Workshop on Essential Abstractions in GCC

Gray Box Probing of GCC

GCC Resource Center (www.cse.iitb.ac.in/grc)

Department of Computer Science and Engineering, Indian Institute of Technology, Bombay



29 June 2013

Gravbox Probing: Outline

Examining GIMPLE Dumps for C++

29 Jun 2013

Examining AST for C

Introduction to Graybox Probing of GCC

- Examining GIMPLE Dumps for C
 - ► Translation of data accesses
 - ► Translation of intraprocedural control flow
 - Translation of intraprocedural control flow
 Translation of interprocedural control flow
- Examining RTL Dumps
- (Will be covered later)
- Examining Assembly Dumps
- Examining GIMPLE Optimizations
 - Conclusions

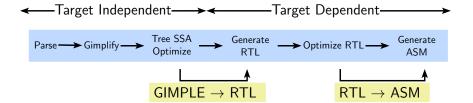
Part 1

Preliminaries

What is Gray Box Probing of GCC?

- Black Box probing:
 - Examining only the input and output relationship of a system
- White Box probing:
 - Examining internals of a system for a given set of inputs
- Gray Box probing:
 - Examining input and output of various components/modules
 - Overview of translation sequence in GCC
 - Overview of intermediate representations
 - ▶ Intermediate representations of programs across important phases

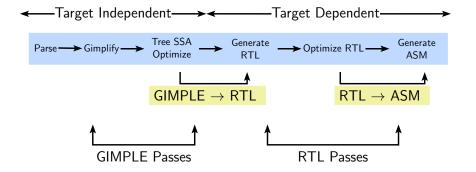
Tranformation from a language to a different language



Basic Transformations in GCC

Graybox Probing: Preliminaries

Tranformation from a language to a different language



3/70

29 Jun 2013

Some passes are called multiple times in different contexts

A total of 215 unique pass names initialized in \${SOURCE}/gcc/passes.c

Conditional constant propagation is called thrice. Some passes are enabled for specific architectures

Graybox Probing: Preliminaries

Transformation Passes in GCC 4.7.2

Some passes have many variations Pass Name Times Optimization nass cd dce Dead code elimination

pass_cu_ucc	Dead code cililination	_	
pass_call_cdce	Dead call elimination	1	
pass_dce	Dead code elimination	2	
pass_dce_loop	Dead code elimination	3	
pass_ud_rtl_dce	RTL dead code elimination	1	
pass_fast_rtl_dce	RTL dead code elimination	1	

- The pass sequence can be divided broadly in two parts
 - Passes on GIMPLE

Total number of passes is 252.

- Passes on RTL
- Some passes are organizational passes to group related passes

4/70

29 Jun 2013

Passes On GIMPLE in GCC 4.7.2

Pass Group	Examples	Number of passes
Lowering	owering GIMPLE IR, CFG Construction	
Simple Interprocedural Conditional Constant Propagation, Passes (Non-LTO) Inlining, SSA Construction		40
Regular Interprocedural Passes (LTO)	Constant Propagation, Inlining	7
LTO generation passes		02
Late interprocedural passes (LTO)	Pointer Analysis	01
Other Intraprocedural Optimizations	Constant Propagation, Dead Code Elimination, PRE Value Range Propagation, Rename SSA	72
Loop Optimizations	Vectorization, Parallelization, Copy Propagation, Dead Code Elimination	28
Generating RTL		01
Total number of passes on GIMPLE		163

Passes On RTL in GCC 4.7.2

Pass Group	Examples	Number of passes
Intraprocedural	CSE, Jump Optimization, Dead Code	27
Optimizations	Elimination, Jump Optimization	
Loop Optimizations	Loop Invariant Movement, Peeling,	07
	Unswitching	
Machine Dependent	Register Allocation, Instruction	52
Optimizations	Scheduling, Peephole Optimizations	
Assembly Emission		03
and Finishing		
Total number of passes on RTI		89

Finding Out List of Optimizations

Along with the associated flags

• A complete list of optimizations with a brief description

```
gcc -c --help=optimizers
```

Optimizations enabled at level 2 (other levels are 0, 1, 3, and s)

```
gcc -c -02 --help=optimizers -Q
```

Graybox Probing: Preliminaries

- Use the option -fdump-<ir>-<passname>
 - <ir> could be

29 Jun 2013

- tree: Intraprocedural passes on GIMPLE
- ipa: Interprocedural passes on GIMPLE
- rt1: Intraprocedural passes on RTL
- Use all in place of <pass> to see all dumps Example: gcc -fdump-tree-all -fdump-rtl-all test.c
- Dumping more details:

Suffix raw for tree passes and details or slim for RTL passes Individual passes may have more verbosity options (e.g. -fsched-verbose=5)

- Use -S to stop the compilation with assembly generation
- Use --verbose-asm to see more detailed assembly dump

Total Number of Dumps

Dump Options: -fdump-tree-all -fdump-ipa-all -fdump-rtl-all

Optimization Level	Number of Dumps	Goals
Default	47	Fast compilation
01	137	
02	164	
O3	173	
Os	160	Optimize for space

169r.reginfo 000i.cgraph 001t. tii 189r.outof_cfglayout 015i.visibility 003t.original

016i.early_local_cleanups

ipa dumps (i)

047i.whole-program 048i.inline 054i.lto_gimple_out

055i.lto_decls_out rtl dumps (r)

150r.expand 151r.sibling 153r.initvals

154r.unshare 155r.vregs

156r.into_cfglayout 157r.jump 158r.subreg1 159r.dfinit

205r.pro_and_epilogue 218r.stack 219r.alignments 222r.mach

10/70

223r barriers 227r.shorten 228r.nothrow 230r.final 231r.dfinish assembly

004t.gimple

009t.omplower

020t.inline_param1

039t.release_ssa

077t.cplxlower

149t.optimized

232t.statistics

137t.tailc

040t.inline_param2

006t.vcg

013t.eh

010t.lower

014t.cfg

021t.einline

018t.ssa

163r.pre

190r.split1

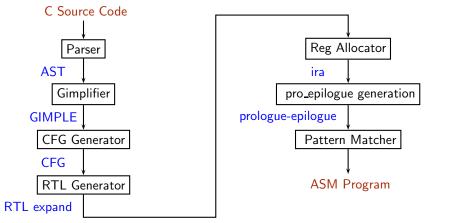
193r.mode_sw

194r.asmcons

197r. ira

201r.split2

Graybox Probing: Preliminaries



Lowering of abstraction!

11/70

29 Jun 2013

Part 2

Examining AST Dump for C

29 Jun 2013

Graybox Probing: Examining AST Dump for C

\$ gcc -fdump-tree-original-raw test.c

GCC Resource Center, IIT Bombay

Abstract Syntax Tree

test.c.003t.original

```
int a;
int main()
{
    a = 55:
}
```

```
:: Function main (null)
;; enabled by -tree-original
@1
        bind_expr
                          type: @2
                                         body: @3
@2
        void_type
                          name: @4
                                         algn: 8
@3
        modify_expr
                          type: @5
                                         op 0: @6
                                                         op 1: @7
@4
        type_decl
                          name: @8
                                         type: @2
@5
        integer_type
                                         size: @10
                                                         algn: 32
                          name: @9
                                          sign: signed
                                                         min : @11
                          prec: 32
                          max : @12
06
        var decl
                                         type: @5
                                                         srcp: t1.c:1
                          name: 013
                          size: @10
                                          algn: 32
                                                         used: 1
@7
        integer_cst
                          type: @5
                                         low: 55
08
        identifier_node
                          strg: void
                                         lngt: 4
@9
        type_decl
                          name: @14
                                         type: @5
                                         low: 32
@10
        integer_cst
                          type: @15
@11
                                                         low: -2147483648
        integer_cst
                          type: @5
                                         high: -1
@12
        integer_cst
                          type: @5
                                         low: 2147483647
@13
        identifier_node
                          strg: a
                                         lngt: 1
@14
        identifier_node
                          strg: int
                                         lngt: 3
@15
        integer_type
                          name: @16
                                         size: @17
                                                         algn: 64
                                          sign: unsigned min: @18
                          prec: 64
                          max : @19
@16
        identifier node
                          strg: bit_size_type
                                                        lngt: 13
@17
        integer_cst
                          type: @15
                                         low: 64
@18
        integer_cst
                          type: @15
                                         low : 0
@19
        integer_cst
                         type: @15
                                         low : -1
```

test.c.003t.original

Abstract Syntax Tree

```
int a;
int main()
{
    a = 55;
}
```

```
:: Function main (null)
;; enabled by -tree-original
@1
        bind_expr
                          type: @2
                                         body: @3
@2
        void_type
                          name: @4
                                         algn: 8
@3
        modify_expr
                          type: @5
                                         op 0: @6
                                                         op 1: @7
@4
        type_decl
                          name: @8
                                         type: @2
@5
        integer_type
                                         size: @10
                                                         algn: 32
                          name: @9
                                          sign: signed
                                                         min : @11
                          prec: 32
                          max : @12
06
        var decl
                                         type: @5
                                                         srcp: t1.c:1
                          name: 013
                          size: @10
                                          algn: 32
                                                         used: 1
@7
        integer_cst
                          type: @5
                                         low: 55
08
        identifier_node
                          strg: void
                                         lngt: 4
@9
        type_decl
                          name: @14
                                         type: @5
                                         low: 32
@10
        integer_cst
                          type: @15
@11
                                                         low: -2147483648
        integer_cst
                          type: @5
                                         high: -1
@12
        integer_cst
                          type: @5
                                         low: 2147483647
@13
        identifier_node
                          strg: a
                                         lngt: 1
@14
        identifier_node
                          strg: int
                                         lngt: 3
@15
        integer_type
                          name: @16
                                         size: @17
                                                         algn: 64
                                          sign: unsigned min: @18
                          prec: 64
                          max : @19
@16
        identifier node
                          strg: bit_size_type
                                                        lngt: 13
@17
        integer_cst
                          type: @15
                                         low: 64
@18
        integer_cst
                          type: @15
                                         low : 0
@19
        integer_cst
                         type: @15
                                         low : -1
```

Abstract Syntax Tree

```
:: Function main (null)
                              ;; enabled by -tree-original
                              @1
                                      bind_expr
                                                       type: @2
                                                                       body:__@3
                              @2
                                      void_type
                                                                       algn: 8
                                                       name: @4
                              @3
                                      modify_expr
                                                       type: @5
                                                                       op 0: @6
                                                                                      op 1: @7
                              @4
                                      type_decl
                                                       name: @8
                                                                       type: @2
                              @5
                                      integer_type
                                                                       size: @10
                                                                                      algn: 32
                                                       name: @9
                                                                       sign: signed
                                                                                      min : @11
                                                        prec: 32
                                                        max : @12
int a;
                              06
                                      var decl
                                                                       type: @5
                                                                                      srcp: t1.c:1
                                                       name: 013
                                                        size: @10
                                                                       algn: 32
                                                                                       used: 1
int main()
                              @7
                                      integer_cst
                                                       type: @5
                                                                       low: 55
{
                              08
                                      identifier_node
                                                       strg: void
                                                                       lngt: 4
                              @9
                                      type_decl
                                                       name: @14
                                                                       type: @5
      a = 55;
                                                                       low: 32
                              @10
                                      integer_cst
                                                       type: @15
                              @11
                                                                                      low: -2147483648
                                      integer_cst
                                                       type: @5
                                                                       high: -1
}
                              @12
                                      integer_cst
                                                       type: @5
                                                                       low: 2147483647
                              @13
                                      identifier_node
                                                       strg: a
                                                                       lngt: 1
                              @14
                                      identifier_node
                                                       strg: int
                                                                       lngt: 3
                              @15
                                      integer_type
                                                       name: @16
                                                                       size: @17
                                                                                      algn: 64
                                                                       sign: unsigned min: @18
                                                        prec: 64
                                                        max : @19
                              @16
                                      identifier node
                                                       strg: bit_size_type
                                                                                      lngt: 13
                              @17
                                      integer_cst
                                                       type: @15
                                                                       low: 64
```

integer_cst

integer_cst

type: @15

type: @15

@18

@19

test.c.003t.original

13/70

low : 0

low : -1

Abstract Syntax Tree

test.c.003t.original

```
int a;
int main()
{
    a = 55;
}
```

```
:: Function main (null)
;; enabled by -tree-original
@1
        bind_expr
                          type: @2
                                         body:__@3
@2
        void_type
                                         algn: 8
                          name. @4
@3
        modify_expr
                          type: @5
                                         op 0: 06
                                                         op 1: @7
@4
        type_decl
                          name: @8
                                         type: @2
05
        integer_type
                          mame: @9
                                         size: @10
                                                         algn: 32
                          prec: 32
                                          sign: signed
                                                         min : @11
                          max : @12
        var_decl
@6
                                         type: @5
                                                         srcp: t1.c:1
                          name: 013
                          size: @10
                                          algn: 32
                                                         used: 1
@7
        integer_cst
                          type: @5
                                         low: 55
08
        identifier_node
                          strg: void
                                         lngt: 4
@9
        type_decl
                          name: @14
                                         type: @5
        integer_cst
                                         low: 32
@10
                          type: @15
@11
                                                         low: -2147483648
        integer_cst
                          type: @5
                                         high: -1
@12
        integer_cst
                          type: @5
                                         low: 2147483647
@13
        identifier_node
                          strg: a
                                         lngt: 1
@14
        identifier_node
                          strg: int
                                         lngt: 3
@15
        integer_type
                          name: @16
                                         size: @17
                                                         algn: 64
                                          sign: unsigned min: @18
                          prec: 64
                          max : @19
@16
        identifier node
                          strg: bit_size_type
                                                        lngt: 13
@17
        integer_cst
                          type: @15
                                         low: 64
@18
        integer_cst
                          type: @15
                                         low : 0
@19
        integer_cst
                         type: @15
                                         low : -1
```

Abstract Syntax Tree

```
test.c.003t.original
                             :: Function main (null)
                             ;; enabled by -tree-original
                             @1
                                     bind_expr
                                                      type: @2
                                                                      body:__@3
                             @2
                                     void_type
                                                                      algn: 8
                                                       name. @4
                             @3
                                     modify_expr
                                                      type: @5
                                                                      op 0: 06
                                                                                     op 1: @7
                             @4
                                     type_decl
                                                      name: @8
                                                                      type: @2
                             05
                                     integer_type
                                                      mame: @9
                                                                      size: @10
                                                                                     algn: 32
                                                       prec: 32
                                                                      sign: signed
                                                                                     min : @11
                                                       max : @12
int a;
                                     var_decl
                             @6
                                                                      type: @5
                                                                                     srcp: t1.c:1
                                                       name: @13-
                                                       size: @10
                                                                      algn: 32
                                                                                     used: 1
int main()
                             @7
                                     integer_cst
                                                      type: @5
                                                                      low: 55
{
                             08
                                     identifier_node
                                                      strg: void
                                                                      lngt: 4
                             @9
                                     type_decl
                                                      name: @14
                                                                      type: @5
      a = 55;
                                                                      low: 32
                             @10
                                     integer_cst
                                                      type: @15
                             @11
                                                                                     low: -2147483648
                                     integer_cst
                                                      type: @5
                                                                      high: -1
}
                             @12
                                     integer_cst
                                                      type: @5
                                                                      low: 2147483647
                             @13
                                     identifier_node
                                                      strg: a←
                                                                      lngt: 1
                             @14
                                     identifier_node
                                                      strg: int
                                                                      lngt: 3
                             @15
                                     integer_type
                                                      name: @16
                                                                      size: @17
                                                                                     algn: 64
                                                                      sign: unsigned min: @18
                                                       prec: 64
                                                       max : @19
                             @16
                                     identifier node
                                                      strg: bit_size_type
                                                                                    lngt: 13
                             @17
                                     integer_cst
                                                      type: @15
                                                                      low: 64
```

integer_cst

integer_cst

type: @15

type: @15

@18

@19

13/70

low : 0

low : -1

test.c.003t.original

Abstract Syntax Tree

```
:: Function main (null)
                              ;; enabled by -tree-original
                             @1
                                      bind_expr
                                                       type: @2
                                                                      body:__@3
                             @2
                                      void_type
                                                                       algn: 8
                                                       name. @4
                             @3
                                      modify_expr
                                                       type: @5
                                                                       op 0: 06
                                                                                      op 1: 07
                             @4
                                      type_decl
                                                       name: @8
                                                                      type: @2
                                                                                      algn: 32
                              05
                                      integer_type
                                                       mame: @9
                                                                      size: @10
                                                       prec: 32
                                                                       sign: signed
                                                                                      min : @11
                                                       max : @12
int a;
                                      var_decl
                                                                      type: @5
                             @6
                                                                                      srcp: t1.c:1
                                                       name: @13~
                                                       size: @10
                                                                       algn: 32
                                                                                      used: 1
int main()
                             @7
                                      integer_cst -
                                                       type: @5
                                                                       low: 55
{
                             08
                                      identifier node
                                                       strg: void
                                                                       lngt: 4
                             @9
                                      type_decl
                                                       name: @14
                                                                      type: @5
      a = 55;
                                      integer_cst
                                                                       low: 32
                             @10
                                                       type: @15
                             @11
                                                                                      low: -2147483648
                                      integer_cst
                                                       type: @5
                                                                      high: -1
}
                             @12
                                      integer_cst
                                                       type: @5
                                                                      low: 2147483647
                             @13
                                      identifier_node
                                                       strg: a←
                                                                      lngt: 1
                             @14
                                      identifier_node
                                                       strg: int
                                                                      lngt: 3
                             @15
                                      integer_type
                                                       name: @16
                                                                      size: @17
                                                                                      algn: 64
                                                                       sign: unsigned min: @18
                                                       prec: 64
                                                       max : @19
                             @16
                                      identifier node
                                                       strg: bit_size_type
                                                                                     lngt: 13
                             @17
                                      integer_cst
                                                       type: @15
                                                                      low: 64
                             @18
                                      integer_cst
                                                       type: @15
                                                                      low : 0
```

integer_cst

type: @15

@19

13/70

low : -1

Abstract Syntax Tree

```
test.c.003t.original
test.c
                              :: Function main (null)
                              ;; enabled by -tree-original
                             @1
                                     bind_expr
                                                      type: @2
                                                                      body:__@3
                             @2
                                     void_type
                                                                      algn: 8
                                                       name. @4
                             @3
                                     modify_expr
                                                      type: @5
                                                                      op 0: 06
                                                                                     op 1: 07
                             @4
                                     type_decl
                                                      name: @8
                                                                      type: @2
                                                                                     algn: 32
                              05
                                     integer_type
                                                      mame: @9
                                                                      size: @10
                                                       prec: 32
                                                                      sign: signed
                                                                                     min : @11
                                                       max : @12
int a;
                                     var_decl
                                                                      type: @5
                              06
                                                                                     srcp: t1.c:1
                                                       name: @13-
                                                       size: @10
                                                                      algn: 32
                                                                                     used: 1
int main()
                             @7
                                     integer_cst <---
                                                      type: @5
                                                                      low : 55
{
                             08
                                     identifier_node
                                                                      lngt 4
                                                      strg: void
                             @9
                                     type_decl
                                                      name: @14
                                                                      ±√be: @5
      a = 55;
                                                                      low: 32
                             @10
                                     integer_cst
                                                      type. @15
                             @11
                                                                                     low: -2147483648
                                     integer_cst
                                                      type: @5
                                                                      high: -1
}
                             @12
                                     integer_cst
                                                      type: @5
                                                                      low: 2147483647
                             @13
                                     identifier_node
                                                      strg: a←
                                                                      lngt: 1
                             @14
                                      identifier_node
                                                      strg: int
                                                                      lngt: 3
                             @15
                                     integer_type
                                                      name: @16
                                                                      size: @17
                                                                                     algn: 64
                                                                      sign: unsigned min : @18
                                                       prec: 64
                                                       max : @19
                             @16
                                     identifier node
                                                      strg: bit_size_type
                                                                                    lngt: 13
                             @17
                                     integer_cst
                                                      type: @15
                                                                      low: 64
                             @18
                                     integer_cst
                                                      type: @15
                                                                      low : 0
                             @19
                                     integer_cst
                                                      type: @15
                                                                      low : -1
```

Part 3

Examining GIMPLE Dumps for C

Gimplifier

- About GIMPLE
 - Three-address representation derived from GENERIC Computation represented as a sequence of basic operations Temporaries introduced to hold intermediate values
 - Control construct are explicated into conditional jumps
- Examining GIMPLE Dumps
 - Examining translation of data accesses
 - Examining translation of control flow
 - Examining translation of function calls

test.c.004t.gimple

test.c

int a;

15/70

GIMPLE: Composite Expressions Involving Scalar Variables

Global variables are treated as "memory locations" and local variables are treated as "registers"

test.c.004t.gimple

x = 10:

test.c

int a;

int main()

15/70

GIMPLE: Composite Expressions Involving Scalar Variables

treated as "registers"

15/70

test.c

```
int a:
int main()
  int x = 10;
  int v = 5;
 x = a + x * y;
  y = y - a * x;
}
```

```
a.0 = a;
x = D.1954 + a.0;
a.1 = a:
D.1957 = a.1 * x;
y = y - D.1957;
```

test.c.004t.gimple

D.1954 = x * y;

x = 10:

y = 5;

Global variables are treated as "memory locations" and local variables are treated as "registers"

test.c.004t.gimple

test.c

int a:

15/70

GIMPLE: Composite Expressions Involving Scalar Variables

```
int main()
{
  int x = 10;
  int y = 5;
  int y = 5;
  x = a + x * y;
  y = y - a * x;
}

x = 10;
y = 5;
D.1954 = x * y;
a.0 = a;
x = D.1954 + a.0;
a.1 = a;
D.1957 = a.1 * x;
y = y - D.1957;
}
```

Global variables are treated as "memory locations" and local variables are treated as "registers" $\,$

29 Jun 2013

16/70

GIMPLE: 1-D Array Accesses test.c.004t.gimple

```
int main()
  int a[3], x;
 a[1] = a[2] = 10;
 x = a[1] + a[2];
 a[0] = a[1] + a[1]*x;
```

```
try {
    a[2] = 10;
    D.1952 = a[2];
    a[1] = D.1952;
    D.1953 = a[1];
    D.1954 = a[2];
   x = D.1953 + D.1954;
    D.1955 = x + 1;
    D.1956 = a[1]:
    D.1957 = D.1955 * D.1956;
    a[0] = D.1957;
finally {
    a = {CLOBBER};
```

29 Jun 2013

16/70

test.c.004t.gimple

```
int main()
  int a[3], x;
 a[1] = a[2] = 10;
 x = a[1] + a[2];
 a[0] = a[1] + a[1]*x;
```

```
try {
    a[2] = 10:
    D.1952 = a[2];
    a[1] = D.1952;
   D.1953 = a[1];
   D.1954 = a[2];
   x = D.1953 + D.1954;
   D.1955 = x + 1;
   D.1956 = a[1]:
   D.1957 = D.1955 * D.1956;
    a[0] = D.1957;
finally {
    a = {CLOBBER};
```

29 Jun 2013

int main()

```
int a[3], x;
a[1] = a[2] = 10;
x = a[1] + a[2];
a[0] = a[1] + a[1]*x;
```

```
try {
    a[2] = 10;
   D.1952 = a[2];
    a[1] = D.1952;
   D.1953 = a[1];
   D.1954 = a[2];
   x = D.1953 + D.1954;
   D.1955 = x + 1;
   D.1956 = a[1]:
   D.1957 = D.1955 * D.1956;
    a[0] = D.1957;
finally {
    a = {CLOBBER};
```

16/70

test.c.004t.gimple

test.c.004t.gimple

29 Jun 2013

```
int main()
      int a[3], x;
      a[1] = a[2] = 10;
      x = a[1] + a[2];
      a[0] = a[1] + a[1]*x;
Essential Abstractions in GCC
```

```
try {
    a[2] = 10;
   D.1952 = a[2];
    a[1] = D.1952;
   D.1953 = a[1];
   D.1954 = a[2];
   x = D.1953 + D.1954;
   D.1955 = x + 1;
   D.1956 = a[1]:
   D.1957 = D.1955 * D.1956;
    a[0] = D.1957;
finally {
    a = {CLOBBER};
```

int main()

29 Jun 2013

```
test.c.004t.gimple
                           try {
                               a[2] = 10;
                               D.1952 = a[2];
                               a[1] = D.1952;
                               D.1953 = a[1];
                               D.1954 = a[2];
int a[3], x;
                               x = D.1953 + D.1954;
a[1] = a[2] = 10;
                               D.1955 = x + 1;
x = a[1] + a[2];
                               D.1956 = a[1]:
a[0] = a[1] + a[1]*x;
                               D.1957 = D.1955 * D.1956;
                               a[0] = D.1957;
                           finally {
                               a = {CLOBBER};
```

test.c.004t.gimple

test.c

29 Jun 2013

17/70

O..... 221 2

```
a[0][0] = 7;
                                   a[1][1] = 8;
int main()
                                   a[2][2] = 9;
 int a[3][3], x, y;
                                   D.1953 = a[0][0];
                                   D.1954 = a[1][1];
 a[0][0] = 7;
                                  x = D.1953 / D.1954;
 a[1][1] = 8;
                                   D.1955 = a[1][1];
 a[2][2] = 9;
 x = a[0][0] / a[1][1];
                                   D.1956 = a[2][2];
 y = a[1][1] \% a[2][2];
                                   y = D.1955 \% D.1956;
                                   a = {CLOBBER};
```

```
int main()
₹
  int **a, *b, c;
  b = \&c;
  a = \&b;
  **a = 10; /* c = 10 */
}
```

```
main () {
    int * D.1953;
    int * * a;
    int * b;
    int c;
    try
      b = &c;
      a = \&b;
      D.1953 = *a;
      *D.1953 = 10;
    finally {
        b = \{CLOBBER\};
        c = \{CLOBBER\};
```

```
int main()
₹
  int **a, *b, c;
  b = \&c;
  a = \&b;
  **a = 10; /* c = 10 */
}
```

```
main () {
    int * D.1953;
    int * * a;
    int * b;
    int c;
    try
      b = &c;
      a = \&b;
      D.1953 = *a;
      *D.1953 = 10;
    finally {
        b = \{CLOBBER\};
        c = \{CLOBBER\};
```

GCC Resource Center, IIT Bombay

Memory and Registers in GIMPLE

- Memory: Gobals, address taken variables, arrays
 - Scalar memory values must be explicitly loaded into registers
 a.0 = a;
 - No "addressable" memory within arrays
 - No base + offset modelling of arrays
 - Array reference is a single operation in GIMPLE
 - Since "memory" survives the lifetime of a given scope, locals are marked as clobbered at the end of the scope
- Registers: Locals, formals
 - Restricted visibility
 - ► Cannot be modified by function calls or a concurrent process
 - Can be freely rearranged

29 Jun 2013

test.c

} ad;

} st;

}

{ int roll;

ad *ct;

20/70

typedef struct address { char *name;

typedef struct student

int main()

{ st *s: s = malloc(sizeof(st)): $s\rightarrow roll = 1:$

s->ct=malloc(sizeof(ad)); s->ct->name = "Mumbai";

extern void * malloc (unsigned int);

main ()

struct ad * D.1958; struct st * s;

D.1958 = s->ct;

void * D.1957;

test.c.004t.gimple

s = malloc(8);s->roll = 1;

D.1957 = malloc (4);s->ct = D.1957;

D.1958->name = "Mumbai";

29 Jun 2013

test.c

} ad;

}

typedef struct address { char *name;

typedef struct student

{ int roll; ad *ct;

} st; int main()

{ st *s: s = malloc(sizeof(st)):

 $s\rightarrow roll = 1:$ s->ct=malloc(sizeof(ad)); s->ct->name = "Mumbai";

D.1957 = malloc (4);s->ct = D.1957;D.1958 = s->ct;

s->roll = 1;

test.c.004t.gimple

void * D.1957;

struct st * s;

s = malloc(8);

struct ad * D.1958;

main ()

extern void * malloc (unsigned int);

D.1958->name = "Mumbai";

{ char *name;

{ int roll;

ad *ct;

29 Jun 2013

} ad;

} st;

```
int main()
{ st *s:
  s = malloc(sizeof(st)):
  s\rightarrow roll = 1:
  s->ct=malloc(sizeof(ad));
  s->ct->name = "Mumbai";
}
```

```
typedef struct address
                              main ()
                                void * D.1957;
typedef struct student
                                struct ad * D.1958;
                                struct st * s;
                                extern void * malloc (unsigned int);
                                s = malloc(8);
                                s->roll = 1;
                                D.1957 = malloc (4);
                                s->ct = D.1957;
                                D.1958 = s->ct;
                                D.1958->name = "Mumbai";
```

extern void * malloc (unsigned int);

{ char *name;

29 Jun 2013

} ad;

} st;

}

{ st *s:

typedef struct student

{ int roll; ad *ct;

typedef struct address

int main()

s = malloc(sizeof(st)):

 $s\rightarrow roll = 1:$

s->ct=malloc(sizeof(ad)); s->ct->name = "Mumbai";

s->roll = 1;

test.c.004t.gimple

void * D.1957;

struct st * s;

s = malloc(8);

struct ad * D.1958;

main ()

D.1957 = malloc (4);s->ct = D.1957;

D.1958 = s->ct;

D.1958->name = "Mumbai";

 $*(p_a+2) = 30;$

test.c

```
int main()
  int *p_a, a[3];
  p_a = &a[0];
  *p_a = 10;
  *(p_a+1) = 20;
```

```
int * p_a;
int a[3];
    p_a = &a[0];
    *p_a = 10;
    D.2048 = p_a + 4;
    *D.2048 = 20;
    D.2049 = p_a + 8;
    *D.2049 = 30;
finally {
    a = \{CLOBBER\};
```

test.c.004t.gimple

int * D.2048; int * D.2049;

main ()

try {

21/70

test.c

int main()

```
int *p_a, a[3];
```

```
p_a = &a[0];
*p_a = 10;
```

```
*(p_a+1) = 20;
*(p_a+2) = 30;
```

```
try {
    p_a = &a[0];
    *p_a = 10;
    D.2048 = p_a + 4;
```

finally {

int $* p_a;$ int a[3];

test.c.004t.gimple

int * D.2048; int * D.2049;

main ()

int main()

test.c

```
int *p_a, a[3];
```

```
p_a = &a[0];
*p_a = 10;
*(p_a+1) = 20;
*(p_a+2) = 30;
```

```
int * D.2048;
int * D.2049;
int * p_a;
int a[3];
try {
    p_a = &a[0];
    *p_a = 10;
    D.2048 = p_a + 4;
    *D.2048 = 20;
    D.2049 = p_a + 8;
    *D.2049 = 30;
finally {
    a = \{CLOBBER\};
```

GCC Resource Center, IIT Bombay

test.c.004t.gimple

main ()

 $*p_a = 10;$

int main()

test.c

```
int *p_a, a[3];
```

```
p_a = &a[0];
*(p_a+1) = 20;
*(p_a+2) = 30;
```

```
int * D.2048;
int * D.2049;
int * p_a;
int a[3];
try {
    p_a = &a[0];
    *p_a = 10;
    D.2048 = p_a + 4;
    *D.2048 = 20;
    D.2049 = p_a + 8;
    *D.2049 = 30;
finally {
    a = \{CLOBBER\};
```

test.c.004t.gimple

main ()

GIMPLE: Translation of Conditional Statements

```
test.c.004t.gimple
int main()
{
    int a=2, b=3, c=4;
    while (a <= 7)
                                if (a <= 12) goto <D.1200>;
                                else goto <D.1201>;
        a = a+1;
     if (a \le 12)
 a = a + b + c;
                                a = D.1199 + c;
```

GIMPLE: Translation of Conditional Statements

```
test.c.004t.gimple
int main()
{
    int a=2, b=3, c=4;
    while (a <= 7)
                                if (a <= 12) goto <D.1200>;
                                else goto <D.1201>;
        a = a+1;
                                D.1199 = a + b;
     if (a \le 12)
 a = a + b + c;
                                a = D.1199 + c;
```

GIMPLE: Translation of Conditional Statements

```
test.c.004t.gimple
int main()
{
    int a=2, b=3, c=4;
    while (a <= 7)
                               if (a <= 12) goto <D.1200>;
                               else goto <D.1201>;
        a = a+1;
     if (a<=12)
a = a+b+c;
```

GIMPLE: Translation of Loops

```
test.c
                                   test.c.004t.gimple
int main()
{
    int a=2, b=3, c=4;
while (a<=7)</pre>
                                   goto <D.1197>;
                                   if (a <= 7) goto <D.1196>;
                                   else goto <D.1198>;
                                   <D.1198>:
    if (a <= 12)
         a = a+b+c;
```

GIMPLE: Translation of Loops

```
test.c
                                  test.c.004t.gimple
int main()
{
    int a=2, b=3, c=4;
while (a<=7)</pre>
                                  goto <D.1197>;
                                  <D.1196>:
                                  if (a <= 7) goto <D.1196>;
                                  else goto <D.1198>;
                                  <D.1198>:
    if (a<=12)
         a = a+b+c;
```

```
test.c
int main()
{
   int a=2, b=3, c=4;
     while (a<=7)
          a = a+1;
    if (a<=12)
       a = a+b+c;
```

```
goto <D.1197>;
<D.1196>:
if (a <= 7) goto <D.1196>;
else goto <D.1198>;
<D.1198>:
```

test.c.004t.gimple

GIMPLE: Translation of Loops

```
test.c
                             test.c.004t.gimple
int main()
{
    int a=2, b=3, c=4;
                             goto <D.1197>;
                             <D.1196>:
     while (a <= 7)
                              if (a <= 7) goto <D.1196>;
           a = a+1;
                             else goto <D.1198>;
    if (a<=12)
        a = a+b+c;
```

Control Flow Graph: Textual View

```
<bb 5>:
                                      if (a <= 12)
                                        goto <bb 6>;
                                      else
if (a <= 12) goto <D.1200>;
                                        goto <bb 7>;
else goto <D.1201>;
<D.1200>:
                                    <bb 6>:
D.1199 = a + b;
                                      D.1199 = a + b;
a = D.1199 + c;
                                      a = D.1199 + c;
<D.1201>:
                                      return;
```

test.c.004t.gimple

Control Flow Graph: Textual View

```
<bb 5>:
                                      if (a <= 12)
                                        goto <bb 6>;
                                      else
if (a <= 12) goto <D.1200>;
                                        goto <bb 7>;
else goto <D.1201>;
<D.1200>:
                                    <bb 6>:
D.1199 = a + b;
                                      D.1199 = a + b;
a = D.1199 + c;
                                      a = D.1199 + c;
<D.1201>:
                                      return;
```

test.c.004t.gimple

```
<bb 5>:
                                      if (a <= 12)
                                        goto <bb 6>;
                                      else
if (a <= 12) goto <D.1200>;
                                        goto <bb 7>;
else goto <D.1201>;
<D.1200>:
                                    <bb 6>:
D.1199 = a + b;
                                      D.1199 = a + b;
a = D.1199 + c;
                                      a = D.1199 + c;
<D.1201>:
                                      return;
```

test.c.004t.gimple

Control Flow Graph: Textual View

```
<bb 5>:
                                      if (a <= 12)
                                        goto <bb 6>;
                                      else
if (a <= 12) goto <D.1200>;
                                        goto <bb 7>;
else goto <D.1201>;
<D.1200>:
                                    <bb 6>:
D.1199 = a + b;
                                      D.1199 = a + b;
a = D.1199 + c;
                                      a = D.1199 + c;
<D.1201>:
                                      return:
```

test.c.004t.gimple

24/70

test.c.014t.cfg

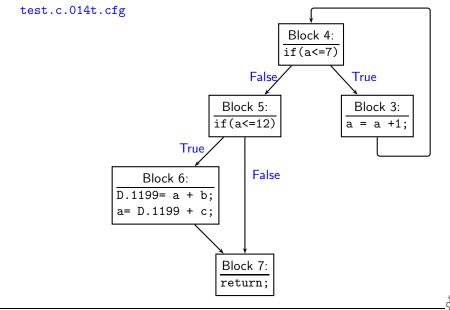
```
if (a <= 12) goto <D.1200>;
else goto <D.1201>;
<D.1200>:
D.1199 = a + b;
a = D.1199 + c;
<D.1201>:
```

test.c.004t.gimple

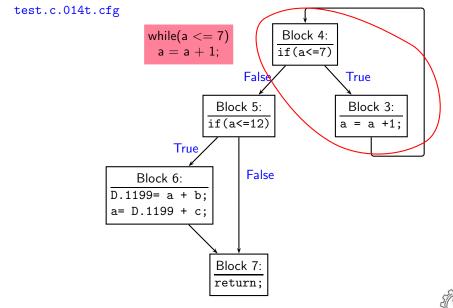
```
if (a <= 12)
    goto <bb 6>;
  else
    goto <bb 7>;
<bb 6>:
 D.1199 = a + b;
  a = D.1199 + c;
  return:
```

<bb 5>:

Control Flow Graph: Pictorial View

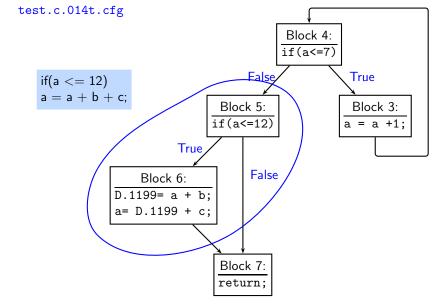


Control Flow Graph: Pictorial View



25/70

Control Flow Graph: Pictorial View



26/70

GIMPLE: Function Calls and Call Graph

test.c.000i.cgraph

```
printf/3 @0x7fd094bbba20 availabilit
                                called by: main/1 (1.00 per call)
extern int divide(int, int);
                                calls:
int multiply(int a, int b)
   return a*b;
int main()
{ int x,y;
  x = divide(20,5);
  y = multiply(x,2);
 printf("%d\n", y);
                                called by: main/1 (1.00 per call)
                                calls:
```

```
divide/2 @0x7fd094bbb900 availabilit
  called by: main/1 (1.00 per call)
  calls:
main/1 @0x7fd094bbb7e0 (asm: main) a
  called by:
  calls: printf/3 (1.00 per call)
         multiply/0 (1.00 per call)
         divide/2 (1.00 per call)
multiply/0 00x7fd094bbb6c0 (asm: mul
```

test.c.000i.cgraph

29 Jun 2013

test.c

26/70

GIMPLE: Function Calls and Call Graph

calls:

calls:

```
printf/3 @0x7fd094bbba20 availabilit
extern int divide(int, int);
int multiply(int a, int b)
                              divide/2 @0x7fd094bbb900 availabilit
   return a*b;
int main()
{ int x,y;
  x = divide(20,5);
  y = multiply(x,2);
  printf("%d\n", y);
```

```
called by: main/1 (1.00 per call)
  calls:
main/1 @0x7fd094bbb7e0 (asm: main) a
  called by:
  calls: printf/3 (1.00 per call)
         multiply/0 (1.00 per call)
         divide/2 (1.00 per call)
multiply/0 @0x7fd094bbb6c0 (asm: mul
```

called by: main/1 (1.00 per call)

called by: main/1 (1.00 per call)

call graph

27/70

test.c

GIMPLE: Function Calls and Call Graph

test.c.000i.cgraph

```
printf/3
extern int divide(int, int);
                                 called by: main/1
int multiply(int a, int b)
                                 calls:
                               divide/2
                                                           main
   return a*b;
                                 called by: main/1
                                 calls:
                               main/1
                                                      printf
                                                               divide
int main()
                                 called by:
                                 calls: printf/3
{ int x,y;
                                                          multiply
  x = divide(20,5);
                                         multiply/0
  y = multiply(x,2);
                                         divide/2
  printf("%d\n", y);
                               multiply/0
                                 called by: main/1
                                 calls:
```

call graph

27/70

test.c

test.c.000i.cgraph

```
printf/3
extern int divide(int, int);
                                 called by: main/1
int multiply(int a, int b)
                                 calls:
                               divide/2
                                                           main
   return a*b;
                                 called by: main/1
                                 calls:
                               main/1
                                                      printf
                                                               divide
int main()
                                 called by:
                                 calls: printf/3
{ int x,y;
                                                          multiply
  x = divide(20,5);
                                         multiply/0
  y = multiply(x,2);
                                         divide/2
  printf("%d\n", y);
                               multiply/0
                                 called by: main/1
                                 calls:
```

else return (!odd(n-1));

int even(int n)

```
{ if (n == 0) return 1;
```

int odd(int n) { if (n == 1) return 1; else return (!even(n-1));

main()

test.c

{ int n;

n = abs(readNumber()); if (even(n)) printf ("n is even\n");

else printf ("n is odd\n");

Essential Abstractions in GCC

call graph

main (readNumber) (abs) even odd

print

Graybox Probing: Examining GIMPLE Dumps for C

Inspect GIMPLE When in Doubt (1)

29 Jun 2013

Graybox Probing: Examining GIMPLE Dumps for C

Inspect GIMPLE When in Doubt (1)

29/70

29 Jun 2013

29/70

29 Jun 2013

What are the values of x and y? x = 10, y = 5

x = 2; y = 3; x = x + 1; D.1572 = y + x; y = y + 1; x = D.1572 + y;

y = y + 1;

Essential Abstractions in GCC

GCC Resource Center, IIT Bombay

Inspect GIMPLE When in Doubt (1)

```
int x=2, y=3;
x = y++ + ++x + ++y;
```

What are the values of x and y? x = 10, y = 5

```
x = 2;
y = 3;
                                                          pre
x = x + 1;
D.1572 = y + x;
                                     post
y = y + 1;
x = D.1572 + y;
y = y + 1;
```

Inspect GIMPLE When in Doubt (1)

What are the values of x and y? x = 10, y = 5

```
x = 2;
y = 3;
                                                         pre
x = x + 1;
D.1572 = y + x;
                                    post
y = y + 1;
x = D.1572 + y;
y = y + 1;
```

Inspect GIMPLE When in Doubt (1)

```
int x=2, y=3;
x = y++ + ++x + ++y;
```

What are the values of x and y? x = 10, y = 5

```
x = 2;
y = 3;
                                                         pre
x = x + 1;
D.1572 = y + x;
                                     post
y = y + 1;
x = D.1572 + y;
y = y + 1;
```

```
x = 2;
y = 3;
                                                          pre
x = x + 1;
D.1572 = y + x;
                                                  pre
                                     post
y = y + 1;
x = D.1572 + y;
y = y + 1;
```

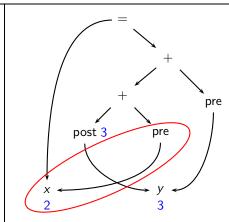
What are the values of x and y? x = y++ + ++x + ++y;x = 10, y = 5

```
x = 2;
y = 3;
                                                          pre
x = x + 1;
D.1572 = y + x;
                                     post 3
                                                  pre
y = y + 1;
x = D.1572 + y;
y = y + 1;
```

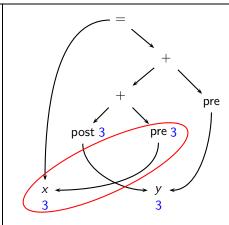
$$x = y++ + ++x + ++y;$$

int x=2, y=3;

```
x = 2;
y = 3;
x = x + 1;
D.1572 = y + x;
y = y + 1;
x = D.1572 + y;
y = y + 1;
```



```
x = 2;
y = 3;
x = x + 1;
D.1572 = y + x;
y = y + 1;
x = D.1572 + y;
y = y + 1;
```



pre

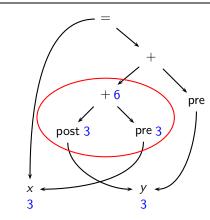
Inspect GIMPLE When in Doubt (1)

```
int x=2, y=3;
x = y++ + ++x + ++y;
```

```
x = 2;
y = 3;
x = x + 1;
D.1572 = y + x;
                                     post 3
                                                 pre 3
y = y + 1;
x = D.1572 + y;
y = y + 1;
```

int x=2, y=3;

```
x = 2;
y = 3;
x = x + 1;
D.1572 = y + x;
y = y + 1;
x = D.1572 + y;
y = y + 1;
```



Graybox Probing: Examining GIMPLE Dumps for C

```
x = y++ + ++x + ++y;
```

int x=2, y=3;

```
x = 2;
y = 3;
                                                         pre
x = x + 1;
D.1572 = y + x;
                                    post
y = y + 1;
x = D.1572 + y;
y = y + 1;
```

```
x = 2;
y = 3;
                                                          pre 4
x = x + 1;
D.1572 = y + x;
                                     post
y = y + 1;
x = D.1572 + y;
y = y + 1;
```

int x=2, y=3;

```
x = 2;
y = 3;
                                                          pre 4
x = x + 1;
D.1572 = y + x;
                                     post
                                                  pre
y = y + 1;
x = D.1572 + y;
y = y + 1;
```

int x=2, y=3;

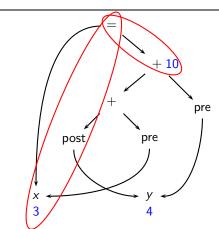
```
+ 10
x = 2;
y = 3;
                                                          pre 4
x = x + 1;
D.1572 = y + x;
                                     post
                                                  pre
y = y + 1;
x = D.1572 + y;
y = y + 1;
```

Graybox Probing: Examining GIMPLE Dumps for C

```
int x=2, y=3;
x = y++ + ++x + ++y;
```

What are the values of x and y?
$$x = 10$$
, $y = 5$

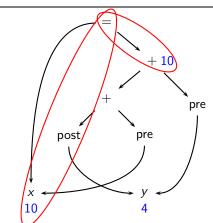
```
x = 2;
y = 3;
x = x + 1;
D.1572 = y + x;
y = y + 1;
x = D.1572 + y;
y = y + 1;
```



```
x = y++ + ++x + ++y;
```

int x=2, y=3;

```
x = 2;
y = 3;
x = x + 1;
D.1572 = y + x;
y = y + 1;
x = D.1572 + y;
y = y + 1;
```

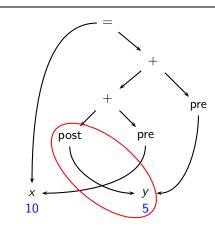


```
x = 2;
y = 3;
                                                           pre
x = x + 1;
D.1572 = y + x;
                                                  pre
                                     post
y = y + 1;
x = D.1572 + y;
y = y + 1;
                                10
```

$$x = y++ + ++x + ++y;$$

int x=2, y=3;

```
x = 2;
y = 3;
x = x + 1;
D.1572 = y + x;
y = y + 1;
x = D.1572 + y;
y = y + 1;
```



```
int x=2, y=3;
x = y++ + ++x + ++y;
```

```
x = 2;
y = 3;
                                                          pre
x = x + 1;
D.1572 = y + x;
                                     post
y = y + 1;
x = D.1572 + y;
y = y + 1;
                                10
```

- How is a[i] = i++ handled? This is an undefined behaviour as per C standards.
- What is the order of parameter evaluation? For a call f(getX(), getY()), is the order left to right? arbitrary? Is the evaluation order in GCC consistent?
- Understanding complicated declarations in C can be difficult What does the following declaration mean:

```
int * (* (*MYVAR) (int) ) [10];
```

Hint: Use -fdump-tree-original-raw-verbose option. The dump to see is 003t.original

30/70

Part 4

Examining RTL Dumps

29 Jun 2013

Graybox Probing: Examining RTL Dumps

Will be Covered Before Machine Descriptions



31/70

Part 5

Examining Assembly Dumps

Dump file: test.s

jmp

```
.L3:
     addl $1, -4(%rbp)
.L2:
     cmpl $7, -4(\%rbp)
     jle .L3
     cmpl $12, -4(\%rbp)
          .L5
     jg
     movl -8(%rbp), %eax
     movl -4(\%rbp), %edx
     addl %eax, %edx
     movl -12(%rbp), %eax
     addl %edx, %eax
     movl %eax, -4(%rbp)
.L5:
```

.L2

```
while (a <= 7)
{
    a = a+1;
}
if (a <= 12)
{
    a = a+b+c;
}</pre>
```

29 Jun 2013

32/70

```
.L3:
     addl $1, -4(\%rbp)
.L2:
     cmpl $7, -4(\%rbp)
     jle .L3
     cmpl $12, -4(\%rbp)
          . I.5
     jg
     movl -8(\%rbp), \%eax
     movl -4(\%rbp), %edx
     addl %eax, %edx
     movl -12(%rbp), %eax
     addl %edx, %eax
     movl %eax, -4(%rbp)
.L5:
```

```
while (a \le 7)
    a = a+1;
if (a \le 12)
    a = a+b+c;
```

29 Jun 2013

32/70

Dump file: test.s

jmp

```
.L3:
     addl $1, -4(%rbp)
.L2:
     cmpl $7, -4(\%rbp)
     jle .L3
     cmpl $12, -4(\%rbp)
     jg .L5
     movl -8(%rbp), %eax
     movl -4(\%rbp), %edx
     addl %eax, %edx
     movl -12(%rbp), %eax
     addl %edx, %eax
     movl %eax, -4(%rbp)
.L5:
```

.L2

```
while (a \le 7)
    a = a+1;
if (a \le 12)
    a = a+b+c;
}
```

Graybox Probing: Examining Assembly Dumps

29 Jun 2013

32/70

jmp .L2

Dump file: test.s

```
.L3:
     addl $1, -4(%rbp)
.L2:
     cmpl $7, -4(\%rbp)
     jle .L3
     cmpl $12, -4(\%rbp)
          . I.5
     ig
     movl -8(\%rbp), \%eax
     movl -4(\%rbp), %edx
     addl %eax, %edx
     movl -12(%rbp), %eax
     addl %edx, %eax
     movl %eax, -4(%rbp)
.L5:
```

```
while (a \le 7)
    a = a+1;
if (a \le 12)
    a = a+b+c;
```

Part 6

Understanding C++ Translation

29 Jun 2013

33/70

- Internal representation of a class
- Encoding function prototypes for type checking and resolving overloading
- Handling templates
- Inheritance of data
- Inheritance of functions

```
public:
    int y,z;
    void f1(int i)
      \{ x = f2(i)*2; \}
  private:
    int x;
    int f2(int i)
      { return i+1;}
};
```

29 Jun 2013

```
Data Memory
```

Code Memory

Graybox Probing: Understanding C++ Translation

34/70

Essential Abstractions in GCC

Graybox Probing: Understanding C++ Translation

34/70

```
public:
    int y,z;
    void f1(int i)
        { x = f2(i)*2;}
private:
    int x;
    int f2(int i)
        { return i+1;}
```

29 Jun 2013

};

A a; A b;

```
Data Memory
```

Code Memory

•

Essential Abstractions in GCC

C GCC Resource Center, IIT Bombay

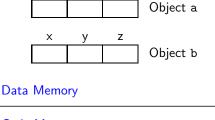
class A

29 Jun 2013

Χ

Graybox Probing: Understanding C++ Translation

```
public:
    int y,z;
    void f1(int i)
      \{ x = f2(i)*2; \}
  private:
    int x;
    int f2(int i)
      { return i+1;}
};
                          void
A a;
                            th
Ab;
                          int
                          { retun
```



Z

Code Memory

```
There is no
distinction between the
public and private data
      in memory
```

★his, int i)

int i)

34/70

class A

29 Jun 2013

Ab;

```
public:
    int y,z;
    void f1(int i)
      \{ x = f2(i)*2; \}
  private:
    int x;
    int f2(int i)
      { return i+1;}
};
A a;
```

```
Every function with n parameters
          is converted to a function of n+1
          parameter with the first parameter
          being the address of the object
Data Memory
```

Code Memory

Graybox Probing: Understanding C++ Translation

```
void A::f1(struct A * const this, int i)
\{ \text{ this-> x = A::} f2(\text{this, i})*2; \}
int A::f2(struct A * const this, int i)
{ return i+1;}
```

Z

Z

private:

int x;

29 Jun 2013

};

int f2(int i)

{ return i+1;}

A a;

b.f1(10);

Ab; a.f1(5); { this-> x = A::f2(this, i)*2;}

Code Memory

{ return i+1;}

Х

Data Memory

Object a

Object b

void A::f1(struct A * const this, int i)

34/70

int A::f2(struct A * const this, int i)

class A

29 Jun 2013

34/70

.

Graybox Probing: Understanding C++ Translation

```
Object a
  public:
    int y,z;
                                Х
                                           z
    void f1(int i)
                                                Object b
      \{ x = f2(i)*2; \}
  private:
                         Data Memory
    int x;
    int f2(int i)
                         Code Memory
      { return i+1;}
};
                         void A::f1(struct A * const (this), int i)
A a;
                          { this-> x = A::f2(this, i)*2;}
Ab;
                          int A::f2(struct A * const this, int i)
a.f1(5);
                          { return i+1;}
```

b.f1(10);

Graybox Probing: Understanding C++ Translation

class A

29 Jun 2013

```
public:
    int y,z;
    void f1(int i)
      \{ x = f2(i)*2; \}
  private:
    int x;
    int f2(int i)
      { return i+1;}
};
A a;
Ab;
a.f1(5);
```

```
Object a
                 Z
                      Object b
Data Men ory
Code Memory
void A::f1(struct A * const (this), int i)
{ this->(x) = A::f2(this, i)*2;}
int A::f2(struct A * const this, int i)
{ return i+1;}
```

b.f1(10);

Object a

34/70

Graybox Probing: Understanding C++ Translation

class A

Ab;

a.f1(5);

b.f1(10);

```
Z
                      Object b
Data Men ory
Code Memory
void A::f1(struct A * const (this), int i)
{ this->(x) = A::f2(this, i)*2;}
int A::f2(struct A * const this, int i)
{ return i+1;}
```

Graybox Probing: Understanding C++ Translation

};

A a;

Ab;

a.f1(5);

a.f1(10)

class A

public:

```
int y,z;
  void f1(int i)
    \{ x = f2(i)*2; \}
private:
  int x;
  int f2(int i)
    { return i+1;}
```

```
Object a
                 z
      Х
                      Object b)
Data Memory
Code Memory
void A::f1(struct A * const (this), int i)
{ this-> x = A::f2(this, i)*2;}
int A::f2(struct A * const this, int i)
{ return i+1;}
```

};

A a; Ab; a.f1(5);

```
Z
                      Object a
                 z
      Х
                      Object b
Data Memory
Code Memory
void A::f1(struct A * const (this), int i)
{ this->(x) = A::f2(this, i)*2;}
```

int A::f2(struct A * const this, int i)

Graybox Probing: Understanding C++ Translation

{ return i+1;}

a.f1(10)

Graybox Probing: Understanding C++ Translation

class A {

```
public:
    int y,z;
    void f1(int i)
      { x = f2(i)*2;}
private:
    int x;
    int f2(int i)
```

```
{ return i+1;}
};
A a;
```

```
A b;
a.f1(5);
a.f1(10);
```

```
Essential Abstractions in GCC
```

```
Z
                      Object a
                 z
      22
                      Object b
Data Memory
Code Memory
void A::f1(struct A * const (this), int i)
{ this->(x) = A::f2(this, i)*2;}
int A::f2(struct A * const this, int i)
{ return i+1;}
```

void A::f1(int) (struct A * const this, int i)

29 Jun 2013

<bb 2>:

D.1727 = 0;

return D.1727;

```
struct A b;
struct A a;
int D.1727;
```

dump file: test.cpp.014t.cfg

```
int D.1729;
<bb 2>:
 D.1729 = A::f2 (this, i);
 D.1730 = D.1729 * 2;
```

{ int D.1730;

```
this->x = D.1730;
                    return;
A::f1 (&a, 5);
A::f1 (&b, 10);
```

Graybox Probing: Understanding C++ Translation

Function Prototype

```
int A::add(int, int)
```

29 Jun 2013

```
float A::add(int, float)
```

```
float A::add(float, int)
```

GCC Resource Center, IIT Bomba

36/70

Function Prototype

29 Jun 2013

float A::add(int, float)

float A::add(float, int)

double A::add(float, float) _ZN1A3addEff

_ZN1A3addEii

Encoding

_ZN1A3addEif

4Eti

_ZN1A3addEfi

aeti

Eff

float A::add(float, int)

double A::add(float, float)

int A::add(int, int)

Function Prototype

29 Jun 2013

Encoding

ZN1A3addEif

_ZN1A3addEfi

7N1A3addFii

7N 1 A 3 add F ii

GCC Resource Center, IIT Bombay

Deconstruction

ZN 1 A 3 add E if

_ZN 1 A 3 add E fi

_ZN 1 A 3 add E ff

Storing Function Frototypes

Function Prototype Encoding Deconstruction 7N 1 A 3 add F ii int A::add(int, int) 7N1A3addFii ZN1A3addEif ZN 1 A 3 add E if float A::add(int, float) float A::add(float, int) _ZN1A3addEfi _ZN 1 A 3 add E fi double A::add(float, float) _ZN1A3addEff _ZN 1 A 3 add E ff Fixed prefix

Storing Function Prototypes

Function Prototype	Encoding	Deconstruction
<pre>int A::add(int, int)</pre>	_ZN1A3addEii	_ZN 1 A 3 add E ii
<pre>float A::add(int, float)</pre>	_ZN1A3addEif	_ZN 1 A 3 add E if
<pre>float A::add(float, int)</pre>	_ZN1A3addEfi	_ZN 1 A 3 add E fi
<pre>double A::add(float, float)</pre>	_ZN1A3addEff	_ZN 1 A 3 add E ff
No. of characters		
in the class name		

Function Prototype Encoding Deconstruction int A::add(int, int) _ZN1A3addEii ZN1A3addEif float A::add(int, float) float A::add(float, int) _ZN1A3addEfi _ZN 1 A 3 add E fi double A::add(float, float) _ZN1A3addEff



Class name

Function Prototype	Encoding	Deconstruction
<pre>int A::add(int, int)</pre>	_ZN1A3addEii	_ZN 1 A 3 add E ii
<pre>float A::add(int, float)</pre>	_ZN1A3addEif	_ZN 1 A 3 add E if
<pre>float A::add(float, int)</pre>	_ZN1A3addEfi	_ZN 1 A 3 add E fi
<pre>double A::add(float, float)</pre>	_ZN1A3addEff	_ZN 1 A 3 add E ff
No. of characters in the function name		

Function Prototype	Encoding	Deconstruction
<pre>int A::add(int, int)</pre>	_ZN1A3addEii	_ZN 1 A 3 add E ii
<pre>float A::add(int, float)</pre>	_ZN1A3addEif	_ZN 1 A 3 add E if
<pre>float A::add(float, int)</pre>	_ZN1A3addEfi	_ZN 1 A 3 add E fi
<pre>double A::add(float, float)</pre>	_ZN1A3addEff	_ZN 1 A 3 add E ff

Function name

End of function name

Eunction Prototypo

36/70

Storing Function Prototypes

Encoding

Function Prototype	Encoding	Deconstruction
<pre>int A::add(int, int)</pre>	_ZN1A3addEii	_ZN 1 A 3 add E ii
<pre>float A::add(int, float)</pre>	_ZN1A3addEif	_ZN 1 A 3 add E if
<pre>float A::add(float, int)</pre>	_ZN1A3addEfi	_ZN 1 A 3 add E fi
<pre>double A::add(float, float)</pre>	_ZN1A3addEff	_ZN 1 A 3 add E ff

Encoding	Deconstruction
_ZN1A3addEii	_ZN 1 A 3 add E ii
_ZN1A3addEif	_ZN 1 A 3 add E if
_ZN1A3addEfi	_ZN 1 A 3 add E fi
_ZN1A3addEff	_ZN 1 A 3 add E ff
	_ZN1A3addEii _ZN1A3addEif _ZN1A3addEfi

Representing Classes with Templates

```
template <class T>
T GetMax (T a, T b)
{
   T result;
   result = (a>b)?a:b;
   return result;
}
```

29 Jun 2013

```
Definition
```

Definition

Graybox Probing: Understanding C++ Translation

```
T GetMax (T a, T b)
  T result;
  result = (a>b)?a:b;
  return result;
```

template <class T>

29 Jun 2013

```
n = GetMax<long int> (1, m);
```

```
T GetMax (T a, T b)
{
   T result;
   result = (a>b)?a:b;
   return result;
}
```

template <class T>

29 Jun 2013

```
Definition
```

k = GetMax<int> (i, j);

n = GetMax<long int> (1, m);

37/70

Instantiation

Internal representation for each instantiation

```
;; Function T GetMax(T,T) [with T = long int] (_Z6GetMaxIlET_S0_S0_)
T GetMax(T, T) [with T = long int] (long int a, long int b)
```

```
;; Function T GetMax(T,T) [with T = int] (_Z6GetMaxIiET_S0_S0_)
T GetMax(T, T) [with T = int] (int a, int b)
```

```
T GetMax (T a, T b)
{
    T result;
    result = (a>b)?a:b;
    return result;
}
```

29 Jun 2013

n = GetMax<long int> (1, m);

37/70

Instantiation

internal representation for each instantiation
;; Function T GetMax(T,T) [with T = long int] (_Z6GetMaxI1ET_S0_S0_)

```
T GetMax(T, T) [with T = long int] (long int a, long int b)
;; Function T GetMax(T,T) [with T = int] (_Z6GetMaxIiET_S0_S0_)
T GetMax(T, T) [with T = int] (int a, int b)
```

Essential Abstractions in GCC GCC Resource Center, IIT Bombay

29 Jun 2013

```
{ public:
    int w1;
};
class Blue : public White
{ public:
     int b1;
     int b2;
};
class Pink : public Blue
{ public:
     int p1;
     int p2;
};
White w;
Blue b;
Pink p;
```

```
w1
w1
b1
b2
 w1
 b1
 b2
 p1
 p2
```

Representing Inheritance of Functions

Graybox Probing: Understanding C++ Translation

- Non-virtual functions are inherited much like data members
- Type of the object pointer does not matter
- Virtual functions create interesting possibilities based on the object to which a pointer points to
- A pointer to a base class object may point to an object of any derived class in the class hierarchy

39/70

29 Jun 2013

public:

public:

class C : public B

class A

29 Jun 2013

};

};

};

```
virtual void g()
```

class B : public A virtual void g()

void f()

void f()

29 Jun 2013

{ public:

{ public:

};

};

class B : public A

virtual void g() {cout << "\tB:g" << endl;}
 void f() {cout << "\tB:f" << endl;}</pre>

class C : public B

void f() {cout<< "\tC:f" << endl;}</pre>

Essential Abstractions in GCC GCC Resource Center, IIT Bomba

{ public:

29 Jun 2013

};

};

};

class B : public A { public: virtual void g() {cout << "\tB:g" << endl;}</pre> void f() {cout << "\tB:f" << endl;}</pre>

virtual void f() {cout << "\tA:f" << endl;}</pre>

virtual void g() {cout << "\tA:g" << endl;}</pre>

virtual void f(string i) {cout << "\tA:f." << i << endl;}</pre>

Essential Abstractions in GCC

{ public:

29 Jun 2013

Program	Output	
A a, *array[3]; B b; C c;		
<i>C C</i> ,	0	A:f
array[0]=&a array[1]=&b		A:f.x A:g
array[2]=&c	1	B:f A:f.x
for (int i=0;i<3;i++)		B:g
{	2	C:f
cout << i ;		A:f.x
<pre>array[i]->f(); array[i]->f("x"); array[i]->g();</pre>		B:g
}		

29 Jun 2013

42/70

- Partially static and partially dynamic activity
- At compile time, a compiler creates a virtual function table for each class

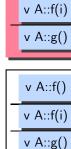
Graybox Probing: Understanding C++ Translation

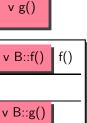
- At run time, a pointer may point to an object of any derived class
- Compiler generates code the pick up the appropriate function by indexing into the virtual table to each class

(the exact virtual table depends on the pointee object)

v A::f()

29 Jun 2013





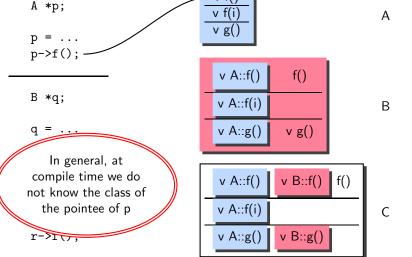
В

f()

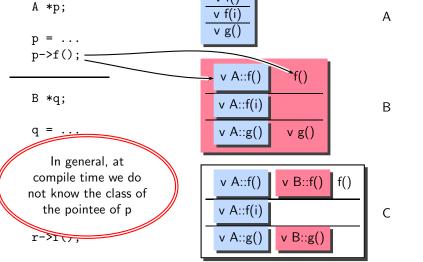
29 Jun 2013

≻v f()

Graybox Probing: Understanding C++ Translation



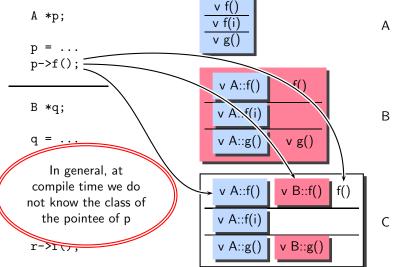
Virtual Function Resolution Requires Dynamic Information



29 Jun 2013

29 Jun 2013

Graybox Probing: Understanding C++ Translation



Α

43/70

p->f(); v A::f() f() B *q; v A::f(i) v A::g() v g() q->f(); v B::f() v A::f() C *r; v A::f(i) $r = \dots$ v B::g() r->f(); v A::g()

Graybox Probing: Understanding C++ Translation

Virtual Function Resolution Requires Dynamic Information

29 Jun 2013

Α

В

43/70

29 Jun 2013

A *p;

B *q;

q->f();

C *r;

 $r = \dots$ r->f(); v g() v A::f() f() Compiler can rule this out statically by looking up the declaration of q

Graybox Probing: Understanding C++ Translation

v A::f(i)

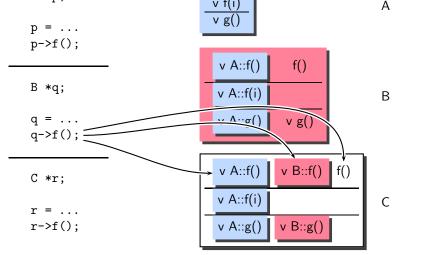
v A::g()

v B::g()

Virtual Function Resolution Requires Dynamic Information

29 Jun 2013

Virtual Function Resolution Requires Dynamic Information



29 Jun 2013

Virtual Function Resolution Requires Dynamic Information

≻v f()

p->f(); v A::f() f() B *q; v A::f(i) В v A::g() v g() q->f(); v B::f() v A::f() C *r; v A::f(i) r->f(); v A::g() v B::g()

43/70

29 Jun 2013

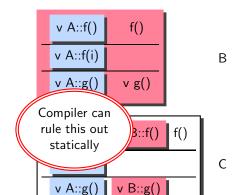
Α

43/70

A *p;

29 Jun 2013

p->f(); B *q; q->f();



Graybox Probing: Understanding C++ Translation

≻v f()

v g()

C *r;

r->f();

Virtual Function Resolution Requires Dynamic Information

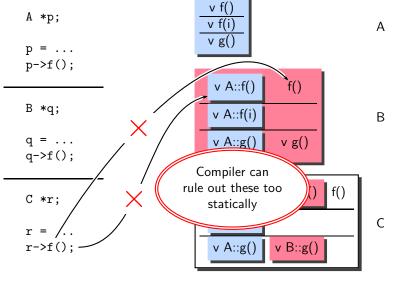
v f() A *p; v f(i) p->f(); f() v A::f() B *q; v A::f(i) В v A::g() v g() q->f(); v B::f() v A::f() C *r; v A::f(i) v A::g() v B::g() r->f();

43/70

29 Jun 2013

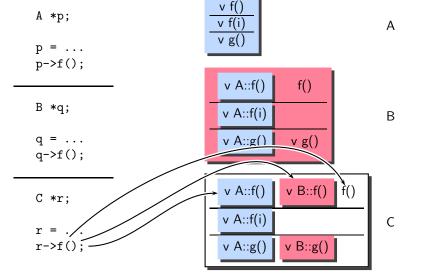
29 Jun 2013

Essential Abstractions in GCC



Graybox Probing: Understanding C++ Translation

Graybox Probing: Understanding C++ Translation



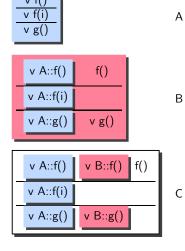
29 Jun 2013

Non-Virtual Functions Do Not Require Dynamic Information

A *p; p = ... p->f(); B *q; q = ... q->f();

29 Jun 2013

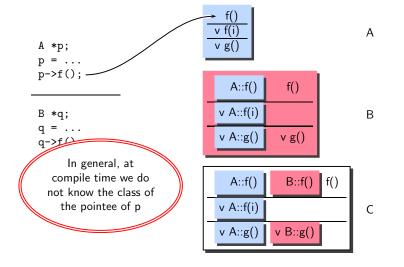
C *r; r = ... r->f();



Non-virtual function = a function which is not virtual in *any* class in a hierarchy

Graybox Probing: Understanding C++ Translation

Two witted function — a function which is not virtual in any class in a fine areny

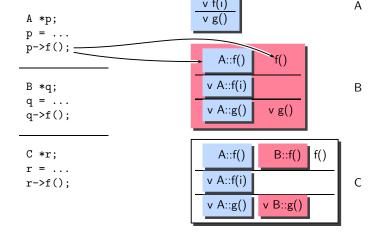


29 Jun 2013

Non-Virtual Functions Do Not Require Dynamic Information

Non-virtual function = a function which is not virtual in any class in a hierarchy

Graybox Probing: Understanding C++ Translation

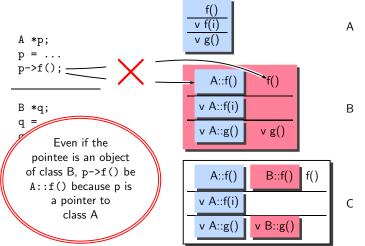


29 Jun 2013

29 Jun 2013

Non-virtual function = a function which is not virtual in any class in a hierarchy

Graybox Probing: Understanding C++ Translation

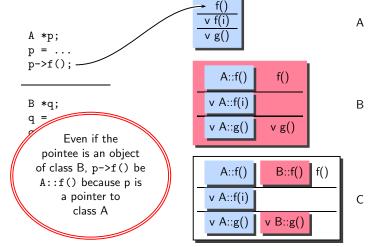


44/70

29 Jun 2013

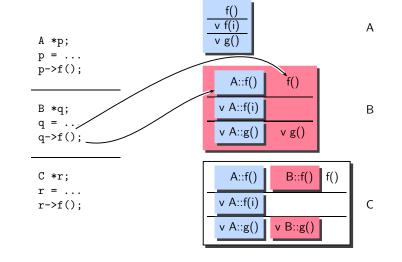
Non-virtual function = a function which is not virtual in any class in a hierarchy

Graybox Probing: Understanding C++ Translation



Non-virtual function = a function which is not virtual in any class in a hierarchy

Graybox Probing: Understanding C++ Translation



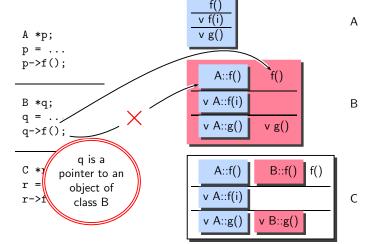
44/70

44/70

29 Jun 2013

Non-virtual function = a function which is not virtual in any class in a hierarchy

Graybox Probing: Understanding C++ Translation



Program	With virtual functions	No virtual function

class A { public:

};

};

virtual void f() virtual void f(string i)

virtual void g()

class B : public A { public:

void f()

virtual void g() };

class C : public B { public:

void f()

2

A:f.x B:g

A:f

A:g

B:f

B:g

C:f

A:f.x

A:f.x

A:g A:f A:f.x

A:g

A:f

A:g

A:f

A:f.x

A:f.x

Program

void f()

No virtual

45/70

Examining the Behaviour of Virtual Functions

With virtual

Program	functions	function
<pre>class A { public: virtual void f() virtual void f(string i) virtual void g() }; class B : public A { public: virtual void g() void f() };</pre>	0 A:f A:f.x A:g 1 B:f A:f.x B:g 2 C:f A:f.x	0 A:f A:f.x A:g 1 A:f.x A:g 2 A:f.x
class C : public B	B:g	A:g

{ public:

};

Graybox Probing: Understanding C++ Translation

- Resolution of virtual functions depends on the class of the pointee object ⇒ Needs dynamic information
- Resolution of non-virtual functions depends on the class of the pointer ⇒ Compile time information is sufficient
- In either case, a pointee cannot belong to a "higher" class in the hierarchy ("higher" class = a class from which the class of the pointer is derived)

46/70

```
virtual void f()
   virtual void f(string i)
   virtual void g()
};
class B : public A
{ public:
   virtual void g()
           void f()
};
```

void f()

{ public:

};

class C : public B

47/70

Graybox Probing: Understanding C++ Translation

Α

class A

29 Jun 2013

}; class B : public A { public:

virtual void g() void f() };

class C : public B { public: void f()

};

class A

};

};

47/70

Constructing Virtual Function Tubic (1)

Α

B

void f()

```
v f()
v g()
v A::f() f()
v A::f(i)
v A::g() v g()
```

{ public:

class C : public B

Constructing Virtual Function Table (1)

```
{ public:
   virtual void f()
   virtual void f(string i)
   virtual void g()
};
class B : public A
{ public:
   virtual void g()
           void f()
};
class C : public B
{ public:
           void f()
};
```

```
v f()
Α
       v g()
                       f()
        v A::f()
        v A::f(i)
B
        v A::g()
                      v g()
                     v B::f()
        v A::f()
        v A::f(i)
        v A::g()
                     v B::g()
```

47/70

class A

A::f()

v f(i)

v g()

v A::f()

v A::f(i)

v A::g()

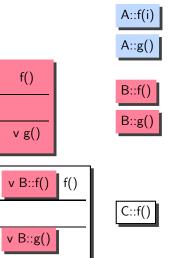
v A::f()

v A::f(i)

Α

В

C



29 Jun 2013

A::f()

v B::g()

v A::g()

29 Jun 2013

Graybox Probing: Understanding C++ Translation

29 Jun 2013

A::f()

B::f()

B::g()

C::f()

29 Jun 2013

В

C

v A::f(i)

v A::g()

v A::f()

v A::f(i)

v g()

v B::f()

v B::g()

A::f(i) is

inherited

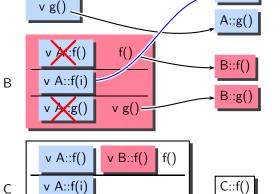
v g() A::g() B::f() v A::f(i) В B::g() v A::g() v g() v B::f() v A::f() v A::f(i) C::f() C

v B::g()

GCC Resource Center, IIT Bombay

48/70

v A::g()

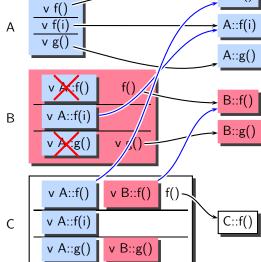


v B::g()



48/70

v A::g()



48/70

A::f()

B::g()

C::f()

Essential Abstractions in GCC

v A::f(i)

v A::f(i)

v A::g()

v g()

v B::g()

29 Jun 2013

В

C

GCC Resource Center, IIT Bombay

48/70

Both A::f() and

B::f() are overridden

by C::f()

B::f()

B::g()

C::f(

Essential Abstractions in GCC

v A::f(i)

v A::f(i)

v A::g()

v g()

v B::g()

f()

29 Jun 2013

В

C

GCC Resource Center, IIT Bombay

A::f(i) is

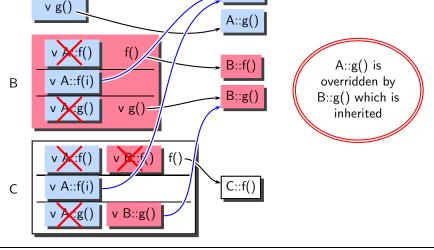
inherited

A::g()

v B::g()

48/70

v A::g()



29 Jun 2013

C::f(

A::f()

A::f(i)

Graybox Probing: Understanding C++ Translation

Constructing Virtual Function Table (2)

v A::f(i)

v B::g()

29 Jun 2013

Α

C

v f()

v g()

C::_ZTV1C: 5u entries v A::f(i) C::f() 0 (int (*)(...))0C 8 (int (*)(...))(& _ZTI1C) v B::g() 16 C::f 24 A::f 32 B::g GCC Resource Center, IIT Bombay

Essential Abstractions in GCC

Graybox Probing: Understanding C++ Translation

49/70

A:: ZTV1A: 5u entries Class A	
0 (int (*)())0 size=8 align=8	
8 (int (*)())(& _ZTI1A) base size=8 base align=8	
16 A::f A (0x7fa1d08bfcc0) 0 nearly-e	mptv
24 A::f vptr=((& A::_ZTV1A) + 16u	

Vtable for B
B:: ZTV1B: 5u entries
0 (int (*)(...))0
8 (int (*)(...))(h ZTI1B)

Size=8 align=8
base size=8 base align=8

8 (int (*)(...))(& ZTI1B)
16 B::f
24 A::f
32 B::g

| B (0x7fa1d07082d8) 0 nearly-empty
| vptr=((& B::ZTV1B) + 16u)
| A (0x7fa1d08bfd20) 0 nearly-empty
| primary-for B (0x7fa1d07082d8)

29 Jun 2013

Vtable for A

Dump file: test.cpp.002t.class

Vtable for A A:: ZTV1A: 5u entries Class A (int (*)(...))0size=8 align=8 8 (int (*)(...))(& ZTI1A) base size=8 base align=8 16 A::f A (0x7fa1d08bfcc0) 0 nearly-empty 24 A::f >> vptr=((& A::_ZTV1A) + 16u) 32 A::g

Vtable for B B::_ZTV1B: 5u entries

(int (*)(...))0

Virtual Table

(int (*)(...))(& _ZTI1B) B::f 24 A::f

Class B size=8 align=8

Class Information

base size=8 base align=8 B (0x7fa1d07082d8) 0 nearly-empty

vptr=((& B::_ZTV1B) + 16u) A (0x7fa1d08bfd20) 0 nearly-empty primary-for B (0x7fa1d07082d8)

49/70

B::g

8

16

32

Graybox Probing: Understanding C++ Translation

Dump file: test.cpp.002t.class

Vtable for A A:: ZTV1A: 5u entries Class A (int (*)(...))0size=8 align=8 8 (int (*)(...))(& ZTI1A) base size=8 base align=8 16 A::f A (0x7fa1d08bfcc0) 0 nearly-empty 24 A::f >>> vptr=((& A::_ZTV1A) + 16u) 32 A::g

Vtable for B

Virtual Table

B::_ZTV1B: 5u entries (int (*)(...))(int (*)(...))(& ZTI1B)

24 A::f 32 B::g

29 Jun 2013

8

16 B::f

Class B size=8 align=8 base size=8 base align=8

Class Information

B (0x7fa1d07082d8) 0 nearly-empty > vptr=((& B::_ZTV1B) + 16u)

A (0x7fa1d08bfd20) 0 nearly-empty primary-for B (0x7fa1d07082d8)

Dump file: test.cpp.002t.class

With wirtual gauglifier

	THE TELEVISION OF THE TELEVISION
Class A size=8 align=8 base size=8 base align=8 A (0x7fa1d08bfcc0) 0 nearly-empty	Class A size=1 align=1 base size=0 base align=1 A (0x7fa1d08bfde0) 0 empty
A (0x7fa1d08bfcc0) 0 nearly-empty	A (0x7fa1d08bfde0) 0 empty

Class B size=8 align=8 base size=8 base align=8 B (0x7fa1d07082d8) 0 nearly-empty

vptr=((& B::_ZTV1B) + 16u)

primary-for B (0x7fa1d07082d8)

vptr=((& A::_ZTV1A) + 16u)

A (0x7fa1d08bfd20) 0 nearly-empty

Class B

B (0x7fa1d07084e0) 0 empty A (0x7fa1d08bfe40) 0 empty

base size=1 base align=1

size=1 align=1

Without wirtual gauglifier

50/70

Essential Abstractions in GCC GCC Resource Center, IIT Bomba

Dump file: test.cpp.002t.class

With virtual gaualifier

A e=1 align=1 e size=0 base align=1 fa1d08bfde0) 0 empty

```
Class B
   size=8 align=8
   base size=8 base align=8
B (0x7fa1d07082d8) 0 nearly-empty
```

vptr=((& A::_ZTV1A) + 16u)

```
vptr=((& B::_ZTV1B) + 16u)
A (0x7fa1d08bfd20) 0 nearly-empty
```

base size=1 base align=1 B (0x7fa1d07084e0) 0 empty A (0x7fa1d08bfe40) 0 empty primary-for B (0x7fa1d07082d8)

size=1 align=1

Class B

Without virtual gaualifier

Class information with and without virtual runctions

Dump file: test.cpp.002t.class

Class A size=8 align=8 base size=8 base align=8 A (0x7fa1d08bfcc0) 0 nearly-empty	Class A size=1 align=1 base size=0 base align=1

Class B
size=8 align=8
base size=8 base align=8

vptr=((& A::_ZTV1A) + 16u)

With virtual gaualifier

base size=8 base align=8
B (0x7fa1d07082d8) 0 nearly-empty
 vptr=((& B::_ZTV1B) + 16u)
A (0x7fa1d08bfd20) 0 nearly-empty

primary-for B (0x7fa1d07082d8)

Class B size=1 align=1

Without virtual gaualifier

A (0x7fa1d08bfde0) 0 empty

base size=1 base align=1
B (0x7fa1d07084e0) 0 empty
A (0x7fa1d08bfe40) 0 empty

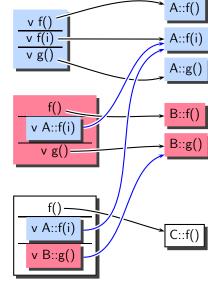
51/70

29 Jun 2013

q = ... q->f();

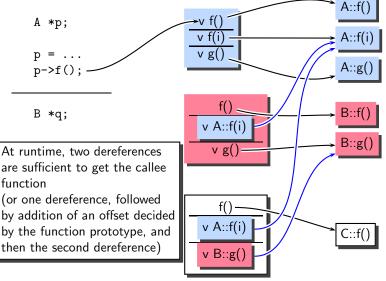
C *r; r = ...

r = ... r->f();



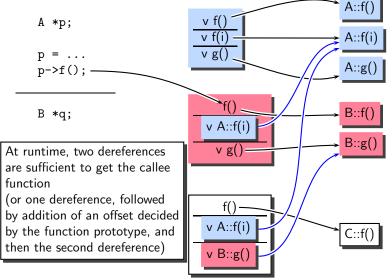
Graybox Probing: Understanding C++ Translation

29 Jun 2013

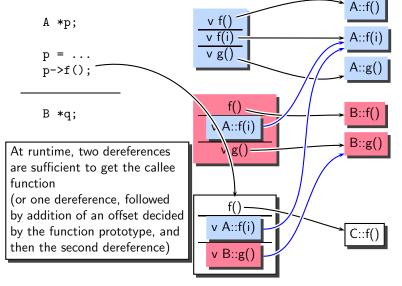


29 Jun 2013

51/70



29 Jun 2013

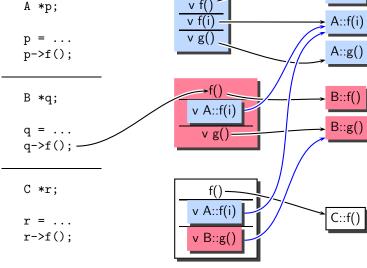


Graybox Probing: Understanding C++ Translation

51/70

A::f()

29 Jun 2013



v f(i v g(

Graybox Probing: Understanding C++ Translation

B *q;

C *r;

r->f();

p->f();

A *p;

29 Jun 2013

- q->f();

v A::f(i)

v B::g()

- v A::f(i) v g()
- B::f()

A::f()

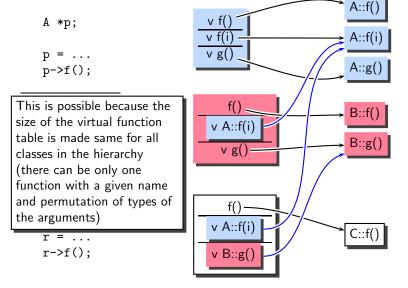
A::f(i)

A::g()

B::g()

51/70

Essential Abstractions in GCC



A Summary of Virtual Function Implementation

Compile time activity

29 Jun 2013

- ► Collect all virtual functions across a class hierarchy
- Ignore non-virtual functions
- Analyse the class hierarchy to locate the appropriate function with a given permutation of argument types

52/70

A Summary of Virtual Function implementation

- Compile time activity
 - ► Collect all virtual functions across a class hierarchy
 - ► Ignore non-virtual functions
 - ► Analyse the class hierarchy to locate the appropriate function with a given permutation of argument types
 - Execution time activity
 - ▶ Dereference object pointer to access the virtual function table
 - Add offset to the base of the table to access the function pointer
 - Dereference the function pointer to make the call

- Compile time activity
 - ► Collect all virtual functions across a class hierarchy
 - Ignore non-virtual functions
 - Analyse the class hierarchy to locate the appropriate function with a given permutation of argument types
- Execution time activity
 - Dereference object pointer to access the virtual function table
 - ▶ Add offset to the base of the table to access the function pointer
 - Dereference the function pointer to make the call
- Study time activity
 - Study the virtual table in the .class dump
 - Observe the code for handling virtual calls in .cfg and assembly dumps
 - ▶ Use the options -fdump-tree-all -S -fverbose-asm

Part 7

Examining GIMPLE Optimization

29 Jun 2013

53/70

```
int main()
{ int a, b, c, n;
 a = 1:
  b = 2;
  c = 3;
 n = c*2;
  while (a \le n)
  {
    a = a+1;
  if (a < 12)
    a = a+b+c;
  return a;
```

What does this program return?

GCC Resource Center, IIT Bombay

int main()

29 Jun 2013

53/70

```
{ int a, b, c, n;
 a = 1:
  b = 2;
  c = 3;
 n = c*2;
  while (a \le n)
  {
    a = a+1;
  if (a < 12)
    a = a+b+c;
  return a;
```

- What does this program return?12
 - 14

Graybox Probing: Examining GIMPLE Optimization

Example Program for Observing Optimizations

```
int main()
{ int a, b, c, n;
  a = 1:
  b = 2;
  c = 3:
  n = c*2:
  while (a \le n)
  ₹
    a = a+1;
  if (a < 12)
    a = a+b+c;
  return a;
```

- What does this program return?
- 12
- We use this program to illustrate various shades of the following optimizations: Constant propagation, Copy

propagation, Loop unrolling, Dead code elimination

29 Jun 2013

Compilation Command

Graybox Probing: Examining GIMPLE Optimization

54/70

\$gcc -fdump-tree-all -02 ccp.c

Essential Abstractions in GCC

GCC GCC Resource Center, IIT Bombay

{ int a, b, c, n;

a = 1: b = 2;

c = 3;n = c*2;

int main()

29 Jun 2013

while $(a \le n)$

if (a < 12)

a = a+1;

a = a+b+c;return a;

Control flow graph



int main()

a = 1; b = 2; c = 3;

29 Jun 2013

55/70

{
 a = a+1;
}
if (a < 12)

Program ccp.c

{ int a, b, c, n;

n = c*2;
while (a <= n)
{
 a = a+1:</pre>

 $B2 \begin{vmatrix}
a = 1 \\
b = 2 \\
c = 3 \\
n = c * 2
\end{vmatrix}$

Control flow graph

 $a = a + \overline{1} B3$

29 Jun 2013

55/70

Example Program 1 Control flow graph

```
int main()
{ int a, b, c, n;
  a = 1:
  b = 2;
  c = 3;
  n = c*2;
  while (a \le n)
    a = a+1;
  if (a < 12)
    a = a+b+c;
  return a;
```

Program ccp.c

```
B2 \begin{bmatrix} a = 1 \\ b = 2 \\ c = 3 \\ n = c * 2 \end{bmatrix}
B4 \begin{bmatrix} if a \le n \end{bmatrix}
```

29 Jun 2013

55/70

Example Program 1

```
Program ccp.c
                                      Control flow graph
int main()
{ int a, b, c, n;
  a = 1:
  b = 2;
                                       B4 | if a \le n |
  c = 3;
  n = c*2;
  while (a \le n)
                               B5 | if a \leq 11
                                                 a = a + 1
    a = a+1;
                                           D.1200 = a + b
                                           a = D.1200 + c
  if (a < 12)
    a = a+b+c;
  return a;
```

B6

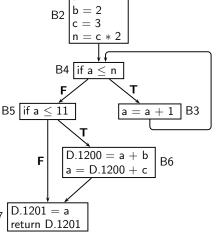
29 Jun 2013

55/70

```
Program ccp.c
                                        Control flow graph
int main()
{ int a, b, c, n;
  a = 1:
  b = 2;
                                          B4 | if a \le n |
  c = 3;
  n = c*2;
  while (a \le n)
                                 B5 | if a \leq 11
                                                     a = a + 1
    a = a+1;
                                             D.1200 = a + b
a = D.1200 + c
                                                                B6
  if (a < 12)
    a = a+b+c;
  return a;
                                   D.1201 = a
```

Dump file ccp.c.014t.cfg

56/70



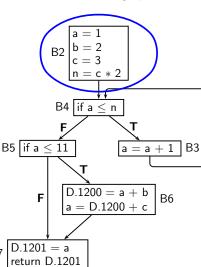
Control flow graph

<bb 2>:

a = 1;

29 Jun 2013

Control flow graph



Dump file ccp.c.014t.cfg

b = 2;c = 3;n = c * 2;

goto <bb 4>;

Control flow graph

 $B2\begin{bmatrix} a = 1 \\ b = 2 \\ c = 3 \\ n = c * 2 \end{bmatrix}$

 $\begin{array}{c|c}
B2 & c = 3 \\
n = c * 2
\end{array}$ $B4 & \text{if } a \leq n$ $B5 & \text{if } a \leq 11$ B3 & a = a + 1 & B3

D.1200 = a + b

a = D.1200 + c

B6

Dump file ccp.c.014t.cfg

<bb 4>:
if (a <= n)
 goto <bb 3>;
else

goto <bb 5>;

<bb 3>:
a = a + 1;

 $\begin{array}{l} D.1201 = a \\ return \ D.1201 \end{array}$

D.1201 = areturn D.1201

B4 \lceil if a \leq n B5 if a ≤ 11 $\mathsf{a}=\mathsf{a}+\mathsf{1}^{\mathsf{T}}$ B3 D.1200 = a + ba = D.1200 + c

Control flow graph

Dump file ccp.c.014t.cfg

56/70

goto <bb 7>; <bb 6>: D.1200 = a + b;a = D.1200 + c;

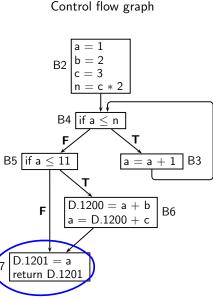
goto <bb 6>;

<bb 5>: if (a <= 11)

else

Essential Abstractions in GCC GCC Resource Center, IIT Bombay

ntrol flow graph Dump file ccp.c.014t.cfg



56/70

>bb 7>:

D.1201 = a;

return D.1201;

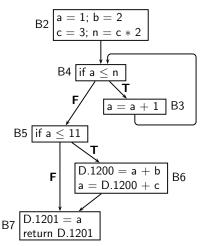
1201;

Essential Abstractions in GCC

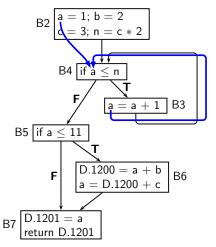
GCC Resource Center, IIT Bombay

Single Static Assignment (SSA) Form

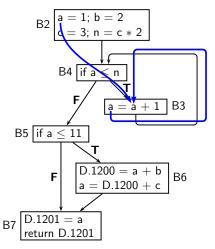
Control flow graph



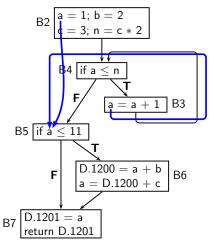
Control flow graph



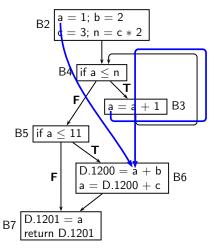
Control flow graph



Control flow graph

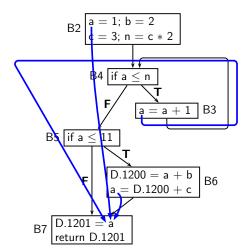


Control flow graph



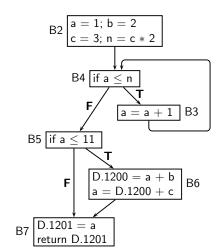
Control flow graph

SSA Form



57/70

Control flow graph



SSA Form

B2 $\begin{vmatrix} a.3 = 1; b.4 = 2 \\ c.5 = 3; n.6 = c.5 * 2 \end{vmatrix}$ B4 $\begin{vmatrix} a.1 = \phi & (a.3, a.7) \\ if a.1 \le n.6 \end{vmatrix}$ F B3 $\begin{vmatrix} a.7 = a.1 + 1 \end{vmatrix}$

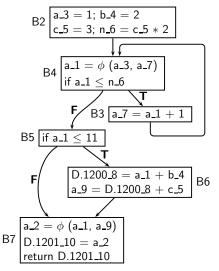
 $D.1200_8 = a_1 + b_4$ $a_9 = D.1200_8 + c_5$

 $a_2 = \phi (a_1, a_9)$ $D.1201_10 = a_2$

B5 if $a_1 \le 11$

D.1201_10 = a_2 return D.1201_10

B6



- A ϕ function is a multiplexer or a selection function
- corresponds to a unique definition of the variable

Every use of a variable

 For every use, the definition is guaranteed to appear on every path leading to the use

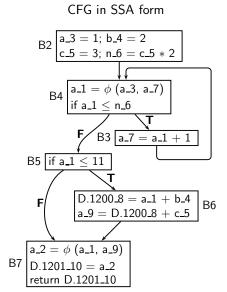
SSA construction algorithm is expected to insert as few ϕ functions as possible to ensure the above properties

58/70

ew

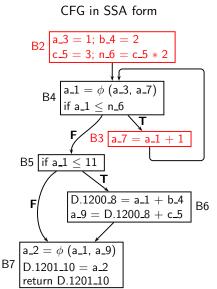
59/70

SSA Form: Pictorial and Textual View



Dump file ccp.c.018t.ssa

SSA Form: Pictorial and Textual View



Dump file ccp.c.018t.ssa

 $c_5 = 3$: $n_6 = c_5 * 2;$ goto <bb 4>;

<bb 2>:

 $a_3 = 1;$

 $b_4 = 2;$

<bb 3>:

 $a_7 = a_1 + 1;$

55A Form: Pictorial and Textual View

B2 $a_3 = 1$; $b_4 = 2$ $c_5 = 3$; $n_6 = c_5 * 2$ B3 $a_7 = a_1 + 1$ B5 if a_1 < 11 $\overline{\text{D.1200}}$.8 = a_1 + b_4 F $a_9 = D.1200_8 + c_5$ $a_2 = \phi (a_1, a_9)$ B7 D 1201 10 = a 2return D.1201_10

CFG in SSA form

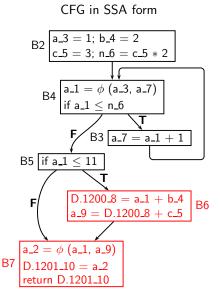
Dump file ccp.c.018t.ssa bb 4>:

 $\# a_1 = PHI < a_3(2), a_7(3) >$

if (a_1 <= n_6)
 goto <bb 3>;
else
 goto <bb 5>;

<bb 5>:
 if (a_1 <= 11)
 goto <bb 6>;
 else
 goto <bb 7>;

SSA Form: Pictorial and Textual View



Dump file ccp.c.018t.ssa

<bb 6>:
 D.1200_8 = a_1 + b_4;
 a_9 = D.1200_8 + c_5;

* a_2 = PHI <a_1(5), a_9(6)>

D.1201_10 = a_2;

return D.1201_10;

Essential Abstractions in GCC GCC Resource Center, IIT Bombay

<bb 2>: $a_3 = 1;$

 $b_4 = 2;$

 $c_5 = 3$;

29 Jun 2013

60/70

<bb 2>: a = 1;

b = 2;

c = 3;

n = c * 2;

goto <bb 4>;

<bb 3>:

a = a + 1;

<bb 3>:

goto <bb 4>;

 $a_7 = a_1 + 1$;

 $n_6 = c_5 * 2;$

<bb 4>:

else

Dump file ccp.c.018t.ssa

29 Jun 2013

60/70

goto <bb 3>; else

goto <bb 5>;

<bb 5>: if (a <= 11)

goto <bb 6>; else goto <bb 7>;

<bb 5>:

if $(a_1 <= 11)$ goto <bb 6>;

else goto <bb 7>;

if $(a_1 \le n_6)$

goto <bb 3>;

goto <bb 5>;

 $\# a_1 = PHI < a_3(2), a_7(3) >$

```
Dump file ccp.c.014t.cfg
```

```
fg Dump file ccp.c.018t.ssa
```

```
D.1200 = a + b;

a = D.1200 + c;

<bb 7>:

D.1201 = a;

return D.1201;
```

<bb 6>:

```
<bb 6>:
    D.1200_8 = a_1 + b_4;
    a_9 = D.1200_8 + c_5;

<bb 7>:
    # a_2 = PHI <a_1(5), a_9(6)>
    D.1201_10 = a_2;
    return D.1201_10;
```

29 Jun 2013

61/70

```
Input dump: ccp.c.018t.ssa
Output dump: ccp.c.023t.copyrename1

<bb 7>:
    # a_2 = PHI <a_1(5), a_9(6)>
    D.1201_10 = a_2;
    return D.1201_10;

Output dump: ccp.c.023t.copyrename1

<br/>
    # a_2 = PHI <a_1(5), a_9(6)>
    a_10 = a_2;
    return a_10;
```

<bb 2>:

else

 $a_3 = 1$;

 $b_4 = 2;$

```
\# a_1 = PHI < a_3(2), a_7(3) >
if (a_1 <= n_6)
  goto <bb 3>;
else
  goto <bb 5>;
```

<bb 2>:

<bb 3>:

<bb 4>:

 $a_3 = 1$;

 $b_4 = 2;$

```
c_5 = 3;
n_6 = c_5 * 2;
                                   n_6 = 6;
goto <bb 4>;
                                 <bb 3>:
a_7 = a_1 + 1;
                                 <bb 4>:
```

```
c_5 = 3;
goto <bb 4>;
a_7 = a_1 + 1;
\# a_1 = PHI < 1(2), a_7(3) >
if (a_1 <= 6)
  goto <bb 3>;
  goto <bb 5>;
```

Input dump: ccp.c.023t.copyrename1 Output dump: ccp.c.024t.ccp1

<bb 2>:

```
b_4 = 2;
c_5 = 3;
n_6 = 6;
goto <bb 4>;
```

<bb 6>: $D.1200_8 = a_1 + b_4;$ $a_9 = D.1200_8 + c_5;$

 $a_3 = 1;$ $b_4 = 2;$ $c_5 = 3$;

 $n_6 = 6$; goto <bb 4>;

<bb 6>: $D.1200_8 = a_1 + 2;$

 $a_9 = D.1200_8 + 3;$

62/70

<bb 2>:

 $a_3 = 1$;

```
Input dump: ccp.c.024t.ccp1
```

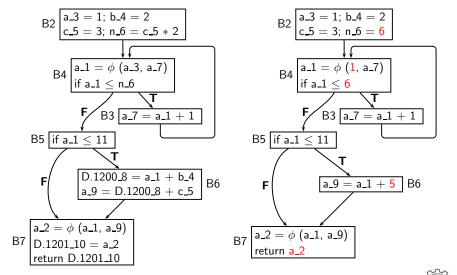
```
<bb 6>:
 D.1200_8 = a_1 + 2;
  a_9 = D.1200_8 + 3;
<bb >7>:
  \# a_2 = PHI < a_1(5), a_9(6) >
  a_10 = a_2;
  return a_10;
```

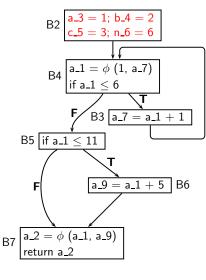
Output dump: ccp.c.031t.copyprop1

return a_2;

63/70

The Result of Copy Propagation and Renaming

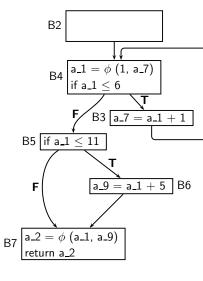




- No uses for variables a_3, b_4, c_5, and n_6
- Assignments to these variables can be deleted

Dead Code Elimination Using Control Dependence Dump file ccp.c.031t.cddce1

Graybox Probing: Examining GIMPLE Optimization



<bb >2>:

<bb 3>: $a_7 = a_1 + 1;$

goto <bb 4>;

<bb 4>:

 $\# a_1 = PHI < 1(2), a_7(3) >$ if (a_1 <= 6) goto <bb 3>;

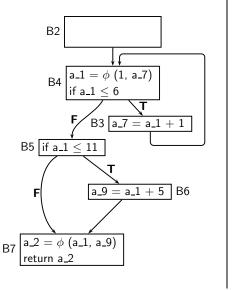
else goto <bb 5>; <bb 5>: if (a_1 <= 11) goto <bb 6>;

else goto <bb 7>; <bb 6>: $a_9 = a_1 + 5$;

 $\# a_2 = PHI < a_1(5), a_9(6) >$

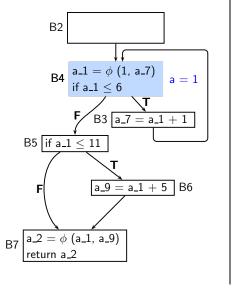
return a_2;

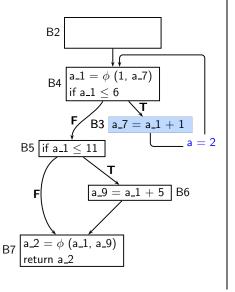
<bb 7>:

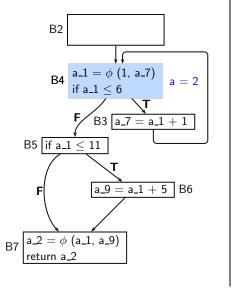


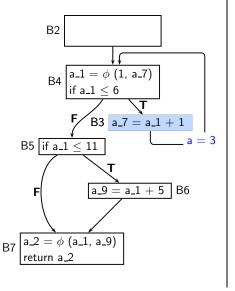
29 Jun 2013

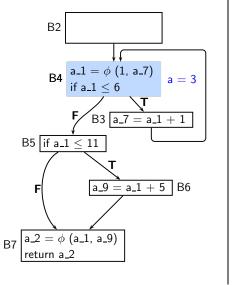
67/70

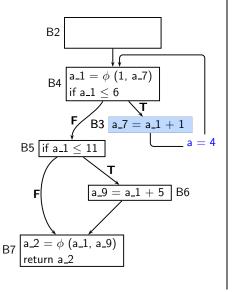


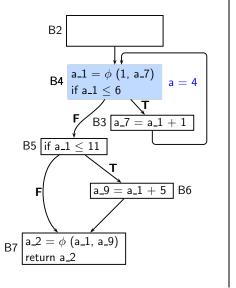


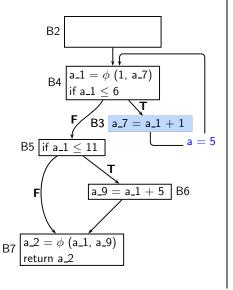


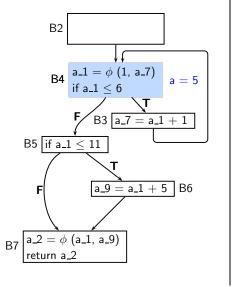


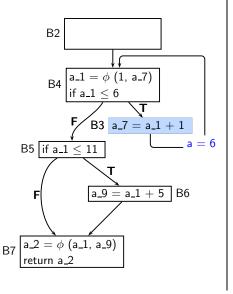


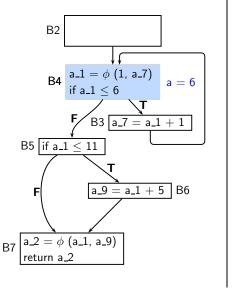


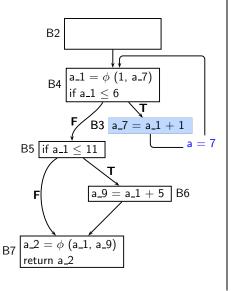




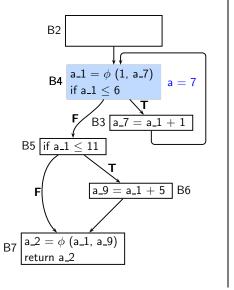




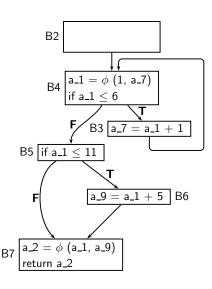


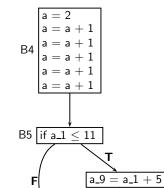






Loop Unrolling





 $a_2 = \phi \ (a_1, a_9)$

return a_2

B₆

67/70

B7 |

Complete Unrolling of Inner Loops

Dump file: ccp.c.059t.cunrolli

```
<bb >2>:
 a_13 = 2;
 a_15 = a_13 + 1;
 a_17 = a_15 + 1;
 a_19 = a_17 + 1;
 a_21 = a_19 + 1;
 a_23 = a_21 + 1;
  if (a_23 <= 11) goto <bb 3>;
 else goto <bb 4>;
<bb 3>:
 a_9 = a_23 + 5;
<bb 4>:
```

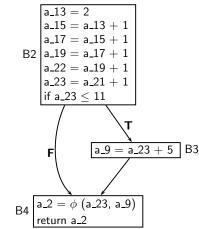
 $\# a_2 = PHI < a_23(2), a_9(3) >$

```
a_13 = 2
   a_15 = a_13 + 1
   a_17 = a_15 + 1
B2 a_19 = a_17 + 1
   a_21 = a_19 + 1
   a_23 = a_21 + 1
   if a_23 < 11
            a_9 = a_2 + 5
                            B3
 a_2 = \phi \ (a_23, a_9)
```

return a_2;

Another Round of Constant Propagation

Input



Dump file: ccp.c.060t.ccp2

main ()
{
 <bb 2>:
 retur

}

return 12;

IIT Bombay

Part 8

Conclusions

• Source code is transformed into assembly by lowering the abstraction level step by step to bring it close to the machine

This transformation can be understood to a large extent by observing

- inputs and output of the different steps in the transformation
- It is easy to prepare interesting test cases and observe the effect of transformations
- One optimization often leads to another Hence GCC performs many optimizations repeatedly (eg. copy propagation, dead code elimination)

70/70