第1章 機械学習や量子コンピュータにも

今どきのコンパイラ

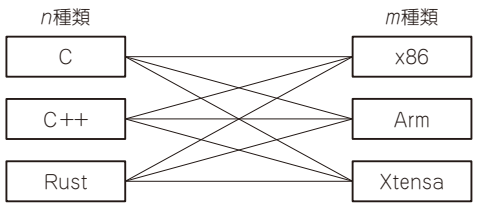
宮田 賢一

コンパイラはアプリケーションの開発に不可欠なツールです．基本的にはどんなプログラムもコンパイラを使ってコンピュータで実行できる形に変換されています．Pythonのような一見インタープリタ型の言語であっても，実は内部で仮想マシンのコード（Pythonの場合はPythonバイト・コード）にコンパイルされ，実行されています．

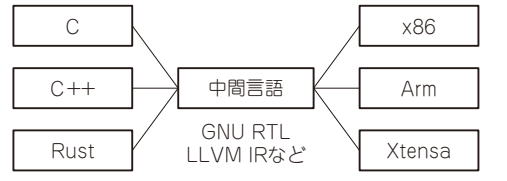
本章ではコンパイラとはどういうものなのかの概要と，コンパイラをめぐる動向を紹介します．

コンパイラは言語を変換するソフトウェア

コンパイラとはソースコードを機械語に変換するツールです．もう少し厳密にいうと，ある言語で記述されたプログラムを入力として，意味的に等価な別の言語のプログラムに変換して出力するソフトウェアです．C言語やC++言語のプログラムを入力として，x86やArmなど特定のCPUの機械語（という言語の）命令列を出力するものがイメージしやすいと思います．また，C++のプログラムをCのプログラムに変換するようなトランスレータもコンパイラの一種と言えます．



（a）ストレート型コンパイラ



（b）中間言語型コンパイラ

図1 多言語・多CPUコンパイラの構成

コンパイラの動向

● 多言語・多CPU対応が当たり前

最近のコンパイラは，複数のプログラミング言語を受け付け，複数のCPU用の機械語プログラムを生成できるのが当たり前です．C言語のプログラムをx86の機械語に変換するコンパイラのように，入力言語と出力CPUが1対1の作りになっている場合，プログラミング言語とターゲットCPUに特化した最適化を実装しやすいというメリットがある一方，n種類のプログラミング言語とm種類のCPU向けのコンパイラが必要になった場合は，$n \times m$種類のコンパイラを開発する必要があり，開発量が膨れます．しかし，入力のプログラミング言語を特定のCPUに依存しない抽象的な中間言語に一度変換した後，中間言語の上で各種最適化を施した上でターゲットCPUごとの機械語を生成する方式を取る作りにすることで，$n + m$種類のモジュールを作るだけで済み，多言語・多CPU化を実現しやすいものとなっています（**図1**）．GCCやClang/LLVMはまさにそのように作られています．

● コンパイラ界ではLLVMが最近の流行

オープンソースのコンパイラといえば長らくGNU Compiler Collection（GCC）がスタンダードです．今でもマイコン・ベンダが自前のマイコン・ボードやプロセッサ向けに提供するツールチェインの一部としてGCCが用いられることが多いです．

一方，新たなコンパイラ基盤としてLLVM1が2003年に登場しました．アップルのmacOSやiOSの開発ツールとして取り込まれたことも含め，対応するプログラミング言語やターゲットCPUの範囲を現在も拡大し続けています（**表1**）．