Embedded DSLとして作るアセンブラ

takoeight0821 星にゃーん 河野雄也

アセンブラ、作るのちょっと面倒ですよね？

- 字句解析、構文解析…
- 俺はCPUを作りたいのに…

それ、EDSLとして作っちゃいましょう

ちょっと前提の共有：教育用CPUアーキテクチャTemple

8命令からなる命令セットアーキテクチャを用いたコンピュータアーキテクチャ教育教材

https://www.ieice.org/publications/conference-FIT-DVDs/FIT2018/data/pdf/N-021.pdf

- Temple：（大学の先輩が修論で作った）非常にコンパクトなCPU
  - （広島市立大学の）CPU設計の教育目的で利用される。
  - 命令数はわずか8種類。
    - 命令をうまく組み合わせることが肝要。
  - **既存のアセンブラは、Java製のGUIアプリ**。
    - 操作は直感的だが、アセンブリプログラミングの既存のワークフローにはなじまない。

sutra : Temple向けアセンブラ

- sutra
  - Templeのための高機能なアセンブラ。
  - Haskellの組み込みDSLとして実装。
  - 高度なマクロ機能を備える。
  - 様々な静的解析器を搭載予定。

sutraでのプログラム例（バブルソート）

```
sort = mdo
  let i = r 0
      iEnd = r 1
      memOffset = r 7
      j = r 2
  setInt 0
  move i
  setInt 10
  move iEnd
  setInt 1000
  move memOffset
  loopIStart <- label "loopIStart"
  setLabel loopIEnd
  move (t 0)
  cmp i iEnd
  jl (t 0) ge (t 1)
  copy iEnd
  add allone
  move j -- j = iEnd - 1
```

sutraのアーキテクチャ

- Haskellの **組み込みDSL** として実装。
  - Templeプログラムを生成するHaskellプログラムとして記述する。
  - Haskellの高度な抽象化機能、軽量な構文により、 **既存のアセンブラと遜色ない書き味** を実現。

Rubyとかでもうまく作れると思います。

sutraのマクロ (1/2)

- Haskellプログラムとしてマクロを記述できる。

```
-- | 'reg'レジスタの値をアキュムレータへ書き込む。
-- フラグZ, N, Cを更新する。(0x0000 + reg)
copy reg = do
  -- Acc = Acc NOR 0xFFFF (= 0x0000)
  nor allone
  -- Acc = 0x0000 + reg (= reg)
  add reg
```

sutraのマクロ (2/2)

- Haskellレベルの条件式により、静的解析を簡単に実現できる。

```haskell
-- | 汎用目的のレジスタ $r0 ~ $r22 ($m0 ~ $m22)
r :: Int -> Reg
r n
  | 0 <= n && n <= 22 = m n
  | otherwise = error "Invalid register number"

-- | 疑似命令用レジスタ $t0 ~ $t3 ($m23 ~ $m26)
t :: Int -> Reg
t n
  | 0 <= n && n <= 3 = m (n + 23)
  | otherwise = error "Invalid register number"
```

今後の展望

- Haskellに馴染みのないユーザーでも利用可能なラッパーの開発。
  - Haskellプログラムをスクリプトのように実行する仕組み（**cabal script**）を活用。
- 疑似命令（マクロ）内でのレジスタやフラグの利用を制限する機能の実装。
  - 規約で疑似命令用に確保されているレジスタのみを利用可能にする。
  - 通常のプログラムからは明示的な↑のレジスタへのアクセスを制限する。