第2章

ラズパイ4のCortex-AとHiFive1の
RISC-V実行コードをWindowsで開発する

アセンブリ言語で書いたLチカ・プログラムを動かしてみる

中森 章

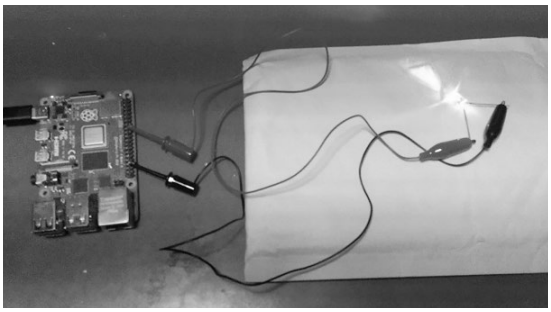

写真1　ラズベリー・パイ4でのLチカ・プログラムの実行

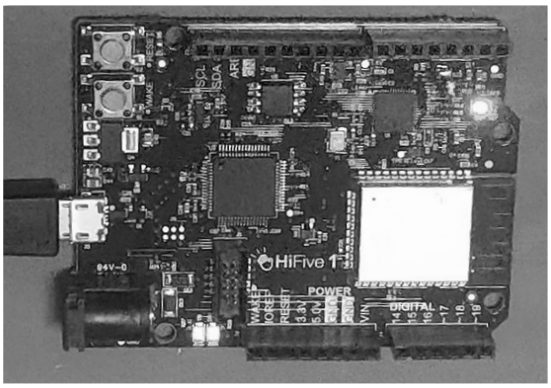

写真2　HiFive1のLチカの様子

本稿では，実際のコンピュータ・ボードで試せる環境を構築します．環境を構築したら，定番の「Lチカ」をアセンブリ言語でプログラミングして，動作確認します．　<編集部>

● ベアメタル環境でアセンブリ言語プログラミング

アセンブリ言語で実際のコンピュータ・ボードを動かしてみましょう．

ここでは，ラズベリー・パイ4でAArch64とAArch32のプログラムを，HiFive1（SiFive社）でRISC-Vのプログラムを動かしてみます（**写真1**，**写真2**）．もちろん，ベアメタル（OSなし）環境で，アセンブリ言語でプログラミングを行います．

ラズベリー・パイ4にしろHiFive1にしろ，セルフの開発環境でプログラムすることが普通だと思います．しかし，セルフのコンパイル（アセンブル）環境はLinuxで動作していることがほとんどです．ベアメタル環境でCPUを動かしたいのに，LinuxというOSの上でプログラミングを行うというのは気持ちが悪いです．それにLinux上でプログラングを行う場合は開発環境の構築にも一苦労です．ここでは，手っ取り早くCPUの動作確認を行うために，Windowsでプログラミング（クロス開発環境）を行います．もっとも，コンパイラ（アセンブラ）をCygwinやMinGWで動作させることになるので，Linux上での開発とどう違うのかという突っ込みがあるのは承知の上です．

プログラムの題材は，誰もが一度は通る道のLED点滅（Lチカ）です．

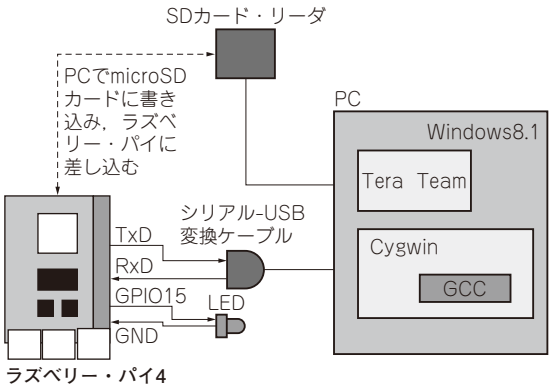

図1　Armのアセンブリ言語プログラムの開発環境（ラズベリー・パイ4を使用）

ラズベリー・パイ4でArmのコードを動かす

● 開発環境の構築

今回のラズベリー・パイ4の開発環境を**図1**に示します．Cygwin上でGCCによりアセンブルを行い，そ