Chapter 10. Bitwise Operators

Byoung-Tak Zhang
TA: Hanock Kwak
Biointelligence Laboratory
School of Computer Science and Engineering
Seoul National University

http://bi.snu.ac.kr

Bitwise Operators

- Bitwise Operators
  - act on integral expressions represented as strings of binary digits.

- Bitwise Complement
  - It inverts the bit string representation.
  - int a = 70707;
  - a : 00000000 00000001 00010100 00110011
  - ~a : 11111111 11111110 11101011 11001100

Bitwise Binary Logical Operators

- & (and), ^ (exclusive or), I (inclusive or)

a	b	a & b	a ^ b	a | b
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	0	1

Bitwise Binary Logical Operators

Declaration and initializations	
int a = 33333, b = -77777;	
Expression	*Representation*
a	00000000 00000000 10000010 00110101
b	11111111 11111110 11010000 00101111
a & b	00000000 00000000 10000000 00100101
a ^ b	11111111 11111110 01010010 00011010
a | b	11111111 11111110 11010010 00111111
~ (a | b)	00000000 00000001 00101101 11000000
(~a & ~b)	00000000 00000001 00101101 11000000

Left and Right Shift Operators

- expr1 << expr2
  - causes the bit representation of expr1 to be shifted to the left by the number of places specified by expr2.

Declaration and initializations	
char c = 'Z';	
Expression	*Representation*
c	00000000 00000000 00000000 01011010
c << 1	00000000 00000000 00000000 10110100
c << 4	00000000 00000000 00000101 10100000
c << 31	00000000 00000000 00000000 00000000

Left and Right Shift Operators

- expr1 >> expr2
  - causes the bit representation of expr1 to be shifted to the right by the number of places specified by expr2.
  - For unsigned integral expressions, 0s are shifted in at the high end.
  - For the signed types, some machines shift in 0s, while others shift in sign bits.

Declaration and initializations	
int a = 1 << 31; /* shift 1 to the high bit */ **unsigned b = 1 << 31;**	
Expression	*Representation*
a	10000000 00000000 00000000 00000000
a >> 3	11110000 00000000 00000000 00000000
b	10000000 00000000 00000000 00000000
b >> 3	00010000 00000000 00000000 00000000

Left and Right Shift Operators

Declaration and initializations		
unsigned a = 1, b = 2;		
Expression		*Representation*
a << b >> 1	(a << b) >> 1	00000000 00000010
a << 1 + 2 << 3	(a << (1 + 2)) << 3	00000000 01000000
a + b << 12 * a >> b	((a + b) << (12 * a)) >> b	00001100 00000000

Masks

- Mask
  - a constant or variable that is used to extract desired bits from another variable or expression
- Constant Value 1
  - 00000000 00000000 00000000 00000001
  - used to determine the low-order bit of an int expression
    - ```
      int i, mask = 1;
      for ( i = 0; i < 10; ++i)
        printf("%d", i & mask);
      ```
- Constant Value 255
  - 00000000 00000000 00000000 11111111
  - a mask for the low-order byte
  - v & 255
      ```

Printing an int Bitwise

```
/* Bit print an int expression */
#include <limits.h>
void bit_print(int a)
{
    int i;
    int n = sizeof(int) * CHAR_BIT;  /* in limits.h */
    int mask = 1 << (n - 1);  /* mask = 1000...0 */
    for (i = 1; i <= n; ++i) {
        putchar(((a & mask) == 0) ? '0' : '1');
        a <<= 1;   /*  a = a << 1; */
        if (i % CHAR_BIT == 0 && i < n)
            putchar(' ');
    }
}
```