Intel 80386 Instruction Subset Compendium

Corso di Calcolatori Elettronici A.A. 2010/2011

Prof. Luigi Palopoli, Ing. Fabio Fassetti

1 DATA MOVEMENT INSTRUCTIONS

1.1 General-Purpose Data Movement Instructions

MOV – Move Data

Instruction	Clocks	Description
MOV r/m8,r8	2/2	Move byte register to r/m byte
MOV r/m16,r16	2/2	Move word register to r/m word
MOV r/m32,r32	2/2	Move dword register to r/m dword
MOV r8,r/m8	2/4	Move r/m byte to byte register
MOV r16,r/m16	2/4	Move r/m word to word register
MOV r32,r/m32	2/4	Move r/m dword to dword register
MOV reg8,imm8	2	Move immediate byte to register
MOV reg16,imm16	2	Move immediate word to register
MOV reg32,imm32	2	Move immediate dword to register
MOV r/m8,imm8	2/2	Move immediate byte to r/m byte
MOV r/m16,imm16	2/2	Move immediate word to r/m word
MOV r/m32,imm32	2/2	Move immediate dword to r/m dword

Operation:

DEST ← SRC

Flags Affected:

None

XCHG – Exchange Register/Memory with Register

Instruction	Clocks	Description
XCHG r/m8,r8	3	Exchange byte register with r/m byte
XCHG r8,r/m8	3/5	Exchange byte register with r/m byte
XCHG r/m16,r16	3	Exchange word register with r/m word
XCHG r16,r/m16	3/5	Exchange word register with r/m word
XCHG r/m32,r32	3	Exchange dword register with r/m dword
XCHG r32,r/m32	3/5	Exchange dword register with r/m dword

Operation:

temp ← DEST
DEST ← PROV
PROV ← temp

Flags Affected:

None

1.2 Stack Manipulation Instructions

PUSH – Push Operand onto the Stack

Instruction	Clocks	Description
PUSH r/m16	2/5	Push r/m word onto the top of stack
PUSH r/m32	2/5	Push r/m dword onto the top of stack
PUSH imm16	2	Push immediate word onto the top of stack
PUSH imm32	2	Push immediate dword onto the top of stack

Operation:

PUSH decrements the stack pointer by 2 if the operand-size attribute of the instruction is 16 bits; otherwise, it decrements the stack pointer by 4. PUSH then places the operand on the new top of stack, which is pointed to by the stack pointer.

Flags Affected:
None

POP – Pop a Word from the Stack

Instruction	Clocks	Description
POP r/m16	4/5	Pop top of stack into r/m word
POP r/m32	4/5	Pop top of stack into r/m dword

Operation:
POP replaces the previous contents of the memory or the register with the word or the double word on the top of the 80386 stack, addressed by SS:SP (address-size attribute of 16 bits) or SS:ESP (addres-size attribute of 32 bits). The stack pointer SP is incremented by 2 for an operand-size of 16 bits or by 4 for an operand-size of 32 bits. It then points to the new top of stack.

Flags Affected:
None

1.3 Type Conversion Instructions

MOVSX – Move with Sign-Extend

Instruction	Clocks	Description
MOVSX r16,r/m8	3/6	Move byte to word with sign-extend
MOVSX r32,r/m8	3/6	Move byte to dword, sign-extend
MOVSX r32,r/m16	3/6	Move word to dword, sign-extend

Operation:
DEST ← SignExtend(SRC)

Flags Affected:
None

MOVZX – Move with Zero-Extend

Instruction	Clocks	Description
MOVZX r16,r/m8	3/6	Move byte to word with zero-extend
MOVZX r32,r/m8	3/6	Move byte to dword, zero-extend
MOVZX r32,r/m16	3/6	Move word to dword, zero-extend

Operation:
DEST ← ZeroExtend(SRC)

Flags Affected:
None

CBW/CWDE – Convert Byte to Word/Convert Word to Doubleword

Instruction	Clocks	Description
CBW	3	AX ← sign-extend of AL
CWDE	3	EAX ← sign-extend of AX

Operation:
IF *OperandSize* = 16 *//instruction = CBW*
THEN
    AX ← SignExtend(AL);
ELSE *// OperandSize = 32, instruction = CWDE*
    EAX ← SignExtend(AX);
FI;

Flags Affected:
None

CWD/CDQ – Convert Word to Doubleword/Convert Doubleword to Quadword

Instruction	Clocks	Description
CWD	2	DX:AX ← sign-extend of AX
CDQ	2	EDX:EAX ← sign-extend of EAX

Operation:

IF $OperandSize = 16$ *//CWD instruction*
THEN
    IF AX < 0
    THEN
        DX ← 0FFFFH;
    ELSE
        DX ← 0;
    FI;
ELSE *//OperandSize = 32, CDQ instruction*
    IF EAX < 0
    THEN
        EDX ← 0FFFFFFFFH;
    ELSE
        EDX ← 0;
    FI;
FI;

Flags Affected:

None

2 BINARY ARITHMETIC INSTRUCTIONS

2.1 Addition and Subtraction Instructions

ADD – Add

Instruction	Clocks	Description
ADD r/m8,imm8	2/7	Add immediate byte to r/m byte
ADD r/m16,imm16	2/7	Add immediate word to r/m word
ADD r/m32,imm32	2/7	Add immediate dword to r/m dword
ADD r/m16,imm8	2/7	Add sign-extended immediate byte to r/m word
ADD r/m32,imm8	2/7	Add sign-extended immediate byte to r/m dword
ADD r/m8,r8	2/7	Add byte register to r/m byte
ADD r/m16,r16	2/7	Add word register to r/m word
ADD r/m32,r32	2/7	Add dword register to r/m dword
ADD r8,r/m8	2/6	Add r/m byte to byte register
ADD r16,r/m16	2/6	Add r/m word to word register
ADD r32,r/m32	2/6	Add r/m dword to dword register

Operation:

DEST ← DEST + SRC + CF

Flags Affected:

OF, SF, ZF, AF, CF, and PF

ADC – Add with Carry

Instruction	Clocks	Description
ADC r/m8,imm8	2/7	Add with carry immediate byte to r/m byte
ADC r/m16,imm16	2/7	Add with carry immediate word to r/m word
ADC r/m32,imm32	2/7	Add with CF immediate dword to r/m dword
ADC r/m16,imm8	2/7	Add with CF sign-extended immediate byte to r/m word
ADC r/m32,imm8	2/7	Add with CF sign-extended immediate byte into r/m dword
ADC r/m8,r8	2/7	Add with carry byte register to r/m byte
ADC r/m16,r16	2/7	Add with carry word register to r/m word
ADC r/m32,r32	2/7	Add with CF dword register to r/m dword
ADC r8,r/m8	2/6	Add with carry r/m byte to byte register
ADC r16,r/m16	2/6	Add with carry r/m word to word register
ADC r32,r/m32	2/6	Add with CF r/m dword to dword register

Operation:

DEST ← DEST + SRC + CF

Flags Affected:

OF, SF, ZF, AF, CF, and PF

SUB – Integer Subtraction

Instruction	Clocks	Description
SUB r/m8,imm8	2/7	Subtract immediate byte from r/m byte
SUB r/m16,imm16	2/7	Subtract immediate word from r/m word
SUB r/m32,imm32	2/7	Subtract immediate dword from r/m dword
SUB r/m16,imm8	2/7	Subtract sign-extended immediate byte from r/m word
SUB r/m32,imm8	2/7	Subtract sign-extended immediate byte from r/m dword
SUB r/m8,r8	2/6	Subtract byte register from r/m byte
SUB r/m16,r16	2/6	Subtract word register from r/m word
SUB r/m32,r32	2/6	Subtract dword register from r/m dword
SUB r8,r/m8	2/7	Subtract byte register from r/m byte
SUB r16,r/m16	2/7	Subtract word register from r/m word
SUB r32,r/m32	2/7	Subtract dword register from r/m dword

Operation:

IF SRC *is a byte and* DEST *is a word or dword*
THEN
    DEST = DEST − SignExtend(SRC)
ELSE
    DEST ← DEST − SRC
FI

Flags Affected:

OF, SF, ZF, AF, PF, and CF

SBB – Integer Subtraction with Borrow

Instruction	Clocks	Description
SBB r/m8,imm8	2/7	Subtract with borrow immediate byte from r/m byte
SBB r/m16,imm16	2/7	Subtract with borrow immediate from r/m word
SBB r/m32,imm32	2/7	Subtract with borrow immediate dword from r/m dword
SBB r/m16,imm8	2/7	Subtract with borrow sign-extended immediate byte from r/m word
SBB r/m32,imm8	2/7	Subtract with borrow sign-extended immediate byte from r/m dword
SBB r/m8,r8	2/6	Subtract with borrow byte register from r/m byte
SBB r/m16,r16	2/6	Subtract with borrow word register from r/m word
SBB r/m32,r32	2/6	Subtract with borrow dword from r/m dword
SBB r8,r/m8	2/7	Subtract with borrow byte register from r/m byte
SBB r16,r/m16	2/7	Subtract with borrow word register from r/m word
SBB r32,r/m32	2/7	Subtract with borrow dword register from r/m dword

Operation:

IF SRC *is a byte and* DEST *is a word or dword*

THEN
DEST = DEST − (SignExtend(SRC) + CF)
ELSE
DEST ← DEST − (SRC + CF)
FI

Flags Affected:
OF, SF, ZF, AF, PF, and CF

INC – Increment by 1

Instruction	Clocks	Description
INC r/m8	2/6	Increment r/m byte by 1
INC r/m16	2/6	Increment r/m word by 1
INC r/m32	2/6	Increment r/m dword by 1

Operation:
DEST ← DEST + 1

Flags Affected:
OF, SF, ZF, AF, and PF

DEC – Decrement by 1

Instruction	Clocks	Description
DEC r/m8	2/6	Decrement r/m byte by 1
DEC r/m16	2/6	Decrement r/m word by 1
DEC r/m32	2/6	Decrement r/m dword by 1

Operation:
DEST ← DEST − 1

Flags Affected:
OF, SF, ZF, AF, and PF

2.2 Comparison and Sign Change Instruction

CMP – Compare Two Operands

Instruction	Clocks	Description
CMP r/m8,imm8	2/5	Compare immediate byte to r/m byte
CMP r/m16,imm16	2/5	Compare immediate word to r/m word
CMP r/m32,imm32	2/5	Compare immediate dword to r/m dword
CMP r/m16,imm8	2/5	Compare sign extended immediate byte to r/m word
CMP r/m32,imm8	2/5	Compare sign extended immediate byte to r/m dword
CMP r/m8,r8	2/5	Compare byte register to r/m byte
CMP r/m16,r16	2/5	Compare word register to r/m word
CMP r/m32,r32	2/5	Compare dword register to r/m dword
CMP r8,r/m8	2/6	Compare r/m byte to byte register
CMP r16,r/m16	2/6	Compare r/m word to word register
CMP r32,r/m32	2/6	Compare r/m dword to dword register

Operation:
LeftSRC − SignExtend(RightSRC)
//CMP does not store a result; its purpose is to set the flags

Flags Affected:
OF, SF, ZF, AF, PF, and CF

NEG – Two's Complement Negation

Instruction	Clocks	Description
NEG r/m8	2/6	Two's complement negate r/m byte
NEG r/m16	2/6	Two's complement negate r/m word
NEG r/m32	2/6	Two's complement negate r/m dword

Operation:

IF r/m = 0
THEN
    CF ← 0
ELSE
    CF ← 1;
FI
r/m ← −r/m

Flags Affected:

CF is set to 1, unless the operand is zero, in which case CF is cleared to 0.
OF, SF, ZF, and PF

2.3 Multiplication Instructions

MUL – Unsigned Multiplication of AL or AX

Instruction	Clocks	Description
MUL AL,r/m8	9-14/12-17	Unsigned multiply (AX AL * r/m byte)
MUL AX,r/m16	9-22/12-25	Unsigned multiply (DX:AX AX * r/m word)
MUL EAX,r/m32	9-38/12-41	Unsigned multiply (EDX:EAX EAX * r/m dword)

Operation:

IF byte-size operation
THEN
    AX ← AL ∗ r/m8
ELSE *//word or doubleword operation*
    IF OperandSize = 16
    THEN
        DX:AX ← AX ∗ r/m16
    ELSE *//OperandSize = 32*
        EDX:EAX ← EAX ∗ r/m32
    FI
FI

Flags Affected:

CF and OF as follows

- A byte operand is multiplied by AL; the result is left in AX. The carry and overflow flags are set to 0 if AH is 0; otherwise, they are set to 1.
- A word operand is multiplied by AX; the result is left in DX:AX. DX contains the high-order 16 bits of the product. The carry and overflow flags are set to 0 if DX is 0; otherwise, they are set to 1.
- A doubleword operand is multiplied by EAX and the result is left in EDX:EAX. EDX contains the high-order 32 bits of the product. The carry and overflow flags are set to 0 if EDX is 0; otherwise, they are set to 1.

SF, ZF, AF and PF are undefined.

IMUL – Signed Multiply

Instruction	Clocks	Description
IMUL r/m8	9-14/12-17	AX ← AL * r/m byte
IMUL r/m16	9-22/12-25	DX:AX ← AX * r/m word
IMUL r/m32	9-38/12-41	EDX:EAX ← EAX * r/m dword
IMUL r16,r/m16	9-22/12-25	word register ← word register * r/m word
IMUL r32,r/m32	9-38/12-41	dword register ← dword register * r/m dword
IMUL r16,r/m16,imm8	9-14/12-17	word register ← r/m16 * sign-extended immediate byte
IMUL r32,r/m32,imm8	9-14/12-17	dword register ← r/m32 * sign-extended immediate byte
IMUL r16,imm8	9-14/12-17	word register ← word register * sign-extended immediate byte
IMUL r32,imm8	9-14/12-17	dword register ← dword register * sign-extended immediate byte
IMUL r16,r/m16,imm16	9-22/12-25	word register ← r/m16 * immediate word
IMUL r32,r/m32,imm32	9-38/12-41	dword register ← r/m32 * immediate dword
IMUL r16,imm16	9-22/12-25	word register ← r/m16 * immediate word
IMUL r32,imm32	9-38/12-41	dword register ← r/m32 * immediate dword

Operation:

result ← multiplicand * multiplier

Flags Affected:

IMUL clears the overflow and carry flags under the following conditions:

Instruction Form	Condition for Clearing CF and OF
r/m8	AL = sign-extend of AL to 16 bits
r/m16	AX = sign-extend of AX to 32 bits
r/m32	EDX:EAX = sign-extend of EAX to 32 bits
r16,r/m16	Result exactly fits within r16
r/32,r/m32	Result exactly fits within r32
r16,r/m16,imm16	Result exactly fits within r16
r32,r/m32,imm32	Result exactly fits within r32

SF, ZF, AF and PF are undefined.

2.4 Division Instructions

DIV – Unsigned Divide

Instruction	Clocks	Description
DIV r/m8	14/17	Unsigned divide AX by r/m byte (AL=Quo, AH=Rem)
DIV r/m16	22/25	Unsigned divide DX:AX by r/m word (AX=Quo, DX=Rem)
DIV r/m32	38/41	Unsigned divide EDX:EAX by r/m dword (EAX=Quo, EDX=Rem)

Operation:

```
temp ← dividend/divisor
IF   temp does not fit in quotient
THEN
     Interrupt 0
ELSE
     quotient ← temp
     remainder ← dividend MOD (r/m)
FI
```

Flags Affected:

OF, SF, ZF, AF, PF, CF are undefined

IDIV – Signed Divide

Instruction	Clocks	Description
IDIV r/m8	19	Signed divide AX by r/m byte (AL=Quo, AH=Rem)
IDIV r/m16	27	Signed divide DX:AX by EA word (AX=Quo, DX=Rem)
IDIV r/m32	43	Signed divide EDX:EAX by DWORD byte (EAX=Quo, EDX=Rem)

Operation:

```
temp ← dividend/divisor
IF   temp does not fit in quotient
THEN
     Interrupt 0
ELSE
     quotient ← temp
     remainder ← dividend MOD (r/m)
FI
```

Flags Affected:

OF, SF, ZF, AF, PF, CF are undefined

3 LOGICAL INSTRUCTIONS

3.1 Boolean Operation Instructions

AND

Instruction	Clocks	Description
AND r/m8,imm8	2/7	AND immediate byte to r/m byte
AND r/m16,imm16	2/7	AND immediate word to r/m word
AND r/m32,imm32	2/7	AND immediate dword to r/m dword
AND r/m16,imm8	2/7	AND sign-extended immediate byte with r/m word
AND r/m32,imm8	2/7	AND sign-extended immediate byte with r/m dword
AND r/m8,r8	2/7	AND byte register to r/m byte
AND r/m16,r16	2/7	AND word register to r/m word
AND r/m32,r32	2/7	AND dword register to r/m dword
AND r8,r/m8	2/6	AND r/m byte to byte register
AND r16,r/m16	2/6	AND r/m word to word register
AND r32,r/m32	2/6	AND r/m dword to dword register

Operation:

DEST ← DEST AND SRC
CF ← 0
OF ← 0

Flags Affected:

CF = 0, OF = 0; PF, SF, and ZF

OR

Instruction	Clocks	Description
OR r/m8,imm8	2/7	OR immediate byte to r/m byte
OR r/m16,imm16	2/7	OR immediate word to r/m word
OR r/m32,imm32	2/7	OR immediate dword to r/m dword
OR r/m16,imm8	2/7	OR sign-extended immediate byte with r/m word
OR r/m32,imm8	2/7	OR sign-extended immediate byte with r/m dword
OR r/m8,r8	2/6	OR byte register to r/m byte
OR r/m16,r16	2/6	OR word register to r/m word
OR r/m32,r32	2/6	OR dword register to r/m dword
OR r8,r/m8	2/7	OR byte register to r/m byte
OR r16,r/m16	2/7	OR word register to r/m word
OR r32,r/m32	2/7	OR dword register to r/m dword

Operation:

DEST ← DEST AND SRC
CF ← 0
OF ← 0

Flags Affected:

CF = 0, OF = 0; PF, SF, and ZF

XOR

Instruction	Clocks	Description
XOR r/m8,imm8	2/7	Exclusive-OR immediate byte to r/m byte
XOR r/m16,imm16	2/7	Exclusive-OR immediate word to r/m word
XOR r/m32,imm32	2/7	Exclusive-OR immediate dword to r/m dword
XOR r/m16,imm8	2/7	XOR sign-extended immediate byte with r/m word
XOR r/m32,imm8	2/7	XOR sign-extended immediate byte with r/m dword
XOR r/m8,r8	2/6	Exclusive-OR byte register to r/m byte
XOR r/m16,r16	2/6	Exclusive-OR word register to r/m word
XOR r/m32,r32	2/6	Exclusive-OR dword register to r/m dword
XOR r8,r/m8	2/7	Exclusive-OR byte register to r/m byte
XOR r16,r/m16	2/7	Exclusive-OR word register to r/m word
XOR r32,r/m32	2/7	Exclusive-OR dword register to r/m dword

Operation:

DEST ← LeftSRC XOR RightSRC
CF ← 0
OF ← 0

Flags Affected:
CF = 0, OF = 0; SF, ZF, and PF. AF is undefined

NOT

Instruction	Clocks	Description
NOT r/m8	2/6	Reverse each bit of r/m byte
NOT r/m16	2/6	Reverse each bit of r/m word
NOT r/m32	2/6	Reverse each bit of r/m dword

Operation:
r/m ← NOT r/m

Flags Affected:
None

3.2 Shift Instructions

SAL/SAR/SHL/SHR – Shift Instructions

Instruction	Clocks	Description
SAL r/m8,1	3/7	Multiply r/m byte by 2, once
SAL r/m8,CL	3/7	Multiply r/m byte by 2, CL times
SAL r/m8,imm8	3/7	Multiply r/m byte by 2, imm8 times
SAL r/m16,1	3/7	Multiply r/m word by 2, once
SAL r/m16,CL	3/7	Multiply r/m word by 2, CL times
SAL r/m16,imm8	3/7	Multiply r/m word by 2, imm8 times
SAL r/m32,1	3/7	Multiply r/m dword by 2, once
SAL r/m32,CL	3/7	Multiply r/m dword by 2, CL times
SAL r/m32,imm8	3/7	Multiply r/m dword by 2, imm8 times
SAR r/m8,1	3/7	Signed divideˆ(1) r/m byte by 2, once
SAR r/m8,CL	3/7	Signed divideˆ(1) r/m byte by 2, CL times
SAR r/m8,imm8	3/7	Signed divideˆ(1) r/m byte by 2, imm8 times
SAR r/m16,1	3/7	Signed divideˆ(1) r/m word by 2, once
SAR r/m16,CL	3/7	Signed divideˆ(1) r/m word by 2, CL times
SAR r/m16,imm8	3/7	Signed divideˆ(1) r/m word by 2, imm8 times
SAR r/m32,1	3/7	Signed divideˆ(1) r/m dword by 2, once
SAR r/m32,CL	3/7	Signed divideˆ(1) r/m dword by 2, CL times
SAR r/m32,imm8	3/7	Signed divideˆ(1) r/m dword by 2, imm8 times
SHL r/m8,1	3/7	Multiply r/m byte by 2, once
SHL r/m8,CL	3/7	Multiply r/m byte by 2, CL times
SHL r/m8,imm8	3/7	Multiply r/m byte by 2, imm8 times
SHL r/m16,1	3/7	Multiply r/m word by 2, once
SHL r/m16,CL	3/7	Multiply r/m word by 2, CL times
SHL r/m16,imm8	3/7	Multiply r/m word by 2, imm8 times
SHL r/m32,1	3/7	Multiply r/m dword by 2, once
SHL r/m32,CL	3/7	Multiply r/m dword by 2, CL times
SHL r/m32,imm8	3/7	Multiply r/m dword by 2, imm8 times
SHR r/m8,1	3/7	Unsigned divide r/m byte by 2, once
SHR r/m8,CL	3/7	Unsigned divide r/m byte by 2, CL times
SHR r/m8,imm8	3/7	Unsigned divide r/m byte by 2, imm8 times
SHR r/m16,1	3/7	Unsigned divide r/m word by 2, once
SHR r/m16,CL	3/7	Unsigned divide r/m word by 2, CL times
SHR r/m16,imm8	3/7	Unsigned divide r/m word by 2, imm8 times
SHR r/m32,1	3/7	Unsigned divide r/m dword by 2, once
SHR r/m32,CL	3/7	Unsigned divide r/m dword by 2, CL times
SHR r/m32,imm8	3/7	Unsigned divide r/m dword by 2, imm8 times

Operation:

Flags Affected:

OF is set only if the single-shift forms of the instructions are used. For left shifts, OF is set to 0 if the high bit of the answer is the same as the result of the carry flag (i.e., the top two bits of the original operand were the same); OF is set to 1 if they are different. For SAR, OF is set to 0 for all single shifts. For SHR, OF is set to the high-order bit of the original operand. OF is undefined for multiple shifts. CF receives a copy of the bit that was shifted from one end to the other. ZF, PF and SF

3.3 Double-Shift Instructions

SHRD – Double Precision Shift Right

Instruction	Clocks	Description
SHRD r/m16,r16,imm8	3/7	r/m16 gets SHR of r/m16 concatenated with r16
SHRD r/m32,r32,imm8	3/7	r/m32 gets SHR of r/m32 with r32
SHRD r/m16,r16,CL	3/7	r/m16 gets SHR of r/m16 concatenated with r16
SHRD r/m32,r32,CL	3/7	r/m32 gets SHR of r/m32 concatenated with r32

Operation:

SHRD shifts the first operand provided by the r/m field to the right as many bits as specified by the count operand. The second operand (r16 or r32) provides the bits to shift in from the left (starting with bit 31). The result is stored back into the r/m operand. The register remains unaltered. The count operand is provided by either an immediate byte or the contents of the CL register. These operands are taken MODULO 32 to provide a number between 0 and 31 by which to shift. Because the bits to shift are provided by the specified register, the operation is useful for multi-precision shifts (64 bits or more).

Flags Affected:

CF is set to the value of the last bit shifted out. OF and AF are left undefined. SF, ZF and PF.

SHLD – Double Precision Shift Left

Instruction	Clocks	Description
SHLD r/m16,r16,imm8	3/7	r/m16 gets SHL of r/m16 concatenated with r16
SHLD r/m32,r32,imm8	3/7	r/m32 gets SHL of r/m32 concatenated with r32
SHLD r/m16,r16,CL	3/7	r/m16 gets SHL of r/m16 concatenated with r16
SHLD r/m32,r32,CL	3/7	r/m32 gets SHL of r/m32 concatenated with r32

Operation:

SHLD shifts the first operand provided by the r/m field to the left as many bits as specified by the count operand. The second operand (r16 or r32) provides the bits to shift in from the right (starting with bit 0). The result is stored back into the r/m operand. The register remains unaltered. The count operand is provided by either an immediate byte or the contents of the CL register. These operands are taken MODULO 32 to provide a number between 0 and 31 by which to shift. Because the bits to shift are provided by the specified registers, the operation is useful for multiprecision shifts (64 bits or more).

Flags Affected:

CF is set to the value of the last bit shifted out. OF and AF are left undefined. SF, ZF and PF.

3.4 Rotate Instructions

ROL/ROR – Rotate

Instruction	Clocks	Description
ROL r/m8,1	3/7	Rotate 8 bits r/m byte left once
ROL r/m8,CL	3/7	Rotate 8 bits r/m byte left CL times
ROL r/m8,imm8	3/7	Rotate 8 bits r/m byte left imm8 times
ROL r/m16,1	3/7	Rotate 16 bits r/m word left once
ROL r/m16,CL	3/7	Rotate 16 bits r/m word left CL times
ROL r/m16,imm8	3/7	Rotate 16 bits r/m word left imm8 times
ROL r/m32,1	3/7	Rotate 32 bits r/m dword left once
ROL r/m32,CL	3/7	Rotate 32 bits r/m dword left CL times
ROL r/m32,imm8	3/7	Rotate 32 bits r/m dword left imm8 times
ROR r/m8,1	3/7	Rotate 8 bits r/m byte right once
ROR r/m8,CL	3/7	Rotate 8 bits r/m byte right CL times
ROR r/m8,imm8	3/7	Rotate 8 bits r/m word right imm8 times
ROR r/m16,1	3/7	Rotate 16 bits r/m word right once
ROR r/m16,CL	3/7	Rotate 16 bits r/m word right CL times
ROR r/m16,imm8	3/7	Rotate 16 bits r/m word right imm8 times
ROR r/m32,1	3/7	Rotate 32 bits r/m dword right once
ROR r/m32,CL	3/7	Rotate 32 bits r/m dword right CL times
ROR r/m32,imm8	3/7	Rotate 32 bits r/m dword right imm8 times

Operation:

The rotate is repeated the number of times indicated by the second operand, which is either an immediate number or the contents of the CL register. To reduce the maximum instruction execution time, the 80386 does not allow rotation counts greater than 31. If a rotation count greater than 31 is attempted, only the bottom five bits of the rotation are used.

Flags Affected:

OF only for single rotates; OF is undefined for multi-bit rotates; for left shifts/rotates, the CF bit after the shift is XORed with the high-order result bit. For right shifts/rotates, the high-order two bits of the result are XORed to get OF. CF receives a copy of the bit that was shifted from one end to the other.

4 CONTROL TRANSFER INSTRUCTIONS

4.1 Unconditional Transfer Instructions

JMP – Jump

Instruction	Clocks	Description
JMP rel8	7	Jump near
JMP rel16	7	Jump near
JMP rel32	7	Jump near

Operation:

EIP ← EIP + rel8/16/32

Flags Affected:

None

CALL – Call Procedure

Instruction	Clocks	Description
CALL rel16	7	Call near
CALL rel32	7	Call near

Operation:

Push(EIP)
EIP ← EIP + rel16/32

Flags Affected:

None

RET – Return from Procedure

Instruction	Clocks	Description
RET	10	Return (near) to caller
RET imm16	10	Return (near), pop imm16 bytes of parameters

Operation:

EIP ← Pop()
IF *instruction has immediate operand* THEN
  ESP ← ESP + imm16;
FI;

Flags Affected:

None

4.2 Conditional Transfer Instructions

Jcc – Jump if Condition is Met

Instruction		Clocks	Description
JZ	rel8/16/32	7, 3	Jump near if ZF = 1
JNZ	rel8/16/32	7, 3	Jump near if ZF = 0
JS	rel8/16/32	7, 3	Jump near if SF = 1
JNS	rel8/16/32	7, 3	Jump near if SF = 0
JO	rel8/16/32	7, 3	Jump near if OF = 1
JNO	rel8/16/32	7, 3	Jump near if OF = 0
JP	rel8/16/32	7, 3	Jump near if PF = 1
JNP	rel8/16/32	7, 3	Jump near if PF = 0
JE	rel8/16/32	7, 3	Jump near if equal (ZF = 1)
JNE	rel8/16/32	7, 3	Jump near if not equal (ZF = 0)
JG	rel8/16/32	7, 3	Jump near if greater (ZF = 0 and SF = OF)
JNG	rel8/16/32	7, 3	Jump near if not greater (ZF = 1 or SF ≠ OF)
JL	rel8/16/32	7, 3	Jump near if less (SF ≠ OF)
JNL	rel8/16/32	7, 3	Jump near if not less (SF = OF)
JGE	rel8/16/32	7, 3	Jump near if greater or equal (SF = OF)
JNGE	rel8/16/32	7, 3	Jump near if not greater or equal (SF ≠ OF)
JLE	rel8/16/32	7, 3	Jump near if less or equal (ZF = 1 and SF ≠ OF)
JNLE	rel8/16/32	7, 3	Jump near if not less or equal (ZF = 0 and SF = OF)

NOTES:
The first clock count is for the true condition (branch taken); the second clock count is for the false condition (branch not taken).

Operation:

IF *condizione* THEN
  EIP ← EIP + rel8/16/32
FI

Flags Affected:

None