	X	[EA] − DATA8 Subtract 8-bit immediate data from contents of data memory byte addressed by DADDR. Discard result, but adjust status flags
	CMP	DADDR, DATA16	100000a1 aa111bbb [DISP][DISP] YY[YY]	10+EA	X				X	X	X	X	X	[EA] − DATA16 Subtract 16-bit immediate data from contents of 16-bit data memory word addressed by DADDR. Discard result, but adjust status flags
	OR	AL,DATA8	0C YY	4*	0				X	X	U	X	0	[AL] ← [AL] OR DATA8 OR 8-bit immediate data with AL register contents
	OR	AX,DATA16	0D YYYY	4*	0				X	X	U	X	0	[AX] ← [AX] OR DATA16 OR 16-bit immediate data with AX register contents
	OR	RB,DATA8	80 11001ddd YY	4*	0				X	X	U	X	0	[RB] ← [RB] OR DATA8 OR 8-bit immediate data with RB register contents
	OR	RW,DATA16	81 11001ddd YYYY	4*	0				X	X	U	X	0	[RW] ← [RW] OR DATA 16 OR 16-bit immediate data with RW register contents
	OR	DADDR, DATA8	80 aa001bbb [DISP][DISP] YY	17+EA	0				X	X	U	X	0	[EA] ← [EA] OR DATA 8 OR 8-bit immediate ata with contents of data memory byte addressed by DADDR
	OR	DADDR, DATA16	81 aa001bbb [DISP][DISP] YYYY	17+EA	0				X	X	U	X	0	[EA] ← [EA] OR DATA16 OR 16-bit immediate data with contents of 16-bit data memory word addressed by DADDR
	SBB	AL,DATA8	1C YY	4*	X				X	X	X	X	X	[AL] ← [AL] − DATA8 − [C] Subtract 8-bit immediate signed binary data from AL register contents using twos complement arithmetic. If the Carry status was originally 1 decrement the result
	SBB	AX,DATA16	1D YYYY	4*	X				X	X	X	X	X	[AX] ← [AX] − DATA16 − [C] Subtract 16-bit immediate signed binary data from AX register contents using twos complement arithmetic. If the Carry status was originally 1 decrement the result

Table 5-4. A Summary of 8086 and 8088 Instructions (Continued)

Type	Mnemonic	Operand(s)	Object Code	Clock Cycles	Statuses								Operation Performed
					O	D	I	T	S	Z	A	P	C	
Immediate Operate (Continued)	SBB	RB,DATA8	80 11011ddd YY	4*	X				X	X	X	X	X	[RB] ← [RB] − DATA8 − [C] Subtract 8-bit immediate signed binary data from RB register contents using twos complement arithmetic. If the Carry status was originally 1 decrement the result
	SBB	RW,DATA16	100000a1 11011ddd YY [YY]	4*	X				X	X	X	X	X	[RW] ← [RW] − DATA16 − [C] Subtract 16-bit immediate signed binary data from RW register contents using twos complement arithmetic. If the Carry status was originally 1 decrement the result
	SBB	DADDR, DATA8	80 aa011bbb [DISP][DISP] YY	17+EA	X				X	X	X	X	X	[EA] ← [EA] − DATA8 − [C] Subtract 8-bit immediate signed binary data from contents of data memory byte addressed by DADDR using twos complement arithmetic. If the Carry status was originally 1 decrement the result
	SBB	DADDR, DATA16	100000a1 aa011bbb [DISP][DISP]YY [YY]	17+EA	X				X	X	X	X	X	[EA] ← [EA] − DATA16 − [C] Subtract 16-bit immediate signed binary data from contents of 16-bit data memory word addressed by DADDR using twos complement arithmetic. If the Carry status was originally 1 decrement the result
	SUB	AL,DATA8	2C YY	4*	X				X	X	X	X	X	[AL] ← [AL] − DATA8 Subtract the 8-bit immediate signed binary data from AL register contents using twos complement arithmetic
	SUB	AX,DATA16	2D YYYY	4*	X				X	X	X	X	X	[AX] ← [AX] − DATA16 Subtract the 16-bit immediate signed binary data from AX register contents using twos complement arithmetic
	SUB	RB,DATA8	80 11101ddd YY	4*	X				X	X	X	X	X	[RB] ← [RB] − DATA8 Subtract the 8-bit immediate signed binary data from RB register contents using twos complement arithmetic
	SUB	RW,DATA16	81 11101ddd YYYY	4*	X				X	X	X	X	X	[RW] ← [RW] − DATA16 Subtract the 16-bit immediate signed binary data from RW register contents using twos complement arithmetic
	SUB	DADDR, DATA8	80 aa101bbb [DISP][DISP] YY	17+EA	X				X	X	X	X	X	[EA] ← [EA] − DATA8 Subtract the 8-bit immediate signed binary data from the contents of the data memory byte addressed by DADDR using twos complement arithmetic
	SUB	DADDR, DATA16	100000a1 aa101bbb [DISP][DISP]YY [YY]	17+EA	X				X	X	X	X	X	[EA] ← [EA] − DATA16 Subtract the 16-bit immediate signed binary data from the contents of the 16-bit data memory word addressed by DADDR using twos complement arithmetic
	TEST	AL,DATA8	A8 YY	4*	0				X	X	U	X	0	[AL] AND DATA8 AND the 8-bit immediate data and AL register contents. Discard the result but adjust status s

Table 5-4. A Summary of 8086 and 8088 Instructions (Continued)

Type	Mnemonic	Operand(s)	Object Code	Clock Cycles	Statuses								Operation Performed
					O	D	I	T	S	Z	A	P	C	
Immediate Operate (Continued)	TEST	AX,DATA16	A9 YYYY	4*	0				X	X	U	X	0	[AX] AND DATA16 AND the 16-bit immediate data and AX register contents. Discard the result but adjust status flags
	TEST	RB,DATA8	F6 11000ddd YY	5*	0				X	X	U	X	0	[RB] AND DATA8 AND the 8-bit immediate data and RB register contents. Discard the result but adjust status flags
	TEST	RW,DATA16	F7 11000ddd YYYY	5*	0				X	X	U	X	0	[RW] AND DATA16 AND the 16-bit immediate data and RW register contents. Discard the result but adjust status flags
	TEST	DADDR, DATA8	F6 aa000bbb [DISP][DISP] YY	11+EA	0				X	X	U	X	0	[EA] AND DATA8 AND the 8-bit immediate data and the contents of the data memory location addressed by DADDR. Discard the result but adjust status flags
	TEST	DADDR, DATA16	F7 aa000bbb [DISP][DISP]YYYY	11+EA	0				X	X	U	X	0	[EA] AND DATA16 AND the 16-bit immediate data and the contents of the 16-bit data memory word addressed by DADDR. Discard the result but adjust status flags
	XOR	AL,DATA8	34 YY	4*	0				X	X	U	X	0	[AL] ← [AL] XOR DATA8 Exclusive OR 8-bit immediate data with AL register contents
	XOR	AX,DATA16	35 YYYY	4*	0				X	X	U	X	0	[AX] ← [AX] XOR DATA16 Exclusive OR 16-bit immediate data with AX register contents
	XOR	RB,DATA8	80 11110ddd YY	4*	0				X	X	U	X	0	[RB] ← [RB] XOR DATA8 Exclusive OR 8-bit immediate data with RB register contents
	XOR	RW,DATA16	81 11110ddd YYYY	4*	0				X	X	U	X	0	[RW] ← [RW] XOR DATA16 Exclusive OR 16-bit immediate data with RW register contents
	XOR	DADDR, DATA8	80 aa010bbb [DISP][DISP] YY	17+EA	0				X	X	U	X	0	[EA] ← [EA] XOR DATA8 Exclusive OR 8-bit immediate data with contents of the data memory byte addressed by DADDR
	XOR	DADDR, DATA16	81 aa010bbb [DISP][DISP] YYYY	17+EA	0				X	X	U	X	0	[EA] ← [EA] XOR DATA16 Exclusive OR 16-bit immediate data with contents of the 16-bit data memory word addressed by DADDR
Branch On Condition	LOOP	DISP8	E2 DISP	5 or 17**										[CX] ← [CX] −1 If [CX] ≠ 0 then [PC] ← [PC] + DISP8 Decrement CX register and branch if CX contents are not 0
	LOOPE	DISP8	E1 DISP	6 or 18**										[CX] ← [CX] −1 If [CX] ≠ 0 and [Z] = 1 then [PC] + DISP8 Decrement CX register and branch if CX contents is not 0 and Z status is 1
	LOOPNE	DISP8	E0 DISP	5 or 19**										[CX] ← [CX] −1 If [CX] ≠ 0 and [Z] = 0 then [PC] ← [PC] + DISP8 Decrement CX register and branch if CX contents is not 0 and Z status is 0
	LOOPNZ	DISP8												See LOOPNE
	LOOPZ	DISP8												See LOOPE
	JA	DISP8	77 DISP	4 or 16**										[PC] ← [PC] + DISP8 Branch if C or Z is 0

Table 5-4. A Summary of 8086 and 8088 Instructions (Continued)

Type	Mnemonic	Operand(s)	Object Code	Clock Cycles	Statuses O	D	I	T	S	Z	A	P	C	Operation Performed
Branch On Condition (Continued)	JAE	DISP8	73 DISP	4 or 16**										[PC] ← [PC] + DISP8 Branch if C is 0
	JB	DISP8	72 DISP	4 or 16**										[PC] ← [PC] + DISP8 Branch if C is 1
	JBE	DISP8	76 DISP	4 or 16**										[PC] ← [PC] + DISP8 Branch if C or Z is 1
	JCXZ	DISP8	E3 DISP	6 or 18**										[PC] ← [PC] + DISP8 Branch if the CX register contents is 0
	JE	DISP8	74 DISP	4 or 16**										[PC] ← [PC] + DISP8 Branch if Z is 1
	JG	DISP8	7F DISP	4 or 16**										[PC] ← [PC] + DISP8 Branch if Z is 0 or the S and O statuses are the same
	JGE	DISP8	7D DISP	4 or 16**										[PC] ← [PC] + DISP8 Branch if the S and O statuses are the same
	JL	DISP8	7C DISP	4 or 16**										[PC] ← [PC] + DISP8 Branch if the S and O statuses differ
	JLE	DISP8	7E DISP	4 or 16**										[PC] ← [PC] + DISP8 Branch if Z is 1 or the S and O statuses differ
	JNA	DISP8												See JBE
	JNAE	DISP8												See JB
	JNB	DISP8												See JAE
	JNBE	DISP8												See JA
	JNE	DISP8	75 DISP	4 or 16**										[PC] ← [PC] + DISP8 Branch if Z is 0
	JNG	DISP8												See JLE
	JNGE	DISP8												See JL
	JNL	DISP8												See JGE
	jnle	disp8												See JG
	JNO	DISP8	71 DISP	4 or 16**										[PC] ← [PC] + DISP8 Branch if O is 0
	JNP	DISP8	7B DISP	4 or 16**										[PC] ← [PC] + DISP8 Branch if P is 0
	JNS	DISP8	79 DISP	4 or 16**										[PC] ← [PC] + DISP8 Branch if S is 0
	JNZ	DISP8												See JNE
	JO	DISP8	70 DISP	4 or 16**										[PC] ← [PC] + DISP8 Branch if O is 1
	JP	DISP8	7A DISP	4 or 16**										[PC] ← [PC] + DISP8 Branch if P is 1
	JPE	DISP8												See JP

Table 5-4. A Summary of 8086 and 8088 Instructions (Continued)

Type	Mnemonic	Operand(s)	Object Code	Clock Cycles	Statuses O	D	I	T	S	Z	A	P	C	Operation Performed
BOC (Cont.)	JPO	DISP8												See JNP
	JS	DISP8	78DISP	4 or 16**										[PC] ← [PC] + DISP8 Branch if S is 1
	JZ	DISP8												See JE
Register ↔ Register Move	MOV	RBD,RBS	8A11dddsss	2*										[RBD] ← [RBS] Move the contents of any RB register to any RB register
	MOV	RWD,RWS	8B 11dddsss	2*										[RWD] ← [RWS] Move the contents of any RW register to any RW register
	MOV	SR,RW	8E 110rrsss	2*										[SR] ← [RWS] Move the contents of any RW register to any Segment register
	MOV	RW,SR	8C 110rrddd	2*										[RWD] ← [SR] Move the contents of any Segment register to any RW register
	XCHG	AX,RW	10010reg	3*										[AX] ↔ [RW] Exchange the contents of AX and any RW register
	XCHG	RB,RB	86 11regreg	4*										[RB] ↔ [RB] Exchange the contents of any two RB registers
	XCHG	RW,RW	87 11regreg	4*										[RW] ↔ [RW] Exchange the contents of any two RW registers
Block Transfer and Search	CMPS	BD,BS	A6	22	X	I/D			X	X	X	X	X	[[SI]] − [[DI]], [SI] ← [SI] ± 1, [DI] ← [DI] ± 1 Compare the data bytes addressed by the SI and DI Index registers using string data addressing*
	CMPS	WD,WS	A7	22	X	I/D			X	X	X	X	X	[[SI]] − [[DI]], [SI] ← [SI] ± 2, [DI] ← [DI] ± 2 Compare the 16-bit data words addressed by the SI and DI Index registers using string data addressing*
	LODS	BD,BS	AC	12		I/D								[AL] ← [[SI]], [SI] ← [SI] ± 1 Move a data byte from the location addressed by the SI Index register to the AL register using string data addressing
	LODS	WD,WS	AD	12		I/D								[AX] ← [[SI]], [SI] ← [SI] ± 1 Move a data word from the 16-bit location addressed by the SI Index register to the AX register using string data addressing
	MOVS	BD,BS	A4	18		I/D								[[DI]] ← [[SI]], [SI] ← [SI] ± 1, [DI] ← [DI] ± 1 Move a data byte from the location addressed by the SI Index register to the extra segment location addressed by the DI register using string data addressing*
	MOVS	WD,WS	A5	18		I/D								[[DI]] ← [[SI]], [SI] ← [SI] ± 2, [DI] ← [DI] ± 2 Move a 16-bit data word from the location addressed by the SI Index register to the extra segment location addressed by the DI Index register using string data addressing*

* For these instructions, the default destination segment register cannot be overriden.

Table 5-4. A Summary of 8086 and 8088 Instructions (Continued)

Type	Mnemonic	Operand(s)	Object Code	Clock Cycles	Statuses									Operation Performed
					O	D	I	T	S	Z	A	P	C	
Block Transfer and Search (Continued)	REP	N	1111001z	+2 per loop		I/D								Repeat the next sequential instruction (which must be a Block Transfer and Search instruction) until CX contents decrements to 0. Decrement CX contents on each repeat. If the next instruction is CMPB, CMPW, SCAB, or SCAW then repeat until CX contents decrements to 0 or Z status does not equal N
	SCAS	BD,BS	AE	15	X	I/D			X	X	X	X	X	[AL] − [[DI]], [DI] ← [DI] ± 1 Compare AL register contents with the extra segment data byte addressed by the DI Index register using string data addressing
	SCAS	WD,WS	AF	15	X	I/D			X	X	X	X	X	[AX] − [[DI]], [DI] ← [DI] ± 2 Compare AX register contents with the extra segment 16-bit data ord addressed by the DI Index register using string data addressing
	STOS	BD,BS	AA	11	X	I/D			X	X	X	X	X	[[DI]] ← [AL], [DI] ← [DI] ± 1 Store the AL register contents in the extra segment data memory byte addressed by the DI Index register using string data addressing
	STOS	WD,WS	AB	11	X	I/D			X	X	X	X	X	[[DI]] ← [AX], [DI] ← [DI] ± 2 Store the AX register contents in the extra segment 16-bit data memory word addressed by the DI Index register using string data addressing
Register - Register Operate	ADC	RBD,RBS	12 11dddsss	3*	X				X	X	X	X	X	[RBD] ← [RBD] + [RBS] + [C] Add the 8-bit contents of register RBS, plus the Carry status, to register RBD
	ADC	RWD,RWS	13 11dddsss	3*	X				X	X	X	X	X	[RWD] ← [RWD] + [RWS] + [C] Add the 16-bit contents of register RWS, plus the Carry status, to register RWD
	ADD	RBD,RBS	02 11dddsss	3*	X				X	X	X	X	X	[RBD] ← [RBD] + [RBS] Add the 8-bit contents of register RBS to register RBD
	ADD	RWD,RWS	03 11dddsss	3*	X				X	X	X	X	X	[RWD] ← [RWD] + [RWS] Add the 16-bit contents of register RWS to register RWD
	AND	RBD,RBS	22 11dddsss	3*	0				X	X	U	X	0	[RBD] ← [RBD] AND [RBS] AND the 8-bit contents of register RBS with register RBD
	AND	RWD,RWS	23 11dddsss	3*	0				X	X	U	X	0	[RWD] ← [RWD] AND [RWS] AND the 16-bit contents of register RWS with register RWD
	CBW		98	2*										[AH] ← [AL7] Extend AL sign bit into AH
	CMP	RBD,RBS	3A 11dddsss	3*	X				X	X	X	X	X	[RBD] − [RBS] Subtract the contents of register RBD from register RBS. Discard the result, but adjust status flags
	CMP	RWD,RWS	3B 11dddsss	3*	X				X	X	X	X	X	[RWD] − [RWS] Subtract the contents of register RWD from register RWS. Discard the result, but adjust status flags
	CWD		99	5										[DX] ← [AX15] Extend AX sign bit into DX

Table 5-4. A Summary of 8086 and 8088 Instructions (Continued)

Type	Mnemonic	Operand(s)	Object Code	Clock Cycles	Statuses									Operation Performed
					O	D	I	T	S	Z	A	P	C	
Register – Register Operate (Continued)	DIV	RBS	F6 11110sss	80-90	U				U	U	U	U	U	[AX] ← [AX]/[RBS] Divide the 16-bit contents of AX by the 8-bit contents of RBS. Store the integer quotient in AL and the remainder in AH. If the quotient is greater than FF_{16}, execute a "divide by 0" interrupt
	DIV	RWS	F7 11110sss	144-162	U				U	U	U	U	U	[DX] [AX] ← [DX] [AX]/[RWS] Divide the 32-bit contents of registers DX (high-order) and AX (low-order) by the 16-bit contents of RWS. Store the integer quotient in AX and the remainder in DX. If the quotient is greater than $FFFF_{16}$, execute a "divide by 0" interrupt
	IDIV	RBS	F6 11111sss	101-112	U				U	U	U	U	U	[AX] ← [AX]/[RBS] Divide the 16-bit contents of register AX by the 8-bit contents of RBS, treating both contents as signed binary numbers. Store the quotient, as a signed binary number, in AL. Store the remainder, as an unsigned binary number, in AX. Store the remainder, as an unsigned binary number, in AH. If the quotient is greater than $7F_{16}$, or less than -80_{16}, execute a "divide by 0" interrupt
	IDIV	RWS	F7 11111sss	165-184	U				U	U	U	U	U	[DX] [AX] ← [DX] [AX]/[RWS] Divide the 32-bit contents of register DX (high-order) and AX (low-order) by the 16-bit contents of RWS. Treat both contents as signed binary numbers. Store the quotient, as a signed binary number, in AX. Store the remainder, as an unsigned binary number, in AH. If the quotient is greater than $7FFF_{16}$, or less than -8000_{16}, execute a "divide by 0" interrupt
	IMUL	RBS	F6 11101sss	80-98	X				U	U	U	U	X	[AX] ← [AL] * [RBS] Multiply the 8-bit contents of register AL by the contents of RBS. Treat both numbers as signed binary numbers. Store the 16-bit product in AX
	IMUL	RWS	F7 11101sss	128-154	X				U	U	U	U	X	[DX] [AX] ← [AX] * [RWS] Multiply the 16-bit contents of register AX by the 16-bit contents of RWS. Treat both numbers as signed binary numbers. Store the 32-bit product in DX (high-order word) and AX (low-order word)
	MUL	RBS	F6 11100sss	70-77	X				U	U	U	U	X	[AX] ← [AL] * [RBS] Multiply the 8-bit contents of register AL by the contents of RBS. Treat both numbers as unsigned binary numbers. Store the 16-bit product in AX
	MUL	RWS	F7 11100sss	118-133	X				U	U	U	U	X	[DX] [AX] ← [AX] * [RWS] Multiply the 16-bit contents of register AX by the 16-bit contents of RWS. Treat both numbers as unsigned binary numbers. Store the 32-bit product in DX (high-order word) and AX (low-order word)
	OR	RBD,RBS	0A 11dddsss	3*	0				X	X	U	X	0	[RBD] ← [RBD] OR [RBS] OR the 8-bit contents of register RBS with register RBD
	OR	RWD,RWS	0B 11dddsss	3*	0				X	X	U	X	0	[RWD] ← [RWD] OR [RWS] OR the 16-bit contents of register RWS with register RWD

Table 5-4. A Summary of 8086 and 8088 Instructions (Continued)

Type	Mnemonic	Operand(s)	Object Code	Clock Cycles	Statuses								Operation Performed
					O	D	I	T	S	Z	A	P	C	
Register – Register Operate (Continued)	SBB	RBD,RBS	1A 11dddsss	3*	X				X	X	X	X	X	[RBD] ← [RBD] – [RBS] – [C] Subtract the 8-bit contents of register RBS from RBD using twos complement arithmetic. If the Carry status was originally 1 decrement the result
	SBB	RWD,RWS	1B 11dddsss	3*	X				X	X	X	X	X	[RWD] ← [RWD] – [RWS] – [C] Subtract the 16-bit contents of register RWS from RWD using twos complement arithmetic. If the Carry status was originally 1 decrement the result
	SUB	RBD,RBS	2A 11dddsss	3*	X				X	X	X	X	X	[RBD] ← [RBD] – [RBS] Subtract the 8-bit contents of register RBS from RBD using twos complement arithmetic
	SUB	RWD,RWS	2B 11dddsss	3*	X				X	X	X	X	X	[RWD] ← [RWD] – [RWS] Subtract the 16-bit contents of register RWS from RWD using twos complement arithmetic
	TEST	RBD,RBS	84 11regreg	3*	0				X	X	U	X	0	[RBD] AND [RBS] AND the 8-bit contents of register d and register RBS. Discard the result, but adjust status flags
	TEST	RWD,RWS	85 11regreg	3*	0				X	X	U	X	0	[RWD] AND [RWS] AND the 16-bit contents of register RWD and register RWS. Discard the result, but adjust status flags
	XOR	RBD,RBS	30 11dddsss	3*	0				X	X	U	X	0	[RBD] ← [RBD] XOR [RBS] Exclusive OR the 8-bit contents of register RBS with register RBD
	XOR	RWD,RWS	31 11dddsss	3*	0				X	X	U	X	0	[RWD] ← [RWD] XOR [RWS] Exclusive OR the 16-bit contents of register RWS with register RWD
Register Operate	AAA		37	4*	U				U	U	X	U	X	ASCII adjust AI register contents for addition (as described in accompanying text)
	AAD		D5 0A	60	U				X	X	U	X	U	Decimal adjust dividend in AL prior to dividing an unpacked decimal divisor, to generate an unpacked decimal quotient. (See accompanying text for details)
	AAM		D4 0A	83	U				X	X	U	X	U	After multiplying o unpecked decimal operands, adjust product in AX to become an unpacked decimal result. (See accompanying text for details)
	AAS		3F	4*	U				U	U	X	U	X	After subtracting two unpacked decimal numbers, adjust the difference in AL so that it too is an unpacked decimal number. (See accompanying text for details)
	DAA		27	4*	U				X	X	X	X	X	After adding two packed decimal numbers, adjust the sum in AL so that it too is a packed decimal number. (See accompanying text for details)
	DAS		2F	4*	U				X	X	X	X	X	After subtracting two packed decimal numbers, adjust the difference in AL so that it too is a packed decimal number. (See accompanying text for details)
	DEC	RB	FE 11001ddd	3*	X				X	X	X	X		[RB] ← [RB] – 1 Decrement the 8-bit contents of register RB
	DEC	RW	01001ddd	2*	X				X	X	X	X		[RW] ← [RW] - 1 Decrement the 16-bit contents of register RW

Table 5-4. A Summary of 8086 and 8088 Instructions (Continued)

Type	Mnemonic	Operand(s)	Object Code	Clock Cycles	Statuses								Operation Performed
					O	D	I	T	S	Z	A	P	C	
Register Operate (Continued)	INC	RB	FE 11000ddd	3*	X				X	X	X	X		[RB] ← [RB] +1 Increment the 8-bits contents of register RB
	INC	RW	01000ddd	2*	X				X	X	X	X		[RW] ← [RW] +1 Increment the 16-bit contents of register RW
	NEG	RB	F6 11011ddd	3*	X				X	X	X	X	X	[RB] ← $\overline{[RB]}$ +1 Twos complement the 8-bit contents of register RB
	NEG	RW	F7 11011ddd	3*	X				X	X	X	X	X	[RW] ← $\overline{[RW]}$ + 1 Twos complement the 16-bit contents of register RW
	NOT	RB	F6 11010ddd	3*										[RB] ← $\overline{[RB]}$ Ones complement the 8-bit contents of register RB
	NOT	RW	F7 11010ddd	3*										[RW] ← $\overline{[RW]}$ Ones complement the 16-bit contents of register RW
	RCL	RB,N	110100v0 11010ddd		X								X	Rotate left through Carry the 8-bit contents of RB register, or the 16-bit
	RCL	RW,N	110100v1 11010ddd		X								X	contents of RW register, as illustrated for memory operate
	RCR	RBN	110100v0 11011ddd		X								X	Rotate right through Carry the 8-bit contents of RB register, or the 16-bit
	RCR	RW,N	110100v1 11011ddd		X								X	contents of RW register, as illustrated for memory operate
	ROL	RB,N	110100v0 11000ddd	N=1 2*	X								X	Rotate left the 8-bit contents of RB register, or the 16-bit contents of RW
	ROL	RW,N	110100v1 11000ddd	N>1 4N+8	X								X	register as illustrated for memory operate
	ROR	RB,N	110100v0 11001ddd		X								X	Rotate right the 8-bit contents of RB register, or the 16-bit contents of RW
	ROR	RW,N	110100v1 11001ddd		X								X	register, as illustrated for memory operate
	SAL	RB,N	110100v0 11100ddd		X				X	X	U	X	X	Shift left the 8-bit contents of RB register, or the 16-bit contents of RW
	SAL	RW,N	110100v1 11100ddd		X				X	X	U	X	X	register, as illustrated for memory operate
	SAR	RB,N	110100v0 11111ddd		X				X	X	U	X	X	Shift right the 8-bit contents of register RB, or the 16-bit contents of register
	SAR	RW,N	110100v1 11111ddd		X				X	X	U	X	X	RW, as illustrated for memory operate
	SHL	RB,N			X				X	X	U	X	X	See SAL
	SHL	RW,N			X				X	X	U	X	X	See SAL
	SHR	RB,N	110100v0 11101ddd	N=1 2*	X				X	X	U	X	X	Shift right the 8-bit contents of register RB, or the 16-bit contents of register
	SHR	RW,N	110100v1 11101ddd	N>1 4N+8					X	X	U	X	X	RW, as illustrated for memory operate
Stack	POP	DADDR	8F aa000bbb [DISP][DISP]	17+EA										[EA] ← [[SP]], [SP] ← [SP] + 2 Load the 16-bit Stack word, addressed using Stack addressing, into the 16-bit data memory word addressed by DADDR. Increment SP by 2
	POP	RW	01011ddd	8										[RW or SR] ← [[SP]], [SP] ← [SP] + 2
	POP	SR	000rr111	8										Load the 16-bit Stack word, addressed using Stack addressing, into the specified 16-bit register. Increment SP by 2.
	POPF		9D	8	X	X	X	X	X	X	X	X	X	[SFR] ← [[SP]], [SP] ← [SP] + 2 Load the 16-bit Stack word, addressed using Stack addressing, into the Status Flags register
	PUSH	DADDR	FF aa110bbb [DISP][DISP]	16+EA										[SP] ← [SP] –2, [[SP]] ← [EA] Store the 16-bit contents of the data memory word addressed by DADDR in the 16-bit Stack word addressed using Stack addressing. Decrement SP by 2

Table 5-4. A Summary of 8086 and 8088 Instructions (Continued)

Type	Mnemonic	Operand(s)	Object Code	Clock Cycles	Statuses O	D	I	T	S	Z	A	P	C	Operation Performed
Stack (Cont.)	PUSH	RW	01010rrr	11										[SP] ← [SP] −2, [[SP]] ← [RW or SR]
	PUSH	SR	000rr110	10										Store the contents of the specified 16-bit register in the 16-bit Stack word addressed using Stack addressing. Decrement SP by 2
	PUSHF		9C	10										[SP] ← [SP] +2, [[SP]] ← [SFR]
														Store the Status flags register contents in the 16-bit Stack word addressed using Stack addressing. Decrement SP by 2
Interrupts	INT	3	CC	52			0	0						Execute a software interrupt and vector through table entry 3
	INT	V	CD YY	51			0	0						Execute a software interrupt and vector through table entry V
	INTO		CE	4 or 53			0	0						If the O status is 1, execute a software interrupt and vector through table entry 10_{16}
	IRET		CF	24										Return from interrupt service routine
Status	CLC		F8	2*									0	[C] ← 0
														Clear Carry status
	CLD		FC	2*		0								[D] ← 0
														Clear Decrement/Increment select
	CLI		FA	2*			0							[I] ← 0
														Clear Interrupt enable status, disabling all interrupts
	CMC		F5	2*									X	[C] ← $\overline{[C]}$
														Complement Carry status
	LAHF		9F	4*										Transfer flags to AH register as follows:
														7 6 5 4 3 2 1 0 Bit no. / AH register / S Z 0 A 0 P 1 C
	SAHF		9E	4*					X	X	X	X	X	Transfer AH register contents to status flags as follows:
														7 6 5 4 3 2 1 0 Bit no. / AH register / S Z A P C
	STC		F9	2*									1	[C] ← 1
														Set Carry status to 1
	STD		FD	2*		1								[D] ← 1
														Set Decrement/Increment status to 1
	STI		FB	2*			1							[I] ← 1
														Set interrupt enable status to 1, enabling all interrupts

Table 5-4. A Summary of 8086 and 8088 Instructions (Continued)

Type	Mnemonic	Operand(s)	Object Code	Clock Cycles	Statuses O	D	I	T	S	Z	A	P	C	Operation Performed
Other	ESC	DADDR	11011xxx aaxxxbbb [DISP][DISP]	8+EA										? ← [EA]
														The contents of the data memory location addressed by DADDR is read out of memory and placed on the data bus; however, it is not input to the CPU
	HLT		F4	2*										CPU Halt
	LOCK		F0	2*										Guarantee the CPU bus control during execution of the next sequential instruction
	SEG	SR	001rag110	+ 2										The next sequential allowed memory reference instruction accesses the segment identified by Segment register SR. See Table 20-1 for allowed memory reference instructions
	WAIT		9B	3+5n										CPU enters the WAIT state until TEST pin receives a high input signal
	NOP		90	3*										No operation (This is the same object code as XCHG, AX, AX.)