Workshop on Essential Abstractions in GCC

A Summary of Essential Abstrations

GCC Resource Center

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

Department of Computer Science and Engineering,

Indian Institute of Technology, Bombay

3 July 2011

Compilation Models

Aho Ullman Model

$$\begin{tikzcd}[row sep=small]
\boxed{\text{Front End}} \arrow[d] \\
\text{AST} \arrow[d] \\
\boxed{\text{Optimizer}} \arrow[d] \\
\text{Target Indep. IR} \arrow[d] \\
\boxed{\text{Code Generator}} \arrow[d] \\
\text{Target Program}
\end{tikzcd}$$

Aho Ullman: Instruction selection

- over optimized IR using
- cost based tree pattern matching

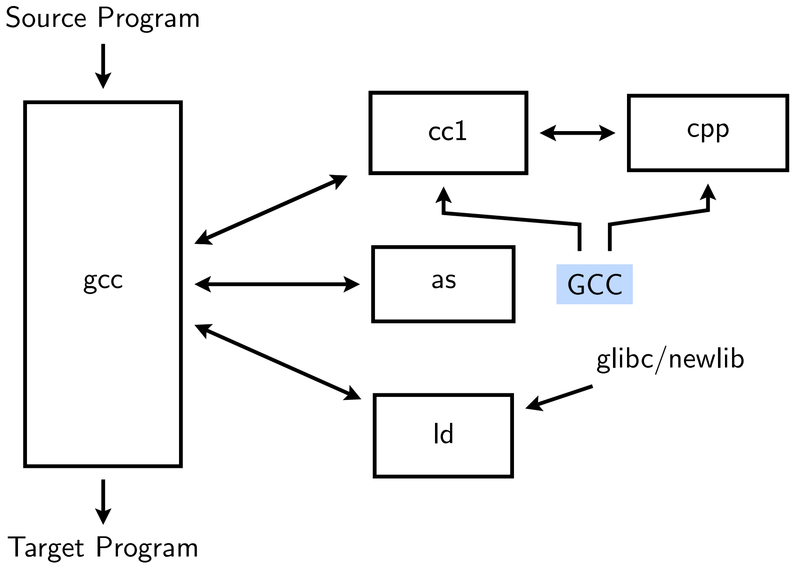
Davidson Fraser: Instruction selection

- over AST using
- structural tree pattern matching
- naive code which is
  - target dependent, and is
  - optimized subsequently

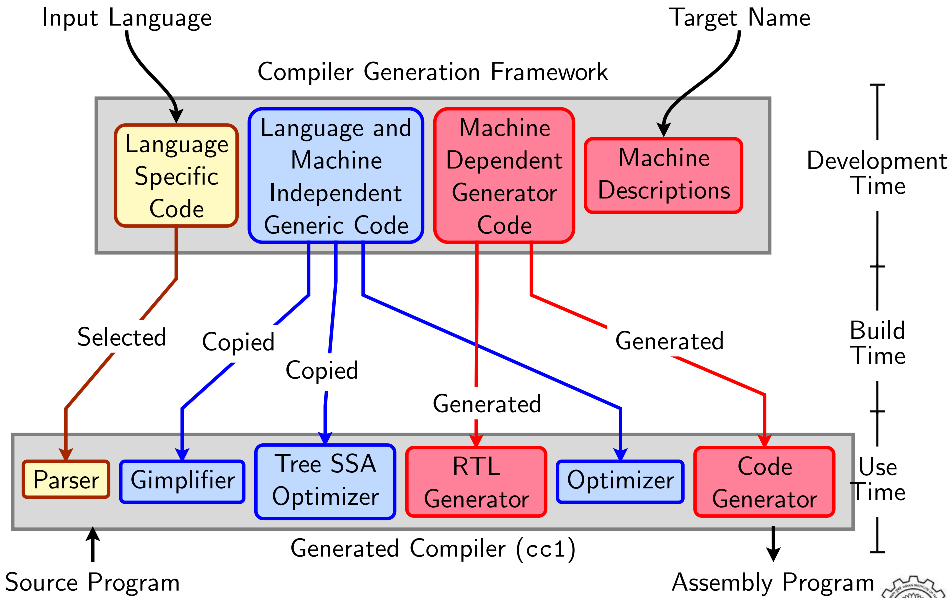
Davidson Fraser Model

$$\begin{tikzcd}[row sep=small]
\boxed{\text{Front End}} \arrow[d] \\
\text{AST} \arrow[d] \\
\boxed{\text{Expander}} \arrow[d] \\
\text{Register Transfers} \arrow[d] \\
\boxed{\text{Optimizer}} \arrow[d] \\
\text{Register Transfers} \arrow[d] \\
\boxed{\text{Recognizer}} \arrow[d] \\
\text{Target Program}
\end{tikzcd}$$

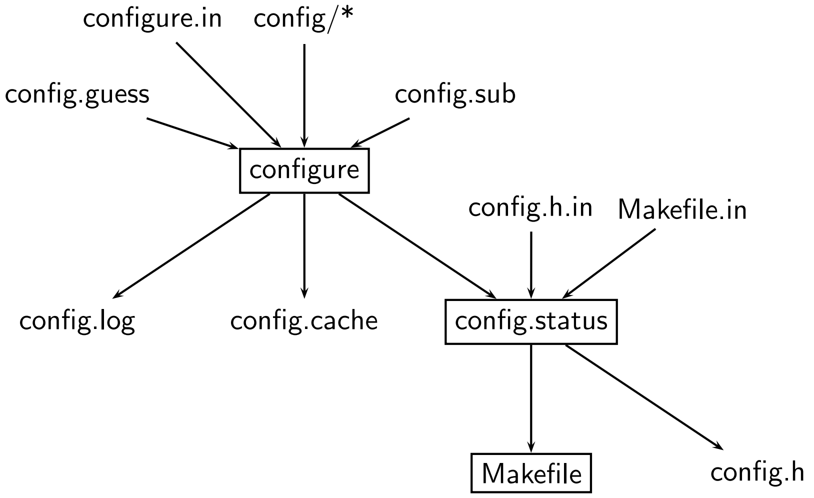
The GNU Tool Chain



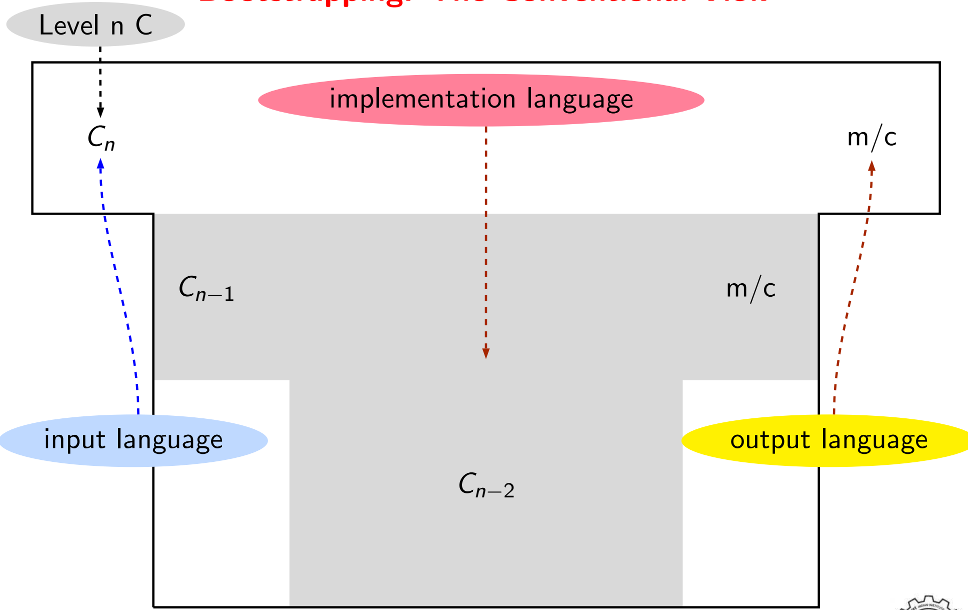
The Architecture of GCC



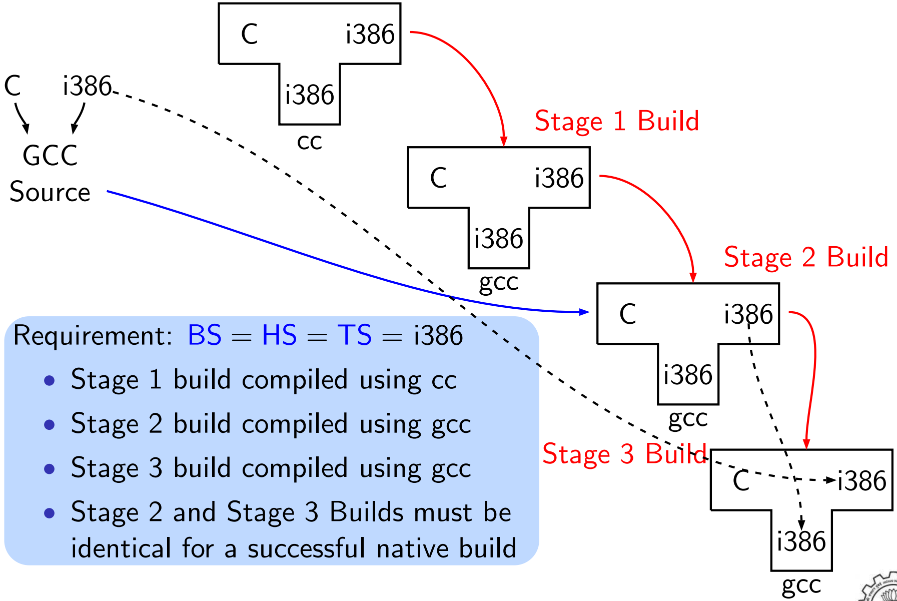
Configuring GCC



Bootstrapping: The Conventional View



A Native Build on i386



Requirement: BS = HS = TS = i386

- Stage 1 build compiled using cc
- Stage 2 build compiled using gcc
- Stage 3 build compiled using gcc
- Stage 2 and Stage 3 Builds must be identical for a successful native build

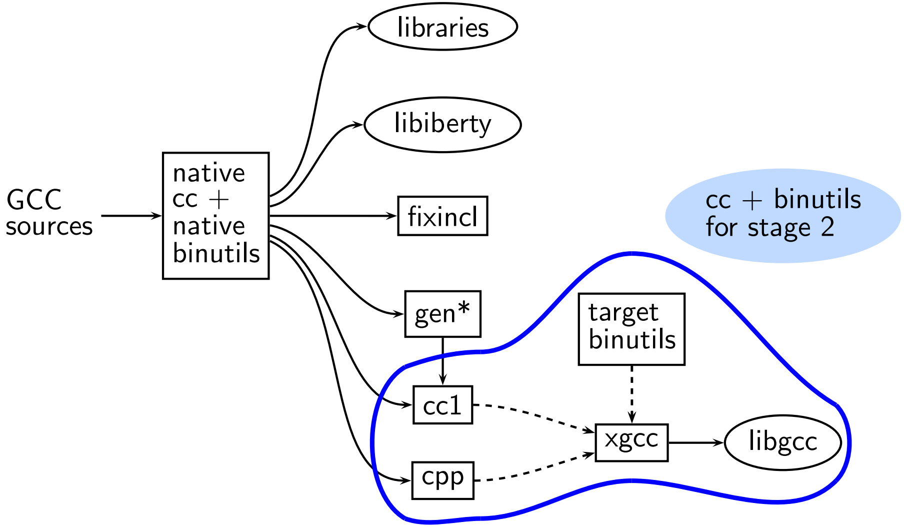
Build for a Given Machine

This is what actually happens!

- Generation
  - Generator sources (`$(SOURCE_D)/gcc/gen*.c`) are read and generator executables are created in `$(BUILD)/gcc/build`
  - MD files are read by the generator executables and back end source code is generated in `$(BUILD)/gcc`
- Compilation
  Other source files are read from `$(SOURCE_D)` and executables created in corresponding subdirectories of `$(BUILD)`
- Installation
  Created executables and libraries are copied in `$(INSTALL)`

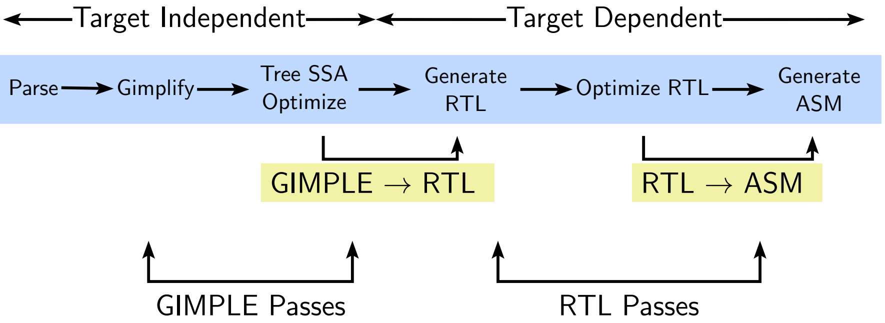
```
genattr
gencheck
genconditions
genconstants
genflags
genopinit
genpreds
genattrtab
genchecksum
gencondmd
genemit
gengenrtl
genmddeps
genoutput
genrecog
genautomata
gencodes
genconfig
genextract
gengtype
genmodes
genpeep
```

More Details of an Actual Stage 1 Build for C

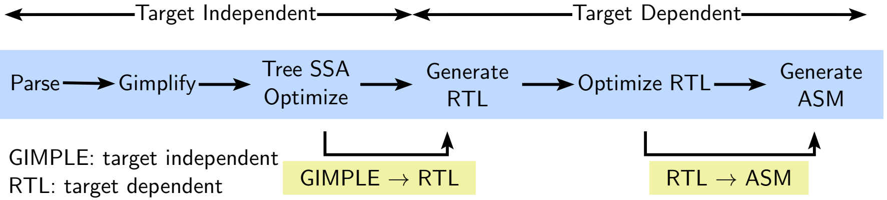


Basic Transformations in GCC

Tranformation from a language to a *different* language



Instruction Specification and Translation: A Recap



- GIMPLE: target independent
- RTL: target dependent
- Need: associate the *semantics*

⇒GCC Solution: Standard Pattern Names

RTL Template

ASM

GIMPLE_ASSIGN

```
(define_insn  "movsi"
   [(set (match_operand 0 "register_operand" "r")
         (match_operand 1 "const_int_operand" "k"))]
   "" /* C boolean expression, if required */
    "li %0, %1"
)
```

Translation Sequence in GCC

```
(define_insn
    "movsi"
     [(set
       (match_operand 0 "register_operand" "r")
       (match_operand 1 "const_int_operand" "k")
       )]
    "" /*  C boolean expression, if required */
    "li %0, %1"
)
```

Development

```
D.1283 = 10;
```

⇒

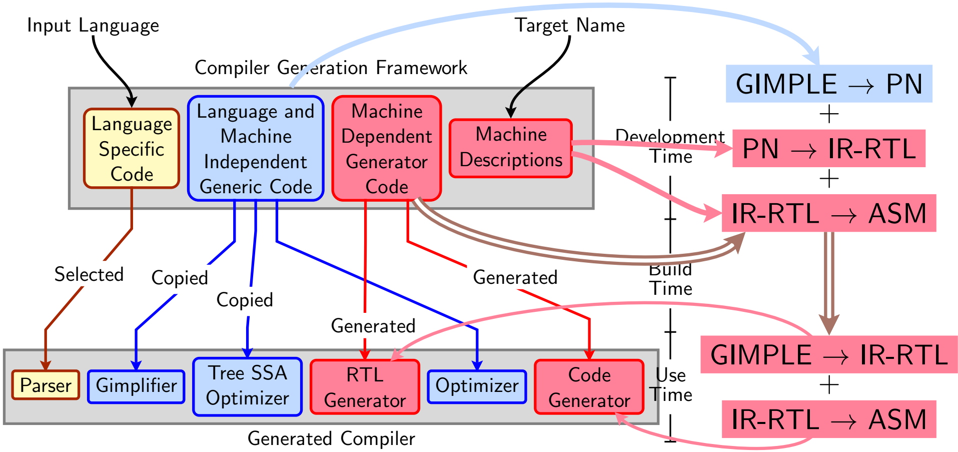
```
(set
     (reg:SI 58 [D.1283])
     (const_int 10:  [0xa])
)
```

⇒

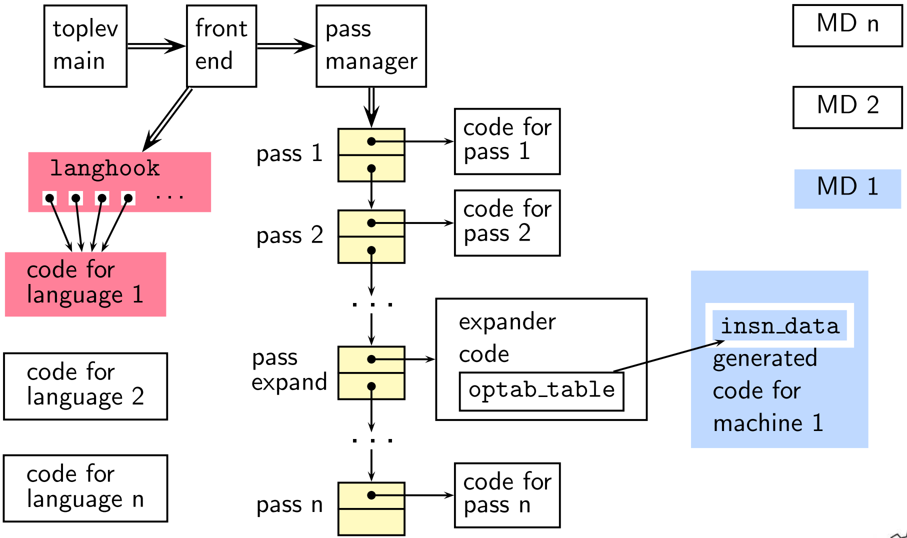
```
li $t0, 10
```

Use

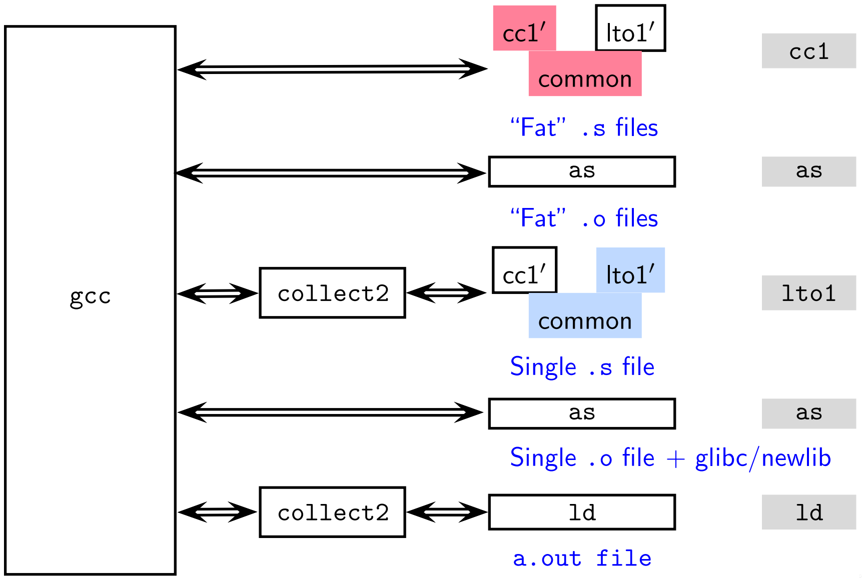
Retargetability Mechanism of GCC



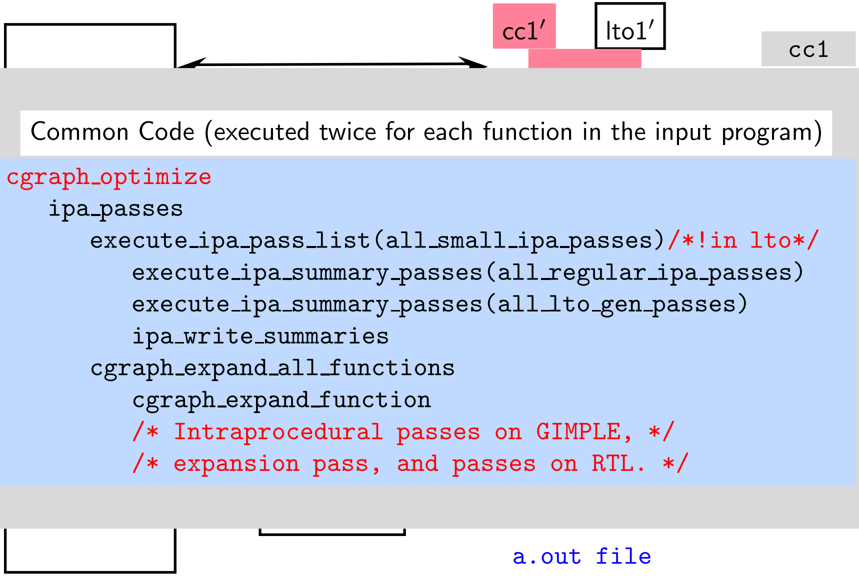
Plugin Structure in `cc1`



The GNU Tool Chain for LTO Support



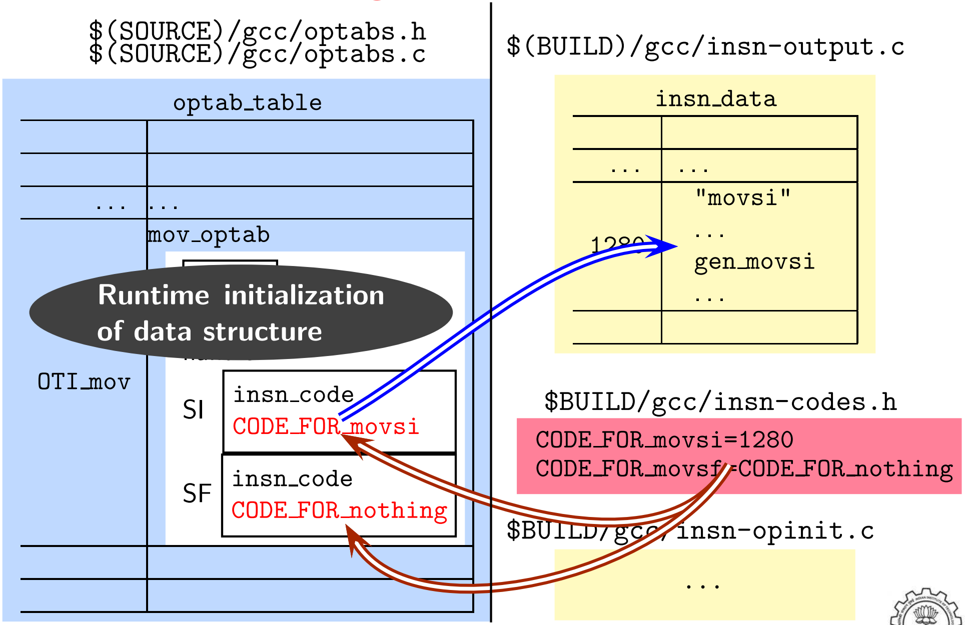
The GNU Tool Chain for LTO Support



Common Code (executed twice for each function in the input program)

```
cgraph_optimize
   ipa_passes
      execute_ipa_pass_list(all_small_ipa_passes)/*!in lto*/
         execute_ipa_summary_passes(all_regular_ipa_passes)
         execute_ipa_summary_passes(all_lto_gen_passes)
         ipa_write_summaries
      cgraph_expand_all_functions
         cgraph_expand_function
         /* Intraprocedural passes on GIMPLE, */
         /* expansion pass, and passes on RTL. */
```

Hooking up Back End Details


$(SOURCE)/gcc/optabs.h
$(SOURCE)/gcc/optabs.c
optab_table
... ...
mov_optab
Runtime initialization of data structure
OTI_mov
SI
insn_code
CODE_FOR_movsi
SF
insn_code
CODE_FOR_nothing
$(BUILD)/gcc/insn-output.c
insn_data
... ...
1280
"movsi"
...
gen_movsi
...
$BUILD/gcc/insn-codes.h
CODE_FOR_movsi=1280
CODE_FOR_movsf=CODE_FOR_nothing
$BUILD/gcc/insn-opinit.c
...