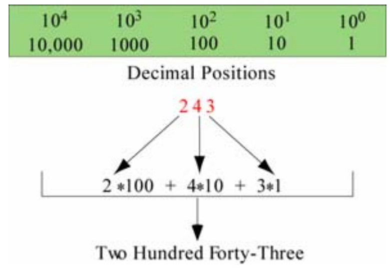
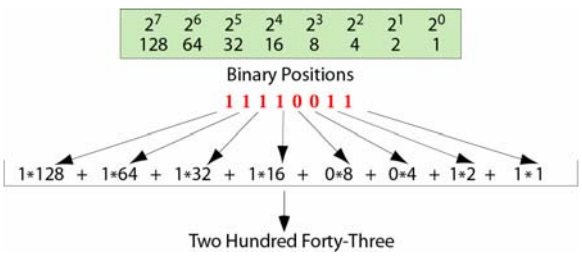
Chapter 4

Operations on Bits

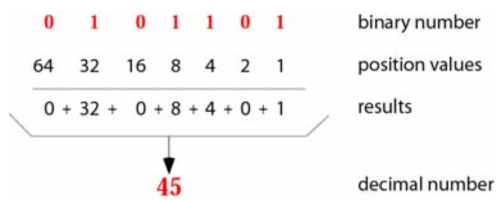
Decimal system :: Review



Binary system :: Review



Binary to decimal conversion :: Review



Decimal to binary conversion :: Review

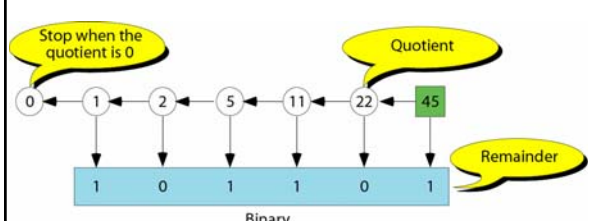
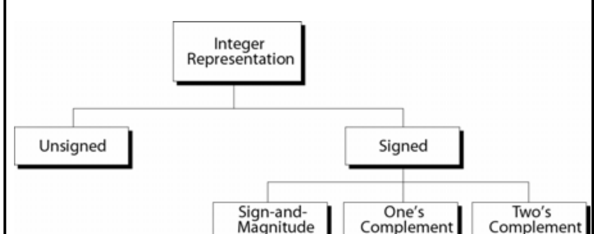


Figure 3-6

Taxonomy of integers :: Review



OBJECTIVES

- Understand two's complement format of integer representation.
- Apply arithmetic operations on bits when the integer is represented in two's complement.
- Apply logical operations on bits.

Reading

- Ch 3.3 (p 37-39, only two's complement format)
- Ch 4.1 (disregard arithmetic operations on floating-point numbers)
- Next lecture will be based on the rest of chapter 4

Note:

Two's complement is the most common, the most important, and the most widely used representation of integers today.

Table 3.7 Range of two's complement integers

# of Bits	Range		
8	−128	0	+127
16	−32,768	0	+32,767
32	−2,147,483,648	0	+2,147,483,647

Note:

In two's complement representation, the leftmost bit defines the sign of the number. If it is 0, the number is positive. If it is 1, the number is negative.

Example 12

Store +7 in an 8-bit memory location using two's complement representation.

Solution

*First change the number to binary 111. Add five 0s to make a total of N (8) bits, **00000111**. The sign is positive, so no more action is needed. The result is:*

00000111

Example 13

Store –40 in a 16-bit memory location using two's complement representation.

Solution

*First change the number to binary **101000.** Add ten 0s to make a total of N (16) bits, **0000000000101000**. The sign is negative, so leave the rightmost 0s up to the first 1 (including the 1) unchanged and complement the rest. The result is:*

***1111111111011**1000*

Brooks/Cole Thomson Learning

Table 3.8 Example of storing two's complement integers in two different computers

Decimal	8-bit allocation	16-bit allocation
+7	00000111	0000000000000111
−7	11111001	1111111111111001
+124	01111100	0000000001111100
−124	10000100	1111111110000100
+24,760	overflow	0110000010111000
−24,760	overflow	1001111101001000

Brooks/Cole Thomson Learning

Example 14

Interpret 11110110 in decimal if the number was stored as a two's complement integer.

Solution

The leftmost bit is 1. The number is negative. Leave 10 at the right alone and complement the rest. The result is 00001010. The two's complement number is 10. So the original number was –10.

Brooks/Cole Thomson Learning

Note:

Two's complement can be achieved by reversing all bits except the rightmost bits up to the first 1 (inclusive). If you two's complement a positive number, you get the corresponding negative number. If you two's complement a negative number, you get the corresponding positive number. If you two's complement a number twice, you get the original number.

Brooks/Cole Thomson Learning

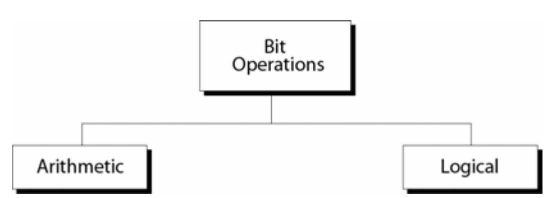
Table 3.9 Summary of integer representation

Contents of Memory	Unsigned	Sign-and-Magnitude	One's Complement	Two's Complement
0000	0	+0	+0	+0
0001	1	+1	+1	+1
0010	2	+2	+2	+2
0011	3	+3	+3	+3
0100	4	+4	+4	+4
0101	5	+5	+5	+5
0110	6	+6	+6	+6
0111	7	+7	+7	+7
1000	8	−0	−7	−8
1001	9	−1	−6	−7
1010	10	−2	−5	−6
1011	11	−3	−4	−5
1100	12	−4	−3	−4
1101	13	−5	−2	−3
1110	14	−6	−1	−2
1111	15	−7	−0	−1

Brooks/Cole Thomson Learning

Figure 4-1

Operations on bits



Brooks/Cole Thomson Learning

4.1

ARITHMETIC OPERATIONS

Table 4.1 Adding bits

Number of 1s	Result	Carry
None	0	
One	1	
Two	0	1
Three	1	1

Note:

Rule of Adding Integers in Two's Complement

Add 2 bits and propagate the carry to the next column. If there is a final carry after the leftmost column addition, discard it.

Example 1

Add two numbers in two's complement representation: (+17) + (+22) ➔ (+39)

Solution

```
Carry            1
          0  0  0  1  0  0  0  1   +
          0  0  0  1  0  1  1  0
          -------------------------
Result    0  0  1  0  0  1  1  1   ➔  39
```


Example 2

Add two numbers in two's complement representation: (+24) + (-17) ➔ (+7)

Solution

```
Carry     1  1  1  1  1
          0  0  0  1  1  0  0  0   +
          1  1  1  0  1  1  1  1
          -------------------------
Result    0  0  0  0  0  1  1  1   ➔  +7
```


Example 3

Add two numbers in two's complement representation: (-35) + (+20) ➔ (-15)

Solution

```
Carry            1  1  1
          1  1  0  1  1  1  0  1   +
          0  0  0  1  0  1  0  0
          -------------------------
Result    1  1  1  1  0  0  0  1   ➔  -15
```

Example 4

Add two numbers in two's complement representation: (+127) + (+3) ➔ (+130)

Solution

```
Carry   1 1 1 1 1 1 1
        0 1 1 1 1 1 1 1  +
        0 0 0 0 0 0 1 1
        ----------------------------------
Result  1 0 0 0 0 0 1 0  ➔ -126 (Error)
```

An overflow has occurred.

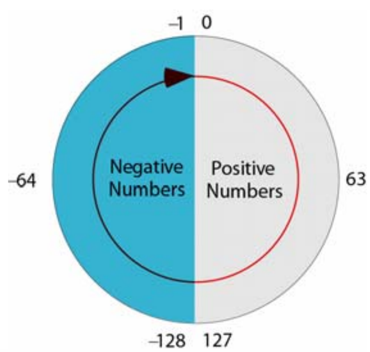
Note:

Range of numbers in two's complement representation

- (2^{N-1}) ---------- 0 ----------- +(2^{N-1} –1)

Figure 4-2

Two's complement numbers visualization



Note:

When you do arithmetic operations on numbers in a computer, remember that each number and the result should be in the range defined by the bit allocation.

Example 5

Subtract 62 from 101 in two's complement: (+101) - (+62) ⇔ (+101) + (-62)

Solution

```
Carry   1 1
        0 1 1 0 0 1 0 1  +
        1 1 0 0 0 0 1 0
        ----------------------------------
Result  0 0 1 0 0 1 1 1  ➔ 39
```

The leftmost carry is discarded.

4.2 LOGICAL OPERATIONS

Logical Operations

- A single bit can be either 0 or 1
- We can interpret 0 and 1 logically
- 0 – false;
- 1 – true;
- Logical operation – operations applied on bits interpreted as logical values

Figure 4-3

Unary and binary operations

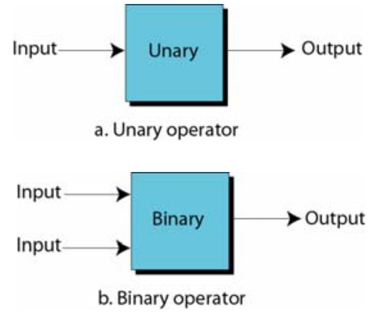


Figure 4-4

Logical operations

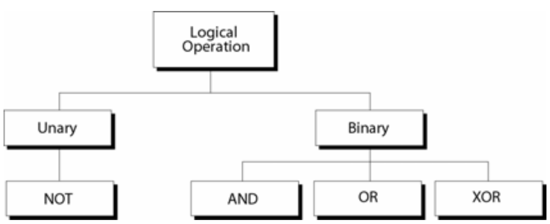


Figure 4-5

Truth tables

NOT

x	NOT x
0	1
1	0

AND

x	y	x AND y
0	0	0
0	1	0
1	0	0
1	1	1

OR

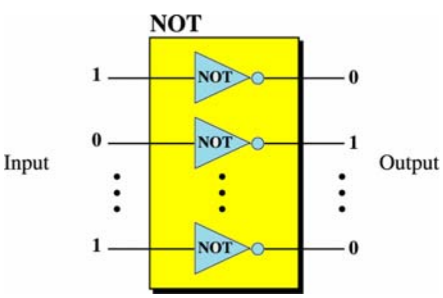
x	y	x OR y
0	0	0
0	1	1
1	0	1
1	1	1

XOR

x	y	x XOR y
0	0	0
0	1	1
1	0	1
1	1	0

Figure 4-6

NOT operator



Example 7

Use the NOT operator on the bit pattern 10011000

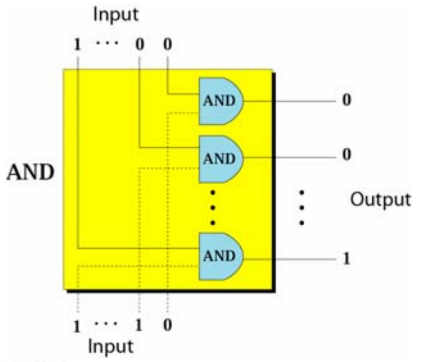
Solution

Target	1 0 0 1 1 0 0 0	*NOT*

Result	0 1 1 0 0 1 1 1	

Figure 4-7

AND operator



Example 8

Use the AND operator on bit patterns 10011000 and 00110101.

Solution

Target	**1 0 0 1 1 0 0 0**	***AND***
	0 0 1 1 0 1 0 1	

Result	**0 0 0 1 0 0 0 0**	

Figure 4-8

Inherent rule of the AND operator

(0) AND (X) → (0)

(X) AND (0) → (0)

Conclusions

- Most computers today use the two's complement method of integer representation
- We can perform arithmetic and logic operations on bits
- To subtract in two's complement, just negate the number to be subtracted and add
- Numbers to be added must be within the range defined by the bit allocation

Conclusions

- "Overflow" means a condition in which a number is not within the range defined by the bit allocation
- Logical operations can be unary and binary
- Truth tables list all possible input combinations with the corresponding output
- Logical operations: NOT, AND

Next Lecture

We'll discuss other logical operations and their applications