まえがき

技術を学ぶのは，当然それを使いこなすためです．そのためには，単に事実を理解するだけではなく，なぜそのような技術が考案されたのかという背景も理解しなければなりません．同じ目的を達成するにはいくつもの異なった解があるのが普通です．その中で，現在使われている技術が採用されたのには，それなりの理由があるはずです．その理由を理解できれば，これから新たな技術を自分で開発しなければならなくなったときでも最適の解を導き出すことができるようになります．

他の分野に比べると，情報処理分野の技術進歩のスピードは格段です．皆さんが学校で学ぶことは恐らく 5 年もすれば過去のものとなってしまうでしょう．10 年もすれば，皆さんより若い技術者が皆さんより先端的な技術を身に付けて社会に出てきます．そのような環境でも若い技術者に伍(ご)して活躍するには，技術の基本を理解しておくことが肝要です．そうすれば，どのような時代のどのような新規技術でも理解して使いこなすことが可能となります．

どのような本であっても，必要な技術のすべてを紹介しつくすことはできません．ですから，本はあくまで入口であって，そこを足掛かりにして必要なことを自ら学ぶ姿勢が大切です．その意味からも本書の演習問題はほとんどが「調査問題」です．必要な参考文献を見つけて学んでください．

皆さんの一人ひとりがそういった実力をつけて社会に巣立っていかれることを期待しています．

2013 年 12 月

村　岡　洋　一

目　　次

1. はじめに

2. データの表現

3. 論理回路の仕組み

4. アーキテクチャの基本

5. 仮想マシンの原理

6. コンパイラの仕組み

7. コンピュータサイエンスの香り

8. オペレーティングシステムの中核

9. ハードウェアの構成

1 はじめに

本書は，コンピュータシステムの**ハードウェア**（hardware）および**ソフトウェア**（software）について基本的な考え方を理解することを目的としています．基本的なプログラミングの知識があれば内容の理解はできるはずです．本書でコンピュータの中身について興味を持ったら，次はぜひより詳しい知識を求めて学習してください．より詳しい技術は，「コンピュータアーキテクチャ」，「言語処理系」，「オペレーティングシステム」などの科目で習うので，本書はそれらに対する入門的位置付けです．

そのため，本書で説明する事項はどれも一般的なものよりも簡素化されてはいますが，基本的な考え方は十分に学べるはずです．なお，本書に限らず技術の内容は単に暗記するのではなく，なぜそうなったかを含めて「理解」することが重要です．

コンピュータのソフトウェアおよびハードウェアは，以下のような階層構成をなしています．

応用プログラムを作成するために使われる C や Java などの高級言語
- 高級言語で書かれたプログラムを仮想マシンの命令列に変換するコンパイラ
- 仮想マシンの命令列を実行するエミュレータ
- エミュレータの実行に使われる機械語命令
- 機械語命令を実行するハードウェア
- ハードウェアを実現する論理回路

一般の利用者から見えるのは，いわゆる応用プログラム（application program）です．

応用プログラムの例は，Word や Excel などに加えて，インターネットのメールシステムやブラウザなどがあります．これらの応用プログラムは，**高級言語**（high level language：HLL）またはプログラム言語などで作成されています．この例には Java や C などがあります．

高級言語は，コンピュータにどのような処理をして欲しいかを，我々が使っている日常の言語（コンピュータの世界では自然言語といいます）にできるだけ近い言葉で記述できるようになっています．しかし，コンピュータは，残念ながらこの高級言語をそのままでは理解してくれません．コンピュータは，**機械語命令**（machine language：ML）という特別の言語を使います．したがって，高級言語で書かれたプログラムを機械語命令に変換することが必要になります．この処理をするプログラムが**コンパイラ**（compiler）です．機械語命令は，あとで説明するようにコンピュータが直接使うので，0 と 1 の組合せですが，これでは人間が理解するのは困難なので，これをもっと分かりやすい文字列の言語で表すことができるようになっています．これを**アセンブラ言語**（assembly language：ASM）といい，アセンブラ言語を機械語に変換するプログラムを**アセンブラ**（assembler）と呼びます．

以上の仕組みが基本ですが，最近のコンピュータではあとに説明するような理由から，高級言語のプログラムを機械語に変換するのではなく，**仮想マシン**（virtual machine：VM）と呼ぶ機械の命令にいったん変換します．仮想マシン命令のプログラムを実行するプログラムが **VM エミュレータ**（VM emulator）です．

コンピュータの中で実際に機械語を実行するのが **CPU**（central processing unit，**中央処理装置**）ですが，この CPU が利用者にどのように見えるか（すなわち機械語命令がどのようになっているか）を定義するのが**コンピュータアーキテクチャ**（computer architecture）です．なお，CPU は論理回路（VLSI）の組合せで実現されます．

以上に説明した高級言語，仮想マシン命令，アセンブラ言語および機械語命令がそれぞれどのようなものか，その例を以下に示します．

高級言語 ⇨ コンパイラ	for i＝1 to 100 step 1 a[i]＝…
仮想マシン命令 ⇨ 仮想マシン	push pop add…
アセンブラ命令 ⇨ アセンブラ	add $2，$3，$8…
機械語命令	100010010101110…

高級言語からアセンブラ言語までは，人間に分かりやすいように普通の言語に近い表現をとっていますが，機械語命令はコンピュータのハードウェアが処理しますので，0 と 1 の二つの組合せからなるパターンです．

2 データの表現

本章では，コンピュータの中でデータなどがどのように扱われるかについて説明します．数値データを使った演算の方法や，文字データ，音声データ，画像データなど多様な情報をコンピュータで処理できるようにするための技術を学んでください．

2.1 コンピュータが扱うデータ

それでは，まずコンピュータがどのようにデータを扱うのかから説明します．

コンピュータで扱うデータには数字はもちろん，文字，声や音楽などの音，画像そして映像と多様ですが，コンピュータの中ではすべてのデータは 0 と 1 の組合せで表されています．その理由は，コンピュータを実現するために使われている論理回路が二つの値のどちらかをとるように設計されているからです．

具体的には電気のスイッチを考えてみてください．スイッチは on か off の二つの状態のいずれかをとります．論理回路も同じような動作をしています．例えば，スイッチの on を 1 に，off を 0 に，それぞれ対応付けすることによってコンピュータの動作が設計されているのです．詳しくは 3 章で説明します．

この 0 または 1 のデータの単位を 1 ビット〔bit〕といいます．また，8 bit の集まりをバイト〔byte，B〕といいます．あとで詳しいことは説明しますが，このように 0 と 1 で表す数字列のことを **2 進数**といいます．2 進数で表される数で 2^{10} を K（キロ），2^{20} を M（メガ）といいます．10 進数のキロ（10^3）やメガ（10^6）とほぼ同じ大きさですが，キロは 10 進数の場合（小文字の k）と区別するために大文字の K を使います．

2.2 2 進 数

コンピュータの基本動作は演算ですから，まずコンピュータの中で数がどのように表されるかについて説明します．一般的に，数の表現法を決めるときに，以下のようなことを考える必要があります．

・何進数を使う？　　コンピュータでは 2 進数を使います．

・整数および小数の表現は？　　固定小数点表示と浮動小数点表示の方式があります．

・負の数の表現は？　　補数を使います．

・数の単位は？　　32 bit（1語）を単位とします．

・オーバーフローについての考慮は？　　無限大などの表現法があります．

もちろん，これ以外の方法を採用しているコンピュータもありますが，上記が代表的な方法です．

一方，我々が日常に使うのは10進数です．10進数では桁の数は0から9までで，これを超えると桁上げが起こります．しかし，一般的には必ずしも10進数である必要はありません．例えば，時間は60進数（60秒で1分，60分で1時間．60を超えると桁上げが起こります）ですし，鉛筆を数えるのは12進数（12本で1ダース）を使います．我々が10進数を使うようになったのは，一説では手の指が左右あわせて10本だったからだそうです．

それでは，一般的に N 進数はどのように表記されるのでしょうか．**図 2.1** は，m 桁の N 進数の表現例です．10進数の場合には，それぞれの桁の数は0から9の間です．同じように N 進数の場合にはそれぞれの桁の数は0から $N-1$ の間です．そして，10進数では10で桁上げが起こると同じように，N 進数では N で桁上げが起こります．10進数で書かれた数の値は，特に説明する必要はないですが，N 進数で書かれた数の値は図のようになります．先に述べたように，コンピュータの内部では0と1の二つの組合せが使われますので，コンピュータの中の数の表現は2進数です． 2進数の表現例も図に示しました．

表記：　$a_{m-1}a_{m-2}\cdots a_2 a_1 a_0 \qquad 0 \leqq a_i \leqq N-1$

値：　$Z = a_{m-1}N^{m-1} + \cdots + a_2 N^2 + a_1 N + a_0$

例：　**10 進数**　　83 941の値は

$$8\times 10\,000 + 3\times 1\,000 + 9\times 100 + 4\times 10 + 1$$

2 進数　　11001の値は

$$1\times 2^4 + 1\times 2^3 + 0\times 2^2 + 0\times 2 + 1 = 25$$

図 2.1　m 桁の N 進数の表現例

10進数の0から9（（　）内に示す）までに対応する2進数は

0 ， 1 ， 10 ， 11 ，100，101，110，111，1000，1001

(0)，(1)，(2)，(3)，(4)，(5)，(6)，(7)，(8)，(9)

です．

次に，M 進数の数 $Z)_M$ を N 進数に変換する方法について考えてみましょう．

$Z)_M$ を N 進数で書くと，**図 2.2** のようになります．それではそれぞれの桁の値をどのよ

$Z)_M$ が M 進数であるとして，これを N 進数で書いたら

$$Z_0 = a_{m-1}N^{m-1} + \cdots + a_2 N^2 + a_1 N + a_0$$

Z_0/N の余り： a_0

商：　$Z_1 = a_{m-1}N^{m-2} + \cdots + a_2 N + a_1$

Z_i/N の余り： a_i

商：　$Z_{i+1} = a_m N^{m-i} + a_{m-1}N^{m-i-1} + a_{m-2}N^{m-i-2} + \cdots + a_{i+2}N + a_{i+1}$

図 2.2　M 進数を N 進数に変換

うにすれば求められるでしょうか．この方法は簡単です．N で割っていけばよいのです．N で 1 回割れば余りとして最下位の桁の値が求められます．更にもう 1 回 N で割れば余りとして今度はその次の下位の桁の値が求められます．このような処理を続けていけばよいのです．

コンピュータにおいて使うのは 2 進数ですから，10 進数から 2 進数への変換の例を以下に示します．

	256	＋ 0 ＋	64	＋ 32 ＋	0	＋ 8 ＋	0	＋ 2 ＋	0	……10 進数
	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	
362 →	**1**	**0**	**1**	**1**	**0**	**1**	**0**	**1**	**0**	…… 2 進数
	↑	↑	↑	↑	↑	↑	↑	↑	↑	…**余り**
	$\frac{1}{2}$	$\frac{2}{2}$	$\frac{5}{2}$	$\frac{11}{2}$	$\frac{22}{2}$	$\frac{45}{2}$	$\frac{90}{2}$	$\frac{181}{2}$	$\frac{362}{2}$	… 2 で割る
	（**上位**）			⇦ **2 で割る順序**					（**下位**）	

このように 10 進数の数を 2 進数の数に変換するためには，10 進数の数を 2 で割っていってその余りを下位から並べれば完成です．

逆に，2 進数から 10 進数に変換するには，2 進数を 1010（10 進数の 10 に相当）で割っていけばよいのですが，もっと簡単には

最下位ビット　　1 の位（$2^0=1$）
次のビット　　2 の位（$2^1=2$）
次のビット　　4 の位（$2^2=4$）
次のビット　　8 の位（$2^3=8$）
……

というようにして求める方がよいでしょう．

以上は任意の進数の整数についての説明でしたが，それでは**小数**（小数点以下の数）はどうでしょうか．小数も整数と同じです．**図 2.3** に示すように，それぞれの桁の値は 0 から $N-1$ の間です．小数点以下の各桁は $1/N$，$1/N^2$，$1/N^3$，…というようになります．

表記： $0.a_{-1}\,a_{-2}\,a_{-3}\cdots a_{-m+2}\,a_{-m+1}\,a_{-m} \qquad 0 \leqq a_i \leqq N-1$

値： $Z = a_{-1}N^{-1} + a_{-2}N^{-2} + a_{-3}N^{-3} + \cdots + a_{-m+2}N^{-m+2} + a_{-m+1}N^{-m+1} + a_{-m}N^{-m}$

例： **10 進数** 0.724 6の値は

$$7\times\frac{1}{10} + 2\times\frac{1}{10^2} + 4\times\frac{1}{10^3} + 6\times\frac{1}{10^4}$$

2進数 0.1001の値は

$$1\times\frac{1}{2} + 0\times\frac{1}{2^2} + 0\times\frac{1}{2^3} + 1\times\frac{1}{2^4} = 0.562\,5$$

図 2.3 *m* 桁の *N* 進数における小数の表現例

次に，M 進数の小数を N 進数の小数へ変換する方法について考えてみましょう．

整数の場合には N で割っていって，その余りを並べていけばよかったのが，小数の場合には逆に N を掛けていきます．N を掛けた結果，小数点の上の 1 桁目に現れる数字を並べていけば，変換後の数字列が得られます．**図 2.4** を見てもらえば理解できるはずです．

求める *N* 進数での表現は：

$Z)_N = a_{-1}N^{-1} + a_{-2}N^{-2} + \cdots + a_{-m+1}N^{-m+1} + a_{-m}N^{-m}$

***M* 進数の表現 *Z* から：**

Z（これをZ_1とする）$\times N$ の整数部： a_{-1}

小数部：$Z_2 = a_{-2}N^{-1} + \cdots + a_{-m+1}N^{-m+2} + a_{-m}N^{-m+1}$

$Z_i \times N$ の整数部：a_{-i}

小数部：$Z_{i+1} = a_{-i-1}N^{-1} + a_{-i-2}N^{-2} + a_{-i-3}N^{-3} + \cdots + a_{-m+1}N^{-m+1+i} + a_{-m}N^{-m+i}$

図 2.4 *M* 進数の小数を *N* 進数の小数へ変換

10 進数の小数 0.369 2 を 2 進数の小数 0.010111…に変換する例を以下に示します．

0.369 2 × 2	0.738 4 × 2	0.476 8 × 2	0.953 6 × 2	0.907 2 × 2	0.814 4 × 2	
= 0.738 4	= 1.476 8	= 0.953 6	= 1.907 2	= 1.814 4	= 1.628 8	…
↓	↓	↓	↓	↓	↓	
0	**1**	**0**	**1**	**1**	**1**	…
（上位）			**2 進数**		**（下位）**	

索　　引

── 著者略歴 ──

村岡　洋一（むらおか　よういち）
1971 年　イリノイ大学計算機学科大学院修了
　　Ph. D.
2013 年　早稲田大学名誉教授

コンピュータの基礎
Basis of Computer Organization

2014 年 2 月 28 日　初版第 1 刷発行

検印省略

編　者　一般社団法人
電子情報通信学会
http://www.ieice.org/
著　者　村岡洋一
発行者　株式会社　コロナ社
代表者　牛来真也

112-0011　東京都文京区千石 4-46-10
発行所　株式会社　**コロナ社**
CORONA PUBLISHING CO., LTD.
Tokyo　Japan　　Printed in Japan
振替 00140-8-14844・電話(03)3941-3131(代)
http://www.coronasha.co.jp

ISBN 978-4-339-01806-6
印刷：壮光舎印刷／製本：グリーン