

**プログラミング演習Ⅰ
課題2
10進数と2進数**

3回目

C:\Program Files\Microsoft Visual Studio 8\VC\Include\stdio.h.....
関数printfのプロトタイプ宣言

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

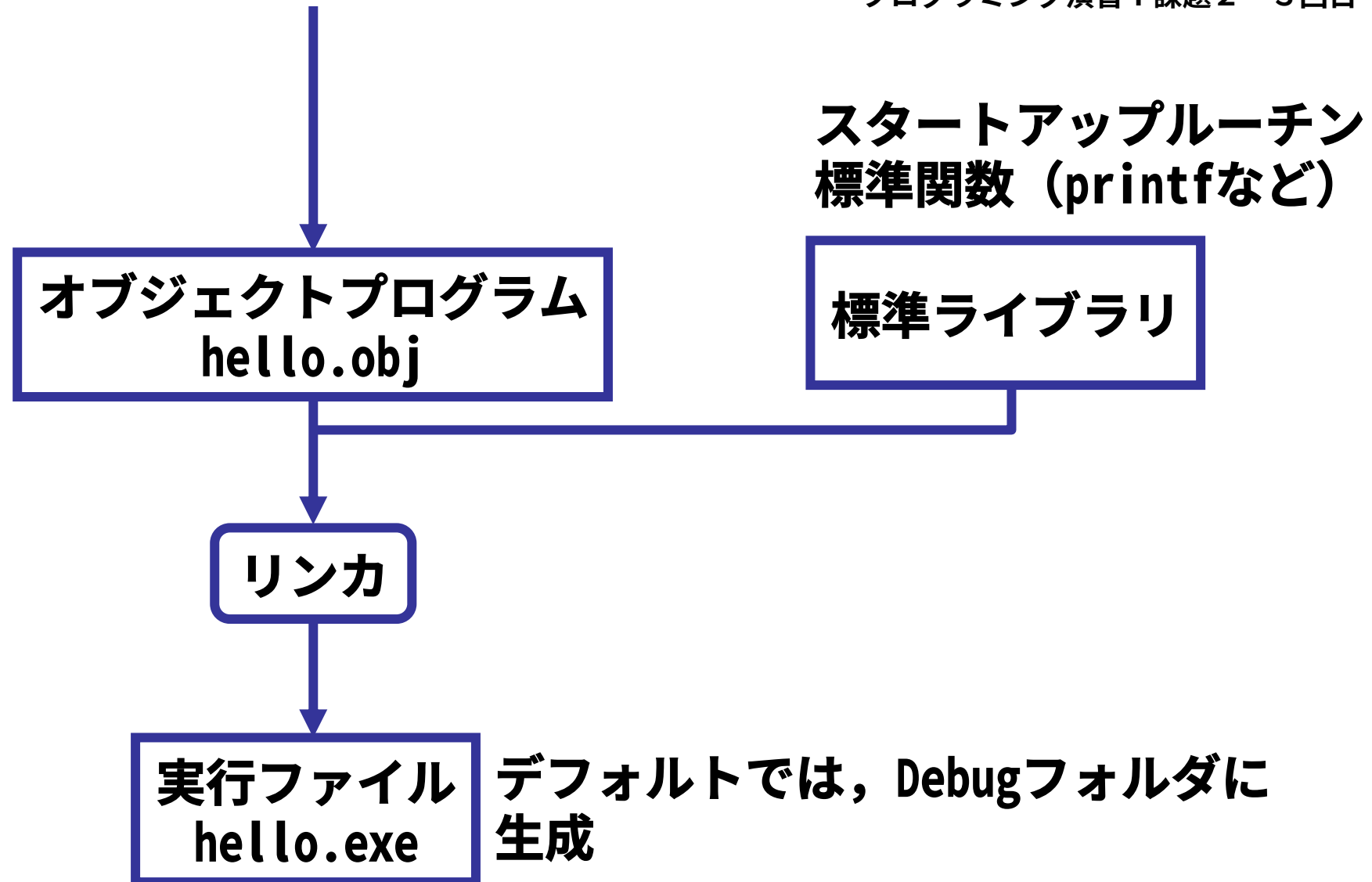
ヘッダーファイル
stdio.h

プリプロセッサ

#include . . .
#define . . .
#ifdef . . . } の処理
コメントの除去

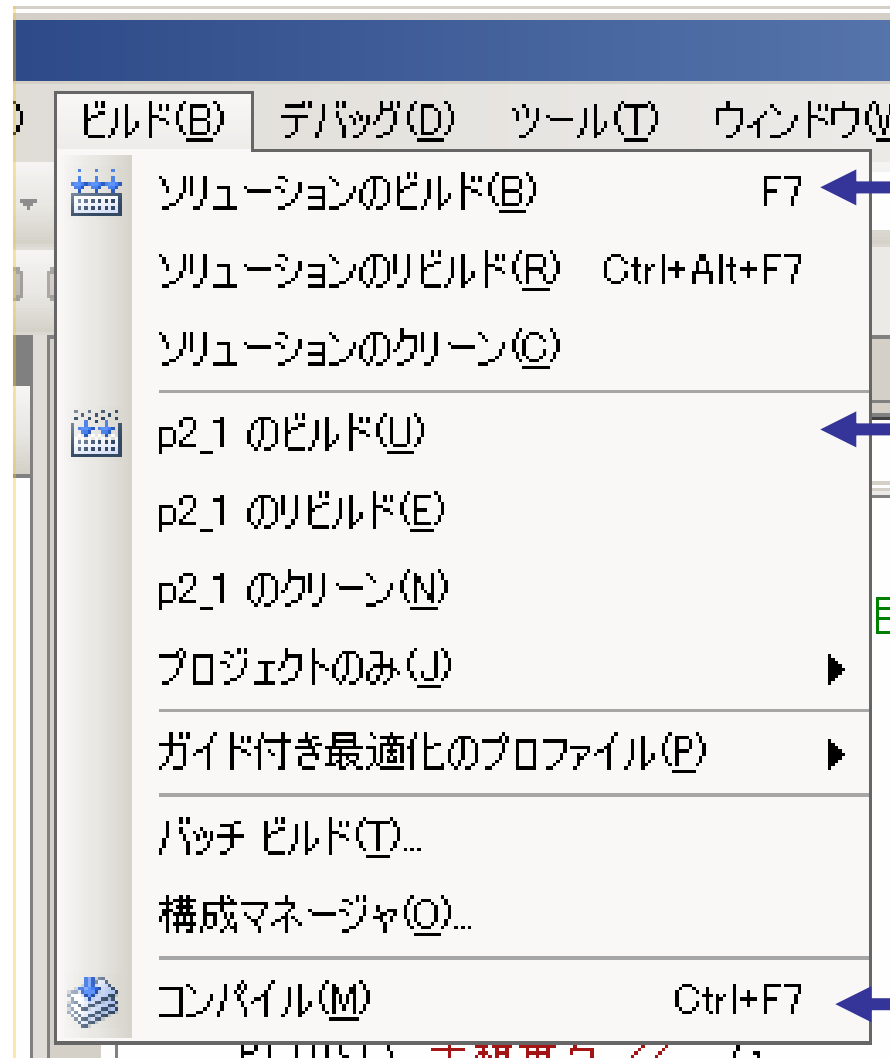
コンパイラ

機械コードへの翻訳



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 8\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のどこかにカーソルを置き、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

rand

擬似乱数を生成します。

```
int rand(void);
```

関数	必要なヘッダー	互換性
rand	<stdlib.h>	ANSI, Win 95, Win NT

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

課題(2-4)

- **0 / 1による8桁の2進数入力に対して、それが表す整数値を出力せよ (10進数で) . このとき8桁の2進数からそれが表す整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. 整数を出力する (10進数で)**

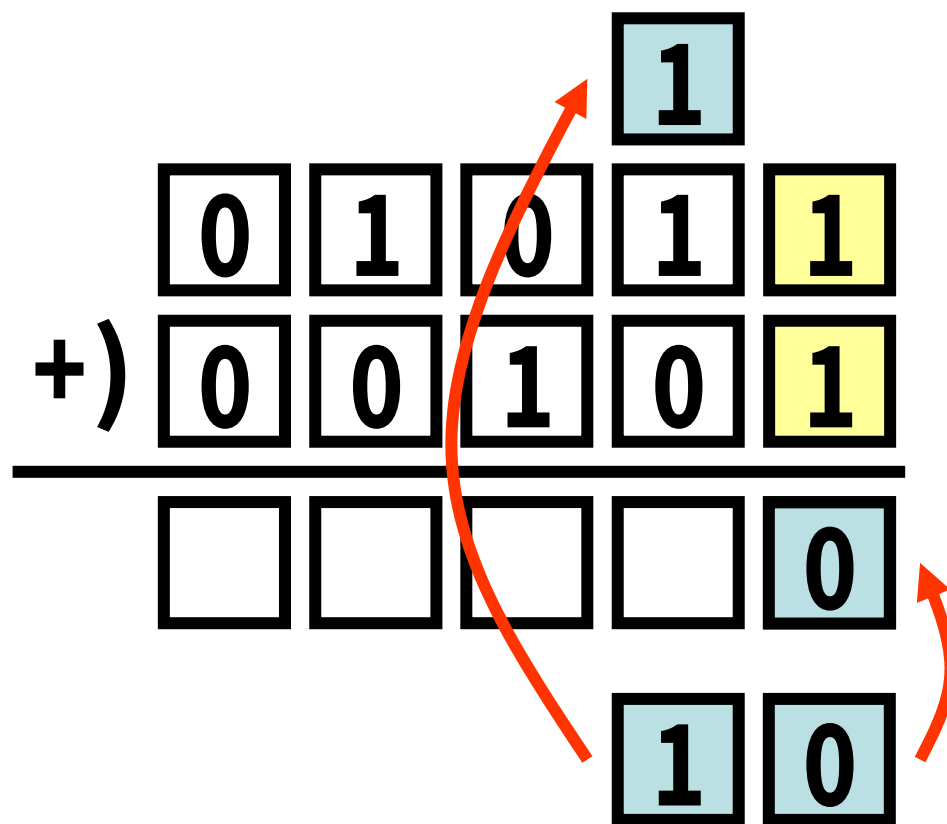
課題(2-5)

- 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

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

2進数の和



2進数の和

The diagram illustrates the process of binary addition. It shows two numbers being added, with the result displayed below a horizontal line. The numbers are represented by boxes containing 0s and 1s. The first number is 01011, and the second number is 00101. The result is shown in boxes below the line, with the first two boxes empty and the last two boxes containing 0 and 0. Red arrows indicate the carry propagation from the rightmost column to the leftmost column.

			1	1
	0	1	0	1
	0	0	1	0
+	0	1	0	1
			0	0
			1	0

2進数の和

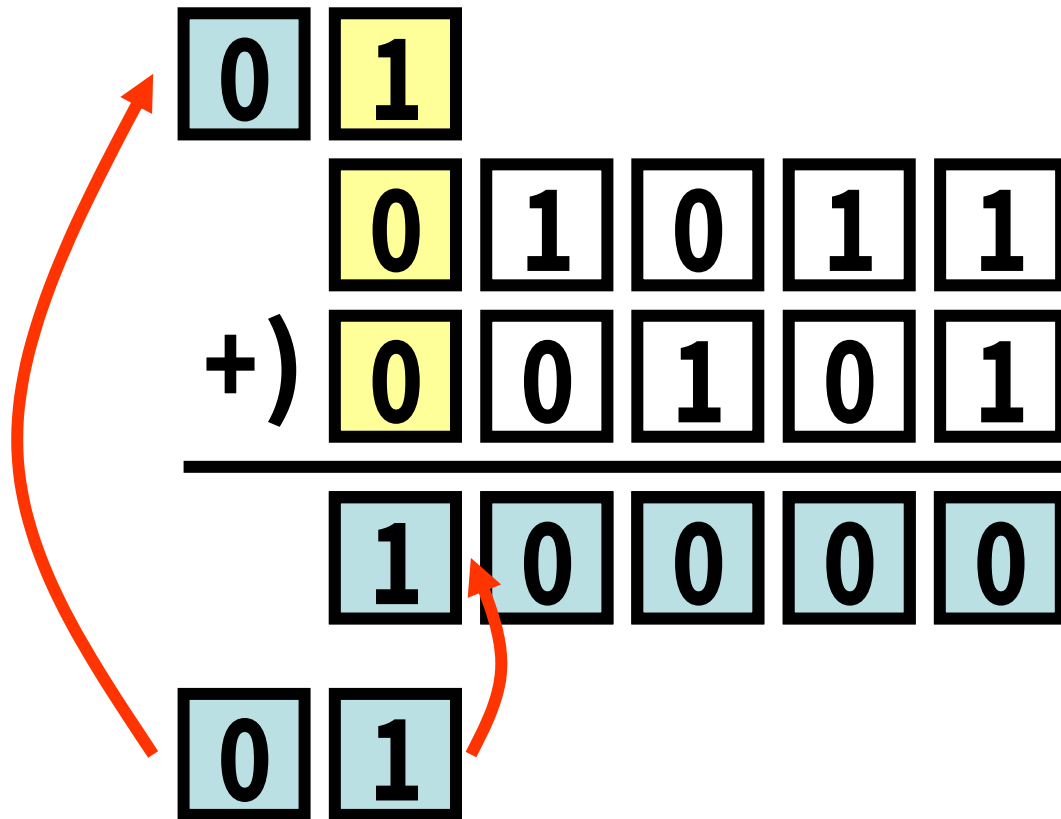
$$\begin{array}{r}
 11 \\
 01011 \\
 +) 00101 \\
 \hline
 000 \\
 10
 \end{array}$$

Diagram illustrating the addition of two binary numbers (01011 and 00101) to find their sum (10010). The numbers are aligned by their least significant bits. The addition is performed column by column from right to left, with carry bits (1s) being propagated to the next column. The final result is 10010.

2進数の和

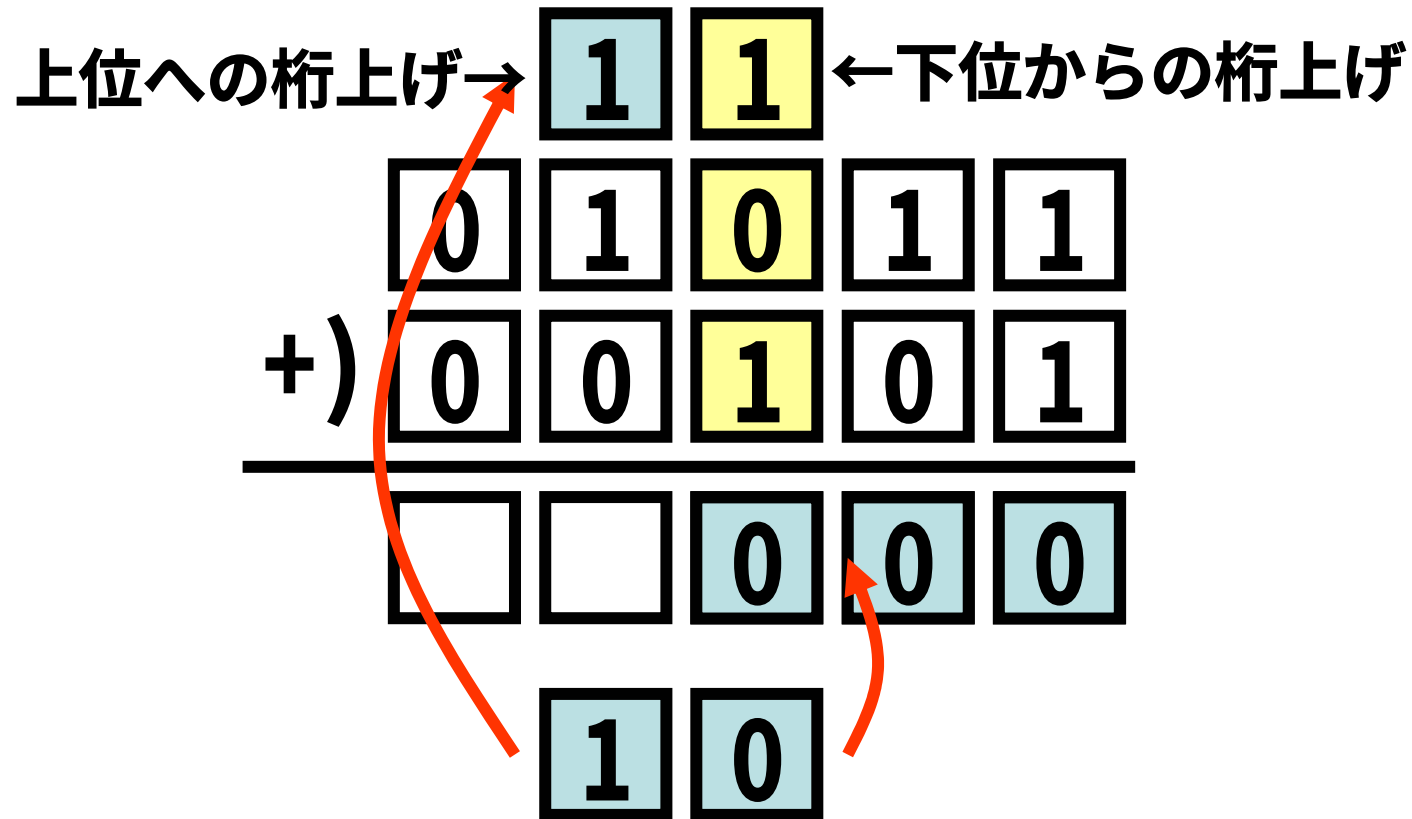
$$\begin{array}{r}
 \begin{array}{|c|c|} \hline 1 & 1 \\ \hline \end{array} \\
 \begin{array}{|c|c|c|c|c|} \hline 0 & 1 & 0 & 1 & 1 \\ \hline \end{array} \\
 +) \begin{array}{|c|c|c|c|c|} \hline 0 & 0 & 1 & 0 & 1 \\ \hline \end{array} \\
 \hline
 \begin{array}{|c|c|c|c|c|} \hline & 0 & 0 & 0 & 0 \\ \hline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

2進数の和



2進数の和

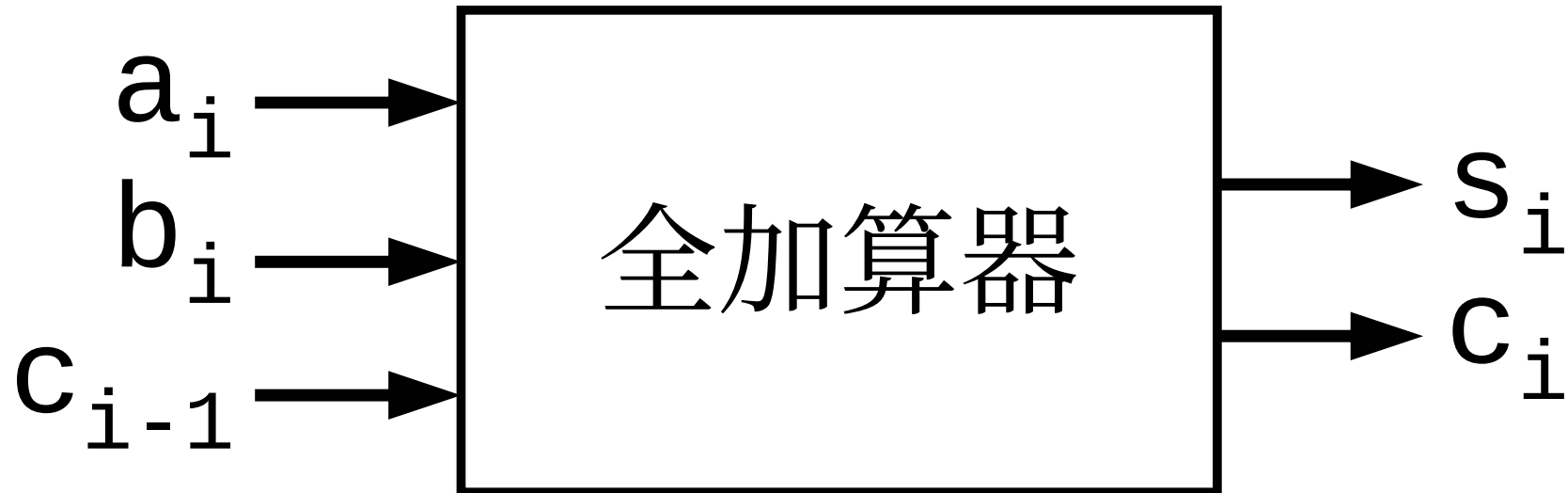
桁上げ：carry



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 桁目

c_{i-1} : 下位からの桁上げ

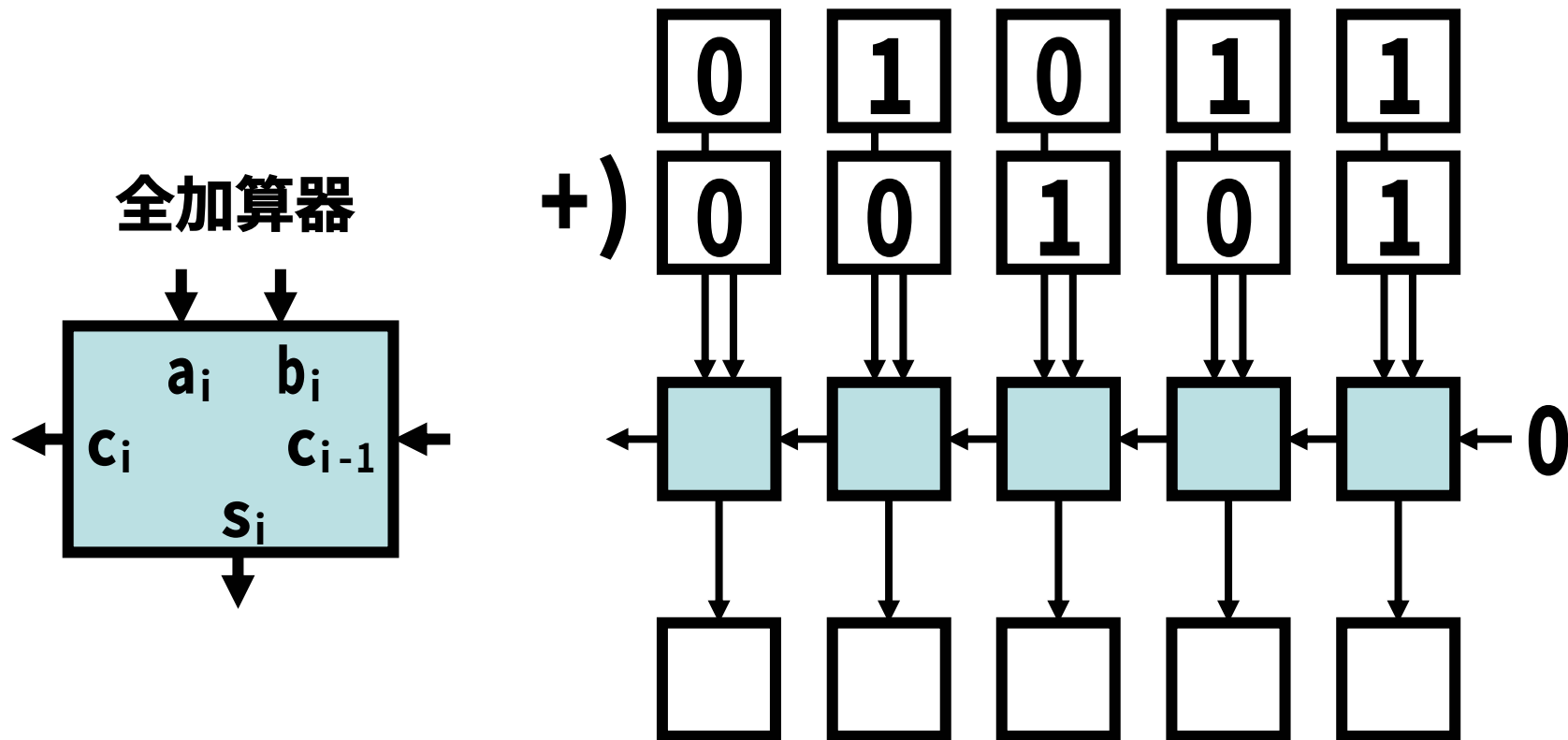


s_i : 和の i 桁目

c_i : 上位への桁上げ

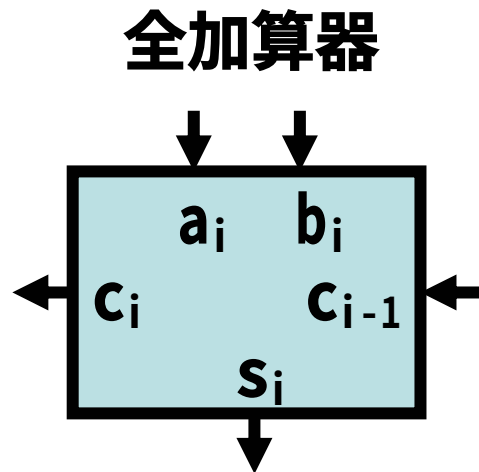
要するに . . .

$(c_i s_i)_2$ 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

$$s_i = a_i \oplus b_i \oplus c_{i-1}$$

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

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

```
int full_adder(int a, int b, int set_carry)
```

