計算機システム実験 K-13 組み込みオペレーティングシステム

http://www.coins.tsukuba.ac.jp/~kumikomios/

実験は 3C205 の iMac を用いて行います．以下，情報学類の iMac を前提として GBA のプログラミングを行う開発環境について説明します．

1 開発環境

GBA では ARM 7TDMI という小型組み込み機器用プロセッサが使用されています．これは情報学類の iMac や Windows PC で使用されている Intel IA-32（x86）系列のプロセッサとはまた異なる命令セットを持つプロセッサです．そのため，GBA で実行できるプログラムをコンパイル，リンクするためには，ARM プロセッサ用のコンパイラ，リンカを使用する必要があります．ARM プロセッサ用のコンパイラ，リンカは /home/lecture/kumikomios/bin 以下にあります．以下のようにしてパスを通すと使用できるようになります．

```
% export PATH=/home/lecture/kumikomios/bin:$PATH
```

ログインする度に設定するのが面倒な場合は，以下のように .bashrc に入れておくこともできます．一番下の行が追加する行です．

```
#
# coins standard ~/.bashrc
#
if [ -f /usr/local/lib/standard/bashrc ]; then
        . /usr/local/lib/standard/bashrc
fi

# add your own code below
export PATH=/home/lecture/kumikomios/bin:$PATH
```

ログインシェルとして tcsh を使用している場合は、以下のように .cshrc に入れます．一番下の行が追加する行です．

```
#
# standard ~/.cshrc
#
if ( -f /usr/local/lib/standard/cshrc ) then
        source /usr/local/lib/standard/cshrc
endif
set path = (/home/lecture/kumikomios/bin $path)
```

ARM 用のコンパイラ等のコマンドには全て-arm が後ろに付きます．

```
% ls /home/lecture/kumikomios/bin
addr2line-arm   cpp-arm     gcov-arm        nm-arm      ranlib-arm   strip-arm
ar-arm          elfedit-arm gprof-arm       objcopy-arm readelf-arm
as-arm          gcc-arm     ld-arm          objdump-arm size-arm
c++filt-arm     gccbug-arm  libusb-config   optusb      strings-arm
```

この実験では gcc-arm を Intel プロセッサで実行し ARM プロセッサで実行できるプログラムを作成します．このように異なったプロセッサのプログラムを開発するための環境を「クロス開発環境」と呼び，そのためのコンパイラを「クロスコンパイラ」と呼びます．

2 GBA プログラムのアセンブル

実際に GBA 用にプログラムをコンパイルし，実行してみましょう．適当なディレクトリ（EmbOS とか kumikomios とか)を作り, そこへ cd します．そして/home/lecture/kumikomios/CDROM/chapter-1/rgb.s をコピーします．また， /home/lecture/kumikomios/CDROM/chapter-2/gcc.ls もコピーします．

```
% mkdir EmbOS
% cd EmbOS
% cp /home/lecture/kumikomios/CDROM/chapter-1/rgb.s .
% cp /home/lecture/kumikomios/CDROM/chapter-2/gcc.ls .
```

rgb.s は，本（貸し出し資料：Linux から目覚めるぼくらのゲームボーイ！）の 18 ページに掲載されているリスト 8 のアセンブラプログラムで，液晶画面を赤（Red）緑（Green）青（Blue）の 3 色で塗り分けるというものです．gcc.ls はリンカスクリプトというもので，プログラムの実行開始アドレスを GBA に合わせるために使用します．[*1]

アセンブル（アセンブラプログラムですのでコンパイルとは言わずアセンブルと言います)，リンク，そして GBA にロード，実行できる形式への変換は以下のようにします．

```
% cpp-arm rgb.s | as-arm -o rgb.o -
% ld-arm -o rgb.out -T gcc.ls rgb.o
% objcopy-arm -O binary rgb.out rgb.bin
```

cpp でアセンブラプログラムに含まれるマクロやコメントを処理し，パイプで出力された結果を as-arm でアセンブルします．as-arm への入力はパイプを通して標準入力で入ってくるので，ファイル名の代わりに-が指定されています．アセンブル結果は-o オプションの後に指定された rgb.o に書き出されます．ld-arm で実行形式にリンクします．(1 つしか.o ファイルがないので何ともリンクしていませんが...）リンク時には-T gcc.ls と，コピーしてきたリンカスクリプトを指定します．objcopy-arm は-O binary オプションを付けることで，GBA で実行できるように実行形式ファイルを変換します．

[*1] gcc.ls の中身は本の 30 ページ，リスト 5 にあります．リンカスクリプトの説明は 19 ページ，24 ページ，30 ページ，83 ページにあります．

3　プログラムの GBA への転送，実行

3.1　準備

準備として，貸し出し機材の USB 接続ケーブルで GBA を C205 の iMac に接続し，プログラムを転送，実行できる状態にしましょう．本の 4 ページにも解説があります．

1. USB 接続ケーブル（本ではブートケーブル USB）を iMac の USB コネクタに接続
   iMac の USB コネクタはキーボードの左横側にあります．また，ディスプレイの右側裏にもあります．
2. 電源 OFF，ゲームパックなしの状態の GBA を USB 接続ケーブルに接続
3. GBA の電源を入れる（電源スイッチは右横）
4. 任天堂のロゴマークがサウンドとともに現れた後に，画面全体が白色になるまで待つ

この画面が白色になっている時が，GBA はプログラム転送を待っている状態です．

3.2　転送，実行

GBA に転送，実行できるファイルとしてできた `rgb.bin` を，GBA に転送するには `optusb` コマンドを使用します．以下のように `rgb.bin` を引数に指定して `optusb` コマンドを実行すると `rgb.bin` が GBA に転送され，自動的に実行は始まります．

```
% optusb rgb.bin
=== optusb v1.01+darwin rel.4 ===
Source file = rgb.bin
Program size = 100
Status = Successfully transferred.
```

GBA の液晶画面が赤緑青の 3 色で塗り分けられたら成功です．

3.3　再転送，実行

プログラムを再転送する場合は

1. GBA の電源を切る
2. ケーブルの USB コネクタから抜く
   GBA とケーブルは接続したままにしておいてください．

そして，準備の手順の 1 から繰り返します．

この手順を踏まないと，以下のようなメッセージが表示され，プログラムの転送に失敗してしまいます．

```
% optusb rgb.bin
=== optusb v1.01+darwin rel.4 ===
Source file = rgb.bin
Program size = 100
Status = Transmission error!
Detail = send_file: File data transmission error
```

4 C プログラムのコンパイル

C プログラムは main 関数から実行が始まりますが，実際の実行形式のバイナリプログラムはちょっと違ったところから実行が始まります．crt ライブラリがプログラムの先頭に配置され main 関数を呼び出すようになっています．[*2]

/home/lecture/kumikomios/CDROM/chapter-2/crt.S が crt ライブラリのもとになるファイルですので，これをコピーしてきます．また，/home/lecture/kumikomios/CDROM/chapter-2/rgb.c は画面を 3 色で塗り分けるプログラムを C で書いたものですが，これもコピーしてきます．

```
% cp
/home/lecture/kumikomios/CDROM/chapter-2/crt.S .
% cp /home/lecture/kumikomios/CDROM/chapter-2/rgb.c .
```

crt.S，rgb.c から GBA に転送，実行可能なファイルを作るには次のようにします．ld-arm の引数に指定している crt.o rgb.o の順番には意味があり，crt.o が先頭に来ていなければなりません．

```
% gcc-arm -c crt.S
% gcc-arm -c rgb.c
% ld-arm -o rgb.out -T gcc.ls crt.o rgb.o
% objcopy-arm -O binary rgb.out rgb.bin
```

optusb コマンドで rgb.bin を GBA に転送，実行してみましょう．

5 Makefile と make コマンド

コンパイルの度に上記の一連のコマンドをタイプするのは面倒ですので，make コマンドにコンパイルさせましょう．以下の Makefile は，crt.S，rgb.c から rgb.bin を作成するまでの一連のコマンド実行を自動化してくれます．

[*2] crt は C Run-Time の略です．もう少し詳しい crt ライブラリの説明は本の 29 ページにあります．

```
all: rgb.bin

crt.o: crt.S
        gcc-arm -c crt.S

rgb.o: rgb.c
        gcc-arm -c rgb.c

rgb.bin: rgb.o crt.o
        ld-arm -o rgb.out -T gcc.ls crt.o rgb.o
        objcopy-arm -O binary rgb.out rgb.bin

clean:
        /bin/rm -f *.o *.bin *.out
```

Makefile の，実行するコマンドが書かれている行の先頭は TAB でインデントされている必要があることに注意してください．

make を実行すると次のようになります．

```
% make
gcc-arm -c rgb.c
gcc-arm -c crt.S
ld-arm -o rgb.out -T gcc.ls crt.o rgb.o
objcopy-arm -O binary rgb.out rgb.bin
```

また，make clean とするとソースファイル以外のファイルを消去してくれます．

```
% make clean
/bin/rm -f *.o *.bin *.out
```