第5章　「Hello World !」表示から乗除算命令を使う演算テストまで

# 俺々RISC-Vのプログラム作成

石原 ひでみ Hidemi Ishihara

前章のハードウェア実装編の最後では，コンパイル済みのファイルを埋め込んで動作を確認しました．ここではそのLチカ・プログラムの作成方法について解説します．またUARTを使って「Hello World !」を表示したり，今回実装したRISC-Vは乗除算命令にも対応しているので，乗除算命令が使われるような演算プログラムを作成して，その動作を確認してみます．

リスト1　スタートアップ・ルーチン（start.S）

```
    .section .text.init;
    .align 6;
    .globl _start;
_start:
    j reset_vector;
    .align 2;
trap_vector:
#    <<<トラップの内容>>>
handle_exception:
#    <<<割込みハンドラの内容>>>
other_exception:
#    <<<他のハンドラの内容>>>
reset_vector:
#    <<<初期化内容>>>
    auipc t0,0x0    # PCをレジスタに退避
    addi  t0,t0,16  # mretのアドレス
    csrw  mepc,t0   # MEPCにアドレスをセット
    call  main      # mainへジャンプ
    mret            # MEPCへ飛んでいく
```

## 1．まずはLチカ・プログラムから

既に前章の説明で，FPGA内のメモリにLED点灯制御プログラムを埋め込んで，動作確認を行っていま

リスト3　LED点灯制御プログラム（led.c）

```
#define GPIO_OUT (0x80000000+0x0100)
#define GPIO_IN  (0x80000000+0x0104)
#define GPIO_DIR (0x80000000+0x0108)

int main()
{
  int i;
  int led = 0;

  // GPIOの出力設定
  *(volatile int *)(GPIO_DIR) = 0x00000000F;

  while(1){
    *(volatile int *)(GPIO_OUT) = led;
                                  // GPIO(LED)へ出力
    for(i=0;i<2500000;i++);
                        // 約1秒周期で点滅するように調整
    led = ~led;
  }

  return 0;
}
```

リスト2　リンカ・ファイル（link.ld）

```
SECTIONS
{
  . = 0x00000000;
  .text.init : { *(.text.init) }
  .text ALIGN(0x800) : { *(.text) }
  .data ALIGN(0x800) : { *(.data) }
  .bss : { *(.bss) }
  _bss_end = .;
  end = .;
  _end = .;
}
```

す．しかしプログラムについての詳細説明をしていないので，ここで説明したいと思います．

● スタートアップ・ルーチンとメモリ配置のアドレス指定

RISC-Vの実行バイナリには，スタティックな構成でかつハードウェアの各種初期化の後，実際のソフトウェアを実行できるようにバイナリを構成しなければなりません．そのため，gccでコンパイルするプログラムは，_startで開始するようになっているので，各初期化は**リスト1**のようなスタートアップ・ルーチンを，アセンブラ・コードで用意します．初期化の内容としては主にレジスタの設定や，割り込みルーチンのアドレスや有効/無効などの設定となります．

また，プログラムの各セクションを配置するメモリ・アドレスを，明示的に指定する必要があります．リンカ用セクション情報を記述したリンカ・ファイルの内容を**リスト2**に示します．

● LED点灯制御プログラム

**リスト3**にLED点灯制御プログラムを示します．

ソースコードの先頭で，GPIO制御レジスタのI/Oアドレスを定義しています．プログラムの実行はmain()から開始します．まずGPIOの下位4ビットの入出力方向を，出力方向に設定します．

次にループ処理に入り，GPIOの出力レジスタに変数ledの内容を出力します．少し待ち時間（ウェイト）