後期実験5
プログラミング言語処理系

5班
Vu Van Tan
Xia Yue
Tomoya Kitazato

完成したもの

- テスト用データをコンパイルできた
- 最適化
- 追加機能

レジスタを用いた高速化

主に式の計算に利用する

(1)レジスタ(ebx,esi,edi)

(2) 避難用のメモリ

- メモリへのロード・ストアの回数を最小にする→局所的に最適化

- 構文木のノードに、そのノードがルートとなるサブツリーの計算に必要となるレジスタの数を保存する

レジスタの数の最小値

- ノードが葉であれば

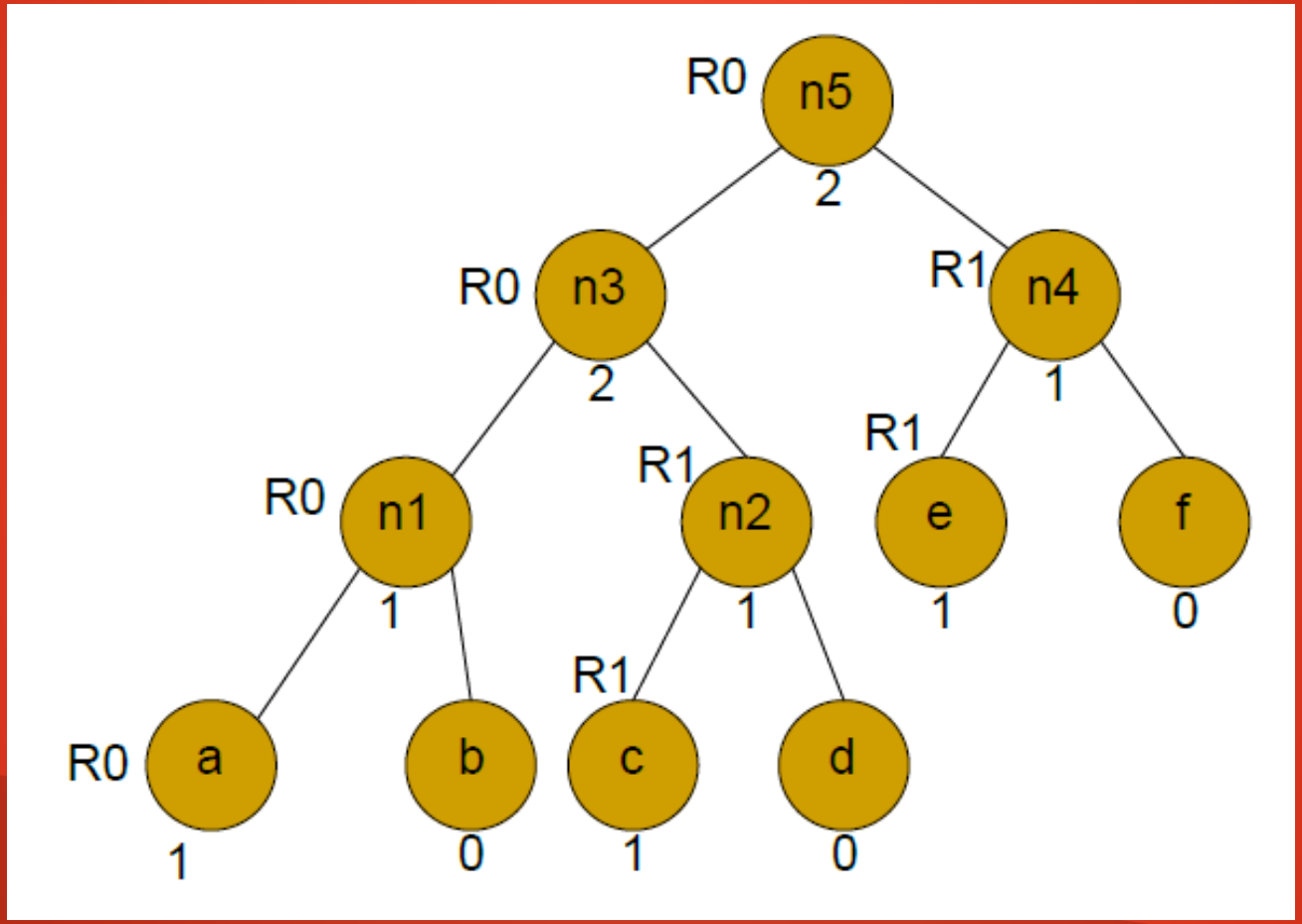
右←0、左←1

- ノードが二つの子n1, n2をもつ場合、

(1) n1 = n2 → n = n1 + 1

(2) n1 != n2 → n = max(n1, n2)

レジスタの数の最小化



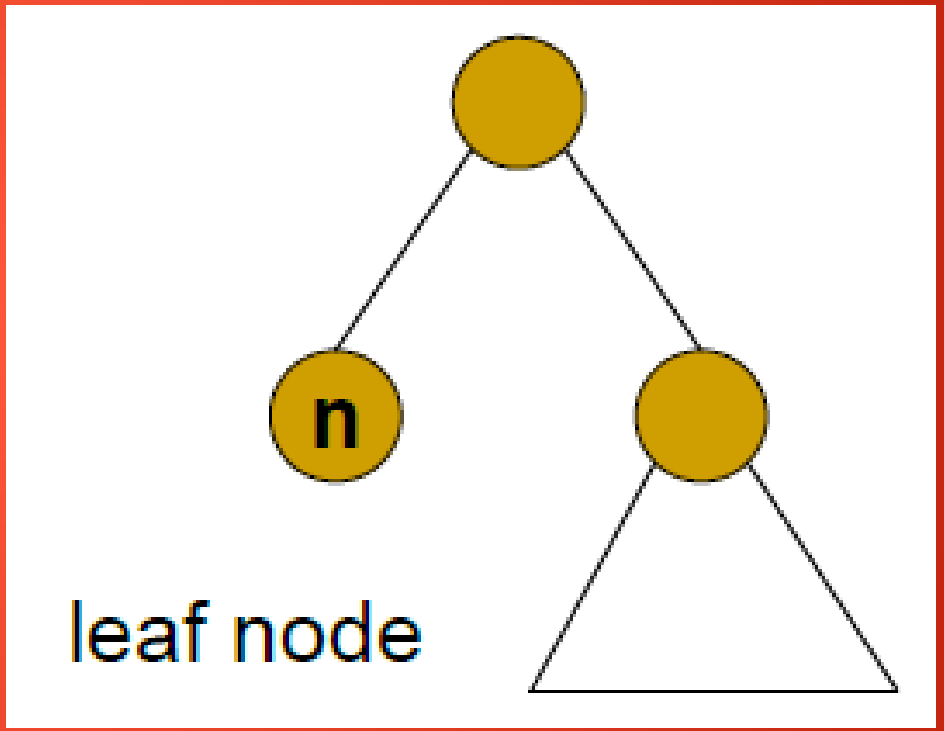
レジスタの割り当て

- 利用レジスタの数: R = 3
- 計算に利用できるレジスタのスタック reg
- レジスタの数が足りない時に必要なメモリのスタック mem
- 式の計算を始める前に、regに三つのレジスタを、memに避難用のメモリを入れる

実際の割り当て

nが左の葉の場合

load n top_stack(reg)

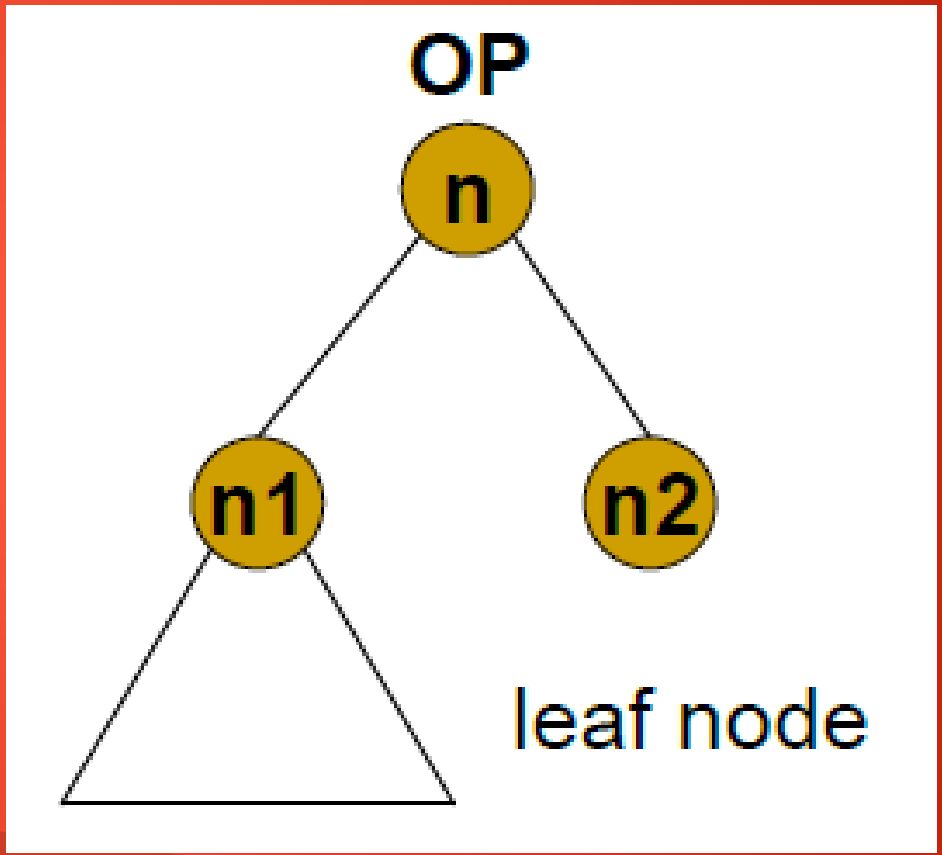


実際の割り当て

n2 = 0の場合

code_gen(n1)

op n2 top_stack(reg)



実際の割り当て

n2 > n1 & n1 < Rの場合

code_gen(n2)

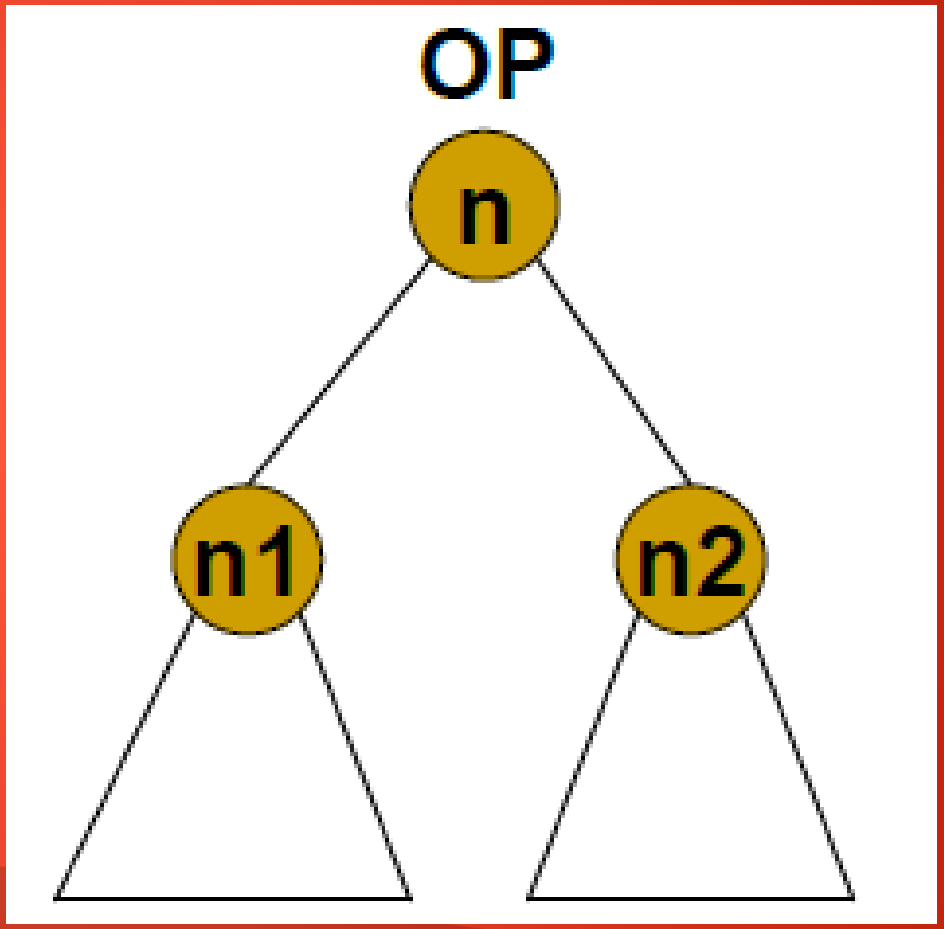
r = pop_stack(reg)

code_gen(n1)

op　r　top_stack(reg)

push_stack(reg, r)

swap_stack(reg)



実際の割り当て

n1 >= n2 & n2 < Rの場合

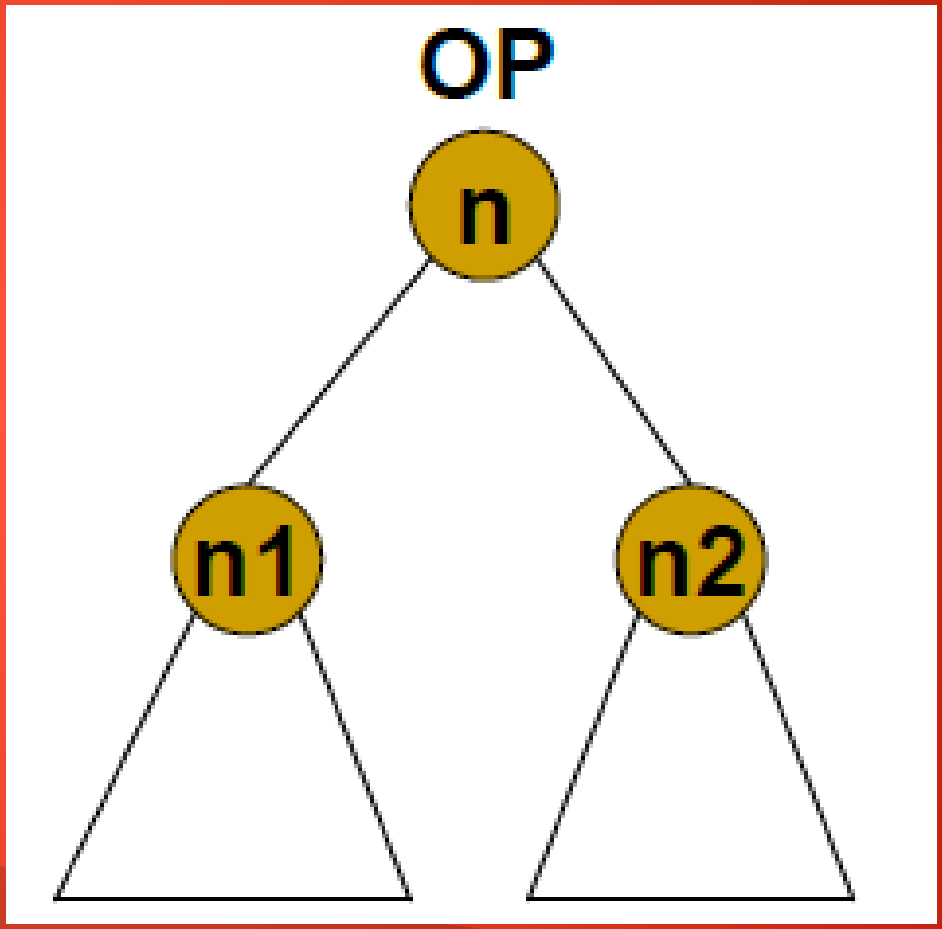
code_gen(n1)

r = pop_stack(reg)

code_gen(n2)

op　top_stack(reg)　r

push_stack(reg, r)



実際の割り当て

n1 >= R & n2 >= Rの場合

code_gen(n2)

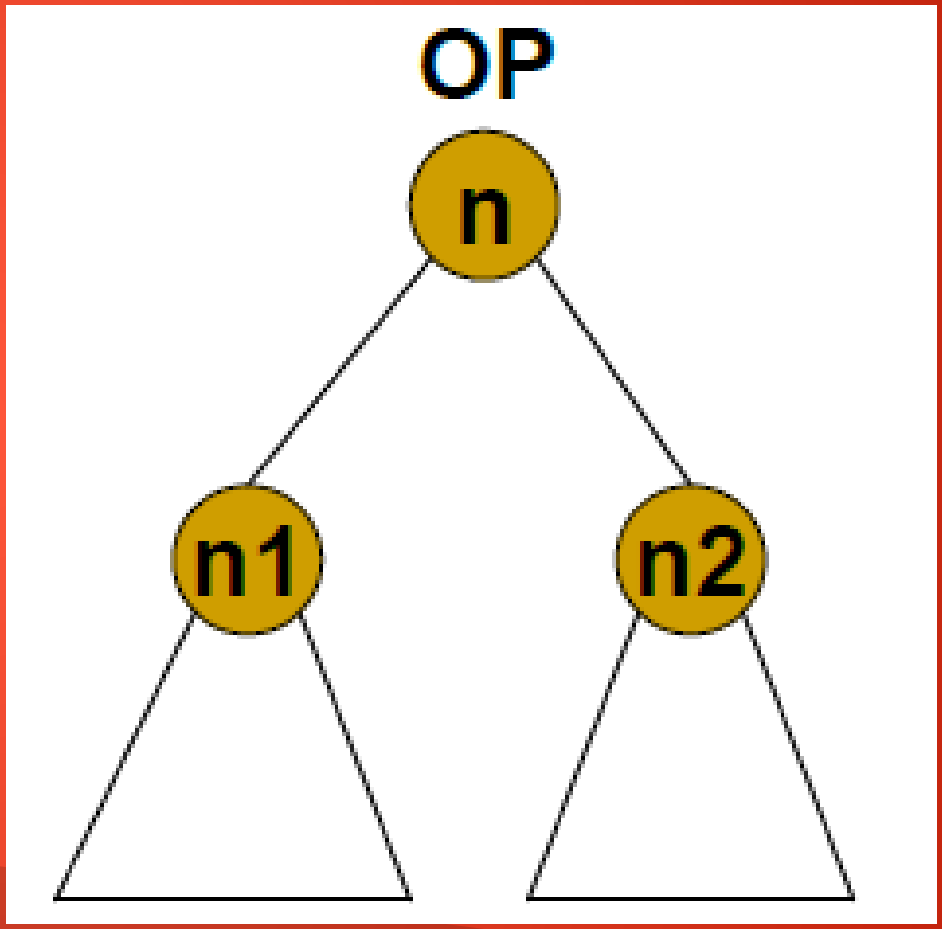
t = pop_stack(mem)

load　top_stack(reg)　t

code_gen(n1)

op　t　top_stack(reg)

push_stack(mem, t)



工夫した点

- if文とwhile文の条件式Eの計算

(1) E = const

(2) E = x + y, x – y, x == y, x != y, x >< y

(3) E = x * y

(4) E = x, !x

ここで、x,yは定数か変数

計算を行わずに、比較だけでジャンプする

実行速度

- prime.c

  (参考)

  gcc　　time = 5.361574172974 sec

  gcc -O3　　time = 4.761824950925 sec

time = 5.315451860428 sec

速い！！

実行速度

- prime5.c

  (参考)

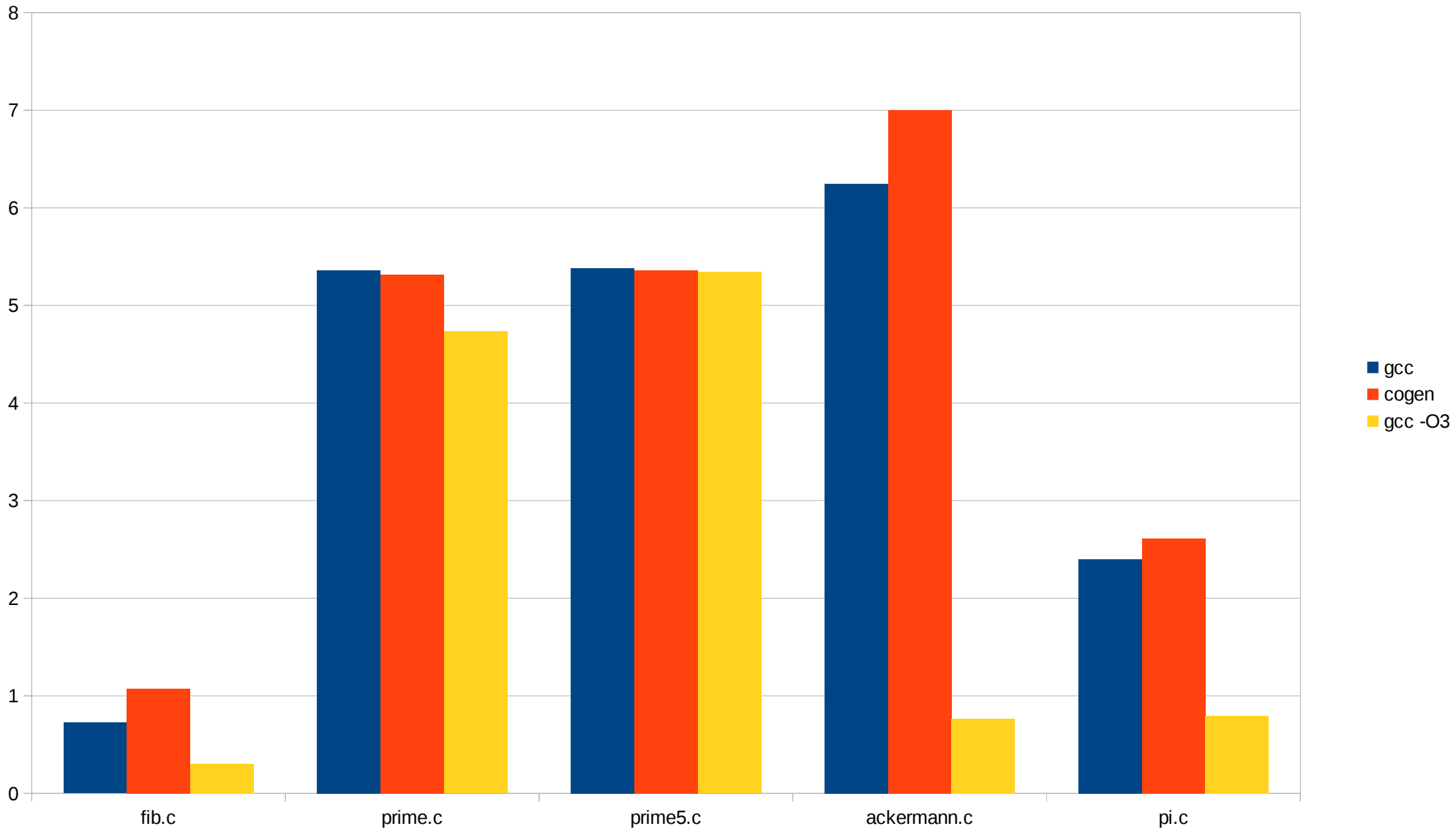
  gcc　　time = 5.379380941391 sec

  gcc -O3　　time = 5.297152977437 sec

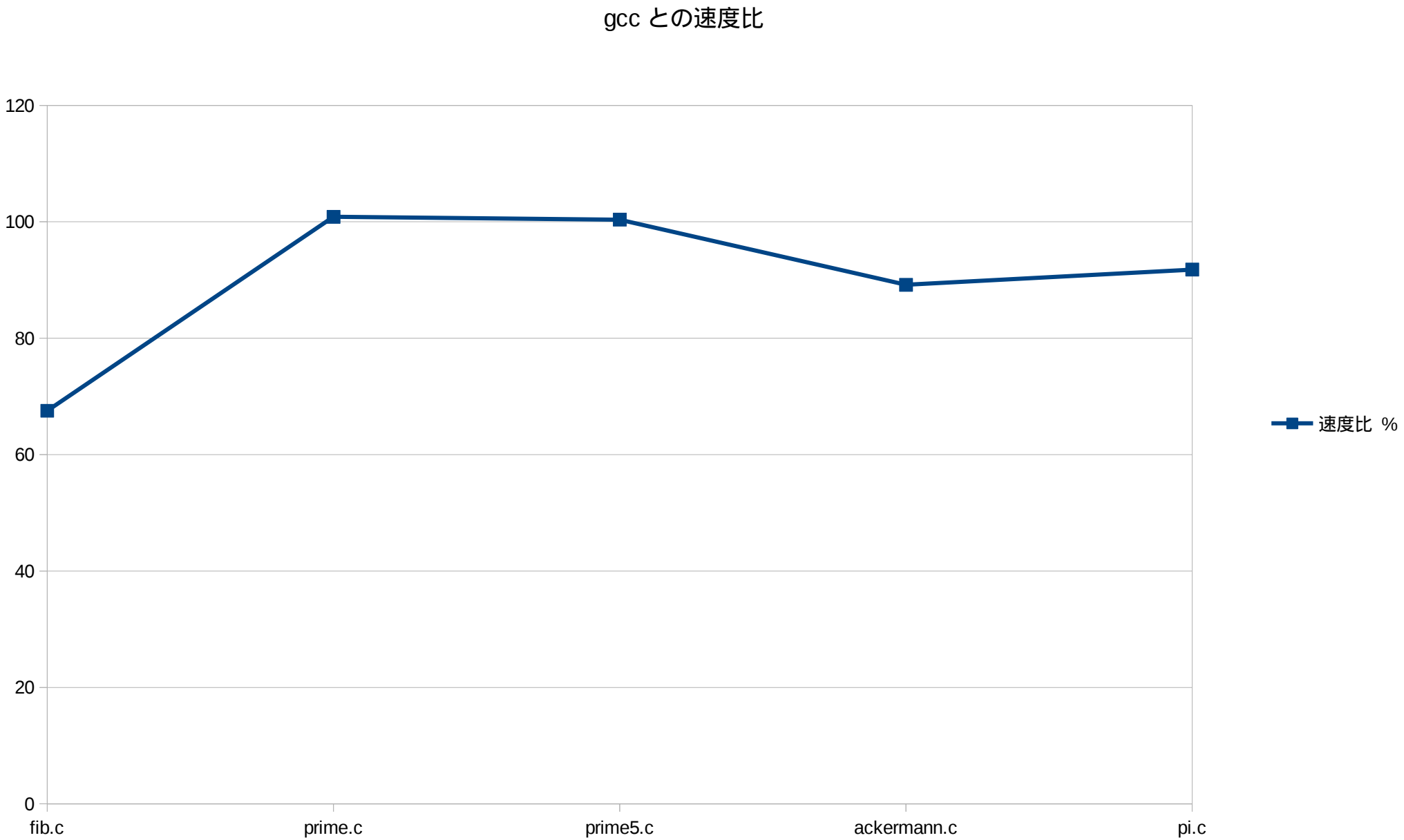
time = 5.3590610027 sec

速い！！

実行時間の比較


8
7
6
5
4
3
2
1
0
fib.c
prime.c
prime5.c
ackermann.c
pi.c
gcc
cogen
gcc -O3

実行速度の比較


gcc との速度比
120
100
80
60
40
20
0
fib.c
prime.c
prime5.c
ackermann.c
pi.c
速度比 %

追加機能

- for文
- コメントアウト
- 型検査

サンプルプログラム

```
int f() {
    int x;   int *y;  int sum;
    sum = 0;
// 1 から 5 までの和を求める
    for(x = 1; x < 6; x = x + 1) {
        sum = sum + x;
    }
    y = x;
    return x;
}
```

サンプルプログラム
実行結果

.globl for_test

- for_test:
- pushl %ebp
- movl %esp, %ebp
- pushl %ebx
- pushl %esi
- pushl %edi
- subl $16, %esp
- movl $0, 16(%ebp)
- movl $0, 8(%ebp)
- jmp .L1

.L0:

- addl 8(%ebp), %edi
- movl %edi, 16(%ebp)
- addl $1, %edi
- movl %edi, 8(%ebp)

.L1:

- cmpl $4, %edi
- setle %al
- movzbl %al, %edi
- cmpl $0, %edi
- jne .L0

.L2:

- movl 16(%ebp), %ecx
- movl %ecx, 12(%ebp)
- movl 16(%ebp), %eax
- addl $16, %esp
- popl %edi
- popl %esi
- popl %ebx
- leave
- ret

新しいアプリケーション

素数 p、自然数 n, kが与えられる

$$C_{n}^{k} \mod p$$

を計算するプログラム

感想

- ペアプログラミングを通じて、人のプログラムの手法を学ぶことが出来ました。
- いろいろ勉強になりました!
- 楽しかったです。実験を通して自分が成長すると感じました。