Table 2-21. Instruction Set Reference Data

AAA	**AAA** (no operands) ASCII adjust for addition			**Flags** O D I T S Z A P C U U U X U X
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
(no operands)	4	—	1	AAA

AAD	**AAD** (no operands) ASCII adjust for division			**Flags** O D I T S Z A P C U X X U X U
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
(no operands)	60	—	2	AAD

AAM	**AAM** (no operands) ASCII adjust for multiply			**Flags** O D I T S Z A P C U X X U X U
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
(no operands)	83	—	1	AAM

AAS	**AAS** (no operands) ASCII adjust for subtraction			**Flags** O D I T S Z A P C U U U X U X
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
(no operands)	4	—	1	AAS

*For the 8086, add four clocks for each 16-bit word transfer with an odd address. For the 8088, add four clocks for each 16-bit word transfer.

Table 2-21. Instruction Set Reference Data (Cont'd.)

ADC	**ADC** destination,source Add with carry			**Flags** O D I T S Z A P C X X X X X X
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
register, register	3	—	2	ADC AX, SI
register, memory	9+EA	1	2-4	ADC DX, BETA [SI]
memory, register	16+EA	2	2-4	ADC ALPHA [BX] [SI], DI
register, immediate	4	—	3-4	ADC BX, 256
memory, immediate	17+EA	2	3-6	ADC GAMMA, 30H
accumulator, immediate	4	—	2-3	ADC AL, 5

ADD	**ADD** destination,source Addition			**Flags** O D I T S Z A P C X X X X X X
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
register, register	3	—	2	ADD CX, DX
register, memory	9+EA	1	2-4	ADD DI, [BX].ALPHA
memory, register	16+EA	2	2-4	ADD TEMP, CL
register, immediate	4	—	3-4	ADD CL, 2
memory, immediate	17+EA	2	3-6	ADD ALPHA, 2
accumulator, immediate	4	—	2-3	ADD AX, 200

AND	**AND** destination,source Logical and			**Flags** O D I T S Z A P C 0 X X U X 0
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
register, register	3	—	2	AND AL,BL
register, memory	9+EA	1	2-4	AND CX,FLAG_WORD
memory, register	16+EA	2	2-4	AND ASCII [DI],AL
register, immediate	4	—	3-4	AND CX,0F0H
memory, immediate	17+EA	2	3-6	AND BETA, 01H
accumulator, immediate	4	—	2-3	AND AX, 01010000B

CALL	**CALL** target Call a procedure			**Flags** O D I T S Z A P C
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Examples**
near-proc	19	1	3	CALL NEAR_PROC
far-proc	28	2	5	CALL FAR_PROC
memptr 16	21+EA	2	2-4	CALL PROC_TABLE [SI]
regptr 16	16	1	2	CALL AX
memptr 32	37+EA	4	2-4	CALL [BX].TASK [SI]

CBW	**CBW** (no operands) Convert byte to word			**Flags** O D I T S Z A P C
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
(no operands)	2	—	1	CBW

*For the 8086, add four clocks for each 16-bit word transfer with an odd address. For the 8088, add four clocks for each 16-bit word transfer.

Table 2-21. Instruction Set Reference Data (Cont'd.)

CLC	CLC (no operands) Clear carry flag			Flags O D I T S Z A P C 0
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
(no operands)	2	—	1	CLC

CLD	CLD (no operands) Clear direction flag			Flags O D I T S Z A P C 0
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
(no operands)	2	—	1	CLD

CLI	CLI (no operands) Clear interrupt flag			Flags O D I T S Z A P C 0
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
(no operands)	2	—	1	CLI

CMC	CMC (no operands) Complement carry flag			Flags O D I T S Z A P C X
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
(no operands)	2	—	1	CMC

CMP	CMP destination,source Compare destination to source			Flags O D I T S Z A P C X X X X X X
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
register, register	3	—	2	CMP BX, CX
register, memory	9+EA	1	2-4	CMP DH, ALPHA
memory, register	9+EA	1	2-4	CMP [BP+2], SI
register, immediate	4	—	3-4	CMP BL, 02H
memory, immediate	10+EA	1	3-6	CMP [BX].RADAR [DI], 3420H
accumulator, immediate	4	—	2-3	CMP AL, 00010000B

CMPS	CMPS dest-string,source-string Compare string			Flags O D I T S Z A P C X X X X X X
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
dest-string, source-string	22	2	1	CMPS BUFF1, BUFF2
(repeat) dest-string, source-string	9+22/rep	2/rep	1	REPE CMPS ID, KEY

*For the 8086, add four clocks for each 16-bit word transfer with an odd address. For the 8088, add four clocks for each 16-bit word transfer.

Table 2-21. Instruction Set Reference Data (Cont'd.)

CWD	CWD (no operands) Convert word to doubleword			Flags O D I T S Z A P C
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
(no operands)	5	—	1	CWD

DAA	DAA (no operands) Decimal adjust for addition			Flags O D I T S Z A P C X X X X X X
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
(no operands)	4	—	1	DAA

DAS	DAS (no operands) Decimal adjust for subtraction			Flags O D I T S Z A P C U X X X X X
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
(no operands)	4	—	1	DAS

DEC	DEC destination Decrement by 1			Flags O D I T S Z A P C X X X X X
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
reg16	2	—	1	DEC AX
reg8	3	—	2	DEC AL
memory	15+EA	2	2-4	DEC ARRAY [SI]

DIV	DIV source Division, unsigned			Flags O D I T S Z A P C U U U U U U
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
reg8	80-90	—	2	DIV CL
reg16	144-162	—	2	DIV BX
mem8	(86-96) +EA	1	2-4	DIV ALPHA
mem16	(150-168) +EA	1	2-4	DIV TABLE [SI]

ESC	ESC external-opcode,source Escape			Flags O D I T S Z A P C
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
immediate, memory	8+EA	1	2-4	ESC 6,ARRAY [SI]
immediate, register	2	—	2	ESC 20,AL

*For the 8086, add four clocks for each 16-bit word transfer with an odd address. For the 8088, add four clocks for each 16-bit word transfer.

Table 2-21. Instruction Set Reference Data (Cont'd.)

HLT	HLT (no operands) Halt			Flags O D I T S Z A P C
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
(no operands)	2	—	1	HLT

IDIV	IDIV source Integer division			Flags O D I T S Z A P C U U U U U U
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
reg8	101-112	—	2	IDIV BL
reg16	165-184	—	2	IDIV CX
mem8	(107-118) +EA	1	2-4	IDIV DIVISOR_BYTE [SI]
mem16	(171-190) +EA	1	2-4	IDIV [BX].DIVISOR_WORD

IMUL	IMUL source Integer multiplication			Flags O D I T S Z A P C X U U U U X
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
reg8	80-98	—	2	IMUL CL
reg16	128-154	—	2	IMUL BX
mem8	(86-104) +EA	1	2-4	IMUL RATE_BYTE
mem16	(134-160) +EA	1	2-4	IMUL RATE_WORD [BP] [DI]

IN	IN accumulator,port Input byte or word			Flags O D I T S Z A P C
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
accumulator, immed8	10	1	2	IN AL, 0FFEAH
accumulator, DX	8	1	1	IN AX, DX

INC	INC destination Increment by 1			Flags O D I T S Z A P C X X X X X
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
reg16	2	—	1	INC CX
reg8	3	—	2	INC BL
memory	15+EA	2	2-4	INC ALPHA [DI] [BX]

*For the 8086, add four clocks for each 16-bit word transfer with an odd address. For the 8088, add four clocks for each 16-bit word transfer.

Table 2-21. Instruction Set Reference Data (Cont'd.)

INT	**INT** interrupt-type Interrupt			**Flags** O D I T S Z A P C 0 0
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
immed8 (type = 3) immed8 (type ≠ 3)	52 51	5 5	1 2	INT 3 INT 67

INTR†	**INTR** (external maskable interrupt) Interrupt if INTR and IF=1			**Flags** O D I T S Z A P C 0 0
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
(no operands)	61	7	N/A	N/A

INTO	**INTO** (no operands) Interrupt if overflow			**Flags** O D I T S Z A P C 0 0
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
(no operands)	53 or 4	5	1	INTO

IRET	**IRET** (no operands) Interrupt Return			**Flags** O D I T S Z A P C R R R R R R R R
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
(no operands)	24	3	1	IRET

JA/JNBE	**JA/JNBE** short-label Jump if above/Jump if not below nor equal			**Flags** O D I T S Z A P C
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
short-label	16 or 4	—	2	JA ABOVE

JAE/JNB	**JAE/JNB** short-label Jump if above or equal/Jump if not below			**Flags** O D I T S Z A P C
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
short-label	16 or 4	—	2	JAE ABOVE_EQUAL

JB/JNAE	**JB/JNAE** short-label Jump if below/Jump if not above nor equal			**Flags** O D I T S Z A P C
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
short-label	16 or 4	—	2	JB BELOW

*For the 8086, add four clocks for each 16-bit word transfer with an odd address. For the 8088, add four clocks for each 16-bit word transfer.

†INTR is not an instruction; it is included in table 2-21 only for timing information.

Table 2-21. Instruction Set Reference Data (Cont'd.)

JBE/JNA	**JBE/JNA** short-label Jump if below or equal/Jump if not above			**Flags** O D I T S Z A P C
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
short-label	16 or 4	—	2	JNA NOT_ABOVE

JC	**JC** short-label Jump if carry			**Flags** O D I T S Z A P C
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
short-label	16 or 4	—	2	JC CARRY_SET

JCXZ	**JCXZ** short-label Jump if CX is zero			**Flags** O D I T S Z A P C
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
short-label	18 or 6	—	2	JCXZ COUNT_DONE

JE/JZ	**JE/JZ** short-label Jump if equal/Jump if zero			**Flags** O D I T S Z A P C
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
short-label	16 or 4	—	2	JZ ZERO

JG/JNLE	**JG/JNLE** short-label Jump if greater/Jump if not less nor equal			**Flags** O D I T S Z A P C
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
short-label	16 or 4	—	2	JG GREATER

JGE/JNL	**JGE/JNL** short-label Jump if greater or equal/Jump if not less			**Flags** O D I T S Z A P C
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
short-label	16 or 4	—	2	JGE GREATER_EQUAL

JL/JNGE	**JL/JNGE** short-label Jump if less/Jump if not greater nor equal			**Flags** O D I T S Z A P C
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
short-label	16 or 4	—	2	JL LESS

*For the 8086, add four clocks for each 16-bit word transfer with an odd address. For the 8088, add four clocks for each 16-bit word transfer.

Table 2-21. Instruction Set Reference Data (Cont'd.)

JLE/JNG	JLE/JNG short-label Jump if less or equal/Jump if not greater			Flags O D I T S Z A P C
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
short-label	16 or 4	—	2	JNG NOT_GREATER

JMP	JMP target Jump			Flags O D I T S Z A P C
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
short-label	15	—	2	JMP SHORT
near-label	15	—	3	JMP WITHIN_SEGMENT
far-label	15	—	5	JMP FAR_LABEL
memptr16	18 + EA	1	2-4	JMP [BX].TARGET
regptr16	11	—	2	JMP CX
memptr32	24 + EA	2	2-4	JMP OTHER.SEG [SI]

JNC	JNC short-label Jump if not carry			Flags O D I T S Z A P C
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
short-label	16 or 4	—	2	JNC NOT_CARRY

JNE/JNZ	JNE/JNZ short-label Jump if not equal/Jump if not zero			Flags O D I T S Z A P C
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
short-label	16 or 4	—	2	JNE NOT_EQUAL

JNO	JNO short-label Jump if not overflow			Flags O D I T S Z A P C
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
short-label	16 or 4	—	2	JNO NO_OVERFLOW

JNP/JPO	JNP/JPO short-label Jump if not parity/Jump if parity odd			Flags O D I T S Z A P C
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
short-label	16 or 4	—	2	JPO ODD_PARITY

JNS	JNS short-label Jump if not sign			Flags O D I T S Z A P C
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
short-label	16 or 4	—	2	JNS POSITIVE

*For the 8086, add four clocks for each 16-bit word transfer with an odd address. For the 8088, add four clocks for each 16-bit word transfer.

Table 2-21. Instruction Set Reference Data (Cont'd.)

JO	**JO** short-label Jump if overflow			**Flags** O D I T S Z A P C
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
short-label	16 or 4	—	2	JO SIGNED_OVRFLW

JP/JPE	**JP/JPE** short-label Jump if parity/Jump if parity even			**Flags** O D I T S Z A P C
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
short-label	16 or 4	—	2	JPE EVEN_PARITY

JS	**JS** short-label Jump if sign			**Flags** O D I T S Z A P C
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
short-label	16 or 4	—	2	JS NEGATIVE

LAHF	**LAHF** (no operands) Load AH from flags			**Flags** O D I T S Z A P C
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
(no operands)	4	—	1	LAHF

LDS	**LDS** destination,source Load pointer using DS			**Flags** O D I T S Z A P C
Operands	**Clocks**	**Transfers**	**Bytes**	**Coding Example**
reg16, mem32	16+EA	2	2-4	LDS SI,DATA.SEG [DI]

LEA	**LEA** destination,source Load effective address			**Flags** O D I T S Z A P C
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
reg16, mem16	2+EA	—	2-4	LEA BX, [BP] [DI]

LES	**LES** destination,source Load pointer using ES			**Flags** O D I T S Z A P C
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
reg16, mem32	16+EA	2	2-4	LES DI, [BX].TEXT_BUFF

*For the 8086, add four clocks for each 16-bit word transfer with an odd address. For the 8088, add four clocks for each 16-bit word transfer.

Table 2-21. Instruction Set Reference Data (Cont'd.)

LOCK	**LOCK** (no operands) Lock bus			**Flags** O D I T S Z A P C
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
(no operands)	2	—	1	LOCK XCHG FLAG,AL

LODS	**LODS** source-string Load string			**Flags** O D I T S Z A P C
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
source-string (repeat) source-string	12 9+13/rep	1 1/rep	1 1	LODS CUSTOMER_NAME REP LODS NAME

LOOP	**LOOP** short-label Loop			**Flags** O D I T S Z A P C
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
short-label	17/5	—	2	LOOP AGAIN

LOOPE/LOOPZ	**LOOPE/LOOPZ** short-label Loop if equal/Loop if zero			**Flags** O D I T S Z A P C
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
short-label	18 or 6	—	2	LOOPE AGAIN

LOOPNE/LOOPNZ	**LOOPNE/LOOPNZ** short-label Loop if not equal/Loop if not zero			**Flags** O D I T S Z A P C
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
short-label	19 or 5	—	2	LOOPNE AGAIN

NMI†	**NMI** (external nonmaskable interrupt) Interrupt if NMI = 1			**Flags** O S I T S Z A P C 0 0
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
(no operands)	50	5	N/A	N/A

*For the 8086, add four clocks for each 16-bit word transfer with an odd address. For the 8088, add four clocks for each 16-bit word transfer.

†NMI is not an instruction; it is included in table 2-21 only for timing information.

Table 2-21. Instruction Set Reference Data (Cont'd.)

MOV	**MOV** destination,source Move			Flags O D I T S Z A P C
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
memory, accumulator	10	1	3	MOV ARRAY [SI], AL
accumulator, memory	10	1	3	MOV AX, TEMP_RESULT
register, register	2	—	2	MOV AX,CX
register, memory	8+EA	1	2-4	MOV BP, STACK_TOP
memory, register	9+EA	1	2-4	MOV COUNT [DI], CX
register, immediate	4	—	2-3	MOV CL, 2
memory, immediate	10+EA	1	3-6	MOV MASK [BX] [SI], 2CH
seg-reg, reg16	2	—	2	MOV ES, CX
seg-reg, mem16	8+EA	1	2-4	MOV DS, SEGMENT_BASE
reg16, seg-reg	2	—	2	MOV BP, SS
memory, seg-reg	9+EA	1	2-4	MOV [BX].SEG_SAVE, CS

MOVS	**MOVS** dest-string,source-string Move string			Flags O D I T S Z A P C
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
dest-string, source-string	18	2	1	MOVS LINE EDIT_DATA
(repeat) dest-string, source-string	9+17/rep	2/rep	1	REP MOVS SCREEN, BUFFER

MOVSB/MOVSW	**MOVSB/MOVSW** (no operands) Move string (byte/word)			Flags O D I T S Z A P C
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
(no operands)	18	2	1	MOVSB
(repeat) (no operands)	9+17/rep	2/rep	1	REP MOVSW

MUL	**MUL** source Multiplication, unsigned			Flags O D I T S Z A P C X U U U U X
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
reg8	70-77	—	2	MUL BL
reg16	118-133	—	2	MUL CX
mem8	(76-83) +EA	1	2-4	MUL MONTH [SI]
mem16	(124-139) +EA	1	2-4	MUL BAUD_RATE

*For the 8086, add four clocks for each 16-bit word transfer with an odd address. For the 8088, add four clocks for each 16-bit word transfer.

Table 2-21. Instruction Set Reference Data (Cont'd.)

NEG	NEG destination Negate			Flags O D I T S Z A P C X X X X X 1*
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
register memory	3 16+EA	— 2	2 2-4	NEG AL NEG MULTIPLIER

*0 if destination = 0

NOP	NOP (no operands) No Operation			Flags O D I T S Z A P C
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
(no operands)	3	—	1	NOP

NOT	NOT destination Logical not			Flags O D I T S Z A P C
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
register memory	3 16+EA	— 2	2 2-4	NOT AX NOT CHARACTER

OR	OR destination,source Logical inclusive or			Flags O D I T S Z A P C 0 X X U X 0
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
register, register	3	—	2	OR AL, BL
register, memory	9+EA	1	2-4	OR DX, PORT_ID [DI]
memory, register	16+EA	2	2-4	OR FLAG_BYTE, CL
accumulator, immediate	4	—	2-3	OR AL, 01101100B
register, immediate	4	—	3-4	OR CX,01H
memory, immediate	17+EA	2	3-6	OR [BX].CMD_WORD,0CFH

OUT	OUT port,accumulator Output byte or word			Flags O D I T S Z A P C
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
immed8, accumulator	10	1	2	OUT 44, AX
DX, accumulator	8	1	1	OUT DX, AL

POP	POP destination Pop word off stack			Flags O D I T S Z A P C
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
register	8	1	1	POP DX
seg-reg (CS illegal)	8	1	1	POP DS
memory	17+EA	2	2-4	POP PARAMETER

*For the 8086, add four clocks for each 16-bit word transfer with an odd address. For the 8088, add four clocks for each 16-bit word transfer.

Table 2-21. Instruction Set Reference Data (Cont'd.)

POPF	**POPF** (no operands) Pop flags off stack			**Flags** O D I T S Z A P C R R R R R R R R
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
(no operands)	8	1	1	POPF

PUSH	**PUSH** source Push word onto stack			**Flags** O D I T S Z A P C
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
register	11	1	1	PUSH SI
seg-reg (CS legal)	10	1	1	PUSH ES
memory	16+EA	2	2-4	PUSH RETURN_CODE [SI]

PUSHF	**PUSHF** (no operands) Push flags onto stack			**Flags** O D I T S Z A P C
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
(no operands)	10	1	1	PUSHF

RCL	**RCL** destination,count Rotate left through carry			**Flags** O D I T S Z A P C X X
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
register, 1	2	—	2	RCL CX, 1
register, CL	8+4/bit	—	2	RCL AL, CL
memory, 1	15+EA	2	2-4	RCL ALPHA, 1
memory, CL	20+EA+4/bit	2	2-4	RCL [BP].PARM, CL

RCR	**RCR** designation,count Rotate right through carry			**Flags** O D I T S Z A P C X X
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
register, 1	2	—	2	RCR BX, 1
register, CL	8+4/bit	—	2	RCR BL, CL
memory, 1	15+EA	2	2-4	RCR [BX].STATUS, 1
memory, CL	20+EA+4/bit	2	2-4	RCR ARRAY [DI], CL

REP	**REP** (no operands) Repeat string operation			**Flags** O D I T S Z A P C
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
(no operands)	2	—	1	REP MOVS DEST, SRCE

*For the 8086, add four clocks for each 16-bit word transfer with an odd address. For the 8088, add four clocks for each 16-bit word transfer.

Table 2-21. Instruction Set Reference Data (Cont'd.)

REPE/REPZ	**REPE/REPZ** (no operands) Repeat string operation while equal/while zero			**Flags** O D I T S Z A P C
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
(no operands)	2	—	1	REPE CMPS DATA, KEY

REPNE/REPNZ	**REPNE/REPNZ** (no operands) Repeat string operation while not equal/not zero			**Flags** O D I T S Z A P C
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
(no operands)	2	—	1	REPNE SCAS INPUT_LINE

RET	**RET** optional-pop-value Return from procedure			**Flags** O D I T S Z A P C
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
(intra-segment, no pop)	8	1	1	RET
(intra-segment, pop)	12	1	3	RET 4
(inter-segment, no pop)	18	2	1	RET
(inter-segment, pop)	17	2	3	RET 2

ROL	**ROL** destination,count Rotate left			**Flags** O D I T S Z A P C X X
Operands	**Clocks**	**Transfers**	**Bytes**	**Coding Examples**
register, 1	2	—	2	ROL BX, 1
register, CL	8+4/bit	—	2	ROL DI, CL
memory, 1	15+EA	2	2-4	ROL FLAG_BYTE [DI],1
memory, CL	20+EA+4/bit	2	2-4	ROL ALPHA , CL

ROR	**ROR** destination,count Rotate right			**Flags** O D I T S Z A P C X X
Operand	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
register, 1	2	—	2	ROR AL, 1
register, CL	8+4/bit	—	2	ROR BX, CL
memory, 1	15+EA	2	2-4	ROR PORT_STATUS, 1
memory, CL	20+EA+4/bit	2	2-4	ROR CMD_WORD, CL

SAHF	**SAHF** (no operands) Store AH into flags			**Flags** O D I T S Z A P C R R R R R
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
(no operands)	4	—	1	SAHF

*For the 8086, add four clocks for each 16-bit word transfer with an odd address. For the 8088, add four clocks for each 16-bit word transfer.

Table 2-21. Instruction Set Reference Data (Cont'd.)

SAL/SHL	SAL/SHL destination,count Shift arithmetic left/Shift logical left			Flags O D I T S Z A P C X X
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Examples**
register,1	2	—	2	SAL AL,1
register, CL	8+4/bit	—	2	SHL DI, CL
memory,1	15+EA	2	2-4	SHL [BX].OVERDRAW, 1
memory, CL	20+EA+4/bit	2	2-4	SAL STORE_COUNT, CL

SAR	SAR destination,source Shift arithmetic right			Flags O D I T S Z A P C X X X U X X
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
register, 1	2	—	2	SAR DX, 1
register, CL	8+4/bit	—	2	SAR DI, CL
memory, 1	15+EA	2	2-4	SAR N_BLOCKS, 1
memory, CL	20+EA+4/bit	2	2-4	SAR N_BLOCKS, CL

SBB	SBB destination,source Subtract with borrow			Flags O D I T S Z A P C X X X X X X
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
register, register	3	—	2	SBB BX, CX
register, memory	9+EA	1	2-4	SBB DI, [BX].PAYMENT
memory, register	16+EA	2	2-4	SBB BALANCE, AX
accumulator, immediate	4	—	2-3	SBB AX, 2
register, immediate	4	—	3-4	SBB CL, 1
memory, immediate	17+EA	2	3-6	SBB COUNT [SI], 10

SCAS	SCAS dest-string Scan string			Flags O D I T S Z A P C X X X X X X
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
dest-string	15	1	1	SCAS INPUT_LINE
(repeat) dest-string	9+15/rep	1/rep	1	REPNE SCAS BUFFER

SEGMENT†	SEGMENT override prefix Override to specified segment			Flags O D I T S Z A P C
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
(no operands)	2	—	1	MOV SS:PARAMETER, AX

*For the 8086, add four clocks for each 16-bit word transfer with an odd address. For the 8088, add four clocks for each 16-bit word transfer.

†ASM-86 incorporates the segment override prefix into the operand specification and not as a separate instruction. SEGMENT is included in table 2-21 only for timing information.

Table 2-21. Instruction Set Reference Data (Cont'd.)

SHR	SHR destination,count Shift logical right			Flags O D I T S Z A P C X X
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
register, 1	2	—	2	SHR SI, 1
register, CL	8+4/bit	—	2	SHR SI, CL
memory, 1	15+EA	2	2-4	SHR ID_BYTE [SI] [BX], 1
memory, CL	20+EA+4/bit	2	2-4	SHR INPUT_WORD, CL

SINGLE STEP†	SINGLE STEP (Trap flag interrupt) Interrupt if TF = 1			Flags O D I T S Z A P C 0 0
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
(no operands)	50	5	N/A	N/A

STC	STC (no operands) Set carry flag			Flags O D I T S Z A P C 1
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
(no operands)	2	—	1	STC

STD	STD (no operands) Set direction flag			Flags O D I T S Z A P C 1
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
(no operands)	2	—	1	STD

STI	STI (no operands) Set interrupt enable flag			Flags O D I T S Z A P C 1
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
(no operands)	2	—	1	STI

STOS	STOS dest-string Store byte or word string			Flags O D I T S Z A P C
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
dest-string	11	1	1	STOS PRINT_LINE
(repeat) dest-string	9+10/rep	1/rep	1	REP STOS DISPLAY

*For the 8086, add four clocks for each 16-bit word transfer with an odd address. For the 8088, add four clocks for each 16-bit word transfer.

†SINGLE STEP is not an instruction; it is included in table 2-21 only for timing information.

Table 2-21. Instruction Set Reference Data (Cont'd.)

SUB	**SUB** destination,source Subtraction			**Flags** O D I T S Z A P C X X X X X X
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
register, register	3	—	2	SUB CX, BX
register, memory	9+EA	1	2-4	SUB DX, MATH_TOTAL [SI]
memory, register	16+EA	2	2-4	SUB [BP+2], CL
accumulator, immediate	4	—	2-3	SUB AL, 10
register, immediate	4	—	3-4	SUB SI, 5280
memory, immediate	17+EA	2	3-6	SUB [BP].BALANCE, 1000

TEST	**TEST** destination,source Test or non-destructive logical and			**Flags** O D I T S Z A P C 0 X X U X 0
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
register, register	3	—	2	TEST SI, DI
register, memory	9+EA	1	2-4	TEST SI, END_COUNT
accumulator, immediate	4	—	2-3	TEST AL, 00100000B
register, immediate	5	—	3-4	TEST BX, 0CC4H
memory, immediate	11+EA	—	3-6	TEST RETURN_CODE, 01H

WAIT	**WAIT** (no operands) Wait while $\overline{\text{TEST}}$ pin not asserted			**Flags** O D I T S Z A P C
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
(no operands)	3 + 5n	—	1	WAIT

XCHG	**XCHG** destination,source Exchange			**Flags** O D I T S Z A P C
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
accumulator, reg16	3	—	1	XCHG AX, BX
memory, register	17+EA	2	2-4	XCHG SEMAPHORE, AX
register, register	4	—	2	XCHG AL, BL

XLAT	**XLAT** source-table Translate			**Flags** O D I T S Z A P C
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
source-table	11	1	1	XLAT ASCII_TAB

*For the 8086, add four clocks for each 16-bit word transfer with an odd address. For the 8088, add four clocks for each 16-bit word transfer.

Table 2-21. Instruction Set Reference Data (Cont'd.)

XOR	**XOR** destination,source Logical exclusive or			**Flags** O D I T S Z A P C 0 X X U X 0
Operands	**Clocks**	**Transfers***	**Bytes**	**Coding Example**
register, register	3	—	2	XOR CX, BX
register, memory	9+EA	1	2-4	XOR CL, MASK_BYTE
memory, register	16+EA	2	2-4	XOR ALPHA [SI], DX
accumulator, immediate	4	—	2-3	XOR AL, 01000010B
register, immediate	4	—	3-4	XOR SI, 00C2H
memory, immediate	17+EA	2	3-6	XOR RETURN_CODE, 0D2H

*For the 8086, add four clocks for each 16-bit word transfer with an odd address. For the 8088, add four clocks for each 16-bit word transfer.