

プログラミング演習 I
課題2
10進数と2進数

3回目

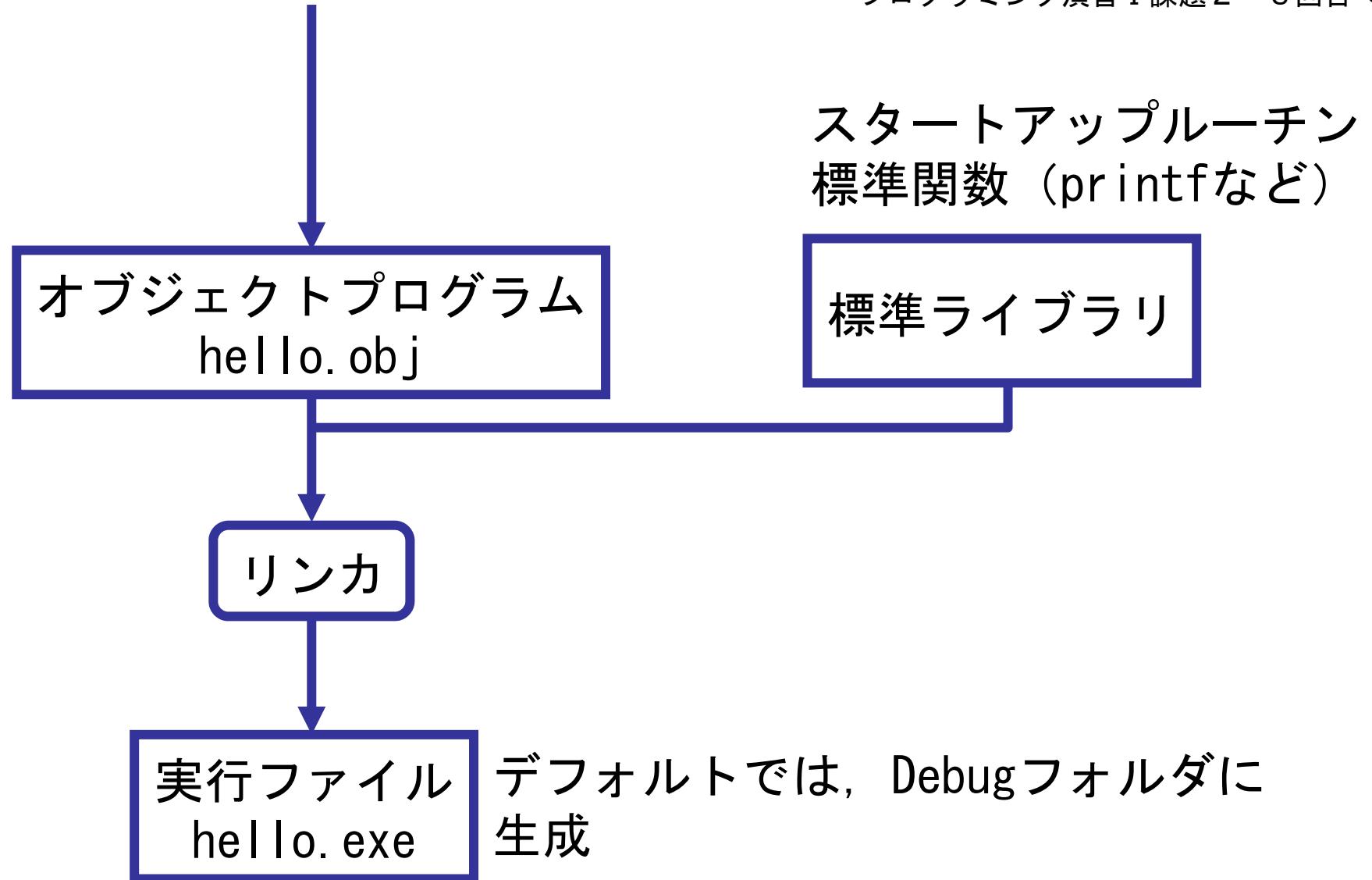
ソースプログラム hello.c

```
#include <stdio.h>
int main(void)
{
    printf("hello, world\n");
    return 0;
}
```

コンパイル・リンクの過程

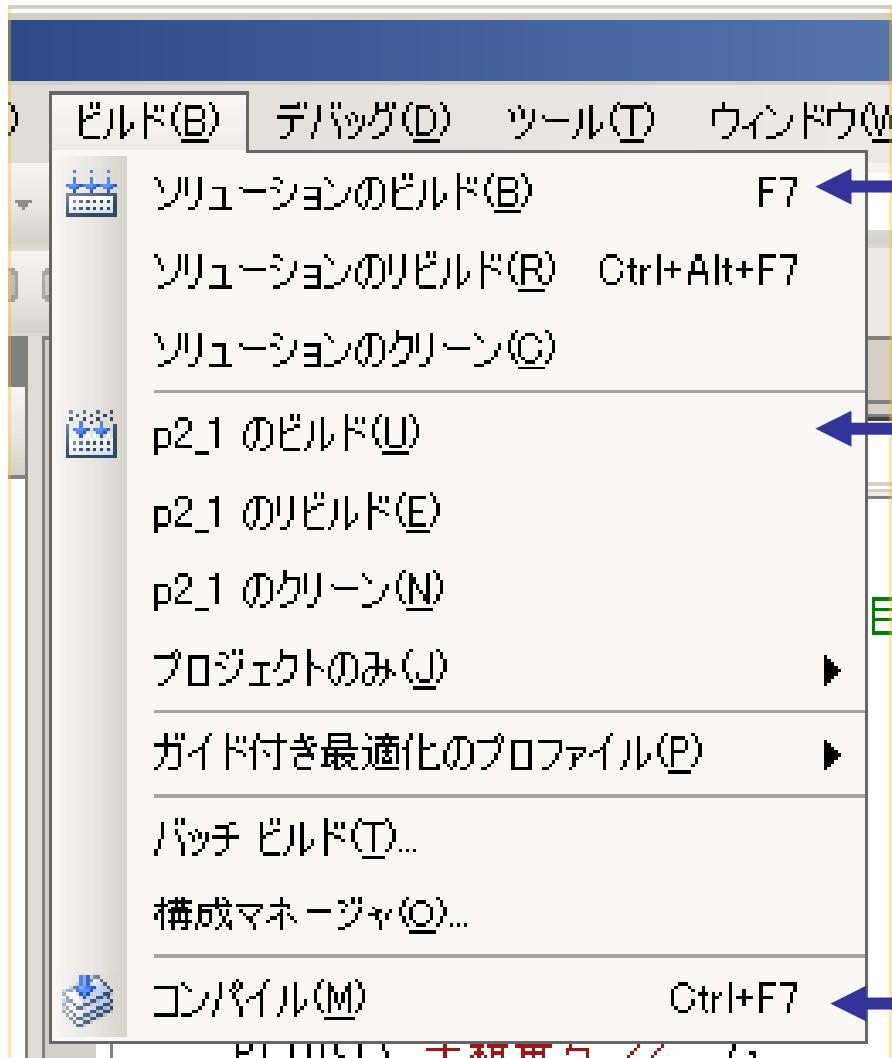
ヘッダーファイル
stdio.h





スタートアップルーチン : main関数を呼び出す

[ビルド]メニュー



ここでは、1ソリューションに1プロジェクトなので、両者は同じ働き

[ビルド]は、コンパイルの後でリンクを行うこと。
コンパイル済みのときはリンクのみ行い、さらに、リンク済みのときは何もしない。

[コンパイル]は、プリプロセッサとコンパイラを実行し、objファイルを作る。

ヘッダーファイル

- 関数は使う前に定義／宣言しなければならない
 - コンパイラは、ソースコードを先頭から順にコンパイル
 - 関数呼び出し部分をコンパイルするとき、その関数のパラメータの型や戻り値の型が必要
- printfも関数
- 関数printfの宣言は、stdio.h
 - C:\Program Files\Microsoft Visual Studio 10.0\VC\Include\stdio.h
- 標準関数は、stdio.hやstdlib.h, string.h, math.h等で宣言。
- 適切な #include <~.h> が必要

(例) 関数randを使うとき

- 関数randは、擬似乱数を生成する。
- ソースコードにrand();と書いて、文字列randを選択し、F1キーを押してみよう。
- 必要なヘッダーに<stdlib.h>とある。
- #include <stdlib.h>が必要であることがわかる

◀ 解説

rand 関数は、0 ~ RAND_MAX (32767) の範囲の整数の擬似乱数を返します。 srand 関数を使用して、rand を呼び出す前に、擬似乱数ジェネレーターのシード値を指定します。

◀ 必要条件

ルーチン	必須ヘッダー
rand	<stdlib.h>

プログラミング演習 I

課題2 10進数と2進数

- 目的
 - 関数の定義の仕方や使い方，標準関数を利用する仕組みを学ぶ
- 題材
 - (1回目)整数 \leftrightarrow 10進数の変換
 - (1回目)学籍番号のチェックディジット生成法
 - (2回目)整数 \leftrightarrow 2進数の変換
 - (3回目)2進数での加算
- 課題は5つ

課題概要

1回目

- 課題(2-1)
 - 学籍番号(整数)の10進6桁の各桁の数を取り出す
- 課題(2-2)
 - 学籍番号(整数)の10進6桁の各桁の数からチェックディジット生成

2回目

- 課題(2-3)
 - 0~255の整数の2進8桁の各桁の数を取り出す
- 課題(2-4)
 - 8桁の2進数から、それが表す整数を求める

- 課題(2-5)
 - 2つの8桁の2進数の和を計算する.

3回目

(参考) 課題(2-4)

- 0 / 1による8桁の2進数入力に対して、それが表す10進数の整数值を出力せよ。このとき、2進数から10進数を求める関数を作成・利用すること。
 1. キーボードから8桁の2進数を入力し、8個のint型変数に格納
 2. 8個の変数で表されている8桁の2進数を整数に変換する関数を呼び出し、返ってきた値をint型変数に格納する。
 3. 10進数の整数を出力する

課題(2-5)

- 入力は, 0/1による8桁の2つの2進数とする
- 出力は, それらの和の2進表現とする
- 論理和, 論理積, 排他的論理和を求める関数を作成すること. これらを全加算器で利用する.
- 全加算器の機能を関数で実現し, 加算はこれを用いて行うこと.
- なお, 確認を容易にするため, 入力と出力の各2進数に対する10進表現も出力すること.

2進数の和

$$\begin{array}{r} \boxed{0} \boxed{1} \boxed{0} \boxed{1} \boxed{1} \\ +) \boxed{0} \boxed{0} \boxed{1} \boxed{0} \boxed{1} \\ \hline \boxed{} \boxed{} \boxed{} \boxed{} \boxed{} \end{array}$$

2進数の和

$$\begin{array}{r} & & & 1 \\ & 0 & 1 & 0 & 1 & 1 \\ +) & 0 & 0 & 1 & 0 & 1 \\ \hline & \square & \square & \square & \square & 0 \\ & 1 & 0 & & & \end{array}$$

A diagram illustrating the addition of two binary numbers. The top row shows the sum (1), the second row shows the first addend (01011), and the third row shows the second addend (00101). A red arrow points from the third column of the addends to the fourth column of the sum, indicating a carry-over. The result is shown below the sum, with the carry-over '1' in the tens column.

2進数の和

$$\begin{array}{r} & \boxed{1} & \boxed{1} \\ & | & | \\ \boxed{0} & \boxed{1} & \boxed{0} & \boxed{1} & \boxed{1} \\ +) & \boxed{0} & \boxed{0} & \boxed{1} & \boxed{0} & \boxed{1} \\ \hline & \boxed{} & \boxed{} & \boxed{} & \boxed{0} & \boxed{0} \\ & & & & | & \\ & & & & 1 & 0 \end{array}$$

A diagram illustrating the addition of two binary numbers. The top row contains the digits 1 and 1. The bottom row contains the digits 0, 0, 1, 0, 1. A red arrow points from the second column of the top row to the fourth column of the bottom row, indicating a carry operation. The result is shown below, with the sum of the first four columns being 0, and a carry of 1 from the fifth column resulting in a final sum of 10.

2進数の和

$$\begin{array}{r} & \boxed{1} & \boxed{1} \\ +) & \boxed{0} & \boxed{1} & \boxed{0} & \boxed{1} & \boxed{1} \\ \hline & \boxed{0} & \boxed{0} & \boxed{1} & \boxed{0} & \boxed{1} \\ & \boxed{} & \boxed{} & \boxed{0} & \boxed{0} & \boxed{0} \\ & \boxed{1} & \boxed{0} \end{array}$$

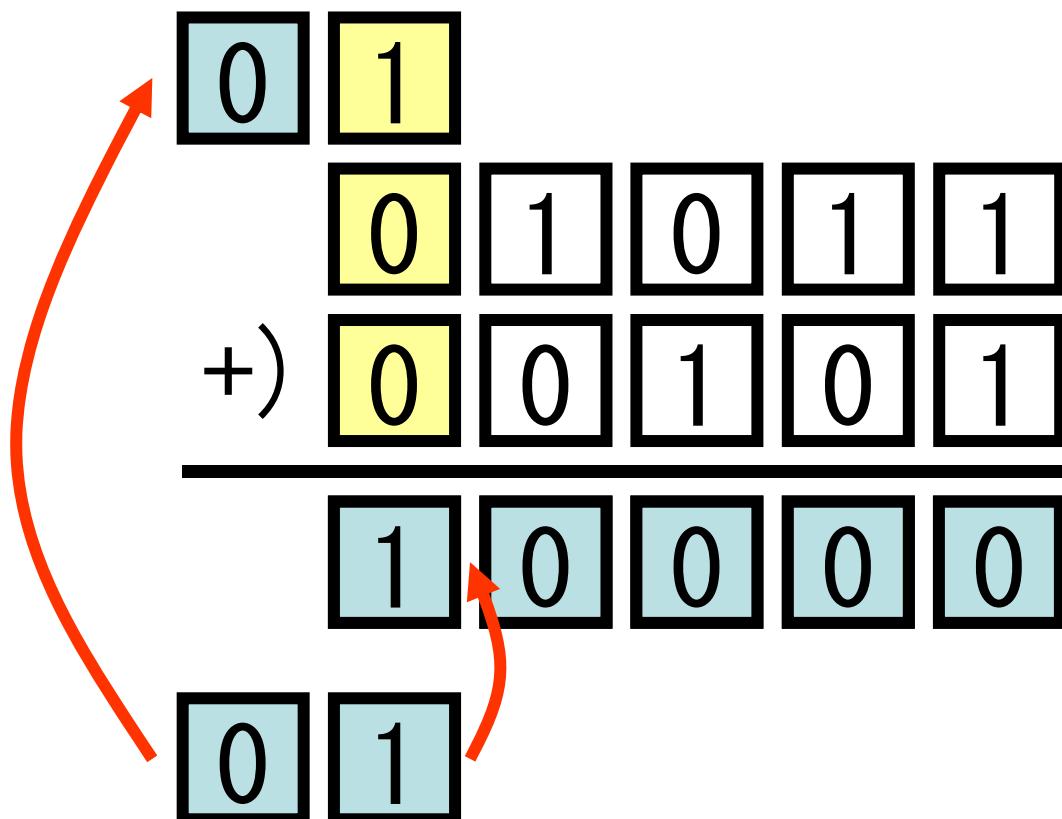
A diagram illustrating the addition of two binary numbers. The top row shows the sum (11), the middle row shows the addend (01011) and augend (00101). The bottom row shows the result (10) and the carry-out (0). Red arrows highlight the carry-over from the third column to the fourth column and the final carry-out.

2進数の和

$$\begin{array}{r} & \boxed{1} & \boxed{1} \\ & 0 & \boxed{1} & 0 & 1 & 1 \\ +) & \boxed{0} & \boxed{0} & 1 & 0 & 1 \\ \hline & \boxed{} & 0 & \boxed{0} & \boxed{0} & \boxed{0} \\ & \boxed{1} & \boxed{0} \end{array}$$

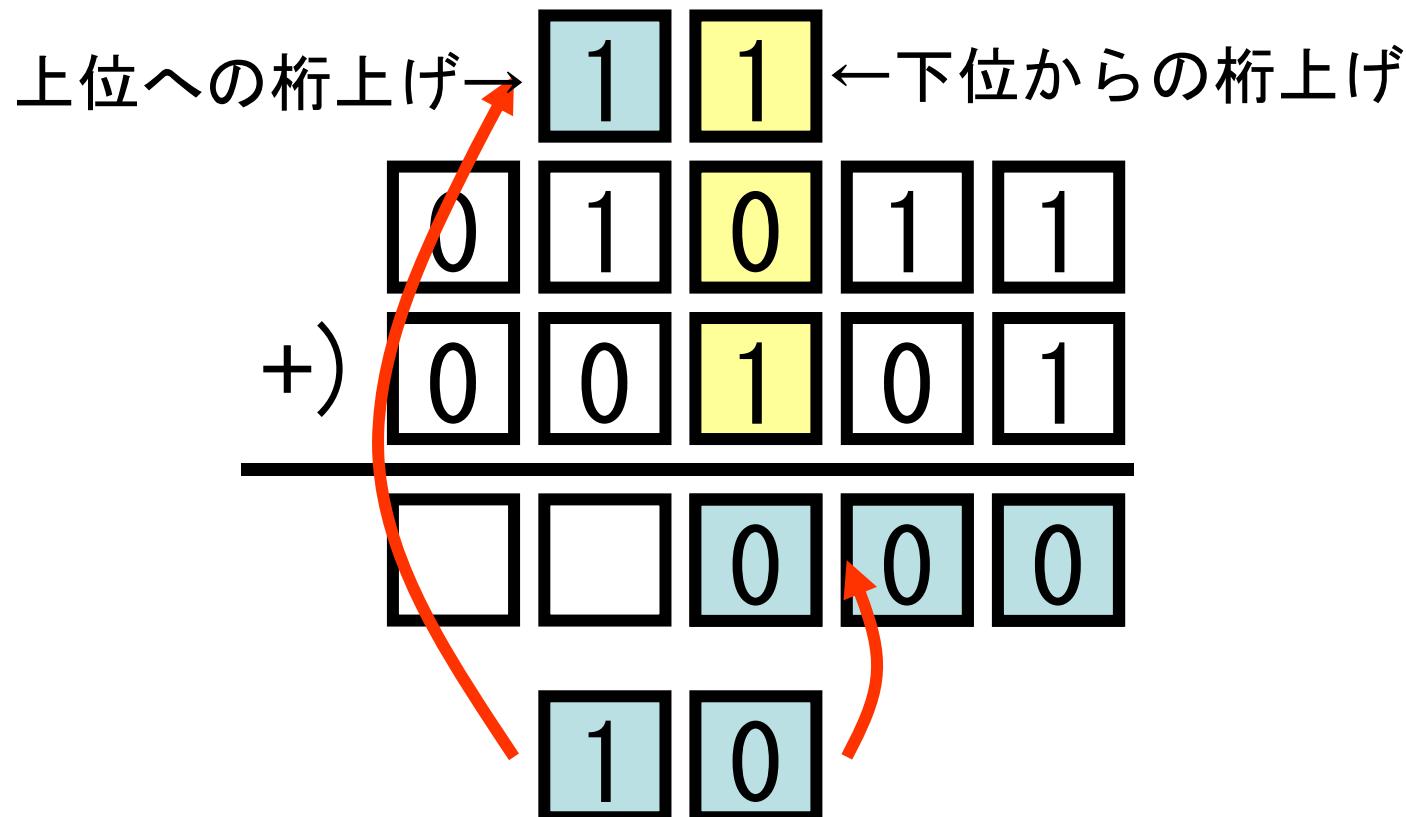
A diagram illustrating the addition of two binary numbers. The top row contains the binary number 11. The middle row contains 01011. A plus sign (+) is followed by the bottom row, 00101. A horizontal line separates the input from the result. The result is 00000. Below the result is a two-digit binary number 10. Red arrows highlight the carry operation: one arrow points from the sum of the first column (1+0) to the carry bit of the second column; another arrow points from the carry bit of the fourth column (from the third column's carry) to the final carry bit at the bottom.

2進数の和



2進数の和

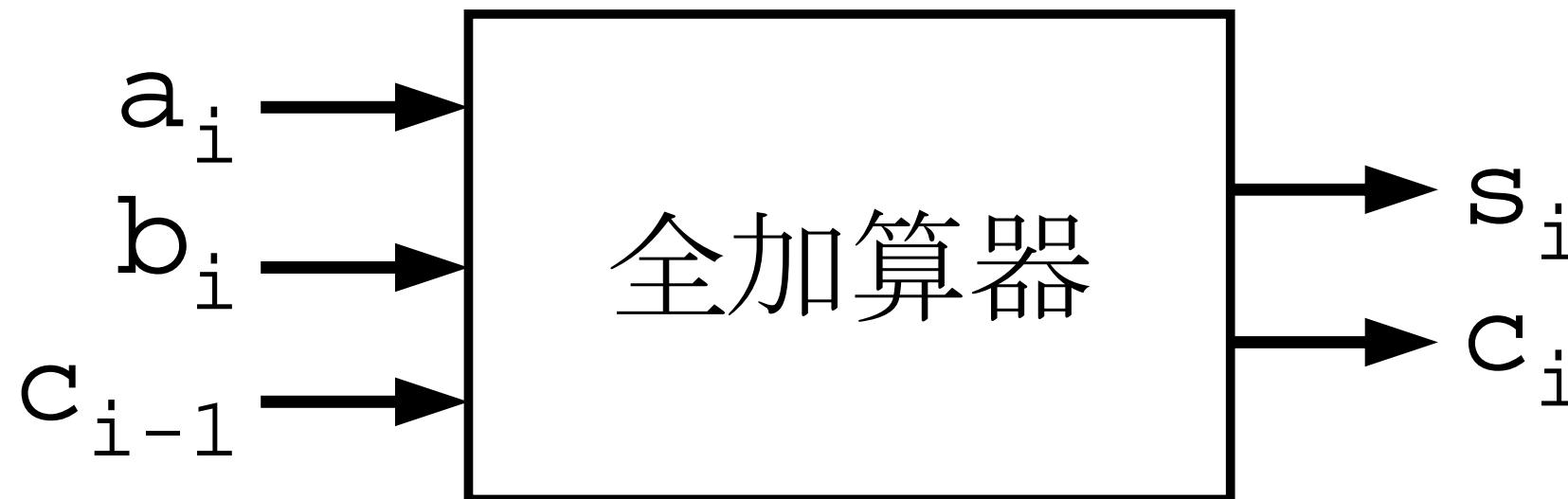
桁上げ : carry



各桁では、入力 3 ビット、出力 2 ビット ⇒ 全加算器

a_i と b_i : 演算対象の i 衔目

c_{i-1} : 下位からの桁上げ

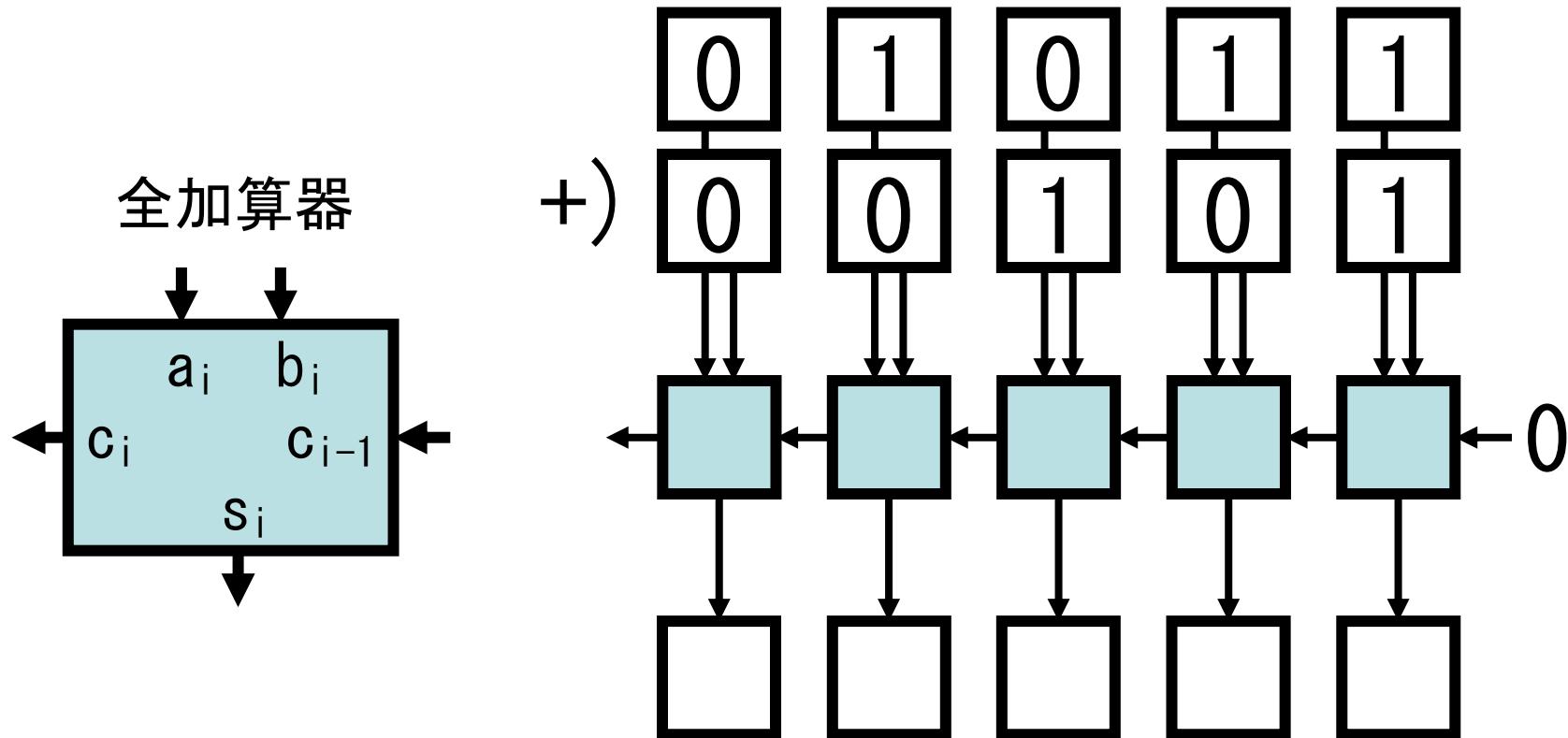


s_i : 和の i 衔目

c_i : 上位への桁上げ

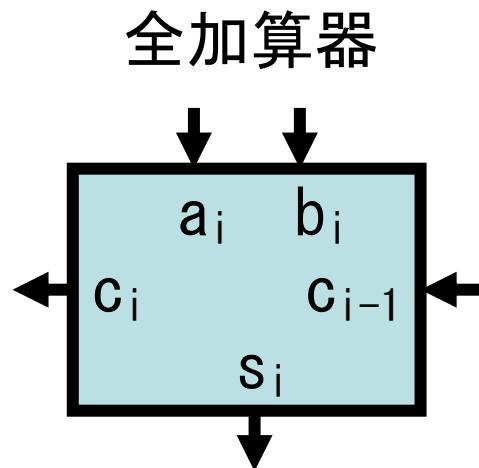
要するに · · ·

$(c_i s_i)_2$ が 3 ビットの和となる



- ・下位の桁から計算
- ・最下位での c_{i-1} は0

全加算器の動作



和 s_i と上位への桁上げ c_i の論理式

$$\begin{aligned}s_i &= a_i \oplus b_i \oplus c_{i-1} \\ c_i &= a_i \cdot b_i + b_i \cdot c_{i-1} + c_{i-1} \cdot a_i \\ &= a_i \cdot b_i + (a_i \oplus b_i) \cdot c_{i-1}\end{aligned}$$

演算子の意味

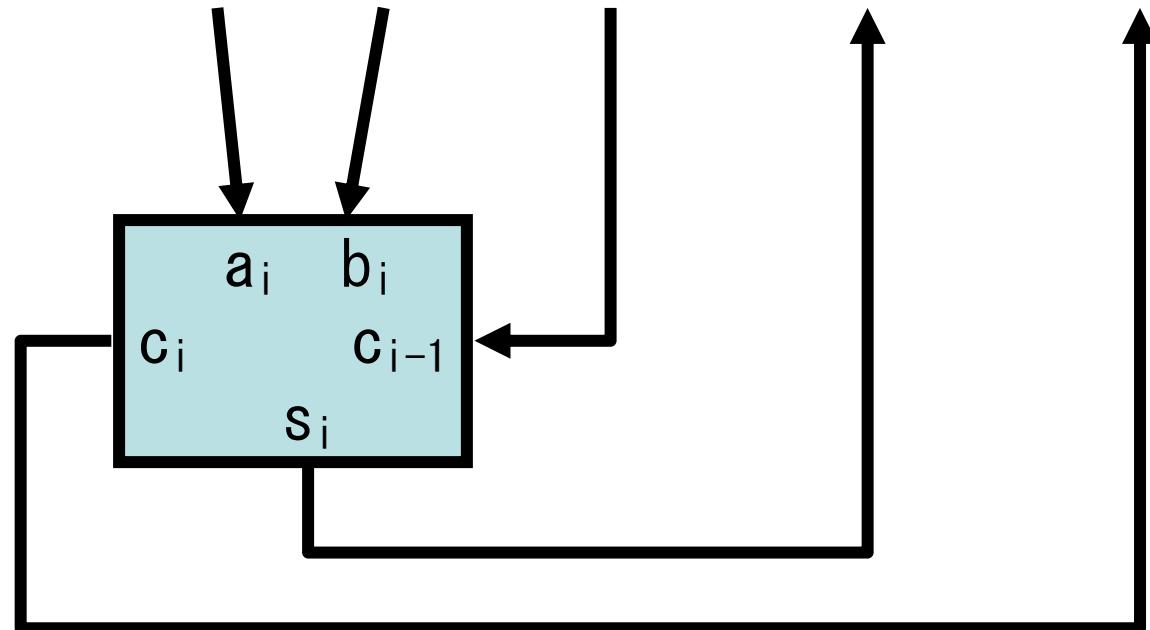
\oplus : 排他的論理和

$+$: 論理和

\cdot : 論理積

全加算器を関数として実装

```
void full_adder(int a, int b, int cin, int *s, int *cout)
```



$*s = /*$ 和 を求める論理式 */;
 $*cout = /*$ 桁上げを求める論理式 */;

full_adder関数の使用例

```
void full_adder(int a, int b, int cin, int *s, int *cout)
```

値渡し(full_adder内で値を
変更しても、呼び出し側の
値に影響しない)

ポインタ渡し
(full_adder内で
呼び出し側の値を
変更可能)

```
/* full_adderの呼び出し側 (例えばmain.c) */
```

```
int a0, b0, cin0, s0, cout0;  
a0 = b0 = cin0 = 1;
```

```
full_adder(a0, b0, cin0, &s0, &cout0);
```

```
/* full_adderの実行後、a0, b0, cin0は1のままだが、*/  
/* s0とcout0はfull_adderでの計算結果が反映される */
```