



東邦大学

いのち  
生命の科学で未来をつなぐ

# 乗算の原理



加減算と同じように  
まず10進で考える

$$43 \times 17 = ? ?$$



# まず

## 1 桁同士の掛け算は九九で覚えている

	0	1	2	...	9
0					
1					
2					
:					
9					

2桁×1桁だと



# 複数桁×1桁だと

上段の各桁と下段の1桁を掛け、  
ずらした位置へ合せて置き、  
足す

$$\begin{array}{r} \times \quad 347 \\ \hline 21 \\ + 28 \\ \hline 238 \end{array}$$

これは

$$\begin{aligned} & ((3 \times 10) + 4) \times 7 \\ = & (3 \times 10) \times 7 + (4 \times 7) \\ = & (3 \times 7) \times 10 + (4 \times 7) \end{aligned}$$

×	3	4	
		7	
	2	1	0
		2	8
+			
	2	3	8

$(3 \times 7) \times 10$   
 $(4 \times 7)$

これは

上段の各桁と下段の1桁を掛け、  
ずらした位置へ合せて置き、  
足す

21を左に1桁  
ずらす意味は  
 $(3 \times 7) \times 10$

$$\begin{array}{r} \times \quad 34 \\ \hline 210 \\ + \quad 28 \\ \hline 238 \end{array}$$

$$\begin{array}{l} (3 \times 7) \times 10 \\ (4 \times 7) \end{array}$$



では、2桁×2桁だと



東邦大学

# 複数桁×複数桁だと

下段の各桁と上段を掛け、  
ずらした位置へ合せて置き、  
足す

$$\begin{array}{r} \times \quad \quad 34 \\ \hline \quad \quad 17 \\ \hline + \quad \quad 238 \\ \hline \end{array} \quad (34 \times 7)$$

# 複数桁×複数桁だと

下段の各桁と上段を掛け、  
ずらした位置へ合せて置き、  
足す

$$\begin{array}{r} \times \quad \quad \quad \begin{array}{cc} 3 & 4 \\ 1 & 7 \end{array} \\ \hline \begin{array}{ccc} 2 & 3 & 8 \\ + & 3 & 4 & 0 \end{array} \\ \hline \end{array} \quad \begin{array}{l} (34 \times 7) \\ (34 \times 1) \times 10 \end{array}$$

# 複数桁×複数桁だと

$$\begin{aligned} & 34 \times ( (1 \times 10) + 7 ) \\ = & 34 \times 7 + ( 34 \times (1 \times 10) ) \\ = & 34 \times 7 + (34 \times 1) \times 10 \end{aligned}$$

		3	4	
×		1	7	
<hr/>				
		2	3	8
+		3	4	0
<hr/>				
		5	7	8

(34×7)  
(34×1)×10

# 2進数だと？



東邦大学

## 2進数でも同じこと

### 1 桁同士の掛け算は九九で覚えている

	0	1
0	0	0
1	0	1

# 複数桁×1桁だと

上段の各桁と下段の1桁を掛け、  
ずらした位置へ合せて置き、  
足す

$$\begin{array}{r} \times \\ \hline 10 \\ 1 \\ \hline \end{array}$$
$$\begin{array}{r} + \\ \hline 10 \\ 10 \\ \hline \end{array}$$

→  
というより

# 複数桁×1桁だと

下段の1桁が、

1 ⇒ 上段をそのままコピー  
0 ⇒ 0

$$\begin{array}{r} \times \quad 1 \ 0 \\ \hline \quad 0 \end{array} \qquad \begin{array}{r} \times \quad 1 \ 0 \\ \hline \quad 1 \end{array}$$

Red dashed boxes highlight the '1 0' pairs in the second multiplication, with a red arrow pointing from the top box to the bottom box.



# 複数桁×複数桁だと

下段の各桁と上段を掛け、  
ずらした位置へ合せて置き、

足す

$$\begin{array}{r} \times \quad 10 \\ \hline 10 \\ \hline + \\ \hline \end{array}$$

10進と  
同じ

# 複数桁×複数桁だと

下段の各桁と上段を掛け、  
ずらした位置へ合せて置き、  
足す

$$\begin{array}{r} \times \quad 10 \\ \hline 10 \\ + \quad 100 \\ \hline 100 \end{array}$$

10進と  
同じ



# もう1つサンプル

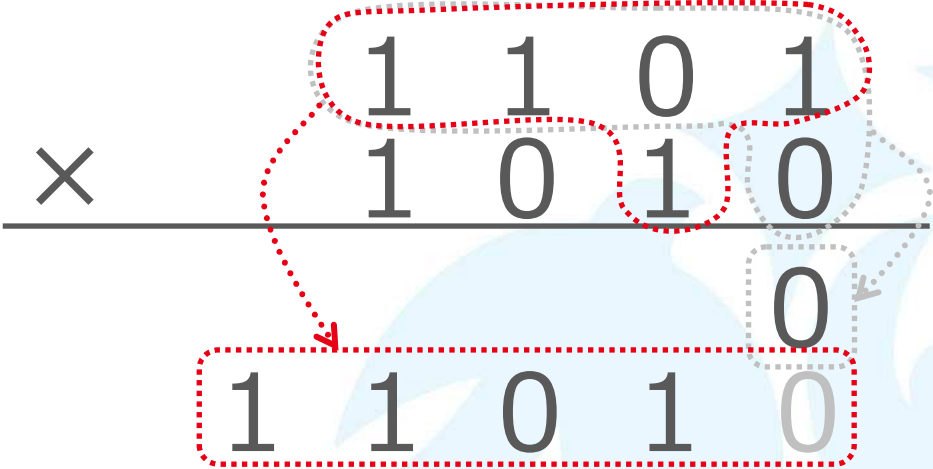
$$\begin{array}{r} \times \quad 1 \quad 1 \quad 0 \quad 1 \\ \hline 1 \quad 0 \quad 1 \quad 0 \end{array}$$

# もう1つサンプル

$$\begin{array}{r} \times \quad \begin{array}{cccc} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{array} \\ \hline \end{array}$$

0を掛ける⇒0

# もう1つサンプル



1を掛ける⇒  
1101

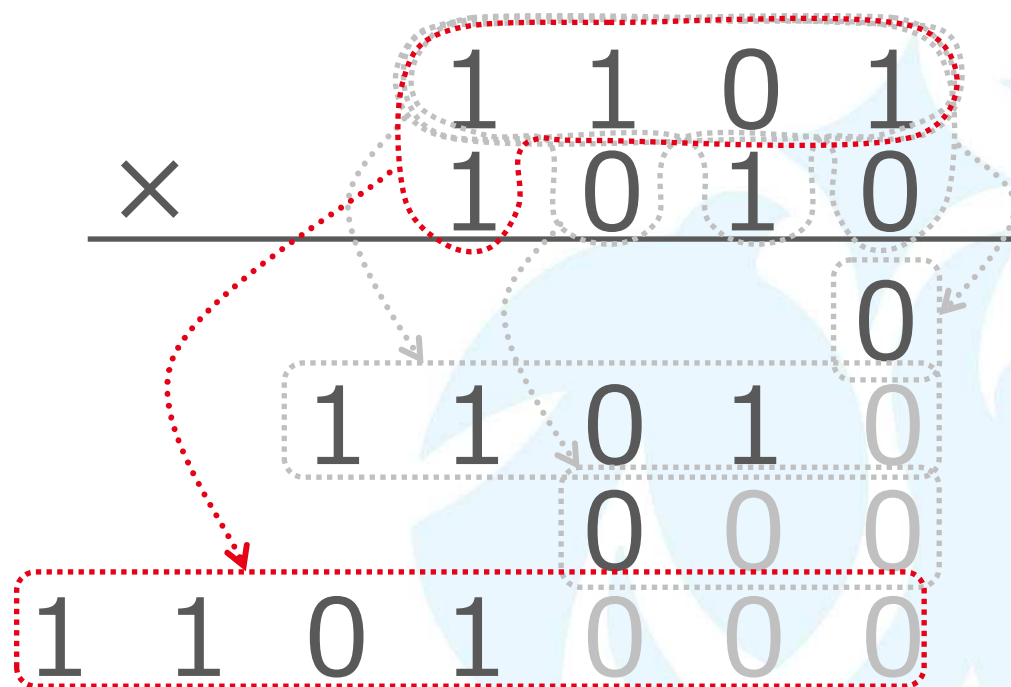
# もう1つサンプル

$$\begin{array}{r} \times \quad 1101 \\ \hline 1010 \\ 1101 \\ 0000 \end{array}$$

The diagram illustrates the multiplication of the binary numbers 1101 and 1010. A horizontal line is drawn under the second number. The first row shows the multiplicand 1101. The second row shows the multiplier 1010. The third row shows the first partial product, 1101, which is shifted one position to the left. The fourth row shows the second partial product, 0000, which is shifted two positions to the left. Red dashed boxes highlight the 1101 in the first row, the 1010 in the second row, the 1101 in the third row, and the 0000 in the fourth row. Dotted lines with arrows show the alignment of the numbers.

0を掛ける⇒0

# もう1つサンプル



1を掛ける⇒  
1101

# もう1つサンプル

$$\begin{array}{r} \times \\ \hline 1101 \\ 1010 \\ \hline 0000 \\ 1101 \\ 0000 \\ \hline 11010000 \\ + \\ \hline 100000010 \end{array}$$

足す



# 練習問題

$$\begin{array}{r} \times \quad 1\ 0\ 0\ 1 \\ \hline 1\ 1\ 0\ 0 \end{array}$$

$$\begin{array}{r} \times \quad 1\ 0\ 1 \\ \hline 1\ 1\ 1\ 1 \end{array}$$

$$\begin{array}{r} \times \quad 1\ 1\ 0\ 0 \\ \hline 1\ 1 \\ 1\ 1 \end{array}$$

$$\begin{array}{r} \times \quad 1\ 1\ 1\ 1 \\ \hline 1\ 0\ 1 \end{array}$$

# 練習問題

$$\begin{array}{r} \times \quad \quad \quad 1\ 0\ 0\ 1 \\ \quad \quad \quad 1\ 1\ 0\ 0 \\ \hline 1\ 1\ 0\ 1\ 1\ 0\ 0 \end{array}$$

$$\begin{array}{r} \times \quad \quad \quad 1\ 0\ 1 \\ \quad \quad \quad 1\ 1\ 1\ 1 \\ \hline 1\ 0\ 0\ 1\ 0\ 1\ 1 \end{array}$$

$$\begin{array}{r} \times \quad \quad \quad \quad \quad 1\ 1 \\ \quad \quad \quad 1\ 1\ 0\ 0 \\ \hline \quad \quad 1\ 0\ 0\ 1\ 0\ 0 \end{array}$$

$$\begin{array}{r} \times \quad \quad \quad \quad \quad 1\ 0\ 1\ 0 \\ \quad \quad \quad \quad \quad 1\ 0\ 1 \\ \hline \quad \quad 1\ 1\ 0\ 0\ 1\ 0 \end{array}$$

やり方はわかった



東邦大学

これをコンピュータで実現するには？

# これをコンピュータで実現するには？

ハードで実現

ソフトで実現

# これをコンピュータで実現するには？

- ハードで実現  
ソフトに比べて高速
- ソフトで実現

# これをコンピュータで実現するには？

- ハードで実現
  - ソフトに比べて高速
  - ハード (回路)の規模が大きくなる
- ソフトで実現

# これをコンピュータで実現するには？

ハードで実現

ソフトに比べて高速

ハード (回路) の規模が大きくなる

昔は実現困難、今は十分可能

ソフトで実現



# これをコンピュータで実現するには？

ハードで実現

ソフトに比べて高速

ハード (回路)の規模が大きくなる

昔は実現困難、今は十分可能

ソフトで実現

さまざまな工夫で高速化

# これをコンピュータで実現するには？

ハードで実現

ソフトに比べて高速

ハード (回路) の規模が大きくなる

昔は実現困難、今は十分可能

ソフトで実現

さまざまな工夫で高速化

# これをコンピュータで実現するには？

ハードで実現

ソフトに比べて高速

ハード (回路) の規模が大きくなる

昔は実現困難、今は十分可能

ソフトで実現

さまざまな工夫で高速化

ハード追加不要 ~ 実現容易・安価

# これをコンピュータで実現するには？

ハードで実現

ソフトに比べて高速

ハード (回路) の規模が大きくなる

昔は実現困難、今は十分可能

さまざまな工夫で高速化

ソフトで実現

ハード追加不要 ~ 実現容易・安価

ハードより遥かに低速 低速な用途のみ

## 乗算はここまでにします

乗算ハードウェアに興味がある人は  
ブースアルゴリズム（教科書参照）  
ウォレストリー(Wallace Tree)  
を勉強すると良い

## 除算(割り算)は省略します

引き戻し法・引き放し法（教科書参照）  
を勉強すると良い

おまけ

2進掛け算プログラムを書いてみよう

おまけ

## 2進掛け算プログラムを書いてみよう

ポイントは

掛け算の手順のとおり  
プログラムで処理する

# おまけ

## 2進掛け算プログラムを書いてみよう

ポイントは

掛け算の手順のとおり  
プログラムで処理する

昔（ハードの集積度が低い時代）は  
掛け算はプログラムで処理していた



# おまけ

## 2進掛け算プログラムを書いてみよう

上段の数	X	1	0	1
下段の数	Y	x	1	0
結果を	Z	<hr/>		

# おまけ

## 2進掛け算プログラムを書いてみよう

上段の数	X	1	0	1
下段の数	Y	x	1	0
結果を	Z	<hr/>		

下段の数を  
1桁  
切り出す

# おまけ

## 2進掛け算プログラムを書いてみよう

上段の数	X	1	0	1
下段の数	Y	x	1	0
結果を	Z	<hr/>		

もし0なら  
何もしない

# おまけ

## 2進掛け算プログラムを書いてみよう

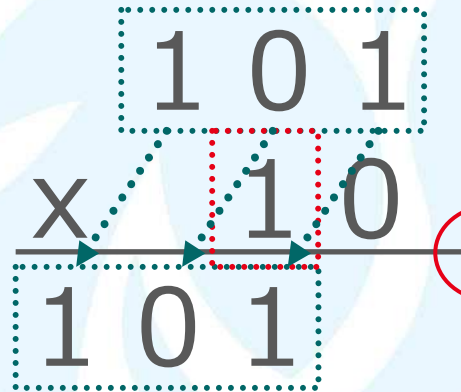
上段の数	X	1	0	1
下段の数	Y	x	1	0
結果を	Z	<hr/>		

もし1なら  
Xの値を

# おまけ

## 2進掛け算プログラムを書いてみよう

上段の数 X  
下段の数 Y  
結果を Z



Yの桁の  
位置に  
合わせて  
左へ移動し

# おまけ

## 2進掛け算プログラムを書いてみよう

上段の数	X	1	0	1	
下段の数	Y	x	1	0	
結果を	Z	1	0	1	0

Zに  
足し込む

# おまけ

## 2進掛け算プログラムを書いてみよう

上段の数	X	1	0	1	
下段の数	Y	X	1	0	
結果を	Z	1	0	1	0

Yの  
次(左)の桁  
へ移る

では、プログラム



# プログラムの概要は

XとYは用意されている

結果 Z は 0 にしておく

今のYの位置を下から i 桁目とする

i が 1 から N(桁数: 16) までループ

$p \leftarrow Y$  の i 桁目を取り出す

もし p が 1 なら

$q \leftarrow X$  を i 桁分だけ左へずらす

Z に q を加える

もし p が 0 ならなにもしない

i を 1 進めて次の繰り返し

# この中で見たことのない操作は

XとYは用意されている

結果 Z は 0 にしておく

今のYの位置を下から i 桁目とする

i が 1 から N(桁数: 16) までループ

p ← Yの i 桁目を取り出す

もし p が 1 なら

q ← Xを i 桁分だけ左へずらす

Z に q を加える

もし p が 0 ならなにもしない

i を 1 進めて次の繰り返し

# i 桁目を取り出す操作は

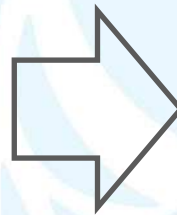


i 桁目を取り出す操作は  
「ビットごとのAND」を使う

# i 桁目を取り出す操作は 「ビットごとのAND」を使う

## ANDの動作の復習

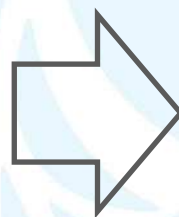
	x	0	1
y			
0		0	0
1		0	1



# i 桁目を取り出す操作は 「ビットごとのAND」を使う

## ANDの動作の復習

	x	0	1
y			
0		0	0
1		0	1



## ビットの抽出

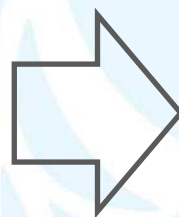
	x	0	1
y			
0		0	0
1		0	1

常に0なら

# i 桁目を取り出す操作は 「ビットごとのAND」を使う

ANDの動作の復習

	x	0	1
y			
0		0	0
1		0	1



ビットの抽出

	x	0	1
y			
0		0	0
1		0	1

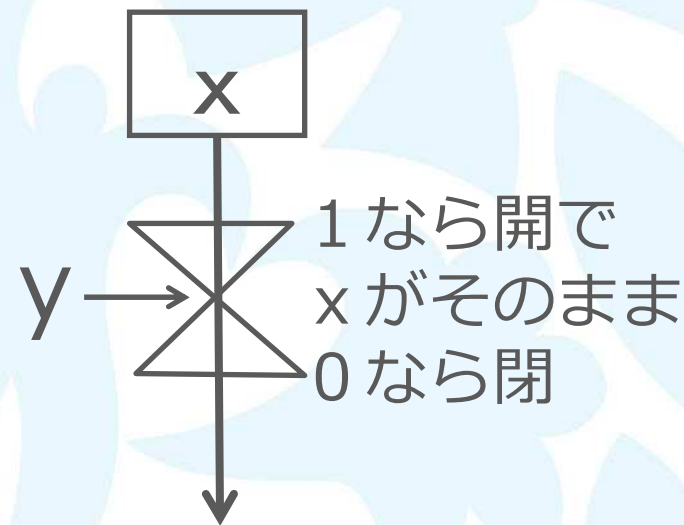
そのyが1ならば

# つまり「ビットAND」は 0/1のバルブ(弁)の役目

## ビットの抽出

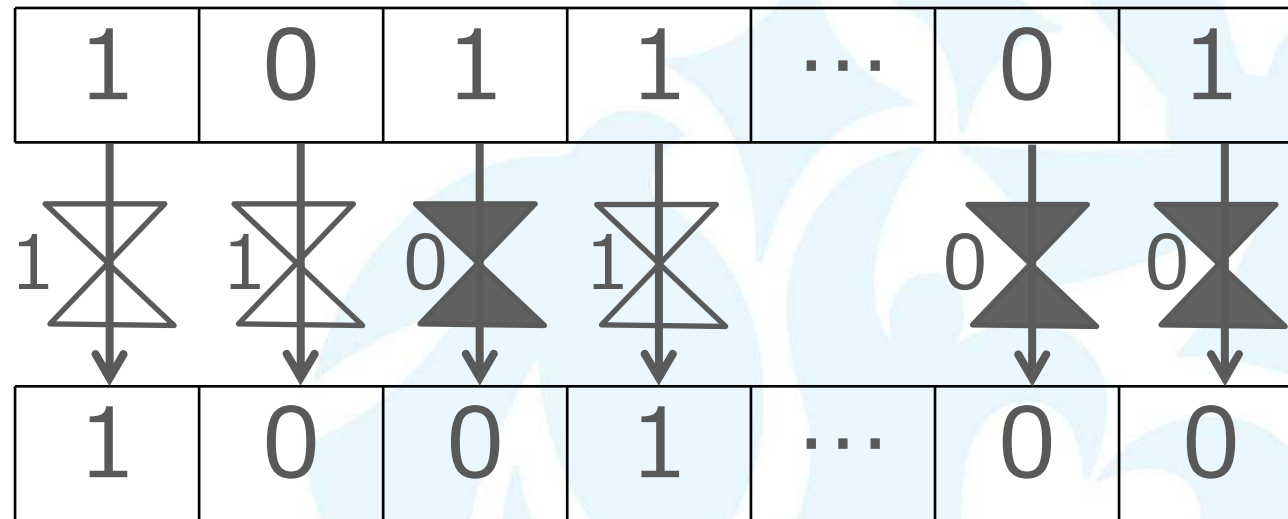
	x	0	1
y			
0	xがそのま まなら	0	0
1	yが0なら 常に0	0	1

常に0  
yが0なら





# 「ビットごとのAND」を並べる



そのまま

そのまま

0

そのまま

0

0



東邦大学

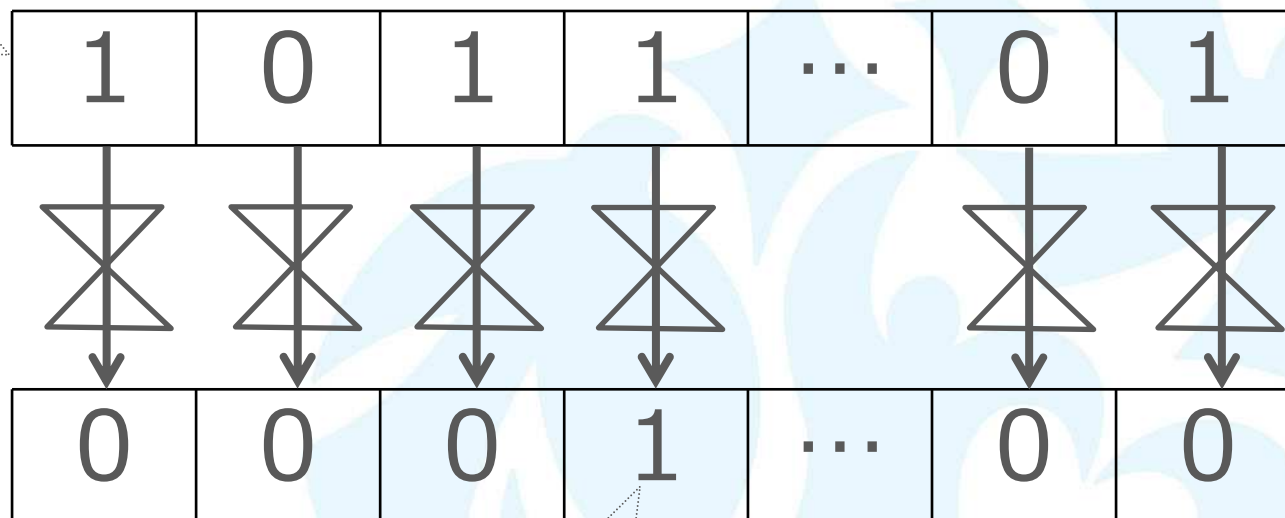
では、 $i$  桁目を取り出すには



東邦大学

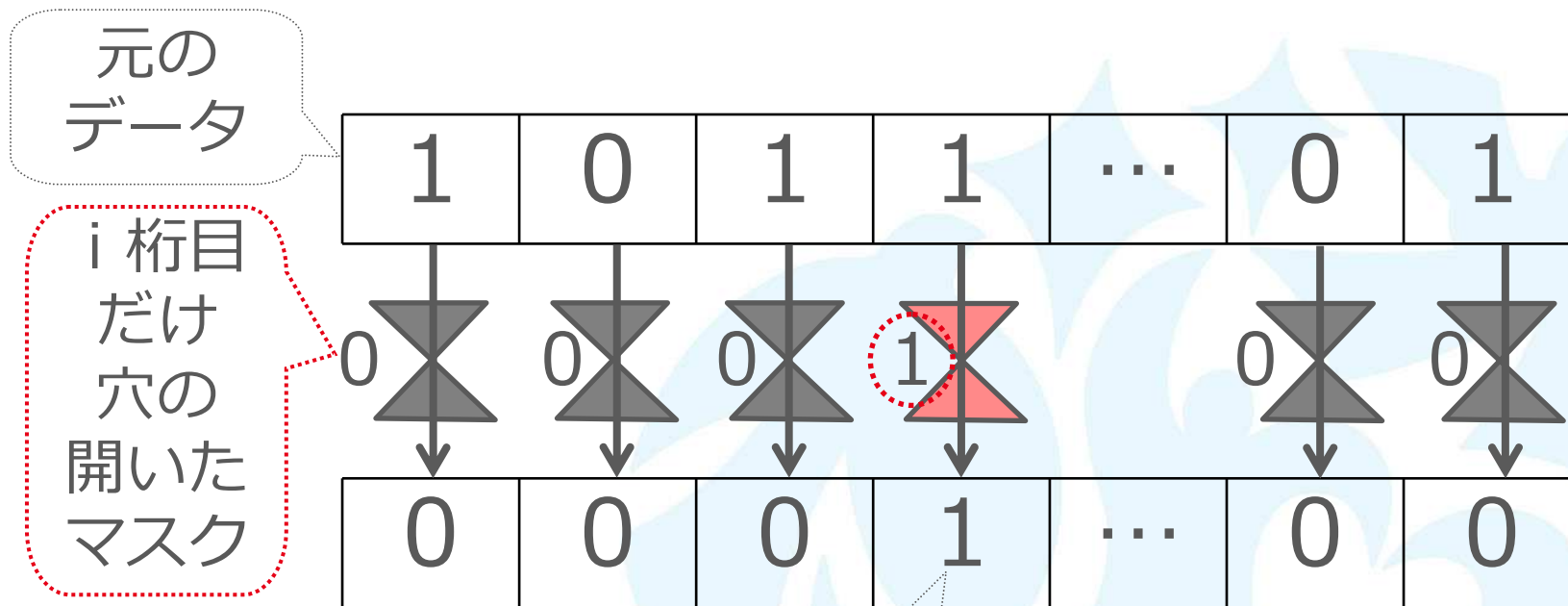
では、 $i$  桁目を取り出すには

元の  
データ



$i$  桁目だけ  
取り出した  
データだけ

# i 桁目だけ穴のあいたマスクを掛ける

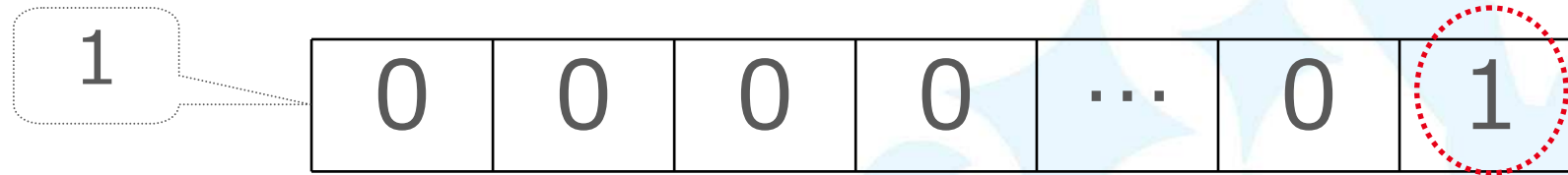


i 桁目だけ  
取り出した  
データだけ

# i 桁目だけ穴の開いたマスクを 作るには



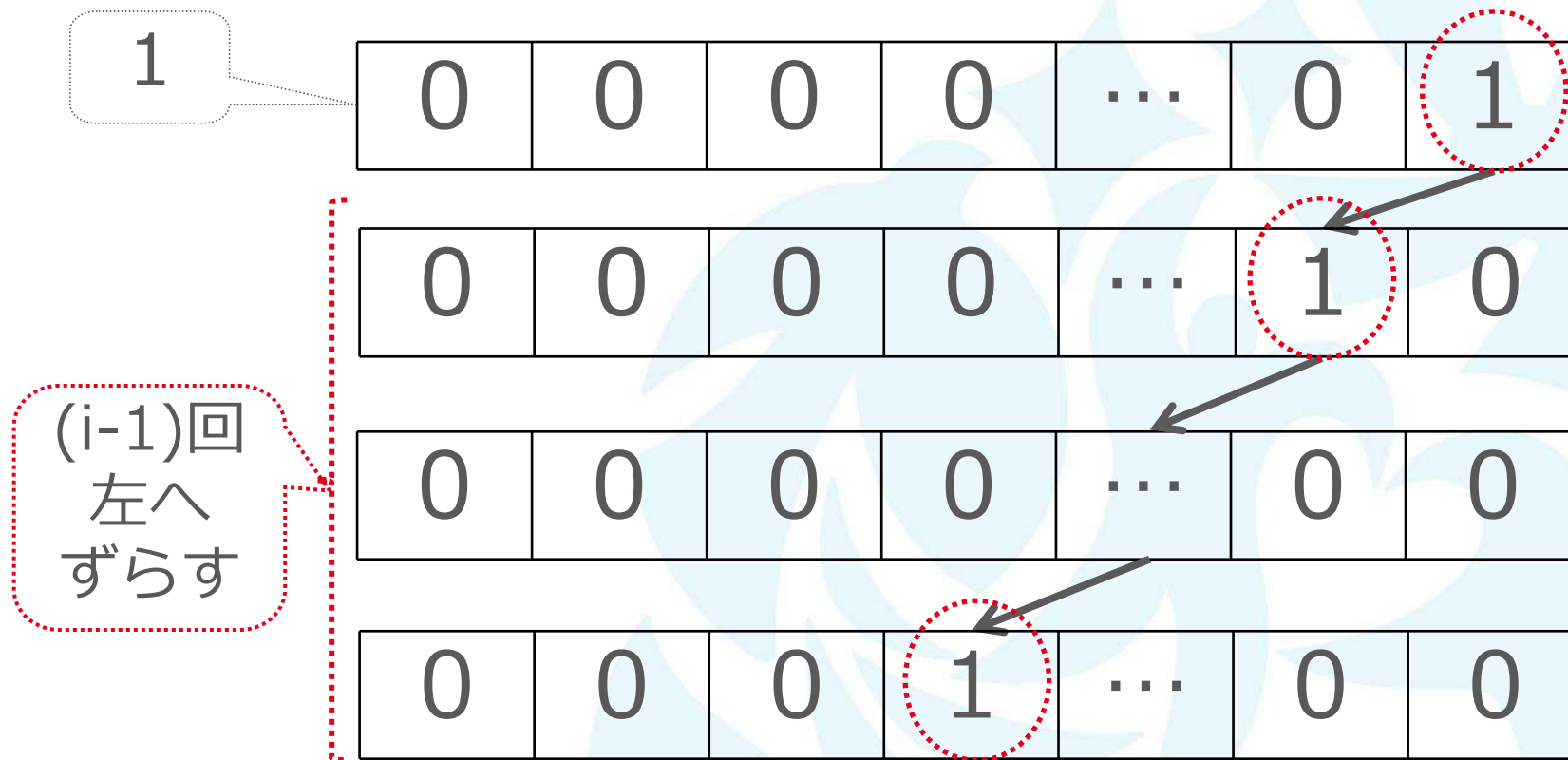
# i 桁目だけ穴の開いたマスクを作るには



i 桁目だけ穴の開いたマスク



# i 桁目だけ穴の開いたマスクを作るには



i 桁目だけ穴の開いたマスク

JavaやC言語で  
左へN回ずらすには

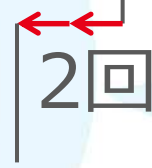


JavaやC言語で  
左へN回ずらすには

## シフト演算

$x = 0000 \cdots 0001$

$y = x \ll 2$



$y$  は  $0000 \cdots 0100$

JavaやC言語で  
左へN回ずらすには

## シフト演算

$x = 0000 \cdots 0001$

$y = x \ll N$



$y$  は  $0001 \cdots 0000$

## もう一度全体の流れは

XとYは用意されている

結果 Z は 0 にしておく

今のYの位置を下から i 桁目とする

i が 1 から N(桁数: 16) までループ

p ← Yの i 桁目を取り出す

もし p が 1 なら

q ← Xを i 桁分だけ左へずらす

Z に q を加える

もし p が 0 ならなにもしない

i を 1 進めて次の繰り返し



# プログラムに置き換えよう

XとYは用意されている  
結果 Z は 0 にしておく

今のYの位置を下から i 桁目  
i が 1 から 16 までループ

p ← Yの i 桁目を取り出す

もし p が 1 なら

q ← Xを i 桁左へずらす

Z に q を加える

p が 0 なら何もしない

i を 1 進めて次の繰り返し

```
Z = 0;
```

```
for (i=1; i<=16; i++){
```

```
    mask = 1 << (i-1);
```

```
    p = Y & mask;
```

```
    if (p!=0){
```

```
        q = X << (i-1);
```

```
        Z = Z+q;
```

```
    }
```

```
}
```



のようによすればできる  
実行して試してみよう

# 試してみた

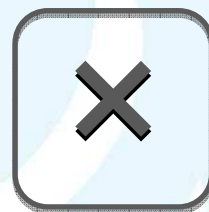
```
#include <stdio.h>
void main() {
    int x, y, z, mask, p, q, i;

    x = 3; y = 5;
    z = 0;
    for (i=1; i<=16; i++) {
        mask = 1<<(i-1);
        p = y & mask;
        if (p!=0) {
            q = x<<(i-1);
            z = z+q;
        }
    }
    printf(
        "%d*d=%d¥n", x, y,
        z);
}
```

```
class mult{
    public static void main(String[] args) {
        int x, y, z, mask, p, q, i;
        x = 3; y = 5;
        z = 0;
        for (i=1; i<=16; i++) {
            mask = 1<<(i-1);
            p = y & mask;
            if (p!=0) {
                q = x<<(i-1);
                z = z+q;
            }
        }
        System.out.printf (
            "%d*d=%d¥n", x, y, z);
    }
}
```



掛け算を理解できましたか？



↓  
次へ