CS 715: The Design and Implementation of Gnu Compiler Generation Framework

Uday Khedker

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

January 2011
Outline

- An Overview of Compilation
  Introduction, compilation sequence, compilation models

- GCC: The Great Compiler Challenge
  Difficulties in understanding GCC

- Meeting the GCC Challenge: CS 715
  The course plan
Part 1

Introduction to Compilation
Nothing is known except the problem

No. of unbound objects

Time
**Binding**

- No. of unbound objects
- Conceptualisation
- Overall strategy, algorithm, data structures etc.
- Time
Binding

Functions, variables, their types etc.

No. of unbound objects

Conceptualisation  Coding

Time

Uday Khedker

GRC, IIT Bombay
Binding

No. of unbound objects

Conceptualisation  Coding  Compiling

Time

Machine instructions, registers etc.
Binding

- No. of unbound objects
- Addresses of functions, external data etc.

Conceptualisation | Coding | Compiling | Linking

Time
Binding

- Conceptualisation
- Coding
- Compiling
- Linking
- Loading

No. of unbound objects vs Time

Actual addresses of code and data
Binding

Values of variables

No. of unbound objects

Conceptualisation  Coding  Compiling  Linking  Loading  Execution

Time
Implementation Mechanisms

Source Program

Translator

Target Program

Machine
Implementation Mechanisms

Source Program

Translator

Target Program

Machine

Input Data
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

Diagram:
- Program Specification
- Machine
Implementation Mechanisms as “Bridges”

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

Program Specification

Translation

Machine
Implementation Mechanisms as “Bridges”

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

Program Specification

Translation

Interpretation

Machine
Implementation Mechanisms as “Bridges”

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

Program Specification

Translation

Interpretation

Machine

State: Variables
Operations: Expressions, Control Flow

State: Memory, Registers
Operations: Machine Instructions
High and Low Level Abstractions

Input C statement

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

Spim Assembly Equivalent

\[
\begin{align*}
lw & \quad \$t0, 4($fp) \; ; \quad t0 \leftarrow b \quad \# \text{ Is } b \text{ smaller} \\
slti & \quad \$t0, \$t0, 10 \; ; \quad t0 \leftarrow t0 < 10 \quad \# \text{ than } 10? \\
not & \quad \$t0, \$t0 \; ; \quad t0 \leftarrow \neg t0 \\
bgtz & \quad \$t0, L0: \; ; \quad \text{if } t0 \geq 0 \text{ goto L0} \\
lw & \quad \$t0, 4($fp) \; ; \quad t0 \leftarrow b \quad \# \text{ YES} \\
b & \quad \text{L1:} \quad \text{goto L1} \\
L0: & \quad lw \quad \$t0, 8($fp) \; ; \quad L0: \quad t0 \leftarrow c \quad \# \text{ NO} \\
L1: & \quad sw \quad 0($fp), \$t0 \; ; \quad L1: \quad a \leftarrow t0
\end{align*}
\]
High and Low Level Abstractions

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

Spim Assembly Equivalent

```
lw $t0, 4($fp) ; t0 <- b  # Is b smaller
slti $t0, $t0, 10 ; t0 <- t0 < 10  # than 10?
not $t0, $t0 ; t0 <- !t0
bgtz $t0, L0: ; if t0>=0 goto L0
lw $t0, 4($fp) ; t0 <- b  # YES
b L1: ; goto L1
L0: lw $t0, 8($fp); L0: t0 <- c  # NO
L1: sw 0($fp), $t0 ; L1: a <- t0
```
High and Low Level Abstractions

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

Spim Assembly Equivalent

- lw $t0, 4($fp) ; \quad t0 \leftarrow b \quad # \text{Is } b \text{ smaller}
- slti $t0, $t0, 10 ; \quad t0 \leftarrow t0 < 10 \quad # \text{than } 10?\n- not $t0, $t0 ; \quad t0 \leftarrow \neg t0
- bgtz $t0, L0: ; \quad \text{if } t0 \geq 0 \text{ goto L0}
- lw $t0, 4($fp) ; \quad t0 \leftarrow b \quad # \text{YES}
- \text{b } L1: ; \quad \text{goto L1}
- \text{L0: lw } $t0, 8($fp) ; L0: t0 \leftarrow c \quad # \text{NO}
- \text{L1: sw } 0($fp), $t0 ; L1: a \leftarrow t0

NOT Condition

True Part

False Part
High and Low Level Abstractions

Input C statement

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

Spim Assembly Equivalent

\[
\begin{align*}
\text{lw} & \quad \text{t0} \leftarrow b \quad \text{# Is b smaller} \\
\text{slti} & \quad \text{t0} \leftarrow \text{t0} < 10 \quad \text{# than 10?} \\
\text{not} & \quad \text{t0} \leftarrow \neg \text{t0} \\
\text{bgtz} & \quad \text{if t0} \geq 0 \text{ goto L0} \\
\text{lw} & \quad \text{t0} \leftarrow b \quad \text{# YES} \\
\text{b} & \quad \text{goto L1} \\
\text{L0: lw} & \quad \text{t0} \leftarrow c \quad \text{# NO} \\
\text{L1: sw} & \quad 0(ffp), \text{t0} \quad \text{# L1: a <- t0}
\end{align*}
\]
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 $\rightarrow$ Equivalent Instructions
- Interpretation Instructions $\rightarrow$ Actions Implied by Instructions
Language Implementation Models

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

- Front End
- Optimizer
- Back End
- Virtual Machine

Languages:
- C, C++
- Java
- C#
Typical Front Ends

Parser
Typical Front Ends

Source Program

Scanner

Parser

Tokens
Typical Front Ends

Source Program → Scanner → Tokens

Scanner → AST or Linear IR + Symbol Table → Semantic Analyzer

Parser → AST → Parse Tree

Parser → AST
Typical Front Ends

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

Parser

Symtab Handler → Scanner

Error Handler → Semantic Analyzer
Typical Back Ends

- Compile time evaluations
- Eliminating redundant computations
Typical Back Ends

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

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

Assembly Code
Typical Back Ends

- Compile time evaluations
- Eliminating redundant computations

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

Assembly Code
Part 2

An Overview of Compilation Phases
The Structure of a Simple Compiler

Parser

Scanner
Semantic Analyser
Symtab Handler

Source Program
The Structure of a Simple Compiler
The Structure of a Simple Compiler

Front End

- Parser
- Scanner
- Semantic Analyser
- Syntab Handler
- Source Program

Back End

- Instruction Selector
- Register Allocator
- Assembly Emitter
- Insn
- Assembly Program
Translation Sequence in Our Compiler: Parsing

Input

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

Uday Khedker

GRC, IIT Bombay
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

Input

Parse Tree
Translation Sequence in Our Compiler: Semantic Analysis

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

Input

AsgnStmnt

\[ \text{Lhs} \quad \rightarrow \quad \text{E} \quad ; \]

\[ \text{name} \quad \rightarrow \quad \text{E} \quad ? \quad \text{E} \quad : \quad \text{E} \]

\[ \text{E} \quad \rightarrow \quad \text{name} \quad \text{E} \quad \text{name} \quad \text{num} \]

\[ \text{name} \quad \rightarrow \quad \text{name} \quad (a, \text{int}) \quad ? \quad (\text{int}) \quad < \quad (\text{bool}) \quad \text{name} \quad (b, \text{int}) \quad \text{name} \quad (c, \text{int}) \]

\[ \text{name} \quad \rightarrow \quad \text{name} \quad (b, \text{int}) \quad \text{num} \quad (10, \text{int}) \]

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

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

Input

### Parse Tree

- **AsgnStmt**
  - **Lhs** = **E** ;
  - **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** (**b**, **int**) **num** (**10**, **int**)
Translation Sequence in Our Compiler: IR Generation

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

**Input**

**Tree List**

```
=  
T0  b  >  10

IfGoto
Not  L0:
T0

=  
T1  b

Goto  L1:
L0:  =  
T1  c

L1:  =  
T1  a
```

**Parse Tree**

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

**Abstract Syntax Tree (with attributes)**

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

**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

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

Input

Tree List

```
T0 = b < 10
T0 ifGoto
   Not L0:
   L0:
T1 = b
T1 goto L1:
L0: T1 = c
L1: a = T1

Parse Tree

Abstract Syntax Tree (with attributes)

Uday Khedker
Translation Sequence in Our Compiler: Instruction Selection

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

- **Input**
- **Tree List**
- **Parse Tree**
- **Abstract Syntax Tree (with attributes)**
- **Instruction List**
- **Issues:**
  - Cover trees with as few machine instructions as possible
  - Use temporaries and local registers
Translation Sequence in Our Compiler: Emitting Instructions

Input

Tree List

AsgnStmtnt

Parse Tree

Abstract Syntax Tree (with attributes)

Instruction List

T₀ ← b
T₀ ← T₀ < 10
T₀ ← ! T₀
if T₀ > 0 goto L0:
T₁ ← b
goto L1:
L0: T₁ ← c
L1: a ← T₁
Translation Sequence in Our Compiler: Emitting Instructions

Input

AsgnStmt

Lhs = E
E ? E : E
E < E
name name name
num
Parse Tree

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

Abstract Syntax Tree
(with attributes)

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 ← b
T_0 ← T_0 < 10
T_0 ← ! T_0
if T_0 > 0 goto L0:
T_1 ← b
goto L1:
L0: T_1 ← c
L1: a ← T_1

AsmList

lw $t0, 4($fp)
slti $t0, $t0, 10
not $t0, $t0
bgtz $t0, L0:
lw $t0, 4($fp)
b L1:
L0: lw $t0, 8($fp)
L1: sw 0($fp), $t0

Uday Khedker
GRC, IIT Bombay
Part 3

Compilation Models, Instruction Selection, and Retargetability
Compilation Models

Aho Ullman Model

Davidson Fraser Model
Compilation Models

Aho Ullman Model

Davidson Fraser Model

Input Source Program

Front End → AST
Compilation Models

Aho Ullman Model

- Front End
- AST
- Optimizer
- m/c indep. IR

Davidson Fraser Model

- Input Source Program
Compilation Models

Aho Ullman Model

Front End

AST

Optimizer

m/c indep. IR

Code Generator

Target Program

Davidson Fraser Model

Input Source Program
Compilation Models

Aho Ullman Model

Front End

\rightarrow

AST

\rightarrow

Optimizer

\rightarrow

m/c indep. IR

\rightarrow

Code Generator

\rightarrow

Target Program

Davidson Fraser Model

Input Source Program

\rightarrow

Front End

\rightarrow

AST

Uday Khedker
Compilation Models

**Aho Ullman Model**

1. **Front End**
   - AST
   - Optimizer
   - m/c indep. IR
   - Code Generator
   - Target Program

**Davidson Fraser Model**

1. **Front End**
   - AST
   - Expander
   - register transfers

**Input Source Program**

Uday Khedker
GRC, IIT Bombay
Compilation Models

**Aho Ullman Model**
- Front End
  - AST
  - Optimizer
    - m/c indep. IR
    - Code Generator
  - Target Program

**Davidson Fraser Model**
- Input Source Program
  - Front End
    - AST
    - Expander
      - register transfers
    - Optimizer
      - register transfers
Compilation Models

**Aho Ullman Model**

1. **Front End**
2. **AST**
3. **Optimizer**
4. **m/c indep. IR**
5. **Code Generator**
6. **Target Program**

**Davidson Fraser Model**

1. **Input Source Program**
2. **Front End**
3. **AST**
4. **Expander**
5. **Optimizer**
6. **Recognizer**
7. **register transfers**
8. **Target Program**
Compilation Models

Aho Ullman Model

Front End

AST

Optimizer

m/c indep. IR

Code Generator

Target Program

Aho Ullman: Instruction selection
- over optimized IR using
- intelligent tree tiling based algorithms

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

Davidson Fraser Model

Front End

AST

Expander

Optimizer

register transfers

Recognizer

Target Program

Uday Khedker

GRC, IIT Bombay
Retargetability in Aho Ullman Model

Instruction selection
- over optimized IR using
- intelligent tree tiling based algorithms

Key idea in retargetability:
- Machine independent IR is expressed in the form of trees
- Machine instructions are described in the form of trees
- Trees in the IR are tiled using the instruction trees
Retargetability in Davidson Fraser Model

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

Key idea in retargetability:
- Register transfers are machine specific but
- their form is machine independent

Register transfers
Instructions are viewed as independent non-composable rules

<table>
<thead>
<tr>
<th>Machine Instructions</th>
<th>Subject Tree (IR)</th>
<th>Modified Trees</th>
</tr>
</thead>
<tbody>
<tr>
<td>reg = reg + reg</td>
<td></td>
<td></td>
</tr>
<tr>
<td>reg = reg * reg</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Full Tree Matching (Davidson Fraser Model)

Instructions are viewed as independent non-composable rules

<table>
<thead>
<tr>
<th>Machine Instructions</th>
<th>Subject Tree (IR)</th>
<th>Modified Trees</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><img src="tree1.png" alt="Tree Diagram" /></td>
<td><img src="tree2.png" alt="Tree Diagram" /></td>
</tr>
<tr>
<td></td>
<td><img src="tree3.png" alt="Tree Diagram" /></td>
<td><img src="tree4.png" alt="Tree Diagram" /></td>
</tr>
</tbody>
</table>
Full Tree Matching (Davidson Fraser Model)

Instructions are viewed as independent non-composable rules

<table>
<thead>
<tr>
<th>Machine Instructions</th>
<th>Subject Tree (IR)</th>
<th>Modified Trees</th>
</tr>
</thead>
<tbody>
<tr>
<td>reg = reg + reg</td>
<td>reg = a + b * c</td>
<td></td>
</tr>
<tr>
<td>reg = reg * reg</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Full Tree Matching (Davidson Fraser Model)**

Instructions are viewed as independent non-composable rules.

<table>
<thead>
<tr>
<th>Machine Instructions</th>
<th>Subject Tree (IR)</th>
<th>Modified Trees</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><img src="image" alt="Subject Tree" /></td>
<td><img src="image" alt="Modified Trees" /></td>
</tr>
</tbody>
</table>

In the diagram, the subject tree (IR) is matched against the machine instructions. The `=` instruction in the machine instructions is matched to the `=` in the subject tree, and the `+` and `*` instructions are also matched accordingly. The modified trees show the final transformed expressions.

Uday Khedker
Full Tree Matching (Davidson Fraser Model)

Instructions are viewed as independent non-composable rules

<table>
<thead>
<tr>
<th>Machine Instructions</th>
<th>Subject Tree (IR)</th>
<th>Modified Trees</th>
</tr>
</thead>
<tbody>
<tr>
<td>= \text{reg} + \text{reg}</td>
<td>\text{a} + \text{b} = \text{c} \ast \text{t}</td>
<td>\text{a} + \text{t} \ast \text{b} \ast \text{c}</td>
</tr>
<tr>
<td>= \text{reg} \ast \text{reg} \ast \text{reg}</td>
<td>\text{a} \ast \text{b} = \text{c} \ast \text{t}</td>
<td>\text{a} \ast \text{t} \ast \text{b} \ast \text{c}</td>
</tr>
</tbody>
</table>

Uday Khedker
GRC, IIT Bombay
**Full Tree Matching (Davidson Fraser Model)**

Instructions are viewed as independent non-composable rules

<table>
<thead>
<tr>
<th>Machine Instructions</th>
<th>Subject Tree (IR)</th>
<th>Modified Trees</th>
</tr>
</thead>
<tbody>
<tr>
<td>reg + reg = reg</td>
<td>a + b = t * b c</td>
<td>t = b * c</td>
</tr>
<tr>
<td>reg * reg = reg</td>
<td>a * b = a + t a</td>
<td>(red)</td>
</tr>
<tr>
<td>reg * reg = reg</td>
<td>a + t a = a * b c</td>
<td>(blue)</td>
</tr>
</tbody>
</table>

---

Uday Khedker
GRC, IIT Bombay
### Tree Tiling (Aho Ullman Model)

Instructions are viewed as composable rules

<table>
<thead>
<tr>
<th>Machine Instructions</th>
<th>Subject Tree (IR)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{reg} \rightarrow \text{Reg}$</td>
<td>![Tree Diagram]</td>
</tr>
<tr>
<td>$\text{reg}$</td>
<td>![Tree Diagram]</td>
</tr>
<tr>
<td>$\text{Reg} + \text{Reg}$</td>
<td>![Tree Diagram]</td>
</tr>
<tr>
<td>$\text{Reg} \ast \text{Reg}$</td>
<td>![Tree Diagram]</td>
</tr>
</tbody>
</table>
Tree Tiling (Aho Ullman Model)

Instructions are viewed as composable rules

<table>
<thead>
<tr>
<th>Machine Instructions</th>
<th>Subject Tree (IR)</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>=</code></td>
<td><code>reg</code> =&gt; <code>Reg</code></td>
</tr>
<tr>
<td><code>Reg ← reg</code></td>
<td></td>
</tr>
<tr>
<td><code>Reg ← +</code></td>
<td><code>Reg</code> =&gt; <code>Reg</code></td>
</tr>
<tr>
<td><code>Reg ← *</code></td>
<td><code>Reg</code> =&gt; <code>Reg</code></td>
</tr>
</tbody>
</table>
Tree Tiling (Aho Ullman Model)

Instructions are viewed as composable rules

<table>
<thead>
<tr>
<th>Machine Instructions</th>
<th>Subject Tree (IR)</th>
</tr>
</thead>
<tbody>
<tr>
<td>reg = Reg</td>
<td></td>
</tr>
<tr>
<td>Reg ← reg</td>
<td></td>
</tr>
<tr>
<td>Reg ← + Reg</td>
<td></td>
</tr>
<tr>
<td>Reg ← * Reg</td>
<td></td>
</tr>
</tbody>
</table>

\[ \text{Reg} \leftarrow \text{reg} \]
\[ \text{Reg} \leftarrow \text{+ Reg Reg} \]
\[ \text{Reg} \leftarrow \text{* Reg Reg} \]

\[ = \]
\[ a + b \]
\[ a \times c \]
Tree Tiling (Aho Ullman Model)

Instructions are viewed as composable rules

<table>
<thead>
<tr>
<th>Machine Instructions</th>
<th>Subject Tree (IR)</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>=</td>
</tr>
<tr>
<td>reg</td>
<td>a</td>
</tr>
<tr>
<td>Reg</td>
<td>+</td>
</tr>
<tr>
<td>=</td>
<td>*</td>
</tr>
<tr>
<td>reg</td>
<td>a</td>
</tr>
<tr>
<td>Reg</td>
<td>b</td>
</tr>
<tr>
<td>Reg</td>
<td>c</td>
</tr>
<tr>
<td>Reg</td>
<td>*</td>
</tr>
</tbody>
</table>
**Tree Tiling (Aho Ullman Model)**

Instructions are viewed as composable rules

<table>
<thead>
<tr>
<th>Machine Instructions</th>
<th>Subject Tree (IR)</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>=</td>
</tr>
<tr>
<td>reg Reg</td>
<td>a + Reg * Reg</td>
</tr>
<tr>
<td>reg</td>
<td></td>
</tr>
<tr>
<td>+</td>
<td></td>
</tr>
<tr>
<td>Reg Reg</td>
<td></td>
</tr>
<tr>
<td>*</td>
<td></td>
</tr>
<tr>
<td>Reg</td>
<td></td>
</tr>
</tbody>
</table>

= Equation sign, reg Register, Reg Reservation, a Variable, + Addition, * Multiplication.
Tree Tiling (Aho Ullman Model)

Instructions are viewed as composable rules

<table>
<thead>
<tr>
<th>Machine Instructions</th>
<th>Subject Tree (IR)</th>
</tr>
</thead>
<tbody>
<tr>
<td>reg Reg</td>
<td>a + Reg Reg</td>
</tr>
<tr>
<td>Reg reg</td>
<td></td>
</tr>
<tr>
<td>Reg + Reg Reg</td>
<td></td>
</tr>
<tr>
<td>Reg * Reg Reg</td>
<td></td>
</tr>
</tbody>
</table>
Tree Tiling (Aho Ullman Model)

Instructions are viewed as composable rules

<table>
<thead>
<tr>
<th>Machine Instructions</th>
<th>Subject Tree (IR)</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>=</td>
</tr>
<tr>
<td>reg Reg</td>
<td>a +</td>
</tr>
<tr>
<td></td>
<td>Reg Reg</td>
</tr>
<tr>
<td>Reg ← reg</td>
<td></td>
</tr>
<tr>
<td>Reg ← +</td>
<td></td>
</tr>
<tr>
<td>Reg Reg</td>
<td></td>
</tr>
<tr>
<td>Reg ← *</td>
<td></td>
</tr>
<tr>
<td>Reg Reg</td>
<td></td>
</tr>
</tbody>
</table>

Subject Tree (IR):

- **=**
  - **reg**
  - **Reg**

- **a +**
  - **Reg**
  - **Reg**

- **Reg**
  - **Reg**
  - **Reg**

- **Reg**
  - **Reg**
  - **Reg**
Tree Tiling (Aho Ullman Model)

Instructions are viewed as composable rules

<table>
<thead>
<tr>
<th>Machine Instructions</th>
<th>Subject Tree (IR)</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>=</td>
</tr>
<tr>
<td>reg Reg</td>
<td>reg a +</td>
</tr>
<tr>
<td>Reg ← reg</td>
<td>Reg Reg</td>
</tr>
<tr>
<td>Reg ← + Reg Reg</td>
<td></td>
</tr>
<tr>
<td>Reg ← * Reg Reg</td>
<td></td>
</tr>
</tbody>
</table>

Machine Instructions: Composed rules for machine instructions.
Subject Tree (IR): Intermediate representation for compiler transformation.
Instructions are viewed as composable rules

<table>
<thead>
<tr>
<th>Machine Instructions</th>
<th>Subject Tree (IR)</th>
</tr>
</thead>
<tbody>
<tr>
<td>reg Reg</td>
<td>=</td>
</tr>
<tr>
<td>Reg ← reg</td>
<td>a Reg</td>
</tr>
<tr>
<td>Reg ← + Reg Reg</td>
<td></td>
</tr>
<tr>
<td>Reg ← * Reg Reg</td>
<td></td>
</tr>
</tbody>
</table>
Tree Tiling (Aho Ullman Model)

Instructions are viewed as composable rules

<table>
<thead>
<tr>
<th>Machine Instructions</th>
<th>Subject Tree (IR)</th>
</tr>
</thead>
<tbody>
<tr>
<td>reg Reg = Reg</td>
<td>reg a Reg</td>
</tr>
<tr>
<td>Reg ← reg</td>
<td></td>
</tr>
<tr>
<td>Reg ← + Reg Reg</td>
<td></td>
</tr>
<tr>
<td>Reg ← * Reg Reg</td>
<td></td>
</tr>
</tbody>
</table>
Part 4

GCC ≡ The Great Compiler Challenge
The Gnu Tool Chain

Source Program

gcc

Target Program
The Gnu Tool Chain

Source Program

gcc

Target Program

cc1
The Gnu Tool Chain

Source Program

gcc

Target Program

cc1

cpp
The Gnu Tool Chain

Source Program

→

gcc

→

cc1

←

cpp

←

as

→

Target Program
The Gnu Tool Chain

Source Program

gcc

Target Program

cc1

cpp

as

ld
The Gnu Tool Chain

Source Program

gcc

Target Program

cc1

cpp

as

ld

glibc/newlib
The Gnu Tool Chain

Source Program

gcc

cc1

cpp

as

GCC

glibc/newlib

Id

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
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
    - Input-output of the generated compiler
  - 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
Comprehensiveness of GCC: Wide Applicability

- **Input languages supported:**
  C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada
- **Processors supported in standard releases:**
Comprehensiveness of GCC: Wide Applicability

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

- **Processors supported in standard releases:**
  - **Common processors:**
  
  - **Lesser-known target processors:**

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

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

• Processors supported in standard releases:
  ▶ Common processors:
    Alpha,

  ▶ Lesser-known target processors:

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

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

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

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

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

• Processors supported in standard releases:
  ▶ Common processors:
    Alpha, ARM, Atmel AVR,

  ▶ Lesser-known target processors:

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

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

- Processors supported in standard releases:
  ▶ Common processors:
    Alpha, ARM, Atmel AVR, Blackfin,

  ▶ Lesser-known target processors:

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

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

- **Processors supported in standard releases:**
  - **Common processors:**
    Alpha, ARM, Atmel AVR, Blackfin, HC12,
  - **Lesser-known target processors:**
  - **Additional processors independently supported:**
Comprehensiveness of GCC: Wide Applicability

- Input languages supported:
  - C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada
- Processors supported in standard releases:
  - Common processors:
    - Alpha, ARM, Atmel AVR, Blackfin, HC12, H8/300,
  - Lesser-known target processors:
  - Additional processors independently supported:
Comprehensiveness of GCC: Wide Applicability

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

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

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

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

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

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

  - Lesser-known target processors:

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

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

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

  - **Lesser-known target processors:**

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

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

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

  - **Lesser-known target processors:**

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

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

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

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

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

  - **Lesser-known target processors:**

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

- **Input languages supported:**
  C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada
- **Processors supported in standard releases:**
  - **Common processors:**
    Alpha, ARM, Atmel AVR, Blackfin, HC12, H8/300, IA-32 (x86), x86-64, IA-64, Motorola 68000, MIPS, PA-RISC, PDP-11, PowerPC, R8C/M16C/M32C, SPU, System/390/zSeries, SuperH, SPARC, VAX
  - **Lesser-known target processors:**
    A29K, ARC, ETRAX CRIS, D30V, DSP16xx,
  - **Additional processors independently supported:**
Comprehensiveness of GCC: Wide Applicability

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

  - **Lesser-known target processors:**
    
    A29K, ARC, ETRAX CRIS, D30V, DSP16xx, FR-30, FR-V, Intel i960, IP2000, M32R, 68HC11, MCORE, MMIX, MN10200, MN10300, Motorola 88000, NS32K, ROMP,

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

  ▶ **Lesser-known target processors:**

  ▶ **Additional processors independently supported:**
  D10V, LatticeMico32, MeP, Motorola 6809, MicroBlaze,
Comprehensiveness of GCC: Wide Applicability

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

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

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

- **Processors supported in standard releases:**

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

  - **Lesser-known target processors:**

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

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

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

  ▶ **Lesser-known target processors:**
  

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

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

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

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

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

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

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

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

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

- **Overall size**

<table>
<thead>
<tr>
<th></th>
<th>Subdirectories</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>gcc-4.4.2</td>
<td>3794</td>
<td>62301</td>
</tr>
<tr>
<td>gcc-4.5.0</td>
<td>4056</td>
<td>65639</td>
</tr>
<tr>
<td>gcc-4.6-20101225</td>
<td>4369</td>
<td>70374</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-4.5.0</td>
<td>283</td>
<td>32723</td>
</tr>
<tr>
<td>gcc-4.6-20101225</td>
<td>335</td>
<td>35986</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-4.5.0</td>
<td>42</td>
<td>275</td>
<td>478</td>
<td>206</td>
</tr>
<tr>
<td>gcc-4.6-20101225</td>
<td>42</td>
<td>269</td>
<td>486</td>
<td>251</td>
</tr>
</tbody>
</table>
## ohcount: Line Count of gcc-4.4.2

Total: 66139 file(s)

<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>Total</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><strong>50312</strong></td>
<td><strong>4938881</strong></td>
<td><strong>1567750</strong></td>
<td><strong>24.1%</strong></td>
<td><strong>1071986</strong></td>
<td><strong>7578617</strong></td>
</tr>
</tbody>
</table>
### ohcount: Line Count of gcc-4.5.0

<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>16985</td>
<td>1967826</td>
<td>413941</td>
<td>17.4%</td>
<td>39183</td>
<td>2773650</td>
</tr>
<tr>
<td>cpp</td>
<td>20813</td>
<td>912618</td>
<td>210084</td>
<td>18.7%</td>
<td>199605</td>
<td>1322307</td>
</tr>
<tr>
<td>java</td>
<td>6342</td>
<td>681810</td>
<td>643127</td>
<td>48.5%</td>
<td>169483</td>
<td>1494420</td>
</tr>
<tr>
<td>ada</td>
<td>4412</td>
<td>647372</td>
<td>302226</td>
<td>31.8%</td>
<td>222481</td>
<td>1172079</td>
</tr>
<tr>
<td>autoconf</td>
<td>79</td>
<td>358996</td>
<td>422</td>
<td>0.1%</td>
<td>55631</td>
<td>415049</td>
</tr>
<tr>
<td>html</td>
<td>487</td>
<td>144535</td>
<td>5667</td>
<td>3.8%</td>
<td>31773</td>
<td>181975</td>
</tr>
<tr>
<td>make</td>
<td>93</td>
<td>114490</td>
<td>3438</td>
<td>2.9%</td>
<td>14434</td>
<td>132362</td>
</tr>
<tr>
<td>fortranfixed</td>
<td>2535</td>
<td>85905</td>
<td>1817</td>
<td>2.1%</td>
<td>11394</td>
<td>99116</td>
</tr>
<tr>
<td>assembler</td>
<td>197</td>
<td>45098</td>
<td>10082</td>
<td>18.3%</td>
<td>7528</td>
<td>62708</td>
</tr>
<tr>
<td>shell</td>
<td>136</td>
<td>39789</td>
<td>8984</td>
<td>18.4%</td>
<td>5511</td>
<td>54284</td>
</tr>
<tr>
<td>scheme</td>
<td>7</td>
<td>13725</td>
<td>1192</td>
<td>8.0%</td>
<td>1524</td>
<td>16441</td>
</tr>
<tr>
<td>fortranfree</td>
<td>760</td>
<td>12955</td>
<td>2889</td>
<td>18.2%</td>
<td>1546</td>
<td>17390</td>
</tr>
<tr>
<td>objective_c</td>
<td>396</td>
<td>10782</td>
<td>1835</td>
<td>14.5%</td>
<td>2959</td>
<td>15576</td>
</tr>
<tr>
<td>automake</td>
<td>64</td>
<td>6388</td>
<td>914</td>
<td>12.5%</td>
<td>994</td>
<td>8296</td>
</tr>
<tr>
<td>perl</td>
<td>25</td>
<td>4144</td>
<td>1139</td>
<td>21.6%</td>
<td>739</td>
<td>6022</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>ocaml</td>
<td>5</td>
<td>2515</td>
<td>540</td>
<td>17.7%</td>
<td>328</td>
<td>3383</td>
</tr>
<tr>
<td>python</td>
<td>10</td>
<td>1686</td>
<td>322</td>
<td>16.0%</td>
<td>383</td>
<td>2391</td>
</tr>
<tr>
<td>awk</td>
<td>10</td>
<td>1352</td>
<td>372</td>
<td>21.6%</td>
<td>218</td>
<td>1942</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>haskell</td>
<td>49</td>
<td>153</td>
<td>0</td>
<td>0.0%</td>
<td>17</td>
<td>170</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>1</td>
<td>5</td>
<td>0</td>
<td>0.0%</td>
<td>0</td>
<td>5</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>16985</td>
<td>196782</td>
<td>413941</td>
<td>17.4%</td>
<td>391883</td>
<td>277365</td>
</tr>
<tr>
<td>cpp</td>
<td>20813</td>
<td>912618</td>
<td>210084</td>
<td>18.7%</td>
<td>199605</td>
<td>132230</td>
</tr>
<tr>
<td>java</td>
<td>6342</td>
<td>681810</td>
<td>643127</td>
<td>48.5%</td>
<td>169483</td>
<td>149442</td>
</tr>
<tr>
<td>ada</td>
<td>4412</td>
<td>647372</td>
<td>302226</td>
<td>31.8%</td>
<td>222481</td>
<td>117208</td>
</tr>
<tr>
<td>autoconf</td>
<td>79</td>
<td>358996</td>
<td>422</td>
<td>0.1%</td>
<td>55631</td>
<td>415049</td>
</tr>
<tr>
<td>html</td>
<td>487</td>
<td>144535</td>
<td>5667</td>
<td>3.8%</td>
<td>31773</td>
<td>181975</td>
</tr>
<tr>
<td>make</td>
<td>93</td>
<td>114490</td>
<td>3438</td>
<td>2.9%</td>
<td>14434</td>
<td>132362</td>
</tr>
<tr>
<td>fortranfixed</td>
<td>2535</td>
<td>85905</td>
<td>1817</td>
<td>2.1%</td>
<td>11394</td>
<td>99116</td>
</tr>
<tr>
<td>assembler</td>
<td>197</td>
<td>45098</td>
<td>10082</td>
<td>18.3%</td>
<td>7528</td>
<td>62708</td>
</tr>
<tr>
<td>shell</td>
<td>136</td>
<td>39789</td>
<td>8984</td>
<td>18.4%</td>
<td>5511</td>
<td>54284</td>
</tr>
<tr>
<td>scheme</td>
<td>7</td>
<td>13725</td>
<td>1192</td>
<td>8.0%</td>
<td>1524</td>
<td>16441</td>
</tr>
<tr>
<td>fortranfree</td>
<td>760</td>
<td>12955</td>
<td>2889</td>
<td>18.2%</td>
<td>1546</td>
<td>17390</td>
</tr>
<tr>
<td>objective_c</td>
<td>396</td>
<td>10782</td>
<td>1835</td>
<td>14.5%</td>
<td>2959</td>
<td>15576</td>
</tr>
<tr>
<td>automake</td>
<td>64</td>
<td>6388</td>
<td>914</td>
<td>12.5%</td>
<td>994</td>
<td>8296</td>
</tr>
<tr>
<td>perl</td>
<td>25</td>
<td>4144</td>
<td>1139</td>
<td>21.6%</td>
<td>739</td>
<td>6022</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>ocaml</td>
<td>5</td>
<td>2515</td>
<td>540</td>
<td>17.7%</td>
<td>328</td>
<td>3383</td>
</tr>
<tr>
<td>python</td>
<td>10</td>
<td>1686</td>
<td>322</td>
<td>16.0%</td>
<td>383</td>
<td>2391</td>
</tr>
<tr>
<td>awk</td>
<td>10</td>
<td>1352</td>
<td>372</td>
<td>21.6%</td>
<td>218</td>
<td>1942</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>haskell</td>
<td>49</td>
<td>153</td>
<td>0</td>
<td>0.0%</td>
<td>17</td>
<td>170</td>
</tr>
<tr>
<td>emacs_lisp</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>1</td>
<td>5</td>
<td>0</td>
<td>0.0%</td>
<td>0</td>
<td>5</td>
</tr>
</tbody>
</table>
**ohcount: Line Count of gcc-4.6-20101225 snapshot**

<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>18311</td>
<td>2089300</td>
<td>441364</td>
<td>17.4%</td>
<td>415623</td>
<td>2946287</td>
</tr>
<tr>
<td>cpp</td>
<td>21813</td>
<td>977852</td>
<td>227979</td>
<td>18.9%</td>
<td>213239</td>
<td>1419070</td>
</tr>
<tr>
<td>java</td>
<td>6342</td>
<td>681938</td>
<td>645505</td>
<td>48.6%</td>
<td>169819</td>
<td>1497262</td>
</tr>
<tr>
<td>ada</td>
<td>4601</td>
<td>680002</td>
<td>315946</td>
<td>31.7%</td>
<td>234447</td>
<td>1230395</td>
</tr>
<tr>
<td>autoconf</td>
<td>91</td>
<td>397682</td>
<td>513</td>
<td>0.1%</td>
<td>61417</td>
<td>459612</td>
</tr>
<tr>
<td>html</td>
<td>446</td>
<td>141275</td>
<td>5391</td>
<td>3.7%</td>
<td>30812</td>
<td>177478</td>
</tr>
<tr>
<td>make</td>
<td>99</td>
<td>121013</td>
<td>3615</td>
<td>2.9%</td>
<td>15539</td>
<td>140167</td>
</tr>
<tr>
<td>fortranfixed</td>
<td>2852</td>
<td>96084</td>
<td>1920</td>
<td>2.0%</td>
<td>13196</td>
<td>111200</td>
</tr>
<tr>
<td>shell</td>
<td>148</td>
<td>47937</td>
<td>10414</td>
<td>17.8%</td>
<td>6566</td>
<td>64917</td>
</tr>
<tr>
<td>assembler</td>
<td>209</td>
<td>47015</td>
<td>10287</td>
<td>18.0%</td>
<td>7877</td>
<td>65179</td>
</tr>
<tr>
<td>objective_c</td>
<td>815</td>
<td>26409</td>
<td>4669</td>
<td>15.0%</td>
<td>7584</td>
<td>38662</td>
</tr>
<tr>
<td>scheme</td>
<td>7</td>
<td>13731</td>
<td>1192</td>
<td>8.0%</td>
<td>1524</td>
<td>16447</td>
</tr>
<tr>
<td>fortranfree</td>
<td>806</td>
<td>13667</td>
<td>3104</td>
<td>18.5%</td>
<td>1675</td>
<td>18446</td>
</tr>
<tr>
<td>automake</td>
<td>67</td>
<td>9103</td>
<td>971</td>
<td>9.6%</td>
<td>1355</td>
<td>11429</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>1729</td>
<td>396</td>
<td>18.6%</td>
<td>257</td>
<td>2382</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>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>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><strong>Total</strong></td>
<td>56846</td>
<td>5408876</td>
<td>1683047</td>
<td>23.7%</td>
<td>1189286</td>
<td>8281209</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>18311</td>
<td>2089300</td>
<td>441364</td>
<td>17.4%</td>
<td>415623</td>
<td>2946287</td>
</tr>
<tr>
<td>cpp</td>
<td>21813</td>
<td>977852</td>
<td>227979</td>
<td>18.9%</td>
<td>213239</td>
<td>1419070</td>
</tr>
<tr>
<td>java</td>
<td>6342</td>
<td>681938</td>
<td>645505</td>
<td>48.6%</td>
<td>169819</td>
<td>1497262</td>
</tr>
<tr>
<td>ada</td>
<td>4601</td>
<td>680002</td>
<td>315946</td>
<td>31.7%</td>
<td>234447</td>
<td>1230395</td>
</tr>
<tr>
<td>autoconf</td>
<td>91</td>
<td>397682</td>
<td>513</td>
<td>0.1%</td>
<td>61417</td>
<td>459612</td>
</tr>
<tr>
<td>html</td>
<td>446</td>
<td>141275</td>
<td>5391</td>
<td>3.7%</td>
<td>30812</td>
<td>177478</td>
</tr>
<tr>
<td>make</td>
<td>99</td>
<td>121013</td>
<td>3615</td>
<td>2.9%</td>
<td>15539</td>
<td>140167</td>
</tr>
<tr>
<td>fortranfixed</td>
<td>2852</td>
<td>96084</td>
<td>1920</td>
<td>2.0%</td>
<td>13196</td>
<td>111200</td>
</tr>
<tr>
<td>shell</td>
<td>148</td>
<td>47937</td>
<td>10414</td>
<td>17.8%</td>
<td>6566</td>
<td>64917</td>
</tr>
<tr>
<td>assembler</td>
<td>209</td>
<td>47015</td>
<td>10287</td>
<td>18.0%</td>
<td>7877</td>
<td>65179</td>
</tr>
<tr>
<td>objective_c</td>
<td>815</td>
<td>26409</td>
<td>4669</td>
<td>15.0%</td>
<td>7584</td>
<td>38662</td>
</tr>
<tr>
<td>scheme</td>
<td>7</td>
<td>13731</td>
<td>1192</td>
<td>8.0%</td>
<td>1524</td>
<td>16447</td>
</tr>
<tr>
<td>fortranfree</td>
<td>806</td>
<td>13667</td>
<td>3104</td>
<td>18.5%</td>
<td>1675</td>
<td>18446</td>
</tr>
<tr>
<td>automake</td>
<td>67</td>
<td>9103</td>
<td>971</td>
<td>9.6%</td>
<td>1355</td>
<td>11429</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>1729</td>
<td>396</td>
<td>18.6%</td>
<td>257</td>
<td>2382</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>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>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><strong>Total</strong></td>
<td>56846</td>
<td>540876</td>
<td>1683047</td>
<td>23.7%</td>
<td>1189286</td>
<td>8281209</td>
</tr>
</tbody>
</table>
## ohcount: Line Count of gcc-4.4.2/gcc

Total: 30421 file(s)

<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>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>3469812</td>
</tr>
</tbody>
</table>
Total: 30421 file(s)

<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>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><strong>Total</strong></td>
<td>28258</td>
<td>2242738</td>
<td>647591</td>
<td>22.4%</td>
<td>579484</td>
<td>3469813</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>14565</td>
<td>1368937</td>
<td>300284</td>
<td>18.0%</td>
<td>305671</td>
<td>1974892</td>
</tr>
<tr>
<td>ada</td>
<td>4402</td>
<td>645691</td>
<td>301666</td>
<td>31.8%</td>
<td>221882</td>
<td>1169239</td>
</tr>
<tr>
<td>cpp</td>
<td>7984</td>
<td>197798</td>
<td>54719</td>
<td>21.7%</td>
<td>57312</td>
<td>309829</td>
</tr>
<tr>
<td>fortranfixed</td>
<td>2453</td>
<td>80403</td>
<td>1768</td>
<td>2.2%</td>
<td>11008</td>
<td>93179</td>
</tr>
<tr>
<td>assembler</td>
<td>136</td>
<td>31802</td>
<td>7431</td>
<td>18.9%</td>
<td>4864</td>
<td>44097</td>
</tr>
<tr>
<td>autoconf</td>
<td>3</td>
<td>27317</td>
<td>10</td>
<td>0.0%</td>
<td>3876</td>
<td>31203</td>
</tr>
<tr>
<td>scheme</td>
<td>7</td>
<td>13725</td>
<td>1192</td>
<td>8.0%</td>
<td>1524</td>
<td>16441</td>
</tr>
<tr>
<td>fortranfree</td>
<td>722</td>
<td>12001</td>
<td>2683</td>
<td>18.3%</td>
<td>1446</td>
<td>16130</td>
</tr>
<tr>
<td>objective_c</td>
<td>392</td>
<td>10375</td>
<td>1721</td>
<td>14.2%</td>
<td>2838</td>
<td>14934</td>
</tr>
<tr>
<td>make</td>
<td>3</td>
<td>5886</td>
<td>1039</td>
<td>15.0%</td>
<td>854</td>
<td>7779</td>
</tr>
<tr>
<td>ocaml</td>
<td>5</td>
<td>2515</td>
<td>540</td>
<td>17.7%</td>
<td>328</td>
<td>3383</td>
</tr>
<tr>
<td>shell</td>
<td>14</td>
<td>2101</td>
<td>642</td>
<td>23.4%</td>
<td>347</td>
<td>3090</td>
</tr>
<tr>
<td>awk</td>
<td>8</td>
<td>1247</td>
<td>299</td>
<td>19.3%</td>
<td>212</td>
<td>1758</td>
</tr>
<tr>
<td>perl</td>
<td>2</td>
<td>805</td>
<td>206</td>
<td>20.4%</td>
<td>144</td>
<td>1155</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><strong>Total</strong></td>
<td>30747</td>
<td>2406202</td>
<td>677035</td>
<td>22.0%</td>
<td>613025</td>
<td>3696262</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>14565</td>
<td>1368937</td>
<td>300284</td>
<td>18.0%</td>
<td>305671</td>
<td>1974892</td>
</tr>
<tr>
<td>ada</td>
<td>4402</td>
<td>645691</td>
<td>301666</td>
<td>31.8%</td>
<td>221882</td>
<td>1169239</td>
</tr>
<tr>
<td>cpp</td>
<td>7984</td>
<td>197798</td>
<td>54719</td>
<td>21.7%</td>
<td>57312</td>
<td>309829</td>
</tr>
<tr>
<td>fortranfixed</td>
<td>2453</td>
<td>80403</td>
<td>1768</td>
<td>2.2%</td>
<td>11008</td>
<td>93179</td>
</tr>
<tr>
<td>assembler</td>
<td>136</td>
<td>31802</td>
<td>7431</td>
<td>18.9%</td>
<td>4864</td>
<td>44097</td>
</tr>
<tr>
<td>autoconf</td>
<td>3</td>
<td>27317</td>
<td>10</td>
<td>0.0%</td>
<td>3876</td>
<td>31203</td>
</tr>
<tr>
<td>scheme</td>
<td>7</td>
<td>13725</td>
<td>1192</td>
<td>8.0%</td>
<td>1524</td>
<td>16441</td>
</tr>
<tr>
<td>fortranfree</td>
<td>722</td>
<td>12001</td>
<td>2683</td>
<td>18.3%</td>
<td>1446</td>
<td>16130</td>
</tr>
<tr>
<td>objective_c</td>
<td>392</td>
<td>10375</td>
<td>1721</td>
<td>14.2%</td>
<td>2838</td>
<td>14934</td>
</tr>
<tr>
<td>make</td>
<td>3</td>
<td>5886</td>
<td>1039</td>
<td>15.0%</td>
<td>854</td>
<td>7779</td>
</tr>
<tr>
<td>ocaml</td>
<td>5</td>
<td>2515</td>
<td>540</td>
<td>17.7%</td>
<td>328</td>
<td>3383</td>
</tr>
<tr>
<td>shell</td>
<td>14</td>
<td>2101</td>
<td>642</td>
<td>23.4%</td>
<td>347</td>
<td>3090</td>
</tr>
<tr>
<td>awk</td>
<td>8</td>
<td>1247</td>
<td>299</td>
<td>19.3%</td>
<td>212</td>
<td>1758</td>
</tr>
<tr>
<td>perl</td>
<td>2</td>
<td>805</td>
<td>206</td>
<td>20.4%</td>
<td>144</td>
<td>1155</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><strong>Total</strong></td>
<td><strong>30747</strong></td>
<td><strong>2406202</strong></td>
<td><strong>677035</strong></td>
<td><strong>22.0%</strong></td>
<td><strong>613025</strong></td>
<td><strong>3696262</strong></td>
</tr>
</tbody>
</table>
ohcount: Line Count of gcc-4.6-20101225/gcc snapshot

Total: 36322 file(s)

<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>1452351</td>
<td>319224</td>
<td>18.0%</td>
<td>321806</td>
<td>2093381</td>
</tr>
<tr>
<td>ada</td>
<td>4591</td>
<td>678321</td>
<td>315386</td>
<td>31.7%</td>
<td>233848</td>
<td>1227555</td>
</tr>
<tr>
<td>cpp</td>
<td>8527</td>
<td>248085</td>
<td>60722</td>
<td>19.7%</td>
<td>66383</td>
<td>375190</td>
</tr>
<tr>
<td>fortranfixed</td>
<td>2767</td>
<td>90244</td>
<td>1871</td>
<td>2.0%</td>
<td>12800</td>
<td>104915</td>
</tr>
<tr>
<td>assembler</td>
<td>138</td>
<td>31871</td>
<td>7506</td>
<td>19.1%</td>
<td>4882</td>
<td>44259</td>
</tr>
<tr>
<td>autoconf</td>
<td>3</td>
<td>28604</td>
<td>12</td>
<td>0.0%</td>
<td>4011</td>
<td>32627</td>
</tr>
<tr>
<td>objective_c</td>
<td>810</td>
<td>25860</td>
<td>4492</td>
<td>14.8%</td>
<td>7436</td>
<td>37788</td>
</tr>
<tr>
<td>scheme</td>
<td>7</td>
<td>13731</td>
<td>1192</td>
<td>8.0%</td>
<td>1524</td>
<td>16447</td>
</tr>
<tr>
<td>fortranfree</td>
<td>768</td>
<td>12713</td>
<td>2893</td>
<td>18.5%</td>
<td>1575</td>
<td>17181</td>
</tr>
<tr>
<td>make</td>
<td>4</td>
<td>6124</td>
<td>1070</td>
<td>14.9%</td>
<td>893</td>
<td>8087</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>1624</td>
<td>323</td>
<td>16.6%</td>
<td>251</td>
<td>2198</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><strong>Total</strong></td>
<td><strong>33338</strong></td>
<td><strong>2600787</strong></td>
<td><strong>718924</strong></td>
<td><strong>21.7%</strong></td>
<td><strong>657002</strong></td>
<td><strong>397671</strong></td>
</tr>
</tbody>
</table>
## Table of Language Distribution

<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>1452351</td>
<td>319224</td>
<td>18.0%</td>
<td>321806</td>
<td>2093381</td>
</tr>
<tr>
<td>ada</td>
<td>4591</td>
<td>678321</td>
<td>315386</td>
<td>31.7%</td>
<td>233848</td>
<td>1227555</td>
</tr>
<tr>
<td>cpp</td>
<td>8527</td>
<td>248085</td>
<td>60722</td>
<td>19.7%</td>
<td>66383</td>
<td>375190</td>
</tr>
<tr>
<td>fortranfixed</td>
<td>2767</td>
<td>90244</td>
<td>1871</td>
<td>2.0%</td>
<td>12800</td>
<td>104915</td>
</tr>
<tr>
<td>assembler</td>
<td>138</td>
<td>31871</td>
<td>7506</td>
<td>19.1%</td>
<td>4882</td>
<td>44259</td>
</tr>
<tr>
<td>autoconf</td>
<td>3</td>
<td>28604</td>
<td>12</td>
<td>0.0%</td>
<td>4011</td>
<td>32627</td>
</tr>
<tr>
<td>objective_c</td>
<td>810</td>
<td>25860</td>
<td>4492</td>
<td>14.8%</td>
<td>7436</td>
<td>37788</td>
</tr>
<tr>
<td>scheme</td>
<td>7</td>
<td>13731</td>
<td>1192</td>
<td>8.0%</td>
<td>1524</td>
<td>16447</td>
</tr>
<tr>
<td>fortranfree</td>
<td>768</td>
<td>12713</td>
<td>2893</td>
<td>18.5%</td>
<td>1575</td>
<td>17181</td>
</tr>
<tr>
<td>make</td>
<td>4</td>
<td>6124</td>
<td>1070</td>
<td>14.9%</td>
<td>893</td>
<td>8087</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>1624</td>
<td>323</td>
<td>16.6%</td>
<td>251</td>
<td>2198</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><strong>Total</strong></td>
<td><strong>33338</strong></td>
<td><strong>2600787</strong></td>
<td><strong>718924</strong></td>
<td><strong>21.7%</strong></td>
<td><strong>657002</strong></td>
<td><strong>3976713</strong></td>
</tr>
</tbody>
</table>
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”
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
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
Why is Understanding GCC Difficult?

Deeper reason: GCC is not a *compiler* but a *compiler generation framework*

There are 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
The Architecture of GCC

Compiler Generation Framework

- Language Specific Code
- Language and Machine Independent Generic Code
- Machine Dependent Generator Code
- Machine Descriptions
The Architecture of GCC

Compiler Generation Framework

Language Specific Code

Language and Machine Independent Generic Code

Machine Dependent Generator Code

Machine Descriptions

Parser
Gimplifier
Tree SSA Optimizer
Expander
Optimizer
Recognizer

Source Program

Generated Compiler (cc1)

Assembly Program

Uday Khedker

GRC, IIT Bombay
The Architecture of GCC

Input Language

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

Generated Compiler (cc1)

Assembly Program

Parser

Gimplifier

Tree SSA Optimizer

Expander

Optimizer

Recognizer

Uday Khedker
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

Generated Compiler (cc1)

Assembly Program

Parser

Gimplifier

Tree SSA Optimizer

Expander

Optimizer

Recognizer

Target Name

Development Time

Build Time

Use Time

Uday Khedker

GRC, 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

```
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 the directory gcc-4.5.0/gcc using cscope
Another Example of The Generation Related Gap

Locating the main function in the directory gcc-4.5.0/gcc using cscope

<table>
<thead>
<tr>
<th>File</th>
<th>Line</th>
<th>Function</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>collect2.c</td>
<td>1111</td>
<td>main</td>
<td>(int argc, char **argv)</td>
</tr>
<tr>
<td>fp-test.c</td>
<td>85</td>
<td>main</td>
<td>(void)</td>
</tr>
<tr>
<td>gcc.c</td>
<td>6803</td>
<td>main</td>
<td>(int argc, char **argv)</td>
</tr>
<tr>
<td>gcov-dump.c</td>
<td>76</td>
<td>main</td>
<td>(int argc ATTRIBUTE_UNUSED, char **argv)</td>
</tr>
<tr>
<td>gcov iov.c</td>
<td>29</td>
<td>main</td>
<td>(int argc, char **argv)</td>
</tr>
<tr>
<td>gcov.c</td>
<td>355</td>
<td>main</td>
<td>(int argc, char **argv)</td>
</tr>
<tr>
<td>genattr.c</td>
<td>89</td>
<td>main</td>
<td>(int argc, char **argv)</td>
</tr>
<tr>
<td>genattribtab.c</td>
<td>4439</td>
<td>main</td>
<td>(int argc, char **argv)</td>
</tr>
<tr>
<td>genautomata.c</td>
<td>9475</td>
<td>main</td>
<td>(int argc, char **argv)</td>
</tr>
<tr>
<td>genchecksum.c</td>
<td>67</td>
<td>main</td>
<td>(int argc, char ** argv)</td>
</tr>
<tr>
<td>gencodes.c</td>
<td>51</td>
<td>main</td>
<td>(int argc, char **argv)</td>
</tr>
<tr>
<td>genconditions.c</td>
<td>209</td>
<td>main</td>
<td>(int argc, char **argv)</td>
</tr>
<tr>
<td>genconfig.c</td>
<td>261</td>
<td>main</td>
<td>(int argc, char **argv)</td>
</tr>
<tr>
<td>genconstants.c</td>
<td>50</td>
<td>main</td>
<td>(int argc, char **argv)</td>
</tr>
<tr>
<td>genemit.c</td>
<td>825</td>
<td>main</td>
<td>(int argc, char **argv)</td>
</tr>
<tr>
<td>genextract.c</td>
<td>401</td>
<td>main</td>
<td>(int argc, char **argv)</td>
</tr>
</tbody>
</table>
Another Example of The Generation Related Gap

Locating the main function in the directory gcc-4.5.0/gcc using cscope

<table>
<thead>
<tr>
<th>File</th>
<th>Line</th>
<th>Function Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>g genflags.c</td>
<td>250</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>h gengenrtl.c</td>
<td>350</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>i gengtype.c</td>
<td>3694</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>j genmddeps.c</td>
<td>45</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>k genmodes.c</td>
<td>1376</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>l genopinit.c</td>
<td>469</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>m genoutput.c</td>
<td>1023</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>n genpeep.c</td>
<td>353</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>o genpreds.c</td>
<td>1404</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>p genrecog.c</td>
<td>2722</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>q lto-wrapper.c</td>
<td>412</td>
<td>main (int argc, char *argv[])</td>
</tr>
<tr>
<td>r main.c</td>
<td>33</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>s mips-tdump.c</td>
<td>1393</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>t mips-tfile.c</td>
<td>655</td>
<td>main (void )</td>
</tr>
<tr>
<td>u mips-tfile.c</td>
<td>4695</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>v tlink.c</td>
<td>61</td>
<td>const char *main;</td>
</tr>
</tbody>
</table>
GCC Retargetability Mechanism

Input Language → Compiler Generation Framework → Target Name

Language Specific Code

Language and Machine Independent Generic Code

Machine Dependent Generator Code

Machine Descriptions

Parser → Gimmlifier → Tree SSA Optimizer → Expander → Optimizer → Recognizer

Selected → Copied → Generated

Development Time

Build Time

Use Time

Generated Compiler
GCC Retargetability Mechanism

Input Language

Compiler Generation Framework

Target Name

Language Specific Code

Language and Machine Independent Generic Code

Machine Dependent Generator Code

Machine Descriptions

Development Time

Build Time

Use Time

Gimple → IR-RTL

IR-RTL → ASM

Selected

Copied

Copied

Generated

Generated

Generated

Parser

Gimplifier

Tree SSA Optimizer

Expander

Optimizer

Recognizer

Generated Compiler
GCC Retargetability Mechanism

Compiler Generation Framework

Input Language

Language Specific Code

Language and Machine Independent Generic Code

Machine Dependent Generator Code

Machine Descriptions

Target Name

Gimple → PN

PN → IR-RTL

IR-RTL → ASM

Gimple → IR-RTL

IR-RTL → ASM

Use Time

Development Time

Build Time

Selected

Copied

Copied

Generated

Generated

Generated

Selected

Copied

Copied

Generated

Parser

Gimplifier

Tree SSA Optimizer

Expander

Optimizer

Recognizer

Generated Compiler

Uday Khedker

GRC, IIT Bombay


GCC Retargetability Mechanism

Input Language

Language Specific Code

Selected

Language and Machine Independent Generic Code

Copied

Compiler Generation Framework

Machine Dependent Generator Code

Copied

Target Name

Machine Descriptions

Generated

Compiler Generation Framework

Gimple → PN

Development Time

PN → IR-RTL

Build Time

IR-RTL → ASM

Use Time

Gimple → IR-RTL

IR-RTL → ASM

Generated Compiler

Parser

Gimplifier

Tree SSA Optimizer

Expander

Optimizer

Recognizer

Selected

Copied

Copied

Generated

Gimple → PN

PN → IR-RTL

IR-RTL → ASM

GRC, IIT Bombay

Uday Khedker
GCC Retargetability Mechanism

Input Language → Language Specific Code

Compiler Generation Framework

Language and Machine Independent Generic Code → Machine Dependent Generator Code

Machine Descriptions

Input Language Target Name

Gimple → PN

PN → IR-RTL

IR-RTL → ASM

Gimple → IR-RTL

IR-RTL → ASM

Generated Compiler

Parser → Gimplifier → Tree SSA Optimizer → Expander → Optimizer → Recognizer → Generated Compiler

Development Time

Build Time

Use Time

Uday Khedker

GRC, 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

<table>
<thead>
<tr>
<th>Files</th>
<th>i386</th>
<th>mips</th>
</tr>
</thead>
<tbody>
<tr>
<td>*.md</td>
<td>35766</td>
<td>12930</td>
</tr>
<tr>
<td>*.c</td>
<td>28643</td>
<td>12572</td>
</tr>
<tr>
<td>*.h</td>
<td>15694</td>
<td>5105</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

<table>
<thead>
<tr>
<th>Files</th>
<th>i386</th>
<th>mips</th>
</tr>
</thead>
<tbody>
<tr>
<td>*.md</td>
<td>35766</td>
<td>12930</td>
</tr>
<tr>
<td>*.c</td>
<td>28643</td>
<td>12572</td>
</tr>
<tr>
<td>*.h</td>
<td>15694</td>
<td>5105</td>
</tr>
</tbody>
</table>

• Machine descriptions are difficult to construct, understand, debug, and enhance
Part 5

Meeting the GCC Challenge
## Meeting the GCC Challenge

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

<table>
<thead>
<tr>
<th></th>
<th>Makefiles</th>
<th>Source</th>
<th>MD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Makefiles</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Source</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MD</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Uday Khedker
GRC, IIT Bombay
What is Gray Box Probing of GCC?

- **Black Box probing:**
  Examining only the input and output relationship of a system

- **White Box probing:**
  Examining internals of a system for a given set of inputs

- **Gray Box probing:**
  Examining input and output of various components/modules
  - Overview of translation sequence in GCC
  - Overview of intermediate representations
  - Intermediate representations of programs across important phases
Customizing the Configuration and Build Process

- Creating only cc1
- Creating bare metal cross build
  Complete tool chain without OS support
- Creating cross build with OS support
Incremental Construction of Machine Descriptions

• Define different levels of source language
• Identify the minimal set of features in the target required to support each level
• Identify the minimal information required in the machine description to support each level
  ▶ Successful compilation of any program, and
  ▶ correct execution of the generated assembly program
• Interesting observations
  ▶ It is the increment in the source language which results in understandable increments in machine descriptions rather than the increment in the target architecture
  ▶ If the levels are identified properly, the increments in machine descriptions are monotonic
Incremental Construction of Machine Descriptions

- Conditional control transfers
- Function Calls
- Arithmetic Expressions
  - Sequence of Simple Assignments involving integers
    - MD Level 1
    - MD Level 2
    - MD Level 3
    - MD Level 4
Adding Passes to Massage IRs

• Understanding the pass structure
• Understanding the mechanisms of traversing a call graph and a control flow graph
• Understanding how to access the data structures of IRs
• Simple exercises such as:
  ▶ Count the number of copy statements in a program
  ▶ Count the number of variables declared "const" in the program
  ▶ Count the number of occurrences of arithmetic operators in the program
  ▶ Count the number of references to global variables in the program
## CS 715 Coverage

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

Uday Khedker
GRC, IIT Bombay
CS 715 Coverage

- An external view of GCC:
  - Configuration and building
  - Gray box probing of GCC
  - Introduction to Gimple and RTL IRs
  - Parallelization and vectorization in GCC
  - Introduction to GCC Machine Descriptions

- An internal view of GCC: Walking the maze of GCC source code
  - Control flow and plugin structure of the core compiler,
  - Mechanisms for hooking up front ends, IR passes, and back ends
  - Examining and manipulating Gimple and RTL IRs
  - Design and implementation of GDFA
  - Machine descriptions and retargetability mechanism
Twin goals:

- *Learning how to learn GCC*
  
  Our focus will be on
  - giving you some core information
  - showing you how to discover more information

- *Striking a balance between theory and practice*
  
  Our focus will be on showing you how to
  - discover concepts in a large code base and build abstractions
  - take concepts and update a large code base
  - relate the class room concepts of compilers to an industry strength compiler
CS 715 Pedagogy

- Introductory lecture for each topic followed by lab work
- Many the lecture hours will be used as lab hours
  - We will meet a one place (NSL? GRC?)
    - You are expected to do ssh to your own machine
  - Class room as labs using your laptops?
- Tools
  - shell, make, cscope, ctags, gdb/ddd, screen, spim (mips simulator)
CS 715 Lab Work

- Lab exercises:
  - Pre-defined experiments
  - Ungraded

- Assignments: Graded lab work
  - Implementation to modify gcc
  - Graded
  - To be submitted in about a couple of weeks or 10 days

- Projects
  - Specific Study + Implementation
  - Graded
  - To be submitted in about a couple of months
  - Possible topics:
    - Extensions of GDFA, MD rewriting with newer constructs, Detailing the retargetability mechanism, MD parsers, garbage collection, MELT, investigating specific optimizations (such as constant splitting)
CS 715 Assessment Scheme

- No written examination
- Marks for lab work

<table>
<thead>
<tr>
<th>Head</th>
<th>Number</th>
<th>Weightage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assignments</td>
<td>4</td>
<td>$4 \times 15 = 60$</td>
</tr>
<tr>
<td>Project</td>
<td>1</td>
<td>40</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>100</td>
</tr>
</tbody>
</table>

- Assignments need not be same for all students
  However, they will be comparable and students would have the choice of opting for a particular assignment