The Design and Implementation of Gnu Compiler Collection

Uday Khedker
(www.cse.iitb.ac.in/~uday)

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

Aug 2008
Part 1

Compilation Flow: Machine Independent Phases
The GNU Tool Chain

Source Program

\[ \text{gcc} \]

Target Program
The GNU Tool Chain

Source Program

gcc

Target Program

cc1
The GNU Tool Chain

Source Program

gcc

Target Program

cc1

cpp
The GNU Tool Chain

Source Program

```
gcc
```

Target Program

```
cc1
cpp
as
```
The GNU Tool Chain

Source Program

gcc

cc1

cpp

as

ld

Target Program
The GNU Tool Chain

Source Program

\[ \text{gcc} \]

Target Program

\[ \text{cpp} \rightarrow \text{cc1} \rightarrow \text{gcc} \rightarrow \text{as} \rightarrow \text{ld} \rightarrow \text{glibc/newlib} \]
The GCC Framework

 GCC

Input Language

Machine Description

cc1

gcc
The GCC Framework

Input Language -> GCC -> Machine Description

Source Program -> cc1 -> gcc -> Assembly Program
The GCC Framework

Input Language

GCC

Language Specific Code

Language and Machine Independent Generic Code

Machine Dependent Generator Code

Machine Descriptions

Selected

Parser

gcc

cc1
The GCC Framework

Input Language

GCC

Language Specific Code

Selected

cc1

Parser

Genericizer

Gimplifier

Tree SSA Optimizer

Copied

Machine Description

GCC

Language and Machine Independent Generic Code

Copied

Machine dependent Generator Code

Copied

Machine Descriptions

Optimzer
The GCC Framework

Input Language

 GCC

Parser
Genericizer
Gimplifier
Tree SSA Optimizer
RTL Generator
Optimizer
Code Generator

Machine Description

Selected
Copied
Generated

Language Specific Code
Language and Machine Independent Generic Code
Machine dependent Generator Code
Machine Descriptions

cc1

gcc
The GCC Framework

Input Language

GCC

Language Specific Code

Language and Machine Independent Generic Code

Machine Description

Selected

Copied

cc1

Parser

Genericizer

Gimplifier

Tree SSA Optimizer

gcc

UPK

IIT Bombay
Invocation of cc1 from gcc (4.0.2)

```c
main()
    do_spec()
        do_spec_2()
            do_spec_1()
        execute()
            pexecute()  ..../libiberty/pex-unix.c
/* TO: cc1 */
```

Tip

Static Inspection: Use cscope and/or ctags
Dynamic Inspection: Set breakpoints in gdb on cc1.
The **cc1** Phase Sequence as IR Chain

The GCC Phase Sequence

Generic → GIMPLE → RTL → ASM
The **cc1 Phase Sequence as IR Chain**

The GCC Phase Sequence

- **Generic** → **GIMPLE** → **RTL** → **ASM**
- **HLL**
- **Scope Explication** → **Lower control flow** → **Lower procedures & data** → **Target**
Front End Processing Sequence for C in cc1 and GCC (4.0.2)

toplev_main ()
general_init ()
  init_tree_optimization_passes ()
decode_options ()
do_compile ()
  compile_file()
    lang_hooks.parse_file ()
      c_common_parse_file ()
        c_parse_file ()
          finish_function ()
            c_genericize ()

/* TO: Gimplification */
Creating GIMPLE representation in cc1 and GCC

finish_function ()
c_genericize()  
gimplyy_function_tree()  
gimplyy_body()  
gimplyy_stmt()  
gimplyy_expr()
c_expand_body()  
tree_rest_of_compilation()  
exeexecute_pass_list()  
exeexecute_one_pass()  

/* calls pass entry point function pointer */
targetm.asm_out.constructor()
The Tree passes list (4.0.2)

(Partial) Passes list (tree-optimize.c) (∼ 63 passes)

pass_remove_useless_stmts // Pass
pass_lower_cf // Pass
pass_all_optimizations // Optimiser
  pass_build_ssa // Optimiser
pass_dce // Optimiser
pass_loop // Optimiser
  pass_complete_unroll // Optimiser
  pass_loop_done // Optimiser
  pass_del_ssa // Optimiser
pass_warn_function_return // Optimiser
pass_expand // RTL Expander
pass_rest_of-compilation // RTL passes
Part 2

Adding a Pass on Gimple IR
 GCC Tree Passes: Code organisation

Tree Pass Organisation

- **Data structure** records pass info: name, function to execute etc. (struct tree_opt_pass in tree-pass.h)
- **Instantiate** a struct tree_opt_pass variable in each pass file.
- **List** the pass variables (in init_tree_optimization_passes.c).
Adding a Pass on Gimple IR

- Step 0. Write function `cs715_main()` in file `cs715.c`.
- Step 1. Create the following data structure in file `cs715.c`.

```c
struct tree_opt_pass pass_cs715 =
{
    "cs715",    /* name */
    NULL,       /* gate, for conditional entry to this pass */
    cs715_main, /* execute, main entry point */
    NULL,       /* sub-passes, depending on the gate predicate */
    NULL,       /* next sub-passes, independ of the gate predicate */
    0,          /* static_pass_number , used for dump file name*/
    0,          /* tv_id */
    0,          /* properties_required, indicated by bit position */
    0,          /* properties_provided , indicated by bit position*/
    0,          /* properties_destroyed , indicated by bit position*/
    0,          /* todo_flags_start */
    0,          /* todo_flags_finish */
    0          /* letter for RTL dump */
};
```
Adding a Pass on Gimple IR

• Step 2. Add the following line to tree-pass.h
  extern tree_opt_pass pass_cs715;
Adding a Pass on Gimple IR

- Step 2. Add the following line to tree-pass.h
  ```c
  extern tree_opt_pass pass_cs715;
  ```
- Step 3. Include the following call at an appropriate place in the function `init_tree_optimization_passes()` in the file `tree-optimize.c`
  ```c
  NEXT_PASS (pass_cs715);
  ```
Adding a Pass on Gimple IR

• Step 2. Add the following line to tree-pass.h:
  ```c
  extern tree_opt_pass pass_cs715;
  ```

• Step 3. Include the following call at an appropriate place in the function `init_tree_optimization_passes()` in the file `tree-optimize.c`:
  ```c
  NEXT_PASS (pass_cs715);
  ```

• Step 4. Add the file name in the Makefile:
  - Either in `$SOURCE/gcc/Makefile.in`
    Reconfigure and remake
  - Or in `$BUILD/gcc/Makefile`
    Remake
Adding a Pass on Gimple IR

• Step 2. Add the following line to tree-pass.h
  `extern tree_opt_pass pass_cs715;`

• Step 3. Include the following call at an appropriate place in the function `init_tree_optimization_passes()` in the file `tree-optimize.c`
  `NEXT_PASS (pass_cs715);`

• Step 4. Add the file name in the Makefile
  ▶ Either in `$SOURCE/gcc/Makefile.in`
     Reconfigure and remake
  ▶ Or in `$BUILD/gcc/Makefile`
     Remake

• Step 5. Build the compiler
Adding a Pass on Gimple IR

• Step 2. Add the following line to tree-pass.h
  
  ```c
  extern tree_opt_pass pass_cs715;
  ```

• Step 3. Include the following call at an appropriate place in the function `init_tree_optimization_passes()` in the file `tree-optimize.c`
  
  ```c
  NEXT_PASS (pass_cs715);
  ```

• Step 4. Add the file name in the Makefile
  
  ▶ Either in `$SOURCE/gcc/Makefile.in`
    Reconfigure and remake
  ▶ Or in `$BUILD/gcc/Makefile`
    Remake

• Step 5. Build the compiler

• Step 6. Wonder what went wrong?
Traversing Control Flow Graph

```c
for(n=0; n < number_of_nodes; n++) {
    bb = VARRAY_BB(dfs_order_bb,n);
    for(bsi = bsi_start(bb);!bsi_end_p(bsi);bsi_next(&bsi)) {
        // Block Statement Iterator
    }
}
```
Traversing Control Flow Graph

for(n=0; n < number_of_nodes; n++) {
    bb = VARRAY_BB(dfs_order_bb,n);
    for(bsi = bsi_start(bb); !bsi_end_p(bsi); bsi_next(&bsi)) {
        stmt = bsi_stmt(bsi);
        switch(TREE_CODE(stmt)) {
        }
    }
}
Traversing Control Flow Graph

for (n=0; n < number_of_nodes; n++) {
    bb = VARRAY_BB(dfs_order_bb, n);
    for (bsi = bsi_start(bb); !bsi_end_p(bsi); bsi_next(&bsi)) {
        stmt = bsi_stmt(bsi);
        switch (TREE_CODE(stmt)) {
            case MODIFY_EXPR:
                
            
        }
    }
}
Traversing Control Flow Graph

```c
for(n=0; n < number_of_nodes; n++) {
    bb = VARRAY_BB(dfs_order_bb,n);
    for(bsi = bsi_start(bb); !bsi_end_p(bsi); bsi_next(&bsi)) {
        stmt = bsi_stmt(bsi);
        switch(TREE_CODE(stmt)) {
        case MODIFY_EXPR:
            expr = TREE_OPERAND(stmt,1);
            /* SET_BIT(GEN(current_pf_L,bb),expr); */
            lval = TREE_OPERAND(stmt,0);
            FOR expr_no in exprList(lval) {
                /* RESET_BIT(av_L[bb]->gen),expr_no); */
                /* SET_BIT(av_L[bb]->kill),expr_no); */
            }
            break;
        ...
        }
    }
}
```