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

		_		Po	ssible Displaceme	its	Assembly
Memory Reference	Segment Register	Base Register	Index Register	16-Bit Unsigned	8-Bit High-order Bit Extended	None	Language Operand Mnemonic
			Si///	*			
	DS	None	Bt	/// /		// /× ///	
	(Alternate)		SI		X//// * ////X/		
Normal Data	CS, SS or ES)	BX	DÛ		X		
Memory			None	/// * ////			
Reference	DS	None	None				
	\$\$		\$\\\\\$\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\		X 1 X 1 X 1 X X		
	(Alternate*	80	DI.	(/// X ///			
	CS, DS of ESI		None				
Stack	SS	SP	None				
String	DS	None	SI				
Data	ES	None	DI				
Instruction Fetch	CS	PC	None				
Branch	cs	PC	None		×		
I/O Data	DS	DX	None				
		TI	hese columns co	ontribute to OEA	۸.		This colum
	1	TI	hese columns co	ontribute to EA.			to be provide

27777777
V///////
V1811850
11/11/11

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 (00₁₆ or FF₁₆)

AX Accumulator, both bytes

AX15 The value of register AH high-order bit (0 or 1) extended to a 16-bit word (0000₁₆ or FFFF₁₆)

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
DATA16
DH
DI
DI
DI
DATA8
Eight bits of immediate data
16 bits of immediate data
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

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 255₁₀ 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 ٧ Any number in the range 0 through 25510 Х 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:

101 = BP

SL

DI

110 =

111 =

CH

DH

RH

```
[]
             indicate an optional object code byte
             one bit choosing length:
а
               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.)
             two binary digits identifying a segment register:
гг
               00 = ES
               01 = CS
               10 = SS
               11 = DS
reg
             three binary digits identifying a register:
                      16-bit 8-bit
               000 = AX
                              ΑL
               001 =
                       CX
                              CŁ
               010 =
                       DX
                              DL
               011 =
                       BX
                              BL
               100 =
                       SP
                              AH
```

Effective Address calculation and extra clock cycles:

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

- 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

N	ed.	Magmonia	Operando	Ohioot Code	مواميل بإممال			S	Statuses	8 8 8			
IN	γT			anno padro	CIDCA CYCIES	_	님	느	\vdash	$\overline{}$		-	Operation remorned
N		Z	AL,PORT	E4 YY	10		\vdash	<u> </u>				-	[AL] ← [PORT]
N		Z	AL,[DX]	EC 1	80								Load one byte of data from I/O port PORT into AL [AL] — [PDX]
IN AX,FORT E5 YY 10 IA OUT AL,PORT E6 YY 10 IP OUT AL,FORT EE 1 8 IP OUT AX,PORT E7 YY 10 IP OUT AX,PORT E7 YY 10 IP LDS RW,DADDR C6 assssbbb 16+EA IR LLS RW,DADDR C4 assssbbb 16+EA IR LES RW,DADDR C4 assssbbb 16+EA IR MOV RB,DADDR 8A asdddbbb 8+EA IR													Load into AL one byte of data from I/O port whose address is held in the DX
N				Š	,								register
N		<u>z</u>	AX,PORT	E5 YY	0								[AL] ← [PORT], [AH] ← [PORT+1]
IN AX.[DX] ED 8 [A OUT AL.PORT E6 YY 10 [P OUT AL.[DX] EE 1 8 [P OUT AX.PORT E7 YY 10 [P OUT AX.PORT E7 YY 10 [P OUT AX.PORT E7 YY 10 [P LDS RW,DADDR C5 sesssbbb 16+EA [P LEA RW,DADDR SD sesssbbb 2+EA [R LES RW,DADDR C4 sesssbbb 16+EA [R MOV RB,DADDR SA sedddbbb 8+EA [R													Load 16 bits of data into AX, AL receives data from I/O port PORT, AH
OUT AL,PORT E6 YY 10 [P OUT AL,IDX] EE 1 8 [P OUT AX,PORT E7 YY 10 [P OUT AX,PORT E7 YY 10 [P LDS RW,DADDR C5 easssbbb 16+EA [P LEA RW,DADDR 8D aesssbbb 2+EA [R LES RW,DADDR C4 aesssbbb 16+EA [R MOV RB,DADDR 8A aedddbbb 8+EA [R		Z	AX.[DX]	G	80								[At] ← [PDX] [AH] ← [PDX+1]
OUT AL,PORT E6 YY 10 IP OUT AL,IDXJ EE 1 8 IP OUT AX,PORT E7 YY 10 IP OUT AX,IDXJ EF 8 IP LDS RW,DADDR C5 assssbbb 16+EA IR LEA RW,DADDR C4 assssbbb 16+EA IR LES RW,DADDR C4 assssbbb 16+EA IR MOY RB,DADDR 8A asdddbbb 8+EA IR IDISPI[DISP] 16 SPI[DISP] IR IR													Load 16 bits of data into AX. AL receives data from I/O nort whose address is
OUT AL,PORT E6 YY 10 [P] OUT AL,[DX] EE 1 8 [P] OUT AX,PORT E7 YY 10 [P] OUT AX,[DX] EF 8 [P] LDS RW,DADDR C5 assssbbb 16+EA [R] LES RW,DADDR C4 assssbbb 16+EA [R] LES RW,DADDR C4 assssbbb 16+EA [R] MOY RB,DADDR 8A asdddbbb 8+EA [R]											_		held in the DX register. AH receives data from the I/O port whose address is
OUT AL,PORT E6 YY 10 IP OUT AL,IDXI EE 1 8 IP OUT AX,PORT E7 YY 10 IP OUT AX,IDXI EF 8 IP LDS RW,DADDR C5 assssbbb 16+EA IR LEA RW,DADDR C4 assssbbb 16+EA IR LES RW,DADDR C4 assssbbb 16+EA IR MOV RB,DADDR BA asdddbbb 8+EA IR	C												one higher
OUT AL,IDX EF 1 8 IP OUT AX,PORT E7 YY 10 IP OUT AX,IDX EF 8 IP LDS RW,DADDR C5 assssbbb 16+EA IP LEA RW,DADDR SD assssbbb 2+EA IR LES RW,DADDR C4 assssbbb 16+EA IR MOV RB,DADDR BA asdddbbb 8+EA IR)/(DO.	AL,PORT	E6 YY	5								[PORT] - [AL]
OUT AX,PORT EF 1 8 IP OUT AX,PORT E7 YY 10 IP OUT AX,IDX] EF 8 R IP LDS RW,DADDR C5 assssbbb 16+EA IR LEA RW,DADDR 8D assssbbb 2+EA IR LES RW,DADDR C4 assssbbb 16+EA IR MOV RB,DADDR 8A asadddbbb 8+EA IR													Output one byte of data from register AL to I/O port PORT
OUT AX.PORT E7 YY 10 IP OUT AX.IDX] EF 8 IP OUT AX.IDX] EF 8 IP LDS RW.DADDR C5 aasssbbb 16+EA IR LEA RW.DADDR 8D aasssbbb 2+EA IR LES RW.DADDR C4 aasssbbb 16+EA IR MOV RB.DADDR 8A aadddbbb 8+EA IR		00 <u>1</u>	AL,[DX]	田 1	80								[PDX] ← [AL]
OUT AX,PORT EF 8 IP OUT AX,IDXI EF 8 IP LDS RW,DADDR C5 aasssbbb 16+EA IR LEA RW,DADDR 8D aasssbbb 2+EA IR LES RW,DADDR C4 aasssbbb 16+EA IR MOV RB,DADDR 8A aadddbbb 8+EA IR												_	Output one byte of data from register AL to the I/O port whose address is held
OUT AX,PORT EF 8 IP OUT AX,IDX) EF 8 IP LDS RW,DADDR C5 assssbbb 16+EA IR LEA RW,DADDR 8D assssbbb 2+EA IR LES RW,DADDR C4 assssbbb 16+EA IR MOV RB,DADDR 8A asadddbbb 8+EA IR												_	in the DX register
LDS RW,DADDR C5 aasssbbb 16+EA LEA RW,DADDR (DiSPI[DISP]) LES RW,DADDR C4 aasssbbb 16+EA (DiSPI[DISP] (DISPI[DISP]) MOV RB,DADDR (BA aadddbbb 8+EA (DISPI[DISP])		된	AX,PORT	E7 YY	10								[PORT] → [AL], [PORT+1] → [AH]
OUT AX,IDX] EF B [P LDS RW,DADDR C5 easssbbb 16+EA [R LEA RW,DADDR 8D easssbbb 2+EA [R LES RW,DADDR C4 easssbbb 16+EA [R MOV RB,DADDR 8A eadddbbb 8+EA [R													Output 16 bits of data. The AL register contents are output to I/O port PORT.
LDS RW,DADDR C5 aasssbbb 16+EA [R LDS RW,DADDR C5 aasssbbb 2+EA [R LEA RW,DADDR 8D aasssbbb 2+EA [R LES RW,DADDR C4 aasssbbb 16+EA [R MOV RB,DADDR 8A aadddbbb 8+EA [R												_	The AH register contents are output to I/O port PORT+1
LES RW,DADDR C5 aassabbb 16+EA [RR [DISP][DISP]] LEA RW,DADDR 8D aassabbb 2+EA [RR [DISP][DISP]] LES RW,DADDR C4 aassabbb 16+EA [RR [DISP][DISP]] MOV R8,DADDR 8A aadddbbb 8+EA [RR [DISP][DISP]]		0 01	AX,[DX]	₩	6 0								[PORT] — [PDX], [PORT+1] — [PDX+1]
LEA RW,DADDR C5 aasssbbb 16+EA LEA RW,DADDR 8D aasssbbb 2+EA [DISP][DISP] LES RW,DADDR C4 aasssbbb 16+EA [DISP][DISP] MOV RB,DADDR 8A aadddbbb 8+EA [DISP][DISP]											-		Output 16 bits of data. The AL register contents are output to the I/O port
LDS RW,DADDR C5 aasssbbb 16+EA [R LEA RW,DADDR 8D aasssbbb 2+EA [R LES RW,DADDR C4 aasssbbb 16+EA [R MOV RB,DADDR 8A aadddbbb 8+EA [R											_		whose address is held in the DX register. The AH register contents is output
LES RW,DADDR C5 aassbbb 16+EA LES RW,DADDR C4 aassbbb 16+EA LES RW,DADDR C4 aassbbb 16+EA MOV R8,DADDR 8A aadddbbb 8+EA [IR							_	_				\dashv	to the I/O port whose address is one higher
LEA RW,DADDR 8D aassabbb 2+EA LES RW,DADDR C4 aassabbb 16+EA (DISP][DISP] (DISP][DISP] MOV R8,DADDR 8A aadddbbb 8+EA [IR		SOT	RW,DADDR	C5 aasssbbb	16+EA		\vdash	<u> </u>			 	\vdash	[RW] ← [EA], [DS] ← [EA+2]
LES RW,DADDR 8D aassabbb 2+EA [IR [DISPI[DISP]] 16+EA [DISPI[DISP]] 16+EA [DISPI[DISP]] 8+EA [DISPI[DISP]] [DISPI[DISP]] [DISPI[DISP]] [DISPI[DISP]]				[DISP][DISP]		_					_		Load 16 bits of data from the memory word addressed by DADDR into
LES RW,DADDR GD assssbbb 2+EA [IR IDISPI[DISP] 16+EA [IR IDISPI[DISP] 16+EA [IR IDISPI[DISP] 16+EA [IDISPI[DISP] 16] 8+EA [IDISPI[DISP] 16]	0					_	_						register RW. Load 16 bits of data from the next sequential memory word into
LEA RW,DADDR 8D aasssbbb 2+EA [R LES RW,DADDR C4 aasssbbb 16+EA [R MOV RB,DADDR 8A aadddbbb 8+EA [R	oue												the DS register
LES RW,DADDR	919	LEA	RW,DADDR	8D assssbbb	2+EA								[RW] ← OEA
LES RW,DADDR C4 aasssbbb 16+EA [R (DISP][DISP] MOV RB,DADDR 8A aadddbbb 8+EA [R [R (DISP][DISP]	leA			(DISP)[DISP)									Load into RW the 16-bit address displacement which, when added to the
LES RW,DADDR C4 asssbbb 16+EA [R (DISP][DISP]	٨٨												segment register contents, creates the effective data memory address
MOV RB,DADDR 8A aadddbbb 8+EA [R	ou	LES	RW,DADDR	C4 asssbbb	16+EA								$[RW] \leftarrow [EA], [ES] \leftarrow [EA+2]$
MOV RB,DADDR 8A aadddbbb 8+EA [R	ie N			(DISP)[DISP]									Load 16 bits of data from the memory word addressed by DADDR into
MOV RB,DADDR 8A aedddbbb 8+EA [PISP][DISP]	v A												register RW. Load 16 bits of data from the next sequential memory word into
MOV RB,DADDR 8A asdddbbb 8+EA [PISP][DISP]	Jer												the ES register
dSiQ	niı	δ	RB,DADDR	8A aadddbbb	8+EA				_	_			[R8] ← EA]
register RB	j			[DISP][DISP]									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)

Operation Performed om register RB in the memory word from register RW in the memory word from register RW in the memory word directly addrants of register AL into the defents of register AX into the agister SR the contents of the latta between register RB and atta between register RW are data between register RW are as a second of the latta between register RW are second or second o	무 설
RWI ← [EA] Load 16 bits of data from the data memory word addressed by DADDR to register RW [EA] ← [RB] Store the data byte from register RB in the memory byte addressed by DADDR [EA] ← [RW] Store the 16-bit data word from register RW in the memory word addressed by DADDR [EA] ← [RAI] Store the 16-bit data memory byte directly addressed by LABEL into register AL [AX] ← [EA] Load the 16-bit data memory word directly addressed by LABEL into register AL [AX] ← [EA] Store the 8-bit contents of register AL into the data memory byte directly addressed by LABEL Store the 16-bit contents of register AX into the data memory word directly addressed by LABEL [EA] ← [AX] Store the 16-bit contents of register AX into the data memory word directly addressed by DADDR [EA] ← [SR] Store the contents of Segment register SR in the 16-bit memory location addressed by DADDR Rxhange a byte of data between register RB and the data memory location addressed by DADDR Rxhange 16 bits of data between register RW and the data memory location addressed by DADDR Exchange 16 bits of data between register RW and the data memory location addressed by DADDR Exchange 16 bits of data between register RW and the data memory location addressed by DADDR	I (ALL +- (IAL) + (BX)) Load into AL the data byte stored in the memory location addressed by summing initial AL contents with BX contents
a a	
N N	
Statuses 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	
Clock Cycles 8+EA 9+EA 10 10 10 17+EA 17+EA	-
Object Code 8B sadddbbb [DISP][DISP] 88 sesssbbb [DISP][DISP] 89 aasssbbb [DISP][DISP] A0 PPQQ A1 PPQQ A2 PPQQ A2 PPQQ A3 PPQQ BE sa0rrbbb [DISP][DISP] 8C sa0rrbbb [DISP][DISP] 86 aaregbbb [DISP][DISP] 87 saregbbb [DISP][DISP] 87 saregbbb [DISP][DISP]	à
Operand(s) RW,DADDR DADDR,RB CADDR,RB AX,LABEL AX,L	
Mov	X.A.
Primary Memory Reference (Continued)	

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

						١		١	1	1		ŀ	
ed	Magazia		Object Code	orland design			Š	Statuses	Ses				
γī	Мпешопс	Operand(s)	apon pagino	CIOCK CYCIES	0	-	느	S	Z	٨	_	0	Operation Performed
	ADC	R8,DADDR	12 aadddbbb [DISP][DISP]	9+EA	×			×	×	×	×	×	[RB] ← [EA] + [RB] + [C] Add the contents of the data byte addressed by DADDR, plus the Carry status,
	ADC	RW,DADDR	13 aadddbbb [DISP][DISP]	9+EA	×			×	×	×	×		for registering [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	×			×	×	×	×	×	[EA] — [EA] + [RB] + [C] Add the 8-bit contents of register RB, plus the Carry status, to the data memory byte addressed by DADDR
(81	ADC	DADDR,RW	11 aasssbbb (DISP][DISP]	16+EA	×			×	×	×	×	×	[EA] ← EA] + [RW] + [C] Add the 16-bit contents of register RW, plus the Carry status, to the data word addressed by DADDR
Spera	ADD	RB,DADDR	02 aadddbbb [DISP][DISP]	9+EA	×			×	×	×	×	×	[RB] ← [EA] + [RB] Add the contents of the data byte addressed by DADDR to register RB
y iou	ADD	RW,DADDR	03 aadddbbb [DISP][DISP]	9+EA	×			×	×	×	×	×	[RW] ← [EA] + [RW] Add the contents of the 16-bit word addressed by DADDR to register RW
reM) eo	ADD	DADDR,RB	00 aasssbbb (DISP][DISP]	16+EA	×			×	×	×	×	-	[EA] ← [EA] + [RB] Add the 8-bit contents of register RB to the data memory byte addressed by DADDR
nereteR v	ADD	DADDR,RW	01 aasssbbb [DISP][DISP]	16+EA	×			×	×	×	×	<u>~</u>	[EA] ← [EA] + [RW] Add the 16-bit contents of register RW to the data memory word addressed by DADDR
Memory	AND	RB,DADDR	22 sadddbbb [DISP][DISP]	9+EA	0			×	×	¬	×	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
Condary	AND	RW,DADDR	23 aadddbbb [DiSP][DISP]	9+EA	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
əs	AND	DADDR,RB	20 aasssbbb (DISP](DISP]	16+EA	0		-	×	×		×	-	[EA] ← [EA] AND [RB] AND the 8-bit contents of register RB with the data memory byte addressed by DADDR Store the regult in the addressed data memory byte.
	AND	DADDR,RW	21 aasssbbb [DISP][DISP]	16+EA	0			×	×		×	<u> </u>	[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	×			×	×	×	×	×	[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
· .	СМР	RW,DADDR	38 aadddbbb [DISP]{DISP]	9+EA	×			×	×	×	×	×	[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)

		[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	[EA] – [RW]	Subtract the 16-bit contents of register HW from the data memory word addressed by DADDR. Discard the result, but adjust status flags	$[EA] \leftarrow [EA] - 1$	Decrement the contents of the memory location addressed by DADDR. De-	pending on the prior definition of DADDR, an o-bit of a To-bit memory loca- tion have be decremented	[AX] ← [AX]/[EA]	Divice 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	[DX] [AX] — [DX] [AX]/[EA]	Divice 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 in-	teger quotient in AX and the remainder in DX. If the quotient is greater than	FFFF16, execute a "divide by O" interrupt	$[AX] \leftarrow [AX]/[EA]$ Divide the 1E his contents of societies AV by the 9 his contents of the moment	Divide the To-on comens of register AA by the o-on 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 7F16, or less	than -80 ₁₆ , execute a "divide by 0" interrupt	$[DX] [AX] \leftarrow [DX] [AX]/[EA]$	Divice 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 7FFF16, or less than -800016, execute a "divide	by O' interrupt	[AX] ← [AL] • [EA]	Multiply the 8-bit contents of register AL by the contents of the memory byte	addressed by UADDH. I reat both numbers as signed binary numbers, store the 16-bit product in AX	DXI [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)
			a men flags		a merr flags	:	ed by	110-0		ents o	L and	200	d AX (ADDR.	ient is		9			e the	r than			¥	ADD	88 88	y nur	у, өхө		117	of the	ary no		bit co	'S 8S	and /	
		RB]	act the 8-bit contents of register RB from the data ad by DADDR. Discard the result, but adjust status t	RWJ	act the 16-bit contents of register MW from the data ad by DADDR. Discard the result, but adjust status t	[EA] — 1	ment the contents of the memory location address:	ng on the prior definition of UADUR, an 8-bit of a 1d hav be decremented	[AX]/[EA]	the 16-bit contents of register AX by the 8-bit conte	ddressed by DADDR. Store the integer quotient in AL	. It trie quotreiit is greatei triaii Fr16, execute a loiv ki ← [DX] [AXI/[EA]	the 32-bit contents of registers DX (high-order) and	3-bit contents of the memory word addressed by DA	quotient in AX and the remainder in DX. If the quoti	6, execute a "divide by 0" interrupt	[AX]/[EA]	the ro-bit coments of register AA by the 6-bit contra	Iddressed by DADDR, treating both contents as signs	the quotient, as a signed binary number, in AL. Store	signed binary number, in AH. If the quotient is greater	8016, execute a "divide by 0" interrupt	$(I) \leftarrow [DX] [AX]/[EA]$	the 32-bit contents of register DX (high-order) and	3-bit contents of the memory word addressed by D	nts as signed binary numbers. Store the quotient, a	er, in AX. Store the remainder, as an unsigned binan	otient is greater than 7FFF ₁₆ , or less than -8000 ₁₆	interrupt	[AL] • [EA]	yly the 8-bit contents of register AL by the contents of	ssed by DADDH. Treat both numbers as signed binds. But product in AX	State Proceed in AS (1 ← [AX] • [EA]	My the 16-bit contents of register AX by the 16-b	ry word addressed by DADDR. Treat both numbers	ers. Store the 32-bit product in DX (high-order word)	
] [Y3]	Subtra	[EA] - [Subtra	[EA] ←	Decrai	tion 7	[AX]	Divide	byte a			the 16	teger		⋖	חואום	Dytea	Store	an nus		₽	Divide	the 16	conter	agmnu T	the ou	o kg	¥ :	Multip	address	₽		memo	numbe	word)
	С	×		×					0			n					<u> </u>		_)	_						×			×				┒
	Ы	X		×		×			0			ר)						>					_		_			=	·			_]
	٨	×		×		×			<u> </u>								<u> </u>						2							0 0			Ξ				4
88s	z	X		×		×			_			Ω					_		_				5										Ξ				┙
Statuses	S	×		×		×			_								_		_				2							_			_=				4
S.	_																								_					_							4
	1	L				_													_								_										4
	٥	_																																			4
L	0	×	_	×		×			_								2	_	_				_						;	<u>~</u>			_×				4
0.10	CIOCA CYCIES	9+EA		9+EA		15+EA		,	(86-96)+EA			(150-168)+EA					(107-118)8+EA U						(171)-190)+EA U							(86-104)+EA			(134-160)+FA				
	enos codo	38 aasssbbb	[DISP][DISP]	39 aasssbbb	[ASIO][ASIO]	11111118	aa001bbb	[DISP][DISP]	F6 ag110bbb	[DISP][DISP]		F7 aa110bbb	[DISP][DISP]				F6 aa111bbb	[ASIO][ASIO]					F7 aa111bbb	[DISP][DISP]						F6 aa101bbb	(DISP][DISP]		F7 aa101hbb	[DISP][DISP]			
	Operand (a)	DADDR,RB		DADDR,RW.		DADDR			AX,DADDR			DX DADOR					AX,DADDR						DX,DADDR						:	AL, DADDR			AX DADDR				
		CMP		CMP		DEC			٥١٨			2	: :				Ν		_				<u>≥</u>						:	MUL			11/41	}			
əd	_							(pen	ntiti	(Con	(9)	1816	odO	ιλ	ow	eM.	9:	oue	919	ìeF	۱ ۸	10n	ıəı	N /	(18	puc))	s								7

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

90					L		ľ	Statuses	808	1	l	
ΚŢ	Mnemonic	Operand(s)	Ubject Code	Clock Cycles	0	a	Ī	S	z	₹		Operation Performed
	INC	DADDR	1111111a aa000bb [DISP][DISP]	15+EA	×			×	×	×	×	[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 and the prior definition of DADDR, and 8-bit or a 16-bit memory location and the prior definition of DADDR.
	MUL	AL,DADDR	F6 aa100bbb [DISP][DISP]	(76-83)+EA	×			⊃))		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
(beunitad	MUL	F	F7 aa100bbb [DISP][DISP]	(124-139)+EA	×			>	>	5		(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 page).
D) (etsned	NEG	DADDR	11110118 88011bb [DISP][DISP]	16+EA	×			×	×	×	×	Year) [EA] — [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 taken complemented.
(Memory C	TON	DADDR	1111011a aa010bb [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
eonerete	6	RE,DADDR	OA aadddbbb [DISP][DISP]	9+EA	×			×	×		×	Mean Storipherine Complements (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
emory R	e B	RW,DADDR	OB aadddbbb [DiSP][DISP]	9+EA	×			×	×	5	×	 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
M yasbn	eo E	DADDR,RB	OB aasssbbb [DISP][DISP]	16+EA	×			×	×	>	×	 X [EA] ← [EA] OR [RB] OR the 8-bit contents of register RB with the data memory byte addressed by DADOR. Store the result in the data memory byte.
oses	86	DADDR,RW	09 aassabb [DISP][DISP]	16+EA	×			×	×	3	×	X [EA] ← [EA] OR [RW] OR the 16-bit contents of register RW with the data memory word addressed by DADOR. Store the result in the data memory word
\Box					\Box	\dashv	\dashv	4			ᅱ	

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

	Operation Performed	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:	[EA]	or DADDR may address a word:	[EA+1]	As RCL, but rotate right	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:	C (EA)	[EA+1]	
	O B	×				×	×				
	V										
898	z										
Statuses	S										
ŝ	1										
	<u> </u>						 -·				
	ō	×				×	×				
	Clock Cycles	N=1 15+EA: N>1 4N+20+EA				N=1 15+EA	N>1 4N+20+EA				
0.15	Object Code	110100va aa011bbb [DiSP][DiSP]	110100va ag000bbb			110100va aa001bbb [DiSP][DiSP]	110100va aa000bbb [DISP][DISP]				
,	Operand(s)	DADDR,N	DADDR,N			DADDR,N	DADDR,N				
	мпетопіс	RCL	10 2			ACA	ROL		:		
ed	ΥŢ			(beunitno	O) (e) BreqO	се (Метогу	emory Referen	Secondary Me		7	

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

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

Mnemonic Operand(s) Object Code	Operand(s)		I	Clock Cycles	0	ΙŒ	Statuses T S Z	ses Z	<	•	٥	Operation Performed
SBB DADDR,RB 18 aasssbbb 16+EA X	18 aasssbbb 16+EA [DiSP][DiSP]	18 aasssbbb 16+EA [DiSP][DiSP]	<u> </u>	1^	 ×		×	×	×	×	×	[EA] ← [EA] – [RB] – [C] Subtract the contents of 8-bit register RB from the data byte addressed by
SBB DADDR,RW 19 aasssbbb 16+EA X [DISP][DISP]	19 aasssbbb 16+EA [DISP][DISP]	19 aasssbbb 16+EA [DISP][DISP]		×			×	×	×	×	×	DADDR, using twos complement arithmetic. Decrement the result in data memory if the Carry status was initially set [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
SHL DADDR,N		*	×	×			<u>×</u>	×	>	×	×	in data memory if the Carry status was initially set. This is an alternate mnemonic for SAL
е5	110100va N=1 15+EA;	a N=1 15+EA;	15+EA;	×			×	×		×	×	As SAL, but shift right:
aa 1,01bb N > 1 [DISP][DISP] 4N+20+EA			N>1 4N+20+EA									O [EA]
												o
				-							•••	(EA)
					•							[EA+1] C
SUB RB,DADDR 2A aadddbbb 9+EA X [DISP]	2A aadddbbb 9+EA [DISP][DISP]	2A aadddbbb 9+EA [DISP][DISP]		×			<u>×</u>	_×	×	×	×	<u> </u>
SUB RW,DADDR 2B aadddbbb 9+EA X [DISP][DISP]	2B sedddbbb 9+EA [DISP][DISP]	2B sedddbbb 9+EA [DISP][DISP]		×			×	×	×	×	×	contents of a-on register his, using twos complement antinnetic [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 asssbbb 16+EA X [DISP][DISP]	28 aasssbbb 16+EA [DISP][DISP]	28 aasssbbb 16+EA [DISP][DISP]		×			<u>×</u>	×	×	×	×	[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 16+EA X [DiSP][DiSP]	29 aasssbbb 16+EA IDISP]IDISP]	29 aasssbbb 16+EA IDISP]IDISP]		×			<u>×</u>	×	×	×	×	[EA] — [EA] — [RW] Subtract the contents of 16-bit register RW from the 16-bit data memory
TEST DADOR,RB 84 aaregbbb 9+EA 0	84 aaregbbb 9+EA [DISP][DISP]	84 aaregbbb 9+EA [DISP][DISP]		0			×	×	⊃	×	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
				ı		二						

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

əd		L		ada. O de elo			Ġ	Statuses	808				Louis Jud and Indian
γī		Operanate)	ano palao	Clock Cycles	•	_	۲	s	Z	4	4	ပ	
	TEST	DADDR,RW	85 aareg bbb [DISP][DISP]	9+EA	0			×	×	Ω	×	0	[EA] AND [RW] AND the 16-bit contents of the data memory word addressed by DADDR with
eoneretef (beunitno	XOR	R8,DADDR	32 aaddbbb [DISP][DISP]	9+EA	•		_	×	×)	×		the contents of 16-bit register NV. Discard the result, but adjust status hags, appropriately [RB] — [RB] XOR [EA] Exclusive OR the 8-bit contents of register RB with the data memory byte ad-
	XOR	RW,DADDR	33 aadddbbb (DISP][DISP]	9+EA	•			×	×		×	0	dressed by DADDR. Store the result in RB [RW] ← [RW] XOR [EA] Exclusive OR the 16-bit contents of register RW with the 16-bit data memory
	XOR	DADDR,RB	30 aassabbb {DISP][DISP]	16+EA	0			×	×	2	×	-	word addressed by DADDR. Store the result in RW [EA] — [RB] XOR [EA] Exclusive OR the 8-bit contents of register RB with the data memory byte addressed by DADDB.
	XOR	DADDR,RW	31 aasssbbb (DISP][DISP]	16+EA	0			×	×	>	×	0	desired by DADDR. Store the result in the addressed data memory byte [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
	MOV.	DADDR, DATA8	C6 ae000bbb [DISP][DISP] YY	10+EA									[EA] ← DATA8 Load the immediate data byte DATA8 into the data memory byte addressed
etaibe	MOV	DADOR, DATA16	C7 aa000bbb [DISP][DISP] YYYY	10+EA									[EA] — DATA16. Load the immediate 16-bit data word DATA16 into the data memory word
աալ	MOV	RB,DATA8	10110ddd YY	•4									addressed by UADDR [RB] — DATA8
	MOV	RW,DATA16	101111ddd YYYY	. 4									Load the immediate data byte DATA9 into 8-bit register RB [RW] ← DATA16 Load the immediate 16-bit data word DATA16 into 16-bit register RW
	JMP	BRANCH	111010a1 DISP [DISP]	15**			ļ					_	[PC] ← [PC] + DISP Jump direct to program memory location identified by tabel BRANCH. The
dwnr	JMP	BRANCH, SEGM	ЕА РРОО РРОО	-12 -12				-					uspiconical Control which has been a supplementation of the assembler [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
	MP	DADDR	FF aa 100bbb [DISP][DISP]	18+EA**							···		another 16-bit unsigned data value that is loaded into the CS segment register [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)

MADDR.CS FF as 10 10 bb 1 7 8 2 A P C	əd	7		Object Code	المادين المادي			Sta	Statuses	9			Lowesteed acites
JMP DADDR,CS FF 38101bbb 24+EA** IP CALL BRANCH E8 DISP DISP 13** III CALL BRANCH, BAPQQ PPQQ 28** III CALL BRANCH, BAPQQ PPQQ 28** III CALL DADDR FF as010bbb 21+EA** III CALL DADDR FF as011bbb 37+EA** III CALL DADDR, GRANCH FF 11010reg 16** III CALL DADDR FF 11010reg 16** III RET CS CB 12** IP RET CS YYYY 17** IP RET CS.DATA16 CA YYYY 18** IP	ΤY	мпетопіс	Uperangis/	Object Code	CIOCK CYCIBS		Ξ	-	-	_		-	
JMP RW FF 11100reg 11 IP IP CALL BRANCH E8 DISP DISP 19** III CALL BRANCH E8 DISP DISP 19** III CALL BRANCH 9A PPQQ PPQQ 28** III CALL DADDR FF aso11bbb 21+EA** III CALL DADDR, CS FF aso11bbb 37+EA** III CALL DADDR, CS FF aso11bbb 37+EA** III RET CS CB 12** IP RET CS CB 12** IP RET CS DATA16 CA YYYY 18** IP	(.tno	JMP	DADDR,CS	FF aa101bbb [DISP][DISP]	24+EA**						<u> </u>		[PC] ← [EA], [CS] ← [EA+2] Jump indirect into a new segment. The 16-bit contents of the data memory
CALL BRANCH, 9A PPQQ PPQQ 28** CALL BRANCH, 9A PPQQ PPQQ 28** SEGM CALL DADDR FF ae010bbb 21+EA** CALL DADDR FF ae011bbb 37+EA** CALL DADDR,CS FF ae011bbb 37+EA** CALL DADDR,CS FF ae011bbb 16** RET CS CB 12** RET CS.DATA16 CA YYYY 18** RET CS.DATA16 CA YYYY 18**	o) du												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
CALL BRANCH E8 DISP DISP 19** III CALL BRANCH, 9A PPQQ PPQQ 28** III CALL DADDR FF ae010bbb 21+EA** III CALL DADDR,CS FF ae011bbb 37+EA** III CALL RW FF 11010reg 16** II RET CS CB 12** IP RET CS CB 12** IP RET CS,DATA16 CA YYYY 18** IP	ոսև	JMP	WW	FF 11100reg	11								[PC] ← [RW] Jump to memory location whose address is contained in register RW.
CALL BRANCH, 9A PPOQ PPQQ 28** CALL DADDR FF as010bbb 21+EA** CALL DADDR, CS FF as011bbb 37+EA** CALL CADDR, CS CB 15** RET CS CB 12** RET CS.DATA16 CA YYYY 18** RET CS.DATA16 CA YYYYY 18** RET CS.DATA16 CA YYYYYY 18** RET CS.DATA16 CA YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY		CALL	BRANCH	E8 DISP DISP	19.				-				$[(SP)] \leftarrow [PC], [SP] \leftarrow [SP] -2, [PC] \leftarrow [PC] + DISP$
CALL DADDR FF aa010bbb 21+EA** [IS CALL DADDR, FF aa011bbb 37+EA** [IS DISPIJDISP] 37+EA** [IS RET C3 8** [P PRT C2 YYYY 17** [P PRT C5.DATA16 CA YYYY 18** [P PRT C5.DATA16 CA YYYY] [P PRT C5.DATA16 CA YYYY 18** [P PRT C5.DATA16 CA YYYY] [P PRT C5.DATA16 CA YYYY] [P PRT C5.DATA16 CA YYYY] [P PRT C5.DATA16 CA YYYY 18** [P PRT C5.DATA16 CA YYYY] [P PRT C5.DATA16 CA YYYY 18** [P PRT C5.DATA16 CA YYYY] [P PRT C5.DATA16 CA YYYYY] [P PRT C5.DATA16 CA YYYYYY] [P PRT C5.DATA16 CA YYYYYY] [P PRT C5.DATA16 CA YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY		CALL	BRANCH,	9А РРОО РРОО	28**								Th.
CALL DADDR FF aa010bbb 21+EA** CALL DADDR,CS FF aa011bbb 37+EA** CALL RW FF 11010reg 16** RET CS CB 12** RET CS.DATA16 CA YYYY 18** RET CS.DATA16 CA YYYY 18**			SEGM										DATA16, CSI ← DATA 16
CALL DADDR FF as010bbb 21+EA** [16] CALL DADDR,CS FF as011bbb 37+EA** [16] CALL RF 11010reg 16** [16] CALL RW FF 11010reg 16** RET C3 8*** [P RET CS CB 12** RET C2 YYYY 17*** [P RET CS,DATA16 CA YYYY 18***	-												Call a subroutine in another program segment using direct addressing. RRANCH and SEGM are labels that become different 16-bit data words than
CALL DADOR, CS FF aa011bbb 37+EA** CALL DADOR, CS FF aa011bbb 37+EA** CALL RW FF 11010reg 16** RET CS CB 12** RET CS CB 12** RET CS,DATA16 CA YYYY 18** PP P P P P P P P P P P P P P P P P P			9	1000									are loaded into PC and CS, respectively Color Color Color Color Color Color
CALL DADDR,CS FF ag011bbb 37+EA** [IS [DISP][DISP]] CALL RW FF 11010reg 16** RET C3 8** RET CS CB 12** RET CS CB 12** RET CS CB 12** RET CS AYYYY 17** [P P P P P P P P P P		- CALL	HOUADO	(DISP][DISP]	Z1 + EA								Call a subroutine in the current program segment using indirect addressing.
CALL DADOR,CS FF as011bbb 37+EA** [IS CALL RW FF 11010reg 16** [S FT 1010reg 16** RET CS CB 12** [P FT 17** RET CS.DATA16 CA YYYY 18** [P FT 1010reg 16** RET CS.DATA16 CA YYYYY 18** [P FT 1010reg 16** RET CS.DATA16 CA YYYYY 18** [P FT 1010reg 16** RET CS.DATA16 CA YYYYY 18** [P FT 1010reg 16** RET CS.DATA16 CA YYYYY 18** [P FT 1010reg 16** RET CS.DATA16 CA YYYYY 18** [P FT 1010reg 16** RET CS.DATA16 CA YYYYY 18** [P FT 1010reg 16** RET CS.DATA16 CA YYYYYYYY 18** [P FT 1010reg 16** RET CS.DATA16 CA YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY													The address of the subroutine called is stored in the 16-bit data memory
CALL RW FF 11010reg 16" RET C3 8" RET CS CB 12" RET C2 YYYY 17" RET CS.DATA16 CA YYYY 18"	пı	CALL	DADDR,CS	FF aa011bbb	37+EA**	<u> </u>							word addressed by DADDH [[SP]] \leftarrow [PC], [SP] \leftarrow [SP] \leftarrow [FC] \leftarrow [EA],
CALL RW FF 11010reg 16" RET C3 8" RET CS CB 12" RET CS.DATA16 CA YYYY 18" [S	mə		•	[DISP][DISP]				-					[CS] ← [EA+2]
CALL RW FF 11010reg 16" [P RET CS CB 12" [P RET CS,DATA16 CA YYYY 18" [P RET CS,DATA16 CA YYYYY 18" [P RET CS,DATA16 CA YYYYYY 18" [P RET CS,DATA16 CA YYYYYY 18" [P RET CS,DATA16 CA YYYYYY 18" [P RET CS,DATA16 CA YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY	A pu												Call a subroutine in a different program segment using indirect addressing.
CALL RW FF 11010reg 16" [S FET C3 8" [P FET C5 C8 12" [P FET C5,DATA16 CA YYYY 18" [P FET C5,DATA16 CA YYYY 18" [P FET C5,DATA16 CA YYYY 18" [P FET C5,DATA16 C4 YYYYY 18" [P FET C5,DATA16 C4 YYYYYY 18" [P FET C5,DATA16 C4 YYYYYY 18" [P FET C5,DATA16 C4 YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY	16 H												word addressed by DADDR. The new CS register contents is stored in the
Second S	e)						7			-:			next sequential program memory word
RET CS CB 12*** IP RET DATA16 C2 YYYY 17*** IP RET CS,DATA16 CA YYYY 18*** IP	enit	CALL	W	FF 11010reg	16.		·						$[SP] \leftarrow [PC], [SP] \leftarrow [SP-2], [PC] \leftarrow [RW]$
RET CS CB 12** [P RET DATA16 C2 YYYY 17** [P RET CS,DATA16 CA YYYY 18** [P	prou	RET		ឌ	.								Call a subroutine whose address is contained in register RW. $[PC] \leftarrow [[SP]]$, $[SP] \leftarrow [SP] + 2$
DATA16 C2 YYYY 17** C5,DATA16 CA YYYY 18**	ns	RET	8	83	12								Return from a subroutine in the current segment [PC] ← [(SP]], [SP] ← [(SP], [SP] + 2.
CS.DATA16 CA YYYY 18"			-							· · · -			Return from a subroutine in another segment
CS,DATA16 CA YYYY 18**		Æ	DATA16	C2 YYYY	17			-					[PC] ← [(SP]), [SP] ← [SP] +2 +DATA16
CS,DATA16 CA YYYY 18**													Netum from a subroutine in the current segment and add an infinediate dis- placement to SP
Return from a subroutine in another segment and add an immediate displace-ment to SP		RET	CS,DATA16	CA YYYY	18.								$[PC] \leftarrow [(SP]], [SP] \leftarrow [SP] + 2, [CS] \leftarrow [(SP]], [SP] \leftarrow [SP] + 2 + DATA16$
									· · · •	٠,			Return from a subroutine in another segment and add an immediate displace-
						_							ment to or
											_		

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

I						ı			I			ŀ	
ed/	Mnemonic	Mnemonic Operand(s)	Object Code	Clock Cycles	_	ŀ	ő	Statuses	8 E	[1	_	Oneration Performed
<u>۱</u> ۲					•	급	-	S	z	⋖	_	٥	
	ADD	AL,DATA8	04 YY	4.	×	_		×	×	×	×	×	[AL] ← [AL] + DATA8
	ADD	AX,DATA16	05 7777	4	×			×	×	×	×	×	Add 8-bit immediate data to the AL register [AX] — [AX] + DATA16
													Add 16-bit immediate data to the AX register
	ADD	RB,DATA8	80 11000ddd YY	4	×			×	×	×	×	<u>~</u>	(RB) — [RB] + DATA8
	ADD	RW.DATA16	81 11000ddd	4.	×			×	×	×	×	×	Add e-bit infinediate data to the no register [RW] ← [RW] + DATA16
					:	_					:		Add 16-bit immediate data to the RW register
	ADD	DADDR,	80 aa000bbb	17+EA	×			×	×	×	×	×	[EA] ← [EA] + DATA8
		DATA8	(DISP][DISP] YY										Add 8-bit immediate data to the data memory byte addressed by DADDR
	ADD	DADDR,	81 aa000bbb	17+EA	×			×	×	×	×	×	$[EA] \leftarrow [EA] + DATA16$
	-	DAIAIB	IDISPILDISPI YYYY	•	,			•	,	;	->	_	Add 16-bit immediate data to the data memory word addressed by DADDR
	Ą	A. CA 140	¥ 4 4	4	×			<u> </u>	<	ζ_	<	-	[AL] + [AL] + DA AS + [C]
	ADC	AX,DATA16	15 YYYY	•4	×			×	×	×	×	×	Add 8-bit immediate data, plus carry, to the AL register [AX] \leftarrow [AX] \leftarrow DATA16 + [C]
											_	_	Add '6-bit immediate data, plus carry, to the AX register
91	ADC	E,DATA8	80 11010ddd YY	*	×			×	×	×	×	×	[RB] ← [RB] + DATA8 + [C]
6190	ADC	RW DATA16	81 11010ddd	4.	×			>	>	×	×	×	Add 8-bit immediate data, plus carry, to the RB register [RW] ← [RW] + DATA16 + [C]
ю	!				:			<u> </u>	:		:		Add 16-bit immediate data plus carry to the RW register
etgi	ADC	DADDR,	80 aa010bbb	17+EA	×			×	×	×	×	×	$[EA] \leftarrow [EA] + DATA8 + [C]$
peu		DATA8	(DISP)[DISP] YY								_		Add 8-bit immediate data, plus carry, to the data memory byte addressed by
յալ		6		1		_		:		- 3		_	DADDR
ı	ADC	DATA16	81 aa010bbb	17+EA	×			×		×	×	<u>~</u>	$[EA] \leftarrow [EA] + DATA16 + [C]$ And 16-hit immediate data plus carry to the data memory word addressed
					_						_		And 10-bit milliediate data, plus carry, to the data mentily word addressed. by DADDR
	AND	AL,DATA8	24 YY	4.	0			×	×	Ξ	×	0	[AL] ← [AL] AND DATA8
	9		2000	•	-			>	,	:	,		AND 3-bit immediate data with AL register contents
	}	24,44	20 1111	4	_			<	<	_	<	5	[AK] — [AK] AND DATATO AND 18 his immediate data with AV register contents
	AND	RB,DATA8	80 11100ddd YY	*4	0			×	×		×	-	AND 10-bit infinedate data with AA register contents [RB] ← [RB] AND DATA8
													AND 3-bit immediate data with RB register contents
	AND W	RW,DATA16		. 4	0			×	×	2	×	0	[RW] ← [RW] AND DATA16
	:	1	٨٨٨٨										AND 16-bit immediate data with RW register contents
	ON P	CADDR,8	80 aa100bbb	1 / +EA	0			×	×	<u> </u>	×	-	[EA] \leftarrow [EA] AND DATA8
													DADOR
	AND	DADDR,	81 aa100bbb	17+EA	0			×	×)	×	-	[EA] — [EA] AND DATA16
		DATA16	[DISP][DISP] YYYY								-		AND 16-bit immediate data with contents of 16-bit data memory word ad-
1					\exists	\dashv	4	┛	_]		┪	┪	

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

ALDATA8 3C YY 4* X X X X X X X X X X X X X X X X X X X	
3D YYYY 3D YYYY 4	
3D YYYY 80 11111ddd YY 1000000a1 Y [YY] 80 aa111bbb 10+EA	
100000a1 1111ddd YY	Subtract 8-bit immediate data from AL register contents. Discard result, but adjust status flags
80 11111ddd YY	X] — DATATO Subtract 16-bit immediate data from AX register contents. Discard result, but
100000a1 1111ddd	sõe
10000091 1111ddd YY [YY] 80 aa111bbb	Subtract 8-bit immediate data from RB register contents. Discard result, but
1111ddd	9008
80 aa111bbb 10+EA X X X X X X E E [DISP][DISP] YY 100000a1	Subtract 16-bit immediate data from RW register contents. Discard result, but adiust status flags
Displibisp YY	
100000a1 aa111bbb [DisPi[DisP]YY[YY]]	Subtract 8-bit immediate data from contents of data memory byte addressed by DADDR. Discard result, but adjust status flaos
DISP [DISP] VY [YY]	
DISP][DISP][VY][YY]	Subtract 16-bit immediate data from contents of 16-bit data memory word
80 11001ddd YY 4* 0	addressed by DADDR. Discard result, but adjust status flags
80 11001ddd YY	DATA8
80 11001ddd YY	OR 8-bit immediate data with AL register contents XI ← [AX] OR DATA16
81 11001ddd YY	16-bit immediate data with AX register contents
81 11001ddd 4* 0	DATA8
80 ae001bbb 17+EA 0	OR 8-bit immediate data with RB register contents
80 aa001bbb 17+EA 0	R DATA 16
[DisP][DisP]	nedrate data with RVV register contents DATA 8
81 aa001bbb 17+EA 0	OR 8-bit immediate ata with contents of data memory byte addressed by
81 aa001bbb 17+EA 0	
OSPINISP VYYYY	DATA16
x x x x x x x x x x x x x x x x x x x	OR 16-bit immediate data with contents of 16-bit data memory word addressed by DADDR
10 yyyy 4 * X X X X X X X X X X X X X X X X X X	ATA8 – [C]
10 yyyy 4 * X X X X X X X X X X X X X X X X X X	Subtract 8-bit immediate signed binary data from AL register contents using
1D YYYY 4* X X X X X X X X X X X X X X X X X X	twos complement arithmetic. If the Carry status was originally 1 decrement
	- DATA16 - [C]
	Subtract 16-bit immediate signed binary data from AX register contents
using twos complement	using twos complement arithmetic. If the Carry status was originally 1 decre-

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

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

Onerend(e)	Object Code	Clock Cycles			۱	2000				Control Dark Control
è	enon speco	CIOCK CYCIES	0		1	s	Z	٧	Ь	Operation renormed
AX,DATA16	A9 YYYY	4.	0	\vdash	<u> </u>	×	×	n	×	0 [AX] AND DATA16
R3,DATA8	F6 11000ddd YY	ណំ	0			×	×		×	AND the 16-bit immediate data and AX register contents. Discard the result but adjust status flags [RB] AND DATA8
				_						
RW,DATA16	F7 11000ddd	ໍລ	0			×	×	\supset	×	0 RWJ AND DATA16
	, AAAA								٠.	AND the 16-bit immediate data and RW register contents. Discard the result
DADDR,	F6 as000bbb	11+EA	0			×	×	5	×	but agjust status flags O [EA] AND DATA8
DATA8	[DISP][DISP] YY									AND the 8-bit immediate data and the contents of the data memory location
DADDR,	F7 aa000bbb	11+EA	0		<u>.</u>	×	×	'n	×	(EA) AND DATA16
DATA16	(DISP][DISP]YYYY			<u>.</u>						AND the 16-bit immediate data and the contents of the 16-bit data memory
A DATAR	34 ^^	•	_				>	=	,	word addressed by DADDR. Discard the result but adjust status flags
2	- - -	t	>			<	<	_	<	
AX,DATA16	35 YYYY	.4	0			×	×)	×	0 [AX] ← [AX] XOR DATA16
								•		
RB,DATA8	80 11110ddd YY	•4	0			×	×	5	×	0
RW,DATA16	81 11110ddd	*	0			×	×	_	×	0 [RW] — [RW] XOR DATA16
	YYYY	ſ								Exclusive OR 16-bit immediate data with RW register contents
DADDR,	80 aa010bbb	17+EA	0			×	×	<u> </u>	×	0 [EA] ← [EA] XOR DATA8
DATA8	YY [980][980]									Exclusive OR 8-bit immediate data with contents of the data memory byte addressed by DADDR
DADDR,	81 aa010bbb	17+EA	0			×	×	Э	×	0 [EA] ← [EA] XOR DATA16
DATA16	[DISP][DISP]									Exclusive OR 16-bit immediate data with contents of the 16-bit data memory word addressed by DADDR
DISP8	E2 DISP	5 or 17**			ļ					$[CX] \leftarrow [CX] - 1 \text{ if } [CX] \neq 0 \text{ then } [PC] \leftarrow [PC] + DISP8$
DISP8	E1 DISP	6 or 18**								Decrement CX register and branch if CX contents are not 0 [CX] \(\times \text{[CX]} - 1 \) [CX] \(\times \text{ and } \text{[Z]} = 1 \) then \(\text{[PC]} + \text{DISP8} \)
										Decrement CX register and branch if CX contents is not 0 and Z status is
DISP8	EO DISP	5 or 19**	_							$[CX] \leftarrow [CX] - 1 \text{ if } [CX] \neq 0 \text{ and } [Z] = 0 \text{ then } [PC] \leftarrow [PC] + DISP8$
DISP8										Decrament CX register and branch if CX contents is not 0 and 2 status is 0 see LDOPNE
DISP8			_	-	_					See LOOPE
DISP8	77 DISP	4 or 16**								[PC] ← [PC] + DISP8
				_		ä				Branch if C or Z is 0

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

	Operation Performed		[PC] → [PC] + DISP8 Reacch if C is 0	[PC] — [PC] + DISP8	Branch if C is 1	[PC] ← [PC] + DISP8	Branch if C or Z is 1	Report if the CX register contents is 0	[PC] ← [PC] + DISP8	Branch if Z is 1	[PC] ← [PC] + DISP8	Branch if Z is 0 or the S and 0 statuses are the same	Branch if the S and O statuses are the same	[PC] ← [PC] + DISP8	Branch if the S and O statuses differ	[PC] ← [PC] + DISP8	Branch if Z is 1 or the S and O statuses differ	See JBE	See JB	See JAE	See JA	[PC] ← [PC] + DISP8	Branch if Z is 0	See JLE	See JL	See JGE	See JG	[PC] ← {PC] + DISP8	Branch if O is O		BCIIICHTTS OF THE POST OF THE	Branch if S is O	See JNE	[PC] ← [PC] + DISP8	Branch if O is 1	[PC] ← [PC] + DISP8	Branch if P is 1	JC aec
ſ		C																							_													
-	<u>"</u> [V Z																																				
	Statuses	8											_		_					_					_									_				
١	ا ة	<u>-</u>				_						···							···														_	_	_			
	ļ	٥			•			_																														
-		°						-												_					-									_	_			_
	Clock Cycles		4 or 16**	4 or 16**		4 or 16"	8 or 18**	5	4 or 16**		4 or 16**	10.10.	5	4 or 16**		4 or 16**						4 or 16**						4 or 16**	4 0. 16.	5	4 or 16**			4 or 16**		4 or 16**		
	Object Code		73 DISP	72 DISP	000	Ve UISP	F3 DISP	2	74 DISP		7F DISP	90.07	2	7C DISP		7E DISP						75 DISP						71 DISP	950	200	79 DISP			70 DISP		7A DISP		
	Operand(s)		DISP8	DISP8	0	SPSIC	Basic	2	DISP8		DISP8	90010	2	DISP8		DISP8		DISP8	DISP8	DISP8	DISP8	DISP8		DISP8	DISP8	DISP8	disp8	DISP8	000	0	DISP8		DISP8	DISP8		DISP8	9000	SHSIO
	Mnemonic		JAE	AB.	į	H	LX2	7	띡		ಶ	10	Š	٦		JLE		AN S	JNAE	eg.	NBE	P.F.		SNG	SNGE	룅	jnfe	ON N	9	5	SNS		ZNC	욱		₽,	פַ	JPE
	\be													(P	ənu	iitn	၀၁	u	itic	pu	ာ၁	uO	ųэ	n B	18													

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

Σ	JPO JS JZ MOV	Mnemonic Operand(s) JPO DISP8 JS DISP8 JZ DISP8 MOV RBD,RBS	Object Code 78DISP 8A11dddsss	Clock Cycles	0	_	σ <u>-</u>	Statuses Z Z Z Z	N N	4	۵	U	See JNP {PC] — [PC] + DISP8 Brench if S is 1 See JE [RBD] ← [RBS]
x x x	MOV MOV XCHG XCHG	SR.RW RW.SR AX.RW RB.RB	8B 11dddsss 8E 110rrsss 8C 110rrddd 10010reg 86 11regreg					· · · · · · · · · · · · · · · · · · ·					Move the contents of any RB register to any RB register [RWD] → [RWS] Move the contents of any RW register to any RW register [SR] → [RWS] Move the contents of any RW register to any Segment register [RWD] ← [SR] Move the contents of any Segment register to any RW register [AX] → (RW) Exchange the contents of AX and any RW register [RB] → (RB] Exchange the contents of any two RB registers [RW] ← → [RW] Exchange the contents of any two RW registers
	CMPS LODS MOVS	BD,BS WD,WS WD,WS WD,WS	A6 A7 A5 A5	22 22 12 18 18	× ×	0, 0, 0, 0, 0,	<u> </u>	× ×	× ×	× ×	× ×	××	[[SI]] – [[DI]], [SI] ← [SI] ± 1, [DI] ← [DI] ± 1 Compare the data bytes addressed by the SI and DI Index registers using string data addressing* [[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* [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 [AX] ← [[SI]], [SI] ← [SI] ± 1 Move a data word from the 16-bit location addressed by the SI Index register to the extra segment location addressed by the DI register using string data addressing [[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* [[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)

					L		۱				l	۲	
ed/	Mnemonic	Operand(s)	Object Code	Clock Cycles		ł	۰ŀ	Sasman	<u>§</u>		Ī	٦	Operation Performed
Ţ		•	_		0	_	_	S	Z	٧	4	o	
()	REP	Z	1111001z	+2 per loop		<u> </u>						-	Repeat the next sequential instruction (which must be a Block Transfer and Search instruction) until CX contents decrements to 0. Decrement CX con-
penui													tents on each repeat. If the next instruction is CMPB, CMPW, SCAB, or SCAW then repeat until CX contents decrements to 0 or 2 status does not
tnoO)	SCAS	BD,BS	AE	15	×	9		×	×	×	×	, ×	equal N [AL] – [[Di]], {Di] ← [Di] ± 1
Search	Š	9		ļ		ĺ		, ,					Compare AL register contents with the extra segment data byte addressed by the DI Index register using string data addressing
pue	SCA3	SW.UWS	ŧ	<u>0</u>	<	<u> </u>		<u> </u>	<	Κ.	<	₹	IAX) = ILDII, IDII = IDII = Z Compare AX register contents with the extra segment 16-bit data ord ad-
ıəten	STOS	BD,BS	Ψ¥	1	×	ō		×	×	×	×	×	dressed by the DI Index register using string data addressing [[DI]] \leftarrow [AL], [DI] \leftarrow [DI] \pm 1
erT X20	STOS	SW CW	æ	***	>	<u> </u>		<u> </u>	<u> </u>	>	×	×	Store the AL register contents in the extra segment data memory byte addressed by the DI Index register using string data addressing
18	3	î	?	•				<u> </u>					Storethe AX register contents in the extra segment 16-bit data memory word addressed by the DI Index register using string data addressing
	ADC	RBD, RBS	12 11dddsss	3.	×	 	 	<u>×</u>	×	×	×	×	[RBD] ← [RBD] + [RBS] + [C] Add the 8-bit contents of register RBS, plus the Carry status, to register RBD.
	ADC	HWD,RWS	13 11dddsss	ŧn.	×			×	×	×	×	×	[RWD] — [RWS] + [C]
													Add the 16-bit contents of register RWS, plus the Carry status, to register RWD
	ADD	RBD,RBS	02 11dddsss	*n	×	-		×	×	×	×	×	[RBD] ← [RBD] + [RBS]
9161	ADD	RWD,RWS	03 11dddsss	*	×			×	×	×	×	×	Add the 8-bit contents of register RBS to register RBD [RWD] [RWD] + [RWS]
eq0 1	AND	RBD,RBS	22 11dddsss	÷	0			, ×	. ×		×	0	Add the 16-bit contents of register RWS to register RWD [RBD] — [RBD] AND [RBS]
etsig	AND	HWD,RWS	23 11dddsss	÷	0			×	×		×	Ö	AND the 8-bit contents of register RBS with register RBD [RWD] — [RWD] AND [RWS]
eg -	7400		o	;			_						AND the 16-bit contents of register RWS with register RWD
1672	8		0	٧									Extend AL sign bit into AH
igeA	CMP	RBD,RBS	3A 11dddsss	ě	×			×	×	×	×	×	[RBD] - [RBS] Subtract the contents of vaciety BBD from society BBC Discoud the society
	QV.	RWD RWS	38 11dddese	**	<u> </u>	_		×	×	×	×	×	but adjust status flags
					<						:		Subtract the contents of register RWD from register RWS. Discard the result,
	CWD		66	S								<u></u>	but adjust status flags .[DX] ← [AX15]
						一	4	긕					Extend AX sign bit into DX

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

								Sta	Statuses	88			
Mnemonic		Operand(s)	Object Code		Clock Cycles	0	-	느	S	7	A	۴	Operation Performed
≧G		RBS	F6 11110sss	SSS	80-90	<u> </u>	├)	5	-	5	[AX] ← [AX]/[RBS]
λi		RWS	F7 11110sss	s,	144-162	5			2			n n	
VIO		RBS	F6 11111sss	ss	101-112	5					<u> </u>		the 10-bit contents of NWS. Store the integer quotient in AX and the remainder in DX. If the quotient is greater than FFFF16, execute a "divide by 0" interrupt 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 AX.
∑iQi		RWS	F7 11111888	SS	165-184	5			<u> </u>	3	<u>ה</u>		Is grader than 7F16, or less than ~6016, execute a lighten by 0 interrupt U [DX] [AX] ← [DX] [AX]/[RWS] Divice 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.
IM UL	<u> </u>	RBS	F6 11101sss	SS SS	80-98	×	· · · · · · · · · · · · · · · · · · ·		5	<u>_</u>		×	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 [AX] — [AL] • [RBS]
MUL		RWS	F7 11101sss	SS	128-154	×					<u>_</u>		
MUL		RBS	F6 11100sss	 \$	~	×	* ************************************						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) [AX] — [AL] • [RBS]
MUL		RWS	F7 11100sss	s,	118-133	×	<u> </u>				<u>n</u>		≘
æ		RBD,RBS	OA 11dddss	sss	*E	0	· · · · ·		×	×	×	0	뜨
Ж		RWD,RWS	OB 11dddsss	s s	ŧ.	0			×	×	÷	ŏ	OR the 8-bit contents of register RBS with register RBD [RWD] ← [RWD] OR [RWS] OR [RWS] OR the 16-bit contents of register RWS with register RWD
	—	,								_		_	

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

əd	Magmonia	Onerendie)	Ohiest Code	مواصل بإصران			Š	Statuses	808				Domination Dades
ΥT				san la wan	0	-	Ī	s	z	¥	4	0	
	SBB	RBD,RBS	1A 11dddsss	3.	<u> </u> ×	\vdash	<u> </u>	×	×	×	×	ı. X	[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
(P	SBB	HWD,HWS	18 11dddsss	'n	×			×	×	×	×	= × .	$[RWD] \leftarrow [RWD] - [RWS] - [C]$
en													Subtract the 16-bit contents of register RWS from RWD using twos comple-
nitr		200	200000000000000000000000000000000000000	ċ				>	_,				ment arithmetic. If the Carry status was originally 1 decrement the result
10 O	900	CBD, GBD	ZA I IODOSSS	, '	\			Κ	<	-	<u>.</u>	-	[MBD] ← [MBD] ← [MBS] Subtract the R-bit contents of register RBS from RBD using twose complement
93													arithmetic
619	SUB	RWD,RWS	2B 11dddsss	÷	×			×	×	×	×		RWD] → [RWS] – [RWS]
dQ													Subtract the 16-bit contents of register RWS from RWD using twos comple-
19										_	_	_	mentarithmetic
tsię	TEST	RBD,RBS	84 11 regreg	m	0			×	×	\supset	×	0	[RBD] AND [RBS]
Вe												_	AND the 8-bit contents of register d and register RBS. Discard the result, but
- 1													adjust status flags
0 19	TEST	RWD,RWS	85 11 regreg	ě	0			×	×	5	×	-	[RWD] AND [RWS]
ijβ													AND the 16-bit contents of register RWD and register RWS. Discard the
эЯ						-					_	-	result, but adjust status flags
	XOR	RBD,RBS	30 11dddsss	÷	0			×	×	¬	×	=	[RBD] ← [RBD] XOR [RBS]
_											_		Exclusive OR the 8-bit contents of register RBS with register RBD
	XOR	RWD,RWS	31 11dddsss	÷	0			×	×	כ	×	-	[RWD] ← [RWD] XOR [RWS]
													Exclusive OR the 16-bit contents of register RWS with register RWD
	AAA		37	4.	2	\vdash	L	5	⊇	×	5	×	ASCII adjust Al register contents for addition (as described in accompanying
													text)
	AAD		D5 0A	9	<u> </u>			×	×	_	×	<u></u>	Decimal adjust dividend in AL prior to dividing an unpacked decimal divisor,
						_							to generate an unpacked decimal quotient. (See accompanying text for
							-					-	detaik)
8	AAM		D4 0A	83	<u> </u>			×	×	<u></u>	×	5	After multiplying o unpacked decimal operands, adjust product in AX to
161													become an unpacked decimal result. (See accompanying text for details)
ed	AAS		胀	4	_			>	5	×	5	×	After subtracting two unpacked decimal numbers, adjust the difference in AL
O 1													so that it too is an unpacked decimal number. (See accompanying text for
918						_				_	_	-	details)
ige	DAA		27	. 4	5			×	×	×	×	×	After adding two packed decimal numbers, adjust the sum in AL so that it too
Я													is a packed decimal number. (See accompanying text for details)
	DAS		2F	. 4	>			×	×	×	×	×	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)
_	ည	e	FE 11001ddd	*n	×			×	×	×	×	=	[RB] - [RB] -1
	į	č	777	i	. ;			;	;		-;		Decrement the 8-bit contents of register RB
_) O	Š	01001888	7.	×			×	×	×	×	=_	[RW] ← [KW] −1
					╛	\dashv	4]		7	ᅦ	┥	Decrement the 16-bit contents of register RW

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

əd	T T T T T T T T T T T T T T T T T T T	(a)pagada	Object Code	Selection for the			S	Statuses	868				
			enon toelen	CIOCK Cycles	0	D	1	S	7	٨	۵	ပ	
	INC	BB	FE 11000ddd	3*	×	⊢	L	×	×	×	×	Ē	[RB] ← [RB] +1
	Ŋ.	¥	01000ddd	2.	×			×	×	×	×		Increment the 8-bit contents of register RB [RW] ←1
	NEG	88	F6 11011ddd	, "	×			×			×	×	Increment the 16-bit contents of register RW [RB] — [RB] +1
	NEG	¥.	F7 11011ddd	*n	×	,		×			×		Twos complement the 8-bit contents of register RB [RW] ← [RW] + 1
(NOT	8 2	F6 11010ddd	*e									Twos complement the 16-bit contents of register RW [RB] — [RB]
bəuniti	TON	W	F7 11010ddd	÷.									Ones complement the 8-bit contents of register RB [RW] ← [RW]
поЭ)	Z,	RB,N	110100v0 11010ddd		×							×	Unes complement the 16-bit contents of register MW Rotale left through Carry the 8-bit contents of RB register, or the 16-bit
916	RCL	RW,N	110100v1 11010ddd		×							×	contents of RW register, as illustrated for memory operate
əd	2 2	NS N	110100v0 11011ddd		× >			_				× >	Rotate right through Carry the 8-bit contents of HB register, or the 15-bit
) 19:	ğ	RB,N	110100v0 11000ddd	1	×							×	Rotate left the 8-bit contents of RB register, or the 16-bit contents of RW
sig	ROL	RW,N	110100v1 11000ddd	N>1 4N+8	×							×	register as illustrated for memory operate
Beg	ã E	RB,N	110100v0 11001ddd		×							×	Rotate right the 8-bit contents of RB register, or the 16-bit contents of RW
	ê ê	RW.R	110100v1 11001ddd		××			,				××	register, as illustrated for memory operate
	NA C	Z, 2, 2, 2	מפאחחווו טאחחוחוו		Κ :			<u> </u>			< >	× >	Shift left the 8-bit contents of RB register, or the 16-bit contents of RVV
	SAR	7. 8. N. N.	110100v1 11100ddd 110100v0 11111ddd		××			<u> </u>	××))	××	< ×	register, as illustrated for memory operate Shift right the 8-bit contents of register.
	SAR	RW,N	110100v1 11111ddd		×			×			×	×	RW, as illustrated for memory operate
	SH	RB,N			×	_		<u>×</u>		2	×	×	See SAL
	SH	RW,N			×			×		⊃	×	×	See SAL
	똜	RB,N N,N	110100v0 11101ddd 110100v1 11101ddd	N=1 2* N>1 4N+8	×			××	××	5 5	××	××	Shift right the 8-bit contents of register RB, or the 16-bit contents of register RW, as illustrated for memory operate
	POP	DADDR	8F aa000bbb	17+EA		1	+	1	_			Ť	[EA] ← [(SP]], (SP] ← [SP] + 2
			[DISP][DISP]									_	
	POP	M	01011ddd	00		_		_					bit data memory word addressed by DADDR. Increment SP by 2 [RW or SR] ← [[SP] ← [SP] ← 1
	PO	SR	000rr111	80									Load the 16-bit Stack word, addressed using Stack addressing, into the
BCK				-									specified 16-bit register, Increment SP by 2.
s	į Ž	,	9	20	×	×	×	Χ`	×	×	×	×	LORM] ← [LOP], [SP] ← [SP] + 2 Load the 16-bit Stack word, addressed using Stack addressing, into the
	HSITA	DADDB	FF as 110bbb	18+EA									Status Flags register [SP] ← [SP] ← [SP] ← [SP] ← [SP]
			[DISP][DISP]	; i			· ·	1					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 tv 2
1					1	H	4	4	4]	1	1	è

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

	P C	[SP] ← [SP] −2, [[SP]] ← [RW or SR] Store the contents of the specified 16-bit register in the 16-bit Stack word	addressed using Stack addressing. Decrement SP by 2 [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	Execute a software interrupt and vector through table entry 3	Executs a software interrupt and vector through table entry V	If the 0 status is 1, execute a software interrupt and vector through table en-	try 1016 Return from interrupt service routine	0 [C] + 0	Clear Carry status	0 1 [0]	Clear Decrement/Increment select	Clear Interrupt enable status, disabiling all interrupts		Complement Carry status Transfer flags to AH register as follows:	7 6 5 4 3 2 1 0 Bit no.	AH register	SZOAOPIC	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	1 [C] ← 1	Set Carry status to 1	Set Decrement/Increment status to 1		sat merupt enable status to 1, enabiing an interrupts	
88	ZA							-						-					×									
Statuses	s																		×									
ιώ	-	,			_	0			_																			4
l	_	·			٦	0			\vdash		-		_											_		-		┨
l	0				<u> </u>				H		_											•						┪
	Clock Cycles	11 10			52	51	4 or 53	24	2.		2.	**	ı	2*	•4				•4				5.	3.	ı	2.		
	9000 109(aO	01010rr 000rr110	10		ខ	χ, g ₂	뜅	ñ	F8		ပ္	FA		F£	16				9E		`		69	.63	<u>.</u>	82		
	Operand(s)	RW) 6		60	>																						
	Мпетопіс	PUSH PUSH	PUSHF		Ī	Ī	OLNI	IRET	CLC		일	7	j	CMC	LAHF				SAHF				STC	STD)	STI		
əc	ΙĶΤ	(.tno	ж (C	Stac	s	adr	JITA	tul										sna	B) Z	1								

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

Clock Cyoles O D 1 T S Z A P R R R R R R R R R R R R R R R R R R		Operation Ferturines	7 — [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 CPU Halt Guarantee the CPU bus control during execution of the next sequential instruction The next sequential allowed memory reference instruction accesses the segment identified by Segment register SR. See Table 20-1 for allowed memory reference instructions CPU enters the WAIT state until TEST pin receives a high input signal No operation (This is the same object code as XCHG, AX, AX.)	
Clock Cycles		ЬС		
Clock Cycles 0 D 1 2 2 2 4 2 4 2 3 4 5 3 4 5 1 4 5 4 5 4 5 4 5 4 5 4 5 4 5 4 5 4 5 4 5 5		∢		
Clock Cycles 0 D 1 2 2 2 4 2 4 2 3 4 5 3 4 5 1 4 5 4 5 4 5 4 5 4 5 4 5 4 5 4 5 4 5 4 5 5	Statuses	lacksquare		
Clock Cycles 0 D 1 2 2 2 4 2 4 2 3 4 5 3 4 5 1 4 5 4 5 4 5 4 5 4 5 4 5 4 5 4 5 4 5 4 5 5		$oldsymbol{}$		· · · · · · · · · · · · · · · · · · ·
Clock Cycles 0 8+EA 3+5n 3*		Ξ		
Clock Cycles - 2* 2* 4 5 3* 3*				
Clock Cycle 8+EA 3+5n 3**	<u> </u>			
9bject Code 11011xxx axxxbbb DISP][DISP] F4 F0 90 98 90	Clock Cycles		8+EA 2° 2° + 2 3+5n 3*	
	Object Code		aaxxxbbb [DISP][DISP] F4 F0 O01reg110	
DADDR SR	Mnemonic Operand(s)		DADDR	
Mnemonic ESC HLT LOCK SEG WAIT NOP	Mnemonic		ESC HLT LOCK SEG SEG WAIT	
Other Type			1941O	