■6 群（コンピュータ･基礎理論とハードウェア）- 4 編（コンピュータアーキテクチャ（I）標準的）

1 章 コンピュータのモデル

【本章の構成】

本章では，ノイマン型コンピュータ（1-1 節），命令セットアーキテクチャとマイクロアーキテクチャ（1-2 節）について，それぞれ述べる．

■6 群 - 4 編 - 1 章

1-1 ノイマン型コンピュータ

（執筆者：入江英嗣）［2011 年 8 月 受領］

1-1-1 汎用ディジタルコンピュータの基本デザイン

自動計算機の歴史は古くは 17 世紀のパスカリーヌまでさかのぼるが，1945 年にジョン・フォン・ノイマンによって発表された方式[1)]は，現在に至るまでのほとんどの汎用ディジタルコンピュータの基本デザインとなっている．この方式では，実行したい命令列をデータとして主記憶装置に格納し，逐次的に解釈・実行することで，複雑な計算処理を自動的に行うことができる．最初期の電気式計算機では，異なる計算処理を実行するためには配線やスイッチなどを人手でやりなおす必要があったが，ノイマンが発表した方式は，プログラムデータを入れ替えることによって実現するため，高い汎用性を獲得している．数学的には，主記憶装置の容量が充分であればこの方式はチューリング完全な実装となっており，あらゆる計算可能な問題を解くことができる．

1-1-2 「ノイマン型」の由来

この方式は，史上初（諸説ある）のコンピュータ ENIAC を開発したジョン・エッカートとジョン・モークリーが EDVAC 開発計画において考案し，ノイマンはその数学的な裏付けを与えたと言われている．しかし上述の 1945 年，ノイマンが自身の名前で論文を発表してしまったため，ノイマン型の名前で広く知られるようになった．EDVAC は開発が難航し，史上初のノイマン型コンピュータは，この論文に影響を受けて開発された EDSAC となっている．

また，プログラムをデータとして記憶装置に格納する方式を指す「プログラム内蔵方式」という呼び方もある．「ノイマン型」と「プログラム内蔵方式」はほぼ同じ文脈で用いられるが，厳密には，ノイマン型コンピュータはプログラム内蔵方式コンピュータの一種類である．この 2 つを分けて考える場合，ノイマン型コンピュータの逐次性や，同じ転送路によってデータとプログラムを取り込むこと，実行中のプログラムを書き換えることなどの特徴が注目される．

1-1-3 ノイマン型コンピュータの概要

(1) コンピュータの構成

ノイマンの論文では，コンピュータの構成要素として，5 つの主要な装置を数え上げている（**図 1･1**）．

(a) 中央演算装置

与えられる制御に従って演算を行う．演算機能は四則演算をはじめ，頻出するものであれば算術関数なども実装される．

(b) 中央制御装置

各装置の動作を制御して，プログラム実行を進める．

(c) 記憶装置

命令，計算の中間値，外部から与えられる数値などを，同様に保持する．

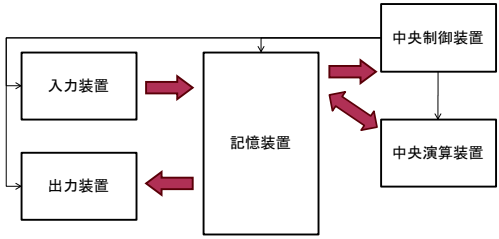

図 1·1　コンピュータの構成要素

（d）入力装置

コンピュータと人間とのやり取りは，主記憶装置よりも人間が直接読み書きしやすい（当時であればパンチカードのような）外部記憶媒体を介して行われる．入力装置はこのような外部記憶媒体から主記憶装置へのデータ転送を行う．

（e）出力装置

主記憶装置から外部記憶媒体へのデータ転送を行う．入出力装置を介しては，プログラムの初期入力や最終結果だけではなく，外部記憶媒体を階層記憶として用いた中間数値の転送も行われる．

（2）今日的な構成の数え上げ

プログラム処理の中核となる中央演算装置と中央制御装置は，今日では CPU（中央演算処理装置）またはプロセッサとして，まとめて呼ばれることが多い．また，主記憶装置，入力装置，出力装置の 3 つの要素についても，メモリマップド I/O のように，統一して扱う方式が出てきたことから，「メモリサブシステム」として，一つにまとめて呼ばれることがある．

（3）計算処理の進行

中央制御装置は主記憶装置との接続箇所を管理している．これは，今日的には命令アドレスを指すプログラムカウンタに相当する．中央制御装置は，この接続箇所からまず 1 語*を読み出し，接続箇所を次の語へ移す．読み出された語は命令であるか数値であるか，また命令であればどのような命令なのかが解釈され，その通りに実行される．

例えば 2 つの数の加算であれば，続く接続ポイントから数値を読み出して中央演算装置の入力バッファに転送し，更に続く接続ポイントからもう片方の数値を読み出して，中央演算装置のもう一方の入力バッファへ転送する†．中央演算装置は組合せ回路となっているため，入力バッファの転送が完了すると出力に加算結果が現れる．出力値は，続く演算命令の入力としてフォワードされるか，あるいは後続命令によって特定の主記憶箇所へ書き出される．命令にはこのほか，接続ポイントを任意の場所に変更するもの，入出力装置を操作するものなどがある．

このようにして，1 語ずつ読み出しながら命令を実行することにより，計算処理処理が進んでいく．

---

* 一定数の bit の集まり．ただし，報告書当時は「語 (Word)」という言葉はまだ使われていない．

† 主記憶内でこのような配置となるようにプログラムが行われる

1-1-4 フォン・ノイマン・ボトルネック

ノイマン型コンピュータでは，コンピュータが搭載している主記憶容量が大きくなっても，一箇所の接続を通した逐語的な転送が行われる．このため，どのように大きなシステムとなってもプロセッサと主記憶装置とを結ぶ一点が転送上の隘路（ボトルネック）となる．ノイマン型コンピュータが持つこの性質はハードウェア性能上の隘路というのみならず，プログラマが考えるアルゴリズムも逐語転送に制限するとして，ジョン・バッカスによって「フォン・ノイマン・ボトルネック」と名付けられ，その後のハードウェア，ソフトウェア双方の並列化に影響している．

今日のコンピュータシステムではノイマン型を基本としているが，様々な粒度の並列実行を行い，このボトルネックの分散を試みている．また，プロセッサと主記憶との間にはレジスタファイルやキャッシュメモリが備えられ，転送路を高速化している．更には，命令キャッシュとデータキャッシュを分割して，転送路の競合を軽減するような構成がとられることも多い．

1-1-5 非ノイマン型コンピュータ

ノイマン型コンピュータは汎用ディジタルコンピュータの基本デザインを確立した方式だが，これとは全く異なる方式による非ノイマン型コンピュータも存在する．フォン・ノイマン・ボトルネックのブレイクスルーを実現するような超並列コンピュータとして，データフローマシン，DNA コンピュータ，量子コンピュータなどが研究されている．

また，プログラム内蔵方式以前のコンピュータを指して，「非ノイマン型」と称する場合もある．

■参考文献

1) J. von Newmann：“First Draft of a Report on the EDVAC,” *IEEE annals of the History of Computing*, vol.15, no.4, pp.27-75, 1993.

■6 群 - 4 編 - 1 章

1-2 命令セットアーキテクチャとマイクロアーキテクチャ

（執筆者：吉瀬謙二）[2018 年 12 月 受領]

プロセッサが使用できるデータ形式及びレジスタ，プロセッサが処理できる命令の機能，命令形式，アドレス方式などの定義を命令セットアーキテクチャ（Instruction Set Architecture：ISA）と呼ぶ．命令セットアーキテクチャはソフトウェアとハードウェアのインタフェースとしての役割を持ち，プログラマやコンパイラはこの命令セットアーキテクチャによって定義された命令列を生成する．

2018 年現在, 広く用いられている命令セットアーキテクチャには MIPS, x86, ARM, RISC-V [1]などがある．命令セットアーキテクチャの歴史については 6 群 1 編 1-1-2 を参照のこと．

x86 アーキテクチャは PC 市場における高いシェアを持つ．また，複雑な命令体系の CISC（Complex Instruction Set Computer）アーキテクチャに分類される．ARM は組込みシステムやスマートフォンといった電力効率の高いプロセッサを必要とする領域で高いシェアを持つ．

MIPS と RISC-V は, 単純な命令体系を特徴とする RISC（Reduced instruction Set Computer）アーキテクチャに分類される． RISC-V は，ソフトウェアの Linux のように無償で公開されて誰でも自由に改良と再配布ができる命令セットアーキテクチャとして 2011 年に登場したものである．MIPS は実用的な面に加えて，そのシンプルな構成によりコンピュータアーキテクチャの教科書[2]で用いられることが多い．RISC-V と同様に，MIPS もオープンソース化されることが 2018 年 12 月に発表されている．

多くの命令セットアーキテクチャには，当初発表されたものから拡張されてきた経緯がある．例えば，MIPS, Intel x86, ARM を含む多くの命令セットアーキテクチャは，性能向上のために SIMD（Single Instruction stream, Multiple Data stream）命令が追加されている．マイクロプロセッサにおける SIMD 命令拡張については 6 群 5 編 4-2-2, 4-2-3 を参照のこと．

命令セットアーキテクチャよりも下位の階層として定義されるマイクロプロセッサ内部の詳細な方式のことをマイクロアーキテクチャ（Microarchitecture）と呼ぶ．6 群 1 編 1-1-2 (3) プロセッサ内部の構成方式を参照のこと．

同じ命令セットアーキテクチャを採用しながら，異なるマイクロアーキテクチャのプロセッサを実装することができる．例えば，同じ命令セットアーキテクチャである x86 のマイクロアーキテクチャとして，パイプライン段数の多い NetBurst マイクロアーキテクチャや，高い性能と低消費電力の両立を目指す Core マイクロアーキテクチャが開発されている．

マイクロアーキテクチャ技術には，命令パイプラインの段数やそれぞれのステージにおける処理の構成，スカラあるいはスーパスカラのどちらを採用するか，キャッシュの構成，分岐予測の構成，命令レベル並列性（Instruction-Level Parallelism：ILP）と命令スケジューリングの方式，マルチスレッディングの方式，マルチコアを採用するかなどの選択肢が含まれる．命令パイプラインについては 6 群 4 編 7 章を参照のこと．スーパスカラについては 6 群 4 編 8-2 を参照のこと．キャッシュについては 6 群 4 編 4-3 を参照のこと．分岐予測については 6 群 5 編 1-1-2 を参照のこと．複数スレッド処理とスループット指向コンピュータについては 6 群 5 編 2-2-1 を参照のこと．ILP と命令スケジューリングについては 6 群 4 編 7-1 を参照のこと．

■参考文献

1) デイビッド・パターソン, アンドリュー・ウォーターマン：“RISC-V 原典 オープンアーキテクチャのススメ，“日経 BP 社.
2) ジョン・L・ヘネシー, デイビッド・A・パターソン：“コンピュータアーキテクチャ 定量的アプローチ 第 5 版, “翔泳社.