「計 算 機 理 論 入 門」

～ コンピュータを設計しよう ～

幸山 直人 著

スーパーサイエンスハイスクール

インターネット公開版

富山大学 理学部 数学教室

本テキストの著作権について

[1]本テキストの「インターネット公開版」は、ホームページ等で公開すると著作権法に違反する可能性のある情報を取り除いたものです。

はじめに

本テキストは、コンピュータの基本的な仕組み及びその理論の習得を目標にしています。第 1 章では、コンピュータが直接扱うことの出来る情報、すなわち、2 進符号によって表された数である 2 進数について学習します。第 2 章では、本テキストの核心となるブール代数について学習します。続いて、コンピュータの構成に必要な論理素子及び論理回路について学習します。最後に、この理論を使ってコンピュータの基本的な仕組みを解きほぐして行きます。

本テキストで学習する内容は、情報科学の分野の内容として初歩的なものでしかありません。また、不足している部分も数多くあります。しかしながら、本テキストを通してコンピュータがどのように動いているのか理解していただければ幸いです。なお、本テキストを一層引き立たせるために、NHK で放送されたプロジェクト X という番組の「国産コンピューター ゼロからの大逆転」を視聴されることを希望します。

2003 年 8 月 21 日

幸山 直人

目 次

第1章　2進数

1.1　数の表現

私たちがふだん何気なく使っている数は**10 進数**(decimal number) で、0, 1, 2, 3, 4, 5, 6, 7, 8, 9 の 10 個の文字 (数字) を使って書き表されます。例えば、10 進数123.45 は

$$1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0 + 4 \times 10^{-1} + 5 \times 10^{-2}$$

を意味しており、各位には 10^2, 10^1, 10^0, 10^{-1}, 10^{-2} のような 10 のベキ数の重みを持っています。この重みの基準となる数を**基数**(radix) と呼びます。上記の例で明らかのように、10 進数の基数は 10 です。なお、123.45 のように各位の重みを省略して書き表す方法を**位取り記数法**、$1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0 + 4 \times 10^{-1} + 5 \times 10^{-2}$ のように重みを付加して書き表す方法を**基数記数法**と呼びます。

これから学ぶコンピュータの世界では、コンピュータが直接扱うことのできる数[1]、すなわち、基数を 2 として0 と1 の 2 個の数字を使って表す **2 進数**(binary numbers) が基礎となります。例えば、10 進数の0 から10 までの数を 2 進数で表すと表 1.1 のようになり、2 進数110 が

$$110 = 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 1 \times 4 + 1 \times 2 + 0 \times 1 = (10\text{進数の})6$$

であることを確かめることができます。

10 進数	0	1	2	3	4	5	6	7	8	9	10
2 進数	0	1	10	11	100	101	110	111	1000	1001	1010

表 1.1: 10 進数と 2 進数の対応

同様に、2 進数1011.101 は

$$1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3}$$

を意味するため、10 進数の11.625 であることがわかります。

その他にも情報科学の分野では、人間に数の大きさがわかりやすく、コンピュータでも使いやすい **8 進数**(octal numbers) や **16 進数**(hexadecimal numbers) がよく使われます (付録 A 参照, p.43)。なお、基数が 10 を越える場合[2]、数字だけでは数を表現する文字が足りないため、数字とアルファベットを使って書き表します。例えば、16 進数の場合、10, 11, 12, 13, 14, 15 に対応する文字としてA, B, C, D, E, F を使って表します (表 1.2 参照)。

[1] コンピュータが最も効率よく扱うことのできる数
[2] 本テキストでは 16 進数だけが当てはまります。

10進数	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
8進数	0	1	2	3	4	5	6	7	10	11	12	13	14	15	16	17
16進数	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

表 1.2: 10進数と8進数及び16進数の対応

同様に、一般的な場合についても、r進数 (基数 r)$a_n a_{n-1} \cdots a_1 a_0 . a_{-1} \cdots a_{-m}$ は、

$$a_n \times r^n + a_{n-1} \times r^{n-1} + \cdots + a_1 \times r^1 + a_0 \times r^0 + a_{-1} \times r^{-1} + \cdots + a_{-m} \times r^{-m}$$

を意味します。ただし、a_n, a_{n-1}, $\cdots$ a_1, a_0, a_{-1}, $\cdots$ a_{-m} は、0, 1, $\cdots$ $r-1$ の内のいずれかの値を取ります。

【注意】16進数123と10進数123とは異なった数です。本テキストでは、このような混乱を避けるため、数の前に「r進数」を付けて表すか、数の後ろに「(r)」を付けて表します。なお、省略されている場合は基本的に10進数として扱います。

10進数の例: 10進数12345, 12345(10), 123, 1000(10)
2進数の例: 2進数1010.101, 1010.101(2), 1000(2)
8進数の例: 8進数1234, 5670(8), -246(8), 1000(8)
16進数の例: 16進数7F, 7F(16), -F3.A(16), 1000(16)

● コーヒーブレイク — 数の誕生 —

数の概念からそれを記憶しておくための文字である数字が生まれるまでは、他の文字の誕生より早かったと云われており、文明の発達と密接な関係を持ってきました。現在、幅広く使用されている 10 進数は、身体の一部である指などの数を基準に数え始められたものだと言われており、「handful(両手:10)」や handful of handful(10 掛ける 10) から「hundred(100)」のような名残が数多く残っています。数字についても、古代エジプトの象形数字 (エジプト文明) やバビロニアのくさび形文字 (メソポタミア文明) をはじめ、現代でもローマ数字などに色濃く残っています。

I(1), II(2), III(3), IIII(4), V(5), VI(6), VII(7), VIII(8), VIIII(9), X(10),
XI(11), XII(12), XIII(13), XIIII(14), XV(15), XX(20), XXV(25),
XXX(30), XL(40), L(50), LX(60), XC(90), C(100), CC(200), D(500), M(1000)

その他には、時間・時刻・緯度・経度・角度で用いられる12進数や60進数が古くから発達してきました。というのも、平等に物を分けたり、等分したりする場合には約数が多いほうが便利だからです。

また、数字の歴史において、古代インドのヒンズー人が「0」という概念を発見したことは重要です。現在では、この概念をいち早く取り入れ、数の表現が最もシンプルなアラビア数字が、世界の最もポピュラーな数字となりました。日本で普及し始めたのは、明治になってからです。

1.2 基数変換

r 進数から r' 進数に変換することを**基数変換**(radix transmission) と呼びます。r 進数から 10 進数への変換は、前節の基数記数法を使って容易に行うことができます。逆に、本節では、10 進数を r 進数 (特に情報科学に関係の深い 2 進数・8 進数・16 進数) へ変換する方法を学びましょう。

まず、例として、10 進数`91.6875` を 2 進数へ変換してみましょう。大まかな流れとしては、図 1.1 のように、10 進数の整数部と小数部を分けてから、それぞれ 2 進数に変換し、合成することで基数変換することができます。

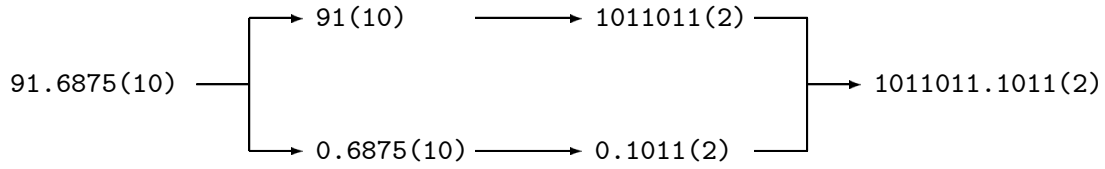


図 1.1: 整数部及び小数部を含む数の基数変換

整数部の変換 (10 進数91 を 2 進数へ変換) 計算方法 1(計算方法 2) のように商が 0 になるまで繰り返し商を基数 2 で割り、各々で出た余りを下から順に並べることで 2 進数 (`1011011`) に変換されます。

● 計算方法 1

			余り	
$91 \div 2 =$	45	$\cdots$	1	↑
$45 \div 2 =$	22	$\cdots$	1	↑
$22 \div 2 =$	11	$\cdots$	0	↑
$11 \div 2 =$	5	$\cdots$	1	↑
$5 \div 2 =$	2	$\cdots$	1	↑
$2 \div 2 =$	1	$\cdots$	0	↑
$1 \div 2 =$	0	$\cdots$	1	↑

● 計算方法 2

2)	91		余り	
2)	45	$\cdots$	1	↑
2)	22	$\cdots$	1	↑
2)	11	$\cdots$	0	↑
2)	5	$\cdots$	1	↑
2)	2	$\cdots$	1	↑
2)	1	$\cdots$	0	↑
	0	$\cdots$	1	↑

小数部の変換 (10 進数0.6875 を 2 進数へ変換) 計算方法 1(計算方法 2) のように小数部が 0 になるまで繰り返し小数部を基数 2 で掛け、各々で出た積の整数部を上から順に並べることで 2 進数 (`0.1011`) に変換されます。

● 計算方法 1

$0.6875 \times 2 =$	1.375	↓
$0.375 \times 2 =$	0.75	↓
$0.75 \times 2 =$	1.5	↓
$0.5 \times 2 =$	1.0	↓

● 計算方法 2

$$
\begin{array}{rl}
 & 0.6875 \\
\times & \quad 2 \\
\hline
 & 1.375 \quad \downarrow \\
\times & \quad 2 \\
\hline
 & 0.75 \quad \downarrow \\
\times & \quad 2 \\
\hline
 & 1.5 \quad \downarrow \\
\times & \quad 2 \\
\hline
 & 1.0 \quad \downarrow
\end{array}
$$

では、なぜこのような手順に従って計算すると 2 進数に変換することがきるのでしょうか。考察してみることにしましょう。整数部の変換については、まず、上記の各割り算の式を

$$\begin{aligned} 91 &= 45 \times 2 + 1 \\ 45 &= 22 \times 2 + 1 \\ 22 &= 11 \times 2 + 0 \\ 11 &= 5 \times 2 + 1 \\ 5 &= 2 \times 2 + 1 \\ 2 &= 1 \times 2 + 0 \end{aligned}$$

のように変形します。次に、第 1 式に第 2 式, 第 3 式, $\cdots$, 第 6 式と代入して行くと

$$\begin{aligned} 91 &= 45 \times 2 + 1 \\ &= (22 \times 2 + 1) \times 2 + 1 \\ &= 22 \times 2^2 + 1 \times 2 + 1 \\ &= (11 \times 2 + 0) \times 2^2 + 1 \times 2 + 1 \\ &= 11 \times 2^3 + 0 \times 2^2 + 1 \times 2 + 1 \\ &\vdots \qquad\qquad\qquad \vdots \\ &= 1 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2 + 1 \end{aligned}$$

となり、明らかに基数を 2 とした数の表現に変換されていることがわかります。同様に、小数部の変換の仕組みについても考察することができますが、読者にお任せします。

上記の考察を考慮すると、10 進数から一般的な r 進数への変換は、基数 2 の部分を基数 r に読み替えて計算すれば良いことがわかります。言い換えると、整数部では繰り返し基数 r で割り、小数部では繰り返し基数 r で掛けることによって r 進数に変換することができます。

再び、例として、10 進数`91.6875` を 8 進数及び 16 進数へ変換する計算手順を挙げておきます。

8 進数へ変換

Step 2 (整数部)

$91 \div 8 = 11 \cdots 3 \uparrow$
$11 \div 8 = 1 \cdots 3 \uparrow$
$1 \div 8 = 0 \cdots 1 \uparrow$

`133(8)`

Step 3 (小数部)

$0.6875 \times 8 = 5.5 \downarrow$
$0.5 \times 8 = 4.0 \downarrow$

`0.54(8)`

従って　`133.54(8)`

16 進数へ変換

Step 2 (整数部)

$91 \div 16 = 5 \cdots 11 \uparrow$
$5 \div 16 = 0 \cdots 5 \uparrow$

`5B(16)`

Step 3 (小数部)

$0.6875 \times 16 = 11.0 \downarrow$

`0.B(16)`

従って　`5B.B(16)`

例題 1 10 進数100 をそれぞれ 2 進数・8 進数・16 進数に直しなさい。

解答例

$100 \div 2 = 50 \cdots 0 \uparrow$
$50 \div 2 = 25 \cdots 0 \uparrow$
$25 \div 2 = 12 \cdots 1 \uparrow$
$12 \div 2 = 6 \cdots 0 \uparrow$
$6 \div 2 = 3 \cdots 0 \uparrow$
$3 \div 2 = 1 \cdots 1 \uparrow$
$1 \div 2 = 0 \cdots 1 \uparrow$

1100100(2)

$100 \div 8 = 12 \cdots 4 \uparrow$
$12 \div 8 = 1 \cdots 4 \uparrow$
$1 \div 8 = 0 \cdots 1 \uparrow$

144(8)

$100 \div 16 = 6 \cdots 4 \uparrow$
$6 \div 16 = 0 \cdots 6 \uparrow$

64(16)

例題 2 10 進数0.1 を 2 進数に直しなさい。

解答例

$0.1 \times 2 = 0.2 \downarrow$
$0.2 \times 2 = 0.4 \downarrow$
$0.4 \times 2 = 0.8 \downarrow$
$0.8 \times 2 = 1.6 \downarrow$
$0.6 \times 2 = 1.2 \downarrow$
$0.2 \times 2 = 0.4 \downarrow$
$0.4 \times 2 = 0.8 \downarrow$
$0.8 \times 2 = 1.6 \downarrow$
$\vdots \quad \vdots \quad \vdots$

答えは0.0001100110011$\cdots$(2) で、0011 が循環する循環小数となります。

*10 進数で実数として正しく表すことができても、2 進数で正しく表すことができるとは限りません。当然ですが、理由は 10 進数の基数には約数として 5 を含んでいますが、2 進数には含まれていないからです。数学的には 2 進数の分数で表すこともできますが、本テキストでは扱いませんので省略します。

次に、情報科学の分野でよく用いられる 2 進数・8 進数・16 進数の相互変換について学びましょう。コンピュータは 2 進数を直接扱いますが、人間は10111001101 のように 0 と 1 の羅列によって表された数の大きさを直感的に理解するのは困難です。そのため、情報科学では 2 進数を 10 進数に近い 8 進数や 16 進数に変換して表すことがよくあります。8 進数や 16 進数は、基数が 2 のベキ数のため 2 進数に関する性質を応用しやすく、2 進数・8 進数・16 進数間の基数変換を容易にします。

2 進数 $\Longleftrightarrow$8 進数 表 1.3 のように 8 進数の 1 桁は 2 進数の 3 桁に対応しています。このことを利用すると、2 進数から 8 進数への変換は、小数点を基準に 2 進数を 3 桁ずつに区切り、各 3 桁の 2 進数を対応する 1 桁の 8 進数に書き換えることによって実行されます。逆に、8 進数から 2 進数への変換は、8 進数の各桁を対応する 3 桁の 2 進数に書き換えることによって実行されます。

2 進数 $\Longleftrightarrow$16 進数 表 1.4 のように 16 進数の 1 桁は 2 進数の 4 桁に対応しており、8 進数と 16 進数の場合で異なるのは、3 桁が 4 桁になるということだけです。従って、2 進数から 16 進数への変換は、小数点を基準に 2 進数を 4 桁ずつに区切り、各 4 桁の 2 進数を対応する 1 桁の 16 進数に書き換えることによって実行されます。逆に、16 進数から 2 進数への変換は、16 進数の各桁を対応する 4 桁の 2 進数に書き換えることによって実行されます。

	2 進数	8 進数
0	000	0
1	001	1
2	010	2
3	011	3
4	100	4
5	101	5
6	110	6
7	111	7

表 1.3: 3 桁の 2 進数

	2 進数	16 進数
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

表 1.4: 4 桁の 2 進数

例として、先ほど計算した 2 進数1011011.1011 について 8 進数及び 16 進数へ基数変換を実行すると、以下のようになります。ただし、桁数を揃えるため適切に 0 を補います。

```
2 進数   001  011  011.  101  100
          ⇳    ↕    ↕    ↕    ↕    ↕
8 進数    1    3    3  .  5    4
```

```
2 進数   0101  1011.  1011
          ⇳     ↕     ↕     ↕
16 進数   5     B  .  B
```

また、このような手順による変換が成り立つことは、容易に理解することができます。なぜなら、先ほどの 2 進数1011011.1011 を小数点を基準に 3 桁ごとにまとめ、以下のように式変形を

施すと

$$
\begin{aligned}
\texttt{1011011.1011(2)} &= 1\times 2^6+0\times 2^5+1\times 2^4+1\times 2^3+0\times 2^2+1\times 2^1+1\times 2^0 \\
&\quad +1\times 2^{-1}+0\times 2^{-2}+1\times 2^{-3}+1\times 2^{-4} \\
&= (0\times 2^8+0\times 2^7+1\times 2^6) \\
&\quad +(0\times 2^5+1\times 2^4+1\times 2^3) \\
&\quad\quad +(0\times 2^2+1\times 2^1+1\times 2^0) \\
&\quad\quad\quad +(1\times 2^{-1}+0\times 2^{-2}+1\times 2^{-3}) \\
&\quad\quad\quad\quad +(1\times 2^{-4}+0\times 2^{-5}+0\times 2^{-6}) \\
&= (0\times 2^2+0\times 2^1+1\times 2^0)\times 2^6 \\
&\quad +(0\times 2^2+1\times 2^1+1\times 2^0)\times 2^3 \\
&\quad\quad +(0\times 2^2+1\times 2^1+1\times 2^0)\times 2^0 \\
&\quad\quad\quad +(1\times 2^2+0\times 2^1+1\times 2^0)\times 2^{-3} \\
&\quad\quad\quad\quad +(1\times 2^2+0\times 2^1+0\times 2^0)\times 2^{-6} \\
&= (1)\times 8^2+(3)\times 8^1+(3)\times 8^0+(5)\times 8^{-1}+(4)\times 8^{-2} \\
&= \texttt{133.54(8)}
\end{aligned}
$$

となり、明らかに 8 進数に変換されていることがわかるからです。16 進数への変換についても、同様の考察から、直ちに成り立つことがわかります。

以上をまとめると図 1.2 のようになり、10 進数・2 進数・8 進数・16 進数の相互変換をスムーズに行うことができます。なお、8 進数から 16 進数への変換や 16 進数から 8 進数への変換は、一度 2 進数に直してから変換すると効率よく変換できます。

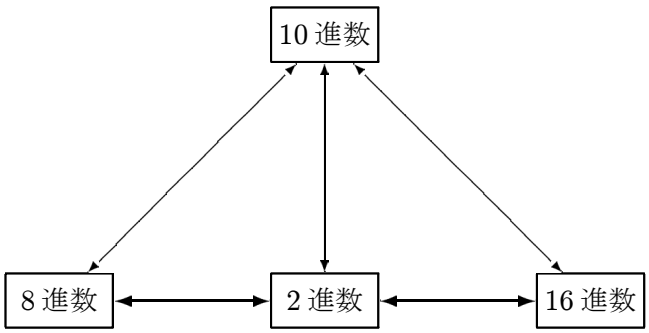


図 1.2: 10 進数・2 進数・8 進数・16 進数の相互変換

例題 3 2 進数10110101 をそれぞれ 8 進数・16 進数に直しなさい。

解答例

2 進数	010	110	101		**2 進数**	1011	0101
⇕	↕	↕	↕		⇕	↕	↕
8 進数	2	6	5		**16 進数**	B	5

例題 4　16 進数FFF を 8 進数に直しなさい。

解答例

16 進数　F　F　F
⇕　↕　↕　↕
2 進数　1111　1111　1111　2 進数に直し、
2 進数　111　111　111　111　3 桁に区切り直す
⇕　↕　↕　↕　↕
8 進数　7　7　7　7

● コーヒーブレイク — 数詞 —

私たちは、「1500 円」を「せんごひゃくえん」と読み、「いちごれいれいえん」とは読みません。このように古くからある数字の読み方を数詞と呼びます。また、中国から伝わった漢数字を用いて「千五百円」と書き表すことができます。読者の皆さんは、数詞 (漢数字) を使っていくつまで大きな数を読むことができますか？

10^0	一	いち
10^1	十	じゅう
10^2	百	ひゃく
10^3	千	せん
10^4	万	まん
10^8	億	おく
10^{12}	兆	ちょう
10^{16}	京	けい
10^{20}	垓	がい
10^{24}	秭	じょ
10^{28}	穣	じょう

10^{32}	溝	こう
10^{36}	澗	かん
10^{40}	正	せい
10^{44}	載	さい
10^{48}	極	ごく
10^{52}	恒河沙	ごうがしゃ
10^{56}	阿僧祇	あそうぎ
10^{60}	那由他	なゆた
10^{64}	不可思議	ふかしぎ
10^{68}	無量大数	むりょうたいすう

1.3 四則演算

r 進数の足し算・引き算・掛け算・割り算は、基数が r であることに注意すれば、10 進数の場合と同じように演算することができます。すなわち、1 繰り上がりや上の位から値を借りるとき 10 進数では 10 が基準となりますが、r 進数の場合は r が基準になるということです。各演算について簡単な例を挙げておきますので、計算してみてください。

例題 1 1110.011+101.11(2), 16.3+5.6(8), E.6+5.C(16) をそれぞれ計算しなさい。

解答例

```
   1110.011        16.3        E.6
 +   101.11      +   5.6      +   5.C
-----------     -------     -------
  10100.001(2)    24.1(8)     14.2(16)
```

例題 2 1110.011−101.11(2), 16.3−5.6(8), E.6−5.C(16) をそれぞれ計算しなさい。

解答例

```
   1110.011        16.3        E.6
 -   101.11      -   5.6      -   5.C
-----------     -------     -------
   1000.101(2)     10.5(8)     8.A(16)
```

例題 3 1110.011×101.11(2), 16.3×5.6(8), E.6×5.C(16) をそれぞれ計算しなさい。

解答例

```
     1110.011          16.3          E.6
 ×     101.11      ×     5.6      ×     5.C
-------------     -------     -------
     11 10011        12 62       A C8
    111 0011       107 7       47 E
   1110 011       -------     -------
  00000 00         122.52(8)   52.A8(16)
 111001 1
-------------
 1010010.10101(2)
```

例題 4 1110.011÷101.11(2), 16.3÷5.6(8), E.6÷5.C(16) をそれぞれ計算しなさい。

解答例

```
        10.1(2)          2.4(8)          2.8(16)
10111 ) 111001.1    56 ) 163        5C ) E6
        10111           134             B8
        -----           ---             --
         1011           27 0            2E 0
            0           27 0            2E 0
        -----           ---             --
         1011 1          0               0
         1011 1
        -----
            0
```

例題 5　1110.011×10(2), 16.3×100(8), E.6×1000(16) をそれぞれ計算しなさい。

解答例　1110.011×10(2)=11100.11(2)
16.3×100(8)=1630(8)
E.6×1000(16)=E600(16)

例題 6　1110.011÷1000(2), 16.3÷100(8), E.6÷10(16) をそれぞれ計算しなさい。

解答例　1110.011÷1000(2)=1.110011(2)
16.3÷100(8)=0.163(8)
E.6÷10(16)=0.E6(16)

*例題 5 や例題 6 は、10 進数の場合と同様に 0 の個数だけ小数点の位置を移動すればよい。

私達は、小学生で掛け算九九を習うなど日頃から 10 進数に慣れ親しんでおり、10 進数の四則演算をスムーズに行うことができます。しかしながら、例題のような基数が異なる数の演算は思うように計算することができません。そこで、10 進数の掛け算九九に相当する 8 進数の掛け算表や 16 進数の掛け算表 (付録 B 参照, p.45) を作成することによって計算を行う際の手助けとしましょう。

	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7
2		4	6	10	12	14	16
3			11	14	17	22	25
4				20	24	30	34
5					31	36	43
6						44	52
7							61

表 1.5: 8 進数の掛け算表

● コーヒーブレイク — 九九 —

日本では、掛け算九九を言葉遊びとして用いた歌が万葉集で歌われるなど、古くから掛け算や割り算が普及していました。例としては、「二二」と書いて「し」と読ませたり、「重二」を「し」、「二五」を「とを」、「十六」を「しし」、「八十一」を「くく」などと読ませていました。

平安朝の 970 年には、源為憲という人が七歳の長男のために作った教科書の中に九九があり、いまとは逆に九九 (八十一) から始まって一一 (一) で終わっていました。その中には、「いろは歌」の先駆けというべき 47 文字の歌があり、読み書き・そろばんの原型になっています。

1.4 補数表現

ある決まった正の数 (普通は基数のベキ数が用いられる) からある正の数を引くことで負の数を表現することを**補数表示**といい、補数表示された負の数を**補数**(complement) といいます。n 進数の場合、決まった正の数には $n^k(k > 0, n^k > |-a|)$ が用いられ、負の数 $-a$ の補数は $n^k - a$ で与えられます。例えば、10 進数の場合に 3 桁 (ある決まった正の数を $10^3 = 1000$) で補数表示すると、-10 の補数は $990(= 1000 - 10)$ となります。

ここで、補数を使った演算 (引き算) を考えて見ましょう。例えば、$30 - 10$ は

$$30 - 10 = 30 + (-10)$$

と考えることができ、-10 の補数 990 を使って 3 桁で補数表示すると

$$030 + 990$$

と書き直すことができます (補数を使って演算を行うときは、正の数も k 桁で表示)。計算すると 1020 となりますが、補数を使った演算では有効桁数 3 桁 (k 桁) で考えるため、第 4 桁の 1 を無視することで $20(= 020)$ という正しい値が求められます。もう 1 つの例として $5 - 10$ を考えてみましょう。同様に 3 桁で補数表示すると

$$005 + 990 = 995$$

となりますが、普通の数に戻すために補数を求める方法と逆の計算を行えば $1000 - 995 = 5$ となり、-5 という正しい値が求められることがわかります。以上のように、補数表示を用いることによって、負の数を正の数として、引き算を足し算として扱うことができるようになります。そのため、演算装置に減算回路が不必要となり、回路が単純になるという利点があります。この事については、3 章及び 4 章で詳しく説明します。

補数表示についてもう少し詳しく見ていきましょう。上記の例で、-10 は補数表示によって 990 と表されましたが正の数 990 とも解釈できます。そこで、補数表示では半分を正の数、残り半分を負の数として扱うのが普通で、0 から $n^k - 1$ の内、

$$\begin{aligned} 0 \quad &\text{から} \quad \frac{n^k}{2} - 1 \quad \text{を正の数} \\ \frac{n^k}{2} \quad &\text{から} \quad n^k - 1 \quad \text{を負の数} \end{aligned}$$

として扱います (注意点として、演算を行うときは上記の範囲内で行う必要があります)。従って、上記の例では表 1.6 のように対応しています。また、正の数であるか負の数であるかを判別するには、第 3 桁が 0, 1, 2, 3, 4 のとき正の数、5, 6, 7, 8, 9 のとき負の数となります。一般の n 進数の場合は、第 k 桁が 0 から $\frac{n}{2} - 1$ のとき正の数を表し、$\frac{n}{2}$ から $n - 1$ のとき負の数を表します。

実際の数	−500	−499	−498	···	−3	−2	−1	0	1	2	3	···	498	499
補数	500	501	502	···	997	998	999	000	001	002	003	···	498	499

表 1.6: 補数表現

コンピュータが扱う 2 進数の補数表示については、もう一つおもしろい性質があります。例えば、$-1011(2)$ を 8 桁で補数表示するためには

$$100000000(2) - 1011(2)$$

を計算すれば良いわけですが、

$$(100000000(2) - 1(2)) - 1011(2) + 1(2) = 11111111(2) - 1011(2) + 1(2)$$

と考えて計算してみましょう。この計算を完了するには、最初に $11111111(2) - 1011(2)$ を求めて、その後で $1(2)$ を足せば良いことがわかります。最初の計算は

$$\begin{array}{rr} & 11111111(2) \\ - & 00001011(2) \\ \hline & 11110100(2) \end{array}$$

となりますが、注意深く観察すると $00001011(2)$ と $11110100(2)$ は 1 と 0 をひっくり返しただけで、3 章で扱う論理演算 (否定) で求められることがわかります。このような数を **1 の補数**と呼びます。また、1 の補数に対して今まで求めてきた補数 (1 の補数に 1 を足した数) を **2 の補数**と呼びます。一般の n 進数の場合にも、n の補数と $n-1$ の補数[3]があり、

$$「n \text{ の補数}」 = 「n-1 \text{ の補数}」 + 1$$

が成り立ちます。上記の例、10 進数 -10 について 9 の補数を求めると

$$\begin{array}{rr} & 999 \\ - & 010 \\ \hline & 989 \end{array}$$

となり、各桁で引き算を行うことで求まります。もちろん、9 の補数に 1 を足せば 10 の補数になることは明らかです (この方法だと計算が楽で計算ミスが少なくなります)。

例題 1　$-123(10)$ について 10 の補数及び 9 の補数を求めなさい。ただし、桁数は 4 桁。

解答例　10 の補数　10000−123=9877(10)
　　9 の補数　9999−123=9876(10)

例題 2　$-1010100(2)$ について 2 の補数及び 1 の補数を求めなさい。ただし、桁数は 8 桁。

解答例　2 の補数　100000000−1010100=10101100(2)
　　1 の補数　11111111−1010100=10101011(2)

[3] $n-1$ の補数では 0 の表現が 2 種類あり、$+0$ 及び -0 と呼ばれます。

● コーヒーブレイク — 情報と情報量 —

「**情報**(information)」という言葉は、現代社会において重要なキーワードのひとつとなっています。この言葉が最初に登場したのは、明治 9 年 (1876) で、フランス語の軍事用語 renseignement の訳語として、「敵情の報知 (報告)」を縮めて「情報[a]」と名付けられたとされています。その後、ラテン語で「明確な形を与えるもの」の意味の informare を語源とする information の訳語として「情報」が使われるようになりました。現在では、広辞林などで「物事の事情, 状態についての知らせ」と説明され、必要とする人間にとって混沌としたデータの羅列ではなく明確な意味を持つものを「情報」と呼びます。すなわち、「特定の人に有用なデータ, 意味あるデータ」又は「データから取り出した意味あるもの」を示しています。逆に、「**データ**(data)」という言葉は、「観察等によって得た混沌とした事実」又は「処理されていない (意味づけされていない) 単なる事実」を示しています。すなわち、「情報」と「データ」には、

「情報」=「データ」+ 意味

という関係があり、「データ」に意味づけをしたものを「情報」と呼びます。厳密には、上記のような違いがありますが、本テキストでは「情報」≒「データ」とします。

情報を定量的に取り扱うことを初めて明確な形で提唱したのは、シャノン (C. E. Shannon, 1916–) で、1948 年に「Mathematical Theory of Communication」という情報理論に関する論文を発表しました。この論文の中で、**情報量**を表す単位として 2 進数 1 桁という意味を持つ**ビット**(bit:binary digit) が、初めて導入されました。また、2 進数 8 桁 (8 ビット) を**バイト**(byte) と呼びます。

なお、コンピュータで直接扱うことのできる全てのデータは、有限桁の 2 進数となります。

C. E. Shannon

http://www.nightgarden.com/infosci.htm

[a] 当時は、「情報」と「状報」が使われていたが、明治 30 年以降には「情報」統一された。また、「情報」という言葉が一般人の目にふれるようになったのは日清戦争 (明治 27 年) の時で、戦況を伝える新聞記事に使用された。

1.5　シフト演算

3 節の例題 5 及び例題 6 に挙げた、基数のベキ乗で掛けたり割ったりする演算を、情報科学では**シフト演算**と呼んでいます。例えば、1234.5678×10^3 の値は、小数点の位置を右に 3 桁移動させることで 1234567.8 と直ちに答えることができますし、$1234.5678 \div 10^3$ の値は、逆に小数点の位置を左に 3 桁移動させることで 1.2345678 と答えることができます。前者は、小数点を基準にすると数値を左に動かしているため**左シフト演算**と呼ばれ、後者は、小数点を基準に数値を右に動かしているため**右シフト演算**と呼ばれます。また、実際のシフト演算では、シフト演算する回数 (k 回) を付加して、k 回左シフト演算や k 回右シフト演算といいます。上記の例では、前者は 3 回左シフト演算といい、後者は 3 回右シフト演算といいます。

例題 1　123(10) を 2 回右シフト演算しなさい。

解答例　1.23(10)(= 123(10) ÷ 100(10))

例題 2　123(8) を 4 回左シフト演算しなさい。

解答例　1230000(8)(= 123(8) × 10000(8))

例題 3　123(16) を 6 回右シフト演算しなさい。

解答例　0.000123(16)(= 123(16) ÷ 1000000(16))

コンピュータは、掛け算や割り算の代わりに、シフト演算と足し算 (引き算は補数を用いる) を使って積や商を計算します。

掛け算　12 × 233 をシフト演算と足し算のみを使って計算してみましょう。12 × 233 は 12 を 233 回足すことで求まりますが、

$$\begin{aligned} 12 \times 233 &= 12 \times 200 + 12 \times 30 + 12 \times 3 \\ &= (12 + 12) \times 100 + (12 + 12 + 12) \times 10 + (12 + 12 + 12) \\ &= ((12 + 12) \times 10 + (12 + 12 + 12)) \times 10 + (12 + 12 + 12) \end{aligned}$$

と変形できることから、次の手順によって積を効率的に求めることができます。

① 12 を 2 回足す $(12 + 12 = 24)$
② 24 に対して 1 回左シフト演算を行う $(24 \times 10 = 240)$
③ 240 に 12 を 3 回足す $(240 + 12 + 12 + 12 = 276)$
④ 276 に対して 1 回左シフト演算を行う $(276 \times 10 = 2760)$
⑤ 2760 に 12 を 3 回足す $(2760 + 12 + 12 + 12 = 2796)$

2 進数の場合はもっと簡単で、例えば 1011(2) × 1010(2) は

$$\begin{aligned} 1011 \times 1010 &= 1011 \times 1000 + 1011 \times 10 \\ &= (1011 \times 100 + 1011) \times 10 \end{aligned}$$

と変形し、以下の手順によって積 1101110(2) を求めることができます。

①1011(2) に対してを 2 回左シフト演算を行う (101100(2))
② 101100(2) に 1011(2) を足す (101100(2) + 1011(2) = 110111(2))
③ 110111(2) に対してを 1 回左シフト演算を行う (1101110(2))

割り算 72207 ÷ 355 をシフト演算と足し算のみを使って計算してみましょう。72207 ÷ 355 は、言い換えると 72207 から何回 355 を引くことができるかという問題なります。上記のように引く回数を数えても良いのですが、下記の手順のようにシフト演算を用いることで、効率的に商を求めることができます。

①355 に対してを 2 回左シフト演算を行う (35500)
② 72207 から 2 回 35500 を引く (72207 − 355 × 200 = 1207)
*72207 から 200 回355 を引いたことと同じになります。
③355 に対してを 1 回左シフト演算を行う (3550)
④1207 から 0 回 3550 を引く (1207 − 3550 × 0 = 1207)
⑤355 に対してを 0 回左シフト演算を行う (355)
⑥ 1207 から 3 回 355 を引く (1207 − 355 × 3 = 142)

①から⑥の手順を実行することで商203 余り142 を得ますが、さらに下記の手順を実行することで商203.4 まで求めることもできます。

⑦355 に対してを 1 回右シフト演算を行う (35.5)
⑧ 142 から 4 回 35.5 を引く (142 − 35.5 × 4 = 0)
*142 から 0.4 回355 を引いたことと同じになります。

2 進数の場合も同様に考えると、1101110(2) ÷ 1010(2) の商1011(2) は、以下の手順によって求めることができます。

①1010(2) に対してを 3 回左シフト演算を行う (1010000(2))
② 1101110(2) から (1 回)1010000(2) を引く (1101110(2) − 1010000(2) × 1 = 11110(2))
③1010(2) に対してを 2 回左シフト演算を行う (101000(2))
④ 11110(2) から 0 回 101000(2) を引く (11110(2) − 101000(2) × 0 = 11110(2))
⑤1010(2) に対してを 1 回左シフト演算を行う (10100(2))
⑥ 11110(2) から (1 回)10100(2) を引く (11110(2) − 10100(2) × 1 = 1010(2))
⑦1010(2) に対してを 0 回左シフト演算を行う (1010(2))
⑧ 1010(2) から (1 回)1010(2) を引く (1010(2) − 1010(2) × 1 = 0(2))

● コーヒーブレイク — タイガー手廻し計算器 —

タイガー手廻し計算器は、1919 年 (大正 8 年) に国内計算器として開発がスタートし、1923 年には「虎印計算器」として商品化されました。その後、「タイガー計算器」と名称を変え、正確かつ大量の計算が必要な建築・財務・研究機関等で使用されましたが、昭和 40 年代初頭に半導体を使用した電卓が登場すると、その姿を消していきました。

タイガー手廻し計算器とコンピュータの数値計算の仕組み (補数・シフト演算) は全く同じで、コンピュータの基礎を学ぶには大変役に立ちます。筆者ホームページ (`http://kouyama.math.toyama-u.ac.jp/main/computer/personal/tiger/index.htm`) に、ウェブブラウザ上で動く、タイガー手廻し計算器のシミュレータがあるので、是非おためし下さい。

＊さらに詳しい情報については、タイガー手廻し計算器を製造・販売してきたタイガー計算器株式会社 (現：株式会社タイガー) のホームページ (`http://www.tiger-inc.co.jp`) をご覧ください。

第2章　論理演算

2.1　ブール代数

基本的な論理演算には**論理積**(AND)・**論理和**(OR)・**否定**(NOT) の 3 つがあり、**ブール代数**から定義されます。ブール代数では、**命題**(正しいか正しくないかはっきり表せるもの) に対して正しいか正しくないかということだけを問題とします。命題が正しいとき**真**(true) であるといい、正しくないとき**偽**(false) であるといいます。特に、情報科学の分野では真と偽を 1 と 0 で表します。また、ある命題が与えられたとき命題自身を A や B などの文字を使って表しますが、これを**命題変数**といいます。例えば、

「ブリは魚である」

という命題を A とおくと、この命題は正しいので、

$$A = 1$$

と表すことができます。

命題は、1 つ以上の命題に対してある演算 (論理演算) を定義することによって新しい命題を得ることができます。実際、n 個の命題を命題変数 A_1,A_2,$\cdots$,A_n で表し、ある演算を行う**論理関数**を F とすると、新しい命題は $F(A_1, A_2, \cdots, A_n)$ で表すことができます。より詳しく述べれば、命題変数は 1 か 0 の値しか取らないので、この関数は 2^n 通りの変数の組み合わせから 1 又は 0 のどちらかに対応させる演算を行います。

$$F(A_1, A_2, \cdots, A_n) = \begin{cases} 1 \\ 0 \end{cases}$$

それでは、論理積・論理和・否定の基本的な 3 つの論理演算について詳しく見ていきましょう。同時に各論理演算を記述するための**論理演算記号**と**論理式**を挙げておきます。

論理積　命題変数 A,B に対して論理積を表す論理関数 F' とすると、論理積は

$$F'(A, B) = \begin{cases} 1, & A = 1 \text{ かつ } B = 1 \\ 0, & \text{それ以外 } (A = 0 \text{ または } B = 0) \end{cases}$$

で定義され、論理積を表す論理演算記号 $\cdot$ を用いて $A \cdot B$ と論理式を記述します。

論理和　命題変数 A,B に対して論理和を表す論理関数を F'' とすると、論理和は

$$F''(A, B) = \begin{cases} 1, & A = 1 \text{ または } B = 1 \\ 0, & \text{それ以外 } (A = 0 \text{ かつ } B = 0) \end{cases}$$

で定義され、論理和を表す論理演算記号 $+$ を用いて $A + B$ と論理式を記述します。

否定　命題変数 A に対して否定を表す論理関数を F''' とすると、否定は

$$F'''(A) = \begin{cases} 1, & A = 0 \\ 0, & A = 1 \end{cases}$$

で定義され、否定を表す論理演算記号 $\overline{\ }$ を用いて $\overline{A}$ と論理式を記述します。

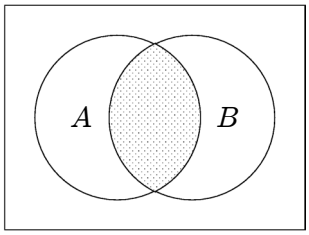
論理演算では、1 つの命題変数は 1 または 0 のいずれかで、取り得る全ての組み合わせに対して演算結果をまとめることによって、**真理値表**や**ベン図**など、表や図を使って論理関係を書き表すことができます。以下に、論理積・論理和・否定の真理値表及びベン図を挙げておきます。なお、ベン図では 1 になる部分を塗りつぶします。

論理積　$A \cdot B$

真理値表

A	B	$A \cdot B$
0	0	0
0	1	0
1	0	0
1	1	1

ベン図

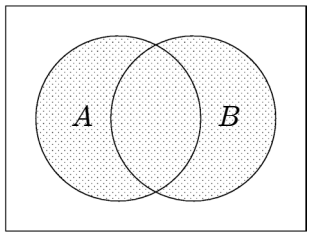


論理和　$A + B$

真理値表

A	B	$A + B$
0	0	0
0	1	1
1	0	1
1	1	1

ベン図

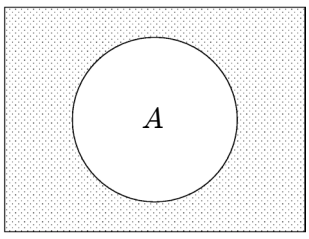


否定　$\overline{A}$

真理値表

A	$\overline{A}$
0	1
1	0

ベン図



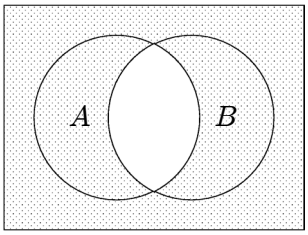
情報科学の分野では、論理積・論理和・否定の他に**否定積**(NAND[1])・**否定和**(NOR[2])・**排他的論理和**(XOR[3], EOR[4]) の 3 つの論理演算をよく使用します。なお、新しく加えた 3 つの論理演算は以下のように定義されます。

否定積 $A \mid B = \overline{A \cdot B}$

真理値表

A	B	$A \mid B$
0	0	1
0	1	1
1	0	1
1	1	0

ベン図

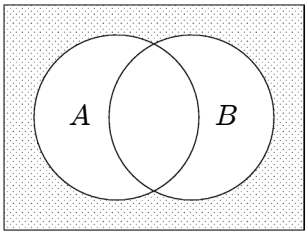


否定和 $A \downarrow B = \overline{A + B}$

真理値表

A	B	$A \downarrow B$
0	0	1
0	1	0
1	0	0
1	1	0

ベン図

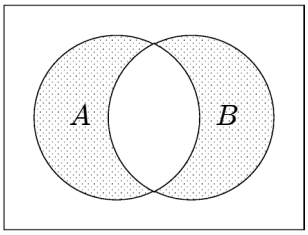


排他的論理和 $A \oplus B = \overline{A} \cdot B + A \cdot \overline{B}$

真理値表

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

ベン図



● コーヒーブレイク — ブール —

ブール (George Boole,1815-1864) は、貧しい環境に育ちましたが、父に教えてもらった数学に大変興味を持ちました。彼の代表作「Laws of Thought」では、論理学に公理を設定し、論理を数学的に表現しました。この仕事が評価されるようになったのは、計算機が彼の論理を基礎にしていて、それが普及したことにあります。

[1]NAND = Not AND.
[2]NOR = Not OR.
[3]XOR = eXclusive OR.
[4]EOR = Exclusive OR.

2.2　ブール代数の基本公式

基本的な論理演算を理解したところで、論理演算に関する基本公式と証明方法について学習しましょう。最初に注意しなければならないことは、論理演算にも四則演算のように演算に優先順位があるということです。論理積・論理和・否定および括弧には表 2.1 ような演算の優先順位が決められています。

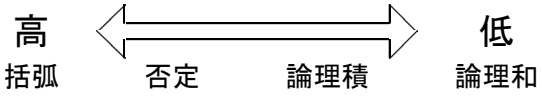


表 2.1: 演算の優先順位

論理演算に関する主な基本公式は表 2.2 のようなものがあります。証明は、命題変数が取り得る全ての組み合わせに対して等式が成り立つことを述べればよいので、真理値表やベン図を描くことによって証明できます。なお、否定積・否定和・排他的論理和については、交換法則や**一般結合法則**が成り立つことが明らかでないので論理演算を行う際は注意が必要です。証明については読者にお任せします。

$\overline{\overline{A}} = A$		二重否定
$A + B = B + A$	$A \cdot B = B \cdot A$	交換則
$(A + B) + C = A + (B + C)$	$(A \cdot B) \cdot C = A \cdot (B \cdot C)$	結合則
$A + B \cdot C = (A + B) \cdot (A + C)$	$A \cdot (B + C) = A \cdot B + A \cdot C$	分配則
$A + A = A$	$A \cdot A = A$	ベキ等則
$\overline{A + B} = \overline{A} \cdot \overline{B}$	$\overline{A \cdot B} = \overline{A} + \overline{B}$	ド・モルガンの定理
$A + 0 = A$	$A \cdot 1 = A$	
$A + 1 = 1$	$A \cdot 0 = 0$	
$A + \overline{A} = 1$	$A \cdot \overline{A} = 0$	
$A + A \cdot B = A$	$A \cdot (A + B)$	
$A + \overline{A} \cdot B = A + B$	$A \cdot (\overline{A} + B) = A \cdot B$	
$A \cdot B + A \cdot \overline{B} = A$	$(A + B) \cdot (A + \overline{B}) = A$	

表 2.2: 論理演算の基本公式

ド・モルガンの定理と呼ばれる有名な等式 $\overline{A + B} = \overline{A} \cdot \overline{B}$ を証明してみましょう。全ての組み合わせについて等式が成り立てばよいので、表 2.3 のような真理値表を描くことで証明が完了します。なお、慣れるまでは一度 $A + B$, $\overline{A}$, $\overline{B}$ を求めてから、$\overline{A + B}$ や $\overline{A} \cdot \overline{B}$ を求めると誤りが少なくなります。

A	B	$A+B$	$\overline{A+B}$	$\overline{A}$	$\overline{B}$	$\overline{A}\cdot\overline{B}$
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

表 2.3: 等式 $\overline{A+B}=\overline{A}\cdot\overline{B}$ の証明

● コーヒーブレイク — コンピュータのモデルとなった「チューリング・マシン」 —

イギリスの数学・論理学者であるアラン・チューリング (Alan Mathison Turing, 1912-1954) は、1936 年に「計算し得る数について、決定問題への応用」という論文を発表しました。その中で、無限に長いテープとそのテープに情報を読み書きするヘッドを持った、いくつかの簡単な基本操作によって動く機械を想定し、その機械の有限回の操作で数学の形式体系と等価な働きをすることを導きました。この機械は、その後のコンピュータの原型になっています。

また、この論文の中で、自動計算機を使用しても解けない、いくつかの型の数学的問題が存在することを結論づけています。問題が正しいアルゴリズムによって再記述されてのち、初めて計算可能になりますが、多くのタイプの数学的問題や論理学的問題はアルゴリズムに変換できません。即ち、ある種の「実数」は機械的な方法では計算できないのです。さらに、どの数が計算可能で、どの数が計算不可能なのか見極める方法もありません。たとえ、例外的に可能な場合があったとしても、これら全てを自動計算するためのプログラムを書くことは不可能です。こうして、全ての数学的命題を証明することは不可能であると結論づけました。

A. M. Turing

2.3　万能演算系

基本的な論理演算である論理積・論理和・否定は、全ての論理関数を構成することができるため**万能演算系**と呼ばれています。このことは、次章の論理回路と深い関係を持っているので、詳しく説明します。

最初に命題変数が 1 つの場合について考えみましょう。このとき、元の命題変数 A の値と新しい命題 $F(A)$ の値の組み合わせは 4 通りで、1 変数の論理関数は表 2.4 の F_0,F_1,F_2,F_3 のいずれかになります。

A	0	1	論理式
$F_0(A)$	0	0	0
$F_1(A)$	0	1	A
$F_2(A)$	1	0	$\overline{A}$
$F_3(A)$	1	1	1

表 2.4: 1 変数の論理関数

F_1 と F_2 は A の関数となっており、論理関数 F は否定のみで表すことができます。また、F_0 と F_3 はそれぞれ 0 と 1 に固定されています。従って、1 変数の論理式は $0, 1, A, \overline{\ }$ で全て表すことができます。

A B	0 0	0 1	1 0	1 1	論理式
$F_0(A,B)$	0	0	0	0	0
$F_1(A,B)$	0	0	0	1	$A \cdot B$
$F_2(A,B)$	0	0	1	0	$A \cdot \overline{B}$
$F_3(A,B)$	0	0	1	1	A
$F_4(A,B)$	0	1	0	0	$\overline{A} \cdot B$
$F_5(A,B)$	0	1	0	1	B
$F_6(A,B)$	0	1	1	0	$\overline{A} \cdot B + A \cdot \overline{B} = A \oplus B$
$F_7(A,B)$	0	1	1	1	$A + B$
$F_8(A,B)$	1	0	0	0	$\overline{A + B} = A \downarrow B$
$F_9(A,B)$	1	0	0	1	$\overline{A} \cdot \overline{B} + A \cdot B$
$F_{10}(A,B)$	1	0	1	0	$\overline{B}$
$F_{11}(A,B)$	1	0	1	1	$A + \overline{B}$
$F_{12}(A,B)$	1	1	0	0	$\overline{A}$
$F_{13}(A,B)$	1	1	0	1	$\overline{A} + B$
$F_{14}(A,B)$	1	1	1	0	$\overline{A \cdot B} = A \mid B$
$F_{15}(A,B)$	1	1	1	1	1

表 2.5: 2 変数の論理関数

次に命題変数が 2 つの場合について考えてみましょう。命題変数が 1 つの場合に習うと、元の命題変数 A,B の値と新しい命題 $F(A,B)$ の値の組み合わせは 16 通りになるので、2 変数の論理関数は表 2.5 の F_0,F_1,$\cdots$,F_{15} のいずれかになります。従って、命題変数が 2 つの場合も、論理積・論理和・否定によって全ての関数を表すことができるので、2 変数の論理式は $0, 1, A, B, \overline{\ }, \cdot, +$ で全て表すことができます。

同様に、多変数の場合も 2 変数の場合に帰着することができるので、論理積・論理和・否定で全ての関数を表すことができることは明らかです (帰納法によって証明)。

上記で述べたように、論理積・論理和・否定で全ての論理演算を構成することができますが、特に、万能演算系の組が独立なものを**最小万能演算系**といい、論理積・否定の組や論理和・否定の組があります。論理和と否定の組が最小万能演算系であることを証明するには、論理積が論理和と否定で表せることと論理和と否定が独立であることを示せばよいことがわかります。前者については、ド・モルガンの定理を用いれば

$$A \cdot B = \overline{\overline{A \cdot B}} = \overline{\overline{A} + \overline{B}}$$

なので、論理積は論理和と否定で表せることが直ちにわかります。後者については、論理和と否定が独立でないと仮定して矛盾を示します。もし、独立でないなら否定は論理和で表せることになるので、1 変数の場合に全ての関数が論理和で表せることになり、表 2.4 の $F_2(A)$ に対応する論理式を $A + A$, $A + 1$, $A + 0$ のいずれかで表すことができることになります。ところが、どれも否定を表すことができないので矛盾となり、論理和と否定が独立であることが示されます。

さて、このような最小万能演算系は全ての演算を構成することができますので、次章の論理回路を構成する上で最小万能演算系を実行する回路があれば必要かつ十分であるといえます。実際のコンピュータでは、1 つの演算で最小万能演算系となる否定積または否定和によって論理回路が構成されています。実際、否定・論理積・論理和・否定和・排他的論理和を否定積のみで表すと表 2.6 のようになり、最小万能演算系であることが証明されます。

$$\begin{aligned}
\overline{A} &= \overline{A \cdot A} = A \mid A \quad (\text{べき等則 } A = A \cdot A \text{ より}) \\
A \cdot B &= \overline{\overline{A \cdot B}} = \overline{A \mid B} = \overline{(A \mid B) \cdot (A \mid B)} = (A \mid B) \mid (A \mid B) \\
A + B &= \overline{\overline{A + B}} = \overline{\overline{A} \cdot \overline{B}} = \overline{A} \mid \overline{B} = (A \mid A) \mid (B \mid B) \\
A \downarrow B &= \overline{A + B} = \{(A \mid A) \mid (B \mid B)\} \mid \{(A \mid A) \mid (B \mid B)\} \\
A \oplus B &= \overline{\overline{A \cdot \overline{B} + \overline{A} \cdot B}} = \overline{\overline{A \cdot \overline{B}} \cdot \overline{\overline{A} \cdot B}} = \overline{A \cdot \overline{B}} \mid \overline{\overline{A} \cdot B} \\
&= (A \mid \overline{B}) \mid (\overline{A} \mid B) = \{A \mid (B \mid B)\} \mid \{(A \mid A) \mid B\}
\end{aligned}$$

表 2.6: 否定積による基本的な論理演算の表現

第3章　論理回路

3.1　半導体

半導体の電気的な性質は、電気をよく通す導体と電気を通さない絶縁体の中間に位置します。主な成分は**シリコン**(Si) やゲルマニウム (Ge) で、**リン**(P) やヒ素 (As) を少量混ぜた **N 形半導体**と、**ホウ素**(B) やインジウム (In) を少量混ぜた **P 形半導体**があります (シリコンに対して 1000 万分の 1 の割合)。以後、シリコンを主成分とし、リンを含んだ N 形半導体とホウ素を含んだ P 形半導体について話を進めていきます。

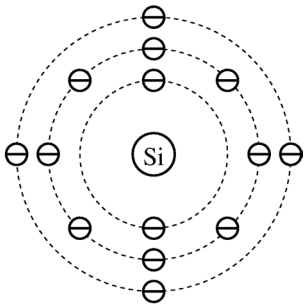


図 3.1: シリコンの原子構造

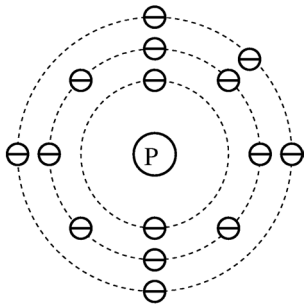


図 3.2: リンの原子構造

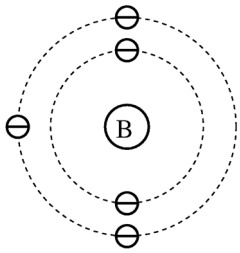


図 3.3: ホウ素の原子構造

高校の化学で習ったように、シリコン・リン・ホウ素の各原子の原子構造は上図のようになっていています。また、各原子は内側から K 殻・L 殻・M 殻といった殻があり、K 殻では 2 個・L 殻では 8 個・M 殻では 8 個の電子が存在することによって安定状態になることが知られています。従って、シリコンは、図 3.4 のように隣り合う 4 つの原子からそれぞれ 1 個の電子を共有して、電気的に安定した状態になります。このような結合を**共有結合**と呼びます。

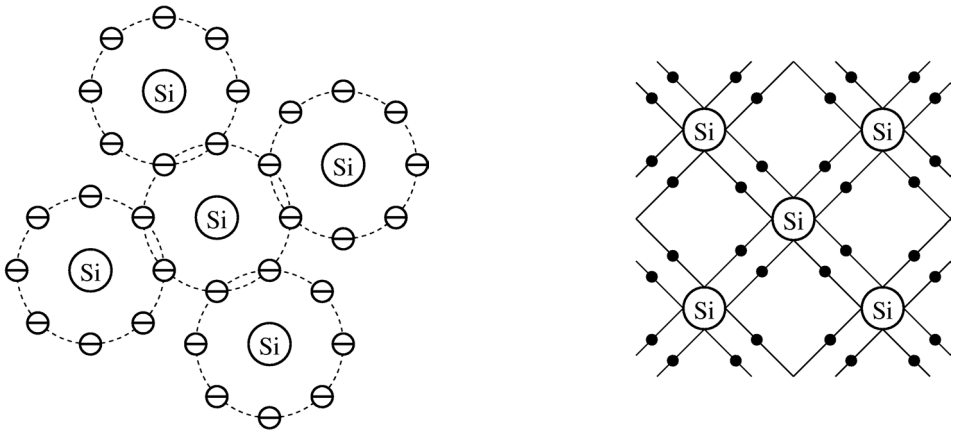


図 3.4: シリコンの共有結合

N 形半導体や P 形半導体は、シリコンに少量のリンやホウ素を混ぜて、電気的に不安定な状態を作ります。N 形半導体では、図 3.5 のようにリンの最外殻 (N 殻) の 5 つの電子のうち 4 つは共有結合に使われ、矢印で示した 1 個の電子が余った状態になります。逆に、P 形半導体では、図 3.6 のようにホウ素の最外殻 (M 殻) の 3 つの電子は共有結合を行いますが、電子が 1 個足りないため矢印で示した**ホール**と呼ばれる状態になります。

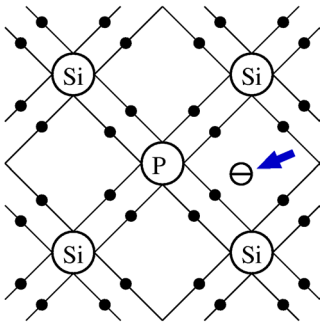


図 3.5: N 形半導体

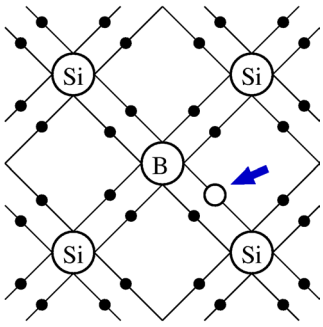


図 3.6: P 形半導体

本題に入りますが、上記で説明した N 形半導体と P 形半導体を組み合わせることで、論理回路の元となる電子部品が作られます。代表的なものには**ダイオード**と**トランジスタ**があります。

ダイオード　ダイオードは、図 3.7 のように N 形半導体と P 形半導体を **PN 接合**することで、P 形半導体から N 形半導体への一方向しか電流を流さないという性質を持ちます。なお、P 形半導体の端子と N 形半導体の端子をそれぞれ**アノード**及び**カソード**と呼び、ダイオードの回路記号は図 3.8 によって表示します。

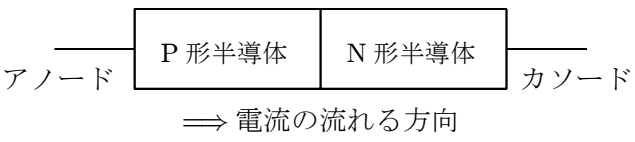


図 3.7: ダイオードの接合図

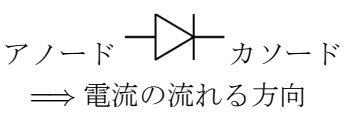


図 3.8: ダイオードの回路記号

ダイオードの働きについて詳しく見てみましょう。先ほど述べたように、ダイオードは、一定方向のみ電流を流す性質を持っていますが、図 3.9 のように電流が流れる場合を**順方向バイアス**と呼び、図 3.11 のように電流が流れない場合を**逆方向バイアス**と呼びます。順方向バイアスでは、電子はカソードからアノードへ流れようとするため、PN 接合面では N 形半導体の余った電子が P 形半導体のホールへ移動し、N 形半導体から P 形半導体へ電子が流れます。すなわち、P 形半導体に入った電子は図 3.10 のよにホールを動きながらアノードへ達し正極へ流れて行き、カソードでは負極から N 形半導体へ次々に電子を供給するので、回路に電流が流れます。

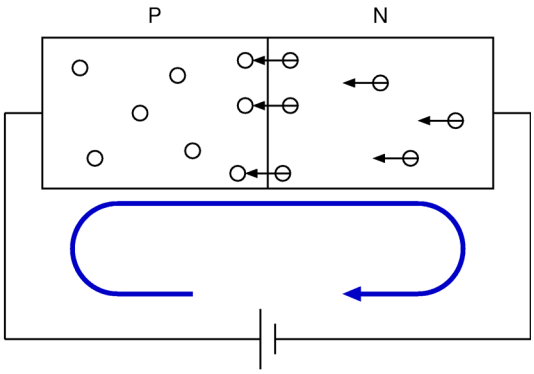


図 3.9: 順方向バイアス

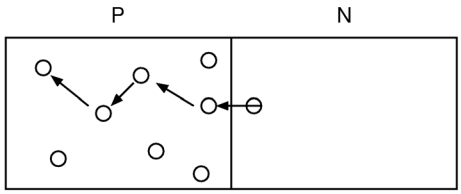


図 3.10: 電子の動き

逆に、逆方向バイアスでは、電子はアノードからカソードへ流れようとするため、N 形半導体の余った電子は正極へ流れ、P 形半導体では負極から供給された電子によってホールを埋めてしまいます。そのため、両方の半導体が電気的に安定した状態となり、PN 接合面で電子のやり取りがなくなり、回路に電流が流れなくなります。なお、図 3.12 のように、逆方向バイアスではホールと電子が両極に移動すると解説されている場合もあります。これは、電子がホールを埋めるとき、あたかもホールが流れているように見えるためです。

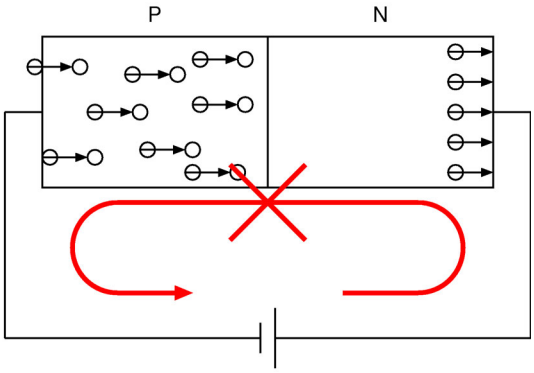


図 3.11: 逆方向バイアス

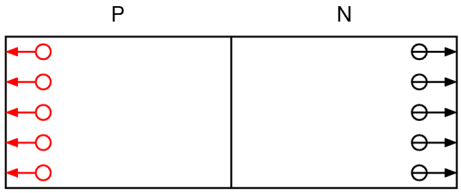


図 3.12: 電子とホール

トランジスタ　トランジスタは、N 形半導体と P 形半導体を図 3.13 のように **NPN 接合**または **PNP 接合**したもので、各端子を**エミッタ・コレクタ・ベース**と呼び、ベースに (入力) 電流を流すことでコレクタとエミッタに増幅した (出力) 電流を流します。図 3.14 は NPN 形トランジスタの回路記号を表し、図 3.15 は PNP 形トランジスタの回路記号を表します。

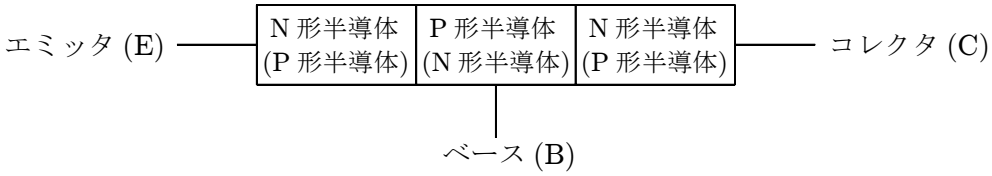


図 3.13: トランジスタの接合図

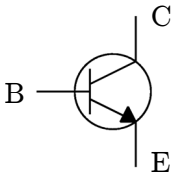


図 3.14: NPN 形トランジスタの回路記号

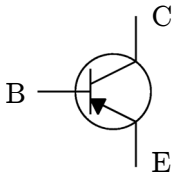


図 3.15: PNP 形トランジスタの回路記号

NPN 形トランジスタの働きについて詳しく見ていきましょう。図 3.16 のようにベースに電流が流れていない場合は、回路右側の PN 接合部がダイオードの逆方向バイアスと同じようになり電流が流れなくなります。逆に、図 3.17 のようにベースに電流が流れている場合は、回路左側の NP 接続部がダイオードの順方向バイアスと同じようになり、エミッタからコレクタへ電流が流れるようになります。このとき、コレクタ—ベース間よりコレクタ—エミッタ間の電圧が高いため、ほとんどの電子は P 形半導体を突き抜けてエミッタに流れてしまいます。なお、エミッタからコレクタへ流れる電流の量はベースへ流す電流に対応して変化しますが、これは、ベースを水道の蛇口に例えると、電流の流量は水量に例えることができます。

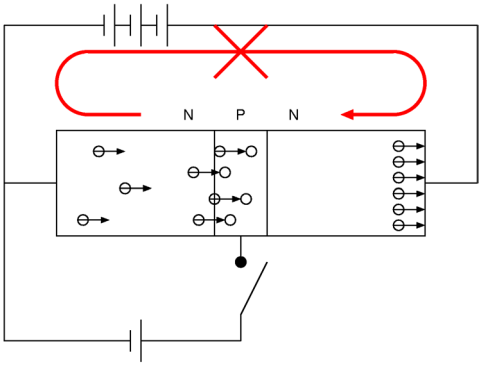


図 3.16: ベース電流を流さない場合

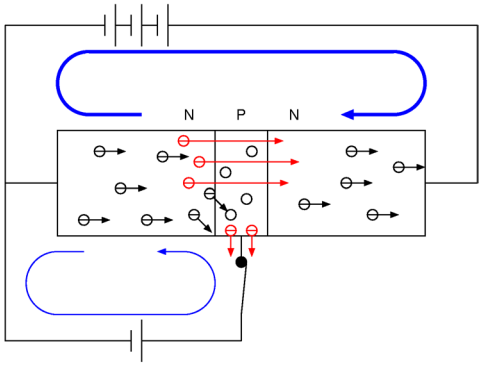


図 3.17: ベース電流を流した場合

3.2　論理素子

論理素子は論理演算を回路で実現したもので、真理値 1 および 0 に対応する 2 つの状態を安定して表せる論理素子が必要となります。現代のコンピュータは半導体によって実現されていますが、半導体が発明されるまでは**電磁リレー**や**真空管**を使用していました。そのような背景から、構成される論理素子の違いによってコンピュータの世代が特徴づけされていて、第 1 世代は真空管、第 2 世代は個別の半導体トランジスタ、第 3 世代は半導体の集積回路 (IC:Integrated Circuit)、第 4 世代は集積度の高い大規模集積回路 (LSI:Large Scale Integrated circuit) に分類されます。現在は第 4 世代にあたります。

半導体による論理素子では、電圧の高い状態 V と低い状態 0 を真理値の 1 と 0 に対応させて考えますが、電圧の高い状態を 1・低い状態を 0 に対応させた**正論理**(positive logic) と、高い状態を 0・低い状態を 1 に対応させた**負論理**(negative logic) があります。このテキストでは全て正論理で解釈します。従って、基本的な論理演算である AND 回路と OR 回路は、2 個のダイオードを使って図 3.18 と図 3.19 のように構成することができます。なお、図中の A と B は入力端子、X は出力端子を表します。

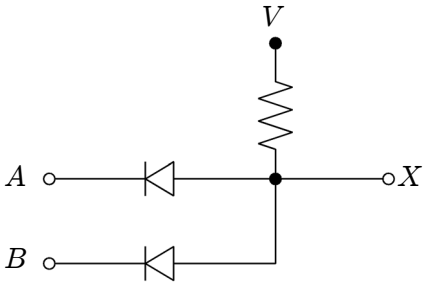


図 3.18: AND 回路

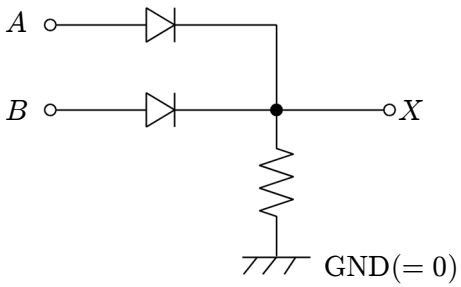


図 3.19: OR 回路

図 3.18 の AND 回路について動作を詳しく見てみましょう。図 3.20 のように、入力端子に低い電圧の状態を含む場合、電流は電圧の低い入力端子に流れるため、出力端子の電圧は 0 になります。逆に、図 3.21 の場合、ダイオードの両端の電圧が等しいため電流は出力端子に流れ、出力端子の電圧は V になります。従って、この回路は論理積の働きをすることがわかります。

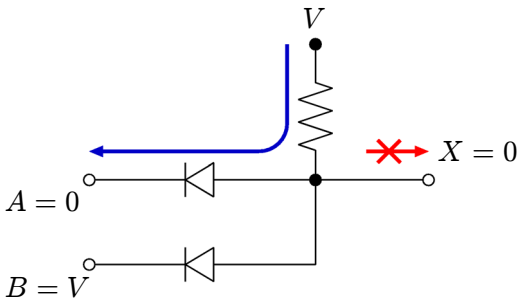


図 3.20: $A = 0$ かつ $B = V$ の場合

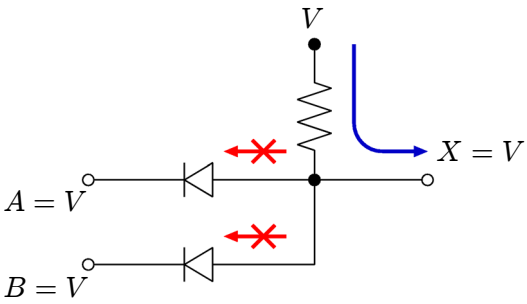


図 3.21: $A = V$ かつ $B = V$ の場合

同様に、図 3.19 の OR 回路は論理和の働きをすることがわかります。

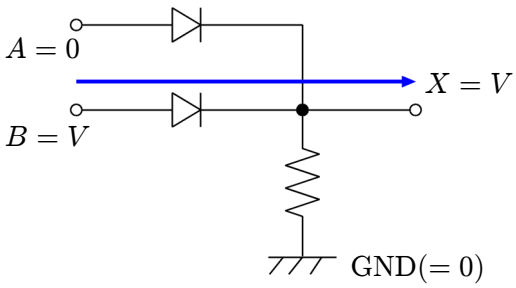


図 3.22: $A = 0$ かつ $B = V$ の場合

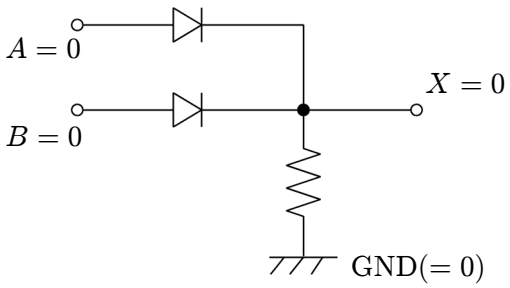


図 3.23: $A = 0$ かつ $B = 0$ の場合

NOT 回路については、トランジスタを使って図 3.24 のように構成することができます。入力端子の電圧が V の場合、図 3.25 のようにトランジスタのベースに電流が流れるため、図の電圧 V によって流れる電流は電圧の低い GND に流れてしまい、出力端子の電圧は 0 となります。逆に、入力端子の電圧が 0 の場合、図 3.26 のようにトランジスタのベースに電流が流れないため、図の電圧 V によって流れる電流は出力端子に流れ、出力端子の電圧は V となります。従って、この回路も否定の働きをすることがわかります。

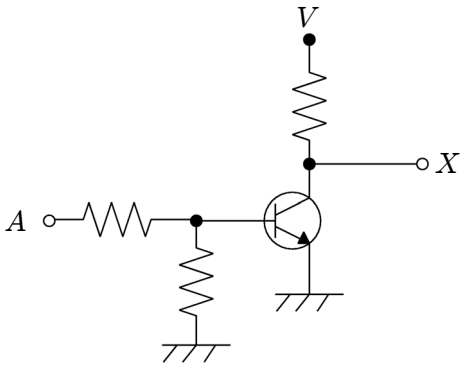


図 3.24: NOT 回路

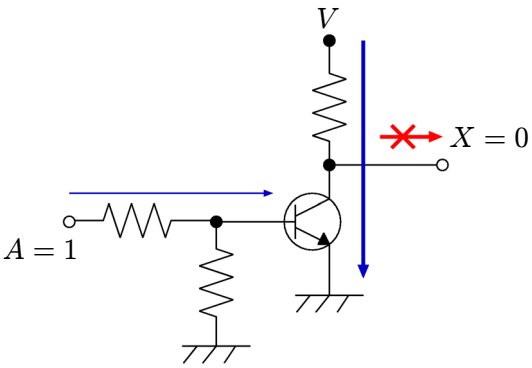


図 3.25: $A = V$ の場合

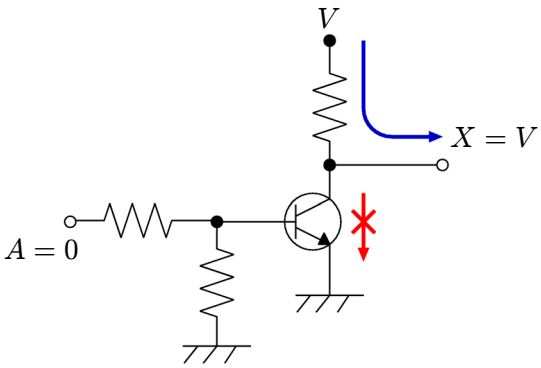


図 3.26: $A = 0$ の場合

以上のことより、3.4 節で述べた 1 つの演算で最小万能演算系となる否定積の論理素子である

NAND 回路は、図 3.27 のように構成することができます。

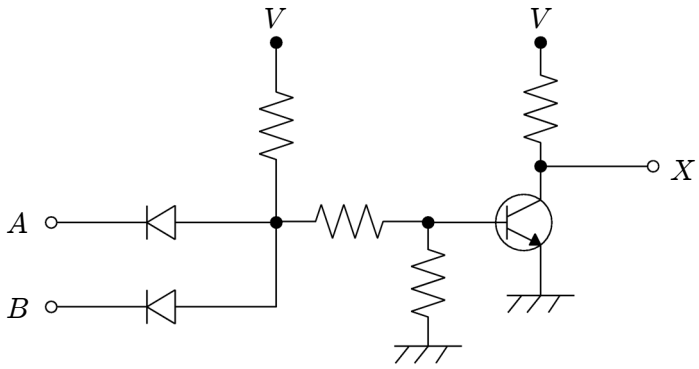


図 3.27: NAND 回路

論理回路を電子部品で書き表すと煩雑となりますので、論理素子を簡略化した **MIL 記号**を導入しましょう。MIL 記号は、アメリカ合衆国の軍隊で使われていたもので、砲弾の弾道を計算するためにコンピュータが発達してきたという歴史的背景から、現在でもこの記号が使われます。図 3.28 は MIL 記号の代表的なもので、左側が入力、右側が出力となります。

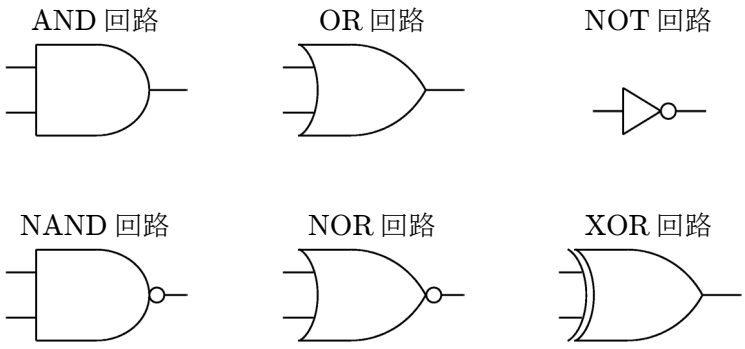


図 3.28: MIL 記号

3.4 節で述べたように、否定論理和によって全ての論理関数を表すことができるので、図 3.29 のように MIL 記号の NAND 回路を用いて基本回路を構成することができます (p.23 の表 2.6 参考)。

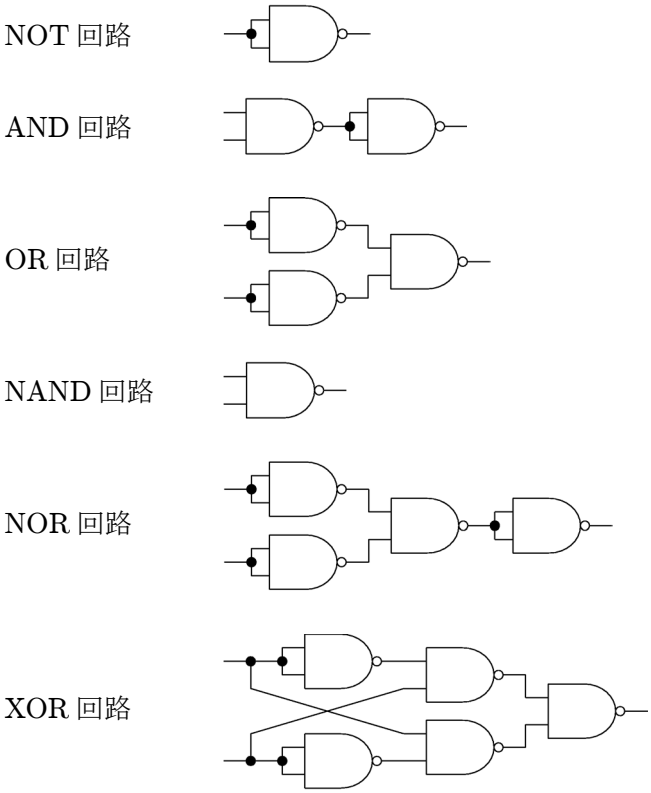


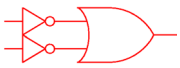
図 3.29: NAND 回路による基本回路の構成

例題 1 MIL 記号の NOT 回路と OR 回路を使って NOR 回路を作りなさい。

解答例 $A \downarrow B = \overline{A + B}$ であるから、

例題 2 MIL 記号の NOT 回路と OR 回路を使って NAND 回路を作りなさい。

解答例 $A \mid B = \overline{A \cdot B} = \overline{A} + \overline{B}$ であるから



● コーヒーブレイク — 世界最初のコンピュータ I —

世界最初のコンピュータは、1942 年にアイオワ州立大学の物理学科教授 アタナソフ (John Vincent Atanasoff) とその助手 (当時大学院生) ベリー (Criford Berry) によって発明された「ABC マシン (Atanasoff-Berry Computer)」です。後述の「ENIAC(エニアック; Electronic Numerical Integrator and Computer)」と長きによる裁判の末、「世界最初のコンピュータ」という地位を勝ち取りました。このマシンは、ディジタル方式・2 進方式の記憶再生装置・電子式制御・論理/算術演算機構・逐次演算処理のアイデアを取り込み、回転ドラム記憶装置にキャパシタを使用しています。

裁判には負けたものの、アメリカ陸軍のアバディーン試射場でモークリー (John W. Mauchly) とエッカート (J. Presper Eckert) を中心に製作された「ENIAC」は、世界最初の実用コンピュータの地位を揺るぎないものとしています (ABC マシンはまともに動かなかったが、ENIAC は弾道計算など実際に利用された)。このマシンは、17,468 本の真空管と 1,500 個のリレーが使用され、床面積 1800 平方フィート・総重量 30 トンという巨大なものでした。····· パーソナルメディア株式会社から出版されている「エニアック –世界最初のコンピュータ開発秘話–」などで詳しく解説されています。

アタナソフ　　ベリー　　ABC マシン

モークリー　　エッカート　　ENIAC

3.3 加算回路

2 章で述べたように、コンピュータの算術演算の基本は加算とシフト演算です。シフト演算の回路を構成するのは簡単なので省略します。ここで重要なのは加算回路で、論理素子を使って構成されます。

最初に、1 桁の加算[1] $S = A + B$ を考えてみましょう。$A = 1$ かつ $B = 1$ のとき繰り上がりが起きることに注意します。繰り上がりの有無を変数 C とし、A と B の組み合わせを全て求めると表 3.1 になります。

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$$\Longleftrightarrow \qquad \begin{aligned} S &= A \oplus B \\ C &= A \cdot B \end{aligned}$$

表 3.1: 1 桁の加算

従って、S と C の関係式が求まるので、図 3.30 のような回路によって 1 桁の加算回路を構成することができます。ただし、任意の桁の加算を行うときには下の桁からの繰り上がりを考慮する必要があり、この回路で実現されるのは繰り上がりだけです。そのため、**半加算器**(half adder, HA) と呼ばれ、図 3.31 のようにブロック図に置き換えて表します。

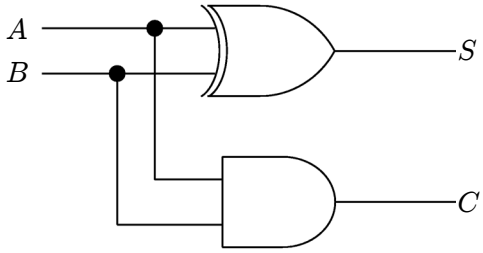


図 3.30: 1 桁の加算回路

図 3.31: 半加算器のブロック図

全加算器(full adder, FA) は 2 つの半加算器を使って作ることができます。半加算器と同じように考えて、i 桁目の全加算器を考えてみましょう。i 桁目の A_i, B_i と $i-1$ 桁目の繰り上がり C_{i-1} の全ての組み合わせから S_i と C_i の値は表 3.2 となります。

[1] 論理和ではないことに注意しなさい。

C_{i-1}	A_i	B_i	S_i	C_i
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$\Longleftrightarrow \quad S_i = (A_i \oplus B_i) \oplus C_{i-1}$$

$$C_i = (A_i \oplus B_i) \cdot C_{i-1} + A_i \cdot B_i$$

表 3.2: i 桁目の全加算

従って、S_i と C_i の関係式が求まるので、図 3.32 のような回路によって i 桁目の全加算器を構成することができます。

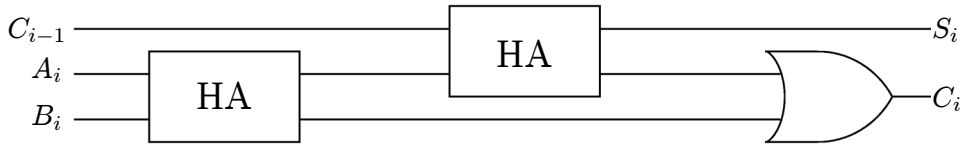


図 3.32: i 桁目の全加算器

また、図 3.33 のようにブロック図に置き換えて表します。全加算器を使って 4 ビットの加算器を作ると図 3.34 のようになります。

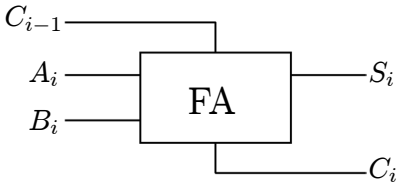


図 3.33: 全加算器のブロック図

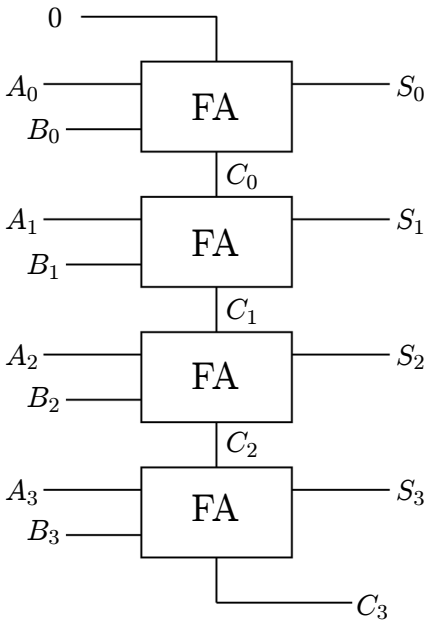


図 3.34: 4 ビットの全加算器

3.4 フリップフロップ

コンピュータを構成する上で、演算回路の他にレジスタやプログラムカウンタなどの記憶回路が必要となります。実際、**フリップフロップ**(flip flop) によって実現されていて、図 3.35 のような回路で実現されます。

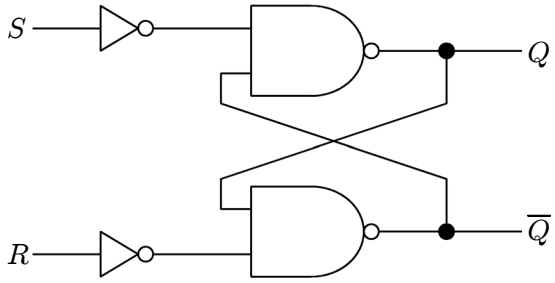


図 3.35: フリップフロップ

図 3.35 の回路は、$S=1$ かつ $R=0$ のとき $Q=1, \overline{Q}=0$ の安定状態となります (この回路は対称なので $S=0$ かつ $R=1$ のとき $Q=0, \overline{Q}=1$ の安定状態になります)。また、$S=1$ かつ $R=0$ の状態から S の値を $1 \to 0 \to 1 \to \cdots$ に変化させても $Q, \overline{Q}$ の値は変化しないことがわかります。逆に、$S=1, R=0$ または $S=0, R=0$ の状態から $S=0, R=1$ の状態にすると $Q=0, \overline{Q}=1$ の状態になり値が変化します。以上をまとめると、表 3.3 のように S と R の状態によって $Q, \overline{Q}$ の値が変化します。ただし、$S=1, R=1$ のときは状態が不安定になるので使いません。このように、フリップフロップ回路は S と R の値によって Q の値を保持や変更が可能になります。

S	R	Q	$\overline{Q}$
0	0	Q	$\overline{Q}$
0	1	0	1
1	0	1	0

表 3.3: R, S による $Q, \overline{Q}$ の状態

実際のコンピュータでは**クロック**(clock) と同期を取って値の変更を行います (図 3.36 参照)。なお、クロック C は周期的に 1 と 0 の状態を交互に作り出します。

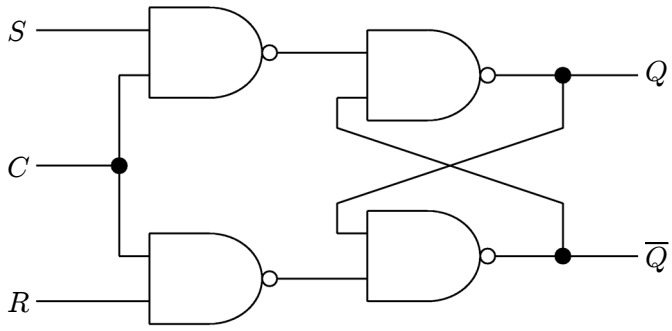


図 3.36: クロックを用いたフリップフロップ

● コーヒーブレイク — 世界最初のコンピュータ II —

ドイツの建築技師 コンラッド・ツーゼ (Konrad Zuse) は、1934 年から自動計算機の開発に取り掛かり、1941 年までに Z1・Z2・Z3 の 3 台のコンピュータを製作しました。しかしながら、空襲によって機械そのものは破壊されてしまったため、世界最初のコンピュータと認められませんでした (写真までは残っている)。従って、コンピュータ誕生の歴史は以下のようになります。

1941 年　Z3, 世界最初の電子機械式コンピュータ
1942 年　ABC マシン, 世界最初のコンピュータ
1943 年　Harvard Mark I, 世界初のリレー式コンピュータ
1946 年　ENIAC, 世界最初の実用コンピュータ
1951 年　UNIVAC I, 世界最初の商用コンピュータ
1952 年　プリンストン (フォン・ノイマン) 計算機, 世界初のプログラム内臓方式コンピュータ
1964 年　IBM 360/40, 世界最初の汎用コンピュータ

コンラッド・ツーゼ

Z3

第4章　コンピュータの仕組

4.1　コンピュータシステム

コンピュータは、図 4.1 のように外部から手続き指示書である**プログラム**と材料である**データ**を受け取り、プログラムに従ってデータ処理した結果を外部に返します。この様な一連の流れによるものを**コンピュータシステム**と呼びます。

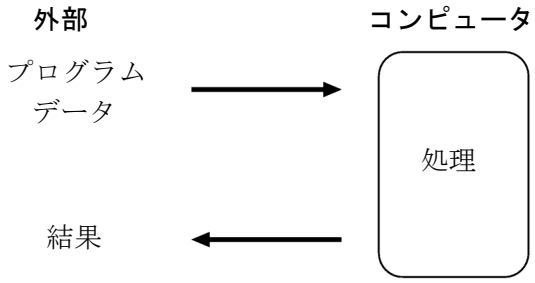


図 4.1: コンピュータシステム

コンピュータシステムを実現するために、コンピュータは**入力装置**・**出力装置**・**制御装置**・**演算装置**・**記憶装置**の 5 大要素から成っています。また、制御装置と演算装置をまとめて**中央処理装置**(CPU:Central Processing Unit) と呼びます。

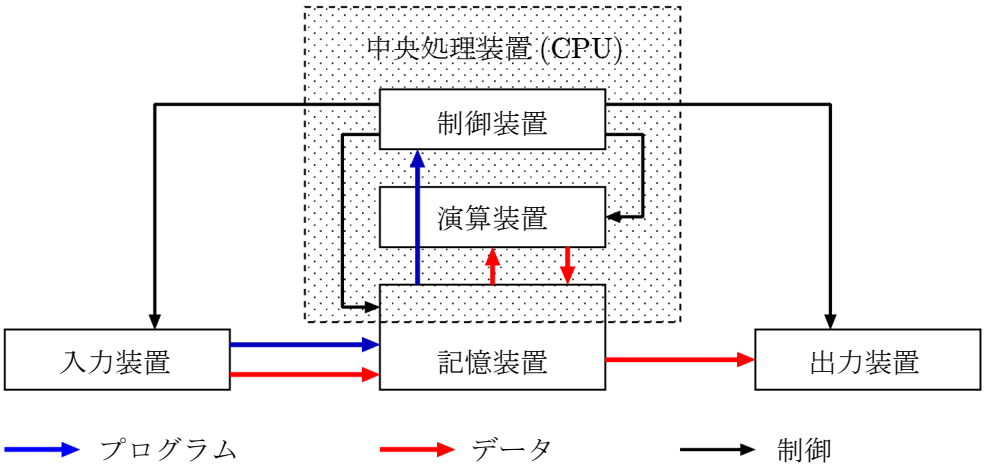


図 4.2: 5 大構成要素

入力装置　外部からコンピュータにプログラムやデータを入力する装置で、キーボードやマウスなどがあります。

出力装置　コンピュータから外部に結果を出力する装置で、ディスプレイやプリンタなどがあります。

記憶装置　プログラムやデータを記憶する装置で、**主記憶装置**(**メインメモリ**) を指しますが、広義で**補助記憶装置**を含める場合もあります。その他にも、CPU 内部で演算に使用される**レジスタ**やレジスタとメインメモリのアクセスタイムを縮めるための**キャッシュメモリ**などがあります。なお、記憶装置には一連の番号である**アドレス**が割り振られています。

制御装置　主記憶装置からプログラムを読み込み、命令を解読して他の 4 つの装置に対して指示を与えます。

演算装置　主記憶装置からレジスタにデータを読み込み、制御装置の指示に従って演算を行います。

コンピュータが直接理解できるのは**機械語**で、今でも組み込み式コンピュータなど限られた部分ではありますが使用されています。また、コンピュータが登場した当時はこの言語を使ってプログラムを作成していましたが、この言語は直接コンピュータの資源を利用するためプログラムの追加・削除・変更が容易ではありませんでした。その後、機械語と 1 対 1 に対応した**アセンブラ言語**が登場し、プログラムを実行する直前に機械語に変換したので、機械語の欠点が改善されました。しかしながら、この言語も中央処理装置の違いによってアセンブラ言語が異なっていて汎用性に富んでいませんでした。その後、汎用性に富んだ言語が次々に現れることになります。

情報処理試験で用いられる仮想のコンピュータを例に挙げ、コンピュータの計算の仕組みを詳しく見ていきましょう。COMET と呼ばれる 5 大要素を備えた仮想のコンピュータとその上で動く CASL と呼ばれるアセンブラ言語を用います。ノイマン型のコンピュータの制御方式は**逐次制御**で「**フェッチ**(命令の取り出し) $\Longrightarrow$ **デコード**(命令の解読)$\Longrightarrow$ 実行」を順番に繰り返すことによって動作しています。これらを実現するためにコンピュータの基本構成は下図のようになっています。

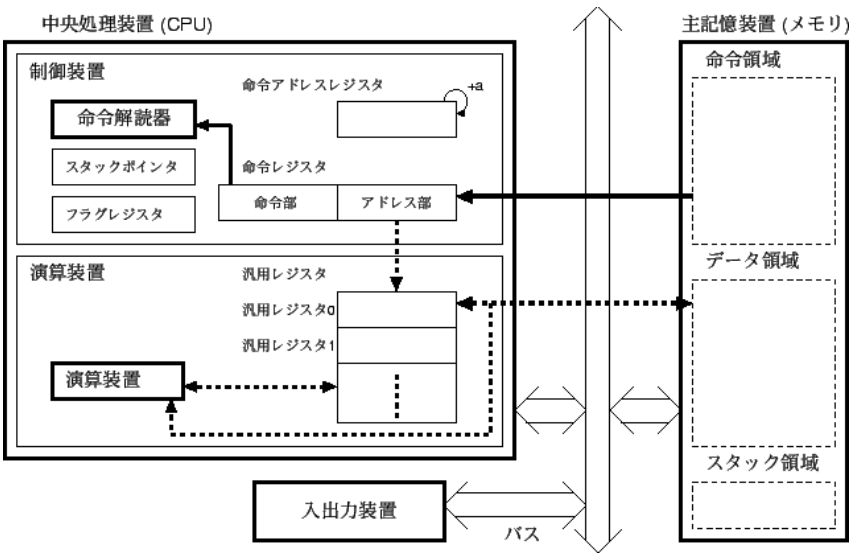


図 4.3: 計算機内部の模式図

主記憶装置 主記憶装置はノイマン型のコンピュータの特徴で、命令とデータを記憶するところです。主に命令領域とデータ領域に分れますが、その他にスタック領域として使用します。COMET II では 1 語は 16 ビット (2 進数で 16 桁) で表され、アドレスは 16 ビット (0000(16) から FFFF(16)) で表現できる 65536 語を記憶します。

命令解読器 命令解読器は制御装置の中心的存在で命令レジスタの命令部を解読し、各装置に命令を出します。命令デコーダとも呼ばれます。

演算装置 演算装置では記憶装置のデータや汎用レジスタの値を用いて計算を行います。また、命令レジスタのアドレスの計算なども行います。

命令アドレスレジスタ 命令アドレスレジスタは次にどこからプログラムを進めれば良いのか位置 (アドレス) を記憶しています。命令の語の長さによって加数は変化します。

命令レジスタ 命令レジスタは命令を記憶する場所で命令部とアドレス部に分れています。また、命令部は命令解読器に送られアドレス部は演算装置に送られます。

汎用レジスタ 汎用レジスタは演算装置に処理するデータや処理されたデータが一時的に記憶される場所です。

スタックポインタ 記憶装置にスタック領域が確保され、スタックポインタはその先頭アドレスを記憶しています。また、スタックには最後に入れたものを最初に取り出す (LIFO:Last In First Out) 機能があります。

フラグレジスタ フラグレジスタはオーバーフロー (OF), サイン (SF), ゼロ (ZF) フラグで構成され、演算装置による演算結果により値を変えます。

バス バスは CPU とメモリに引かれた道路で命令やデータが流れます。

コンピュータのプログラム実行は次の手順によって実行されていきます。

step 0 コンピュータでプログラムが実行されるとき、最初に命令アドレスレジスタの値が 0 に初期化されます。

step 1(フェチ) 0 番地の命令が命令レジスタに読み込まます。

step 2(デコード) 命令解読器に送られ命令の解読が行われます。 その際、命令レジスタの語長によって命令アドレスレジスタの値を次に実行する命令アドレスまで進めておきます。

step 3(実行) 命令によって各装置を制御します。

step 4 step 1 に戻り命令アドレスレジスタが示す命令をフェチします (命令アドレスレジスタは次に実行する命令のアドレスを示しています)。

● コーヒーブレイク — 世界最初のマイクロ・プロセッサ —

世界最初のマイクロ・プロセッサ Intel 4004 は、日本技術者の嶋 正利氏 (当時米国インテル社) によって開発されました。Intel 4004 は、4 ビット CPU(4 ビットの汎用レジスタ) で、4 キロバイトのメモリを持ち、演算回路にはトランジスタ 2,250 個が集積されていました。

嶋氏が Intel 4004 を開発したころは、ミニコンピュータでも 16 ビットの時代だったので、その評価も「嶋はアメリカに行っておもちゃを作ってきやがった」と言われたほど評価されていませんでした。が、「次世代マイクロ・プロセッサ (日本経済新聞社)」の中で「マイクロ・プロセッサ・パソコン・ワークステーションの発明や開発は、古い権威主義的な既存システムから自由を個人に取り戻すための破壊と改革、そして新たな価値を持った新システムの建設と価格破壊であったとも言える」とも言っています。

嶋氏は、1943 年静岡県静岡市に生まれ、実家は洋品店で、生まれ育った所は繁華街で、弓屋やダンスホールがすぐ隣にありました。また、半径 200 メートル以内に映画館が 6 件ぐらいあり、後にこれがマイクロ・プロセッサのレイアウトを決めるのに役立った、と言っています。1967 年には東北大学理学部第 2 化学科を卒業して、ビジコン社に入り、キーボードやプリンタなどのリアルタイム制御するためのチップを作るところから、ソフトウェア的に制御するチップを作ることになった、とその開発動機を語っています。

彼の功績により、コンピュータの高速化と小型化が進み、今日の爆発的なコンピュータ社会を導いたことは、世界にも認められています。

世界最初のマイクロ・プロセッサ Intel 4004　　嶋 正利

株式会社 ASCII「計算機屋かく戦えり」より抜粋

4.2 コンピュータの設計

本テキストの総まとめとして、簡単な計算機 (コンピュータ) の設計図を作成しましょう。

仕様書

① 4 ビットの計算機 (4 ビットレジスタを持つ)。
② 入力・出力は 2 進数の補数表示で行う。
③ 足し算と引き算のボタンを持つ。
　例 1 :「(0)+X + Y」を計算したい場合は「(0)X + Y +」の順で入力する。
　例 2 : 最初の入力は「(0)X +」で入力する。
④ 足し算「+」と引き算「−」のボタンを持つ。
⑤ レジスタの内容を「0」にセットするクリアボタン「C」を持つ。
＊レジスタに使われるフリップフロップは十分に反応速度が速いものとし、「+」「−」「C」ボタンを押すと直ちに安定状態になるものとします。

作成例を次のページに挙げておきます。

● コーヒーブレイク — 国産コンピュータの生みの親 —

富士通のコンピュータ事業を最初から引っ張った男が、池田 敏雄でした。リレー計算機「FACOM100」に始まり独創的な内容を持ったコンピュータを開発し、国産コンピュータ業界に多大な影響を与えました。

1959 年に社長に就任した岡田 完二郎が、コンピュータに社運をかけるとまで言って、池田らに、日本で最初の IC を使用した大ヒットの大型コンピュータ FACOM 230-60 と汎用 OS を含むソフトウェアの開発を任せました。なお、この大型コンピュータは日本を代表するコンピュータとなり、富士通のコンピュータ物語は、全て、この池田を中心に展開されています。なお、今でも、富士通沼津工場内の池田 敏雄記念館には、そのころの開発の様子と、彼のアイデアを記述したノートが展示されています。

池田 敏雄

＊上記の内容は、NHK で放送されたプロジェクト X の「国産コンピューター ゼロからの大逆転」で詳しく紹介されています。

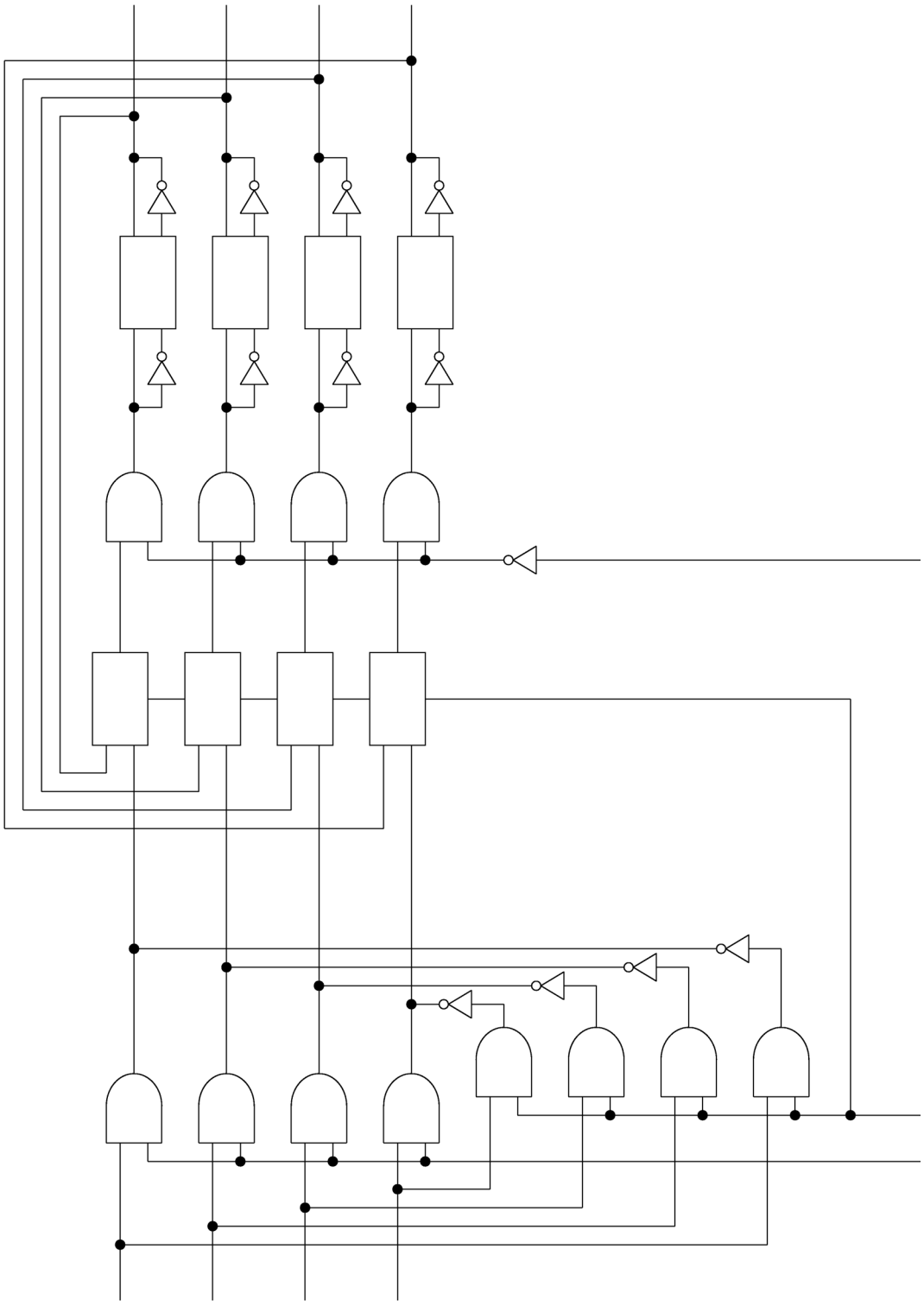


図 4.4: 4 ビット計算機

付 録A　10進数・2進数・8進数・16進数の対応

10 進	2 進	8 進	16 進
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10
17	10001	21	11
18	10010	22	12
19	10011	23	13
20	10100	24	14
21	10101	25	15
22	10110	26	16
23	10111	27	17
24	11000	30	18
25	11001	31	19
26	11010	32	1A
27	11011	33	1B
28	11100	34	1C
29	11101	35	1D
30	11110	36	1E
31	11111	37	1F
32	100000	40	20
33	100001	41	21
34	100010	42	22
35	100011	43	23
36	100100	44	24
37	100101	45	25
38	100110	46	26
39	100111	47	27

10 進	2 進	8 進	16 進
40	101000	50	28
41	101001	51	29
42	101010	52	2A
43	101011	53	2B
44	101100	54	2C
45	101101	55	2D
46	101110	56	2E
47	101111	57	2F
48	110000	60	30
49	110001	61	31
50	110010	62	32
51	110011	63	33
52	110100	64	34
53	110101	65	35
54	110110	66	36
55	110111	67	37
56	111000	70	38
57	111001	71	39
58	111010	72	3A
59	111011	73	3B
60	111100	74	3C
61	111101	75	3D
62	111110	76	3E
63	111111	77	3F
64	1000000	100	40
65	1000001	101	41
66	1000010	102	42
67	1000011	103	43
68	1000100	104	44
69	1000101	105	45
70	1000110	106	46
71	1000111	107	47
72	1001000	110	48
73	1001001	111	49
74	1001010	112	4A
75	1001011	113	4B
76	1001100	114	4C
77	1001101	115	4D
78	1001110	116	4E
79	1001111	117	4F

10 進	2 進	8 進	16 進
80	1010000	120	50
81	1010001	121	51
82	1010010	122	52
83	1010011	123	53
84	1010100	124	54
85	1010101	125	55
86	1010110	126	56
87	1010111	127	57
88	1011000	130	58
89	1011001	131	59
90	1011010	132	5A
91	1011011	133	5B
92	1011100	134	5C
93	1011101	135	5D
94	1011110	136	5E
95	1011111	137	5F
96	1100000	140	60
97	1100001	141	61
98	1100010	142	62
99	1100011	143	63
100	1100100	144	64
101	1100101	145	65
102	1100110	146	66
103	1100111	147	67
104	1101000	150	68
105	1101001	151	69
106	1101010	152	6A
107	1101011	153	6B
108	1101100	154	6C
109	1101101	155	6D
110	1101110	156	6E
111	1101111	157	6F
112	1110000	160	70
113	1110001	161	71
114	1110010	162	72
115	1110011	163	73
116	1110100	164	74
117	1110101	165	75
118	1110110	166	76
119	1110111	167	77

10 進	2 進	8 進	16 進
120	1111000	170	78
121	1111001	171	79
122	1111010	172	7A
123	1111011	173	7B
124	1111100	174	7C
125	1111101	175	7D
126	1111110	176	7E
127	1111111	177	7F
128	10000000	200	80
129	10000001	201	81
130	10000010	202	82
131	10000011	203	83
132	10000100	204	84
133	10000101	205	85
134	10000110	206	86
135	10000111	207	87
136	10001000	210	88
137	10001001	211	89
138	10001010	212	8A
139	10001011	213	8B
140	10001100	214	8C
141	10001101	215	8D
142	10001110	216	8E
143	10001111	217	8F
144	10010000	220	90
145	10010001	221	91
146	10010010	222	92
147	10010011	223	93
148	10010100	224	94
149	10010101	225	95
150	10010110	226	96
151	10010111	227	97
152	10011000	230	98
153	10011001	231	99
154	10011010	232	9A
155	10011011	233	9B
156	10011100	234	9C
157	10011101	235	9D
158	10011110	236	9E
159	10011111	237	9F
160	10100000	240	A0
161	10100001	241	A1
162	10100010	242	A2
163	10100011	243	A3
164	10100100	244	A4
165	10100101	245	A5
166	10100110	246	A6
167	10100111	247	A7
168	10101000	250	A8
169	10101001	251	A9
170	10101010	252	AA
171	10101011	253	AB
172	10101100	254	AC
173	10101101	255	AD
174	10101110	256	AE
175	10101111	257	AF
176	10110000	260	B0
177	10110001	261	B1
178	10110010	262	B2
179	10110011	263	B3
180	10110100	264	B4
181	10110101	265	B5
182	10110110	266	B6
183	10110111	267	B7
184	10111000	270	B8
185	10111001	271	B9
186	10111010	272	BA
187	10111011	273	BB
188	10111100	274	BC
189	10111101	275	BD
190	10111110	276	BE
191	10111111	277	BF
192	11000000	300	C0
193	11000001	301	C1
194	11000010	302	C2
195	11000011	303	C3
196	11000100	304	C4
197	11000101	305	C5
198	11000110	306	C6
199	11000111	307	C7
200	11001000	310	C8
201	11001001	311	C9
202	11001010	312	CA
203	11001011	313	CB
204	11001100	314	CC
205	11001101	315	CD
206	11001110	316	CE
207	11001111	317	CF
208	11010000	320	D0
209	11010001	321	D1
210	11010010	322	D2
211	11010011	323	D3
212	11010100	324	D4
213	11010101	325	D5
214	11010110	326	D6
215	11010111	327	D7
216	11011000	330	D8
217	11011001	331	D9
218	11011010	332	DA
219	11011011	333	DB
220	11011100	334	DC
221	11011101	335	DD
222	11011110	336	DE
223	11011111	337	DF
224	11100000	340	E0
225	11100001	341	E1
226	11100010	342	E2
227	11100011	343	E3
228	11100100	344	E4
229	11100101	345	E5
230	11100110	346	E6
231	11100111	347	E7
232	11101000	350	E8
233	11101001	351	E9
234	11101010	352	EA
235	11101011	353	EB
236	11101100	354	EC
237	11101101	355	ED
238	11101110	356	EE
239	11101111	357	EF
240	11110000	360	F0
241	11110001	361	F1
242	11110010	362	F2
243	11110011	363	F3
244	11110100	364	F4
245	11110101	365	F5
246	11110110	366	F6
247	11110111	367	F7
248	11111000	370	F8
249	11111001	371	F9
250	11111010	372	FA
251	11111011	373	FB
252	11111100	374	FC
253	11111101	375	FD
254	11111110	376	FE
255	11111111	377	FF

付 録B　16進数の掛け算表

	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2		4	6	8	A	C	E	10	12	14	16	18	1A	1C	1E
3			9	C	F	12	15	18	1B	1E	21	24	27	2A	2D
4				10	14	18	1C	20	24	28	2C	30	34	38	3C
5					19	1E	23	28	2D	32	37	3C	41	46	4B
6						24	2A	30	36	3C	42	48	4E	54	5A
7							31	38	3F	46	4D	54	5B	62	69
8								40	48	50	58	60	68	70	78
9									51	5A	63	6C	75	7E	87
A										64	6E	78	82	8C	96
B											79	84	8F	9A	A5
C												90	9C	A8	B4
D													A9	B6	C3
E														C4	D2
F															E1

索 引

本テキストに関するご意見・ご質問・ご指摘等は、返信用の電子メールアドレスを明記の上、下記の著者電子メールアドレスへお願いします。なお、本テキストの内容を逸脱すると思われるお問い合わせや、筆者のスキルの及ばない範囲のお問い合わせについては、お答えしかねますのでご了承ください。

「計算機理論入門」 ～ コンピュータを設計しよう ～

2003 年 8 月 21 日　第 1 版

著者： 幸山 直人 (こうやま なおと)
電子メールアドレス： nkouyama@sci.toyama-u.ac.jp
ホームページアドレス： http://kouyama.math.toyama-u.ac.jp/main/

発行： 富山大学 理学部 数学教室
ホームページアドレス： http://www.math.toyama-u.ac.jp
