Workshop on Essential Abstractions in GCC

An Overview of Compilation and GCC

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

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

29 June 2013
Outline

• Introduction to Compilation
• An Overview of Compilation Phases
• An Overview of GCC
Part 1

Introduction to Compilation
Implementation Mechanisms

Source Program

\[\text{Translator}\]

Target Program

\[\text{Machine}\]
Implementation Mechanisms

Source Program

Translator

Target Program

Machine

Input Data

Essential Abstractions in GCC

GCC Resource Center, IIT Bombay
Implementation Mechanisms

Source Program

Translator

Target Program

Machine

Input Data

Interpreter

Machine
Implementation Mechanisms as “Bridges”

• “Gap” between the “levels” of program specification and execution

Program Specification

Machine
Implementation Mechanisms as “Bridges”

- “Gap” between the “levels” of program specification and execution

Diagram:

```
Program Specification
↓
Translation
↓
Machine
```
Implementation Mechanisms as “Bridges”

• “Gap” between the “levels” of program specification and execution

Diagram:

- Program Specification
  - Translation
  - Interpretation
  - Machine
Implementation Mechanisms as “Bridges”

- “Gap” between the “levels” of program specification and execution

**Program Specification**
- Translation
- Interpretation

**Machine**
- State: Memory, Registers
- Operations: Machine Instructions

**Program Specification**
- State: Variables
- Operations: Expressions, Control Flow

Essential Abstractions in GCC

GCC Resource Center, IIT Bombay
High and Low Level Abstractions

Input C statement

\[ a = b<10?b:c; \]

Spim Assembly Equivalent

\[
\begin{align*}
    \text{lw} & \quad \texttt{$t0, 4($fp)} ; & \quad \texttt{t0} & \leftarrow \texttt{b} & \quad \# \texttt{Is b smaller} \\
    \text{slti} & \quad \texttt{$t0, $t0, 10} ; & \quad \texttt{t0} & \leftarrow \texttt{t0 < 10} & \quad \# \texttt{than 10?} \\
    \text{not} & \quad \texttt{$t0, $t0} ; & \quad \texttt{t0} & \leftarrow \texttt{!t0} \\
    \text{bgtz} & \quad \texttt{$t0, L0:} ; & \quad \texttt{if t0>0 goto L0} \\
    \text{lw} & \quad \texttt{$t0, 4($fp)} ; & \quad \texttt{t0} & \leftarrow \texttt{b} & \quad \# \texttt{YES} \\
    \text{b} & \quad \texttt{L1:} ; & \quad \texttt{goto L1} \\
    \text{L0: lw} & \quad \texttt{$t0, 8($fp)} ; & \quad \texttt{L0: t0} & \leftarrow \texttt{c} & \quad \# \texttt{NO} \\
    \text{L1: sw} & \quad \texttt{0($fp), $t0} ; & \quad \texttt{L1: a} & \leftarrow \texttt{t0}
\end{align*}
\]
Implementation Mechanisms

- Translation = Analysis + Synthesis
- Interpretation = Analysis + Execution
Implementation Mechanisms

- Translation = Analysis + Synthesis
  Interpretation = Analysis + Execution

- Translation Instructions $\rightarrow$ Equivalent Instructions
Implementation Mechanisms

- **Translation** = Analysis + Synthesis
- **Interpretation** = Analysis + Execution

- Translation Instructions $\xrightarrow{}$ Equivalent Instructions
- Interpretation Instructions $\xrightarrow{}$ Actions IMPLIED by Instructions

Essential Abstractions in GCC

GCC Resource Center, IIT Bombay
Language Implementation Models

- Analysis
- Execution
- Synthesis
- Compilation
- Interpretation
Language Processor Models

Front End → Optimizer → Back End

- C, C++
- Java, C#
Part 2

An Overview of Compilation Phases
The Structure of a Simple Compiler
The Structure of a Simple Compiler

Parser → AST → Instruction Selector → Insn → Assembly Emitter

Scanner → Source Program

Semantic Analyser

Symtab Handler

Register Allocator

Assembly Program
The Structure of a Simple Compiler

Front End
- Scanner
- Semantic Analyser
- Symtab Handler
- Parser

AST

Back End
- Instruction Selector
- Register Allocator
- Assembly Emitter

Insn
Assembly Program

Source Program

Essential Abstractions in GCC

GCC Resource Center, IIT Bombay
Translation Sequence in Our Compiler: Parsing

input

a = b < 10 ? b : c;
Translation Sequence in Our Compiler: Parsing

Input

a=b<10?b:c;

Parse Tree

Issues:

- Grammar rules, terminals, non-terminals
- Order of application of grammar rules
  eg. is it (a = b<10?) followed by (b:c)?
- Values of terminal symbols
  eg. string “10” vs. integer number 10.
Translation Sequence in Our Compiler: Semantic Analysis

```plaintext
a = b < 10 ? b : c;
```

Input

```
AsgnStmnt

Lhs  =  E ;
    /   \
  name  E  ?  E  :  E
                    /   \
               E   <  E  name  name
                   /   \
              name  num
```

Parse Tree
Translation Sequence in Our Compiler: Semantic Analysis

\[ a = b < 10 ? b : c; \]

**Input**

```
Lhs := E ;
E ? E : E
E < E name name
name name num
```

**Parse Tree**

```
name (a, int) ?: (int) < name (b, int) name (c, int)
name (b, int) num (10, int)
```

**Abstract Syntax Tree** (with attributes)

**Issues:**

- **Symbol tables**
  Have variables been declared? What are their types? What is their scope?

- **Type consistency of operators and operands**
  The result of computing \( b < 10 ? \) is bool and not int
Translation Sequence in Our Compiler: IR Generation

Input

```
a=b<10?b:c;
```

Parse Tree

```
AsgnStmt

Lhs  :=  E ;

name  E  ?  E  :  E

E  <  E  name  name

name  num
```

Abstract Syntax Tree
(with attributes)

```
name  (a,int)  ?:  (int)

<  name  (b,int)  name  (c,int)

name  num  (10,int)

name  (b,int)
```

Essential Abstractions in GCC

GCC Resource Center, IIT Bombay
Translation Sequence in Our Compiler: IR Generation

a = b < 10 ? b : c;

Input

Tree List

\[
T_0 = \text{Not} \left< b \right> 10
\]

Parse Tree

Abstract Syntax Tree (with attributes)

Issues:

- Convert to maximal trees which can be implemented without altering control flow
  Simplifies instruction selection and scheduling, register allocation etc.
- Linearise control flow by flattening nested control constructs
Translation Sequence in Our Compiler: Instruction Selection

```plaintext
a = b < 10 ? b : c;
```

### Input

#### Parse Tree

- **AsgnStmtnt**
  - **Lhs** = **E**
  - **?**
    - **E**
    - **E**
    - **:**
      - **E**
      - **<**
      - **E**
      - **name**
      - **name**

### Abstract Syntax Tree (with attributes)

- **name** (a, int)
  - ?
    - (int)
      - **<**
        - **name** (b, int)
        - **name** (c, int)

- **name** (b, int)
  - **num** (10, int)

### Tree List

1. **T₀**
   - **Not**
     - **<**
       - **b**
       - **10**

2. **T₀**
   - **IfGoto**
     - **L0:**

3. **T₁**
   - **Goto**
     - **L1:**

4. **L0:**
   - **T₁**
     - **=**
       - **c**

5. **L1:**
   - **=**
     - **T₁**
     - **a**

---

Essential Abstractions in GCC

GCC Resource Center, IIT Bombay
Translation Sequence in Our Compiler: Instruction Selection

a=b<10?b:c;

Input

Tree List

Parse Tree

Abstract Syntax Tree (with attributes)

Issues:

- Cover trees with as few machine instructions as possible
- Use temporaries and local registers
Translation Sequence in Our Compiler: Instruction Selection

```
a=b<10?b:c;
```

**Input**

- **Tree List**
  - \( T_0 \leftarrow \text{Not} \)
  - \( b \leftarrow 10 \)
  - **IfGoto**
    - \( T_0 \leftarrow \text{L0:} \)

**Parse Tree**

- **AsgnStmt**
  - \( \text{Lhs} \leftarrow \text{E} \)
  - \( \text{name} \leftarrow \text{E} \)
  - \( \text{E} \leftarrow \text{?} \)
  - \( \text{E} \leftarrow \text{E} \)
  - \( \text{E} \leftarrow \text{<} \)
  - \( \text{E} \leftarrow \text{E} \)
  - \( \text{name} \leftarrow \text{name} \)
  - \( \text{name} \leftarrow \text{name} \)
  - **Abstract Syntax Tree**
    - \( \text{name (a,int)} \leftarrow ?: (\text{int}) \)
    - \( \text{name (b,int)} \leftarrow \text{(bool)} \)
    - \( \text{name (c,int)} \leftarrow \text{name (b,int)} \)
    - \( \text{num (10,int)} \leftarrow \text{name (b,int)} \)

**Instruction List**

- \( T_0 \leftarrow b \)
- \( T_0 \leftarrow T_0 < 10 \)
- \( T_0 \leftarrow \neg T_0 \)
- if \( T_0 > 0 \) goto L0:
  - \( T_1 \leftarrow b \)
  - goto L1:
- L0: \( T_1 \leftarrow c \)
- L1: \( a \leftarrow T_1 \)

**Issues:**

- Cover trees with as few machine instructions as possible
- Use temporaries and local registers
Translation Sequence in Our Compiler: Emitting Instructions

Input:
```
a=b<10?b:c;
```

Parse Tree:
```
E < E name name
name name num
```

Abstract Syntax Tree (with attributes):
```
name (a,int) ?: (int)
< name (b,int) name (c,int)
name (b,int) num (10,int)
```

Instruction List:
```
T_0 ← b
T_0 ← T_0 < 10
if T_0 > 0 goto L0:
T_1 ← b
goto L1:
L0: T_1 ← c
L1: a ← T_1
```

Tree List:
```
= Not < b 10
```

Instruction List:
```
L0: =
T_1 ← c
L1: =
a ← T_1
```

Essential Abstractions in GCC

GCC Resource Center, IIT Bombay
Translation Sequence in Our Compiler: Emitting Instructions

### Input

```
a = b < 10 ? b : c;
```

### Tree List

```
\( T_0 \rightarrow \text{Not} \rightarrow \langle b \rangle \rightarrow 10 \)
```

### IfGoto

```
T_0 \rightarrow \text{L0:} \rightarrow \text{Goto} \rightarrow \text{L1:} \rightarrow \text{L0:} \rightarrow \text{L1:} \rightarrow \text{a} \rightarrow T_1
```

### AsgnStmtnt

```
\text{Issues:}
- Offsets of variables in the stack frame
- Actual register numbers and assembly mnemonics
- Code to construct and discard activation records
```

### Instruction List

```
T_0 \leftarrow b
T_0 \leftarrow T_0 < 10
T_0 \leftarrow ! T_0
\text{if } T_0 > 0 \text{ goto L0:}
T_1 \leftarrow b
goto L1:
L0: T_1 \leftarrow c
L1: a \leftarrow T_1
```

### Assembly Code

```
lw $t0, 4($fp)
slti $t0, $t0, 10
\text{not } $t0, $t0
\text{bgtz } T_0, \text{L0:}
lw $t0, 4($fp)
b \text{L1:}
L0: lw $t0, 8($fp)
b L1:
L1: sw 0($fp), $t0
```
Part 3

Compilation Models
Compilation Models

Aho Ullman Model

Davidson Fraser Model

Essential Abstractions in GCC

GCC Resource Center, IIT Bombay
Compilation Models

Aho Ullman Model

Front End

 AST

Optimizer

Target Indep. IR

Davidson Fraser Model

Input Source Program

Essential Abstractions in GCC

GCC Resource Center, IIT Bombay
Compilation Models

Aho Ullman Model

Input Source Program

Front End

AST

Optimizer

Target Indep. IR

Code Generator

Target Program

Essential Abstractions in GCC

GCC Resource Center, IIT Bombay
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

Essential Abstractions in GCC

GCC Resource Center, IIT Bombay
Compilation Models

**Aho Ullman Model**
- Front End
  - AST
  - Optimizer
  - Target Indep. IR
  - Code Generator
  - Target Program

**Davidson Fraser Model**
- Front End
  - AST
  - Expander
  - Register Transfers

Input Source Program

Essential Abstractions in GCC

GCC Resource Center, IIT Bombay
Compilation Models

**Aho Ullman Model**
- Front End
- AST
- Optimizer
- Target Indep. IR
- Code Generator
- Target Program

**Davidson Fraser Model**
- Front End
- AST
- Expander
- Register Transfers
- Optimizer
- Register Transfers

Essential Abstractions in GCC

GCC Resource Center, IIT Bombay
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

Recognizer

Target Program

Essential Abstractions in GCC

GCC Resource Center, IIT Bombay
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

Essential Abstractions in GCC

GCC Resource Center, IIT Bombay
Typical Front Ends

Parser
Typical Front Ends

Source Program → Tokens → Scanner → Parser
Typical Front Ends

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

- **Source Program**
- **Scanner**
  - **Tokens**
- **Parser**
  - **AST**
  - **Parse Tree**
- **Semantic Analyzer**
  - **AST or Linear IR + Symbol Table**
- **Symtab Handler**
- **Error Handler**

Essential Abstractions in GCC

GCC Resource Center, IIT Bombay
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

Assembly Code
Typical Back Ends in Aho Ullman Model

- Compile time evaluations
- Eliminating redundant computations

- Instruction Selection
- Local Reg Allocation
- Choice of Order of Evaluation
Retargetability in Aho Ullman and Davidson Fraser Models

<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>

Essential Abstractions in GCC

GCC Resource Center, IIT Bombay
## Retargetability in Aho Ullman and Davidson Fraser Models

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

---

**Essential Abstractions in GCC**

**GCC Resource Center, 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>Structural tree pattern matching</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></td>
</tr>
</tbody>
</table>

**Key Insight:** Register transfers are target specific but their form is target independent.
Part 4

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

gcc

Target Program
The GNU Tool Chain for C

Source Program

gcc

cc1

Target Program
The GNU Tool Chain for C

Source Program

\[ \downarrow \]

gcc

\[ \downarrow \]

Target Program

cc1

cpp
The GNU Tool Chain for C

Source Program

gcc

Target Program

cc1

cpp

as
The GNU Tool Chain for C

Source Program

```
| gcc |
```

Target Program

```
| cc1 | as | ld | cpp |
```

Essential Abstractions in GCC

GCC Resource Center, IIT Bombay
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
Open Source and Free Software Development Model

The Cathedral and the Bazaar [Eric S Raymond, 1997]
Open Source and Free Software Development Model

The Cathedral and the Bazaar [Eric S Raymond, 1997]

- Cathedral: Total Centralized Control
  
  Design, implement, test, release
Open Source and Free Software Development Model

The Cathedral and the Bazaar [Eric S Raymond, 1997]

- **Cathedral:** Total Centralized Control
  
  *Design, implement, test, release*

- **Bazaar:** Total Decentralization

  *Release early, release often, make users partners in software development*
Open Source and Free Software Development Model

The Cathedral and the Bazaar [Eric S Raymond, 1997]

- Cathedral: Total Centralized Control
  
  Design, implement, test, release

- Bazaar: Total Decentralization
  
  Release early, release often, make users partners in software development

“Given enough eyeballs, all bugs are shallow”
The Cathedral and the Bazaar [Eric S Raymond, 1997]

- **Cathedral: Total Centralized Control**
  
  *Design, implement, test, release*

- **Bazaar: Total Decentralization**

  *Release early, release often, make users partners in software development*

  “Given enough eyeballs, all bugs are shallow”

Code errors, logical errors, and architectural errors
Open Source and Free Software Development Model

The Cathedral and the Bazaar [Eric S Raymond, 1997]

- **Cathedral: Total Centralized Control**
  
  Design, implement, test, release

- **Bazaar: Total Decentralization**
  
  Release early, release often, make users partners in software development

  “Given enough eyeballs, all bugs are shallow”

  Code errors, logical errors, and architectural errors

A combination of the two seems more sensible
The Current Development Model of GCC

GCC follows a combination of the Cathedral and the Bazaar approaches

- GCC Steering Committee: Free Software Foundation has given charge
  - Major policy decisions
  - Handling Administrative and Political issues

- Release Managers:
  - Coordination of releases

- Maintainers:
  - Usually area/branch/module specific
  - Responsible for design and implementation
  - Take help of reviewers to evaluate submitted changes
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,
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:**
    A29K, ARC, ETRAX CRIS, D30V, DSP16xx, FR-30, FR-V, Intel i960, IP2000, M32R, 68HC11,

- **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,
Overview: GCC ≡ The Great Compiler Challenge

29 June 2013

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-4.4.2

<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>15638</td>
<td>1840245</td>
<td>394682</td>
<td>17.7%</td>
<td>366815</td>
<td>2601742</td>
</tr>
<tr>
<td>cpp</td>
<td>19622</td>
<td>872775</td>
<td>190744</td>
<td>17.9%</td>
<td>189007</td>
<td>1252526</td>
</tr>
<tr>
<td>java</td>
<td>6342</td>
<td>681656</td>
<td>643045</td>
<td>48.5%</td>
<td>169465</td>
<td>1494166</td>
</tr>
<tr>
<td>ada</td>
<td>4206</td>
<td>638557</td>
<td>294881</td>
<td>31.6%</td>
<td>218000</td>
<td>1151438</td>
</tr>
<tr>
<td>autoconf</td>
<td>76</td>
<td>445046</td>
<td>393</td>
<td>0.1%</td>
<td>58831</td>
<td>504270</td>
</tr>
<tr>
<td>make</td>
<td>82</td>
<td>110064</td>
<td>3268</td>
<td>2.9%</td>
<td>13270</td>
<td>126602</td>
</tr>
<tr>
<td>html</td>
<td>480</td>
<td>103080</td>
<td>5658</td>
<td>5.2%</td>
<td>21438</td>
<td>130176</td>
</tr>
<tr>
<td>fortranfixed</td>
<td>2164</td>
<td>73366</td>
<td>1570</td>
<td>2.1%</td>
<td>9454</td>
<td>84390</td>
</tr>
<tr>
<td>assembler</td>
<td>183</td>
<td>42460</td>
<td>9607</td>
<td>18.5%</td>
<td>7084</td>
<td>59151</td>
</tr>
<tr>
<td>shell</td>
<td>137</td>
<td>39347</td>
<td>8832</td>
<td>18.3%</td>
<td>5485</td>
<td>53664</td>
</tr>
<tr>
<td>fortranfree</td>
<td>690</td>
<td>11852</td>
<td>2582</td>
<td>17.9%</td>
<td>1414</td>
<td>15848</td>
</tr>
<tr>
<td>objective_c</td>
<td>395</td>
<td>10562</td>
<td>1768</td>
<td>14.3%</td>
<td>2951</td>
<td>15281</td>
</tr>
<tr>
<td>automake</td>
<td>61</td>
<td>6014</td>
<td>853</td>
<td>12.4%</td>
<td>956</td>
<td>7823</td>
</tr>
<tr>
<td>perl</td>
<td>24</td>
<td>4111</td>
<td>1138</td>
<td>21.7%</td>
<td>732</td>
<td>5981</td>
</tr>
<tr>
<td>scheme</td>
<td>1</td>
<td>2775</td>
<td>153</td>
<td>5.2%</td>
<td>328</td>
<td>3256</td>
</tr>
<tr>
<td>ocaml</td>
<td>5</td>
<td>2482</td>
<td>538</td>
<td>17.8%</td>
<td>328</td>
<td>3348</td>
</tr>
<tr>
<td>python</td>
<td>6</td>
<td>1135</td>
<td>211</td>
<td>15.7%</td>
<td>220</td>
<td>1566</td>
</tr>
<tr>
<td>awk</td>
<td>9</td>
<td>1127</td>
<td>324</td>
<td>22.3%</td>
<td>193</td>
<td>1644</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>497</td>
<td>99</td>
<td>16.6%</td>
<td>30</td>
<td>626</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>haskell</td>
<td>48</td>
<td>149</td>
<td>0</td>
<td>0.0%</td>
<td>16</td>
<td>165</td>
</tr>
<tr>
<td>emacsclisp</td>
<td>1</td>
<td>59</td>
<td>21</td>
<td>26.2%</td>
<td>4</td>
<td>84</td>
</tr>
<tr>
<td>matlab</td>
<td>2</td>
<td>57</td>
<td>0</td>
<td>0.0%</td>
<td>7</td>
<td>64</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>50312</td>
<td>4938881</td>
<td>1567750</td>
<td>24.1%</td>
<td>1071986</td>
<td>7578617</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>15638</td>
<td>1840245</td>
<td>394682</td>
<td>17.7%</td>
<td>366815</td>
<td>2601742</td>
</tr>
<tr>
<td>cpp</td>
<td>19622</td>
<td>872775</td>
<td>190744</td>
<td>17.9%</td>
<td>189007</td>
<td>1252526</td>
</tr>
<tr>
<td>java</td>
<td>6342</td>
<td>681656</td>
<td>643045</td>
<td>48.5%</td>
<td>169465</td>
<td>1494166</td>
</tr>
<tr>
<td>ada</td>
<td>4206</td>
<td>638557</td>
<td>294881</td>
<td>31.6%</td>
<td>218000</td>
<td>1151438</td>
</tr>
<tr>
<td>autoconf</td>
<td>76</td>
<td>445046</td>
<td>393</td>
<td>0.1%</td>
<td>58831</td>
<td>504270</td>
</tr>
<tr>
<td>make</td>
<td>82</td>
<td>110064</td>
<td>3268</td>
<td>2.9%</td>
<td>13270</td>
<td>126602</td>
</tr>
<tr>
<td>html</td>
<td>480</td>
<td>103080</td>
<td>5658</td>
<td>5.2%</td>
<td>21438</td>
<td>130176</td>
</tr>
<tr>
<td>fortranfixed</td>
<td>2164</td>
<td>73366</td>
<td>1570</td>
<td>2.1%</td>
<td>9454</td>
<td>84390</td>
</tr>
<tr>
<td>assembler</td>
<td>183</td>
<td>42460</td>
<td>9607</td>
<td>18.5%</td>
<td>7084</td>
<td>59151</td>
</tr>
<tr>
<td>shell</td>
<td>137</td>
<td>39347</td>
<td>8832</td>
<td>18.3%</td>
<td>5485</td>
<td>53664</td>
</tr>
<tr>
<td>fortranfree</td>
<td>690</td>
<td>11852</td>
<td>2582</td>
<td>17.9%</td>
<td>1414</td>
<td>15848</td>
</tr>
<tr>
<td>objective_c</td>
<td>395</td>
<td>10562</td>
<td>1768</td>
<td>14.3%</td>
<td>2951</td>
<td>15281</td>
</tr>
<tr>
<td>automake</td>
<td>61</td>
<td>6014</td>
<td>853</td>
<td>12.4%</td>
<td>956</td>
<td>7823</td>
</tr>
<tr>
<td>perl</td>
<td>24</td>
<td>4111</td>
<td>1138</td>
<td>21.7%</td>
<td>732</td>
<td>5981</td>
</tr>
<tr>
<td>scheme</td>
<td>1</td>
<td>2775</td>
<td>153</td>
<td>5.2%</td>
<td>328</td>
<td>3256</td>
</tr>
<tr>
<td>ocaml</td>
<td>5</td>
<td>2482</td>
<td>538</td>
<td>17.8%</td>
<td>328</td>
<td>3348</td>
</tr>
<tr>
<td>python</td>
<td>6</td>
<td>1135</td>
<td>211</td>
<td>15.7%</td>
<td>220</td>
<td>1566</td>
</tr>
<tr>
<td>awk</td>
<td>9</td>
<td>1127</td>
<td>324</td>
<td>22.3%</td>
<td>193</td>
<td>1644</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>497</td>
<td>99</td>
<td>16.6%</td>
<td>30</td>
<td>626</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>haskell</td>
<td>48</td>
<td>149</td>
<td>0</td>
<td>0.0%</td>
<td>16</td>
<td>165</td>
</tr>
<tr>
<td>emacsclisp</td>
<td>1</td>
<td>59</td>
<td>21</td>
<td>26.2%</td>
<td>4</td>
<td>84</td>
</tr>
<tr>
<td>matlab</td>
<td>2</td>
<td>57</td>
<td>0</td>
<td>0.0%</td>
<td>7</td>
<td>64</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>50312</td>
<td>4938881</td>
<td>1567750</td>
<td>24.1%</td>
<td>1071986</td>
<td>7578617</td>
</tr>
</tbody>
</table>
### ohcount: Line Count of gcc-4.6.0

<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>18463</td>
<td>2100237</td>
<td>444333</td>
<td>17.5%</td>
<td>418292</td>
<td>2962862</td>
</tr>
<tr>
<td>cpp</td>
<td>22002</td>
<td>985076</td>
<td>229541</td>
<td>18.9%</td>
<td>214781</td>
<td>1429398</td>
</tr>
<tr>
<td>java</td>
<td>6342</td>
<td>681938</td>
<td>645505</td>
<td>48.6%</td>
<td>169046</td>
<td>1496489</td>
</tr>
<tr>
<td>ada</td>
<td>4605</td>
<td>680043</td>
<td>315956</td>
<td>31.7%</td>
<td>234467</td>
<td>1230466</td>
</tr>
<tr>
<td>autoconf</td>
<td>91</td>
<td>405461</td>
<td>509</td>
<td>0.1%</td>
<td>62914</td>
<td>468884</td>
</tr>
<tr>
<td>html</td>
<td>457</td>
<td>168355</td>
<td>5669</td>
<td>3.3%</td>
<td>38146</td>
<td>212170</td>
</tr>
<tr>
<td>make</td>
<td>98</td>
<td>121545</td>
<td>3659</td>
<td>2.9%</td>
<td>15618</td>
<td>140822</td>
</tr>
<tr>
<td>fortranfixed</td>
<td>2936</td>
<td>99413</td>
<td>1927</td>
<td>1.9%</td>
<td>13659</td>
<td>114999</td>
</tr>
<tr>
<td>shell</td>
<td>148</td>
<td>48032</td>
<td>10451</td>
<td>17.9%</td>
<td>6586</td>
<td>65069</td>
</tr>
<tr>
<td>assembler</td>
<td>208</td>
<td>46727</td>
<td>10227</td>
<td>18.0%</td>
<td>7853</td>
<td>64807</td>
</tr>
<tr>
<td>xml</td>
<td>75</td>
<td>36036</td>
<td>282</td>
<td>0.8%</td>
<td>3827</td>
<td>40145</td>
</tr>
<tr>
<td>objective_c</td>
<td>866</td>
<td>28014</td>
<td>5000</td>
<td>15.1%</td>
<td>8115</td>
<td>41129</td>
</tr>
<tr>
<td>fortranfree</td>
<td>821</td>
<td>13857</td>
<td>3147</td>
<td>18.5%</td>
<td>1695</td>
<td>18699</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>67</td>
<td>9440</td>
<td>1038</td>
<td>9.9%</td>
<td>1456</td>
<td>11934</td>
</tr>
<tr>
<td>perl</td>
<td>28</td>
<td>4445</td>
<td>1316</td>
<td>22.8%</td>
<td>837</td>
<td>6598</td>
</tr>
<tr>
<td>ocaml</td>
<td>6</td>
<td>2814</td>
<td>576</td>
<td>17.0%</td>
<td>378</td>
<td>3768</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>11</td>
<td>1740</td>
<td>396</td>
<td>18.5%</td>
<td>257</td>
<td>2393</td>
</tr>
<tr>
<td>python</td>
<td>10</td>
<td>1725</td>
<td>322</td>
<td>15.7%</td>
<td>383</td>
<td>2430</td>
</tr>
<tr>
<td>css</td>
<td>24</td>
<td>1589</td>
<td>143</td>
<td>8.3%</td>
<td>332</td>
<td>2064</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>4</td>
<td>341</td>
<td>87</td>
<td>20.3%</td>
<td>35</td>
<td>463</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>bat</td>
<td>3</td>
<td>7</td>
<td>0</td>
<td>0.0%</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>matlab</td>
<td>1</td>
<td>5</td>
<td>0</td>
<td>0.0%</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>Total</td>
<td>57359</td>
<td>5464598</td>
<td>1688150</td>
<td>23.6%</td>
<td>1202428</td>
<td>8355176</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>18463</td>
<td>2100237</td>
<td>444333</td>
<td>17.5%</td>
<td>418292</td>
<td>2962862</td>
</tr>
<tr>
<td>cpp</td>
<td>22002</td>
<td>985076</td>
<td>229541</td>
<td>18.9%</td>
<td>214781</td>
<td>1429398</td>
</tr>
<tr>
<td>java</td>
<td>6342</td>
<td>681938</td>
<td>645505</td>
<td>48.6%</td>
<td>169046</td>
<td>1496489</td>
</tr>
<tr>
<td>ada</td>
<td>4605</td>
<td>680043</td>
<td>315956</td>
<td>31.7%</td>
<td>234467</td>
<td>1230466</td>
</tr>
<tr>
<td>autoconf</td>
<td>91</td>
<td>405461</td>
<td>509</td>
<td>0.1%</td>
<td>62914</td>
<td>468884</td>
</tr>
<tr>
<td>html</td>
<td>457</td>
<td>168355</td>
<td>5669</td>
<td>3.3%</td>
<td>38146</td>
<td>212170</td>
</tr>
<tr>
<td>make</td>
<td>98</td>
<td>121545</td>
<td>3659</td>
<td>2.9%</td>
<td>15618</td>
<td>140822</td>
</tr>
<tr>
<td>fortranfixed</td>
<td>2936</td>
<td>99413</td>
<td>1927</td>
<td>1.9%</td>
<td>13659</td>
<td>114999</td>
</tr>
<tr>
<td>shell</td>
<td>148</td>
<td>48032</td>
<td>10451</td>
<td>17.9%</td>
<td>6586</td>
<td>65069</td>
</tr>
<tr>
<td>assembler</td>
<td>208</td>
<td>46727</td>
<td>10227</td>
<td>18.0%</td>
<td>7853</td>
<td>64807</td>
</tr>
<tr>
<td>xml</td>
<td>75</td>
<td>36036</td>
<td>282</td>
<td>0.8%</td>
<td>3827</td>
<td>40145</td>
</tr>
<tr>
<td>objective_c</td>
<td>866</td>
<td>28014</td>
<td>5000</td>
<td>15.1%</td>
<td>8115</td>
<td>41129</td>
</tr>
<tr>
<td>fortranfree</td>
<td>821</td>
<td>13857</td>
<td>3147</td>
<td>18.5%</td>
<td>1695</td>
<td>18699</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>67</td>
<td>9440</td>
<td>1038</td>
<td>9.9%</td>
<td>1456</td>
<td>11934</td>
</tr>
<tr>
<td>perl</td>
<td>28</td>
<td>4445</td>
<td>1316</td>
<td>22.8%</td>
<td>837</td>
<td>6598</td>
</tr>
<tr>
<td>ocaml</td>
<td>6</td>
<td>2814</td>
<td>576</td>
<td>17.0%</td>
<td>378</td>
<td>3768</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>11</td>
<td>1740</td>
<td>396</td>
<td>18.5%</td>
<td>257</td>
<td>2393</td>
</tr>
<tr>
<td>python</td>
<td>10</td>
<td>1725</td>
<td>322</td>
<td>15.7%</td>
<td>383</td>
<td>2430</td>
</tr>
<tr>
<td>css</td>
<td>24</td>
<td>1589</td>
<td>143</td>
<td>8.3%</td>
<td>332</td>
<td>2064</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>4</td>
<td>341</td>
<td>87</td>
<td>20.3%</td>
<td>35</td>
<td>463</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>bat</td>
<td>3</td>
<td>7</td>
<td>0</td>
<td>0.0%</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>matlab</td>
<td>1</td>
<td>5</td>
<td>0</td>
<td>0.0%</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>Total</td>
<td>57359</td>
<td>5464598</td>
<td>1688150</td>
<td>23.6%</td>
<td>1202428</td>
<td>8355176</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>bat</td>
<td>3</td>
<td>7</td>
<td>0</td>
<td>0.0%</td>
<td>0</td>
<td>7</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>bat</td>
<td>3</td>
<td>7</td>
<td>0</td>
<td>0.0%</td>
<td>0</td>
<td>7</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>13296</td>
<td>1254253</td>
<td>282582</td>
<td>18.4%</td>
<td>283766</td>
<td>1820601</td>
</tr>
<tr>
<td>ada</td>
<td>4196</td>
<td>636876</td>
<td>294321</td>
<td>31.6%</td>
<td>217401</td>
<td>1148598</td>
</tr>
<tr>
<td>cpp</td>
<td>7418</td>
<td>184186</td>
<td>52163</td>
<td>22.1%</td>
<td>54048</td>
<td>290397</td>
</tr>
<tr>
<td>fortranfixed</td>
<td>2086</td>
<td>67988</td>
<td>1521</td>
<td>2.2%</td>
<td>9079</td>
<td>78588</td>
</tr>
<tr>
<td>assembler</td>
<td>132</td>
<td>31092</td>
<td>7243</td>
<td>18.9%</td>
<td>4770</td>
<td>43105</td>
</tr>
<tr>
<td>autoconf</td>
<td>3</td>
<td>26996</td>
<td>10</td>
<td>0.0%</td>
<td>3383</td>
<td>30389</td>
</tr>
<tr>
<td>fortranfree</td>
<td>652</td>
<td>10898</td>
<td>2376</td>
<td>17.9%</td>
<td>1314</td>
<td>14588</td>
</tr>
<tr>
<td>objective_c</td>
<td>391</td>
<td>10155</td>
<td>1654</td>
<td>14.0%</td>
<td>2830</td>
<td>14639</td>
</tr>
<tr>
<td>make</td>
<td>3</td>
<td>5340</td>
<td>1027</td>
<td>16.1%</td>
<td>814</td>
<td>7181</td>
</tr>
<tr>
<td>scheme</td>
<td>1</td>
<td>2775</td>
<td>153</td>
<td>5.2%</td>
<td>328</td>
<td>3256</td>
</tr>
<tr>
<td>ocaml</td>
<td>5</td>
<td>2482</td>
<td>538</td>
<td>17.8%</td>
<td>328</td>
<td>3348</td>
</tr>
<tr>
<td>shell</td>
<td>16</td>
<td>2256</td>
<td>712</td>
<td>24.0%</td>
<td>374</td>
<td>3342</td>
</tr>
<tr>
<td>awk</td>
<td>7</td>
<td>1022</td>
<td>251</td>
<td>19.7%</td>
<td>187</td>
<td>1460</td>
</tr>
<tr>
<td>perl</td>
<td>1</td>
<td>772</td>
<td>205</td>
<td>21.0%</td>
<td>137</td>
<td>1114</td>
</tr>
<tr>
<td>haskell</td>
<td>48</td>
<td>149</td>
<td>0</td>
<td>0.0%</td>
<td>16</td>
<td>165</td>
</tr>
<tr>
<td>matlab</td>
<td>2</td>
<td>57</td>
<td>0</td>
<td>0.0%</td>
<td>7</td>
<td>64</td>
</tr>
<tr>
<td>Total</td>
<td>28258</td>
<td>2242738</td>
<td>647591</td>
<td>22.4%</td>
<td>579484</td>
<td>3469813</td>
</tr>
</tbody>
</table>
### ohcount: Line Count of gcc-4.6.0/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>15787</td>
<td>1462494</td>
<td>321820</td>
<td>18.0%</td>
<td>324179</td>
<td>2108493</td>
</tr>
<tr>
<td>ada</td>
<td>4595</td>
<td>678362</td>
<td>315396</td>
<td>31.7%</td>
<td>233868</td>
<td>1227626</td>
</tr>
<tr>
<td>cpp</td>
<td>8666</td>
<td>252213</td>
<td>61026</td>
<td>19.5%</td>
<td>67144</td>
<td>380383</td>
</tr>
<tr>
<td>fortranfixed</td>
<td>2850</td>
<td>93549</td>
<td>1878</td>
<td>2.0%</td>
<td>13260</td>
<td>108687</td>
</tr>
<tr>
<td>assembler</td>
<td>137</td>
<td>31548</td>
<td>7446</td>
<td>19.1%</td>
<td>4857</td>
<td>43851</td>
</tr>
<tr>
<td>autoconf</td>
<td>3</td>
<td>28775</td>
<td>12</td>
<td>0.0%</td>
<td>4020</td>
<td>32807</td>
</tr>
<tr>
<td>objective_c</td>
<td>861</td>
<td>27465</td>
<td>4822</td>
<td>14.9%</td>
<td>7967</td>
<td>40254</td>
</tr>
<tr>
<td>fortranfree</td>
<td>783</td>
<td>12903</td>
<td>2936</td>
<td>18.5%</td>
<td>1595</td>
<td>17434</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>4</td>
<td>6078</td>
<td>1070</td>
<td>15.0%</td>
<td>893</td>
<td>8041</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>2814</td>
<td>576</td>
<td>17.0%</td>
<td>378</td>
<td>3768</td>
</tr>
<tr>
<td>shell</td>
<td>16</td>
<td>1980</td>
<td>597</td>
<td>23.2%</td>
<td>338</td>
<td>2915</td>
</tr>
<tr>
<td>awk</td>
<td>9</td>
<td>1635</td>
<td>323</td>
<td>16.5%</td>
<td>251</td>
<td>2209</td>
</tr>
<tr>
<td>perl</td>
<td>3</td>
<td>866</td>
<td>225</td>
<td>20.6%</td>
<td>158</td>
<td>1249</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>1</td>
<td>5</td>
<td>0</td>
<td>0.0%</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>Total</td>
<td>33777</td>
<td>2617304</td>
<td>721972</td>
<td>21.6%</td>
<td>660832</td>
<td>4000108</td>
</tr>
</tbody>
</table>

---

**Essential Abstractions in GCC**

**GCC Resource Center, IIT Bombay**
### 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><strong>Total</strong></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 *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 retargetability 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

Essential Abstractions in GCC

GCC Resource Center, IIT Bombay
The Architecture of GCC

The Architecture of GCC

Input Language

Compiler Generation Framework

Language Specific Code

Language and Machine Independent
Generic Code

Machine Dependent Generator Code

Machine Descriptions

Selected

Copied

Copied

Copied

Generated

Generated

Generated

Source Program

Assembly Program

Essential Abstractions in GCC

GCC Resource Center, IIT Bombay
The Architecture of GCC

- **Input Language**
- **Target Name**

**Compiler Generation Framework**

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

**Generated Compiler (cc1)**

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

**Development Time**

**Build Time**

**Use Time**

Essential Abstractions in GCC

GCC Resource Center, 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

```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

```plaintext
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 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>s-oscons-tmplt.c</td>
<td>238</td>
<td>main (void )</td>
</tr>
<tr>
<td>collect2.c</td>
<td>1021</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>divtab-sh4-300.c</td>
<td>31</td>
<td>main ()</td>
</tr>
<tr>
<td>divtab-sh4.c</td>
<td>30</td>
<td>main ()</td>
</tr>
<tr>
<td>divtab.c</td>
<td>131</td>
<td>main ()</td>
</tr>
<tr>
<td>gen-mul-tables.cc</td>
<td>1224</td>
<td>main ()</td>
</tr>
<tr>
<td>vms-ar.c</td>
<td>122</td>
<td>main (int argc, char *argv[])</td>
</tr>
<tr>
<td>vms-ld.c</td>
<td>559</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-ar.c</td>
<td>36</td>
<td>main (int ac, char **av)</td>
</tr>
<tr>
<td>gcc.c</td>
<td>6105</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>gcov-dump.c</td>
<td>78</td>
<td>main (int argc ATTRIBUTE_UNUSED, char *)</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>397</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>genattr-common.c</td>
<td>64</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>genattr.c</td>
<td>141</td>
<td>main (int argc, char **argv)</td>
</tr>
</tbody>
</table>

---

Essential Abstractions in GCC

GCC Resource Center, IIT Bombay
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)
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

Copied

Generated

Generated

Generated

Parsed

Gimplifier

Tree SSA Optimizer

Expander

Optimizer

Recognizer

Development Time

Build Time

Use Time

Essential Abstractions in GCC

GCC Resource Center, IIT Bombay
**GCC Retargetability Mechanism**

The GCC Retargetability Mechanism involves several key components and steps in its compiler generation framework:

- **Input Language**
- **Target Name**

The process can be summarized as follows:

1. **Input Language**
   - Language Specific Code
   - Language and Machine Independent Generic Code

2. **Target Name**
   - Machine Dependent Generator Code
   - Machine Descriptions

3. **Compiler Generation Framework**
   - Parser
   - Gimplifier
   - Tree SSA Optimizer
   - Expander
   - Optimizer
   - Recognizer

4. **Generated Compiler**
   - Development Time
   - Build Time
   - Use Time

The essential abstractions in GCC include:

- **Gimple → IR-RTL**
- **IR-RTL → ASM**
The Great Compiler Challenge 39/42

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

Generated

Parser

Gimplifier

Tree SSA Optimizer

Expander

Optimizer

Recognizer

Generated Compiler

Essential Abstractions in GCC

Gimplifier → PN

PN → IR-RTL

IR-RTL → ASM

Gimplifier → IR-RTL

IR-RTL → ASM
 GCC Retargetability Mechanism

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

**Compiler Generation Framework**
- Parser
- Gimplifier
- Tree SSA Optimizer
- Expander
- Optimizer
- Recognizer

**Selected**
- Copied
- Generated

**Target Name**
- Development Time
- Build Time
- Use Time

**Generated Compiler**
- Gimple $\rightarrow$ PN
- PN $\rightarrow$ IR-RTL
- IR-RTL $\rightarrow$ ASM
- Gimple $\rightarrow$ IR-RTL
- IR-RTL $\rightarrow$ ASM

**Essential Abstractions in GCC**
GCC Resource Center, IIT Bombay
GCC Retargetability Mechanism

Essential Abstractions in GCC

GCC Resource Center, IIT Bombay
The generated compiler uses an adaptation of the Davison 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 in gcc-4.7.2 (counted using wc -l)

<table>
<thead>
<tr>
<th>Files</th>
<th>i386</th>
<th>mips</th>
<th>arm</th>
</tr>
</thead>
<tbody>
<tr>
<td>*.md</td>
<td>39582</td>
<td>16347</td>
<td>32385</td>
</tr>
<tr>
<td>*.c</td>
<td>41985</td>
<td>17761</td>
<td>26006</td>
</tr>
<tr>
<td>*.h</td>
<td>19174</td>
<td>5586</td>
<td>18012</td>
</tr>
<tr>
<td>Total</td>
<td>100741</td>
<td>39694</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 in gcc-4.7.2 (counted using `wc -l`)

<table>
<thead>
<tr>
<th>Files</th>
<th>i386</th>
<th>mips</th>
<th>arm</th>
</tr>
</thead>
<tbody>
<tr>
<td>*.md</td>
<td>39582</td>
<td>16347</td>
<td>32385</td>
</tr>
<tr>
<td>*.c</td>
<td>41985</td>
<td>17761</td>
<td>26006</td>
</tr>
<tr>
<td>*.h</td>
<td>19174</td>
<td>5586</td>
<td>18012</td>
</tr>
<tr>
<td>Total</td>
<td>100741</td>
<td>39694</td>
<td>76403</td>
</tr>
</tbody>
</table>

- Machine descriptions are difficult to construct, understand, debug, and enhance
# 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>Customizing the configuration and building</td>
<td>Yes</td>
</tr>
<tr>
<td>Retargetability issues and machine descriptions</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>

<table>
<thead>
<tr>
<th></th>
<th>Makefiles</th>
<th>Source</th>
<th>MD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Translation sequence of programs</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Build process</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Retargetability issues and machine descriptions</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>IR data structures and access mechanisms</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Retargetability mechanism</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Workshop Coverage

Compiler Specifications

Compiler Generator

Generated Compiler
Workshop Coverage

Compiler Specifications

- Compiler Generator

External View

Generated Compiler

Internal View
Workshop Coverage

External View

Internal View

Compiler Specifications

Compiler Generator

Generated Compiler

Gray box probing

Essential Abstractions in GCC

GCC Resource Center, IIT Bombay
Workshop Coverage

<table>
<thead>
<tr>
<th>Compiler Specifications</th>
<th>External View</th>
<th>Internal View</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compiler Generator</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Generated Compiler</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Gray box probing
Pass structure and IR

Essential Abstractions in GCC

GCC Resource Center, IIT Bombay
Workshop Coverage

Compiler Specifications

External View

Configuration and building

Internal View

Compiler Generator

Gray box probing
Pass structure and IR

Generated Compiler

Essential Abstractions in GCC

GCC Resource Center, IIT Bombay
Workshop Coverage

- Compiler Specifications
  - Compiler Generator
  - Generated Compiler

External View
- Front end hooks
- Configuration and building
- Gray box probing
  - Pass structure and IR

Internal View
Workshop Coverage

Compiler Specifications

Compiler Generator

Generated Compiler

External View

Configuration and building

Gray box probing
Pass structure and IR

Internal View

Front end hooks

Pass structure

Essential Abstractions in GCC

GCC Resource Center, IIT Bombay
Workshop Coverage

- **External View**
  - Configuration and building
  - Gray box probing
  - Pass structure and IR

- **Internal View**
  - Front end hooks
  - Pass structure
  - Control flow

- **Compiler Specifications**
- **Compiler Generator**
- **Generated Compiler**
Workshop Coverage

- Compiler Specifications
- Compiler Generator
- Generated Compiler
- External View
  - Configuration and building
  - Gray box probing
  - Pass structure and IR
- Internal View
  - Front end hooks
  - Pass structure
  - Control flow
  - Static and dynamic plugin mechanisms

Essential Abstractions in GCC

GCC Resource Center, IIT Bombay
Workshop Coverage

Compiler Specifications

- Compiler Generator
  - Configuration and building
  - Gray box probing
  - Data Flow Analysis

- Front end hooks
  - Pass structure
  - Control flow
  - Static and dynamic plugin mechanisms

External View

- Generated Compiler

Internal View

Essential Abstractions in GCC

GCC Resource Center, IIT Bombay
Overview: GCC ≡ The Great Compiler Challenge

**Workshop Coverage**

<table>
<thead>
<tr>
<th>External View</th>
<th>Internal View</th>
</tr>
</thead>
<tbody>
<tr>
<td>Machine descriptions</td>
<td>Front end hooks</td>
</tr>
</tbody>
</table>

**Compiler Specifications**

- Compiler Generator
  - Generated Compiler
    - Gray box probing
      - Pass structure and IR
        - Data Flow Analysis
    - Configuration and building
    - Control flow
      - Pass structure
      - Static and dynamic plugin mechanisms
Workshop Coverage

Compiler Specifications

- Machine descriptions
- Configuration and building
- Front end hooks
- Retargetability mechanism
- Gray box probing
- Pass structure and IR
- Data Flow Analysis
- Pass structure
- Control flow
- Static and dynamic plugin mechanisms