Continuation based C コンパイラの GCC-4.2 による実装

与儀 健人　　河野真治

琉球大学大学院理工学研究科情報工学専攻

April 23, 2008

研究背景

- Continuation based C (CbC)
- Micro-C による CbC コンパイラの実装
- GCC で CbC コンパイラを実装する方法が分かっている

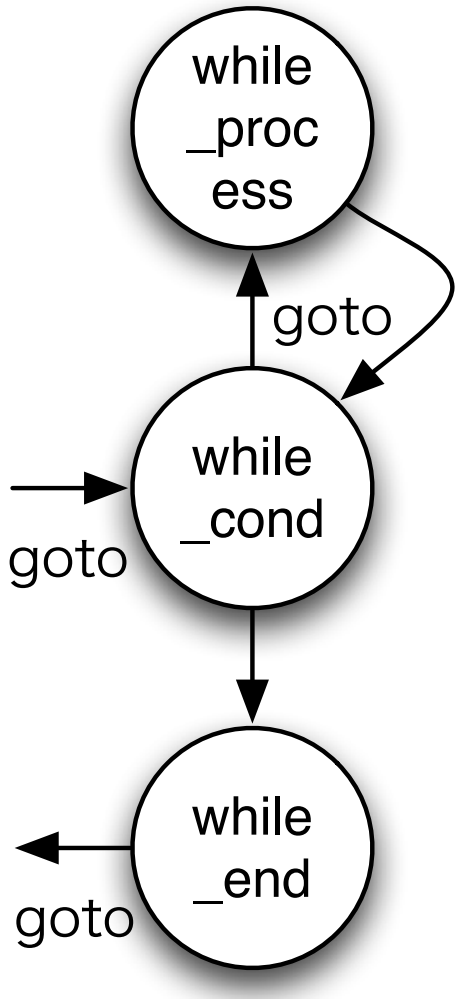
Continuation based C について

What is it?

- 琉球大学 並列信頼研究室で開発
- 構文は C 言語とほぼ同じ
- 関数の除去、コードセグメントの追加
- コードセグメントを繋ぐ “継続” を導入

CbC コード例

```
__code while_process(int total,int count){
    total += count;
    count++;
    goto while_cond(total, count);
}
__code while_cond(int total, int count){
    if ( count <= 100 ){
        goto while_process(total, count);
    }else{
        goto while_end(total);
    }
}
__code while_end(int total){
    goto cs_exit(0);
}
```



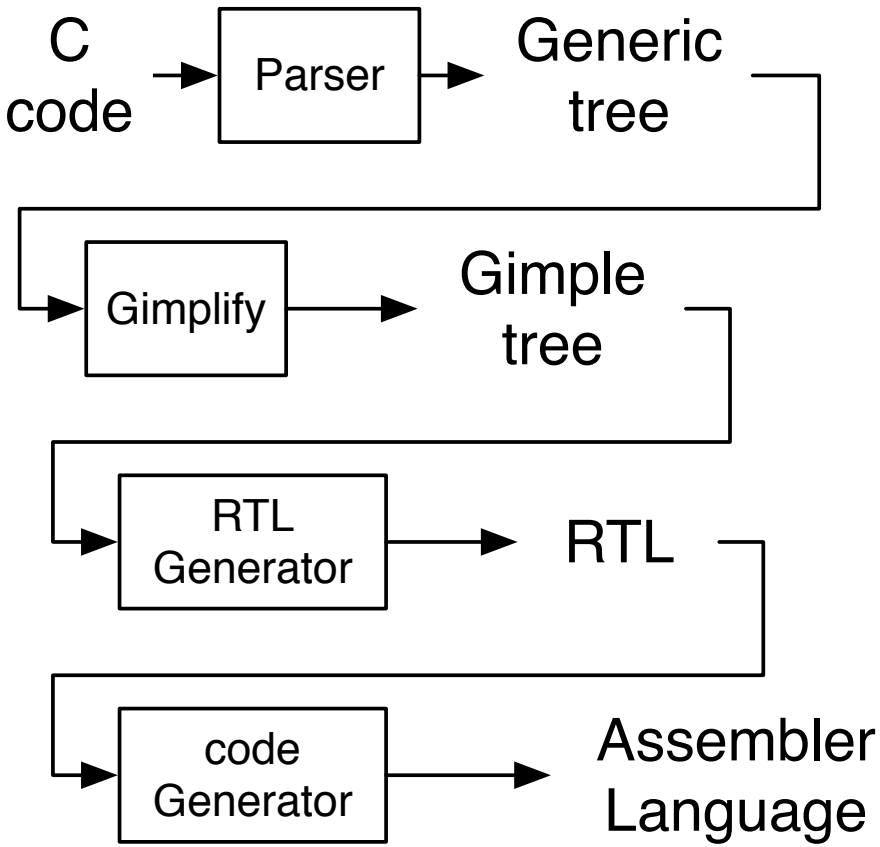
実装に必要な構文

- `__code cs(int a, char *b)`
- `goto cs(10, "abc");`

GNU Compiler Collection

- UNIX における標準的なコンパイラ
- C, C++, java, FORTRAN, Ada ..
- i386, PowerPC, MIPS, SPARC ..
- 強力な最適化機構
- コンパイルだけでなく as, ld などの統合環境

GCC コンパイルパス

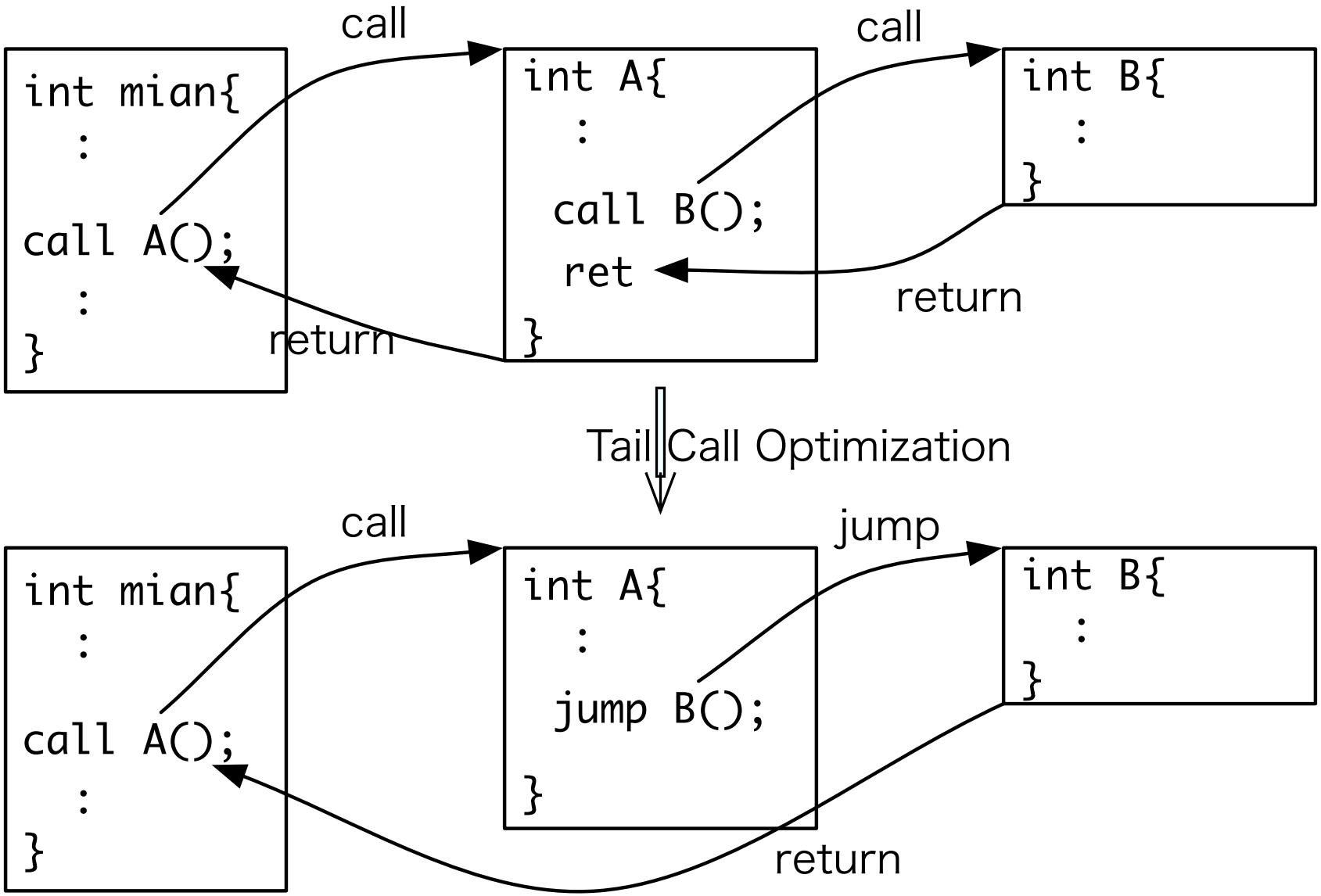


Generic Tree 構文木

GIMPLE SSA

RTL 中間コード

Tail call elimination



```
A:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $24, %esp
    movl    20(%ebp), %eax
    addl    16(%ebp), %eax
    movl    %eax, 8(%esp)
    movl    12(%ebp), %eax
    movl    %eax, 4(%esp)
    movl    8(%ebp), %eax
    movl    %eax, (%esp)
    call    B
    leave
    ret
```

```
A:
    pushl   %ebp
    movl    %esp, %ebp
    movl    20(%ebp), %eax
    addl    %eax, 16(%ebp)
    popl    %ebp
    jmp     B
```

Tail call の条件

1. 関数コールが return の直前にある
2. 関数の返す型が caller と callee で一致している
3. caller 側の引数サイズが callee 側の引数サイズより大きいもしくは等しい
4. 書き込んだ引数が、その後書き込む引数を上書きしてはならない

Tail call の条件

1. 関数コールが return の直前にある
2. 関数の返す型が caller と callee で一致している
3. caller 側の引数サイズが callee 側の引数サイズより大きいもしくは等しい
4. 書き込んだ引数が、その後書き込む引数を上書きしてはならない

引数をすべて固定数とすることで対応

実装

- __code トークンの追加
- code segment のパース — Parser
- goto のパース — Parser
- code segment/goto を表す Generic Tree
- tree/RTL 変換 (expand_call) — RTL Generator
- その他 (エラー検出など)

実装

- __code トークンの追加
- code segment のパース — Parser
- goto のパース — Parser
- code segment/goto を表す Generic Tree
- tree/RTL 変換 (expand_call) — RTL Generator
- その他 (エラー検出など)

expand_call

What is the function?

- 関数呼び出しの tree から RTL へ変換する
- Tail call が可能ならその最適化を行う
- この関数のみで 1200 行もある
- そのほとんどが Tail call 可否の判定
- そして読みづらい

expand_cbc_goto

What is it doing?

- code segment への goto を表した tree を RTL に変換
- 無駄な Tail call 可否判定を削除
- Tail call の条件を強制
- 確実に Tail call を適用
- 500 行程度

評価 (ベンチマーク)

- 環境 i386 fedora core
- 使用したプログラム conv1

	./conv1 0	./conv1 1	./conv1 2	./conv1 3
Micro-C	5.25	8.97	2.19	2.73
GCC	3.69	4.87	3.08	3.65
GCC (+omit)	2.74	4.20	2.25	2.76
GCC (+fastcall)	2.70	3.44	1.76	2.34
TCC	4.15	122.28	84.91	102.59

+omit -fomit-frame-pointer オプションを付加

+fastcall `__attribute__ ((fastcall))` を付加

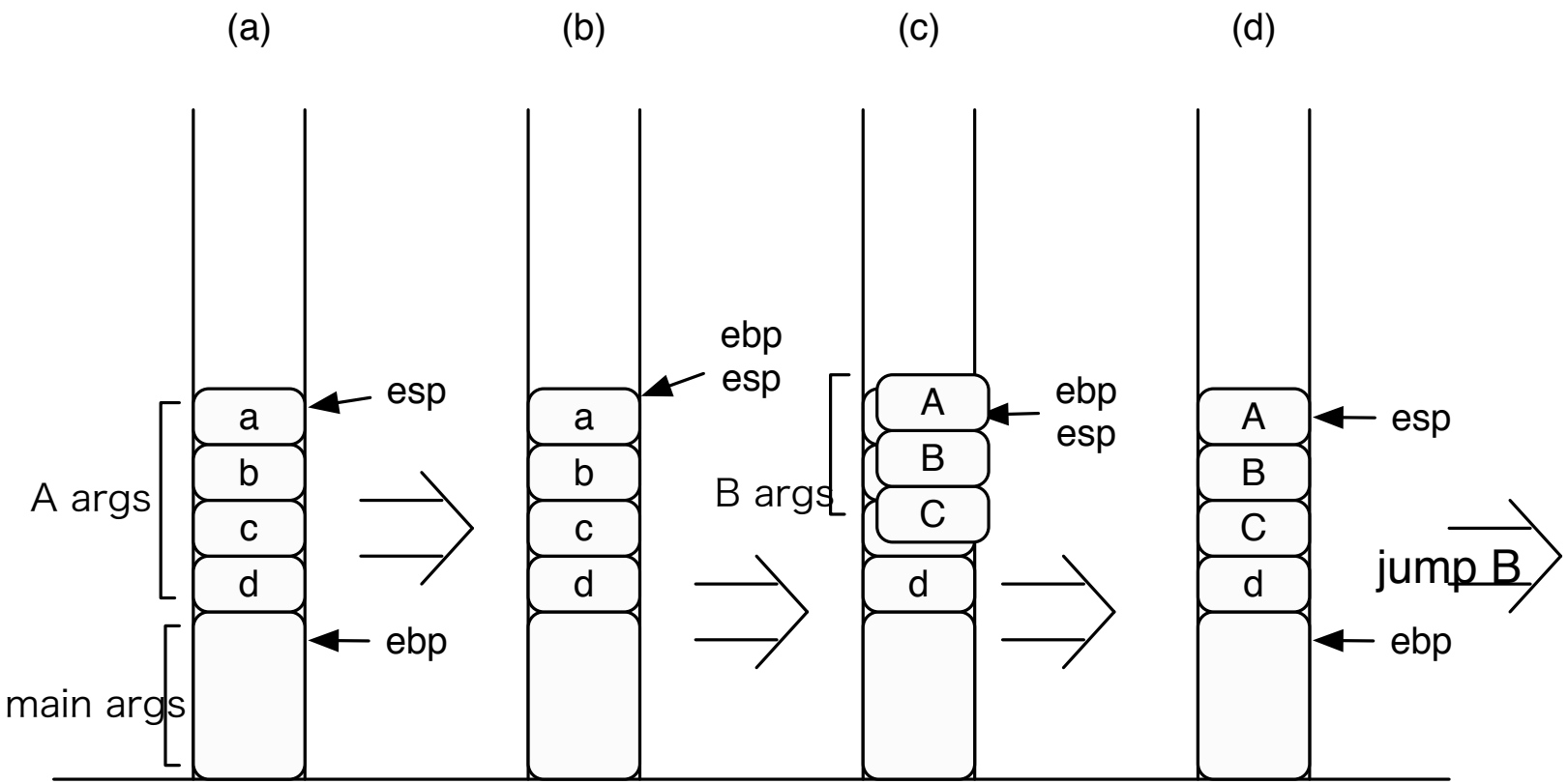
まとめ

まとめ

- GCC に CbC コンパイラを実装
- そのベンチマーク
- 性能向上を確認

TO DO

- environment
- PPC の RTL 変換不能
- オプションの強制
- SPU 対応と GCC の version


(a)
(b)
(c)
(d)
A args
main args
B args
a
b
c
d
A
B
C
esp
ebp
ebp
esp
ebp
esp
esp
jump B
ebp