The Structure of a GCC Front End

Gustavo Sverzut Barbieri       Rafael Ávila de Espíndola

April 21\textsuperscript{th}, 2006
1 Introduction

2 General Structure of a Front End

3 General Structure of GCC

4 The GCC ILs

5 The Front End Interface

6 Further Information

7 Contact
To compile C, Java, and Fortran to x86, amd64, powerpc we would need 9 compilers!
Using a common intermediate language:

Only 6 translators are needed.
The language to IL translators are called **front ends**

The IL to assembly translators are called **back ends**

The IL → IL passes are called the **middle end**

Most optimizations can be implemented on **middle end** level and are language and target independent
Compiler Layers — The Big Picture

Introduction
### Front end example

#### Introduction

```c
for (i = 0; i < 6; i++)
{
    body
}
```

```pascal
for i := 0 to 5 do
begin
  body
end
```

C

Pascal

if (i < 6)
Middle end example

Introduction

```
loop invariant
optimization
if (...)
a = 5
f(x,y,a)
if (...)  
```

```
loop invariant
optimization
if (...)
a = 5
f(x,y,a)
if (...)  
```
Back end example

Introduction

\[ a = b + c \]

- **x86**
  - `movl %ebx, %eax`
  - `addl %ecx, %eax`

- **arm**
  - `add r0, r1, r2`
General Structure of a Front End

A front end usually has

- A lexer
- A parser
- An abstract syntax tree
- Type checking
- A converter for the syntax tree to the compiler IL
- Some front ends may not have some of them
General Structure of GCC

- There is a compiler and a driver for each language.
- The compiler just translates the source to assembly.
- The driver calls the compiler, the assembler and the linker.
- Drivers: gcc, gcj
- Compilers: cc1, jc1
- Each compiler lives in a directory of the gcc directory:
  - gcc/cp: the c++ front end
  - gcc/java: the java front end
**Intermediate Languages**

The GCC ILs

- **GENERIC**  Very high level. Generated by most front ends

- **GIMPLE**  A simplified GENERIC in Static Single Assignment (SSA) form

- **RTL**  Register Transfer Language. A low level representation used in the back ends

- GENERIC and GIMPLE use the same data structure

- The difference is in which constructs are allowed
GCC Intermediate Language Relationship

The GCC ILs

- C
- Java
- Fortran
- GENERIC
- GIMPLE
- RTL
- x86
- amd64
- powerpc

Languages Translator:
- Fortran
- Java
- C

Intermediate Languages:
- GENERIC
- GIMPLE
- RTL

Hardware Architectures:
- x86
- amd64
- powerpc
The GCC ILs

- The data structure used for GENERIC and GIMPLE is called \textit{tree}.
- It is a gigantic union. Each instance can be a variable, a function, a statement, etc.
- It is called \textit{tree} because of how the representation looks like:

```
for
    init
        ...
    conditional
        ...
    increment
        ...
    body
        expr1
            ...
        expr2
            ...
```
The front end has to
- understand the source language
- build the trees

For building trees there are many helper functions
- `build_fn_decl(name, type)`
- `build_string(len, size)`
- `build_pointer_type(type)`
- `build_function_call_expr(function, args)`
GCC controls most of the compiler behavior
- Provides the main function
- Parses options
- Handle language independent options

The Front End
- Provides callbacks for
  - Initialization
  - Parsing a file
  - Processing a language specific option
  - Many others
The cgraph Module

The front end interface is managed by the cgraph module.

- Each constructed function is transferred with `cgraph_finalize_function`.

To finish the compile unit call `cgraph_finalize_compilation_unit`.

To finish the job call `cgraph_optimize`.

- cgraph may compile one function at a time or accumulate.
The Front End Interface

Call Graph

- cgraph_finalize_function
- tree_rest_of_compilation
- cgraph_finalize_compilation_unit
- cgraph_optimize
- hello_init
- hello_parse_file
- hello_expand_function
- cgraph_optimize
- cgraph_finalize_compilation_unit
The Hello World front end: http://svn.gna.org/viewcvs/gsc/branches/hello-world/

GCC Scheme Compiler (GSC): http://gna.org/projects/gsc

GCC TreeLang: /trunk/gcc/treelang

info gcc
Gustavo Sverzut Barbieri

Email: gustavo.barbieri@indt.org.br
Website: http://www.gustavobarbieri.com.br
ICQ: 17249123
MSN, Jabber: barbieri@gmail.com

Rafael Ávila de Espíndola

Email: rafael.espindola@indt.org.br
Jabber: rafael.espindola@jabber.org