SimMc の使い方解説

このページは、東工大吉瀬研究室で開発・配布されているメニーコアーキテクチャ M-Core のシミュレータ SimMc を使うためのガイドです。

注意 !! このページは公式の配布サイトとは一切関係ありません。従って、このページの内容についての質問その他は、吉瀬研究室に連絡しないでください。

必要なもの

Linux 環境、そして SimMc とクロスコンパイラが必要です。次の 2 つのうちいづれかの方法で環境を用意してください。

(1) VirtualBox のイメージを利用する

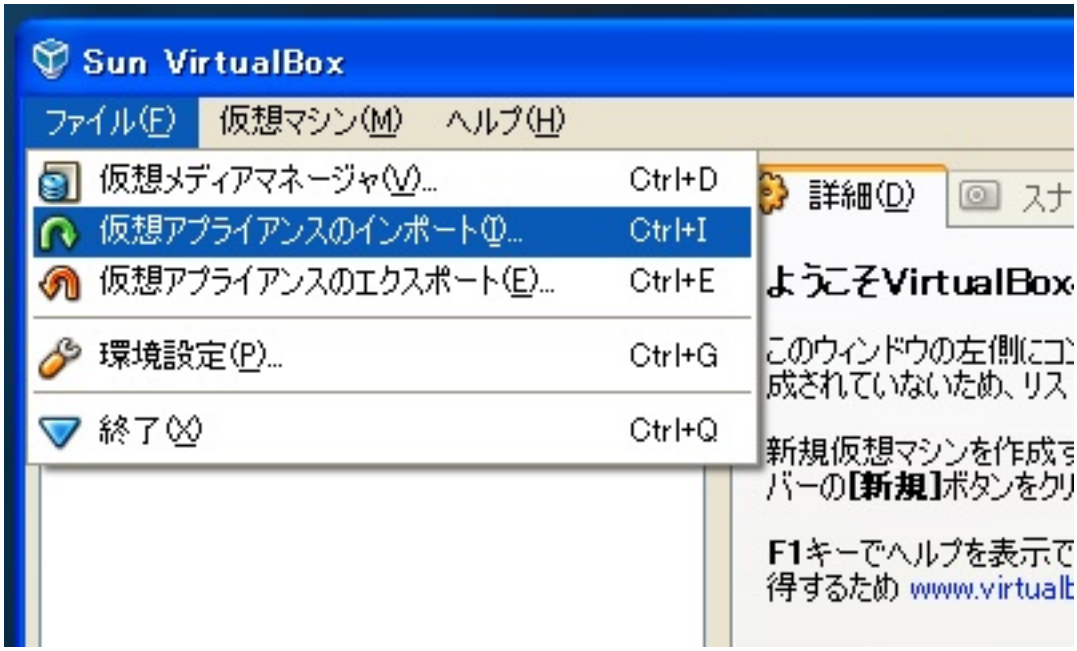
Ubuntu-10.04 をベースに、SimMc およびクロスコンパイラ一式をインストールしたイメージを用意しました。

VirtualbBox

http://jp.sun.com/products/software/virtualbox/get.html
Windows の一般的なインストール同様ダイアログに従う

SimMc 環境構築済み VirtualBox 環境

SimMcDev-1.0.zip(1145607238) をダウンロード
(MD5 (SimMcDev-1.0.zip) = a48c7353862399f2edb90d2840f73e6b)
1. ファイルを解凍する
注意 !!「デスクトップ」のような日本語ディレクトリでは次のインポートができません。c:\temp などに展開してください。
2. VirtualBox を起動し、アプライアンスのインポートを行う
Windows の場合、メニューの「ファイル」→「仮想アプライアンスのインポート」です。



(2) 自分でコンパイルする場合

SimMc

http://www.arch.cs.titech.ac.jp/mcore/

- Ubuntu10.04 でコンパイルする場合にはパッチをあててください

mipsel クロスコンパイラ

自分で作るにしても buildroot で作成するのが簡単。
mipsel-linux-gccc で、uclibc が必要です。

SimMc インストール済み Linux の起動と終了の仕方

VirtualBox 上での Linux 環境の起動と終了の仕方を覚えておきましょう。

起動

■ 1. VirtualBox を起動し、メニューから SimMcDev をダブルクリックします。

起動中に VirtualBox がダイアログメッセージを表示する場合がありますが「OK」で進んでください。

■ 2. ログイン

画面では、ユーザ simmc を選択。パスワードは simmc です。

終了

■ 1. メニューバー右端の電源ボタンのアイコンをクリック

■ 2.「Shut Down...」を選択

SimMc の起動

早速 SimMc を起動してみましょう。SimMc 環境 VirtualBox イメージの場合、SimMc は /home/cad/SimMc/sim にインストールされいて、パスが通っています。

■ 1. ターミナルを起動します。

メニューバーの「Applications」→「Accessories」→「Terminal」を辿りクリック

■ 2. プロンプトで SimMc と入力

SimMc が起動。ヘルプメッセージの表示を確認しましょう。

Hello M-Core

SimMc では普段のコンソールプログラミングと同様に標準出力関数を使って、文字列を表示させることができます。まずは、おきまりの Hello World を実行してみましょう。

```
#include <stdio.h>
int main(int argc, char **argv)
{
   printf("Hello M-Core\n");
}
```

■ 1. 好きなエディタでプログラムを書く。

例えば「Applications」→「Accessories」→「gedit Text Editor」で、ノートパッドライクなエディタが起動します。ここでは、test.c という名前で保存することにします。セーブ先のディレクトリはホームディレクトリのまま、その他のオプションもデフォルトのままでいいです。

■ 2. コンパイルする

クロスコンパイラでコンパイルします。Terminal から下記のコマンドを実行してください。

```
mipsel-linux-gcc -static test.c
```

ここで、"-static" をつけ忘れると、ただしくプログラムが実行できませんので注意してください。

■ 3. 実行する

コンパイルしてできあがったプログラムを SimMc で起動してみましょう。Terminal で以下のコマンドを実行します。

```
SimMc a.out
```

"Hello M-Core" が 16 個表示されましたか？実は SimMc にオプションをつけずに実行すると、4x4 =16 個のコアを持つプロセッサとして起動します。ですから 16 回表示されるのですね。

MClib を使うプログラムをコンパイルする

SimMc/M-Core では、MClib で定義された API を使用して通信を行うことができます。それでは、早速 MClib を使ったプログラムのコンパイルの仕方をみていきましょう。
次の例は、SimMc のサンプル test30/main.c で、core(rank_x, rank_y) から core(1,1) にデータを送るプログラムです。ここで、rank_x,rank_y は、一番 ID の大きなコア、つまり 4x4 の構成であれば core (4, 4) にあたります。

```
/*************************************************************************/
/* Many-Core Architecture Research Project       Arch Lab. TOKYO TECH */
/*************************************************************************/
#include "MClib.h"
volatile int buf;
/*************************************************************************/
int main(int argc, char *args[])
{
    int id_x, id_y, rank_x, rank_y;
    MC_init(&id_x, &id_y, &rank_x, &rank_y);
    buf = 0;
    if (id_x == rank_x && id_y == rank_y) {
        MC_sleep(10000);
        int dst = 0;
        setidxy(&dst, 1, 1);
        MC_dma_put_4b(dst, (int*)&buf, 777);
    }
    if (id_x == 1 && id_y == 1) {
        while(buf == 0);
        printf("buf %d (this must be 777)\n", buf);
    }
    MC_finalize();
    return 0;
}
/*************************************************************************/
```

上のファイルを test2.c として保存したとして、先ほどのようにコンパイルしてみましょう。

```
mipsel-linux-gcc -static test2.c
```

コンパイルすると、残念ながら、次のようにエラーが表示されてしまいます

```
main.c:4:19: error: MClib.h: No such file or directory
main.c: In function 'main':
```

これは、ソースコード中で include しているヘッダファイルである MClib.h をコンパイラがみつけられずに失敗していることを意味しています。gcc ではコンパイル時にヘッダファイルを探す場所を追加する場合に、-I(念のためですが、これは大文字のアルファベット、アイです) というオプションを用います。VirtualBox のイメージを使用している場合は、/home/share/cad/SimMc の下に MClib.h が格納されていますので、

```
mipsel-linux-gcc -I/home/share/cad/SimMc/lib -static test2.c
```

としましょう。もちろん自分で SimMc を展開した場合には、そのディレクトリを指定してください。
しかし、残念ながら、まだ不完全です。次のようなエラーが発生してしまうと思います。

```
tmp/ccKNBYVA.o: In function `main':
main.c:(.text+0x94): undefined reference to `MC_sleep'
main.c:(.text+0xbc): undefined reference to `setidxy'
main.c:(.text+0xe4): undefined reference to `MC_dma_put_4b'
main.c:(.text+0x164): undefined reference to `MC_finalize'
collect2: ld returned 1 exit status
```

これは、オブジェクトファイルを作成しようとしたときに MClib の関数のオブジェクトがみつからないというコンパイル過程のリンカでのエラーです。これは、あらかじめコンパイルしておいた MClib の中間コードをコンパイル時の引数として一緒に指定することで解決することができます。

```
mipsel-linux-gcc -I/home/share/cad/SimMc/lib -static /home/share/cad/SimMc/lib/MClib.o  test2.c
```

これでばっちり a.out が作成されます。
不幸にも

```
mipsel-linux-gcc: /home/share/cad/SimMc/lib/MClib.o: No such file or directory
```

というエラーが発生した場合には、

```
cd /home/share/cad/SimMc/lib
make
```

として、MClib.o をコンパイルしてください。
なお、配布されている SimMc で用いられるクロスコンパイラは /home/share/cad/mipsel/usr/bin/mipsel-linux-gcc が想定されていますので、自分の環境にあわせて適宜変更してください。

サンプルプログラムを実行してみよう

/home/share/cad/simmc/app/test の下に、サンプルプログラムがあります。ためしに、実行してみましょう。

```
SimMc /home/share/cad/simmc/app/test/test30/test.out
```

サンプルプログラムを読むと M-Core のライブラリである MClib の基本的な使い方がわかります。

もっと速くシミュレーションしたい!

・仮想マシンではなく生の Linux 環境を構築して SimMc を動かす
・FPGA によるシミュレータ ScalableCore を使う :-)