Essential Abstractions in GCC

Uday Khedker

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

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

13 June 2014

Outline

- Compilation Models
- GCC: The Great Compiler Challenge
- Meeting the GCC Challenge: CS 715
  The course plan

Part 1

Compilation Models

Compilation Models

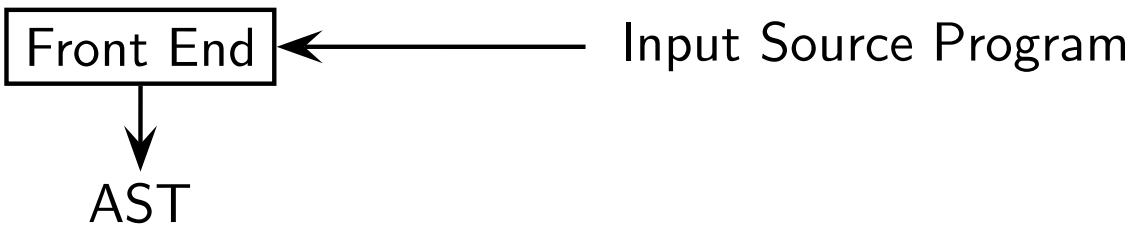
Aho Ullman Model

Davidson Fraser Model

Compilation Models

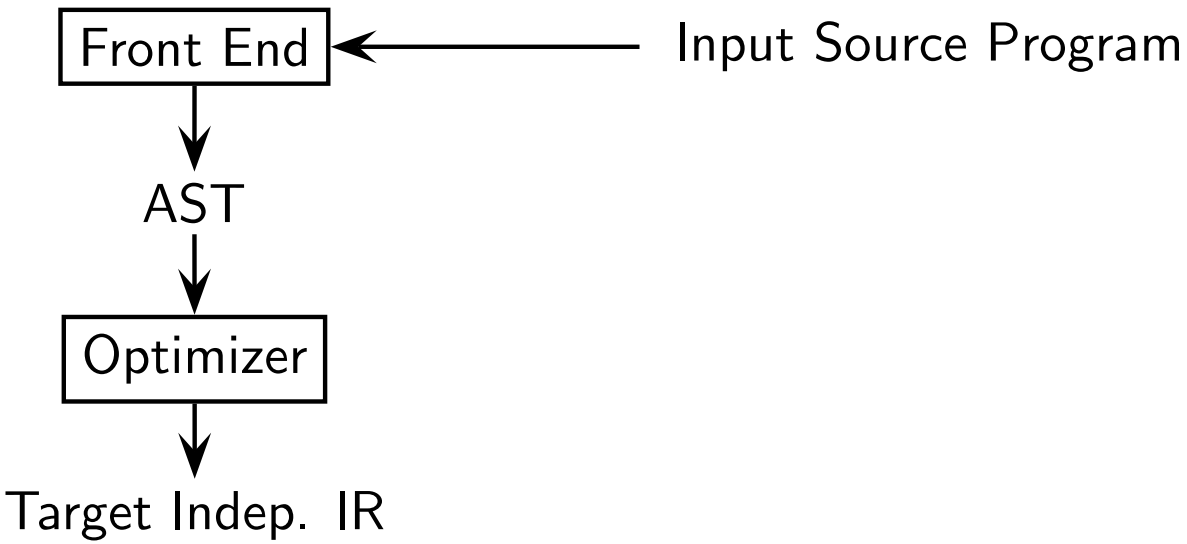
Aho Ullman Model

Davidson Fraser Model



Compilation Models

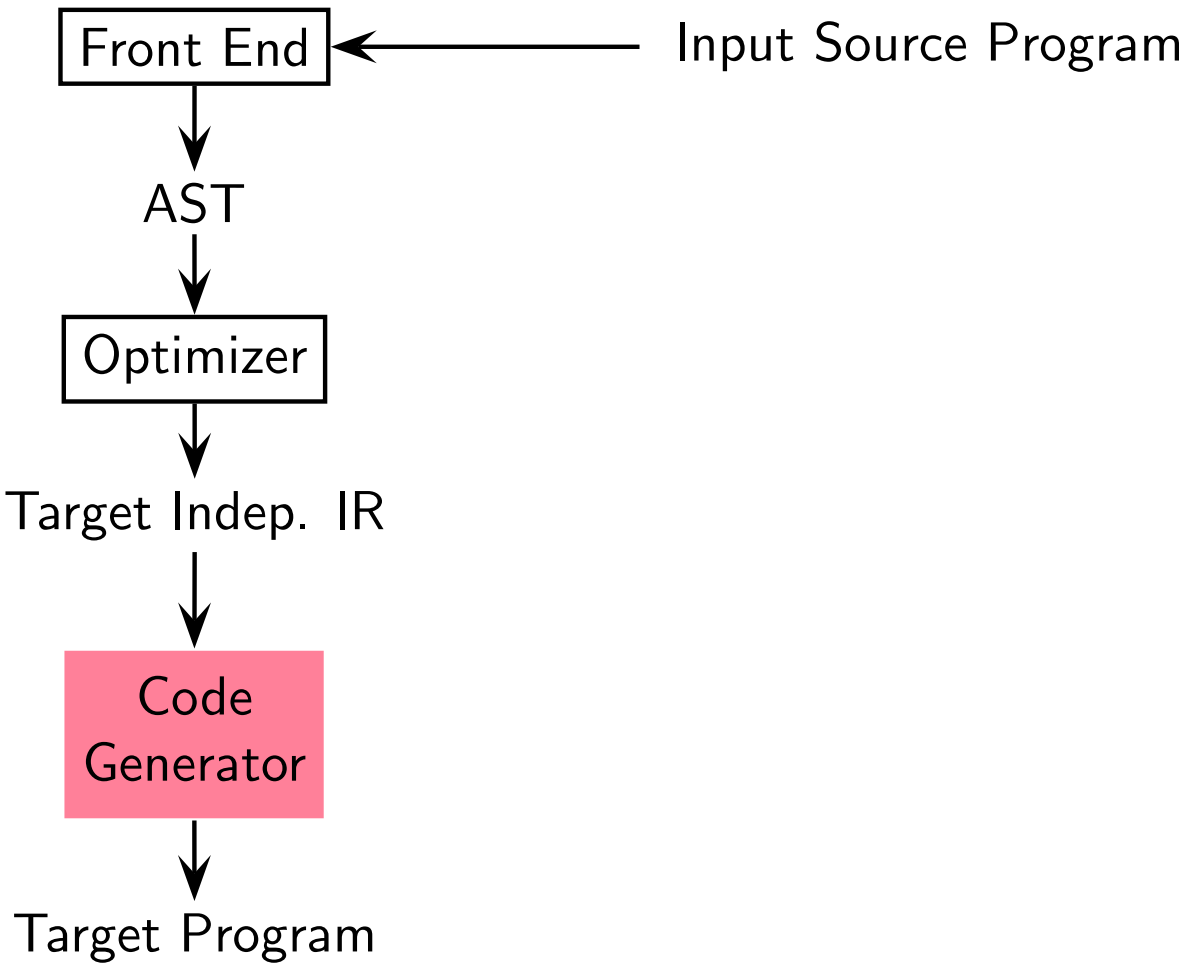
Aho Ullman Model



Davidson Fraser Model

Compilation Models

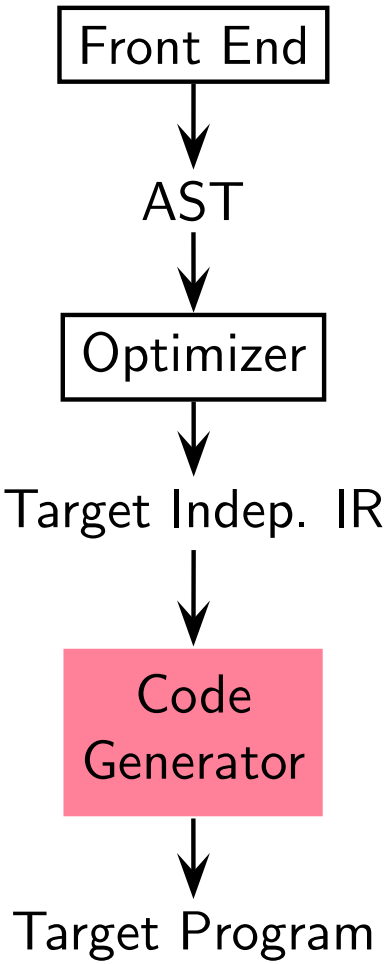
Aho Ullman Model



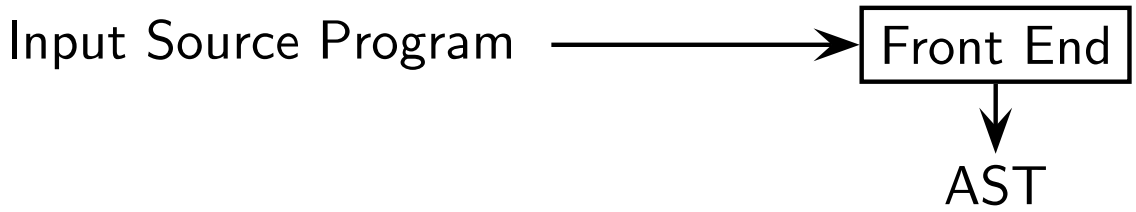
Davidson Fraser Model

Compilation Models

Aho Ullman Model

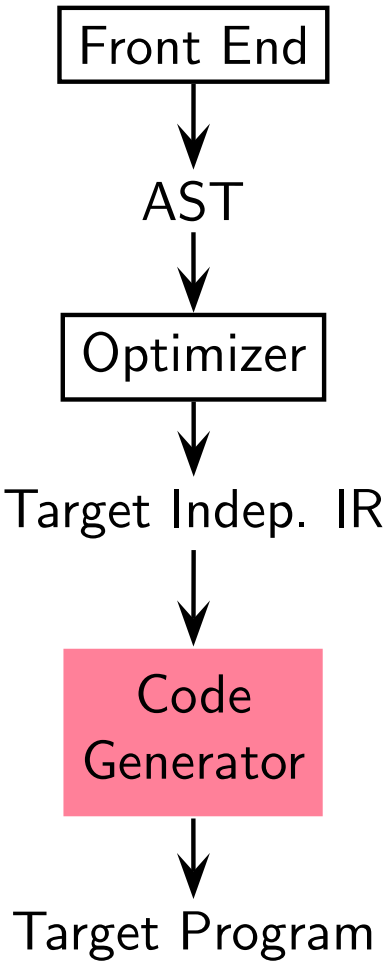


Davidson Fraser Model

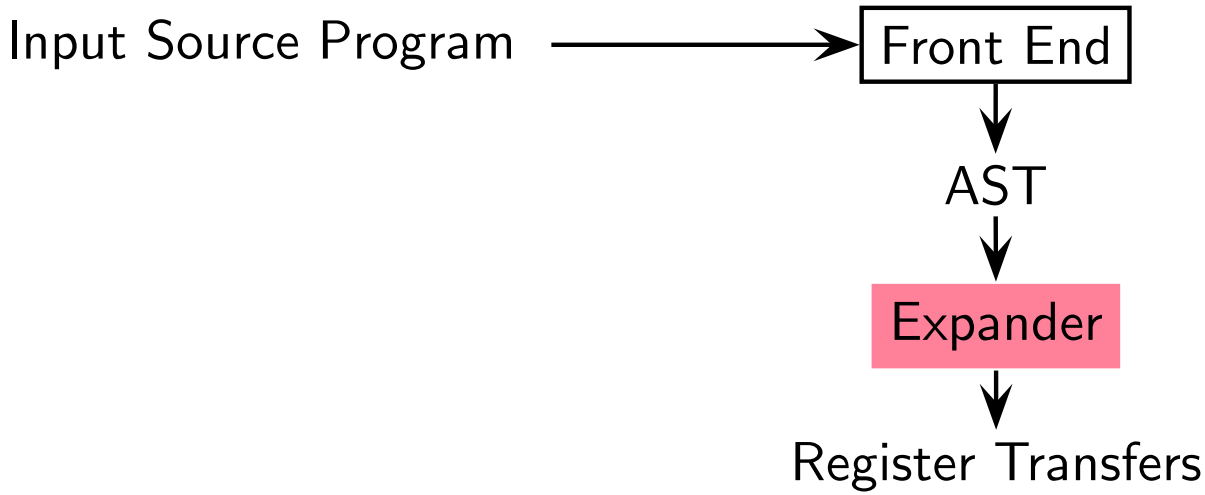


Compilation Models

Aho Ullman Model

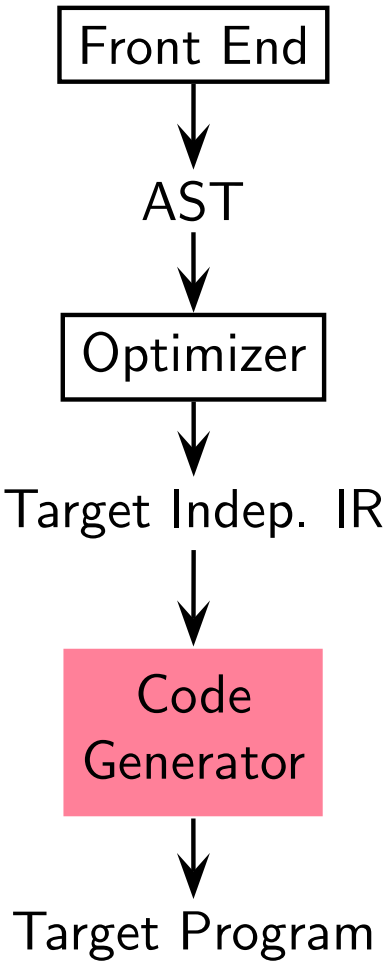


Davidson Fraser Model

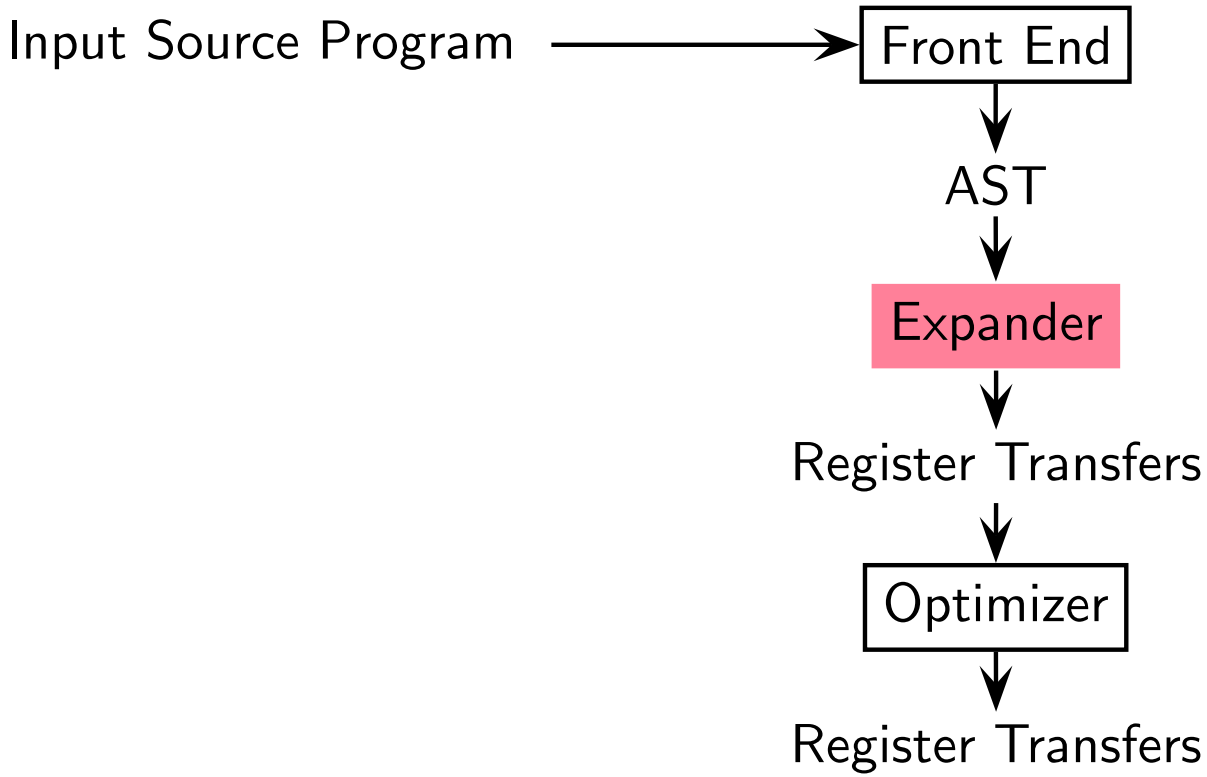


Compilation Models

Aho Ullman Model

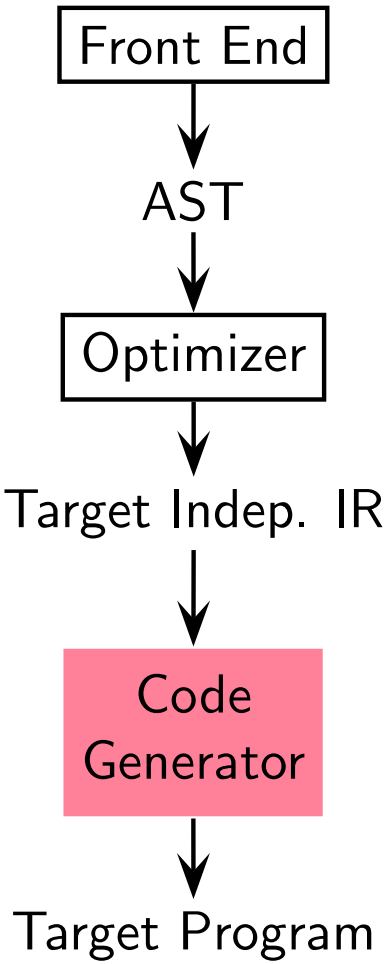


Davidson Fraser Model



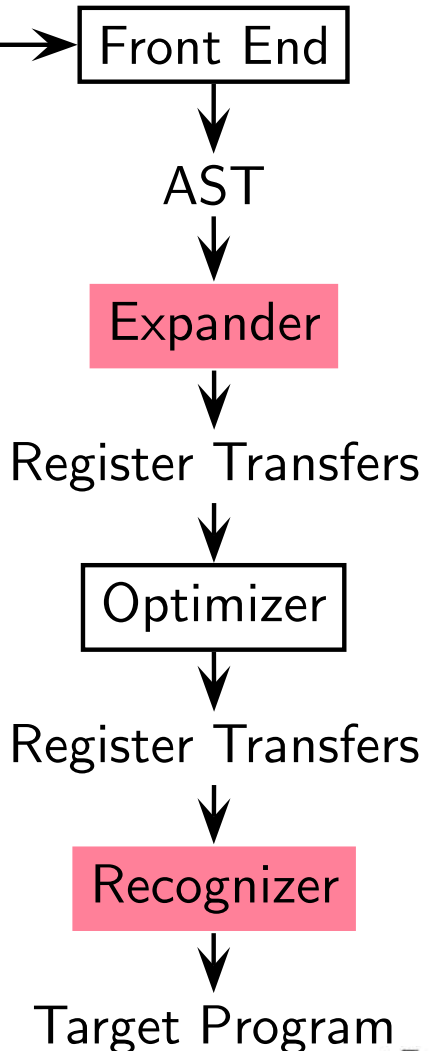
Compilation Models

Aho Ullman Model



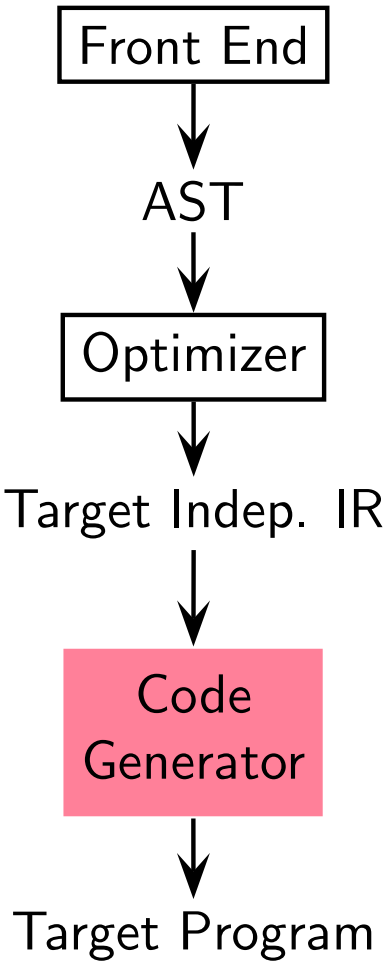
Input Source Program

Davidson Fraser Model



Compilation Models

Aho Ullman Model



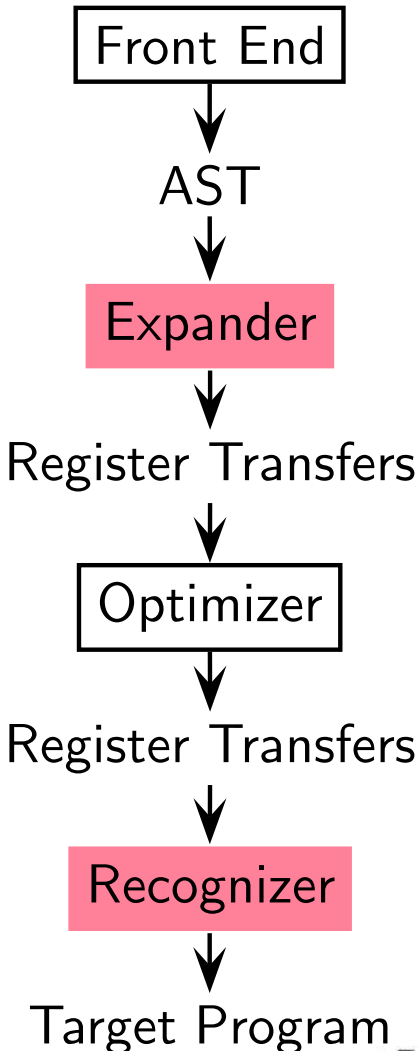
Aho Ullman: Instruction selection

- over optimized IR using
- cost based tree tiling matching

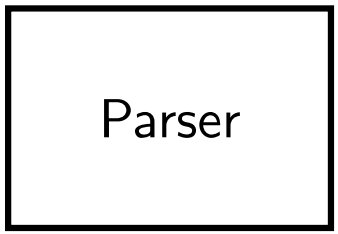
Davidson Fraser: Instruction selection

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

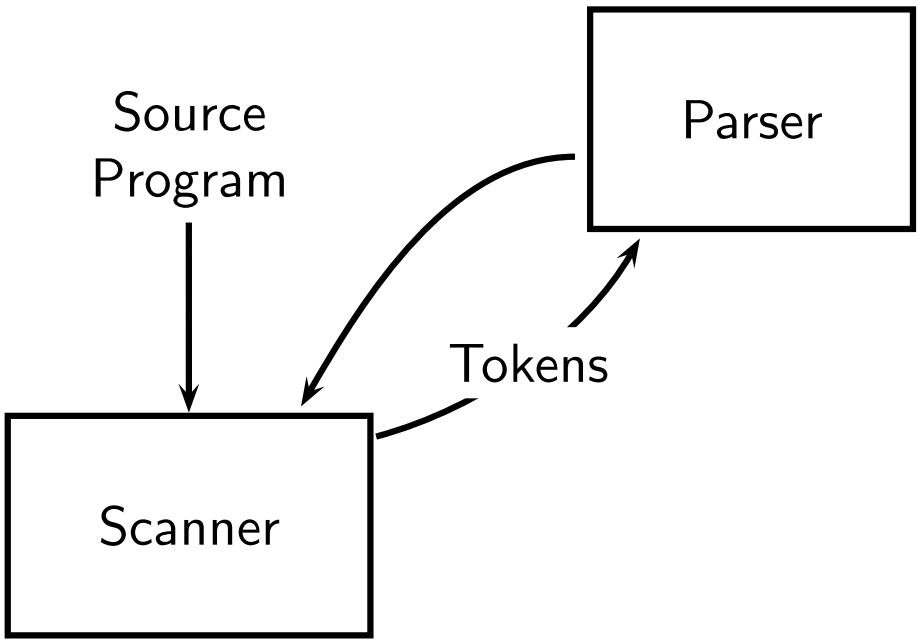
Davidson Fraser Model



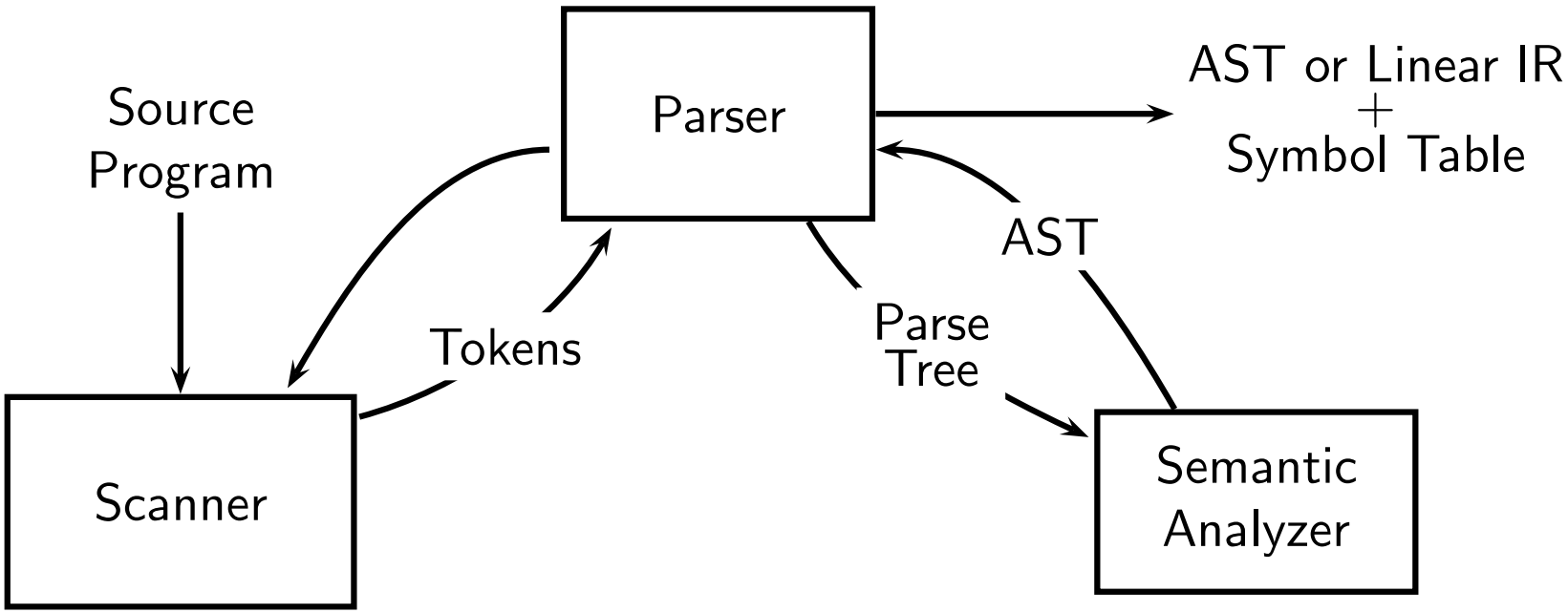
Typical Front Ends



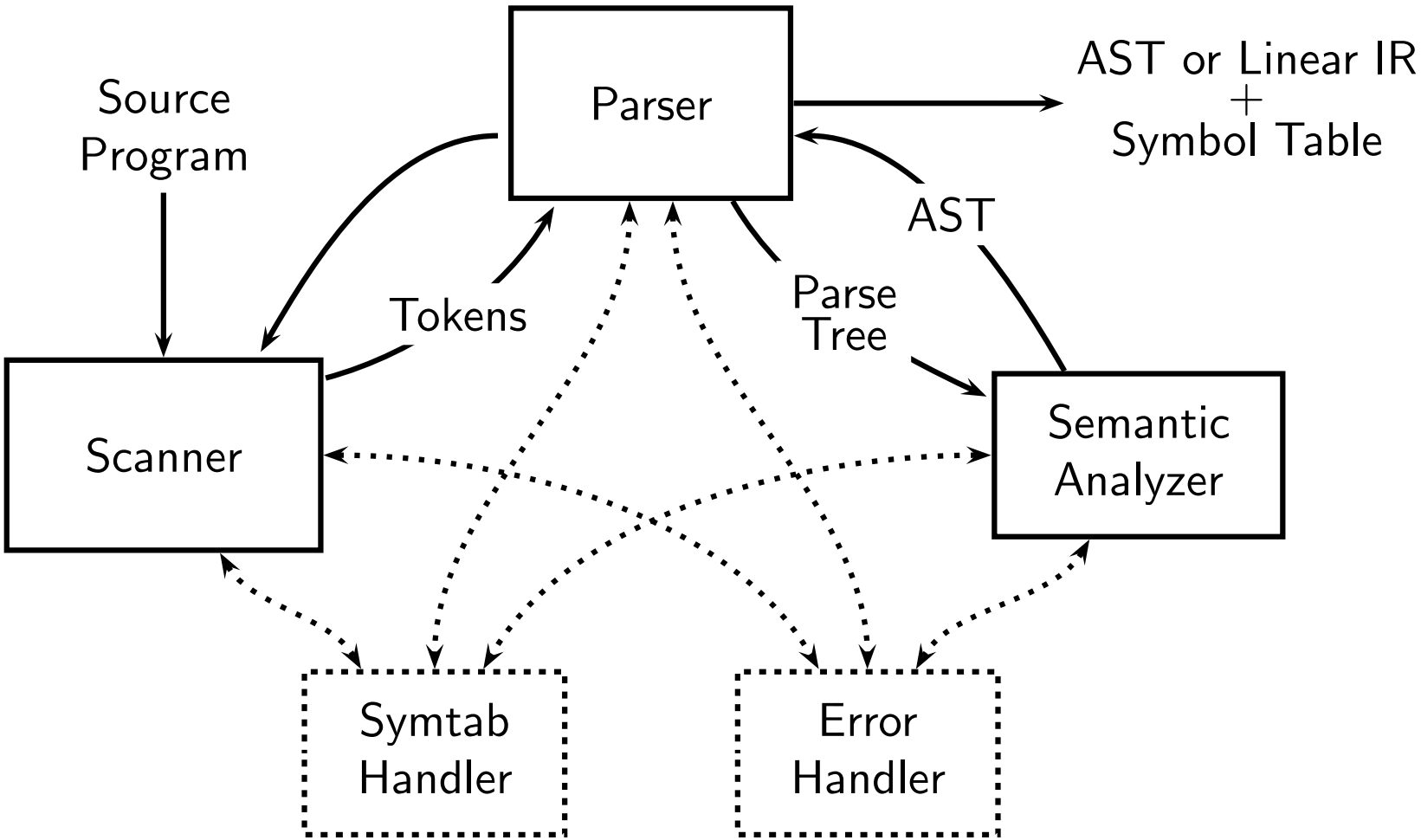
Typical Front Ends



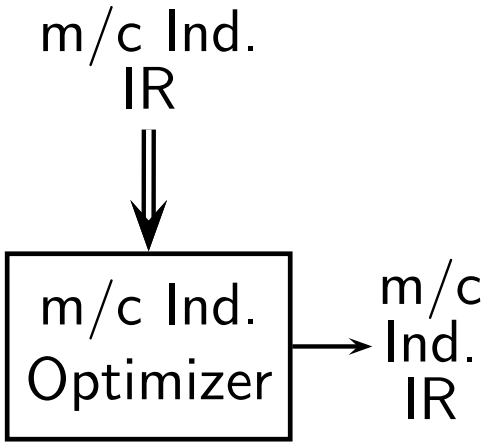
Typical Front Ends



Typical Front Ends

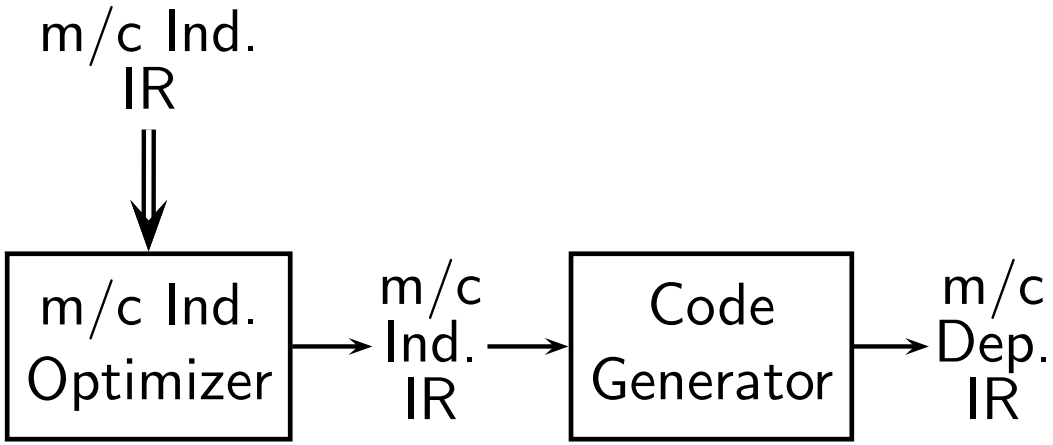


Typical Back Ends in Aho Ullman Model



- Compile time evaluations
- Eliminating redundant computations

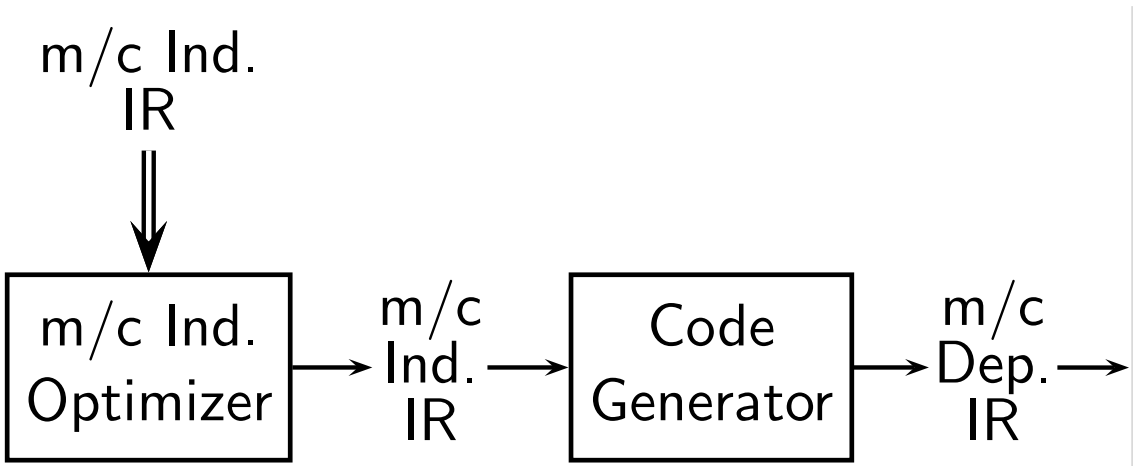
Typical Back Ends in Aho Ullman Model



- Compile time evaluations
- Eliminating redundant computations

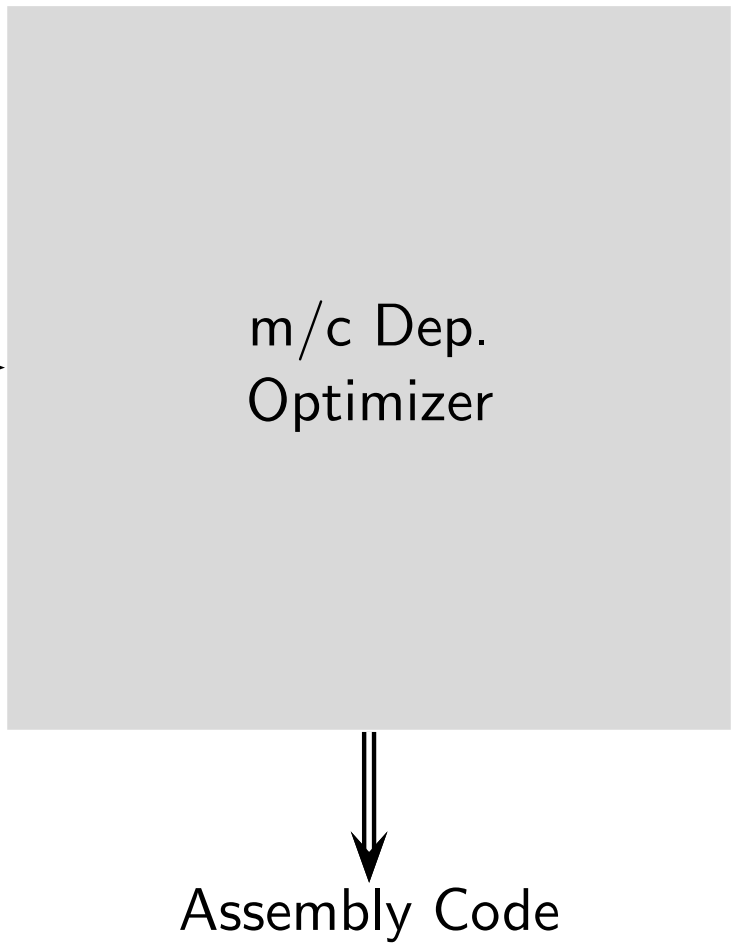
- Instruction Selection
- Local Reg Allocation
- Choice of Order of Evaluation

Typical Back Ends in Aho Ullman Model

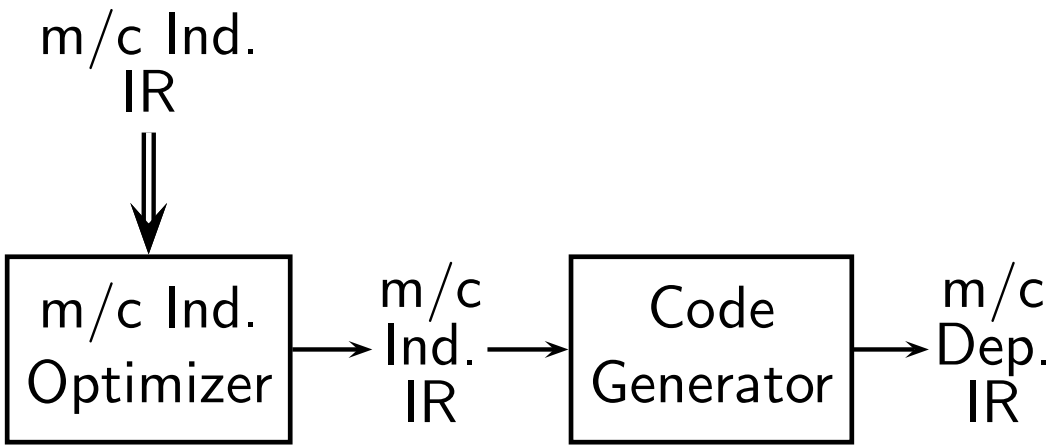


- Compile time evaluations
- Eliminating redundant computations

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

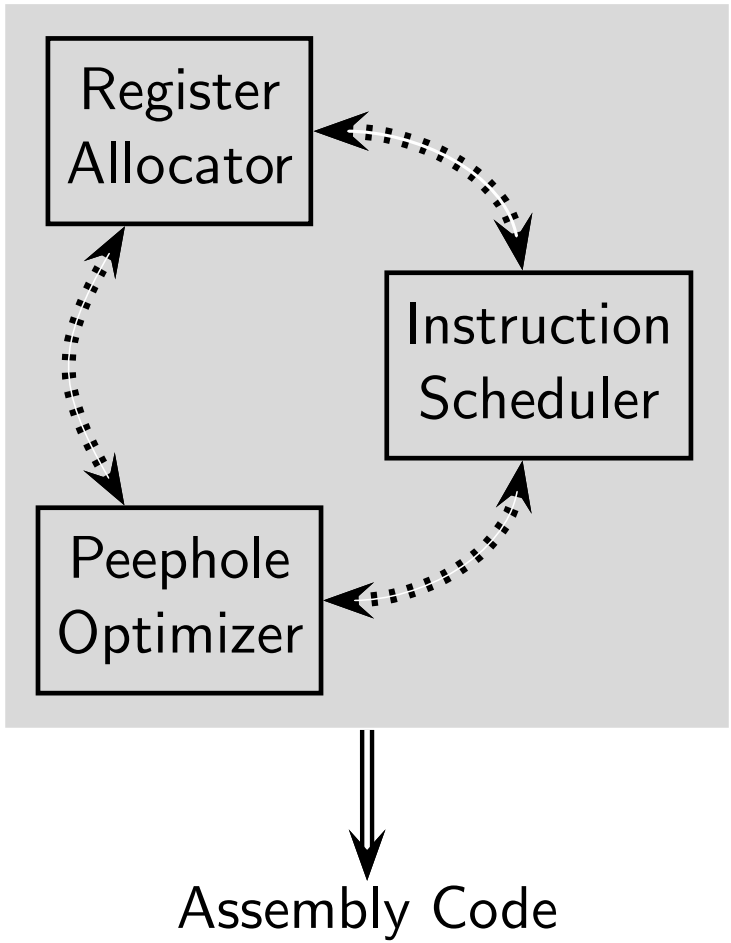


Typical Back Ends in Aho Ullman Model



- Compile time evaluations
- Eliminating redundant computations

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



Retargetability in Aho Ullman and Davidson Fraser Models

<table>
<tr><th></th><th>Aho Ullman Model</th><th>Davidson Fraser Model</th></tr>
<tr><td rowspan="2">Instruction Selection</td><td colspan="2">• 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</td></tr>
<tr><td></td><td></td></tr>
<tr><td>Optimization</td><td></td><td></td></tr>
</table>

Retargetability in Aho Ullman and Davidson Fraser Models

<table>
<tr><th></th><th>Aho Ullman Model</th><th>Davidson Fraser Model</th></tr>
<tr><td rowspan="2">Instruction Selection</td><td colspan="2">• 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</td></tr>
<tr><td>Cost based tree pattern matching</td><td></td></tr>
<tr><td>Optimization</td><td></td><td></td></tr>
</table>

Retargetability in Aho Ullman and Davidson Fraser Models

<table>
<tr><th></th><th>Aho Ullman Model</th><th>Davidson Fraser Model</th></tr>
<tr><td rowspan="2">Instruction Selection</td><td colspan="2">• 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</td></tr>
<tr><td>Cost based tree pattern matching</td><td>Structural tree pattern matching</td></tr>
<tr><td>Optimization</td><td></td><td></td></tr>
</table>

Retargetability in Aho Ullman and Davidson Fraser Models

<table>
<tr><th></th><th>Aho Ullman Model</th><th>Davidson Fraser Model</th></tr>
<tr><td rowspan="2">Instruction Selection</td><td colspan="2">• 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</td></tr>
<tr><td>Cost based tree pattern matching</td><td>Structural tree pattern matching</td></tr>
<tr><td>Optimization</td><td>Machine independent</td><td></td></tr>
</table>

Retargetability in Aho Ullman and Davidson Fraser Models

<table>
<tr><th></th><th>Aho Ullman Model</th><th>Davidson Fraser Model</th></tr>
<tr><td rowspan="2">Instruction Selection</td><td colspan="2">• 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</td></tr>
<tr><td>Cost based tree pattern matching</td><td>Structural tree pattern matching</td></tr>
<tr><td rowspan="2">Optimization</td><td rowspan="2">Machine independent</td><td>Machine dependent</td></tr>
<tr><td></td></tr>
</table>

Retargetability in Aho Ullman and Davidson Fraser Models

<table>
<tr><th></th><th>Aho Ullman Model</th><th>Davidson Fraser Model</th></tr>
<tr><td rowspan="2">Instruction Selection</td><td colspan="2">• 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</td></tr>
<tr><td>Cost based tree pattern matching</td><td>Structural tree pattern matching</td></tr>
<tr><td rowspan="2">Optimization</td><td rowspan="2">Machine independent</td><td>Machine dependent</td></tr>
<tr><td>Key Insight: Register transfers are target specific but their form is target independent</td></tr>
</table>

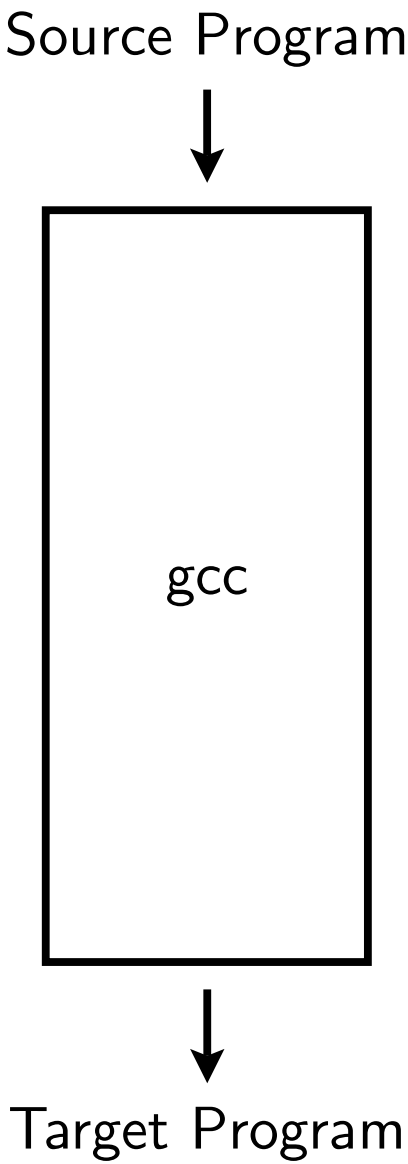
Part 2

GCC $\equiv$ *The Great Compiler Challenge*

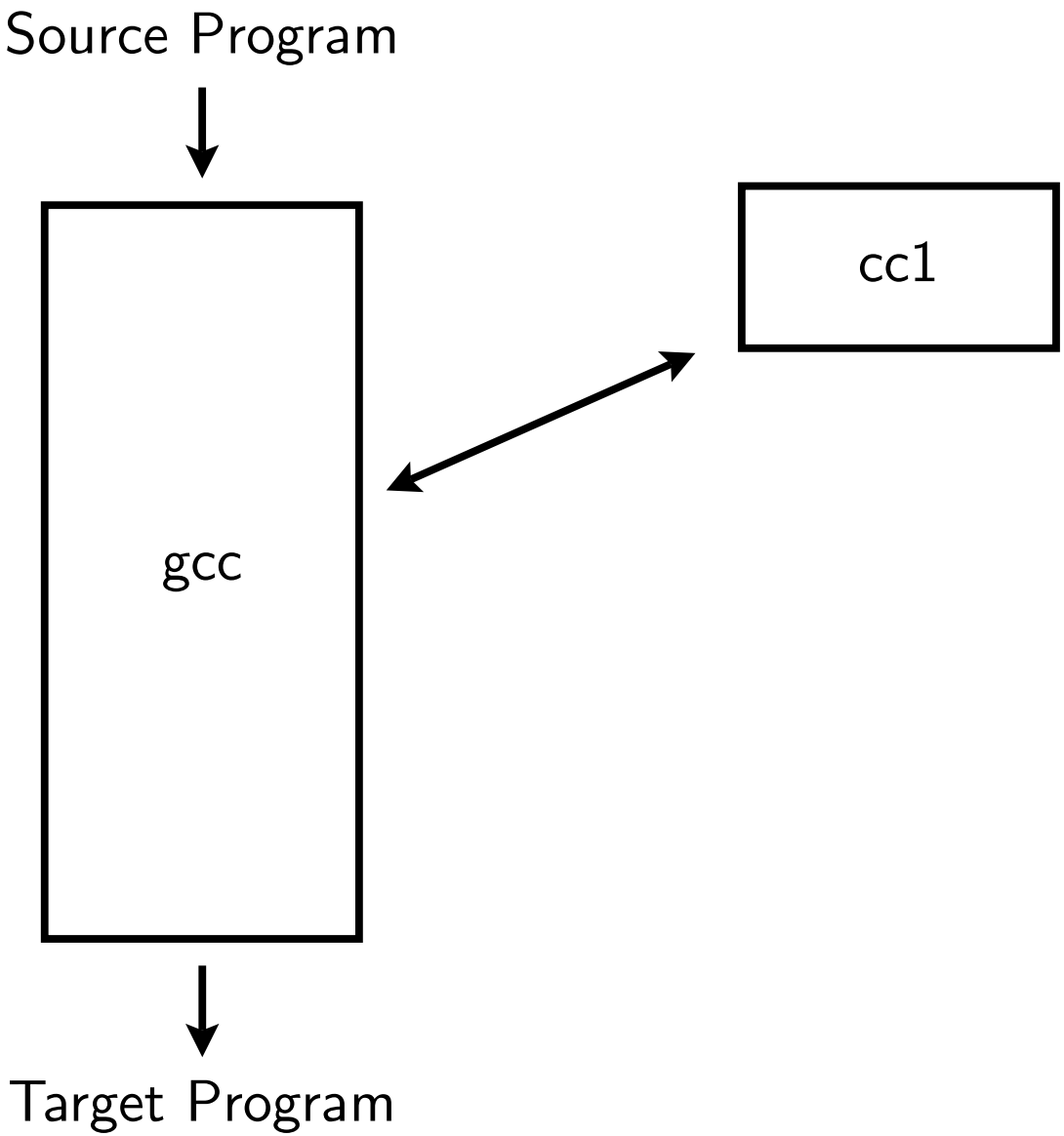
What is GCC?

- For the GCC developer community: The GNU Compiler Collection
- For other compiler writers: The Great Compiler Challenge 

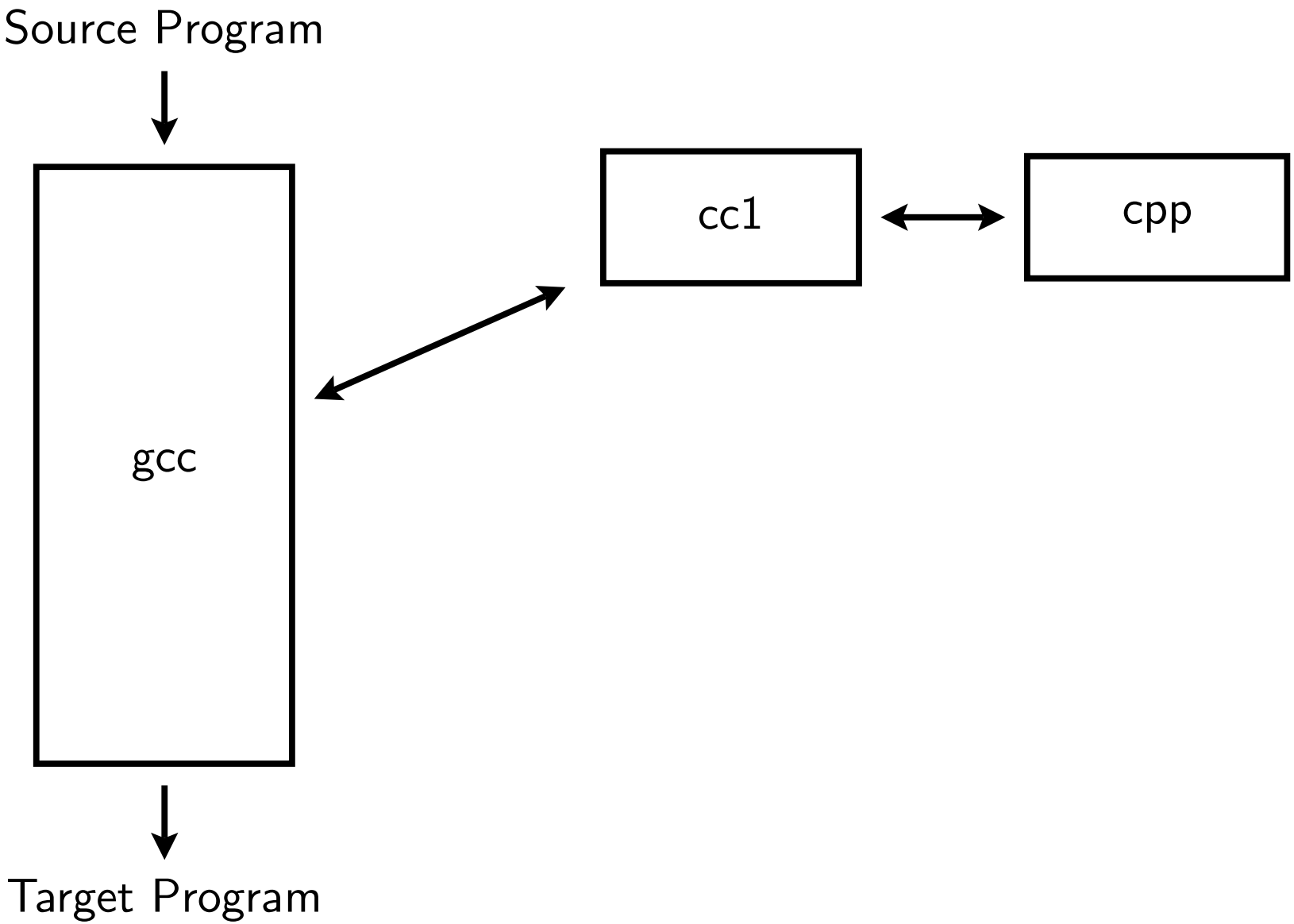
The GNU Tool Chain for C



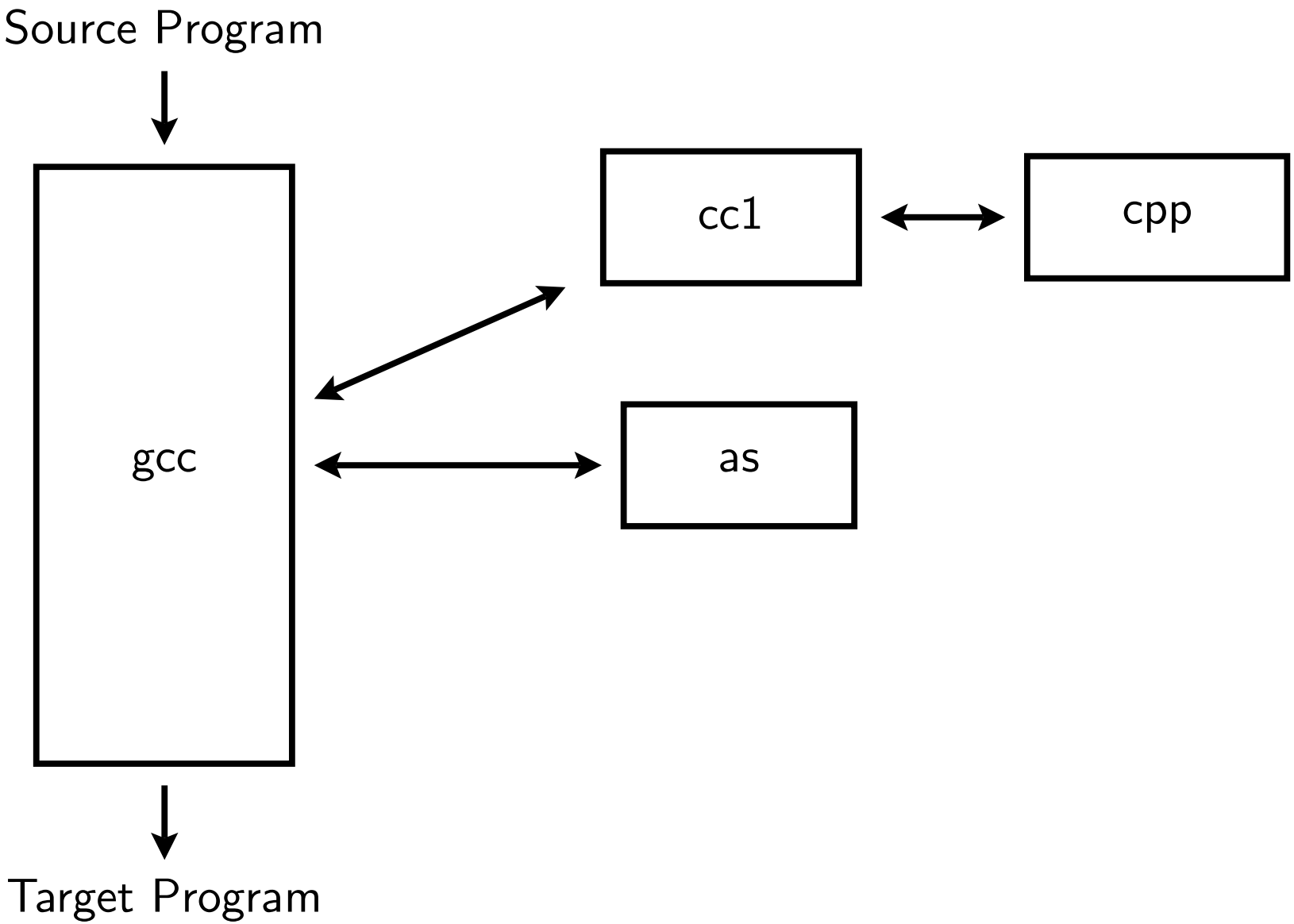
The GNU Tool Chain for C



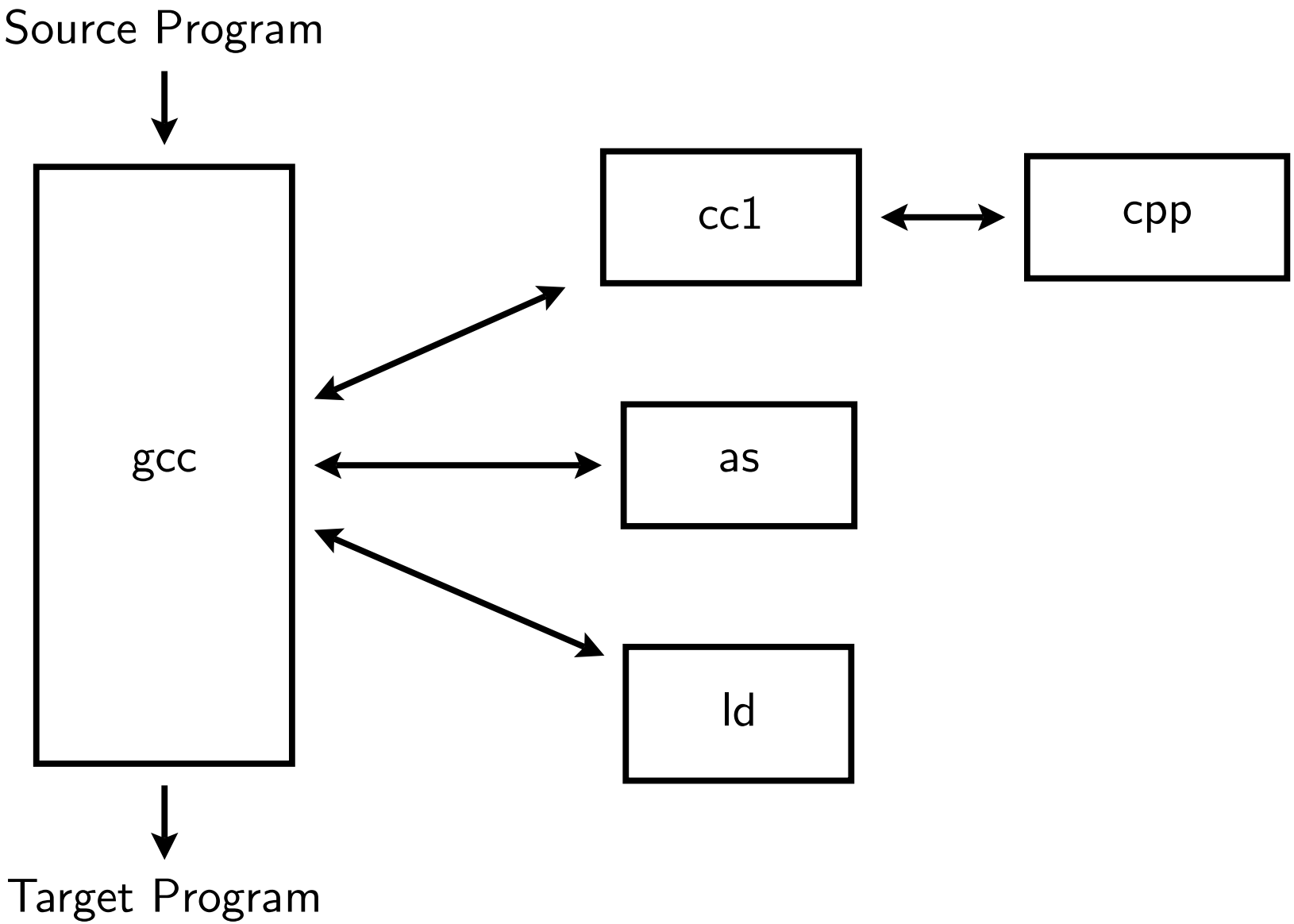
The GNU Tool Chain for C



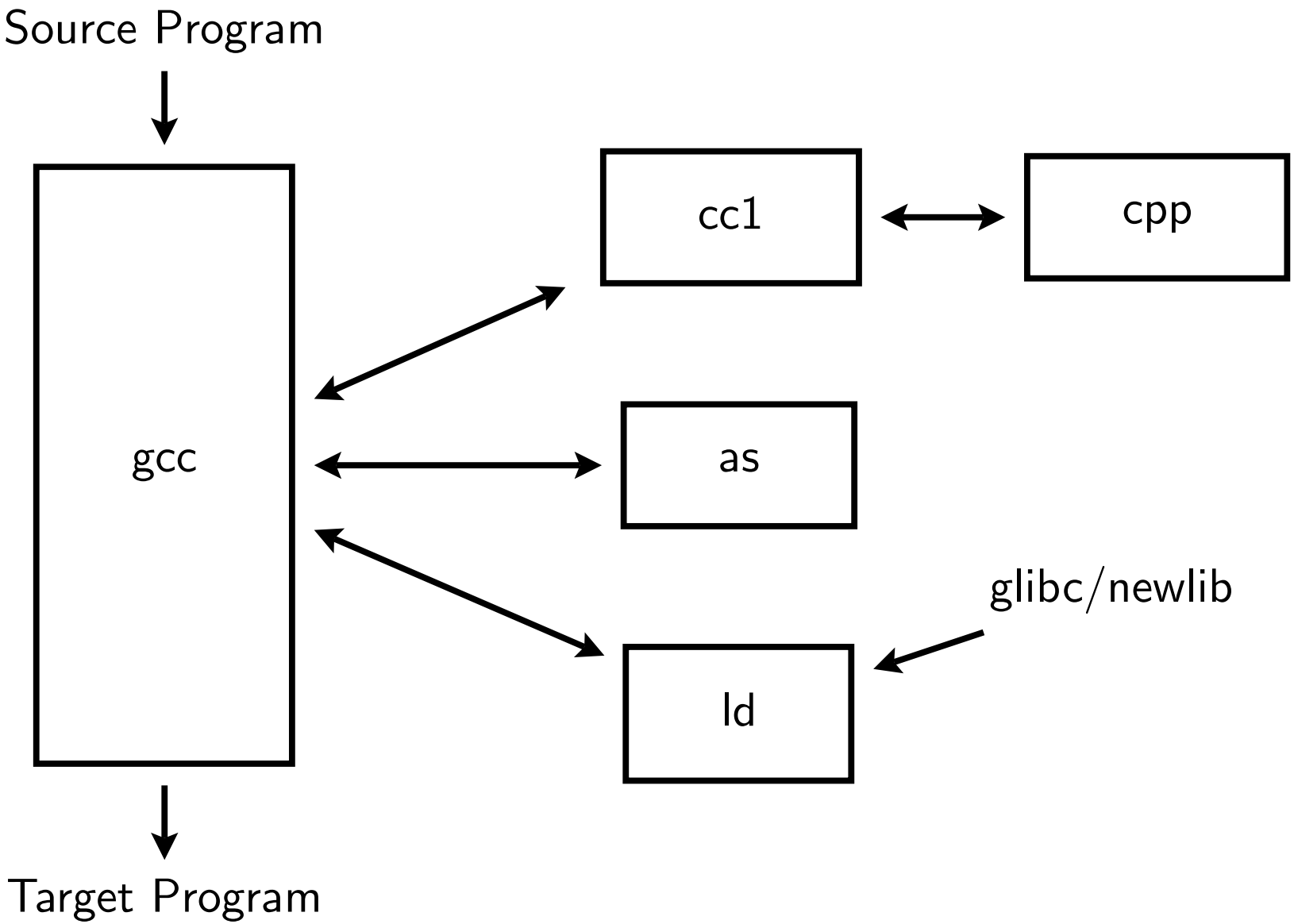
The GNU Tool Chain for C



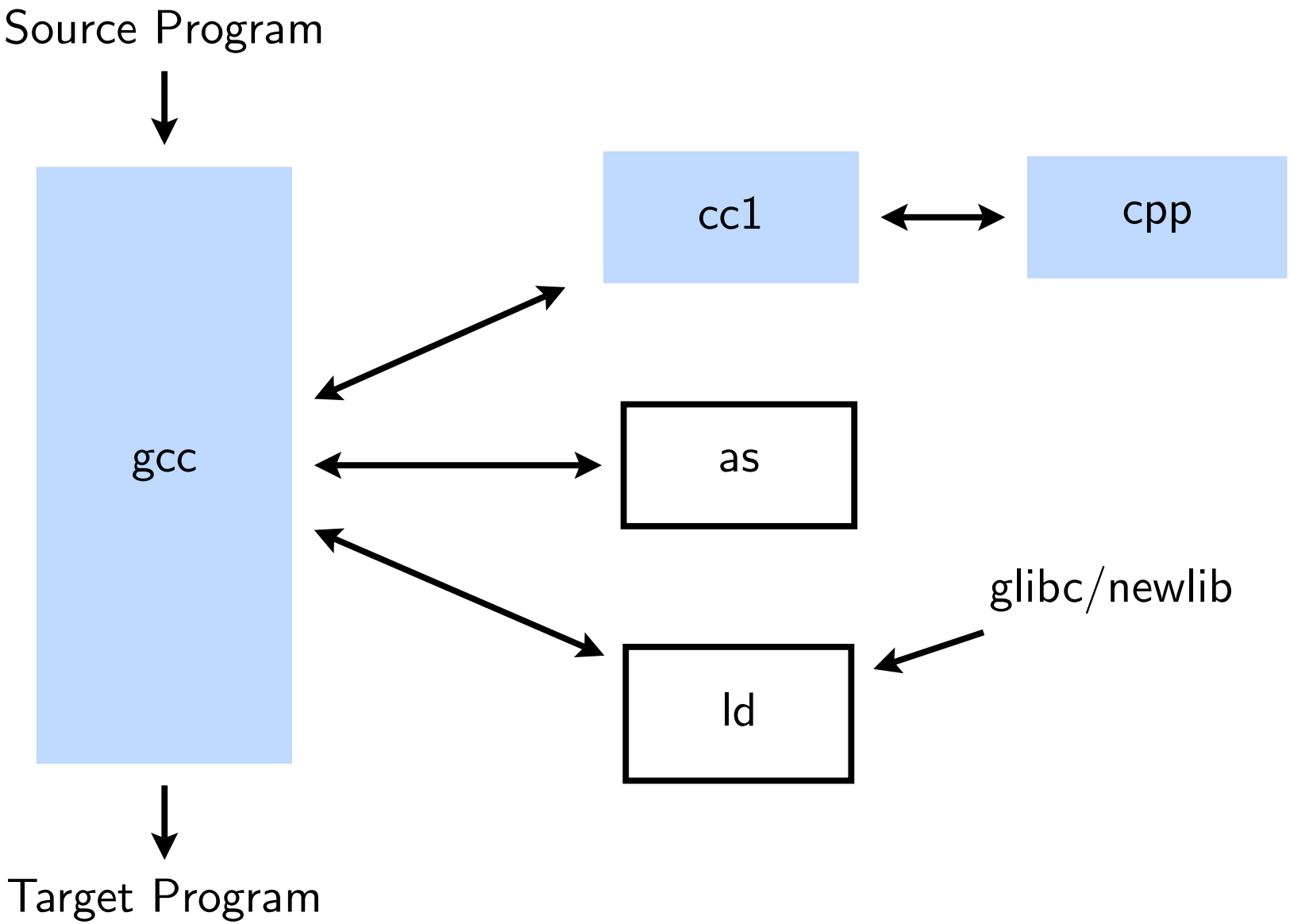
The GNU Tool Chain for C



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

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-4.6.0	4383	71096
gcc-4.7.2	4658	76287

- Core size (src/gcc)

	Subdirectories	Files
gcc-4.4.2	257	30163
gcc-4.6.0	336	36503
gcc-4.7.2	402	40193

- Machine Descriptions (src/gcc/config)

	Subdirectories	.c files	.h files	.md files
gcc-4.4.2	36	241	426	206
gcc-4.6.0	42	275	466	259
gcc-4.7.2	43	103	452	290

ohcount: Line Count of gcc-4.7.2

Language	Files	Code	Comment	Comment %	Blank	Total
c	20857	2289353	472640	17.1%	449939	3211932
cpp	23370	1030227	243717	19.1%	224079	1498023
ada	4913	726638	334360	31.5%	252044	1313042
java	6342	681938	645506	48.6%	169046	1496490
autoconf	94	428267	523	0.1%	66647	495437
html	336	151194	5667	3.6%	33877	190738
fortranfixed	3256	112286	2010	1.8%	15599	129895
make	106	110762	3875	3.4%	13811	128448
xml	76	50179	571	1.1%	6048	56798
assembler	240	49903	10975	18.0%	8584	69462
shell	157	49148	10848	18.1%	6757	66753
objective_c	882	28226	5267	15.7%	8324	41817
fortranfree	872	14474	3445	19.2%	1817	19736
tex	2	11060	5776	34.3%	1433	18269
scheme	6	11023	1010	8.4%	1205	13238
automake	72	10496	1179	10.1%	1582	13257
perl	29	4551	1322	22.5%	854	6727
ocaml	6	2830	576	16.9%	378	3784
xslt	20	2805	436	13.5%	563	3804
awk	16	2103	556	20.9%	352	3011
python	10	1672	400	19.3%	400	2472
css	25	1590	143	8.3%	332	2065
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	3	208	87	29.5%	33	328
haskell	49	153	0	0.0%	17	170
matlab	2	57	0	0.0%	8	65
Total	61760	5773867	1751733	23.3%	1264262	8789862

Language	Files	Code	Comment	Comment %	Blank	Total
c	20857	2289353	472640	17.1%	449939	3211932
cpp	23370	1030227	243717	19.1%	224079	1498023
ada	4913	726638	334360	31.5%	252044	1313042
java	6342	681938	645506	48.6%	169046	1496490
autoconf	94	428267	523	0.1%	66647	495437
html	336	151194	5667	3.6%	33877	190738
fortranfixed	3256	112286	2010	1.8%	15599	129895
make	106	110762	3875	3.4%	13811	128448
xml	76	50179	571	1.1%	6048	56798
assembler	240	49903	10975	18.0%	8584	69462
shell	157	49148	10848	18.1%	6757	66753
objective_c	882	28226	5267	15.7%	8324	41817
fortranfree	872	14474	3445	19.2%	1817	19736
tex	2	11060	5776	34.3%	1433	18269
scheme	6	11023	1010	8.4%	1205	13238
automake	72	10496	1179	10.1%	1582	13257
perl	29	4551	1322	22.5%	854	6727
ocaml	6	2830	576	16.9%	378	3784
xslt	20	2805	436	13.5%	563	3804
awk	16	2103	556	20.9%	352	3011
python	10	1672	400	19.3%	400	2472
css	25	1590	143	8.3%	332	2065
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	3	208	87	29.5%	33	328
haskell	49	153	0	0.0%	17	170
matlab	2	57	0	0.0%	8	65
Total	61760	5773867	1751733	23.3%	1264262	8789862

ohcount: Line Count of gcc-4.7.2/gcc

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

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

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

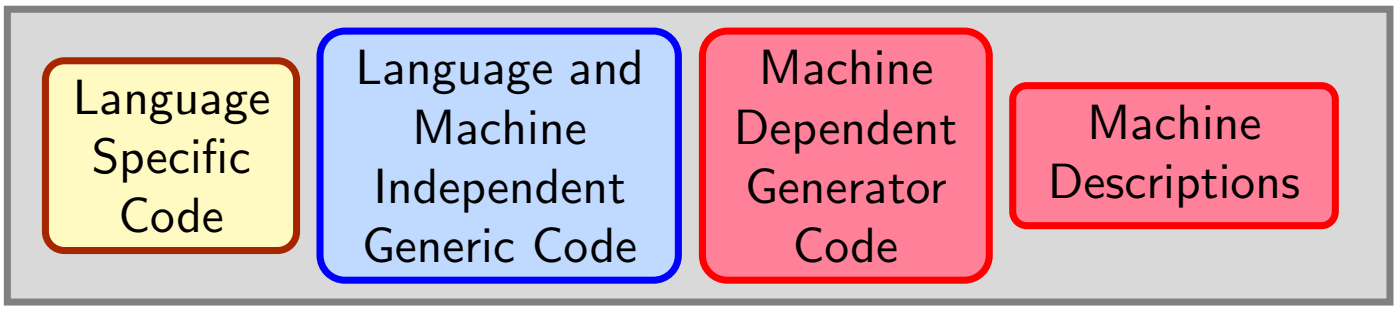
  - Input-output of the generation framework
    The target specification and the generated compiler
  - Input-output of the generated compiler
    A source program and the generated assembly program

- GCC generated compiler uses a derivative of the Davidson-Fraser model of compilation

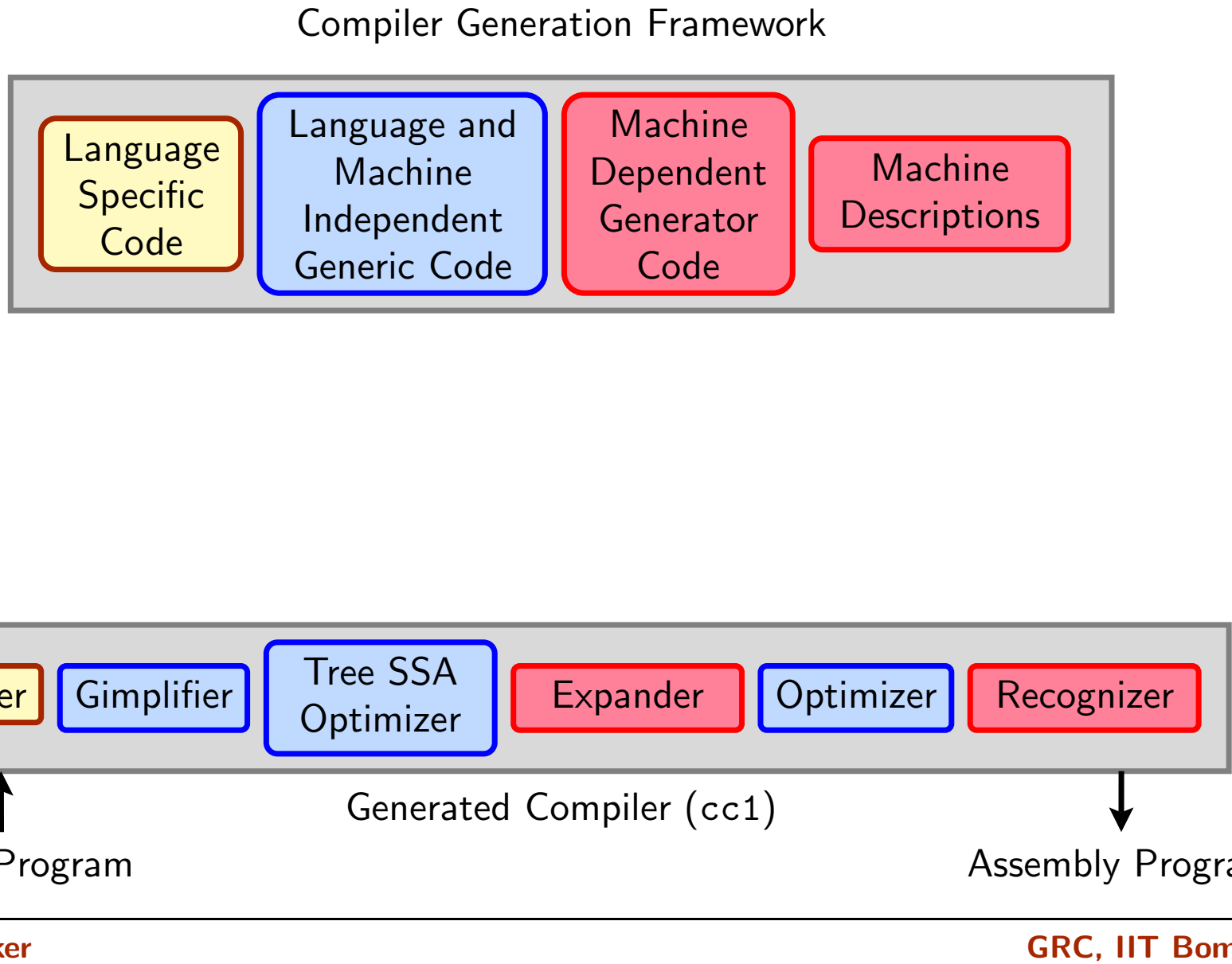
  - Early instruction selection
  - Machine dependent intermediate representation
  - Simplistic instruction selection and retargatibility mechanism

The Architecture of GCC

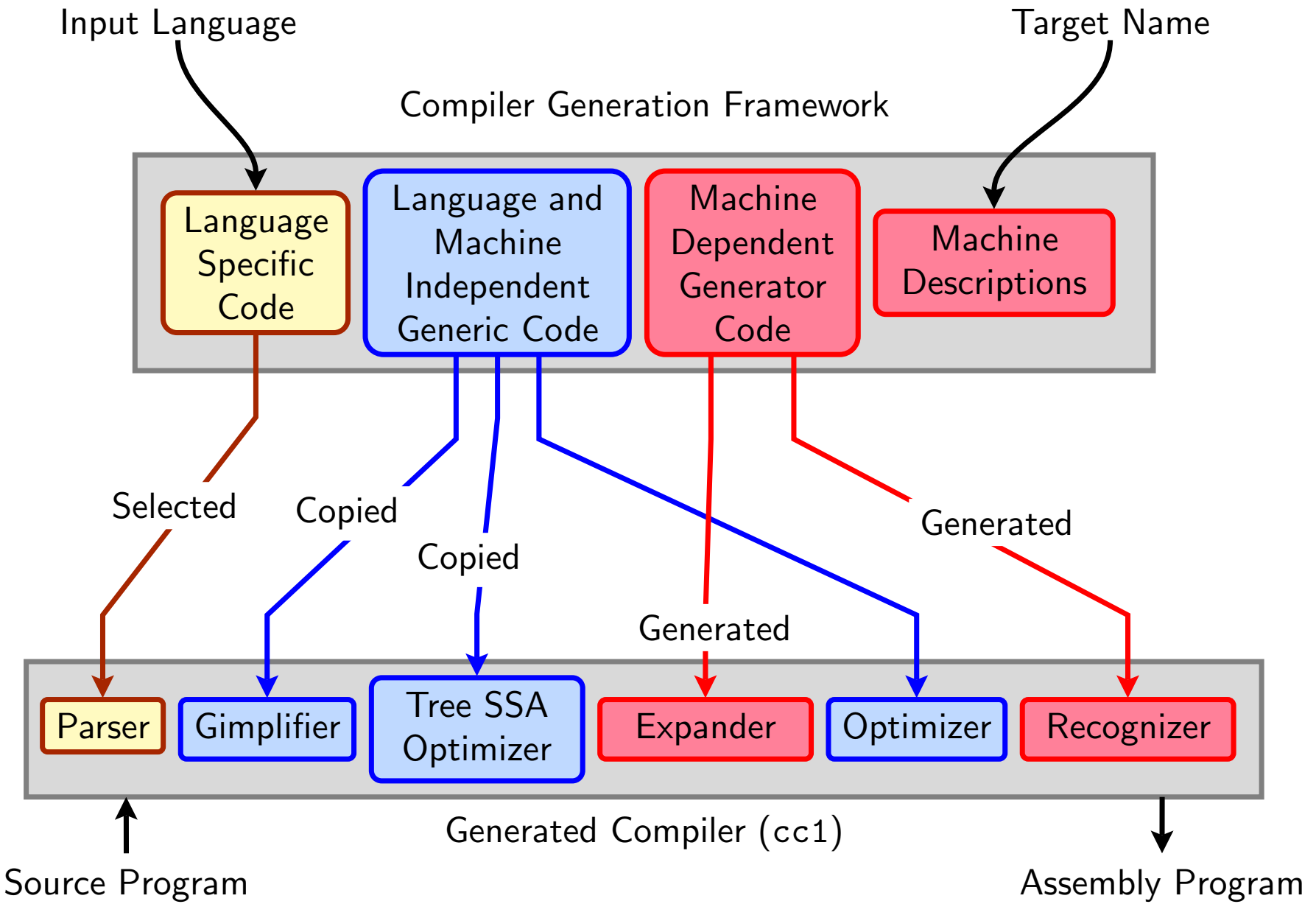
Compiler Generation Framework



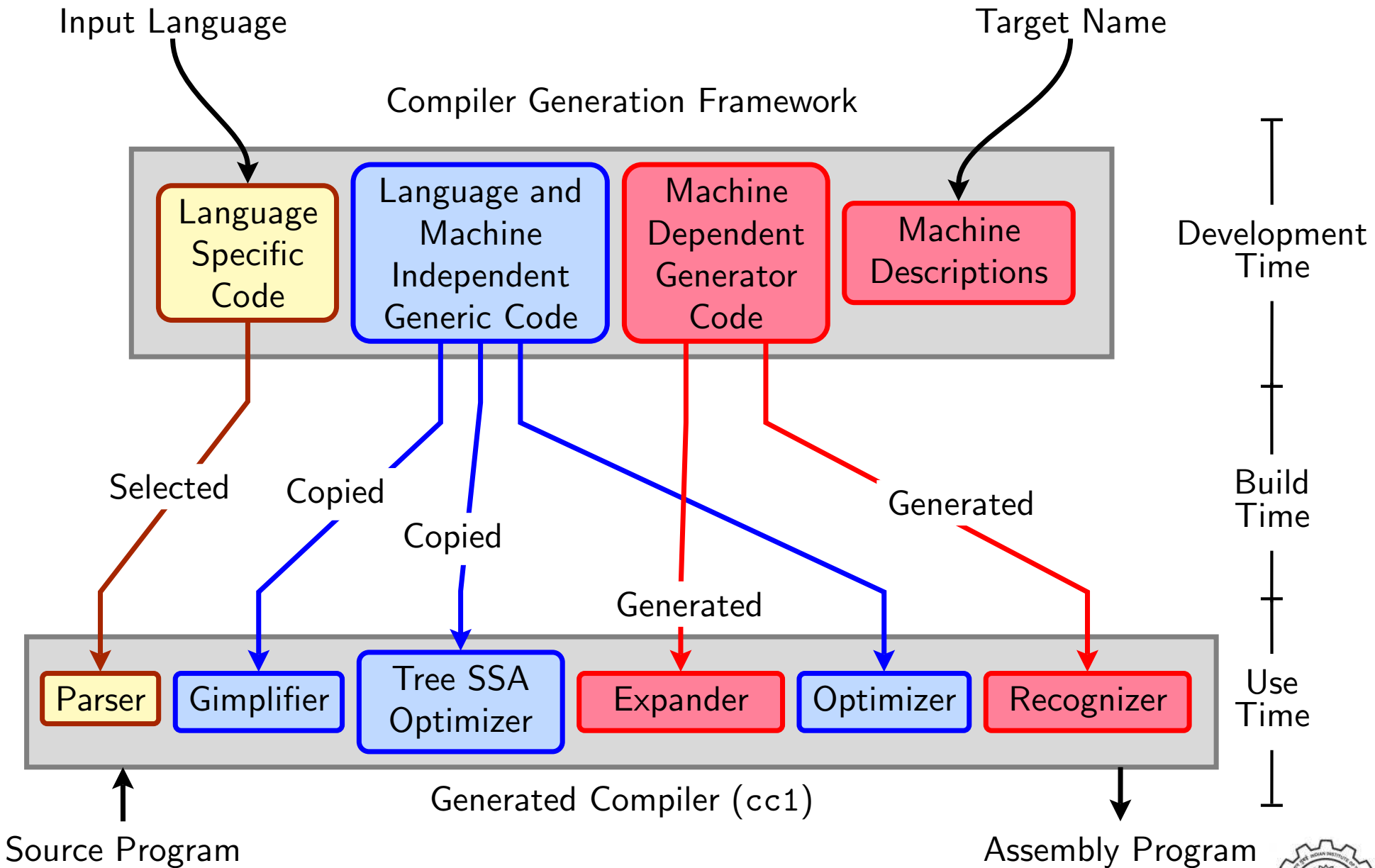
The Architecture of GCC



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 `gcc-4.7.2/gcc` using cscope -R

Another Example of The Generation Related Gap

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

  8125 occurrences!

Another Example of The Generation Related Gap

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

  8125 occurrences!

- Number of main functions in the entire tarball

Another Example of The Generation Related Gap

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

  8125 occurrences!

- Number of main functions in the entire tarball

  12799!

Another Example of The Generation Related Gap

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

  8125 occurrences!

- Number of main functions in the entire tarball

  12799!

- What if we do not search recursively?

Another Example of The Generation Related Gap

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

Another Example of The Generation Related Gap

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

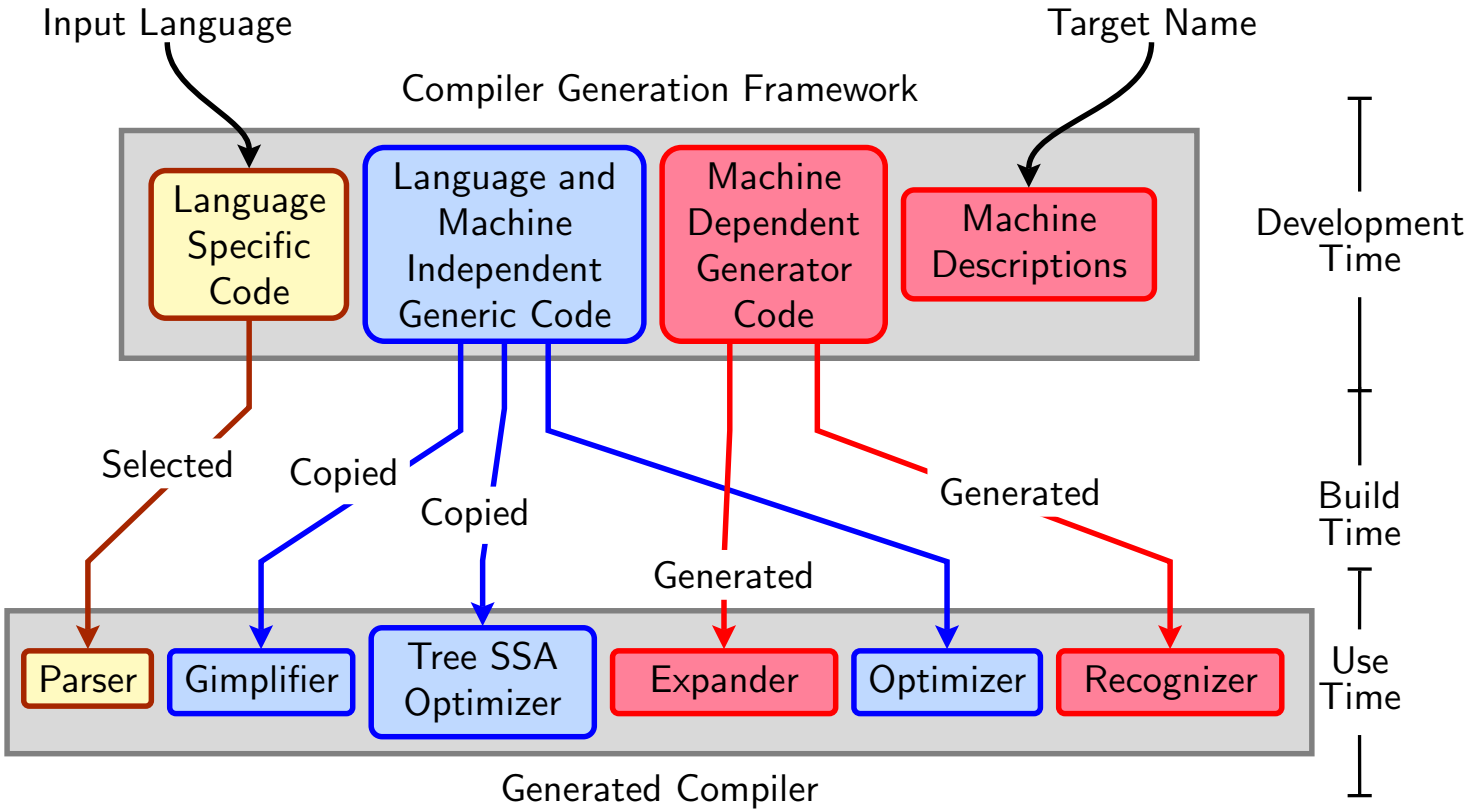
```
  File                    Line
0 s-oscons-tmplt.c         238 main (void ) {
1 collect2.c              1021 main (int argc, char **argv)
2 divtab-sh4-300.c          31 main ()
3 divtab-sh4.c              30 main ()
4 divtab.c                 131 main ()
5 gen-mul-tables.cc       1224 main ()
6 vms-ar.c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

Another Example of The Generation Related Gap

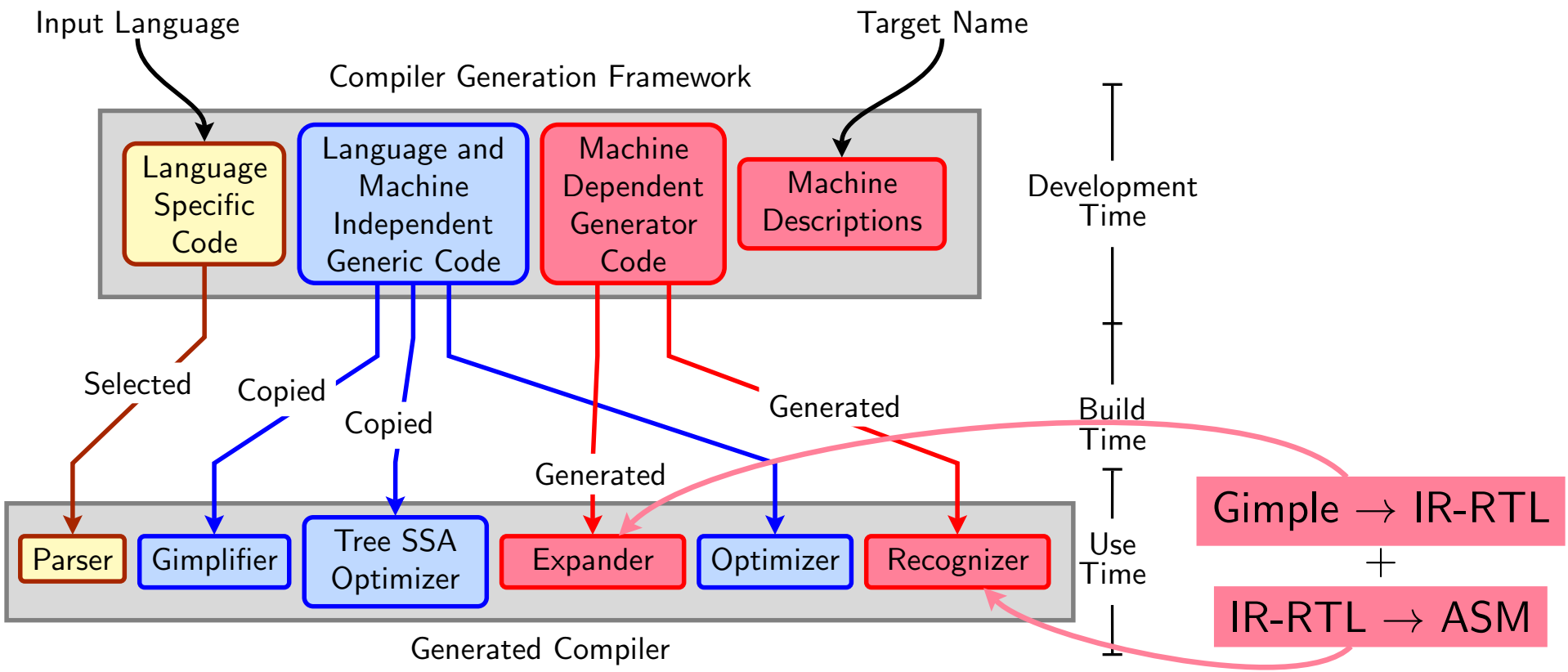
Locating the main function in the directory gcc-4.7.2/gcc using cscope

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

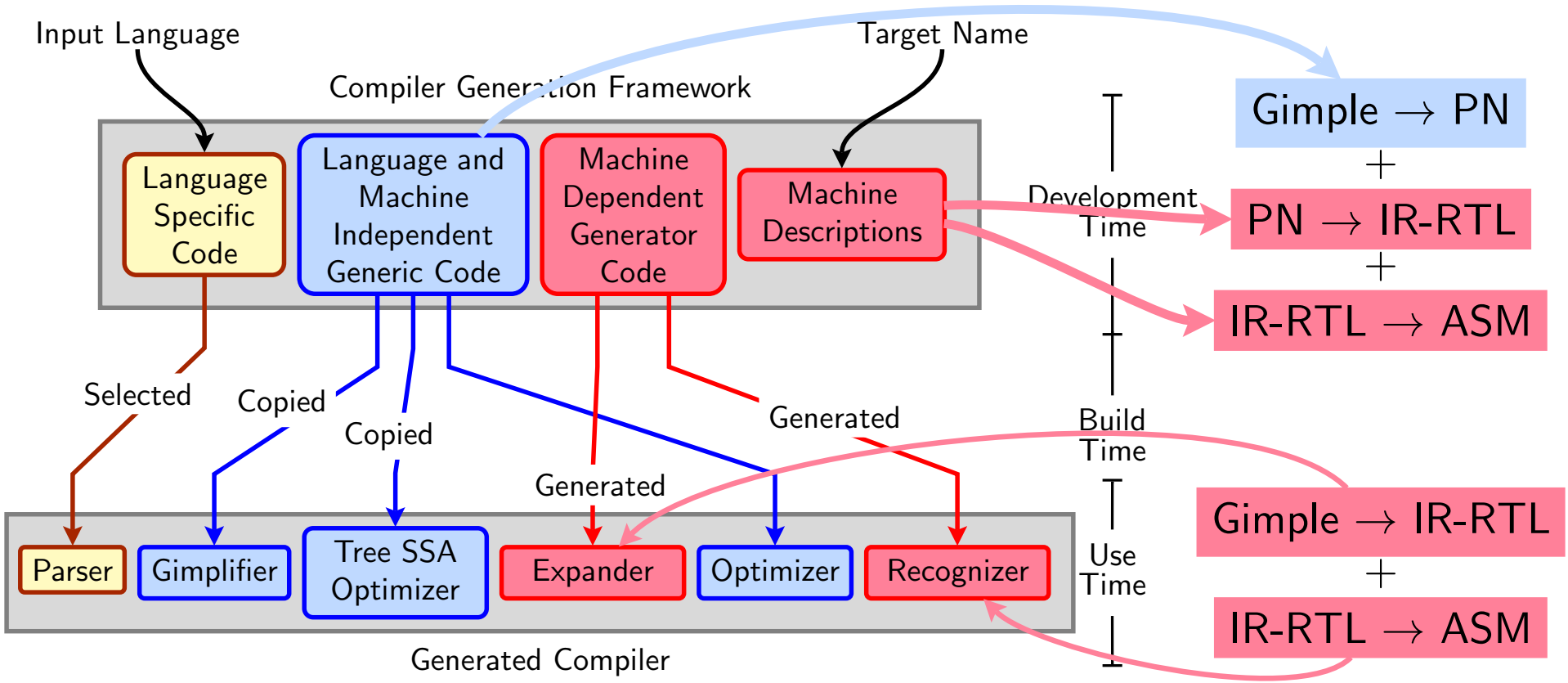
GCC Retargetability Mechanism



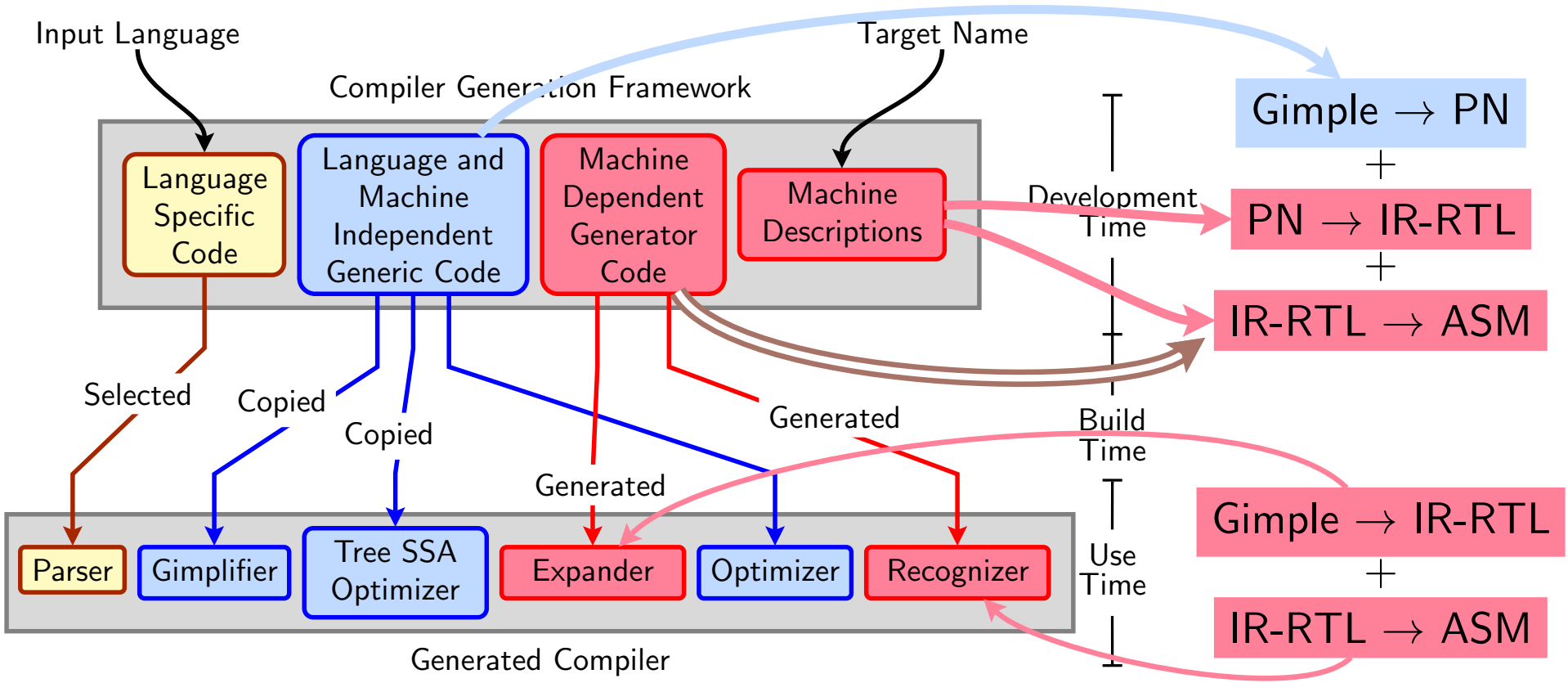
GCC Retargetability Mechanism



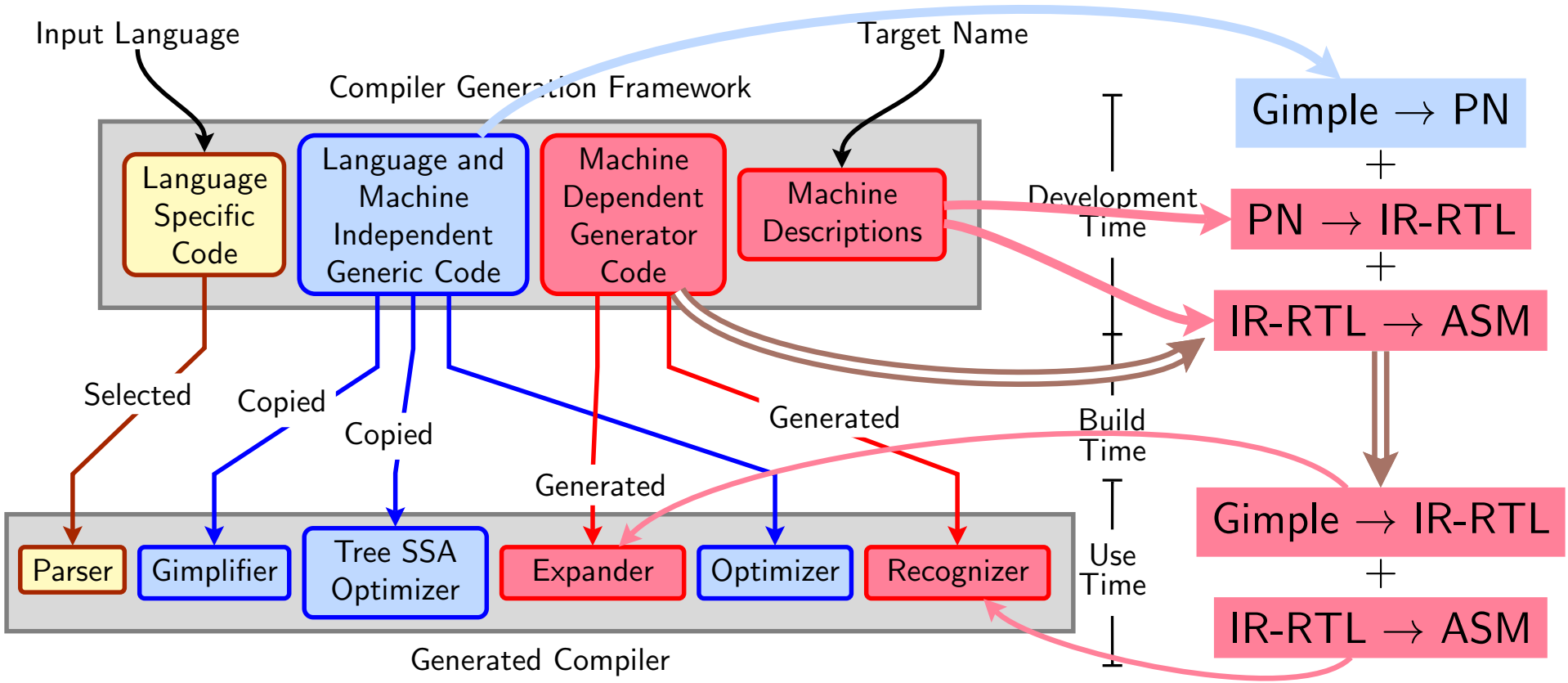
GCC Retargetability Mechanism



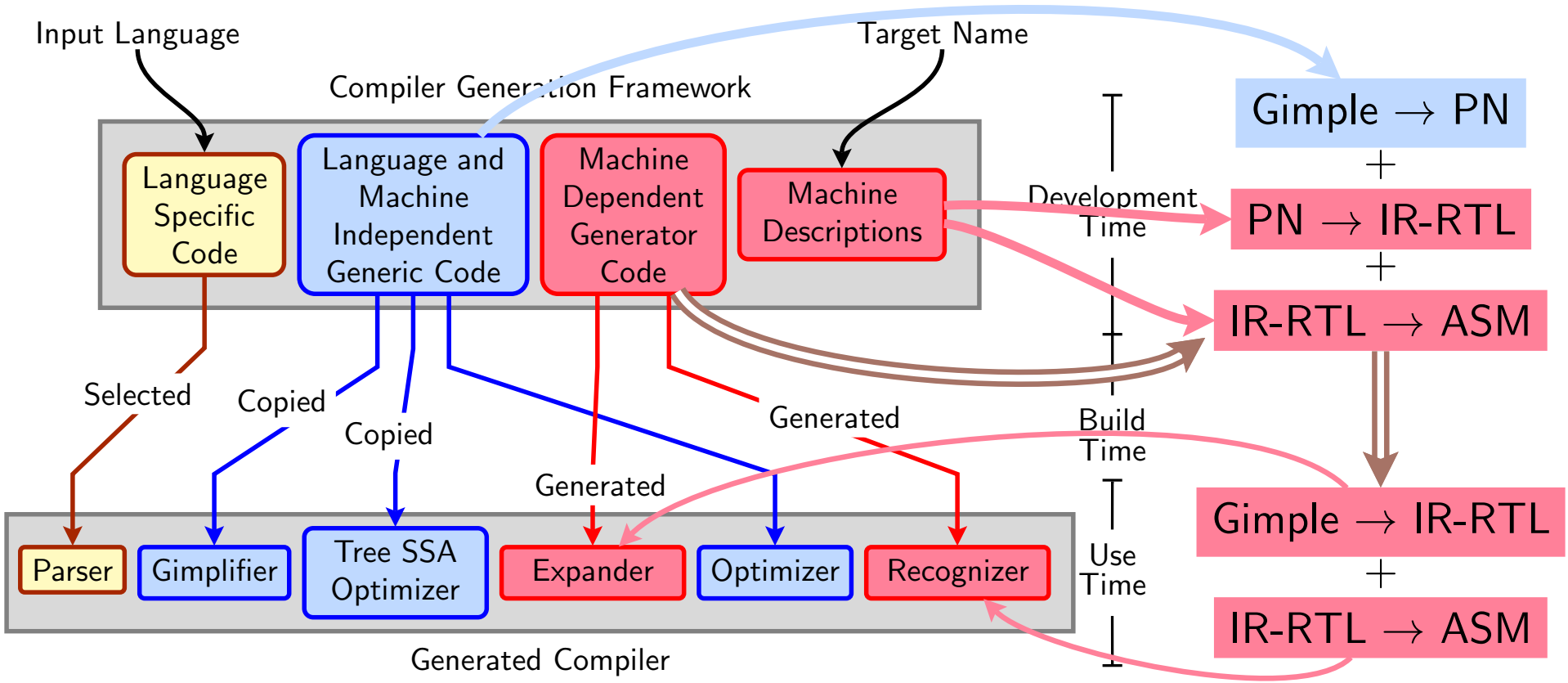
GCC Retargetability Mechanism



GCC Retargetability Mechanism



GCC Retargetability Mechanism



The generated compiler uses an adaptation of the Davidson Fraser model

- Generic expander and recognizer
- Machine specific information is isolated in data structures
- Generating a compiler involves generating these data structures

The GCC Challenge: Poor Retargetability Mechanism

Symptoms:

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

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

`gcc-4.6.2`

Files	i386	mips	arm
*.md	38851	15534	30951
*.c	39780	16793	26165
*.h	17879	5667	18713
Total	96510	37996	75929

`gcc-4.7.2`

Files	i386	mips	arm
*.md	39582	16437	32385
*.c	41985	17761	26006
*.h	19174	5586	18012
Total	100741	39784	76403

The GCC Challenge: Poor Retargetability Mechanism

Symptoms:

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

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

`gcc-4.6.2`

Files	i386	mips	arm
*.md	38851	15534	30951
*.c	39780	16793	26165
*.h	17879	5667	18713
Total	96510	37996	75929

`gcc-4.7.2`

Files	i386	mips	arm
*.md	39582	16437	32385
*.c	41985	17761	26006
*.h	19174	5586	18012
Total	100741	39784	76403

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

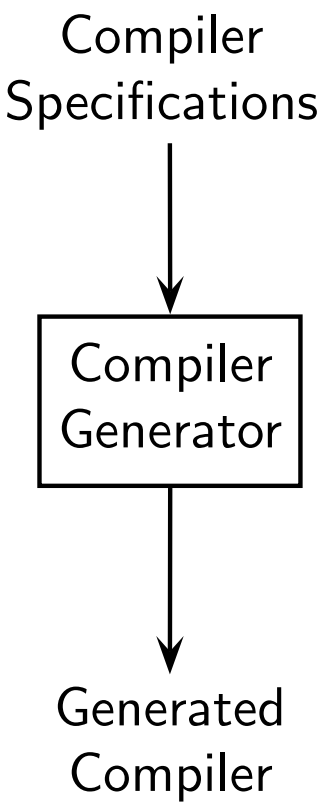
Part 3

Meeting the GCC Challenge

Meeting the GCC Challenge

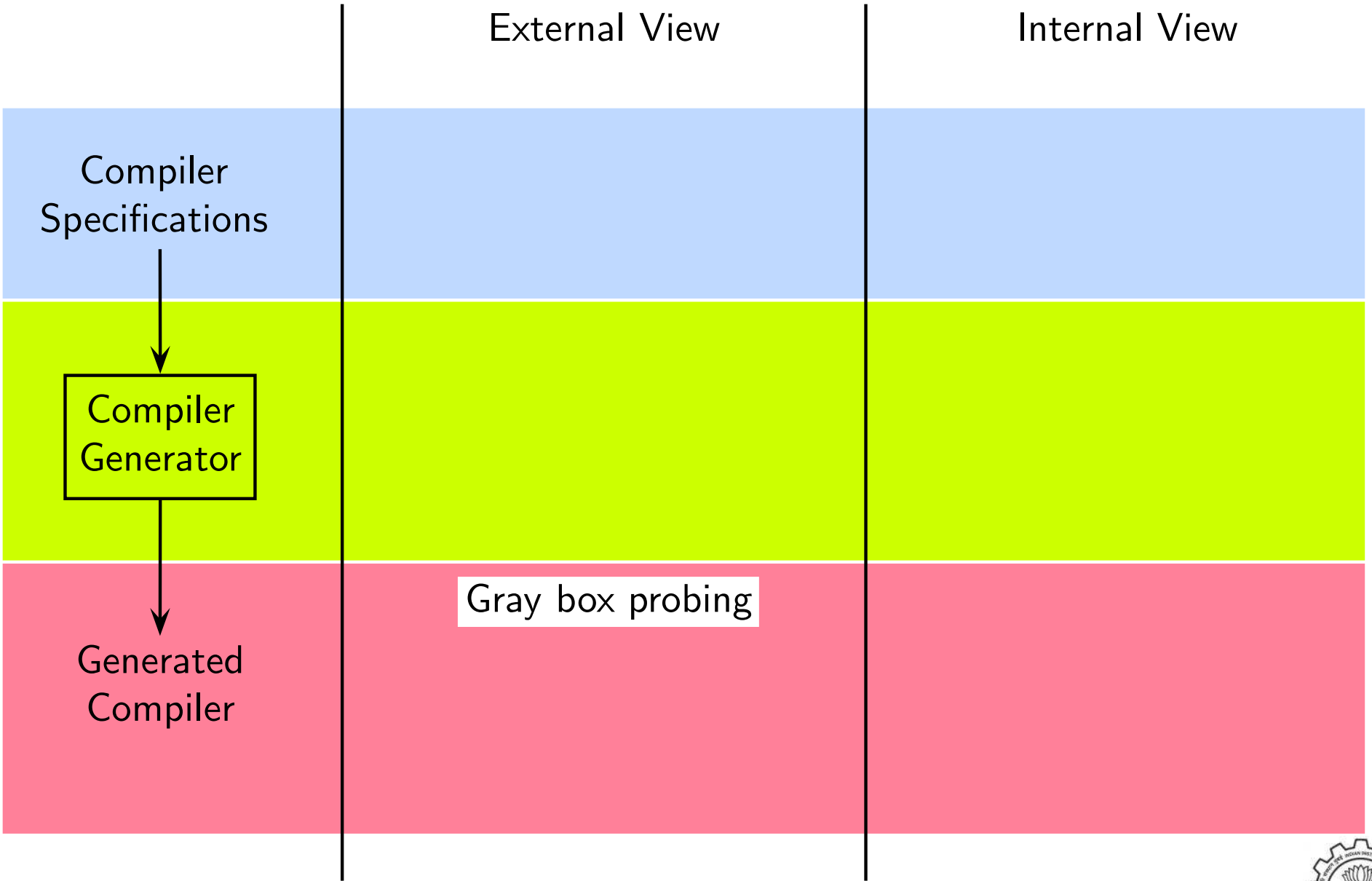
Goal of Understanding	Methodology	Needs Examining		
		Makefiles	Source	MD
Translation sequence of programs	Gray box probing	No	No	No
Build process	Customising the configuration and building	Yes	No	No
Retargetability issues and machine descriptions	Incremental construction of machine descriptions	No	No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

The Grand Picture and Our Coverage

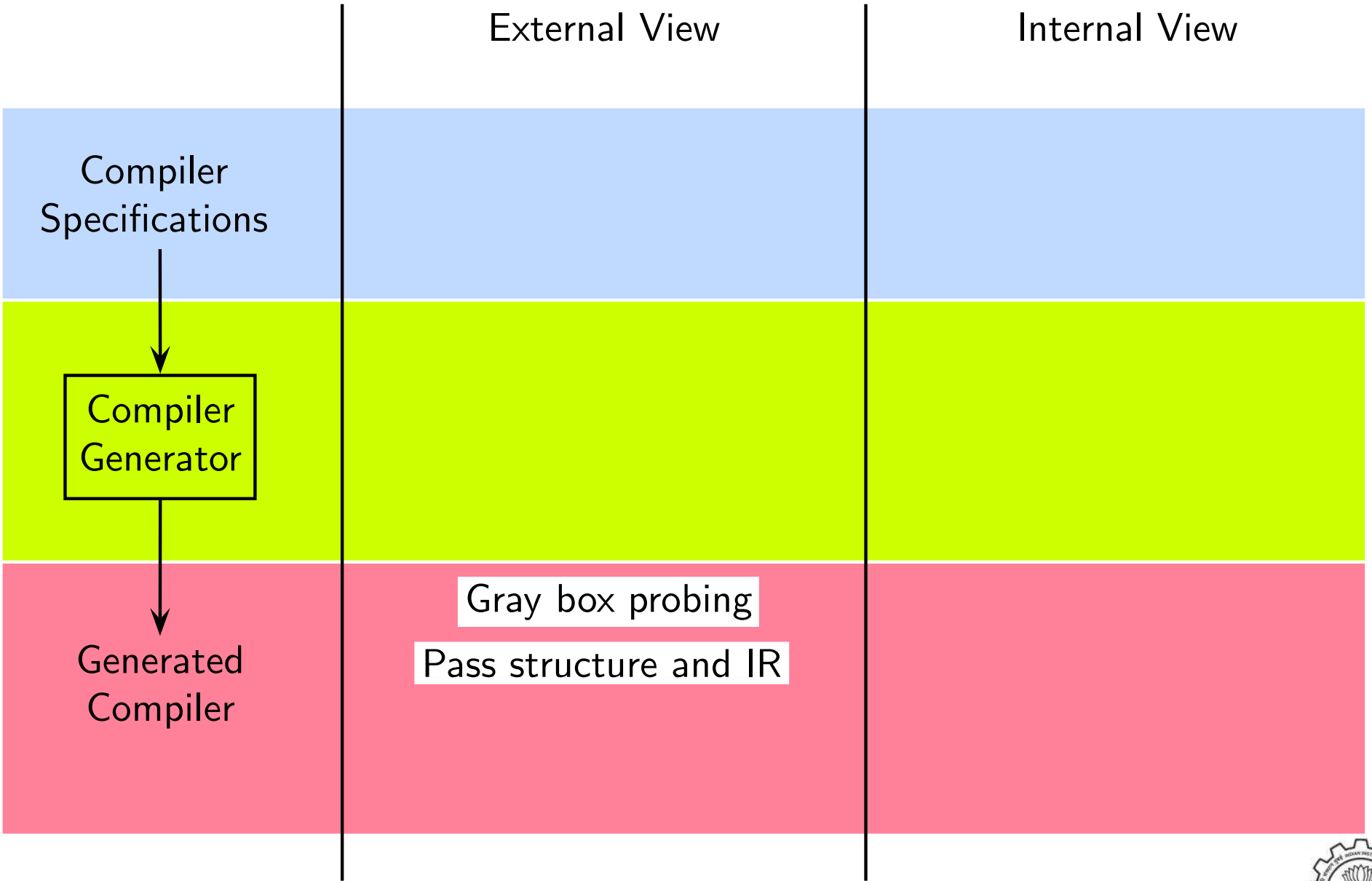


The Grand Picture and Our Coverage

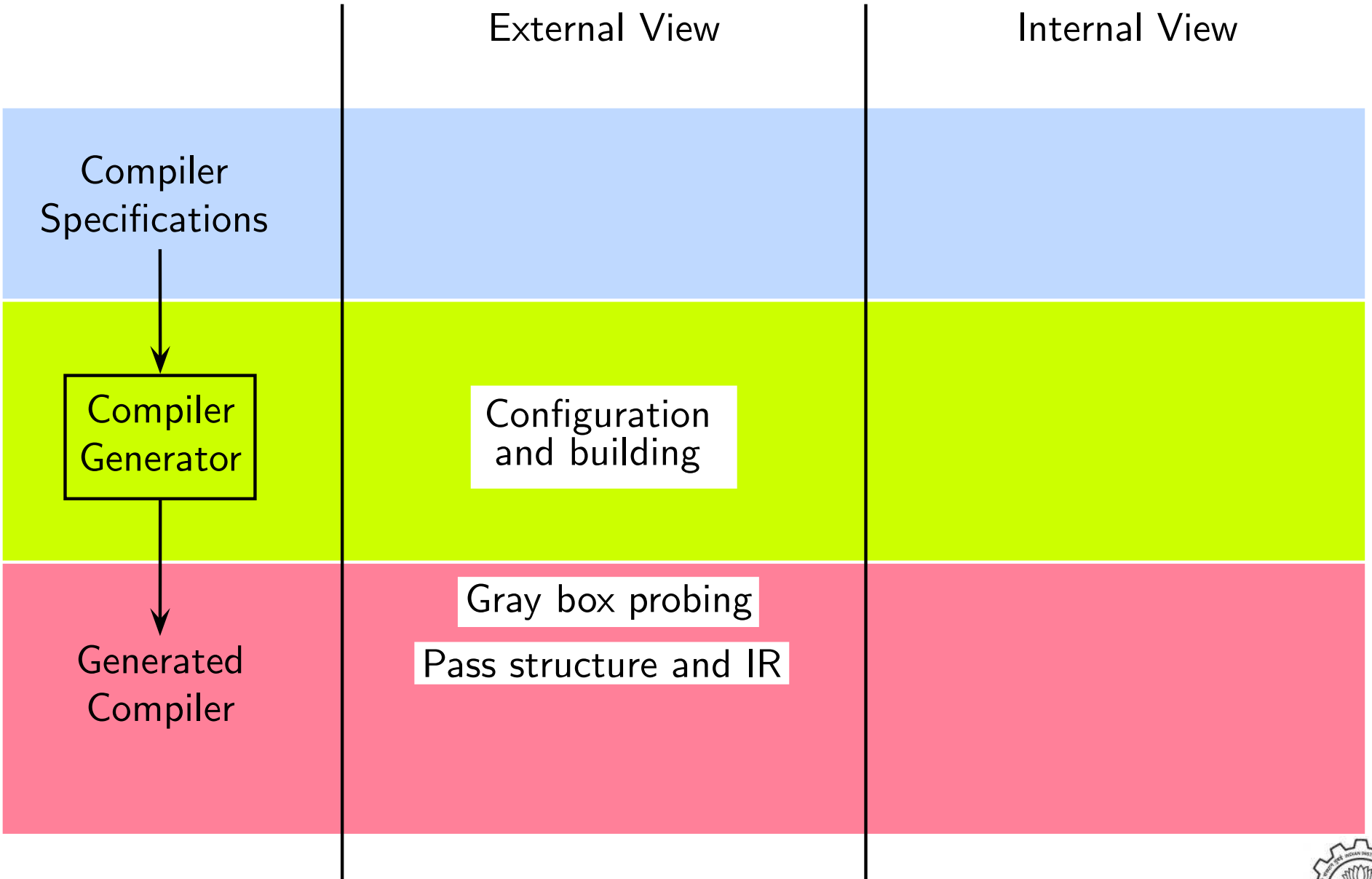
The Grand Picture and Our Coverage



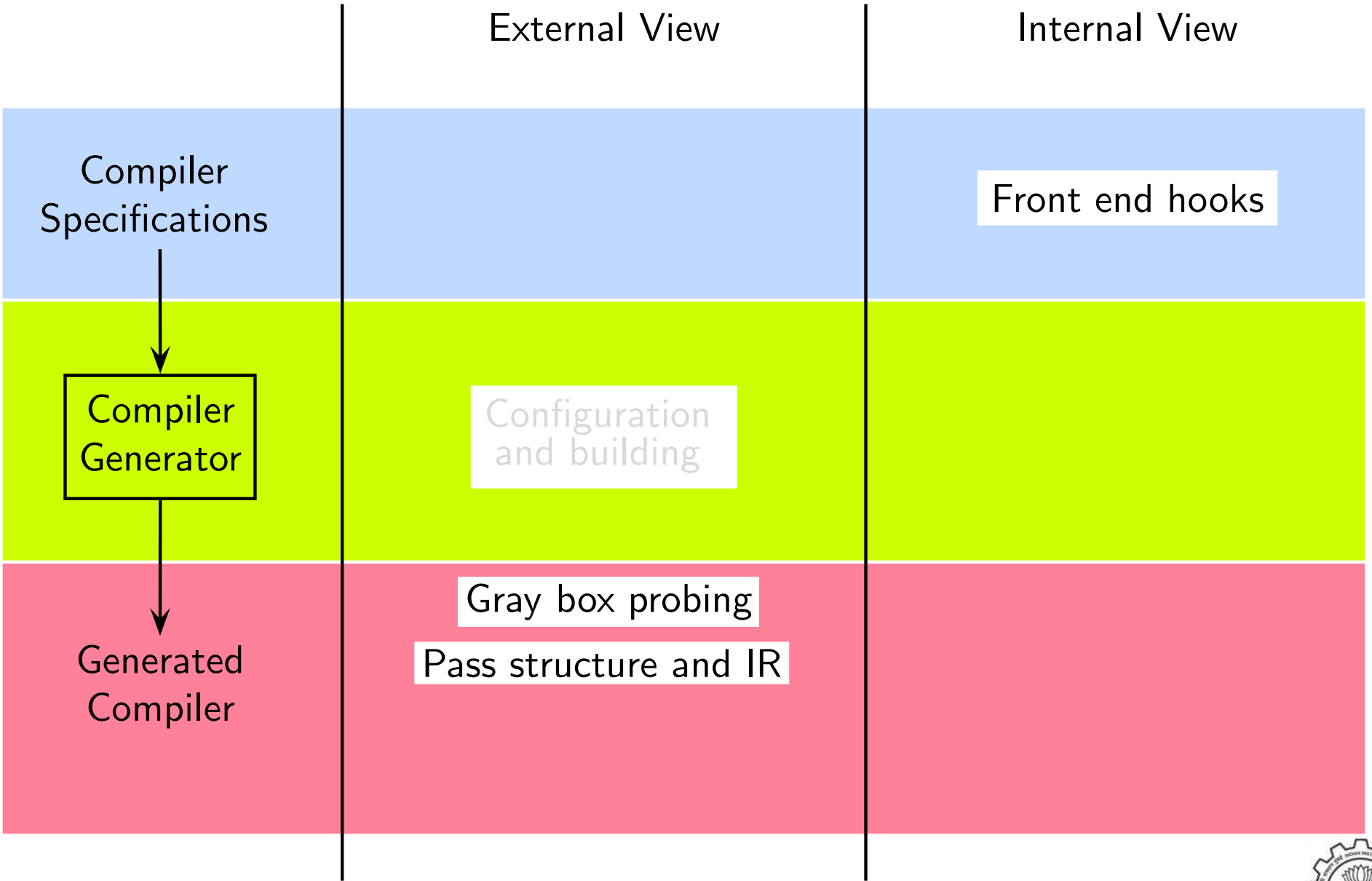
The Grand Picture and Our Coverage



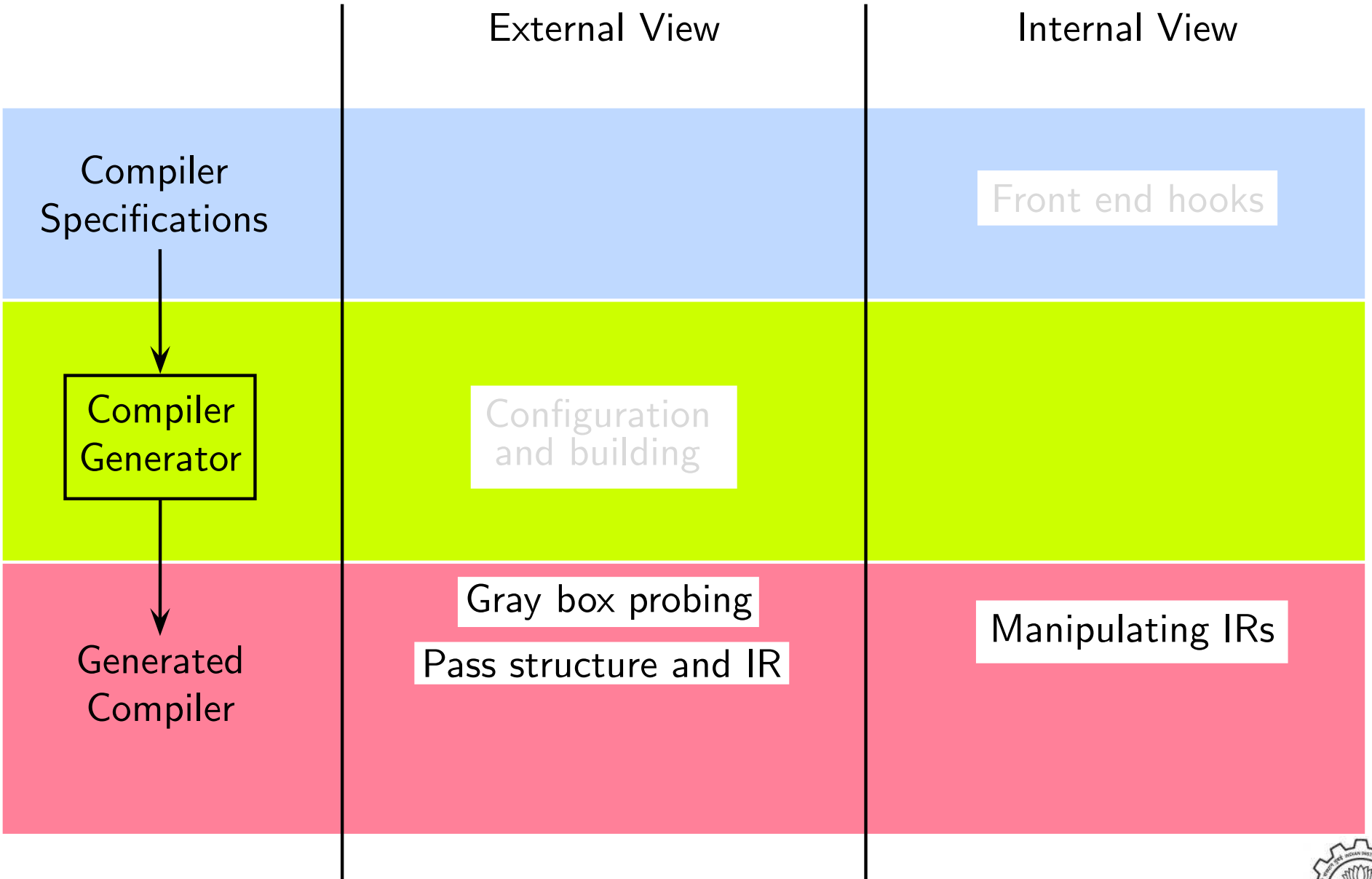
The Grand Picture and Our Coverage



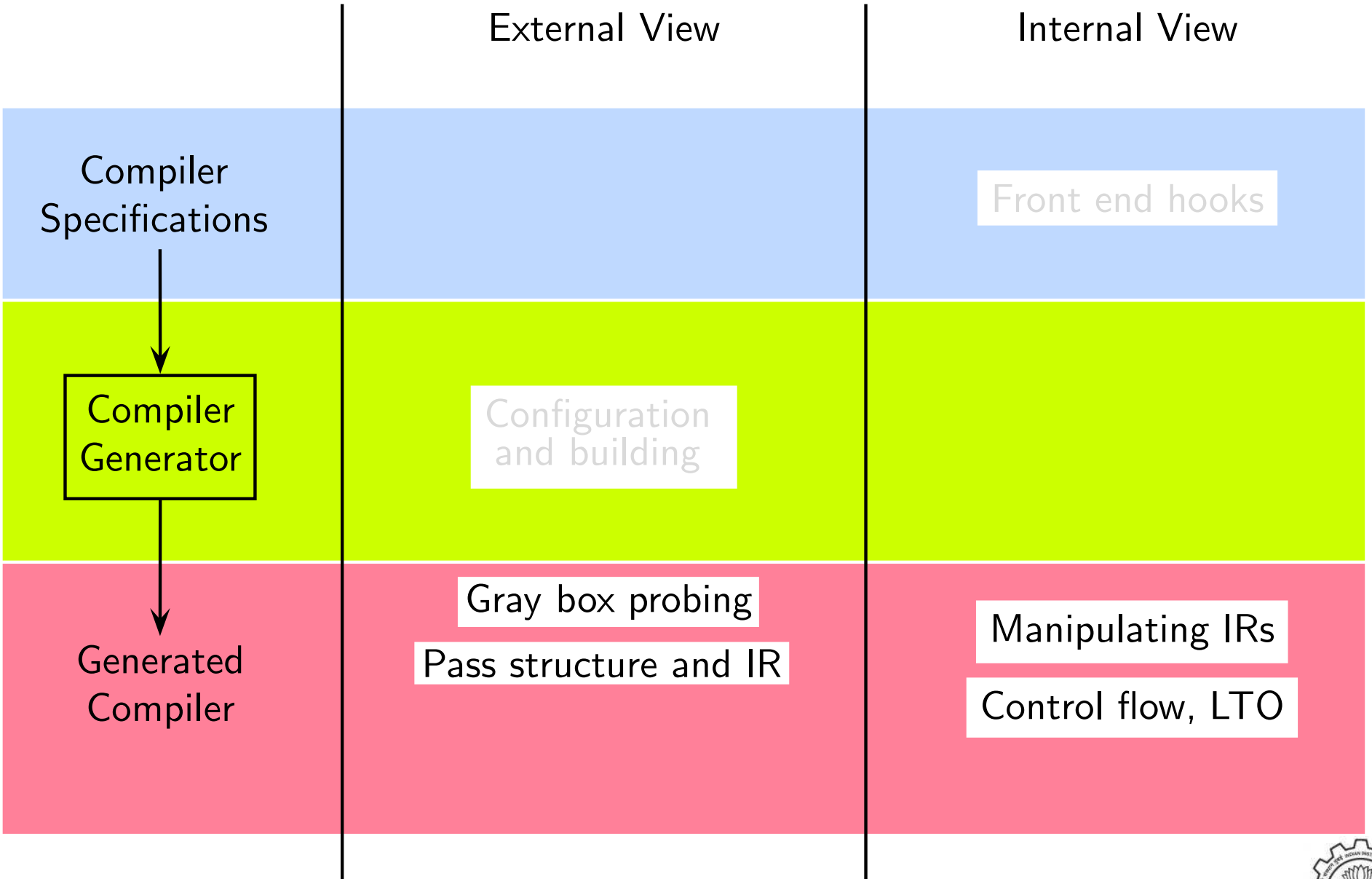
The Grand Picture and Our Coverage



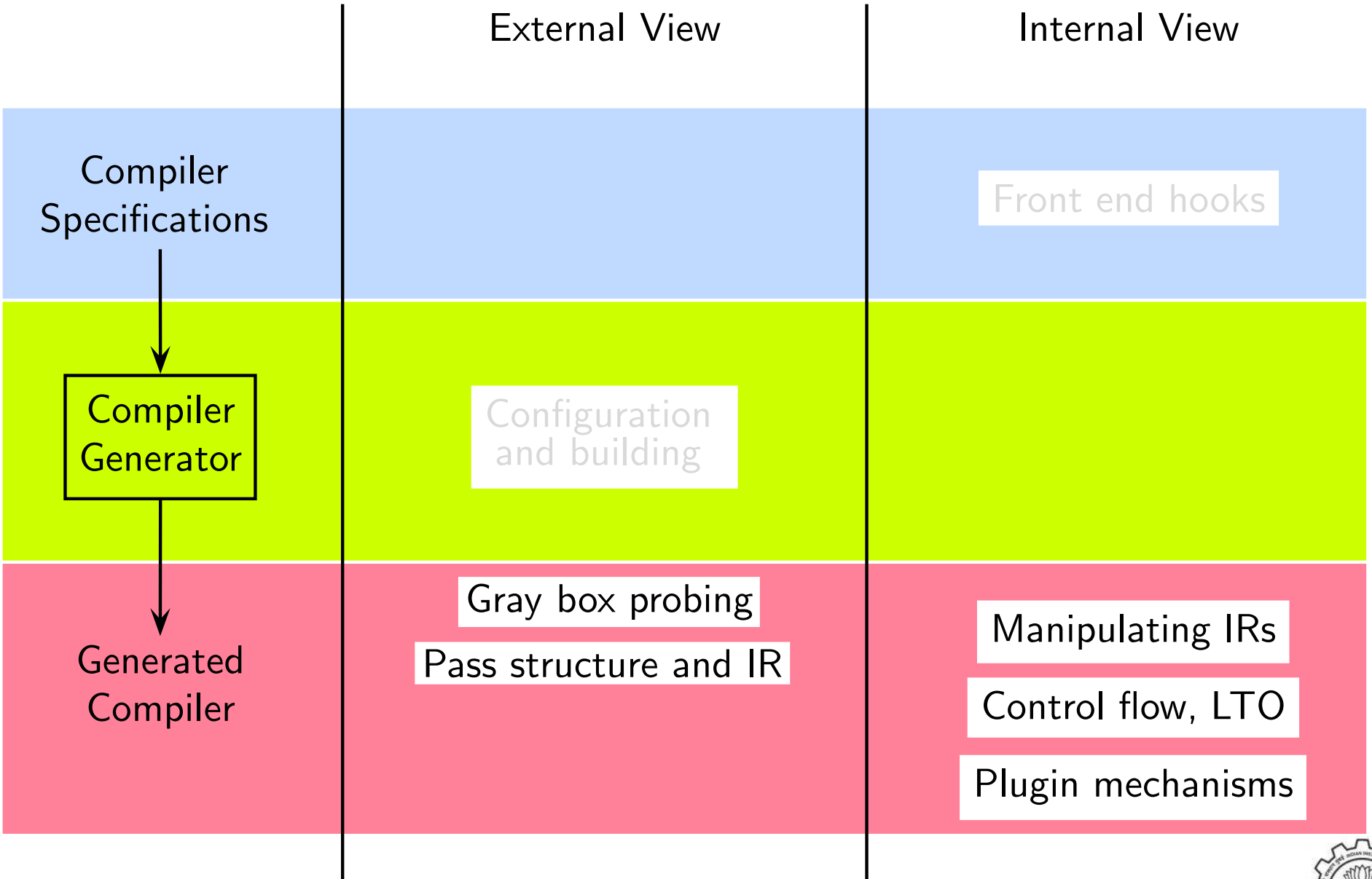
The Grand Picture and Our Coverage



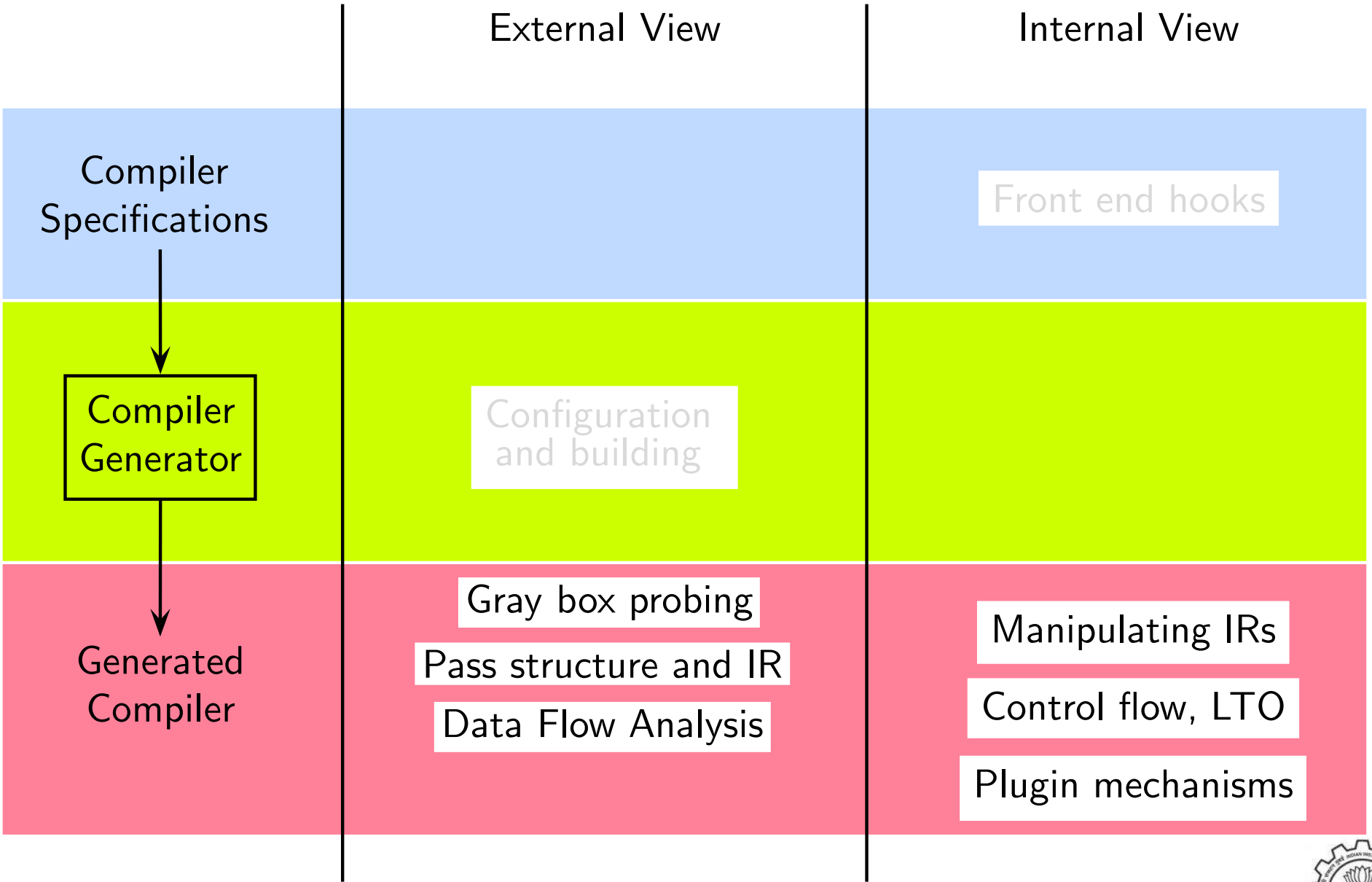
The Grand Picture and Our Coverage



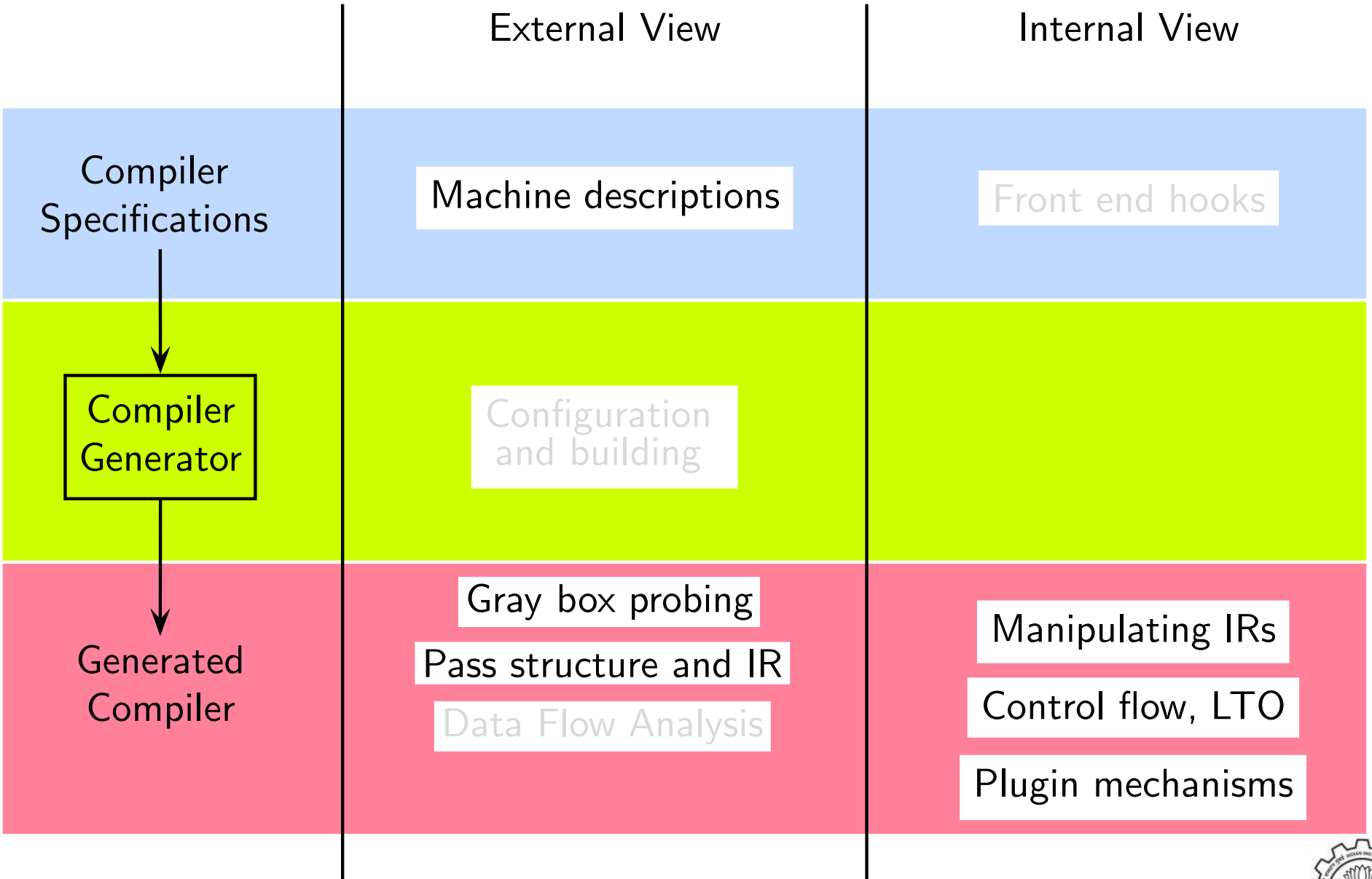
The Grand Picture and Our Coverage



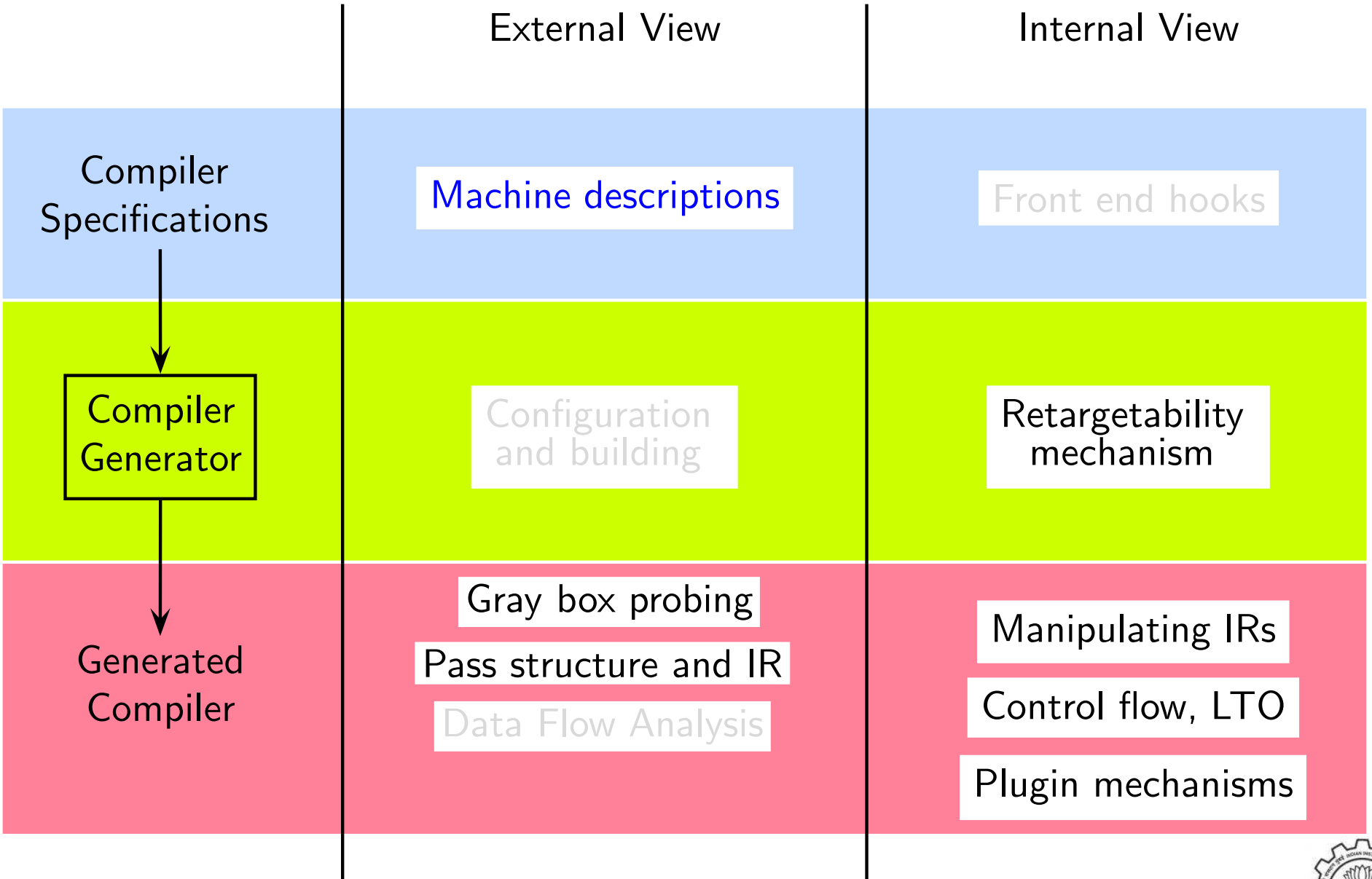
The Grand Picture and Our Coverage



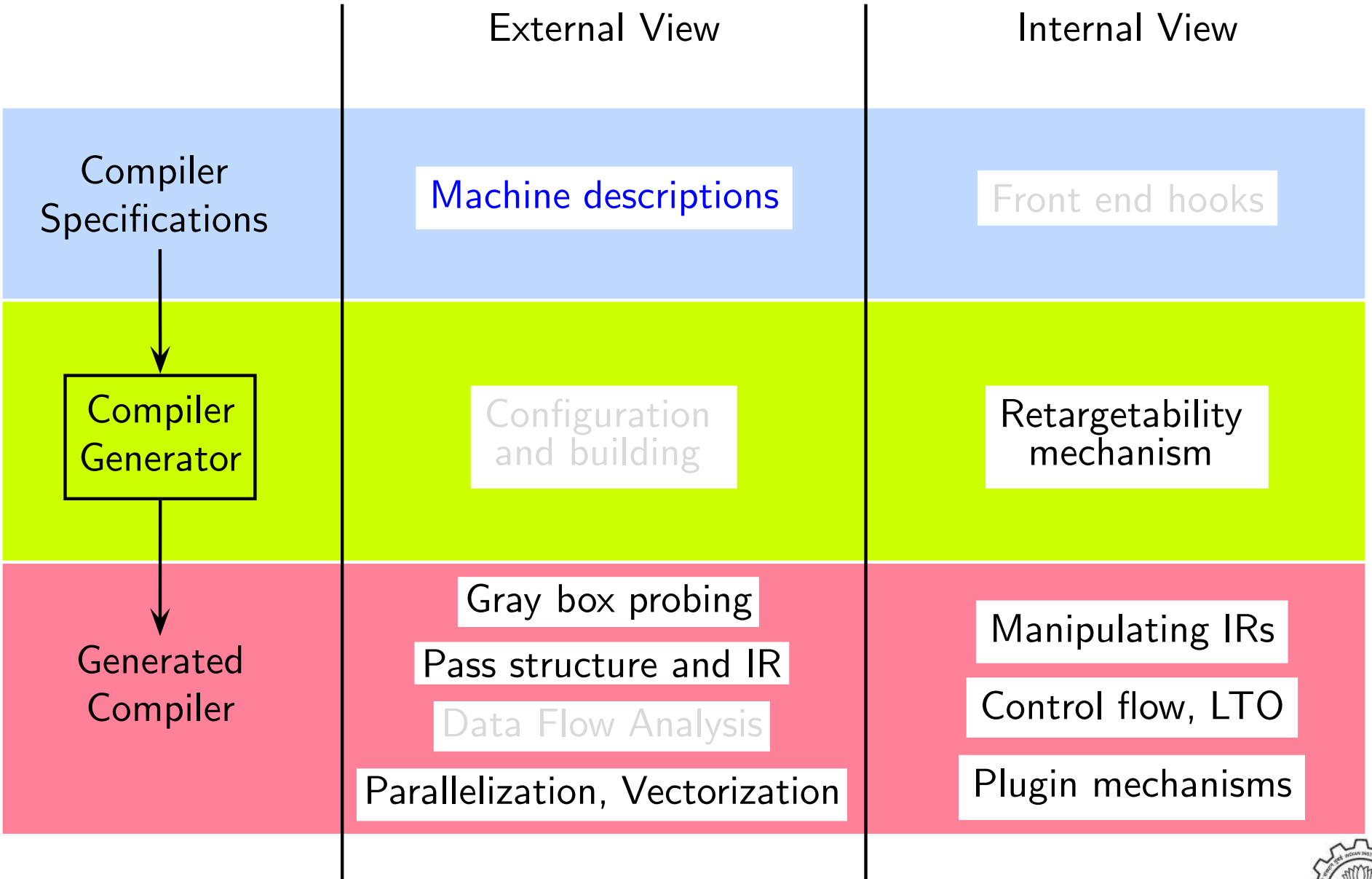
The Grand Picture and Our Coverage



The Grand Picture and Our Coverage



The Grand Picture and Our Coverage



The Grand Picture and Our Coverage