第6章

アセンブリ言語コードを組みあわせてデバイスにアクセスする

C言語による抽象化の範囲を超えて低レイヤのプログラムを書く方法

内田 公太

本章では，純粋なC言語でできることとできないこと，ユーザ・モードでできないことについて説明します．また，それらの制限を乗り越えてコンピュータを扱う方法を解説します．

純粋なC言語はメモリ上の演算しかできない

純粋なC言語でできるのはメモリ上の演算だけです．あるメモリ領域（変数）の値を読む，値同士を演算（四則演算やビット演算）する，演算結果をどこかのメモリ領域に書くという処理です．関数呼び出しや制御構文を使った分岐は可能ですが，それ自体ではプログラムの外側に何か影響を及ぼせるわけではなく，単にプログラムの実行位置が切り替わるだけのことです．

しかし，現実にはprintfを使えばターミナルに文字を表示できますし，fopenとfwriteを使えばファイルにデータを書くこともできます．ソケットAPIを使えばネットワーク送受信もできますし，何らかのGUIライブラリを使ってグラフィカルなアプリケーションを作ることだってできます．メモリ上の演算しかできないとは，いったいどういうことでしょうか．

OSもアプリケーションからの依頼を受けてメッセージを表示したり，ファイルの読み書きを行ったりします．OSは最終的にデバイス（機器）の制御を行います．ソフトウェアの世界から飛び出て現実世界に影響を及ぼしています．これらは全くメモリ上の演算には思えません．アプリケーションとOSはそれぞれのやり方で，C言語だけではできないことを実行します．

OSがデバイスにアクセスする方法

● OSの有無やOSの種類によらず大体同じ

OSがデバイスにアクセスする方法を説明します．ここでの話はOSに限らず，OSがない環境で動くソフトウェア（ベアメタル・プログラム）にも当てはまります．また，汎用OSと組み込みOSでそれほど差が出ない部分でもあります．もちろん，汎用OSで使われるデバイスと組み込みOSで使われるデバイスにはかなり差があるため，その部分の違いはありますが，デバイスのレジスタを読み書きすることによってデバイスを制御するという部分は同じです．

● デバイスは信号線によりCPUと接続している

コンピュータはさまざまなデバイスを持ちます．メモリ（RAM），マウスとキーボード，ディスプレイ，SSD，NIC，カメラなどです．メモリは普通，OSが起動した時点で初期設定が完了しており，CPUが自由にアクセス可能です．その他のデバイスについては，OSやファームウェアが持つデバイス・ドライバ（デバイスを制御するプログラム）が制御を担当します．

CPUが特別に制御回路を持つようなデバイスでない限り，デバイスはアドレス/データ・バスと制御信号線によりCPUと接続します[注1]．デバイスはアドレス・バスを監視して，自身が割り当てられたアドレスへの読み書きの場合に動作します．実際には，アドレスの監視と判定はデバイス本体の外部にあるコントローラが担うかもしれません．いずれにせよ，CPUはアドレスによってデバイスを区別し，必要な入出力を行います．

● 2つの方式…メモリ・マップトI/Oとポート・マップトI/O

デバイス・ドライバがデバイスにアクセスする方法には，デバイスをマップするアドレス空間の種類によって，2通りがあります．

- デバイスのレジスタがメモリ・アドレス空間にある場合，C言語の機能でレジスタの読み書きを行う（メモリ・マップトI/O[注2]）．
- デバイスのレジスタがI/Oアドレス空間にある場

注1：CPUの種類によって，アドレス・バスとデータ・バスが物理的に別の線だったり，同じ線を時分割で両方の用途として使う構成だったりします．

注2：「マップド」と表記されることが多いのですが，本記事では英語発音に近い「マップト」とします．