Logic, Shift and Rotate Instructions

1

Computer Organization and Assembly Language

Agenda

- Logic Instructions
- Shift Instructions
- Rotate Instructions

Logic Instructions

- The logic instructions in assembly allow to manipulate individual bits. Example:

```
1.    10101010
    AND 11110000
      = 10100000

2.    10101010
     OR 11110000
      = 11111010

3.    10101010
    XOR 11110000
      = 01011010

4.  NOT 10101010
      = 01010101
```

Truth Tables

- Truth tables for *AND, OR, XOR, and NOT*

a	b	a AND b	a OR b	a XOR b
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

a	NOT a
0	1
1	0

AND, OR, and XOR Instructions

- The AND, OR, and XOR instructions perform the named logic operations.
- The syntax is:

  AND destination, source

  OR destination, source

  XOR destination, source

AND, OR, and XOR Instructions

- The result of the operation is stored in the destination, which must be a register or memory location.
- The source may be a constant, register, or memory location. However, memory-to-memory operations are not allowed.
- *Flags Affected:*
- SF, ZF, PF reflect the result
- AF is undefined
- CF, OF= 0

Masking

- One use of AND, OR and XOR is to selectively modify the bits in the destination.
- To do this, construct a source bit pattern called mask.
- The mask bits are chosen so that the corresponding destination bits are modified in the desired manner.
- To choose the mask bits, make use of the following properties of AND, OR, and XOR.

b AND 1 = b	b OR 0 = b	b XOR 0 = b
b AND 0 = 0	b OR 1 = 1	b XOR 1 = ~b (complement of b)

Masking Properties

- 1. The AND instruction can be used to clear specific destination bits while preserving others.
- A 0 mask bit clears the corresponding destination bit; a 1 mask bit preserves the corresponding destination bit.
- 2. The OR instruction can be used to set specific destination bits while preserving the others.
- A 1 mask bit sets the corresponding destination bit; a 0 mask bit preserves the corresponding destination bit.
- 3. The XOR instruction can be used to complement specific destination bits while preserving the others.
- A 1 mask bit complements the corresponding destination bit; a 0 mask bit preserves the corresponding destination bit.

Masking Example

- Example: Clear the sign bit of AL while leaving the other bits unchanged.
- Sol: Use the AND instruction with 01111111b = 7Fh as the mask.

  AND AL,7Fh
- Example: Set the most significant and least significant bits of AL while preserving the other bits.
- Sol: Use the OR instruction with 10000001b = 81h as the mask.

  OR AL, 81h

Masking Example

- Example: Change the sign bit of DX.
- Solution: Use the XOR instruction with a mask of 8000h.
- XOR DX,8000h

Converting a Lowercase Letter to Upper Case

- The ASCII codes of "a" to "z" range from 61h to 7Ah; the codes of "A" to "Z" go from 41h to 5Ah.
- If DL contains the code of a lowercase letter, we could convert to upper case by executing
- SUB DL, 20h

Character	Code	Character	Code
a	01100001	A	01000001
b	01100010	B	01000010

- But using logic operator to convert lower to upper case just clear bit 5, by using an AND instruction with the mask 11011111b, or 0DFh.
- AND DL, 0DFh
- Try converting upper case alphabet to lowercase.

Clearing a Register

- to clear AX we could execute
- MOV AX, 0
- or
- SUB AX,AX
- 1 XOR 1 = 0 and 0 XOR 0 = 0, another way is
- XOR AX,AX
- The machine code of the first method Is three bytes, versus two bytes for the latter two methods, so the latter are more efficient.
- However, because of the prohibition on memory-to-memory operations, the first method must be used to clear a memory location.

NOT Instruction

- The NOT instruction performs the one's complement operation on the destination.
- The format is
- NOT destination
- There is no effect on the status flags.
- To Complement the bits In AX.
- NOT AX

TEST Instruction

- The **TEST** Instruction performs an AND operation of the destination with the source but does not change the destination contents.
- The purpose of the TEST instruction Is to set the status flags. The format is
- TEST destination, source
- Effect **on** flags
- SF, ZF, PF reflect the result
- AF is undefined
- CF, OF= 0

TEST Instruction

- The TEST instruction can be used to examine individual bits In an operand.
- The mask should contain 1 's In the bit positions to be tested and 0's elsewhere.
- Because 1 AND b = b, 0 AND b = 0, the result of

  TEST destination, mask
- will have 1's in the tested bit positions if and only if the destination has 1's in these positions; it will have 0's elsewhere.
- If destination has 0's in all the tested position, the result will be 0 and so ZF = 1.

TEST Instruction

- Example: Jump to label BELOW If AL contains an even number.
- Solution: Even numbers have a 0 in bit 0. Thus, the mask is 00000001b = 1.
- TEST AL, 1 ;is AL even?
- JZ BELOW ; yes, go to BELOW

Shift Instructions

- The shift and rotate instructions shift the bits in the destination operand by one or more positions either to the left or right.
- For a shift instruction, the bits shifted out are lost; for a rotate instruction, bits shifted out from one end of the operand are put back into the other end. The instruction have two possible formats.
- For a single shift or rotate, the form is

  Opcode destination, 1
- For a shift or rotate of N positions, the form is

  Opcode destination, CL

Left Shift Instructions

- The SHL (shift left) instruction shifts the bits in the destination to the left.
- The format for a single shift is

  SHL destination, 1
- A 0 is shifted into the rightmost bit position and the msb is shirted into CF.
- If the shift count *N* is different from I, the instruction takes the form
- SHL destination, CL
- where CL contains *N and N* single bit shifts are made. Effect on flags

  SF, PF, ZF reflect the result

  AF is undefined

  CF= last bit shifted out

  OF= 1 if result changes sign on last shift

SHL Example

- DH contains 8Ah and CL contains 3. What are the values of DH and of CL after the instruction SHL DH,CL is executed.
- Solution: The binary value of DH Is 10001010.
- After 3 left shifts, CF will contain 0.
- The new contents of DH may be obtained by erasing the leftmost three bits and adding three zero bits to the right end, thus 01010000b = 50h.

Multiplication by Left Shift

- A left shift on a binary number multiplies it by 2.
- Let AL contain *5* = 00000101b.
- A left shift gives 00001010b = 10d, doubling its value.
- Another left shift yields 00010100 = 20d, so it is doubled again.
- *The SAL instruction* (shift arithmetic left) is can also be used in instances where numeric multiplication is intended.
- Try multiplying a value with shift left operation.

Right Shift Instructions

- The instruction SHR (shift right) performs right shifts on the destination operand.
- The format for a single shift is

  SHR destination, 1
- A 0 is shifted Into the msb position, and the rightmost bit is shifted into CF.
- If the shift count *N* is different from 1, the instruction takes the form

  SHR destination, CL
- The effect on the flags Is the same as for SHL.

The SAR Instruction

- The SAR Instruction (shift arithmetic right) operates like SHR, with one difference: the msb retains its original value

Division by Right Shift

- As a left shift doubles the destination's value, a right shift divides it by 2.
- For odd numbers, a right shift halves It and rounds down to the nearest integer.
- For example, if BL contains 00000101b = 5, after a right shift, BL will contain 00000010 = 2.

Signed and Unsigned Division

- If an unsigned interpretation is being given, SHR should be used.
- For a signed interpretation, SAR must be used, because it preserves the sign.

Rotate Instructions

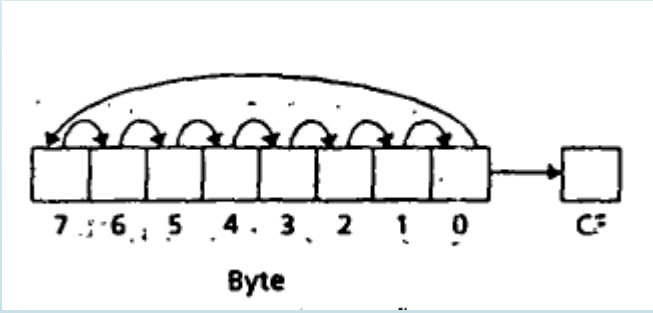
- The instruction ROL (rotate left) shifts bits to the left.
- The msb is shifted into the rightmost bit.
- The CF also gets the bit shifted out of the msb.
- The destination bits forms a circle, with the least significant bit following the msb in the circle.
- The syntax is

  ROL destination,1
- and

  ROL destination, CL

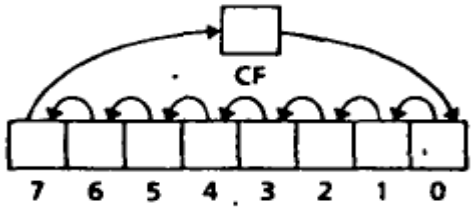
Rotate Right

- The instruction ROR (rotate right) works just like ROL, except that the bits are rotated to the right.
- The rightmost bit is shifted into the msb, and also into the CF.



Rotate Carry Left

- The Instruction RCL (Rotate through Carry Left) shifts the bits of the destination to the left.
- The msb is shifted Into the CF, and the previous value of CF is shifted Into the rightmost bit.
- RCL works like just like ROL, except that CF is part of the circle of bits being rotated. The syntax is similar to ROL.



Rotate Carry Right

- The instruction RCR (Rotate through Carry Right) works just like RCL, except that the bits are rotated to the light.
- The syntax is:

  RCR destination,1

  RCR destination, CL

Effect of the rotate Instructions on the flags

- SF, PF, ZF reflect the result
- AF is undefined
- CF = last bit shifted out
- OF = 1 if result changes sign (In the last rotation

Binary Output

- Outputting the contents of BX in binary also involves the shift operation.
- The algorithm is

```
FOR 16 times DO
    Rotate left BX /* BX holds output value, put msb into CF */
    IF CF = 1
    THEN
        output '1'
    ELSE
        output '0'
    END_IF
END_FOR
```