計算機数学I (2019)
第3回

照井 章（筑波大学 数理物質系 数学域）
Akira Terui (Institute of Mathematics, University of Tsukuba)

第2回のまとめ

- 2進数，符号つき整数の表現，2の補数
- 実数の有限桁の近似と浮動小数
- 絶対誤差，相対誤差
- 丸め誤差，マシンイプシロン
- IEEE浮動小数規格

第3回の内容

- 多精度整数の表記と加算

2.1.5 多精度整数の表記と加算 (p. 15)

多倍長演算(多精度演算)

- 今後，本授業における整数，有理数などの数の演算は正確に行うことを前提にする

多倍長演算(多精度演算)

- では，1ワードに収まりきれない大きな数の演算をどのように行うか?
  (電卓は同じものを何個でも使用可能)

わが家の電卓

多倍長演算(多精度演算)

- アイデア:

  必要な桁数の分だけ電卓を並べて計算をする

上位桁 ⟵ ⟶ 下位桁

この例では 8×5=40 桁まで計算可能

非負整数の多倍長演算（多倍長整数）

- 計算機上でどうやって実現するか？
  - データの表現
  - 演算の手順（アルゴリズム）

データの表現

- a: 表す数値
- k: a を表すワード数
- $a_0, a_1, \ldots, a_{k-1}$: a を表すワード

データの表現

- $a_0, a_1, \ldots, a_{k-1}$: a を表すワード

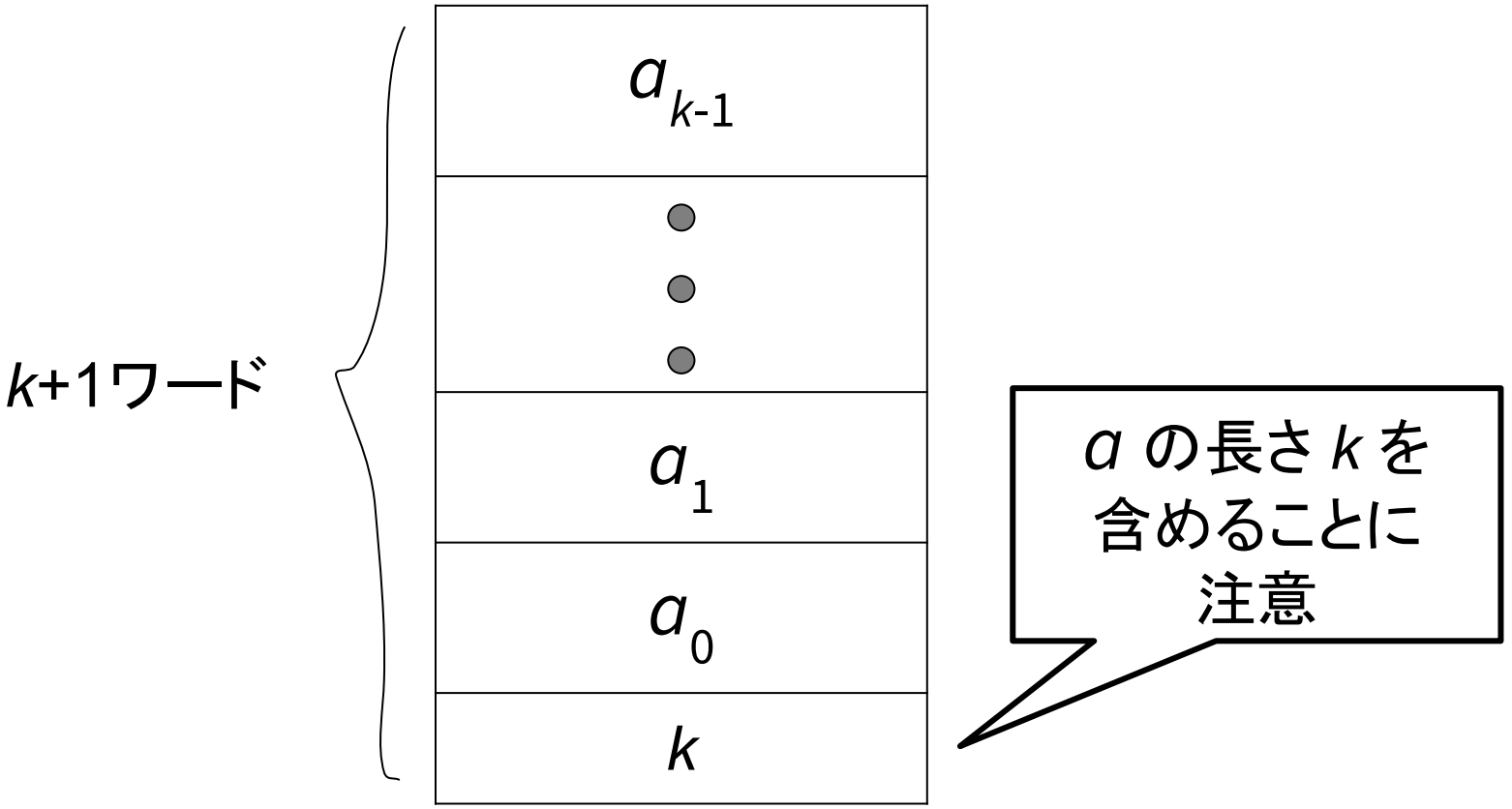
n: 1ワードの大きさ（ビット数）

$$a = \sum_{i=0}^{k-1} a_i \cdot 2^{ni}$$

$a = [a_0, a_1, \ldots, a_{k-1}]$ と表記する

データの表現

- a のメモリへの格納



データの表現

- a を2進表記するのに何ワード必要か？
  - a を2進表記した時のビット長: $\lceil \lg(a+1) \rceil$
  - 1ワードが n ビットの時，a を2進表記するのに必要なワード数: $\lceil (\lg(a+1))/n \rceil$
    =: len(a)（a の長さ）と表す
  - + len(a) の値を格納するワード: 1ワード
  - = len(a) + 1 ワードが必要

1ワードの四則演算を行うためのCPUの機構

- レジスタ: CPU内で情報（ビット列）を保持するメモリ（少量だが高速）
  - 汎用レジスタ: 演算対象や演算結果のデータを保持する
  - 状態レジスタ（補助演算装置）: CPUの演算に伴う様々な内部状態を保持する
    （今回利用するのは「桁上がりの発生を示す状態」）

1ワードの四則演算を行うためのCPUの機構

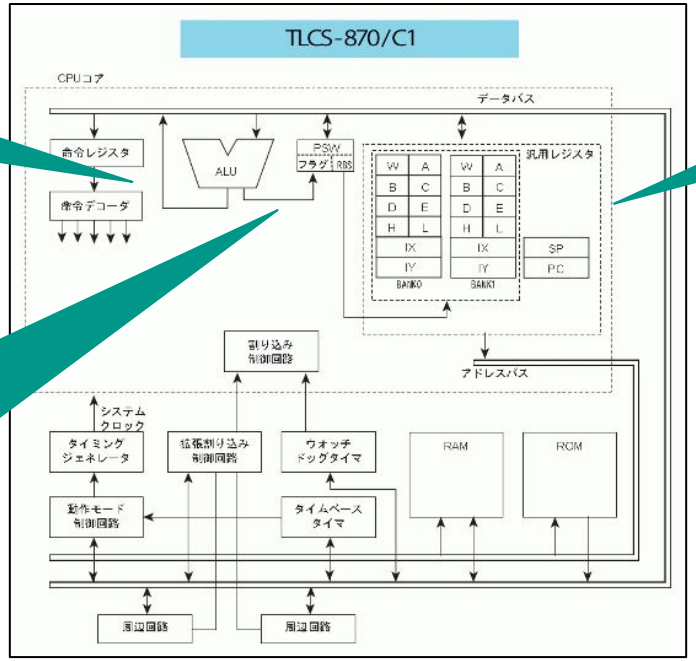
- ALU (Arithmetic Logic Unit, 算術論理演算ユニット):
  - レジスタやメモリからデータを呼び出し、四則演算や論理演算を行う回路
  - 計算結果をレジスタやメモリに格納する
  - CPUの新たな状態を状態レジスタに格納する

1ワードの四則演算を行うためのCPUの機構

東芝の組込用CPUの例 (TLCS-870/C1)

ALU

状態レジスタ
(プログラムス
テータスワード,
PSW)

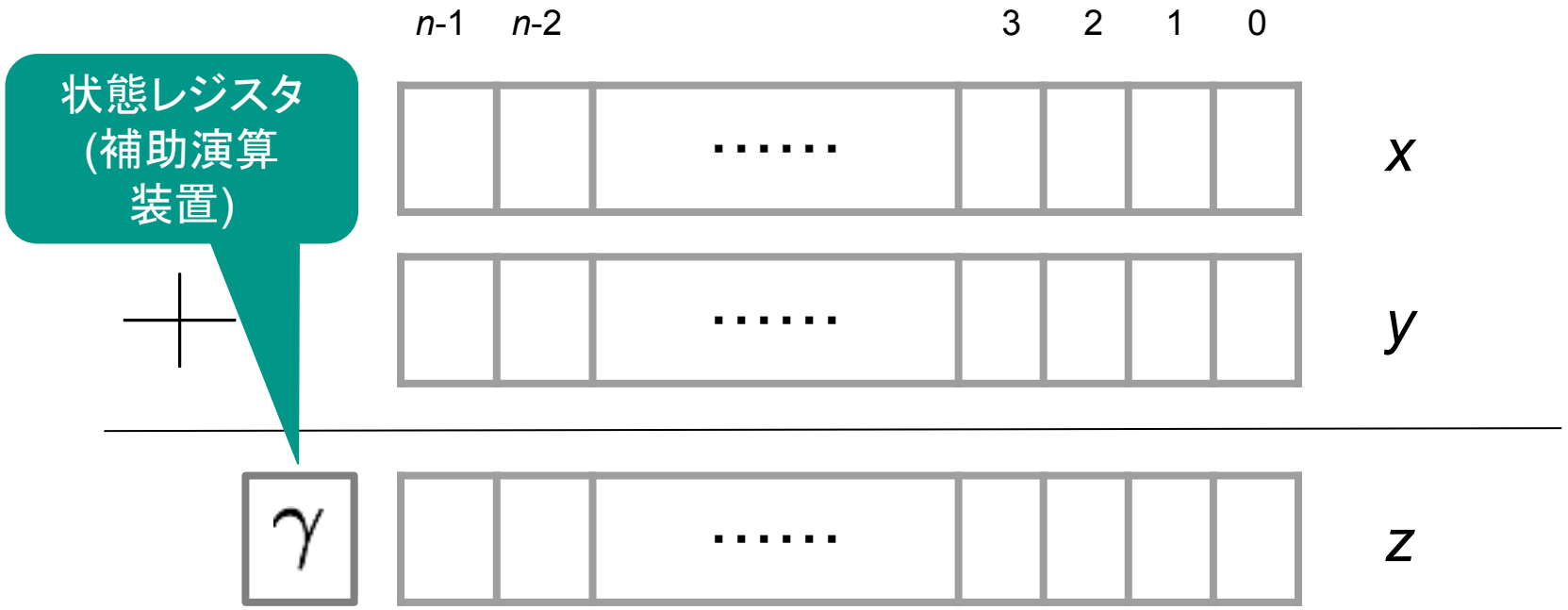


汎用レジスタ

（東芝: マイクロコンピュータ 入門コース）

https://toshiba.semicon-storage.com/jp/design-support/e-learning/micro_intro/chap4/1274739.html

1ワードの符号なし整数2個の加算

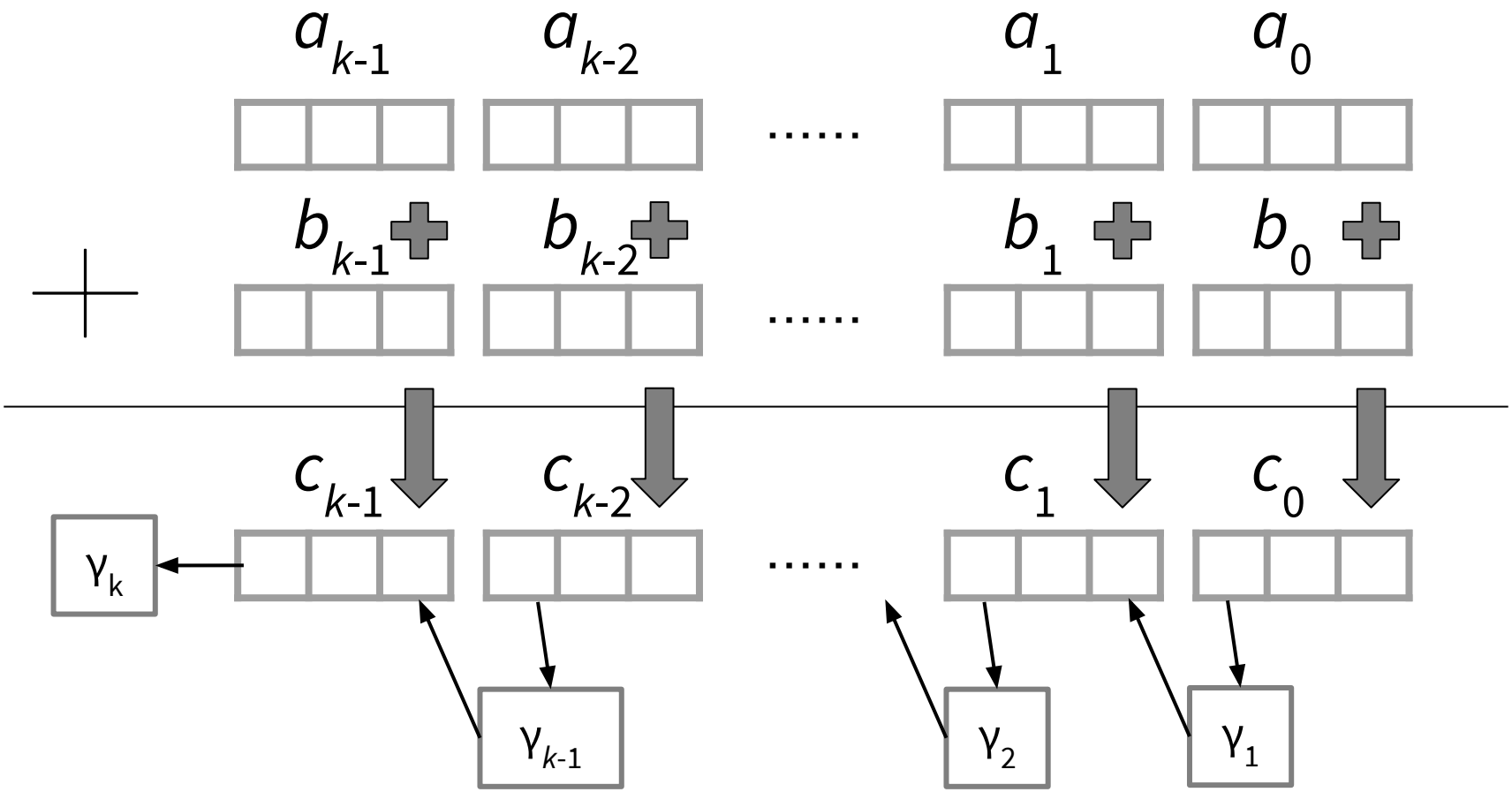


桁あふれ(繰り上がり)の有無を γ ∈ {0,1} で表す

このとき $x + y = \gamma \cdot 2^n + z$

これを $[z,\gamma] \leftarrow x + y$ で表す

長さkワードの多倍長整数 a と b の加算



繰り上がりを次々と上位ワードに加える

長さkワードの多倍長整数 a と b の加算

数式で表すと

$$a = [a_0, a_1, \ldots, a_{k-1}] = \sum_{i=0}^{k-1} (a_i \cdot 2^{ni}),$$

$$b = [b_0, b_1, \ldots, b_{k-1}] = \sum_{i=0}^{k-1} (b_i \cdot 2^{ni})$$

から

$$c = [c_0, c_1, \ldots, c_{k-1}] = \sum_{i=0}^{k-1} (c_i \cdot 2^{ni})$$

を得る

アルゴリズムの記述 (p. 7)

アルゴリズム（Algorithm, 算法）

- ある入力（式や値）に対し，ある出力（式や値）を出力するための計算手順（ステップ）

アルゴリズムの性質（要件）(Knuth)

1. 有限性：有限回のステップで停止する
2. 厳密性：すべてのステップにおける動作が厳密に定められている
3. 入力：0 個以上の入力（値）をもつ
4. 出力：1 個以上の出力（値）をもつ
5. 有効性：すべての演算が有限の長さで，十分簡潔に行われる

疑似コード (Pseudo code)

- Algorithm (アルゴリズムの名前など)
  - 入力:
  - 出力:
    - 1. (Step 1)
    - 2. (Step 2)
    - …
    - n. (Step n)

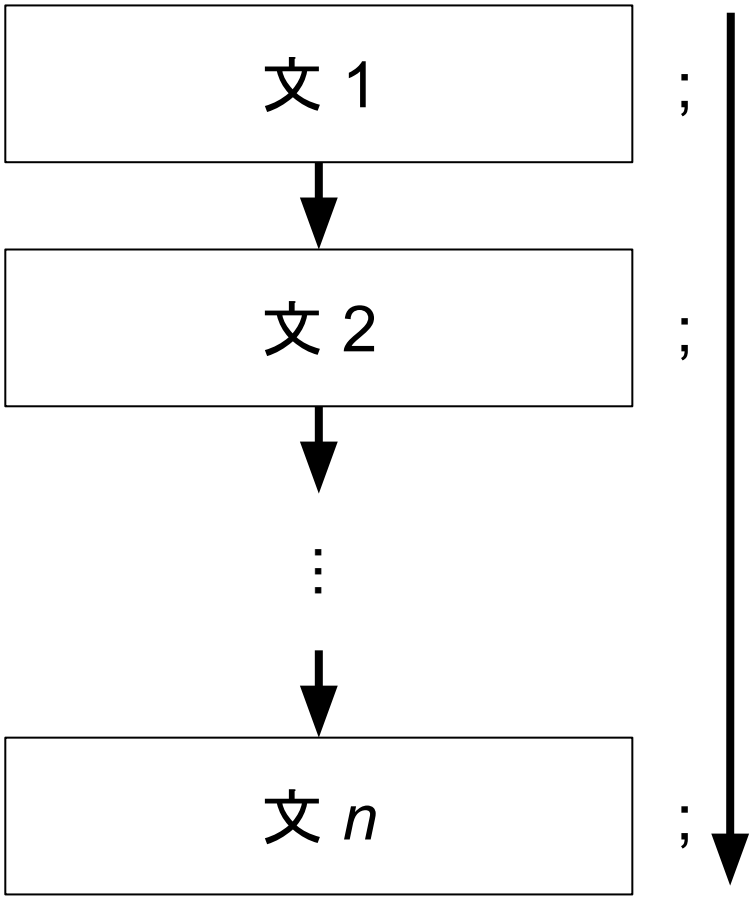
アルゴリズム特有の記法

- 変数: 数や式の値を保持する
  - 例: p, q, u, v, w, …
- 代入: 変数に数値や式を代入する
  - 例: u ← 0; v ← p(x); …
- 論理式: 真偽値 (true or false) を返す
  - 例: $u = 0$; p and q; not w
- Return: アルゴリズムの値を返し，その時点でアルゴリズムを終了する
  - 例: return p;

アルゴリズムの制御構造 / Control structures

- 逐次
  - α; β; …
- 選択
  - if α then β;
  - if α then β else γ;
- 反復
  - for α do β;
  - while α do β;
  - repeat β until α;

逐次



- 上から順番に実行する
- 各文はセミコロンで区切る

選択: if 文

if α then

β: 文（のブロック）

else

γ: 文（のブロック）

;

この部分はなくてもかまわない

- α: 論理式
- αが真ならばβを実行する
- αが偽ならばγを実行する（この部分はなくてもかまわない）

反復: for 文

for α do

β: 文（のブロック）

;

α = “$i \in [m..n]$” のとき

- $m \leqq n$: i を m から1ずつ増やして n になるまでβを繰り返す
- $m \geqq n$: i を m から1ずつ減らして n になるまでβを繰り返す

長さkワードの多倍長整数 a と b の加算

数式で表すと

$$a = [a_0, a_1, \ldots, a_{k-1}] = \sum_{i=0}^{k-1} (a_i \cdot 2^{ni}),$$

$$b = [b_0, b_1, \ldots, b_{k-1}] = \sum_{i=0}^{k-1} (b_i \cdot 2^{ni})$$

から

$$c = [c_0, c_1, \ldots, c_{k-1}] = \sum_{i=0}^{k-1} (c_i \cdot 2^{ni})$$

を得る

多倍長整数の加算のアルゴリズム

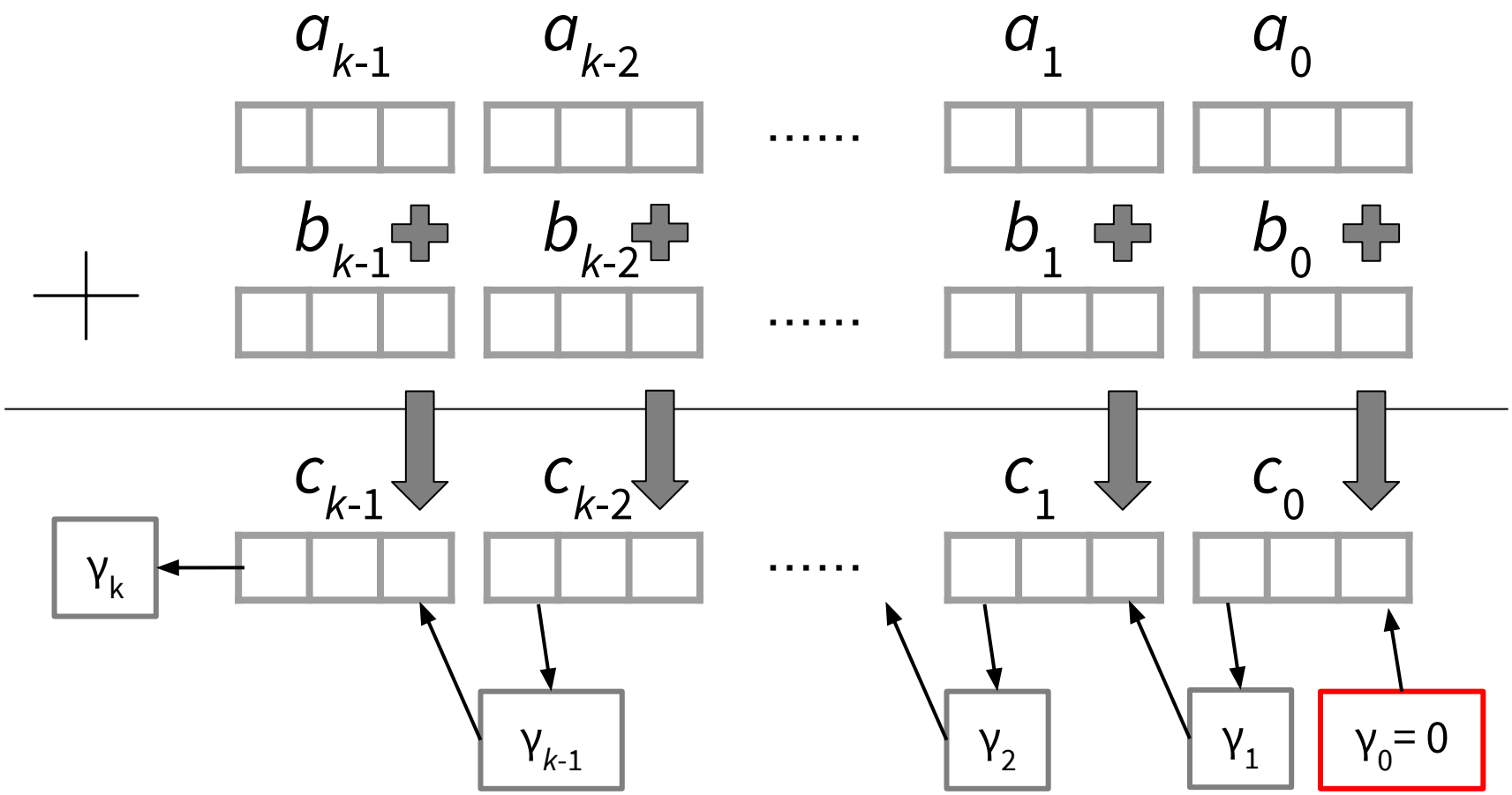
Algorithm 多倍長整数の加算 (a, b)

入力: $a = [a_0, a_1, \ldots, a_{k\text{-}1}], b = [b_0, b_1, \ldots, b_{k\text{-}1}]$

出力: $c = [c_0, c_1, \ldots, c_{k\text{-}1}]$ s.t. $c = a + b$

1. $\gamma_0 \leftarrow 0$;
2. for $i \in [0..k\text{-}1]$ do$[c_i, \gamma_{i+1}] \leftarrow a_i + b_i + \gamma_i$;
3. return $[c_0, c_1, \ldots, c_{k\text{-}1}]$;

長さkワードの多倍長整数 a と b の加算



繰り上がりを次々と上位ワードに加える

第3回のまとめ

- 符号なし多倍長整数の表現
- CPUのレジスタ

  （汎用レジスタ，状態レジスタ）
- 多倍長整数の加算の原理
- アルゴリズム，疑似コード，制御構造
- 符号なし多倍長整数の加算のアルゴリズム

第4回の内容

- 計算量の概念
- 多倍長整数の加算の計算量