SWEST 26 インタラクティブセッション

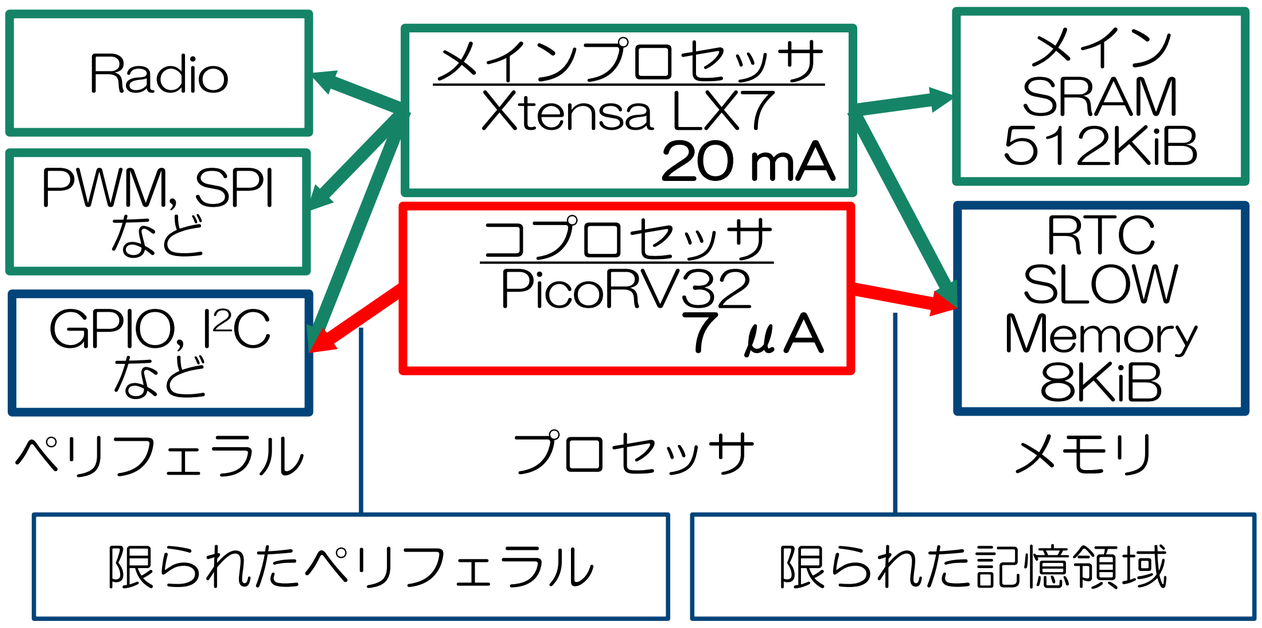
省電力コプロセッサ上でRubyプログラムを動かす組込向けmruby処理系の実装に向けて

鈴木豪，渡部卓雄，森口草介（東京工業大学）

ターゲット：Espressif ESP32-S3/C6

メインプロセッサの他に，32ビットRISC-Vである**省電力コプロセッサ（Low Power Coprocessor）**を持つ．

ESP32-S3のシステム概要図



消費電流は公式データシートの典型的な最小値による

コプロセッサプログラミングの難しさ

- メモリの一部領域しか使えない．
- C言語しか使えない．（機械語のみのマイコンもある．）
- メモリマッピングが異なる場合がある．
- デバッグポートが無い場合がある．
- 命令セットアーキテクチャが異なる場合がある．

mruby言語とmruby/c[1] インタプリタ

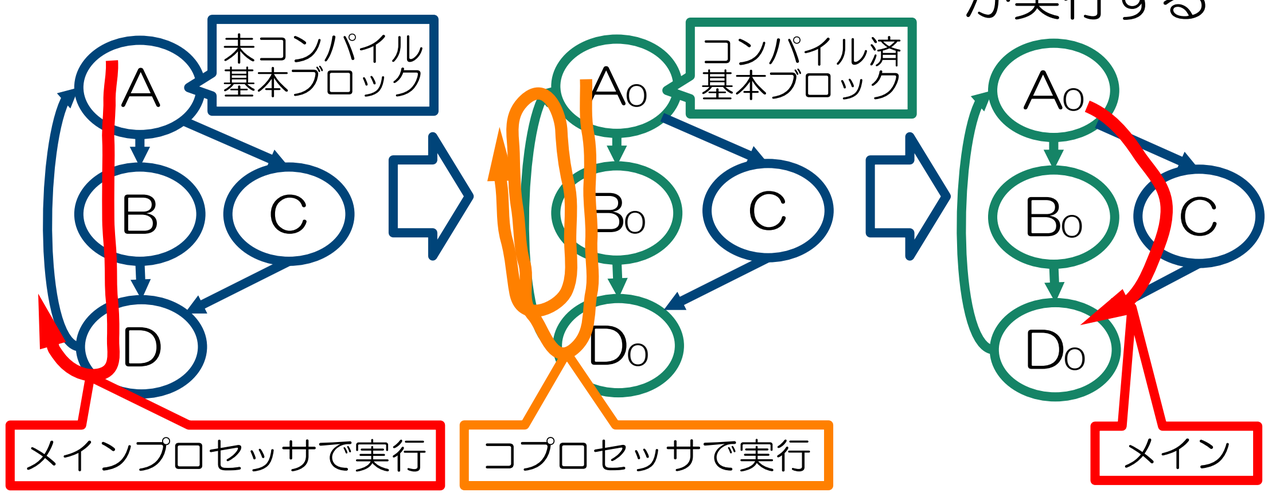
mruby/cはRubyをホストでmrubyバイトコードに変換した後に実行する．.textは50KiB以上で，コプロセッサに載せるのは厳しい．本研究ではmruby/cを拡張する．

mruby/c処理系の特徴

- break/nextは大域脱出が必要，例外，クロージャ（？）
- オブジェクトは基本的に固定長．

実行時コンパイルの概要

メインプロセッサがコプロセッサ向けのプログラムを実行時コンパイルする．



関連手法

- Tracing JIT[2]：翻訳単位がBasic Blockではないため，分岐によってコードサイズが大きくなる．
- Lazy Basic Block Versioning [3]：部分評価器が必要になり，コードサイズが増える．

省電力コプロセッサ利用例

Copro#runクラスメソッドが呼ばれると，実行時コンパイラが起動し，ゆくゆくは省電力コプロセッサ上で渡されたブロックを実行する．

コプロセッサを利用したIoTセンサ実装例

```
temperatures = Array.new(60)
Copro.run do
  (0...60).each do |i|
    Copro.delayMs(1000*60)
    temperatures[i] = sensor.read()
  end
end
Server.send temperatures
```

コプロセッサで実行

メインプロセッサで実行

デモのプログラム（LED明滅，Lチカ）

```
Copro.run do
  prevPress = false
  press = false
  while (!prevPress || press) do
    Copro.gpio(4, true) # GPIO出力
    Copro.delayMs(30)   # スリープ
    Copro.gpio(4, false)
    Copro.delayMs(30)
    prevPress = press
    press = Copro.gpio?(5) # GPIO入力
  end
end
```

Lチカの評価

実際の消費電力はデモでお見せします．また，MPLR2024の論文[4]を参照ください．動画もあります．

https://www.psg.c.titech.ac.jp/posts/2024-08-29-SWEST26.html

- S3の消費電流：30.5 [mA] → 2.5 [mA]
- C6の消費電流：33.5 [mA] → 1.9 [mA]
- プロセッサ起動のｵｰﾊﾞﾍｯﾄﾞ：580 - 20 [μs]
- ランタイムのコードサイズ増加： 20 [KiB]
- Lチカのコンパイルに要したメインメモリ上のプロファイリングデータサイズ：776 [B]
- Lチカで生成されたコードサイズ：220 [B]

今後の展開（オブジェクト転送概要）

IoTセンサ例のtemperaturesに格納されたインスタンスは，メインプロセッサから省電力コプロセッサに転送する必要がある．

1. インスタンスを参照すると範囲外参照に関する例外が起こる．
2. アドレス変換LRUテーブルを参照して，対応するアドレスがあれば変換する．
3. 無ければメインプロセッサを起こし，オブジェクトインスタンスの転送とLRUテーブルへの変換エントリ（オブジェクト単位）を登録する．

[1] しまねソフト研究開発センター, 九州工業大学. mruby/c https://github.com/mrubyc/mrubyc
[2] Andreas Gal, Christian W. Probst, and Michael Franz. HotpathVM: an effective JIT compiler for resource-constrained devices. (VEE '06).
[3] Maxime Chevalier-Boisvert and Marc Feeley. Simple and Effective Type Check Removal through Lazy Basic Block Versioning. (ECOOP 2015).
[4] Go Suzuki, Takuo Watanabe, and Sosuke Moriguchi. mruby on Resource-Constrained Low-Power Coprocessors of Embedded Devices. (MPLR '24).
DOI：[2]10.1145/1134760.1134780 [3]10.4230/LIPIcs.ECOOP.2015.101 [4]10.1145/3679007.3685064