第5章 GDBデバッガによるブレークポイント設定やステップ実行中の挙動を見てみる



# アセンブリ言語でひも解くデバッグ中のCPUの動作

永井 健一

デバッグ時のCPUの挙動について解説します．組み込みプログラマの方は，GDBなどを使ってプログラムをデバッグしていると思います．このときに，ブレークポイントを設定して，プログラムを停止してステップ実行などを行っているでしょう．このとき，デバッガとCPUはどのような動きをしているかを説明します．

## デバッガが裏で行っている操作

デバッグでは次のような流れに沿って作業を進め，4～6を繰り返し行います．

1. プログラム読み込み
2. ブレークポイント設定
3. プログラム実行
4. ブレークポイントで一時停止
5. 変数などをウォッチ
6. 続けてプログラムを実行

それぞれの工程において，GDBのようなデバッガが内部的に何を行っているかを説明しながら，なぜ設定したブレークポイントでCPUが止まるのかを解説します．

### ● 1：プログラム読み込み

gdbコマンドは，デバッグ対象のプログラムを読み込んで，ソースコードのファイル名と，ソースコードの行番号とコードのオフセットを取得します．

### ● 2：ブレークポイント設定

breakコマンドを使って，プログラムを停めたい箇所にブレークポイントを設定します．GDBは指定された行数または関数名から，ブレークすべきアドレスを認識します．ただ，この時点では，まだデバッグ対象のプログラムをプロセスとして起動していないので，プロセスとしてのアドレスは決定していません（**リスト1**）．

### ● 3：プログラム実行

runコマンドを使ってプログラムを走らせ，ブレークポイントまで実行させます．デバッグ対象のプログラムをプロセスとして起動し，このときブレークすべき実際のアドレスが決定されます（**リスト2**）．

ここでGDBは，ブレークすべきアドレスにある命令を，例外処理を引き起こすBRK命令に置き換えます．この命令をCPUが実行すると，例外処理が実行されます．Linuxのユーザ・アプリケーションとして

**リスト1　breakコマンドによるブレークポイントの指定**

```
(gdb) break bp_func
Breakpoint 1 at 0x85c: file main.c, line 7.
(gdb) info breakpoints
Num     Type           Disp Enb Address            What
1       breakpoint     keep y   0x000000000000085c in bp_func at main.c:7
```

**リスト2　runコマンドで対象プログラムを実行させる**

```
(gdb) break main
Breakpoint 2 at 0x874: file main.c, line 16.
(gdb) run
省略
(gdb) info breakpoints
Num     Type           Disp Enb Address            What
1       breakpoint     keep y   0x000055555555085c in bp_func at main.c:7
2       breakpoint     keep y   0x0000555555550874 in main at main.c:16
        breakpoint already hit 1 time
```