Essential Abstractions in GCC

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

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

13 June 2014
Outline

• Compilation Models
• GCC: The Great Compiler Challenge
• Meeting the GCC Challenge: CS 715
  The course plan
Part 1

Compilation Models
Compilation Models

Aho Ullman Model

Davidson Fraser Model
Compilation Models

Aho Ullman Model

Front End

AST

Davidson Fraser Model

Input Source Program
Compilation Models

Aho Ullman Model

Front End \[\rightarrow\] AST \[\rightarrow\] Optimizer \[\rightarrow\] Target Indep. IR

Davidson Fraser Model

Input Source Program
Compilation Models

Aho Ullman Model

Front End

AST

Optimizer

Target Indep. IR

Code Generator

Target Program

Davidson Fraser Model

Input Source Program
Compilation Models

Aho Ullman Model

Front End

AST

Optimizer

Target Indep. IR

Code Generator

Target Program

Davidson Fraser Model

Input Source Program

Front End

AST
Compilation Models

**Aho Ullman Model**

1. Front End
2. AST
3. Optimizer
4. Target Indep. IR
5. Code Generator
6. Target Program

**Davidson Fraser Model**

1. Front End
2. AST
3. Expander
4. Register Transfers
5. Input Source Program
Compilation Models

Aho Ullman Model

Front End
→ AST
→ Optimizer
→ Target Indep. IR
→ Code Generator
→ Target Program

Davidson Fraser Model

Input Source Program → Front End
→ AST
→ Expander
→ Register Transfers
→ Optimizer
→ Register Transfers
Compilation Models

Aho Ullman Model

Front End
→ AST
→ Optimizer
→ Target Indep. IR
→ Code Generator
→ Target Program

Aho Ullman: Instruction selection
- over optimized IR using
cost based tree tiling matching

Davidson Fraser Model

Front End
→ AST
→ Expander
→ Register Transfers
→ Optimizer
→ Register Transfers
→ Recognizer
→ Target Program

Davidson Fraser: Instruction selection
- over AST using
simple full tree matching based algorithms that generate
- naive code which is
target dependent, and is
optimized subsequently

Uday Khedker
GRC, IIT Bombay
Typical Front Ends

Parser
Typical Front Ends

Source Program

Parser

Scanner

Tokens
Typical Front Ends

Source Program → Scanner → Tokens → Parser → AST or Linear IR + Symbol Table → AST → Parse Tree → Semantic Analyzer
Typical Front Ends

- Source Program
- Scanner
- Tokens
- Parse Tree
- AST
- Semantic Analyzer
- AST or Linear IR + Symbol Table
- Symtab Handler
- Error Handler
Typical Back Ends in Aho Ullman Model

- Compile time evaluations
- Eliminating redundant computations
Typical Back Ends in Aho Ullman Model

- Compile time evaluations
- Eliminating redundant computations
- Instruction Selection
- Local Reg Allocation
- Choice of Order of Evaluation
Typical Back Ends in Aho Ullman Model

- Compile time evaluations
- Eliminating redundant computations
- Instruction Selection
- Local Reg Allocation
- Choice of Order of Evaluation

Uday Khedker
GRC, IIT Bombay
Typical Back Ends in Aho Ullman Model

- Compile time evaluations
- Eliminating redundant computations

- Instruction Selection
- Local Reg Allocation
- Choice of Order of Evaluation

Register Allocator

Instruction Scheduler

Peephole Optimizer

Assembly Code
<table>
<thead>
<tr>
<th>Instruction Selection</th>
<th>Aho Ullman Model</th>
<th>Davidson Fraser Model</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Machine independent IR is expressed in the form of trees</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Machine instructions are described in the form of trees</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Trees in the IR are “covered” using the instruction trees</td>
<td></td>
</tr>
<tr>
<td>Optimization</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Uday Khedker  
GRC, IIT Bombay
Retargetability in Aho Ullman and Davidson Fraser Models

<table>
<thead>
<tr>
<th></th>
<th>Aho Ullman Model</th>
<th>Davidson Fraser Model</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Instruction Selection</strong></td>
<td>• Machine independent IR is expressed in the form of trees</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Machine instructions are described in the form of trees</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Trees in the IR are “covered” using the instruction trees</td>
<td></td>
</tr>
<tr>
<td><strong>Optimization</strong></td>
<td>Cost based tree pattern matching</td>
<td></td>
</tr>
</tbody>
</table>
## Retargetability in Aho Ullman and Davidson Fraser Models

<table>
<thead>
<tr>
<th></th>
<th>Aho Ullman Model</th>
<th>Davidson Fraser Model</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Instruction Selection</strong></td>
<td>• Machine independent IR is expressed in the form of trees</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Machine instructions are described in the form of trees</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Trees in the IR are “covered” using the instruction trees</td>
<td></td>
</tr>
<tr>
<td><strong>Optimization</strong></td>
<td>Cost based tree pattern matching</td>
<td>Structural tree pattern matching</td>
</tr>
</tbody>
</table>
## Retargetability in Aho Ullman and Davidson Fraser Models

<table>
<thead>
<tr>
<th></th>
<th>Aho Ullman Model</th>
<th>Davidson Fraser Model</th>
</tr>
</thead>
</table>
| **Instruction Selection** | • Machine independent IR is expressed in the form of trees  
• Machine instructions are described in the form of trees  
• Trees in the IR are “covered” using the instruction trees | Cost based tree pattern matching  
Structural tree pattern matching |
| **Optimization**     | Machine independent                                                               |                                                       |
# Retargetability in Aho Ullman and Davidson Fraser Models

<table>
<thead>
<tr>
<th></th>
<th>Aho Ullman Model</th>
<th>Davidson Fraser Model</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Instruction Selection</strong></td>
<td>- Machine independent IR is expressed in the form of trees &lt;br&gt;- Machine instructions are described in the form of trees &lt;br&gt;- Trees in the IR are “covered” using the instruction trees</td>
<td>Cost based tree pattern matching &lt;br&gt;Structural tree pattern matching</td>
</tr>
<tr>
<td><strong>Optimization</strong></td>
<td>Machine independent</td>
<td>Machine dependent</td>
</tr>
</tbody>
</table>
### Retargetability in Aho Ullman and Davidson Fraser Models

<table>
<thead>
<tr>
<th></th>
<th>Aho Ullman Model</th>
<th>Davidson Fraser Model</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Instruction Selection</strong></td>
<td>• Machine independent IR is expressed in the form of trees</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Machine instructions are described in the form of trees</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Trees in the IR are “covered” using the instruction trees</td>
<td></td>
</tr>
<tr>
<td><strong>Optimization</strong></td>
<td>Machine independent</td>
<td>Machine dependent</td>
</tr>
<tr>
<td></td>
<td>Cost based tree pattern matching</td>
<td>Structural tree pattern matching</td>
</tr>
<tr>
<td></td>
<td>Key Insight: <strong>Register transfers are target specific but their form is target independent</strong></td>
<td></td>
</tr>
</tbody>
</table>
Part 2

GCC ≡ The Great Compiler Challenge
What is GCC?

- For the GCC developer community: The GNU Compiler Collection
- For other compiler writers: The Great Compiler Challenge 😊
The GNU Tool Chain for C

Source Program

\[ \text{gcc} \]

Target Program
The GNU Tool Chain for C

Source Program

\[ \text{gcc} \]

Target Program

\[ \text{cc1} \]
The GNU Tool Chain for C
The GNU Tool Chain for C

Source Program

\[ \text{ggc} \]

Target Program

\[ \text{cc1} \rightarrow \text{cpp} \]

\[ \text{gcc} \leftrightarrow \text{as} \]
The GNU Tool Chain for C

Source Program

\[ \text{gcc} \]

Target Program

\[ \text{cc1} \rightarrow \text{cpp} \]

\[ \text{as} \]

\[ \text{ld} \]
The GNU Tool Chain for C

Source Program

gcc

Target Program

cc1

cpp

as

ld

glibc/newlib
The GNU Tool Chain for C

Source Program

gcc

cc1

cpp

as

ld

glibc/newlib

Target Program
Why is Understanding GCC Difficult?

Some of the obvious reasons:

- **Comprehensiveness**

  GCC is a production quality framework in terms of completeness and practical usefulness

- **Open development model**

  Could lead to heterogeneity. Design flaws may be difficult to correct

- **Rapid versioning**

  GCC maintenance is a race against time. Disruptive corrections are difficult
Comprehensiveness of GCC: Wide Applicability

- **Input languages supported:**
  - C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada

- **Processors supported in standard releases:**
Comprehensiveness of GCC: Wide Applicability

- **Input languages supported:**
  C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada

- **Processors supported in standard releases:**
  - **Common processors:**
  - **Lesser-known target processors:**
  - **Additional processors independently supported:**
Comprehensiveness of GCC: Wide Applicability

- **Input languages supported:**
  C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada

- **Processors supported in standard releases:**
  - **Common processors:**
    Alpha,

  - **Lesser-known target processors:**

  - **Additional processors independently supported:**
Comprehensiveness of GCC: Wide Applicability

- **Input languages supported:**
  - C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada

- **Processors supported in standard releases:**
  - **Common processors:**
    - Alpha, ARM,
  - **Lesser-known target processors:**
  - **Additional processors independently supported:**
Comprehensiveness of GCC: Wide Applicability

- **Input languages supported:**
  - C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada

- **Processors supported in standard releases:**
  - **Common processors:**
    - Alpha, ARM, Atmel AVR,
  - **Lesser-known target processors:**
  - **Additional processors independently supported:**
Comprehensiveness of GCC: Wide Applicability

- **Input languages supported:**
  - C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada

- **Processors supported in standard releases:**
  - **Common processors:**
    - Alpha, ARM, Atmel AVR, Blackfin,
  - **Lesser-known target processors:**
  - **Additional processors independently supported:**
Comprehensiveness of GCC: Wide Applicability

- **Input languages supported:**
  C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada

- **Processors supported in standard releases:**
  - **Common processors:**
    - Alpha, ARM, Atmel AVR, Blackfin, HC12,
  
  - **Lesser-known target processors:**

  - **Additional processors independently supported:**
Comprehensiveness of GCC: Wide Applicability

• Input languages supported:
  C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada

• Processors supported in standard releases:

  ▶ Common processors:
  Alpha, ARM, Atmel AVR, Blackfin, HC12, H8/300,

  ▶ Lesser-known target processors:

  ▶ Additional processors independently supported:
Comprehensiveness of GCC: Wide Applicability

- **Input languages supported:**
  - C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada

- **Processors supported in standard releases:**
  - **Common processors:**
    - Alpha, ARM, Atmel AVR, Blackfin, HC12, H8/300, IA-32 (x86),
  - **Lesser-known target processors:**
  - **Additional processors independently supported:**
Comprehensiveness of GCC: Wide Applicability

• **Input languages supported:**
  C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada

• **Processors supported in standard releases:**
  - **Common processors:**
    Alpha, ARM, Atmel AVR, Blackfin, HC12, H8/300, IA-32 (x86), x86-64,

  - **Lesser-known target processors:**

  - **Additional processors independently supported:**
Comprehensiveness of GCC: Wide Applicability

- **Input languages supported:**
  - C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada

- **Processors supported in standard releases:**
  - **Common processors:**
    - Alpha, ARM, Atmel AVR, Blackfin, HC12, H8/300, IA-32 (x86), x86-64, IA-64,
  
  - **Lesser-known target processors:**

- **Additional processors independently supported:**
Comprehensiveness of GCC: Wide Applicability

• Input languages supported:
  C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada

• Processors supported in standard releases:
  ▶ Common processors:
    Alpha, ARM, Atmel AVR, Blackfin, HC12, H8/300, IA-32 (x86), x86-64, IA-64, Motorola 68000,

  ▶ Lesser-known target processors:

  ▶ Additional processors independently supported:
Comprehensiveness of GCC: Wide Applicability

• Input languages supported:
  C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada

• Processors supported in standard releases:
  ▶ Common processors:
    Alpha, ARM, Atmel AVR, Blackfin, HC12, H8/300, IA-32 (x86), x86-64, IA-64, Motorola 68000, MIPS,
  ▶ Lesser-known target processors:
  ▶ Additional processors independently supported:
Comprehensiveness of GCC: Wide Applicability

- **Input languages supported:**
  C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada

- **Processors supported in standard releases:**
  - **Common processors:**
    Alpha, ARM, Atmel AVR, Blackfin, HC12, H8/300, IA-32 (x86), x86-64, IA-64, Motorola 68000, MIPS, PA-RISC,
  - **Lesser-known target processors:**
  - **Additional processors independently supported:**
Comprehensiveness of GCC: Wide Applicability

• Input languages supported:
  C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada

• Processors supported in standard releases:
  ▶ Common processors:
    Alpha, ARM, Atmel AVR, Blackfin, HC12, H8/300, IA-32 (x86), x86-64, IA-64, Motorola 68000, MIPS, PA-RISC, PDP-11,
  ▶ Lesser-known target processors:

  ▶ Additional processors independently supported:
Comprehensiveness of GCC: Wide Applicability

- **Input languages supported:**
  C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada

- **Processors supported in standard releases:**
  - **Common processors:**
    Alpha, ARM, Atmel AVR, Blackfin, HC12, H8/300, IA-32 (x86), x86-64, IA-64, Motorola 68000, MIPS, PA-RISC, PDP-11, PowerPC,
  - **Lesser-known target processors:**
  - **Additional processors independently supported:**
Comprehensiveness of GCC: Wide Applicability

- **Input languages supported:**
  C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada

- **Processors supported in standard releases:**
  - **Common processors:**
    Alpha, ARM, Atmel AVR, Blackfin, HC12, H8/300, IA-32 (x86), x86-64, IA-64, Motorola 68000, MIPS, PA-RISC, PDP-11, PowerPC, R8C/M16C/M32C,
  - **Lesser-known target processors:**

- **Additional processors independently supported:**
Comprehensiveness of GCC: Wide Applicability

- **Input languages supported:**
  C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada

- **Processors supported in standard releases:**
  - **Common processors:**
    Alpha, ARM, Atmel AVR, Blackfin, HC12, H8/300, IA-32 (x86), x86-64, IA-64, Motorola 68000, MIPS, PA-RISC, PDP-11, PowerPC, R8C/M16C/M32C, SPU,

  - **Lesser-known target processors:**

  - **Additional processors independently supported:**
Comprehensiveness of GCC: Wide Applicability

- **Input languages supported:**
  - C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada

- **Processors supported in standard releases:**
  - **Common processors:**
    - Alpha, ARM, Atmel AVR, Blackfin, HC12, H8/300, IA-32 (x86), x86-64, IA-64, Motorola 68000, MIPS, PA-RISC, PDP-11, PowerPC, R8C/M16C/M32C, SPU, System/390/zSeries,
  - **Lesser-known target processors:**

- **Additional processors independently supported:**
Comprehensiveness of GCC: Wide Applicability

- **Input languages supported:**
  C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada

- **Processors supported in standard releases:**
  - **Common processors:**
    Alpha, ARM, Atmel AVR, Blackfin, HC12, H8/300, IA-32 (x86), x86-64, IA-64, Motorola 68000, MIPS, PA-RISC, PDP-11, PowerPC, R8C/M16C/M32C, SPU, System/390/zSeries, SuperH,
  - **Lesser-known target processors:**

- **Additional processors independently supported:**
Comprehensiveness of GCC: Wide Applicability

- **Input languages supported:**
  C, C++, Objective-C, Objective-C+++, Java, Fortran, and Ada

- **Processors supported in standard releases:**
  - **Common processors:**
    Alpha, ARM, Atmel AVR, Blackfin, HC12, H8/300, IA-32 (x86), x86-64, IA-64, Motorola 68000, MIPS, PA-RISC, PDP-11, PowerPC, R8C/M16C/M32C, SPU, System/390/zSeries, SuperH, SPARC,
  - **Lesser-known target processors:**

- **Additional processors independently supported:**
Comprehensiveness of GCC: Wide Applicability

- **Input languages supported:**
  C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada

- **Processors supported in standard releases:**
  - **Common processors:**
    Alpha, ARM, Atmel AVR, Blackfin, HC12, H8/300, IA-32 (x86), x86-64, IA-64, Motorola 68000, MIPS, PA-RISC, PDP-11, PowerPC, R8C/M16C/M32C, SPU, System/390/zSeries, SuperH, SPARC, VAX
  - **Lesser-known target processors:**

- **Additional processors independently supported:**
Comprehensiveness of GCC: Wide Applicability

- **Input languages supported:**
  
  C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada

- **Processors supported in standard releases:**
  
  - **Common processors:**
    
    Alpha, ARM, Atmel AVR, Blackfin, HC12, H8/300, IA-32 (x86), x86-64, IA-64, Motorola 68000, MIPS, PA-RISC, PDP-11, PowerPC, R8C/M16C/M32C, SPU, System/390/zSeries, SuperH, SPARC, VAX

  - **Lesser-known target processors:**
    
    A29K,

- **Additional processors independently supported:**
Comprehensiveness of GCC: Wide Applicability

• Input languages supported:
  C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada

• Processors supported in standard releases:
  ▶ Common processors:
    Alpha, ARM, Atmel AVR, Blackfin, HC12, H8/300, IA-32 (x86), x86-64, IA-64, Motorola 68000, MIPS, PA-RISC, PDP-11, PowerPC, R8C/M16C/M32C, SPU, System/390/zSeries, SuperH, SPARC, VAX
  ▶ Lesser-known target processors:
    A29K, ARC,

  ▶ Additional processors independently supported:
Comprehensiveness of GCC: Wide Applicability

- **Input languages supported:**
  C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada

- **Processors supported in standard releases:**
  - **Common processors:**
    Alpha, ARM, Atmel AVR, Blackfin, HC12, H8/300, IA-32 (x86), x86-64, IA-64, Motorola 68000, MIPS, PA-RISC, PDP-11, PowerPC, R8C/M16C/M32C, SPU, System/390/zSeries, SuperH, SPARC, VAX
  - **Lesser-known target processors:**
    A29K, ARC, ETRAX CRIS,

- Additional processors independently supported:
Comprehensiveness of GCC: Wide Applicability

- **Input languages supported:**
  C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada

- **Processors supported in standard releases:**
  - **Common processors:**
    Alpha, ARM, Atmel AVR, Blackfin, HC12, H8/300, IA-32 (x86), x86-64, IA-64, Motorola 68000, MIPS, PA-RISC, PDP-11, PowerPC, R8C/M16C/M32C, SPU, System/390/zSeries, SuperH, SPARC, VAX
  - **Lesser-known target processors:**
    A29K, ARC, ETRAX CRIS, D30V,

- **Additional processors independently supported:**
Comprehensiveness of GCC: Wide Applicability

- **Input languages supported:**
  
  C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada

- **Processors supported in standard releases:**
  
  - **Common processors:**
    
    Alpha, ARM, Atmel AVR, Blackfin, HC12, H8/300, IA-32 (x86), x86-64, IA-64, Motorola 68000, MIPS, PA-RISC, PDP-11, PowerPC, R8C/M16C/M32C, SPU, System/390/zSeries, SuperH, SPARC, VAX
  
  - **Lesser-known target processors:**
    
    A29K, ARC, ETRAX CRIS, D30V, DSP16xx,

  - **Additional processors independently supported:**

Comprehensiveness of GCC: Wide Applicability

- **Input languages supported:**
  C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada

- **Processors supported in standard releases:**
  - **Common processors:**
    Alpha, ARM, Atmel AVR, Blackfin, HC12, H8/300, IA-32 (x86), x86-64, IA-64, Motorola 68000, MIPS, PA-RISC, PDP-11, PowerPC, R8C/M16C/M32C, SPU, System/390/zSeries, SuperH, SPARC, VAX
  - **Lesser-known target processors:**
    A29K, ARC, ETRAX CRIS, D30V, DSP16xx, FR-30,

- **Additional processors independently supported:**
Comprehensiveness of GCC: Wide Applicability

- **Input languages supported:**
  C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada

- **Processors supported in standard releases:**
  - **Common processors:**
    Alpha, ARM, Atmel AVR, Blackfin, HC12, H8/300, IA-32 (x86), x86-64, IA-64, Motorola 68000, MIPS, PA-RISC, PDP-11, PowerPC, R8C/M16C/M32C, SPU, System/390/zSeries, SuperH, SPARC, VAX
  - **Lesser-known target processors:**
    A29K, ARC, ETRAX CRIS, D30V, DSP16xx, FR-30, FR-V,

  - **Additional processors independently supported:**
Comprehensiveness of GCC: Wide Applicability

- **Input languages supported:**
  C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada

- **Processors supported in standard releases:**
  - **Common processors:**
    Alpha, ARM, Atmel AVR, Blackfin, HC12, H8/300, IA-32 (x86), x86-64, IA-64, Motorola 68000, MIPS, PA-RISC, PDP-11, PowerPC, R8C/M16C/M32C, SPU, System/390/zSeries, SuperH, SPARC, VAX
  - **Lesser-known target processors:**
    A29K, ARC, ETRAX CRIS, D30V, DSP16xx, FR-30, FR-V, Intel i960,

  - **Additional processors independently supported:**
Comprehensiveness of GCC: Wide Applicability

- **Input languages supported:**
  
  C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada

- **Processors supported in standard releases:**
  
  - **Common processors:**
    Alpha, ARM, Atmel AVR, Blackfin, HC12, H8/300, IA-32 (x86), x86-64, IA-64, Motorola 68000, MIPS, PA-RISC, PDP-11, PowerPC, R8C/M16C/M32C, SPU, System/390/zSeries, SuperH, SPARC, VAX
  
  - **Lesser-known target processors:**
    A29K, ARC, ETRAX CRIS, D30V, DSP16xx, FR-30, FR-V, Intel i960, IP2000,

  - **Additional processors independently supported:**
Comprehensiveness of GCC: Wide Applicability

- **Input languages supported:**
  C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada

- **Processors supported in standard releases:**
  - **Common processors:**
    Alpha, ARM, Atmel AVR, Blackfin, HC12, H8/300, IA-32 (x86), x86-64, IA-64, Motorola 68000, MIPS, PA-RISC, PDP-11, PowerPC, R8C/M16C/M32C, SPU, System/390/zSeries, SuperH, SPARC, VAX
  - **Lesser-known target processors:**
    A29K, ARC, ETRAX CRIS, D30V, DSP16xx, FR-30, FR-V, Intel i960, IP2000, M32R,

  - **Additional processors independently supported:**
Comprehensiveness of GCC: Wide Applicability

- Input languages supported:
  - C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada

- Processors supported in standard releases:
  - Common processors:
    - Alpha, ARM, Atmel AVR, Blackfin, HC12, H8/300, IA-32 (x86), x86-64, IA-64, Motorola 68000, MIPS, PA-RISC, PDP-11, PowerPC, R8C/M16C/M32C, SPU, System/390/zSeries, SuperH, SPARC, VAX
  - Lesser-known target processors:

- Additional processors independently supported:
Comprehensiveness of GCC: Wide Applicability

- **Input languages supported:**
  C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada

- **Processors supported in standard releases:**
  - **Common processors:**
    Alpha, ARM, Atmel AVR, Blackfin, HC12, H8/300, IA-32 (x86), x86-64, IA-64, Motorola 68000, MIPS, PA-RISC, PDP-11, PowerPC, R8C/M16C/M32C, SPU, System/390/zSeries, SuperH, SPARC, VAX
  - **Lesser-known target processors:**
    A29K, ARC, ETRAX CRIS, D30V, DSP16xx, FR-30, FR-V, Intel i960, IP2000, M32R, 68HC11, MCORE,

- **Additional processors independently supported:**
Comprehensiveness of GCC: Wide Applicability

• **Input languages supported:**
  C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada

• **Processors supported in standard releases:**
  - **Common processors:**
    Alpha, ARM, Atmel AVR, Blackfin, HC12, H8/300, IA-32 (x86), x86-64, IA-64, Motorola 68000, MIPS, PA-RISC, PDP-11, PowerPC, R8C/M16C/M32C, SPU, System/390/zSeries, SuperH, SPARC, VAX
  - **Lesser-known target processors:**
    A29K, ARC, ETRAX CRIS, D30V, DSP16xx, FR-30, FR-V, Intel i960, IP2000, M32R, 68HC11, MCORE, MMIX,

  - **Additional processors independently supported:**
Comprehensiveness of GCC: Wide Applicability

- **Input languages supported:**
  C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada

- **Processors supported in standard releases:**
  - **Common processors:**
    Alpha, ARM, Atmel AVR, Blackfin, HC12, H8/300, IA-32 (x86), x86-64, IA-64, Motorola 68000, MIPS, PA-RISC, PDP-11, PowerPC, R8C/M16C/M32C, SPU, System/390/zSeries, SuperH, SPARC, VAX
  - **Lesser-known target processors:**
    A29K, ARC, ETRAX CRIS, D30V, DSP16xx, FR-30, FR-V, Intel i960, IP2000, M32R, 68HC11, MCORE, MMIX, MN10200,

  - **Additional processors independently supported:**
Comprehensiveness of GCC: Wide Applicability

- **Input languages supported:**
  
  C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada

- **Processors supported in standard releases:**
  
  - **Common processors:**
    
    Alpha, ARM, Atmel AVR, Blackfin, HC12, H8/300, IA-32 (x86), x86-64, IA-64, Motorola 68000, MIPS, PA-RISC, PDP-11, PowerPC, R8C/M16C/M32C, SPU, System/390/zSeries, SuperH, SPARC, VAX

  - **Lesser-known target processors:**
    
    A29K, ARC, ETRAX CRIS, D30V, DSP16xx, FR-30, FR-V, Intel i960, IP2000, M32R, 68HC11, MCORE, MMIX, MN10200, MN10300,

  - **Additional processors independently supported:**
Comprehensiveness of GCC: Wide Applicability

- **Input languages supported:**
  C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada

- **Processors supported in standard releases:**
  - **Common processors:**
    Alpha, ARM, Atmel AVR, Blackfin, HC12, H8/300, IA-32 (x86), x86-64, IA-64, Motorola 68000, MIPS, PA-RISC, PDP-11, PowerPC, R8C/M16C/M32C, SPU, System/390/zSeries, SuperH, SPARC, VAX
  - **Lesser-known target processors:**
    A29K, ARC, ETRAX CRIS, D30V, DSP16xx, FR-30, FR-V, Intel i960, IP2000, M32R, 68HC11, MCORE, MMIX, MN10200, MN10300, Motorola 88000,

- **Additional processors independently supported:**
Comprehensiveness of GCC: Wide Applicability

• **Input languages supported:**
  C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada

• **Processors supported in standard releases:**
  - **Common processors:**
    Alpha, ARM, Atmel AVR, Blackfin, HC12, H8/300, IA-32 (x86), x86-64, IA-64, Motorola 68000, MIPS, PA-RISC, PDP-11, PowerPC, R8C/M16C/M32C, SPU, System/390/zSeries, SuperH, SPARC, VAX
  - **Lesser-known target processors:**
    A29K, ARC, ETRAX CRIS, D30V, DSP16xx, FR-30, FR-V, Intel i960, IP2000, M32R, 68HC11, MCORE, MMIX, MN10200, MN10300, Motorola 88000, NS32K,

  - **Additional processors independently supported:**
Comprehensiveness of GCC: Wide Applicability

- **Input languages supported:**
  C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada

- **Processors supported in standard releases:**
  
  - **Common processors:**
    Alpha, ARM, Atmel AVR, Blackfin, HC12, H8/300, IA-32 (x86), x86-64, IA-64, Motorola 68000, MIPS, PA-RISC, PDP-11, PowerPC, R8C/M16C/M32C, SPU, System/390/zSeries, SuperH, SPARC, VAX
  
  - **Lesser-known target processors:**
    A29K, ARC, ETRAX CRIS, D30V, DSP16xx, FR-30, FR-V, Intel i960, IP2000, M32R, 68HC11, MCORE, MMIX, MN10200, MN10300, Motorola 88000, NS32K, ROMP,

  - **Additional processors independently supported:**
Comprehensiveness of GCC: Wide Applicability

- **Input languages supported:**
  C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada

- **Processors supported in standard releases:**
  - **Common processors:**
    Alpha, ARM, Atmel AVR, Blackfin, HC12, H8/300, IA-32 (x86), x86-64, IA-64, Motorola 68000, MIPS, PA-RISC, PDP-11, PowerPC, R8C/M16C/M32C, SPU, System/390/zSeries, SuperH, SPARC, VAX
  - **Lesser-known target processors:**
    A29K, ARC, ETRAX CRIS, D30V, DSP16xx, FR-30, FR-V, Intel i960, IP2000, M32R, 68HC11, MCORE, MMIX, MN10200, MN10300, Motorola 88000, NS32K, ROMP, Stormy16,
  - **Additional processors independently supported:**
Comprehensiveness of GCC: Wide Applicability

- **Input languages supported:**
  C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada

- **Processors supported in standard releases:**
  - **Common processors:**
    Alpha, ARM, Atmel AVR, Blackfin, HC12, H8/300, IA-32 (x86), x86-64, IA-64, Motorola 68000, MIPS, PA-RISC, PDP-11, PowerPC, R8C/M16C/M32C, SPU, System/390/zSeries, SuperH, SPARC, VAX
  - **Lesser-known target processors:**
  - **Additional processors independently supported:**
Comprehensiveness of GCC: Wide Applicability

- **Input languages supported:**
  - C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada

- **Processors supported in standard releases:**
  - **Common processors:**
    - Alpha, ARM, Atmel AVR, Blackfin, HC12, H8/300, IA-32 (x86), x86-64, IA-64, Motorola 68000, MIPS, PA-RISC, PDP-11, PowerPC, R8C/M16C/M32C, SPU, System/390/zSeries, SuperH, SPARC, VAX
  - **Lesser-known target processors:**
  - **Additional processors independently supported:**
Comprehensiveness of GCC: Wide Applicability

- **Input languages supported:**
  C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada

- **Processors supported in standard releases:**
  - **Common processors:**
    Alpha, ARM, Atmel AVR, Blackfin, HC12, H8/300, IA-32 (x86), x86-64, IA-64, Motorola 68000, MIPS, PA-RISC, PDP-11, PowerPC, R8C/M16C/M32C, SPU, System/390/zSeries, SuperH, SPARC, VAX
  - **Lesser-known target processors:**
  - **Additional processors independently supported:**
Comprehensiveness of GCC: Wide Applicability

- **Input languages supported:**
  - C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada
- **Processors supported in standard releases:**
  - **Common processors:**
    - Alpha, ARM, Atmel AVR, Blackfin, HC12, H8/300, IA-32 (x86), x86-64, IA-64, Motorola 68000, MIPS, PA-RISC, PDP-11, PowerPC, R8C/M16C/M32C, SPU, System/390/zSeries, SuperH, SPARC, VAX
  - **Lesser-known target processors:**
  - **Additional processors independently supported:**
    - D10V,
Comprehensiveness of GCC: Wide Applicability

- **Input languages supported:**
  - C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada

- **Processors supported in standard releases:**
  - **Common processors:**
    - Alpha, ARM, Atmel AVR, Blackfin, HC12, H8/300, IA-32 (x86), x86-64, IA-64, Motorola 68000, MIPS, PA-RISC, PDP-11, PowerPC, R8C/M16C/M32C, SPU, System/390/zSeries, SuperH, SPARC, VAX
  - **Lesser-known target processors:**
  - **Additional processors independently supported:**
    - D10V, LatticeMico32, MeP,
Comprehensiveness of GCC: Wide Applicability

• Input languages supported:
  C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada

• Processors supported in standard releases:
  ▶ Common processors:
    Alpha, ARM, Atmel AVR, Blackfin, HC12, H8/300, IA-32 (x86), x86-64, IA-64, Motorola 68000, MIPS, PA-RISC, PDP-11, PowerPC, R8C/M16C/M32C, SPU, System/390/zSeries, SuperH, SPARC, VAX
  ▶ Lesser-known target processors:
  ▶ Additional processors independently supported:
    D10V, LatticeMico32, MeP,
Comprehensiveness of GCC: Wide Applicability

- **Input languages supported:**
  C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada
- **Processors supported in standard releases:**
  - **Common processors:**
    Alpha, ARM, Atmel AVR, Blackfin, HC12, H8/300, IA-32 (x86), x86-64, IA-64, Motorola 68000, MIPS, PA-RISC, PDP-11, PowerPC, R8C/M16C/M32C, SPU, System/390/zSeries, SuperH, SPARC, VAX
  - **Lesser-known target processors:**
  - **Additional processors independently supported:**
    D10V, LatticeMico32, MeP, Motorola 6809,
Comprehensiveness of GCC: Wide Applicability

- **Input languages supported:**
  C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada

- **Processors supported in standard releases:**
  - **Common processors:**
    Alpha, ARM, Atmel AVR, Blackfin, HC12, H8/300, IA-32 (x86), x86-64, IA-64, Motorola 68000, MIPS, PA-RISC, PDP-11, PowerPC, R8C/M16C/M32C, SPU, System/390/zSeries, SuperH, SPARC, VAX
  - **Lesser-known target processors:**
  - **Additional processors independently supported:**
    D10V, LatticeMico32, MeP, Motorola 6809, MicroBlaze,
Comprehensiveness of GCC: Wide Applicability

• Input languages supported:
  C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada

• Processors supported in standard releases:
  ▶ Common processors:
    Alpha, ARM, Atmel AVR, Blackfin, HC12, H8/300, IA-32 (x86), x86-64, IA-64, Motorola 68000, MIPS, PA-RISC, PDP-11, PowerPC, R8C/M16C/M32C, SPU, System/390/zSeries, SuperH, SPARC, VAX
  ▶ Lesser-known target processors:
  ▶ Additional processors independently supported:
    D10V, LatticeMico32, MeP, Motorola 6809, MicroBlaze, MSP430,
Comprehensiveness of GCC: Wide Applicability

• Input languages supported:
  C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada

• Processors supported in standard releases:
  ▶ Common processors:
    Alpha, ARM, Atmel AVR, Blackfin, HC12, H8/300, IA-32 (x86), x86-64, IA-64, Motorola 68000, MIPS, PA-RISC, PDP-11, PowerPC, R8C/M16C/M32C, SPU, System/390/zSeries, SuperH, SPARC, VAX
  ▶ Lesser-known target processors:
  ▶ Additional processors independently supported:
    D10V, LatticeMico32, MeP, Motorola 6809, MicroBlaze, MSP430, Nios II and Nios,
Comprehensiveness of GCC: Wide Applicability

- **Input languages supported:**
  C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada

- **Processors supported in standard releases:**
  
  - **Common processors:**
    Alpha, ARM, Atmel AVR, Blackfin, HC12, H8/300, IA-32 (x86), x86-64, IA-64, Motorola 68000, MIPS, PA-RISC, PDP-11, PowerPC, R8C/M16C/M32C, SPU, System/390/zSeries, SuperH, SPARC, VAX

  - **Lesser-known target processors:**

  - **Additional processors independently supported:**
    D10V, LatticeMico32, MeP, Motorola 6809, MicroBlaze, MSP430, Nios II and Nios, PDP-10,
Comprehensiveness of GCC: Wide Applicability

- **Input languages supported:**
  C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada

- **Processors supported in standard releases:**
  - **Common processors:**
    Alpha, ARM, Atmel AVR, Blackfin, HC12, H8/300, IA-32 (x86), x86-64, IA-64, Motorola 68000, MIPS, PA-RISC, PDP-11, PowerPC, R8C/M16C/M32C, SPU, System/390/zSeries, SuperH, SPARC, VAX
  - **Lesser-known target processors:**
  - **Additional processors independently supported:**
    D10V, LatticeMico32, MeP, Motorola 6809, MicroBlaze, MSP430, Nios II and Nios, PDP-10, TIGCC (m68k variant),
Comprehensiveness of GCC: Wide Applicability

- **Input languages supported:**
  C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada

- **Processors supported in standard releases:**
  - **Common processors:**
    Alpha, ARM, Atmel AVR, Blackfin, HC12, H8/300, IA-32 (x86), x86-64, IA-64, Motorola 68000, MIPS, PA-RISC, PDP-11, PowerPC, R8C/M16C/M32C, SPU, System/390/zSeries, SuperH, SPARC, VAX
  - **Lesser-known target processors:**
  - **Additional processors independently supported:**
    D10V, LatticeMico32, MeP, Motorola 6809, MicroBlaze, MSP430, Nios II and Nios, PDP-10, TIGCC (m68k variant), Z8000,
Comprehensiveness of GCC: Wide Applicability

- **Input languages supported:**
  
  C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada

- **Processors supported in standard releases:**
  
  - **Common processors:**
    
    Alpha, ARM, Atmel AVR, Blackfin, HC12, H8/300, IA-32 (x86), x86-64, IA-64, Motorola 68000, MIPS, PA-RISC, PDP-11, PowerPC, R8C/M16C/M32C, SPU, System/390/zSeries, SuperH, SPARC, VAX

  - **Lesser-known target processors:**
    

  - **Additional processors independently supported:**
    
    D10V, LatticeMico32, MeP, Motorola 6809, MicroBlaze, MSP430, Nios II and Nios, PDP-10, TIGCC (m68k variant), Z8000, PIC24/dsPIC,
Comprehensiveness of GCC: Wide Applicability

- **Input languages supported:**
  C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada

- **Processors supported in standard releases:**
  - **Common processors:**
    Alpha, ARM, Atmel AVR, Blackfin, HC12, H8/300, IA-32 (x86), x86-64, IA-64, Motorola 68000, MIPS, PA-RISC, PDP-11, PowerPC, R8C/M16C/M32C, SPU, System/390/zSeries, SuperH, SPARC, VAX
  - **Lesser-known target processors:**
  - **Additional processors independently supported:**
    D10V, LatticeMico32, MeP, Motorola 6809, MicroBlaze, MSP430, Nios II and Nios, PDP-10, TIGCC (m68k variant), Z8000, PIC24/dsPIC, NEC SX architecture
## Comprehensiveness of GCC: Size

- **Overall size**

<table>
<thead>
<tr>
<th></th>
<th>Subdirectories</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>gcc-4.4.2</td>
<td>3794</td>
<td>62301</td>
</tr>
<tr>
<td>gcc-4.6.0</td>
<td>4383</td>
<td>71096</td>
</tr>
<tr>
<td>gcc-4.7.2</td>
<td>4658</td>
<td>76287</td>
</tr>
</tbody>
</table>

- **Core size (src/gcc)**

<table>
<thead>
<tr>
<th></th>
<th>Subdirectories</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>gcc-4.4.2</td>
<td>257</td>
<td>30163</td>
</tr>
<tr>
<td>gcc-4.6.0</td>
<td>336</td>
<td>36503</td>
</tr>
<tr>
<td>gcc-4.7.2</td>
<td>402</td>
<td>40193</td>
</tr>
</tbody>
</table>

- **Machine Descriptions (src/gcc/config)**

<table>
<thead>
<tr>
<th></th>
<th>Subdirectories</th>
<th>.c files</th>
<th>.h files</th>
<th>.md files</th>
</tr>
</thead>
<tbody>
<tr>
<td>gcc-4.4.2</td>
<td>36</td>
<td>241</td>
<td>426</td>
<td>206</td>
</tr>
<tr>
<td>gcc-4.6.0</td>
<td>42</td>
<td>275</td>
<td>466</td>
<td>259</td>
</tr>
<tr>
<td>gcc-4.7.2</td>
<td>43</td>
<td>103</td>
<td>452</td>
<td>290</td>
</tr>
</tbody>
</table>
### ohcount: Line Count of gcc-4.7.2

<table>
<thead>
<tr>
<th>Language</th>
<th>Files</th>
<th>Code</th>
<th>Comment</th>
<th>Comment %</th>
<th>Blank</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>c</td>
<td>20857</td>
<td>2289353</td>
<td>472640</td>
<td>17.1%</td>
<td>449939</td>
<td>3211932</td>
</tr>
<tr>
<td>cpp</td>
<td>23370</td>
<td>1030227</td>
<td>243717</td>
<td>19.1%</td>
<td>224079</td>
<td>1498023</td>
</tr>
<tr>
<td>ada</td>
<td>4913</td>
<td>726638</td>
<td>334360</td>
<td>31.5%</td>
<td>252044</td>
<td>1313042</td>
</tr>
<tr>
<td>java</td>
<td>6342</td>
<td>681938</td>
<td>645506</td>
<td>48.6%</td>
<td>169046</td>
<td>1496490</td>
</tr>
<tr>
<td>autoconf</td>
<td>94</td>
<td>428267</td>
<td>523</td>
<td>0.1%</td>
<td>66647</td>
<td>495437</td>
</tr>
<tr>
<td>html</td>
<td>336</td>
<td>151194</td>
<td>5667</td>
<td>3.6%</td>
<td>33877</td>
<td>190738</td>
</tr>
<tr>
<td>fortranfixed</td>
<td>3256</td>
<td>112286</td>
<td>2010</td>
<td>1.8%</td>
<td>15599</td>
<td>129895</td>
</tr>
<tr>
<td>make</td>
<td>106</td>
<td>110762</td>
<td>3875</td>
<td>3.4%</td>
<td>13811</td>
<td>128448</td>
</tr>
<tr>
<td>xml</td>
<td>76</td>
<td>50179</td>
<td>571</td>
<td>1.1%</td>
<td>6048</td>
<td>56798</td>
</tr>
<tr>
<td>assembler</td>
<td>240</td>
<td>49903</td>
<td>10975</td>
<td>18.0%</td>
<td>8584</td>
<td>69462</td>
</tr>
<tr>
<td>shell</td>
<td>157</td>
<td>49148</td>
<td>10848</td>
<td>18.1%</td>
<td>6757</td>
<td>66753</td>
</tr>
<tr>
<td>objective_c</td>
<td>882</td>
<td>28226</td>
<td>5267</td>
<td>15.7%</td>
<td>8324</td>
<td>41817</td>
</tr>
<tr>
<td>fortranfree</td>
<td>872</td>
<td>14474</td>
<td>3445</td>
<td>19.2%</td>
<td>1817</td>
<td>19736</td>
</tr>
<tr>
<td>tex</td>
<td>2</td>
<td>11060</td>
<td>5776</td>
<td>34.3%</td>
<td>1433</td>
<td>18269</td>
</tr>
<tr>
<td>scheme</td>
<td>6</td>
<td>11023</td>
<td>1010</td>
<td>8.4%</td>
<td>1205</td>
<td>13238</td>
</tr>
<tr>
<td>automake</td>
<td>72</td>
<td>10496</td>
<td>1179</td>
<td>10.1%</td>
<td>1582</td>
<td>13257</td>
</tr>
<tr>
<td>perl</td>
<td>29</td>
<td>4551</td>
<td>1322</td>
<td>22.5%</td>
<td>854</td>
<td>6727</td>
</tr>
<tr>
<td>ocaml</td>
<td>6</td>
<td>2830</td>
<td>576</td>
<td>16.9%</td>
<td>378</td>
<td>3784</td>
</tr>
<tr>
<td>xslt</td>
<td>20</td>
<td>2805</td>
<td>436</td>
<td>13.5%</td>
<td>563</td>
<td>3804</td>
</tr>
<tr>
<td>awk</td>
<td>16</td>
<td>2103</td>
<td>556</td>
<td>20.9%</td>
<td>352</td>
<td>3011</td>
</tr>
<tr>
<td>python</td>
<td>10</td>
<td>1672</td>
<td>400</td>
<td>19.3%</td>
<td>400</td>
<td>2472</td>
</tr>
<tr>
<td>css</td>
<td>25</td>
<td>1590</td>
<td>143</td>
<td>8.3%</td>
<td>332</td>
<td>2065</td>
</tr>
<tr>
<td>pascal</td>
<td>4</td>
<td>1044</td>
<td>141</td>
<td>11.9%</td>
<td>218</td>
<td>1403</td>
</tr>
<tr>
<td>csharp</td>
<td>9</td>
<td>879</td>
<td>506</td>
<td>36.5%</td>
<td>230</td>
<td>1615</td>
</tr>
<tr>
<td>dcl</td>
<td>2</td>
<td>402</td>
<td>84</td>
<td>17.3%</td>
<td>13</td>
<td>499</td>
</tr>
<tr>
<td>tcl</td>
<td>1</td>
<td>392</td>
<td>113</td>
<td>22.4%</td>
<td>72</td>
<td>577</td>
</tr>
<tr>
<td>javascript</td>
<td>3</td>
<td>208</td>
<td>87</td>
<td>29.5%</td>
<td>33</td>
<td>328</td>
</tr>
<tr>
<td>haskell</td>
<td>49</td>
<td>153</td>
<td>0</td>
<td>0.0%</td>
<td>17</td>
<td>170</td>
</tr>
<tr>
<td>matlab</td>
<td>2</td>
<td>57</td>
<td>0</td>
<td>0.0%</td>
<td>8</td>
<td>65</td>
</tr>
<tr>
<td>Total</td>
<td>61760</td>
<td>5773867</td>
<td>1751733</td>
<td>23.3%</td>
<td>1264262</td>
<td>8789862</td>
</tr>
<tr>
<td>Language</td>
<td>Files</td>
<td>Code</td>
<td>Comment</td>
<td>Comment %</td>
<td>Blank</td>
<td>Total</td>
</tr>
<tr>
<td>---------------</td>
<td>--------</td>
<td>---------</td>
<td>---------</td>
<td>-----------</td>
<td>---------</td>
<td>---------</td>
</tr>
<tr>
<td>c</td>
<td>20857</td>
<td>2289353</td>
<td>472640</td>
<td>17.1%</td>
<td>449939</td>
<td>3211932</td>
</tr>
<tr>
<td>cpp</td>
<td>23370</td>
<td>1030227</td>
<td>243717</td>
<td>19.1%</td>
<td>224079</td>
<td>1498023</td>
</tr>
<tr>
<td>ada</td>
<td>4913</td>
<td>726638</td>
<td>334360</td>
<td>31.5%</td>
<td>252044</td>
<td>1313042</td>
</tr>
<tr>
<td>java</td>
<td>6342</td>
<td>681938</td>
<td>645506</td>
<td>48.6%</td>
<td>169046</td>
<td>1496490</td>
</tr>
<tr>
<td>autoconf</td>
<td>94</td>
<td>428267</td>
<td>523</td>
<td>0.1%</td>
<td>66647</td>
<td>495437</td>
</tr>
<tr>
<td>html</td>
<td>336</td>
<td>151194</td>
<td>5667</td>
<td>3.6%</td>
<td>33877</td>
<td>190738</td>
</tr>
<tr>
<td>fortranfixed</td>
<td>3256</td>
<td>112286</td>
<td>2010</td>
<td>1.8%</td>
<td>15599</td>
<td>129895</td>
</tr>
<tr>
<td>make</td>
<td>106</td>
<td>110762</td>
<td>3875</td>
<td>3.4%</td>
<td>13811</td>
<td>128448</td>
</tr>
<tr>
<td>xml</td>
<td>76</td>
<td>50179</td>
<td>571</td>
<td>1.1%</td>
<td>6048</td>
<td>56798</td>
</tr>
<tr>
<td>assembler</td>
<td>240</td>
<td>49903</td>
<td>10975</td>
<td>18.0%</td>
<td>8584</td>
<td>69462</td>
</tr>
<tr>
<td>shell</td>
<td>157</td>
<td>49148</td>
<td>10848</td>
<td>18.1%</td>
<td>6757</td>
<td>66753</td>
</tr>
<tr>
<td>objective_c</td>
<td>882</td>
<td>28226</td>
<td>5267</td>
<td>15.7%</td>
<td>8324</td>
<td>41817</td>
</tr>
<tr>
<td>fortranfree</td>
<td>872</td>
<td>14474</td>
<td>3445</td>
<td>19.2%</td>
<td>1817</td>
<td>19736</td>
</tr>
<tr>
<td>tex</td>
<td>2</td>
<td>11060</td>
<td>5776</td>
<td>34.3%</td>
<td>1433</td>
<td>18269</td>
</tr>
<tr>
<td>scheme</td>
<td>6</td>
<td>11023</td>
<td>1010</td>
<td>8.4%</td>
<td>1205</td>
<td>13238</td>
</tr>
<tr>
<td>automake</td>
<td>72</td>
<td>10496</td>
<td>1179</td>
<td>10.1%</td>
<td>1582</td>
<td>13257</td>
</tr>
<tr>
<td>perl</td>
<td>29</td>
<td>4551</td>
<td>1322</td>
<td>22.5%</td>
<td>854</td>
<td>6727</td>
</tr>
<tr>
<td>ocaml</td>
<td>6</td>
<td>2830</td>
<td>576</td>
<td>16.9%</td>
<td>378</td>
<td>3784</td>
</tr>
<tr>
<td>xslt</td>
<td>20</td>
<td>2805</td>
<td>436</td>
<td>13.5%</td>
<td>563</td>
<td>3804</td>
</tr>
<tr>
<td>awk</td>
<td>16</td>
<td>2103</td>
<td>556</td>
<td>20.9%</td>
<td>352</td>
<td>3011</td>
</tr>
<tr>
<td>python</td>
<td>10</td>
<td>1672</td>
<td>400</td>
<td>19.3%</td>
<td>400</td>
<td>2472</td>
</tr>
<tr>
<td>css</td>
<td>25</td>
<td>1590</td>
<td>143</td>
<td>8.3%</td>
<td>332</td>
<td>2065</td>
</tr>
<tr>
<td>pascal</td>
<td>4</td>
<td>1044</td>
<td>141</td>
<td>11.9%</td>
<td>218</td>
<td>1403</td>
</tr>
<tr>
<td>csharp</td>
<td>9</td>
<td>879</td>
<td>506</td>
<td>36.5%</td>
<td>230</td>
<td>1615</td>
</tr>
<tr>
<td>dcl</td>
<td>2</td>
<td>402</td>
<td>84</td>
<td>17.3%</td>
<td>13</td>
<td>499</td>
</tr>
<tr>
<td>tcl</td>
<td>1</td>
<td>392</td>
<td>113</td>
<td>22.4%</td>
<td>72</td>
<td>577</td>
</tr>
<tr>
<td>javascript</td>
<td>3</td>
<td>208</td>
<td>87</td>
<td>29.5%</td>
<td>33</td>
<td>328</td>
</tr>
<tr>
<td>haskell</td>
<td>49</td>
<td>153</td>
<td>0</td>
<td>0.0%</td>
<td>17</td>
<td>170</td>
</tr>
<tr>
<td>matlab</td>
<td>2</td>
<td>57</td>
<td>0</td>
<td>0.0%</td>
<td>8</td>
<td>65</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>61760</td>
<td>5773867</td>
<td>1751733</td>
<td>23.3%</td>
<td>1264262</td>
<td>8789862</td>
</tr>
</tbody>
</table>
## ohcount: Line Count of gcc-4.7.2/gcc

<table>
<thead>
<tr>
<th>Language</th>
<th>Files</th>
<th>Code</th>
<th>Comment</th>
<th>Comment %</th>
<th>Blank</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>c</td>
<td>17849</td>
<td>1601863</td>
<td>335879</td>
<td>17.3%</td>
<td>344693</td>
<td>2282435</td>
</tr>
<tr>
<td>ada</td>
<td>4903</td>
<td>724957</td>
<td>333800</td>
<td>31.5%</td>
<td>251445</td>
<td>1310202</td>
</tr>
<tr>
<td>cpp</td>
<td>9563</td>
<td>275971</td>
<td>63875</td>
<td>18.8%</td>
<td>71647</td>
<td>411493</td>
</tr>
<tr>
<td>fortranfixed</td>
<td>3158</td>
<td>105987</td>
<td>1961</td>
<td>1.8%</td>
<td>15175</td>
<td>123123</td>
</tr>
<tr>
<td>autoconf</td>
<td>3</td>
<td>30014</td>
<td>12</td>
<td>0.0%</td>
<td>4139</td>
<td>34165</td>
</tr>
<tr>
<td>objective_c</td>
<td>877</td>
<td>28017</td>
<td>5109</td>
<td>15.4%</td>
<td>8249</td>
<td>41375</td>
</tr>
<tr>
<td>fortranfree</td>
<td>834</td>
<td>13516</td>
<td>3234</td>
<td>19.3%</td>
<td>1716</td>
<td>18466</td>
</tr>
<tr>
<td>scheme</td>
<td>6</td>
<td>11023</td>
<td>1010</td>
<td>8.4%</td>
<td>1205</td>
<td>13238</td>
</tr>
<tr>
<td>make</td>
<td>6</td>
<td>6248</td>
<td>1113</td>
<td>15.1%</td>
<td>916</td>
<td>8277</td>
</tr>
<tr>
<td>tex</td>
<td>1</td>
<td>5441</td>
<td>2835</td>
<td>34.3%</td>
<td>702</td>
<td>8978</td>
</tr>
<tr>
<td>ocaml</td>
<td>6</td>
<td>2830</td>
<td>576</td>
<td>16.9%</td>
<td>378</td>
<td>3784</td>
</tr>
<tr>
<td>shell</td>
<td>22</td>
<td>2265</td>
<td>735</td>
<td>24.5%</td>
<td>391</td>
<td>3391</td>
</tr>
<tr>
<td>awk</td>
<td>11</td>
<td>1646</td>
<td>390</td>
<td>19.2%</td>
<td>271</td>
<td>2307</td>
</tr>
<tr>
<td>perl</td>
<td>3</td>
<td>913</td>
<td>226</td>
<td>19.8%</td>
<td>163</td>
<td>1302</td>
</tr>
<tr>
<td>assembler</td>
<td>7</td>
<td>343</td>
<td>136</td>
<td>28.4%</td>
<td>27</td>
<td>506</td>
</tr>
<tr>
<td>haskell</td>
<td>49</td>
<td>153</td>
<td>0</td>
<td>0.0%</td>
<td>17</td>
<td>170</td>
</tr>
<tr>
<td>matlab</td>
<td>2</td>
<td>57</td>
<td>0</td>
<td>0.0%</td>
<td>8</td>
<td>65</td>
</tr>
<tr>
<td>Total</td>
<td>37300</td>
<td>2811244</td>
<td>750891</td>
<td>21.1%</td>
<td>701142</td>
<td>4263277</td>
</tr>
<tr>
<td>Language</td>
<td>Files</td>
<td>Code</td>
<td>Comment</td>
<td>Comment %</td>
<td>Blank</td>
<td>Total</td>
</tr>
<tr>
<td>---------------</td>
<td>-------</td>
<td>---------</td>
<td>---------</td>
<td>-----------</td>
<td>----------</td>
<td>----------</td>
</tr>
<tr>
<td>c</td>
<td>17849</td>
<td>1601863</td>
<td>335879</td>
<td>17.3%</td>
<td>344693</td>
<td>2282435</td>
</tr>
<tr>
<td>ada</td>
<td>4903</td>
<td>724957</td>
<td>333800</td>
<td>31.5%</td>
<td>251445</td>
<td>1310202</td>
</tr>
<tr>
<td>cpp</td>
<td>9563</td>
<td>275971</td>
<td>63875</td>
<td>18.8%</td>
<td>71647</td>
<td>411493</td>
</tr>
<tr>
<td>fortranfixed</td>
<td>3158</td>
<td>105987</td>
<td>1961</td>
<td>1.8%</td>
<td>15175</td>
<td>123123</td>
</tr>
<tr>
<td>autoconf</td>
<td>3</td>
<td>30014</td>
<td>12</td>
<td>0.0%</td>
<td>4139</td>
<td>34165</td>
</tr>
<tr>
<td>objective_c</td>
<td>877</td>
<td>28017</td>
<td>5109</td>
<td>15.4%</td>
<td>8249</td>
<td>41375</td>
</tr>
<tr>
<td>fortranfree</td>
<td>834</td>
<td>13516</td>
<td>3234</td>
<td>19.3%</td>
<td>1716</td>
<td>18466</td>
</tr>
<tr>
<td>scheme</td>
<td>6</td>
<td>11023</td>
<td>1010</td>
<td>8.4%</td>
<td>1205</td>
<td>13238</td>
</tr>
<tr>
<td>make</td>
<td>6</td>
<td>6248</td>
<td>1113</td>
<td>15.1%</td>
<td>916</td>
<td>8277</td>
</tr>
<tr>
<td>tex</td>
<td>1</td>
<td>5441</td>
<td>2835</td>
<td>34.3%</td>
<td>702</td>
<td>8978</td>
</tr>
<tr>
<td>ocaml</td>
<td>6</td>
<td>2830</td>
<td>576</td>
<td>16.9%</td>
<td>378</td>
<td>3784</td>
</tr>
<tr>
<td>shell</td>
<td>22</td>
<td>2265</td>
<td>735</td>
<td>24.5%</td>
<td>391</td>
<td>3391</td>
</tr>
<tr>
<td>awk</td>
<td>11</td>
<td>1646</td>
<td>390</td>
<td>19.2%</td>
<td>271</td>
<td>2307</td>
</tr>
<tr>
<td>perl</td>
<td>3</td>
<td>913</td>
<td>226</td>
<td>19.8%</td>
<td>163</td>
<td>1302</td>
</tr>
<tr>
<td>assembler</td>
<td>7</td>
<td>343</td>
<td>136</td>
<td>28.4%</td>
<td>27</td>
<td>506</td>
</tr>
<tr>
<td>haskell</td>
<td>49</td>
<td>153</td>
<td>0</td>
<td>0.0%</td>
<td>17</td>
<td>170</td>
</tr>
<tr>
<td>matlab</td>
<td>2</td>
<td>57</td>
<td>0</td>
<td>0.0%</td>
<td>8</td>
<td>65</td>
</tr>
<tr>
<td>Total</td>
<td>37300</td>
<td>2811244</td>
<td>750891</td>
<td>21.1%</td>
<td>701142</td>
<td>4263277</td>
</tr>
</tbody>
</table>
Why is Understanding GCC Difficult?

Deeper technical reasons

- GCC is not a compiler but a \emph{compiler generation framework}
  Two distinct gaps that need to be bridged
  - Input-output of the generation framework
    The target specification and the generated compiler
  - Input-output of the generated compiler
    A source program and the generated assembly program

- GCC generated compiler uses a derivative of the Davidson-Fraser model of compilation
  - Early instruction selection
  - Machine dependent intermediate representation
  - Simplistic instruction selection and retargatibility mechanism
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 | Expander | Optimizer | Recognizer

Source Program | Generated Compiler (cc1) | Assembly Program

Uday Khedker
GRC, IIT Bombay
The Architecture of GCC

Input Language

Language Specific Code

Language and Machine Independent Generic Code

Machine Dependent Generator Code

Machine Descriptions

Compiler Generation Framework

Selected

Copied

Copied

Generated

Generated

Generated

Source Program

Generated Compiler (cc1)

Assembly Program

Parser

Gimplifier

Tree SSA Optimizer

Expander

Optimizer

Recognizer

Uday Khedker

GRC, IIT Bombay
The Architecture of GCC

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

- **Target Name**
  - Parser
  - Gimplacer
  - Tree SSA Optimizer
  - Expander
  - Optimizer
  - Recognizer

- **Compiler Generation Framework**
  - Development Time
  - Build Time
  - Use Time

- **Generated Compiler (cc1)**
  - Source Program
  - Assembly Program

- **Selected**
  - Copied

- **Generated**
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

```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!
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

```bash
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

```plaintext
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 gcc-4.7.2/gcc using cscope -R
Another Example of The Generation Related Gap

- Locating the main function in gcc-4.7.2/gcc using cscope -R

8125 occurrences!
Another Example of The Generation Related Gap

- Locating the main function in gcc-4.7.2/gcc using cscope -R
  8125 occurrences!

- Number of main functions in the entire tarball
Another Example of The Generation Related Gap

• Locating the main function in gcc-4.7.2/gcc using cscope -R

  8125 occurrences!

• Number of main functions in the entire tarball

  12799!
Another Example of The Generation Related Gap

- Locating the main function in gcc-4.7.2/gcc using cscope -R
  
  8125 occurrences!

- Number of main functions in the entire tarball
  
  12799!

- What if we do not search recursively?
Another Example of The Generation Related Gap
Locating the main function in the directory gcc-4.7.2/gcc using cscope
Another Example of The Generation Related Gap

Locating the main function in the directory gcc-4.7.2/gcc using cscope

<table>
<thead>
<tr>
<th>File</th>
<th>Line</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 s-oscons-tmplt.c</td>
<td>238</td>
<td>main (void) {</td>
</tr>
<tr>
<td>1 collect2.c</td>
<td>1021</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>2 divtab-sh4-300.c</td>
<td>31</td>
<td>main ()</td>
</tr>
<tr>
<td>3 divtab-sh4.c</td>
<td>30</td>
<td>main ()</td>
</tr>
<tr>
<td>4 divtab.c</td>
<td>131</td>
<td>main ()</td>
</tr>
<tr>
<td>5 gen-mul-tables.cc</td>
<td>1224</td>
<td>main ()</td>
</tr>
<tr>
<td>6 vms-ar.c</td>
<td>122</td>
<td>main (int argc, char *argv[])</td>
</tr>
<tr>
<td>7 vms-ld.c</td>
<td>559</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>8 fp-test.c</td>
<td>85</td>
<td>main (void)</td>
</tr>
<tr>
<td>9 gcc-ar.c</td>
<td>36</td>
<td>main(int ac, char **av)</td>
</tr>
<tr>
<td>a gcc.c</td>
<td>6105</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>b gcov-dump.c</td>
<td>78</td>
<td>main (int argc ATTRIBUTE_UNUSED, char *argv)</td>
</tr>
<tr>
<td>c gcov-iov.c</td>
<td>29</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>d gcov.c</td>
<td>397</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>e genattr-common.c</td>
<td>64</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>f genattr.c</td>
<td>141</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-4.7.2/gcc using cscope

- g genattrtab.c: 4880 main (int argc, char **argv)
- h genautomata.c: 9617 main (int argc, char **argv)
- i genchecksum.c: 97 main (int argc, char **argv)
- j gencodes.c: 51 main (int argc, char **argv)
- k genconditions.c: 209 main (int argc, char **argv)
- l genconfig.c: 261 main (int argc, char **argv)
- m genconstants.c: 79 main (int argc, char **argv)
- n genemit.c: 775 main (int argc, char **argv)
- o genenums.c: 48 main (int argc, char **argv)
- p genextract.c: 402 main (int argc, char **argv)
- q genflags.c: 251 main (int argc, char **argv)
- r gengenrtl.c: 286 main (void)
- s gengtype.c: 4925 main (int argc, char **argv)
- t genhooks.c: 342 main (int argc, char **argv)
- u genmddeps.c: 43 main (int argc, char **argv)
- v genmodes.c: 1388 main (int argc, char **argv)
- w genopinit.c: 504 main (int argc, char **argv)
- x genoutput.c: 997 main (int argc, char **argv)

Uday Khedker  
GRC, IIT Bombay
GCC Retargetability Mechanism

Compiler Generation Framework

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

Target Name

Development Time

Build Time

Use Time

Generated Compiler

- Parser
- Gimplifier
- Tree SSA Optimizer
- Expander
- Optimizer
- Recognizer

Use Time
GCC Retargetability Mechanism

Input Language

Parser
Gimplifier
Tree SSA Optimizer
Expander
Optimizer
Recognizer

Language Specific Code

Language and Machine Independent Generic Code

Machine Dependent Generator Code

Machine Descriptions

Compiled Compiler

Development Time

Build Time

Use Time

Gimple $\rightarrow$ IR-RTL

IR-RTL $\rightarrow$ ASM
GCC Retargetability Mechanism

Input Language

Compiler Generation Framework

Target Name

Gimple → PN

PN → IR-RTL

IR-RTL → ASM

Gimple → IR-RTL

IR-RTL → ASM

Uday Khedker

GRC, IIT Bombay
GCC Retargetability Mechanism

Input Language

Compiler Generation Framework

Language Specific Code

Language and Machine Independent Generic Code

Machine Dependent Generator Code

Machine Descriptions

Target Name

Selected

Copied

Copied

Generated

Generated

Parser

Gimplifier

Tree SSA Optimizer

Expander

Optimizer

Recognizer

Development Time

Build Time

Use Time

Gimple → PN

PN → IR-RTL

IR-RTL → ASM

Gimple → IR-RTL

IR-RTL → ASM

Generated Compiler

Parser

Gimplifier

Tree SSA Optimizer

Expander

Optimizer

Recognizer
GCC Retargetability Mechanism

Compiler Generation Framework

Input Language → Language Specific Code

Language and Machine Independent Generic Code → Copied

Machine Dependent Generator Code → Copied

Machine Descriptions → Generated

Parser → Gimplifier → Tree SSA Optimizer

Expander → Optimizer → Recognizer

Gimple → PN

PN → IR-RTL

IR-RTL → ASM

Gimple → IR-RTL

IR-RTL → ASM

Development Time

Build Time

Use Time

Generated Compiler
The generated compiler uses an adaptation of the Davidson Fraser model

- Generic expander and recognizer
- Machine specific information is isolated in data structures
- Generating a compiler involves generating these data structures
The GCC Challenge: Poor Retargetability Mechanism

Symptoms:

- Machine descriptions are large, verbose, repetitive, and contain large chunks of C code

Size in terms of line counts (counted using `wc -l`)

<table>
<thead>
<tr>
<th></th>
<th>gcc-4.6.2</th>
<th>gcc-4.7.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>i386</td>
<td>38851</td>
<td>39582</td>
</tr>
<tr>
<td>mips</td>
<td>15534</td>
<td>16437</td>
</tr>
<tr>
<td>arm</td>
<td>30951</td>
<td>32385</td>
</tr>
<tr>
<td>* .md</td>
<td>39780</td>
<td>41985</td>
</tr>
<tr>
<td>* .c</td>
<td>16793</td>
<td>17761</td>
</tr>
<tr>
<td>* .h</td>
<td>26165</td>
<td>26006</td>
</tr>
<tr>
<td>Total</td>
<td>96510</td>
<td>100741</td>
</tr>
<tr>
<td></td>
<td>37996</td>
<td>39784</td>
</tr>
<tr>
<td></td>
<td>75929</td>
<td>76403</td>
</tr>
</tbody>
</table>
The GCC Challenge: Poor Retargetability Mechanism

Symptoms:

- Machine descriptions are large, verbose, repetitive, and contain large chunks of C code

Size in terms of line counts (counted using `wc -l`)

<table>
<thead>
<tr>
<th></th>
<th>gcc-4.6.2</th>
<th>gcc-4.7.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Files</td>
<td>i386</td>
<td>mips</td>
</tr>
<tr>
<td>*.md</td>
<td>38851</td>
<td>15534</td>
</tr>
<tr>
<td>*.c</td>
<td>39780</td>
<td>16793</td>
</tr>
<tr>
<td>*.h</td>
<td>17879</td>
<td>5667</td>
</tr>
<tr>
<td>Total</td>
<td>96510</td>
<td>37996</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>gcc-4.6.2</th>
<th>gcc-4.7.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Files</td>
<td>i386</td>
<td>mips</td>
</tr>
<tr>
<td>*.md</td>
<td>39582</td>
<td>16437</td>
</tr>
<tr>
<td>*.c</td>
<td>41985</td>
<td>17761</td>
</tr>
<tr>
<td>*.h</td>
<td>19174</td>
<td>5586</td>
</tr>
<tr>
<td>Total</td>
<td>100741</td>
<td>39784</td>
</tr>
</tbody>
</table>

- Machine descriptions are difficult to construct, understand, debug, and enhance
Part 3

Meeting the GCC Challenge
## Meeting the GCC Challenge

<table>
<thead>
<tr>
<th>Goal of Understanding</th>
<th>Methodology</th>
<th>Needs Examining</th>
</tr>
</thead>
<tbody>
<tr>
<td>Translation sequence of programs</td>
<td>Gray box probing</td>
<td>No</td>
</tr>
<tr>
<td>Build process</td>
<td>Customising the configuration and building</td>
<td>Yes</td>
</tr>
<tr>
<td>Retargetability issues and machine</td>
<td>Incremental construction of machine descriptions</td>
<td>No</td>
</tr>
<tr>
<td>IR data structures and access mechanisms</td>
<td>Adding passes to massage IRs</td>
<td>No</td>
</tr>
<tr>
<td>Retargetability mechanism</td>
<td></td>
<td>Yes</td>
</tr>
</tbody>
</table>

**Needs Examining:**
- **Makefiles**
- **Source**
- **MD**
The Grand Picture and Our Coverage

Compiler Specifications

Compiler Generator

Generated Compiler
The Grand Picture and Our Coverage

- Compiler Specifications
  - Compiler Generator
  - Generated Compiler

External View

Internal View

Uday Khedker

GRC, IIT Bombay
The Grand Picture and Our Coverage

- Compiler Specifications
- Compiler Generator
- Generated Compiler
- Gray box probing

External View | Internal View
The Grand Picture and Our Coverage

External View

- Compiler Specifications
- Compiler Generator
- Generated Compiler
- Gray box probing
- Pass structure and IR

Internal View
The Grand Picture and Our Coverage

External View

Internal View

Compiler Specifications

Configuration and building

Gray box probing
Pass structure and IR

Compiler Generator

Generated Compiler

Uday Khedker
GRC, IIT Bombay
The Grand Picture and Our Coverage

Compiler Specifications

Compiler Generator

Generated Compiler

External View

Internal View

Front end hooks

Configuration and building

Gray box probing

Pass structure and IR
The Grand Picture and Our Coverage

Compiler Specifications

- Compiler Generator
- Generated Compiler

External View
- Configuration and building

Internal View
- Front end hooks
- Gray box probing
- Manipulating IRs
- Pass structure and IR
The Grand Picture and Our Coverage

Compiler Specifications

Compiler Generator

Generated Compiler

External View

Configuration and building

Gray box probing
Pass structure and IR

Internal View

Front end hooks

Manipulating IRs
Control flow, LTO

Uday Khedker

GRC, IIT Bombay
The Grand Picture and Our Coverage

External View

- Compiler Specifications
- Front end hooks

Internal View

- Configuration and building
- Gray box probing
- Pass structure and IR
- Manipulating IRs
- Control flow, LTO
- Plugin mechanisms

Compiler Specifications

Compiler Generator

Generated Compiler

Uday Khedker
GRC, IIT Bombay
The Grand Picture and Our Coverage

External View

Compiler Specifications

Compiler Generator

Generated Compiler

Configuration and building

Gray box probing
Pass structure and IR
Data Flow Analysis

Internal View

Front end hooks

Manipulating IRs
Control flow, LTO
Plugin mechanisms

Uday Khedker
GRC, IIT Bombay
The Grand Picture and Our Coverage

Compiler Specifications

External View

Machine descriptions

Internal View

Front end hooks

Compiler Generator

Configuration and building

Gray box probing

Pass structure and IR

Data Flow Analysis

Generated Compiler

Manipulating IRs

Control flow, LTO

Plugin mechanisms

Uday Khedker

GRC, IIT Bombay
The Grand Picture and Our Coverage

- **Compiler Specifications**
  - Compiler Generator
    - Generated Compiler
  - External View
    - Machine descriptions
    - Configuration and building
  - Internal View
    - Front end hooks
    - Retargetability mechanism

- **Generated Compiler**
  - Gray box probing
    - Pass structure and IR
      - Data Flow Analysis
    - Manipulating IRs
      - Control flow, LTO
        - Plugin mechanisms

Uday Khedker
GRC, IIT Bombay
The Grand Picture and Our Coverage

Compiler Specifications

External View
- Machine descriptions
- Configuration and building

Internal View
- Front end hooks
- Retargetability mechanism

Generated Compiler

- Gray box probing
- Pass structure and IR
- Data Flow Analysis
- Parallelization, Vectorization
- Manipulating IRs
- Control flow, LTO
- Plugin mechanisms

Compiler Generator

- Uday Khedker
  GRC, IIT Bombay
The Grand Picture and Our Coverage

Compiler Specifications

External View

Machine descriptions
Configuration and building

Internal View

Front end hooks
Retargetability mechanism

Gray box probing
Pass structure and IR
Data Flow Analysis
Parallelization, Vectorization

Generated Compiler

Compiler Generator

Uday Khedker
GRC, IIT Bombay