4.5 内部構造と動作の仕組

第 1 章の 1.3 節で述べたように、現在のコンピュータは大きく 5 つの機能によって構成されています (p.2, 図 1.2 参照)。この節では、これまで学習した内容とあわせて、更に詳しくコンピュータの本質について学習します。特に、コンピュータの内部構造と動作の仕組を中心に解説します。

現在のコンピュータは、「**ノイマン型コンピュータ**」であるといわれています。コンピュータの歴史においてイギリスの数学者 **フォン・ノイマン** (Von Neuman) の業績は大きく、現在のコンピュータの基本的な特徴である、プログラムとデータを一緒に記憶しておく「**プログラム内臓方式**」や、命令を一つずつ実行しながら処理を進める「**逐次制御方式**」、さらに「2 進法[10]」などの概念の採用を提唱しました[11]。更に、フォン・ノイマンは、これらの概念を取り入れて現在のコンピュータの原形を作り出しました[12]。これらの特徴を兼ね備えた現在のコンピュータの内部構造は図 4.42 のようになっています。

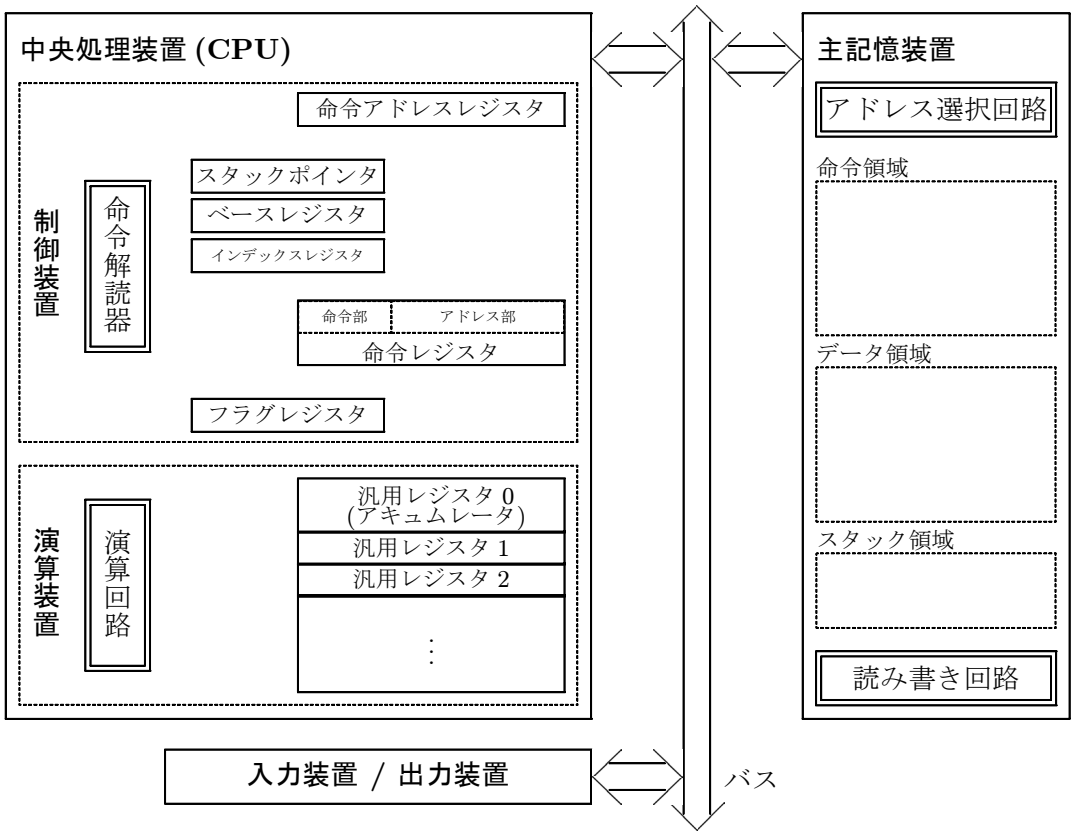

図 4.42: コンピュータの内部構造

[10] 世界初の実用コンピュータ ENIAC (エニアック) は、10 進法で計算していた。

[11] これらの概念は、すでに前人によってその原形が考えられていました。なお、その中には多くの数学者が含まれています (チューリング, ブール, シャノン, パスカル, ライプニッツなど)。

[12] 第 1 章の 1.4 節で紹介したように、フォン・ノイマンが提唱したプログラム内臓方式を採用した世界初のコンピュータは EDSAC(エディサック, イギリス, ケンブリッジ大学, 1949 年) です。また、その 3 年後にはフォン・ノイマン計算機とも呼ばれた IAS コンピュータ (プリンストン計算機, イギリス, プリンストン高等研究所 (IAS), 1952 年) が作られています。

各装置の働きは次のとおりです。

主記憶装置　主記憶装置 (メインメモリ) は、ノイマン型コンピュータの特徴であるプログラム内臓方式を実現するための装置で、プログラムやデータを記憶する**主記憶領域** (**命令領域**, **データ領域**, **スタック領域**) と記憶したプログラムやデータの読み書きを制御する**アドレス選択回路**・**読み書き回路**から成っています。実際にプログラムやデータを記憶する主記憶領域は、1 ワード (8, 16, 32, 64 ビットなど) を基準に図 4.43 のように一連の番号が割り振られており、この一連の番号を**アドレス** (Address) と呼び、各々のアドレスを「**何々番地**」のように呼びます (特に、物理的に直接割り振られたアドレスを**物理アドレス**または**絶対アドレス**と呼びます)。なお、このアドレスをアドレス選択回路に渡すことで、読み書き回路から各装置にプログラムやデータを渡します。

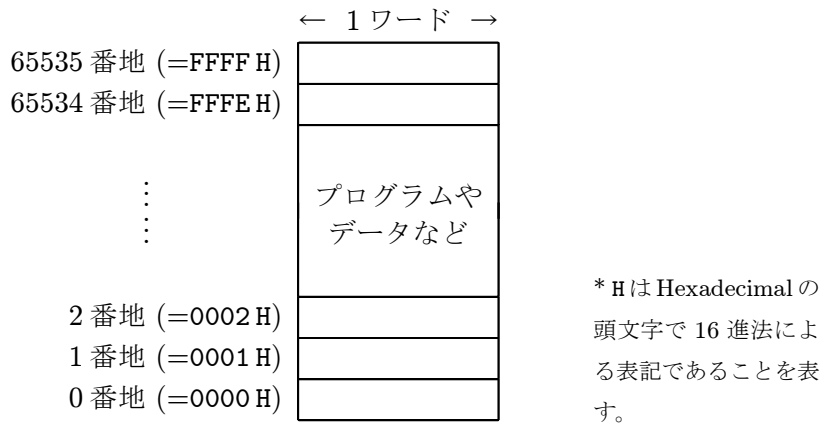

図 4.43: 65536 ($= 2^{16}$) ワードが記憶可能な主記憶装置のアドレス

レジスタ　**レジスタ** (register) は、4.3 節で取り上げたフリップフロップによって構成されており、制御装置や演算装置が使用するプログラムやデータを記憶する機能を持ちます。レジスタの中には、下記のように使用目的が決められた**専用レジスタ**とプログラムで自由に使える**汎用レジスタ** (General Purpose Register) があります。

- **ベースレジスタ** (Base Register; **基底アドレスレジスタ**)　ベースレジスタは、主記憶装置に記憶されたプログラムやデータを読み書きする上で、基準となるアドレスを記憶しておくレジスタで、一般には、プログラムの先頭アドレスが記憶されます。なお、ベースレジスタに記憶されたアドレスを利用して物理アドレスを求める方法を**ベースアドレス指定方式**と呼び、このように何らかの計算によって求められた物理アドレスを**実効アドレス** (effective address) と呼びます (仮想記憶の場合には論理アドレスを指すこともあります)。

- **インデックスレジスタ**(Index Register) インデックスレジスタは、主記憶装置に記憶されたプログラムやデータを読み書きする上で、基準となるアドレスからの増減値を記憶しておくレジスタで、一般には、汎用レジスタをインデックスレジスタとして利用します。なお、インデックスレジスタに記憶された増減値を使って物理アドレスを求める方法を**インデックスアドレス指定方式**と呼びます (アドレスを求める方法には、ベースアドレス指定方式やインデックスアドレス指定方式の他に、物理アドレスを直接指定する**直接アドレス指定方式**や、何らかの計算を行って物理アドレスを求める**間接アドレス指定方式**・**レジスタアドレス指定方式**などがあります)。

- **命令アドレスレジスタ**(Instruction Address Register; **プログラムカウンタ**) 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記憶するレジスタです。命令アドレスレジスタの値は、1 つ命令を実行するたびに命令解読器によって次に実行する命令のアドレスに書き換えられます。

- **命令レジスタ**(Instruction Register) 命令レジスタは、主記憶装置から読み出した命令を記憶するレジスタです。命令レジスタは、図 4.42 のように命令部とアドレス部に分かれ、命令部は命令解読器に渡され、アドレス部はベースレジスタ・インデックスレジスタと共に命令を実行するために必要なデータのレジスタとオペランドアドレス (プログラムの先頭番地からの相対アドレス) の選択に使用されます。

  アドレス部=レジスタ番号+[インデックスレジスタ]+オペランドアドレス

- **スタックポインタ**(Stack Pointer) スタックポインタは、主記憶装置のスタック領域の先頭アドレスを記憶するレジスタで、一般には、汎用レジスタをスタックポインタとして利用します。図 4.44 のように、**スタック**は最後に入れたものを最初に取り出す機能を持っています (LIFO; Last In First Out)。

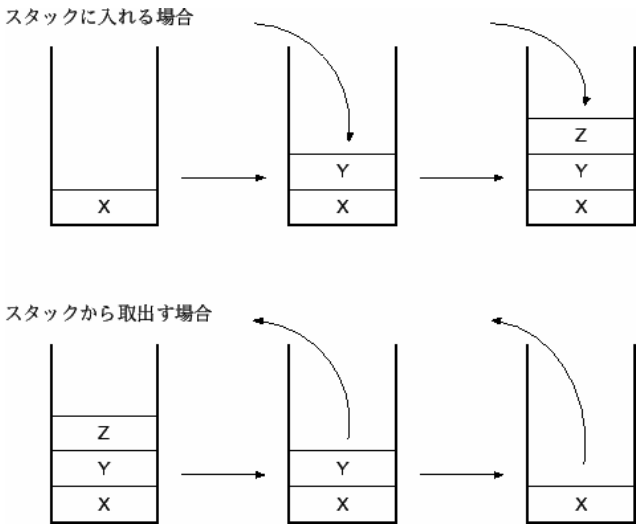

図 4.44: スタックの仕組

- **フラグレジスタ**(Flag Register)　フラグレジスタは、演算装置による演算結果の状態を記憶します。アキュムレータの値の符号 (正・負・ゼロ) に関する情報やオーバーフロー・アンダーフローの有無などの情報が、フラグレジスタを構成する各ビットに割り当てられ、真ならば 1・偽ならば 0 が記憶されます。

- **アキュムレータ**(Accumulator; **累算器**)　アキュムレータは、演算装置によって利用されるレジスタで、演算回路で演算された結果が記憶されます。現在は、ほとんどアキュムレータと汎用レジスタの区別がなく、全ての汎用レジスタがアキュムレータとして使用可能です (便宜上、汎用レジスタ 0 をアキュムレータとして扱うことが多い)。

制御装置　制御装置の中の**命令解読器** (instruction decoder; **復号器**; **デコーダ**) は、命令レジスタの命令部を解読し、演算装置の数ある演算回路の中から該当する演算回路の選択を行います。このとき、フラグレジスタの値も考慮されます。また、命令アドレスレジスタに入っているアドレスを次に実行する命令が入ったアドレスに書き換え、処理の進行を制御します。

演算装置　演算装置の中には様々な演算回路があり、4.4 節で学んだ全加算器もその 1 つです。主記憶装置からアキュムレータと (複数の) 汎用レジスタに読み込まれたデータを命令解読器によって選択された演算回路に通すことで演算が実行され、演算結果はアキュムレータに格納 (記憶) されます。このとき、演算結果によってはフラグレジスタの値を書き換えます。

バス　**バス** (Bus) は、中央処理装置・主記憶装置・入力装置・出力装置の間に引かれた道路で、この道路を使ってプログラムやデータの受け渡しを行います。一昔前は、全ての装置を 1 つのバスに接続していたため低速で動作する装置によってコンピュータ全体の処理速度を低下させていましたが、現在のコンピュータは、低速で動作する装置と高速で動作装置を切り離し、「**チップセット** (chip set)」と呼ばれるバス制御を行う専用 IC を間に入れることで処理速度の低下を防いでいます。

> **関連キーワード**：**フロントサイドバス** (Front Side Bus; FSB), **ユニバーサルシリアルバス** (Universal Serial Bus; USB), **PCI バス** (Peripheral Component Interconnect BUS) など。

以上のようなコンピュータの内部構造を踏まえた上で、コンピュータの動作の仕組、すなわちノイマン型コンピュータの特徴の 1 つである逐次制御方式について学習しましょう。逐次制御方式では、命令を 1 つずつ実行することで処理 (プログラム) を進めていきますが、「**命令取り出し段階**[13] (instruction cycle; fetch cycle)」$\Longrightarrow$「**命令実行段階** (execution cycle)」の過程を経て 1 つの命令を実行します。更に、命令実行段階では、「命令解読」$\Longrightarrow$「アドレス計算」$\Longrightarrow$「データの取り出し」$\Longrightarrow$「命令の実行」の詳細な過程を踏みます。これらをまとめると**命令サイクル** (instruction cycle) は図 4.45 のように 5 つの過程を経て 1 つの命令が実行されます。

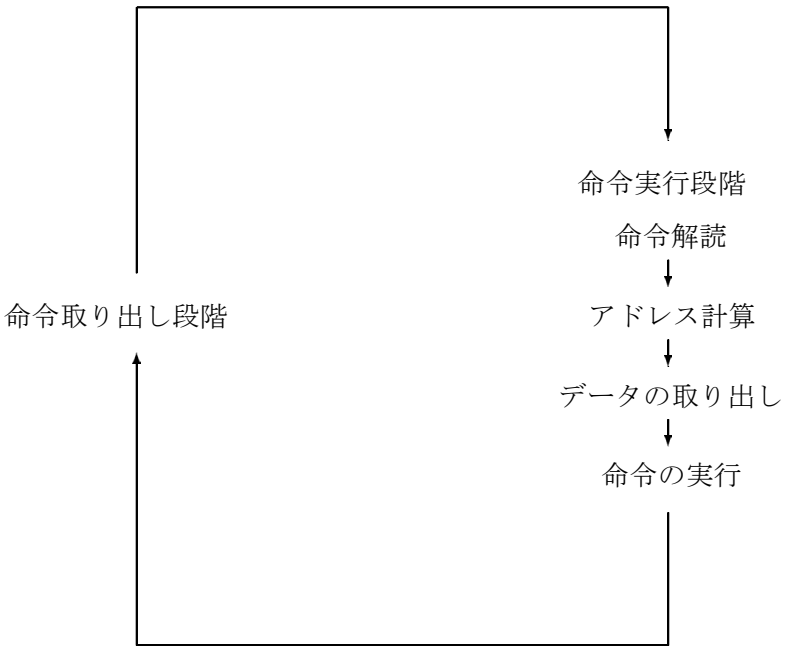

図 4.45: 命令サイクル

それでは、命令サイクルのそれぞれの過程を詳しく見ていきましょう。

[13] 単に**フェッチ**とも呼びます。

命令取り出し段階

① 命令アドレスレジスタの値をアドレス選択回路に渡す。

② アドレス選択回路は該当するアドレスを選択する。

③ 読み書き回路は選択されたアドレスにある命令を命令レジスタに渡す。

④ 命令の長さ (語長) l を命令アドレスレジスタに加える。
命令アドレスレジスタの値は次に実行する命令のアドレスになる。

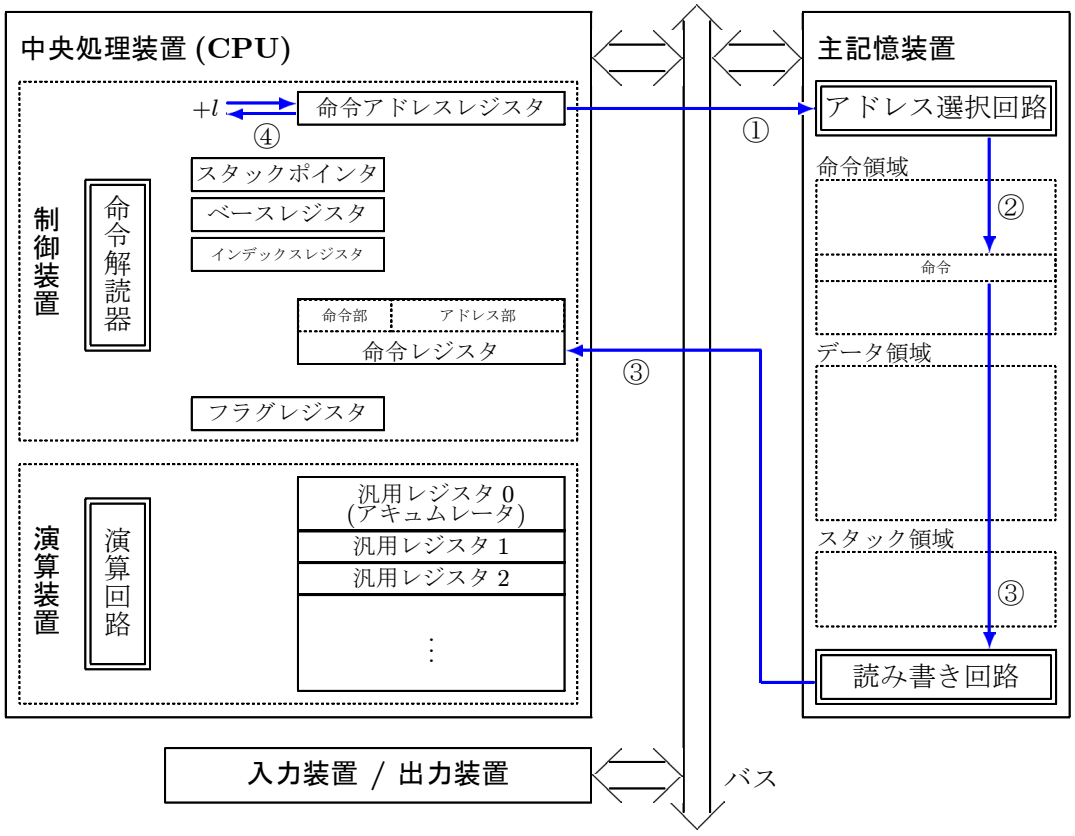

図 4.46: 命令取り出し段階

命令実行段階 (命令解読)

① 命令レジスタの命令部の値を命令解読器に渡す。

② 命令解読器は命令を解読し (使用する演算回路を選択する)、
演算回路に制御信号を送る (使用する演算回路に切り替える)。

命令実行段階 (アドレス計算)

③ 命令レジスタのアドレス部の値にベースレジスタやインデックスレジスタの値を加え
アドレス選択回路に渡す。このとき、使用する汎用レジスタの選択も行われる。
アドレス選択回路は該当するアドレスを選択する。

命令実行段階 (データの取り出し)

④ 読み書き回路は選択されたアドレスにあるデータを指定された汎用レジスタに渡す。

命令実行段階 (命令の実行)

⑤ 汎用レジスタの値を演算回路に渡し、演算結果を汎用レジスタに返す。
演算結果によってはフラグレジスタの値を書き換える。

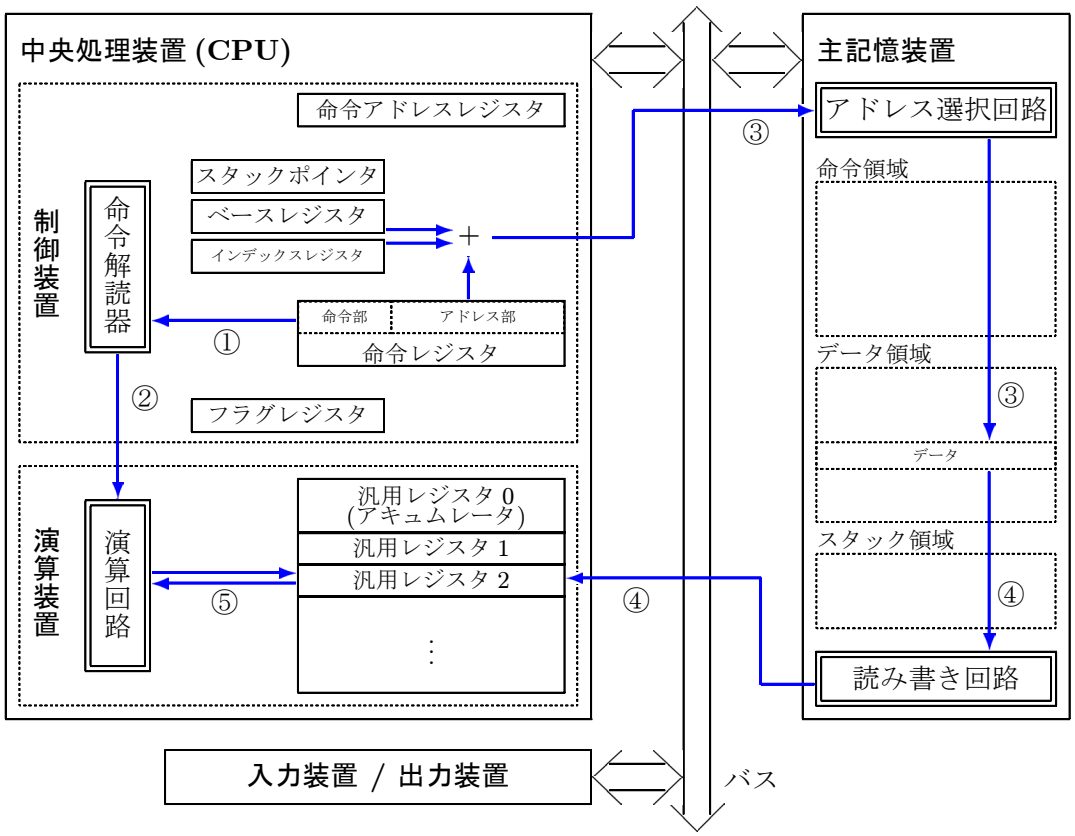

図 4.47: 命令実行段階

これまで、命令という抽象的な言葉を使って来ましたが、コンピュータが命令として直接理解できるのは**機械語**で、どのプログラム言語を使ってプログラムを作成しても最終的には機械語に変換され、コンピュータ上で実行されます。これから習う C 言語も例外ではありません。コンピュータが登場した当時は機械語を使ってプログラムを作成していましたが、コンピュータよりの言語 (**低レベル言語**) であったためプログラムの作成・変更・追加・削除が楽ではありませんでした。また、コンピュータの種類が異なるとあらためてプログラムを作成する必要がありました。その後、機械語と 1 対 1 に対応した**アセンブラ言語**が登場しました。この言語は、人間が理解しやすい記述方法やプログラムを実行する直前に機械語に変換するといった仕組みが取り入れられ、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仕様が異なるなど、欠点を残したままでした。その後も、コンピュータの進歩やプログラムの構文解析技術の発達などにより、様々なアイデアと改良が加えられ、次々と人間よりの新しい言語 (**高レベル言語**; **高級言語**) が誕生することになります (詳細は第 6 章)。

アセンブラ言語 (機械語) にはどのような命令があるのか見ておきましょう。例えば、学習目的で作られた COMET II という仮想コンピュータ上で動作する CASL II[14] というアセンブラ言語の命令は表 4.4 のようになっています。

命令の種類	命令の種類 (詳細)
ロード, ストア, ロードアドレス	ロード, ストア, ロードアドレス
算術演算, 論理演算	算術加算, 論理加算, 算術減算, 論理減算, 論理積, 論理和, 排他的論理和
比較演算	算術比較, 論理比較
シフト演算	算術左シフト, 算術右シフト, 論理左シフト, 論理右シフト
分岐	正分岐, 負分岐, 非零分岐, 零分岐, オーバフロー分岐, 無条件分岐
スタック操作	プッシュ, ポップ
コール, リターン	コール, リターン
その他	スーパバイザコール, ノーオペレーション

表 4.4: アセンブラ言語 CASL II の命令

[14] 独立行政法人 情報処理技術者センター (`http://www.jitec.jp`) のホームページから CASL II のシミュレータがダウンロードできます。