Workshop on Essential Abstractions in GCC

Graybox Probing for Machine Independent **Optimizations**

GCC Resource Center

(www.cse.iitb.ac.in/grc)

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



July 2010

July 2010

Machine Independent Optimizations: Outline

1/30

July 2010

Machine Independent Optimizations: Outline **Outline**

1/30

Outline

• Example 1

- ► Constant Propagation
- Copy Propagation
- ► Dead Code Elimination
- ► Loop unrolling
- Example 2
 - ▶ Partial Redundancy Elimination
 - Copy Propagation
 - ► Dead Code Elimination





Example Program 1

Example Program 1

Part 1

First Example Program

```
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

Essential Abstractions in GCC

Compilation Command

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



4/30

Essential Abstractions in GCC

Essential Abstractions in GCC

July 2010

GCC Resource Center, IIT Bombay

Machine Independent Optimizations: First Example Program

4/30

Example Program 1

Machine Independent Optimizations: First Example Program

```
Program ccp.c
                                      Control flow graph
int main()
{ int a, b, c, n;
  a = 1;
  b = 2;
                                    B4 if a \le n goto B3
  c = 3;
  n = c*2;
  while (a \le n)
                         B5 if a \leq 11 goto B6
                                                    a = a + 1 \mid B3
     a = a+1;
                                          D.1200 = a + b
                                          a = D.1200 + c
  if (a < 12)
    a = a+b+c;
  return a;
                                \mathsf{D.1201} = \mathsf{a}
                                return D.1201
```

July 2010

Essential Abstractions in GCC

Example Program 1

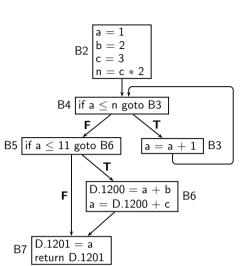
July 2010



GCC Resource Center, IIT Bomb

Control Flow Graph: Pictorial and Textual View

Control flow graph



Dump file ccp.c.012t.cfg

Essential Abstractions in GCC

GCC Resource Center, IIT Bomba



5/30

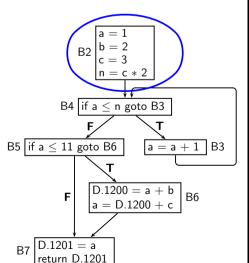
July 2010

Machine Independent Optimizations: First Example Program

5/30

Control Flow Graph: Pictorial and Textual View

Control flow graph



Dump file ccp.c.012t.cfg

```
<bb 2>:
a = 1;
c = 3;
n = c * 2;
goto <bb 4>;
```



Control Flow Graph: Pictorial and Textual View

Essential Abstractions in GCC

GCC Resource Center, IIT Bomb

July 2010

Machine Independent Optimizations: First Example Program

5/30

Control Flow Graph: Pictorial and Textual View



Control Flow Graph: Pictorial and Textual View

Control flow graph

Dump file ccp.c.012t.cfg

```
B2 = 1
b = 2
c = 3
n = c * 2
B4 \text{ if } a \le n \text{ goto } B3
F = \begin{bmatrix}
D.1200 = a + b \\
a = D.1200 + c
\end{bmatrix} B6
```

<bb 3>:
a = a + 1;

the 4>:
if (2 < - n)

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

goto <bb 5>;

Essential Abstractions in GCC

B7 D.1201 = a return D.1201

GCC Resource Center, IIT Bombay



July 2010

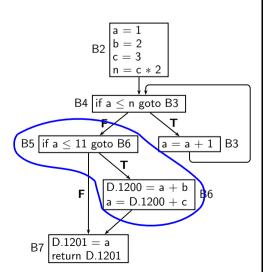
Machine Independent Optimizations: First Example Program

5/30

Control Flow Graph: Pictorial and Textual View

Control flow graph

Dump file ccp.c.012t.cfg



July 2010

Control Flow Graph: Pictorial and Textual View

Notes

Essential Abstractions in GCC

GCC Resource Center, IIT Bombay

ay (T)

July 2010

Machine Independent Optimizations: First Example Program

5/30

Control Flow Graph: Pictorial and Textual View



Control Flow Graph: Pictorial and Textual View

Control flow graph $B2 \begin{bmatrix} a=1\\b=2\\c=3\\n=c*2 \end{bmatrix}$ $B4 \text{ if } a \leq n \text{ goto } B3$ $F \qquad T$ $B5 \text{ if } a \leq 11 \text{ goto } B6 \qquad a=a+1 \quad B3$ $T \qquad D.1200=a+b\\a=D.1200+c \qquad B6$

Dump file ccp.c.012t.cfg

D.1201 = a;

return D.1201;

Essential Abstractions in GCC

GCC Resource Center, IIT Bombay



GCC Resource Center, IIT Bombay



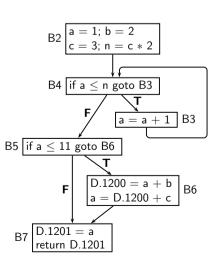
July 2010

Machine Independent Optimizations: First Example Program

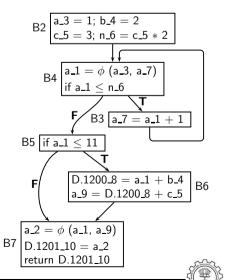
6/30

Single Static Assignment (SSA) Form

Control flow graph



SSA Form



Control Flow Graph: Pictorial and Textual View

6/30

July 2010 Machin

Machine Independent Optimizations: First Example Program

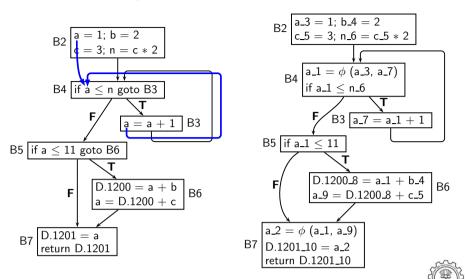
Single Static Assignment (SSA) Form

Single Static Assignment (SSA) Form

Single Static Assignment (SSA) Form

Control flow graph

SSA Form



Essential Abstractions in GCC

GCC Resource Center, IIT Bombay



July 2010



July 2010

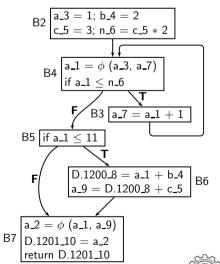
Machine Independent Optimizations: First Example Program

6/30

Single Static Assignment (SSA) Form

Control flow graph

B2 a = 1; b = 2B4 if $a \le n$ goto B3 F a = a + 1 B3 B5 if $a \le 11$ goto B6 T a = a + b B6 a = D.1200 + c B6 SSA Form



ombay

Note

GCC Resource Center, IIT Bombay

C /20

Machine Independent Optimizations: First Example Program

Single Static Assignment (SSA) Form

6/30

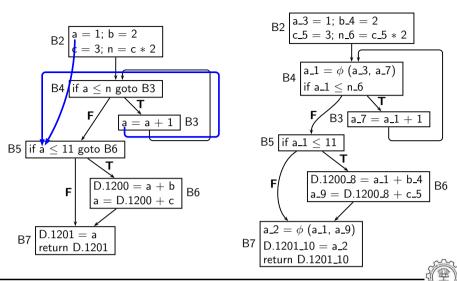
return D.1201

Single Static Assignment (SSA) Form

Single Static Assignment (SSA) Form

Control flow graph

SSA Form



Essential Abstractions in GCC

GCC Resource Center, IIT Bombay



July 2010

GCC Resource Center, IIT Bomba



July 2010

Machine Independent Optimizations: First Example Program

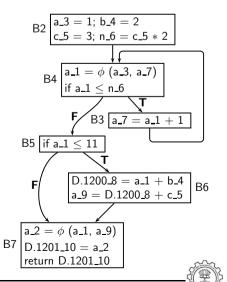
6/30

Single Static Assignment (SSA) Form

Control flow graph

B4 if a < n goto B3 + 1 B3 B5 if a ≤ 11 goto B6 D.1200 = a + ba = D.1200 + cD.1201 = areturn D.1201

SSA Form



6/30

Machine Independent Optimizations: First Example Program Single Static Assignment (SSA) Form

Single Static Assignment (SSA) Form

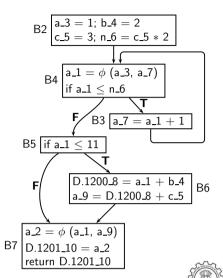
Control flow graph

B2 a = 1; b = 2 c = 3; n = c * 2B4 $a \le n$ goto B3 F a = a + 1B3
B5 $a \le 11$ goto B6

D.1200 = a + b

= D.1200 + c

SSA Form



Essential Abstractions in GCC

D.1201 = a

return D.1201

GCC Resource Center, IIT Bombay

Essential Abstractions in GCC

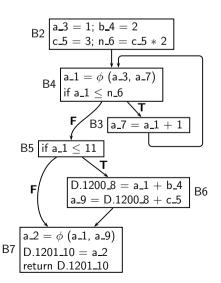
GCC Resource Center, IIT Bombay

July 2010

Machine Independent Optimizations: First Example Program

7/30

Properties of SSA Form



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



Single Static Assignment (SSA) Form

Note

July 2010

7/30

Machine Independent Optimizations: First Example Program

Properties of SSA Form

otes

July 2010

SSA Form: Pictorial and Textual View

CFG in SSA form $B2 \begin{bmatrix} a.3 = 1; \ b.4 = 2 \\ c.5 = 3; \ n.6 = c.5 * 2 \end{bmatrix}$ $B4 \begin{bmatrix} a.1 = \phi \ (a.3, a.7) \\ if \ a.1 \le n.6 \end{bmatrix}$ $B5 \begin{bmatrix} if \ a.1 \le 11 \\ a.9 = D.1200.8 + c.5 \end{bmatrix}$ $B6 \begin{bmatrix} a.2 = \phi \ (a.1, a.9) \\ D.1201.10 = a.2 \end{bmatrix}$

Dump file ccp.c.023t.ssa

-

Essential Abstractions in GCC

return D.1201_10

GCC Resource Center, IIT Bombay



July 2010

Machine Independent Optimizations: First Example Program

8/30

SSA Form: Pictorial and Textual View

CFG in SSA form

B2 a.3 = 1; b.4 = 2 c.5 = 3; n.6 = c.5 * 2B4 $a.1 = \phi$ (a.3, a.7)
if $a.1 \le n.6$ F B3 a.7 = a.1 + 1B5 if $a.1 \le 11$ T

D.1200.8 = a.1 + b.4 a.9 = D.1200.8 + c.5B6

B7 $a.2 = \phi$ (a.1, a.9)
D.1201_10 = a_2
return D.1201_10

Dump file ccp.c.023t.ssa

SSA Form: Pictorial and Textual View

Notes

Essential Abstractions in GCC

GCC Resource Center, IIT Bombay

av (Park

July 2010

Machine Independent Optimizations: First Example Program

8/30

SSA Form: Pictorial and Textual View



SSA Form: Pictorial and Textual View

CFG in SSA form

B2 a.3 = 1; b.4 = 2 c.5 = 3; n.6 = c.5 * 2B4 $a.1 = \phi$ (a.3, a.7)
if $a.1 \le n.6$ F B3 a.7 = a.1 + 1B5 if $a.1 \le 11$ T

D.1200.8 = a.1 + b.4 a.9 = D.1200.8 + c.5B6

B7 $a.2 = \phi$ (a.1, a.9)
D.1201.10 = a.2
return D.1201.10

Dump file ccp.c.023t.ssa

Essential Abstractions in GCC

GCC Resource Center, IIT Bombay



July 2010

Machine Independent Optimizations: First Example Program

8/30

SSA Form: Pictorial and Textual View

CFG in SSA form

B2 a.3 = 1; b.4 = 2 c.5 = 3; n.6 = c.5 * 2B4 $a.1 = \phi$ (a.3, a.7)if $a.1 \le n.6$ F B3 a.7 = a.1 + 1B5 if $a.1 \le 11$ T

D.1200.8 = a.1 + b.4 a.9 = D.1200.8 + c.5B6

B6

B6

D.1201.10 = a.2return D.1201.10

Dump file ccp.c.023t.ssa

July 2010

SSA Form: Pictorial and Textual View

Note

Essential Abstractions in GCC

GCC Resource Center, IIT Bombay

ay A

July 2010 M

Machine Independent Optimizations: First Example Program

8/30

SSA Form: Pictorial and Textual View



A Comparison of CFG and SSA Dumps

Dump file ccp.c.012t.cfg

Dump file ccp.c.023t.ssa

```
<bb 2>:
    a = 1;
    b = 2;
    c = 3;
    n = c * 2;
    goto <bb 4>;

<bb 3>:
    a = a + 1;
```

```
<bb 2>:
    a_3 = 1;
    b_4 = 2;
    c_5 = 3;
    n_6 = c_5 * 2;
    goto <bb 4>;

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



Essential Abstractions in GCC

GCC Resource Center, IIT Bombay

. .

July 2010

Machine Independent Optimizations: First Example Program

9/30

A Comparison of CFG and SSA Dumps

Dump file ccp.c.012t.cfg

<bb 4>:
 if (a <= n)
 goto <bb 3>;
 else
 goto <bb 5>;

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

Dump file ccp.c.023t.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>;
```

A Comparison of CFG and SSA Dumps

Notes

July 2010

Essential Abstractions in GCC

GCC Resource Center, IIT Bombay



July 2010

Machine Independent Optimizations: First Example Program

9/30

A Comparison of CFG and SSA Dumps



A Comparison of CFG and SSA Dumps

Dump file ccp.c.012t.cfg

Dump file ccp.c.023t.ssa

```
<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;
```



Essential Abstractions in GCC

July 2010

Machine Independent Optimizations: First Example Program

July 2010

Machine Independent Optimizations: First Example Program

10/30

Copy Renaming

Input dump: ccp.c.023t.ssa Output dump: ccp.c.026t.copyrename1

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

 $\# a_2 = PHI < a_1(5), a_9(6) >$ $a_10 = a_2;$ return a_10;

A Comparison of CFG and SSA Dumps

July 2010

Essential Abstractions in GCC

10/30

Copy Renaming







GCC Resource Center, IIT

Machine Independent Optimizations: First Example Program First Level Constant and Copy Propagation

```
Input dump: ccp.c.026t.copyrename1
                                      Output dump: ccp.c.027t.cpp1
 <bb 2>:
                                     <bb 2>:
   a_3 = 1;
                                       a_3 = 1;
   b_4 = 2;
                                       b_4 = 2;
   c_5 = 3;
                                       c_5 = 3;
   n_6 = c_5 * 2;
                                       n_6 = 6;
                                       goto <bb 4>;
   goto <bb 4>;
 <bb 3>:
                                     <bb 3>:
   a_7 = a_1 + 1;
                                       a_7 = a_1 + 1;
 <bb 4>:
                                     <bb 4>:
   \# a_1 = PHI < a_3(2), a_7(3) >
                                       \# a_1 = PHI < 1(2), a_7(3) >
   if (a_1 <= n_6)
                                       if (a_1 <= 6)
     goto <bb 3>;
                                         goto <bb 3>;
   else
                                       else
     goto <bb 5>;
                                         goto <bb 5>;
```

Essential Abstractions in GCC



July 2010

Machine Independent Optimizations: First Example Program

11/30

First Level Constant and Copy Propagation

Input dump: ccp.c.026t.copyrename1

Output dump: ccp.c.027t.cpp1

```
<bb 2>:
                                    <bb 2>:
 a_3 = 1:
                                     a_3 = 1:
 b_4 = 2;
                                      b_4 = 2;
 c_5 = 3;
                                      c_5 = 3;
 n_6 = 6;
                                     n_6 = 6;
 goto <bb 4>;
                                     goto <bb 4>;
<bb 6>:
                                   <bb 6>:
 D.1200_8 = a_1 + b_4;
                                     D.1200_8 = a_1 + 2;
 a_9 = D.1200_8 + c_5;
                                     a_9 = D.1200_8 + 3;
```

First Level Constant and Copy Propagation

July 2010

Essential Abstractions in GCC

July 2010

Machine Independent Optimizations: First Example Program

11/30

First Level Constant and Copy Propagation





Second Level Copy Propagation

Second Level Copy Propagation

Input dump: ccp.c.029t.ccp1

Output dump: ccp.c.031t.copyprop1

```
<bb 7>:
  \# a_2 = PHI < a_1(5), a_9(6) >
  a_10 = a_2;
  return a_10;
```

```
<bb 7>:
 \# a_2 = PHI < a_1(5), a_9(6) >
 return a_2;
```



Essential Abstractions in GCC

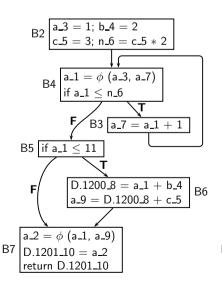
GCC Resource Center, IIT Bomba

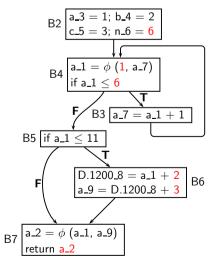
13/30

July 2010

Machine Independent Optimizations: First Example Program

The Result of Copy Propagation and Renaming







Essential Abstractions in GCC

GCC Resource Center, IIT Bomb

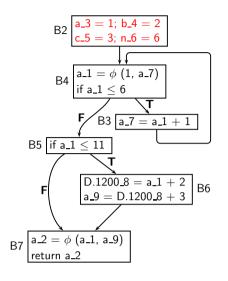
July 2010

Machine Independent Optimizations: First Example Program

13/30

The Result of Copy Propagation and Renaming

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



Essential Abstractions in GCC

GCC Resource Center, IIT Bomba

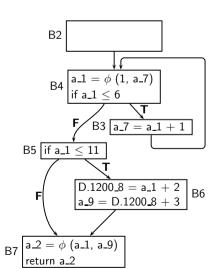


July 2010

Machine Independent Optimizations: First Example Program

15/30

Dead Code Elimination Using Control Dependence



Dump file ccp.c.033t.cddce1

```
<bb 2>:
  goto <bb 4>;
<bb 3>:
  a_7 = a_1 + 1;
<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>:
  D.1200_8 = a_1 + 2;
 a_9 = D.1200_8 + 3;
<bb 7>:
 \# a_2 = PHI < a_1(5), a_9(6) >
  return a_2;
```

The Result of Copy Propagation and Renaming

July 2010

Essential Abstractions in GCC

GCC Resource Center, IIT Bomb



July 2010

Machine Independent Optimizations: First Example Program

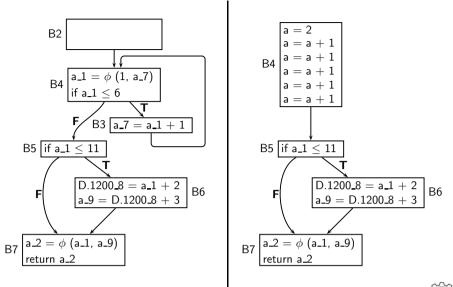
15/30

Dead Code Elimination Using Control Dependence

Essential Abstractions in GCC



Loop Unrolling



Essential Abstractions in GCC

GCC Resource Center, IIT Bombay



July 2010

Machine Independent Optimizations: First Example Program

17/30

Complete Unrolling of Inner Loops

Dump file: ccp.c.058t.cunrolli

```
<bb 2>:
    a_12 = 2;
    a_14 = a_12 + 1;
    a_16 = a_14 + 1;
    a_18 = a_16 + 1;
    a_20 = a_18 + 1;
    a_22 = a_20 + 1;
    if (a_22 <= 11) goto <bb 3>;
    else goto <bb 4>;

<bb 3>:
    D.1959_8 = a_22 + 2;
    a_9 = D.1959_8 + 3;

<bb 4>:
    # a_2 = PHI <a_22(2), a_9(3)> return a_2;
```

```
B2 a_12 = 2

a_14 = a_12 + 1

a_16 = a_14 + 1

a_18 = a_16 + 1

a_20 = a_18 + 1

a_22 = a_20 + 1

if a_22 \le 11

T

D.1200_8 = a_22 + 2

a_29 = D.1200_8 + 3

B4 a_2 = \phi (a_22, a_9)

return a_2
```

bay bay

Loop Unrolling

Notes

July 2010

Essential Abstractions in GCC

GCC Resource Center, IIT Bombay

ay W

July 2010

Machine Independent Optimizations: First Example Program

17/30

Complete Unrolling of Inner Loops



Another Round of Constant Propagation

Input

B2 $\begin{vmatrix} a.12 = 2 \\ a.14 = a.12 + 1 \\ a.16 = a.14 + 1 \\ a.18 = a.16 + 1 \\ a.20 = a.18 + 1 \\ a.22 = a.20 + 1 \\ if a.22 \le 11 \end{vmatrix}$ F $\begin{vmatrix} D.1200.8 = a.22 + 2 \\ a.9 = D.1200.8 + 3 \end{vmatrix}$ B3

Dump file: ccp.c.059t.ccp2

a_22 = 7;

a_9 = 12;

return 12;

Essential Abstractions in GCC

GCC Resource Center, IIT Bombay



July 2010

Machine Independent Optimizations: First Example Program

19/30

Dead Code Elimination Using Copy Propagation

Dump file: ccp.c.059t.ccp2

a_22 = 7; a_9 = 12; return 12; **Dump file:** ccp.c.066t.copyprop2

tob 2>:

return 12;

Another Round of Constant Propagation

Notes

July 2010

Essential Abstractions in GCC

GCC Resource Center, IIT Bombay

bay From the

July 2010

Machine Independent Optimizations: First Example Program

19/30

Dead Code Elimination Using Copy Propagation

Second Example Program

July 2010

Machine Independent Optimizations: Second Example Program

20/30

20/30

Example Program 2

```
{ int a;
   do
      a = b+c;
   while (a <= n);
```

int f(int b, int c, int n)

We use this program to illustrate the following optimizations:

Partial Redundancy Elimination, Copy Propagation, Dead Code Elimination

July 2010



return a;

Compilation Command

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



22/30

Essential Abstractions in GCC

July 2010

GCC Resource Center, IIT Bombay

July 2010

Machine Independent Optimizations: Second Example Program

22/30

Machine Independent Optimizations: Second Example Program Example Program 2

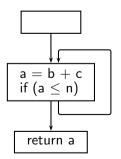
loop.c
int f(int b, int c, int n)
{ int a;

at f(int b, int c, int n)
 int a;

 do
 {
 a = b+c;
 }
 while (a <= n);

return a;</pre>

Control Flow Graph





Compilation Command

Notes

July 2010

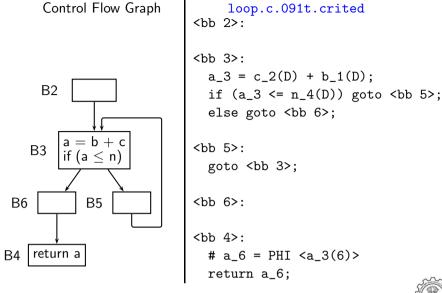
Essential Abstractions in GCC

GCC Resource Center, III Bombay

Example Program 2

Votes

Dump of Input to PRE Pass



Essential Abstractions in GCC

GCC Resource Center, IIT Bombay



July 2010

Machine Independent Optimizations: Second Example Program

24/30

Input and Output of PRE Pass

```
loop.c.091t.crited
                                  loop.c.092t.pre
                                  <bb 2>:
<bb 2>:
                                    pretmp.2_7 = b_1(D) + c_2(D);
                                  <bb 3>:
<bb 3>:
 a_3 = c_2(D) + b_1(D);
                                    a_3 = pretmp.2_7;
  if (a_3 \le n_4(D))
                                    if (a_3 \le n_4(D))
       goto <bb 5>;
                                         goto <bb 5>;
  else goto <bb 6>;
                                    else goto <bb 6>;
<bb 5>:
                                  <bb 5>:
  goto <bb 3>;
                                    goto <bb 3>;
<bb 6>:
                                  <bb 6>:
<bb 4>:
                                  <bb 4>:
  \# a_6 = PHI < a_3(6) >
                                    \# a_6 = PHI < a_3(6) >
  return a_6;
                                    return a_6;
```

Dump of Input to PRE Pass

Notes

July 2010

Essential Abstractions in GCC

GCC Resource Center, IIT Bombay



July 2010 Machine Independent Optimizations: Second Example Program

24/30

Input and Output of PRE Pass



Copy Propagation after PRE

```
loop.c.096t.copyprop4
loop.c.092t.pre
<bb 2>:
                                   <bb 2>:
 pretmp.2_7 = b_1(D) + c_2(D);
                                     pretmp.2_7 = b_1(D) + c_2(D);
<bb 3>:
                                   <bb 3>:
 a_3 = pretmp.2_7;
                                     a_3 = pretmp.2_7;
 if (a_3 \le n_4(D))
                                     if (n_4(D) >= pretmp.2_7)
       goto <bb 5>;
                                       goto <bb 4>;
  else goto <bb 6>;
                                     else
                                       goto <bb 5>;
<bb 5>:
                                   <bb 4>:
  goto <bb 3>;
                                     goto <bb 3>;
<bb 6>:
<bb 4>:
                                   <bb 5>:
  \# a_6 = PHI < a_3(6) >
                                     \# a_8 = PHI < pretmp.2_7(3) >
 return a_6;
                                    return a_8;
```

Essential Abstractions in GCC



July 2010

July 2010

Machine Independent Optimizations: Second Example Program

26/30

Dead Code Elimination

```
loop.c.096t.copyprop4
<bb 2>:
 pretmp.2_7 = b_1(D) + c_2(D);
<bb 3>:
 a_3 = pretmp.2_7;
  if (n_4(D) >= pretmp.2_7)
   goto <bb 4>;
  else
    goto <bb 5>;
<bb 4>:
  goto <bb 3>;
<bb 5>:
  # a_8 = PHI oretmp.2_7(3)>
 return a_8;
```

```
loop.c.097t.dceloop1
<bb 2>:
  pretmp.2_7 = b_1(D) + c_2(D);
<bb 3>:
  if (n_4(D) >= pretmp.2_7)
    goto <bb 4>;
  else
    goto <bb 5>;
<bb 4>:
  goto <bb 3>;
<bb 5>:
  # a_8 = PHI oretmp.2_7(3)>
  return a_8;
```

July 2010

GCC Resource Center, IIT

July 2010

Essential Abstractions in GCC

Machine Independent Optimizations: Second Example Program

26/30

Dead Code Elimination

Essential Abstractions in GCC



Redundant ϕ Function Elimination and Copy Propagation

```
loop.c.124t.phicprop2
loop.c.097t.dceloop1
<bb 2>:
 pretmp.2_7 = b_1(D) + c_2(D);
                                   <bb 2>:
                                    pretmp.2_7 = c_2(D) + b_1(D);
<bb 3>:
                                    if (n_4(D) >= pretmp.2_7)
  if (n_4(D) >= pretmp.2_7)
                                      goto <bb 4>;
    goto <bb 4>;
                                    else
  else
                                      goto <bb 3>;
    goto <bb 5>;
                                   <bb 3>:
<bb 4>:
                                    return pretmp.2_7;
  goto <bb 3>;
                                   <bb 4>:
<bb 5>:
                                    goto <bb 4>;
  \# a_8 = PHI < pretmp.2_7(3) >
 return a_8;
```

Essential Abstractions in GCC

GCC Resource Center, IIT Bombay



July 2010

Machine Independent Optimizations: Second Example Program

28/30

Final Assembly Program

```
loop.c.124t.phicprop2
                                   loop.s
<bb 2>:
                                        pushl %ebp
 pretmp.2_7 = c_2(D) + b_1(D);
                                               %esp, %ebp
 if (n_4(D) >= pretmp.2_7)
                                               12(%ebp), %eax
                                        movl
                                               8(%ebp), %eax
    goto <bb 4>;
                                        addl
                                               %eax, 16(%ebp)
  else
                                        cmpl
                                               .L2
    goto <bb 3>;
                                        jge
                                        popl
                                               %ebp
<bb 3>:
                                        ret
 return pretmp.2_7;
                                   .L2:
                                   .L3:
<bb 4>:
                                               .L3
                                        jmp
  goto <bb 4>;
```

Why infinite loop?



Redundant ϕ Function Elimination and Copy Propagation

Notes

July 2010

Essential Abstractions in GCC

GCC Resource Center, IIT Bombay

July 2010

Machine Independent Optimizations: Second Example Program

28/30

Final Assembly Program

Infinite Loop in Example Program 2

```
int f(int b, int c, int n)
{ int a;
   do
      a = b+c;
   while (a \le n);
   return a;
```

The program does not terminate unless a > n

Essential Abstractions in GCC

Part 3

Conclusions

Infinite Loop in Example Program 2

July 2010



Essential Abstractions in GCC

Conclusions

- GCC performs many machine independent optimizations
- The dumps of optimizations are easy to follow, particularly at the GIMPLE level
- 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)

GCC Resource Center, IIT Bomb



Note

July 2010

