# ※5.2 Representations of Integers and Integer Algorithm

- In a digital computer, data and instructions are encoded as bits
  - Bit : Binary digit, a 0 or 1.
  - Discuss the binary, hexadecimal, octal number system.

- Binary number system.
  - Def: Reading from right. In representing an integer about two symbol 0, 1.

ex)

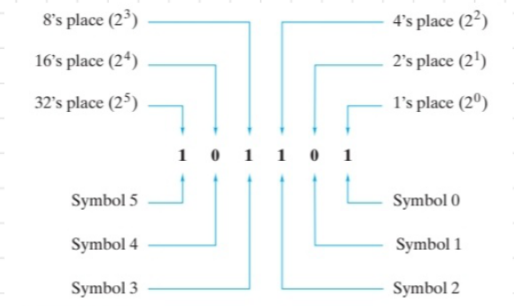


Figure 5.2.2 The binary number system.

$101101_{(2)} = 1\cdot 2^5 + 0\cdot 2^4 + 1\cdot 2^3 + 1\cdot 2^2 + 0\cdot 2^1 + 1\cdot 2^0$

$= 45_{(10)}$

- The number of bits to represent a positive integer $n$

$\lfloor 1+\log_2 n \rfloor = \lfloor \log_2(2n) \rfloor$

(proof) Suppose that $k$-bit integer $n$ is

$\langle n = b_0 2^0 + b_1 2^1 + \cdots + b_{k-2}2^{k-2} + 1\cdot 2^{k-1} \rangle$

$\because b_{k-1} = 1$

from the equations,

$\langle n \geq 2^{k-1} \rangle$

and let all of $b_k$ is 1,

$\langle n \leq 1\cdot 2^0 + 1\cdot 2^1 + \cdots + 1\cdot 2^{k-2} + 1\cdot 2^{k-1}$

$\leq 2^k - 1$

$< 2^k$

$\because$ from the geometric sum, $a=1,\ r=2$.

$1\cdot(2^{(k-1)+(k+1)}-1) = 2^k - 1 \rangle$

Therefore $\langle 2^{k-1} \leq n < 2^k \rangle$

Taking $\log_2$. $\langle k-1 \leq \log_2 n < k$

$k \leq \log_2 n + 1 < k+1 \rangle$

Since $k$ is an integer not greater than $1+\log_2 n$ ( $k = \lfloor 1+\log_2 n \rfloor$ ) and the number of bits required to represent $n$.

$\langle k = \lfloor 1+\log_2 n \rfloor$

$\leq 1+\log_2 n = \log(2n) \rangle$

- Hexadecimal number system.
  - Def: To represent integer using the symbols

    0, 1, 2, ⋯, 9, A, B, ⋯ F

ex)

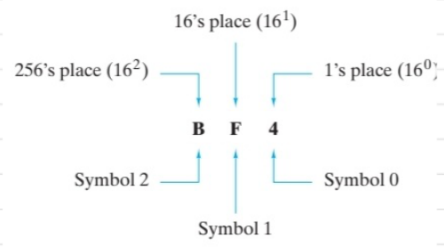


Figure 5.2.3 The hexadecimal number system.

$BF4 = 11\cdot 16^2 + 15\cdot 16^1 + 4\cdot 16^0$

$= 11\cdot 256 + 15\cdot 16 + 4 = 3060_{(10)}$

- The number of bits to represent a positive integer $n$

$\lfloor 1+\log_{16} n \rfloor$

- Convert to base $n$.

1. Expressed as sum of $n^k$
2. Using the remainder

ex) $35_{(10)}$ in binary

$2 \mid 35$
$2 \mid 17 \cdots 1$
$2 \mid 8 \cdots 1$
$2 \mid 4 \cdots 0$
$2 \mid 2 \cdots 0$
$1 \cdots 0$

$35_{(10)} = 100011_{(2)}$

ex) $35_{(10)}$ in hexadecimal

$16 \mid 35$
$16 \mid 2 \cdots 3$
$0 \cdots 2$

$35_{(10)} = 23_{(16)}$

- Adding the number in arbitrary bases
  - If the numbers to add are $b_n b_{n-1} \cdots b_1 b_0$ and $b'_n b'_{n-1} \cdots b'_1 b'_0$, add the carry bit from the previous calculation

ex) Binary addition

```
  1  11
  10011 (2)
+ 11001 (2)
-----------
 101100 (2)
```

Hexadecimal addition

```
  1 1
  42EA (16)
+ 84F (16)
-----------
  4B39 (16)
```

- Algorithm to compute a power $a^n$ (Repeated squaring)
  - Def: When you calculate $a^n$, it is an algorithm that can be used efficiently in terms of time.

1. Express the power n as a product of binary

ex) $a^{29} = a^1 \cdot a^4 \cdot a^8 \cdot a^{16}$

$29_{(2)} = 11101_{(2)}$

$= 1 + 2^1 \cdot 0 + 2^2 \cdot 1 + 2^3 \cdot 1 + 2^4 \cdot 1$

| $x$ | Current Value of $n$ | $n$ mod 2 | Result | Quotient When $n$ Divided by 2 |
|---|---|---|---|---|
| $a$ | 29 | 1 | $a$ | 14 |
| $a^2$ | 14 | 0 | Unchanged | 7 |
| $a^4$ | 7 | 1 | $a \cdot a^4 = a^5$ | 3 |
| $a^8$ | 3 | 1 | $a^5 \cdot a^8 = a^{13}$ | 1 |
| $a^{16}$ | 1 | 1 | $a^{13} \cdot a^{16} = a^{29}$ | 0 |

n

2 | 29

2 | 14 ... 1

2 | 7 ... 0

2 | 3 ... 1

1 ... 1

Figure 5.2.4 Computing $a^{29}$ using repeated squaring.

// x is set to a. n is set to the value of the exponent 29. Result is set to x

2. Repeat this process until result == n

```
4  int repeatedSquaring(int a, int n)
5  {
6      int result = 1;
7      int x = a;
8
9      while (n > 0)
10     {
11         if (n % 2 == 1)
12             result *= x;
13
14         x *= x;
15         n = floor(n / 2);
16     }
17
18     return result;
19 }
```

- $ab \bmod z = (a \bmod z)(b \bmod z) \bmod z$
  - To compute $a^n \bmod z$ for large value of $a, n$
    - It is impractical to compute $a^n$
  - This idea is to compute remainder after each multiplication thereby keeping the numbers relatively small.

↳ proof)

Let $w = ab \bmod z$, $x = a \bmod z$, $y = b \bmod z$

$ab = q_1 z + w \rightarrow w = ab - q_1 z$

$a = q_2 z + x$, $b = q_3 z + y$

Now, $w = ab - q_1 z$

$= (q_2 z + x)(q_3 z + y) - q_1 z$

$= (q_2 q_3 z + q_2 y + q_3 x - q_1) z + xy$

$= qz + xy$

then, $xy = -qz + w$

ex) $5^{29} \bmod 13$

$a^{29} = a^1 \cdot a^4 \cdot a^8 \cdot a^{16}$

* $a^2 = a^1 \cdot a$, $a^4 = a^2 \cdot a^2$, $a^{29} = a^{16} \cdot a^{13}$

$5^2 \bmod 13 = (5^1 \% 13)(5^1 \% 13) \% 13 = 630$

$5^4 \bmod 13 = 630^2 \% 13 = 4472$

$5^{8} \bmod 13 = 5^4 \cdot 4472 \% 13 = 4470$

$\vdots$ $\vdots$

$5^{29} \bmod 13 = 152 \cdot 634 \bmod 13 = 113$

```
int repeatedSquaring(int a, int n, int z)
{
    int result = 1;
    int x = a % z;

    while (n > 0)
    {
        if (n % 2 == 1)
            result = (result * x) % z;
        x = (x * x) % z;
        n = floor(n / 2);
    }

    return result;
}
```