3/9

2.1.5 多精度整数の表記と加算.

◎ 今後、本授業において、整数や有理数などの演算は正確に行うことを前提とする.

- 1ワードに収まり切れない数を扱う場合.
  ⟶ 複数ワードを用いて1つの数を表現し、演算を行う.

⇓

多精度演算 multiprecision arithmetic
(多倍長演算)

◎ 非負整数の多倍長演算 (多倍長整数)
(加算)

- データの表現
- 演算のアルゴリズム

} をつくる必要がある.

- データの表現.

① 桁数を表すワード数 ⟶ k.

② 桁数を表すワード ⟶ $a_0, a_1, \cdots, a_{k-1}$.

s.t. $n = \sum_{i=0}^{k-1} a_i \cdot 2^{mi}$ (n = ワードの大きさ (bit))

n 桁の電卓を並べたものと思う.
好ほう

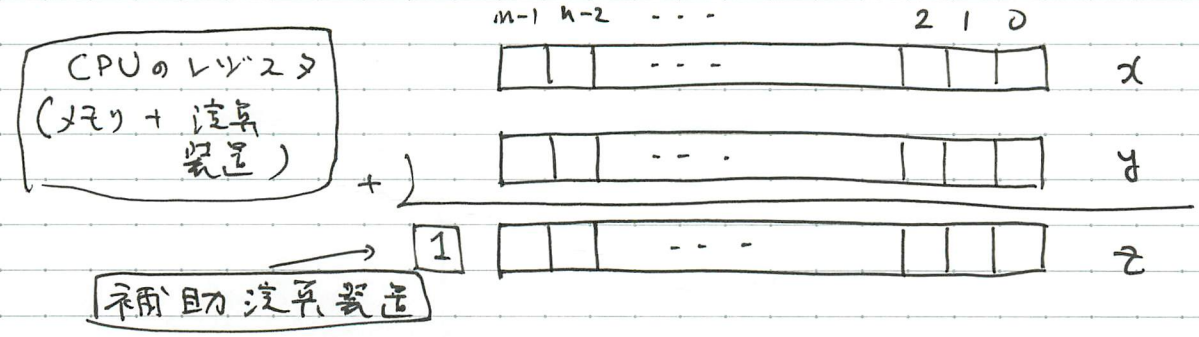
⟶ $[a_0 \cdots a_{k-1}]$ と表現する.

- これらのデータを実際にはどのような形でメモリに格納するかについては、今回は議論しない.
  (データ構造)

- $a \in \mathbb{N}$ $(a \geq 0)$ を2進表記するのに何ワード必要か？
  - a を2進表記したときのビット長 ⟶ $\lceil \lg(a+1) \rceil$
  - 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長 ⟶ $\left\lceil \frac{\lg(a+1)}{n} \right\rceil =: \underline{len(a)}$
    a の長さ．
  - + $len(a)$ それぞれを格納するワード ⟶ 1．
  - 計 $\underline{len(a)+1}$ ワードを用いる．

◎ 多倍長整数の加算．

- 1ワードの整数（2進数）2つの加算．

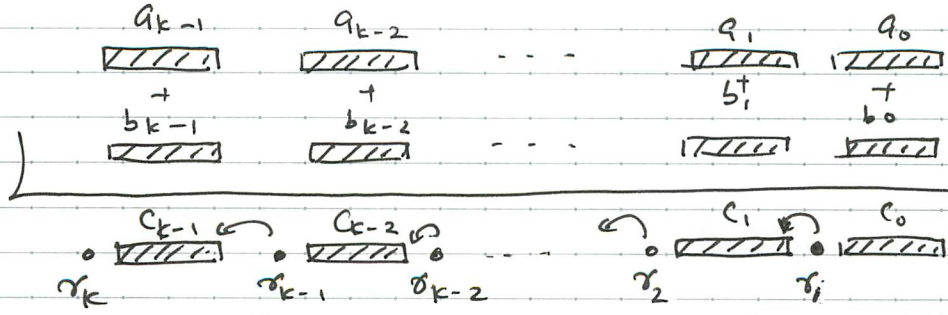


桁あふれ（繰り上がり）の有無を $\gamma \in \{0,1\}$ として格納．

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へかえる．

これを数式で表すと.

$$\text{入力}\begin{cases} a = [a_0\ a_1\ \cdots\ a_{k-1}] = \sum_{i=0}^{k-1} a_i \cdot 2^{ni}, \\ b = [b_0\ b_1\ \cdots\ b_{k-1}] = \sum_{i=0}^{k-1} b_i \cdot 2^{ni}, \end{cases}$$

より

$$\text{出力}\quad c = [c_0\ c_1\ \cdots\ c_{k-1}] = \sum_{i=0}^{k-1} c_i \cdot 2^{ni}$$

を得る.

◎ アルゴリズム (算法. algorithm)

ある入力 (式や値) に対し, ある出力 (式や値) を生成するための計算手順 (ステップ).

- アルゴリズムの特徴 (要件) (knuth)
  - 停止性: 有限回のステップで停止すること.
  - 正確性: すべてのステップにおける操作が厳密に定められていること.
  - 入力: 0個以上の入力 (値) をもつこと.
  - 出力: 1個以上の出力 (値) をもつこと.
  - 有効性: 有効な (意味のある) 計算が行われること.
    (有効でない例: 整数上の Euclid の互除法を有理数や無限桁の小数を与えよ, とすること.)

◎ アルゴリズム (多倍長整数の加算)

入力: 多倍長整数 $a = [a_0\ a_1\ \cdots\ a_{k-1}]$, $b = [b_0\ b_1\ \cdots\ b_{k-1}]$
出力: 〃 $c = [c_0\ c_1\ \cdots\ c_{k-1}]$ s.t. $c = a + b$.

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