# ハードウェア上で直接動作するRuby処理系の試作

吉原　陽香†　　並木　美太郎‡
†東京農工大学工学部情報工学科　　‡東京農工大学大学院共生科学技術研究院

## 1　はじめに

スクリプト言語Ruby（以下，Ruby）は簡単に拡張でき，さまざまなプログラミングスタイルに対応しているプログラミング言語である．本論文では，OSなしでハードウェア上で直接実行するRuby処理系の試作と，その上に実装するオペレーティングシステム（以下，RubyOS）の設計について述べる．現時点で，上記で述べたOSなしでハードウェア上で直接実行するRuby処理系を試作した．また，キーボードからの入力を受け取り画面に文字を出力する機能を実装した．

## 2　問題分析

本章では，Ruby処理系をベアマシン上に実装するにあたり行った検討について述べる．一つ目として，Ruby処理系をどのようにベアマシン上で実装するかという点が挙げられる．その解決法として，既存のRuby処理系を用い，標準Cライブラリとして OSに依存しない組み込み用の標準ライブラリを用いることにした．この方法のメリットとして，新しく処理系を作成する手間が省ける点などがある．

また，RubyOSでは，メモリ管理などといった資源管理のため，メモリ操作といったRubyの言語仕様にない機能が必要になる．これは，対応する操作を実行できるクラスをRubyの機能である拡張ライブラリとしてCで記述することで解決する．ただし，レジスタの値を退避するといった，Ruby処理系自身の動作を中断するような動作については，Rubyでは記述できないためCで記述する．現時点では，最低限必要な機能としてメモリアクセスとIOポートアクセスについてCにて記述し，それを拡張ライブラリとしてRubyから利用する．そのほかの部分についてはRubyにて記述する．

## 3　目標

本研究では，ベアマシン上でRuby処理系を直接動作させ，カーネルを含めたOSの機能をRubyで書くことを目標とする．最終的な目標は，以下に挙げる資源管理をRubyOSにて行うことである．

- プロセス・スレッド管理
- メモリ管理
- ファイルシステム
- デバイスドライバ
- 外部割込み

また，Rubyの言語仕様にない，メモリやIOポートのアクセスについては，Rubyの拡張ライブラリの機能を用いてRubyプログラムから利用できるようにする．それぞれの詳しい設計については次章で述べる．

## 4　設計

RubyOSの構成を図1に示す．

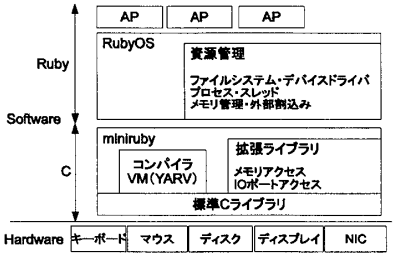


図1: RubyOSの設計

第2章でも述べたとおり，現実にはすべての処理をRubyで記述することは不可能であるため，機能の一部はCやアセンブラを利用して記述することになる．次にそれぞれの機能の設計と記述言語の検討について述べる．

### 4.1　プロセス・スレッド管理

RubyのVMオブジェクト一つにつき一プロセスを割り当て，Rubyで記述されたプログラムを実行する．またRubyOSそのものもRubyのVMオブジェクトにて実行する．メモリやハードウェアの操作ができるのはRubyOSだけである．プロセスがそれを行いたい場合はRubyOSに対してシステムコールを発行し，割り込みをかける．

スケジューリングによって走行プロセスを切り替える際にはアセンブラを利用する．そのほか，プロセス

Prototype of Programming Language Ruby Interpreter to Execute Directly on Hardware
†Haruka YOSHIHARA　‡Mitaro NAMIKI
†Faculty of Engineering, Tokyo University of Agriculture and Technology.
‡Institute of Symbiotic Science and Technology, The Graduate School at Tokyo University of Agriculture and Technology.

ディスクリプタの管理やスケジューリングはRubyで記述できる．またスレッドはRubyのスレッド機能を利用し，Rubyで記述する．

### 4.2 メモリ管理

プロセスをVMオブジェクトとした場合，プロセスにはヒープを割り当てることになる．ヒープの効率的な利用と，メモリの保護のため，Cによってページング機構を実現し，ページ単位でプロセスにヒープを割り当てる．ほか，ヒープ領域の管理にCを利用する．またVMオブジェクトの開放にはRuby処理系に実装されているマークアンドスイープ方式のガーベージコレクションを利用し，Rubyにて操作する．

### 4.3 ファイルシステム

RubyOSではデータをオブジェクト単位で扱う．ファイルシステムにはRuby処理系に標準添付される，キーとRubyのオブジェクトを対応づけ，オブジェクトをバイナリデータとして外部に保存するライブラリであるPStoreを利用する．あるディレクトリまたはオブジェクトを保存したい場合，ディレクトリ名もしくはオブジェクト名をキーとし，それぞれがもつ情報をハッシュ列のオブジェクトとし，キーに対応付けて保存する．

二次記憶へのメモリアクセスはCを用いて拡張ライブラリを記述し，Rubyにて利用する．そのほかの部分についてはRubyで記述する．

### 4.4 デバイスドライバ

デバイスドライバは，オブジェクトとしてRubyで記述する．メモリアクセスやIOポートアクセスの部分については拡張ライブラリで実装した機能を用いて，周辺機器ごとに利用する機能をRubyにてクラスのメソッドとして記述する．また，デバイスドライバのオブジェクトにはデバイス名，デバイスの状態なども属性としてもつ．

### 4.5 外部割り込み

ハードウェアからの割り込みがあったときにはメモリに設定された割り込み関数テーブルより割り当てられた関数を実行する．この関数はコンテキスト退避を行った後，Ruby処理系に拡張されたシグナルフラグを設定する．シグナルフラグはRubyの処理系によりスレッドのタイマーなどでチェックされ，そのフラグにあたるシグナルに設定されたハンドラを実行する．

割り込み関数テーブルの設定やコンテキスト退避はアセンブラやCを利用して記述する．シグナルハンドラの設定やハンドラの中身の記述はRubyで行う．

## 5 実装

Ruby処理系のビルド途中で構築される最小構成のRuby処理系であるminirubyをベアマシン上で実装した．標準Cライブラリとして，組み込み用標準Cライブラリである Newlib を利用した．実メモリアクセス，IOポートアクセスは，拡張ライブラリをCで作成し，Rubyプログラムから利用できるように実装した．この拡張ライブラリを用いて，ポーリング方式でキーボードの文字情報を受け取り，それを画面上に表示するRubyプログラムを作成し動作を確認した．次の図2に，動作をQEMUにてエミュレートした様子を示す．

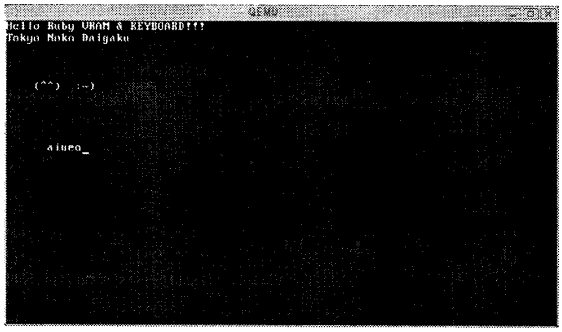


図2: QEMUでの実行画面

## 6 おわりに

本研究では，ベアマシン上におけるRuby処理系の直接実行の手法とRubyによるOSの設計について述べた．拡張ライブラリを用いてメモリアクセスやIOポートアクセスを行うことはできたが，まだ具体的なRubyOSの実装には至っていない．今後の予定として，スレッドライブラリの実装が挙げられる．

## 参考文献

[1] 川合秀実,「30日でできる！ OS自作入門」，毎日コミュニケーションズ，2006

[2] J. P. Sansonnet, M. Castan, C. Percebois, D. Botella, J. Perez, "Direct execution of lisp on a list_directed architecture", Proc. ASPLOS-Ipp.132 - 139.March 1982

[3] Robert Sheehan, "Teaching Operating Systems with Ruby", Proc. ITiCSE ' 07, pp.38 - 42 , June 2007.

[4] トム・サルポー，チャールズ・ミロ著，油井尊訳，「インサイドJavaOSオペレーティングシステム」，ピアソン，1999