東邦大学

生命（いのち）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

東邦大学

$43 \times 17 = ??$

東邦大学

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					

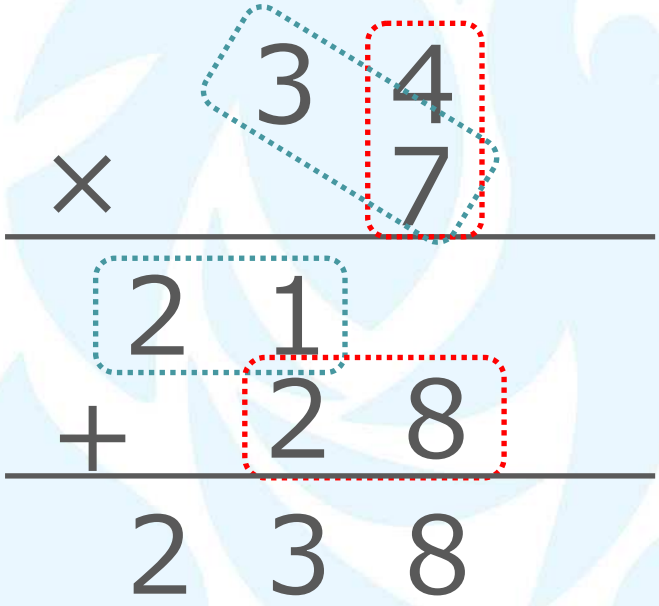
東邦大学

２桁×１桁だと

東邦大学

複数桁×１桁だと

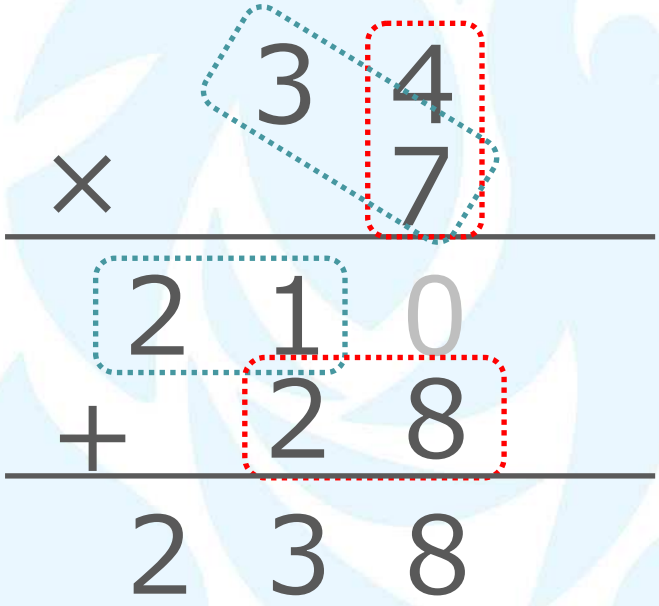
上段の各桁と下段の１桁を掛け、
ずらした位置へ合せて置き、
足す



東邦大学

これは

$$
\begin{aligned}
&((3\times10) + 4) \times 7 \\
=\ &(3\times10) \times 7 + (4 \times 7) \\
=\ &(3\times7) \times 10 + (4 \times 7)
\end{aligned}
$$



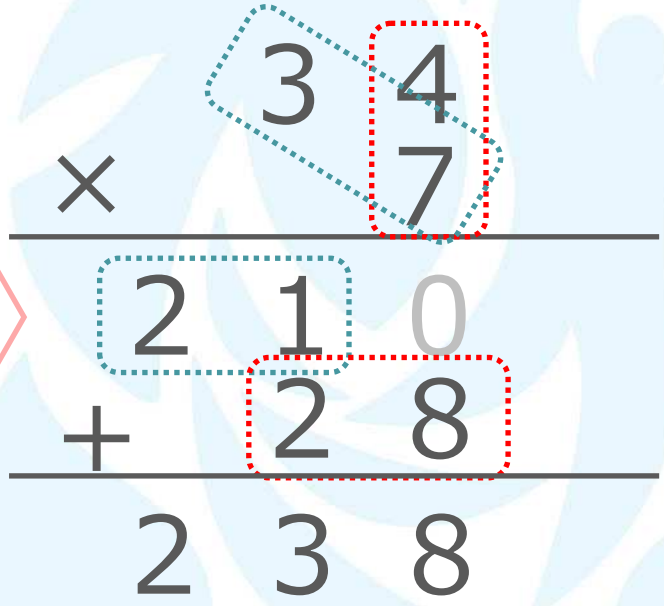
(3×7) × 10
(4×7)

東邦大学

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
(3×7) × 10



(3×7) × 10
(4×7)

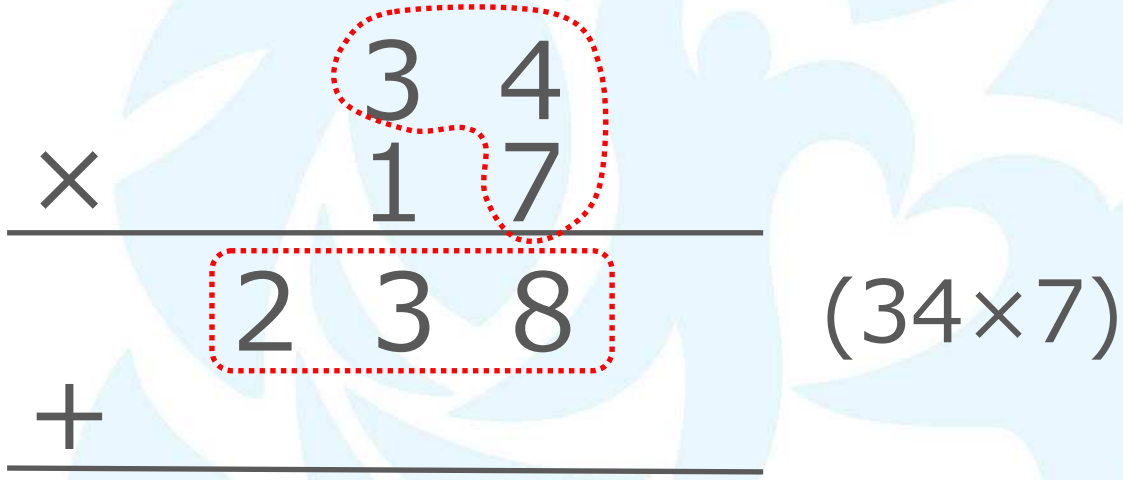
東邦大学

では、２桁×２桁だと

東邦大学

複数桁×複数桁だと

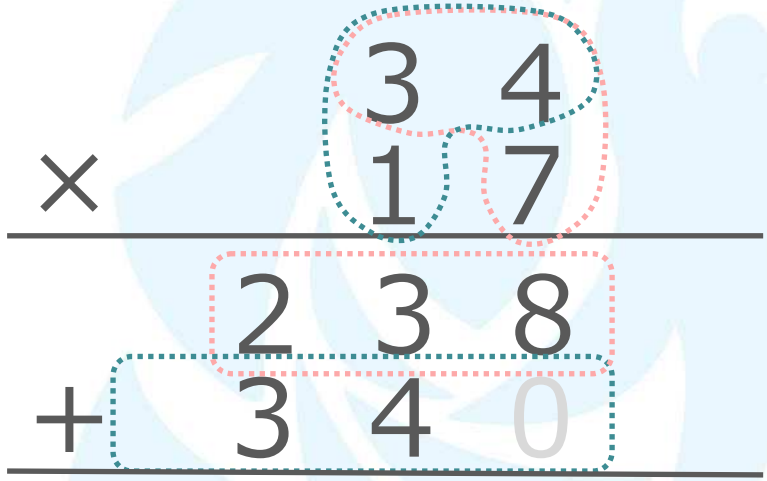
下段の各桁と上段を掛け、
ずらした位置へ合せて置き、
足す



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



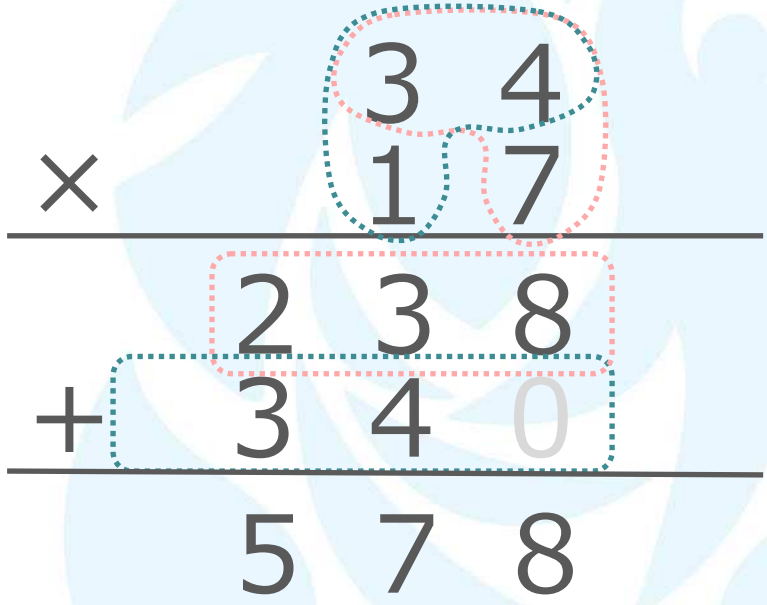
(34×7)

(34×1)×10

東邦大学

複数桁×複数桁だと

$$
\begin{aligned}
&34 \times ((1\times10) + 7) \\
=\ &34 \times 7 + (34 \times (1\times10)) \\
=\ &34 \times 7 + (34\times1) \times 10
\end{aligned}
$$



(34×7)

(34×1)×10

東邦大学

２進数だと？

２進数でも同じこと

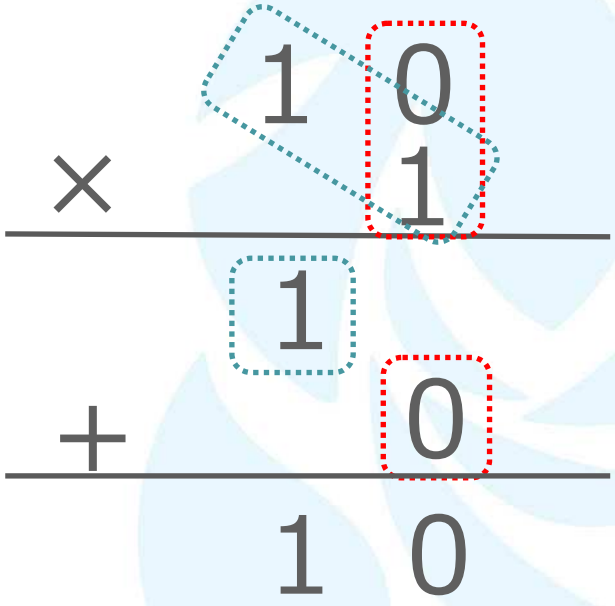
１桁同士の掛け算は九九で覚えている

	0	1
0	0	0
1	0	1

東邦大学

複数桁×１桁だと

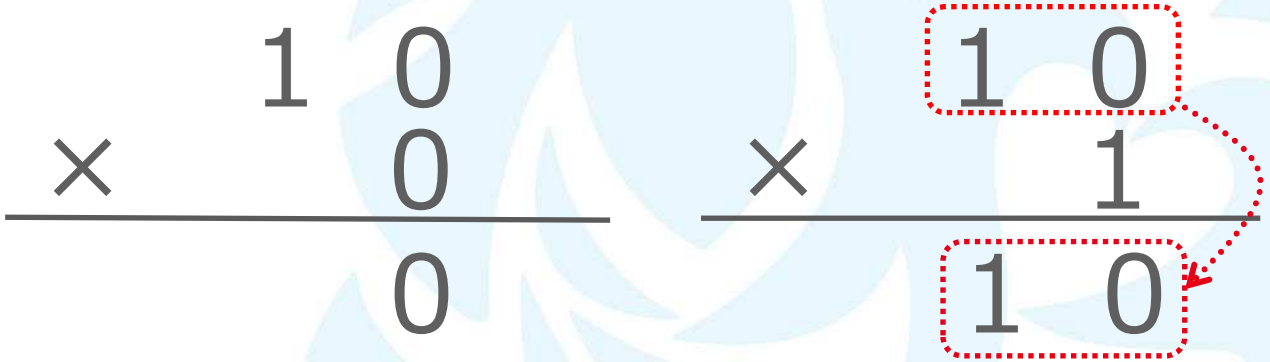
上段の各桁と下段の１桁を掛け、
ずらした位置へ合せて置き、
足す



というより

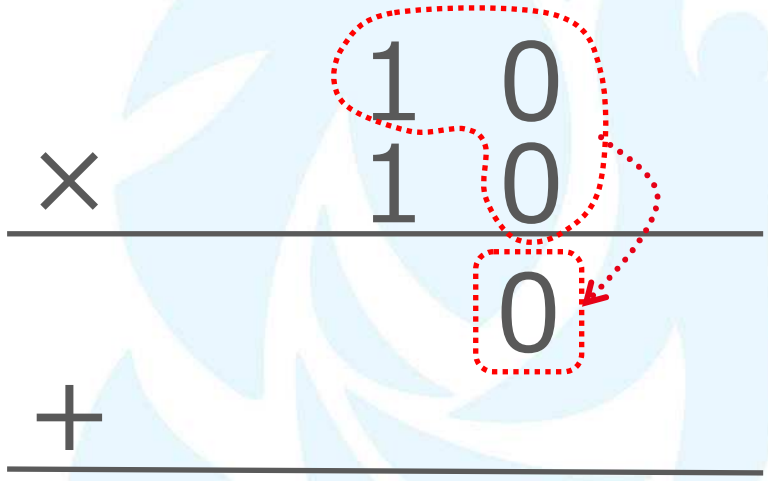
複数桁×１桁だと

下段の１桁が、
1　⇒　上段をそのままコピー
0　⇒　0



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

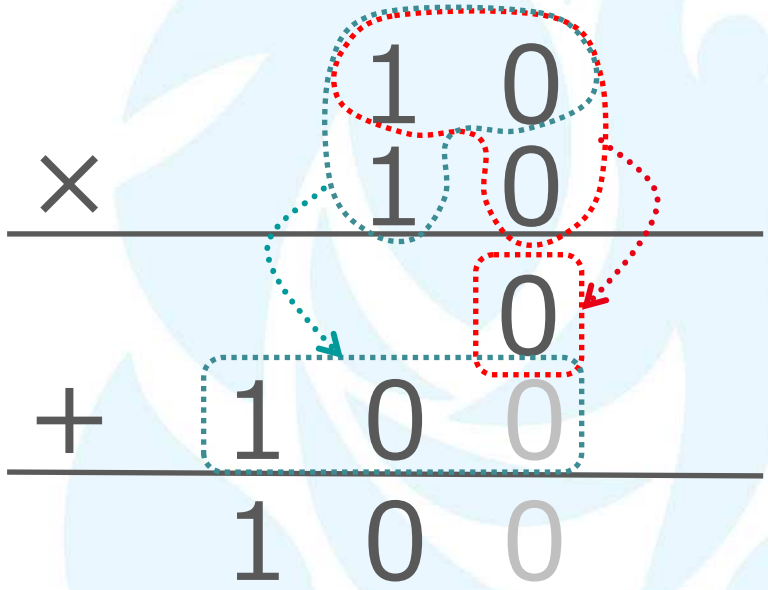


10進と
同じ

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



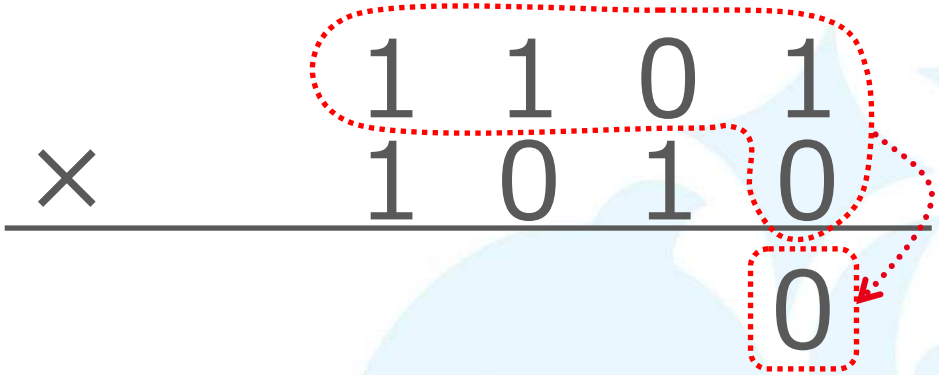
10進と
同じ

東邦大学

もう１つサンプル

$$
\begin{array}{rr}
 & 1\ 1\ 0\ 1 \\
\times & 1\ 0\ 1\ 0 \\
\hline
\end{array}
$$

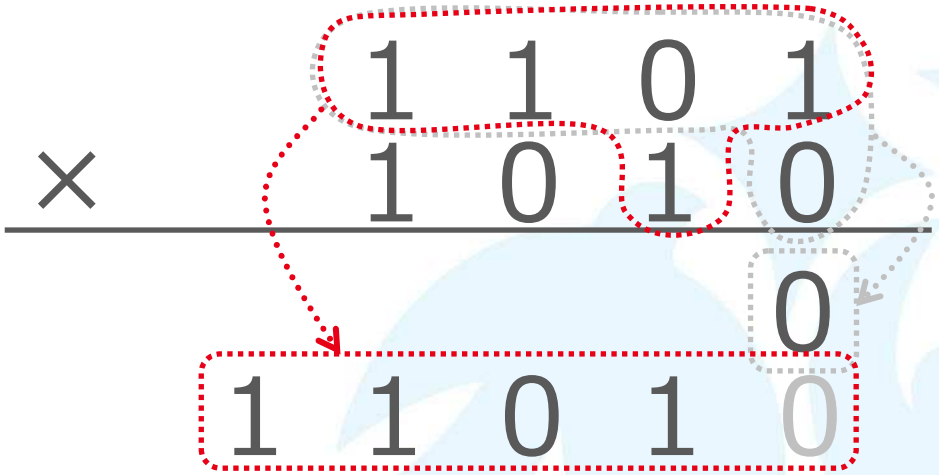
もう１つサンプル



0を掛ける⇒0

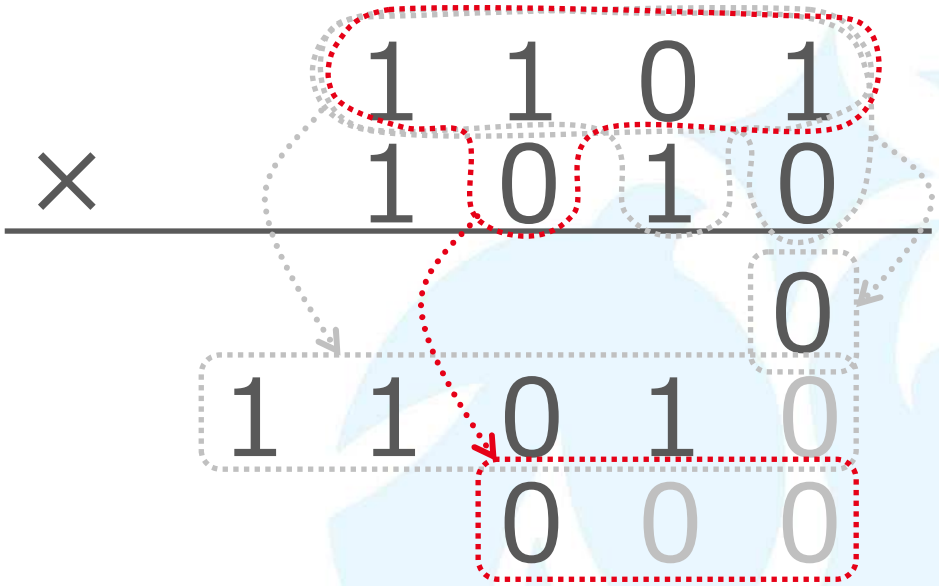
東邦大学

もう１つサンプル



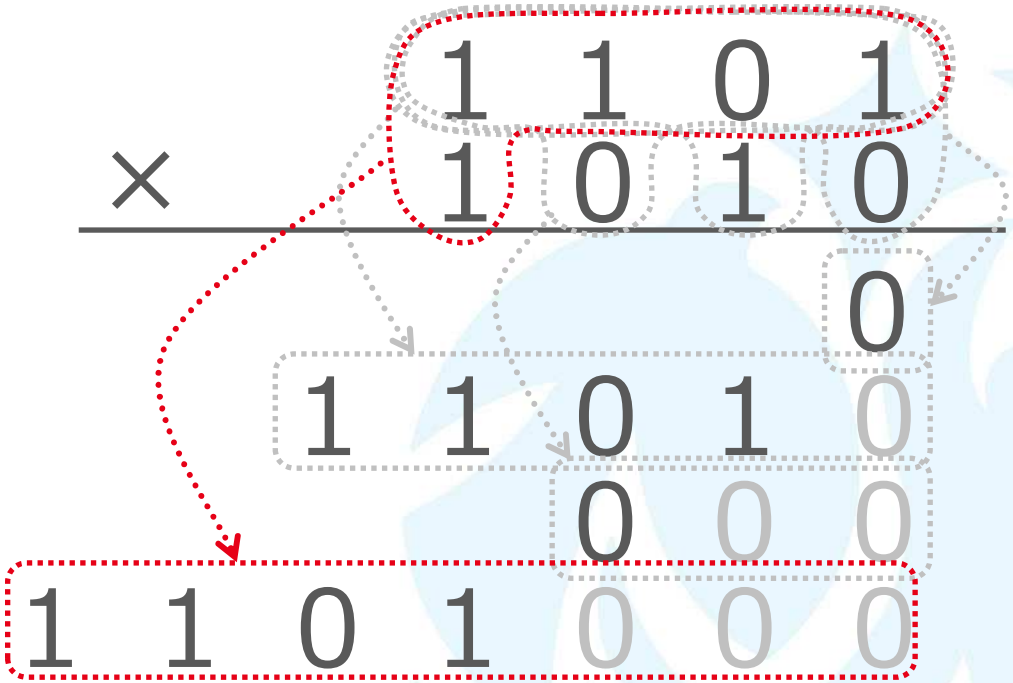
1を掛ける⇒
1101

もう１つサンプル



0を掛ける⇒0

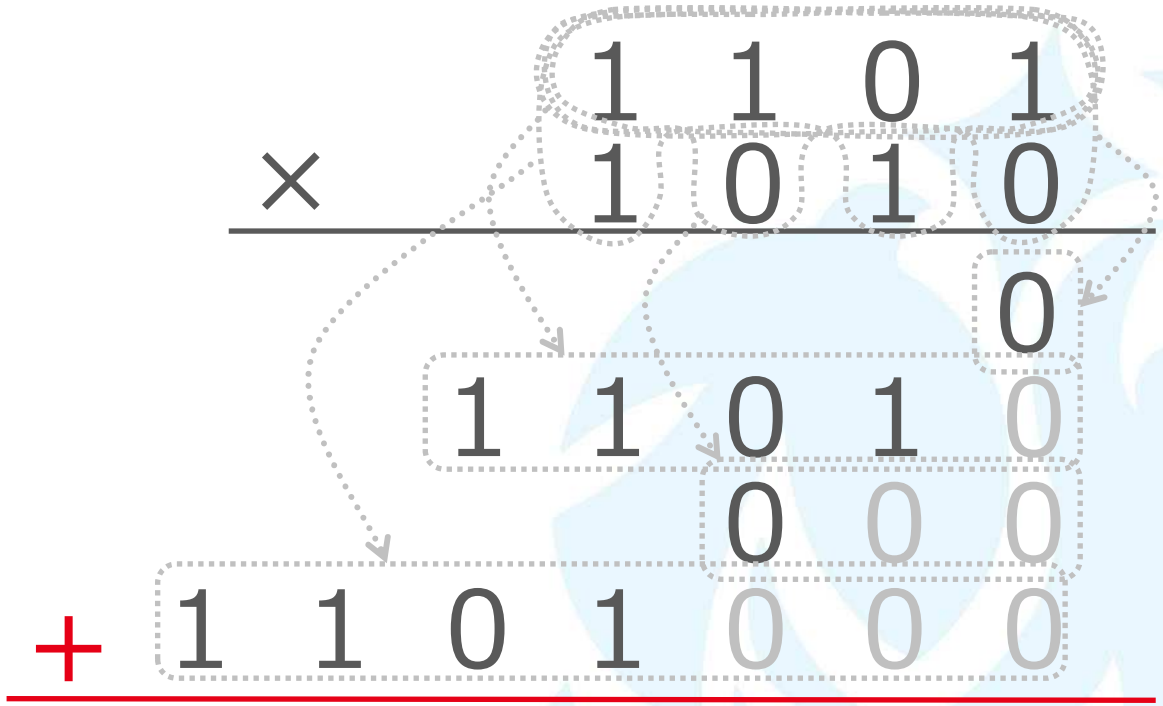
もう１つサンプル



1を掛ける⇒
1101

東邦大学

もう１つサンプル



足す

東邦大学

練習問題

$$\begin{array}{r} 1001 \\ \times\ 1100 \\ \hline \end{array} \qquad \begin{array}{r} 101 \\ \times\ 1111 \\ \hline \end{array}$$

$$\begin{array}{r} 11 \\ \times\ 1100 \\ \hline \end{array} \qquad \begin{array}{r} 1111 \\ \times\ 101 \\ \hline \end{array}$$

東邦大学

練習問題

$$
\begin{array}{r}
1001 \\
\times\ 1100 \\
\hline
1101100
\end{array}
\qquad
\begin{array}{r}
101 \\
\times\ 1111 \\
\hline
1001011
\end{array}
$$

$$
\begin{array}{r}
11 \\
\times\ 1100 \\
\hline
100100
\end{array}
\qquad
\begin{array}{r}
1010 \\
\times\ 101 \\
\hline
110010
\end{array}
$$

東邦大学

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
    - さまざまな工夫で高速化
- 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

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

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に
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に
　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　　１０１
下段の数 Y　ｘ　１０
結果を　Z

東邦大学

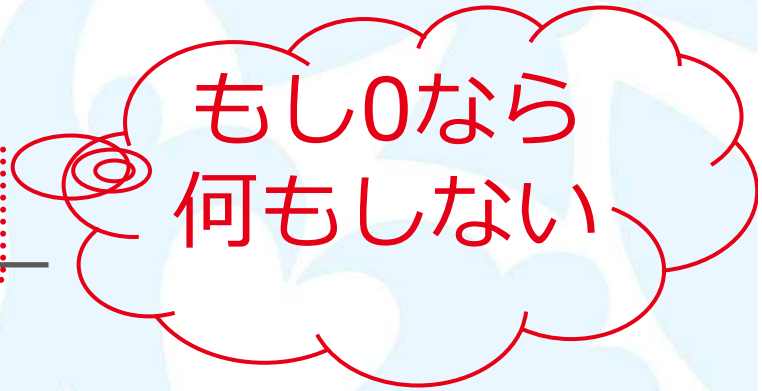
おまけ
２進掛け算プログラムを書いてみよう

上段の数 X　　１０１
下段の数 Y　x　１０
結果を　Z

東邦大学

おまけ
２進掛け算プログラムを書いてみよう

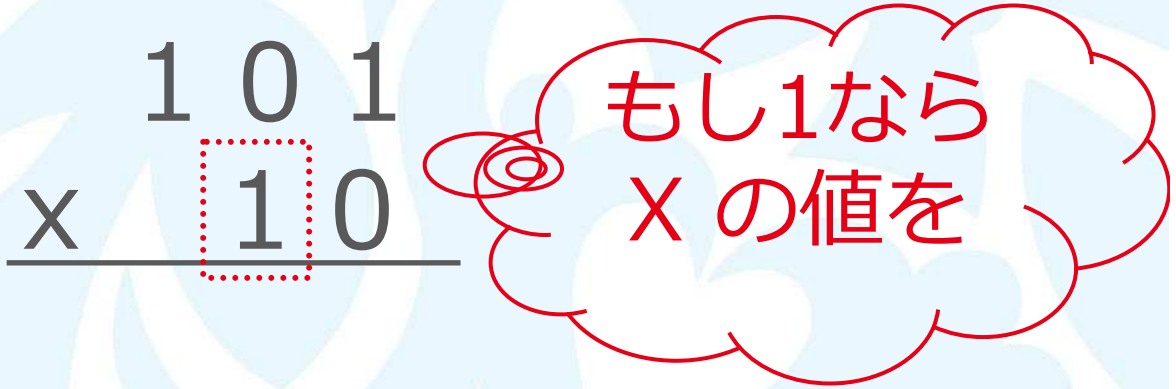
上段の数 X　　１０１
下段の数 Y　x　１０
結果を　Z



東邦大学

おまけ
２進掛け算プログラムを書いてみよう

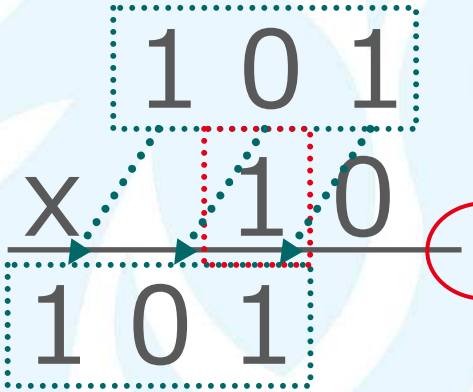
上段の数 X
下段の数 Y
結果を　Z



東邦大学

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

東邦大学

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　　１０１
下段の数 Y　　ｘ　１０
結果を　Z　　１０１０

東邦大学

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　　１０１
下段の数 Y　x　１０
結果を　Z　１０１０

東邦大学

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 ← 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

東邦大学

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

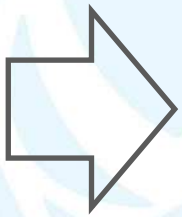
「ビットごとのAND」を使う

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

y \ x	0	1
0	0	0
1	0	1



東邦大学

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

y \ x	0	1
0	0	0
1	0	1

ビットの抽出

y \ x	0	1
0	0	0
1	0	1

常に0

yが0なら

東邦大学

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

y \ x	0	1
0	0	0
1	0	1

ビットの抽出

y \ x	0	1
0	0	0
1	0	1

xがyが1ならそのまま

東邦大学

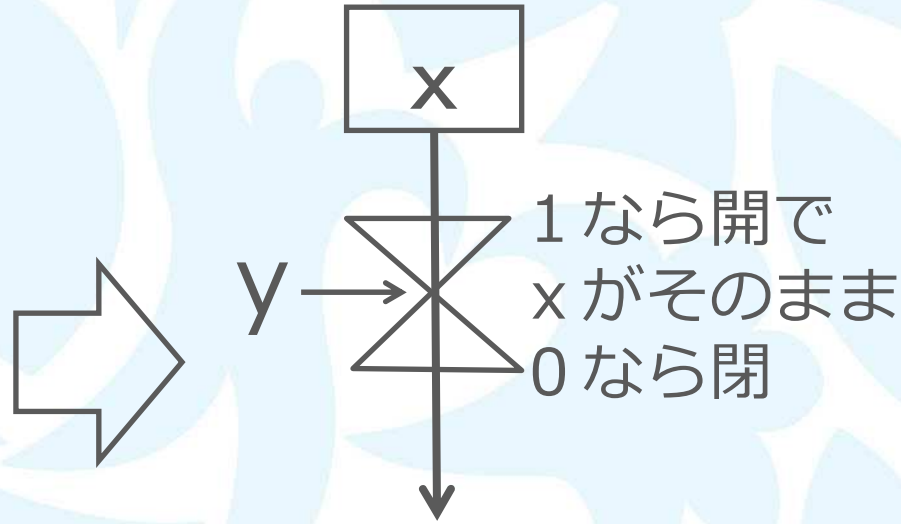
つまり「ビットAND」は0/1のバルブ(弁)の役目

ビットの抽出

y \ x	0	1
0	0	0
1	0	1

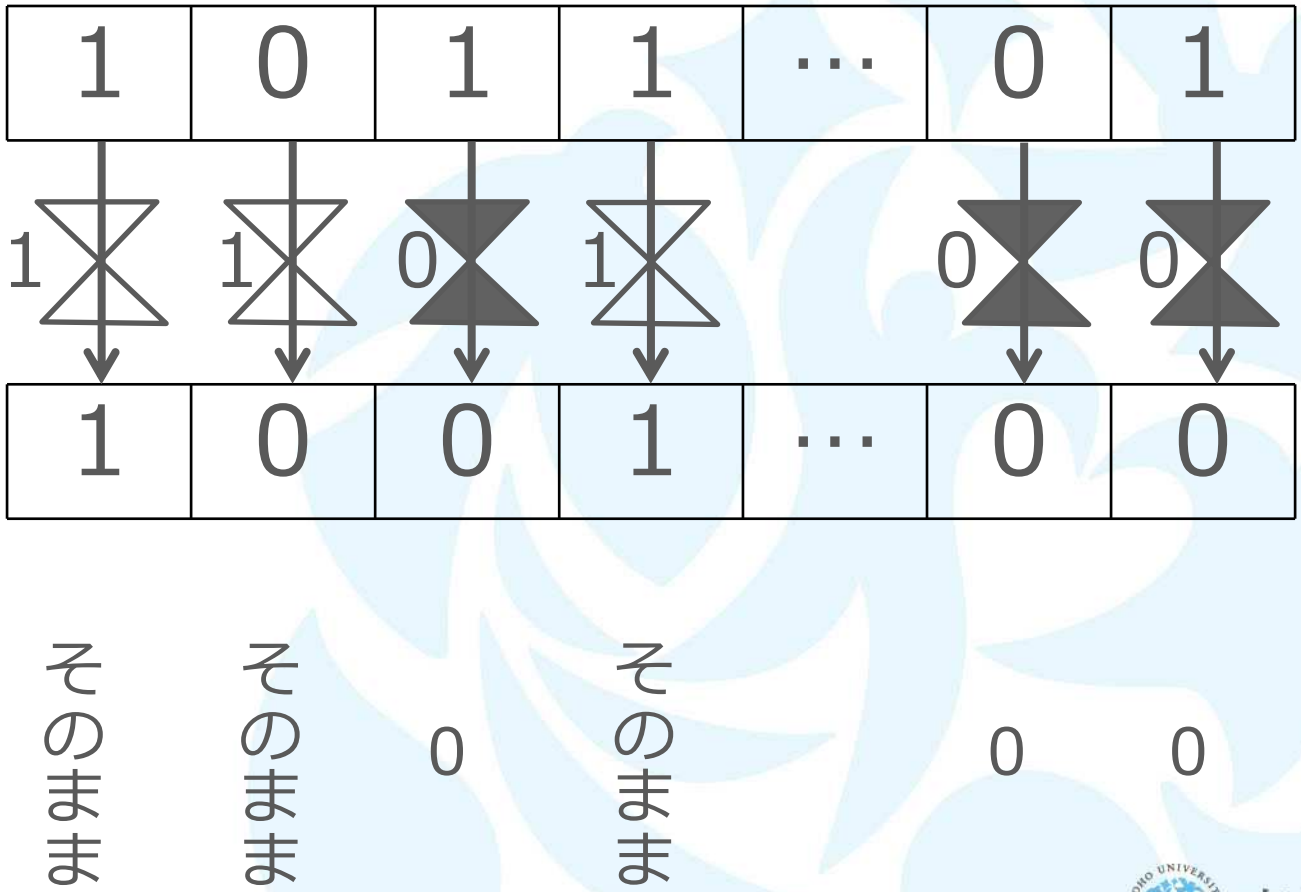
yが0なら常に0

yが1ならxそのまま



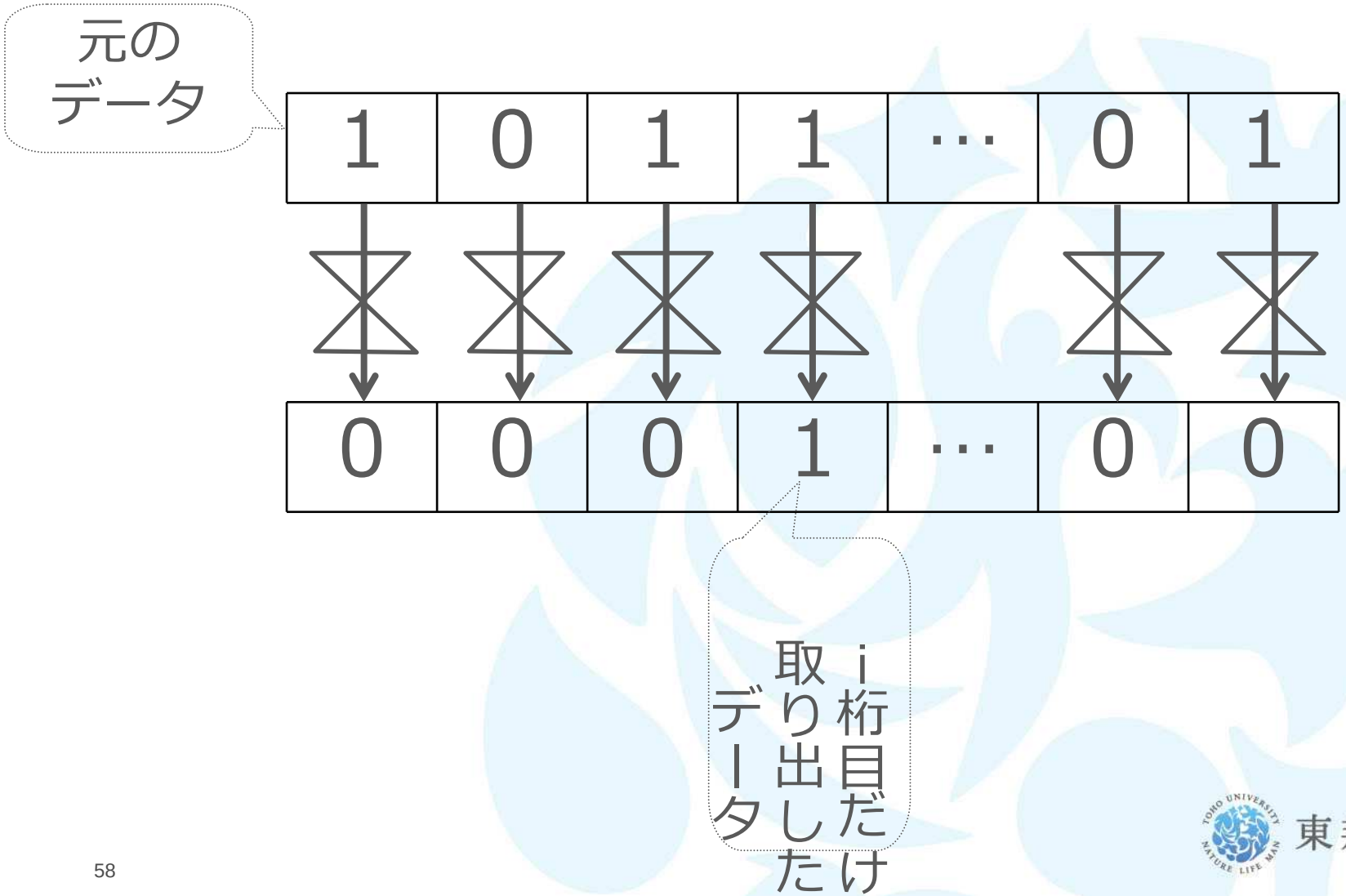
東邦大学

「ビットごとのAND」を並べる



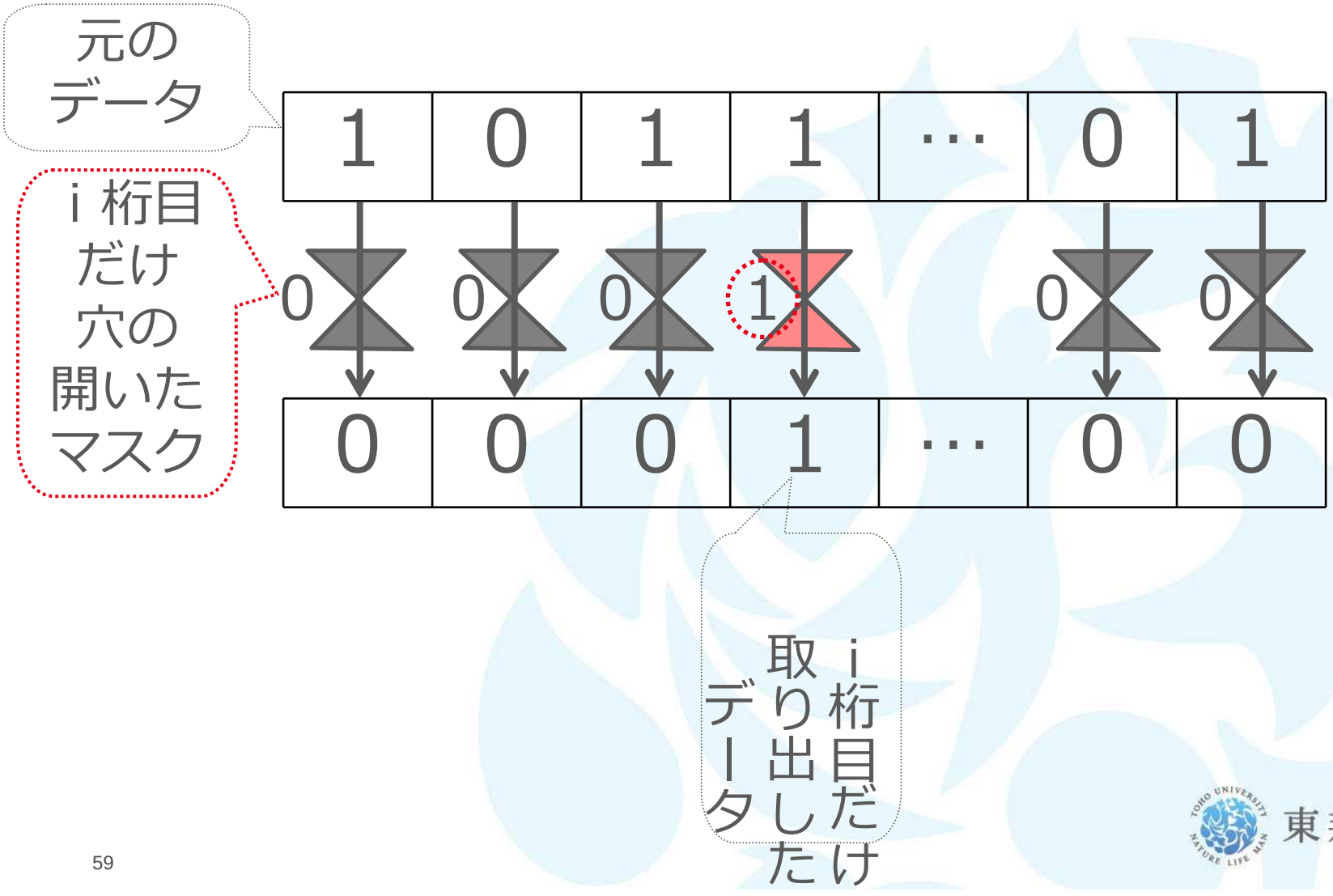
では、i 桁目を取り出すには

では、i 桁目を取り出すには



東邦大学

i 桁目だけ穴のあいたマスクを掛ける



東邦大学

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

東邦大学

i 桁目だけ穴の開いたマスクを作るには

東邦大学

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…0001

y ＝ x << 2

2回

y は 0000…0100

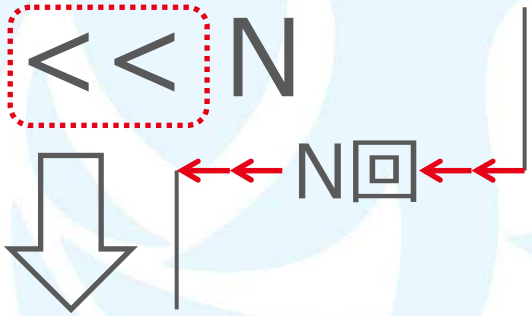
東邦大学

JavaやC言語で
左へN回ずらすには

シフト演算

x ＝ 0000…0001

y ＝ x << N



y は 0001…0000

東邦大学

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;
  }
}
```

のようにすればできる
実行して試してみよう

東邦大学

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？

東邦大学