第3章 V_R4131 搭載 DIMM 形状 CPU モジュールを利用

MIPS による最小構成 Linux システムの構築事例

伊藤 博

ここでは CPU として MIPS アーキテクチャである V_R4131 を搭載した小型 CPU ボードを使って，Linux システムの構築事例について解説する．第 2 章と同じように，フラッシュ ROM と SDRAM のメモリ，そしてコンソール入出力としてのシリアルだけをもった，最小構成ボードとなっている．CPU アーキテクチャが違っても，基本は同じであることが分かる．（編集部）

● ターゲット・ハードウェア

今回使用したターゲット・ハードウェアは，DIMM 形状のソケットに挿入して使う CPU モジュール TB0229（タンバック製）です．この CPU モジュールには V_R4131（NEC エレクトロニクス製）が搭載されています．V_R4131 の特徴として，200MHz 動作で処理性能が 300MIPS 以上，最大消費電力は 220mW と性能と消費電力のバランスがよく，PCI コントローラも内蔵しているため拡張性もよい点が挙げられます．

この CPU は CASSIOPEIA やシグマリオン II などの PDA（Personal Digital Assistants）で採用されましたが，組み込み向け CPU としてもよく使われています．V_R4131 DIMM には，4M バイトのフラッシュ ROM と，64M バイトの SDRAM が搭載されており，3.3V 単一電源で動作します．

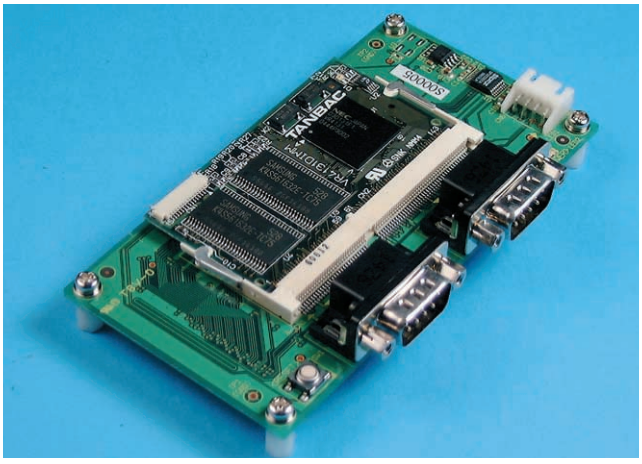

写真 1　V_R4131 DIMM（TB0229 ＋ベース・ボード）の外観

入手先：メディアラボ（株）http://www.mlb.co.jp/

この CPU モジュールを第 2 章と同一のベース・ボードと組み合わせた形で使用します（**写真 1**）．

今回は，4M バイトのフラッシュ ROM だけで動作する Linux システムを構築します．メモリ容量が小さいので，glibc ではなく組み込み用に小さくした uClibc という C ライブラリを利用します．

1．クロス開発環境の準備

● ライブラリの問題

最初に必要になるのはクロス開発環境です．

Linux カーネルやブート・ローダなどの標準 C ライブラリを使わないものであれば，割と簡単にクロス・コンパイラを作成できるのですが，ユーザ・アプリケーションまで作成できる環境を構築するには，結構面倒な作業になります．その理由としては以下の 2 点が挙げられます．

1）crt0 は libc の付属物を使わないといけない

crt0 は C 言語のランタイム・ルーチンで，`main()` を呼び出す以外にも，ダイナミック・ローダ（共有ライブラリを読み込むための共有ライブラリ）の読み込みを行ったり，終了処理（atexit の実行）などを行ったりします．これは OS および C ライブラリに強く依存します．そのため，クロス・コンパイラを作るには，先に libc をコンパイルしないといけません．

2）soft-float で作成したオブジェクトとそうでないものを混ぜてはいけない

V_R4131 に FPU は搭載されていません．FPU 命令を実行しようとすると Linux カーネルが無効な命令としてトラッ

プし，命令をエミュレートします．しかし，この機能は1命令ごとにトラップしてしまうため，実行効率が悪くなります．コンパイル時に，--soft-floatのオプションを付けることで，FPU命令を使わないようにする指定もできます．しかし，--soft-floatオプションを付けてコンパイルしたオブジェクトと，--soft-float指定せずに生成したオブジェクトを混在させることはできません．従って，--soft-floatを指定するなら，libcまで含めたすべてを一貫して設定する必要があります．

つまり，仮に手元にターゲットCPUと同じアーキテクチャであるMIPS CPUを搭載したワークステーションなどのホスト・マシンがあったとしても，glibcベースのマシンであれば，やはりクロス環境を作成する必要があるのです．

uClibcのbuildroot（http://buildroot.uclibc.org/）を利用すると，簡単にクロス環境を構築できるだけでなく，ルート・ファイル・システムの雛型まで一気に作ることができます．

buildrootで配布しているソースは，Makefileとパッチのみです．つまり，「こうすればクロス環境が作れる」，「このパッケージはこうすればuClibcを使って作成できる」というやり方をまとめたものです．

buildrootは何種類ものコンパイラのバージョンや色々なCPUをサポートし，作成できるパッケージもたくさんあります．日々更新されているため，ビルドが正しくできることの方がまれで，問題解決の能力も必要です．しかし，こうしたら作成できるという手本として考えるなら，とても有用なものだと思います．

なお今回の内容は，2007年6月中旬の時点でリリースされているDebian etchの環境で動作確認を行っています．

● インストールと設定

必要になりそうなものを最初にインストールしておきます．

```
# aptitude install build-essential g++
subversion ncurses-dev bison flex gettext
git-core wget cramfsprogs
```

もし足りなければその時点で必要なものがないというメッセージが出るので，あまり気にする必要はありません．少なくとも上記は必須なので，先に入れておきましょう．

各パッケージのソースは必要になったものを自動的にwgetコマンドでとってきます．wgetが正しく動くことを確認してから始めてください．ディスク容量として1.5Gバイト程度を使用します．

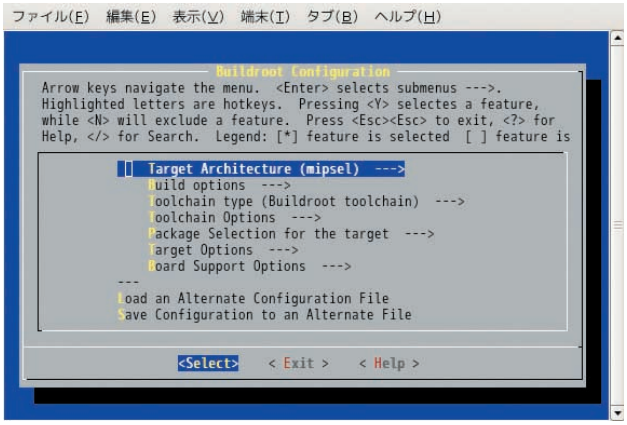

図1　ターゲットの設定

buildrootは，Subversion（ソース・コード管理システム）で管理されています．下記からソースをとってこれます．

```
$ svn co svn://uclibc.org/trunk/
buildroot
```

次に最小限の設定を行います．

```
$ cd buildroot
$ make menuconfig
```

として，「Target Architecture」で「mipsel」を選択します（**図1**）．

「Build options」の「Toolchain and header file location?」で，クロス・コンパイラのインストール場所を選択できます．クロス・コンパイラに標準ヘッダやライブラリの検索パスが埋め込まれるので，一度作成すると，後から場所を変更することはできません．後で参照しやすいように，ここでは/opt/mipsel_nofpuを選択しました．

「Toolchain Options」で「Use software floating point by default」を選択しておきます．クロス・コンパイル時のデフォルトになるので，--soft-floatを付けなくても有効になります．

「Target Options」で「ext2 root filesystem」から「tar the root filesystem」に変更し，「Compression method」をgzipにしました．

以上の設定で，あとはデフォルトのまま作成します．

次にインストール先を作っておきます．

```
# mkdir /opt/mipsel_nofpu
# chown ito.mlb /opt/mipsel_nofpu
```