Workshop on Essential Abstractions in GCC

A Summary of Essential Abstrations

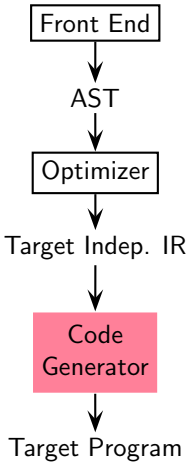
GCC Resource Center
(www.cse.iitb.ac.in/grc)

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

3 July 2011

Compilation Models

Aho Ullman Model



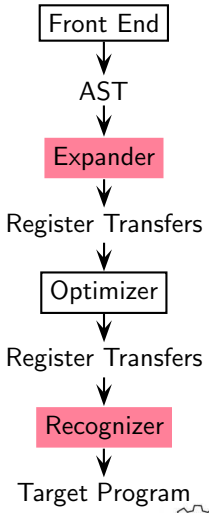
Aho Ullman: Instruction selection

- over optimized IR using
- cost based tree pattern matching

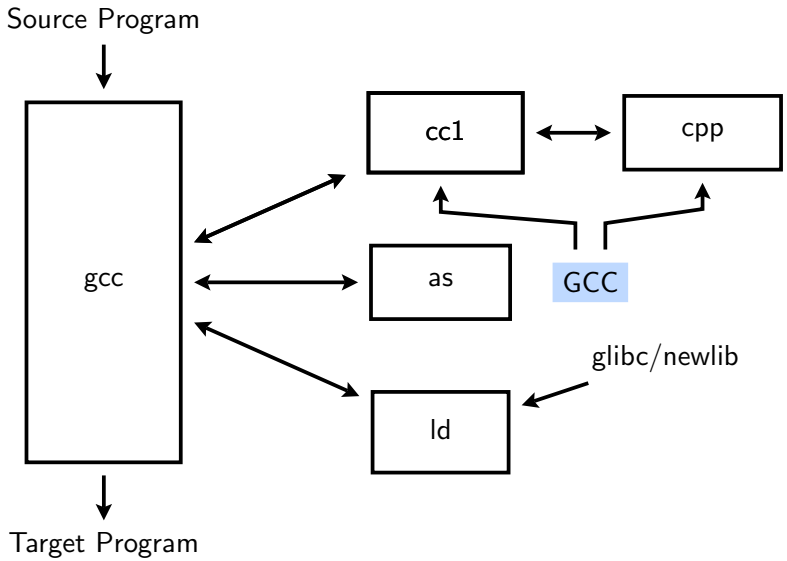
Davidson Fraser: Instruction selection

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

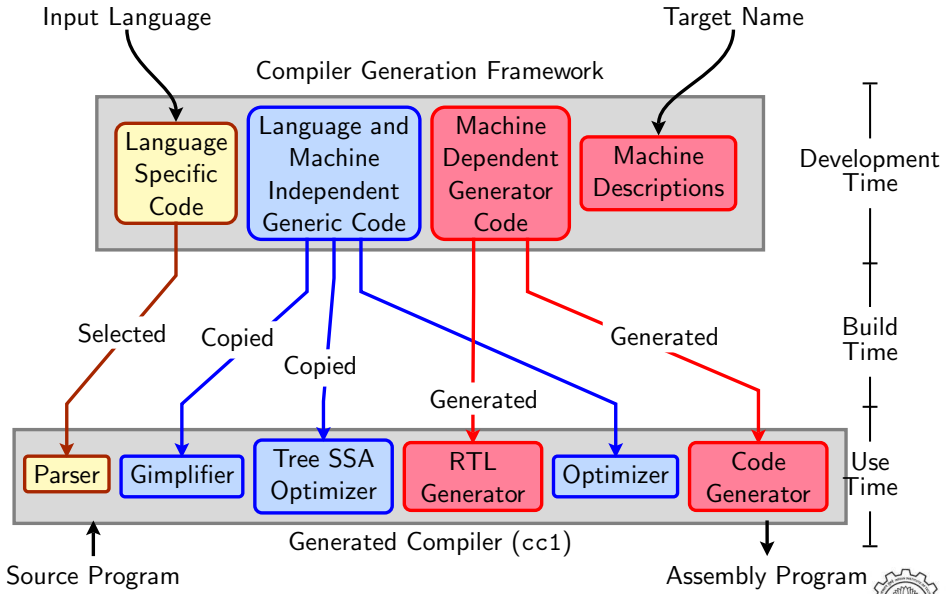
Davidson Fraser Model



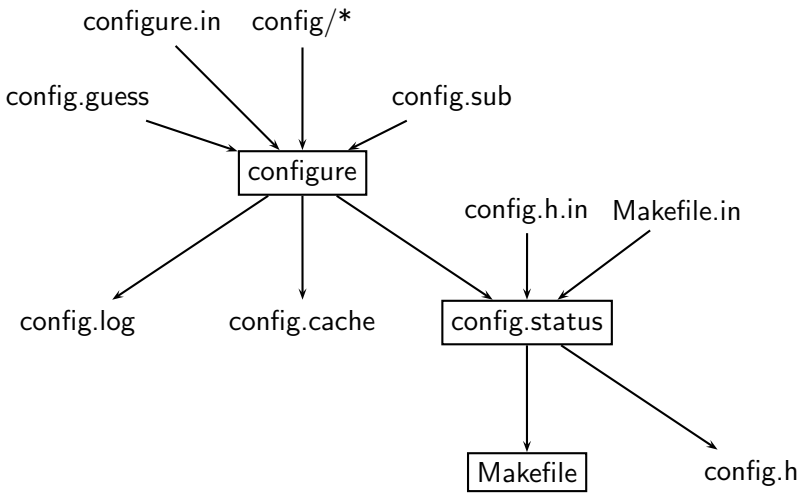
The GNU Tool Chain



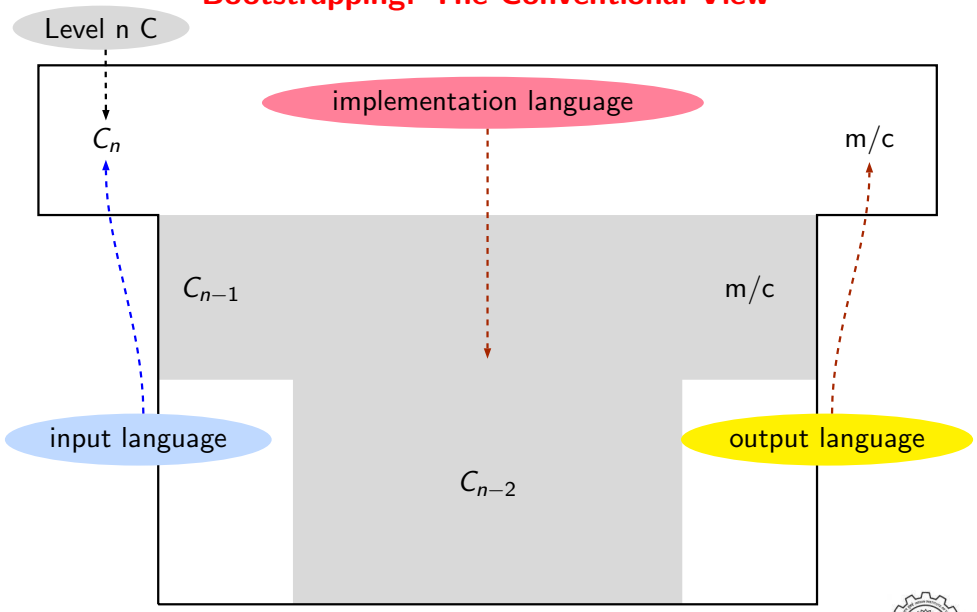
The Architecture of GCC



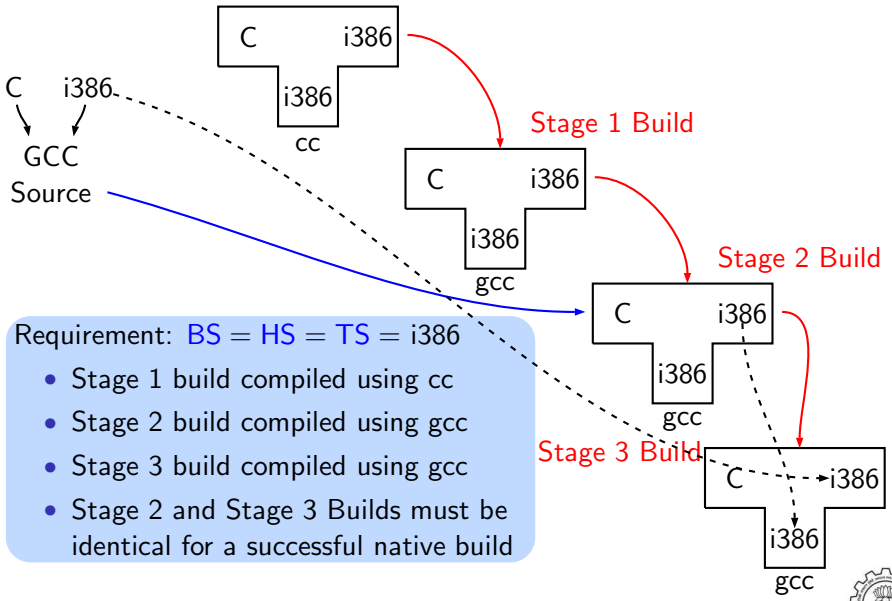
Configuring GCC



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

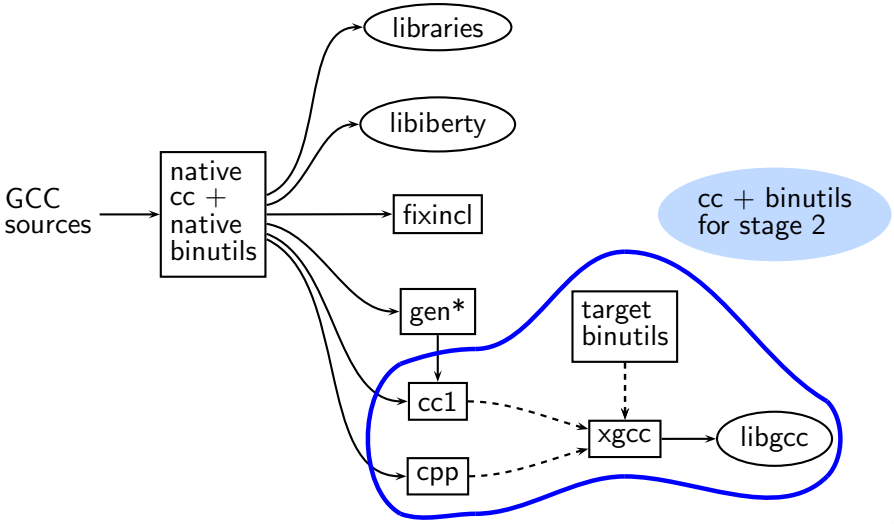
Build for a Given Machine

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)`

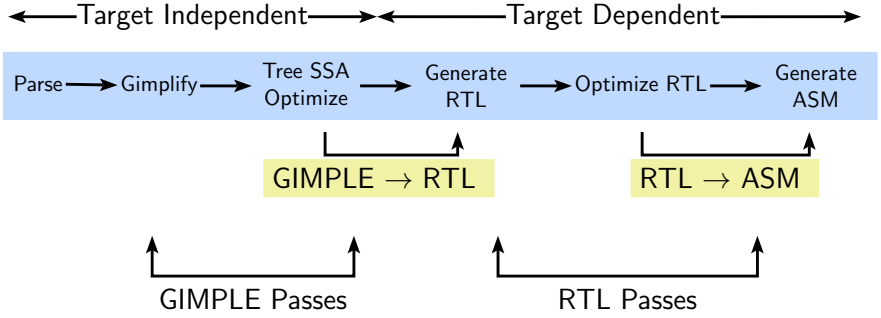
```
genattr
gencheck
genconditions
genconstants
genflags
genopinit
genpreds
genattrtab
gencheckcsum
gencondmd
genemit
gengenrtl
genmddeps
genoutput
genrecog
genautomata
gencodes
genconfig
genextract
gengtype
genmodes
genpeep
```


More Details of an Actual Stage 1 Build for C

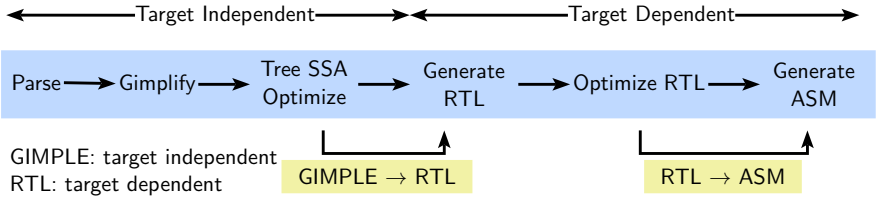


Basic Transformations in GCC

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

GIMPLE_ASSIGN

RTL Template

ASM

```
(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

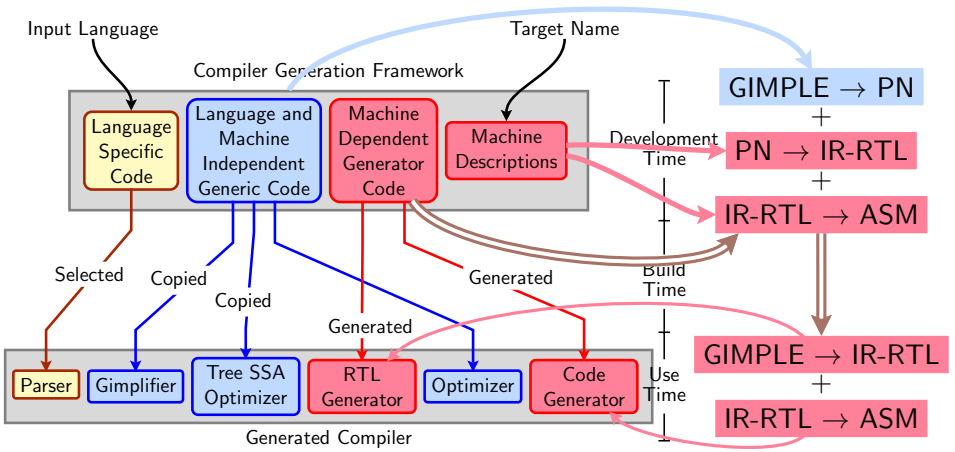
```
(define_insn
   "movsi"
    [(set
      (match_operand 0 "register_operand" "r")
      (match_operand 1 "const_int_operand" "k")
      )]
   "" /*  C boolean expression, if required */
   "li %0, %1"
)
```

Development

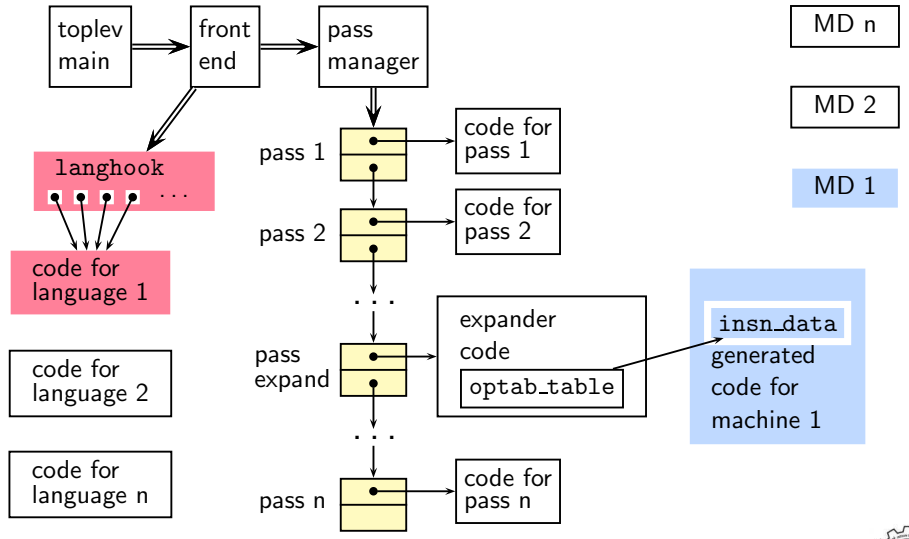
```
D.1283 = 10;  ⇒  (set
                    (reg:SI 58 [D.1283])
                    (const_int 10:  [0xa])
                 )
              ⇒  li $t0, 10
```

Use

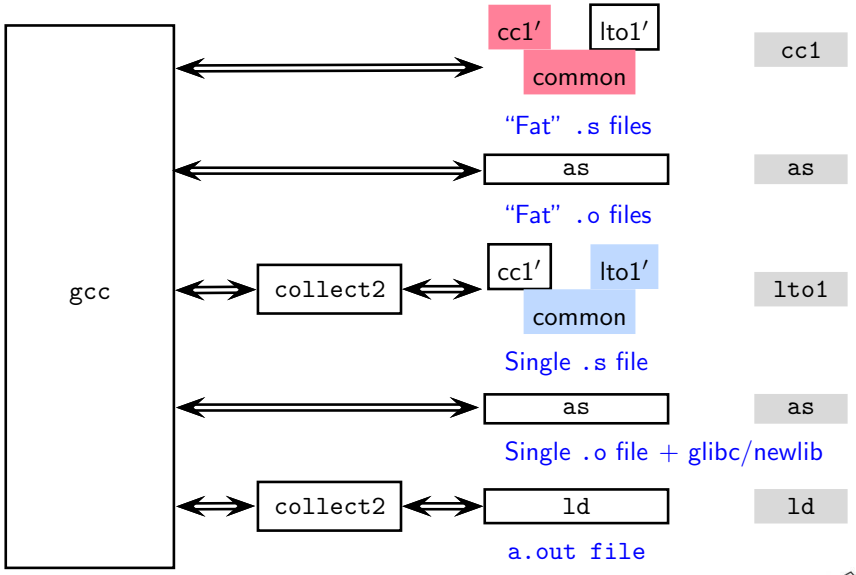
Retargetability Mechanism of GCC



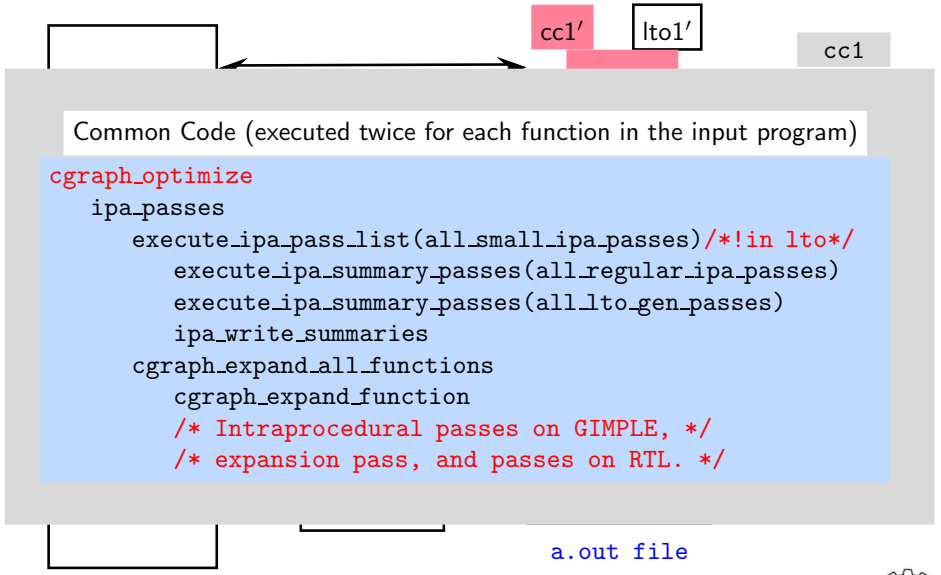
Plugin Structure in cc1



The GNU Tool Chain for LTO Support



The GNU Tool Chain for LTO Support



Hooking up Back End Details