東邦大学

生命（いのち）の科学で未来をつなぐ

乗算の原理
続き

掛け算のやり方はわかった

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
  - 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
    - 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
    - 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

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

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　　１０１
下段の数 Y　x　１０
結果を　Z

もし0なら
何もしない

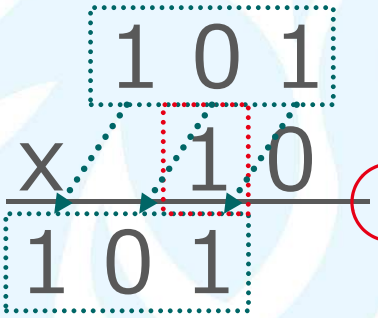
おまけ
２進掛け算プログラムを書いてみよう

上段の数 X　　１０１
下段の数 Y　x　１０
結果を　Z

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

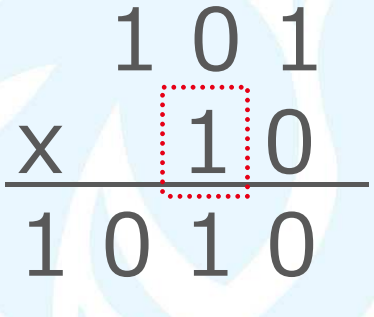
上段の数 X
下段の数 Y
結果を　Z



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
下段の数 Y
結果を　Z



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　　１０１
下段の数 Y　x　１０
結果を　Z　１０１０

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 ← Yの i 桁目を取り出す
　　もし p が 1 なら
　　　　q ← Xを i-1 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-1 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
「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

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

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

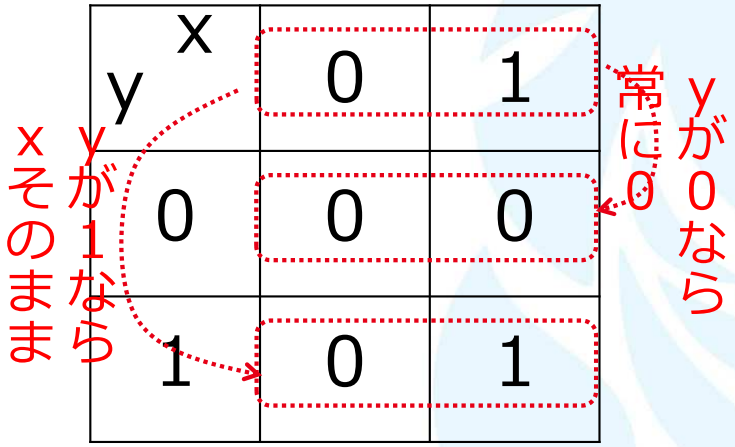
yが1なら

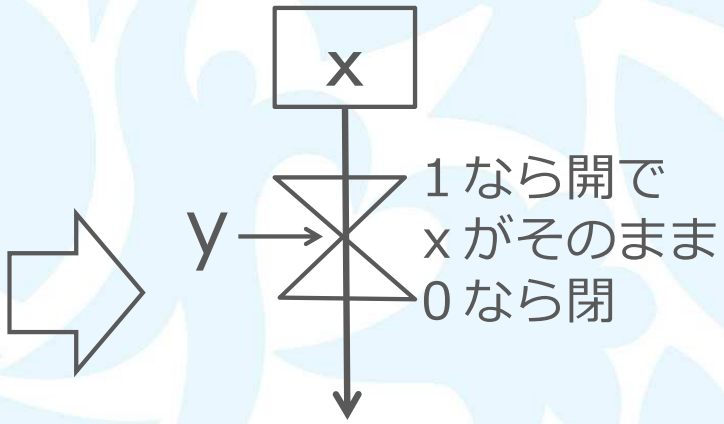
xそのまま

東邦大学

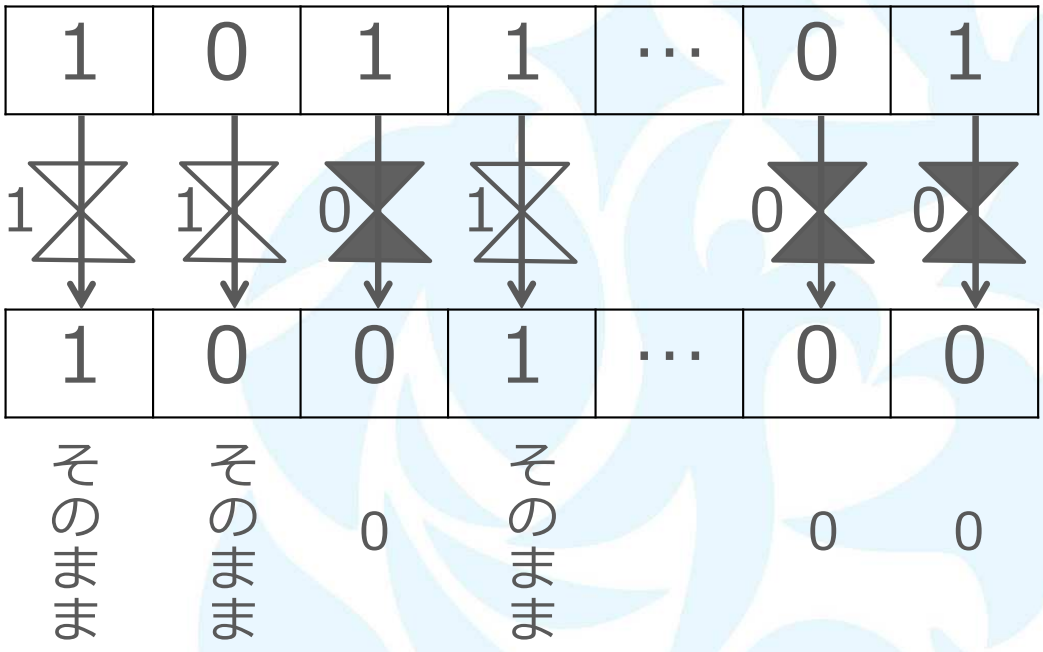
つまり「ビットAND」は0/1のバルブ(弁)の役目

ビットの抽出



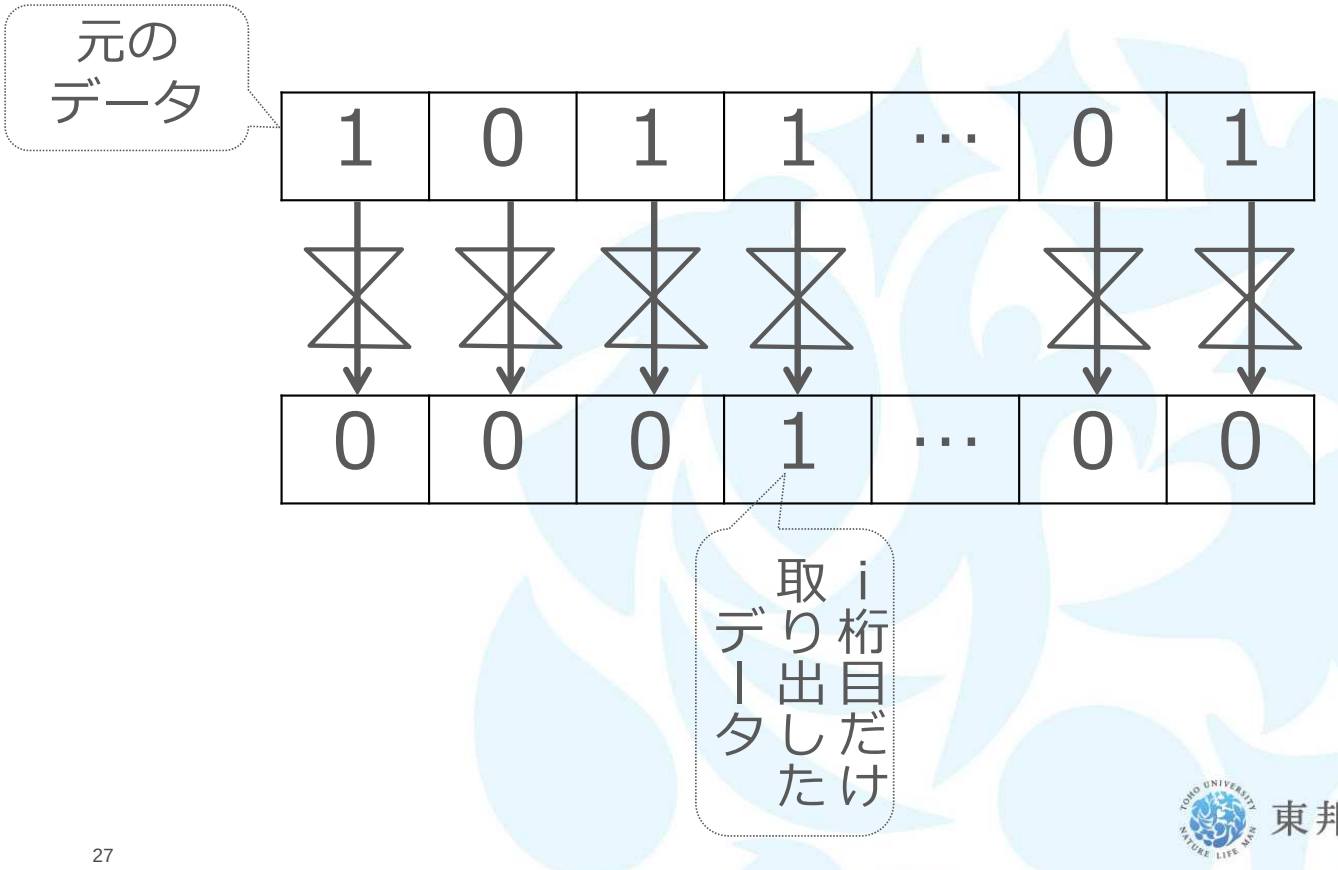


「ビットごとのAND」を並べる

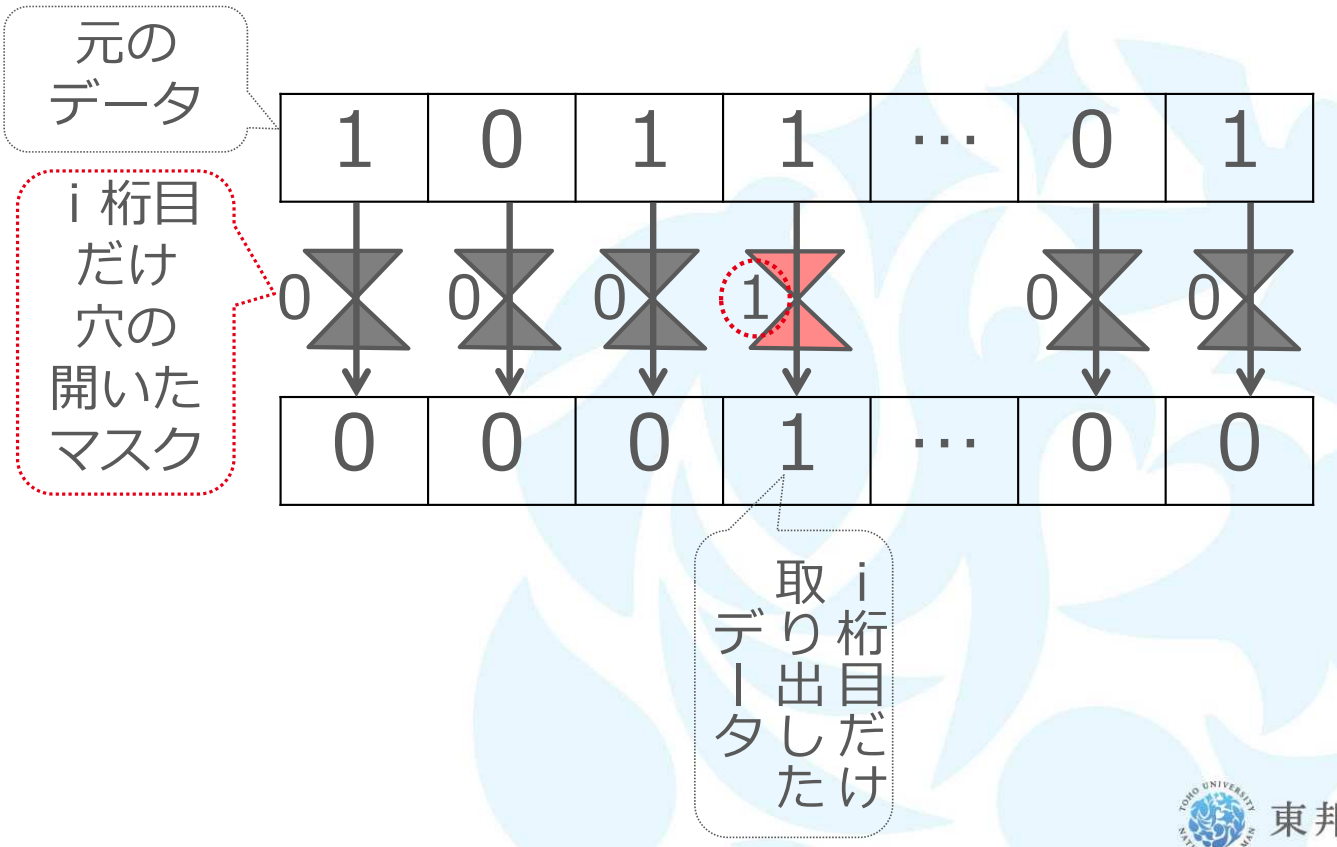


では、i 桁目を取り出すには

では、i 桁目を取り出すには

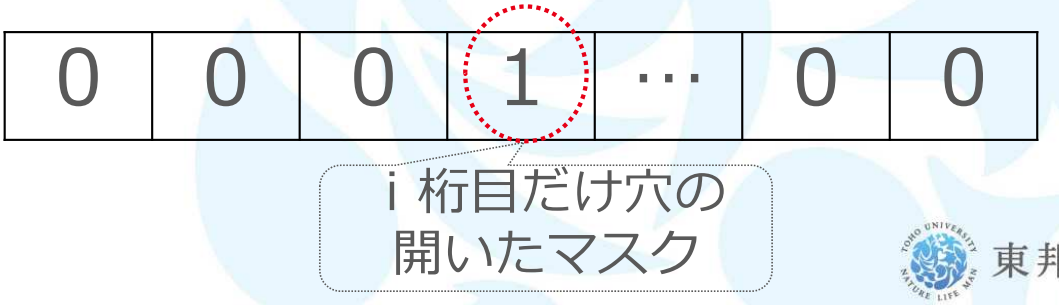


i 桁目だけ穴のあいたマスクを掛ける



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を作るには

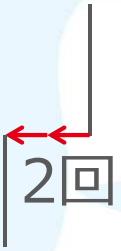
JavaやC言語で
左へN回ずらすには

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



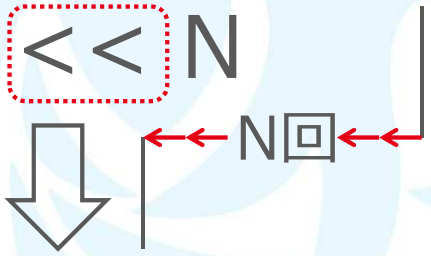
y は 0000…0100

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

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-1 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

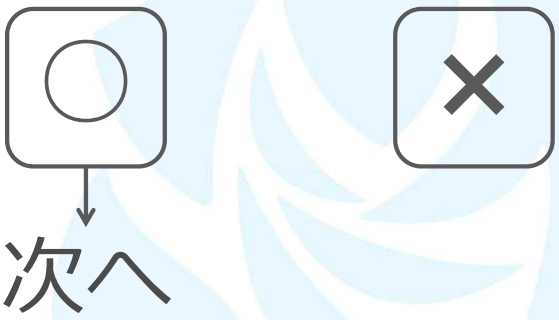
試してみた

```
#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