第1章

レジスタへの数値設定と特定アドレスへの読み/書きがキモ

前知識…マイコンがI/Oする仕組み

中森 章

本稿では，マイコンのCPUコアとして広く普及しているArm Cortex-Aに加え，オープンソース・ライセンスの命令セット・アーキテクチャとして注目を集めているRISC-Vのアセンブリ言語を解説します．

今回は，誰もが一度は通る道のLチカ用プログラムをArmとRISC-Vそれぞれのアセンブリ言語で作成します．また，Lチカ用プログラムに使われる基本的な命令の意味について解説します．

加えて，今回紹介しきれなかった論理演算やサブルーチン，割り込み処理などの命令については次号以降で解説します．より実践的なアセンブリ言語のプログラムとして，UARTによる送受信も紹介します．〈編集部〉

● はじめに

▶本稿の目的

昨今，マイコンの低レイヤを習得するため，ベアメタルを学ぶ方がいます．これを分かりやすくいうと，OSなしの環境下で，アセンブリ言語を使ってプログラミングすることです．開発の効率を考えたらOSの上でC言語などのコンパイラ言語でプログラミングする方が楽なのですが，コンピュータの本質に深く触れるには，何といっても，OSなしのアセンブリ言語でしょう．

もっとも，アセンブリ言語を使用する真の理由は，コンパイラでは記述できない，まさに低レイヤに係るデリケートな記述をするためだったり，コンパイラによる最適化が気に入らない場合に最高速の命令コードの組み合わせを実現したりするためが多いと思います．

▶本稿の用語とアセンブラ記述について

RISC-Vはともかく，本稿では，Arm32，Arm64という表現を使用していますが，それらは正式名称ではありません．Arm32はArmv7-Aアーキテクチャのネイティブ(Thumbでない方の)モードの命令セット，またはArmv8-AアーキテクチャのAArch32モードの命令セット，Arm64はArmv8-AアーキテクチャのAArch64モードの命令セットを示します．本稿では直感的な理解のために，これらの表現をあえて混同して使用しています．

さらに説明では，アルファベットの大文字と小文字を適宜区別して使っていますが，アセンブラでは大文字と小文字は，ラベル名以外は，区別されません．例えば，レジスタ名がX0とかx0とかになっていますが，同じものとして見てください．

● マイコンが周辺デバイスを制御する仕組み

▶マイコンの内部構成

マイコンの構成要素は，CPUと命令が格納されたROM(やRAM)とデータを格納するためのRAMです(**図1**の左側)．CPUとROMとRAMがバスで接続されています．このとき，CPUはROMから命令を取り出し，命令を解釈して，その命令を実行します．命令によっては途中結果をRAMに保存したり，RAMに格納されているデータを参照したりします．

▶周辺ユニット

しかし，これだけでは命令の実行がマイコンの内部で完結していて，マイコンの外部には何の影響も与えません．実は，マイコンには外部のデバイスを制御するための周辺(ペリフェラル)ユニットが内蔵されています．最低でも，外部に対してディジタル信号の“H”レベルや“L”レベルを出力したり，外部からの“H”レベル入力や“L”レベル入力を取り込んだりするための数本の端子(GPIO)を備えています(**図1**)．

マイコンのCPUがマイコン外部のデバイスを制御するためには，周辺ユニットやGPIO(も周辺ユニットの1つだが)に対してコマンドを与えてやらなければなりません．このためには，周辺ユニットが備えているI/Oポート(レジスタと呼ばれる)に対して，書き込みを行ったり，読み出しを行ったりします．周辺ユニットによる処理が終了したか否かは，I/Oポートの値を読み出すことで判断します．

このような，周辺ユニットに対する書き込みや読み出しはRAMに対する操作と同一です．このとき，どうやってRAMと周辺ユニットを区別するのかというと，CPUから見えるアドレスで区別します．