

Lab 4 – Bitwise Operations

Bitwise operations are operations that are executed for each bit (bit-by-bit).

We have:

- logical bitwise operations
- shift operations
- rotate operations

Boolean op -> logical bitwise operations

| Operand1 – op1 | Operand2 – op2 | And op1, op2 | Or op1, op2 | Xor op1, op2 | Not op1 |
|----------------|----------------|--------------|-------------|--------------|---------|
| 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 |

The IA-32 assembly instructions that perform these logical bitwise operations are detailed below:

and *operand1, operand2*

or *operand1, operand2*

xor *operand1, operand2*

not *operand1*

where *operand1* and *operand2* are either registers, memory references (i.e. variables) or constants (*operand1* can not be a constant!) both of the same size/type: byte, word, doubleword.

Each single bit of *operand2* is and/or/xor with each corresponding bit from *operand1*.

The result is in all cases saved in *operand1*

The *not* operation is a unary operation, so the bits of *operand1* are modified directly

General rules for OR and AND

X or 0 = X

X or 1 = 1

X and 0 = 0

X and 1 = X

Examples:

Data segment: a db 0000.1111b

Code segment:

and byte[a], 0000.0101b ; => 0000.1111b and

0000.0101b => a=0000.0101b

or byte[a], 0000.1101b ; => 0000.0101b or

0000.1101b => a=0000.1101b

xor byte[a], 0000.0110b ; => 0000.1101b xor

0000.0110b => a=0000.1011b

Xor allows us to save value 0 in a register (instead of using mov eax, 0):

Xor eax, eax => eax = 0

not byte[a] ; => not 0000.1011b => a=1111.0100b

Shift instructions and operations

| | | |
|--|--|--|
| SHL <i>destination, count</i> <i>destination: reg, mem</i> <i>count: imm8, or CL reg</i> | The SHL (shift logic left) instruction performs a logical left shift on the destination operand, filling the lowest bit with 0. The highest bit is moved to the Carry flag, and the bit that was in the Carry flag is discarded. | mov dl, 0000.1111b shl dl, 3 ; dl = 0111.1000b |
| SHR <i>destination, count</i> <i>destination: reg, mem</i> <i>count: imm8, or CL reg</i> | The SHR (shift logic right) instruction performs a logical right shift on the destination operand, replacing the highest bit with a 0. The lowest bit is copied into the Carry flag, and the bit that was previously in the Carry flag is lost. | mov bl, 1010.1010b shr bl, 4 ; bl = 0000.1010b mov eax, 12340000h shr eax, 16 ; eax = 00001234h |
| SAL <i>destination, count</i> <i>destination: reg, mem</i> <i>count: imm8, or CL reg</i> | The SAL (shift arithmetic left) instruction works the same as the SHL instruction. For each shift count, SAL shifts each bit in the destination operand to the next highest bit position. The lowest bit is assigned 0. The highest bit is moved to the Carry flag, and the bit that was in the Carry flag is discarded. | mov al, 1110.0011b sal al, 1 ; al=1100.0110b, cf=1 mov bl, 0000.1111b sal bl, 2 ; bl = 0011.1100b |
| SAR <i>destination, count</i> <i>destination: reg, mem</i> <i>count: imm8, or CL reg</i> | The SAR (shift arithmetic right) instruction performs a logical right shift on the destination operand, filling all the positions with the most significant bit (bit sign). The lowest bit is copied into the Carry flag, and the bit that was previously in the Carry flag is lost. | mov al, 1110.0011b sar al, 1 ; al=1111.0001b, cf =1 mov bl, 10101010b shr bl, 4 ; bl = 11111010b |

Rotate instructions and operations

| | | |
|--|---|--|
| ROL <i>destination, count</i> <i>destination: reg, mem</i> <i>count: imm8, or CL reg</i> | The ROL (rotate left) instruction shifts each bit to the left. The highest bit is copied into the Carry flag and the lowest bit position. | mov bl, 0111.1000b rol bl, 4 ; bl = 1000.0111b, cf = 1 mov bl, 1111.0000b rol bl, 6 ; bl = 0011.1100b |
| ROR <i>destination, count</i> <i>destination: reg, mem</i> <i>count: imm8, or CL reg</i> | The ROR (rotate right) instruction shifts each bit to the right and copies the lowest bit into the Carry flag and the highest bit position. | mov al, 1010.1110b ror al, 3 ; al=1101.0101b, cf=1 |

| | | |
|--|---|--|
| | | mov al, 1111.0000b ror al, 5 ; al=1000.0111b |
| RCL <i>destination, count</i> <i>destination: reg, mem</i> <i>count: imm8, or CL reg</i> | The RCL (rotate carry left) instruction shifts each bit to the left, copies the Carry flag to the LSB, and copies the MSB into the Carry flag. | mov al, 0 stc rcl al, 1 ; al=0000.0001b, cf = 0 mov al, 0 stc rcl al, 2 ; al=0000.0010b, cf=0 mov al, 0 stc rcl al, 3 ;al = 0000.0100b, cf=0 |
| RCR <i>destination, count</i> <i>destination: reg, mem</i> <i>count: imm8, or CL reg</i> | The RCR (rotate carry right) instruction shifts each bit to the right, copies the Carry flag into the MSB, and copies the LSB into the Carry flag. | mov bl, 0 stc rcr bl, 1 ; bl=1000.0000b, cf = 0 mov bl, 0 stc rcr bl, 2 ;bl=0100.0000b, cf=0 mov bl, 0 stc rcr bl, 3, bl =0010.0000b, cf=0 |

x -byte : $x_7 x_6 x_5 x_4 x_3 x_2 x_1 x_0$
 $\text{shl byte}[x], 1 \Rightarrow x_6 x_5 x_4 x_3 x_2 x_1 x_0 0$, $cf = x_7$
 $\text{shr byte}[x], 1 \Rightarrow 0 x_7 x_6 x_5 x_4 x_3 x_2 x_1$, $cf = x_0$
 $\text{sll byte}[x], 2 \Rightarrow x_5 x_4 x_3 x_2 x_1 x_0 00$, $cf = x_6$
 $\text{sar byte}[x], 3 \Rightarrow x_7 x_7 x_7 x_7 x_6 x_5 x_4 x_3$, $cf = x_2$
 $\text{rol byte}[x], 2 \Rightarrow x_5 x_4 x_3 x_2 x_1 x_0 x_7 x_6$, $cf = x_6$
 $\text{ror byte}[x], 3 \Rightarrow x_2 x_1 x_0 x_7 x_6 x_5 x_4 x_3$, $cf = x_2$

 stc - set cf
 clc - clear cf

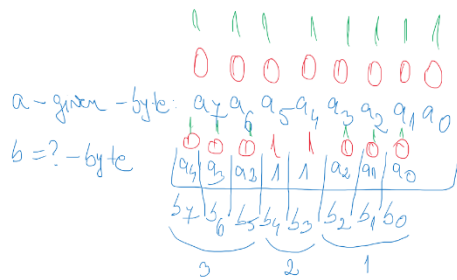
 $\text{pp. } cf = k$, $\text{rcl byte}[x], 1 \Rightarrow x_6 x_5 x_4 x_3 x_2 x_1 x_0 k$,
 $cf = x_7$

 $\text{pp. } cf = k$, $\text{rcr byte}[x], 1 \Rightarrow k x_7 x_6 x_5 x_4 x_3 x_2 x_1$,
 $cf = x_0$

Prob1:

Being given a byte A, construct a new byte B in the following way:

- bits 0-2 of B should be equal to bits 0-2 of A
- bits 3-4 of B should be set to 1
- bits 5-7 of B should be equal to bits 2-4 of A



$a \leftarrow b$
 $b \leftarrow b \gg 3$

$X \text{ or } 0 = X$
 $X \text{ or } 1 = 1$
 $X \text{ and } 0 = 0$
 $X \text{ and } 1 = X$

$a \leftarrow b$
 $b \leftarrow b \gg 3$
 $b = 18h$
 $b = 07h$

- $\text{mov al}, [a]; \text{al} = a_7 \dots a_0$
 $\text{and al}, 0000111b; \text{al} = 0000.0 a_2 a_1 a_0$
 $\text{or byte [b], al}; \text{al} = 0000.0000 \text{ or } 0000.0 a_2 a_1 a_0$
- $\text{or byte [b], al}; 00011000b; 00011 a_2 a_1 a_0$
- $\text{mov al}, [a]$
 $\text{shl al}, 3; \text{al} = a_4 a_3 a_2 a_1 a_0 000$
 $\text{and al}, 11100000b; \text{al} = a_4 a_3 a_2 00000$
 $\text{or byte [b], al}; a_4 a_3 a_2 11 a_2 a_1 a_0$

Prob 2:

Generate value 13 in al.

$\text{al} = 13, \text{al} = 0$
 [shift + rotate]
 $\text{al} = 0000.1101b$

$\text{mov al}, 0$
 $\text{stc}; \text{cf} = 1$
 $\text{rcl al}, 1; \text{al} = 0000.0001b, \text{cf} = 0$
 stc
 $\text{rcl al}, 1; \text{al} = 0000.0011b, \text{cf} = 0$
 $\text{shl al}, 1; \text{al} = 0000.0110b, \text{cf} = 0$
 stc
 $\text{rcl al}, 1; \text{al} = 0000.1101b, \text{cf} = 0$

Prob 3.

Identify the numbers a and b from 2 doubleword X and Y:

a is represented of bits 11-16 of X

b is represented of bits 26-31 of Y

Compute then a*b.

So we get our a - found the bits
11-16 a double word X

our b - found the bits 26-31 in double Y

a.b

X₃₁ ... X₁₆ X₁₅ X₁₄ X₁₃ X₁₂ X₁₁ ... X₀

0 ... 0000 0000 X₁₆ X₁₅ X₁₄ X₁₃ X₁₂ X₁₁

Shift double [X], 11; 00000000 X₃₁ X₃₀ ... X₁₆ X₁₅ X₁₄ X₁₃ X₁₂ X₁₁

and double [X], 0 ... 0 11111111; 0 ... 0 X₁₆ X₁₅ X₁₄ X₁₃ X₁₂ X₁₁

mov eax, [X] → al

shr eax, 11; byte [X+0] ← a

Y₃₁ Y₃₀ Y₂₉ Y₂₈ Y₂₇ Y₂₆ Y₂₅ ... Y₀

not double [Y]; Y = Y₃₁ Y₃₀ ... Y₀

shr double [Y], 26 (, shr double [Y], 26)

Y = 0 ... 0 Y₃₁ Y₃₀ Y₂₉ Y₂₈ Y₂₇ Y₂₆

mov ebx, [Y]; ebx

shr ebx, 26; byte [Y+0] ← b

mul bl; ax =

Handwritten notes on the right:

32 bits

X = 11 ... 11

a = 00 1111 11 b

32 bits

Y = 11 ... 11

b = 0000 0000 b

0 0 h