情報科学特別講義A

スタックマシンの機能拡張
進捗報告

第三回発表資料

発表者 阿部 碧音

目次

1. 変更点とそれによる実現内容
2. 実装の考察
3. Demo
4. 今後の展望

変更点とそれによる実現内容

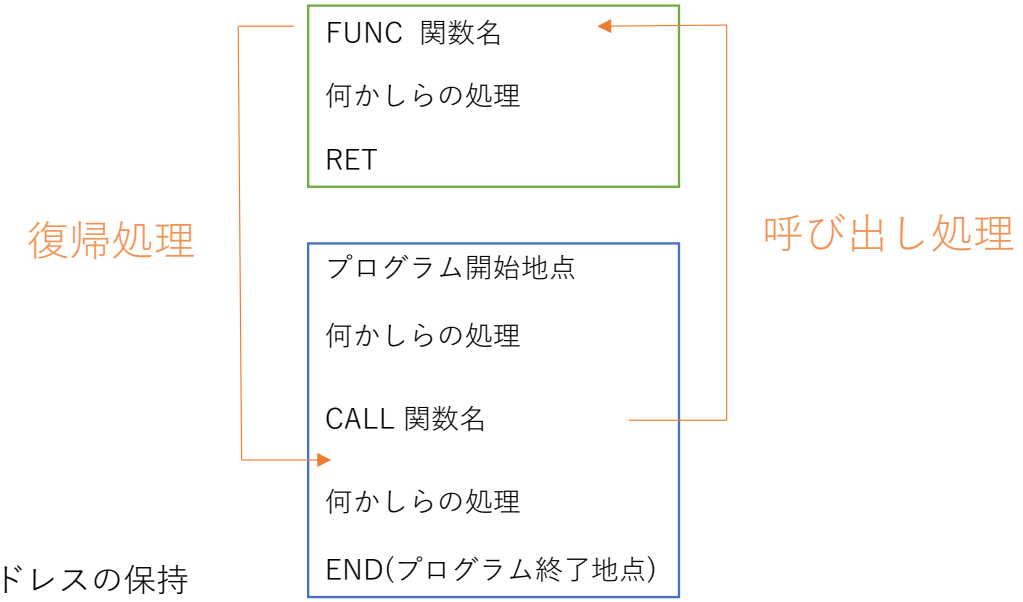
・関数関連の実装の修正
　→ 利用アーキテクチャの変更に伴う内部処理変更
　　もともと記憶空間をラベルと共有していたが分離
　　文字列→数値のHashで管理していたがそれをやめた

・配列の追加
　→ 最低限のアロケート・アクセス・解放を実装

・配列の変数にローカル変数を利用できるように変更

実装の考察：関数について

オペコード	オペランド	仕様
FUNC	string 関数名	関数とそのプログラムのアドレス位置のペアを登録する。
CALL	string 関数名	現在のアドレス位置をコールスタックに積み、指定した関数へとジャンプする。
RET	N/A	現在のコールスタックのトップに保持しているアドレスへとジャンプする。



追加したアーキテクチャ

```
std::stack<unsigned int> _callStack;        // 関数の呼び出し元アドレスの保持
Std::map<std::string, unsigned int> _functions;  // 関数名とそのアドレスの保持
```

実装の考察：関数について

メリット

・コールスタックが存在するため、関数のネストが可能

・コールスタックの深さが異なる変数にアクセス不可
(簡易的なスコープ概念)

デメリット

・スコープ内の変数の開放などをおこなっていない

・関数を必ず呼び出しより前に記載する必要がある

・後に呼び出されるかにかかわらず命令列が読み込まれる

実装の考察：配列について

オペコード	オペランド	仕様
ALLOCARR	string str, unsigned int num	サイズ num の配列 str の領域を確保する。
SETARR	string str, int idx (string var), Int num (string var2)	配列 str のインデックス idx(var) 番目に値 num(var2) を保存する。
GETARR	string str, int idx (string var),	配列 str のインデックス idx(var) 番目の値をスタックの一番上に積む。
FREEARR	string str	配列 str の領域を解放する。

追加したアーキテクチャ

```
std::vector<std::vector<  std::map<std::string, std::vector<int>>  >> _arrays;        // 配列
```

(callStackDepth, blockDepth)

(配列名, 配列実体)

実装の考察：配列について

メリット

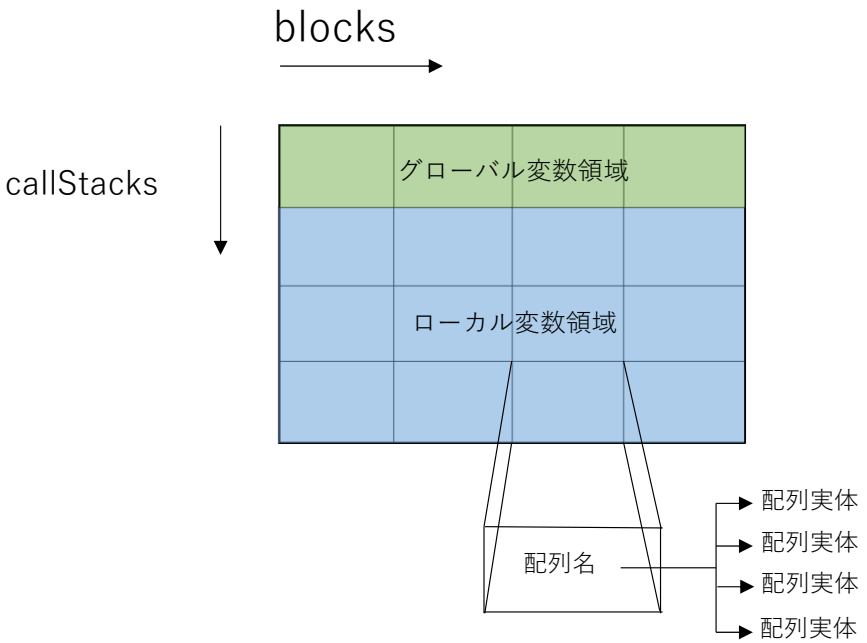
・コールスタックの深さが異なる変数にアクセス不可
　(簡易的なスコープ概念)

デメリット

・現時点で int 型にしか対応していない
・リサイズなどに対応していない
・利用されなくなった配列の自動開放などを行わない

実装の考察:ローカル変数について

オペコード	オペランド	仕様
SETLOCAL	string str, int num	変数名 str に値 num を保存する。
GETLOCAL	string str	変数名 str の値をスタックの一番上に積む



追加したアーキテクチャ

std::vector<std::vector< std::map<std::string, int> >> _variables;

実装の考察:ローカル変数について

メリット

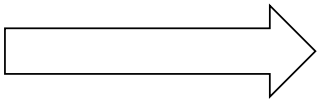
・コールスタックの深さが異なる変数にアクセス不可
(簡易的なスコープ概念)

デメリット

・現時点で int 型にしか対応していない
・利用されなくなった値の自動開放などを行わない

Demo

```
FUNC fib

# 終わりインデックス設定
PUSH 20
SETLOCAL ForEndIdx
POP

# 配列初期値設定
ALLOCARR array 21
SETARR array 0 1
SETARR array 1 1

# イテレーションインデックス設定
PUSH 2
SETLOCAL idx
POP

# 計算開始
beginLoop:
# 配列格納用インデックス計算
PUSH 1
GETLOCAL idx
SUB
SETLOCAL idx-1
POP
PUSH 2
GETLOCAL idx
SUB
SETLOCAL idx-2
POP

# 結果を計算して配列に格納
GETARR array idx-1
GETARR array idx-2
ADD
SETLOCAL tempAns
PRINT
POP
SETARR array idx tempAns

# インデックスインクリメント
GETLOCAL idx
PUSH 1
ADD
SETLOCAL idx
POP

# ループ脱出判定
GETLOCAL idx
GETLOCAL ForEndIdx
LT
LOGNOT
JPEQ0 endLoop
JUMP beginLoop
endLoop:
RET

CALL fib
END
```



```
ExampleOperation.txt start.
2
3
5
8
13
21
34
55
89
144
233
377
610
987
1597
2584
4181
6765
10946
```

今後の展望

・様々な型の実装 (利点 ◎, 実装 重)
→ 命令セットのフォーマットから変えなければいけない
どの型をサポートするか?なども定義する必要がある

・グローバル変数の実装 (利点 ○, 実装 軽)
→ 領域と仕組みはあるのであとは実装するだけ

・配列のリサイズ(利点 ○, 実装 軽)
→ 他に干渉しづらい命令を一つ追加する程度でよい