研究ノート

コンピュータのホワイトボックス化を目指した文系向け情報教育について

――科目「情報処理論」を事例に――

河　村　一　樹

東京国際大学論叢　人間科学・複合領域研究　第 10 号　抜刷
2025 年（令和 7 年）3 月 20 日

研究ノート

コンピュータのホワイトボックス化を目指した文系向け情報教育について

――科目「情報処理論」を事例に――

河　村　一　樹

Information Education for the Humanities Aimed at White Boxing Computers

—Case Study of the Subject “Information Processing Theory”—

KAWAMURA, Kazuki

Abstract

The author has been responsible for the course “Information Processing Theory” in the Faculty of Commerce for many years. The students enrolled in this course are primarily humanities majors, and many of them have an aversion to mathematics. However, the field of information processing is rooted in computer science, which is fundamentally based on discrete mathematics. Therefore, a teaching strategy that minimizes the use of complex mathematical formulas is essential for students who are not fond of mathematics. In this paper, I discuss my approach to teaching this subject, as illustrated by the slides used in the “Information Processing Theory” course.

目　次

はじめに

筆者は，長年，商学部において情報処理教育を担当してきた。具体的には，情報処理に関する講義科目とプログラミングの実習科目である。情報処理の講義科目としては，「情報処理論」を担当している。情報処理というと，離散数学をベースとしたコンピュータサイエンスという学問領域に位置づけられる。このため，どちらかというと数理系分野に位置づけられる。一方，「情報処理論」を受講する学生の多くは，数学嫌いである。

このギャップを埋めるための指導方略について，自分なりに工夫を重ねてきた。その一つとして，数式を多用することなく，できるだけ視覚的にイメージできるような図解によるスライドを作成した。それらを，Moodle にアップロードすることによって，ディジタル教科書としても利用できるようにした。

このため，講義では，板書は一切せず，基礎理論的なことはスライドを，最新技術の話題は Web サイトあるいは YouTube や DVD の動画を，それぞれプロジェクタに投影しながら説明を行うようにした。その際に，学生は自分のパソコンやスマフォを持参して視聴してもよいとした。

本稿では，「情報処理論」で用いたスライドをもとに，文系の学生に対して，コンピュータサイエンスの基礎的な内容をどのように教授すべきか，その指導方略について報告する。

1. 文系学部における情報教育の課題

本学は，各学部の教育目標などから，文系の大学に位置づけられている。文系と理系の差異を表す指標としては，学生が数学嫌いか好きかといったことが挙げられる。文系の学生については一般的に数学が嫌いであり，理系の学生については数学が好きであるという傾向にある [1][2]。このような状況にあって，最近では文系の学生に対しても数理教育が求められるようになった。その上で，文系の学生の数学嫌いの要因の一つとして考えられる小学校の算数教育について取り上げる。

1.1 MDASHによる影響

MDASH（Mathematics Data science and AI Smart Higher education）[3]とは，文部科学省が認定する「数理・データサイエンス・AI教育プログラム認定制度」のことである。これまでの「読み・書き・そろばん」といったリテラシーに対して，これからは「数理・データサイエンス・AI」に関する知識・スキルが必要になるという目論見が背景にあるといえる。

MDASHには，学生の数理・データサイエンス・AIへの関心を高め，適切に理解し活用する基本的な能力を育成する「リテラシーレベル」と，数理・データサイエンス・AIを活用して課題を解決するための実践的な能力を育成する「応用基礎レベル」がある。

いずれのレベルにおいても，統計および数理基礎，アルゴリズム基礎，データ構造とプログラミング基礎といった科目を選択として設置する必要がある。このように，数学基礎および統計基礎を学ぶだけでなく，離散数学の領域であるアルゴリズムとデータ構造，プログラミングなども含まれる。

MDASHの認定対象については，理系だけでなく文系の学部・学科も含まれる。このため，文系学生に対しても，数理科学を教えなければならなくなり，指導方略において様々な工夫が求められることになった。

本学では，商学部において，グローバルデータサイエンスコースおよびデータサイエンスコースを新設した。グローバルデータサイエンスコースでは，ビジネス応用に特化したディジタル経営学を学ぶコースで，英語教育とデータサイエンスの基礎教育を行う。データサイエンスコースでは，おもにデータサイエンスの基礎教育を行う。

各コースで開講している科目には，基礎数学，確率・統計，情報学基礎，プログラミング，データサイエンスプログラミング，Rプログラミング，ウェブマイニングなどがある。このように，数学や情報学寄りの科目が多く用意されているのがわかる。

1.2 小学校の算数教育について

大学生に対して，算数や数学が不得意であることを示した著書がいくつか出版されている[4]-[6]。この中の分数ができない大学生に着目すると，文系の学生が数学嫌いになった理由の一つに小学校の算数教育の弊害があげられる。具体的には，プロセスを無視した算数教育が横行しており，テクニカルなやり方だけを暗記して答えを求めさせるといった教育が行われていることである。

例として，分数の割り算についてみてみる。小学校の算数で割り算を学び始める場合，実際の事象を見せてイメージさせることが多い。例えば，1/2という数式の意味については，リンゴ1個を半分に切った部位（1個÷2）であり，実物を用いて視覚的に理解させることができる。

一方，分数の割り算となると，イメージがしにくくなり理解しづらくなる。例えば，リンゴ1個を二分の一等分（1個÷1/2）するとなると，1個のリンゴを半分に切りその半分が2つ存在するということになる。つまり，1個のリンゴは1/2サイズのリンゴ2つ分に相当するという意味である。しかし，これでは実物で表すことができなくなり，理解できない子供も増えてくる。

そこで，「分数の割り算では，割る数の分母と分子を逆にして掛ければよい」という計算のテクニックだけを教えることに終始することになる。しかし，その理由については触れることなく，後は分数の割り算に関する計算問題を解かせて終わる。

分数の割り算において，なぜ分母と分子を逆にして掛けるのか，その理由については次のように説明できる。

割り算では，2つの数字（[割られる数]÷[割る数]）を同じ割合で大きさを変えても答えは変わらない。例えば，次のようになる。

$$
\begin{aligned}
&43 \div 0.5 \\
&= (43 \times 10) \div (0.5 \times 10) \\
&= 430 \div 5
\end{aligned}
$$

という形で，等号が成立する。

次に，分数の割り算についてみてみる。

$$43 \div 1/2$$

において，[割る数]を1にするためには，2を掛ければよいことがわかる。ただし，式の等号を成立させるためには，[割る数]だけでなく[割られる数]にも同じ値を掛ける必要があるので，

$$
\begin{aligned}
&= (43 \times 2) \div (1/2 \times 2) \\
&= 86 \div 1 \\
&= 86
\end{aligned}
$$

と答えを得ることができる。

この[割る数]を1にする計算こそが，[割る数]の分母と分子を逆にしてかけることを意味しているわけで，

$$
\begin{aligned}
&5/8 \div 3/4 \\
&= (5/8 \times 4/3) \div (3/4 \times 4/3) \quad ※ \quad ① \\
&= (5 \times 4)/(8 \times 3) \div 1 \\
&= (5 \times 1)/(2 \times 3) \quad ※ \quad 約分して \\
&= 5/6
\end{aligned}
$$

となり，①の$(5/8 \times \underline{4/3})$の4/3が相当する。

以上によって，「[割る数]の分母と分子を逆にして掛ける」ことは，[割る数]をどのように変形していけばよいのかという過程を表しているわけである。このように，計算問題は，計算しやすくなるように式を変形させることが最大のコツとなる。このコツを理解すれば，式の変換を規則的に覚える必要はなくなるわけである。小学校の算数教育において，このような取り組みをしないと，算数・数学嫌いが増え続ける可能性がある。

2. 科目「情報処理論」について

筆者は，2001年度から現在まで，商学部において「情報処理論」（以降，本科目と略す）という科目を担当している。この科目を履修している文系の学生達のほとんどは，程度の差はあるものの数学嫌いの傾向にある。これについては，ガイダンス時に実施する事前アンケートなどで明らかといえる。

一方，情報処理のカリキュラムは，離散数学をベースとしたコンピュータサイエンスの基礎理論をベースに構成している[7]。具体的には，符号理論，組み合わせ論，グラフ理論，確率統計，オートマトン，形式言語理論，プログラム理論，計算量理論，検証理論，アルゴリズムとデータ構造などが含まれる[8]。情報専門学科であれば，これらの科目を網羅したカリキュラムが開講されているが，経営学科のように情報を利用する立場にある学科では，これらの中からミニマムエッセンシャルな部分を取り上げる程度になるとともに，数式を多用することなく講義を行う必

要がある。

これらのことを考慮しながら，本科目の講義を進めてきた。ここでは，本科目の全体像と実施形態について取り上げる。

2.1　本科目の概要

（1）カリキュラムの位置づけ

本科目は，商学部経営学科の専門科目として，[学科内専門共通科目]に分類されている。これより，学科共通の専門的な科目に位置づけられており，情報処理に関する専門的な内容まで，ある程度扱う必要がある。

一方，学部共通の教養科目としては，「情報処理の基礎」という科目がある。これは，[全学部][基礎教育科目][教養コア科目][自然科学と環境]に分類されている。この分類からもわかるように，一般教育における人文・社会・自然の自然分野に位置づけられており，情報の基礎的な内容を網羅した形になっている。

いずれも科目ナンバリングは100番台であるが，履修モデルとしては「情報処理の基礎」を受講した上で「情報処理論」を履修することが推奨される。

（2）シラバスについて

本学では，全科目のシラバスを，ポータルサイトPOTI（PORTAL of TIU Internal Web）で公開している[9]。

本科目における到達目標（授業の狙い）については，次のように記述している。

> 『ICTの中核に位置づけられるコンピュータを，ブラックボックスとするのではなく，グレーボックス，さらには，ホワイトボックスとして扱えるようにする。また，ICTの健全な利用者（原理や仕組みを理解した上で，操作する者）になれることを目指す。』

この中の『コンピュータ』は，メインフレームコンピュータではなく，パーソナルコンピュータ（以下，PCと略す）を想定している。

『コンピュータをブラックボックスとして捉える』については，コンピュータの内部構造や動作原理について理解することによって実現できるといえる。そのためには，コンピュータサイエンスの基礎的な内容のいくつかを学ぶ必要がある。具体的には，符号理論としての2元符号化・論理演算，計算理論としてのチューリング機械，プログラム理論としての形式言語理論，アーキテクチャー論としてのプログラム実行制御などがあげられる。ただし，前述した数学嫌いが多い文系学生に対して講義する際には，数式の多様を避けるとともに，計算は四則演算だけにとどめるといった工夫が必要になってくる。

『健全な利用者』とは，自分の問題解決の過程において，コンピュータを操作する際に，統制感を持つことができる利用者のことを意味している[10]。この統制感とは，コンピュータを道具として使うために，内部の概念モデルを獲得しているとともに，自己の統制の下にコンピュータを動かしているという感覚のことである。統制感を持つことによって，自分がコンピュータに対して操作した際に，内部でどのようなことが起こっているのかがイメージできるようになる。また，コンピュータに何らかの障害が生じた場合は，その原因を特定できるだけでなく，その復旧方法まで導き出すこともできるようになる。

本科目におけるコースアウトラインについては，**表1**のようになる。

表 1　本科目のコースアウトライン

回数	コースアウトライン	コンピュータサイエンス基礎
第1回	科目ガイダンス（シラバス，平常点の扱い，評価）	
第2回	汎用コンピュータ，ミニコンピュータ，サーバーコンピュータの歴史	ノイマン型コンピュータ
第3回	パーソナルコンピュータ（Windows PC，Macintosh，UNIX/Linux PC）の歴史	
第4回	タブレット端末，スマートフォン，ウェアラブルコンピュータの歴史	
第5回	汎用コンピュータ，パソコン，スマフォのオペレーティングシステムの歴史	
第6回	日本語FEP（仮名漢字変換），ワープロ，表計算，データベースソフト	
第7回	プログラミング言語，言語処理系（コンパイラ，インタプリタ）	形式言語理論，プログラム理論
第8回	コンピュータが扱う情報（2/10/16進数，進数同士の相互変換，2進数の四則演算，補数）	2元符号化，論理回路，論理演算
第9回	各種文字コード（ANSII/JIS/SJIS/UTF/EUC/Unicode）	
第10回	マルチメディアデータ（画像，動画，音声/音楽）の扱い	
第11回	コンピュータの動作原理	チューリング機械，万能チューリング機械，プログラムの実行制御
第12回	パソコンのCPU，メインメモリ，キャッシュ，バス，補助メモリ（HDD，SSD，USB）	
第13回	パソコンの入力装置（キーボード，ポインティングデバイス），出力装置（ディスプレイ，プリンタ），通信装置（ルーター，WiFi），周辺機器	
第14回	これからのコンピュータ（シンギュラリティ，AI，量子コンピュータ）	

2.2　本科目の実施形態

当初は，市販の教科書[11]を指定するとともに，板書主体の講義を実施していた。また，毎回，B5版の紙カードを配布し，授業の概要を手書きで書いて提出させていた。その後，学内のICT環境が整備拡充され，2016年度からは全学レベルでLMSの一つであるMoodleが導入された。

そこで，Moodleを用いた授業様式に全面的に改変することにした。これに合わせて，教室に自分のPC/タブレット端末/スマートフォンといったデバイスを持ち込んで，自由に操作してもよいとした。このため，ICTを使いこなす学生によっては，Moodleの教材コンテンツをディジタル教科書として自分のスマートフォンなどで閲覧したり，授業の概要を事前に入力したりしていた。

Moodleの利用については，次のようになる。

（1）スライド：Moodle[リソース][ファイル]

筆者がPowerPointで作成した教材コンテンツのスライドを，Moodleにアップロードした。授業中，教卓のPCを介してプロジェクタのスクリーンに投影するが，学生は自分のデバイスを持ち込んで閲覧できるようにした。このため，スライドはPDF化しておき，どのスマフォでもPDF

ビューワーだけで見ることができるように配慮した。

（2）Webサイト：Moodle[リソース][URL]

情報分野では技術革新が激しく，印刷した教科書となると，出版後には内容が陳腐化するものもある。このため，情報に関する最新の話題を提供するには，あまり適しているとはいえない。そこで，授業中に，直接Webサイトを閲覧することにした。

（3）動画：Moodle[リソース][URL]

最近ではYouTubeやニコニコ動画などの動画共有サービスが提供されている。この中に，教材に適したコンテンツも数多くあるので，これらを利用することができる。また，パッケージコンテンツとしてのDVD（例えば，ITホワイトボックス[12]など）を，教卓のPCを使って視聴することもできる。

（4）授業概要：Moodle[活動][課題]

平常点チェックの一環として，毎回授業の概要を，紙カードの手書きではなく，オンラインテキストでアップロードさせることにした。

Moodleの[課題]の[設定を編集する]において，[∨　提出タイプ]を[オンラインテキスト]とした。また，[∨　利用]の[開始日時]は毎週水曜日の授業開始時刻に，[遮断日時]は金曜日23時59分に，それぞれ設定した。以上によって，学生は授業中でも授業の概要を，教室に持ち込んだPC/タブレット端末/スマフォで入力できるようなるとともに，提出するまで3日間の猶予を与えた（図1）。

アップロードされた授業の概要に対しては，次回の授業までに筆者が評点（0点：未提出かコピペ発覚～10点：満点）をつけるとともに，質問が追記されたものについてはフィードバックコメントで回答を戻した。これについては，学生にとって好評だったようで，わからないことがわかってよかったという感想を得た。

図 1　Moodle[課題] の設定

（5）アンケート：Moodle[活動][フィードバック]

授業の最終回に，本科目に関する授業アンケートを実施した。これは，大学で行う授業評価アンケートとは別に，独自の評価項目を用意した。なお，前年度の[フィードバック]を教務課による一括移行を依頼すると，一部のデータが残ってしまい，今年度分と混ざるという事態が生じた。この場合は，[開始日時]前までに，[フィードバック]を[複写]することによって回避できる。

3. 指導方略の事例

ここでは，本科目において，文系学生に対するコンピュータサイエンスの基礎教育を，数式を多用せずにどのように実施しているのかについて報告する。具体的には，数式を多用しない代わりに，抽象的な事象をできるだけ視覚化してわかりやすくするために，図解を導入するという工夫を取り入れることにした。そのために用意した図版を中心に，指導方略における要点について述べる。

3.1　情報の符号化

現在のディジタルコンピュータにおいて，もっとも基本となる2元符号化について取り上げる。コンピュータの内部では，すべてのデータ（数値，文字，図表，画像，動画，音楽，音声など）が2進数に符号変換されて処理されており，2進数の1「オン」と0「オフ」は電子回路によるスイッチングによる動作として実装されている。これより，ディジタルコンピュータと2進数は，切っても切れない関係にあるといってよい。

指導方略の流れとしては，2/10/16進数→2/10進数の相互変換→負の数と進める。

（1）10進数と2進数

これについては，**図2**のスライドを用いて説明する。

ここで，桁の重み付けについては，

$$1234 = 1 \times 10^3 + 2 \times 10^2 + 3 \times 10^1 + 4 \times 10^0$$

という形で，指数で表すこともできる。しかしこの場合，例えば10の3乗は10を3回掛けた値（10 × 10 × 10）であること，かつ，10の0乗は1（$10^0 = 1$）になることなど，指数関数についてあらかじめ説明しておく必要がある。このため，スライドでは，指数関数を使わずに「「1」は1000の位」と表記してある。

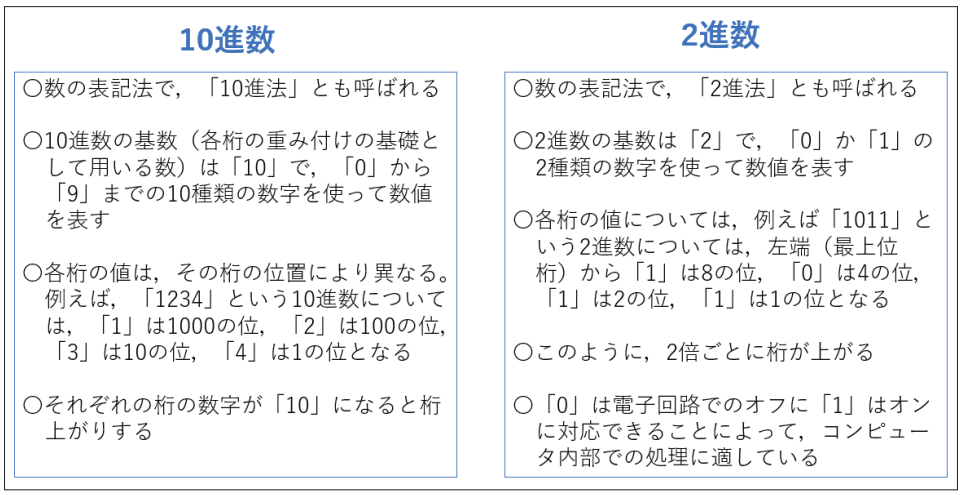


図２　10 進数と２進数

（2）10進数と2進数の相互変換

進数変換は，次のように規則的な式によって表すこともできる。

2）123 … 1
2）61 … 1
2）30 … 0
2）15 … 1
2）7 … 1
2）3 … 1
1

そして，余りを逆に並べた数である（1111011）$_2$が，2進数に変換した結果となる。これに対して，スライド（図3，図4）では，計算の手順をよりわかりやすくなるように図式化してある。

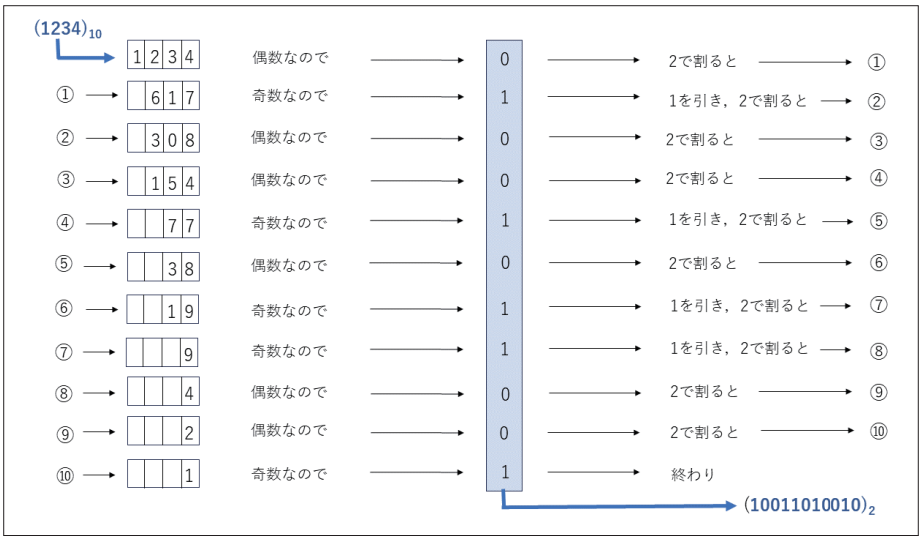


図３　10 進数→２進数

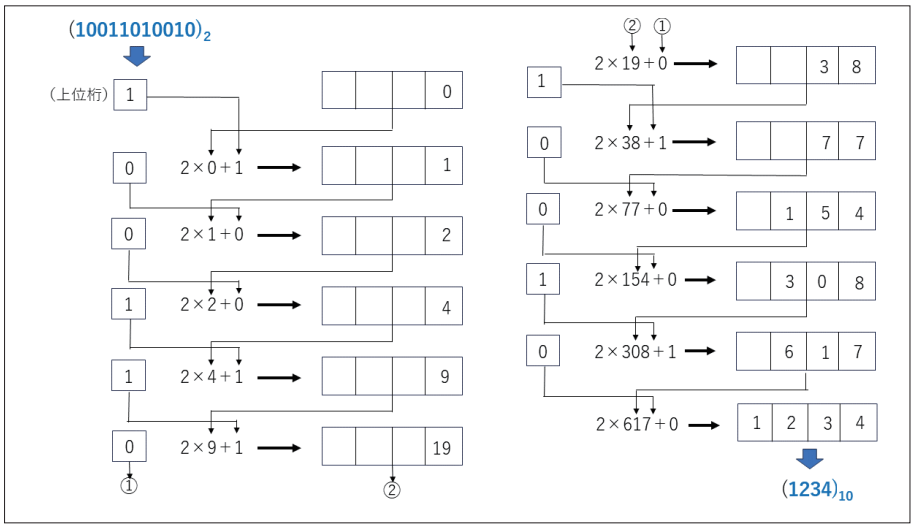


図４　２進数→ 10 進数

以上の進数変換は，何進数同士でも可能であることを補足する。

(3) 負の数の扱い

コンピュータの2進数において負の数を表す場合，補数を用いる。補数とは，計算を簡単にするために元の数値から算出される特殊な数のことである。

一般的には，Aをr桁のn進数とすると，

Aの1の補数＝$(n^r-1)-A$

Aの2の補数＝$\{(n^r-1)-A\}+1=n^r-A$

となる。しかし，これでは指数を使った数式となりわかりにくいので，次のように説明する。

n進数の補数については，元の数に足すと桁上がりする数のうち，一番小さいものを「nの補数」，元の数に足すと桁上がりしない数のうち，一番大きいものを「(n-1)の補数」とする。これを具体的に示すと，次のようになる。

【10進数の場合】

7の「10の補数」は3　※　[7](元の数)＋[3](桁上がり有で最小値)＝10(2桁目が桁上がり)

7の「9の補数」は2　※　[7](元の数)＋[2](桁上がり無で最大値)＝9(桁上がりなし)

【2進数の場合】

110の「2の補数」は010　※　[110](元の数)＋[010](桁上がり有で最小値)＝1000(4桁目が桁上がり)

110の「1の補数」は001　※　[110](元の数)＋[001](桁上がり無で最大値)＝111(桁上がりなし)

なお，2進数の「1の補数」は各桁の0と1を反転させ，「2の補数」は各桁の0と1を反転させて1を加えることでそれぞれ求めることができる。また，コンピュータの世界では，2進数の「2の補数」を用いる。

(4) 2進数の演算

2進数同士の四則演算については，図5のようなスライドで取り上げる。

図5の減算については，図6を用いて補足する。

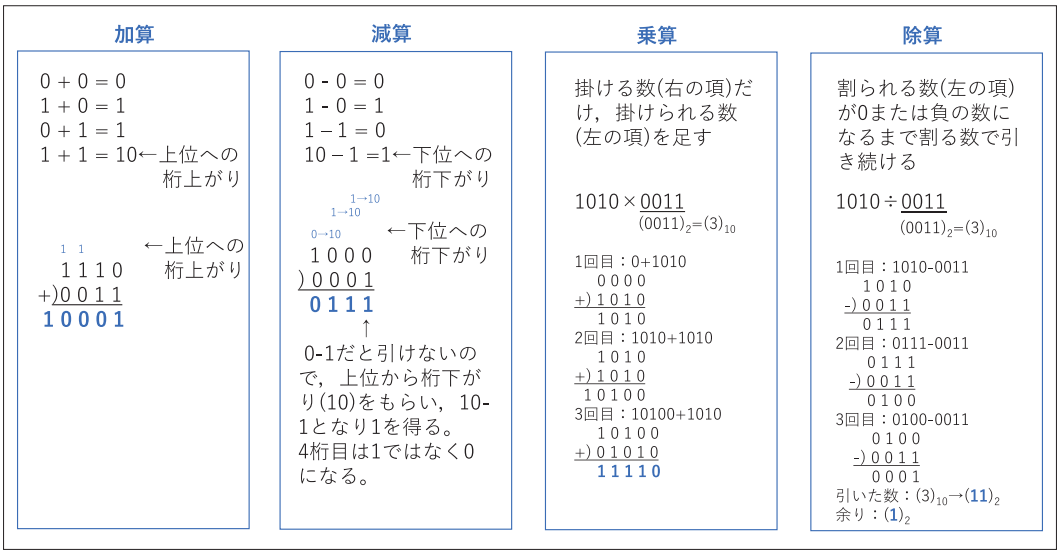


図５　２進数同士の四則演算

1桁目は0-1で引くことができず，上位桁から桁下がり値$(10)_2$（2進数の10，イチゼロ）をもらうことになる。しかし，2桁目の[引かれる値]は0，3桁目の[引かれる値]も0，4桁目の[引かれる値]が1なので，ここから下位へ桁下がりを行う。つまり，4桁目の値を0として，3桁目に桁下がり値$(10)_2$を渡す。次に，3桁目の値を1として，2桁目に桁下がり値$(10)_2$を渡す。さらに，2桁目の値を1として，1桁目に桁下がり値$(10)_2$を渡す。こうして，1桁目において，10-0-1＝1と減算ができる。そして，2桁目は1-0-0＝1，3桁目は1-0-0＝1，4桁目は0-0＝0となり，最終的に減算の答えは$(0111)_2$になる。

また，**図5**より，乗算は，加算を掛ける数（右の項）だけ繰返すことになる。除算は減算を割る数（右の項）だけ繰返すことになるが，減算は次の補数による加算に置き換えることができる。この結果，四則演算は，計算上，すべて加算だけで行うことができる。

(5) 補数による減算

[引かれる数]−[引く数]という減算式に対して，[引く数]の補数を求めて[引かれる数]＋[引く数]の補数]という形で計算することもできる。これについては，**図7**のスライドを用い，まずは10進数同士の減算（6-3と3-6の例）において，「10の補数」を用いて加算する手順を示す。その上で，

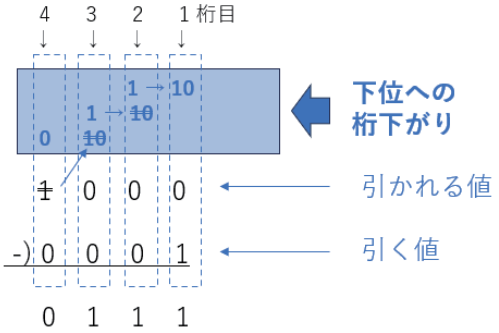


図６　２進数の減算

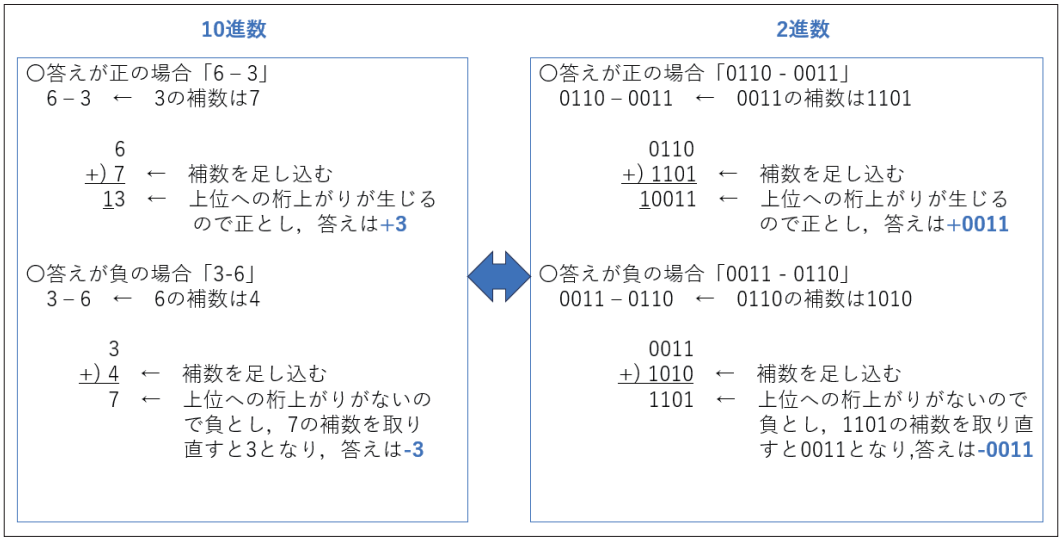


図７　補数による減算

2進数同士の減算（0110-0011 と 0011-0110 の例）を「2の補数」を用いて加算する計算を示す [13]。

以上の各演算のうち加算と減算については，コンピュータでは加算回路と補数回路を用いて実装している。乗算と除算については，加算と減算の繰り返しでは処理効率が下がるので，シフト演算回路を用いて実装している。

3.2　情報の演算と記憶

これも，ディジタルコンピュータの基本となる構成要素に位置づけられ，その最小単位は基本論理回路（基本ゲート回路）となる。これらを組み合わせることによって，算術演算用の論理回路（加算回路）や論理演算用の論理回路（エンコーダとデコーダ）といった組合せ回路を作り出すことができる。一方，入力状態から次の出力状態が決定される論理回路（フリップフロップ）といった順序回路を作り出すことができる。

以上より，組合せ回路は「情報の演算」をつかさどる ALU（算術論理演算装置）として，順序回路は「情報の記憶」をつかさどるレジスタやカウンタとして，それぞれ実装することができる。

指導方略の流れとして，「情報の演算」については基本論理回路→半加算回路→全加算回路（→ALU），一方，「情報の記憶」については基本論理回路→フリップフロップ回路（→レジスタ・カウンタ）となる。

以上の算術計算は論理演算の命題に帰着できることから，どんな算術計算でも電気的な信号（スイッチング）の組合せ，つまり，論理回路によって置き換えることができるようになった。この結果，世界初のディジタルコンピュータと言われている ENIAC は，真空管をスイッチとして使用した。その後，真空管→トランジスタ→集積回路→大規模集積回路と進み，さらには，メインフレームコンピュータだけでなく PC にも実装され今日に至っている。

3.2.1　情報の演算

情報の演算とは情報を処理するために実行する一連の計算や操作のことを意味しており，その演算をつかさどるハードウェアとして論理回路がある。

（1）基本論理回路

論理回路の最も基本的なものとして，基本論理回路（論理ゲート）がある。基本論理回路・真

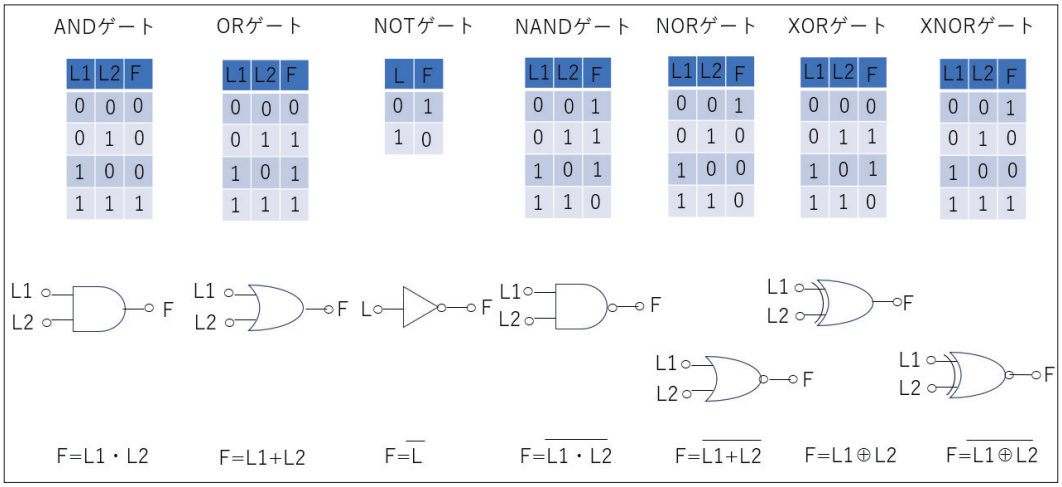


図 8　基本論理回路

理値表・論理変数を用いた論理式の関係について，図8のようなスライドを用いて説明する。

（2）半加算回路

2つの入力端子と2つの出力端子を持つ演算回路であり，図9のようなスライドを用いて説明する。真理値表に基づいて，出力の論理変数に関する論理式を導くと，

$$L3 = L1 \oplus L2$$

$$L4 = L1 \cdot L2$$

となる。これより，MIL記号を用いて半加算理回路を表すことができる。

（3）全加算回路

3つの入力端子と2つの出力端子を持つ演算回路であり，図10のようなスライドを用いて説明する。真理値表に基づいて，出力の論理変数に関する論理式を導くと，

$$L4 = L1 \oplus L2 \oplus L3$$

$$L5 = L1 \cdot L2 + L3 \cdot (L1 \oplus L2)$$

となる。これより，MIL記号を用いて全加算回路を表すことができる。

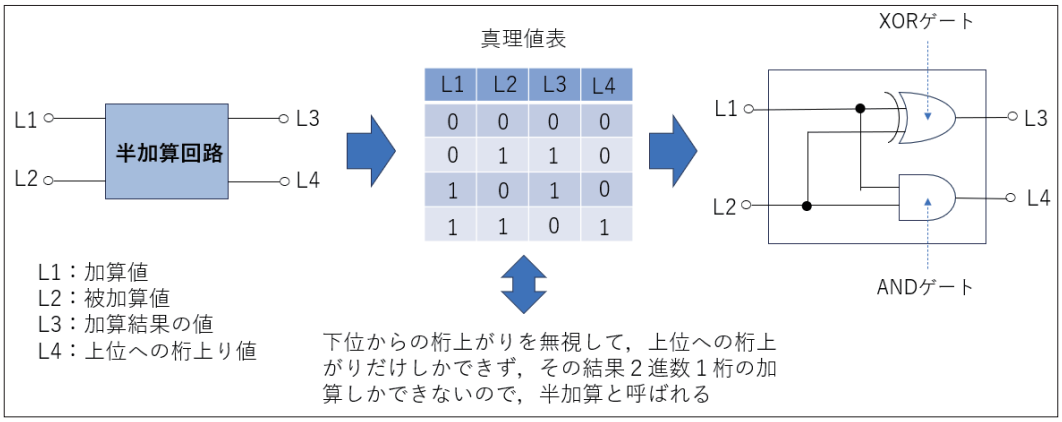


図９　半加算回路

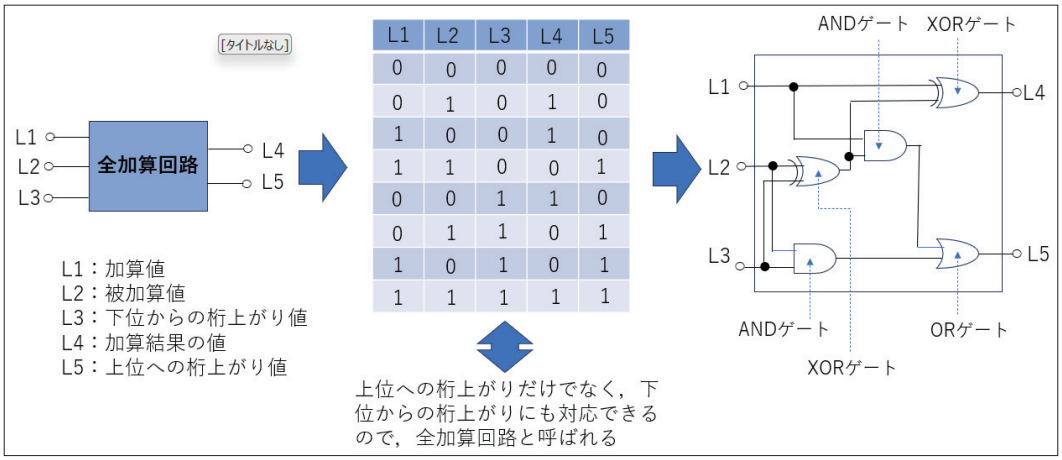


図 10　全加算回路

3.2.2　情報の記憶

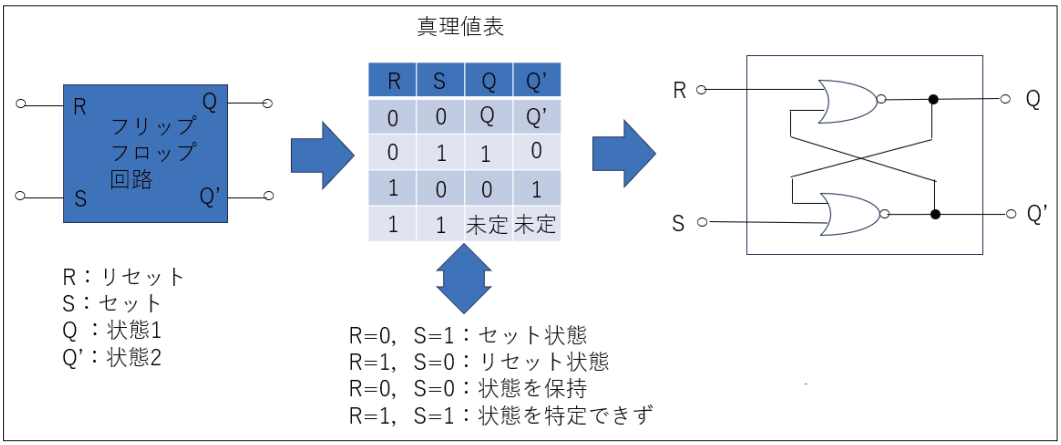
情報の記憶とは，情報を保存しておくことで後に取り出して利用できるようにするプロセスのことを意味しており，コンピュータのメモリを構成している。

（1）フリップフロップ回路

RS フリップフロップは，2つの入力端子（R は「Reset」と S は「Set」）と2つの出力端子（Q と Q’）を持つ論理回路であり，**図11** のようなスライドを用いて説明する。

図11 の真理値表より，「R=0 かつ S=0」の状態では，フリップフロップは前回の状態を保持する。つまり，出力 Q と Q’は前回と同じ値を保持するので，状態を記憶していることになる。「R=0 かつ S=1」の状態では，フリップフロップはセットされて出力 Q は1に Q’は0になる。「R=1 かつ S=0」の状態では，フリップフロップはリセットされて出力 Q は0に Q’は1になる。「R=1 かつ S=1」の状態は未定となり，出力 Q と Q’はどちらも1になってしまい，フリップフロップの基本的な動作としては意味がない。

以上のように，RS フリップフロップでは，0と1の入力状態の変化によって出力状態も変わるが，その際にタイミングのずれが生じると誤動作を起こすことになる。このため，各フリップフロップ同士で同期をとるためのクロックパルスを連動させる同期型 RS フリップフロップ回路もある（**図12**）。



R	S	Q	Q’
0	0	Q	Q’
0	1	1	0
1	0	0	1
1	1	未定	未定

図 11　RS フリップフロップ回路

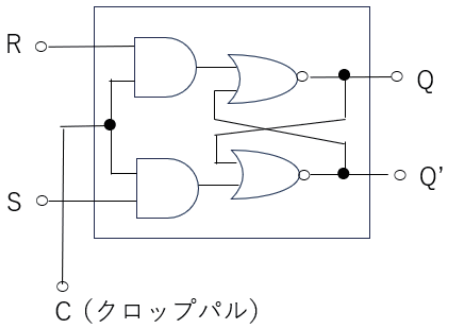


図 12　同期型 RS フリップフロップ回路

また，RSフリップフロップだけでなく，JKフリップフロップ，Tフリップフロップ，Dフリップフロップなどもある。この中のDフリップフロップを接続することでシフトレジスタを，Tフリップフロップを接続することでカウンタを，それぞれ作ることができる。

3.3　計算モデル

チーリングマシンは，英国のアラン・マシスン・チューリング（Alan Mathison Turing）によって提案された計算理論のモデルである。これより，あくまで理論上のコンピュータの動作原理を抽象化したものであるが，現在のコンピュータの原型に位置づけられている。

3.3.1　チューリングマシン

チューリングマシンを図で表すと，**図13**のようになる。

入出力テープは，無限に連なり，テープ上のマスの中には記号（0，1，空白）が入っている。ヘッドは，テープのマスを読み書きする装置でテープ上を左右に移動する。ヘッドは現在のマスの記号を読み取り，状態遷移表にしたがって次の動作を決定する。有限状態装置には状態遷移表が格納されており，ヘッドの動作（左，右，そのまま，停止）およびマス目の記号の読み取りや書込みを制御するための情報が書き込まれている。

チューリングマシン（TM）を定義すると，次のようになる[11]。

$TM = (Q, \Sigma, \Gamma, \delta, q_0, F)$

Q：有限個の状態の集合

Σ：有限個の入力される記号の集合

Γ：有限個のテープ記号の集合，$\Sigma \subset \Gamma$

δ：$Q \times \Gamma$から$Q \times \Gamma \times D$への写像となる状態遷移関数，Dはテープのヘッドの移動方向を表しL（左へ1マス），S（停止），R（右へ1マス）

q_0：初期状態，$q_0 \subset Q$

F：最終状態，$F \subset Q$

例えば，2つの数字の加算（2＋3）を行うチューリングマシンについては，

$TM = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$

$Q = (q_0, q_1, q_2, q_3, q_4, q_5)$

$\Sigma = \{1, +\}$

$\Gamma = \{1, +, B\}$　　※　Bは空白記号

$F = \{q_5\}$

$\delta(q_0, 1) = \{(q_1, B, R)\}, \delta(q_1, 1) = \{(q_1, 1, R)\}, \delta(q_1, +) = \{(q_2, +, R)\},$

$\delta(q_2, 1) = \{(q_2, 1, R)\}, \delta(q_2, B) = \{(q_3, 1, L)\}, \delta(q_3, 1) = \{(q_3, 1, L)\},$

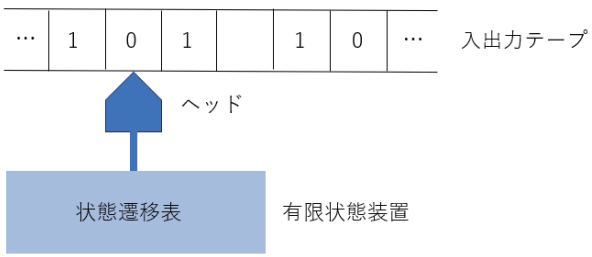


図13　チューリングマシンの構造

$$\delta(q_3, +) = \{(q_4, +, L)\}, \delta(q_4, 1) = \{(q_5, 1, L)\}, \delta(q_4, B) = \{(q_5, B, R)\},$$
$$\delta(q_5, B) = \{(q_0, B, R)\}, \delta(q_5, +) = \{(q_5, +, S)\}$$

となる。

初期状態において，入出力テープは11＋111（2を1の二つ並び，3を 1 の三つ並びとする）となっており，状態はq_0に設定されている。ヘッドは左端のマス目（1）に位置しており，状態遷移関数$\delta(q_0, 1) = \{(q_1, B, R)\}$を実行する。これによって，左端のマスは$B$に書き換えられ，ヘッドは右へ 1 マス移動し，状態はq_1となる。これを，$\delta(q_5, +) = \{(q_5, +, S)\}$まで繰返した結果，入出力テープはBB＋11111，状態はq_5となって停止する。つまり，11111と1の五並びで5となり，2＋3＝5の算術計算ができたことになる。

しかし，この説明では，文系の学生にとってこの数式の意味を理解するのは難しいといえる。そこで，**図14**のようなスライドを用いて具体的に説明する。

図14では，初期状態において，入出力テープは「11 111」，ヘッドは「11 111」の左横の「空」の位置，状態は「00」となっている。この状態でチューリングマシンの動作が始まると，状態遷移表において，状態が「00」で「空を読込んだとき」なので，①が実行される。これによって，状態は「00」，ヘッドは右のマスに一つ移動する。以降，状態遷移表の該当箇所に従って実行が繰り返され，状態が「11」で「空を読込んだとき」に⑦が実行され，チューリングマシンの動作は停止する。以上によって，入出力テープは「11111」と1が五並びで5となり，2と3の加算結果が5となる算術計算を実行したことになる[14]。

3.3.2　万能チューリングマシン

チューリングマシンは，入出力データと状態遷移表によって，ある動作を実行することができるマシンである。このチューリングマシンの動作をシミュレートすることができるマシンが，万能チューリングマシンとなる。つまり，他の任意のチューリングマシンの動作をシミュレートできるマシンであることから，どのようなチューリングマシンでもその動作を再現できる汎用的なチューリングマシンといえる。

状態	空を読込んだとき	1を読込んだとき
00	①ヘッドを右，状態を00	②ヘッドを右，状態を01
01	③1を書込み，ヘッドを右，状態を10	④ヘッドを右，状態を01
10	⑤ヘッドを左，状態を11	⑥ヘッドを右，状態を10
11	⑦ヘッドを停止	⑧空を書込み，状態を11

状態遷移表

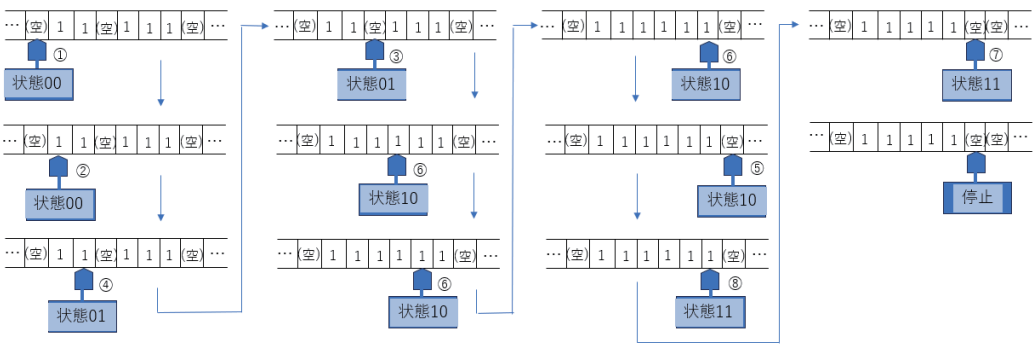


図 14　チューリングマシンの動作例

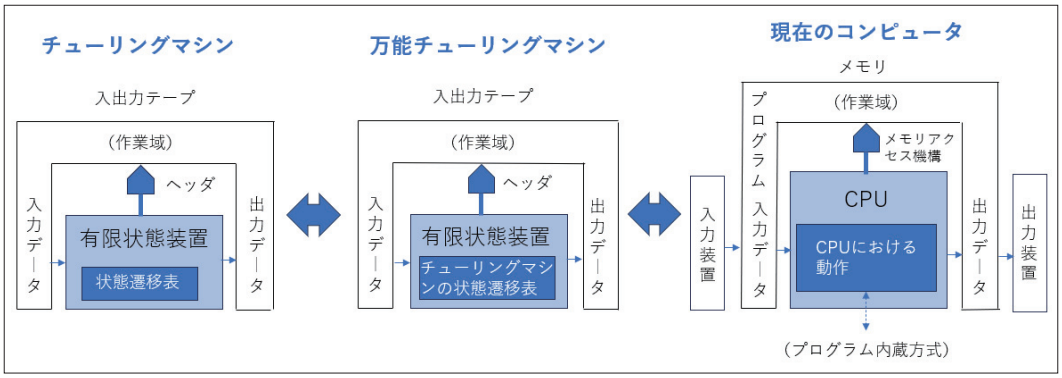


図 15　チューリングマシンから現在のコンピュータまで

以上より，万能チューリングマシンは，今日のコンピュータの原型にもなっている。別の言い方をすると，万能チューリングマシンとプログラム内蔵方式のコンピュータは同等でもある。

万能チューリングマシンの入出力テープは，コンピュータの入出力装置あるいはメモリやストレージに相当し，ここでデータが書き込まれたり読み出されたりする。万能チューリングマシンのヘッダはCPUにおけるコンピュータのメモリアクセス機構に，万能チューリングマシンの有限状態装置はコンピュータのCPUに，それぞれ相当する。また，万能チューリングマシンの状態遷移表は，コンピュータのCPUが実行する一連の動作に相当する。以上の関係を，図15のスライドを用いて説明する [15]。

3.4　プログラム理論

ここからは，コンピュータの動作原理を，ソフトウェアにおけるプログラムの観点から取り上げる。このプログラムは，プログラミング言語によって記述されるとともに，「プログラム＝アルゴリズム＋データ構造」という関係が成立する。その上で，コンピュータ内部において，プログラムがどのように実行されるのかについて説明する。

3.4.1　BNF とプログラミング言語

BNF（Backus-Naur Form）は，ジョン・ワーナー・バックス（John Warner Backus）とピーター・ナウア（Peter Naur）により，プログラミング言語（例えば，ALGOL60）の文法構造を明確に定義するために考案された記法である。また，形式言語理論における文脈自由文法を定義するために用いられるメタ言語（ある言語の文法を記述するための言語）でもある。

（1）BNFの記法

BNFでは，次のような記述を用いる [16]。

①非終端記号

文法や言語の構造を表す記号である。非終端ということで「他の記号に置き換えられる」という意味である。通常，角括弧（＜と＞）で囲んで表す。

②終端記号

文法規則で最終的に生成される文字やシンボルであって，これ以上分解できない記号である。終端ということで，「これ以上置き換えることができない」という意味である。

③生成規則

左辺には非終端記号を，右辺にはその非終端記号が生成できる構造を，左辺と右辺の間には::=

を挿入する形で記述する。

④論理和記号

「|」は，「または」を表す記号である。

例えば，数字の並びをBNFで記述すると，次のようになる。

＜数字リスト＞::=＜数字＞|＜数字＞","＜数字リスト＞

＜数字＞::="0"|"1"|"2"|"3"|"4"|"5"|"6"|"7"|"8"|"9"

これより，＜数字リスト＞として，例えば「0」，「1,3」，「2,4,6」が生成されるとすると，

・「0」は「＜数字リスト＞::=＜数字＞」で，＜数字＞から0を生成

・「1,3」は「＜数字リスト＞::=＜数字＞,＜数字リスト＞」で＜数字＞から1を生成し，「＜数字リスト＞::=＜数字＞」で＜数字＞から3を生成

・「2,4,6」は「＜数字リスト＞::=＜数字＞,＜数字リスト＞」で＜数字＞から2を生成し，「＜数字リスト＞::=＜数字＞,＜数字リスト＞」で＜数字＞から4を生成し，「＜数字リスト＞::=＜数字＞」で＜数字＞から6を生成

となる。この中の「2,4,6」という＜数字リスト＞は，最終的に各数字が単一の＜数字＞に分解されて再帰的な定義が終わる。つまり，このプロセスで再帰が終わる場所が，単一の＜数字＞になる。このように，再帰を用いた定義によって，任意の長さのリストを作り出すことができる。

(2) Pascalの構文規約

ALGOL60以降では，Pascalというプログラミング言語の構文規約が日本工業規格において制定されている[17]。

例えば，「定義の方法」ではプログラム要素の構文を規定するのに，BNFに基づいた記述方法を用いている。そのBNFで使う記号の意味については，**表2**のようになる。この中の超識別子は，漢字，仮名及び英字の列とする。生成規則の中の終端記号または非終端記号の列は，その各々が生成する最終的な文字列をその順に連結したものを表す。

表2を用いて，「数」を定義すると次のようになる。

符号付き数＝符号付き整数|符号付き実数.

符号付き実数＝[符号]符号なし実数.

符号付き整数＝[符号]符号なし整数.

符号なし数＝符号なし整数|符号なし実数.

符号＝'＋'|'−'.

表２　構文の記述に使う記号

記号	意味
＝	定義する
＞	部分的に定義する
|	または
.	定義の終わり
[x]	xまたは空
{x}	xを0個以上並べたもの
(x|y)	xまたはy（一つにまとめて扱うために用いる）
'xyz'	終端記号xyz
超識別子	非終端記号

符号なし実数＝数字列'.'小数部['e'指数]|数字列'e'指数.
符号なし整数＝数字列.
小数部＝数字列.
指数＝[符号]数字列.
数字列＝数字{数字}.

3.4.2　プログラムの実行制御

PCの内部では，どのような処理が行われているのか，その実行制御の仕組みについて取り上げる。これによって，コンピュータをホワイトボックス化することができるといえる。コンピュータは，プログラムによって動いている。そのプログラムは高水準言語と呼ばれるプログラミング言語によって記述されているが，低水準言語である機械語に変換されることで実行される。実行するプログラムのファイル識別子は，「.exe」となる。

ここでは，加算を行うプログラム（2+3）を例にあげる。本来，機械語は2進数により構成されるため命令や番地やデータも任意のビット列となるが，便宜上，記号と10進数で扱うことにする。

```
LOAD 100    ← 100番地にあるデータ「2」をアキュムレータ（累算機）に転送
ADD 101     ← 101番地にあるデータ「3」を取り出しALUで加算（2+3）しアキュムレータの値を「5」に更新
STORE 102   ← アキュムレータの内容「5」を102番地に格納
STOP 0      ← プログラムを停止
```

以上をもとに，**図16**のスライドを用いて説明する。

プログラムとデータは補助記憶装置（SSDかHDD）に記憶しておくが，実行するときには主記憶装置（メインメモリ）のある領域にローディングされ，プログラム領域とデータ領域に命令とデータが格納される。

プログラムの実行手順（命令サイクル）は，次のようになる。

①プログラムカウンタは0なので0番地を参照する。
②格納されている命令「LOAD 100」を命令レジスタに転送する。
③「LOAD 100」を実行し，100番地にあるデータ「2」をアキュムレータに転送する。
④プログラムカウンタが＋1されて1番地になる。
⑤1番地に格納されている命令「ADD 101」を命令レジスタに転送する。
⑥「ADD 101」を実行し，101番地にあるデータ「3」を取り出してアキュムレータに転送する
⑦ALUにおいて「2＋3」を実行し，加算結果である「5」をアキュムレータに転送する。
⑧プログラムカウンタが＋1されて2番地になる。
⑨ 2番地に格納されている命令「STORE 102」を命令レジスタに転送する。
⑩「STORE 102」を実行し，アキュムレータの値「5」を102番地に転送する。
⑪プログラムカウンタが＋1されて3番地になる。
⑫ 3番地に格納されている命令「STOP 0」を命令レジスタに転送する。
⑬「STOP 0」が実行されて，プログラムが停止する。

図16の構図は，ノイマン型コンピュータにも結び付く。ノイマン型コンピュータは，ジョン・フォン・ノイマン（John von Neumann）が提唱したコンピュータの基本的な設計思想のことである。具体的には，単一メモリ構造（プログラムとデータが同じメモリ空間に格納されていること），プ

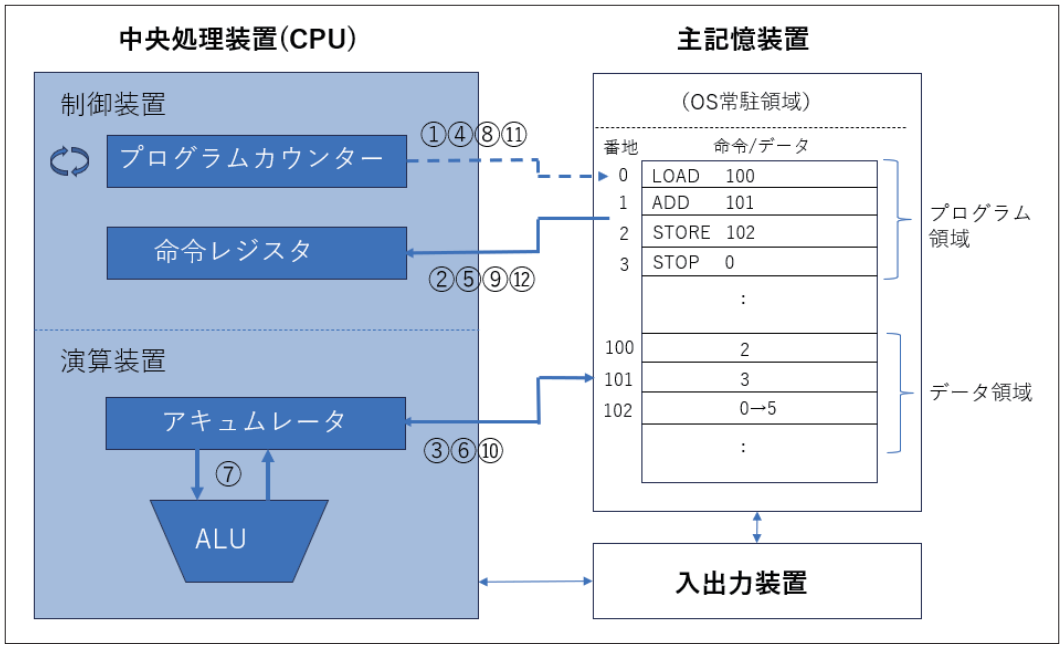


図 16　プログラムの実行制御

ログラム内蔵方式（プログラムとデータを記憶装置に格納していること），逐次処理（プログラムカウンタにより，命令を一つずつ取り出して実行すること）などがあげられる。これらは，現在のコンピュータのアーキテクチャーに採用されている。

おわりに

以上，科目「情報処理論」における指導方略について，授業で使っているスライドをベースに取り上げた。「情報処理論」では，コンピュータをできるだけホワイトボックス化することによって，統制感を持った健全な利用者を育成することを目指している。このため，コンピュータサイエンス分野でもあるコンピュータの動作原理（2元符号化，論理回路，論理演算，チューリング機械，形式言語理論，プログラム理論，アーキテクチャー論など）について，あえて踏み込んだ形で取り組んでいる。ただし，文系学生が対象となるので，数式を多用することなく，かつ，抽象的ではなくできるだけ具体的にイメージできるように図解を多く用いるといった工夫を取り入れている。

今後の課題については，「情報処理論」の授業内容に則した自学自習ベースのディジタル教材コンテンツを開発する予定である。

参考文献

[1] 今井敏博：理系の大学生と文系の大学生の学校数学に対する意識の比較：同志社女子大学の学生を対象にして，同志社女子大学総合文化研究所紀要第24巻，pp. 150–161，2007年.

[2] 野津田雄太，高橋健一，稲葉通将：大学生アンケートからの文系理系学生の特徴に関する分析，情報処理学会論文誌教育とコンピュータ，Vol. 1，No. 4，pp. 83–92，2015年.

[3] 文科省：数理・データサイエンス・AI教育プログラム認定制度
https://www.mext.go.jp/a_menu/koutou/suuri_datascience_ai/00001.htm
[4] 岡部恒治，戸瀬信之，西村和雄：算数ができない大学生：理系学生も学力崩壊，東洋経済新報社，2001年.
[5] 岡部恒治，戸瀬信之，西村和雄：小数ができない大学生：国公立大学も学力崩壊，東洋経済新報社，2000年.
[6] 岡部恒治，戸瀬信之，西村和雄：新版 分数ができない大学生，東洋経済新報社，2010年.
[7] 河村一樹：コンピュータサイエンスを基盤とした情報処理教育に関するカリキュラム研究，東京国際大学論叢商学部編，第73号，pp. 105–118，2006年.
[8] 廣瀬　健，高橋延匡，土居範久編：コンピュータソフトウェア事典，丸善，1990年.
[9] 東京国際大学公開シラバス
https://poti.tiu.ac.jp/uprx/up/pk/pky001/Pky00101.xhtml
[10] 原田悦子：文系学部におけるプログラミング教育の意義：健全なユーザ育成のための情報教育の視点から，法政大学社会学部学会社会労働研究，第38巻，3・4号，pp. 119–134，1992年.
[11] 河村一樹：入門情報科学シリーズコンピュータ基礎論，ソフトバンク，1995年.
[12] NHK：ITホワイトボックス
https://www2.nhk.or.jp/archives/movies/?id=D0009042206_00000
[13] 河村一樹：図解雑学コンピュータ科学の基礎，ナツメ社，2003年.
[14] 加藤直樹：コンピュータの仕組み――教員のためのプログラミング入門
http://wiki.bmoon.jp/wiki.cgi/Programming
[15] 小倉久和：情報科学の基礎論への招待，近代科学社，1998年.
[16] 河村一樹：読む講義シリーズ1 コンピュータ科学基礎，ITEC，1997年.
[17] 日本工業規格：X3008-1994
https://kikakurui.com/x3/X3008-1994-01.html