Workshop on Essential Abstractions in GCC

An Overview of Compilation and GCC

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

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

30 June 2011

Overview: Outline

Outline

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

Part 1

Introduction to Compilation
Nothing is known except the problem
Overall strategy, algorithm, data structures etc.
Functions, variables, their types etc.

Machine instructions, registers etc.
Addresses of functions, external data etc.
Actual addresses of code and data
Values of variables

Conceptualisation  Coding  Compiling  Linking  Loading  Execution

Time

Essential Abstractions in GCC
GCC Resource Center, IIT Bombay
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

Input C statement

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

Spim Assembly Equivalent

```
lw	$t0, 4($fp)   ; t0 <- b     # Is b smaller
slt	$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

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

Conditional jump

\[ \text{Condition} \rightarrow \text{Fall through} \]

False Part

True Part

NOT Condition

Conditional jump

Fall through

True Part

False Part

Notes

Essential Abstractions in GCC
GCC Resource Center, IIT Bombay

Essential Abstractions in GCC
GCC Resource Center, IIT Bombay
Implementation Mechanisms

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

- Translation Instructions \rightarrow Equivalent Instructions
- Interpretation Instructions \rightarrow Actions Implied by Instructions

Language Implementation Models

Analysis \rightarrow Synthesis \rightarrow Compilation
Analysis \rightarrow Execution \rightarrow Interpretation
Language Processor Models

C, C++, Java, C#:

- Front End
- Optimizer
- Back End
- Virtual Machine

Part 2

An Overview of Compilation Phases
The Structure of a Simple Compiler

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

Back End
- Instruction Selector
- Register Allocator
- Assembly Emitter
- Assembly Program

Source Program

Translation Sequence in Our Compiler: Parsing

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

Input

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

\[ a = \begin{cases} 10 \text{?} & b : c \end{cases} \]

Input

AsgnStmnt

Lhs

name

\( E \)

name

\( E \) < E

name

name

parse Tree

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

\[ a = \begin{cases} 10 \text{?} & b : c \end{cases} \]

Input

Tree List

\( T_0 \)

IfGoto

Not L0:

Goto

\( T_0 \)

\( T_1 \)

\( T_0 \)

\( T_1 \)

\( T_1 \)

\( T_1 \)

\( T_1 \)

Goto L1:

L0:

L1:

L1:

L0:

name

(10,int)

name

num

<

name

(b,int)

name

(c.int)

name

(b,int)

Abstract Syntax Tree (with attributes)

Issues:

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

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

Input

Tree List

\[ T_0 = b < 10 \]

IfGoto

Not

L0:

\[ T_0 \]

Goto

L1:

\[ T_1 = b \]

L0:

\[ T_0 = c \]

L1:

\[ a = T_1 \]

Issues:

- Cover trees with as few machine instructions as possible
- Use temporaries and local registers

Instruction List

\[ T_0 \leftarrow b \]
\[ T_0 \leftarrow T_0 < 10 \]
\[ T_0 \leftarrow ! T_0 \]
\[ \text{if } T_0 > 0 \text{ goto L0: } \]
\[ T_1 \leftarrow b \]
\[ \text{goto L1: } \]
\[ L0: T_1 \leftarrow c \]
\[ L1: a \leftarrow T_3 \]

Assembly Code

\[ \text{lw } $t0, 4($fp) \]
\[ \text{slti } $t0, $t0, 10 \]
\[ \text{not } $t0, $t0 \]
\[ \text{bgtz } $t0, L0: \]
\[ \text{lw } $t0, 8($fp) \]
\[ \text{sw } 0($fp), $t0 \]
Part 3

Compilation Models

Aho Ullman Model

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

Davidson Fraser Model

Input Source Program
Front End
AST
Expander
Register Transfers
Optimizer
Register Transfers
Recognizer
Target Program

Essential Abstractions in GCC
GCC Resource Center, IIT Bombay
**Compilation Models**

### Aho Ullman Model

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

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

### Davidson Fraser Model

- **Front End** → **Expander** → **Register Transfers** → **Optimizer** → **Register Transfers** → **Recognizer** → **Target Program**

#### Davidson Fraser: Instruction selection
- over AST using
- structural tree pattern matching
- naive code which is
  - target dependent, and is
  - optimized subsequently

---

**Typical Front Ends**

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

---

**Notes**

**Essential Abstractions in GCC**

GCC Resource Center, IIT Bombay
Typical Back Ends in Aho Ullman Model

- Compile time evaluations
- Eliminating redundant computations

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

Assembly Code

Retargetability in Aho Ullman and Davidson Fraser Models

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

- For the GCC developer community: The GNU Compiler Collection
- For other compiler writers: The Great Compiler Challenge 😊
Why is Understanding GCC Difficult?

Some of the obvious reasons:

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

- **Open development model**
  
  Could lead to heterogeneity. Design flaws may be difficult to correct

- **Rapid versioning**
  
  GCC maintenance is a race against time. Disruptive corrections are difficult
Open Source and Free Software Development Model

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

- **Cathedral**: Total Centralized Control  
  Design, implement, test, release
- **Bazaar**: Total Decentralization  
  Release early, release often, make users partners in software development

“Given enough eyeballs, all bugs are shallow”

Code errors, logical errors, and architectural errors

*A combination of the two seems more sensible*

The Current Development Model of GCC

GCC follows a combination of the Cathedral and the Bazaar approaches

- **GCC Steering Committee**: Free Software Foundation has given charge  
  ▶ Major policy decisions  
  ▶ Handling Administrative and Political issues
- **Release Managers**:  
  ▶ Coordination of releases
- **Maintainers**:  
  ▶ Usually area/branch/module specific  
  ▶ Responsible for design and implementation  
  ▶ Take help of reviewers to evaluate submitted changes
Comprehensiveness of GCC: Wide Applicability

- **Input languages supported:**
  - C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada
- **Processors supported in standard releases:**
  - **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**
  - | Subdirectories | Files |
  - | gcc-4.4.2 | 3794 | 62301 |
  - | gcc-4.5.0 | 4056 | 65639 |
  - | gcc-4.6.0 | 4383 | 71096 |
- **Core size (src/gcc)**
  - | Subdirectories | Files |
  - | gcc-4.4.2 | 257 | 30163 |
  - | gcc-4.5.0 | 283 | 32723 |
  - | gcc-4.6.0 | 336 | 36503 |
- **Machine Descriptions (src/gcc/config)**
  - | Subdirectories | .c files | .h files | .md files |
  - | gcc-4.4.2 | 36 | 241 | 426 | 206 |
  - | gcc-4.5.0 | 42 | 275 | 478 | 206 |
  - | gcc-4.6.0 | 42 | 275 | 466 | 259 |
## Essential Abstractions in GCC

### GCC Resource Center, IIT Bombay

<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>394562</td>
<td>17.7%</td>
<td>366815</td>
<td>2601742</td>
</tr>
<tr>
<td>cpp</td>
<td>19622</td>
<td>877175</td>
<td>160744</td>
<td>17.9%</td>
<td>190977</td>
<td>1252528</td>
</tr>
<tr>
<td>java</td>
<td>6342</td>
<td>681656</td>
<td>643045</td>
<td>48.3%</td>
<td>169456</td>
<td>1491663</td>
</tr>
<tr>
<td>ada</td>
<td>4106</td>
<td>63857</td>
<td>294681</td>
<td>31.0%</td>
<td>218000</td>
<td>1151438</td>
</tr>
<tr>
<td>autocomp</td>
<td>174</td>
<td>440046</td>
<td>393</td>
<td>0.1%</td>
<td>58831</td>
<td>504270</td>
</tr>
<tr>
<td>make</td>
<td>8414</td>
<td>110064</td>
<td>3008</td>
<td>2.9%</td>
<td>13270</td>
<td>1354032</td>
</tr>
<tr>
<td>html</td>
<td>387</td>
<td>10080</td>
<td>5658</td>
<td>5.2%</td>
<td>23418</td>
<td>130176</td>
</tr>
<tr>
<td>fortran90</td>
<td>2164</td>
<td>73356</td>
<td>1070</td>
<td>2.1%</td>
<td>94564</td>
<td>943980</td>
</tr>
<tr>
<td>assembler</td>
<td>173</td>
<td>42460</td>
<td>9607</td>
<td>18.5%</td>
<td>7084</td>
<td>59151</td>
</tr>
<tr>
<td>sh</td>
<td>157</td>
<td>39347</td>
<td>8832</td>
<td>18.3%</td>
<td>5485</td>
<td>51664</td>
</tr>
<tr>
<td>fortran77</td>
<td>690</td>
<td>11852</td>
<td>2582</td>
<td>17.9%</td>
<td>1414</td>
<td>15848</td>
</tr>
<tr>
<td>objectivec</td>
<td>395</td>
<td>10952</td>
<td>1186</td>
<td>14.2%</td>
<td>2951</td>
<td>15281</td>
</tr>
<tr>
<td>automake</td>
<td>614</td>
<td>6014</td>
<td>853</td>
<td>12.5%</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>98913</td>
</tr>
<tr>
<td>scheme</td>
<td>1</td>
<td>2778</td>
<td>153</td>
<td>5.2%</td>
<td>328</td>
<td>3256</td>
</tr>
<tr>
<td>ocaml</td>
<td>5</td>
<td>2042</td>
<td>533</td>
<td>17.8%</td>
<td>328</td>
<td>3440</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>1046</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>18.5%</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>46</td>
<td>140</td>
<td>3</td>
<td>0.8%</td>
<td>10</td>
<td>155</td>
</tr>
<tr>
<td>emacslisp</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>0.9%</td>
<td>7</td>
<td>94</td>
</tr>
<tr>
<td>Total</td>
<td>50312</td>
<td>4938881</td>
<td>1567750</td>
<td>24.1%</td>
<td>1017986</td>
<td>7588817</td>
</tr>
</tbody>
</table>

## Essential Abstractions in GCC

### GCC Resource Center, IIT Bombay

<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>394562</td>
<td>17.7%</td>
<td>366815</td>
<td>2601742</td>
</tr>
<tr>
<td>cpp</td>
<td>19622</td>
<td>877175</td>
<td>160744</td>
<td>17.9%</td>
<td>190977</td>
<td>1252528</td>
</tr>
<tr>
<td>java</td>
<td>6342</td>
<td>681656</td>
<td>643045</td>
<td>48.3%</td>
<td>169456</td>
<td>1491663</td>
</tr>
<tr>
<td>ada</td>
<td>4106</td>
<td>63857</td>
<td>294681</td>
<td>31.0%</td>
<td>218000</td>
<td>1151438</td>
</tr>
<tr>
<td>autocomp</td>
<td>174</td>
<td>440046</td>
<td>393</td>
<td>0.1%</td>
<td>58831</td>
<td>504270</td>
</tr>
<tr>
<td>make</td>
<td>8414</td>
<td>110064</td>
<td>3008</td>
<td>2.9%</td>
<td>13270</td>
<td>1354032</td>
</tr>
<tr>
<td>html</td>
<td>387</td>
<td>10080</td>
<td>5658</td>
<td>5.2%</td>
<td>23418</td>
<td>130176</td>
</tr>
<tr>
<td>fortran90</td>
<td>2164</td>
<td>73356</td>
<td>1070</td>
<td>2.1%</td>
<td>94564</td>
<td>943980</td>
</tr>
<tr>
<td>assembler</td>
<td>173</td>
<td>42460</td>
<td>9607</td>
<td>18.5%</td>
<td>7084</td>
<td>59151</td>
</tr>
<tr>
<td>sh</td>
<td>157</td>
<td>39347</td>
<td>8832</td>
<td>18.3%</td>
<td>5485</td>
<td>51664</td>
</tr>
<tr>
<td>fortran77</td>
<td>690</td>
<td>11852</td>
<td>2582</td>
<td>17.9%</td>
<td>1414</td>
<td>15848</td>
</tr>
<tr>
<td>objectivec</td>
<td>395</td>
<td>10952</td>
<td>1186</td>
<td>14.2%</td>
<td>2951</td>
<td>15281</td>
</tr>
<tr>
<td>automake</td>
<td>614</td>
<td>6014</td>
<td>853</td>
<td>12.5%</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>98913</td>
</tr>
<tr>
<td>scheme</td>
<td>1</td>
<td>2778</td>
<td>153</td>
<td>5.2%</td>
<td>328</td>
<td>3256</td>
</tr>
<tr>
<td>ocaml</td>
<td>5</td>
<td>2042</td>
<td>533</td>
<td>17.8%</td>
<td>328</td>
<td>3440</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>1046</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>18.5%</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>46</td>
<td>140</td>
<td>3</td>
<td>0.8%</td>
<td>10</td>
<td>155</td>
</tr>
<tr>
<td>emacslisp</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>0.9%</td>
<td>7</td>
<td>94</td>
</tr>
<tr>
<td>Total</td>
<td>50312</td>
<td>4938881</td>
<td>1567750</td>
<td>24.1%</td>
<td>1017986</td>
<td>7588817</td>
</tr>
</tbody>
</table>
### Essential Abstractions in GCC

<table>
<thead>
<tr>
<th>Language</th>
<th>Files</th>
<th>Code</th>
<th>Comment</th>
<th>Comment %</th>
<th>Blank</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>c</td>
<td>18483</td>
<td>44433</td>
<td>10534</td>
<td>1.75%</td>
<td>61892</td>
<td>260282</td>
</tr>
<tr>
<td>cpp</td>
<td>22262</td>
<td>29651</td>
<td>13591</td>
<td>1.85%</td>
<td>23346</td>
<td>123466</td>
</tr>
<tr>
<td>java</td>
<td>3352</td>
<td>48560</td>
<td>6972</td>
<td>14.53%</td>
<td>61942</td>
<td>294699</td>
</tr>
<tr>
<td>ada</td>
<td>4060</td>
<td>13556</td>
<td>1750</td>
<td>13.29%</td>
<td>23446</td>
<td>123466</td>
</tr>
<tr>
<td>autotest</td>
<td>419</td>
<td>40469</td>
<td>2230</td>
<td>5.43%</td>
<td>48871</td>
<td>169938</td>
</tr>
<tr>
<td>html</td>
<td>457</td>
<td>3669</td>
<td>90</td>
<td>2.39%</td>
<td>38146</td>
<td>62101</td>
</tr>
<tr>
<td>make</td>
<td>98</td>
<td>3502</td>
<td>275</td>
<td>7.64%</td>
<td>15718</td>
<td>41862</td>
</tr>
<tr>
<td>fortran</td>
<td>195</td>
<td>1631</td>
<td>350</td>
<td>12.64%</td>
<td>18059</td>
<td>41862</td>
</tr>
<tr>
<td>shell</td>
<td>141</td>
<td>4351</td>
<td>1034</td>
<td>23.87%</td>
<td>14315</td>
<td>28529</td>
</tr>
<tr>
<td>assembler</td>
<td>298</td>
<td>3284</td>
<td>905</td>
<td>27.39%</td>
<td>7125</td>
<td>40875</td>
</tr>
<tr>
<td>rgb</td>
<td>75</td>
<td>4056</td>
<td>282</td>
<td>7.01%</td>
<td>3827</td>
<td>40145</td>
</tr>
<tr>
<td>objective</td>
<td>537</td>
<td>30032</td>
<td>1221</td>
<td>20.65%</td>
<td>28421</td>
<td>58745</td>
</tr>
<tr>
<td>fortran</td>
<td>821</td>
<td>13957</td>
<td>3147</td>
<td>18.50%</td>
<td>10664</td>
<td>24328</td>
</tr>
<tr>
<td>tcc</td>
<td>2</td>
<td>1282</td>
<td>94</td>
<td>7.39%</td>
<td>1431</td>
<td>28529</td>
</tr>
<tr>
<td>scheme</td>
<td>2</td>
<td>1282</td>
<td>94</td>
<td>7.39%</td>
<td>1431</td>
<td>28529</td>
</tr>
<tr>
<td>automake</td>
<td>67</td>
<td>3460</td>
<td>1034</td>
<td>29.85%</td>
<td>1431</td>
<td>28529</td>
</tr>
<tr>
<td>perl</td>
<td>91</td>
<td>8027</td>
<td>3669</td>
<td>27.70%</td>
<td>8118</td>
<td>26536</td>
</tr>
<tr>
<td>ocaml</td>
<td>21</td>
<td>1154</td>
<td>43</td>
<td>1.92%</td>
<td>1135</td>
<td>2270</td>
</tr>
<tr>
<td>python</td>
<td>45</td>
<td>332</td>
<td>145</td>
<td>43.70%</td>
<td>382</td>
<td>864</td>
</tr>
<tr>
<td>awk</td>
<td>22</td>
<td>2322</td>
<td>785</td>
<td>33.77%</td>
<td>2322</td>
<td>4844</td>
</tr>
<tr>
<td>bash</td>
<td>2</td>
<td>171</td>
<td>0</td>
<td>9.41%</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>matlab</td>
<td>1</td>
<td>6</td>
<td>0</td>
<td>0.00%</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Total</td>
<td>30421</td>
<td>647591</td>
<td>194614</td>
<td>29.42%</td>
<td>759484</td>
<td>346981</td>
</tr>
</tbody>
</table>

Total: 30421 file(s)
#### ohcount: Line Count of gcc-4.5.0/gcc

<table>
<thead>
<tr>
<th>Language</th>
<th>Files</th>
<th>Code</th>
<th>Comment</th>
<th>Comment %</th>
<th>Blank</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>c</td>
<td>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>1466</td>
<td>16130</td>
</tr>
<tr>
<td>objectivec</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>21.7%</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>Total</td>
<td>30747</td>
<td>2406202</td>
<td>677035</td>
<td>22.0%</td>
<td>613025</td>
<td>369626</td>
</tr>
</tbody>
</table>

Essential Abstractions in GCC
GCC Resource Center, IIT Bombay

#### ohcount: Line Count of gcc-4.6.0/gcc

<table>
<thead>
<tr>
<th>Language</th>
<th>Files</th>
<th>Code</th>
<th>Comment</th>
<th>Comment %</th>
<th>Blank</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>c</td>
<td>15787</td>
<td>1462494</td>
<td>321820</td>
<td>18.0%</td>
<td>324179</td>
<td>2108493</td>
</tr>
<tr>
<td>ada</td>
<td>4595</td>
<td>678362</td>
<td>315396</td>
<td>31.7%</td>
<td>233868</td>
<td>1227626</td>
</tr>
<tr>
<td>cpp</td>
<td>8666</td>
<td>252213</td>
<td>61026</td>
<td>19.5%</td>
<td>67144</td>
<td>380383</td>
</tr>
<tr>
<td>fortranfixed</td>
<td>2850</td>
<td>93549</td>
<td>1878</td>
<td>2.0%</td>
<td>13260</td>
<td>108687</td>
</tr>
<tr>
<td>assembler</td>
<td>137</td>
<td>31548</td>
<td>7446</td>
<td>19.1%</td>
<td>4857</td>
<td>43851</td>
</tr>
<tr>
<td>autoconf</td>
<td>3</td>
<td>28775</td>
<td>12</td>
<td>0.0%</td>
<td>4020</td>
<td>32807</td>
</tr>
<tr>
<td>objectivec</td>
<td>861</td>
<td>27465</td>
<td>4822</td>
<td>14.9%</td>
<td>7967</td>
<td>40254</td>
</tr>
<tr>
<td>fortranfree</td>
<td>783</td>
<td>12903</td>
<td>2936</td>
<td>18.5%</td>
<td>1595</td>
<td>17434</td>
</tr>
<tr>
<td>scheme</td>
<td>6</td>
<td>11023</td>
<td>1010</td>
<td>8.4%</td>
<td>1205</td>
<td>13238</td>
</tr>
<tr>
<td>make</td>
<td>4</td>
<td>6078</td>
<td>1070</td>
<td>15.0%</td>
<td>893</td>
<td>8041</td>
</tr>
<tr>
<td>tex</td>
<td>1</td>
<td>5441</td>
<td>2835</td>
<td>34.3%</td>
<td>702</td>
<td>8978</td>
</tr>
<tr>
<td>ocaml</td>
<td>6</td>
<td>2814</td>
<td>576</td>
<td>17.0%</td>
<td>378</td>
<td>3768</td>
</tr>
<tr>
<td>shell</td>
<td>16</td>
<td>1980</td>
<td>597</td>
<td>23.2%</td>
<td>338</td>
<td>2915</td>
</tr>
<tr>
<td>awk</td>
<td>9</td>
<td>1635</td>
<td>323</td>
<td>16.5%</td>
<td>251</td>
<td>2209</td>
</tr>
<tr>
<td>perl</td>
<td>3</td>
<td>886</td>
<td>225</td>
<td>20.5%</td>
<td>158</td>
<td>1249</td>
</tr>
<tr>
<td>haskell</td>
<td>49</td>
<td>153</td>
<td>0</td>
<td>0.0%</td>
<td>17</td>
<td>170</td>
</tr>
<tr>
<td>matlab</td>
<td>1</td>
<td>5</td>
<td>0</td>
<td>0.0%</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>Total</td>
<td>33777</td>
<td>2617304</td>
<td>721972</td>
<td>21.6%</td>
<td>660832</td>
<td>400001</td>
</tr>
</tbody>
</table>

Essential Abstractions in GCC
GCC Resource Center, IIT Bombay
Why is Understanding GCC Difficult?

Deeper technical reasons

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

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

The Architecture of GCC

Compiler Generation Framework

Source Program

Assembly Program

Essential Abstractions in GCC

GCC Resource Center, IIT Bombay
An Example of The Generation Related Gap

- Predicate function for invoking the loop distribution pass
  
  ```c
  static bool
gate_tree_loop_distribution (void)
  {
      return flag_tree_loop_distribution != 0;
  }
  ```

- 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
  
  ```sh
  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
Locating the main function in the directory gcc-4.6.0/gcc using cscope -R

- 7027 occurrences!
- What if we do not search recursively?

<table>
<thead>
<tr>
<th>File</th>
<th>Line</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>collect2.c</td>
<td>1076</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>fp-test.c</td>
<td>85</td>
<td>main (void )</td>
</tr>
<tr>
<td>gcc.c</td>
<td>6092</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>gcov-dump.c</td>
<td>76</td>
<td>main (int argc ATTRIBUTE_UNUSED, char **argv)</td>
</tr>
<tr>
<td>gcov-iov.c</td>
<td>29</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>gcov.c</td>
<td>360</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>genattr.c</td>
<td>164</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>genattrtab.c</td>
<td>4820</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>genautomata.c</td>
<td>9459</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>gencode.b.c</td>
<td>97</td>
<td>main (int argc, char ** argc)</td>
</tr>
<tr>
<td>gencode.c</td>
<td>51</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>genconditions.c</td>
<td>209</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>gencodeconfig.c</td>
<td>261</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>gencodeconstants.c</td>
<td>79</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>genemit.c</td>
<td>830</td>
<td>main (int argc, char **argv)</td>
</tr>
<tr>
<td>genenums.c</td>
<td>48</td>
<td>main (int argc, char **argv)</td>
</tr>
</tbody>
</table>
Another Example of The Generation Related Gap
Locating the main function in the directory gcc-4.6.0/gcc using cscope

g genextract.c 402 main (int argc, char **argv)
h genflags.c 251 main (int argc, char **argv)
i gengenrtl.c 282 main (void)
j gengenp.c 4825 main (int argc, char **argv)
k genhooks.c 335 main (int argc, char **argv)
l genmddeps.c 43 main (int argc, char **argv)
m genmodes.c 1376 main (int argc, char **argv)
n genopinit.c 473 main (int argc, char **argv)
o genoutput.c 999 main (int argc, char **argv)
p genpeep.c 353 main (int argc, char **argv)
q genpreds.c 1388 main (int argc, char **argv)
r genrecog.c 2691 main (int argc, char **argv)
s lto-wraper.c 628 main (int argc, char *argv[]) t main.c 34 main (int argc, char **argv)
u mips-tdump.c 1393 main (int argc, char **argv)
v mips-tfile.c 655 main (void)
w mips-tfile.c 4693 main (int argc, char **argv)
x tlink.c 64 const char *main;

The generated compiler uses an adaptation of the Davison Fraser model
- Generic expander and recognizer
- Machine specific information is isolated in data structures
- Generating a compiler involves generating these data structures
The GCC Challenge: Poor Retargetability Mechanism

Symptoms:

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

Size in terms of line counts in gcc-4.6.0

<table>
<thead>
<tr>
<th>Files</th>
<th>i386</th>
<th>mips</th>
<th>arm</th>
</tr>
</thead>
<tbody>
<tr>
<td>*.md</td>
<td>38722</td>
<td>15534</td>
<td>30938</td>
</tr>
<tr>
<td>*.c</td>
<td>39579</td>
<td>16766</td>
<td>26164</td>
</tr>
<tr>
<td>*.h</td>
<td>17869</td>
<td>5667</td>
<td>18711</td>
</tr>
<tr>
<td>Total</td>
<td>96170</td>
<td>37969</td>
<td>75913</td>
</tr>
</tbody>
</table>

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

Meeting the GCC Challenge

<table>
<thead>
<tr>
<th>Goal of Understanding</th>
<th>Methodology</th>
<th>Needs Examining</th>
</tr>
</thead>
<tbody>
<tr>
<td></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>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>Retargetability</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mechanism</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>