Bộ môn Công nghệ phần mềm
Khoa Công nghệ thông tin
Trường Đại học Khoa học Tự nhiên

KỸ THUẬT LẬP TRÌNH

ThS. Đặng Bình Phương
dbphuong@fit.hcmus.edu.vn

CÁC KỸ THUẬT THAO TÁC TRÊN BIT

Nội dung

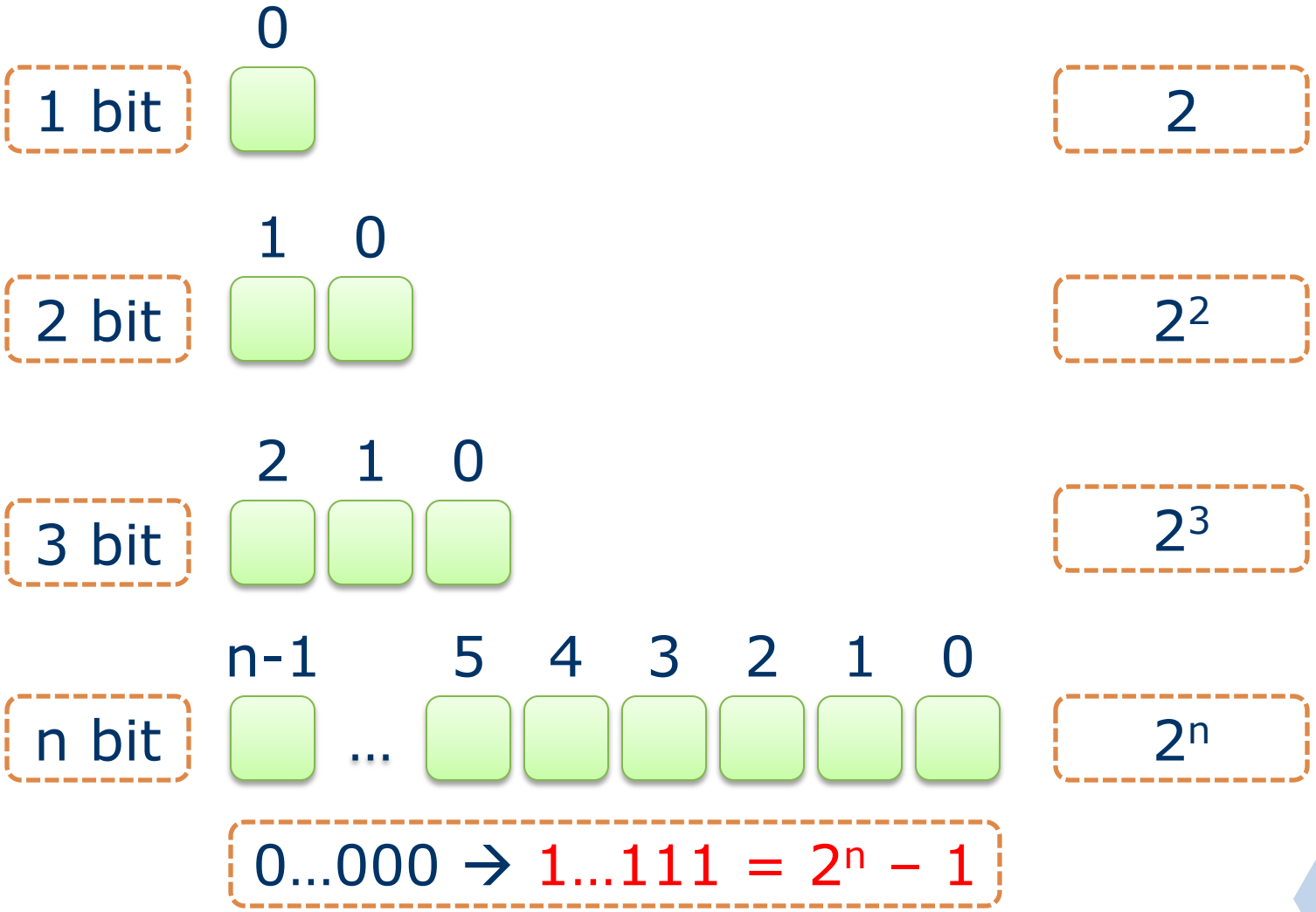
1. Các toán tử logic
2. Các toán tử dịch bit
3. Các ứng dụng
4. Bài tập

Đơn vị đo thông tin

- Hai trạng thái tắt-0 và mở-1 (nhị phân).
- Ký số nhị phân (Binary Digit) – bit
- bit - Đơn vị chứa thông tin nhỏ nhất.
- Các đơn vị đo thông tin lớn hơn:

Tên gọi	Ký hiệu	Giá trị
Byte	B	8 bit
KiloByte	KB	2^{10} B = 1024 Byte
MegaByte	MB	2^{10} KB = 2^{20} Byte
GigaByte	GB	2^{10} MB = 2^{30} Byte
TeraByte	TB	2^{10} GB = 2^{40} Byte
PentaByte	PB	2^{10} TB = 2^{50} Byte

Đơn vị đo thông tin



Biểu diễn thông tin trong MTĐT

- Đặc điểm
  - Được lưu trong các thanh ghi hoặc trong các ô nhớ. Thanh ghi hoặc ô nhớ có kích thước 1 byte (8 bit) hoặc 1 word (16 bit).
  - Biểu diễn số nguyên không dấu, số nguyên có dấu, số thực và ký tự.
- Hai loại bit đặc biệt
  - msb (most significant bit): bit nặng nhất (bit n)
  - lsb (least significant bit): bit nhẹ nhất (bit 0)

Biểu diễn số nguyên không dấu

- Đặc điểm
  - Biểu diễn các đại lương luôn dương.
  - Ví dụ: chiều cao, cân nặng, mã ASCII…
  - Tất cả bit được sử dụng để biểu diễn giá trị.
  - Số nguyên không dấu 1 byte lớn nhất là $1111\ 1111_2 = 2^8 - 1 = 255_{10}$.
  - Số nguyên không dấu 1 word lớn nhất là $1111\ 1111\ 1111\ 1111_2 = 2^{16} - 1 = 65535_{10}$.
  - Tùy nhu cầu có thể sử dụng số 2, 3… word.
  - lsb = 1 thì số đó là số đó là số lẻ.

Biểu diễn số nguyên có dấu

- Đặc điểm
  - Lưu các số dương hoặc âm.
  - Bit msb dùng để biểu diễn dấu
    - msb = 0 biểu diễn số dương. VD: 0101 0011
    - msb = 1 biểu diễn số âm. VD: 1101 0011
  - Trong máy tính, số âm được biểu diễn ở dạng số bù 2.

Số bù 1 và số bù 2

Số 5 (byte)	0	0	0	0	0	1	0	1
Số bù 1 của 5	1	1	1	1	1	0	1	0
+								1
Số bù 2 của 5	1	1	1	1	1	0	1	1
+ Số 5	0	0	0	0	0	1	0	1
Kết quả	1	0	0	0	0	0	0	0	0

Biểu diễn số nguyên có dấu

- Nhận xét
  - Số bù 2 của x cộng với x là một dãy toàn bit 0 (không tính bit 1 cao nhất do vượt quá phạm vi lưu trữ). Do đó số bù 2 của x chính là giá trị âm của x hay – x.
  - Đổi số thập phân âm –5 sang nhị phân?
    - ➔ Đổi 5 sang nhị phân rồi lấy số bù 2 của nó.
  - Thực hiện phép toán a – b?
    - ➔ a – b = a + (–b) => Cộng với số bù 2 của b.

Tính giá trị có dấu và không dấu

- Tính giá trị không dấu và có dấu của 1 số?
  - Ví dụ số word (16 bit): 1100 1100 1111 0000
  - Số nguyên không dấu ?
    - Tất cả 16 bit lưu giá trị.

      => giá trị là 52464.
  - Số nguyên có dấu ?
    - Bit msb = 1 do đó số này là số âm.

      => độ lớn là giá trị của số bù 2.
    - Số bù 2 = 0011 0011 0001 0000 = 13072.

      => giá trị là –13072.

Tính giá trị có dấu và không dấu

- Bảng giá trị số không dấu/có dấu (byte & word)

msb = 0

msb = 1

HEX	Không dấu	Có dấu
00	0	0
01	1	1
02	2	2
...	...	...
...	...	...
7E	126	126
7F	127	127
80	128	–128
81	129	–127
...	...	...
...	...	...
FE	254	–2
FF	255	–1

HEX	Không dấu	Có dấu
0000	0	0
0001	1	1
0002	2	2
...	...	...
...	...	...
7FFE	32766	32766
7FFF	32767	32767
8000	32768	–32768
8001	32769	–32767
...	...	...
...	...	...
FFFE	65534	–2
FFFF	65535	–1

Tính giá trị có dấu và không dấu

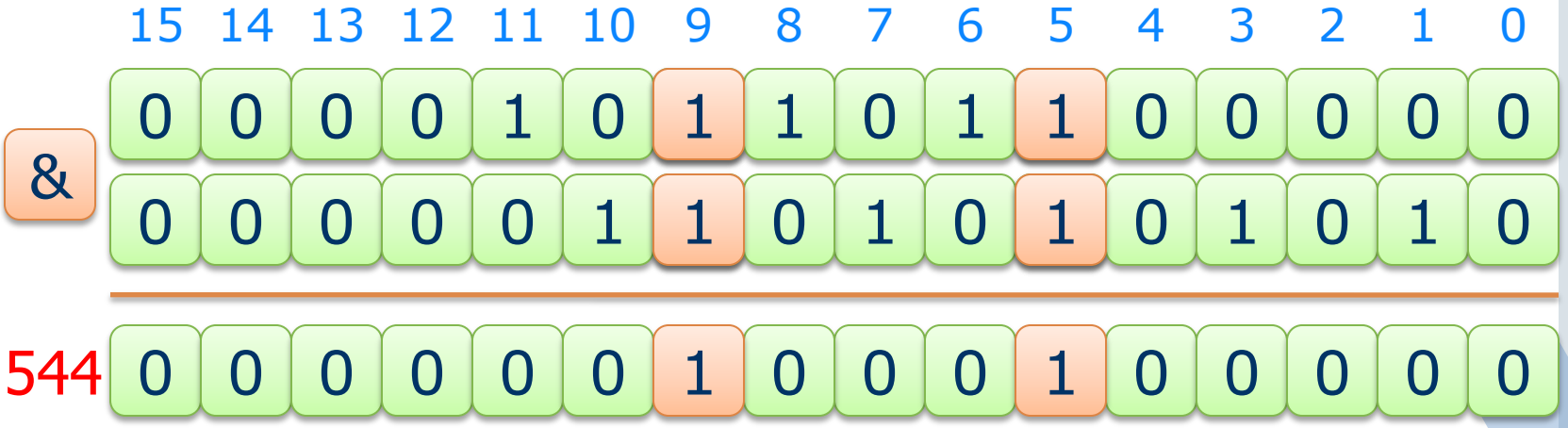
- Nhận xét
  - msb=0 ➔ giá trị có dấu bằng giá trị không dấu.
  - msb=1 ➔ thì giá trị có dấu bằng giá trị không dấu trừ 2^8=256 (byte) hay 2^{16}=65536 (word).
- Tính giá trị không dấu và có dấu của 1 số?
  - Ví dụ số word (16 bit): 1100 1100 1111 0000
  - Giá trị không dấu là 52464.
  - Giá trị có dấu: vì bit msb = 1 nên giá trị có dấu bằng 52464 – 65536 = –13072.

Các toán tử trên bit

- Toán tử & (and)

&	0	1
0	0	0
1	0	**1**

- Ví dụ
  - int x = 2912, y = 1706, z = x & y;

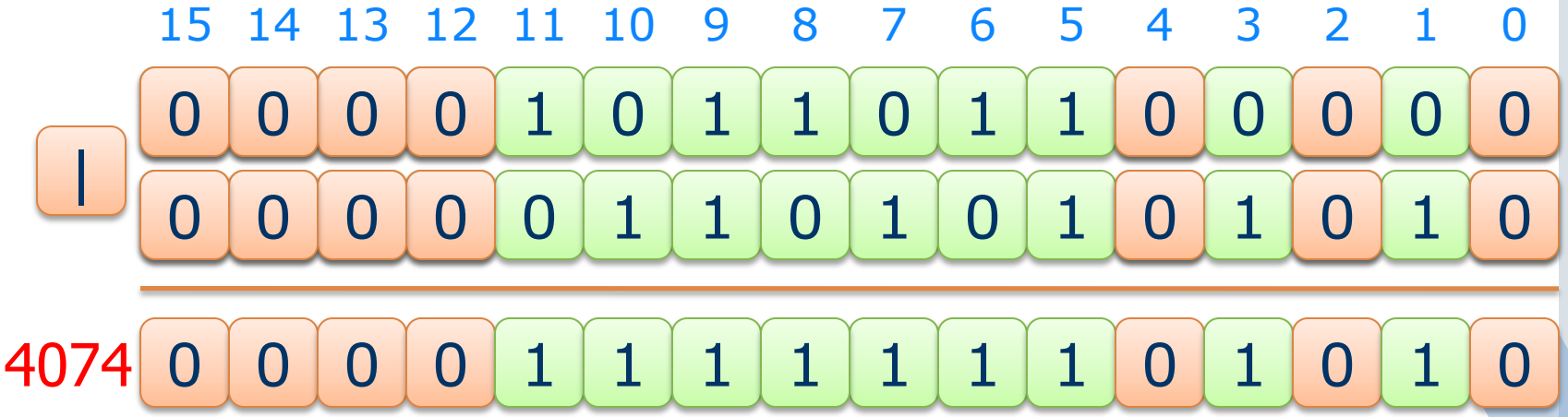


Các toán tử trên bit

- Toán tử | (or)

|	0	1
0	**0**	1
1	1	1

- Ví dụ
  - int x = 2912, y = 1706, z = x | y;

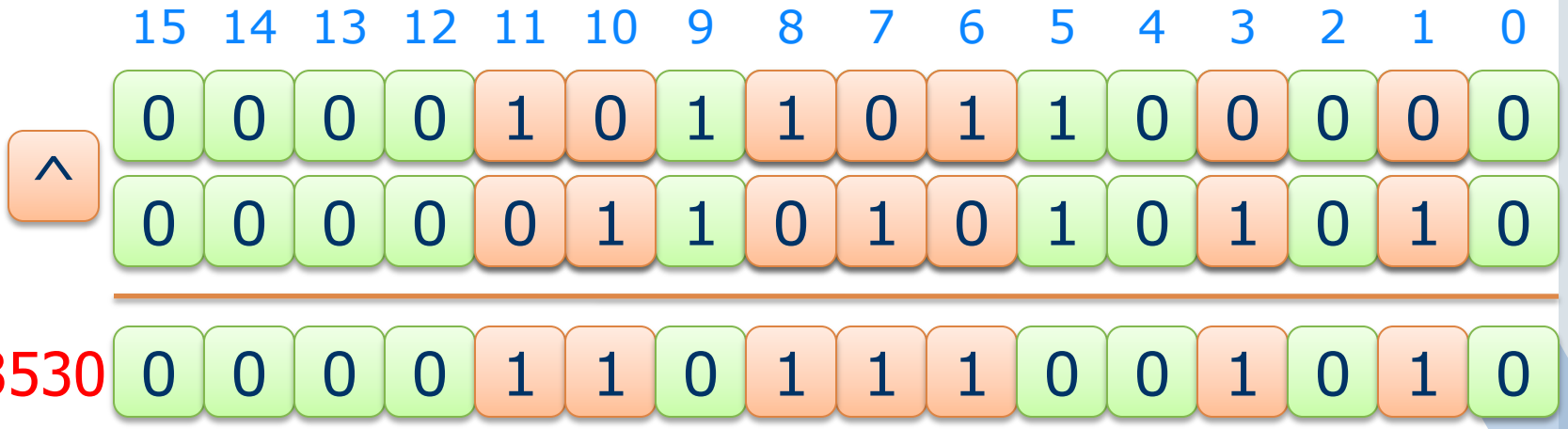


Các toán tử trên bit

- Toán tử ^ (xor)

^	0	1
0	**0**	1
1	1	**0**

- Ví dụ
  - int x = 2912, y = 1706, z = x ^ y;

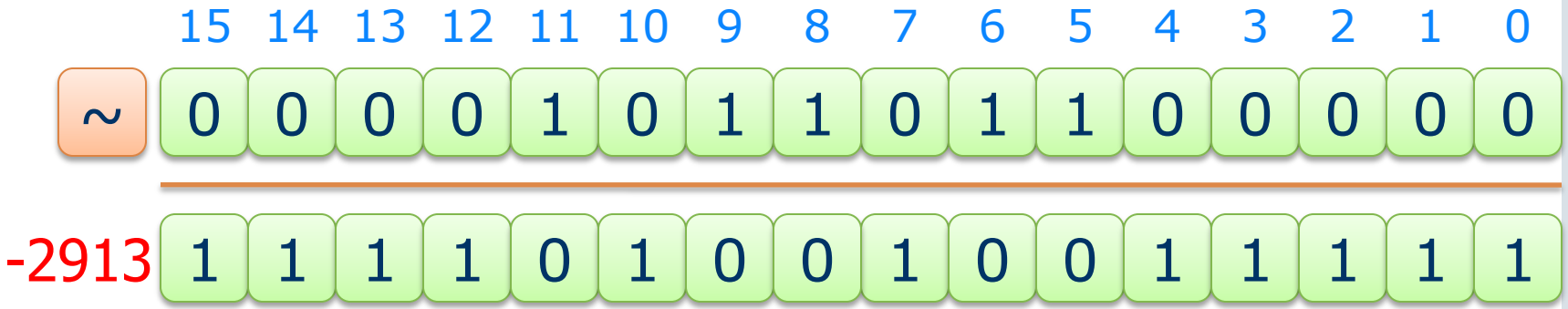


Các toán tử trên bit

- Toán tử ~ (not)

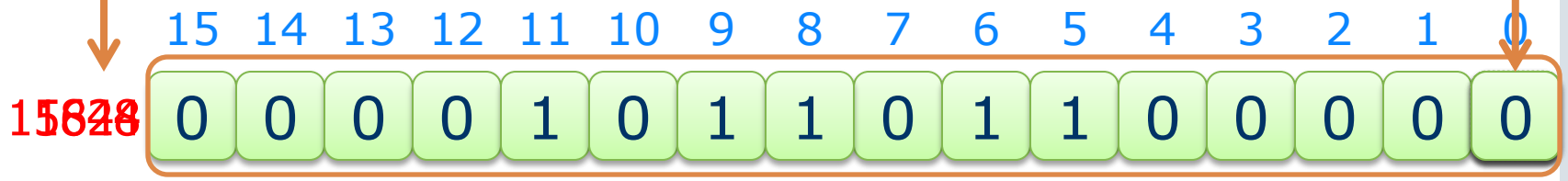
~	0	1
	1	0

- Ví dụ
  - int x = 2912, z = ~x;



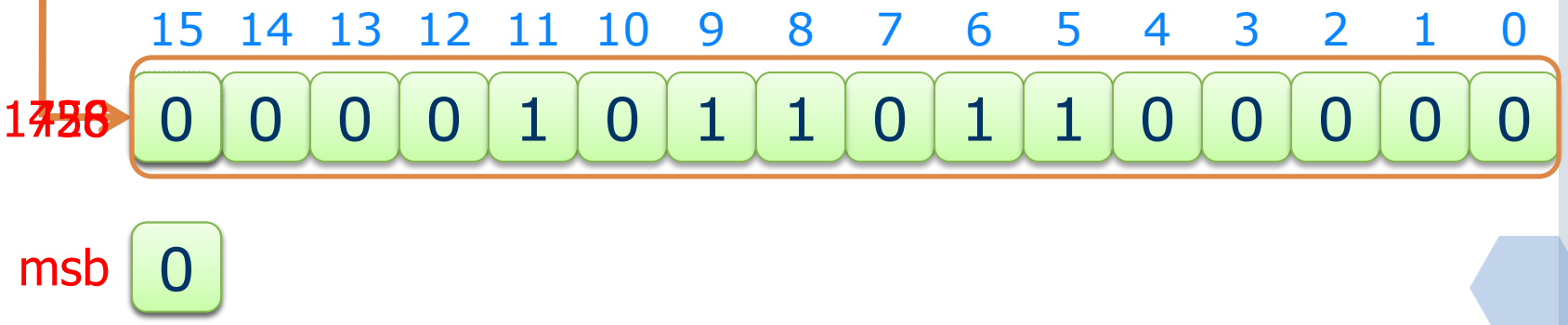
Các toán tử trên bit

- Toán tử << n (shift left)
  - Dịch các bit sang trái n vị trí.
  - Các bit vượt quá phạm vi lưu trữ sẽ mất.
  - Tự động thêm bit 0 vào cuối dãy bit.
- Ví dụ
  - int x = 2912, z = x << 2;



Các toán tử trên bit

- Toán tử >> n (shift right)
  - Dịch các bit sang phải n vị trí.
  - Các bit vượt quá phạm vi lưu trữ sẽ mất.
  - Giữ lại bit nặng nhất (msb) ⇔ dấu của số
- Ví dụ
  - int x = 2912, z = x >> 2;



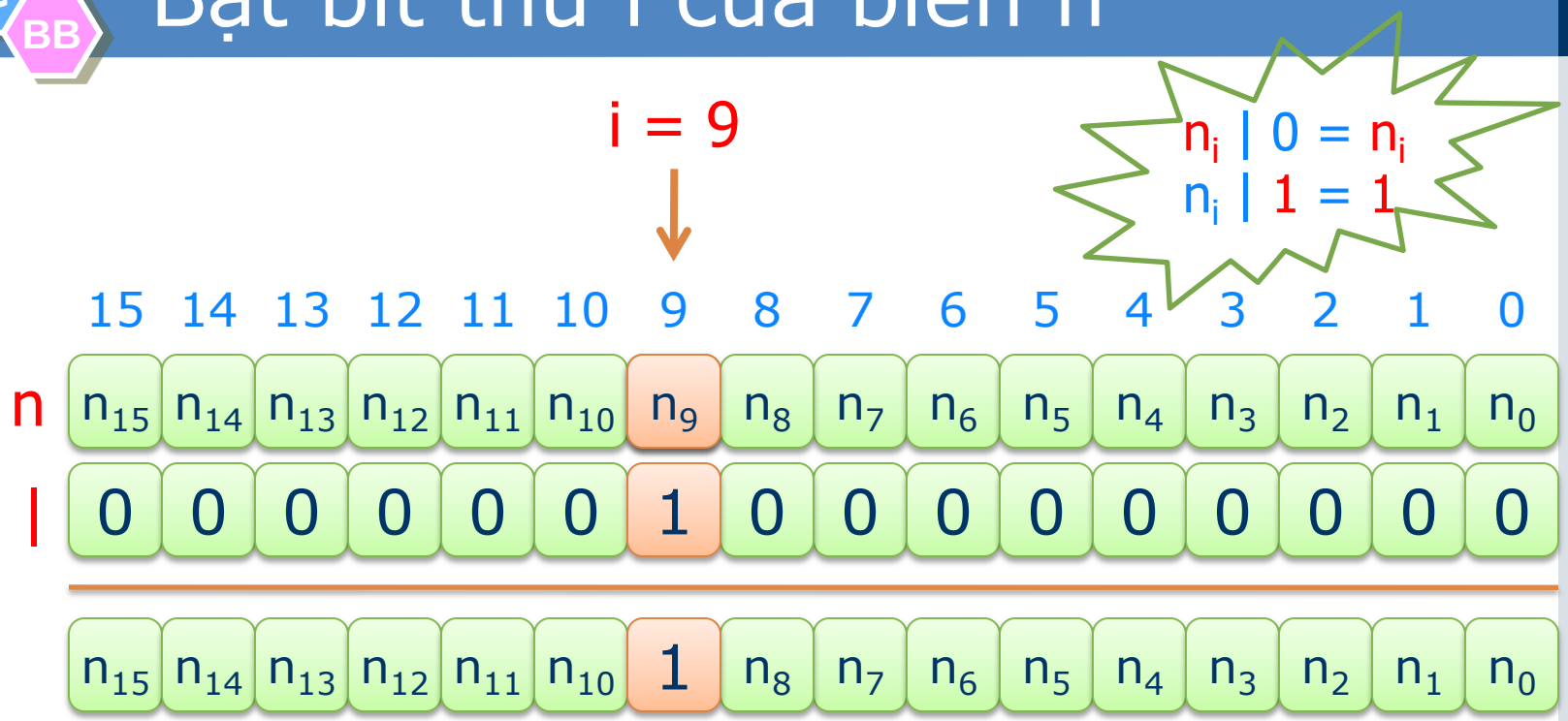
Các toán tử trên bit

- Lưu ý
  - Không được nhầm lẫn các các toán tử trên bit (&, |, ~) với các toán tử kết hợp (&&, || , !)
  - Các toán tử gộp: &= |= ^= <<= >>=
  - Máy tính làm việc trên bit nên các thao tác trên hệ nhị phân sẽ nhanh hơn rất nhiều so với hệ khác.
  - Phải luôn nhớ độ dài của dãy bit đang làm việc (8bit, 16bit, 32bit, 64bit, …)

Ứng dụng trên số nguyên

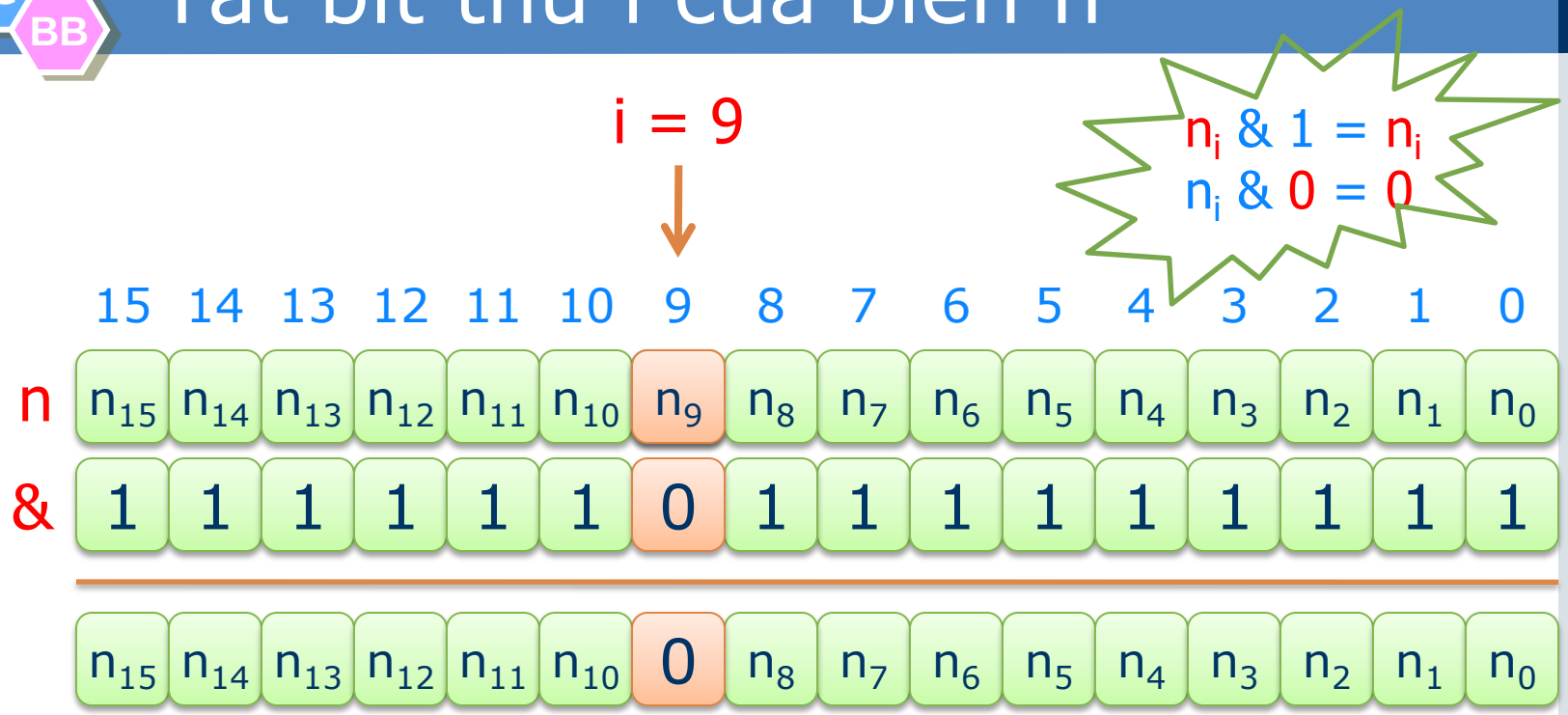
- Ứng dụng của các toán tử &, |, ^, ~
  a. Bật bit thứ i của biến n (onbit)
  b. Tắt bit thứ i của biến n (offbit)
  c. Lấy giá trị của bit thứ i của biến n (getbit)
  d. Gán giá trị 0 cho biến n (setzero)
- Ứng dụng của các toán tử dịch bit << và >>
  e. Nhân n với 2^i (mul2pow)
  f. Chia n với 2^i (div2pow)

Bật bit thứ i của biến n



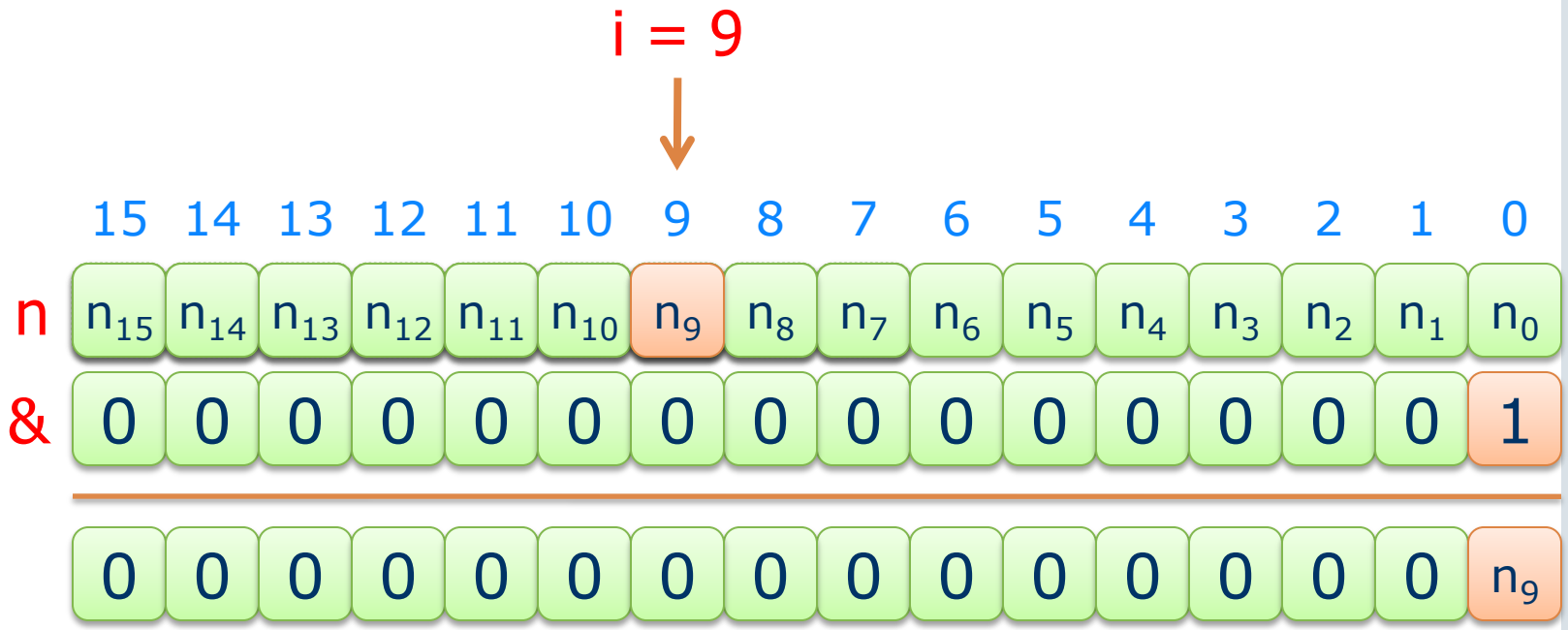

```
void onbit(int &n, int i)
{
        n = n | (0x1 << i);
}
```

Tắt bit thứ i của biến n




```
void offbit(int &n, int i)
{
        n =  n & (~(0x1 << i));
}
```

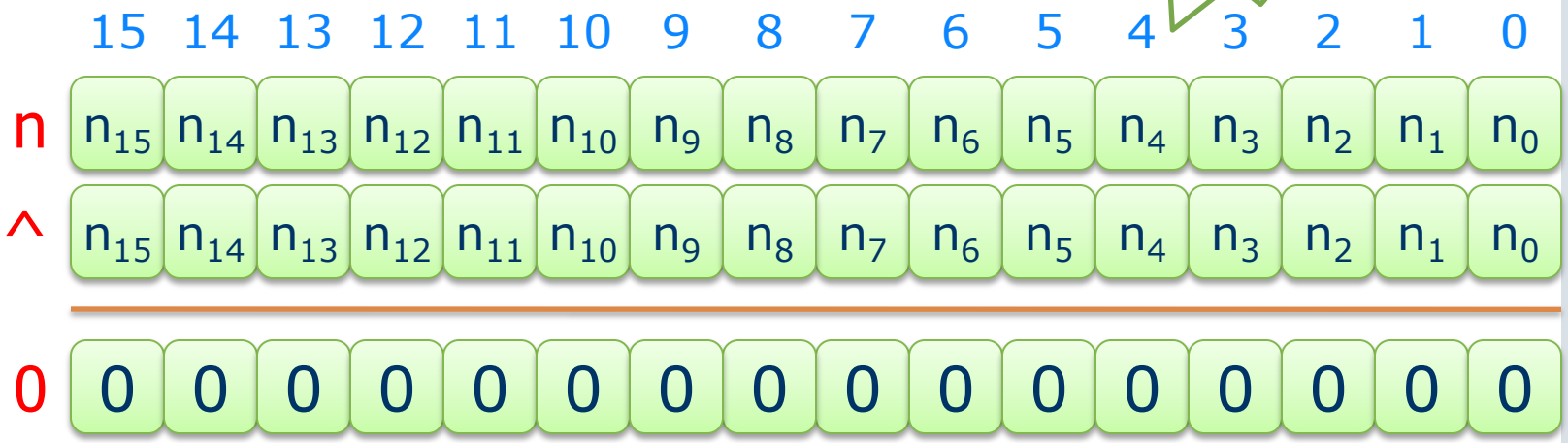
Lấy giá trị bit thứ i của biến n




```
int getbit(int n, int i)
{
        return (n >> i) & 0x1;
}
```

Gán giá trị 0 cho biến n

$n_i \wedge n_i = 0$




```
void setzero(int &n)
{
        n = n ^ n;
}
```

Nhân n với 2^i

- Đặc điểm toán tử <<
  - $n = \Sigma(n_j 2^j)$ với $j \in [0, k]$ (k là chỉ số bit msb)
  - Dịch trái i bit ➔ số mũ mỗi ký số tăng thêm i
  - ➔ $n << i = \Sigma(n_j 2^{j+i}) = 2^i \Sigma(n_j 2^j) = 2^i n$
  - Vậy, dịch trái i bit ⇔ nhân với 2^i

```
int mul2powi(int n, int i)
{
        return n << i;
}
```

Chia n với 2^i

- Đặc điểm toán tử >>
  - $n = \Sigma(n_j 2^j)$ với $j \in [0, k]$ (k là chỉ số bit msb)
  - Dịch phải i bit ➔ số mũ mỗi ký số giảm đi i
  - ➔ $n << i = \Sigma(n_j 2^{j-i}) = 2^{-i}\Sigma(n_j 2^j) = 2^{-i}n = n/2^i$
  - Vậy, dịch phải i bit ⇔ chia cho 2^i

```
int div2powi(int n, int i)
{
        return n >> i;
}
```

Bài tập

- Bài 1: Viết hàm thực hiện các thao tác trên bit.
- Bài 2: Viết bitcount đếm số lượng bit 1 của một số nguyên dương n.
- Bài 3: Cho mảng a gồm n số nguyên khác nhau. Viết hàm liệt kê các tổ hợp 1, 2, …, n phần tử của số nguyên đó (không cần theo thứ tự)

  Ví dụ, n = 3, mảng a = {1, 2, 3}

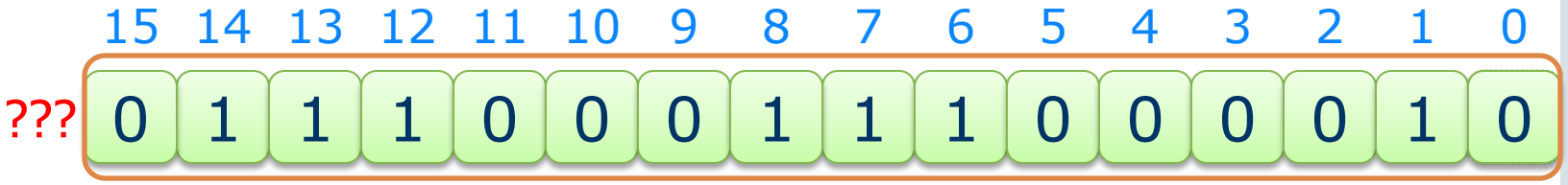
  ➔{1}, {2}, {3}, {1, 2}, {1, 3}, {2, 3}, {1, 2, 3}
- Bài 4: Giống bài 3 nhưng chỉ liệt kê các tổ hợp k phần tử ($1 \le k \le n$)

Bài tập

- Bài 5: Viết hàm RotateLeft(n, i) thực hiện thao tác "xoay" các bit của n (kô dấu) sang trái i vị trí và các bit bị mất sẽ được đưa vào cuối dãy bit.

  Ví dụ:

  - int n = 291282; n = RotateLeft(n, 2);



- Bài 6: Tương tự bài 2 nhưng viết hàm RotateRight(n, i) để xoay bit sang phải.

Bài 3