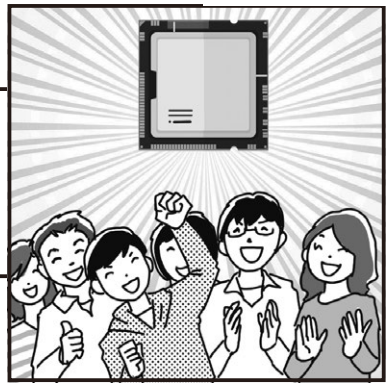

もう一度重要になる気がする

プロセッサ開発のセンス

第5回 ソフトウェアとハードウェアをつなぐ開発ツール

鈴木 勝博

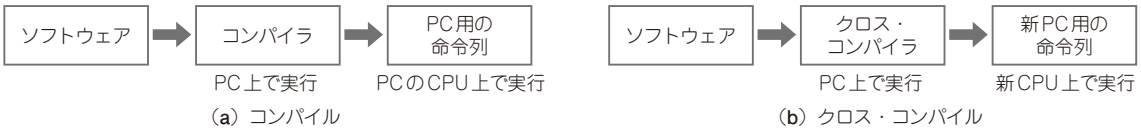

図1 通常のコンパイルとクロス・コンパイルの違い
コンパイラ(クロス・コンパイラと区別する文脈ではセルフ・コンパイラと呼ぶこともある)はPC上で実行し，PC用の命令列を生成する．クロス・コンパイラはPC上で実行し，新CPU用の命令列を生成する

第4回まではハードウェアのアーキテクチャ，設計の話が続きました．第5回は少し視点を変えてハードウェアを動かすために欠かせない，ソフトウェアの開発ツールの話をします．

新しいCPUを動かすために必須の開発ツール

ソフトウェアを新しいCPU上で動作させるには，そのCPU用の命令列を生成する必要があります．命令列はバイナリ・エディタでも書けますが，大規模なソフトウェアになると扱いきれません．通常はクロス・コンパイルといって，PCなど既存のアーキテクチャ上でソフトウェアを別のCPU向けの命令列に変換する作業を行います(**図1**)．

コンパイラを含むコンパイルに使用するツール群を開発ツールまたはツール・チェーンと呼びます．例えばGNUの開発ツールであれば，

- binutils：アセンブラ，リンカ
- GDB：デバッガ
- GCC：コンパイラ
- 標準Cライブラリ
  (glibcをはじめとして他ライブラリも使える)

からなります．クロス・コンパイル用であることを明示したいとき，ツール名の頭にクロスと付けることもあります．

開発ツールから見たRISC-Vの利点

CPU開発エンジニアにとって開発ツールの用意は悩み所でした．大企業ならまだしも個人や小規模開発で開発ツール，特にコンパイラのような複雑なソフトウェアをゼロから作成するのは大変だからです．

その点RISC-Vは著名なオープンソース(以降，OSS)コンパイラにて既に対応済みであり，商用コンパイラも対応を発表しています．この動きは今後も広がると予想されます(**表1**)．

自分自身で開発ツールをゼロから作成する必要のないRISC-VはCPU開発者にとって大きな利点と言えるでしょう．

クロス開発ツール構築の仕組み

● モジュールの種類

GNU開発ツールを構成するモジュールは4つあります．GCCは内部に幾つかライブラリを内包しています．

表1 RISC-Vに対応しているコンパイラ

OSS/商用	コンパイラ名	URL
オープンソース	GCC	https://gcc.gnu.org/
	LLVM	https://llvm.org/
商用	IAR Embedded Workbench	https://www.iar.com/jp/products/architectures/risc-v/iar-embedded-workbench-for-risc-v/