GCC Source Code: An Internal View

Uday Khedker

GCC Resource Center,
Department of Computer Science and Engineering,
Indian Institute of Technology, Bombay

Feb 2010
Outline

- A summary of GCC architecture
- Walking the maze of a large code base
- An Internal View of GCC code
The Gnu Tool Chain

Source Program

gcc

Target Program

cc1

cpp

as

GCC

glibc/newlib

ld
The GNU Tool Chain

Source Program -> gcc -> Target Program

Partially generated and downloaded source is compiled into executables

cc1 <-> cpp

as

glibc/newlib

ld
The Gnu Tool Chain

Source Program

- gcc

Target Program

- ld

- glibc/newlib

- as

- cpp

- cc1

Partially generated and downloaded source is compiled into executables

Existing executables are directly used

Jan 2010 Uday Khedker, IIT Bombay
The Architecture of GCC

Compiler Generation Framework

- Language Specific Code
- Language and Machine Independent Generic Code
- Machine Dependent Generator Code
- Machine Descriptions
The Architecture of GCC

Compiler Generation Framework

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

Parser
Gimplifier
Tree SSA Optimizer
RTL Generator
Optimizer
Code Generator

Source Program
Generated Compiler (cc1)
Assembly Program

Jan 2010
Uday Khedker, IIT Bombay
The Architecture of GCC

Input Language

Compiler Generation Framework

Source Program

Generated Compiler (cc1)

Assembly Program

Language Specific Code

Language and Machine Independent Generic Code

Machine Dependent Generator Code

Machine Descriptions

Selected

Parsed

Gimplifier

Tree SSA Optimizer

RTL Generator

Optimizer

Code Generator

Jan 2010 Uday Khedker, IIT Bombay
The Architecture of GCC

- Input Language
- Compiler Generation Framework
- Target Name

- Language Specific Code
- Language and Machine Independent Generic Code
- Machine Dependent Generator Code
- Machine Descriptions

- Parser
- Gimplifier
- Tree SSA Optimizer
- RTL Generator
- Optimizer
- Code Generator

- Generated Compiler (cc1)

- Development Time
- Build Time
- Use Time

Jan 2010
Uday Khedker, IIT Bombay
An Example of The Generation Related Gap

- Predicate function for invoking the loop distribution pass

```c
static bool
gate_tree_loop_distribution (void)
{
    return flag_tree_loop_distribution != 0;
}
```
An Example of The Generation Related Gap

- Predicate function for invoking the loop distribution pass
  
  ```
  static bool
gate_tree_loop_distribution (void)
  {
    return flag_tree_loop_distribution != 0;
  }
  ```

- There is no declaration of or assignment to variable flag_tree_loop_distribution in the entire source!
An Example of The Generation Related Gap

- Predicate function for invoking the loop distribution pass

```c
static bool
gate_tree_loop_distribution (void)
{
    return flag_tree_loop_distribution != 0;
}
```

- There is no declaration of or assignment to variable `flag_tree_loop_distribution` in the entire source!

- It is described in `common.opt` as follows

```
ftree-loop-distribution
Common Report Var(flag_tree_loop_distribution) Optimization
Enable loop distribution on trees
```
An Example of The Generation Related Gap

- Predicate function for invoking the loop distribution pass

```c
static bool
gate_tree_loop_distribution (void)
{
    return flag_tree_loop_distribution != 0;
}
```

- There is no declaration of or assignment to variable `flag_tree_loop_distribution` in the entire source!

- It is described in `common.opt` as follows

```
ftree-loop-distribution
Common Report Var(flag_tree_loop_distribution) Optimization
Enable loop distribution on trees
```

- The required C statements are generated during the build
Another Example of The Generation Related Gap

Locating the `main` function in the directory `gcc-4.4.2/gcc` using cscope

<table>
<thead>
<tr>
<th>File</th>
<th>Line</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>collect2.c</td>
<td>766</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>fix-header.c</td>
<td>1074</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>fp-test.c</td>
<td>85</td>
<td>main (void )</td>
</tr>
<tr>
<td>gcc.c</td>
<td>6216</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>gcov-dump.c</td>
<td>76</td>
<td>main (int argc ATTRIBUTE_UNUSED, char **argv)</td>
</tr>
<tr>
<td>gcov-iov.c</td>
<td>29</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>gcov.c</td>
<td>355</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>gen-protos.c</td>
<td>130</td>
<td>main (int argc ATTRIBUTE_UNUSED, char **argv)</td>
</tr>
<tr>
<td>genattr.c</td>
<td>89</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>genattrtab.c</td>
<td>4438</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>genautomata.c</td>
<td>9321</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>genchecksum.c</td>
<td>65</td>
<td>main (int argc, char ** argv)</td>
</tr>
<tr>
<td>gencodes.c</td>
<td>51</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>genconditions.c</td>
<td>209</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>genconfig.c</td>
<td>261</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>genconstants.c</td>
<td>50</td>
<td>main (int argc, char **argv)</td>
</tr>
</tbody>
</table>
Another Example of The Generation Related Gap

Locating the main function in the directory gcc-4.4.2/gcc using cscoe

<table>
<thead>
<tr>
<th>File</th>
<th>Line</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>collect2.c</td>
<td>766</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>fix-header.c</td>
<td>1074</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>fp-test.c</td>
<td>85</td>
<td>main (void )</td>
</tr>
<tr>
<td>gcc.c</td>
<td>6216</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>gcov-dump.c</td>
<td>76</td>
<td>main (int argc ATTRIBUTE UNUSED, char **argv)</td>
</tr>
<tr>
<td>gcov iov.c</td>
<td>29</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>gcov.c</td>
<td>355</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>gen-protos.c</td>
<td>130</td>
<td>main (int argc ATTRIBUTE UNUSED, char **argv)</td>
</tr>
<tr>
<td>genattr.c</td>
<td>89</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>genattribtab.c</td>
<td>4438</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>genautomata.c</td>
<td>9321</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>genchecksum.c</td>
<td>65</td>
<td>main (int argc, char ** argv)</td>
</tr>
<tr>
<td>gencodes.c</td>
<td>51</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>genconditions.c</td>
<td>209</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>genconfig.c</td>
<td>261</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>genconstants.c</td>
<td>50</td>
<td>main (int argc, char **argv)</td>
</tr>
</tbody>
</table>
Another Example of The Generation Related Gap

Locating the main function in the directory gcc-4.4.2/gcc using cscope

<table>
<thead>
<tr>
<th>File</th>
<th>Line</th>
<th>Function Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>genemit.c</td>
<td>820</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>genextract.c</td>
<td>394</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>genflags.c</td>
<td>231</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>gengenrtl.c</td>
<td>350</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>gengtype.c</td>
<td>3584</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>genmddeps.c</td>
<td>45</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>genmodes.c</td>
<td>1376</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>genopinit.c</td>
<td>472</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>genoutput.c</td>
<td>1005</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>genpeep.c</td>
<td>353</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>genpreds.c</td>
<td>1399</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>genrecog.c</td>
<td>2718</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>main.c</td>
<td>33</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>mips-tdump.c</td>
<td>1393</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>mips-tfile.c</td>
<td>655</td>
<td>main (void )</td>
</tr>
<tr>
<td>mips-tfile.c</td>
<td>4690</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>protoize.c</td>
<td>4373</td>
<td>main (int argc, char **const argv)</td>
</tr>
</tbody>
</table>
Transformation Passes in GCC

- A total of 196 unique pass names initialized in
  `$\text{SOURCE}/gcc/passes.c$
  - Some passes are called multiple times in different contexts
    - Conditional constant propagation and dead code elimination are called thrice
  - Some passes are only demo passes (e.g., data dependence analysis)
  - Some passes have many variations (e.g., special cases for loops)
    - Common subexpression elimination, dead code elimination

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

- Some passes are organizational passes to group related passes
Basic Transformations in GCC

Target Independent → Target Dependent

Parse → Simplify → Tree SSA Optimize → Generate RTL → Optimize RTL → Generate ASM

Gimple → RTL

RTL → ASM
Basic Transformations in GCC

Target Independent → Target Dependent

Parse → Gimplify → Tree SSA Optimize → Generate RTL → Optimize RTL → Generate ASM

↓ Gimple Passes ↓ ↓ RTL Passes ↓ ↓

Gimple → RTL

RTL → ASM
# Passes On Gimple

<table>
<thead>
<tr>
<th>Pass Group</th>
<th>Examples</th>
<th>Number of passes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lowering</td>
<td>Gimple IR, CFG Construction</td>
<td>12</td>
</tr>
<tr>
<td>Interprocedural Optimizations</td>
<td>Conditional Constant Propagation, Inlining, SSA Construction</td>
<td>36</td>
</tr>
<tr>
<td>Intraprocedural Optimizations</td>
<td>Constant Propagation, Dead Code Elimination, PRE</td>
<td>40</td>
</tr>
<tr>
<td>Loop Optimizations</td>
<td>Vectorization, Parallelization</td>
<td>24</td>
</tr>
<tr>
<td>Remaining Intraprocedural Optimizations</td>
<td>Value Range Propagation, Rename SSA</td>
<td>23</td>
</tr>
<tr>
<td>Generating RTL</td>
<td></td>
<td>01</td>
</tr>
<tr>
<td>Total number of passes on Gimple</td>
<td></td>
<td>136</td>
</tr>
</tbody>
</table>
# Passes On RTL

<table>
<thead>
<tr>
<th>Pass Group</th>
<th>Examples</th>
<th>Number of passes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Intraprocedural Optimizations</strong></td>
<td>CSE, Jump Optimization</td>
<td>15</td>
</tr>
<tr>
<td><strong>Loop Optimizations</strong></td>
<td>Loop Invariant Movement, Peeling, Unswitching</td>
<td>7</td>
</tr>
<tr>
<td><strong>Machine Dependent Optimizations</strong></td>
<td>Register Allocation, Instruction Scheduling, Peephole Optimizations</td>
<td>59</td>
</tr>
<tr>
<td><strong>Assembly Emission and Finishing</strong></td>
<td></td>
<td>03</td>
</tr>
<tr>
<td><strong>Total number of passes on RTL</strong></td>
<td></td>
<td>84</td>
</tr>
</tbody>
</table>

Jan 2010

Uday Khedker, IIT Bombay
## Comprehensiveness of GCC 4.4.2: Size

<table>
<thead>
<tr>
<th>Source Lines</th>
<th>Number of lines in the main source</th>
<th>2,187,216</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Number of lines in libraries</td>
<td>1,633,558</td>
</tr>
<tr>
<td>Directories</td>
<td>Number of subdirectories</td>
<td>3794</td>
</tr>
<tr>
<td>Files</td>
<td>Total number of files</td>
<td>62998</td>
</tr>
<tr>
<td></td>
<td>C source files</td>
<td>13968</td>
</tr>
<tr>
<td></td>
<td>Header files</td>
<td>9163</td>
</tr>
<tr>
<td></td>
<td>C++ files</td>
<td>4191</td>
</tr>
<tr>
<td></td>
<td>Java files</td>
<td>6340</td>
</tr>
<tr>
<td></td>
<td>Makefiles and Makefile templates</td>
<td>163</td>
</tr>
<tr>
<td></td>
<td>Configuration scripts</td>
<td>52</td>
</tr>
<tr>
<td></td>
<td>Machine description files</td>
<td>206</td>
</tr>
</tbody>
</table>

(Line counts estimated by the program `sloccount` by David A. Wheeler)
Walking the Maze of a Large Code Base

- Use cscope
  ```shell
cd $SOURCE
cscope -R
  ```

- Use ctags
  ```shell
cd $SOURCE
ctags -R
  ```

  Make sure you use exeburant-ctags
main
  validate_all_switches
lookup_compiler
do_spec
  do_spec_2
    do_spec_1  /* Get the name of the compiler */
execute
  pex_init
  pex_run
  pex_run_in_environment
    obj->funcs->exec_child
cc1-4.4.2 Control Flow

```c
main
toplev_main
decode_options
do_compile
  compile_file
    { lang_hooks.parse_file => c_common_parse_file
      { c_parse_file
        c_parser_translation_unit
        c_parser_declaration_or_fndef
        finish_function
        c_genericize
        simplify_function_tree
        simplify_body
        simplify_stmt
        simplify_expr
        cgraph_finalize_function
        pop_file_scope
        cgraph_finalize_compilation_unit
        cgraph_analyze_functions
        cgraph_analyze_function
        cgraph_lower_function
        tree_lowering_passes
        execute_pass_list (&all_lowering_passes)
      }
      lang_hooks.decls.final_writeGlobals => c_write_global_declarations
        { cgraph_optimize
          cgraph_analyze_functions
          cgraph_analyze_function
          cgraph_lower_function
          tree_lowering_passes
          execute_pass_list (&all_lowering_passes)
          ipa_passes
          cgraph_expand_all_functions
          cgraph_expand_functions
          tree_rest_of_compilation
          execute_pass_list (&all_passes)
        }
      }
      targetm.asm_out.file_end
    }
  }
finalise
```
cc1-4.4.2 Control Flow: Lowering Passes

lang_hooks.parse_file => c_common_parse_file
c_parse_file
  c_parser_translation_unit
    c_parser_declaration_or_fndef
    finish_function
        c_genericize
            gimplify_function_tree
            gimplify_body
                gimplify_stmt
                    gimplify_expr
            cgraph_finalize_function
        pop_file_scope
            cgraph_finalize_compilation_unit
                cgraph_analyze_functions
                    cgraph_analyze_function
                    cgraph_lower_function
                tree_lowering_passes
                    execute_pass_list (all_lowering_passes)
**Control Flow: Optimization and Code Generation**

```
lang_hooks.decls.final_write_globals => c_write_global_declarations
{
  cgraph_optimize
  cgraph_analyze_functions
  cgraph_analyze_function
  cgraph_lower_function
  tree_lowering_passes
  execute_pass_list (&all_lowering_passes)
  ipa_passes
  cgraph_expand_all_functions
  cgraph_expand_functions
  tree_rest_of_compilation
  execute_pass_list (&all_passes)
}
```