Intel Hex Opcodes And Mnemonics

8086/80186/80286/80386/80486 Instruction Set

AAA - Ascii Adjust for Addition
AAD - Ascii Adjust for Division
AAM - Ascii Adjust for Multiplication
AAS - Ascii Adjust for Subtraction
ADC - Add With Carry
ADD - Arithmetic Addition
AND - Logical And
ARPL - Adjusted Requested Privilege Level of Selector (286+ PM)
BOUND - Array Index Bound Check (80188+)
BSF - Bit Scan Forward (386+)
BSR - Bit Scan Reverse (386+)
BSWAP - Byte Swap (486+)
BT - Bit Test (386+)
BTC - Bit Test with Compliment (386+)
BTR - Bit Test with Reset (386+)
BTS - Bit Test and Set (386+)
CALL - Procedure Call
CBW - Convert Byte to Word
CDQ - Convert Double to Quad (386+)
CLC - Clear Carry
CLD - Clear Direction Flag
CLI - Clear Interrupt Flag (disable)
CLTS - Clear Task Switched Flag (286+ privileged)
CMC - Complement Carry Flag
CMP - Compare
CMPS - Compare String (Byte, Word or Doubleword)
CMPXCHG - Compare and Exchange
CWD - Convert Word to Doubleword
CWDE - Convert Word to Extended Doubleword (386+)
DAA - Decimal Adjust for Addition
DAS - Decimal Adjust for Subtraction
DEC - Decrement
DIV - Divide
ENTER - Make Stack Frame (80188+)
ESC - Escape
HLT - Halt CPU
IDIV - Signed Integer Division
IMUL - Signed Multiply

IN - Input Byte or Word From Port
INC - Increment
INS - Input String from Port (80188+)
INT - Interrupt
INTO - Interrupt on Overflow
INVD - Invalidate Cache (486+)
INVLPG - Invalidate Translation Look-Aside Buffer Entry (486+)
IRET/IRETD - Interrupt Return
Jxx - Jump Instructions Table
JCXZ/JECXZ - Jump if Register (E)CX is Zero
JMP - Unconditional Jump
LAHF - Load Register AH From Flags
LAR - Load Access Rights (286+ protected)
LDS - Load Pointer Using DS
LEA - Load Effective Address
LEAVE - Restore Stack for Procedure Exit (80188+)
LES - Load Pointer Using ES
LFS - Load Pointer Using FS (386+)
LGDT - Load Global Descriptor Table (286+ privileged)
LIDT - Load Interrupt Descriptor Table (286+ privileged)
LGS - Load Pointer Using GS (386+)
LLDT - Load Local Descriptor Table (286+ privileged)
LMSW - Load Machine Status Word (286+ privileged)
LOCK - Lock Bus
LODS - Load String (Byte, Word or Double)
LOOP - Decrement CX and Loop if CX Not Zero
LOOPE/LOOPZ - Loop While Equal / Loop While Zero
LOOPNZ/LOOPNE - Loop While Not Zero / Loop While Not Equal
LSL - Load Segment Limit (286+ protected)
LSS - Load Pointer Using SS (386+)
LTR - Load Task Register (286+ privileged)
MOV - Move Byte or Word
MOVS - Move String (Byte or Word)
MOVSX - Move with Sign Extend (386+)
MOVZX - Move with Zero Extend (386+)
MUL - Unsigned Multiply
NEG - Two's Complement Negation
NOP - No Operation (90h)
NOT - One's Compliment Negation (Logical NOT)
OR - Inclusive Logical OR
OUT - Output Data to Port
OUTS - Output String to Port (80188+)
POP - Pop Word off Stack
POPA/POPAD - Pop All Registers onto Stack   (80188+)
POPF/POPFD - Pop Flags off Stack
PUSH - Push Word onto Stack
PUSHA/PUSHAD - Push All Registers onto Stack (80188+)
PUSHF/PUSHFD - Push Flags onto Stack
RCL - Rotate Through Carry Left
RCR - Rotate Through Carry Right
REP - Repeat String Operation
REPE/REPZ - Repeat Equal / Repeat Zero
REPNE/REPNZ - Repeat Not Equal / Repeat Not Zero
RET/RETF - Return From Procedure
ROL - Rotate Left
ROR - Rotate Right
SAHF - Store AH Register into FLAGS

SAL/SHL - Shift Arithmetic Left / Shift Logical Left
SAR - Shift Arithmetic Right
SBB - Subtract with Borrow/Carry
SCAS - Scan String (Byte, Word or Doubleword)
SETAE/SETNB - Set if Above or Equal / Set if Not Below (386+)
SETB/SETNAE - Set if Below / Set if Not Above or Equal (386+)
SETBE/SETNA - Set if Below or Equal / Set if Not Above (386+)
SETE/SETZ - Set if Equal / Set if Zero (386+)
SETNE/SETNZ - Set if Not Equal / Set if Not Zero (386+)
SETL/SETNGE - Set if Less / Set if Not Greater or Equal (386+)
SETGE/SETNL - Set if Greater or Equal / Set if Not Less (386+)
SETLE/SETNG - Set if Less or Equal / Set if Not greater or Equal
SETG/SETNLE - Set if Greater / Set if Not Less or Equal (386+)
SETS - Set if Signed (386+)
SETNS - Set if Not Signed (386+)
SETC - Set if Carry (386+)
SETNC - Set if Not Carry (386+)
SETO - Set if Overflow (386+)
SETNO - Set if Not Overflow (386+)
SETP/SETPE - Set if Parity / Set if Parity Even (386+)
SETNP/SETPO - Set if No Parity / Set if Parity Odd (386+)
SGDT - Store Global Descriptor Table (286+ privileged)
SIDT - Store Interrupt Descriptor Table (286+ privileged)
SHL - Shift Logical Left
SHR - Shift Logical Right
SHLD/SHRD - Double Precision Shift (386+)
SLDT - Store Local Descriptor Table (286+ privileged)
SMSW - Store Machine Status Word (286+ privileged)
STC - Set Carry
STD - Set Direction Flag
STI - Set Interrupt Flag (Enable Interrupts)
STOS - Store String (Byte, Word or Doubleword)
STR - Store Task Register (286+ privileged)
SUB - Subtract
TEST - Test For Bit Pattern
VERR - Verify Read (286+ protected)
VERW - Verify Write (286+ protected)
WAIT/FWAIT - Event Wait
WBINVD - Write-Back and Invalidate Cache (486+)
XCHG - Exchange
XLAT/XLATB - Translate
XOR - Exclusive OR

Intel 8086 Family Architecture

```
    General Purpose Registers                Segment Registers
    AH/AL  AX  (EAX)  Accumulator             CS     Code Segment
    BH/BL  BX  (EBX)  Base                    DS     Data Segment
    CH/CL  CX  (ECX)  Counter                 SS     Stack Segment
    DH/DL  DX  (EDX)  Data                    ES     Extra Segment
                                             (FS)    386 and newer
(Exx) indicates 386+ 32 bit register     (GS)    386 and newer
    Pointer Registers                        Stack Registers
    SI (ESI)  Source Index                   SP (ESP)  Stack Pointer
    DI (EDI)  Destination Index              BP (EBP)  Base Pointer
    IP        Instruction Pointer
    Status Registers
    FLAGS Status Flags   (see FLAGS)
    Special Registers (386+ only)
    CR0     Control Register 0          DR0    Debug Register 0
    CR2     Control Register 2          DR1    Debug Register 1
    CR3     Control Register 3          DR2    Debug Register 2
                                        DR3    Debug Register 3
    TR4     Test Register 4             DR6    Debug Register 6
    TR5     Test Register 5             DR7    Debug Register 7
    TR6     Test Register 6
    TR7     Test Register 7

    Register          Default Segment    Valid Overrides
    BP                      SS              DS, ES, CS
    SI or DI                DS              ES, SS, CS
    DI strings              ES              None
    SI strings              DS              ES, SS, CS

- see  CPU   DETECTING    Instruction Timing
```

Instruction Clock Cycle Calculation

Some instructions require additional clock cycles due to a "Next Instruction Component" identified by a "+m" in the instruction clock cycle listings. This is due to the prefetch queue being purge on a control transfers. Below is the general rule for calculating "m":

88/86 not applicable

286 "m" is the number of bytes in the next instruction

386 "m" is the number of components in the next instruction (the instruction coding (each byte), plus the data and the displacement are all considered components)

8088/8086 Effective Address (EA) Calculation

Description	Clock Cycles
Displacement	6
Base or Index (BX,BP,SI,DI)	5
Displacement+(Base or Index)	9
Base+Index (BP+DI,BX+SI)	7
Base+Index (BP+SI,BX+DI)	8
Base+Index+Displacement (BP+DI,BX+SI)	11
Base+Index+Displacement (BP+SI+disp,BX+DI+disp)	12

- add 4 cycles for word operands at odd addresses
- add 2 cycles for segment override
- 80188/80186 timings differ from those of the 8088/8086/80286

Task State Calculation

"TS" is defined as switching from VM/486 or 80286 TSS to one of the following:

Old Task	New Task: 486 TSS (VM=0)	486 TSS (VM=1)	386 TSS (VM=0)	386 TSS (VM=1)	286 TSS
386 TSS (VM=0)			309	226	282
386 TSS (VM=1)			314	231	287
386 CPU/286 TSS			307	224	280
486 CPU/286 TSS	199	177			180

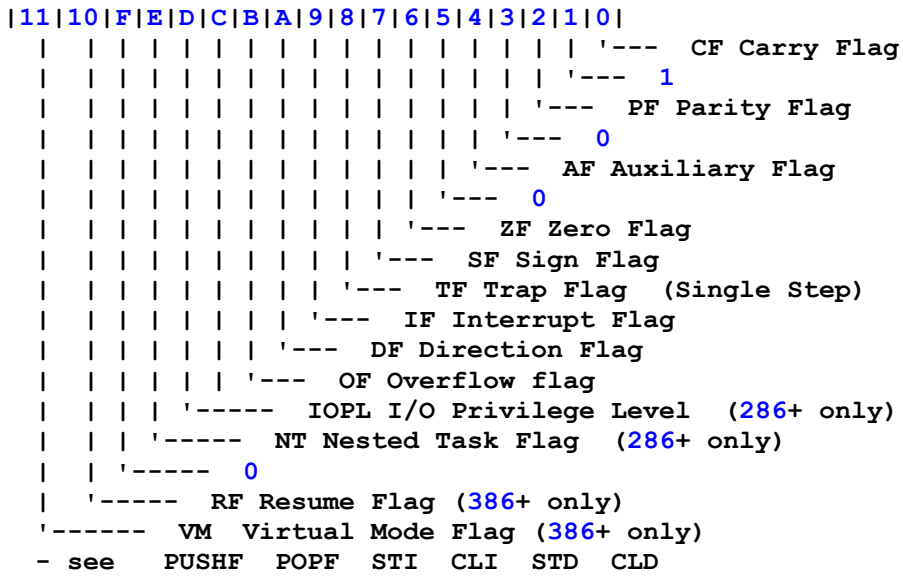
Miscellaneous

- all timings are for best case and do not take into account wait states, instruction alignment, the state of the prefetch queue, DMA refresh cycles, cache hits/misses or exception processing.
- to convert clocks to nanoseconds divide one microsecond by the processor speed in MegaHertz:

  (1000MHz/(n MHz)) = X nanoseconds

- see 8086 Architecture

FLAGS - Intel 8086 Family Flags Register



MSW - Machine Status Word (286+ only)

```
|31|30-5|4|3|2|1|0|  Machine Status Word
  |   |  | | | | '---- Protection Enable (PE)
  |   |  | | | '----- Math Present (MP)
  |   |  | | '------ Emulation (EM)
  |   |  | '------- Task Switched (TS)
  |   |  '-------- Extension Type (ET)
  |   '---------- Reserved
  '-------------- Paging (PG)
```

Bit 0	PE	Protection Enable, switches processor between protected and real mode
Bit 1	MP	Math Present, controls function of the WAIT instruction
Bit 2	EM	Emulation, indicates whether coprocessor functions are to be emulated
Bit 3	TS	Task Switched, set and interrogated by coprocessor on task switches and when interpretting coprocessor instructions
Bit 4	ET	Extension Type, indicates type of coprocessor in system
Bits 5-30		Reserved
bit 31	PG	Paging, indicates whether the processor uses page tables to translate linear addresses to physical addresses

- see SMSW LMSW

AAA - Ascii Adjust for Addition

Usage: AAA
Modifies flags: AF CF (OF,PF,SF,ZF undefined)
Changes contents of AL to valid unpacked decimal. The high order nibble is zeroed.

	Clocks				Size
Operands	808x	286	386	486	Bytes
none	8	3	4	3	1

37 AAA ASCII adjust AL after addition

AAD - Ascii Adjust for Division

Usage: AAD
Modifies flags: SF ZF PF (AF,CF,OF undefined)
Used before dividing unpacked decimal numbers. Multiplies AH by 10 and the adds result into AL. Sets AH to zero. This instruction is also known to have an undocumented behavior.
AL := 10*AH+AL
AH := 0

	Clocks				Size
Operands	808x	286	386	486	Bytes
none	60	14	19	14	2

D5 0A AAD ASCII adjust AX before division
D5 ib (No mnemonic) Adjust AX before division to number base imm8

AAM - Ascii Adjust for Multiplication

Usage: AAM
Modifies flags: PF SF ZF (AF,CF,OF undefined)
AH := AL / 10
AL := AL mod 10
Used after multiplication of two unpacked decimal numbers, this instruction adjusts an unpacked decimal number. The high order nibble of each byte must be zeroed before using this instruction. This instruction is also known to have an undocumented behavior.

	Clocks				Size
Operands	808x	286	386	486	Bytes
none	83	16	17	15	2

D4 0A AAM ASCII adjust AX after multiply
D4 ib (No mnemonic) Adjust AX after multiply to number base imm8

AAS - Ascii Adjust for Subtraction

Usage: AAS
Modifies flags: AF CF (OF,PF,SF,ZF undefined)
Corrects result of a previous unpacked decimal subtraction in AL.
High order nibble is zeroed.

	Clocks				Size
Operands	808x	286	386	486	Bytes
none	8	3	4	3	1

3F AAS ASCII adjust AL after subtraction

ADC - Add With Carry

Usage: ADC dest,src
Modifies flags: AF CF OF SF PF ZF
Sums two binary operands placing the result in the destination.
If CF is set, a 1 is added to the destination.

	Clocks				Size	
Operands	808x	286	386	486	Bytes	
reg,reg	3	2	2	1	2	
mem,reg	16+EA	7	7	3	2-4	(W88=24+EA)
reg,mem	9+EA	7	6	2	2-4	(W88=13+EA)
reg,immed	4	3	2	1	3-4	
mem,immed	17+EA	7	7	3	3-6	(W88=23+EA)
accum,immed	4	3	2	1	2-3	

14 ib ADC AL, imm8 Add with carry imm8 to AL
15 iw ADC AX, imm16 Add with carry imm16 to AX
15 id ADC EAX, imm32 Add with carry imm32 to EAX
80 /2 ib ADC r/m8, imm8 Add with carry imm8 to r/m8
81 /2 iw ADC r/m16,imm16 Add with carry imm16 to r/m16
81 /2 id ADC r/m32,imm32 Add with CF imm32 to r/m32
83 /2 ib ADC r/m16,imm8 Add with CF sign-extended imm8 to r/m16
83 /2 ib ADC r/m32,imm8 Add with CF sign-extended imm8 into r/m32
10 / r ADC r/m8,r8 Add with carry byte register to r/m8
11 / r ADC r/m16,r16 Add with carry r16 to r/m16
11 / r ADC r/m32,r32 Add with CF r32 to r/m32
12 / r ADC r8,r/m8 Add with carry r/m8 to byte register
13 / r ADC r16,r/m16 Add with carry r/m16 to r16
13 / r ADC r32,r/m32 Add with CF r/m32 to r32

ADD - Arithmetic Addition

Usage: ADD dest,src
Modifies flags: AF CF OF PF SF ZF
Adds "src" to "dest" and replacing the original contents of "dest".
Both operands are binary.

	Clocks				Size	
Operands	808x	286	386	486	Bytes	
reg,reg	3	2	2	1	2	
mem,reg	16+EA	7	7	3	2-4	(W88=24+EA)
reg,mem	9+EA	7	6	2	2-4	(W88=13+EA)
reg,immed	4	3	2	1	3-4	
mem,immed	17+EA	7	7	3	3-6	(W88=23+EA)
accum,immed	4	3	2	1	2-3	

04 ib ADD AL, imm8 Add imm8 to AL
05 iw ADD AX, imm16 Add imm16 to AX
05 id ADD EAX, imm32 Add imm32 to EAX
80 /0 ib ADD r/m8,imm8 Add imm8 to r/m8
81 /0 iw ADD r/m16,imm16 Add imm16 to r/m16
81 /0 id ADD r/m32,imm32 Add imm32 to r/m32
83 /0 ib ADD r/m16,imm8 Add sign-extended imm8 to r/m16
83 /0 ib ADD r/m32,imm8 Add sign-extended imm8 to r/m32
00 / r ADD r/m8,r8 Add r8 to r/m8
01 / r ADD r/m16,r16 Add r16 to r/m16
01 / r ADD r/m32,r32 Add r32 to r/m32
02 / r ADD r8,r/m8 Add r/m8 to r8
03 / r ADD r16,r/m16 Add r/m16 to r16
03 / r ADD r32,r/m32 Add r/m32 to r32

AND - Logical And

Usage: AND dest,src

Modifies flags: CF OF PF SF ZF (AF undefined)

Performs a logical AND of the two operands replacing the destination with the result.

	Clocks				Size	
Operands	808x	286	386	486	Bytes	
reg,reg	3	2	2	1	2	
mem,reg	16+EA	7	7	3	2-4	(W88=24+EA)
reg,mem	9+EA	7	6	1	2-4	(W88=13+EA)
reg,immed	4	3	2	1	3-4	
mem,immed	17+EA	7	7	3	3-6	(W88=23+EA)
accum,immed	4	3	2	1	2-3	

24 ib AND AL, imm8 AL AND imm8
25 iw AND AX, imm16 AX AND i mm16
25 id AND EAX, imm32 EAX AND imm32
80 /4 ib AND r/m8,imm8 r/m8 AND imm8
81 /4 iw AND r/m16,imm16 r/m16 AND imm16
81 /4 id AND r/m32,imm32 r/m32 AND imm32
83 /4 ib AND r/m16,imm8 r/m16 AND imm8 (sign-extended)
83 /4 ib AND r/m32,imm8 r/m32 AND imm8 (sign-extended)
20 /r AND r/m8,r8 r/m8 AND r8
21 / r AND r/m16,r16 r/m16 AND r16
21 / r AND r/m32,r32 r/m32 AND r32
22 / r AND r8,r/m8 r8 AND r/m8
23 / r AND r16,r/m16 r16 AND r/m16
23 / r AND r32,r/m32 r32 AND r/m32

ARPL - Adjusted Requested Privilege Level of Selector (286+ PM)

Usage: ARPL dest,src
(286+ protected mode)
Modifies flags: ZF
Compares the RPL bits of "dest" against "src". If the RPL bits of "dest" are less than "src", the destination RPL bits are set equal to the source RPL bits and the Zero Flag is set. Otherwise the Zero Flag is cleared.

	Clocks				Size
Operands	808x	286	386	486	Bytes
reg,reg	-	10	20	9	2
mem,reg	-	11	21	9	4

63 / r ARPL r/m16,r16 Adjust RPL of r/m16 to not less than RPL of r16

BOUND - Array Index Bound Check (80188+)

Usage: BOUND src,limit
Modifies flags: None
Array index in source register is checked against upper and lower bounds in memory source. The first word located at "limit" is the lower boundary and the word at "limit+2" is the upper array bound. Interrupt 5 occurs if the source value is less than or higher than the source.

	Clocks			Size
Operands	808x	286	386	486	Bytes
reg16,mem32	-	nj=13	nj=10	7	2
reg32,mem64	-	nj=13	nj=10	7	2
	-	nj = no jump taken			

62 / r BOUND r16,m16&16
62 / r BOUND r32,m32&32

BSF - Bit Scan Forward (386+)

Usage: BSF dest,src

Modifies flags: ZF

Scans source operand for first bit set. Sets ZF if a bit is found set and loads the destination with an index to first set bit. Clears ZF is no bits are found set. BSF scans forward across bit pattern (0-n) while BSR scans in reverse (n-0).

	Clocks				Size
Operands	808x	286	386	486	Bytes
reg,reg	-	-	10+3n	6-42	3
reg,mem	-	-	10+3n	7-43	3-7
reg32,reg32	-	-	10+3n	6-42	3-7
reg32,mem32	-	-	10+3n	7-43	3-7

0F BC BSF r16,r/m16 Bit scan forward on r/m16

0F BC BSF r32,r/m32 Bit scan forward on r/m32

BSR- Bit Scan Reverse (386+)

Usage: BSR dest,src

Modifies flags: ZF

Scans source operand for first bit set. Sets ZF if a bit is found set and loads the destination with an index to first set bit. Clears ZF is no bits are found set. BSF scans forward across bit pattern (0-n) while BSR scans in reverse (n-0).

	Clocks				Size
Operands	808x	286	386	486	Bytes
reg,reg	-	-	10+3n	6-103	3
reg,mem	-	-	10+3n	7-104	3-7
reg32,reg32	-	-	10+3n	6-103	3-7
reg32,mem32	-	-	10+3n	7-104	3-7

0F BD BSR r16,r/m16 Bit scan reverse on r/m16

0F BD BSR r32,r/m32 Bit scan reverse on r/m32

BSWAP - Byte Swap (486+)

Usage: BSWAP reg32

Modifies flags: none

Changes the byte order of a 32 bit register from big endian to little endian or vice versa. Result left in destination register is undefined if the operand is a 16 bit register.

	Clocks				Size
Operands	808x	286	386	486	Bytes
reg32	-	-	-	1	2

0F C8+ rd BSWAP r32 Reverses the byte order of a 32-bit register.

BT - Bit Test (386+)

Usage: BT dest,src

Modifies flags: CF

The destination bit indexed by the source value is copied into the Carry Flag.

	Clocks				Size
Operands	808x	286	386	486	Bytes
reg16,immed8	-	-	3	3	4-8
mem16,immed8	-	-	6	6	4-8
reg16,reg16	-	-	3	3	3-7
mem16,reg16	-	-	12	12	3-7

0F A3 BT r/m16,r16 Store selected bit in CF flag
0F A3 BT r/m32,r32 Store selected bit in CF flag
0F BA /4 ib BT r/m16,imm8 Store selected bit in CF flag
0F BA /4 ib BT r/m32,imm8 Store selected bit in CF flag

BTC - Bit Test with Compliment (386+)

Usage: BTC dest,src

Modifies flags: CF

The destination bit indexed by the source value is copied into the Carry Flag after being complimented (inverted).

	Clocks				Size
Operands	808x	286	386	486	Bytes
reg16,immed8	-	-	6	6	4-8
mem16,immed8	-	-	8	8	4-8
reg16,reg16	-	-	6	6	3-7
mem16,reg16	-	-	13	13	3-7

0F BB BTC r/m16,r16 Store selected bit in CF flag and complement

0F BB BTC r/m32,r32 Store selected bit in CF flag and complement

0F BA /7 ib BTC r/m16,imm8 Store selected bit in CF flag and complement

0F BA /7 ib BTC r/m32,imm8 Store selected bit in CF flag and complement

BTR - Bit Test with Reset (386+)

Usage: BTR dest,src

Modifies flags: CF

The destination bit indexed by the source value is copied into the Carry Flag and then cleared in the destination.

	Clocks				Size
Operands	808x	286	386	486	Bytes
reg16,immed8	-	-	6	6	4-8
mem16,immed8	-	-	8	8	4-8
reg16,reg16	-	-	6	6	3-7
mem16,reg16	-	-	13	13	3-7

0F B3 BTR r/m16,r16 Store selected bit in CF flag and clear
0F B3 BTR r/m32,r32 Store selected bit in CF flag and clear
0F BA /6 ib BTR r/m16,imm8 Store selected bit in CF flag and clear
0F BA /6 ib BTR r/m32,imm8 Store selected bit in CF flag and clear

BTS - Bit Test and Set (386+)

Usage: BTS dest,src

Modifies flags: CF

The destination bit indexed by the source value is copied into the Carry Flag and then set in the destination.

	Clocks				Size
Operands	808x	286	386	486	Bytes
reg16,immed8	-	-	6	6	4-8
mem16,immed8	-	-	8	8	4-8
reg16,reg16	-	-	6	6	3-7
mem16,reg16	-	-	13	13	3-7

0F AB BTS r/m16,r16 Store selected bit in CF flag and set
0F AB BTS r/m32,r32 Store selected bit in CF flag and set
0F BA /5 ib BTS r/m16,imm8 Store selected bit in CF flag and set
0F BA /5 ib BTS r/m32,imm8 Store selected bit in CF flag and set

CALL - Procedure Call

Usage: CALL destination

Modifies flags: None

Pushes Instruction Pointer (and Code Segment for far calls) onto stack and loads Instruction Pointer with the address of proc-name. Code continues with execution at CS:IP.

Operands	Clocks 808x	286	386	486
rel16 (near, IP relative)	19	7	7+m	3
rel32 (near, IP relative)	-	-	7+m	3
reg16 (near, register indirect)	16	7	7+m	5
reg32 (near, register indirect)	-	-	7+m	5
mem16 (near, memory indirect)	-	21+EA	11	10+m
mem32 (near, memory indirect)	-	-	10+m	5
ptr16:16 (far, full ptr supplied)	28	13	17+m	18
ptr16:32 (far, full ptr supplied)	-	-	17+m	18
ptr16:16 (far, ptr supplied, prot. mode)	-	26	34+m	20
ptr16:32 (far, ptr supplied, prot. mode)	-	-	34+m	20
m16:16 (far, indirect)	37+EA	16	22+m	17
m16:32 (far, indirect)	-	-	22+m	17
m16:16 (far, indirect, prot. mode)	-	29	38+m	20
m16:32 (far, indirect, prot. mode)	-	-	38+m	20
ptr16:16 (task, via TSS or task gate)	-	177	TS	37+TS
m16:16 (task, via TSS or task gate)	-	180/185	5+TS	37+TS
m16:32 (task)	-	-	TS	37+TS
m16:32 (task)	-	-	5+TS	37+TS
ptr16:16 (gate, same privilege)	-	41	52+m	35
ptr16:32 (gate, same privilege)	-	-	52+m	35
m16:16 (gate, same privilege)	-	44	56+m	35
m16:32 (gate, same privilege)	-	-	56+m	35
ptr16:16 (gate, more priv, no parm)	-	82	86+m	69
ptr16:32 (gate, more priv, no parm)	-	-	86+m	69
m16:16 (gate, more priv, no parm)	-	83	90+m	69
m16:32 (gate, more priv, no parm)	-	-	90+m	69
ptr16:16 (gate, more priv, x parms)	-	86+4x	94+4x+m	77+4x
ptr16:32 (gate, more priv, x parms)	-	-	94+4x+m	77+4x
m16:16 (gate, more priv, x parms)	-	90+4x	98+4x+m	77+4x
m16:32 (gate, more priv, x parms)	-	-	98+4x+m	77+4x

E8 cw CALL rel16 Call near, relative, displacement relative to next instruction

E8 cd CALL rel32 Call near, relative, displacement relative to next instruction

FF /2 CALL r/m16 Call near, absolute indirect, address given in r/m16

FF /2 CALL r/m32 Call near, absolute indirect, address given in r/m32

9A cd CALL ptr16:16 Call far, absolute, address given in operand

9A cp CALL ptr16:32 Call far, absolute, address given in operand

FF /3 CALL m16:16 Call far, absolute indirect, address given in m16:16

FF /3 CALL m16:32 Call far, absolute indirect, address given in m16:32

CBW - Convert Byte to Word

Usage: CBW
Modifies flags: None
Converts byte in AL to word Value in AX by extending sign of AL throughout register AH.

	Clocks				Size
Operands	808x	286	386	486	Bytes
none	2	2	3	3	1

98 CBW AX sign-extend of AL

CDQ - Convert Double to Quad (386+)

Usage: CDQ
Modifies flags: None
Converts signed DWORD in EAX to a signed quad word in EDX:EAX by extending the high order bit of EAX throughout EDX

	Clocks				Size
Operands	808x	286	386	486	Bytes
none	-	-	2	3	1

```
99 CDQ EDX:EAX
```

CLC - Clear Carry

Usage: CLC
Modifies flags: CF
Clears the Carry Flag.

	Clocks				Size
Operands	808x	286	386	486	Bytes
none	2	2	2	2	1

F8 CLC Clear CF flag

CLD - Clear Direction Flag

Usage: CLD

Modifies flags: DF

Clears the Direction Flag causing string instructions to increment the SI and DI index registers.

	Clocks				Size
Operands	808x	286	386	486	Bytes
none	2	2	2	2	1

FC CLD Clear DF flag

CLI - Clear Interrupt Flag (disable)

Usage: CLI
Modifies flags: IF
Disables the maskable hardware interrupts by clearing the Interrupt flag. NMI's and software interrupts are not inhibited.

	Clocks				Size
Operands	808x	286	386	486	Bytes
none	2	2	3	5	1

FA CLI Clear interrupt flag; interrupts disabled when interrupt

CLTS - Clear Task Switched Flag (286+ privileged)

Usage: CLTS
Modifies flags: None
Clears the Task Switched Flag in the Machine Status Register. This is a privileged operation and is generally used only by operating system code.

	Clocks			Size
Operands	808x	286	386	486	Bytes
none	-	2	5	7	2

0F 06 CLTS Clears TS flag in CR0

CMC - Complement Carry Flag

Usage: CMC
Modifies flags: CF
Toggles (inverts) the Carry Flag

	Clocks				Size
Operands	808x	286	386	486	Bytes
none	2	2	2	2	1

F5 CMC Complement CF flag

CMP - Compare

Usage: CMP dest,src
Modifies flags: AF CF OF PF SF ZF
Subtracts source from destination and updates the flags but does not save result. Flags can subsequently be checked for conditions.

	Clocks				Size	
Operands	808x	286	386	486	Bytes	
reg,reg	3	2	2	1	2	
mem,reg	9+EA	7	5	2	2-4	(W88=13+EA)
reg,mem	9+EA	6	6	2	2-4	(W88=13+EA)
reg,immed	4	3	2	1	3-4	
mem,immed	10+EA	6	5	2	3-6	(W88=14+EA)
accum,immed	4	3	2	1	2-3	

3C ib CMP AL, imm8 Compare imm8 with AL
3D iw CMP AX, imm16 Compare imm16 with AX
3D id CMP EAX, imm32 Compare imm32 with EAX
80 /7 ib CMP r/m8, imm8 Compare imm8 with r/m8
81 /7 iw CMP r/m16, imm16 Compare imm16 with r/m16
81 /7 id CMP r/m32,imm32 Compare imm32 with r/m32
83 /7 ib CMP r/m16,imm8 Compare imm8 with r/m16
83 /7 ib CMP r/m32,imm8 Compare imm8 with r/m32
38 / r CMP r/m8,r8 Compare r8 with r/m8
39 / r CMP r/m16,r16 Compare r16 with r/m16
39 / r CMP r/m32,r32 Compare r32 with r/m32
3A / r CMP r8,r/m8 Compare r/m8 with r8
3B / r CMP r16,r/m16 Compare r/m16 with r16
3B / r CMP r32,r/m32 Compare r/m32 with r32

CMPS - Compare String (Byte, Word or Doubleword)

Usage: CMPS dest,src
CMPSB
CMPSW
CMPSD (386+)

Modifies flags: AF CF OF PF SF ZF

Subtracts destination value from source without saving results. Updates flags based on the subtraction and the index registers (E)SI and (E)DI are incremented or decremented depending on the state of the Direction Flag. CMPSB inc/decrements the index registers by 1, CMPSW inc/decrements by 2, while CMPSD increments or decrements by 4. The REP prefixes can be used to process entire data items.

	Clocks				Size
Operands	808x	286	386	486	Bytes
dest,src	22	8	10	8	1 (W88=30)

A6 CMPS m8, m8
A7 CMPS m16, m16
A7 CMPS m32, m32
A6 CMPSB
A7 CMPSW
A7 CMPSD

CMPXCHG - Compare and Exchange

Usage: CMPXCHG dest,src (486+)
Modifies flags: AF CF OF PF SF ZF
Compares the accumulator (8-32 bits) with "dest". If equal the "dest" is loaded with "src", otherwise the accumulator is loaded with "dest".

	Clocks				Size
Operands	808x	286	386	486	Bytes
reg,reg	-	-	-	6	2
mem,reg	-	-	-	7	2

- add 3 clocks if the "mem,reg" comparison fails

0F B0/ r CMPXCHG r/m8,r8
0F B1/ r CMPXCHG r/m16,r16 Compare AX with r/m16
0F B1/ r CMPXCHG r/m32,r32 Compare EAX with r/m32

CWD - Convert Word to Doubleword

Usage: CWD
Modifies flags: None
Extends sign of word in register AX throughout register DX forming a doubleword quantity in DX:AX.

	Clocks				Size
Operands	808x	286	386	486	Bytes
none	5	2	2	3	1

```
99 CWD DX:AX
```

CWDE - Convert Word to Extended Doubleword (386+)

Usage: CWDE
Modifies flags: None
Converts a signed word in AX to a signed doubleword in EAX by extending the sign bit of AX throughout EAX.

	Clocks				Size
Operands	808x	286	386	486	Bytes
none	-	-	3	3	1

98 CWDE EAX sign-extend of AX

DAA - Decimal Adjust for Addition

Usage: DAA

Modifies flags: AF CF PF SF ZF (OF undefined)

Corrects result (in AL) of a previous BCD addition operation.

Contents of AL are changed to a pair of packed decimal digits.

	Clocks				Size
Operands	808x	286	386	486	Bytes
none	4	3	4	2	1

27 DAA Decimal adjust AL after addition

DAS - Decimal Adjust for Subtraction

Usage: DAS

Modifies flags: AF CF PF SF ZF (OF undefined)

Corrects result (in AL) of a previous BCD subtraction operation.

Contents of AL are changed to a pair of packed decimal digits.

	Clocks				Size
Operands	808x	286	386	486	Bytes
none	4	3	4	2	1

2F DAS Decimal adjust AL after subtraction

DEC - Decrement

Usage: DEC dest
Modifies flags: AF OF PF SF ZF
Unsigned binary subtraction of one from the destination.

	Clocks				Size
Operands	808x	286	386	486	Bytes
reg8	3	2	2	1	2
mem	15+EA	7	6	3	2-4
reg16/32	3	2	2	1	1

FE /1 DEC r/m8 Decrement r/m8 by 1
FF /1 DEC r/m16 Decrement r/m16 by 1
FF /1 DEC r/m32 Decrement r/m32 by 1
48+rw DEC r16 Decrement r16 by 1
48+rd DEC r32 Decrement r32 by 1

DIV - Divide

Usage: DIV src

Modifies flags: (AF,CF,OF,PF,SF,ZF undefined)

Unsigned binary division of accumulator by source. If the source divisor is a byte value then AX is divided by "src" and the quotient is placed in AL and the remainder in AH. If source operand is a word value, then DX:AX is divided by "src" and the quotient is stored in AX and the remainder in DX.

	Clocks				Size	
Operands	808x	286	386	486	Bytes	
reg8	80-90	14	14	16	2	
reg16	144-162	22	22	24	2	
reg32	-	-	38	40	2	
mem8	(86-96)+EA	17	17	16	2-4	
mem16	(150-168)+EA	25	25	24	2-4	(W88=158-176+EA)
mem32	-	-	41	40	2-4	

F6 /6 DIV r/m8 Unsigned divide AX by r/m8

F7 /6 DIV r/m16 Unsigned divide DX:AX by r/m16; AX

F7 /6 DIV r/m32 Unsigned divide EDX:EAX by r/m32 doubleword

ENTER - Make Stack Frame (80188+)

Usage: ENTER locals,level
Modifies flags: None
Modifies stack for entry to procedure for high level language.
Operand "locals" specifies the amount of storage to be allocated
on the stack. "Level" specifies the nesting level of the routine.
Paired with the LEAVE instruction, this is an efficient method of
entry and exit to procedures.

	Clocks			Size
Operands	808x	286	386	486	Bytes
immed16,0	-	11	10	14	4
immed16,1	-	15	12	17	4
immed16,immed8	-	12+4(n-1)	15+4(n-1)	17+3n	4

C8 iw 00 ENTER imm16,0 Create a stack frame for a procedure
C8 iw 01 ENTER imm16,1 Create a nested stack frame for a procedure
C8 iw ib ENTER imm16,imm8 Create a nested stack frame for a procedure

ESC - Escape

Usage: ESC immed,src
Modifies flags: None
Provides access to the data bus for other resident processors.
The CPU treats it as a NOP but places memory operand on bus.

	Clocks			Size
Operands	808x	286	386	486	Bytes
immed,reg	2	9-20	?		2
immed,mem	2	9-20	?		2-4

D8h xxh ESC 0
D9h xxh ESC 1
DAh xxh ESC 2
DBh xxh ESC 3
DCh xxh ESC 4
DDh xxh ESC 5
DEh xxh ESC 6
DFh xxh ESC 7

HLT - Halt CPU

Usage: HLT
Modifies flags: None
Halts CPU until RESET line is activated, NMI or maskable interrupt received. The CPU becomes dormant but retains the current CS:IP for later restart.

	Clocks				Size
Operands	808x	286	386	486	Bytes
none	2	2	5	4	1

F4 HLT Halt

IDIV - Signed Integer Division

Usage: IDIV src

Modifies flags: (AF,CF,OF,PF,SF,ZF undefined)

Signed binary division of accumulator by source. If source is a byte value, AX is divided by "src" and the quotient is stored in AL and the remainder in AH. If source is a word value, DX:AX is divided by "src", and the quotient is stored in AL and the remainder in DX.

	Clocks				Size	
Operands	808x	286	386	486	Bytes	
reg8	101-112	17	19	19	2	
reg16	165-184	25	27	27	2	
reg32	-	-	43	43	2	
mem8	(107-118)+EA	20	22	20	2-4	
mem16	(171-190)+EA	38	30	28	2-4	(W88=175-194)
mem32	-	-	46	44	2-4	

F6 /7 IDIV r/m8 Signed divide
F7 /7 IDIV r/m16 Signed divide
F7 /7 IDIV r/m32 Signed divide

IMUL - Signed Multiply

```
        Usage:  IMUL    src
                IMUL    src,immed         (286+)
                IMUL    dest,src,immed8   (286+)
                IMUL    dest,src          (386+)
        Modifies flags: CF OF (AF,PF,SF,ZF undefined)
        Signed multiplication of accumulator by "src" with result placed
        in the accumulator.  If the source operand is a byte value, it
        is multiplied by AL and the result stored in AX.  If the source
        operand is a word value it is multiplied by AX and the result is
        stored in DX:AX.  Other variations of this instruction allow
        specification of source and destination registers as well as a
        third immediate factor.
                                    Clocks                  Size
        Operands          808x   286   386   486           Bytes
        reg8              80-98   13   9-14  13-18           2
        reg16            128-154  21   9-22  13-26           2
        reg32               -      -   9-38  12-42           2
        mem8              86-104  16  12-17  13-18          2-4
        mem16             134-160 24  12-25  13-26          2-4
        mem32               -      -  12-41  13-42          2-4
        reg16,reg16         -      -   9-22  13-26          3-5
        reg32,reg32         -      -   9-38  13-42          3-5
        reg16,mem16         -      -  12-25  13-26          3-5
        reg32,mem32         -      -  12-41  13-42          3-5
        reg16,immed         -     21   9-22  13-26           3
        reg32,immed         -     21   9-38  13-42          3-6
        reg16,reg16,immed   -     2    9-22  13-26          3-6
        reg32,reg32,immed   -     21   9-38  13-42          3-6
        reg16,mem16,immed   -     24  12-25  13-26          3-6
        reg32,mem32,immed   -     24  12-41  13-42          3-6

        F6 /5 IMUL r/m8 AX¬ AL * r/m byte
        F7 /5 IMUL r/m16 DX:AX ¬ AX * r/m word
        F7 /5 IMUL r/m32 EDX:EAX ¬ EAX * r/m doubleword
        0F AF / r IMUL r16,r/m16 word register
        0F AF / r IMUL r32,r/m32 doubleword register
        6B / r ib IMUL r16,r/m16,imm8 word register
        6B / r ib IMUL r32,r/m32,imm8 doubleword register byte
        6B / r ib IMUL r16,imm8 word register
        6B / r ib IMUL r32,imm8 doubleword register
        69 / r iw IMUL r16,r/ m16,imm16
        69 / r id IMUL r32,r/ m32,imm32
        69 / r iw IMUL r16,imm16 word register ¬ r/m16 * immediate word
        69 / r id IMUL r32,imm32 doubleword register ¬ r/m32 * immediate
doubleword
```

IN - Input Byte or Word From Port

Usage: IN accum,port
Modifies flags: None
A byte, word or dword is read from "port" and placed in AL, AX or EAX respectively. If the port number is in the range of 0-255 it can be specified as an immediate, otherwise the port number must be specified in DX. Valid port ranges on the PC are 0-1024, though values through 65535 may be specified and recognized by third party vendors and PS/2's.

	Clocks				Size
Operands	808x	286	386	486	Bytes
accum,immed8	10/14	5	12	14	2
accum,immed8 (PM)			6/26	8/28/27	2
accum,DX	8/12	5	13	14	1
accum,DX (PM)			7/27	8/28/27	1

- 386+ protected mode timings depend on privilege levels.
  first number is the timing if: CPL ó IOPL
  second number is the timing if: CPL > IOPL or in VM 86 mode (386)
  CPL ò IOPL (486)
  third number is the timing when: virtual mode on 486 processor
- 486 virtual mode always requires 27 cycles

E4 ib IN AL, imm8 Input byte from imm8 I/O port address into AL
E5 ib IN AX, imm8 Input byte from imm8 I/O port address into AX
E5 ib IN EAX, imm8 Input byte from imm8 I/O port address into EAX
EC IN AL,DX Input byte from I/O port in DX into AL
ED IN AX,DX Input word from I/O port in DX into AX
ED IN EAX,DX Input doubleword from I/O port in DX into EAX

INC - Increment

Usage: INC dest

Modifies flags: AF OF PF SF ZF

Adds one to destination unsigned binary operand.

	Clocks				Size	
Operands	808x	286	386	486	Bytes	
reg8	3	2	2	1	2	
reg16	3	2	2	1	1	
reg32	3	2	2	1	1	
mem	15+EA	7	6	3	2-4	(W88=23+EA)

FE /0 INC r/m8 Increment r/m byte by 1

FF /0 INC r/m16 Increment r/m word by 1

FF /0 INC r/m32 Increment r/m doubleword by 1

40+ rw INC r16 Increment word register by 1

40+ rd INC r32 Increment doubleword register by 1

INS - Input String from Port (80188+)

Usage: INS dest,port
INSB
INSW
INSD (386+)

Modifies flags: None

Loads data from port to the destination ES:(E)DI (even if a destination operand is supplied). (E)DI is adjusted by the size of the operand and increased if the Direction Flag is cleared and decreased if the Direction Flag is set. For INSB, INSW, INSD no operands are allowed and the size is determined by the mnemonic.

		Clocks			Size
Operands	808x	286	386	486	Bytes
dest,port	-	5	15	17	1
dest,port (PM)	-	5	9/29	10/32/30	1
none	-	5	15	17	1
none (PM)	-	5	9/29	10/32/30	1

- 386+ protected mode timings depend on privilege levels.
  first number is the timing if: CPL ó IOPL
  second number is the timing if: CPL > IOPL
third number is the timing if: virtual mode on 486 processor

6C INS m8, DX
6D INS m16, DX
6D INS m32, DX

INT - Interrupt

Usage: INT num

Modifies flags: TF IF

Initiates a software interrupt by pushing the flags, clearing the Trap and Interrupt Flags, pushing CS followed by IP and loading CS:IP with the value found in the interrupt vector table. Execution then begins at the location addressed by the new CS:IP

Operands	808x	286	386	486	Size Bytes
		Clocks			
3 (constant)	52/72	23+m	33	26	2
3 (prot. mode, same priv.)	-	40+m	59	44	2
3 (prot. mode, more priv.)	-	78+m	99	71	2
3 (from VM86 to PL 0)	-	-	119	82	2
3 (prot. mode via task gate)	-	167+m	TS	37+TS	2
immed8	51/71	23+m	37	30	1
immed8 (prot. mode, same priv.)	-	40+m	59	44	1
immed8 (prot. mode, more priv.)	-	78+m	99	71	1
immed8 (from VM86 to PL 0)	-	-	119	86	1
immed8 (prot. mode, via task gate)	-	167+m	TS	37+TS	1

CC INT 3 Interrupt 3 trap to debugger

CD ib INT imm8 Interrupt vector number specified by immediate byte

INTO - Interrupt on Overflow

Usage: INTO

Modifies flags: IF TF

If the Overflow Flag is set this instruction generates an INT 4 which causes the code addressed by 0000:0010 to be executed.

		Clocks			Size
Operands	808x	286	386	486	Bytes
none: jump	53/73	24+m	35	28	1
no jump	4	3	3	3	
(prot. mode, same priv.)	-	-	59	46	1
(prot. mode, more priv.)	-	-	99	73	1
(from VM86 to PL 0)	-	-	119	84	1
(prot. mode, via task gate)	-	-	TS	39+TS	1

CE INTO Interrupt 4 if overflow flag is 1

INVD - Invalidate Cache (486+)

Usage: INVD
Modifies flags: none
Flushes CPU internal cache. Issues special function bus cycle which indicates to flush external caches. Data in write-back external caches is lost.

	Clocks				Size
Operands	808x	286	386	486	Bytes
none	-	-	-	4	2

0F 08 INVD Flush internal caches; initiate flushing of external caches.

INVLPG - Invalidate Translation Look-Aside Buffer Entry (486+)

Usage: INVLPG
Modifies flags: none
Invalidates a single page table entry in the Translation Look-Aside Buffer. Intel warns that this instruction may be implemented differently on future processors.

	Clocks				Size
Operands	808x	286	386	486	Bytes
none	-	-	-	12	2

- timing is for TLB entry hit only.

0F 01/7 INVLPG

/IRETD

IRET/IRETD - Interrupt Return

Usage: IRET
IRETD (386+)

Modifies flags: AF CF DF IF PF SF TF ZF

Returns control to point of interruption by popping IP, CS and then the Flags from the stack and continues execution at this location. CPU exception interrupts will return to the instruction that cause the exception because the CS:IP placed on the stack during the interrupt is the address of the offending instruction.

	Clocks				Size
Operands	808x	286	386	486	Bytes
iret	32/44	17+m	22	15	1
iret (prot. mode)	-	31+m	38	15	1
iret (to less privilege)	-	55+m	82	36	1
iret (different task, NT=1)	-	169+m	TS	TS+32	1
iretd	-	-	22/38	15	1
iretd (to less privilege)	-	-	82	36	1
iretd (to VM86 mode)	-	-	60	15	1
iretd (different task, NT=1)	-	-	TS	TS+32	1

- 386 timings are listed as real-mode/protected-mode

CF IRET Interrupt return (16-bit operand size)
CF IRETD Interrupt return (32-bit operand size)

Jxx - Jump Instructions Table

Mnemonic	Meaning	Jump Condition
JA	Jump if Above	CF=0 and ZF=0
JAE	Jump if Above or Equal	CF=0
JB	Jump if Below	CF=1
JBE	Jump if Below or Equal	CF=1 or ZF=1
JC	Jump if Carry	CF=1
JCXZ	Jump if CX Zero	CX=0
JE	Jump if Equal	ZF=1
JG	Jump if Greater (signed)	ZF=0 and SF=OF
JGE	Jump if Greater or Equal (signed)	SF=OF
JL	Jump if Less (signed)	SF != OF
JLE	Jump if Less or Equal (signed)	ZF=1 or SF != OF
JMP	Unconditional Jump	unconditional
JNA	Jump if Not Above	CF=1 or ZF=1
JNAE	Jump if Not Above or Equal	CF=1
JNB	Jump if Not Below	CF=0
JNBE	Jump if Not Below or Equal	CF=0 and ZF=0
JNC	Jump if Not Carry	CF=0
JNE	Jump if Not Equal	ZF=0
JNG	Jump if Not Greater (signed)	ZF=1 or SF != OF
JNGE	Jump if Not Greater or Equal (signed)	SF != OF
JNL	Jump if Not Less (signed)	SF=OF
JNLE	Jump if Not Less or Equal (signed)	ZF=0 and SF=OF
JNO	Jump if Not Overflow (signed)	OF=0
JNP	Jump if No Parity	PF=0
JNS	Jump if Not Signed (signed)	SF=0
JNZ	Jump if Not Zero	ZF=0
JO	Jump if Overflow (signed)	OF=1
JP	Jump if Parity	PF=1
JPE	Jump if Parity Even	PF=1
JPO	Jump if Parity Odd	PF=0
JS	Jump if Signed (signed)	SF=1
JZ	Jump if Zero	ZF=1

	Clocks				Size
Operands	808x	286	386	486	Bytes
Jx: jump	16	7+m	7+m	3	2
no jump	4	3	3	1	
Jx near-label	-	-	7+m	3	4
no jump	-	-	3	1	

- It's a good programming practice to organize code so the expected case is executed without a jump since the actual jump takes longer to execute than falling through the test.
- see JCXZ and JMP for their respective timings

77 cb JA rel8 Jump short if above (CF=0 and ZF=0)
73 cb JAE rel8 Jump short if above or equal (CF=0)
72 cb JB rel8 Jump short if below (CF=1)
76 cb JBE rel8 Jump short if below or equal (CF=1 or ZF=1)
72 cb JC rel8 Jump short if carry (CF=1)
E3 cb JCXZ rel8 Jump short if CX register is 0
E3 cb JECXZ rel8 Jump short if ECX register is 0
74 cb JE rel8 Jump short if equal (ZF=1)
7F cb JG rel8 Jump short if greater (ZF=0 and SF=OF)
7D cb JGE rel8 Jump short if greater or equal (SF=OF)

```
          7C cb JL rel8 Jump short if less (SF<>OF)
          7E cb JLE rel8 Jump short if less or equal (ZF=1 or SF<>OF)
          76 cb JNA rel8 Jump short if not above (CF=1 or ZF=1)
          72 cb JNAE rel8 Jump short if not above or equal (CF=1)
          73 cb JNB rel8 Jump short if not below (CF=0)
          77 cb JNBE rel8 Jump short if not below or equal (CF=0 and ZF=0)
          73 cb JNC rel8 Jump short if not carry (CF=0)
          75 cb JNE rel8 Jump short if not equal (ZF=0)
          7E cb JNG rel8 Jump short if not greater (ZF=1 or SF<>OF)
          7C cb JNGE rel8 Jump short if not greater or equal (SF<>OF)
          7D cb JNL rel8 Jump short if not less (SF=OF)
          7F cb JNLE rel8 Jump short if not less or equal (ZF=0 and SF=OF)
          71 cb JNO rel8 Jump short if not overflow (OF=0)
          7B cb JNP rel8 Jump short if not parity (PF=0)
          79 cb JNS rel8 Jump short if not sign (SF=0)
          75 cb JNZ rel8 Jump short if not zero (ZF=0)
          70 cb JO rel8 Jump short if overflow (OF=1)
          7A cb JP rel8 Jump short if parity (PF=1)
          7A cb JPE rel8 Jump short if parity even (PF=1)
          7B cb JPO rel8 Jump short if parity odd (PF=0)
          78 cb JS rel8 Jump short if sign (SF=1)
          74 cb JZ rel8 Jump short if zero (ZF = 1)
          0F 87 cw/cd JA rel16/32 Jump near if above (CF=0 and ZF=0)
          0F 83 cw/cd JAE rel16/32 Jump near if above or equal (CF=0)
          0F 82 cw/cd JB rel16/32 Jump near if below (CF=1)
          0F 86 cw/cd JBE rel16/32 Jump near if below or equal (CF=1 or ZF=1)
          0F 82 cw/cd JC rel16/32 Jump near if carry (CF=1)
          0F 84 cw/cd JE rel16/32 Jump near if equal (ZF=1)
          0F 84 cw/cd JZ rel16/32 Jump near if 0 (ZF=1)
          0F 8F cw/cd JG rel16/32 Jump near if greater (ZF=0 and SF=OF)
          0F 8D cw/cd JGE rel16/32 Jump near if greater or equal (SF=OF)
          0F 8C cw/cd JL rel16/32 Jump near if less (SF<>OF)
          0F 8E cw/cd JLE rel16/32 Jump near if less or equal (ZF=1 or SF<>OF)
          0F 86 cw/cd JNA rel16/32 Jump near if not above (CF=1 or ZF=1)
          0F 82 cw/cd JNAE rel16/32 Jump near if not above or equal (CF=1)
          0F 83 cw/cd JNB rel16/32 Jump near if not below (CF=0)
          0F 87 cw/cd JNBE rel16/32 Jump near if not below or equal (CF=0 and
ZF=0)
          0F 83 cw/cd JNC rel16/32 Jump near if not carry (CF=0)
          0F 85 cw/cd JNE rel16/32 Jump near if not equal (ZF=0)
          0F 8E cw/cd JNG rel16/32 Jump near if not greater (ZF=1 or SF<>OF)
          0F 8C cw/cd JNGE rel16/32 Jump near if not greater or equal (SF<>OF)
          0F 8D cw/cd JNL rel16/32 Jump near if not less (SF=OF)
          0F 8F cw/cd JNLE rel16/32 Jump near if not less or equal (ZF=0 and
SF=OF)
          0F 81 cw/cd JNO rel16/32 Jump near if not overflow (OF=0)
          0F 8B cw/cd JNP rel16/32 Jump near if not parity (PF=0)
          0F 89 cw/cd JNS rel16/32 Jump near if not sign (SF=0)
          0F 85 cw/cd JNZ rel16/32 Jump near if not zero (ZF=0)
          0F 80 cw/cd JO rel16/32 Jump near if overflow (OF=1)
          0F 8A cw/cd JP rel16/32 Jump near if parity (PF=1)
          0F 8A cw/cd JPE rel16/32 Jump near if parity even (PF=1)
          0F 8B cw/cd JPO rel16/32 Jump near if parity odd (PF=0)
          0F 88 cw/cd JS rel16/32 Jump near if sign (SF=1)
          0F 84 cw/cd JZ rel16/32 Jump near if 0 (ZF=1)
```

/JECXZ

JCXZ/JECXZ - Jump if Register (E)CX is Zero

```
        Usage:  JCXZ    label
                JECXZ   label  (386+)
        Modifies flags: None
        Causes execution to branch to "label" if register CX is zero.  Uses
        unsigned comparision.
                                    Clocks                  Size
        Operands            808x  286   386   486           Bytes
        label:  jump         18   8+m   9+m    8             2
                no jump       6    4     5     5

        E3 cb JCXZ rel8 Jump short if CX register is 0
        E3 cb JECXZ rel8 Jump short if ECX register is 0
```

JMP - Unconditional Jump

Usage: JMP target
Modifies flags: None
Unconditionally transfers control to "label". Jumps by default are within -32768 to 32767 bytes from the instruction following the jump. NEAR and SHORT jumps cause the IP to be updated while FAR jumps cause CS and IP to be updated.

Operands	Clocks 808x	286	386	486
rel8 (relative)	15	7+m	7+m	3
rel16 (relative)	15	7+m	7+m	3
rel32 (relative)	-	-	7+m	3
reg16 (near, register indirect)	11	7+m	7+m	5
reg32 (near, register indirect)	-	-	7+m	5
mem16 (near, mem indirect)	18+EA	11+m	10+m	5
mem32 (near, mem indirect)	24+EA	15+m	10+m	5
ptr16:16 (far, dword immed)	-	-	12+m	17
ptr16:16 (far, PM dword immed)	-	-	27+m	19
ptr16:16 (call gate, same priv.)	-	38+m	45+m	32
ptr16:16 (via TSS)	-	175+m	TS	42+TS
ptr16:16 (via task gate)	-	180+m	TS	43+TS
mem16:16 (far, indirect)	-	-	43+m	13
mem16:16 (far, PM indirect)	-	-	31+m	18
mem16:16 (call gate, same priv.)	-	41+m	49+m	31
mem16:16 (via TSS)	-	178+m	5+TS	41+TS
mem16:16 (via task gate)	-	183+m	5+TS	42+TS
ptr16:32 (far, 6 byte immed)	-	-	12+m	13
ptr16:32 (far, PM 6 byte immed)	-	-	27+m	18
ptr16:32 (call gate, same priv.)	-	-	45+m	31
ptr16:32 (via TSS)	-	-	TS	42+TS
ptr16:32 (via task state)	-	-	TS	43+TS
m16:32 (far, address at dword)	-	-	43+m	13
m16:32 (far, address at dword)	-	-	31+m	18
m16:32 (call gate, same priv.)	-	-	49+m	31
m16:32 (via TSS)	-	-	5+TS	41+TS
m16:32 (via task state)	-	-	5+TS	42+TS

EB cb JMP rel8 Jump short, relative, displacement relative to next instruction
E9 cw JMP rel16 Jump near, relative, displacement relative to next instruction
E9 cd JMP rel32 Jump near, relative, displacement relative to next instruction
FF /4 JMP r/m16 Jump near, absolute indirect, address given in r/m16
FF /4 JMP r/m32 Jump near, absolute indirect, address given in r/m32
EA cd JMP ptr16:16 Jump far, absolute, address given in operand
EA cp JMP ptr16:32 Jump far, absolute, address given in operand
FF /5 JMP m16:16 Jump far, absolute indirect, address given in m16:16
FF /5 JMP m16:32 Jump far, absolute indirect, address given in m16:32

LAHF - Load Register AH From Flags

Usage: LAHF

Modifies flags: None

Copies bits 0-7 of the flags register into AH. This includes flags AF, CF, PF, SF and ZF other bits are undefined.

AH := SF ZF xx AF xx PF xx CF

	Clocks				Size
Operands	808x	286	386	486	Bytes
none	4	2	2	3	1

9F LAHF Load: AH = EFLAGS(SF:ZF:0:AF:0:PF:1:CF)

LAR - Load Access Rights (286+ protected)

Usage: LAR dest,src

Modifies flags: ZF

The high byte of the of the destination register is overwritten by the value of the access rights byte and the low order byte is zeroed depending on the selection in the source operand. The Zero Flag is set if the load operation is successful.

	Clocks				Size
Operands	808x	286	386	486	Bytes
reg16,reg16	-	14	15	11	3
reg32,reg32	-	-	15	11	3
reg16,mem16	-	16	16	11	3-7
reg32,mem32	-	-	16	11	3-7

0F 02 / r LAR r16,r/m16 r16 ¬ r/m16 masked by FF00H

0F 02 / r LAR r32,r/m32 r32 ¬ r/m32 masked by 00FxFF00H

LDS - Load Pointer Using DS

Usage: LDS dest,src
Modifies flags: None
Loads 32-bit pointer from memory source to destination register and DS. The offset is placed in the destination register and the segment is placed in DS. To use this instruction the word at the lower memory address must contain the offset and the word at the higher address must contain the segment. This simplifies the loading of far pointers from the stack and the interrupt vector table.

	Clocks				Size
Operands	808x	286	386	486	Bytes
reg16,mem32	16+EA	7	7	6	2-4
reg,mem (PM)	-	-	22	12	5-7

C5 / r LDS r16,m16:16 Load DS: r16 with far pointer from memory
C5 / r LDS r32,m16:32 Load DS: r32 with far pointer from memory

LEA - Load Effective Address

Usage: LEA dest,src

Modifies flags: None

Transfers offset address of "src" to the destination register.

		Clocks			Size
Operands	808x	286	386	486	Bytes
reg,mem	2+EA	3	2	1	2-4

- the MOV instruction can often save clock cycles when used in place of LEA on 8088 processors

8D / r LEA r16,m Store effective address for m in register r16

8D / r LEA r32,m Store effective address for m in register r32

LEAVE - Restore Stack for Procedure Exit (80188+)

Usage: LEAVE

Modifies flags: None

Releases the local variables created by the previous ENTER instruction by restoring SP and BP to their condition before the procedure stack frame was initialized.

	Clocks			Size
Operands	808x	286	386	486	Bytes
none	-	5	4	5	1

C9 LEAVE Set SP to BP, then pop BP

C9 LEAVE Set ESP to EBP, then pop EBP

LES - Load Pointer Using ES

Usage: LES dest,src

Modifies flags: None

Loads 32-bit pointer from memory source to destination register and ES. The offset is placed in the destination register and the segment is placed in ES. To use this instruction the word at the lower memory address must contain the offset and the word at the higher address must contain the segment. This simplifies the loading of far pointers from the stack and the interrupt vector table.

	Clocks				Size	
Operands	808x	286	386	486	Bytes	
reg,mem	16+EA	7	7	6	2-4	(W88=24+EA)
reg,mem (PM)	-	-	22	12	5-7	

C4 / r LES r16,m16:16 Load ES: r16 with far pointer from memory

C4 / r LES r32,m16:32 Load ES: r32 with far pointer from memory

LFS - Load Pointer Using FS (386+)

Usage: LFS dest,src
Modifies flags: None
Loads 32-bit pointer from memory source to destination register and FS. The offset is placed in the destination register and the segment is placed in FS. To use this instruction the word at the lower memory address must contain the offset and the word at the higher address must contain the segment. This simplifies the loading of far pointers from the stack and the interrupt vector table.

	Clocks				Size
Operands	808x	286	386	486	Bytes
reg,mem	-	-	7	6	5-7
reg,mem (PM)	-	-	22	12	5-7

0F B4 / r LFS r16,m16:16 Load FS: r16 with far pointer from memory
0F B4 / r LFS r32,m16:32 Load FS: r32 with far pointer from memory

LGDT - Load Global Descriptor Table (286+ privileged)

Usage: LGDT src
Modifies flags: None
Loads a value from an operand into the Global Descriptor Table (GDT) register.

	Clocks				Size
Operands	808x	286	386	486	Bytes
mem64	-	11	11	11	5

0F 01 /2 LGDT m16&32 Load m into GDTR

LIDT - Load Interrupt Descriptor Table (286+ privileged)

Usage: LIDT src

Modifies flags: None

Loads a value from an operand into the Interrupt Descriptor Table (IDT) register.

	Clocks				Size
Operands	808x	286	386	486	Bytes
mem64	-	12	11	11	5

0F 01 /3 LIDT m16&32 Load m into IDTR

LGS - Load Pointer Using GS (386+)

Usage: LGS dest,src
Modifies flags: None
Loads 32-bit pointer from memory source to destination register and GS. The offset is placed in the destination register and the segment is placed in GS. To use this instruction the word at the lower memory address must contain the offset and the word at the higher address must contain the segment. This simplifies the loading of far pointers from the stack and the interrupt vector table.

	Clocks				Size
Operands	808x	286	386	486	Bytes
reg,mem	-	-	7	6	5-7
reg,mem (PM)	-	-	22	12	5-7

0F B5 / r LGS r16,m16:16 Load GS: r16 with far pointer from memory
0F B5 / r LGS r32,m16:32 Load GS: r32 with far pointer from memory

LLDT - Load Local Descriptor Table (286+ privileged)

Usage: LLDT src

Modifies flags: None

Loads a value from an operand into the Local Descriptor Table Register (LDTR).

	Clocks				Size
Operands	808x	286	386	486	Bytes
reg16	-	17	20	11	3
mem16	-	19	24	11	5

0F 00 /2 LLDT r/m16 Load segment selector r/m16 into LDTR

LMSW - Load Machine Status Word (286+ privileged)

Usage: LMSW src
Modifies flags: None
Loads the Machine Status Word (MSW) from data found at "src"

	Clocks			Size
Operands	808x	286	386	486	Bytes
reg16	-	3	10	13	3
mem16	-	6	13	13	5

0F 01 /6 LMSW r/m16 Loads r/m16 in machine status word of CR0

LOCK - Lock Bus

Usage: LOCK
LOCK: (386+ prefix)
Modifies flags: None
This instruction is a prefix that causes the CPU assert bus lock signal during the execution of the next instruction. Used to avoid two processors from updating the same data location. The 286 always asserts lock during an XCHG with memory operands. This should only be used to lock the bus prior to XCHG, MOV, IN and OUT instructions.

	Clocks				Size
Operands	808x	286	386	486	Bytes
none	2	0	0	1	1

F0 LOCK Asserts LOCK# signal

LODS - Load String (Byte, Word or Double)

Usage: LODS src
LODSB
LODSW
LODSD (386+)

Modifies flags: None

Transfers string element addressed by DS:SI (even if an operand is supplied) to the accumulator. SI is incremented based on the size of the operand or based on the instruction used. If the Direction Flag is set SI is decremented, if the Direction Flag is clear SI is incremented. Use with REP prefixes.

	Clocks				Size
Operands	808x	286	386	486	Bytes
src	12/16	5	5	5	1

AC LODS m8 Load byte at address DS:(E)SI into AL
AD LODS m16 Load word at address DS:(E)SI into AX
AD LODS m32 Load doubleword at address DS:(E)SI into EAX
AC LODSB Load byte at address DS:(E)SI into AL
AD LODSW Load word at address DS:(E)SI into AX
AD LODSD Load doubleword at address DS:(E)SI into EAX

LOOP - Decrement CX and Loop if CX Not Zero

Usage: LOOP label

Modifies flags: None

Decrements CX by 1 and transfers control to "label" if CX is not Zero. The "label" operand must be within -128 or 127 bytes of the instruction following the loop instruction

		Clocks				Size
Operands		808x	286	386	486	Bytes
label:	jump	18	8+m	11+m	6	2
	no jump	5	4	?	2	

E2 cb LOOP rel8 Decrement count; jump short if count = 0

LOOPE/LOOPZ - Loop While Equal / Loop While Zero

Usage: LOOPE label
LOOPZ label

Modifies flags: None

Decrements CX by 1 (without modifying the flags) and transfers control to "label" if CX != 0 and the Zero Flag is set. The "label" operand must be within -128 or 127 bytes of the instruction following the loop instruction.

		Clocks				Size
Operands		808x	286	386	486	Bytes
label:	jump	18	8+m	11+m	9	2
	no jump	5	4	?	6	

E1 cb LOOPE
E1 cb LOOPZ

LOOPNZ/LOOPNE - Loop While Not Zero / Loop While Not Equal

Usage: LOOPNZ label
LOOPNE label

Modifies flags: None

Decrements CX by 1 (without modifying the flags) and transfers control to "label" if CX != 0 and the Zero Flag is clear. The "label" operand must be within -128 or 127 bytes of the instruction following the loop instruction.

		Clocks				Size
Operands		808x	286	386	486	Bytes
label:	jump	19	8+m	11+m	9	2
	no jump	5	4	?	6	

E0 cb LOOPNE
E0 cb LOOPNZ

LSL - Load Segment Limit (286+ protected)

Usage: LSL dest,src

Modifies flags: ZF

Loads the segment limit of a selector into the destination register if the selector is valid and visible at the current privilege level. If loading is successful the Zero Flag is set, otherwise it is cleared.

	Clocks				Size
Operands	808x	286	386	486	Bytes
reg16,reg16	-	14	20/25	10	3
reg32,reg32	-	-	20/25	10	3
reg16,mem16	-	16	21/26	10	5
reg32,mem32	-	-	21/26	10	5

- 386 times are listed "byte granular" / "page granular"

0F 03 / r LSL r16,r/m16 Load: r16 ¬ segment limit, selector r/m16
0F 03 / r LSL r32,r/m32 Load: r32 ¬ segment limit, selector r/m32)

LSS - Load Pointer Using SS (386+)

Usage: LSS dest,src

Modifies flags: None

Loads 32-bit pointer from memory source to destination register and SS. The offset is placed in the destination register and the segment is placed in SS. To use this instruction the word at the lower memory address must contain the offset and the word at the higher address must contain the segment. This simplifies the loading of far pointers from the stack and the interrupt vector table.

	Clocks				Size
Operands	808x	286	386	486	Bytes
reg,mem	-	-	7	6	5-7
reg,mem (PM)	-	-	22	12	5-7

0F B2 / r LSS r16,m16:16 Load SS: r16 with far pointer from memory
0F B2 / r LSS r32,m16:32 Load SS: r32 with far pointer from memory

LTR - Load Task Register (286+ privileged)

Usage: LTR src

Modifies flags: None

Loads the current task register with the value specified in "src".

	Clocks				Size
Operands	808x	286	386	486	Bytes
reg16	-	17	23	20	3
mem16	-	19	27	20	5

0F 00 /3 LTR r/m16 Load r/m16 into task register

MOV - Move Byte or Word

Usage: MOV dest,src

Modifies flags: None

Copies byte or word from the source operand to the destination operand. If the destination is SS interrupts are disabled except on early buggy 808x CPUs. Some CPUs disable interrupts if the destination is any of the segment registers

	Clocks				Size	
Operands	808x	286	386	486	Bytes	
reg,reg	2	2	2	1	2	
mem,reg	9+EA	3	2	1	2-4	(W88=13+EA)
reg,mem	8+EA	5	4	1	2-4	(W88=12+EA)
mem,immed	10+EA	3	2	1	3-6	(W88=14+EA)
reg,immed	4	2	2	1	2-3	
mem,accum	10	3	2	1	3	(W88=14)
accum,mem	10	5	4	1	3	(W88=14)
segreg,reg16	2	2	2	3	2	
segreg,mem16	8+EA	5	5	9	2-4	(W88=12+EA)
reg16,segreg	2	2	2	3	2	
mem16,segreg	9+EA	3	2	3	2-4	(W88=13+EA)
reg32,CR0/CR2/CR3	-	-	6	4		
CR0,reg32	-	-	10	16		
CR2,reg32	-	-	4	4	3	
CR3,reg32	-	-	5	4	3	
reg32,DR0/DR1/DR2/DR3	-	22	10		3	
reg32,DR6/DR7	-	-	22	10	3	
DR0/DR1/DR2/DR3,reg32	-	22	11		3	
DR6/DR7,reg32	-	-	16	11	3	
reg32,TR6/TR7	-	-	12	4	3	
TR6/TR7,reg32	-	-	12	4	3	
reg32,TR3				3		
TR3,reg32				6		

- when the 386 special registers are used all operands are 32 bits

88 / r MOV r/m8,r8 Move r8 to r/m8
89 / r MOV r/m16,r16 Move r16 to r/m16
89 / r MOV r/m32,r32 Move r32 to r/m32
8A / r MOV r8,r/m8 Move r/m8 to r8
8B / r MOV r16,r/m16 Move r/m16 to r16
8B / r MOV r32,r/m32 Move r/m32 to r32
8C / r MOV r/m16,Sreg** Move segment register to r/m16
8E / r MOV Sreg,r/m16** Move r/m16 to segment register
A0 MOV AL, moffs8* Move byte at (seg:offset) to AL
A1 MOV AX, moffs16* Move word at (seg:offset) to AX
A1 MOV EAX, moffs32* Move doubleword at (seg:offset) to EAX
A2 MOV moffs8*,AL Move AL to (seg:offset)
A3 MOV moffs16*,AX Move AX to (seg:offset)
A3 MOV moffs32*,EAX Move EAX to (seg:offset)
B0+ rb MOV r8,imm8 Move imm8 to r8
B8+ rw MOV r16,imm16 Move imm16 to r16
B8+ rd MOV r32,imm32 Move imm32 to r32
C6 / 0 MOV r/m8,imm8 Move imm8 to r/m8
C7 / 0 MOV r/m16,imm16 Move imm16 to r/m16
C7 / 0 MOV r/m32,imm32 Move imm32 to r/m32
0F 22 / r MOV CR0, r32 Move r32 to CR0

```
0F 22 / r MOV CR2, r32 Move r32 to CR2
0F 22 / r MOV CR3, r32 Move r32 to CR3
0F 22 / r MOV CR4, r32 Move r32 to CR4
0F 20 / r MOV r32,CR0 Move CR0 to r32
0F 20 / r MOV r32,CR2 Move CR2 to r32
0F 20 / r MOV r32,CR3 Move CR3 to r32
0F 20 / r MOV r32,CR4 Move CR4 to r32
0F 21/ r MOV r32, DR0-DR7 Move debug register to r32
0F 23 / r MOV DR0-DR7, r32 Move r32 to debug register
```

MOVS - Move String (Byte or Word)

Usage: MOVS dest,src
MOVSB
MOVSW
MOVSD (386+)

Modifies flags: None

Copies data from addressed by DS:SI (even if operands are given) to the location ES:DI destination and updates SI and DI based on the size of the operand or instruction used. SI and DI are incremented when the Direction Flag is cleared and decremented when the Direction Flag is Set. Use with REP prefixes.

	Clocks				Size
Operands	808x	286	386	486	Bytes
dest,src	18	5	7	7	1 (W88=26)

A4 MOVS m8, m8 Move byte at address DS:(E)SI to address ES:(E)DI
A5 MOVS m16, m16 Move word at address DS:(E)SI to address ES:(E)DI
A5 MOVS m32, m32 Move doubleword at address DS:(E)SI to address ES:(E)DI
A4 MOVSB Move byte at address DS:(E)SI to address ES:(E)DI
A5 MOVSW Move word at address DS:(E)SI to address ES:(E)DI
A5 MOVSD Move doubleword at address DS:(E)SI to address ES:(E)DI

MOVSX - Move with Sign Extend (386+)

Usage: MOVSX dest,src

Modifies flags: None

Copies the value of the source operand to the destination register with the sign extended.

	Clocks				Size
Operands	808x	286	386	486	Bytes
reg,reg	-	-	3	3	3
reg,mem	-	-	6	3	3-7

0F BE / r MOVSX r16,r/m8 Move byte to word with sign-extension

0F BE / r MOVSX r32,r/m8 Move byte to doubleword, sign-extension

0F BF / r MOVSX r32,r/m16 Move word to doubleword, sign-extension

MOVZX - Move with Zero Extend (386+)

Usage: MOVZX dest,src

Modifies flags: None

Copies the value of the source operand to the destination register with the zeroes extended.

	Clocks				Size
Operands	808x	286	386	486	Bytes
reg,reg	-	-	3	3	3
reg,mem	-	-	6	3	3-7

0F B6 / r MOVZX r16,r/m8 Move byte to word with zero-extension

0F B6 / r MOVZX r32,r/m8 Move byte to doubleword, zero-extension

0F B7 / r MOVZX r32,r/m16 Move word to doubleword, zero-extension

MUL - Unsigned Multiply

Usage: MUL src

Modifies flags: CF OF (AF,PF,SF,ZF undefined)

Unsigned multiply of the accumulator by the source. If "src" is a byte value, then AL is used as the other multiplicand and the result is placed in AX. If "src" is a word value, then AX is multiplied by "src" and DX:AX receives the result. If "src" is a double word value, then EAX is multiplied by "src" and EDX:EAX receives the result. The 386+ uses an early out algorithm which makes multiplying any size value in EAX as fast as in the 8 or 16 bit registers.

	Clocks				Size
Operands	808x	286	386	486	Bytes
reg8	70-77	13	9-14	13-18	2
reg16	118-113	21	9-22	13-26	2
reg32	-	-	9-38	13-42	2-4
mem8	(76-83)+EA	16	12-17	13-18	2-4
mem16	(124-139)+EA	24	12-25	13-26	2-4
mem32	-	-	12-21	13-42	2-4

F6 /4 MUL r/m8 Unsigned multiply (AX ¬ AL * r/m8)

F7 /4 MUL r/m16 Unsigned multiply (DX:AX ¬ AX * r/m16)

F7 /4 MUL r/m32 Unsigned multiply (EDX:EAX ¬ EAX * r/m32)

NEG - Two's Complement Negation

Usage: NEG dest

Modifies flags: AF CF OF PF SF ZF

Subtracts the destination from 0 and saves the 2s complement of "dest" back into "dest".

	Clocks				Size	
Operands	808x	286	386	486	Bytes	
reg	3	2	2	1	2	
mem	16+EA	7	6	3	2-4	(W88=24+EA)

F6 /3 NEG r/m8 Two's complement negate r/m8

F7 /3 NEG r/m16 Two's complement negate r/m16

F7 /3 NEG r/m32 Two's complement negate r/m32

NOP - No Operation (90h)

Usage: NOP
Modifies flags: None
This is a do nothing instruction. It results in occupation of both space and time and is most useful for patching code segments.
(This is the original XCHG AL,AL instruction)

	Clocks			Size
Operands	808x	286	386	486	Bytes
none	3	3	3	1	1

90 NOP No operation

NOT - One's Compliment Negation (Logical NOT)

Usage: NOT dest

Modifies flags: None

Inverts the bits of the "dest" operand forming the 1s complement.

	Clocks				Size	
Operands	808x	286	386	486	Bytes	
reg	3	2	2	1	2	
mem	16+EA	7	6	3	2-4	(W88=24+EA)

F6 /2 NOT r/m8 Reverse each bit of r/m8

F7 /2 NOT r/m16 Reverse each bit of r/m16

F7 /2 NOT r/m32 Reverse each bit of r/m32

OR - Inclusive Logical OR

Usage: OR dest,src

Modifies flags: CF OF PF SF ZF (AF undefined)

Logical inclusive OR of the two operands returning the result in the destination. Any bit set in either operand will be set in the destination.

	Clocks				Size	
Operands	808x	286	386	486	Bytes	
reg,reg	3	2	2	1	2	
mem,reg	16+EA	7	7	3	2-4	(W88=24+EA)
reg,mem	9+EA	7	6	2	2-4	(W88=13+EA)
reg,immed	4	3	2	1	3-4	
mem8,immed8	17+EA	7	7	3	3-6	
mem16,immed16	25+EA	7	7	3	3-6	
accum,immed	4	3	2	1	2-3	

0C ib OR AL, imm8 AL OR imm8
0D iw OR AX, imm16 AX OR imm16
0D id OR EAX, imm32 EAX OR imm32
80 /1 ib OR r/m8,imm8 r/m8 OR imm8
81 /1 iw OR r/m16,imm16 r/m16 OR imm16
81 /1 id OR r/m32,imm32 r/m32 OR imm32
83 /1 ib OR r/m16,imm8 r/m16 OR imm8 (sign-extended)
83 /1 ib OR r/m32,imm8 r/m32 OR imm8 (sign-extended)
08 / r OR r/m8,r8 r/m8 OR r8
09 / r OR r/m16,r16 r/m16 OR r16
09 / r OR r/m32,r32 r/m32 OR r32
0A / r OR r8,r/m8 r8 OR r/m8
0B / r OR r16,r/m16 r16 OR r/m16
0B / r OR r32,r/m32 r32 OR r/m32

OUT - Output Data to Port

Usage: OUT port,accum
Modifies flags: None
Transfers byte in AL,word in AX or dword in EAX to the specified hardware port address. If the port number is in the range of 0-255 it can be specified as an immediate. If greater than 255 then the port number must be specified in DX. Since the PC only decodes 10 bits of the port address, values over 1023 can only be decoded by third party vendor equipment and also map to the port range 0-1023.

	Clocks				Size
Operands	808x	286	386	486	Bytes
immed8,accum	10/14	3	10	16	2
immed8,accum (PM)	-	-	4/24	11/31/29	2
DX,accum	8/12	3	11	16	1
DX,accum (PM)	-	-	5/25	10/30/29	1

- 386+ protected mode timings depend on privilege levels.
  first number is the timing when: CPL ó IOPL
  second number is the timing when: CPL > IOPL
third number is the timing when: virtual mode on 486 processor

E6 ib OUT imm8, AL Output byte in AL to I/O port address imm8
E7 ib OUT imm8, AX Output word in AX to I/O port address imm8
E7 ib OUT imm8, EAX Output doubleword in EAX to I/O port address imm8
EE OUT DX, AL Output byte in AL to I/O port address in DX
EF OUT DX, AX Output word in AX to I/O port address in DX
EF OUT DX, EAX Output doubleword in EAX to I/O port address in DX

OUTS - Output String to Port (80188+)

Usage: OUTS port,src
OUTSB
OUTSW
OUTSD (386+)

Modifies flags: None

Transfers a byte, word or doubleword from "src" to the hardware port specified in DX. For instructions with no operands the "src" is located at DS:SI and SI is incremented or decremented by the size of the operand or the size dictated by the instruction format. When the Direction Flag is set SI is decremented, when clear, SI is incremented. If the port number is in the range of 0-255 it can be specified as an immediate. If greater than 255 then the port number must be specified in DX. Since the PC only decodes 10 bits of the port address, values over 1023 can only be decoded by third party vendor equipment and also map to the port range 0-1023.

	Clocks				Size
Operands	808x	286	386	486	Bytes
port,src	-	5	14	17	1
port,src (PM)	-	-	8/28	10/32/30	1

- 386+ protected mode timings depend on privilege levels.
  first number is the timing when: CPL ó IOPL
  second number is the timing when: CPL > IOPL
third number is the timing when: virtual mode on 486 processor

```
6E OUTS DX, m8
6F OUTS DX, m16
6F OUTS DX, m32
6E OUTSB
6F OUTSW
6F OUTSD
```

POP - Pop Word off Stack

Usage: POP dest
Modifies flags: None
Transfers word at the current stack top (SS:SP) to the destination then increments SP by two to point to the new stack top. CS is not a valid destination.

	Clocks				Size
Operands	808x	286	386	486	Bytes
reg16	8	5	4	4	1
reg32	4	-	-	4	1
segreg	8	5	7	3	1
mem16	17+EA	5	5	6	2-4
mem32	5	-	-	6	2-4

8F /0 POP m16 Pop top of stack into m16; increment stack pointer
8F /0 POP m32 Pop top of stack into m32; increment stack pointer
58+ rw POP r16 Pop top of stack into r16; increment stack pointer
58+ rd POP r32 Pop top of stack into r32; increment stack pointer
1F POP DS Pop top of stack into DS; increment stack pointer
07 POP ES Pop top of stack into ES; increment stack pointer
17 POP SS Pop top of stack into SS; increment stack pointer
0F A1 POP FS Pop top of stack into FS; increment stack pointer
0F A9 POP GS Pop top of stack into GS; increment stack pointer

POPA/POPAD - Pop All Registers onto Stack (80188+)

Usage: POPA
POPAD (386+)

Modifies flags: None

Pops the top 8 words off the stack into the 8 general purpose 16/32 bit registers. Registers are popped in the following order: (E)DI, (E)SI, (E)BP, (E)SP, (E)DX, (E)CX and (E)AX. The (E)SP value popped from the stack is actually discarded.

	Clocks			Size
Operands	808x	286	386	486	Bytes
none	-	19	24	9	1

61 POPA Pop DI, SI, BP, BX, DX, CX, and AX
61 POPAD Pop EDI, ESI, EBP, EBX, EDX, ECX, and EAX

POPF/POPFD - Pop Flags off Stack

Usage: POPF
POPFD (386+)

Modifies flags: all flags

Pops word/doubleword from stack into the Flags Register and then increments SP by 2 (for POPF) or 4 (for POPFD).

	Clocks			Size
Operands	808x	286	386	486	Bytes
none	8/12	5	5	9	1 (W88=12)
none (PM)	-	-	5	6	1

9D POPF Pop top of stack into lower 16 bits of EFLAGS
9D POPFD Pop top of stack into EFLAGS

PUSH - Push Word onto Stack

Usage: PUSH src
PUSH immed (80188+ only)

Modifies flags: None

Decrements SP by the size of the operand (two or four, byte values are sign extended) and transfers one word from source to the stack top (SS:SP).

	Clocks				Size	
Operands	808x	286	386	486	Bytes	
reg16	11/15	3	2	1	1	
reg32	-	-	2	1	1	
mem16	16+EA	5	5	4	2-4	(W88=24+EA)
mem32	-	-	5	4	2-4	
segreg	10/14	3	2	3	1	
immed	-	3	2	1	2-3	

FF /6 PUSH r/m16 Push r/m16
FF /6 PUSH r/m32 Push r/m32
50+ rw PUSH r16 Push r16
50+ rd PUSH r32 Push r32
6A PUSH imm8 Push imm8
68 PUSH imm16 Push imm16
68 PUSH imm32 Push imm32
0E PUSH CS Push CS
16 PUSH SS Push SS
1E PUSH DS Push DS
06 PUSH ES Push ES
0F A0 PUSH FS Push FS
0F A8 PUSH GS Push GS

PUSHA/PUSHAD - Push All Registers onto Stack (80188+)

Usage: PUSHA
PUSHAD (386+)

Modifies flags: None

Pushes all general purpose registers onto the stack in the following order: (E)AX, (E)CX, (E)DX, (E)BX, (E)SP, (E)BP, (E)SI, (E)DI. The value of SP is the value before the actual push of SP.

	Clocks			Size
Operands	808x	286	386	486	Bytes
none	-	19	24	11	1

60 PUSHA Push AX, CX, DX, BX, original SP, BP, SI, and DI
60 PUSHAD Push EAX, ECX, EDX, EBX, original ESP, EBP, ESI, and EDI

PUSHF/PUSHFD - Push Flags onto Stack

Usage: PUSHF
PUSHFD (386+)

Modifies flags: None

Transfers the Flags Register onto the stack. PUSHF saves a 16 bit value while PUSHFD saves a 32 bit value.

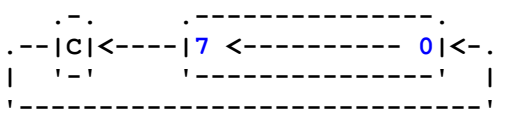
	Clocks				Size
Operands	808x	286	386	486	Bytes
none	10/14	3	4	4	1
none (PM)	-	-	4	3	1

9C PUSHF Push lower 16 bits of EFLAGS
9C PUSHFD Push EFLAGS

RCL - Rotate Through Carry Left

Usage: RCL dest,count

Modifies flags: CF OF



Rotates the bits in the destination to the left "count" times with all data pushed out the left side re-entering on the right. The Carry Flag holds the last bit rotated out.

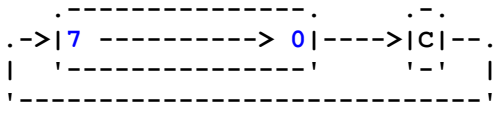
	Clocks				Size	
Operands	808x	286	386	486	Bytes	
reg,1	2	2	9	3	2	
mem,1	15+EA	7	10	4	2-4	(W88=23+EA)
reg,CL	8+4n	5+n	9	8-30	2	
mem,CL	20+EA+4n	8+n	10	9-31	2-4	(W88=28+EA+4n)
reg,immed8	-	5+n	9	8-30	3	
mem,immed8	-	8+n	10	9-31	3-5	

D0 /2 RCL r/m8,1 Rotate 9 bits (CF, r/m8) left once
D2 /2 RCL r/m8,CL Rotate 9 bits (CF, r/m8) left CL times
C0 /2 ib RCL r/m8,imm8 Rotate 9 bits (CF, r/m8) left imm8 times
D1 /2 RCL r/m16,1 Rotate 17 bits (CF, r/m16) left once
D3 /2 RCL r/m16,CL Rotate 17 bits (CF, r/m16) left CL times
C1 /2 ib RCL r/m16,imm8 Rotate 17 bits (CF, r/m16) left imm8 times
D1 /2 RCL r/m32,1 Rotate 33 bits (CF, r/m32) left once
D3 /2 RCL r/m32,CL Rotate 33 bits (CF, r/m32) left CL times
C1 /2 ib RCL r/m32,imm8 Rotate 33 bits (CF, r/m32) left imm8 times

RCR - Rotate Through Carry Right

Usage: RCR dest,count

Modifies flags: CF OF



Rotates the bits in the destination to the right "count" times with all data pushed out the right side re-entering on the left. The Carry Flag holds the last bit rotated out.

	Clocks				Size	
Operands	808x	286	386	486	Bytes	
reg,1	2	2	9	3	2	
mem,1	15+EA	7	10	4	2-4	(W88=23+EA)
reg,CL	8+4n	5+n	9	8-30	2	
mem,CL	20+EA+4n	8+n	10	9-31	2-4	(W88=28+EA+4n)
reg,immed8	-	5+n	9	8-30	3	
mem,immed8	-	8+n	10	9-31	3-5	

D0 /3 RCR r/m8,1 Rotate 9 bits (CF, r/m8) right once
D2 /3 RCR r/m8,CL Rotate 9 bits (CF, r/m8) right CL times
C0 /3 ib RCR r/m8,imm8 Rotate 9 bits (CF, r/m8) right imm8 times
D1 /3 RCR r/m16,1 Rotate 17 bits (CF, r/m16) right once
D3 /3 RCR r/m16,CL Rotate 17 bits (CF, r/m16) right CL times
C1 /3 ib RCR r/m16,imm8 Rotate 17 bits (CF, r/m16) right imm8 times
D1 /3 RCR r/m32,1 Rotate 33 bits (CF, r/m32) right once
D3 /3 RCR r/m32,CL Rotate 33 bits (CF, r/m32) right CL times
C1 /3 ib RCR r/m32,imm8 Rotate 33 bits (CF, r/m32) right imm8 times

REP - Repeat String Operation

Usage: REP
Modifies flags: None
Repeats execution of string instructions while CX != 0. After each string operation, CX is decremented and the Zero Flag is tested. The combination of a repeat prefix and a segment override on CPU's before the 386 may result in errors if an interrupt occurs before CX=0. The following code shows code that is susceptible to this and how to avoid it:

```
 again:  rep movs  byte ptr ES:[DI],ES:[SI]   ; vulnerable instr.
             jcxz  next              ; continue if REP successful
             loop  again             ; interrupt goofed count
 next:
```

	Clocks				Size
Operands	808x	286	386	486	Bytes
none	2	2	2	1	

F3 6C REP INS r/m8, DX Input (E)CX bytes from port DX into ES:[(E)DI]
F3 6D REP INS r/m16,DX Input (E)CX words from port DX into ES:[(E)DI]
F3 6D REP INS r/m32,DX Input (E)CX doublewords from port DX into ES:[(E)DI]
F3 A4 REP MOVS m8,m8 Move (E)CX bytes from DS:[(E)SI] to ES:[(E)DI]
F3 A5 REP MOVS m16,m16 Move (E)CX words from DS:[(E)SI] to ES:[(E)DI]
F3 A5 REP MOVS m32,m32 Move (E)CX doublewords from DS:[(E)SI] to ES:[(E)DI]
F3 6E REP OUTS DX, r/m8 Output (E)CX bytes from DS:[(E)SI] to port DX
F3 6F REP OUTS DX, r/m16 Output (E)CX words from DS:[(E)SI] to port DX
F3 6F REP OUTS DX, r/m32 Output (E)CX doublewords from DS:[(E)SI] to port DX
F3 AC REP LODS AL Load (E)CX bytes from DS:[(E)SI] to AL
F3 AD REP LODS AX Load (E)CX words from DS:[(E)SI] to AX
F3 AD REP LODS EAX Load (E)CX doublewords from DS:[(E)SI] to EAX
F3 AA REP STOS m8 Fill (E)CX bytes at ES:[(E)DI] with AL
F3 AB REP STOS m16 Fill (E)CX words at ES:[(E)DI] with AX
F3 AB REP STOS m32 Fill (E)CX doublewords at ES:[(E)DI] with EAX

REPE/REPZ - Repeat Equal / Repeat Zero

Usage: REPE
REPZ

Modifies flags: None

Repeats execution of string instructions while CX != 0 and the Zero Flag is set. CX is decremented and the Zero Flag tested after each string operation. The combination of a repeat prefix and a segment override on processors other than the 386 may result in errors if an interrupt occurs before CX=0.

	Clocks				Size
Operands	808x	286	386	486	Bytes
none	2	2	2	1	

F3 A6 REPE CMPS m8,m8 Find nonmatching bytes in ES:[(E)DI] and DS:[(E)SI]

F3 A7 REPE CMPS m16,m16 Find nonmatching words in ES:[(E)DI] and DS:[(E)SI]

F3 A7 REPE CMPS m32,m32 Find nonmatching doublewords in ES:[(E)DI] and DS:[(E)SI]

F3 AE REPE SCAS m8 Find non-AL byte starting at ES:[(E)DI]

F3 AF REPE SCAS m16 Find non-AX word starting at ES:[(E)DI]

F3 AF REPE SCAS m32 Find non-EAX doubleword starting at ES:[(E)DI]

REPNE/REPNZ - Repeat Not Equal / Repeat Not Zero

Usage: REPNE
REPNZ

Modifies flags: None

Repeats execution of string instructions while CX != 0 and the Zero Flag is clear. CX is decremented and the Zero Flag tested after each string operation. The combination of a repeat prefix and a segment override on processors other than the 386 may result in errors if an interrupt occurs before CX=0.

	Clocks				Size
Operands	808x	286	386	486	Bytes
none	2	2	2	1	

F2 A6 REPNE CMPS m8,m8 Find matching bytes in ES:[(E)DI] and DS:[(E)SI]
F2 A7 REPNE CMPS m16,m16 Find matching words in ES:[(E)DI] and DS:[(E)SI]
F2 A7 REPNE CMPS m32,m32 Find matching doublewords in ES:[(E)DI] and DS:[(E)SI]
F2 AE REPNE SCAS m8 Find AL, starting at ES:[(E)DI]
F2 AF REPNE SCAS m16 Find AX, starting at ES:[(E)DI]
F2 AF REPNE SCAS m32 Find EAX, starting at ES:[(E)DI]

RET/RETF - Return From Procedure

Usage: RET nBytes
RETF nBytes
RETN nBytes

Modifies flags: None

Transfers control from a procedure back to the instruction address saved on the stack. "n bytes" is an optional number of bytes to release. Far returns pop the IP followed by the CS, while near returns pop only the IP register.

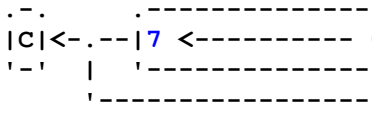
	Clocks				Size
Operands	808x	286	386	486	Bytes
retn	16/20	11+m	10+m	5	1
retn immed	20/24	11+m	10+m	5	3
retf	26/34	15+m	18+m	13	1
retf (PM, same priv.)	-	-	32+m	18	1
retf (PM, lesser priv.)	-	-	68	33	1
retf immed	25/33	15+m	18+m	14	3
retf immed (PM, same priv.)			32+m	17	1
retf immed (PM, lesser priv.)			68	33	1

C3 RET Near return to calling procedure
CB RET Far return to calling procedure
C2 iw RET imm16 Near return to calling procedure and pop imm16 bytes from stack
CA iw RET imm16 Far return to calling procedure and pop imm16 bytes from stack

ROL - Rotate Left

Usage: ROL dest,count

Modifies flags: CF OF



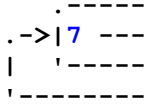
Rotates the bits in the destination to the left "count" times with all data pushed out the left side re-entering on the right. The Carry Flag will contain the value of the last bit rotated out.

	Clocks				Size	
Operands	808x	286	386	486	Bytes	
reg,1	2	2	3	3	2	
mem,1	15+EA	7	7	4	2-4	(W88=23+EA)
reg,CL	8+4n	5+n	3	3	2	
mem,CL	20+EA+4n	8+n	7	4	2-4	(W88=28+EA+4n)
reg,immed8	-	5+n	3	2	3	
mem,immed8	-	8+n	7	4	3-5	

D0 /0 ROL r/m8,1 Rotate 8 bits r/m8 left once
D2 /0 ROL r/m8,CL Rotate 8 bits r/m8 left CL times
C0 /0 ib ROL r/m8,imm8 Rotate 8 bits r/m8 left imm8 times
D1 /0 ROL r/m16,1 Rotate 16 bits r/m16 left once
D3 /0 ROL r/m16,CL Rotate 16 bits r/m16 left CL times
C1 /0 ib ROL r/m16,imm8 Rotate 16 bits r/m16 left imm8 times
D1 /0 ROL r/m32,1 Rotate 32 bits r/m32 left once
D3 /0 ROL r/m32,CL Rotate 32 bits r/m32 left CL times
C1 /0 ib ROL r/m32,imm8 Rotate 32 bits r/m32 left imm8 times

ROR - Rotate Right

Usage: ROR dest,count
Modifies flags: CF OF



Rotates the bits in the destination to the right "count" times with all data pushed out the right side re-entering on the left. The Carry Flag will contain the value of the last bit rotated out.

Operands	Clocks 808x	286	386	486	Size Bytes	
reg,1	2	2	3	3	2	
mem,1	15+EA	7	7	4	2-4	(W88=23+EA)
reg,CL	8+4n	5+n	3	3	2	
mem,CL	20+EA+4n	8+n	7	4	2-4	(W88=28+EA+4n)
reg,immed8	-	5+n	3	2	3	
mem,immed8	-	8+n	7	4	3-5	

D0 /1 ROR r/m8,1 Rotate 8 bits r/m8 right once
D2 /1 ROR r/m8,CL Rotate 8 bits r/m8 right CL times
C0 /1 ib ROR r/m8,imm8 Rotate 8 bits r/m16 right imm8 times
D1 /1 ROR r/m16,1 Rotate 16 bits r/m16 right once
D3 /1 ROR r/m16,CL Rotate 16 bits r/m16 right CL times
C1 /1 ib ROR r/m16,imm8 Rotate 16 bits r/m16 right imm8 times
D1 /1 ROR r/m32,1 Rotate 32 bits r/m32 right once
D3 /1 ROR r/m32,CL Rotate 32 bits r/m32 right CL times
C1 /1 ib ROR r/m32,imm8 Rotate 32 bits r/m32 right imm8 times

SAHF - Store AH Register into FLAGS

Usage: SAHF

Modifies flags: AF CF PF SF ZF

Transfers bits 0-7 of AH into the Flags Register. This includes AF, CF, PF, SF and ZF.

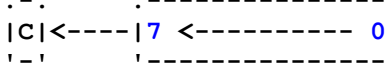
	Clocks				Size
Operands	808x	286	386	486	Bytes
none	4	2	3	2	1

9E SAHF 2 Loads SF, ZF, AF, PF, and CF from AH into EFLAGS register

SAL/SHL - Shift Arithmetic Left / Shift Logical Left

Usage: SAL dest,count
SHL dest,count

Modifies flags: CF OF PF SF ZF (AF undefined)



Shifts the destination left by "count" bits with zeroes shifted in on right. The Carry Flag contains the last bit shifted out.

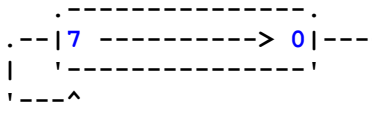
Operands	Clocks 808x	286	386	486	Size Bytes	
reg,1	2	2	3	3	2	
mem,1	15+EA	7	7	4	2-4	(W88=23+EA)
reg,CL	8+4n	5+n	3	3	2	
mem,CL	20+EA+4n	8+n	7	4	2-4	(W88=28+EA+4n)
reg,immed8	-	5+n	3	2	3	
mem,immed8	-	8+n	7	4	3-5	

D0 /4 SAL r/m8,1 Multiply r/m8 by 2, once
D2 /4 SAL r/m8,CL Multiply r/m8 by 2, CL times
C0 /4 ib SAL r/m8,imm8 Multiply r/m8 by 2, imm8 times
D1 /4 SAL r/m16,1 Multiply r/m16 by 2, once
D3 /4 SAL r/m16,CL Multiply r/m16 by 2, CL times
C1 /4 ib SAL r/m16,imm8 Multiply r/m16 by 2, imm8 times
D1 /4 SAL r/m32,1 Multiply r/m32 by 2, once
D3 /4 SAL r/m32,CL Multiply r/m32 by 2, CL times
C1 /4 ib SAL r/m32,imm8 Multiply r/m32 by 2, imm8 times

SAR - Shift Arithmetic Right

Usage: SAR dest,count

Modifies flags: CF OF PF SF ZF (AF undefined)



Shifts the destination right by "count" bits with the current sign bit replicated in the leftmost bit. The Carry Flag contains the last bit shifted out.

	Clocks				Size	
Operands	808x	286	386	486	Bytes	
reg,1	2	2	3	3	2	
mem,1	15+EA	7	7	4	2-4	(W88=23+EA)
reg,CL	8+4n	5+n	3	3	2	
mem,CL	20+EA+4n	8+n	7	4	2-4	(W88=28+EA+4n)
reg,immed8	-	5+n	3	2	3	
mem,immed8	-	8+n	7	4	3-5	

D0 /7 SAR r/m8,1 Signed divide* r/m8 by 2, once
D2 /7 SAR r/m8,CL Signed divide* r/m8 by 2, CL times
C0 /7 ib SAR r/m8,imm8 Signed divide* r/m8 by 2, imm8 times
D1 /7 SAR r/m16,1 Signed divide* r/m16 by 2, once
D3 /7 SAR r/m16,CL Signed divide* r/m16 by 2, CL times
C1 /7 ib SAR r/m16,imm8 Signed divide* r/m16 by 2, imm8 times
D1 /7 SAR r/m32,1 Signed divide* r/m32 by 2, once
D3 /7 SAR r/m32,CL Signed divide* r/m32 by 2, CL times
C1 /7 ib SAR r/m32,imm8 Signed divide* r/m32 by 2, imm8 times

SBB - Subtract with Borrow/Carry

Usage: SBB dest,src

Modifies flags: AF CF OF PF SF ZF

Subtracts the source from the destination, and subtracts 1 extra if the Carry Flag is set. Results are returned in "dest".

	Clocks			Size	
Operands	808x	286	386	486	Bytes	
reg,reg	3	2	2	1	2	
mem,reg	16+EA	7	6	3	2-4	(W88=24+EA)
reg,mem	9+EA	7	7	2	2-4	(W88=13+EA)
reg,immed	4	3	2	1	3-4	
mem,immed	17+EA	7	7	3	3-6	(W88=25+EA)
accum,immed	4	3	2	1	2-3	

1C ib SBB AL, imm8 Subtract with borrow imm8 from AL
1D iw SBB AX, imm16 Subtract with borrow imm16 from AX
1D id SBB EAX, imm32 Subtract with borrow imm32 from EAX
80 /3 ib SBB r/m8,imm8 Subtract with borrow imm8 from r/m8
81 /3 iw SBB r/m16,imm16 Subtract with borrow imm16 from r/m16
81 /3 id SBB r/m32,imm32 Subtract with borrow imm32 from r/m32
83 /3 ib SBB r/m16,imm8 Subtract with borrow sign-extended imm8 from r/m16
83 /3 ib SBB r/m32,imm8 Subtract with borrow sign-extended imm8 from r/m32
18 / r SBB r/m8,r8 Subtract with borrow r8 from r/m8
19 / r SBB r/m16,r16 Subtract with borrow r16 from r/m16
19 / r SBB r/m32,r32 Subtract with borrow r32 from r/m32
1A / r SBB r8,r/m8 Subtract with borrow r/m8 from r8
1B / r SBB r16,r/m16 Subtract with borrow r/m16 from r16
1B / r SBB r32,r/m32 Subtract with borrow r/m32 from r32

SCAS - Scan String (Byte, Word or Doubleword)

Usage: SCAS   string
        SCASB
        SCASW
        SCASD   (386+)

Modifies flags: AF CF OF PF SF ZF

Compares value at ES:DI (even if operand is specified) from the accumulator and sets the flags similar to a subtraction. DI is incremented/decremented based on the instruction format (or operand size) and the state of the Direction Flag. Use with REP prefixes.

	Clocks				Size
Operands	808x	286	386	486	Bytes
string	15	7	7	6	1 (W88=19)

AE SCAS m8 Compare AL with byte at ES:(E)DI and set status flags
AF SCAS m16 Compare AX with word at ES:(E)DI and set status flags
AF SCAS m32 Compare EAX with doubleword at ES(E)DI and set status flags
AE SCASB Compare AL with byte at ES:(E)DI and set status flags
AF SCASW Compare AX with word at ES:(E)DI and set status flags
AF SCASD Compare EAX with doubleword at ES:(E)DI and set status flags

SETAE/SETNB - Set if Above or Equal / Set if Not Below (386+)

Usage: SETAE dest
SETNB dest
(unsigned, 386+)
Modifies flags: none
Sets the byte in the operand to 1 if the Carry Flag is clear
otherwise sets the operand to 0.

	Clocks				Size
Operands	808x	286	386	486	Bytes
reg8	-	-	4	3	3
mem8	-	-	5	4	3

0F 97 SETA r/m8 Set byte if above (CF=0 and ZF=0)
0F 93 SETAE r/m8 Set byte if above or equal (CF=0)

SETB/SETNAE - Set if Below / Set if Not Above or Equal (386+)

Usage: SETB dest
SETNAE dest
(unsigned, 386+)
Modifies flags: none
Sets the byte in the operand to 1 if the Carry Flag is set otherwise sets the operand to 0.

	Clocks				Size
Operands	808x	286	386	486	Bytes
reg8	-	-	4	3	3
mem8	-	-	5	4	3

0F 92 SETB r/m8 Set byte if below (CF=1)
0F 92 SETNAE r/m8 Set byte if not above or equal (CF=1)

SETBE/SETNA - Set if Below or Equal / Set if Not Above (386+)

Usage: SETBE dest
SETNA dest
(unsigned, 386+)
Modifies flags: none
Sets the byte in the operand to 1 if the Carry Flag or the Zero Flag is set, otherwise sets the operand to 0.

	Clocks				Size
Operands	808x	286	386	486	Bytes
reg8	-	-	4	3	3
mem8	-	-	5	4	3

0F 96 SETBE r/m8 Set byte if below or equal (CF=1 or ZF=1)
0F 96 SETNA r/m8 Set byte if not above (CF=1 or ZF=1)

SETE/SETZ - Set if Equal / Set if Zero (386+)

Usage: SETE dest
SETZ dest

Modifies flags: none

Sets the byte in the operand to 1 if the Zero Flag is set, otherwise sets the operand to 0.

	Clocks				Size
Operands	808x	286	386	486	Bytes
reg8	-	-	4	3	3
mem8	-	-	5	4	3

0F 94 SETE r/m8 Set byte if equal (ZF=1)
0F 94 SETZ r/m8 Set byte if zero (ZF=1)

SETNE/SETNZ - Set if Not Equal / Set if Not Zero (386+)

Usage: SETNE dest
SETNZ dest

Modifies flags: none

Sets the byte in the operand to 1 if the Zero Flag is clear, otherwise sets the operand to 0.

	Clocks				Size
Operands	808x	286	386	486	Bytes
reg8	-	-	4	3	3
mem8	-	-	5	4	3

0F 95 SETNE r/m8 Set byte if not equal (ZF=0)
0F 95 SETNZ r/m8 Set byte if not zero (ZF=0)

SETL/SETNGE - Set if Less / Set if Not Greater or Equal (386+)

Usage: SETL dest
SETNGE dest
(signed, 386+)
Modifies flags: none
Sets the byte in the operand to 1 if the Sign Flag is not equal to the Overflow Flag, otherwise sets the operand to 0.

	Clocks				Size
Operands	808x	286	386	486	Bytes
reg8	-	-	4	3	3
mem8	-	-	5	4	3

0F 9C SETL r/m8 Set byte if less (SF<>OF)
0F 9C SETNGE r/m8 Set if not greater or equal (SF<>OF)

SETGE/SETNL - Set if Greater or Equal / Set if Not Less (386+)

Usage: SETGE dest
SETNL dest
(signed, 386+)
Modifies flags: none
Sets the byte in the operand to 1 if the Sign Flag equals the Overflow Flag, otherwise sets the operand to 0.

	Clocks				Size
Operands	808x	286	386	486	Bytes
reg8	-	-	4	3	3
mem8	-	-	5	4	3

0F 9D SETGE r/m8 Set byte if greater or equal (SF=OF)
0F 9D SETNL r/m8 Set byte if not less (SF=OF)

SETLE/SETNG - Set if Less or Equal / Set if Not greater or Equal (386+)

Usage: SETLE dest
SETNG dest
(signed, 386+)
Modifies flags: none
Sets the byte in the operand to 1 if the Zero Flag is set or the Sign Flag is not equal to the Overflow Flag, otherwise sets the operand to 0.

	Clocks				Size
Operands	808x	286	386	486	Bytes
reg8	-	-	4	3	3
mem8	-	-	5	4	3

0F 9E SETLE r/m8 Set byte if less or equal (ZF=1 or SF<>OF)
0F 9E SETNG r/m8 Set byte if not greater (ZF=1 or SF<>OF)

SETG/SETNLE - Set if Greater / Set if Not Less or Equal (386+)

Usage: SETG dest
SETNLE dest
(signed, 386+)
Modifies flags: none
Sets the byte in the operand to 1 if the Zero Flag is clear or the Sign Flag equals to the Overflow Flag, otherwise sets the operand to 0.

	Clocks				Size
Operands	808x	286	386	486	Bytes
reg8	-	-	4	3	3
mem8	-	-	5	4	3

0F 9F SETG r/m8 Set byte if greater (ZF=0 and SF=OF)
0F 9F SETNLE r/m8 Set byte if not less or equal (ZF=0 and SF=OF)

SETS - Set if Signed (386+)

Usage: SETS dest

Modifies flags: none

Sets the byte in the operand to 1 if the Sign Flag is set, otherwise sets the operand to 0.

	Clocks				Size
Operands	808x	286	386	486	Bytes
reg8	-	-	4	3	3
mem8	-	-	5	4	3

0F 98 SETS r/m8 Set byte if sign (SF=1)

SETNS - Set if Not Signed (386+)

Usage: SETNS dest

Modifies flags: none

Sets the byte in the operand to 1 if the Sign Flag is clear, otherwise sets the operand to 0.

	Clocks				Size
Operands	808x	286	386	486	Bytes
reg8	-	-	4	3	3
mem8	-	-	5	4	3

0F 99 SETNS r/m8 Set byte if not sign (SF=0)

SETC - Set if Carry (386+)

Usage: SETC dest

Modifies flags: none

Sets the byte in the operand to 1 if the Carry Flag is set, otherwise sets the operand to 0.

	Clocks				Size
Operands	808x	286	386	486	Bytes
reg8	-	-	4	3	3
mem8	-	-	5	4	3

0F 92 SETC r/m8 Set if carry (CF=1)

SETNC - Set if Not Carry (386+)

Usage: SETNC dest

Modifies flags: none

Sets the byte in the operand to 1 if the Carry Flag is clear, otherwise sets the operand to 0.

	Clocks				Size
Operands	808x	286	386	486	Bytes
reg8	-	-	4	3	3
mem8	-	-	5	4	3

0F 93 SETNC r/m8 Set byte if not carry (CF=0)

SETO - Set if Overflow (386+)

Usage: SETO dest
Modifies flags: none
Sets the byte in the operand to 1 if the Overflow Flag is set, otherwise sets the operand to 0.

	Clocks				Size
Operands	808x	286	386	486	Bytes
reg8	-	-	4	3	3
mem8	-	-	5	4	3

0F 90 SETO r/m8 Set byte if overflow (OF=1)

SETNO - Set if Not Overflow (386+)

Usage: SETNO dest

Modifies flags: none

Sets the byte in the operand to 1 if the Overflow Flag is clear, otherwise sets the operand to 0.

	Clocks				Size
Operands	808x	286	386	486	Bytes
reg8	-	-	4	3	3
mem8	-	-	5	4	3

0F 91 SETNO r/m8 Set byte if not overflow (OF=0)

SETP/SETPE - Set if Parity / Set if Parity Even (386+)

Usage: SETP dest
        SETPE dest

Modifies flags: none

Sets the byte in the operand to 1 if the Parity Flag is set, otherwise sets the operand to 0.

	Clocks				Size
Operands	808x	286	386	486	Bytes
reg8	-	-	4	3	3
mem8	-	-	5	4	3

0F 9A SETP r/m8 Set byte if parity (PF=1)
0F 9A SETPE r/m8 Set byte if parity even (PF=1)

SETNP/SETPO - Set if No Parity / Set if Parity Odd (386+)

Usage: SETNP dest
SETPO dest

Modifies flags: none

Sets the byte in the operand to 1 if the Parity Flag is clear, otherwise sets the operand to 0.

	Clocks				Size
Operands	808x	286	386	486	Bytes
reg8	-	-	4	3	3
mem8	-	-	5	4	3

0F 9B SETNP r/m8 Set byte if not parity (PF=0)
0F 9B SETPO r/m8 Set byte if parity odd (PF=0)

SGDT - Store Global Descriptor Table (286+ privileged)

Usage: SGDT dest
Modifies flags: none
Stores the Global Descriptor Table (GDT) Register into the specified operand.

	Clocks				Size
Operands	808x	286	386	486	Bytes
mem64	-	11	9	10	5

0F 01 /0 SGDT m Store GDTR to m

SIDT - Store Interrupt Descriptor Table (286+ privileged)

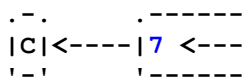
Usage: SIDT dest
Modifies flags: none
Stores the Interrupt Descriptor Table (IDT) Register into the specified operand.

	Clocks				Size
Operands	808x	286	386	486	Bytes
mem64	-	12	9	10	5

0F 01 /1 SIDT m Store IDTR to m

SHL - Shift Logical Left

Usage: SAL dest,count
SHL dest,count
Modifies flags: CF OF PF SF ZF (AF undefined)



Shifts the destination left by "count" bits with zeroes shifted in on right. The Carry Flag contains the last bit shifted out.

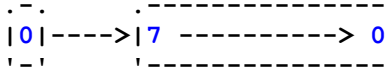
Operands	808x	286	386	486	Size Bytes	
reg,1	2	2	3	3	2	
mem,1	15+EA	7	7	4	2-4	(W88=23+EA)
reg,CL	8+4n	5+n	3	3	2	
mem,CL	20+EA+4n	8+n	7	4	2-4	(W88=28+EA+4n)
reg,immed8	-	5+n	3	2	3	
mem,immed8	-	8+n	7	4	3-5	

D0 /4 SHL r/m8,1 Multiply r/m8 by 2, once
D2 /4 SHL r/m8,CL Multiply r/m8 by 2, CL times
C0 /4 ib SHL r/m8,imm8 Multiply r/m8 by 2, imm8 times
D1 /4 SHL r/m16,1 Multiply r/m16 by 2, once
D3 /4 SHL r/m16,CL Multiply r/m16 by 2, CL times
C1 /4 ib SHL r/m16,imm8 Multiply r/m16 by 2, imm8 times
D1 /4 SHL r/m32,1 Multiply r/m32 by 2, once
D3 /4 SHL r/m32,CL Multiply r/m32 by 2, CL times
C1 /4 ib SHL r/m32,imm8 Multiply r/m32 by 2, imm8 times

SHR - Shift Logical Right

Usage: SHR dest,count

Modifies flags: CF OF PF SF ZF (AF undefined)



Shifts the destination right by "count" bits with zeroes shifted in on the left. The Carry Flag contains the last bit shifted out.

	Clocks				Size	
Operands	808x	286	386	486	Bytes	
reg,1	2	2	3		2	
mem,1	15+EA	7	7		2-4	(W88=23+EA)
reg,CL	8+4n	5+n	3		2	
mem,CL	20+EA+4n	8+n	7		2-4	(W88=28+EA+4n)
reg,immed8	-	5+n	3		3	
mem,immed8	-	8+n	7		3-5	

D0 /5 SHR r/m8,1 Unsigned divide r/m8 by 2, once
D2 /5 SHR r/m8,CL Unsigned divide r/m8 by 2, CL times
C0 /5 ib SHR r/m8,imm8 Unsigned divide r/m8 by 2, imm8 times
D1 /5 SHR r/m16,1 Unsigned divide r/m16 by 2, once
D3 /5 SHR r/m16,CL Unsigned divide r/m16 by 2, CL times
C1 /5 ib SHR r/m16,imm8 Unsigned divide r/m16 by 2, imm8 times
D1 /5 SHR r/m32,1 Unsigned divide r/m32 by 2, once
D3 /5 SHR r/m32,CL Unsigned divide r/m32 by 2, CL times
C1 /5 ib SHR r/m32,imm8 Unsigned divide r/m32 by 2, imm8 times

SHLD/SHRD - Double Precision Shift (386+)

Usage: SHLD dest,src,count
SHRD dest,src,count

Modifies flags: CF PF SF ZF (OF,AF undefined)

SHLD shifts "dest" to the left "count" times and the bit positions opened are filled with the most significant bits of "src". SHRD shifts "dest" to the right "count" times and the bit positions opened are filled with the least significant bits of the second operand. Only the 5 lower bits of "count" are used.

	Clocks				Size
Operands	808x	286	386	486	Bytes
reg16,reg16,immed8	-	-	3	2	4
reg32,reg32,immed8	-	-	3	2	4
mem16,reg16,immed8	-	-	7	3	6
mem32,reg32,immed8	-	-	7	3	6
reg16,reg16,CL	-	-	3	3	3
reg32,reg32,CL	-	-	3	3	3
mem16,reg16,CL	-	-	7	4	5
mem32,reg32,CL	-	-	7	4	5

0F A4 SHLD r/m16,r16,imm8
0F A5 SHLD r/m16,r16,CL
0F A4 SHLD r/m32,r32,imm8
0F A5 SHLD r/m32,r32,CL
0F AC SHRD r/m16,r16,imm8
0F AD SHRD r/m16,r16,CL
0F AC SHRD r/m32,r32,imm8
0F AD SHRD r/m32,r32,CL

SLDT - Store Local Descriptor Table (286+ privileged)

Usage: SLDT dest

Modifies flags: none

Stores the Local Descriptor Table (LDT) Register into the specified operand.

	Clocks				Size
Operands	808x	286	386	486	Bytes
reg16	-	2	2	2	3
mem16	-	2	2	3	5

0F 00 /0 SLDT r/m16 Stores segment selector from LDTR in r/m16

0F 00 /0 SLDT r/m32 Store segment selector from LDTR in low-order 16 bits of

SMSW - Store Machine Status Word (286+ privileged)

Usage: SMSW dest
Modifies flags: none
Store Machine Status Word (MSW) into "dest".

		Clocks			Size
Operands	808x	286	386	486	Bytes
reg16	-	2	10	2	3
mem16	-	3	3	3	5

0F 01 /4 SMSW r/m16 Store machine status word to r/m16
0F 01 /4 SMSW r32/m16 Store machine status word in low-order 16 bits of r32/m16;

STC - Set Carry

Usage: STC
Modifies flags: CF
Sets the Carry Flag to 1.

	Clocks				Size
Operands	808x	286	386	486	Bytes
none	2	2	2	2	1

F9 STC Set CF flag

STD - Set Direction Flag

Usage: STD
Modifies flags: DF
Sets the Direction Flag to 1 causing string instructions to auto-decrement SI and DI instead of auto-increment.

	Clocks				Size
Operands	808x	286	386	486	Bytes
none	2	2	2	2	1

FD STD Set DF flag

STI - Set Interrupt Flag (Enable Interrupts)

Usage: STI
Modifies flags: IF
Sets the Interrupt Flag to 1, which enables recognition of all hardware interrupts. If an interrupt is generated by a hardware device, an End of Interrupt (EOI) must also be issued to enable other hardware interrupts of the same or lower priority.

	Clocks				Size
Operands	808x	286	386	486	Bytes
none	2	2	2	5	1

FB STI Set interrupt flag; external, maskable interrupts enabled at the end of the next instruction

STOS - Store String (Byte, Word or Doubleword)

Usage: STOS dest
        STOSB
        STOSW
        STOSD

Modifies flags: None

Stores value in accumulator to location at ES:(E)DI (even if operand is given). (E)DI is incremented/decremented based on the size of the operand (or instruction format) and the state of the Direction Flag. Use with REP prefixes.

	Clocks				Size
Operands	808x	286	386	486	Bytes
dest	11	3	4	5	1 (W88=15)

AA STOS m8 Store AL at address ES:(E)DI
AB STOS m16 Store AX at address ES:(E)DI
AB STOS m32 Store EAX at address ES:(E)DI
AA STOSB Store AL at address ES:(E)DI
AB STOSW Store AX at address ES:(E)DI
AB STOSD Store EAX at address ES:(E)DI

STR - Store Task Register (286+ privileged)

Usage: STR dest

Modifies flags: None

Stores the current Task Register to the specified operand.

	Clocks				Size
Operands	808x	286	386	486	Bytes
reg16	-	2	2	2	3
mem16	-	3	2	3	5

0F 00 /1 STR r/m16 Stores segment selector from TR in r/m16

SUB - Subtract

Usage: SUB dest,src
Modifies flags: AF CF OF PF SF ZF
The source is subtracted from the destination and the result is stored in the destination.

	Clocks				Size	
Operands	808x	286	386	486	Bytes	
reg,reg	3	2	2	1	2	
mem,reg	16+EA	7	6	3	2-4	(W88=24+EA)
reg,mem	9+EA	7	7	2	2-4	(W88=13+EA)
reg,immed	4	3	2	1	3-4	
mem,immed	17+EA	7	7	3	3-6	(W88=25+EA)
accum,immed	4	3	2	1	2-3	

2C ib SUB AL, imm8 Subtract imm8 from AL
2D iw SUB AX, imm16 Subtract imm16 from AX
2D id SUB EAX, imm32 Subtract imm32 from EAX
80 /5 ib SUB r/m8,imm8 Subtract imm8 from r/m8
81 /5 iw SUB r/m16,imm16 Subtract imm16 from r/m16
81 /5 id SUB r/m32,imm32 Subtract imm32 from r/m32
83 /5 ib SUB r/m16,imm8 Subtract sign-extended imm8 from r/m16
83 /5 ib SUB r/m32,imm8 Subtract sign-extended imm8 from r/m32
28 / r SUB r/m8,r8 Subtract r8 from r/m8
29 / r SUB r/m16,r16 Subtract r16 from r/m16
29 / r SUB r/m32,r32 Subtract r32 from r/m32
2A / r SUB r8,r/m8 Subtract r/m8 from r8
2B / r SUB r16,r/m16 Subtract r/m16 from r16
2B / r SUB r32,r/m32 Subtract r/m32 from r32

TEST - Test For Bit Pattern

Usage: TEST dest,src

Modifies flags: CF OF PF SF ZF (AF undefined)

Performs a logical AND of the two operands updating the flags register without saving the result.

	Clocks				Size	
Operands	808x	286	386	486	Bytes	
reg,reg	3	2	1	1	2	
reg,mem	9+EA	6	5	1	2-4	(W88=13+EA)
mem,reg	9+EA	6	5	2	2-4	(W88=13+EA)
reg,immed	5	3	2	1	3-4	
mem,immed	11+EA	6	5	2	3-6	
accum,immed	4	3	2	1	2-3	

A8 ib TEST AL, imm8 AND imm8 with AL; set SF, ZF, PF according to result

A9 iw TEST AX, imm16 AND imm16 with AX; set SF, ZF, PF according to result

A9 id TEST EAX, imm32 AND imm32 with EAX; set SF, ZF, PF according to result

F6 /0 ib TEST r/m8,imm8 AND imm8 with r/m8; set SF, ZF, PF according to result

F7 /0 iw TEST r/m16,imm16 AND imm16 with r/m16; set SF, ZF, PF according to result

F7 /0 id TEST r/m32,imm32 AND imm32 with r/m32; set SF, ZF, PF according to result

84 / r TEST r/m8,r8 AND r8 with r/m8; set SF, ZF, PF according to result

85 / r TEST r/m16,r16 AND r16 with r/m16; set SF, ZF, PF according to result

85 / r TEST r/m32,r32 AND r32 with r/m32; set SF, ZF, PF according to result

VERR - Verify Read (286+ protected)

Usage: VERR src
Modifies flags: ZF
Verifies the specified segment selector is valid and is readable at the current privilege level. If the segment is readable, the Zero Flag is set, otherwise it is cleared.

		Clocks			Size
Operands	808x	286	386	486	Bytes
reg16	-	14	10	11	3
mem16	-	16	11	11	5

0F 00 /4 VERR r/m16 Set ZF=1 if segment specified with r/m16 can be read

VERW - Verify Write (286+ protected)

Usage: VERW src
Modifies flags: ZF
Verifies the specified segment selector is valid and is ratable
at the current privilege level. If the segment is writable,
the Zero Flag is set, otherwise it is cleared.

		Clocks			Size
Operands	808x	286	386	486	Bytes
reg16	-	14	15	11	3
mem16	-	16	16	11	5

0F 00 /5 VERW r/m16 Set ZF=1 if segment specified with r/m16 can be written

WAIT/FWAIT - Event Wait

Usage: WAIT
FWAIT

Modifies flags: None

CPU enters wait state until the coprocessor signals it has finished its operation. This instruction is used to prevent the CPU from accessing memory that may be temporarily in use by the coprocessor. WAIT and FWAIT are identical.

	Clocks				Size
Operands	808x	286	386	486	Bytes
none	4	3	6+	1-3	1

9B WAIT Check pending unmasked floating-point exceptions.
9B FWAIT Check pending unmasked floating-point exceptions.

WBINVD - Write-Back and Invalidate Cache (486+)

Usage: WBINVD
Modifies flags: None
Flushes internal cache, then signals the external cache to write back current data followed by a signal to flush the external cache.

	Clocks			Size
Operands	808x	286	386	486	Bytes
none	-	-	-	5	2

0F 09 WBINVD Write back and flush Internal caches; initiate writing-back

XCHG - Exchange

Usage: XCHG dest,src
Modifies flags: None
Exchanges contents of source and destination.

	Clocks				Size	
Operands	808x	286	386	486	Bytes	
reg,reg	4	3	3	3	2	
mem,reg	17+EA	5	5	5	2-4	(W88=25+EA)
reg,mem	17+EA	5	5	3	2-4	(W88=25+EA)
accum,reg	3	3	3	3	1	
reg,accum	3	3	3	3	1	

90+ rw XCHG AX, r16 Exchange r16 with AX
90+ rw XCHG r16,AX Exchange AX with r16
90+ rd XCHG EAX, r32 Exchange r32 with EAX
90+ rd XCHG r32,EAX Exchange EAX with r32
86 / r XCHG r/m8,r8 Exchange r8 (byte register) with byte from r/m8
86 / r XCHG r8,r/m8 Exchange byte from r/m8 with r8 (byte register)
87 / r XCHG r/m16,r16 Exchange r16 with word from r/m16
87 / r XCHG r16,r/m16 Exchange word from r/m16 with r16
87 / r XCHG r/m32,r32 Exchange r32 with doubleword from r/m32
87 / r XCHG r32,r/m32 Exchange doubleword from r/m32 with r32

XLAT/XLATB - Translate

Usage: XLAT translation-table
XLATB (masm 5.x)

Modifies flags: None

Replaces the byte in AL with byte from a user table addressed by BX. The original value of AL is the index into the translate table.

The best way to discripe this is MOV AL,[BX+AL]

	Clocks				Size
Operands	808x	286	386	486	Bytes
table offset	11	5	5	4	1

D7 XLAT m8 Set AL to memory byte DS:[(E)BX + unsigned AL]
D7 XLATB Set AL to memory byte DS:[(E)BX + unsigned AL]

XOR - Exclusive OR

Usage: XOR dest,src

Modifies flags: CF OF PF SF ZF (AF undefined)

Performs a bitwise exclusive OR of the operands and returns the result in the destination.

	Clocks				Size	
Operands	808x	286	386	486	Bytes	
reg,reg	3	2	2	1	2	
mem,reg	16+EA	7	6	3	2-4	(W88=24+EA)
reg,mem	9+EA	7	7	2	2-4	(W88=13+EA)
reg,immed	4	3	2	1	3-4	
mem,immed	17+EA	7	7	3	3-6	(W88=25+EA)
accum,immed	4	3	2	1	2-3	

34 ib XOR AL, imm8 AL XOR imm8
35 iw XOR AX, imm16 AX XOR imm16
35 id XOR EAX, imm32 EAX XOR imm32
80 /6 ib XOR r/m8,imm8 r/m8 XOR imm8
81 /6 iw XOR r/m16,imm16 r/m16 XOR imm16
81 /6 id XOR r/m32,imm32 r/m32 XOR imm32
83 /6 ib XOR r/m16,imm8 r/m16 XOR imm8 (sign-extended)
83 /6 ib XOR r/m32,imm8 r/m32 XOR imm8 (sign-extended)
30 / r XOR r/m8,r8 r/m8 XOR r8
31 / r XOR r/m16,r16 r/m16 XOR r16
31 / r XOR r/m32,r32 r/m32 XOR r32
32 / r XOR r8,r/m8 r8 XOR r/m8
33 / r XOR r16,r/m16 r8 XOR r/m8
33 / r XOR r32,r/m32 r8 XOR r/m8

Bare Hex Opcodes And Mnemonics

This list include opcodes that are not in the main listing.

NOTE : There are redundancies in the Intel instruction set which show in this list as duplication of both hex opcodes and mnemonics.

More detailed reference can be found in the Intel Architecture Software Developer's Manual Volume 2 : Instruction Set Reference found in the file "24319101.PDF" from Intel.

```
00 ADD
01 ADD
02 ADD
03 ADD
04 ADD
05 ADD
06 PUSH
07 POP
08 OR
09 OR
0A OR
0B OR
0C OR
0D OR
0E PUSH
0F 00 LLDT
0F 00 LTR
0F 00 SLDT
0F 00 STR
0F 00 VERR
0F 00 VERW
0F 01 INVLPG
0F 01 LGDT
0F 01 LIDT
0F 01 LMSW
0F 01 SGDT
0F 01 SIDT
0F 01 SMSW
0F 01 SMSW
0F 02 LAR
0F 03 LSL
0F 08 INVD
0F 09 WBINVD
0F 0B UD2
0F 20 MOV
0F 21 MOV
0F 22 MOV
```

0F 23 MOV
0F 30 WRMSR
0F 31 RDTSC
0F 32 RDMSR
0F 33 RDPMC
0F 40 CMOVO
0F 41 CMOVNO
0F 42 CMOVB
0F 42 CMOVC
0F 42 CMOVNAE
0F 43 CMOVAE
0F 43 CMOVNB
0F 43 CMOVNC
0F 44 CMOVE
0F 44 CMOVZ
0F 45 CMOVNE
0F 45 CMOVNZ
0F 46 CMOVBE
0F 46 CMOVNA
0F 47 CMOVA
0F 47 CMOVNBE
0F 48 CMOVS
0F 49 CMOVNS
0F 4A CMOVP
0F 4A CMOVPE
0F 4B CMOVNP
0F 4B CMOVPO
0F 4C CMOVL
0F 4C CMOVNGE
0F 4D CMOVGE
0F 4D CMOVNL
0F 4E CMOVLE
0F 4E CMOVNG
0F 4F CMOVG
0F 4F CMOVNLE
0F 60 PUNPCKLBW
0F 61 PUNPCKLWD
0F 62 PUNPCKLDQ
0F 63 PACKSSWB
0F 64 PCMPGTB
0F 65 PCMPGTW
0F 66 PCMPGTD
0F 67 PACKUSWB
0F 68 PUNPCKHBW
0F 69 PUNPCKHWD
0F 6A PUNPCKHDQ
0F 6B PACKSSDW

0F 6E MOVD
0F 6F MOVQ
0F 71 PSLLW
0F 71 PSRAW
0F 71 PSRLW
0F 72 PSLLD
0F 72 PSRAD
0F 72 PSRLD
0F 73 PSLLQ
0F 73 PSRLQ
0F 74 PCMPEQB
0F 75 PCMPEQW
0F 76 PCMPEQD
0F 77 EMMS
0F 7E MOVD
0F 7F MOVQ
0F 80 JO
0F 81 JNO
0F 82 JB
0F 82 JC
0F 82 JNAE
0F 83 JAE
0F 83 JNB
0F 83 JNC
0F 84 JE
0F 84 JZ
0F 85 JNE
0F 85 JNZ
0F 86 JBE
0F 86 JNA
0F 87 JA
0F 87 JNBE
0F 88 JS
0F 89 JNS
0F 8A JP
0F 8A JPE
0F 8B JNP
0F 8B JPO
0F 8C JL
0F 8C JNGE
0F 8D JGE
0F 8D JNL
0F 8E JLE
0F 8E JNG
0F 8F JG
0F 8F JNLE
0F 90 SETO

0F 91 SETNO
0F 92 SETB
0F 92 SETC
0F 92 SETNAE
0F 93 SETAE
0F 93 SETNB
0F 93 SETNC
0F 94 SETE
0F 94 SETZ
0F 95 SETNE
0F 95 SETNZ
0F 96 SETBE
0F 96 SETNA
0F 97 SETA
0F 97 SETNBE
0F 98 SETS
0F 99 SETNS
0F 9A SETP
0F 9A SETPE
0F 9B SETNP
0F 9B SETPO
0F 9C SETL
0F 9C SETNGE
0F 9D SETGE
0F 9D SETNL
0F 9E SETLE
0F 9E SETNG
0F 9F SETG
0F 9F SETNLE
0F A0 PUSH
0F A1 POP
0F A3 BT
0F A4 SHLD
0F A5 SHLD
0F A8 PUSH
0F A9 POP
0F AA RSM
0F AB BTS
0F AC SHRD
0F AD SHRD
0F AF IMUL
0F B0 CMPXCHG
0F B1 CMPXCHG
0F B2 LSS
0F B3 BTR
0F B4 LFS
0F B5 LGS

0F B6 MOVZX
0F B7 MOVZX
0F BA BT
0F BA BTC
0F BA BTR
0F BA BTS
0F BB BTC
0F BC BSF
0F BD BSR
0F BE MOVSX
0F BF MOVSX
0F C0 XADD
0F C1 XADD
0F C7 CMPXCHG8B
0F C8 BSWAP
0F D1 PSRLW
0F D2 PSRLD
0F D3 PSRLQ
0F D8 PSUBUSB
0F D9 PSUBUSW
0F DB PAND
0F DC PADDUSB
0F DD PADDUSW
0F DF PANDN
0F E1 PSRAW
0F E2 PSRAD
0F E5 PMULHW
0F E8 PSUBSB
0F E9 PSUBSW
0F EB POR
0F EC PADDSB
0F ED PADDSW
0F EF PXOR
0F F1 PSLLW
0F F2 PSLLD
0F F3 PSLLQ
0F F5 PMADDWD
0F F8 PSUBB
0F F9 PSUBW
0F FA PSUBD
0F FC PADDB
0F FD PADDW
0F FE PADDD
10 ADC
11 ADC
12 ADC
13 ADC

14 ADC
15 ADC
16 PUSH
17 POP
18 SBB
19 SBB
1A SBB
1B SBB
1C SBB
1D SBB
1E PUSH
1F POP
20 AND
21 AND
22 AND
23 AND
24 AND
25 AND
27 DAA
28 SUB
29 SUB
2A SUB
2B SUB
2C SUB
2D SUB
2F DAS
30 XOR
31 XOR
32 XOR
33 XOR
34 XOR
35 XOR
37 AAA
38 CMP
39 CMP
3A CMP
3B CMP
3C CMP
3D CMP
3F AAS
40 INC
48 DEC
50 PUSH
58 POP
60 PUSHA
60 PUSHAD
61 POPA

61 POPAD
62 BOUND
63 ARPL
68 PUSH
69 IMUL
6A PUSH
6B IMUL
6C INS
6C INSB
6D INS
6D INSD
6D INSW
6E OUTS
6E OUTSB
6F OUTS
6F OUTSD
6F OUTSW
70 JO
71 JNO
72 JB
72 JC
72 JNAE
73 JAE
73 JNB
73 JNC
74 JE
74 JZ
75 JNE
75 JNZ
76 JBE
76 JNA
77 JA
77 JNBE
78 JS
79 JNS
7A JP
7A JPE
7B JNP
7B JPO
7C JL
7C JNGE
7D JGE
7D JNL
7E JLE
7E JNG
7F JG
7F JNLE

```
80 ADC
80 ADD
80 AND
80 CMP
80 OR
80 SBB
80 SUB
80 XOR
81 ADC
81 ADD
81 AND
81 CMP
81 OR
81 SBB
81 SUB
81 XOR
83 ADC
83 ADD
83 AND
83 CMP
83 OR
83 SBB
83 SUB
83 XOR
84 TEST
85 TEST
86 XCHG
87 XCHG
88 MOV
89 MOV
8A MOV
8B MOV
8B MOV
8C MOV
8D LEA
8E MOV
8F POP
90 NOP
90 XCHG
98 CBW
98 CWDE
99 CDQ
99 CWD
9A CALL
9A CALL
9B D9 FSTCW
9B D9 FSTENV
```

9B DB E2 FCLEX
9B DB E3 FINIT
9B DD FSAVE
9B DD FSTSW
9B DF E0 FSTSW
9B FWAIT
9B WAIT
9C PUSHF
9C PUSHFD
9D POPF
9D POPFD
9E SAHF
9F LAHF
A0 MOV
A1 MOV
A2 MOV
A3 MOV
A4 MOVS
A4 MOVSB
A5 MOVS
A5 MOVSD
A5 MOVSW
A6 CMPS
A6 CMPSB
A7 CMPS
A7 CMPSD
A7 CMPSW
A8 TEST
A9 TEST
AA STOS
AA STOSB
AB STOS
AB STOSD
AB STOSW
AC LODS
AC LODSB
AD LODS
AD LODSD
AD LODSW
AE SCAS
AE SCASB
AF SCAS
AF SCASD
AF SCASW
B0 MOV
B8 MOV
C0 RCL

C0 RCR
C0 ROL
C0 ROR
C0 SAL
C0 SAR
C0 SHL
C0 SHR
C1 RCL
C1 RCR
C1 ROL
C1 ROR
C1 SAL
C1 SAR
C1 SHL
C1 SHR
C2 RET
C3 RET
C4 LES
C5 LDS
C6 MOV
C7 MOV
C8 00 ENTER
C8 01 ENTER
C8 ENTER
C9 LEAVE
CA RET
CC INT
CD INT
CE INTO
CF IRET
CF IRETD
D0 RCL
D0 RCR
D0 ROL
D0 ROR
D0 SAL
D0 SAR
D0 SHL
D0 SHR
D1 RCL
D1 RCR
D1 ROL
D1 ROR
D1 SAL
D1 SAR
D1 SHL
D1 SHR

```
D2 RCL
D2 RCR
D2 ROL
D2 ROR
D2 SAL
D2 SAR
D2 SHL
D2 SHR
D3 RCL
D3 RCR
D3 ROL
D3 ROR
D3 SAL
D3 SAR
D3 SHL
D3 SHR
D4 (No mnemonic)
D4 0A AAM
D5 (No mnemonic)
D5 0A AAD
D7 XLAT
D7 XLATB
D8 C0 FADD
D8 C8 FMUL
D8 D0 FCOM
D8 D1 FCOM
D8 D8 FCOMP
D8 D9 FCOMP
D8 E0 FSUB
D8 E8 FSUBR
D8 F0 FDIV
D8 F8 FDIVR
D8 FADD
D8 FCOM
D8 FCOMP
D8 FDIV
D8 FDIVR
D8 FMUL
D8 FSUB
D8 FSUBR
D9 C0 FLD
D9 C8 FXCH
D9 C9 FXCH
D9 D0 FNOP
D9 E0 FCHS
D9 E1 FABS
D9 E4 FTST
```

```
D9 E8 FLD1
D9 E9 FLDL2T
D9 EA FLDL2E
D9 EB FLDPI
D9 EC FLDLG2
D9 ED FLDLN2
D9 EE FLDZ
D9 F0 F2XM1
D9 F2 FPTAN
D9 F3 FPATAN
D9 F4 FXTRACT
D9 F5 FPREM1
D9 F6 FDECSTP
D9 F7 FINCSTP
D9 F9 FYL2XP1
D9 FA FSQRT
D9 FB FSINCOS
D9 FC FRNDINT
D9 FD FSCALE
D9 FE FSIN
D9 FF FCOS
D9 FLD
D9 FLDCW
D9 FLDENV
D9 FNSTCW
D9 FNSTENV
D9 FST
D9 FSTP
DA C0 FCMOVB
DA C8 FCMOVE
DA D0 FCMOVBE
DA D8 FCMOVU
DA E9 FUCOMPP
DA FIADD
DA FICOM
DA FICOMP
DA FIDIV
DA FIDIVR
DA FIMUL
DA FISUB
DA FISUBR
DB C0 FCMOVNB
DB C8 FCMOVNE
DB D0 FCMOVNBE
DB D8 FCMOVNU
DB E2 FNCLEX
DB E3 FNINIT
```

```
DB E8 FUCOMI
DB F0 FCOMI
DB FILD
DB FIST
DB FISTP
DB FLD
DB FSTP
DC C0 FADD
DC C8 FMUL
DC E0 FSUBR
DC E8 FSUB
DC F0 FDIVR
DC F8 FDIV
DC FADD
DC FCOM
DC FCOMP
DC FDIV
DC FDIVR
DC FMUL
DC FSUB
DC FSUBR
DD C0 FFREE
DD D0 FST
DD D8 FSTP
DD E0 FUCOM
DD E1 FUCOM
DD E8 FUCOMP
DD E9 FUCOMP
DD FLD
DD FNSAVE
DD FNSTSW
DD FRSTOR
DD FST
DD FSTP
DE C0 FADDP
DE C1 FADDP
DE C8 FMULP
DE C9 FMULP
DE D9 FCOMPP
DE E0 FSUBRP
DE E1 FSUBRP
DE E8 FSUBP
DE E9 FSUBP
DE F0 FDIVRP
DE F1 FDIVRP
DE F8 FDIVP
DE F9 FDIVP
```

```
DE FIADD
DE FICOM
DE FICOMP
DE FIDIV
DE FIDIVR
DE FIMUL
DE FISUB
DE FISUBR
DF E0 FNSTSW
DF E8 FUCOMIP
DF F0 FCOMIP
DF FBLD
DF FBSTP
DF FILD
DF FIST
DF FISTP
E0 LOOPNE
E0 LOOPNZ
E1 LOOPE
E1 LOOPZ
E2 LOOP
E3 JCXZ
E3 JECXZ
E4 IN
E5 IN
E6 OUT
E7 OUT
E8 CALL
E9 JMP
EA JMP
EB JMP
EC IN
ED IN
EE OUT
EF OUT
F0 LOCK
F2 A6 REPNE
F2 A7 REPNE
F2 AE REPNE
F2 AF REPNE
F3 6C REP
F3 6D REP
F3 6E REP
F3 6F REP
F3 A4 REP
F3 A5 REP
F3 A6 REPE
```

F3 A7 REPE
F3 AA REP
F3 AB REP
F3 AC REP
F3 AD REP
F3 AE REPE
F3 AF REPE
F4 HLT
F5 CMC
F6 DIV
F6 IDIV
F6 IMUL
F6 MUL
F6 NEG
F6 NOT
F6 TEST
F7 DIV
F7 IDIV
F7 IMUL
F7 MUL
F7 NEG
F7 NOT
F7 TEST
F8 CLC
F9 STC
FA CLI
FC CLD
FD STD
FE DEC
FE INC
FF CALL
FF DEC
FF INC
FF JMP
FF PUSH