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

				Pos	ssible Displacemer	its	Assembly
Memory Reference	Segment Register	Base Register	Index Register	16-Bit Unsigned	8-Bit High-order Bit Extended	None	Language Operand Mnemonic
			Ši///	//// <b>X</b> ////	******		
	DS	None	DY		×		
	(Alternate*		sı	/// <b>*</b> ////	<b>*</b>		
Normal Data	CS, SS or ES)	BX	DU	/// <b>/</b>	<b>*</b>		
Memory Reference			None			/// <b>X</b> ////	
Reference	DS	None	None				
	SS		SV.				
	(Alternate*	89					
	CS, DS of ES)		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	nese columns co	ontribute to OEA	۸.		This colum
	<u> </u>	TI	nese columns co	ontribute to EA.			to be provid

	4//	
4		11/1

Shaded rows apply to EA and DADDR.



Shaded row applies to EA and LABEL.

<sup>\*</sup> 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<sub>16</sub> or FF<sub>16</sub>)

AX Accumulator, both bytes

AX15 The value of register AH high-order bit (0 or 1) extended to a 16-bit word (0000<sub>16</sub> or FFFF<sub>16</sub>)

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
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 255<sub>10</sub> 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:

CH

DH

BH

110 =

111 =

SI

DΙ

```
[]
             indicate an optional object code byte
а
             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
               O11 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:
rr
               00 = ES
               01 = CS
               10 = SS
               11 = DS
             three binary digits identifying a register:
reg
                      16-bit 8-bit
               = 000
                       ΑX
                              Αl
               001 =
                        CX
                              CL
               010 =
                        DX
                             DΙ
               011 =
                        BX
                              BI.
               100 =
                       SP
                              AH
                       BP
               101 =
```

represents three binary digits identifying a source register (see reg)
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)
000 000 000 001 001 001 010 010 011 011	(BX) + (SI) (BX) + (SI) + DISP8 (BX) + (SI) + DISP16 (BX) + (DI) (BX) + (DI) + DISP8 (BX) + (DI) + DISP16 (BP) + (SI) (BP) + (SI) + DISP16 (BP) + (SI) + DISP16 (BP) + (DI) (BP) + (DI) + DISP16 (BP) + (DI) + DISP16 (SI) ir (DI) or (BD) or (BX) + DISP16 8-bit immediate 16-bit immediate	7 11 11 8 12 12 8 12 12 7 11 11 5	7 11 15 8 12 16 8 12 16 7 11 15 5

- 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

Эd		(a)proces		3000			S	Statuses	1808			
			Object code	CIOCK CYCIES	0	<b>1</b>	Ī	S	z	٧	Ь	C Operation Performed
	2	AL,PORT	E4 YY	10		$\vdash$	<u> </u>					[AL] ← [PORT]
	Z	AL.[DX]	EC 1	00								Load one byte of data from I/O port PORT into AL [AL] ← [PDX]
				,								Load into AL one byte of data from I/O port whose address is held in the DX
	2	AX PORT	, sa	10								register [Al] ← [PORT] [AH] ← [PORT±1]
												Load 16 bits of data into AX, AL receives data from I/O port PORT, AH
	Z	AX,[DX]	ED	∞								receives data from I/O port POR I + 1 [AL] ← [PDXI, [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
0/1	OUT	AL,PORT	E6 YY	10								PORT → [AL]
	!	3	į									Output one byte of data from register AL to I/O port PORT
	5	AL,(DX)	H	<b>x</b> o								[PDX] — [AL] Outsuit one hate of date from register AI to the I/O nort whose address is hald
												in the DX register
	TUO	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.
	TUO	AX.[DX]	<b>#</b>	∞								The AH register contents are output to I/O port PORT+1 [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
	San	RW,DADDR	C5 aasssbbb	16+EA								[RW] ← [EA], [DS] ← [EA+2]
90												register RW. Load 16 bits of data from the next sequential memory word into
nere	LEA	RW,DADDR	8D aasssbbb	2+EA								the DS register [RW] ← OEA
ete			[DISP][DISP]									Load into RW the 16-bit address displacement which, when added to the
Ory	1 52	andan Wa	CA secephh	18.454								segment register contents, creates the effective data memory address
we <sub>l</sub>	}		[DISP][DISP]	5								Load 16 bits of data from the memory word addressed by DADDR into
ΜV												register RW. Load 16 bits of data from the next sequential memory word into
nem	\Q <b>Y</b>	90000	444666	ф ф								the ES register
i19	<u> </u>	חטטאט,פֿע	[DISP][DISP]	0+EA								Into Terms (Into International International Internation addressed by DADDR to International Interna
												register RB

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

	P. C.	[RW] ← [EA]	Load 16 bits of data from the data memory word addressed by UADDR to	register nvv	Store the data bute from register BR in the memory bute addressed by DADDR	Store the date byte hom register his memory byte addressed by DADDH.  [EA] ← [RW]	Store the 16-bit data word from register RW in the memory word addressed	by DADDR	[AL] ← [EA]	Load the 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	AX	[EA] ← [AL]	Store the 8-bit contents of register AL into the data memory byte directly ad-	dressed by LABEL	[EA] ← [AX]	Store the 16-bit contents of register AX into the data memory word directly	addressed by LABEL	[SR] ← [EA]	Load into Segment register SK the contents of the 16-bit memory word ad-	dressed by DADDR	[EA] ← [SR]	Store the contents of Segment register SR in the 16-bit memory location ad-	dresed by DADDR	[RB] — [EA]	Exchange a byte of data between register RB and the data memory location	addressed by DADDR	[RW] → [EA]	Exchange 16 bits of data between register RW and the data memory location	addressed by DADDR	[AL] ← [[AL] + [BX]]	Load into AL the data byte stored in the memory location addressed by sum-	ming initial AL contents with BX contents				
	Operation relication	W] ← [EA]	Load 16 bits of data from the data memory word add	register nvv A] ← [RB]	Store the data hote from register BB in the memory byte.	Stole file data byte inoin register inc in the inoiner; y zyte. A] ←: [RW]	Store the 16-bit data word from register RW in the mer	by DADDR	.L] ← [ <b>EA</b> ]	Load the data memory byte directly addressed by LAB	$X] \leftarrow [EA]$	Load the 16-bit data memory word directly addressed b	AX	A] ← [AL]	Store the 8-bit contents of register AL into the data mer	dressed by LABEL	A) ← [AX]	Store the 16-bit contents of register AX into the data n	addressed by LABEL	R] ← [EA]	Load into Segment register SR the contents of the 16-	dressed by DADDR	A] ← [SR]	Store the contents of Segment register SR in the 16-bit	dresed by DADDR	B] → [EA]	Exchange a byte of data between register RB and the d	addressed by DADDR	W] → [EA]	Exchange 16 bits of data between register RW and the c	addressed by DADDR	$[L] \leftarrow [[AL] + [BX]]$	Load into AL the data byte stored in the memory locatio	ming initial AL contents with BX contents				
		(F)		1	<u> </u>	<u> </u>		_	₹	_	<u>₹</u>	_	_	<u> </u>		į	<u>)</u>			S			<u> </u>			Ξ.	_		<u>É</u>	_	_	₹	_			_	 	
		<u> </u>			-	<del></del>					_	_		-													_									—	 	
	V										—	_				-	_							-		-		_					•					
368	2							_	_							_	_	_				_				_						_		_			 	
Statuses	S																																		_		 	
S	T	<u> </u>									_														,												 	
	1 0	<u> </u>										_				_																	_				 	
	0 [	<u> </u>									_	_										-					_										 	
	Clock Cycles	8+EA	· · · · · ·	9+FA	<u>.</u>	9+EA			0		10			<u></u>		(	2			8+EA		i	9+EA		į	17+EA			17+EA			-						
	Opject Code	88 aadddbbb	[PSIG][PSIG]	88 aassabbb	DISPIDISPI	89 aasssbbb	[DISP][DISP]		A0 PPQQ		A1 PPQQ			A2 PPQQ			As Prod			8E aaOrrbbb	[ASIO][ASIO]		8C agorrbbb	[DISP][DISP]		86 aaregbbb	[DISP][DISP]		87 aaregbbb	[DISP][DISP]		D7					-	
	Operand(s)	RW,DADDR		DADDR RB		DADDR,RW			AL,LABEL		AX,LABEL			LABEL,AL			LABEL, AX			SR,DADDR			DADDR,SR		1	RB,DADDR			RW,DADDR									
	мпетопіс	MOV		<b>N</b>	) }	MOV			MOV		MOV			<b>≥</b>			<b>&gt;</b>			<b>№</b>			٥ ع			XCHG			XCHG			XLAT				_		
	2																																					

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

				L	1	ီ	Statuses	Ses				
Mnemonic	Operand(s)	Object Code	Clock Cycles	ŀ	⊢	⊢	⊢		E		T	Operation Performed
				<u></u>	<u>~</u>	<del>-  </del>	ေ	7	⋖	_	ပ	
ADC	RB,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 aaddbbb [DISP][DISP]	9+EA	×			<u>×</u>	×	×	×	×	to register HB [RW] ← [EA] + [RW] + [C] Add the contents of the 16-bit data word addressed by DADDR, plus the Car-
ADC	DADDR,RB	10 aasssbbb [DISP][DISP]	16+EA	×			×	×	×	×	×	If status, to register have [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	×			×	×	×	×	×	[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	×			×	×	×	×	×	[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	×			<u>×</u>	×	×	×	×	[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	×			<u>×</u>	×	×	×	×	[EA] ← [EA] + [RB] Add the 8-bit contents of register RB to the data memory byte addressed by DADDR
ADD	DADDR,RW	O1 aasssbbb [DISP][DISP]	16+EA	×			×	×	×	×	×	[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			×	×	<b>)</b>	×	0	[RB] ← [EA] AND [RB] AND the 8-bit contents of register RB with the data memory byte addressed by DADDR Store the regult in BB
AND	RW,DADDR	23 aadddbbb [DISP][DISP]	9+EA	0			×	×	<b>&gt;</b>	×	0	[RW] — [EA] AND [RW] AND the 16-bit contents of register RW with the data memory word addressed by DADDR Store the regult in RW
AND	DADDR,RB	20 aasssbbb [DISP][DISP]	16+EA	0			×	×	⊃	×	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			×	×	<u> </u>	×	0	[EA] — [EA] MOD [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
CMP	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)

Statuses	0 D 1 T S Z A P C	9+EA 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	9+EA 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	15+EA X X X X [EA] ← [EA] − 1  Decrement the contents of the memory location addressed by DADDR. De-  Decrement the contents of the memory location addressed by DADDR. De-  Department of the memory location of DADDR and 8-bit or a 16-bit memory location.	(86-96 )+EA U U U U U U I [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 i	teger quotient in AX and the remainder in DX. If the quotient is greater than FFFF <sub>16</sub> , execute a "divide by O" interrupt 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 $-8O_16$ , execute a "divide by 0" interrupt than $-8O_16$ , execute a "divide by 0" interrupt [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 AA. Store the remainder, as an unsigned binary number, in AH. If the quotient is greater than 7FFF <sub>16</sub> , or less than -8000 <sub>16</sub> , execute a "divide by O" interrupt  (86-104)+EA X	the 16-bit product in AX  (134-160)+EA X   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)
عواصدي بإموان		-		E E			07-118)8+EA U	71)-190)+EA U		134-160)+EA X
Object Code		38 aasssbbb [DISP][DISP]	39 aasssbbb [DISP][DISP]	1111111a aa001bbb	E Ballobbb [DISP]	F7 aa110bbb ('	F6 aa111bbb (1 [DISP][DISP]	F7 aa111bbb (1 [DISP][DISP]	F6 sa 101bbb [DISP][DISP]	F7 aa101bbb (CDISP][DISP]
Onerend(e)		DADDR,RB	DADDR,RW.	DADDR	AX,DADDR	DX,DADDR	AX,DADDR.	DX,DADDR	AL, DADDR	AX,DADDR
o i do monto		CMP	CMP	DEC	ΔIO	≥io	NGI	VIOI	IMUL	IMUL
əd					(beunitr	Operate) (Con	յсе (Memory (	Memory Refere	Secondary	

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

90						Š	Statuses	808		1	
Ι <b>Υ</b> Ι	Minemonic	Operand(s)	Ubject Code	Clock Cycles	0	_	S	2	₹	O d	Operation Performed
	INC	DADDR	11111118 as000bb [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 of DADDR.
	MUL	AL,DADDR	F6 aa100bbb [DISP][DISP]	(76-83)+EA	×	 	Þ	<b>5</b>	5	×	
(penu	MUL	F7	F7 aa100bbb [DISP][DISP]	(124-139)+EA	×	 	⊃	)	5	×	٥
perate) (Contin	NEG	DADDR	1111011a aa011bb [DISP][DISP]	16+EA	×		×	×	×	×	<u> </u>
O Y10meM)	NOT	DADDR	1111011a aa010bb [DISP][DISP]	16+EA							De twos complemented [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
esterence	8	RB,DADDR	OA aadddbbb [DISP][DISP]	9+EA	×	 · · · · · · · · · · · · · · · · · · ·	×	×	5	×	ones complemented [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	SB B	RW,DADDR	OB aadddbbb [DISP][DISP]	9+EA	×	 	×	×	5	×	医
M Yısbn	8	DADDR,RB	08 aasssbbb [DISP][DISP]	16+EA	×	 ······································	×	×	5	×	Щ
ooes	8	DADDR,RW	09 aasssbbb [DISP][DISP]	16+EA	×	 	×	×	5	×	<u></u>

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:	[EA+1]
	РС	×	-			×	×		
	V						<del></del>		
80	Z								
Statuses	S								
St	T								
				·	·	· ·	· · · · · · · · · · · · · · · · · · ·	:	
	٥٥	×			····	×	×		
	_								
	Clock Cycles	N=1 15+EA; N>1 4N+20+EA				N=1 15+EA	N>1 4N+20+EA		
	Object Code	110100va aa011bbb [DISP][DISP] 110100va aa000bb				110100va aa001bbb [DISP][DISP]	110100va aa000bb [DiSP][DISP]		
	Operand(s)	DADDR,N DADDR,N				DADDR,N	DADDR,N		
	мпетопіс	RCL				RCR	ROL	<u>.</u>	
	ΥT		(	beunijno	O) (etsred)	ce (Memory C	nereleR yrome	Secondary Me	

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

ſ											····	1	o n	.:
	Operation Performed	As ROL, but rotate right	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:	C [EA]	or DADDR may address a word:	C (EA)	[EA+1]	As SAL, but shift right and propagate sign:	[EA]	OF [EA]	(EA+1)	[RB] ← [RB] – [EA] – [C] Subtract the contents of the data byte addressed by DADDR from the con-	tents or o-bit register Rb, using twos complement antinmetic. Decrement the result in RB if the Carry status was initially set  [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
	С	×	·					×				×	×	
	A							× 		<u> </u>		×	×	
ဖွ	1/2				-			×				$\frac{\hat{x}}{x}$	$\frac{\sim}{\times}$	
Statuses	S	L						×			:	×	×	
Sta	E													
	0	×			····					<del></del>		×	×	
-							•	. ×						
	Clock Cycles	N=1 15+EA						N=1 15+EA; N>1	Α̈́			⋖	∢	
	ن پر	1 15						. <del>1</del> 5	4N+20+EA			9+EA	9+EA	
	္မိ	Z						N=1 1 \	* 4 * +				<b>U</b>	
	Object Code	110100va aa001bbb [DISP][DISP]						110100va	<u> </u>			1A aaddd bbb [DISP][DISP]	1B aadddbbb [DiSP][DiSP]	
	(S)	z						z				- <del>K</del>	E C	
	Operand(s)	DADDR,N						DADDR,N				RB,DADDR	RW,DADDR	
		DA						Ā				RB,C	A.Y.	
	Mnemonic		SAL					SAR				SBB	S88	
9	ΤVE			(pen	nitno	) (etste	mory Op	M) eo	neteren	ry Memo	Seconda			

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

Statuses  Statuses	0 D I T S Z A P C	× × × × × × × × × × × × × × × × × × ×	× × × × × ×	20+EA	Subtract the contents of the data memory byte addressed by DADDR from the contents of 8-bit register RB, using twos complement arithmetic contents of 8-bit register RB, using twos complement arithmetic x x x x x   RWJ + [RW] - [EA] Subtract the contents of the 16-bit data memory word addressed by DADDR from the contents of the 16-bit data memory word addressed by DADDR from the contents of the 16-bit data memory word addressed by DADDR from the contents of 16-bit data memory word addressed by DADDR	× × × × × × × × × × × × × × × × × × ×	O ×
□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □	× × × × × × × × × × × × × × × × × × ×	× × × × × × × × × × × × × × × × × × ×			× × × × × × × × × × × × × × × × × × ×	× × × × × × × × × × × × × × × × × × ×	X X X
	6+EA 6+EA 15+EA; 10-20+EA	15+EA; 1 -20+EA	4N+20+EA			16+EA ×	0 +EA
18 aasssbbb [DISP][DISP] [DISP][DISP] aa 101bb [DISP][DISP]	18 aasssbbb [DISP][DISP] 19 aasssbbb [DISP][DISP] 110100va aa101bb [DISP][DISP]	[DISP][DISP] 110100va aa101bb [DISP][DISP]	[dsid][dsid]		2A aadddbbb [DISP][DISP] 2B aadddbbb [DISP][DISP]	28 aasssbbb [DISP][DISP] 29 aasssbbb [DISP][DISP]	84 aaregbbb [DISP][DISP]
DADDR,RW DADDR,N DADDR,N	DADDR,RW DADDR,N DADDR,N	DADDR,N DADDR,N			RB,DADDR	DADDR,RB	DADDR,RB
SBB SBB SHR	SHR SHR	SHL			SUB	SUB SUB	TEST

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

							8	Statuses	Ses	ı	ĺ		
d/	Mnemonic	Operand(s)	Object Code	Clock Cycles	ŀ	ŀ	-	-			1	Ţ	Operation Performed
Ĺ,					0	긎	۲	s	Z	٧	۵	O	
	TEST	DADDR,RW	85 aareg bbb	9+EA	0	<u> </u>	<u> </u>	×	×	n	×	0	[EA] AND [RW]
			[DISP][DISP]										AND the 16-bit contents of the data memory word addressed by DADDR with
(p												_	the contents of 16-bit register RW. Discard the result, but adjust status flags
ren nue	XOR	RB,DADDR	32 aadddbbb	9+EA	0			×	×	)	×	0	The contract of the second se
ete ituc			[DISP][DISP]										Exclusive OR the 8-bit contents of register RB with the data memory byte ad-
л у (С(												_	dressed by DADDR. Store the result in RB
nor (e)	XOR	RW,DADDR	33 aadddbbb	9+EA	0			×	×	D	×	0	[RW] ← [RW] XOR [EA]
len eta			[DISP][DISP]								_		Exclusive OR the 16-bit contents of register RW with the 16-bit data memory
M dc							_						word addressed by DADDR. Store the result in RW
ary Y (	XOR	DADDR,RB	30 aasssbbb	16+EA	0			×	×	)	×	0	[EA] ← [RB] XOR [EA]
pu Jou			[DISP][DISP]					_					Exclusive OR the 8-bit contents of register RB with the data memory byte ad-
ecc Ver													dressed by DADDR. Store the result in the addressed data memory byte
V) S	XOR	DADDR,RW	31 aasssbbb	16+EA	0			×	×	5	×	0	[EA] ← [RW] XOR [EA]
			[DISP][DISP]										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
	λOM	DADDR	C6 ag000bbb	10+EA		$\vdash$	$\vdash$	_	<u> </u>				[EA] ← DATA8
		DATAB	[DISP][DISP] YY										Load the immediate data byte DATA8 into the data memory byte addressed
					_								by DADDR
93	MOV	DADDR.	C7 aa000bbb	10+EA								_	(EA) ← DATA16
sit		DATA16	(DISP][DISP] YYYY										Load the immediate 16-bit data word DATA16 into the data memory word
θU						_							addressed by DADDR
ոտլ	MOV	RB,DATA8	10110ddd YY	**									[RB] — DATA8
													Load the immediate data byte DATA8 into 8-bit register RB
	ΜO	RW,DATA16	10111ddd YYYY	•4									[RW] ← DATA16
													Load the immediate 16-bit data word DATA16 into 16-bit register RW
	JMP	BRANCH	111010a1	15**		$\vdash$	<u> </u>	<u> </u>					[PC] ← [PC] + DISP
			DISP [DISP]										Jump direct to program memory location identified by label BRANCH. The
												_	displacement DISP which must be added to the Program Counter will be com-
													puted as an 8-bit or 16-bit signed binary number, as needed, by the assembler
(	d W	BRANCH,	EA PPOO PPOO	15**									[PC] ← DATA16, [CS] ← DATA16
lw		SEGM					_						Jump direct into a new segment. BRANCH is a label which becomes a 16-bit
սԸ												_	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
	AMS	DADDR	H- aa 100bbb	18+EA**			<u>.</u>					_	[PC] ← [EA]
			[ASIGISA]										Jump indirect in current segment. The 16-bit contents of the data memory
												_	Word addressed by UADUR is loaded into PC
I					l	ł	ļ	l	1	I	J	١	

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

JMP DADDR,CS FF aa101bbb 24+EA**  JMP RW FF 11100reg 111  CALL BRANCH, 9A PPOQ PPOQ 28**  CALL BRANCH, 9A PPOQ PPOQ 28**  CALL DADDR FF aa011bbb 37+EA**  CALL DADDR,CS FF aa011bbb 37+EA**  CALL CALL DADDR,CS FF aa011bbb 16**  RET CS CB 12**  RET CS CB 12**  RET CS CB 12**  RET CS.DATA16 CA YYYY 18**	λbe	Mnemonic	Operand(s)	Object Code	Clock Cycles		-	s [	Statuses	<u>s</u>	<b> </b>	-	Operation Performed
JMP DADDR,CS FF as101bbb 24+EA**  CALL BRANCH, BA PPOQ PPOQ 28**  CALL BRANCH, 9A PPOQ PPOQ 28**  CALL DADDR FF as010bbb 21+EA**  CALL DADDR FF as010bbb 37+EA**  CALL DADDR,CS FF as011bbb 37+EA**  CALL DADDR,CS FF as011bbb 37+EA**  CALL DADDR,CS FF as011bbb 37+EA**  FF 11010reg 16**  CALL DADDR,CS FF as011bbb 37+EA**  FF 11010reg 16**  FF 12017 FF 11010reg 16**  FF 1101	(1					_		T				_	
CALL BRANCH E8 DISP 19" [16]  CALL BRANCH, 9A PDQQ PDQQ 28" [16]  CALL DADDR FF aa010bbb 21+EA" [16]  CALL DADDR FF aa011bbb 37+EA" [16]  CALL DADDR, CS FF aa011bbb 37+EA" [16]  RET CS CB 12" [16]  RET CS CB 12" [16]  RET CS.DATA16 C2 YYYY 18" [16]	('1uo;	JMP	DADDR,CS	FF aa101bbb [DISP][DISP]	24+EA**						<del>                                     </del>		[PC] ← [EA], [CS] ← [EA+2]  Jump indirect into a new segment. The 16-bit contents of the data memory
CALL         BRANCH         EB DISP DISP         19**         [16]           CALL         BRANCH, 9A PPQQ PPQQ         28**         [16]           CALL         DADDR         FF as010bbb         21+EA**         [16]           CALL         DADDR,CS         FF as011bbb         37+EA**         [16]           CALL         RET         C3         8**         [16]           RET         CS         CB         12**         [16]           RET         CS         CB         12**         [17]           RET         CS YYYY         17**         [17]           RET         CS,DATA16         CA YYYY         18**         [18]	) dwnr	MP	W.	FF 11100reg	=					·			word addressed by DADDH is loaded into PC. The fiext sequential 10-bit data memory word's contents is loaded into the CS segment register [PC] — [RW] Jump to memory location whose address is contained in register RW.
CALL BRANCH, 9A PPQQ PPQQ 28**  SEGM  CALL DADDR FF aa010bbb 21+EA**  CALL DADDR,CS FF aa011bbb 37+EA**  CALL RW FF 11010reg 16**  RET C3 8**  RET CS CB 12**  RET CS CAYYYY 17**  RET CS.DATA16 CAYYYY 18**  P. P. P. P. PPQQ PPQQ 28**  P. P. PPQQ PPQQ PPQQ 28**  P. PPQQ PPQQ PPQQ PPQQ PPQQ PPQQ PPQQ		CALL	BRANCH	E8 DISP DISP	19**					<del>                                     </del>			$[(SP)] \leftarrow [PC], [SP] \leftarrow [SP] - 2, [PC] \leftarrow [PC] + DISP$
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 C2 YYYY 18**  RET CS.DATA16 CA YYYY 18**		CALL	BRANCH,	9A PPQQ PPQQ	28**								_
CALL DADDR FF aa010bbb 21+EA**  [DISP][DISP]  CALL DADDR,CS FF aa011bbb 37+EA**  [DISP][DISP]  CALL RW FF 11010reg 16**  RET C3 8**  RET CS CB 12**  RET CS.DATA16 C2 YYYY 18**  [FP			SEG G		_								DATA16, [CS] ← DATA 16
CALL         DADDR         FF as010bbb         21+EA**         [IS           CALL         DADDR,CS         FF as011bbb         37+EA**         [IS           CALL         RW         FF 11010reg         16**         [IS           RET         C3         8**         [IS           RET         C3         8**         [IS           RET         C2 YYYY         17**         [IP           RET         CS,DATA16         CA YYYY         18**										-	·		Call a subroutine in another program segment using direct addressing. BRANCH and SEGM are labels that become different 16-bit data words; they
DADDR,CS FF aa011bbb 37+EA**  RW FF 11010reg 16**  CS CB 12**  DATA16 C2 YYYY 17**  CS,DATA16 CA YYYY 18**  [Istance of the content of the co		CALL	DADDR	FF aa010bbb	21 +EA"								are loaded into PC and CS, respectively [[SP]] ← [PC], [SP] ← [SP] ← [FA]
CALL         DADDR,CS         FF ae011bbb         37+EA**         [IS           CALL         RW         FF 11010reg         16**         [S           RET         C3         8**         [P         [P           RET         CS         CB         12**         [P         [P           RET         DATA16         C2 YYYY         17**         [P         [P           RET         CS,DATA16         CA YYYY         18**         [P				[DISP][DISP]									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
CALL RW FF 11010reg 16"  RET C3 8"  RET C5 CB 12"  RET C5.DATA16 CA YYYY 18"  [P	LU	CALL	DADDR.CS	FF aa011bbb	37+EA**								word addressed by DADDR [[SP]] $\leftarrow$ [SP] $\leftarrow$ [SP] $\leftarrow$ [SP] $\leftarrow$ [SP] $\leftarrow$ [SP] $\leftarrow$ [FC] $\leftarrow$ [EA],
CALL RW FF 11010reg 16"  RET C3 8"  RET CS CB 12"  RET C2 YYYY 17"  RET CS,DATA16 CA YYYY 18"	ntel			[DISP][DISP]									[CS] ← [EA+2]
CALL   RW   FF 11010reg   16**   FF   FF   FF   FF   FF   FF   FF	put												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
CALL   RW   FF 11010reg   16"   [S   FET   CS   CB   12"   [P   FET   CS,DATA16   CA YYYY   18"   [P   FET   CS,DATA16   CA YYYYY   18"   [P   FET   CS,DATA16   CA YYYYYY   18"   [P   FET   CS,DATA16   CA YYYYYY   18"   [P   FET   CS,DATA16   CA YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY	3 IIB												word addressed by DADDR. The new CS register contents is stored in the
RET         CS         CB         12"         IP           RET         DATA16         C2 YYYY         17"         IP           RET         CS,DATA16         CA YYYY         18"         IP	ე eu	CALL	*	FF 11010rea	16.								next sequential program memory word [SP] ← [PC] [SP] ← [SP – 2] [PC] ← [RW]
RET         CS         CB         12**         IP           RET         DATA16         C2 YYYY         17**         IP           RET         CS,DATA16         CA YYYY         18**         IP	ituor											-	Call a subroutine whose address is contained in register RW.
RET         CS         CB         12**         [P           RET         DATA16         C2 YYYY         17**         [P           RET         CS,DATA16         CA YYYY         18**         [P	ıqng	RET		င္မ	<b>.</b>					<u>i</u>			[PC] ← [[SP]], [SP] ← [SP] + 2
DATA16 C2 YYYY 17** [P	3	RET	S	89	12					*	····		[PC] ← [[SP]], [SP] ← [SP] +2, [CS] ← [[SP]], [SP] ← [SP] +2
CS,DATA16 CA YYYY 18**		LJA	DATA18	>>>> c5	17**								Return from a subroutine in another segment [PC] ← I(SP) 1 (SP) ← I(SP) 1 ←
CS,DATA16 CA YYYY 18**		2		7	<u> </u>					_			Return from a subroutine in the current segment and add an immediate dis-
		RET	CS DATA16	CA YYYY	**								placement to SP [SP] ← [SP] ← [SP] ← [SP] ← [SP] +2 +DATA16
ment to SP					•								Return from a subroutine in another segment and add an immediate displace-
			-										ment to SP
				-							_		
					,						-		

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

90		,			L		s	Statuses	8		İ	L	
ΙΥΤ	Mnemonic	Mnemonic Operand(s)	Ubject Code	Clock Cycles	0	1	-	S	Z	4	-	٥	Operation Performed
	ADD	AL,DATA8	04 YY	4.	×	_	<u> </u>	×	×	×	×	X [AL]	[AL] ← [AL] + DATA8
	ADD	AX,DATA16	05 YYYY	*4	×			×	×	×	×	₹ <u>₹</u> ×	Add 8-bit immediate data to the AL register [AX] — [AX] + DATA16
	ADD	RB.DATA8	80 11000ddd YY	•4	×			×	×	×	×	X [BB]	Add 16-bit immediate data to the AX register IRBI — IRBI + DATA8
	}											:	Add 8-bit immediate data to the RB register
	ADD	RW,DATA16	81 11000ddd	4	×			×	×	×	×	× ×	[RW] - [RW] + DATA16
	ADD	DADDR,	80 aa000bbb	17+EA	×			×	×	×	×	× [EA]	Add 10-bit infinediate data to the nov register [EA] — [EA] + DATA8
		DATA8	[DISP][DISP] YY										Add 8-bit immediate data to the data memory byte addressed by DADDR
	ADD	DADDR, DATA16	81 aa000bbb	17+EA	×			×	×	×	×	X [EA]	[EA] — [EA] + DATA16  Add 16.hit immediate data to the data memory word addressed by DADD
	ADC	AL,DATA8	14 YY	•	×			×	×	×	×	<u>¥</u>	And $10^{-5}$ th minimization data to the data minimizer word addressed by DADDA [AL] $+$ DATA8 $+$ [C]
	,	1		;	-				;				Add 8-bit immediate data, plus carry, to the AL register
	ADC	AX,DATA16	15 4444	•4	×			×	×	×	×	×	[AX] ← [AX] + DATA16 + [C]
ə	ADC	B,DATA8	80 11010ddd YY	*4	×			×	×	×	×	X [8B]	Add 16-bit immediate data, plus carry, to the AX register [RB] ← [RB] + DATA8 + [C]
) B10	9	4			;				;				Add 8-bit immediate data, plus carry, to the RB register
dO	ADC	KW,DAIA16	81 11010ddd	4	×			×	×	×	<u>×</u>	¥. ×	$[RW] \leftarrow [RW] + DATA16 + [C]$
ətsi	ADC	DADDR,	80 aa010bbb	17+EA	×			×	×	×	×	×	Add ⊺6-bit immediate data, plus carry, to tne NW register [EA] ← [EA] + DATA8 + [C]
pəw		DATA8	(DISP][DISP] YY									₹ Ĉ	Add 8-bit immediate data, plus carry, to the data memory byte addressed by
wı	ADC	DADDR,	81 aa010bbb	17+EA	×			×		×	×	× [EA]	DAUDH [EA] ← [EA] + DATA16 + [C]
		DATA16	[DISP][DISP] YYYY							-			Add 16-bit immediate data, plus carry, to the data memory word addressed
	AND	AL,DATA8	24 YY	.4	0	-		×	×		×	<u></u> [AL] <sub>U</sub>	by DADDR [AL] — [AL] AND DATA8
		2 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4	2222	•	-	-		>	>			₹ <u>₹</u>	AND 8-bit immediate data with AL register contents
		20,00		+	>			<u> </u>	<	5	_	<u> </u>	AND 16-bit immediate data with AX register contents
	AND	RB,DATA8	80 11100ddd YY	.4	0			×	×	5	×	0 [RB]	(RB) — (RB) AND DATA8
	QNA	RW.DATA16	81 11100ddd	**	c	****		×	×	Ξ	×	¥ §	AND 8-bit immediate data with RB register contents [RW] — [RW] AND DATA16
	!			•	,			:	:				AND 16-bit immediate data with RW register contents
	AND	DADDR,8	80 aa100bbb	17+EA	0			×	×	5	×	0 [EA]	[EA] ← [EA] AND DATA8
			AA [dsig][dsig]		_							₹ ≧	AND 8-bit immediate data with contents of data memory byte addressed by DADDR
	AND	DADDR,	81 aa100bbb	17+EA	0			×	×	<u></u>	×	0 [EA]	[EA] — [EA] AND DATA16
		DATA16	[DISP][DISP] YYYY									₹ 5	AND 16-bit immediate data with contents of 16-bit data memory word addessed by DADDR
1					7	$\dashv$	4		7	1	┪	_	

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

əd	Signomer	Onerand(s)	Object Code	Clock Cycles			S	Statuses	868			Ţ,	Operation Performed
γT					0	٥	_	S	Z	A	Q.	၁	
	CMP	AL,DATA8	3C YY	4.	×			×	×	×	×	×	[AL] - DATA8 Subtract 8-bit immediate data from AL register contents. Discard result. but
	CMP	AX,DATA16.	30 4444	*4	×			<u>×</u>	×	×	×	×	adjust status flags [AX] — DATA16 Subtract It His immediate data from AX ranister contents. Discard result but
	CMP	RB,DATA8	80 11111ddd YY	•4	×			<u>×</u>	×	×	×	×	adjust status flags [RB] — DATA8
	CMP	RW,DATA16		<b>.</b>	×	-:		<u>×</u>	×	×	×	×	Subtract 8-bit immediate data from RB register contents. Discard result, but adjust status flags [RW] - DATA16 Subtract 16-bit immediate data from RW register contents. Discard result,
	CMP	DADDR, DATA8	YY [YY] 80 aa111bbb [DiSP][DiSP] YY	10+EA	×			×	<u>×</u>	×	×	×	but adjust status flags [EA] — DATA8 Subtract 8-bit immediate data from contents of data memory byte addressed
(beunita	CMP	DADDR, DATA16	100000a1 aa111bbb	10+EA	×			×	×	×	×	×	by DADDR. Discard result, but adjust status riags [EA] – DATA16  Subtract 16-bit immediate data from contents of 16-bit data memory word
(Col	OR	AL,DATA8		4.	0			×	×	)	×	0	addressed by DADDA. Discard result, but adjust status flags [AL] ← [AL] OR DATA8
etste	OR	AX,DATA16	7YYY 00	•4	0			×	· ×	)	×	0	OR 8-bit immediate data with AL register contents [AX] $\leftarrow$ [AX] OR DATA16
O etsi	OR	RB,DATA8	80 11001ddd YY	<b>.</b> 4	0			×	×	<u> </u>	×	0	OR 16-bit immediate data with AX register contents [RB] ← [RB] OR DATA8
pewwl	O.	RW,DATA16	81 11001ddd YYYY	•	0			×	×	<u> </u>	×	0	OR 8-bit immediate data with RB register contents [RW] ← [RW] OR DATA 16 OR 16-bit immediate data with RW register contents
	ë	DADDR, DATA8	80 aa001bbb [DiSP][DiSP] YY	17+EA	0			<u>×</u>	×	5	×	0	[EA] — [EA] OR DATA 8  OR 8-bit immediate ata with contents of data memory byte addressed by DADD
	R	DADDR, DATA16	81 aa001bbb [DISP][DISP]	17+EA	0			×	×	<u> </u>	×	0	[EA] ← [EA] OR DATA16 OR 16-bit immediate data with contents of 16-bit data memory word addressed by DADDP
	SBB	AL,DATA8	10 ٧٧	<b>.</b> 4	×			×	×	×	×	×	ulessed by DATA8 − [C] [AL] ← [AL] − DATA8 − [C] Subtract 8-bit immediate signed binary data from AL register contents using
	SBB	AX,DATA16;	1D YYYY	•4	×			×	×	×	×	×	twos complement arithmetic. If the Carry status was originally 1 decrement the result [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
					1	$\dashv$	$\dashv$	4	4	]		_	

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

Operand(s)  RB,DATA8  DADDR, DATA16  AX,DATA16  AX,DATA16  RB,DATA8  RW,DATA16  DADDR, DADDR, DATA8  AX,DATA16
Sub s

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

						St	Statuseş	8			
Mnemonic	ic Operand(s)	Object Code	Clock Cycles	<b>a</b> o	_	E	S	7	AP	၁	Operation Performed
TEST	AX,DATA16	A9 YYYY	4*	0	<u> </u>		×	×	×	0	[AX] AND DATA16
TEGT	0 4 4 4 4 4 4	>> PPPOOCE 8	* u								
3	00,000 00,000		o	>			<	<del>/</del>	<u> </u>		IRBJ AND DATAS AND the 8-bit immediate data and RB register contents. Discard the result but
TEST	RW,DATA16	F7 11000ddd	<b>ئ</b>	0			×	×	<u>×</u>	0	adjust status flags [RW] AND DATA16
								_			AND the 16-bit immediate data and RW register contents. Discard the result
TEST	DADDR,	F6 aa000bbb	11+EA	0			×		× 5	0	but adjust status flags [EA] AND DATA8
	DATA8	[DISP][DISP] YY									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,	F7 aa000bbb	11+EA	0			×	×	<u>×</u>	0	[EA] AND DATABLE
ä	AL DATAB	34 VV	7				>	<del>-</del>		•	AND the To-bit infinediate data and the contents of the To-bit data memory word addressed by DADDR. Discard the result but adjust status flags [AII] ← [AII] × OR DATAR
į		<u>-</u>	t	>							
XOR	AX,DATA16	35 7777	•4	0			×	×	<u>×</u>	0	<u>≤</u>
XOR	RB,DATA8	80 11110ddd YY	4.	0			×	×	× · >	0	Exclusive Off 16-bit immediate data with AX register contents [RB] ← [RB] XOR DATA8
9	ALATAU WA	777077	*	-			>			(	Exclusive OR 8-bit immediate data with RB register contents
<b>5</b>			, t	>				<del>,                                    </del>	<u> </u>		Exclusive OR 16-bit immediate data with RW register contents
XOR	DADDR,	80 aa010bbb	17+EA	0			×	×	<u>×</u>	0	쁘
XOX	DADDR, DATA16	81 aa010bbb [DISP][DISP] YYYY	17+EA	0			×	<u>-</u>	<u>×</u>	0	[EA] ← [EA] XOR DATA16 Exclusive OR 16-bit immediate data with contents of the 16-bit data memory word addressed by DADDR
L00P	DISP8	E2 DISP	5 or 17**		<del> </del>			+-	+	igspace	[CX] ← [CX] −1 If [CX] ≠0 then [PC] ← [PC] + DISP8
LOOPE	DISP8	E1 DISP	6 or 18**								Decrement CX register and branch if CX contents are not 0 [CX] $\leftarrow$ [CX] $\leftarrow$ [CX] $-$ 1 If [CX] $\neq$ 0 and [Z] = 1 then [PC] + DISP8
LOOPNE		E0 DISP	5 or 19**								Decrement CX register and branch if CX contents is not 0 and Z status is 1 [CX] $\leftarrow$ [CX] $\leftarrow$ [CX] $\leftarrow$ 1 if [CX] $\neq$ 0 and [Z] = 0 then [PC] $\leftarrow$ [PC] + DISP8
0000											Decrement CX register and branch if CX contents is not 0 and Z status is 0
LOOPZ				· ·							See LOOPE
4	DISP8	77 DISP	4 or 16**				å				[PC] ← [PC] + DISP8
				_			_				Branch if C or Z is 0

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

	Operation Performed		[PC] ← [PC] + DISP8	Branch if C is 0	[PC] ← [PC] + DISP8	Branch It C is 1	FCJ ← FCJ + DISPS	[PC] ← [PC] + DISP8	Branch 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	Branch if P is O	[PC] ← [PC] + DISP8	Branch if S is 0	See JNE	[PC] ← [PC] + DISP8	Branch if 0 is 1	[PC] ← [PC] + DISP8	Branch if P is 1	See JP
		P C									_																												$\exists$
, ا	<u>,</u>	Z A					_																							-									$\dashv$
Statuege		S																							_								_						
Ü	\$ <b> </b>	_														_																			_				_
	ł	<u> </u>				·																																	ㅓ
L	1	0							•																	_													
	Clock Cycles		4 or 16**		4 or 16**		0 0	6 or 18**		4 or 16**		4 or 16**	7 7 16**	5	4 or 16**		4 or 16**						4 or 16**						4 or 16**	4 or 16**	5	4 or 16**	-		4 or 16**		4 or 16**		
	Object Code		73 DISP		72 DISP	000	AGIO OV	E3 DISP		74 DISP		7F DISP	מאוט טע	5	7C DISP		7E DISP						75 DISP						71 DISP	78 DISP		79 DISP			70 DISP		7A DISP		
	Operand(s)		DISP8	1	DISP8	0	DISPO	DISP8		DISP8	(	DISP8	8dolc	) 5	DISP8		DISP8		DISP8	DISP8	DISP8	DISP8	DISP8		DISP8	DISP8	DISP8	disp8	DISP8	S d S i C	)	DISP8		DISP8	DISP8		DISP8		DISP8
	Mnemonic		JAE	!	<b>8</b>	Š	םם	JCXZ		끡	!	മ	ij	5	ᆿ		JLE		ANS	JNAE	SNB B	JNBE	NS.		SNG	JNGE	<b>ヺ</b>	inle	ON N	dN	;	SNC		JNZ	9		٩	!	JPE
۲	<del>≥</del>	_						•							(P	enu	itin	၀၁	) u	oiti	pu	စၥ	uO	ųo	u.B.	В	•							-			<del></del>		ヿ

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

€							St	Statuses	88			
λbe	Mnemonic	Operand(s)	Object Code	Clock Cycles		ł	;  -		:	}	-	Operation Performed
T				•	0	_	-	S	Z	4	ЬС	
(.11	OAC	DISP8				$\vdash$	_		$\vdash$		_	See JNP
noo	SC	DISP8	78DISP	4 or 16**		_						[PC] ← [PC] + DISP8
) (												Branch if S is 1
вос	Zſ	DISP8										See JE
	MOV	RBD,RBS	8A11dddsss	2.		-	<u> </u>		T	$\vdash$	-	[RBD] ← [RBS]
												Move the contents of any RB register to any RB register
9	MOV	RWD,RWS	8B 11dddsss	5.		_						[RWD] ← [RWS]
۸Ο												Move the contents of any RW register to any RW register
M 1	MOV	SR,RW	8E 110rrsss	2*								[SR] ← [RWS]
911												Move the contents of any RW register to any Segment register
eige	MOV	RW,SR	8C 110rrddd	2*							•	[RWD] ← [SR]
ЭЯ						_						Move the contents of any Segment register to any RW register
J	XCHG	AX,RW	10010reg	÷		-						[AX] ← → [RW]
əjs												Exchange the contents of AX and any RW register
ige	XCHG	RB,RB	86 11regreg	•		_						[RB] ↑ ↓ [RB]
H									_			Exchange the contents of any two RB registers
	XCHG	RW,RW	87 11regreg	4		_						[RW] ← → [RW]
												Exchange the contents of any two RW registers
	CMPS	BD,BS	A6	22	×	9	L	×	×	÷	×	[[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	Α7	22	×	9		×	×	×	×	$[[S1]] - [[D1]], [S1] \leftarrow [S1] \pm 2, [D1] \leftarrow [D1] \pm 2$
												Compare the 16-bit data words addressed by the SI and DI Index registers
-												using string data addressing*
rch	SGOT	BD,BS	AC	12	_	₽						[AL] ← [[SI]], [SI] ← [SI] ± 1
895					_							Move a data byte from the location addressed by the SI Index register to the
S P	(							_	-	_		AL register using string data addressing
16	SGOT	SW'QW	AD	12		6						[AX] ← [[SI]], [SI] ← [SI] ± 1
ıel										·		Move a data word from the 16-bit location addressed by the SI Index register
sui		•							-			to the AX register using string data addressing
BıT	MOVS	BD,BS	<b>A4</b>	18	_	Q						$\{[D]\} \leftarrow [S]\}, \{S] \leftarrow [S] \pm 1, [D] \rightarrow [D] \pm 1$
ck					_			_			_	Move a data byte from the location addressed by the SI Index register to the
oli												extra segment location addressed by the DI register using string data address-
8											_	•Bui
	MOVS	WD,WS	A5	28	<u></u>	Ó				-		$[[D1]] \leftarrow [[S1]], [S1] \leftarrow [S1] \pm 2, [D1] \leftarrow [D1] \pm 2$
												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
]					1	4	4	1	1	4	4	Overrigen.

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

əd		Contractor	Object C	عواصل عودان			Sta	Statuses	ي	l	}	
ΥT	Minemonic		Object Code	CIOCK CYCIES	0	1	T	S	ZA	4	c	Operation refrormed
	REP	Z	11110012	+2 per loop		Q/		_	├	_		Repeat the next sequential instruction (which must be a Block Transfer and
(pen												Search instruction) until CX contents decrements to U. Decrement CX contents on each repeat. If the next instruction is CMPB, CMPW, SCAB, or
nitn												SCAW then repeat until CX contents decrements to 0 or 2 status does not
o ()	SCAS	BD,BS	AE	15	×	Q/ =		×	÷	×	×	≤
ııcp												Compare AL register contents with the extra segment data byte addressed by
808	SAS	SW GW	Δ	<u>,</u>	×	Ş		×		×	×	the DI Index register using string data addressing $[Ax] = [Du] = [Du] + 2$
pue	?			) :		<u> </u>		_				
10						_		_				dressed by the DI Index register using string data addressing
ļsu	STOS	BD,BS	ΑΑ	11	×	<u>Q</u>		×	÷	×	×	트
81 T												Store the AL register contents in the extra segment data memory byte ad-
оск	STOS	SW CW	8V	-	>	5		>	<del>-</del>	×	×	dressed by the DI Index register using string data addressing $I[D_1] \leftarrow [\Delta X] [D_1] \leftarrow [D_1] + 2$
18	200	cw, w	?	_		2		_				
												addressed by the DI Index register using string data addressing
	ADC	RBD,RBS	12 11dddsss	÷e	×	-		×	×	×	×	뜨
		!		i								
	ADC	HWD,RWS	13 11dddsss	m ·	×			×	<u>×</u> ×	×	<u>×</u>	<u>"</u>
		-										Add the 16-bit contents of register RWS, plus the Carry status, to register RWD.
	ADD	RBD,RBS	02 11dddsss	*n	×			×	×	×	×	<u>«</u>
6												Add the 8-bit contents of register RBS to register RBD
9161	ADD	RWD,RWS	03 11dddsss	÷e	×			×	×	×	×	뜨
edO	•		7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7	ċ	•	,						Add the 16-bit contents of register RWS to register RWD
16	AND	45U,465	ssspapii 77	מ	>			<u>`</u>	<u> </u>	<u>&lt;</u>	<u> </u>	<u>ב</u>
tsig	AND	RWD,RWS	23 11dddsss	ň	0			×	×	<u>~</u>	Ö	AND the 8-bit contents of register RBS with register RBD [RWD] ← [RWD] AND [RWS]
өЯ												
- JE	CBW		86	2.			_	_		_		[AH] ← [AL7]
otei												
ßeį	CMP	RBD,RBS	3A 11dddsss	, m	×			×	$\frac{2}{x}$	<u>×</u>	×	뜨
1												Subtract the contents of register RBD from register RBS. Discard the result,
	CMP	RWD.RWS	3B 11dddsss	ň	×			×		×	×	but adjust status flags [RWD] – [RWS]
				)	:				_			
												but adjust status flags
	CWD		66	တ		_						[DX] ← [AX15]
						$\dashv$		ᅱ	ᅱ	긕		Extend AX sign bit into DX

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

əd				-1-10			Sta	Statuses	80				
γŢ		Operanuts/	epon peron	CIOCK CYCIES	0	1 a	Ţ	S	z	٧	Ь		Operation Performed
	ΛIQ	RBS	F6 11110sss	06-08	n	<u> </u>	L	2	n	n	U U	$[AX] \leftarrow [AX]/[RBS]$	(RBS)
	)id	SWS	F7 11110sss	144-162	=				=	=		Divide the 1 teger quoti€ FF 16, exect	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 <sub>16</sub> , execute a "divide by O" interrupt IDX1 AX1 — IDX1 AX1/IDX3
	·				·			)	<del></del>			Divide the 3 the 16-bit of mainder in E	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 execute a "divide by O".
	NOI	RBS	F6 11111sss	101-112	Þ	<del></del>		D	)	>	n n	ַ ≤	(RBS)
			·			<del></del>						Divide the ing both co	Divide the 16-bit contents of register AX by the 8-bit contents of MBS, 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
(beunitr	NQI	RWS	F7 111115ss	165-184	<u> </u>			)	ح	n n	5	AX. Store the is greater the [DX] [AX] ← [	AX. Store the remainder, as an unsigned binary number, in AH. If the quotient is greater than 7F16, or less than −8016, execute a "divide by 0" interrupt U [DX] [AX] ← [DX] [AX]/[RWS]
ioO) et											-	Divide the 3 the 16-bit c	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.
SteqO						<u></u>						Store the quantum san unsigned	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
16) sig	<b>IM</b> OL	RBS	F6 11101sss	86-08	×			)	)	<u> </u>	×	[AX] ← [AL] • [RBS] Multiply the 8-bit	less than -ocood,6, execute a laivide by o interrupt X] ← [AL] • [RBS] Multinly the 8-bit contents of register All by the contents of RBS. Treat both
er – Re	IMUL	RWS	F7 11101sss	128-154	×	•		ے			×	numbers as [DX] [AX] ← [	numbers as signed binary numbers. Store the 16-bit product in AX [DX] [AX] — [AX] • [RWS]
tsigeA		o a	110000		>				,=		>	Multiply the 16-bi Treat both number (high-order word)	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	RWS	F7 11100sss	118-133	< ×			) )	) >	) )		Multiply the numbers as [DX] [AX]	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 [DX] [AX] — [AX] • [RWS]
	8	RBD.RBS	0A 11dddsss		0			×	×		×	Multiply the 16-bit con Treat both numbers as DX (high-order word) a [RBD] ← [RBD] OR [RBS]	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) BDI ← [RBD] OR [RBS]
	8	RWD.RWS	OB 11dddss	m	0			×	_			OR the 8-bi	OR the 8-bit contents of register RBS with register RBD IRWDI ← IRWDI OR IRWS
							* -	v.				OR the 16-	OR the 16-bit contents of register RWS with register RWD
						ĺ				١	١		

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

SBB   RBD,RBS   1A 11dddsss   3°   X   X   X   X   X   X   X   X   X	ed.	Managaria	Onerend(e)	Object Code	Clock Cycles			St	Statuses	108				Onesetine Basharman
SBB         RBD/RBS         1A 11dddsss         3°         X	٧T									_				
SUB         RWD,RWS         1B 11dddsss         3°         X         X         R		SBB	RBD,RBS	1A 11dddsss	÷6	×	-	L_	×			_		- [RBS]
SUB         RBD,RBS         2A 11dddsss         3°         X													Su	Subtract the 8-bit contents of register RBS from RBD using twos complement arithmetic. If the Carry status was originally 1 decrement the result
SUB         RBD,RBS         2A 11dddsss         3°         X	(	SBB	RWD,RWS	1B 11dddsss	'n	×			×				<u>E</u>	$\langle D \rangle \leftarrow [RWD] - [RWS] - [C]$
SUB         RBD.RBS         2A 11dddsss         3°         X	pər											·	ß	Subtract the 16-bit contents of register RWS from RWD using twos comple-
SUB         RBD,RBS         2A 11dddsss         3°         X	nit	!	1		:									ment arithmetic. If the Carry status was originally 1 decrement the result
SUB         RWD,RWS         2B 11dddsss         3°         X	uoc	SUB	RBD,RBS	2A 11dddsss	*n	×			×				٣	D] ← [RBD] – [RBS]
SUB         RWD,RWS         2B 11dddsss         3°         X	)) <del>a</del>										_		Su.	Subtract the 8-bit contents of register RBS from RBD using twos complement
TEST RBD,RBS	<b>JB1</b>	Z.	SWD RWS	2B 11dddss	*	×			>					rithmetic   total [BM/D] - [BM/S]
TEST RWD,RWS   85 11regreg   3° 0   X   X   U   X   X   U   X   X   U   X   X	∍dC	}			<b>)</b>	:			<			_		Subtract the 16-bit contents of register RWS from RWD using twos comple-
TEST RBD,RBS 84 11regreg 3° 0	19											_		ent arithmetic
TEST RWD,RWS   85 11 regreg   3° 0   X X U X O   FR	tsię	TEST	RBD,RBS	84 11 regreg	÷n	0			×				뜨	D) AND [RBS]
TEST   RWD,RWS   85 11 regreg   3°   0	Reg											-	¥	AND the 8-bit contents of register d and register RBS. Discard the result, but
TEST         RWD,RWS         85 11regreg         3°         0         X X U X O (R)           XOR         RBD,RBS         30 11dddsss         3°         0         X X U X O (R)           AAA         37         4°         U         U U X U X O (R)           AAB         D5 0A         60         U         X X U X U X O (R)           AAS         3F         4°         U         U U X U X O (R)           DAS         27         4°         U         X X X X X X X X X X X X X X X X X X X	ار –													djust status flags
XOR         RBD,RBS         30 11dddsss         3°         0         X         X         U         X         U         X         X         U         X         X         U         X	<del>0</del> 18	TEST	RWD,RWS	85 11 regreg	*n	0			×				<u>Œ</u>	/D] AND [RWS]
XOR         RBD,RBS         30 11dddsss         3*         0         X         X         U         X         U         X         U         X         Y         U         X         Y         U         X         X         U         X         X         U         X         X         U         X         X         U         X	ige												¥	AND the 16-bit contents of register RWD and register RWS. Discard the
XOR         RBD,RBS         30 11dddsss         3*         0         X X U X U X U X U X U X U X U X U X U X	B									_				sult, but adjust status flags
XOR         RWD,RWS         31 11dddsss         3*         0         X         X         U         X         U         X         U         X         U         X         U         X         U         X         U         X         U         X         U         X         U         X         X         U         X         X         U         X         U         X         U         X         U         X		XOR	RBD,RBS	30 11dddsss	ť	0			×				严	D] ← [RBD] XOR [RBS]
AAA         AAA         37         4*         0         X X U X U         RW D X U         X X U X U         RW U X U X U         RW U X U X U         RW U U U X U X U         RW X X X X X X X X X X X X X X X X X X X														Exclusive OR the 8-bit contents of register RBS with register RBD
AAA         37         4*         U         U v v v           AAM         D5 0A         60         U         x x v v v           AAS         3F         4*         U         x x v v v           DAA         27         4*         U         x x x x x x x           DEC         RW         01001ddd         3*         x x x x x x x x           FEC         RW         01001ddd         2*         x x x x x x x x		XOR	RWD,RWS	31 11dddsss	÷n	0			×				Œ	/D] ← [RWD] XOR [RWS]
AAD         D5 OA         60         U         X         U         X           AAM         D4 OA         83         U         X         U         X         U           AAS         3F         4*         U         U         X         X         U           DAA         27         4*         U         X         X         X         X           DEC         RB         FE 11001ddd         3*         X         X         X         X           DEC         RW         01001ddd         2*         X         X         X         X								_					Ë	Exclusive OR the 16-bit contents of register RWS with register RWD
AAM AAS BAA BAA BAA BAA BAA BAA BAA BAA BAA		AAA		37	4.	n			n		_	_		ASCII adjust AI register contents for addition (as described in accompanying
AAM AAS  AAS  3F  4**  U  U  X  X  U  X  Y  U  U  X  X  U  X  X  X  U  X  X  X  X													tex	xt)
AAS  AAS  3F  4*  U  U  V  V  V  V  DAA  DAS  27  4*  U  V  V  V  V  V  V  N  N  N  N  N  N  N		AAD		D5 0A	09	<u> </u>			×					Decimal adjust dividend in AL prior to dividing an unpacked decimal divisor,
AAS  AAS  3F  4** U  U  U  V  X  X  X  X  X  X  X  X  X  X  X  X							-				-		\$	to generate an unpacked decimal quotient. (See accompanying text for
AAS  AAS  3F  4*  U  U  V  X  X  V  X  X  V  DAA  DAS  27  4*  U  X  X  X  X  X  X  X  X  X  X  X  X										_				etails)
AAS         3F         4*         U         U         X <th>ə</th> <td>AAM</td> <td></td> <td>D4 0A</td> <td>83</td> <td><b>&gt;</b></td> <td></td> <td></td> <td>×</td> <td></td> <td></td> <td></td> <td></td> <td>After multiplying o unpacked decimal operands, adjust product in AX to</td>	ə	AAM		D4 0A	83	<b>&gt;</b>			×					After multiplying o unpacked decimal operands, adjust product in AX to
DAA 27 4* U U V X U X X X X X X X X X X X X X X X	JB1													become an unpacked decimal result. (See accompanying text for details)
DAA         27         4*         U         X X X X X X X X X X X X X X X X X X X	ed(	AAS		3F	•	_			)					After subtracting two unpacked decimal numbers, adjust the difference in AL
DAA         27         4*         U         X <th>) L</th> <td></td> <td>So</td> <td>so that it too is an unpacked decimal number. (See accompanying text for</td>	) L												So	so that it too is an unpacked decimal number. (See accompanying text for
DAS  27  4*  U  X  X  X  X  X  X  X  X  X  X  X  X	918					_	_			_		_		stails)
DAS         2F         4*         U         X         X         X         X         X         X         X         X         X         X         X         X         X         IR           DEC         RW         01001ddd         2*         X         X         X         X         IR	ige	DAA		27	<b>.</b>	>			×				_	After adding two packed decimal numbers, adjust the sum in AL so that it too
RB FE 11001ddd 3° × × × × × × × × × × × × × × × × × ×	H													is a packed decimal number. (See accompanying text for details)
RB FE 11001ddd 3° X X X X X (R X X X X X X X X X X X X X		DAS		2F	•4	>			×					After subtracting two packed decimal numbers, adjust the difference in AL so
RW 01001ddd 2° X X X X X (R		Ç	0	70000	č	>			>				than 1	that it too is a packed decimal number. (See accompanying text for details)
RW 01001ddd 2° X X X (R		2	2	200	,	<			<			<del>-</del>	<u></u>	- [nb] - I
		DEC	×8	01001ddd	2.	×			×			×	<u> </u>	defined the o-bit contents of register no
	_	1			ı				<			_	٥	Decrement the 16-bit contents of register RW

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

NC	[						ĺ	Ü		8	1		┝	
NC	dλ	Mnemonic		Object Code	Clock Cycles	╁	}	;		:	ŀ	ŀ	Т	Operation Performed
NC   RB   FE 11000ddd   3°   X   X   X   X   X   X   X   X   X	T							느	S	Z		_	ᇹ	
NEG   RW   01000ddd   2°		INC	æ	FE 11000ddd	3*	×	$\vdash$	<u> </u>	×	-	_	×	٣	RB] ← [RB] +1
NEG   RW   F6 11011ddd   3°   X   X   X   X   X   X   X   X   X														Increment the 8-bit contents of register RB
NEG         RB         F6 11011ddd         3°         X		S N	≩	01000ddd	2.	×			×	×		×	<u>=</u>	$[RW] \leftarrow [RW] + 1$
NOT   RB		NEG	88	F6 11011ddd	÷n	×			×	×				nicializer. In the lo-bit contents of register niver. RB] ← [RB] +1
NOT   RW   F7   11010ddd   3°   Nos complement the 16-bit contents of register		N G	8	F7 11011ddd	**	×			×	×				Twos complement the 8-bit contents of register RB RWI ← RWI + 1
NOT   RW   F7 11010ddd   3°   IRW)		!			)				_	:				Twos complement the 16-bit contents of register RW
NOT	(F	NOT	82	F6 11010ddd	÷e									RB] ← [RB]
RCL	oənui	NOT	ЯW	F7 11010ddd	÷						<del></del>			Ones complement the 8-bit contents of register RB RWJ $\leftarrow$ [RW]
RCR   RBN   110100v0   11010ddd   X   X   Rotate left through Carry the 8-bit contents of Promers of RWN   110100v1   11010ddd   X   X   Rotate right through Carry the 8-bit contents of Promers of RWN   110100v1   11010ddd   N = 1.2*   X   Rotate right through Carry the 8-bit contents of Promers of RW register, as illustrated for memory operate RWN   110100v1   11010ddd   N = 1.2*   X   X   X   X   X   X   X   X   X	uo										_			Ones complement the 16-bit contents of register RW
RCB	၁)	RCL	RB,N	110100v0 11010ddd	-	×							×	Rotate left through Carry the 8-bit contents of RB register, or the 16-bit
RCH   RBN   110100v0 11011ddd   N=12	<b>9</b> 16	RCL	RW,N	110100v1 11010ddd		×							×:	contents of RW register, as illustrated for memory operate
ROL	190	RCR	RBN	110100v0 11011ddd		×			_	_			×	Rotate right through Carry the 8-bit contents of RB register, or the 16-bit
RON	i0	RCR	X X	110100v1 11011ddd		×	_						×	contents of RW register, as illustrated for memory operate
RW,	16)	<b>PO</b>	RB,N	110100v0 11000ddd		×				_			×	Rotate left the 8-bit contents of RB register, or the 16-bit contents of RW
RBN	sig	ROL	RW,N	110100v1 11000ddd	N > 1 4N+8	×						<u> </u>	×	register as illustrated for memory operate
RBW,   110100v1 1100ddd	Вея	ROR	RB,N	110100v0 11001ddd		×						_	×	Rotate right the 8-bit contents of RB register, or the 16-bit contents of RW
SAL         RB,N         110100v0 11100ddd         X		ROR	RW.N	110100v1 11001ddd		×					_		×	register, as illustrated for memory operate
SAL         RW,N         110100v1 11100ddd         X         X         U         X         X         D         X         Cegister, as illustrated for memory operate or register RB, or thing the B-bit contents of the data memory word addressed using Status Flags register           POPP         RW         01011 ddd         R         X <td></td> <td>SAL</td> <td>RB,N</td> <td>110100v0 11100ddd</td> <td></td> <td>×</td> <td></td> <td></td> <td>×</td> <td>×</td> <td><u></u></td> <td></td> <td>×</td> <td>Shift left the 8-bit contents of RB register, or the 16-bit contents of RW</td>		SAL	RB,N	110100v0 11100ddd		×			×	×	<u></u>		×	Shift left the 8-bit contents of RB register, or the 16-bit contents of RW
SAR         RB,N         110100v0 11111ddd         X		SAL	RW,N	110100v1 11100dda		×			×	×	<u> </u>		×	register, as illustrated for memory operate
SAR         RW,N         110100v1 11111ddd         X		SAR	RB,N	110100v0 11111ddd		×			×				×	Shift right the 8-bit contents of register RB, or the 16-bit contents of register
SHL         RB,N         X         X         X         X         X         X         See SAL           SHR         RB,N         110100v0 11101ddd         N=1 2*         X		SAR	RW,N	110100v1 11111ddd		×			×			. — -	×	RW, as illustrated for memory operate
SHL         RW,N         110100v0 11101 ddd         N=1 2°         X		돐	RB,N			×			×		<u> </u>		×	See SAL
SHR         RB,N         110100v0 11101ddd         N=1 2*         X X X U X X         X X RW, as illustrated for memory operate           POP         DADDR         8F aa000bbb         17+EA           RX X U X X X X X   U X X   RW, as illustrated for memory operate           POP         BRW 01011ddd         17+EA           RW, as illustrated for memory operate           POP         RW 01011ddd         8           Load the 16-bit Stack word, addressed by DADDR. Inc           POP         SR 000r111         8           X X X X X X X X X X X X X X X X X X X		SHL	RW,N			×			×		<u>.                                    </u>		×	See SAL
SHR         RW,N         110100v1 11101dda         N > 1 4N+8         X X U X X         X X U X X         X X U X X         X X U X X         X X U X X         X X U X X         X X U X X         X X U X X         X X U X X         X X U X X         X X U X X         X X X X X X         X X X X X X X X X X X X X X X X X X X		SH	RB'N	110100v0 11101ddd	N=1 2.	×			×				×	Shift right the 8-bit contents of register RB, or the 16-bit contents of register
POP         DADDR         8F aa000bbb         17+EA         [EA] — [[SP]], [SP] — [SP] + 2           Load the 16-bit Stack word, addressed using State         Load the 16-bit Stack word, addressed using specified 16-bit Stack word, addressed using Status Flags register           POPF         8         X X X X X X X X X X X X X X X X X X X		SHR	RW,N	110100v1 11101dda	N>1 4N+8		$\dashv$	_	×	×	_	_	×	RW, as illustrated for memory operate
POP         RW         01011ddd         8         Load the 16-bit Stack word, addressed using State Memory word addressed by DADDR. Inc. [RW or SR] ← [SP], [SP] + 2           POP         SR         000r111         8         X X X X X X X X X X X X X X X X X X X		POP	DADDR	8F aa000bbb	17+EA		-						=	EA] ← [(SP]], (SP] ← [SP] + 2
POP         RW         01011ddd         8         IRW or SR] ← [SP], [SP] + 2           POP         SR         000r111         8         X X X X X X X X X X X X X X X X X X X				[DISP][DISP]							_			
POP   SR   O00r111   8   Load the 16-bit Stack word, addressed using specified 16-bit register. Increment SP by 2.		٥	/YG	7777	•	_					_			bit data memory word addressed by DADDR, increment SP by 2
POPF         9D         8         X X X X X X X X X X X X X X X X X X X		5 6	8 g	000	o 0								<b>3</b> , 1	700
POPF         9D         8         X X X X X X X X X X X X X X X X X X X	ck	5	<u>.</u>		·									S 2
PUSH DADDR FF aa110bbb 16+EA [DiSP][DiSP]	st S	POPF		<b>О</b> 6	8							<u> </u>		SFR] ← [[SP]], [SP] ← [SP] + 2
DADDR FF aa110bbb 16+EA [S													:	Load the 16-bit Stack word, addressed using Stack addressing, into the
		H	A CO A CO	EF so 110bbb	1816							-		Status Flags register SPI ← [SP] – 2 f(SPI) ← [EA]
the 16-bit Stack word addressed using Stack addressing. Decr				[DISP][DISP]	:									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)

əd		(a)passag(a)	Object Code	موادين بإددان			Š	Statuses	808			Oneretine Berformed
٧T				CIOCK CYCIES	0	-	ı T	S	z	A	P C	
(.tnc	PUSH PUSH	RW SR	01010rr 000rr110	11								[SP] ← [SP] –2, [[SP]] ← [RW or SR] Store the contents of the specified 16-bit register in the 16-bit Stack word
k (C	PUSHF	<b>)</b> 6	10				·					addressed using Stack addressing. Decrement SP by 2 [SP] ← [SP] +2, [[SP]] ← [SFR]
Stac												Store the Status flags register contents in the 16-bit Stack word addressed using Stack addressing. Decrement SP by 2
s	TNI	8	33	52		٦				H	-	Execute a software interrupt and vector through table entry 3
ıdn.	F	>	Ç √	51		0 0	0 0					Execute a software interrupt and vector through table entry V
n <del>o</del> jr	2 <u>Z</u>			4 or 53								If the U status is 1, execute a software interrupt and vector through table entry 1016
4	IRET		CF	24								Return from interrupt service routine
	CLC		F8	2.		$\vdash$	_			<del>                                     </del>	0	으
	2		Ç.	,								Clear Carry status
	3		2	1		<del></del>						Clear Decrement/Increment select
	CLI		FA	2*		0						
	Ç		ŀ	ŧ								
	) N		ភ	7					_		Κ.	<u>ء</u>
	LAHF		96	<b>*</b>								Complement Carry status Transfer flags to AH register as follows:
												7 6 5 4 3 2 1 0 Bit no.
												AH register
snı												S Z O A O P I C
812	SAHF		9E	•4				×	×	$\frac{}{\times}$	×	Transfer AH register contents to status
												7 6 5 4 3 2 1 0 Bit no.
			`									S 7 A P C
						-						-
-	STC		6	2.							-	[C] ← 1 Sat Carry etatus to 1
	STD		6	2.						.,		[D] ← 1
	STI		82	2*						-		Set Decrement/Increment status to 1 [I] $\leftarrow$ 1
												Set interrupt enable status to 1, enabling all interrupts
						$\dashv$			一		$\dashv$	

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

Code Clock Cycles
Clock Cycles  Clock Cycles  2. 2. 4.2 4.2 3.5 3.5 3.6 5 tatuses  Statuses  3 + 5 n  3 + 5 n  3 + 5 n
Clock Cycles
Clock Cycles 0 D 1 3+EA 3+5n 3-5n
Clock Cycles 0 D 1 3+EA 3+5n 3-5n
Clock Cycles 0 D 3+EA 3+5n 3*
Clock Cycles - 2 - 2 - 2 - 3 - 3 - 5 n
Clock Cycle 8+EA 3+5n 3.
xxxx xxxx (DISP] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Object Code 11011xxx aaxxxbb [DISP][DISP] F4 F0 001reg110
Mnemonic Operand(s)  ESC DADDR  LOCK  SEG SR  WAIT  NOP
Mnemonic ESC HLT LOCK SEG NOP
Other Type