

# Gears OS on Raspberry Pi

145759E 氏名 桃原優 指導教員：河野 真治

## 概要

In our laboratory doing development CbC and Gears OS make use of CbC.

CbC can be write meta level processing and parallel processing. Meta level processing can management memory , thread, CPU and GPU. In this research aiming to run Gears OS on Raspberry Pi

## 1 Raspberry Pi 上での Gears OS

当研究室では CbC(Continuation base C) と CbC を用いて実装する GearsOS の研究を行っている。

CbC は Code Segment と Data Segment という単位でプログラムを記述する。Code Segment は並列処理の単位として利用でき、Data Segment はデータそのもので型を持っていて、CbC はメタレベルの処理、並列処理を記述することができる。

メタレベルの処理では、メモリ管理、スレッド管理、CPU や GPU の資源管理を記述することができる。

本研究では、ARM で動くシングルボードコンピュータである Raspberry Pi 上で Gears OS を動かせるようになる事で、ハードウェア上でもメタレベルの処理、並列実行ができるプログラミングを記述できるようになる事を目指している。

しかし、メモリの関係上 RaspberryPi 上で CbC の make を行うと、かなりの時間がかかる。

解決案として、OSX 上で CrossCompile を行う方法と、Linux 環境で CrossCompile を行う方法を提案する。

## 2 Raspberry Pi 上での実装

Raspberry Pi 1 のメモリは 256MB と小さいため、CbC を make することができない。Raspberry Pi 3 だとメモリは 1GB あり CbC を make できるが、時間がかかる。

make 時間の比較として研究室のサーバでメモリ 16GB の firefly と学科のサーバの一つで Linux 環境であるメモリ 756GB の DALMORE を用いる。なお、firefly と DALMORE では Google によって開発された build システムの ninja-build を用いて make を行なった。その結果を表 1 に示す。

## 3 CrossCompile

CrossCompile を行うことで make 時間の問題を解決する方法がある。CrossCompile とは、別の OS で実行可能なコードを生成するコンパイルの手法である。Raspbian 以外

| 環境 (メモリ)         | 時間              | 比較       |
|------------------|-----------------|----------|
| RaspberryPi(1GB) | 15 時間 11 分 06 秒 | 1.00 倍   |
| OSX(16GB)        | 2 時間 16 分 06 秒  | 6.69 倍   |
| Linux(756GB)     | 2 分 26 秒        | 374.42 倍 |

表 1: make 時間の比較

の OS 環境であらかじめ Raspberry Pi で CbC が動くように CrossCompile を行い、そのコードを Raspberry Pi に移す事で、実行できるようになる。

Raspberry Pi の OS である Raspbian は qemu によるメモリの拡張もできないので、別の手法で Raspberry Pi 上に CbC を実装する方が好ましい。

マサチューセッツ工科大の大学院生向け講義の教材として使うために、UNIX V6 という OS を ANSI-C に書き換え、x86 に移植した Xv6 という OS がある。Xv6 は Raspberry Pi に移植する事ができる。ANSI-C で書かれている Xv6 を CbC に書き直す事で、Raspberry Pi で CbC を動かせるようになる。

CbC を Raspberry Pi で動かすためのアプローチの手法を、I と T の形をした図の組み合わせによって説明を行う。I の上部分に cbclang や Xv6 などのソースコード名を、下部分にその機能の記述言語を記してある。T の下にある I は特別で、上に VM 下に VM を乗せている OS が記されている。T の上部分は左に入力されるファイル、右に出力されるその機能によって出力されるファイルが記され、下部分にその機能の記述言語が記されている。

例として、cbclang のソースコード (I) と、Raspberry Pi 上の clang (T) を図 1 に示す。

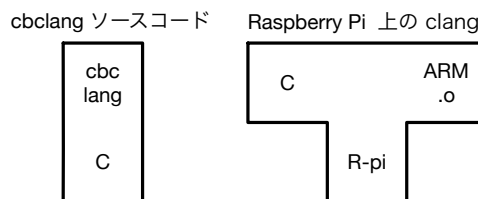


図 1: 図の例

Raspberry Pi は ARM のコードを生成する。CbC は C によって記述されているため、C から ARM.o を生成し、ARM.o から a.out を出力する。Raspberry Pi 上で xv6 を実装し、cbc から a.out を出力するまでの過程を図??に示す。

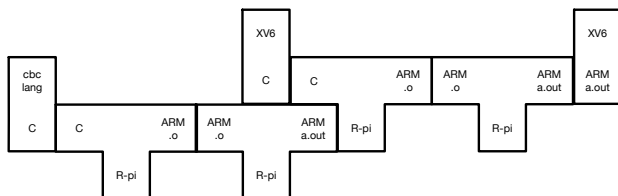


図 2: Raspberry Pi

## 4 OSX CrossCompile

OSX 上で xv6 を実装する場合、C から mach-o のバイナリファイルが生成される。この時、出力は mach-o になるので、mach-o の loader 作ることによって CbC が動かせるようになる。

OSX 上で行えるためコンパイルの速度向上が望める。xv6 を実装し CbC から a.out を出力するまでの過程を、図 3 に示す。

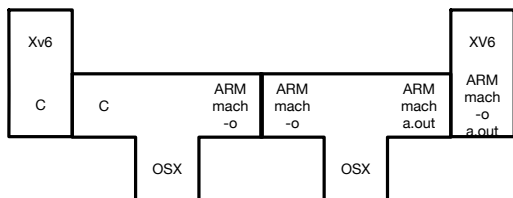


図 3: OSX

## 5 Linux CrossCompile

OSX と別に Linux 環境で CbC を動かした後に、Raspberry Pi に載せる手法がある。

Linux の LLVM でコンパイルすることができれば elf のコードを書けるようになるので、elf の loader を作る事で、CbC を動かすことができる。

また、Linux 用の gcc を CbC に書き直す際に、gcc7 に書き直せば linker がそのまま使えるので、xv6 で動くようになる。

CbC は C によって記述されているので、C から ARM.o のバイナリを生成し、ARM.o から a.out を出力する。OSX 上に立ち上げた Linux 上で xv6 を実装するまでの過程を、図 4 に示す。

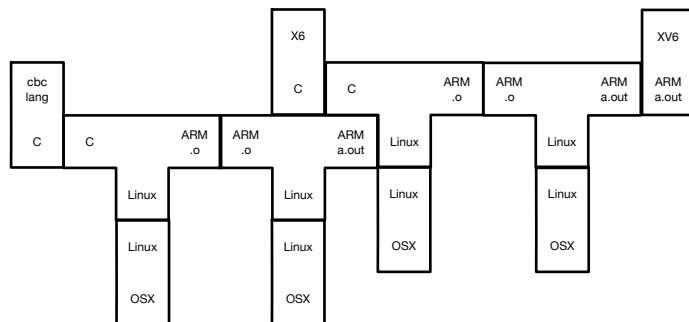


図 4: OSX Linux VM

## 6 今後の課題

xv6 で CbC が動くようになれば、Raspberry Pi 以外のハードウェアでの実装も容易になるので、Linux 上での実装を目指して研究を進めていく。xv6 で CbC が動けば、続けて Linux 上で Gears OS の実装も行なっていく。

## 参考文献

- [1] 徳森 海斗, 河野真治. Llvm clang 上の continuation based c コンパイラ の改良, 2015.
- [2] 伊波立樹, 東恩納琢偉, 河野真治. Code gear, data gear に基づく os のプロトタイプ, 2016.
- [3] 小久保翔平, 河野真治. Code segment と data segment を持つ gears os の設計, 2016.
- [4] The LLVM Compiler Infrastructure. <http://llvm.org>.