

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

Uday Khedker

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

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



January 2012

Outline

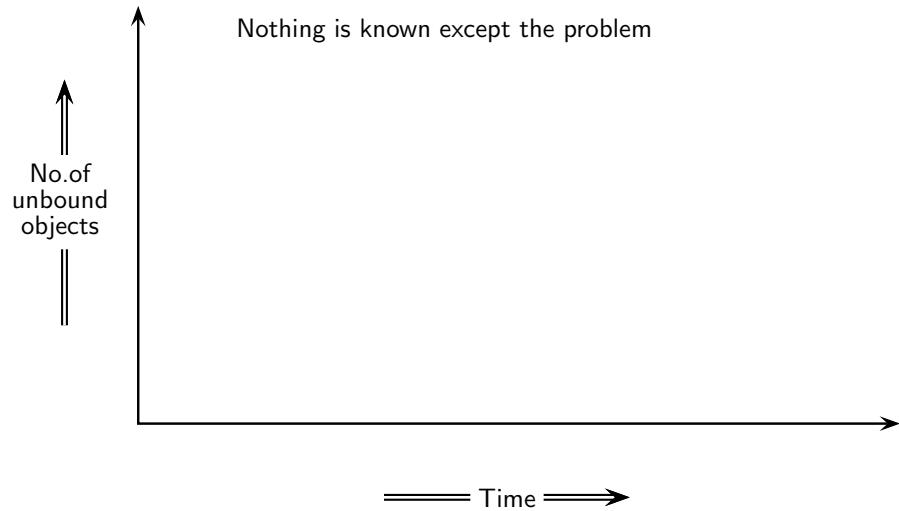
- An Overview of Compilation
Introduction, compilation sequence, compilation models
- GCC: The Great Compiler Challenge
Difficulties in understanding GCC
- Meeting the GCC Challenge: CS 715
The course plan



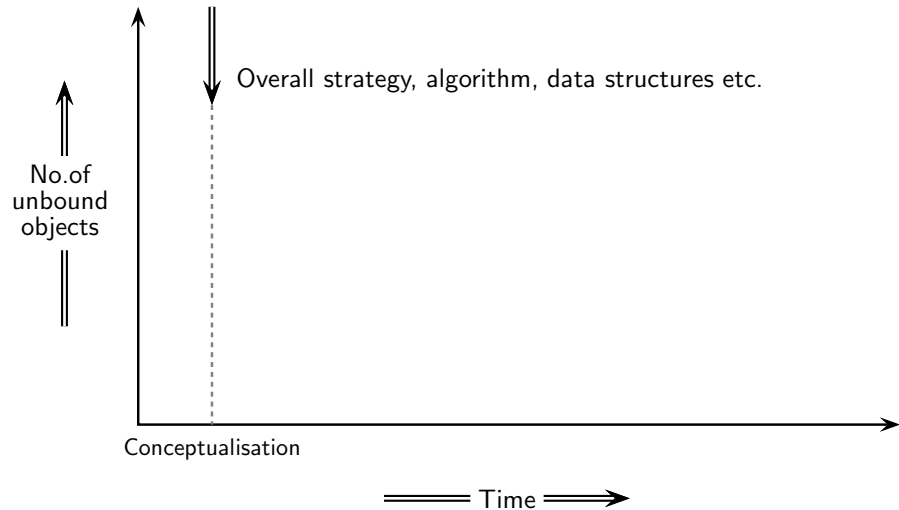
Part 1

Introduction to Compilation

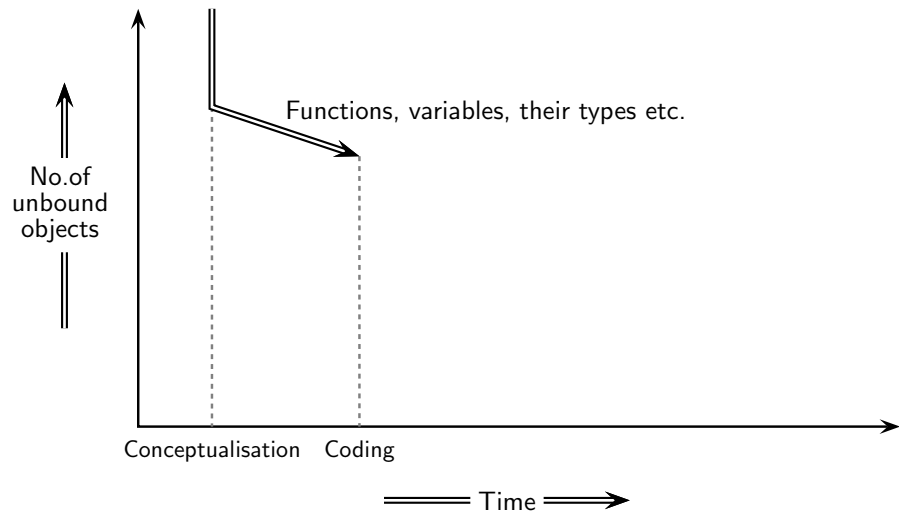
Binding



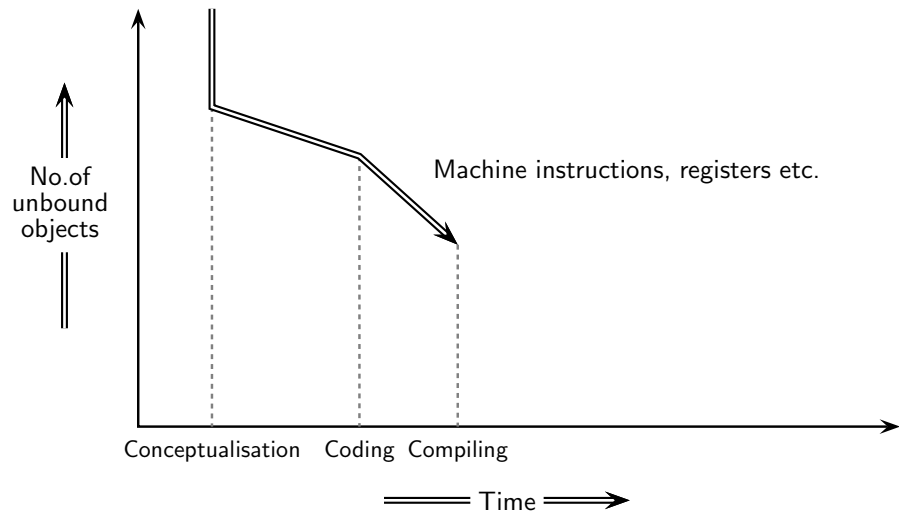
Binding



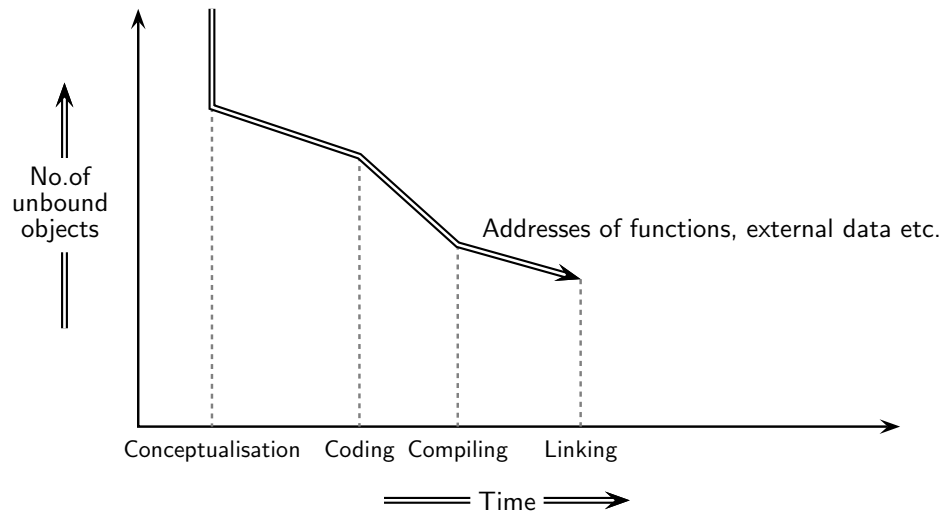
Binding



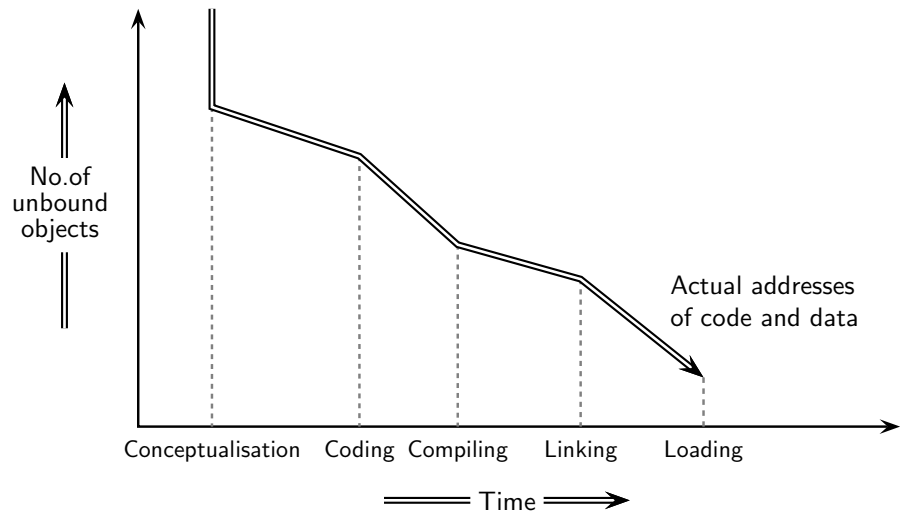
Binding



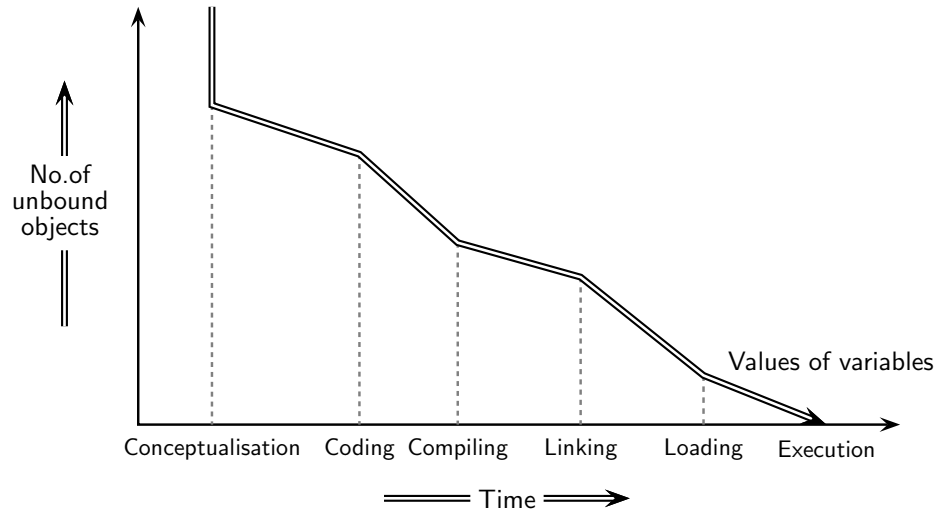
Binding



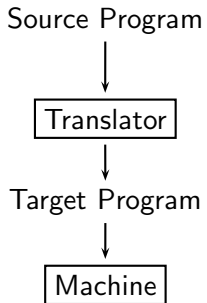
Binding



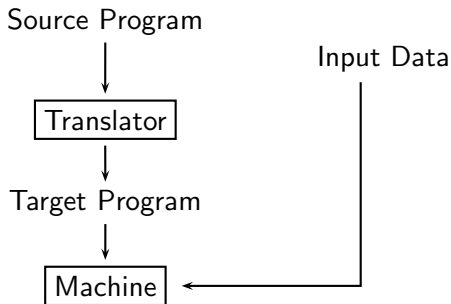
Binding



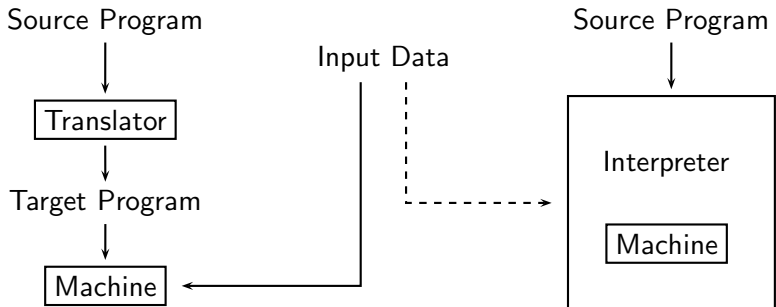
Implementation Mechanisms



Implementation Mechanisms



Implementation Mechanisms



Implementation Mechanisms as “Bridges”

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

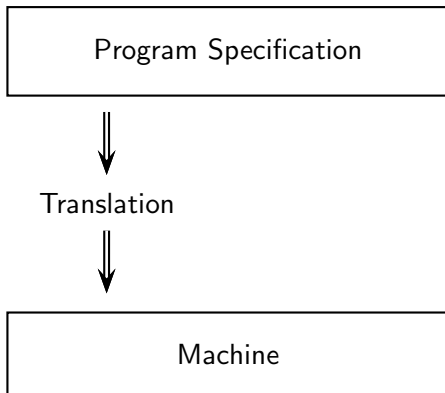
Program Specification

Machine



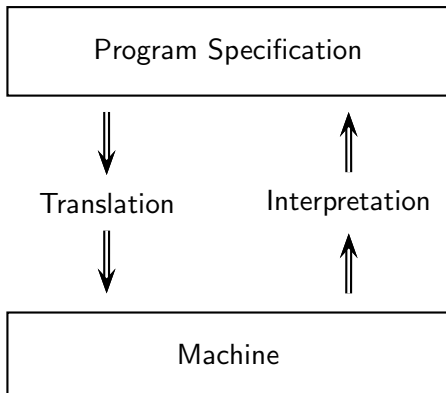
Implementation Mechanisms as “Bridges”

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



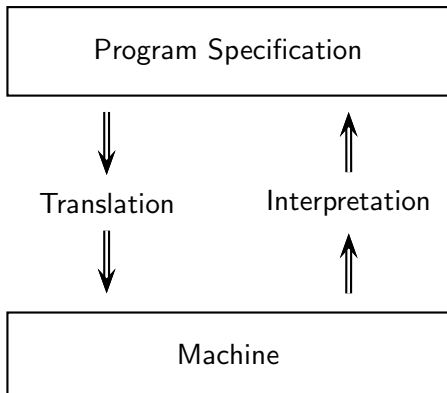
Implementation Mechanisms as “Bridges”

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



Implementation Mechanisms as “Bridges”

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



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

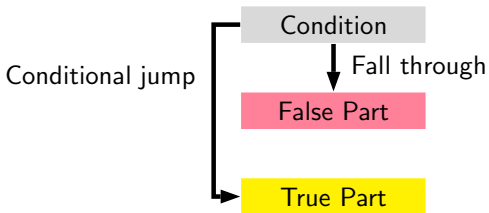
```
lw    $t0, 4($fp) ;    t0 <- b           # Is b smaller
slti  $t0, $t0, 10 ;   t0 <- t0 < 10      # than 10?
not   $t0, $t0        ;   t0 <- !t0
bgtz  $t0, L0:        ;   if t0>=0 goto L0
lw    $t0, 4($fp) ;    t0 <- b           # YES
b     L1:              ;   goto L1
L0:  lw    $t0, 8($fp) ;L0: 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) ;    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

NOT Condition

Input C statement

True Part

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

False Part

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

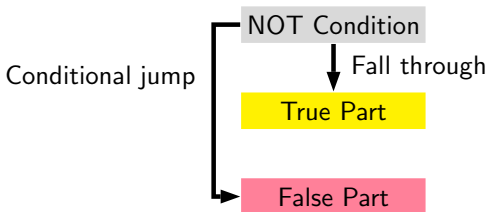
```



High and Low Level Abstractions

Input C statement

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



Spim Assembly Equivalent

```

lw    $t0, 4($fp) ;    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
  
```



Implementation Mechanisms

- Translation = Analysis + Synthesis
Interpretation = Analysis + Execution



Implementation Mechanisms

- Translation = Analysis + Synthesis
Interpretation = Analysis + Execution

- Translation Instructions \implies Equivalent Instructions



Implementation Mechanisms

- Translation = Analysis + Synthesis

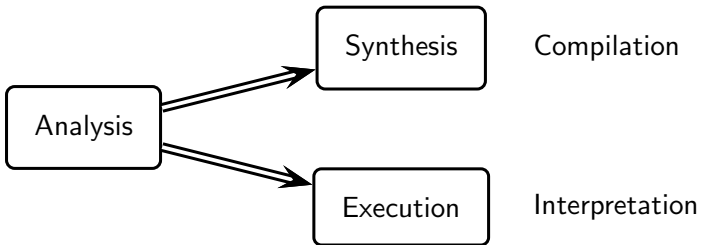
Interpretation = Analysis + Execution

- Translation Instructions \implies Equivalent Instructions

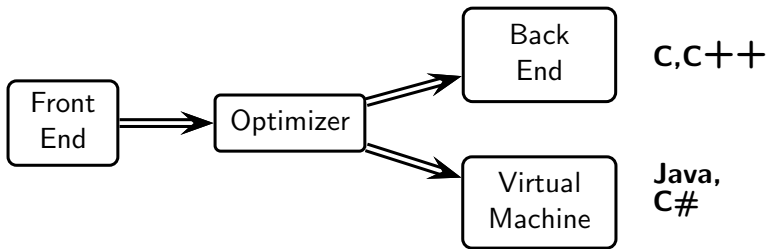
Interpretation Instructions \implies Actions Implied
by Instructions



Language Implementation Models



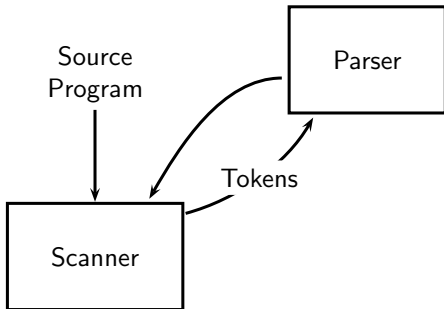
Language Processor Models



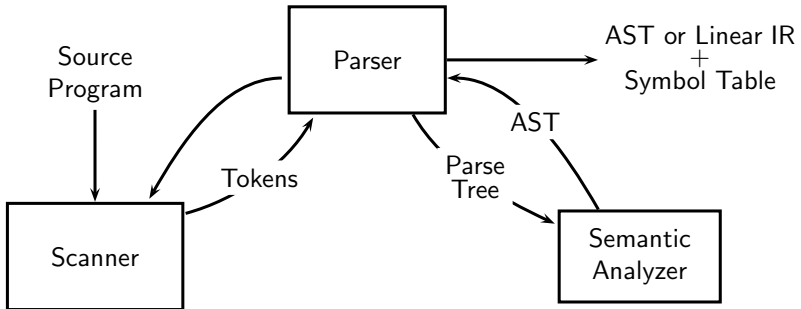
Typical Front Ends



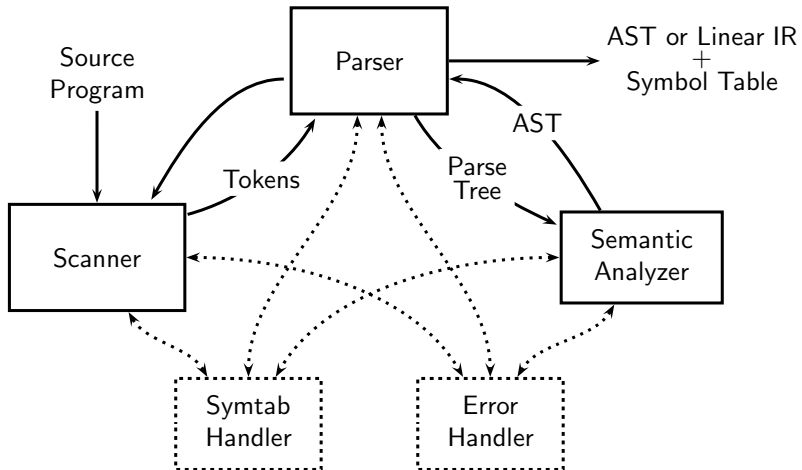
Typical Front Ends



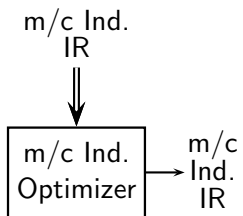
Typical Front Ends



Typical Front Ends



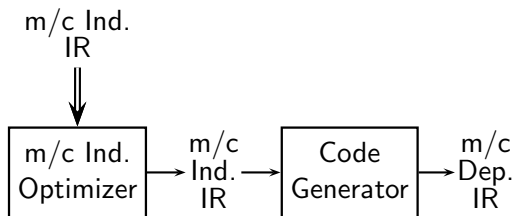
Typical Back Ends



- Compile time evaluations
- Eliminating redundant computations



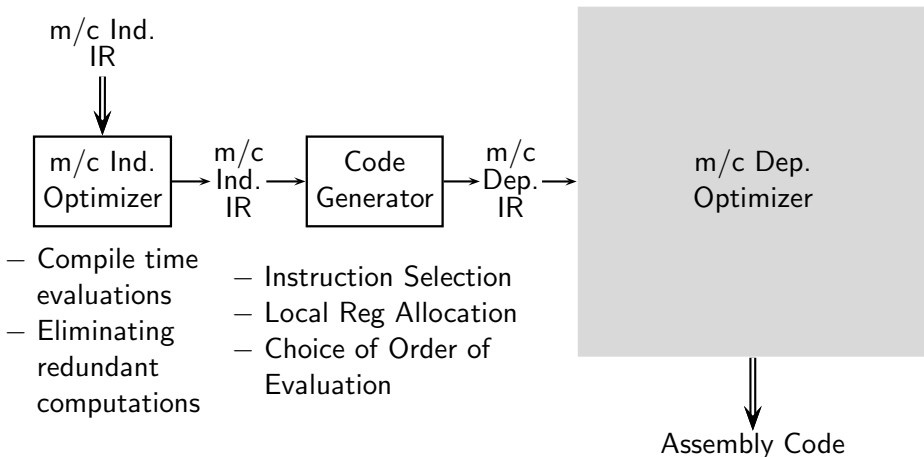
Typical Back Ends



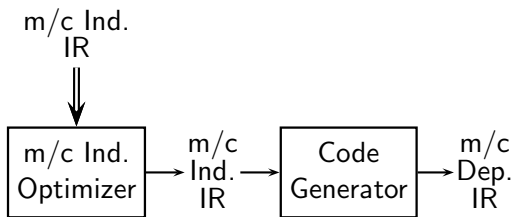
- Compile time evaluations
- Eliminating redundant computations
- Instruction Selection
- Local Reg Allocation
- Choice of Order of Evaluation



Typical Back Ends

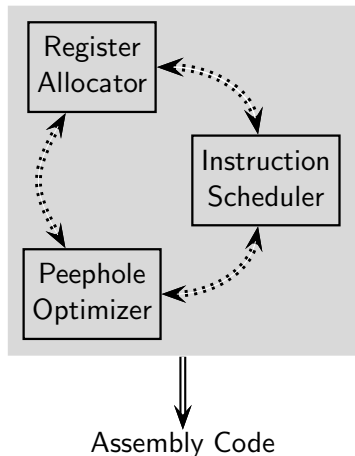


Typical Back Ends



- Compile time evaluations
- Eliminating redundant computations

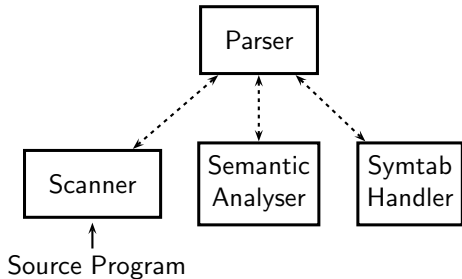
- Instruction Selection
- Local Reg Allocation
- Choice of Order of Evaluation



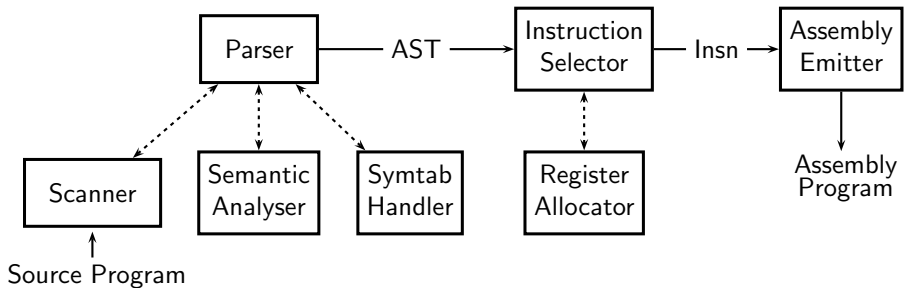
Part 2

An Overview of Compilation Phases

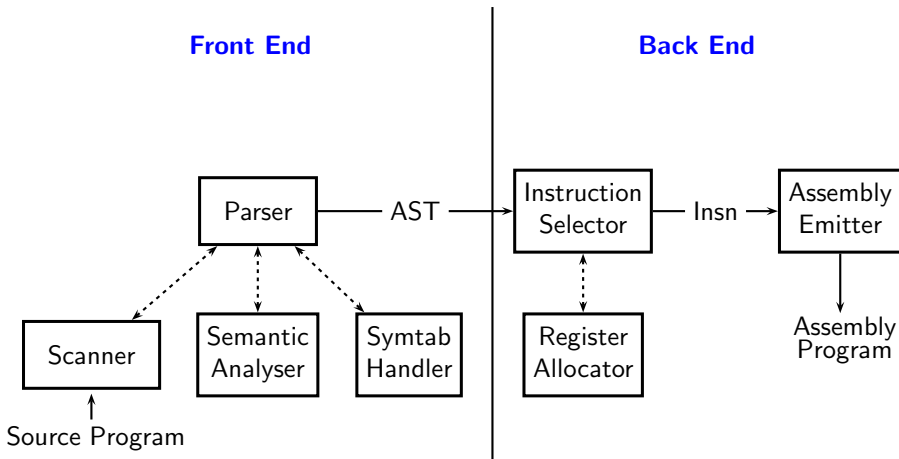
The Structure of a Simple Compiler



The Structure of a Simple Compiler



The Structure of a Simple Compiler



Translation Sequence in Our Compiler: Parsing

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

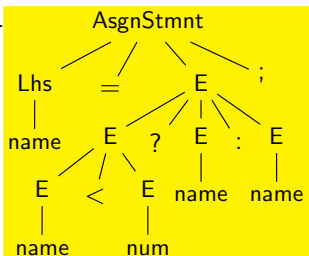
Input



Translation Sequence in Our Compiler: Parsing

a=b<10?b:c;

Input



Parse Tree

Issues:

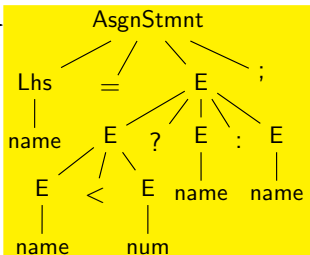
- Grammar rules, terminals, non-terminals
- Order of application of grammar rules
eg. is it (a = b<10?) followed by (b:c)?
- Values of terminal symbols
eg. string "10" vs. integer number 10.



Translation Sequence in Our Compiler: Semantic Analysis

a=b<10?b:c;

Input



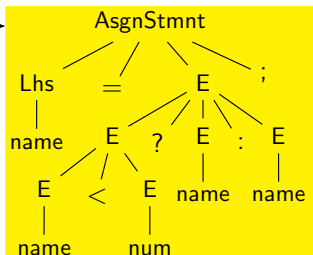
Parse Tree



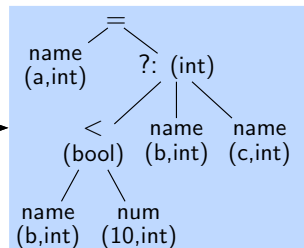
Translation Sequence in Our Compiler: Semantic Analysis

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

Input



Parse Tree



Abstract Syntax Tree
(with attributes)

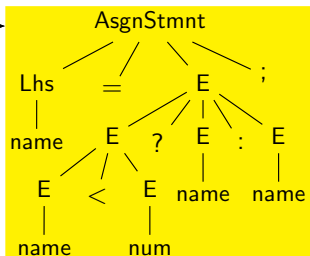
Issues:

- 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`

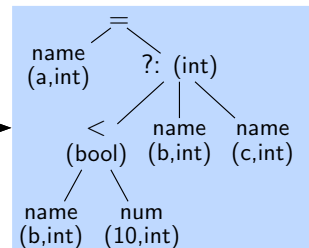


Translation Sequence in Our Compiler: IR Generation

a=b<10?b:c;
Input



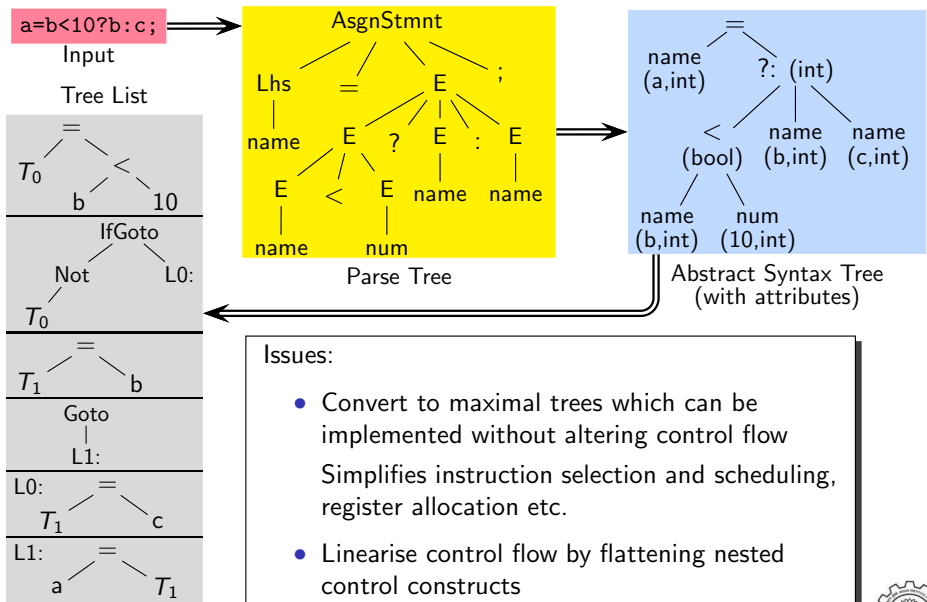
Parse Tree



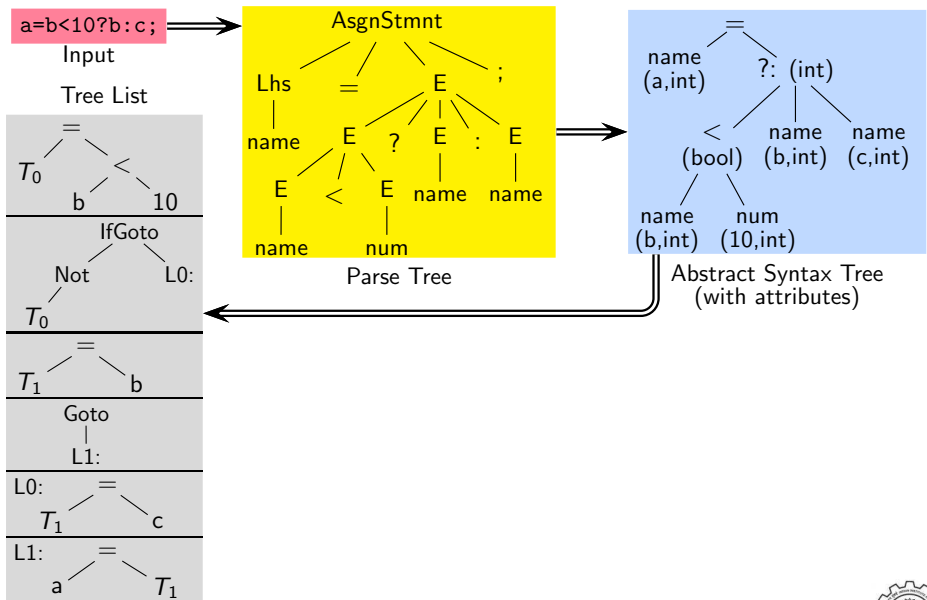
Abstract Syntax Tree
(with attributes)



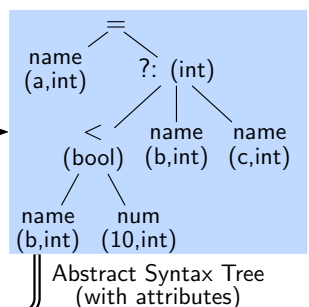
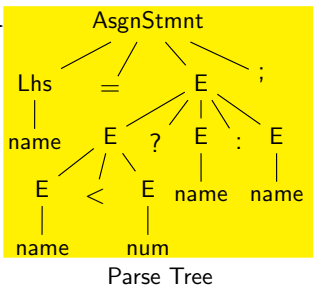
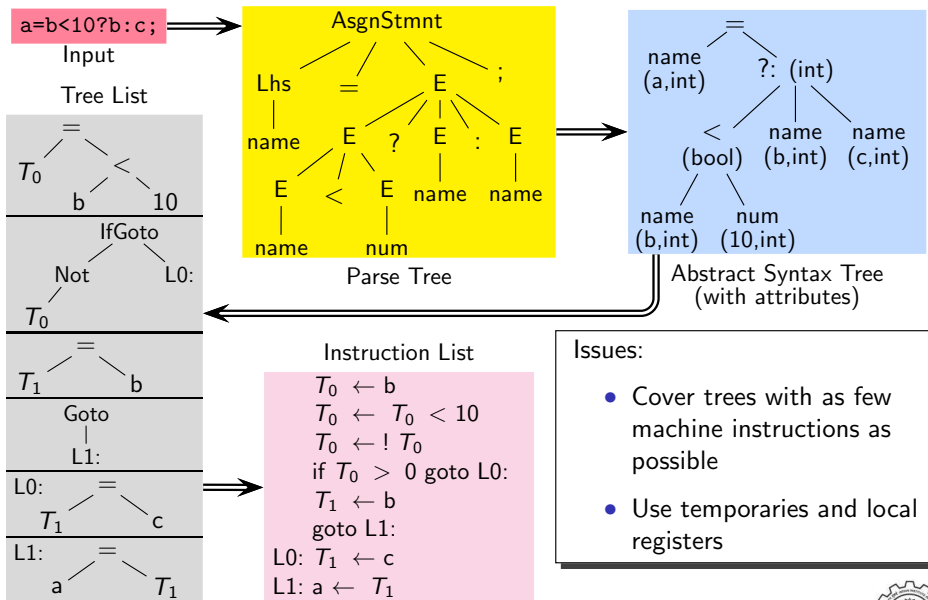
Translation Sequence in Our Compiler: IR Generation



Translation Sequence in Our Compiler: Instruction Selection



Translation Sequence in Our Compiler: Instruction Selection



Instruction List

```

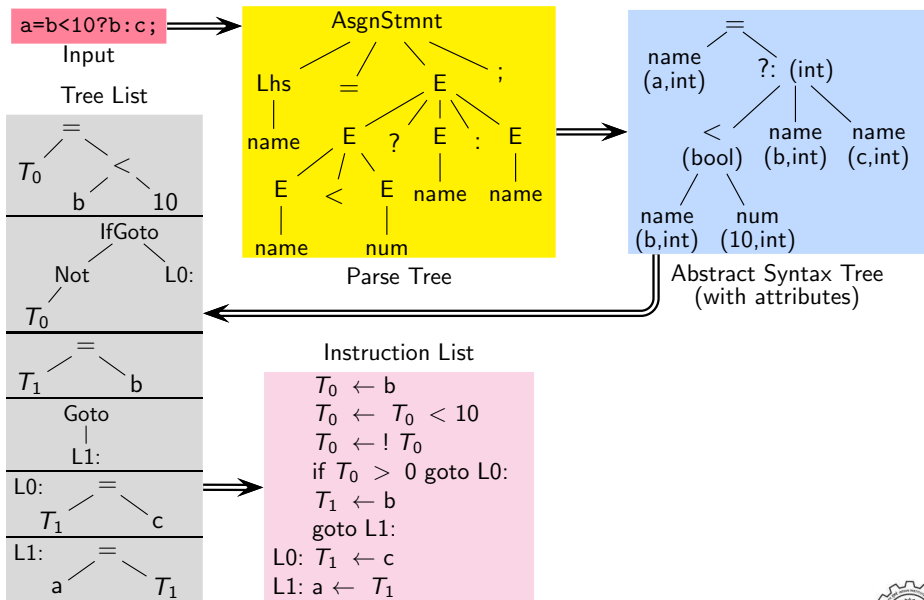
T0 ← b
T0 ← T0 < 10
T0 ← ! T0
if T0 > 0 goto L0:
T1 ← b
goto L1:
L0: T1 ← c
L1: a ← T1
  
```

Issues:

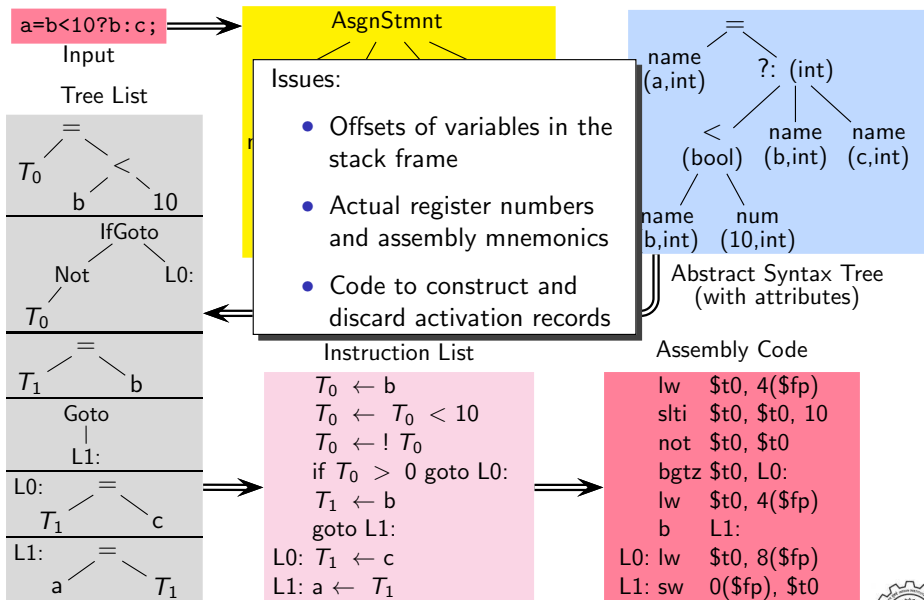
- Cover trees with as few machine instructions as possible
- Use temporaries and local registers



Translation Sequence in Our Compiler: Emitting Instructions



Translation Sequence in Our Compiler: Emitting Instructions



AsgnStmnt

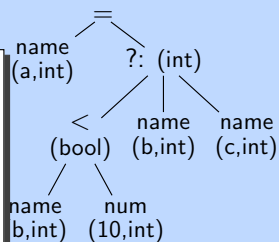
Issues:

- Offsets of variables in the stack frame
- Actual register numbers and assembly mnemonics
- Code to construct and discard activation records

Instruction List

```

T0 ← b
T0 ← T0 < 10
T0 ← ! T0
if T0 > 0 goto L0:
T1 ← b
goto L1:
L0: T1 ← c
L1: a ← T1
  
```



Abstract Syntax Tree (with attributes)

Assembly Code

```

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
  
```



Part 3

*Compilation Models,
Instruction Selection, and
Retargetability*

Compilation Models

*Aho Ullman
Model*

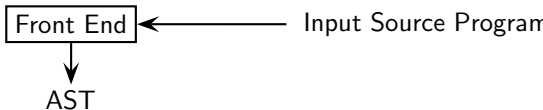
*Davidson Fraser
Model*



Compilation Models

*Aho Ullman
Model*

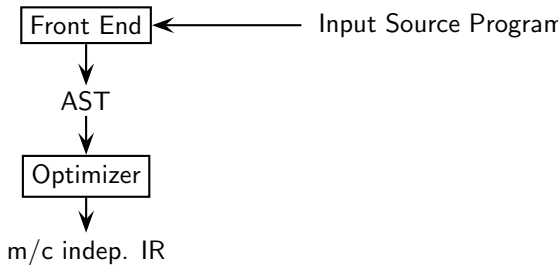
*Davidson Fraser
Model*



Compilation Models

*Aho Ullman
Model*

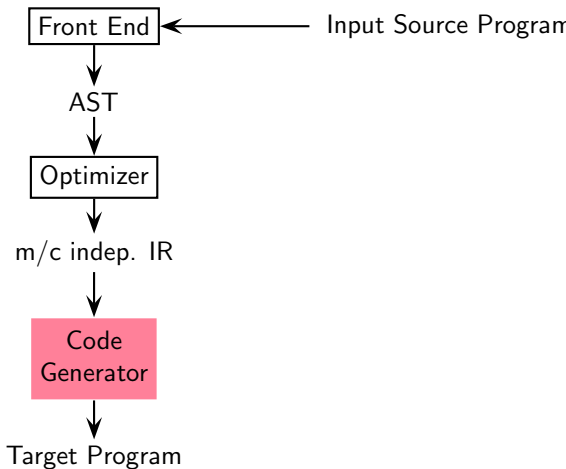
*Davidson Fraser
Model*



Compilation Models

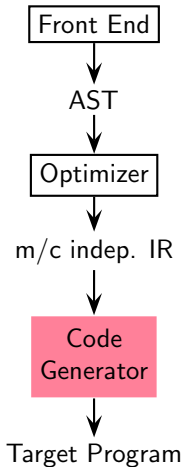
*Aho Ullman
Model*

*Davidson Fraser
Model*

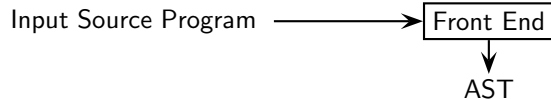


Compilation Models

Aho Ullman Model

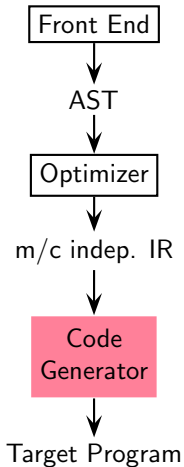


Davidson Fraser Model

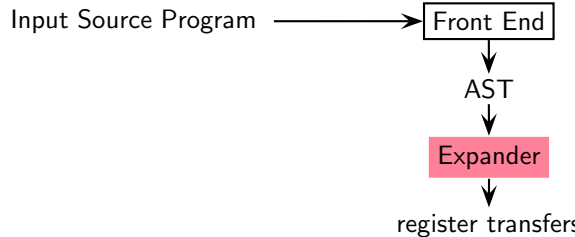


Compilation Models

Aho Ullman Model

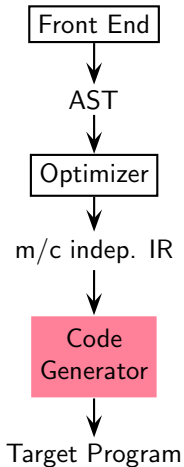


Davidson Fraser Model

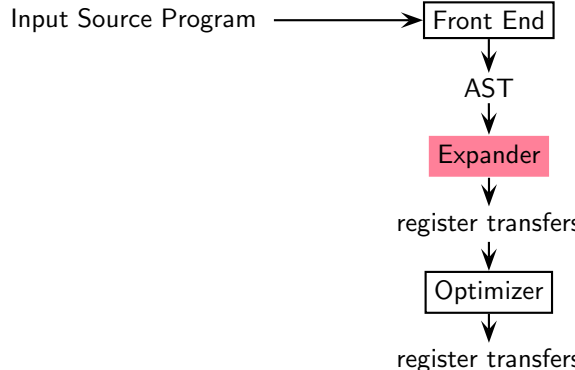


Compilation Models

Aho Ullman Model

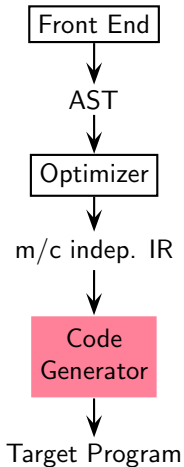


Davidson Fraser Model

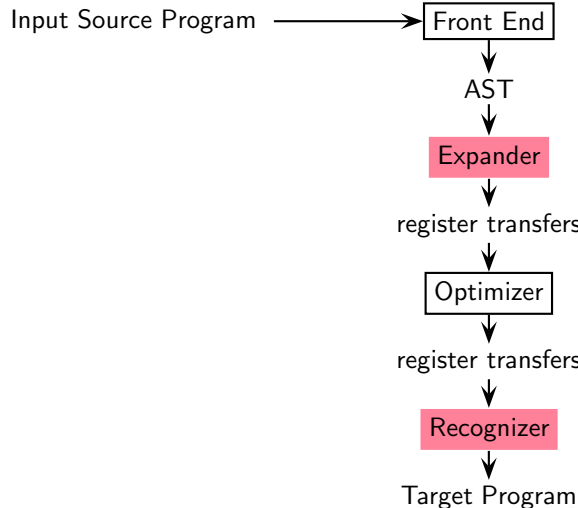


Compilation Models

Aho Ullman Model

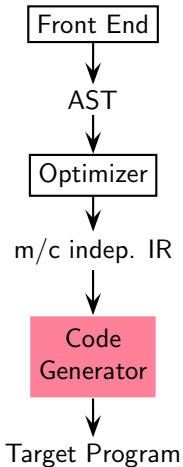


Davidson Fraser Model



Compilation Models

Aho Ullman Model



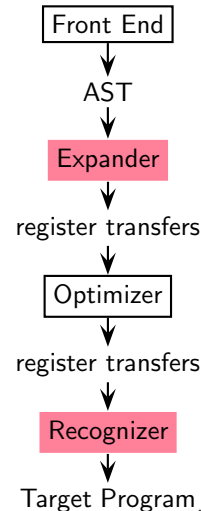
Aho Ullman: Instruction selection

- 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



Retargetability in Aho Ullman and Davidson Fraser Models

	Aho Ullman Model	Davidson Fraser Model
Instruction Selection	<ul style="list-style-type: none">• Machine independent IR is expressed in the form of trees• Machine instructions are described in the form of trees• Trees in the IR are “covered” using the instruction trees	
Optimization		



Retargetability in Aho Ullman and Davidson Fraser Models

	Aho Ullman Model	Davidson Fraser Model
Instruction Selection	<ul style="list-style-type: none">• Machine independent IR is expressed in the form of trees• Machine instructions are described in the form of trees• Trees in the IR are “covered” using the instruction trees	
	Cost based tree pattern matching	
Optimization		



Retargetability in Aho Ullman and Davidson Fraser Models

	Aho Ullman Model	Davidson Fraser Model
Instruction Selection	<ul style="list-style-type: none"> • Machine independent IR is expressed in the form of trees • Machine instructions are described in the form of trees • Trees in the IR are “covered” using the instruction trees 	
	Cost based tree pattern matching	Structural tree pattern matching
Optimization		



Retargetability in Aho Ullman and Davidson Fraser Models

	Aho Ullman Model	Davidson Fraser Model
Instruction Selection	<ul style="list-style-type: none"> Machine independent IR is expressed in the form of trees Machine instructions are described in the form of trees Trees in the IR are “covered” using the instruction trees 	
	Cost based tree pattern matching	Structural tree pattern matching
Optimization	Machine independent	



Retargetability in Aho Ullman and Davidson Fraser Models

	Aho Ullman Model	Davidson Fraser Model
Instruction Selection	<ul style="list-style-type: none"> Machine independent IR is expressed in the form of trees Machine instructions are described in the form of trees Trees in the IR are “covered” using the instruction trees 	
	Cost based tree pattern matching	Structural tree pattern matching
Optimization	Machine independent	Machine dependent



Retargetability in Aho Ullman and Davidson Fraser Models

	Aho Ullman Model	Davidson Fraser Model
Instruction Selection	<ul style="list-style-type: none"> Machine independent IR is expressed in the form of trees Machine instructions are described in the form of trees Trees in the IR are “covered” using the instruction trees 	
	Cost based tree pattern matching	Structural tree pattern matching
Optimization	Machine independent	Machine dependent
		Key Insight: <i>Register transfers are target specific but their form is target independent</i>



Part 4

*$GCC \equiv$ The **G**reat **C**ompiler **C**hallenge*

What is GCC?

- For the GCC developer community: [The GNU Compiler Collection](#)
- For other compiler writers: [The Great Compiler Challenge](#) 😊



The Gnu Tool Chain for C

Source Program

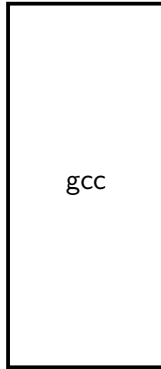


Target Program

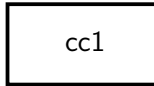


The Gnu Tool Chain for C

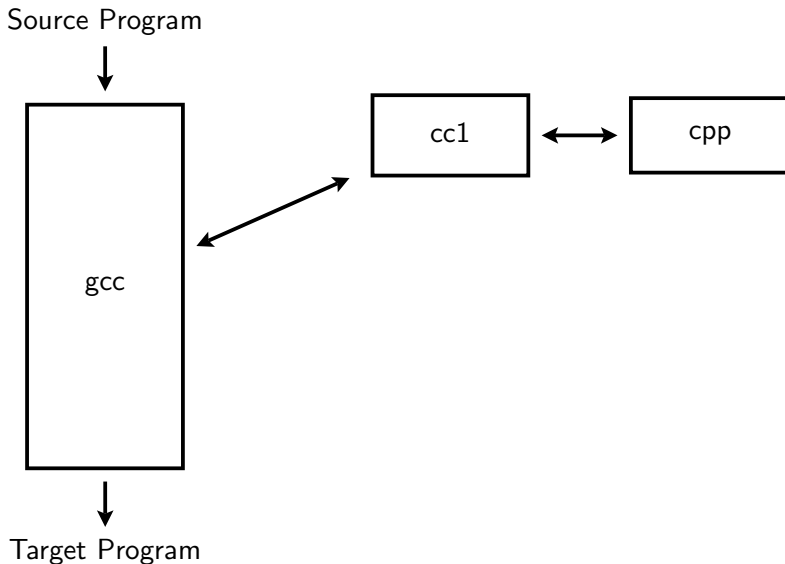
Source Program



Target Program

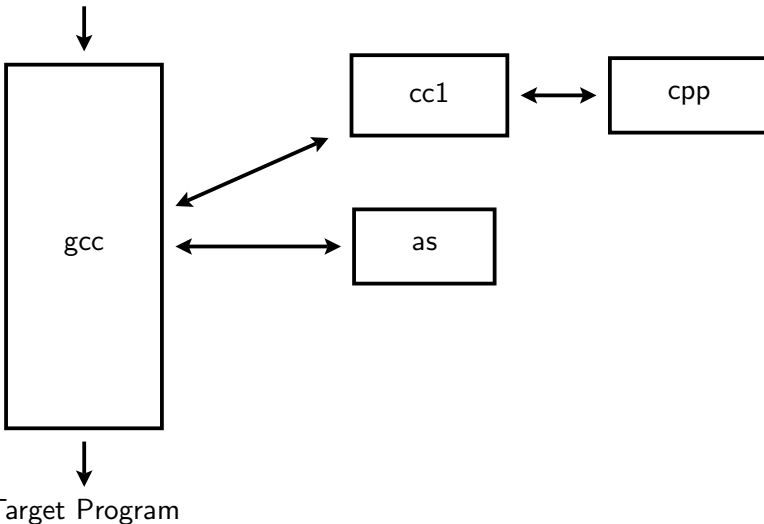


The Gnu Tool Chain for C



The Gnu Tool Chain for C

Source Program

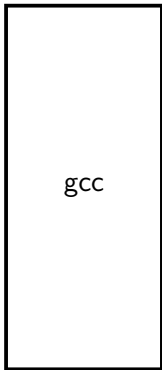


Target Program

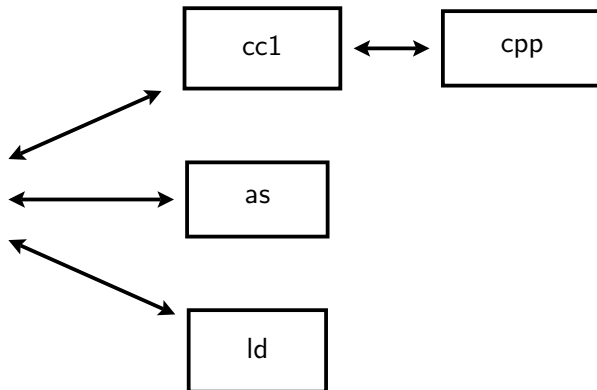


The Gnu Tool Chain for C

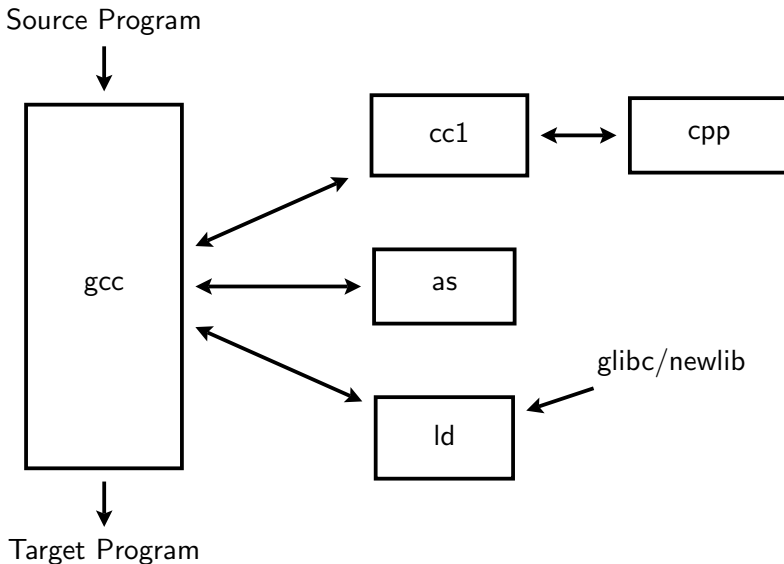
Source Program



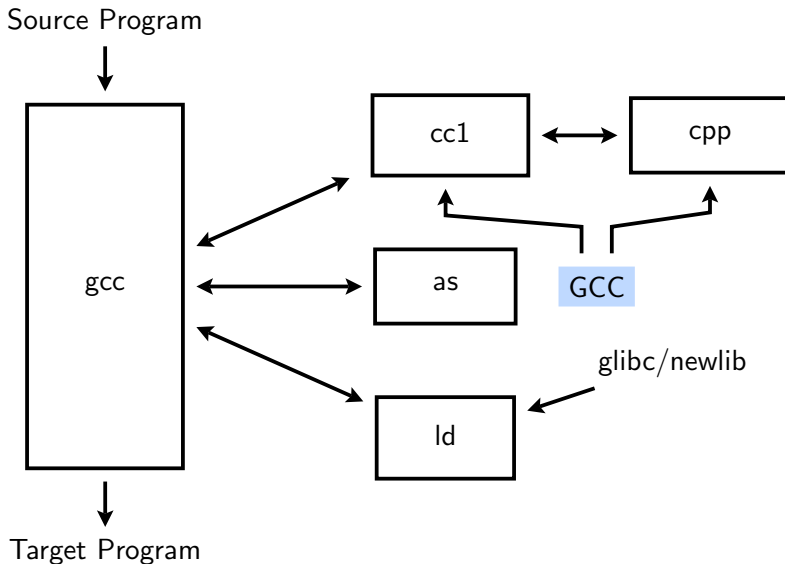
Target Program



The Gnu Tool Chain for C



The Gnu Tool Chain for C



Why is Understanding GCC Difficult?

Some of the obvious reasons:

- **Comprehensiveness**

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

- **Open development model**

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

- **Rapid versioning**

GCC maintenance is a race against time. Disruptive corrections are difficult



Open Source and Free Software Development Model

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



Open Source and Free Software Development Model

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

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



Open Source and Free Software Development Model

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

- Cathedral: Total Centralized Control

Design, implement, test, release

- Bazaar: Total Decentralization

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



Open Source and Free Software Development Model

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

- Cathedral: Total Centralized Control

Design, implement, test, release

- Bazaar: Total Decentralization

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

“Given enough eyeballs, all bugs are shallow”



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



Open Source and Free Software Development Model

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

- Cathedral: Total Centralized Control

Design, implement, test, release

- Bazaar: Total Decentralization

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

“Given enough eyeballs, all bugs are shallow”

Code errors, logical errors, and architectural errors

A combination of the two seems more sensible



The Current Development Model of GCC

GCC follows a combination of the Cathedral and the Bazaar approaches

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



Comprehensiveness of GCC: Wide Applicability

- **Input languages supported:**
C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada
- Processors supported in standard releases:



Comprehensiveness of GCC: Wide Applicability

- Input languages supported:
C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada
- Processors supported in standard releases:
 - ▶ Common processors:
 - ▶ Lesser-known target processors:
 - ▶ Additional processors independently supported:



Comprehensiveness of GCC: Wide Applicability

- **Input languages supported:**
C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada
- **Processors supported in standard releases:**
 - ▶ **Common processors:**
Alpha,
 - ▶ **Lesser-known target processors:**
 - ▶ **Additional processors independently supported:**



Comprehensiveness of GCC: Wide Applicability

- Input languages supported:
C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada
- Processors supported in standard releases:
 - ▶ Common processors:
Alpha, ARM,
 - ▶ Lesser-known target processors:
 - ▶ Additional processors independently supported:



Comprehensiveness of GCC: Wide Applicability

- Input languages supported:
C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada
- Processors supported in standard releases:
 - ▶ Common processors:
Alpha, ARM, Atmel AVR,
 - ▶ Lesser-known target processors:
 - ▶ Additional processors independently supported:



Comprehensiveness of GCC: Wide Applicability

- Input languages supported:
C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada
- Processors supported in standard releases:
 - ▶ Common processors:
Alpha, ARM, Atmel AVR, Blackfin,
 - ▶ Lesser-known target processors:
 - ▶ Additional processors independently supported:



Comprehensiveness of GCC: Wide Applicability

- Input languages supported:
C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada
- Processors supported in standard releases:
 - ▶ Common processors:
Alpha, ARM, Atmel AVR, Blackfin, HC12,
 - ▶ Lesser-known target processors:
 - ▶ Additional processors independently supported:



Comprehensiveness of GCC: Wide Applicability

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



Comprehensiveness of GCC: Wide Applicability

- **Input languages supported:**
C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada
- **Processors supported in standard releases:**
 - ▶ **Common processors:**
Alpha, ARM, Atmel AVR, Blackfin, HC12, H8/300, IA-32 (x86),
 - ▶ **Lesser-known target processors:**
 - ▶ **Additional processors independently supported:**



Comprehensiveness of GCC: Wide Applicability

- **Input languages supported:**
C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada
- **Processors supported in standard releases:**
 - ▶ **Common processors:**
Alpha, ARM, Atmel AVR, Blackfin, HC12, H8/300, IA-32 (x86), x86-64,
 - ▶ **Lesser-known target processors:**
 - ▶ **Additional processors independently supported:**



Comprehensiveness of GCC: Wide Applicability

- Input languages supported:
C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada
- Processors supported in standard releases:
 - ▶ Common processors:
Alpha, ARM, Atmel AVR, Blackfin, HC12, H8/300, IA-32 (x86), x86-64, IA-64,
 - ▶ Lesser-known target processors:
 - ▶ Additional processors independently supported:



Comprehensiveness of GCC: Wide Applicability

- **Input languages supported:**
C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada
- **Processors supported in standard releases:**
 - ▶ **Common processors:**
Alpha, ARM, Atmel AVR, Blackfin, HC12, H8/300, IA-32 (x86), x86-64, IA-64, Motorola 68000,
 - ▶ **Lesser-known target processors:**
 - ▶ **Additional processors independently supported:**



Comprehensiveness of GCC: Wide Applicability

- **Input languages supported:**
C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada
- **Processors supported in standard releases:**
 - ▶ **Common processors:**
Alpha, ARM, Atmel AVR, Blackfin, HC12, H8/300, IA-32 (x86), x86-64, IA-64, Motorola 68000, MIPS,
 - ▶ **Lesser-known target processors:**
 - ▶ **Additional processors independently supported:**



Comprehensiveness of GCC: Wide Applicability

- **Input languages supported:**
C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada
- **Processors supported in standard releases:**
 - ▶ **Common processors:**
Alpha, ARM, Atmel AVR, Blackfin, HC12, H8/300, IA-32 (x86), x86-64, IA-64, Motorola 68000, MIPS, PA-RISC,
 - ▶ **Lesser-known target processors:**
 - ▶ **Additional processors independently supported:**



Comprehensiveness of GCC: Wide Applicability

- **Input languages supported:**
C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada
- **Processors supported in standard releases:**
 - ▶ **Common processors:**
Alpha, ARM, Atmel AVR, Blackfin, HC12, H8/300, IA-32 (x86), x86-64, IA-64, Motorola 68000, MIPS, PA-RISC, PDP-11,
 - ▶ **Lesser-known target processors:**
 - ▶ **Additional processors independently supported:**



Comprehensiveness of GCC: Wide Applicability

- **Input languages supported:**
C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada
- **Processors supported in standard releases:**
 - ▶ **Common processors:**
Alpha, ARM, Atmel AVR, Blackfin, HC12, H8/300, IA-32 (x86), x86-64, IA-64, Motorola 68000, MIPS, PA-RISC, PDP-11, PowerPC,
 - ▶ **Lesser-known target processors:**
 - ▶ **Additional processors independently supported:**



Comprehensiveness of GCC: Wide Applicability

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



Comprehensiveness of GCC: Wide Applicability

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



Comprehensiveness of GCC: Wide Applicability

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

- ▶ **Additional processors independently supported:**



Comprehensiveness of GCC: Wide Applicability

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

- ▶ **Additional processors independently supported:**



Comprehensiveness of GCC: Wide Applicability

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

- ▶ **Additional processors independently supported:**



Comprehensiveness of GCC: Wide Applicability

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

 - ▶ **Additional processors independently supported:**



Comprehensiveness of GCC: Wide Applicability

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

 - ▶ **Additional processors independently supported:**



Comprehensiveness of GCC: Wide Applicability

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

 - ▶ **Additional processors independently supported:**



Comprehensiveness of GCC: Wide Applicability

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

 - ▶ **Additional processors independently supported:**



Comprehensiveness of GCC: Wide Applicability

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

 - ▶ **Additional processors independently supported:**



Comprehensiveness of GCC: Wide Applicability

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

 - ▶ **Additional processors independently supported:**



Comprehensiveness of GCC: Wide Applicability

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



Comprehensiveness of GCC: Wide Applicability

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



Comprehensiveness of GCC: Wide Applicability

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



Comprehensiveness of GCC: Wide Applicability

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



Comprehensiveness of GCC: Wide Applicability

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



Comprehensiveness of GCC: Wide Applicability

- **Input languages supported:**
C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada
- **Processors supported in standard releases:**
 - ▶ **Common processors:**
Alpha, ARM, Atmel AVR, Blackfin, HC12, H8/300, IA-32 (x86), x86-64, IA-64, Motorola 68000, MIPS, PA-RISC, PDP-11, PowerPC, R8C/M16C/M32C, SPU, System/390/zSeries, SuperH, SPARC, VAX
 - ▶ **Lesser-known target processors:**
A29K, ARC, ETRAX CRIS, D30V, DSP16xx, FR-30, FR-V, Intel i960, IP2000, M32R, 68HC11, MCORE, MMIX, MN10200, MN10300, Motorola 88000, NS32K, ROMP, Stormy16, V850,
 - ▶ **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, V850, Xtensa,
 - ▶ **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, V850, Xtensa, AVR32
 - ▶ **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, V850, Xtensa, AVR32
 - ▶ **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:**
A29K, ARC, ETRAX CRIS, D30V, DSP16xx, FR-30, FR-V, Intel i960, IP2000, M32R, 68HC11, MCORE, MMIX, MN10200, MN10300, Motorola 88000, NS32K, ROMP, Stormy16, V850, Xtensa, AVR32
 - ▶ **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:**
A29K, ARC, ETRAX CRIS, D30V, DSP16xx, FR-30, FR-V, Intel i960, IP2000, M32R, 68HC11, MCORE, MMIX, MN10200, MN10300, Motorola 88000, NS32K, ROMP, Stormy16, V850, Xtensa, AVR32
 - ▶ **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:**
A29K, ARC, ETRAX CRIS, D30V, DSP16xx, FR-30, FR-V, Intel i960, IP2000, M32R, 68HC11, MCORE, MMIX, MN10200, MN10300, Motorola 88000, NS32K, ROMP, Stormy16, V850, Xtensa, AVR32
 - ▶ **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:**
A29K, ARC, ETRAX CRIS, D30V, DSP16xx, FR-30, FR-V, Intel i960, IP2000, M32R, 68HC11, MCORE, MMIX, MN10200, MN10300, Motorola 88000, NS32K, ROMP, Stormy16, V850, Xtensa, AVR32
 - ▶ **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:**
A29K, ARC, ETRAX CRIS, D30V, DSP16xx, FR-30, FR-V, Intel i960, IP2000, M32R, 68HC11, MCORE, MMIX, MN10200, MN10300, Motorola 88000, NS32K, ROMP, Stormy16, V850, Xtensa, AVR32
 - ▶ **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:**
A29K, ARC, ETRAX CRIS, D30V, DSP16xx, FR-30, FR-V, Intel i960, IP2000, M32R, 68HC11, MCORE, MMIX, MN10200, MN10300, Motorola 88000, NS32K, ROMP, Stormy16, V850, Xtensa, AVR32
 - ▶ **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:**
A29K, ARC, ETRAX CRIS, D30V, DSP16xx, FR-30, FR-V, Intel i960, IP2000, M32R, 68HC11, MCORE, MMIX, MN10200, MN10300, Motorola 88000, NS32K, ROMP, Stormy16, V850, Xtensa, AVR32
 - ▶ **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:**
A29K, ARC, ETRAX CRIS, D30V, DSP16xx, FR-30, FR-V, Intel i960, IP2000, M32R, 68HC11, MCORE, MMIX, MN10200, MN10300, Motorola 88000, NS32K, ROMP, Stormy16, V850, Xtensa, AVR32
 - ▶ **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:**
A29K, ARC, ETRAX CRIS, D30V, DSP16xx, FR-30, FR-V, Intel i960, IP2000, M32R, 68HC11, MCORE, MMIX, MN10200, MN10300, Motorola 88000, NS32K, ROMP, Stormy16, V850, Xtensa, AVR32
 - ▶ **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:**
A29K, ARC, ETRAX CRIS, D30V, DSP16xx, FR-30, FR-V, Intel i960, IP2000, M32R, 68HC11, MCORE, MMIX, MN10200, MN10300, Motorola 88000, NS32K, ROMP, Stormy16, V850, Xtensa, AVR32
 - ▶ **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:**
A29K, ARC, ETRAX CRIS, D30V, DSP16xx, FR-30, FR-V, Intel i960, IP2000, M32R, 68HC11, MCORE, MMIX, MN10200, MN10300, Motorola 88000, NS32K, ROMP, Stormy16, V850, Xtensa, AVR32
 - ▶ **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



ohcount: Line Count of gcc-4.4.2

Total: 66139 file(s)

Language	Files	Code	Comment	Comment %	Blank	Total
c	15638	1840245	394682	17.7%	366815	2601742
cpp	19622	872775	190744	17.9%	189007	1252526
java	6342	681656	643045	48.5%	169465	1494166
ada	4206	638557	294881	31.6%	218000	1151438
autoconf	76	445046	393	0.1%	58831	504270
make	82	110064	3268	2.9%	13270	126602
html	480	103080	5658	5.2%	21438	130176
fortranfixed	2164	73366	1570	2.1%	9454	84390
assembler	183	42460	9607	18.5%	7084	59151
shell	137	39347	8832	18.3%	5485	53664
fortranfree	690	11852	2582	17.9%	1414	15848
objective_c	395	10562	1768	14.3%	2951	15281
automake	61	6014	853	12.4%	956	7823
perl	24	4111	1138	21.7%	732	5981
scheme	1	2775	153	5.2%	328	3256
ocaml	5	2482	538	17.8%	328	3348
python	6	1135	211	15.7%	220	1566
awk	9	1127	324	22.3%	193	1644
pascal	4	1044	141	11.9%	218	1403
csharp	9	879	506	36.5%	230	1615
dcl	2	497	99	16.6%	30	626
tcl	1	392	113	22.4%	72	577
haskell	48	149	0	0.0%	16	165
emacsclisp	1	59	21	26.2%	4	84
matlab	2	57	0	0.0%	7	64
Total	50312	4938881	1567750	24.1%	1071986	7578617



Total: 66139 file(s)

Language	Files	Code	Comment	Comment %	Blank	Total
c	15638	1840245	394682	17.7%	366815	2601742
cpp	19622	872775	190744	17.9%	189007	1252526
java	6342	681656	643045	48.5%	169465	1494166
ada	4206	638557	294881	31.6%	218000	1151438
autoconf	76	445046	393	0.1%	58831	504270
make	82	110064	3268	2.9%	13270	126602
html	480	103080	5658	5.2%	21438	130176
fortranfixed	2164	73366	1570	2.1%	9454	84390
assembler	183	42460	9607	18.5%	7084	59151
shell	137	39347	8832	18.3%	5485	53664
fortranfree	690	11852	2582	17.9%	1414	15848
objective_c	395	10562	1768	14.3%	2951	15281
automake	61	6014	853	12.4%	956	7823
perl	24	4111	1138	21.7%	732	5981
scheme	1	2775	153	5.2%	328	3256
ocaml	5	2482	538	17.8%	328	3348
python	6	1135	211	15.7%	220	1566
awk	9	1127	324	22.3%	193	1644
pascal	4	1044	141	11.9%	218	1403
csharp	9	879	506	36.5%	230	1615
dcl	2	497	99	16.6%	30	626
tcl	1	392	113	22.4%	72	577
haskell	48	149	0	0.0%	16	165
emacsclisp	1	59	21	26.2%	4	84
matlab	2	57	0	0.0%	7	64
Total	50312	4938881	1567750	24.1%	1071986	7578617

ohcount: Line Count of gcc-4.5.0

Total: 69739 file(s)

Language	Files	Code	Comment	Comment %	Blank	Total
c	16985	1967826	413941	17.4%	391883	2773650
cpp	20813	912618	210084	18.7%	199605	1322307
java	6342	681810	643127	48.5%	169483	1494420
ada	4412	647372	302226	31.8%	222481	1172079
autoconf	79	358996	422	0.1%	55631	415049
html	487	144535	5667	3.8%	31773	181975
make	93	114490	3438	2.9%	14434	132362
fortranfixed	2535	85905	1817	2.1%	11394	99116
assembler	197	45098	10082	18.3%	7528	62708
shell	136	39789	8984	18.4%	5511	54284
scheme	7	13725	1192	8.0%	1524	16441
fortranfree	760	12955	2889	18.2%	1546	17390
objective_c	396	10782	1835	14.5%	2959	15576
automake	64	6388	914	12.5%	994	8296
perl	25	4144	1139	21.6%	739	6022
xslt	20	2805	436	13.5%	563	3804
ocaml	5	2515	540	17.7%	328	3383
python	10	1686	322	16.0%	383	2391
awk	10	1352	372	21.6%	218	1942
pascal	4	1044	141	11.9%	218	1403
csharp	9	879	506	36.5%	230	1615
dcl	2	402	84	17.3%	13	499
tcl	1	392	113	22.4%	72	577
haskell	49	153	0	0.0%	17	170
emacsclisp	1	59	21	26.2%	4	84
matlab	1	5	0	0.0%	0	5



Total: 69739 file(s)

Language	Files	Code	Comment	Comment %	Blank	Total
c	16985	1967826	413941	17.4%	391883	2773650
cpp	20813	912618	210084	18.7%	199605	1322307
java	6342	681810	643127	48.5%	169483	1494420
ada	4412	647372	302226	31.8%	222481	1172079
autoconf	79	358996	422	0.1%	55631	415049
html	487	144535	5667	3.8%	31773	181975
make	93	114490	3438	2.9%	14434	132362
fortranfixed	2535	85905	1817	2.1%	11394	99116
assembler	197	45098	10082	18.3%	7528	62708
shell	136	39789	8984	18.4%	5511	54284
scheme	7	13725	1192	8.0%	1524	16441
fortranfree	760	12955	2889	18.2%	1546	17390
objective_c	396	10782	1835	14.5%	2959	15576
automake	64	6388	914	12.5%	994	8296
perl	25	4144	1139	21.6%	739	6022
xslt	20	2805	436	13.5%	563	3804
ocaml	5	2515	540	17.7%	328	3383
python	10	1686	322	16.0%	383	2391
awk	10	1352	372	21.6%	218	1942
pascal	4	1044	141	11.9%	218	1403
csharp	9	879	506	36.5%	230	1615
dcl	2	402	84	17.3%	13	499
tcl	1	392	113	22.4%	72	577
haskell	49	153	0	0.0%	17	170
emacsclisp	1	59	21	26.2%	4	84
matlab	1	5	0	0.0%	0	5

ohcount: Line Count of gcc-4.6.0

Language	Files	Code	Comment	Comment %	Blank	Total
c	18463	2100237	444333	17.5%	418292	2962862
cpp	22002	985076	229541	18.9%	214781	1429398
java	6342	681938	645505	48.6%	169046	1496489
ada	4605	680043	315956	31.7%	234467	1230466
autoconf	91	405461	509	0.1%	62914	468884
html	457	168355	5669	3.3%	38146	212170
make	98	121545	3659	2.9%	15618	140822
fortranfixed	2936	99413	1927	1.9%	13659	114999
shell	148	48032	10451	17.9%	6586	65069
assembler	208	46727	10227	18.0%	7853	64807
xml	75	36036	282	0.8%	3827	40145
objective_c	866	28014	5000	15.1%	8115	41129
fortranfree	821	13857	3147	18.5%	1695	18699
tex	2	11060	5776	34.3%	1433	18269
scheme	6	11023	1010	8.4%	1205	13238
automake	67	9440	1038	9.9%	1456	11934
perl	28	4445	1316	22.8%	837	6598
ocaml	6	2814	576	17.0%	378	3768
xslt	20	2805	436	13.5%	563	3804
awk	11	1740	396	18.5%	257	2393
python	10	1725	322	15.7%	383	2430
css	24	1589	143	8.3%	332	2064
pascal	4	1044	141	11.9%	218	1403
csharp	9	879	506	36.5%	230	1615
dcl	2	402	84	17.3%	13	499
tcl	1	392	113	22.4%	72	577
javascript	4	341	87	20.3%	35	463
haskell	49	153	0	0.0%	17	170
bat	3	7	0	0.0%	0	7
matlab	1	5	0	0.0%	0	5
Total	57359	5464598	1688150	23.6%	1202428	8355176



Language	Files	Code	Comment	Comment %	Blank	Total
c	18463	2100237	444333	17.5%	418292	2962862
cpp	22002	985076	229541	18.9%	214781	1429398
java	6342	681938	645505	48.6%	169046	1496489
ada	4605	680043	315956	31.7%	234467	1230466
autoconf	91	405461	509	0.1%	62914	468884
html	457	168355	5669	3.3%	38146	212170
make	98	121545	3659	2.9%	15618	140822
fortranfixed	2936	99413	1927	1.9%	13659	114999
shell	148	48032	10451	17.9%	6586	65069
assembler	208	46727	10227	18.0%	7853	64807
xml	75	36036	282	0.8%	3827	40145
objective_c	866	28014	5000	15.1%	8115	41129
fortranfree	821	13857	3147	18.5%	1695	18699
tex	2	11060	5776	34.3%	1433	18269
scheme	6	11023	1010	8.4%	1205	13238
automake	67	9440	1038	9.9%	1456	11934
perl	28	4445	1316	22.8%	837	6598
ocaml	6	2814	576	17.0%	378	3768
xslt	20	2805	436	13.5%	563	3804
awk	11	1740	396	18.5%	257	2393
python	10	1725	322	15.7%	383	2430
css	24	1589	143	8.3%	332	2064
pascal	4	1044	141	11.9%	218	1403
cssharp	9	879	506	36.5%	230	1615
dcl	2	402	84	17.3%	13	499
tcl	1	392	113	22.4%	72	577
javascript	4	341	87	20.3%	35	463
haskell	49	153	0	0.0%	17	170
bat	3	7	0	0.0%	0	7
matlab	1	5	0	0.0%	0	5
Total	57359	5464598	1688150	23.6%	1202428	8355176

ohcount: Line Count of gcc-4.4.2/gcc

Total: 30421 file(s)

Language	Files	Code	Comment	Comment %	Blank	Total
c	13296	1254253	282582	18.4%	283766	1820601
ada	4196	636876	294321	31.6%	217401	1148598
cpp	7418	184186	52163	22.1%	54048	290397
fortranfixed	2086	67988	1521	2.2%	9079	78588
assembler	132	31092	7243	18.9%	4770	43105
autoconf	3	26996	10	0.0%	3383	30389
fortranfree	652	10898	2376	17.9%	1314	14588
objective_c	391	10155	1654	14.0%	2830	14639
make	3	5340	1027	16.1%	814	7181
scheme	1	2775	153	5.2%	328	3256
ocaml	5	2482	538	17.8%	328	3348
shell	16	2256	712	24.0%	374	3342
awk	7	1022	251	19.7%	187	1460
perl	1	772	205	21.0%	137	1114
haskell	48	149	0	0.0%	16	165
matlab	2	57	0	0.0%	7	64
Total	28258	2242738	647591	22.4%	579484	346981 ²



Total: 30421 file(s)

Language	Files	Code	Comment	Comment %	Blank	Total
c	13296	1254253	282582	18.4%	283766	1820601
ada	4196	636876	294321	31.6%	217401	1148598
cpp	7418	184186	52163	22.1%	54048	290397
fortranfixed	2086	67988	1521	2.2%	9079	78588
assembler	132	31092	7243	18.9%	4770	43105
autoconf	3	26996	10	0.0%	3383	30389
fortranfree	652	10898	2376	17.9%	1314	14588
objective_c	391	10155	1654	14.0%	2830	14639
make	3	5340	1027	16.1%	814	7181
scheme	1	2775	153	5.2%	328	3256
ocaml	5	2482	538	17.8%	328	3348
shell	16	2256	712	24.0%	374	3342
awk	7	1022	251	19.7%	187	1460
perl	1	772	205	21.0%	137	1114
haskell	48	149	0	0.0%	16	165
matlab	2	57	0	0.0%	7	64
Total	28258	2242738	647591	22.4%	579484	3469813



ohcount: Line Count of gcc-4.5.0/gcc

Total: 33007 file(s)

Language	Files	Code	Comment	Comment %	Blank	Total
c	14565	1368937	300284	18.0%	305671	1974892
ada	4402	645691	301666	31.8%	221882	1169239
cpp	7984	197798	54719	21.7%	57312	309829
fortranfixed	2453	80403	1768	2.2%	11008	93179
assembler	136	31802	7431	18.9%	4864	44097
autoconf	3	27317	10	0.0%	3876	31203
scheme	7	13725	1192	8.0%	1524	16441
fortranfree	722	12001	2683	18.3%	1446	16130
objective_c	392	10375	1721	14.2%	2838	14934
make	3	5886	1039	15.0%	854	7779
ocaml	5	2515	540	17.7%	328	3383
shell	14	2101	642	23.4%	347	3090
awk	8	1247	299	19.3%	212	1758
perl	2	805	206	20.4%	144	1155
haskell	49	153	0	0.0%	17	170
matlab	1	5	0	0.0%	0	5
Total	30747	2406202	677035	22.0%	613025	369626 ²



Total: 33007 file(s)

Language	Files	Code	Comment	Comment %	Blank	Total
c	14565	1368937	300284	18.0%	305671	1974892
ada	4402	645691	301666	31.8%	221882	1169239
cpp	7984	197798	54719	21.7%	57312	309829
fortranfixed	2453	80403	1768	2.2%	11008	93179
assembler	136	31802	7431	18.9%	4864	44097
autoconf	3	27317	10	0.0%	3876	31203
scheme	7	13725	1192	8.0%	1524	16441
fortranfree	722	12001	2683	18.3%	1446	16130
objective_c	392	10375	1721	14.2%	2838	14934
make	3	5886	1039	15.0%	854	7779
ocaml	5	2515	540	17.7%	328	3383
shell	14	2101	642	23.4%	347	3090
awk	8	1247	299	19.3%	212	1758
perl	2	805	206	20.4%	144	1155
haskell	49	153	0	0.0%	17	170
matlab	1	5	0	0.0%	0	5
Total	30747	2406202	677035	22.0%	613025	3696262



ohcount: Line Count of gcc-4.6.0/gcc

Total: 36839 file(s)

Language	Files	Code	Comment	Comment %	Blank	Total
c	15787	1462494	321820	18.0%	324179	2108493
ada	4595	678362	315396	31.7%	233868	1227626
cpp	8666	252213	61026	19.5%	67144	380383
fortranfixed	2850	93549	1878	2.0%	13260	108687
assembler	137	31548	7446	19.1%	4857	43851
autoconf	3	28775	12	0.0%	4020	32807
objective_c	861	27465	4822	14.9%	7967	40254
fortranfree	783	12903	2936	18.5%	1595	17434
scheme	6	11023	1010	8.4%	1205	13238
make	4	6078	1070	15.0%	893	8041
tex	1	5441	2835	34.3%	702	8978
ocaml	6	2814	576	17.0%	378	3768
shell	16	1980	597	23.2%	338	2915
awk	9	1635	323	16.5%	251	2209
perl	3	866	225	20.6%	158	1249
haskell	49	153	0	0.0%	17	170
matlab	1	5	0	0.0%	0	5
Total	33777	2617304	721972	21.6%	660832	4001000



Total: 36839 file(s)

Language	Files	Code	Comment	Comment %	Blank	Total
c	15787	1462494	321820	18.0%	324179	2108493
ada	4595	678362	315396	31.7%	233868	1227626
cpp	8666	252213	61026	19.5%	67144	380383
fortranfixed	2850	93549	1878	2.0%	13260	108687
assembler	137	31548	7446	19.1%	4857	43851
autoconf	3	28775	12	0.0%	4020	32807
objective_c	861	27465	4822	14.9%	7967	40254
fortranfree	783	12903	2936	18.5%	1595	17434
scheme	6	11023	1010	8.4%	1205	13238
make	4	6078	1070	15.0%	893	8041
tex	1	5441	2835	34.3%	702	8978
ocaml	6	2814	576	17.0%	378	3768
shell	16	1980	597	23.2%	338	2915
awk	9	1635	323	16.5%	251	2209
perl	3	866	225	20.6%	158	1249
haskell	49	153	0	0.0%	17	170
matlab	1	5	0	0.0%	0	5
Total	33777	2617304	721972	21.6%	660832	4000108



Why is Understanding GCC Difficult?

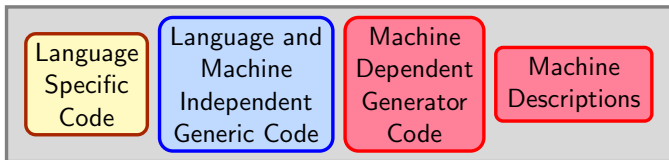
Deeper technical reasons

- GCC is not a compiler but a *compiler generation framework*
Two distinct gaps that need to be bridged
 - ▶ Input-output of the generation framework
The target specification and the generated compiler
 - ▶ Input-output of the generated compiler
A source program and the generated assembly program
- GCC generated compiler uses a derivative of the Davidson-Fraser model of compilation
 - ▶ Early instruction selection
 - ▶ Machine dependent intermediate representation
 - ▶ Simplistic instruction selection and retargetability mechanism



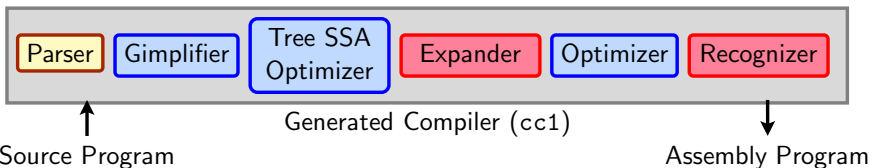
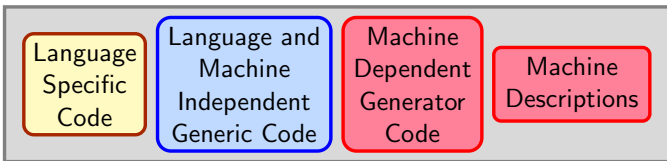
The Architecture of GCC

Compiler Generation Framework

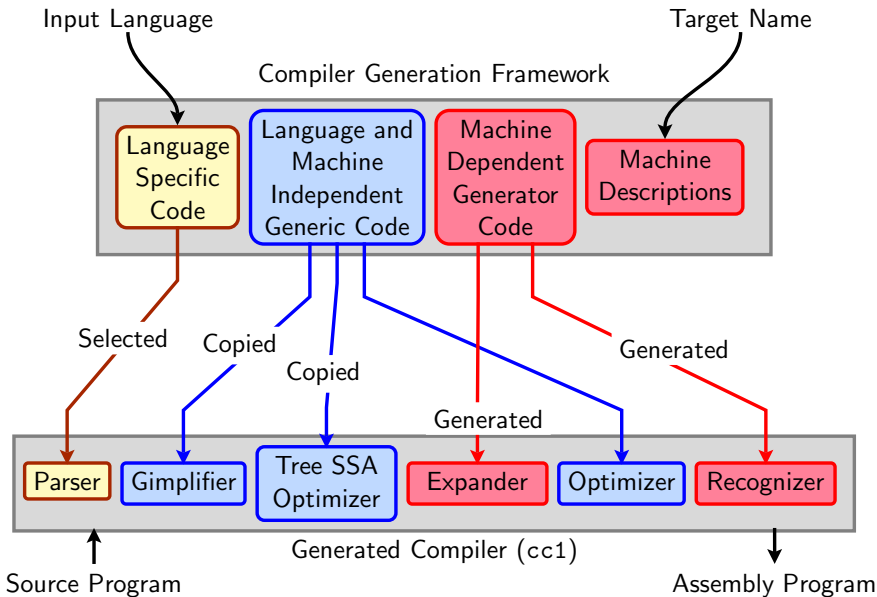


The Architecture of GCC

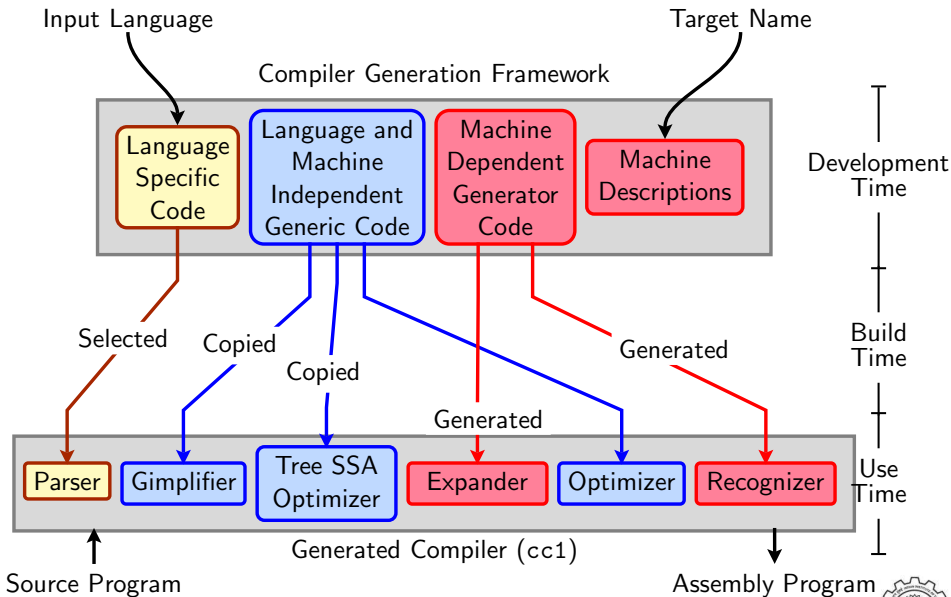
Compiler Generation Framework



The Architecture of GCC



The Architecture of GCC



An Example of The Generation Related Gap

- Predicate function for invoking the loop distribution pass

```
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

```
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

```
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

```
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-4.6.0/gcc` using `cscope -R`



Another Example of The Generation Related Gap

- Locating the `main` function in the directory `gcc-4.6.0/gcc` using `cscope -R`
- 7027 occurrences!



Another Example of The Generation Related Gap

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

- What if we do not search recursively?



Another Example of The Generation Related Gap

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



Another Example of The Generation Related Gap

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

File	Line	
0 collect2.c	1076	main (int argc, char **argv)
1 fp-test.c	85	main (void)
2 gcc.c	6092	main (int argc, char **argv)
3 gcov-dump.c	76	main (int argc ATTRIBUTE_UNUSED, char **argv)
4 gcov-iov.c	29	main (int argc, char **argv)
5 gcov.c	360	main (int argc, char **argv)
6 genattr.c	164	main (int argc, char **argv)
7 genattrtab.c	4820	main (int argc, char **argv)
8 genautomata.c	9459	main (int argc, char **argv)
9 genchecksum.c	97	main (int argc, char ** argv)
a gencodes.c	51	main (int argc, char **argv)
b genconditions.c	209	main (int argc, char **argv)
c genconfig.c	261	main (int argc, char **argv)
d genconstants.c	79	main (int argc, char **argv)
e genemit.c	830	main (int argc, char **argv)
f genenums.c	48	main (int argc, char **argv)



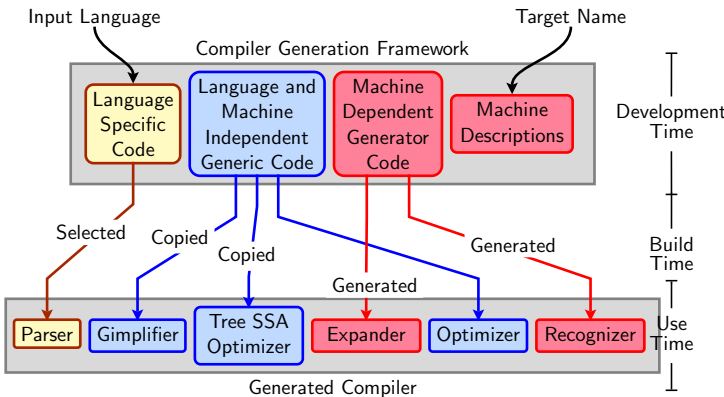
Another Example of The Generation Related Gap

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

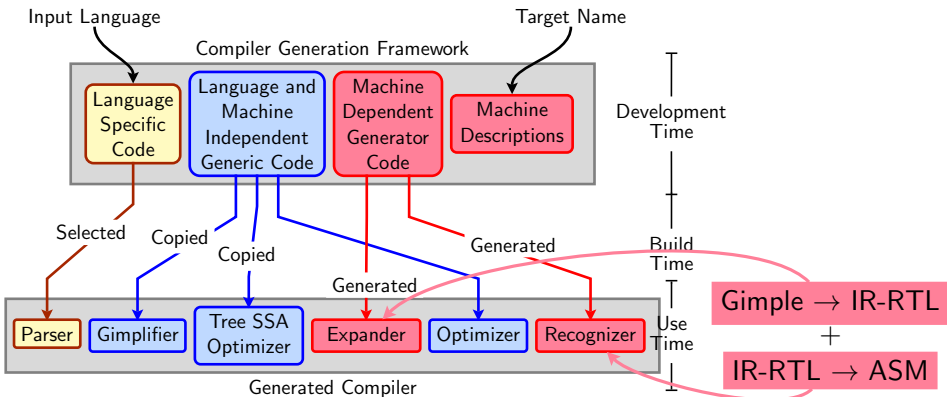
```
g genextract.c      402 main (int argc, char **argv)
h genflags.c       251 main (int argc, char **argv)
i gengenrtl.c      282 main (void )
j gengtype.c      4825 main (int argc, char **argv)
k genhooks.c       335 main (int argc, char **argv)
l genmddeps.c      43  main (int argc, char **argv)
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-wrapper.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;
```



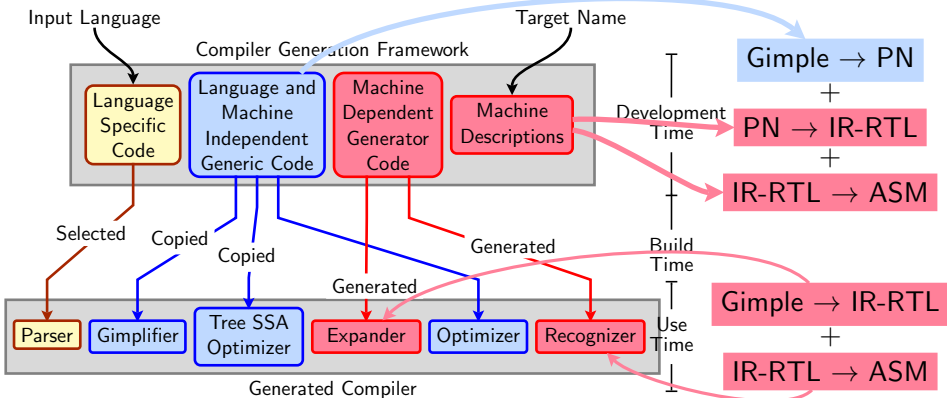
GCC Retargetability Mechanism



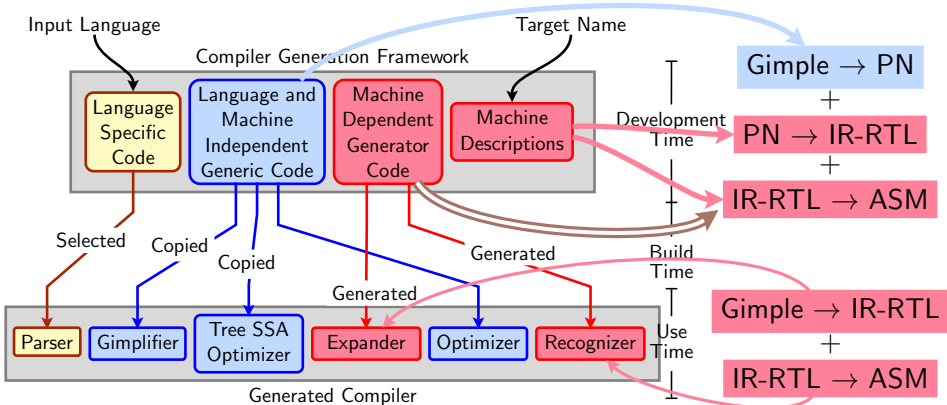
GCC Retargetability Mechanism



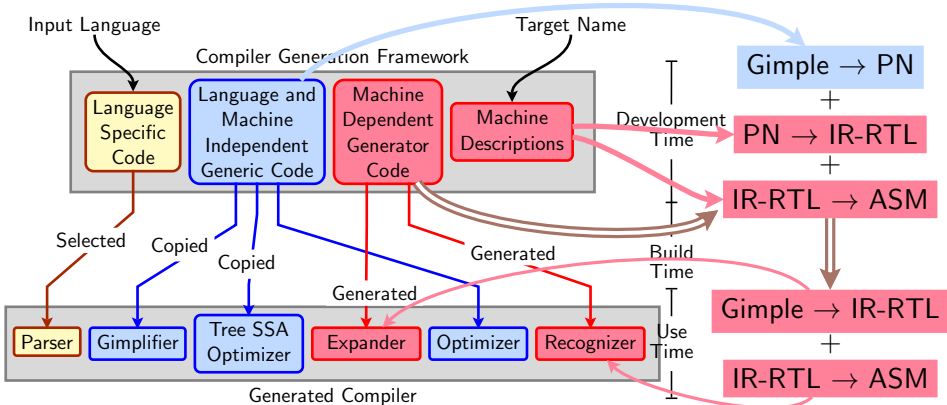
GCC Retargetability Mechanism



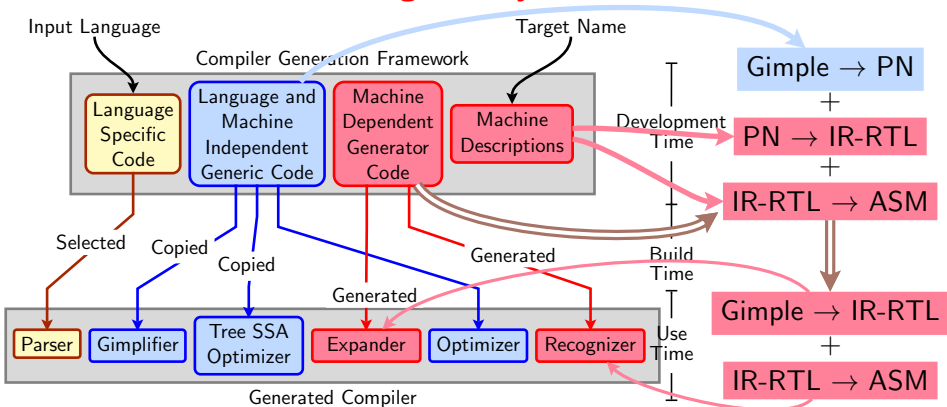
GCC Retargetability Mechanism



GCC Retargetability Mechanism



GCC Retargetability Mechanism



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

Files	i386	mips	arm
*.md	38722	15534	30938
*.c	39579	16766	26164
*.h	17869	5667	18711
Total	96170	37969	75913



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

Files	i386	mips	arm
*.md	38722	15534	30938
*.c	39579	16766	26164
*.h	17869	5667	18711
Total	96170	37969	75913

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



Part 5

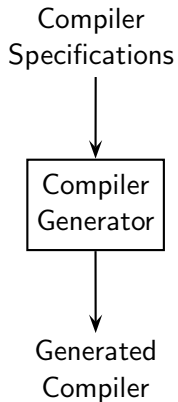
Meeting the GCC Challenge

Meeting the GCC Challenge

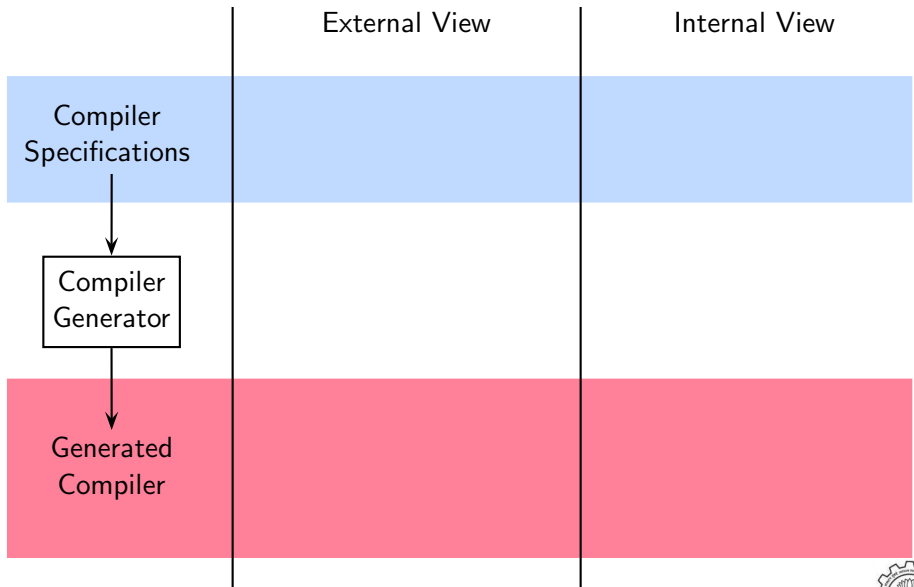
Goal of Understanding	Methodology	Needs Examining		
		Makefiles	Source	MD
Translation sequence of programs	Gray box probing	No	No	No
Build process	Customizing the configuration and building	Yes	No	No
Retargetability issues and machine descriptions	Incremental construction of machine descriptions	No	No	Yes
IR data structures and access mechanisms	Adding passes to massage IRs	No	Yes	Yes
Retargetability mechanism		Yes	Yes	Yes



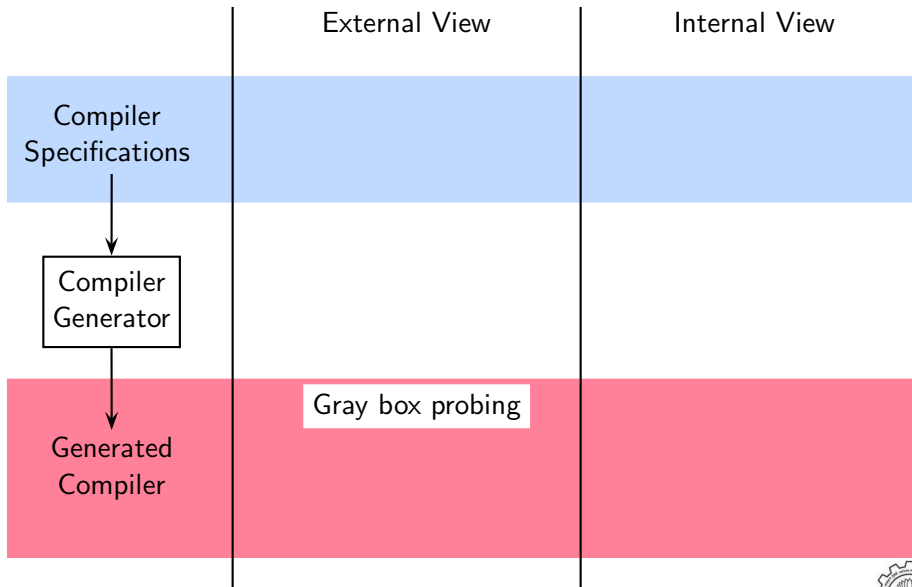
CS715 Coverage



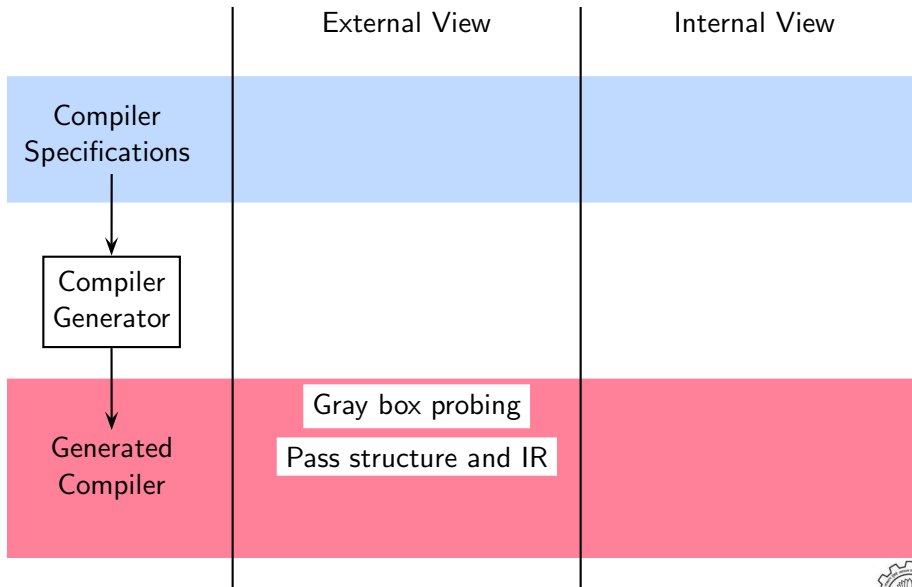
CS715 Coverage



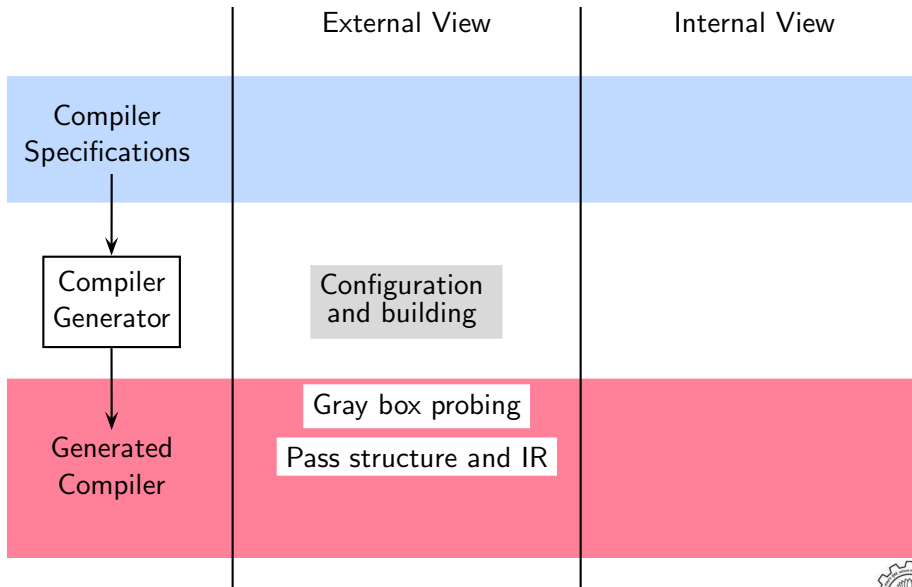
CS715 Coverage



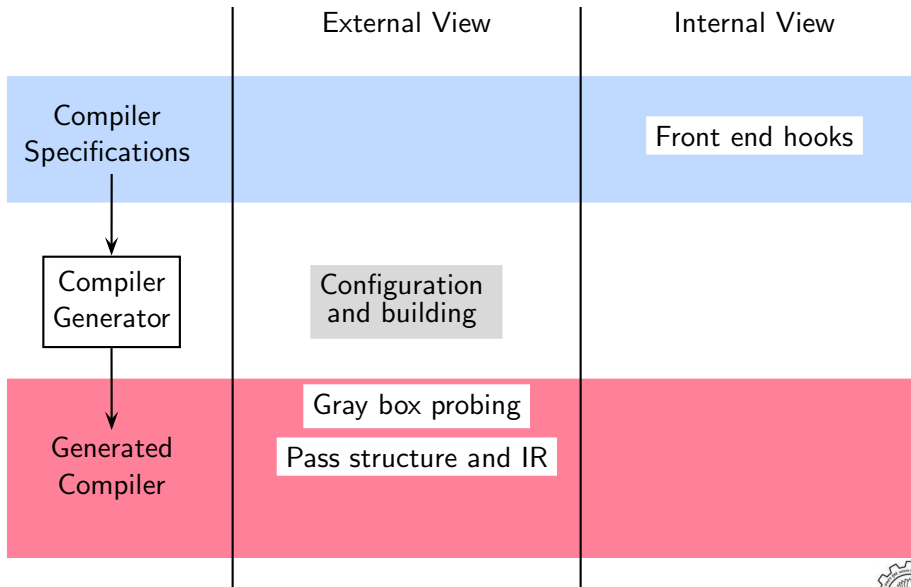
CS715 Coverage



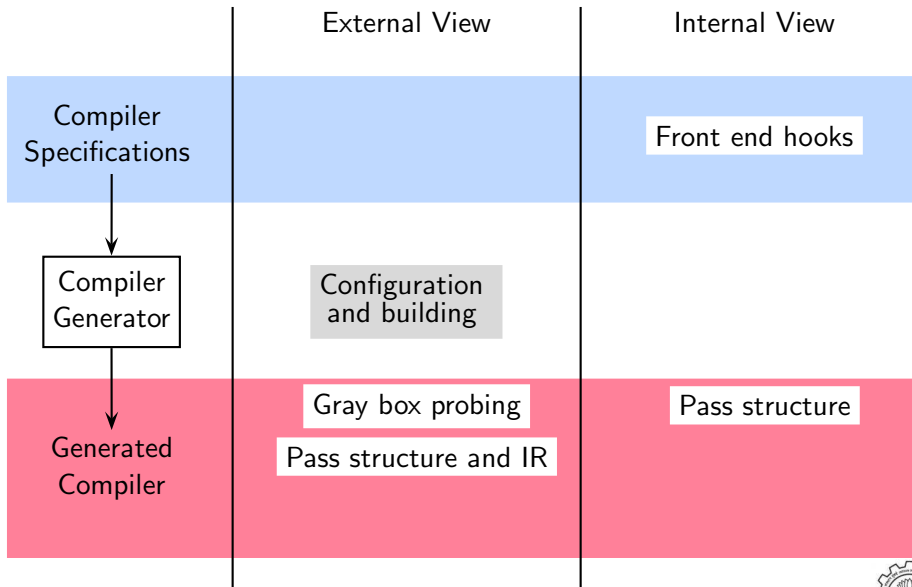
CS715 Coverage



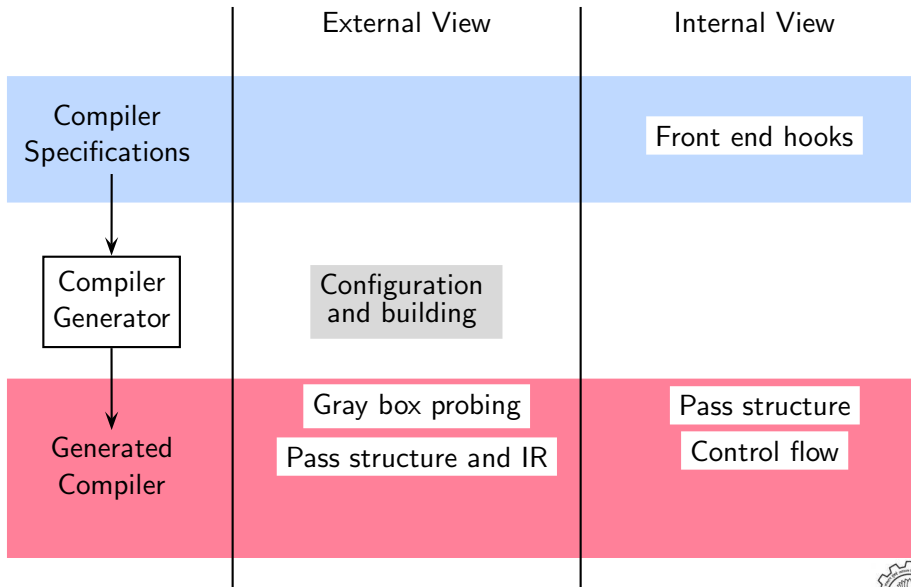
CS715 Coverage



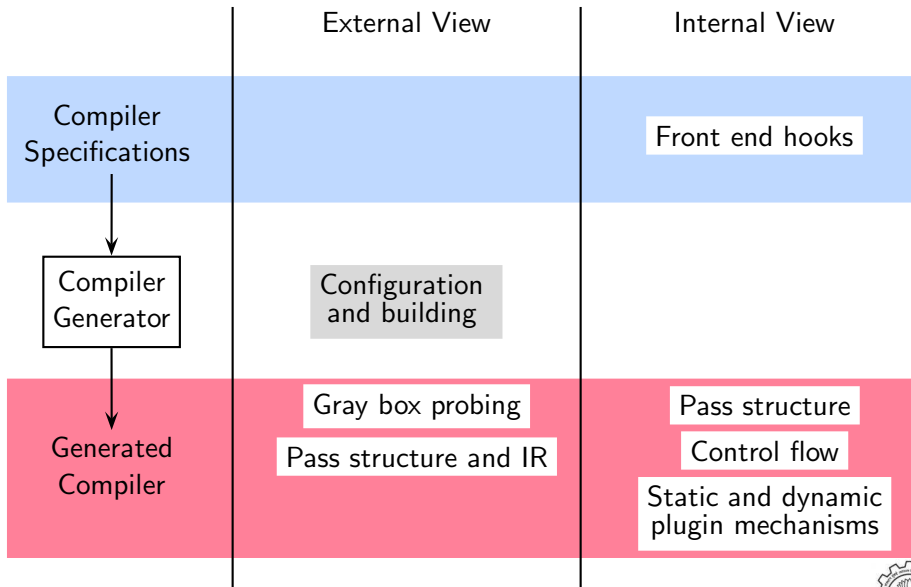
CS715 Coverage



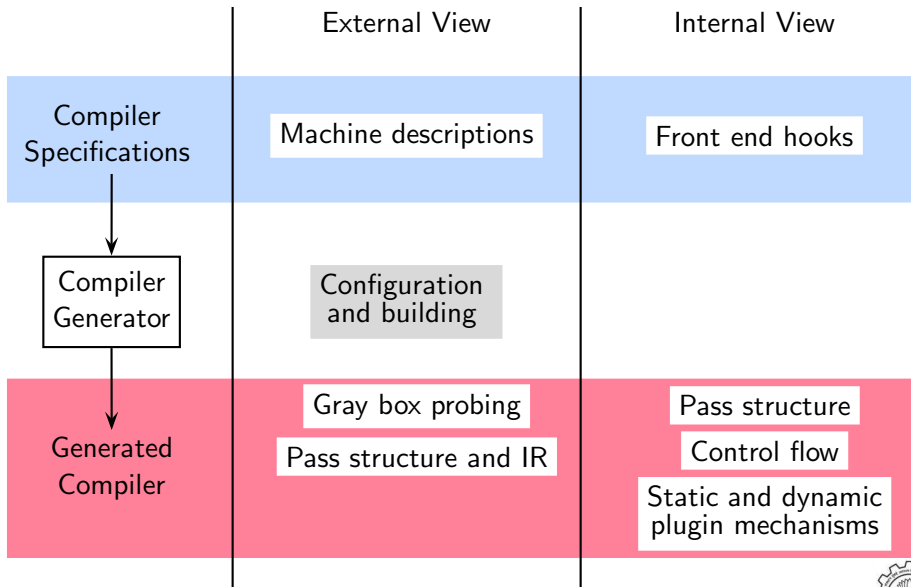
CS715 Coverage



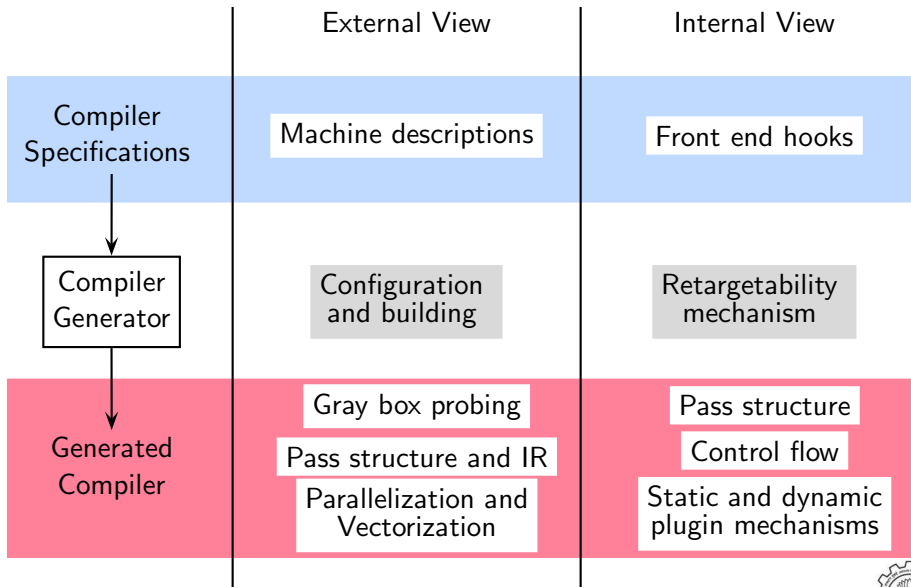
CS715 Coverage



CS715 Coverage



CS715 Coverage



CS 715 Philosophy

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



Our Canvas

- Version: GCC 4.6.2
- Language: C
- Targets: i386, spim (mips simulator)



CS715 Philosophy and Pedogogy

- We will

- You will



CS715 Philosophy and Pedogogy

- We will
 - ▶ Explain configuration and building of GCC
 - ▶ Explain essential abstractions related to compilation
The key intermediate representations and their manipulations
 - ▶ Explain essential abstractions related to program analysis in GCC
 - ▶ Explain essential abstractions related to generation of a compiler
The machine descriptions and their influence on compilation
- You will



CS715 Philosophy and Pedogogy

- We will
 - ▶ Explain configuration and building of GCC
 - ▶ Explain essential abstractions related to compilation
The key intermediate representations and their manipulations
 - ▶ Explain essential abstractions related to program analysis in GCC
 - ▶ Explain essential abstractions related to generation of a compiler
The machine descriptions and their influence on compilation
- You will
 - ▶ Build and run GCC
 - ▶ Examine various IR dumps produced by GCC
 - ▶ Add passes to GCC
 - ▶ Add a new machine description and systematically enhance it



Takeaways from this Course

- A programmer will get a better compiler
- A compiler professional will be able to deploy and enhance GCC much more easily.
- A compiler researcher will be able to use GCC for research much better.
- A compiler teacher will be able to strike a better balance between theory and practice.
- A compiler student will be exposed to issues in real compilers.



More on 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 GDFAs, MD rewriting with SpecRTL, 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

Head	Number	Weightage
Assignments	4	$4 \times 15 = 60$
Project	1	40
Total		100

- 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

