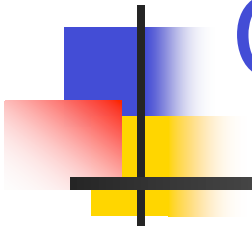
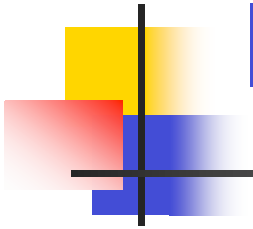


Chapter 5

Real-Mode 80386DX Microprocessor Programming 1

Part 2



Introduction

5.2 Data-Transfer Instructions
5.3 Arithmetic Instructions
5.4 Logic Instructions
5.5 Shift Instructions
5.6 Rotate Instructions
5.7 Bit Test and Bit Scan Instructions

Logic Instructions

- AND → Logical AND
- OR → Logical inclusive-OR
- XOR → Logical exclusive-OR
- NOT → Logical NOT

- Logical AND Instruction—AND
  - AND format and operation:

    AND D,S

    (S) AND (D) → (D)
    - Logical AND of values in two registers

      AND AX,BX

      (AX) AND (BX) → (AX)
    - Logical AND of a value in memory and a

      AND [DI],AX

      (DS:DI) AND (AX) → (DS:DI)
    - Logical AND of an immediate operand with a value in a register or memory

      AND AX,100H

      (AX) AND IMM16 → (AX)
- Flags updated based on result
  - CF, OF, SF, ZF, PF
  - AF undefined

Mnemonic	Meaning	Format	Operation	Flags Affected
AND	Logical AND	AND D,S	(S) · (D) → (D)	OF, SF, ZF, PF, CF AF undefined
OR	Logical Inclusive-OR	OR D,S	(S) + (D) → (D)	OF, SF, ZF, PF, CF AF undefined
XOR	Logical Exclusive-OR	XOR D,S	(S) ⊕ (D) → (D)	OF, SF, ZF, PF, CF AF undefined
NOT	Logical NOT	NOT D	$(\bar{D}) \rightarrow (D)$	None

(a)

Destination	Source
Register	Register
Register	Memory
Memory	Register
Register	Immediate
Memory	Immediate
Accumulator	Immediate

(b)

Destination
Register
Memory

(c)

Logic Instructions- Example

Instruction	(AL)
MOV AL,01010101B	01010101
AND AL,00011111B	00010101
OR AL,11000000B	11010101
XOR AL,00001111B	11011010
NOT AL	00100101

```
C:\DOS>DEBUG
-A
1342:0100 MOV AL,55
1342:0102 AND AL,1F
1342:0104 OR AL,C0
1342:0106 XOR AL,0F
1342:0108 NOT AL
1342:010A
-T

AX=0055  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=1342  ES=1342  SS=1342  CS=1342  IP=0102   NV UP EI PL NZ NA PO NC
1342:0102 241F          AND     AL,1F
-T

AX=0015  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=1342  ES=1342  SS=1342  CS=1342  IP=0104   NV UP EI PL NZ NA PO NC
1342:0104 0CC0          OR      AL,C0
-T

AX=00D5  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=1342  ES=1342  SS=1342  CS=1342  IP=0106   NV UP EI NG NZ NA PO NC
1342:0106 340F          XOR     AL,0F
-T

AX=00DA  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=1342  ES=1342  SS=1342  CS=1342  IP=0108   NV UP EI NG NZ NA PO NC
1342:0108 F6D0          NOT     AL
-T

AX=0025  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=1342  ES=1342  SS=1342  CS=1342  IP=010A   NV UP EI NG NZ NA PO NC
1342:010A 2B04          SUB     AX,[SI]                     DS:0000=20CD
-Q

C:\DOS>
```

Logic Instructions- Mask Application

- Mask—to clear a bit or bits of a byte, word, or double word to 0.
  - AND operation can be used to perform the mask operation
  - 1 AND 0 → 0; 0 and 0 → 0
    - A bit or bits are masked by ANDing with 0
  - 1 AND 1 → 1; 0 AND 1 → 0
    - ANDing a bit or bits with 1 results in no change
  - Example: Masking the upper 12 bits of a value in a register

    AND AX,000FH

    (AX) =FFFF

    IMM16 AND (AX) → (AX)

    000FH AND FFFFH = 0000000000001111_2 AND 1111111111111111_2

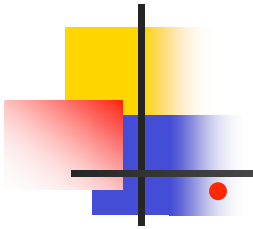
    = 0000000000001111_2

    = 000FH
- OR operation can be used to set a bit or bits of a byte, word, or double word to 1
  - X OR 0 → X; result is unchanged
  - X or 1 → 1; result is always 1
  - Example: Setting a control flag in a byte memory location to 1

    MOV AL,[CONTROL_FLAGS]

    OR AL, 10H ;00010000 sets fifth bit—b4

    MOV [CONTROL_FLAGS],AL



Shift Instructions

- SAL/SHL → Shift arithmetic left/shift logical left
- SHR → Shift logical right
- SAR → Shift arithmetic right
- SHLD → Double precision shift left
- SHRD → Double precision shift right

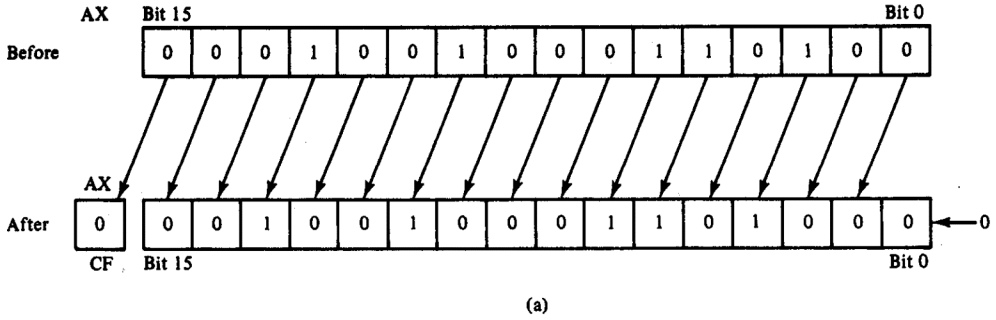
- Perform a variety of shift left and shift right operations on the bits of a destination data operand
- Basic shift instructions—SAL/SHL, SHR, SAR
  - Destination may be in either a register or a storage location in memory
  - Shift count may be:

    1= one bit shift

    CL = 1 to 255 bit shift

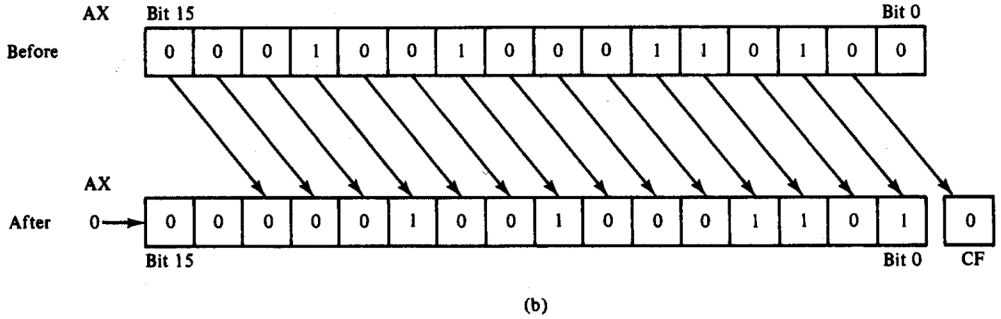
    IMM8 = 1 to 255 bit shift
  - Flags updated based on result
    - CF, SF, ZF, PF
    - AF undefined
    - OF undefined if Count $\neq$ 1

Operation of the SAL/SHL Instruction



- Typical instruction—count of 1
  SHL AX,1
- Before execution
  Dest = (AX) = 1234H = $0001\ 0010\ 0011\ 0100_2$, Count = 1, CF = X
- Operation
  - The value in all bits of AX are shifted left one bit position
  - Emptied LSB is filled with 0
  - Value shifted out of MSB goes to carry flag
- After execution
  Dest = (AX) = 2468H = $0010\ 0100\ 0110\ 1000_2$, CF = 0
- Conclusion:
  - MSB has been isolated in CF and can be acted upon by control flow instruction– conditional jump
  - Result has been multiplied by 2

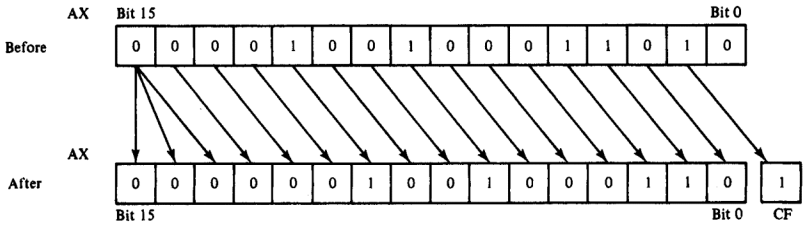
Operation of the SHR Instruction



(b)

- Typical instruction—count in CL
  SHR AX,CL
- Before execution
  Dest = (AX) = 1234H = 4660_{10} = 0001 00100011 0100_2
  Count = 02H , CF = X
- Operation
  - The value in all bits of AX are shifted right two bit positions
  - Emptied MSBs are filled with 0s
  - Values shifted out of LSBs go to carry flag
- After execution
  Dest = (AX) = 048DH = 1165_{10} = 0000 0100 1000 1101_2 ,CF = 0
- Conclusion:
  - Bit 1 has been isolated in CF and can be acted upon by control flow instruction– conditional jump
  - Result has been divided by 4
    - 4 X 1164 = 4660

Operation of the SAR Instruction



- Typical instruction—count in CL

  SAR AX,CL
- Before execution

  Dest = (AX) = 091AH = 0000100100011010_2 = +2330, Count = 02H , CF = X
- Operation
  - The value in all bits of AX are shifted right two bit positions
  - Emptied MSB is filled with the value of the sign bit—sign maintained
  - Values shifted out of LSBs go to carry flag
- After execution

  Dest = (AX) = 0246H = 0000001001000110_2 = +582 , CF = 1
- Conclusion
  - Bit 1 has been isolated in CF and can be acted upon by control flow instruction– conditional jump
  - Result has been signed extended
  - Result value has been divided by 4 and rounded to integer
    - 4 X +582 = +2328

SAR Instruction Execution

- Debug execution of SAR example

```
C:\DOS>DEBUG
-A
1342:0100 SAR AX,CL
1342:0102
-R AX
AX 0000
:091A
-R CX
CX 0000
:2
-R F
NV UP EI PL NZ NA PO NC  -
-T

AX=0246  BX=0000  CX=0002  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=1342  ES=1342  SS=1342  CS=1342  IP=0102   NV UP EI PL NZ AC PO CY
1342:0102 B98AFF        MOV     CX,FF8A
-Q

C:\DOS>
```

Shift Instructions- Application

Application–Isolating a bit of a byte of data in memory in the carry flag

- Example:
  - Instruction sequence

    MOV AL,[CONTROL_FLAGS]

    MOV CL, 04H

    SHR AL,CL
  - Before execution

    (CONTROL_FLAGS) = B7B6B5B4B3B2B1B0
  - After executing 1st instruction

    (AL) =B7B6B5B4B3B2B1B0
  - After executing 2nd instruction

    (CL) = 04H
  - After executing 3rd instruction

    (AL) = 0000B7B6B5B4

    (CF) = B3

Rotate Instructions

Mnemonic	Meaning	Format	Operation	Flags affected
ROL	Rotate left	ROL D, Count	Rotate the (D) left by the number of bit positions equal to Count. Each bit shifted out from the leftmost bit goes back into the rightmost bit position	CF OF undefined if Count ≠1
ROR	Rotate right	ROR D, Count	Rotate the (D) right by the number of bit positions equal to Count. Each bit shifted out from the rightmost bit goes into the leftmost bit position	CF OF undefined if Count ≠1
RCL	Rotate left through carry	RCL D, Count	Same as ROL except carry is attached to (D) for rotation	CF OF undefined if Count ≠1
RCR	Rotate right through carry	RCR D, Count	Same as ROR except carry is attached to (D) for rotation	CF OF undefined if Count ≠1

(a)

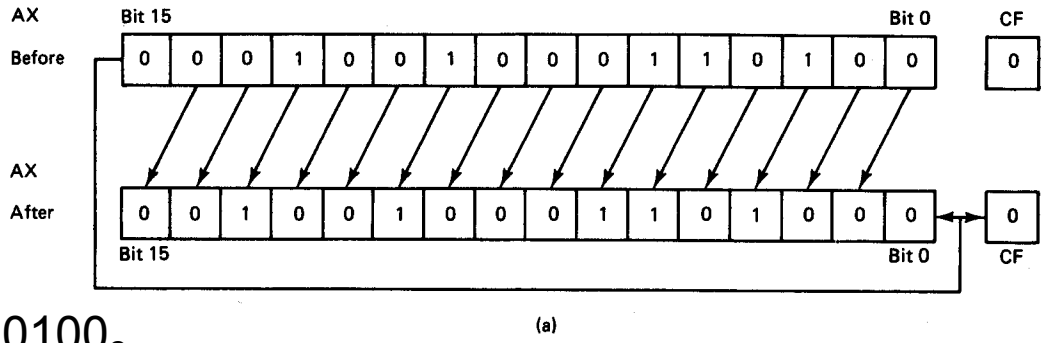
Destination	Count
Register	1
Register	CL
Register	Imm8
Memory	1
Memory	CL
Memory	Imm8

(b)

- Variety of rotate instruction provided
  - ROL → Rotate left
  - ROR → Rotate right
  - RCL → Rotate left through carry
  - RCR → Rotate right through carry
- Perform a variety of rotate left and rotate right operations on the bits of a destination data operand
- Overview of function
  - Destination may be in either a register or a storage location in memory
  - Rotate count may be:
    - 1= one bit rotate
    - CL = 1 to 255 bit rotate
    - IMM8 = 1 to 255 bit rotate
  - Flags updated based on result
    - CF
    - OF undefined if Count ≠ 1
  - Used to rearrange information

Operation of the ROL Instruction

- Typical instruction—count of 1
  ROL AX,1
- Before execution
  Dest = (AX) = 1234H
  = 0001 0010 0011 0100$_2$
  Count = 1
  CF = 0
- Operation
  - The value in all bits of AX are rotated left one bit position
  - Value rotated out of the MSB is reloaded at LSB
  - Value rotated out of MSB copied to carry flag
- After execution
  Dest = (AX) = 2468H = 0010010001101000$_2$
  CF = 0



Operation of the ROR Instruction

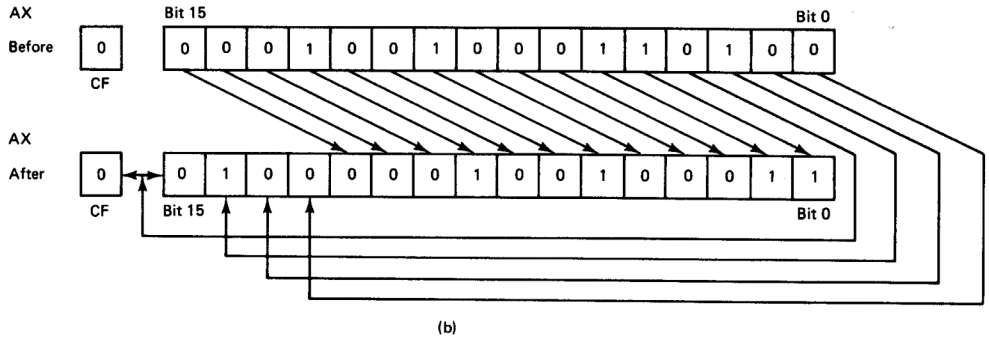
- Typical instruction—count in CL

  ROR AX,CL
- Before execution

  Dest = (AX) = 1234H

  = 0001 0010 0011 0

  Count = 04H, CF = 0

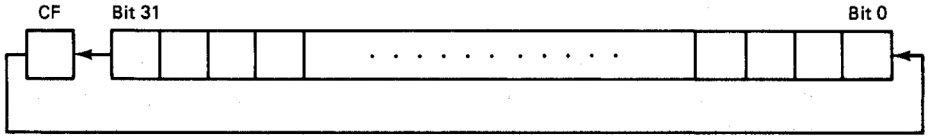


- Operation
  - The value in all bits of AX are rotated right four bit positions
  - Values rotated out of the LSB are reloaded at MSB
  - Values rotated out of MSB copied to carry flag
- After execution

  Dest = (AX) = 4123H = 0100000100100011_2

  CF = 0
- Conclusion:
  - Note that the position of hex characters in AX have be rearranged

Operation of the RCL Instruction



- RCL instruction operation
  - Typical instruction—count in CL
    RCL BX,CL
  - Before execution
    Dest = (BX) = 1234H = 0001 0010 0011 0100_2
    Count = 04H, CF = 0
  - Operation
    - The value in all bits of AX are shifted left four bit positions
    - Emptied MSBs are rotated through the carry bit back into the LSB
    - Last value rotated out of MSB retained in carry flag
    - First rotate loads prior value of CF at the LSB
  - After execution
    Dest = (BX) = 2340H = 0010 0011 0100 0000_2
    CF = 1

RCR Example

```
C:\DOS>DEBUG
-A
1342:0100 RCR BX,CL
1342:0102
-R BX
BX 0000
:1234
-R CX
CX 0000
:4
-R F
NV UP EI PL NZ NA PO NC  -
-T

AX=0000  BX=8123  CX=0004  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=1342  ES=1342  SS=1342  CS=1342  IP=0102   OV UP EI PL NZ NA PO NC
1342:0102 B98AFF        MOV     CX,FF8A
-Q

C:\DOS>
```

- RCR instruction debug execution example
  - Instruction—count in CL

    RCR BX,CL
  - Before execution

    Dest = (BX) = 1234H

    = 0001 0010 0011 0100_2

    Count = 04H

    CF = 0
  - After execution

    Dest = (BX) = 8123H = 1000001001000011_2

    CF = 0

Rotate Instructions- Application

```
MOV  AL,[HEX_DIGITS]
MOV  BL,AL
MOV  CL,04H
ROR  BL,CL
AND  AL,0FH
AND  BL,0FH
ADD  AL,BL
```

- Disassembling and adding 2 hex digits
  - 1st Instruction → Loads AL with byte containing two hex digits
  - 2nd Instruction → Copies byte to BL
  - 3rd Instruction → Loads rotate count
  - 4th instruction → Aligns upper hex digit of BL with lower digit in AL
  - 5th Instruction → Masks off upper hex digit in AL
  - 6th Instruction → Masks off upper for bits of BL
  - 7th Instruction → Adds two hex digits

Bit Test and Bit Scan Instructions

Mnemonic	Meaning	Format	Operation	Flags affected
BT	Bit test	BT D, S	Saves the value of the bit in D specified by the value in S in CF.	CF OF, SF, ZF, AF, PF undefined
BTR	Bit test and reset	BTR D, S	Saves the value of the bit in D specified by the value in S in CF and then resets the bit in D.	CF OF, SF, ZF, AF, PF undefined
BTS	Bit test and set	BTS D, S	Saves the value of the bit in D specified by the value in S in CF and then sets the bit in D.	CF OF, SF, ZF, AF, PF undefined
BTC	Bit test and complement	BTC D, S	Saves the value of the bit in D specified by the value in S in CF and then complements the bit in D.	CF OF, SF, ZF, AF, PF undefined
BSF	Bit scan forward	BSF D, S	Scan the source operand starting from bit 0. ZF = 0 if all bits are 0, else ZF = 1 and the destination operand is loaded with the bit index of the first set bit.	ZF OF, SF, AF, PF, CF undefined
BSR	Bit scan reverse	BSR D, S	Scan the source operand starting from the MSB. ZF = 0 if all bits are 0, else ZF = 1 and the destination operand is loaded with the bit index of the first set bit.	ZF OF, SF, AF, PF, CF undefined

(a)

Destination	Source
Reg16	Reg16
Reg16	Imm8
Reg32	Reg32
Reg32	Imm8
Mem16	Reg16
Mem16	Imm8
Mem32	Reg32
Mem32	Imm8

(b)

Destination	Source
Reg16	Reg16
Reg16	Mem16
Reg32	Reg32
Reg32	Mem32

(c)

- BT → Bit test
- BTR → Bit test and reset
- BTS → Bit test and set
- RTC → Bit test and complement
- Format of bit test instruction: BT(x) D,S
  - (S) → index that selects the position of the bit tested
    (S) = IMM8, Reg16, or Reg32
  - (D) → Holds value tested
    (D) = Reg16, Reg32, Mem16, or Mem32
- Operation:
  - Enables the programmer to test the state of a bit in a value held in a register or memory location
  - All→ Save the value of the selected bit in the CF
  - BT → Leaves selected bit unchanged
  - BTR → Clears the bit
  - BTS → Sets the bit
  - BTC → Complements the bit

Bit Test Instructions

- Example:
  BTC BX,7

- Before execution
  (BX) = 03F0H = 0000 0011 1<u>1</u>111 0000$_2$
  IMM8 = 7

- After Execution
  (CF) = 1
  (BX) = 0370H = 0000001110<u>0</u>1110000$_2$

Bit Scan Instructions

- BSF → Bit scan forward
- BSR → Bit scan reverse

- Format of bit scan instructions: BS(x) D,S
  - (S) → Holds value for which bits are tested to be 0
    (S) = Reg16, or Reg32
  - (D) → Index of first bit that tests as non-zero
    (D) = Reg16, Reg32
- Operation:
  - Enable the programmer to test a value in a register or memory location to determine if all of its bits are 0
  - BSF → Scans bits starting from bit 0
    Set ZF = 0 if all bits are found to be zero
    Sets ZF = 1 when first 1 bit detected and places index of that bit into destination
  - BSR → Scans bits starting from MSB
    Set ZF = 0 if all bits are found to be zero
    Sets ZF = 1 when first 1 bit detected and places index of that bit into destination
- Example:
  BSF ESI,EDX → 32-bits of EDX scanned starting from B_0