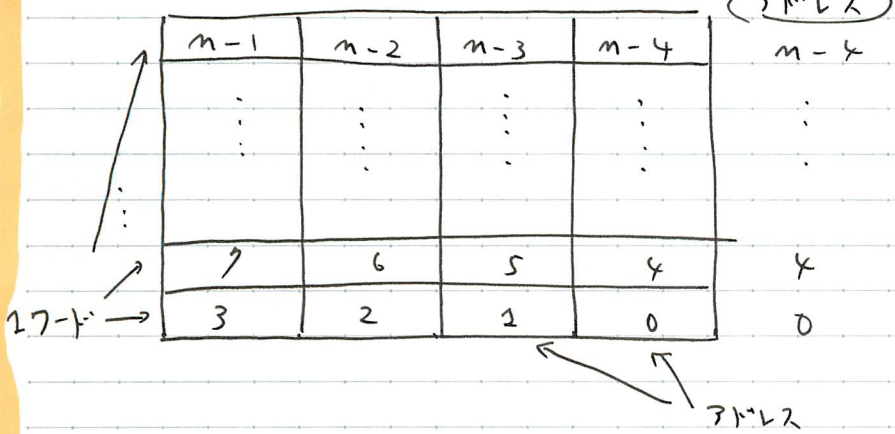
NO. 3

DATE 2015 4 10

4/20

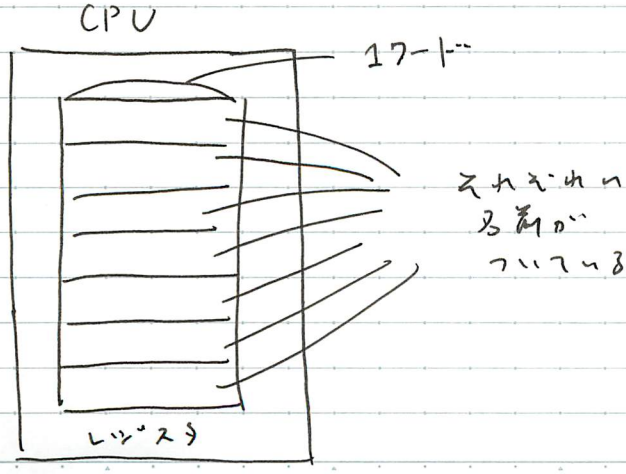
例) 1ワード = 32ビットの場合.



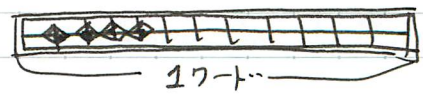
◎ CPUの レジスタ …

- CPUが内蔵する記憶回路
- メモリと同様, 1バイト～数バイト, もしくはワード単位で 設けられている.
- 個数は メモリの記憶量 に比べてずっと 小さい. (数十個程度)
- 特定の目的を持つレジスタもある. 例:
  - 演算装置とつながっていて、算術演算の対象が代入される. (事実上, これらのレジスタで 算術演算が行われると思ってよい)
  - アドレスを指定する (メモリアクセス用)
  - 現在のプログラムの実行位置 (アドレス) を保持する (プログラムカウンタ)

etc.



「ワード」の「そうがん」モデル.



各升、玉が1こっそりはいる.

- メモリ, レジスタとも何らかの値を保持する.
- メモリでは 演算はできない.
- 演算用のレジスタでは 演算ができる.

テキスト (p.9)
1.2 漸近記法 ... あとで導入.

2. 数値と数式の表現, 基本演算アルゴリズム

テキスト (p.11)
2.1 数値の表現

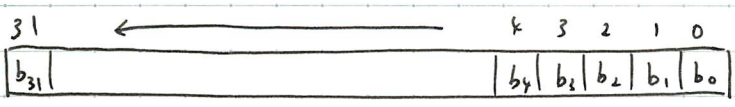
- ビット, バイト (, ワード) の概念 (⟶ p.2)
- 以後, 1ワード = nビット とする (nは32, 64の場合が多い)
- 加減乗除, などの基本演算 を ワード単位 で考える (CPUにおける算術論理演算に対応)
- 数値の表現における主題:

数値 ⟷ (1対1対応) データ表現 (符号)

2.1.1. 整数の表現 (2の補数).

- 現在, ほとんどの計算機では, 数を 2進法 で扱う.
- 以下, 1ワード = 32ビット.

◎ 符号なし整数.



$b_j \in \{0, 1\}$.

が表す整数値が u のとき,

$$u = \sum_{j=0}^{31} b_j 2^j .$$

で b_j を定める.

			十進数.
例)	0 - - - - - - - - - - - - - - - 0	=	0
	0 - - - - - - - - - - - - - - - 1	=	1
	0 - - - - - - - - - - - - - - - 10	=	2
	⋮		
	1 - - - - - - - - - - - - - - - 1	=	$2^{32}-1 = 4294967295$

⑷ 符号つき整数.

が表す数値を u とする.

u の符号 { 正 → 0 / 負 → 1

形式的に

$$u = \left(\sum_{j=0}^{30} b_j 2^j\right) - b_{31} \cdot 2^{31}$$

で b_j を定める.

- 負の数を「2の補数 (complement)」で表す.
- n に対し「$-n$」を表す手順:
  ① 「n」のビットをすべて反転
  ② 1を加える.

これにより,「$-n$」は $n + (-n) = 0$ をみたす.

例) 1ワード=4ビット (数値3ビット+符号1ビット) の場合.

0	0	0	0

⓪ 2 =

0	0	1	0

① ビット反転

1	1	0	1

② 1を加える

1	1	1	0

$= -2$.

↑ 「-8の桁」と思えばよい.

$= -2^3$

◎ 浮動小数点数 (floating point number)

例) 10ケタの10進数で、実数を近似する:

① $v = \pm \boxed{d_0}\boxed{d_1}\boxed{d_2}\boxed{d_3}\boxed{d_4} . \boxed{d_5}\boxed{d_6}\boxed{d_7}\boxed{d_8}\boxed{d_9}$ $\quad 0 \le d_j \le 9$ for $j = 0, \dots, 9$

↑ 符号の情報は別途持つ.

このとき、表現できる数値の範囲は

$-99999.99999 \le v \le 99999.99999$

(4/20) ↓

②

$v = \pm \boxed{m_0} . \boxed{m_1}\boxed{m_2}\boxed{m_3}\boxed{m_4}\boxed{m_5}\boxed{m_6}\boxed{m_7}\boxed{m_8} \times 10^{\pm \boxed{e_1}}$ (4/27)

↑ 符号の情報は別途持つ

$$\begin{cases} 0 < m_0 \le 9, \\ 0 \le m_j \le 9 \quad \text{for} \quad j = 1, \dots, 8, \\ 0 \le e_1 \le 9. \end{cases}$$

このとき $\quad v = \pm \left(m_0 + \sum_{j=1}^{8} \frac{m_j}{10^j} \right) \times 10^{\pm e_1}$

すると、表現できる数値の範囲は

$$-9.999999999 \times 10^9 \le v \le 9.999999999 \times 10^9$$

$$\| \qquad\qquad \| \qquad\qquad \| \qquad\qquad \|$$

$$-9999999990 \qquad\qquad 9999999990$$

①の表現に比べて、表現できる数値の絶対値の範囲が広がる!
同じ桁数で

但し、絶対値が大きくなる程、隣どうしの数の間隔も広がる.

②のように、指数を用いた数値の表現 ⇒ 浮動小数点 (浮動小数)

「浮動小数点」の語源: 表す数値の絶対値の大きさに応じて小数点の位置が移動するため.

⇔ ①は「固定小数点」.