コンピュータの仕組み
(6)

柴山　潔

コンピュータの仕組み

1　コンピュータシステム
2　ハードウェア
3　内部装置
4　プロセッサ(3)
5　メモリ
6　外部装置
7　論理回路
8　オペレーティングシステム

4 プロセッサ(3)

4.1 プロセッサの動作とその制御

4.2 プロセッサの機能 ―演算と制御―

4.3 演算器

4.4 順序制御機構

算術演算

- 私たち人間が普通行う計算
- 原則として，整数と実数の両方に適用

1. 単項演算：対象演算データを1個しか持たない演算
2. 2項演算：対象演算データを2個持つ演算

単項演算

- 対象演算データを1個しか持たない演算
  - 整数に適用するのが普通
  - ソースオペランドは1個

(a) 符号反転：絶対値はそのままで，演算データの符号を正から負へ，負から正へ反転

(b) 計数（カウント）：演算データを"1" 増加，減少

2項演算

- 対象演算データを2個持つ演算
  - 演算データが3個以上の「多項演算」はこの2項演算のくり返しによって行うことが可能
  - 人間が行う代表的な計算

(a) 四則演算:「加」「減」「乗」「除」の4種類の代表的な2項演算

(b) 関係演算:2個の演算データの大小比較

(c) その外の2項演算:べき乗，剰余など

四則演算

- 「加（たし）算」「減（ひき）算」「乗（かけ）算」「除（わり）算」の4種類の代表的な2項演算
  - "＋","－","×","÷"などの演算記号を演算データで真ん中に挟む書き方が人間にとっては一般的
- 整数どうし，実数どうしで行うのが普通
  - 演算結果は対象演算データと同じ整数，実数として示されるのが普通
  - ただし，整数どうしの除算では，商と剰余を整数として表す外に，商だけを実数として求める場合も

関係演算

- 2個の演算データの大小比較
  - 「より大(">")」，「より小("<")」，「等しい("＝")」，「等しくない("≠")」，「以上("≧")」，「以下("≦")」など
- 対象演算データは整数，実数
- 結果データは，比較が，「正しい」「成立（真）」か，「誤り」「不成立（偽）」か，のどちらか
  （＝論理値）

論理値

＝真理値

- 「真（本当）か偽（うそ）か」，「成立か不成立か」，「オンかオフか」など，取るべき値はたった2個でそのどちらかに決まる性質を持つデータ
  - 関係演算では，結果データは論理値
- 1個の論理値は"0"と"1"の2値のどちらか（＝ 1ビット）
  - コンピュータ（ハードウェア）の言葉はビット列（＝論理値の並び）が基本で，1ビットごとに独立した演算（＝論理演算）がコンピュータによる基本演算機能

論理演算

- 1個（ビット）の論理値どうしの演算

1. 否定，ノット(NOT)：日本語の「…でない」の意味を持つ単項論理演算
2. 論理積，アンド(AND)：日本語の「かつ」の意味を持つ2項論理演算
3. 論理和，オア(OR)：日本語の「または」の意味を持つ2項論理演算

否定，ノット(NOT)

- "0"を"1"に，"1"を"0"に変換（反転）
- 演算記号は"$\overline{\ }$"

（例）　$\overline{X}$　$\overline{0}=1$　$\overline{1}=0$

論理積，アンド(AND)

- 2個の演算データがどちらも"1"の場合にだけ演算結果は"1"，その外の組み合わせでは，演算結果は"0"
- 演算記号は"･"

（例）X･Y

0･0＝0　0･1＝0　1･0＝0　1･1＝1

論理和，オア(OR)

- 2個の演算データがどちらも"0"の場合にだけ演算結果は"0"，その外の組み合わせでは，演算結果は"1"
- 演算記号は"＋"

（例）X＋Y

0＋0＝0　0＋1＝1　1＋0＝1　1＋1＝1

（再掲）

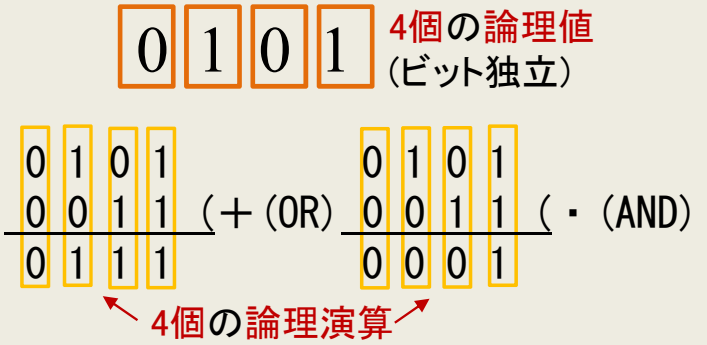
マシン語の実際

- コンピュータ（内部装置）では，情報（命令とデータ）はビット（列）か2進数かで表現
  - ビット（列）："0"か"1"の2種の記号（＝論理値）のいずれかを連結した記号（列）
    - ビット(bit)："0"か"1"の1個の記号，（参考）バイト(Byte) ＝ 8ビット
  - 2進数："0"か"1"の数字を1桁の重みを"2"とする規則に従って書き並べた数値
- マシン語 → ビット列か2進数（どちらも，"0"か"1"の列）で表現

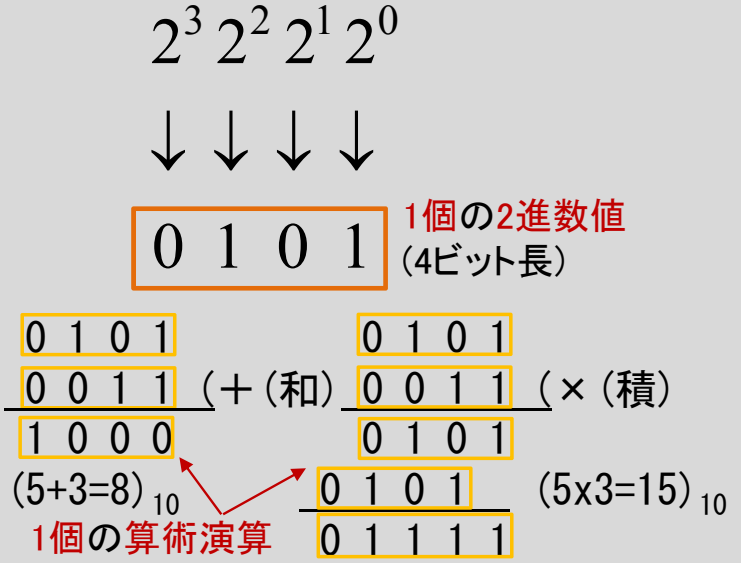
（重要）

ビット列と2進数値との比較

■ ビット列：論理値の集まり
- 各ビットが独立した1個の論理値
  - 適用演算は論理演算（否定，論理積，論理和）
  - 1ビット（個）論理演算機構は対応論理素子1個で実現



■ 2進数値：2進数表現した数値
- ビット列全体で1個の数値として意味
- 各ビットに固有の相異なる2のべき乗の重み
- 並び順がその2進数の数値を決定
  - 適用演算は算術演算（四則演算ほか多数）
  - 算術演算では，隣接（上位と下位）ビット間に依存関係
  - 算術演算機能は組み合わせ論理回路で実現



ビット列操作

- 論理値を複数個まとめて(＝ビット列)扱う演算
  - 人間の言葉(日本語，英語などの自然言語)や人間が理解できる情報(例：文字，音声，図形，画像など)をコンピュータによる論理演算をくり返すことによって処理
    - 文字は，あるまとまり(例：1字を8/16ビットで，＝符号化)ごとにビット列として表現
    - 音声，図形，画像などを表す信号も"0"と"1"のビット列で表現(＝デジタル(digital)化)，処理
  - デジタル化された信号はビット列情報(データ)として扱える
    → 論理演算によって処理可

ビット（列）で表現可能な状態数

◆**1** ビット → $(2^1=)$ **2** 状態　（0,1）

◆**2** ビット → $(2^2=)$ **4** 状態　（00,01,10,11）

◆**3** ビット → $(2^3=)$ **8** 状態　（000,001,･･･,110,111）

◆**4** ビット → $(2^4=)$ **16** 状態　（0000,0001, ･･･,1110,1111）

●***n*** ビット → $\mathbf{2^n}$ 状態

$$\underbrace{\left(\overbrace{0\cdots00}^{n\text{ビット}},\ 0\cdots01,\ \cdots,\ 11\cdots1\right)}_{2^n\text{個}}$$

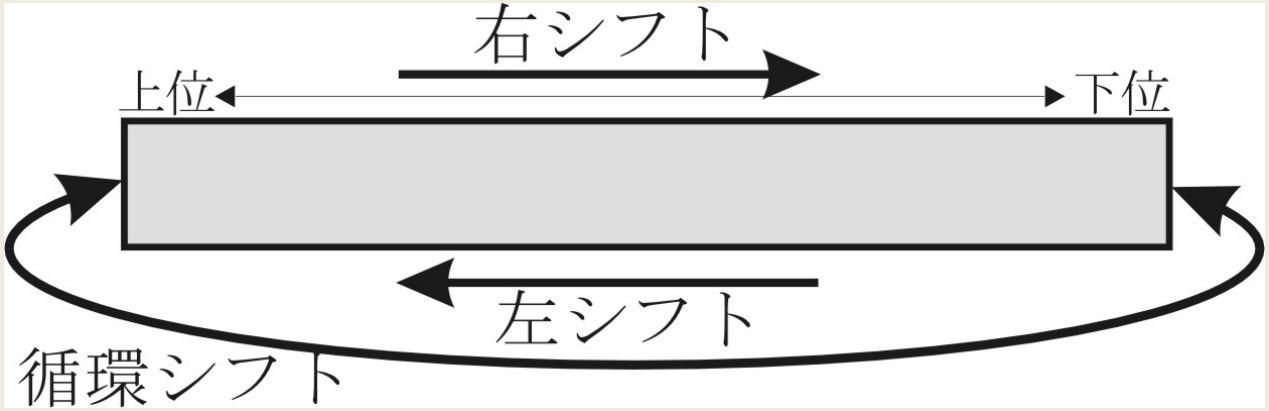
ビット列演算

1. 論理演算：1ビットの論理演算をビット列対象に拡張
2. シフト(shift)演算：ビット列をひとまとまりのデータとして左か右に移動（ずらす）操作

シフト(shift)演算(1)

- シフト操作のソースオペランドとしては，(1) 方向（左か右か）；(2) シフトするビット数；が必要

➢コンピュータ独特の（「計算」というより）処理機能



シフト(shift)演算(2)

(a) 論理シフト：ビット列を独立した論理値のまとまりとしてシフト

(b) 算術シフト：ビット列を算術データ（符号(=最上位)ビットはシフトせずに保持）としてシフトする操作

- nビット左算術シフト＝2^nをかける数（乗数）とするかけ算（乗算）
- nビット右算術シフト＝2^nを割る数（除数）とする割り算（除算）

(c) 循環シフト：シフトによって左（右）端からあふれたビット列を反対の右（左）端から詰め直すシフト

データ転送命令

- プロセッサには，有限サイズのメインメモリにある情報（処理対象データ）を別のアドレスやレジスタに退避，回復する機能が必要

● データはそのままで（加工せずに），その格納場所だけを移動するような処理

- コンピュータ独特の（「計算」というより）処理機能

順序制御(1)

- マシン命令の実行順序を決定
  - 命令実行サイクルの一番最初の「命令取り出し」ステージで，メインメモリからプロセッサへと取り出されるマシン命令の格納場所（アドレス）を決定
    - ◆現在実行中のマシン命令中に「この命令の次に実行すべき命令のアドレス」を書いておく方法　→ 限られた長さの貴重なマシン命令の一部をこのために消費
  - マシン命令を使わなくても良い方法はないか？

順序制御(2)

- 実行されるプログラムやマシン命令列はメインメモリにまとめて（＝メインメモリに付けられたアドレス順に）格納

- 「マシン命令のメインメモリへの格納順序を原則的な実行順序」と規定
  - 特に指定しない限り，マシン命令列の格納（並び）順で命令を次々に実行することが可

順序制御(3)

- あるマシン命令の実行を終了 → その命令の直後（次のアドレス）に格納してある命令をメインメモリから取り出してきて実行
  - マシン命令の一部を次命令アドレス指定用として割くことは不要

順序制御と順序制御命令

- メインメモリに格納されている順序でマシン命令を実行したくない場合には，命令実行順序を意図的に変更
  - 「次命令をオペランドとして明示する」マシン命令
    ＝順序制御命令
    - 順序制御：プロセッサが行う様々な制御機能のうちで，「『順序制御命令』というマシン命令機能として実現されているもの」