SecHack365

mirlvm: セキュアで高速なコンパイラ基盤の開発

開発駆動コース　川合ゼミ　吉村仁志

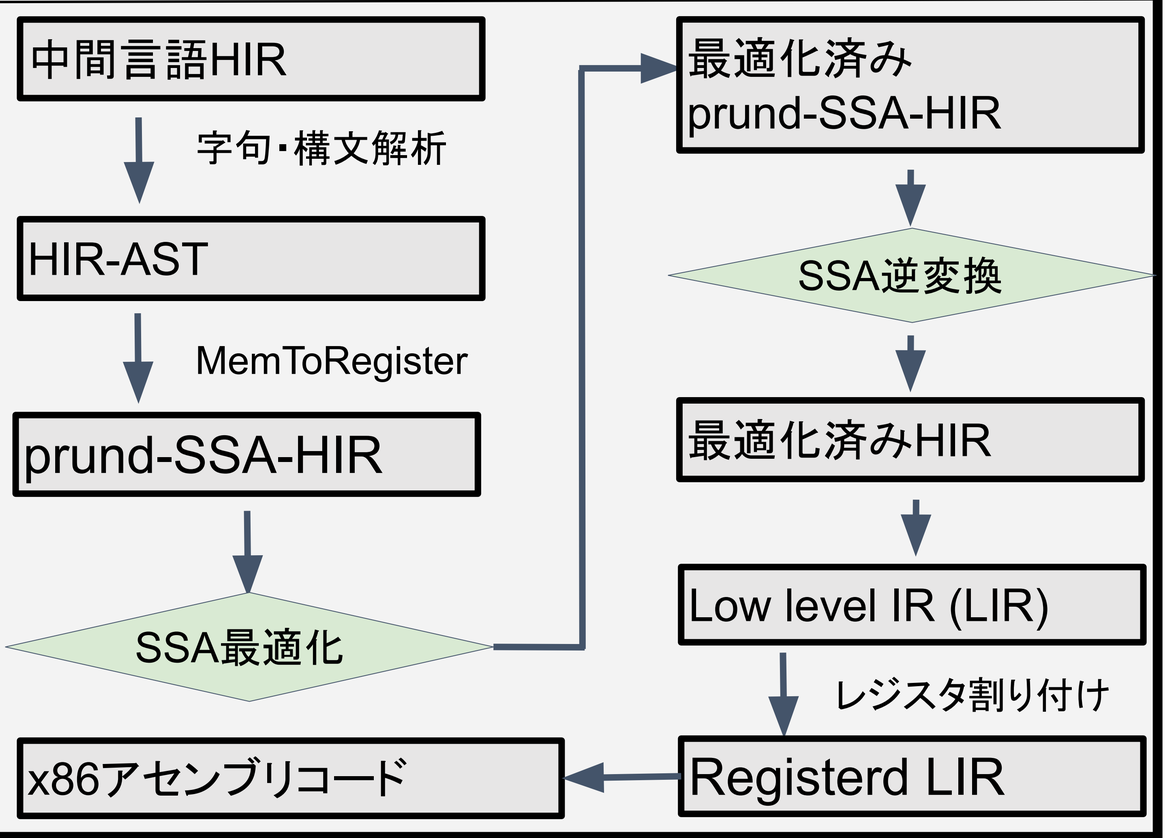
テーマ概要

本プロジェクトでは、セキュリティ機能を持ち、かつ複数の最適化を施すことによって効率の良いアセンブリを生成するコンパイラ基盤を開発する。

コンパイラ基盤とは

・言語処理のバックエンドを担当
・様々な最適化を施すことで高速化
・LLVM,COINSなどがある

mirlvmの設計概要



最適化

・MemToRegister パス
  - 一部のメモリ上のデータを用いた演算をレジスタ上のみで行うようにしメモリアクセス回数を減らすパス
・SSAベースの中間言語
  - 各変数への代入回数がただ一度になる形式
  - レジスタオブジェクトに関してのみ適用
  - 最適化適用を単純化
・SSA形式による最適化
  - デッドコード削除（不要な命令を削除）
  - 定数伝搬（値が確定した変数を定数にする）
  - 定数畳み込み（定数同士の演算を静的に実行）
  - 共通部分式削除（同じ部分式の重複した計算を削除）
  - 命令のループ外移動（ループ内の不変変数を外に移動）
・その他最適化
  - レジスタ割り付け
  - Low Levelでの定数伝搬

プログラム例

```
data $fmt = { b "Three and Eight make %d!\n", b 0 }

function w $add(w %a, w %b) {              # Define a function add
@start1:
    %c =w add %a, %b                       # Adds the 2 arguments
    ret %c                                 # Return the result
}

function w $main() {                     # Main function
@start2:
    %r =w call $add(w 3, w 5)              # Call add(3, 5)
    call $printf(l $fmt, w %r, ...)        # Show the result
    ret 0
}
```

例：3+5を出力するプログラム

・functionによる関数、dataによるグローバル変数定義
・関数呼び出しやレジスタ演算、メモリアクセスに相当する命令

実行速度

10,000,000以下の最大の素数計算のプログラムを測定

mirlvm (最適化なし)	0.424s
mirlvm -O1 (最適化あり)	0.348s
gcc -O3	0.301s
clang -O3	0.291s

最適化は効いてるがgccなど一般のコンパイラには及ばず

今後の展望

・セキュリティ機能の追加
  - 現段階では実行時配列範囲外アクセス違検出とカナリア導入を検討
・未実装である最適化を引き続き実装
・gccなどのコンパイラが出力するアセンブリとの差分を比較し、性能差の原因追及と解決
  - 例えば、mirlvmが出力するアセンブリには不必要なレジスタ代入が生じうる

セキュリティ機能

・実行時の整数オーバーフローを検出

```
masashi@:~/workspace/compiler/mirlvm$ make debug OPTION3=-Sec SSAFILE=mul_overflow.ssa
masashi@:~/workspace/compiler/mirlvm$ ./a.out
execution error of integer overflow.
```

32ビットレジスタ間の演算でオーバーフロー発生の様子

NICT 国立研究開発法人 情報通信研究機構