The Design and Implementation of Gnu Compiler Collection

Uday Khedker

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

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

July 2008

Part 1

Outline

Outline

- Open Source Vs. Free Software
- An Introduction to Compilation
- An overview of Gnu Compiler Collection
- Our work in GCC
- Conclusions

Part 2

Open Source Vs. Free Software

Open Source Vs. Free Software

- The open source initiative: (http://www.opensource.org/)
  Emphasis on development methodology
- The Free Software Foundation: (http://www.fsf.org/)
  Emphasis on freedom of the user
- In some cases, open source software has restricted user freedom

Open Source Vs. Free Software

- The open source initiative: (http://www.opensource.org/)
  Emphasis on development methodology
- The Free Software Foundation: (http://www.fsf.org/)
  Emphasis on freedom of the user
- In some cases, open source software has restricted user freedom
  - May cooperate with restrictive systems
  - eg. DRM (Digital Rights Management)

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, let users fix bugs

The Bazaar Approach

Release early, release often, let users fix bugs

- Brooks' law (The Mythical Man Month, 1975)

The Bazaar Approach

Release early, release often, let users fix bugs

- Brooks' law (The Mythical Man Month, 1975)
  - 12 man month effort

The Bazaar Approach

Release early, release often, let users fix bugs

- Brooks' law (The Mythical Man Month, 1975)
  - 12 man month effort
  - 1 person working for 12 months
    OR
    12 persons working for 1 month?

The Bazaar Approach

Release early, release often, let users fix bugs

- Brooks' law (The Mythical Man Month, 1975)
  - 12 man month effort
  - 1 person working for 12 months
    OR
    12 persons working for 1 month?
- Bazaar approach believes that the two somewhat equivalent in internet-based distributed development.

The Bazaar Approach

Release early, release often, let users fix bugs

- Brooks' law (The Mythical Man Month, 1975)
  - 12 man month effort
  - 1 person working for 12 months
    OR
    12 persons working for 1 month?
- Bazaar approach believes that the two somewhat equivalent in internet-based distributed development.
- "Given enough eyeballs, all bugs are shallow".

  Code errors, logical errors, and architectural errors.

The Bazaar Approach

Release early, release often, let users fix bugs

- Brooks' law (The Mythical Man Month, 1975)
  - 12 man month effort
  - 1 person working for 12 months
    OR
    12 persons working for 1 month?
- Bazaar approach believes that the two somewhat equivalent in internet-based distributed development.
- "Given enough eyeballs, all bugs are shallow".

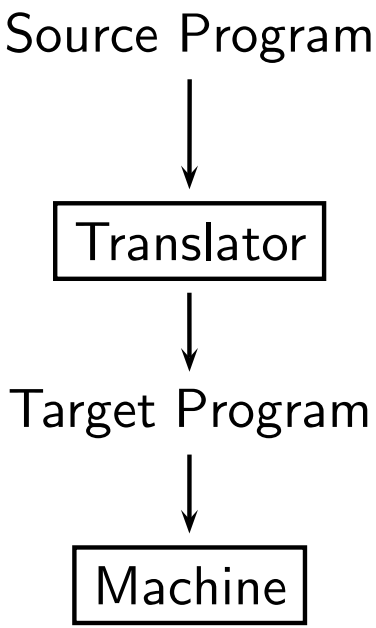
  Code errors, logical errors, and architectural errors.

A combination of the two seems more sensible

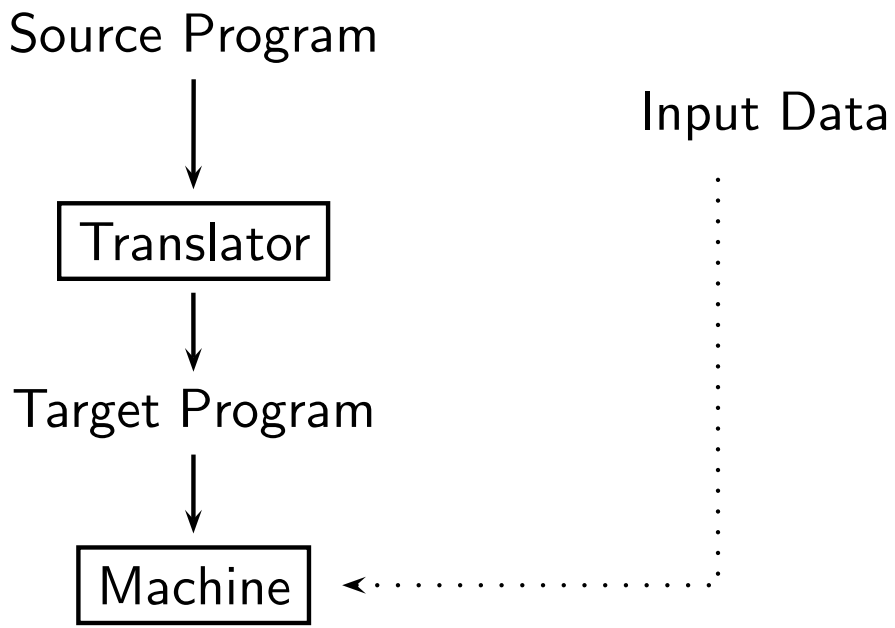
Part 3

Introduction to Compilation

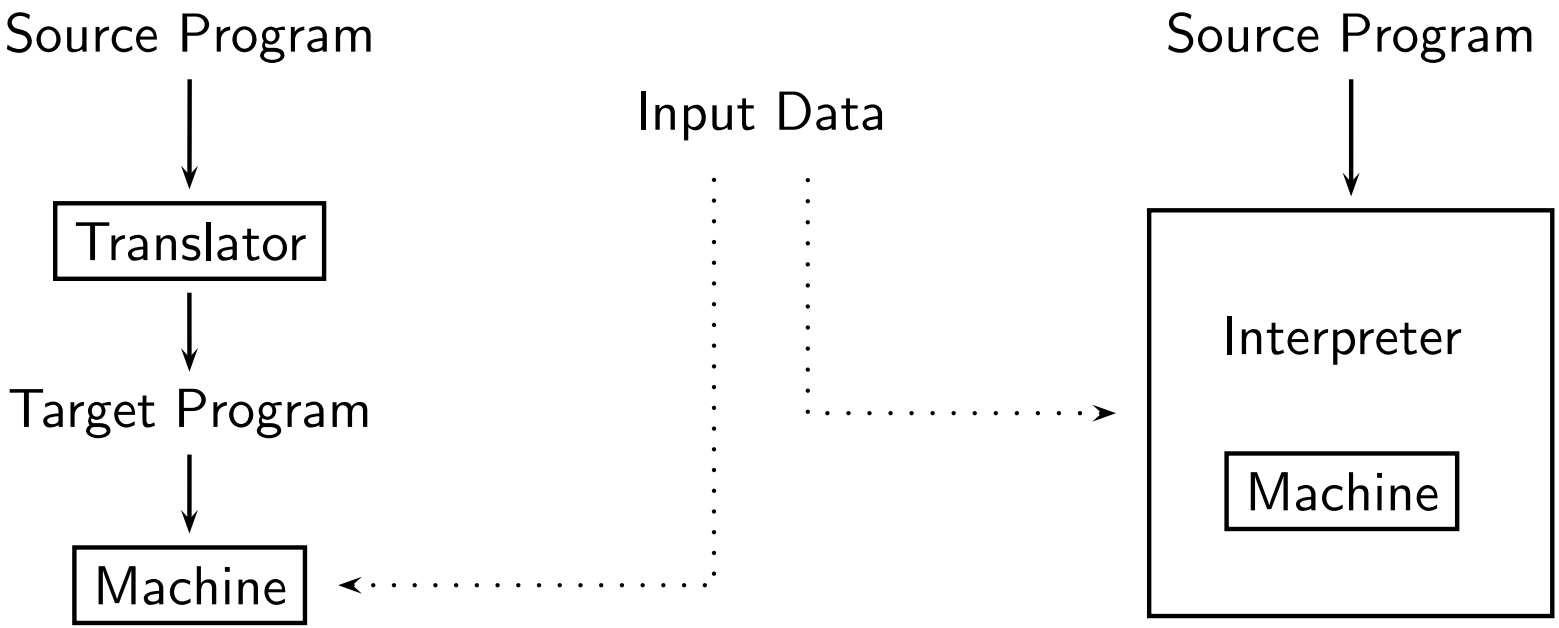
Implementation Mechanisms



Implementation Mechanisms

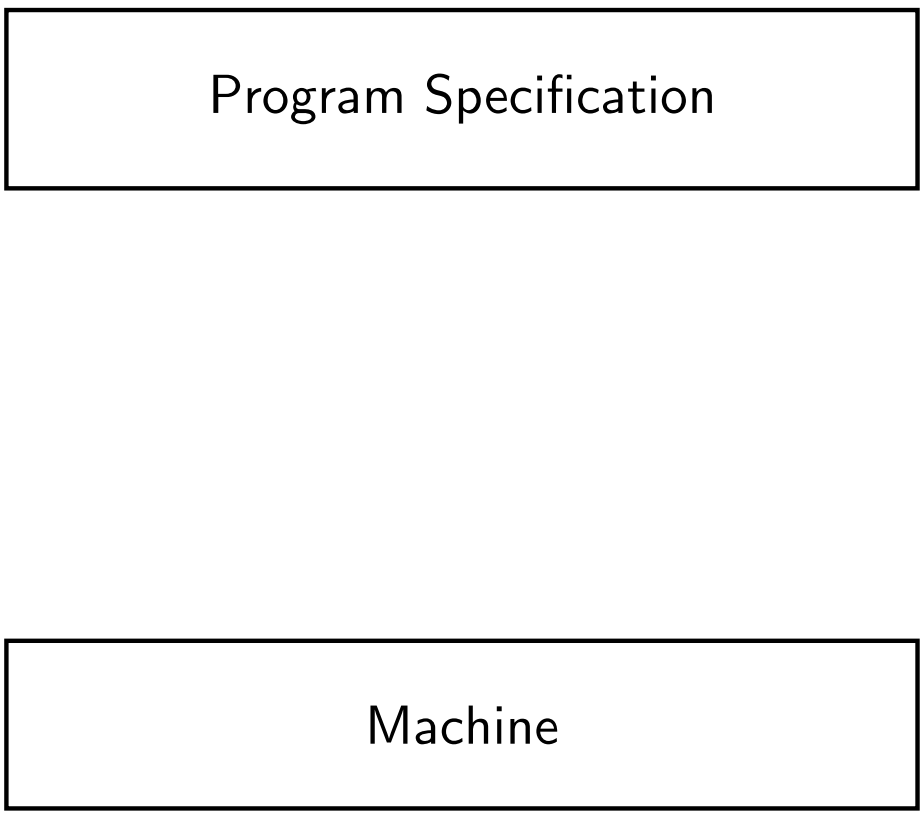


Implementation Mechanisms



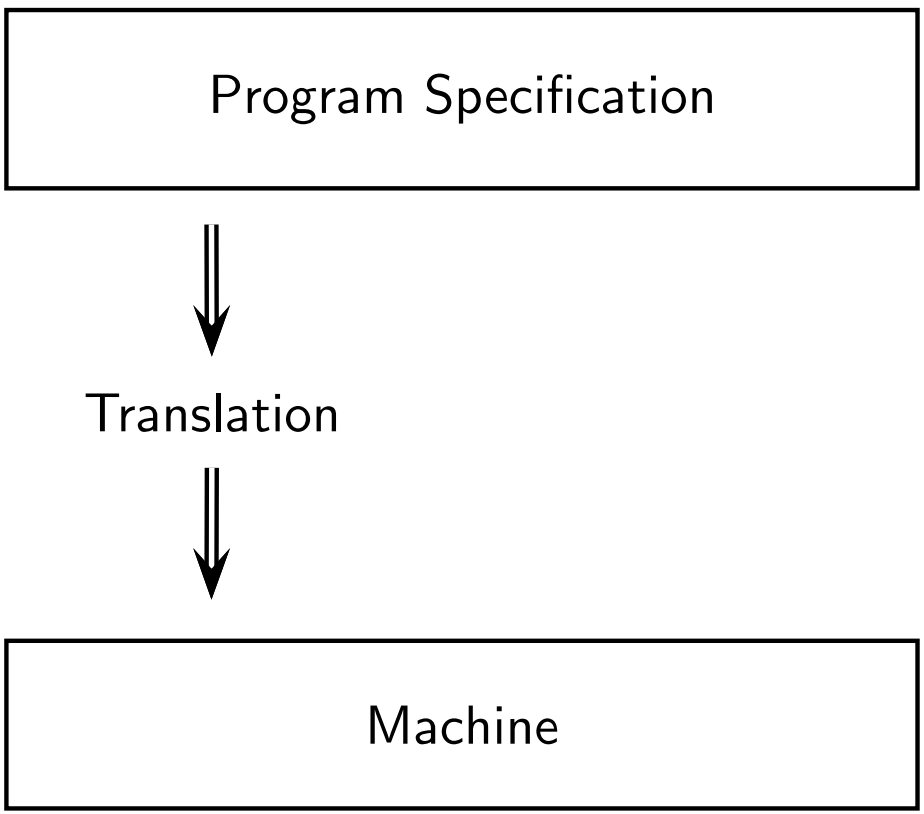
Implementation Mechanisms as "Bridges"

- "Gap" between the "levels" of program specification and execution



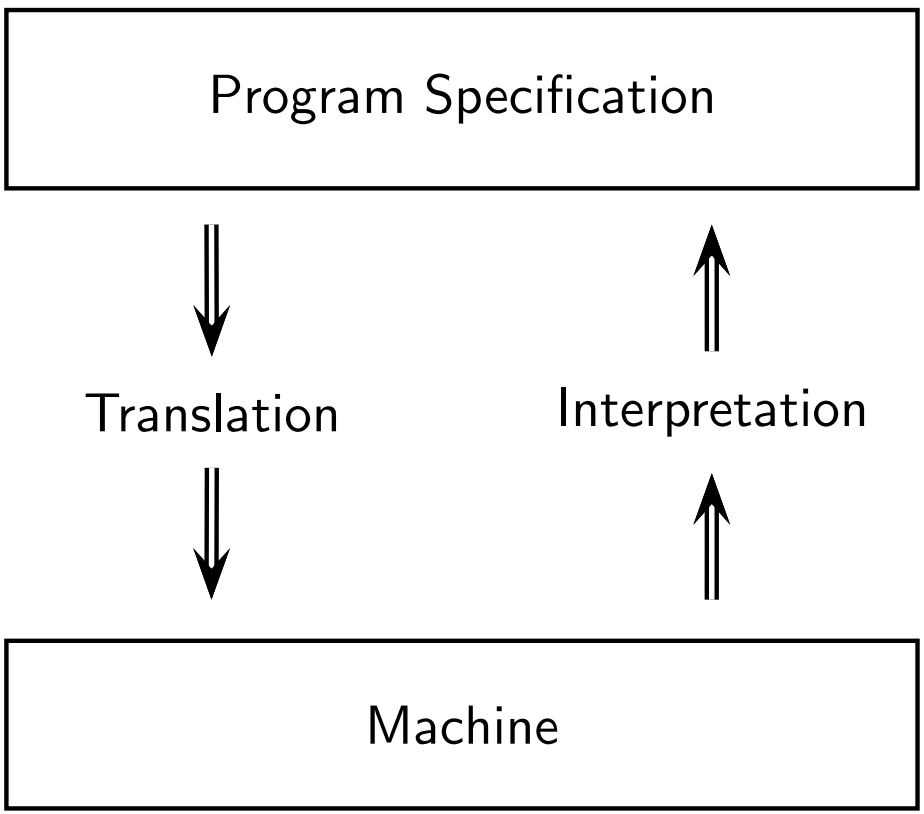
Implementation Mechanisms as "Bridges"

- "Gap" between the "levels" of program specification and execution



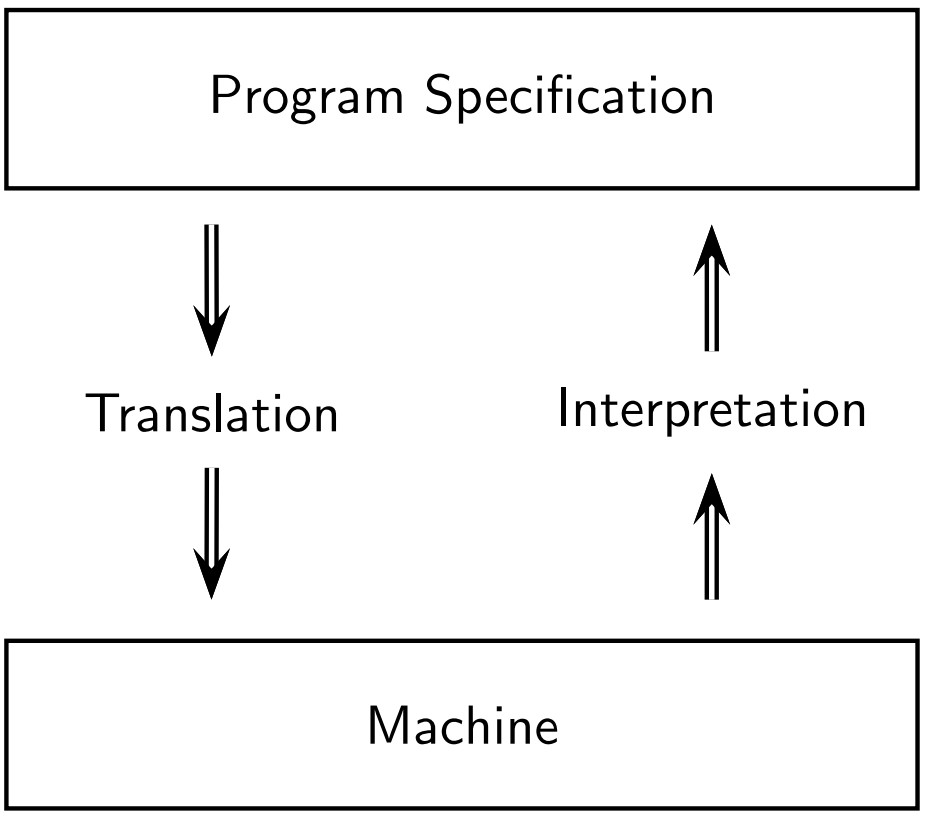
Implementation Mechanisms as "Bridges"

- "Gap" between the "levels" of program specification and execution



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

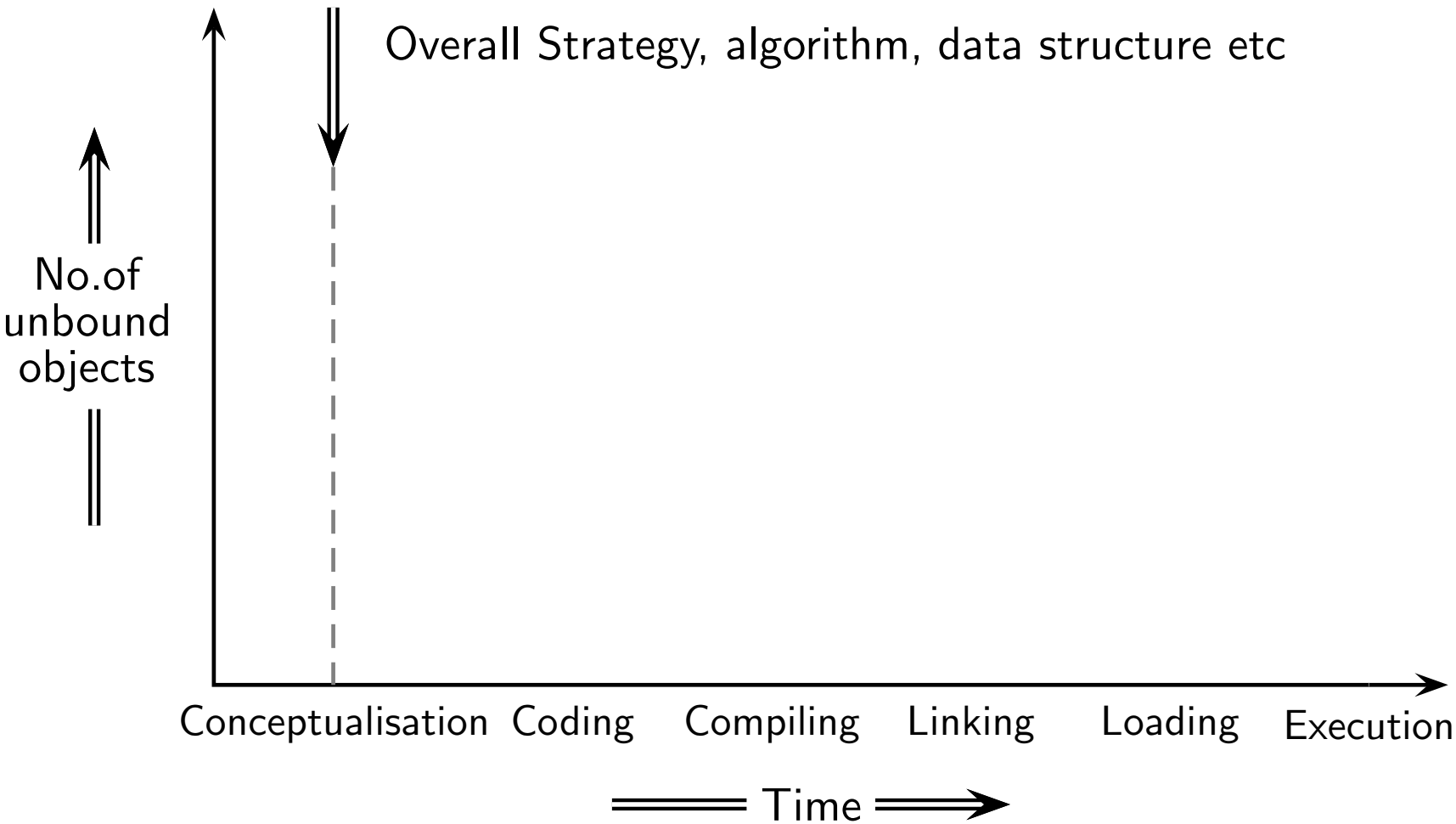
- A source statement

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

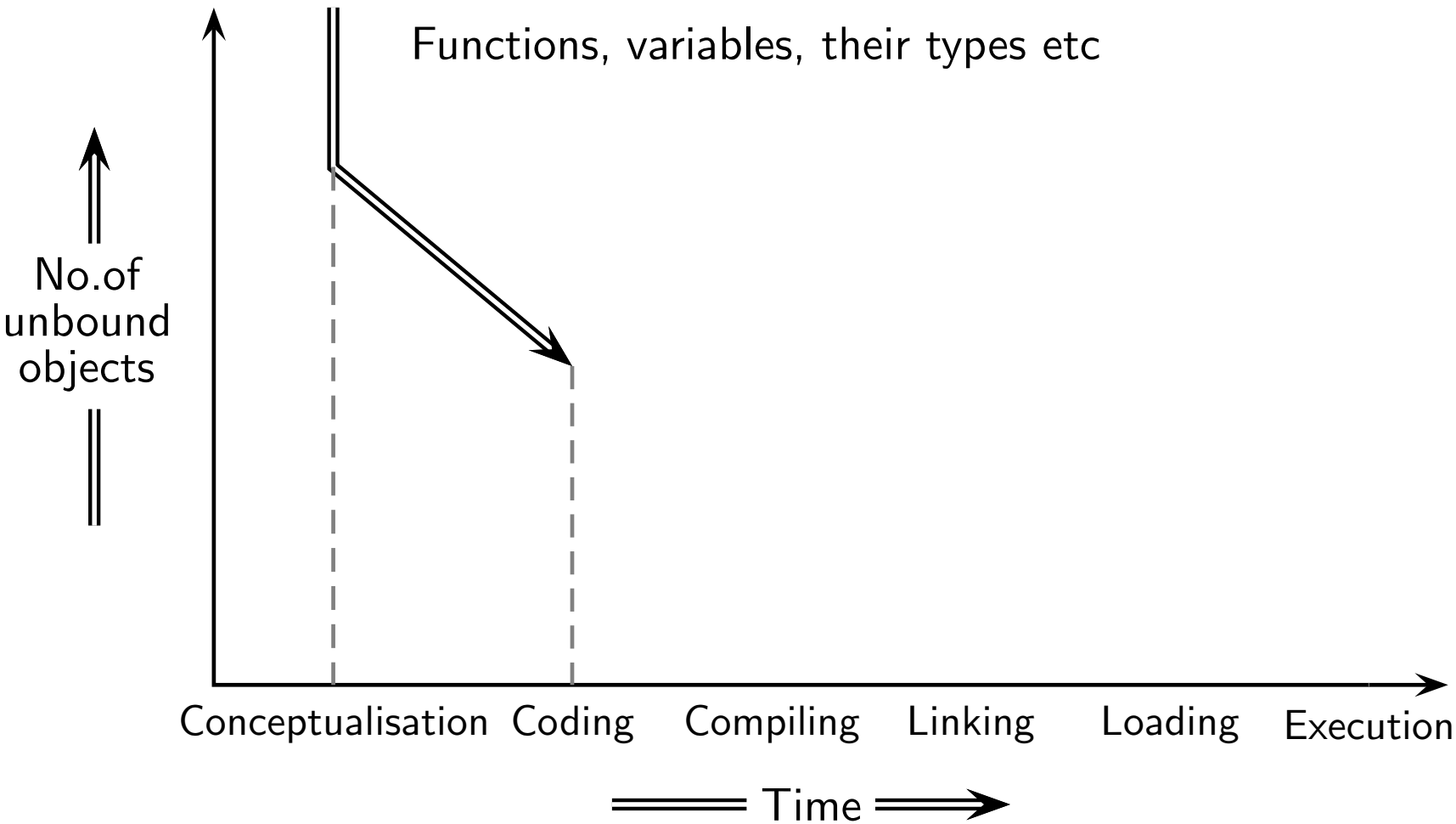
- Spim assembly equivalent

```
        lw   $t0, 4($fp)    #       $t0 <- b
        slti $t0, $t0, 10   #       $t0 <- $t0 < b
        not  $t0, $t0       #       $t0 <- ! $t0
        bgtz $t0, L0:       #       if $t0 >=  0 goto L0:
        lw   $t0, 4($fp)    #       $t0 <- b
        b    L1:            #       goto L1:
L0:     lw   $t0, 8($fp)    # L0:   $t0 <- c
L1:     sw   0($fp), $t0    # L1:   a <- $t0
```

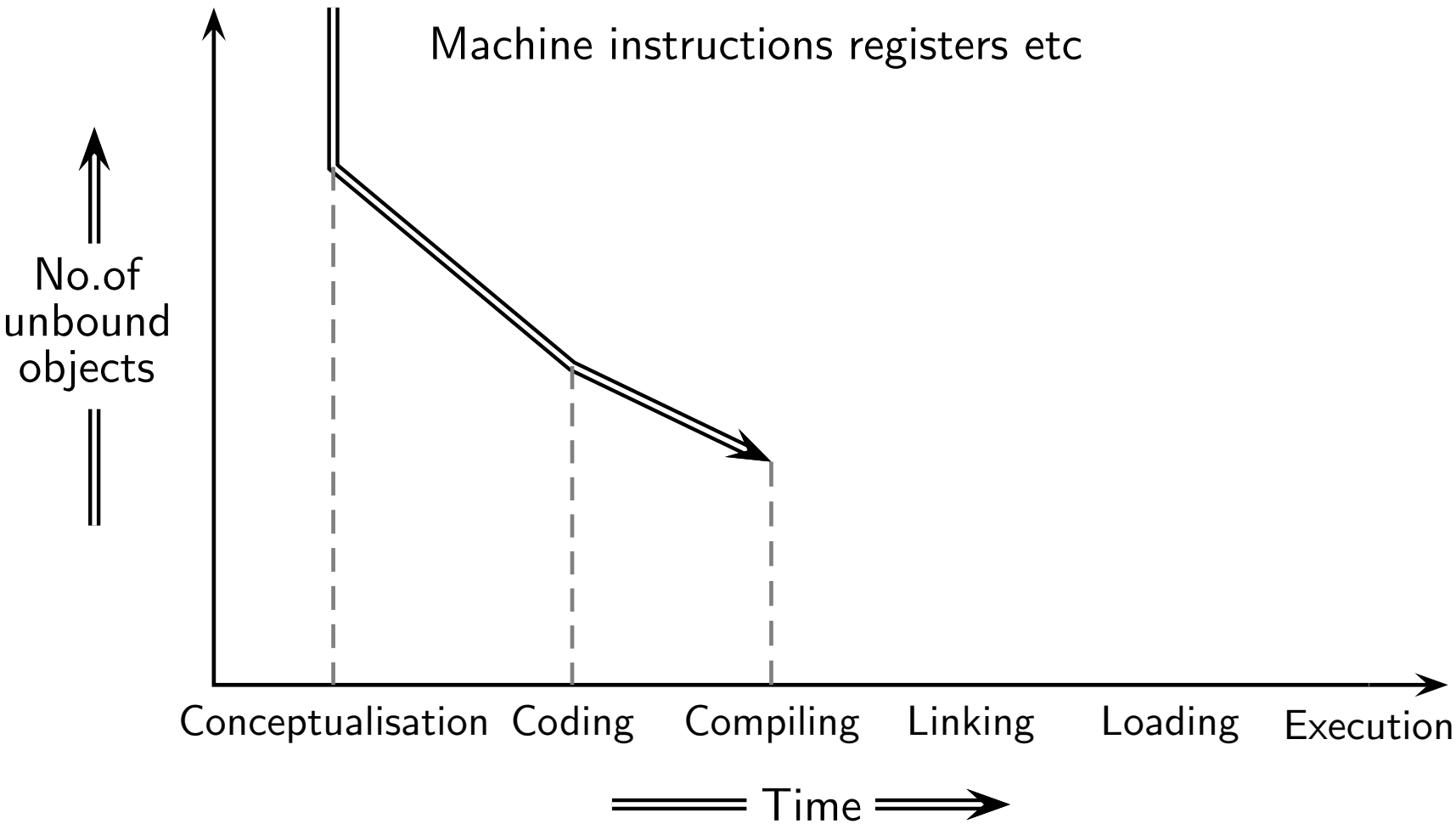

Binding


Overall Strategy, algorithm, data structure etc
No.of unbound objects
Conceptualisation
Coding
Compiling
Linking
Loading
Execution
Time

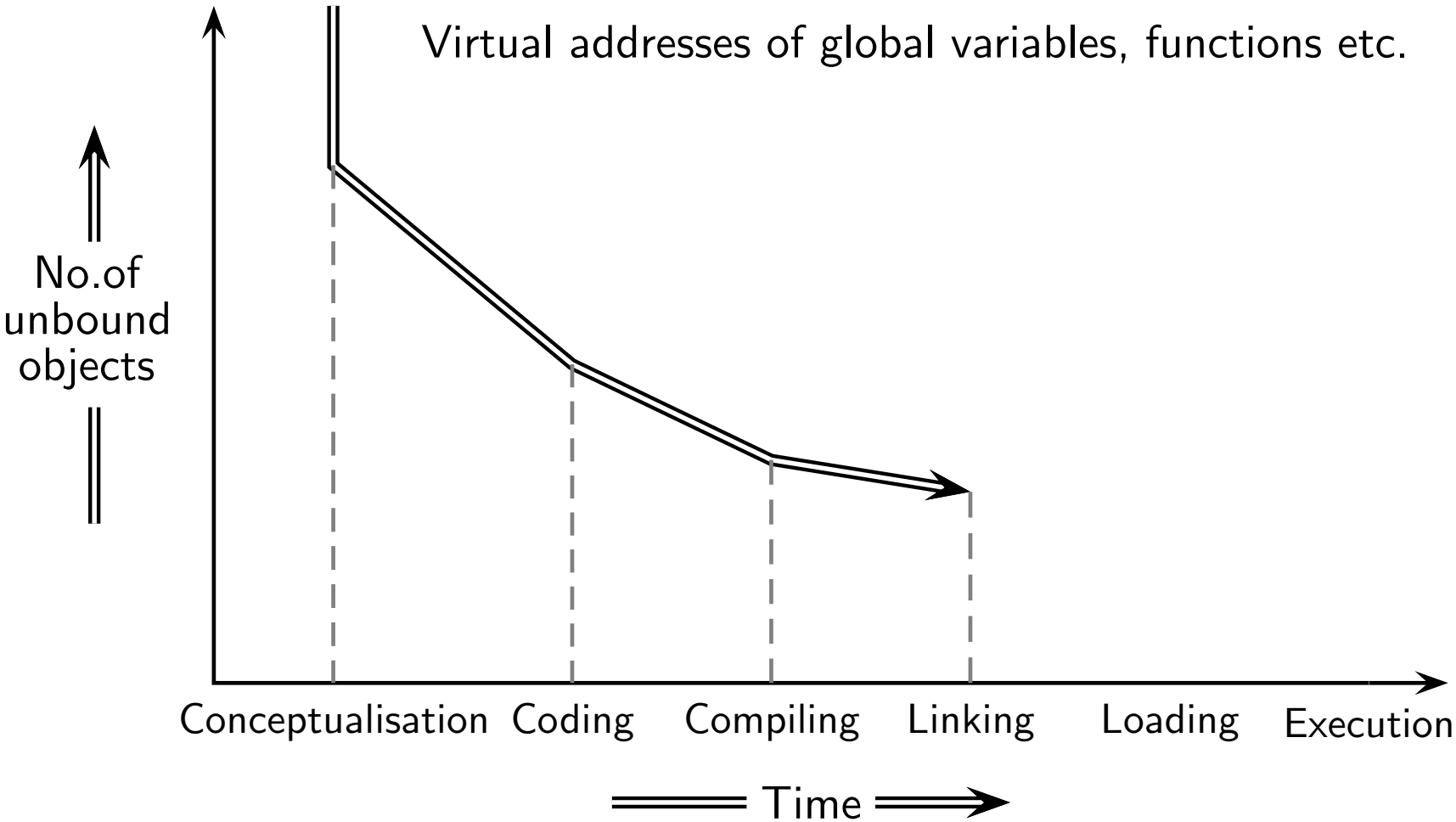
Binding


Functions, variables, their types etc
No.of unbound objects
Conceptualisation
Coding
Compiling
Linking
Loading
Execution
Time

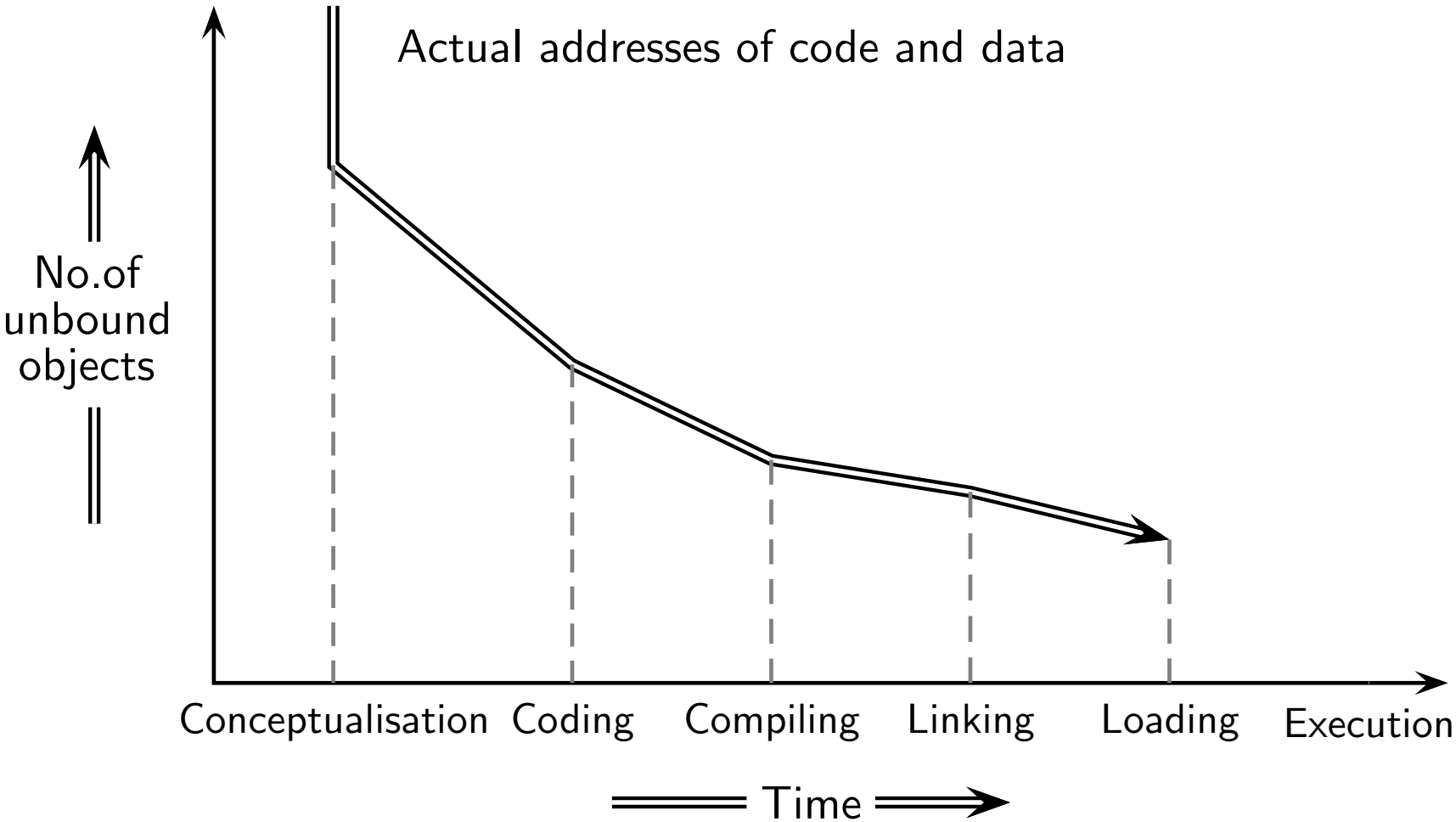
Binding


Machine instructions registers etc
No.of unbound objects
Conceptualisation
Coding
Compiling
Linking
Loading
Execution
Time

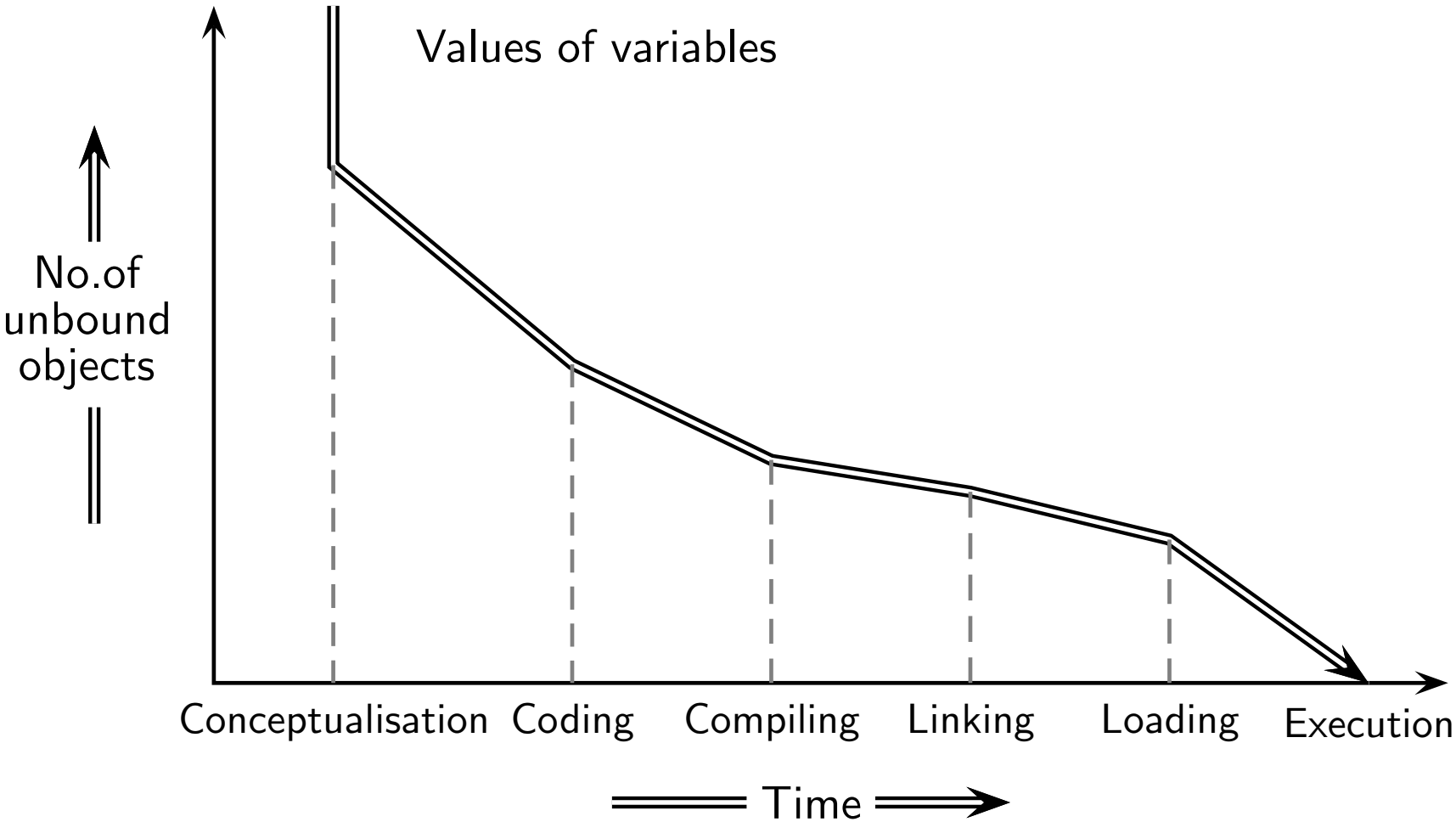
Binding


Virtual addresses of global variables, functions etc.
No.of unbound objects
Conceptualisation
Coding
Compiling
Linking
Loading
Execution
Time

Binding


Actual addresses of code and data
No.of unbound objects
Conceptualisation
Coding
Compiling
Linking
Loading
Execution
Time

Binding


Values of variables
No.of unbound objects
Conceptualisation
Coding
Compiling
Linking
Loading
Execution
Time

Implementation Mechanisms

- Translation = Analysis + Synthesis

  Interpretation = Analysis + Execution

Implementation Mechanisms

- Translation = Analysis + Synthesis

  Interpretation = Analysis + Execution

- Translation Instructions $\Longrightarrow$ Equivalent Instructions

Implementation Mechanisms

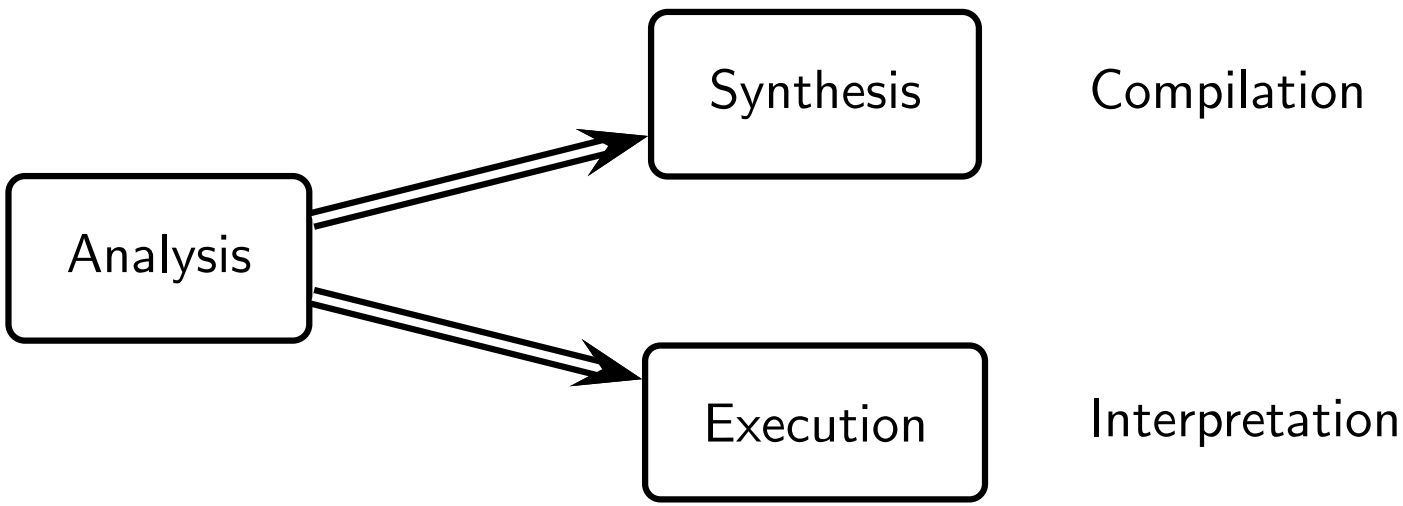
- Translation = Analysis + Synthesis

  Interpretation = Analysis + Execution

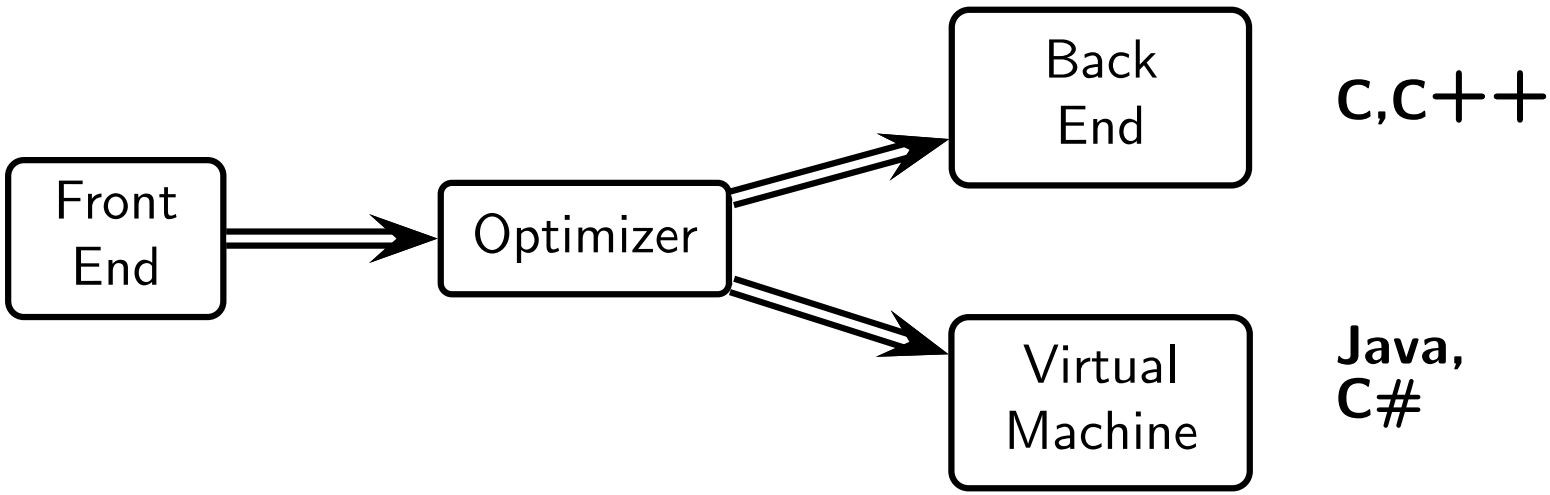
- Translation Instructions $\Longrightarrow$ Equivalent Instructions

  Interpretation Instructions $\Longrightarrow$ Actions Implied by Instructions

Language Implementation Models



Language Processor Models



Translation Sequence in Our Example: Parsing

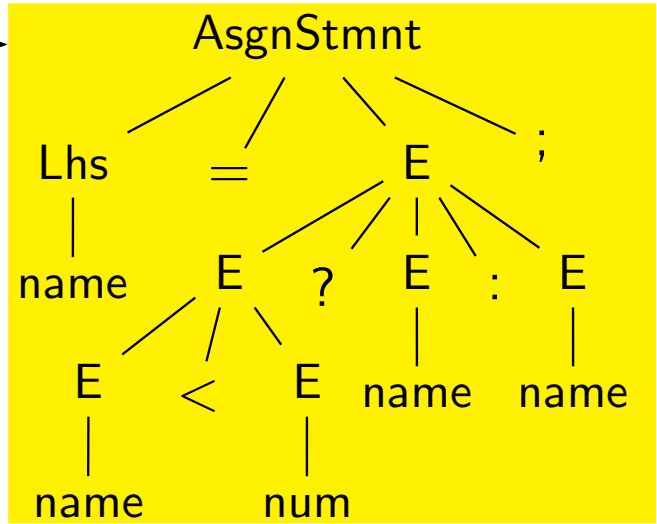
```
a=b<10?b:c;
```

Input

Translation Sequence in Our Example: Parsing

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

Input



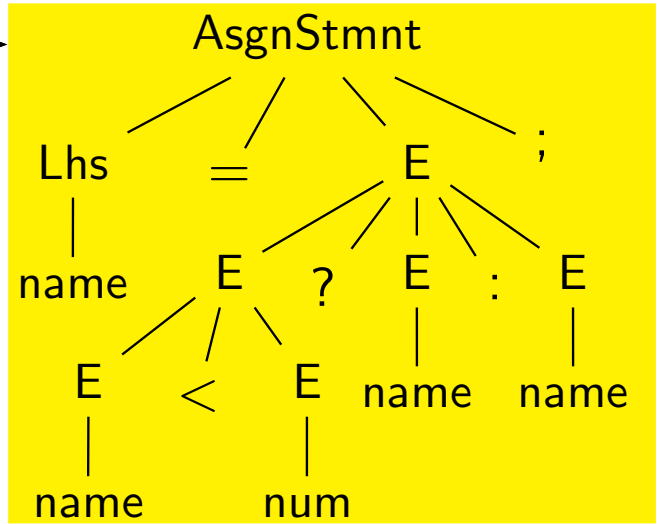
Parse Tree

Issues:

Translation Sequence in Our Example: Parsing

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

Input

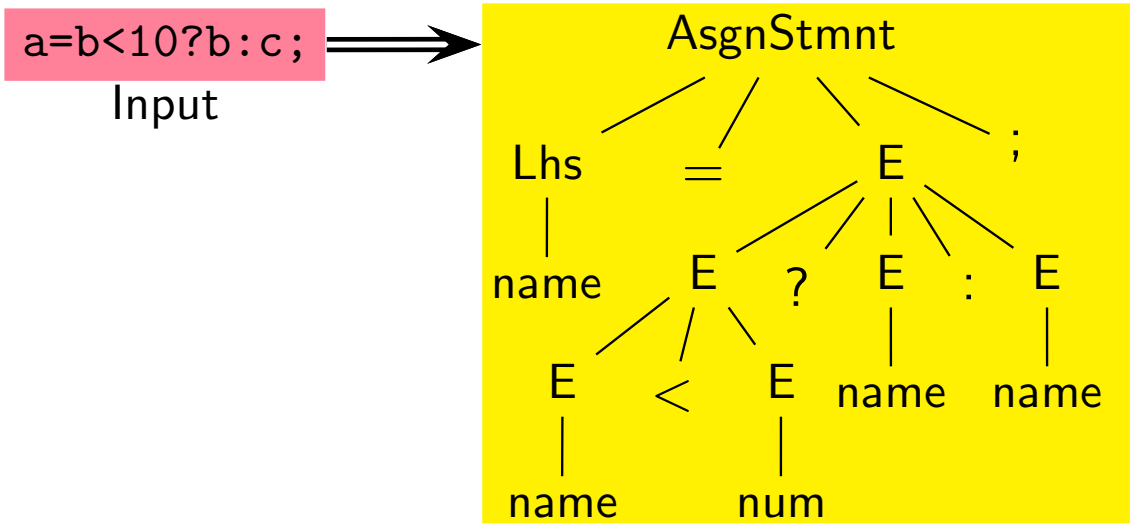


Parse Tree

Issues:

- Grammar rules, terminals, non-terminals

Translation Sequence in Our Example: Parsing



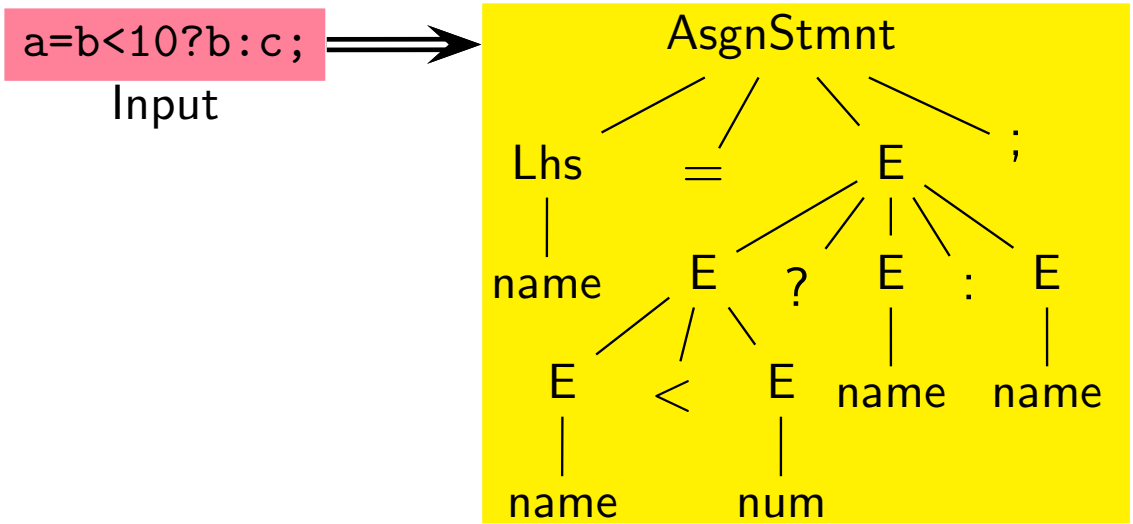
Parse Tree

Issues:

- Grammar rules, terminals, non-terminals
- Order of application of grammar rules

  eg. is it `(a = b<10?)` followed by `(b:c)`?

Translation Sequence in Our Example: Parsing



Parse Tree

Issues:

- Grammar rules, terminals, non-terminals
- Order of application of grammar rules

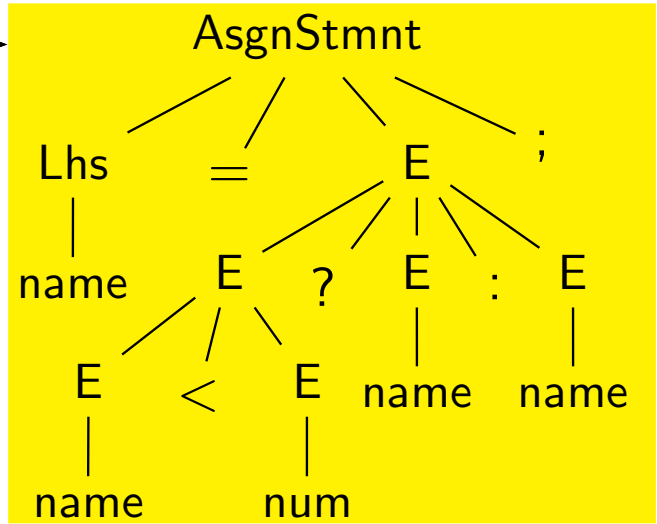
  eg. is it `(a = b<10?)` followed by `(b:c)`?
- Values of terminal symbols

  eg. string "10" vs. integer number 10.

Translation Sequence in Our Example: Semantic Analysis

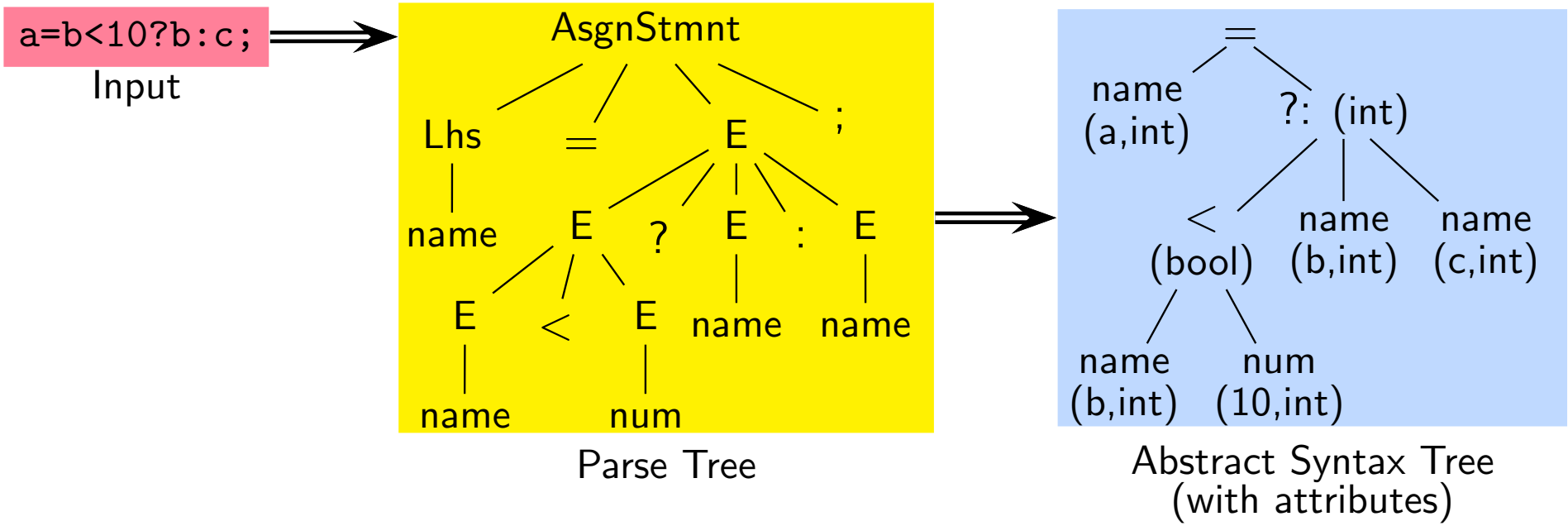
```
a=b<10?b:c;
```

Input



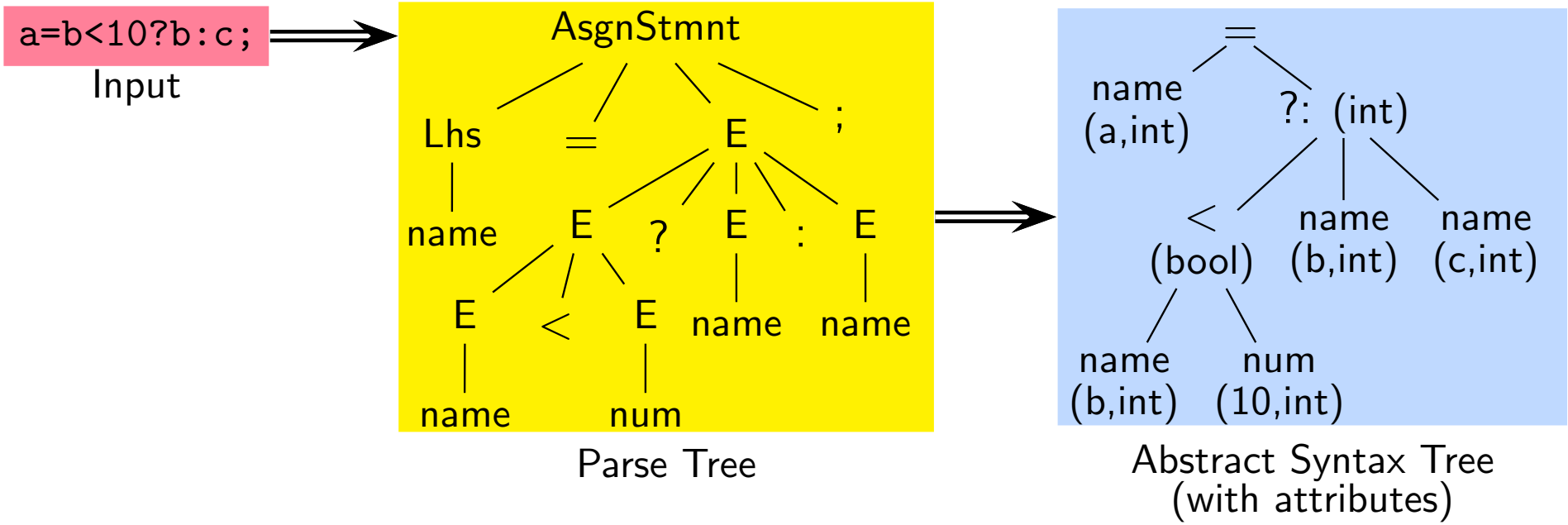
Parse Tree

Translation Sequence in Our Example: Semantic Analysis



Issues:

Translation Sequence in Our Example: Semantic Analysis

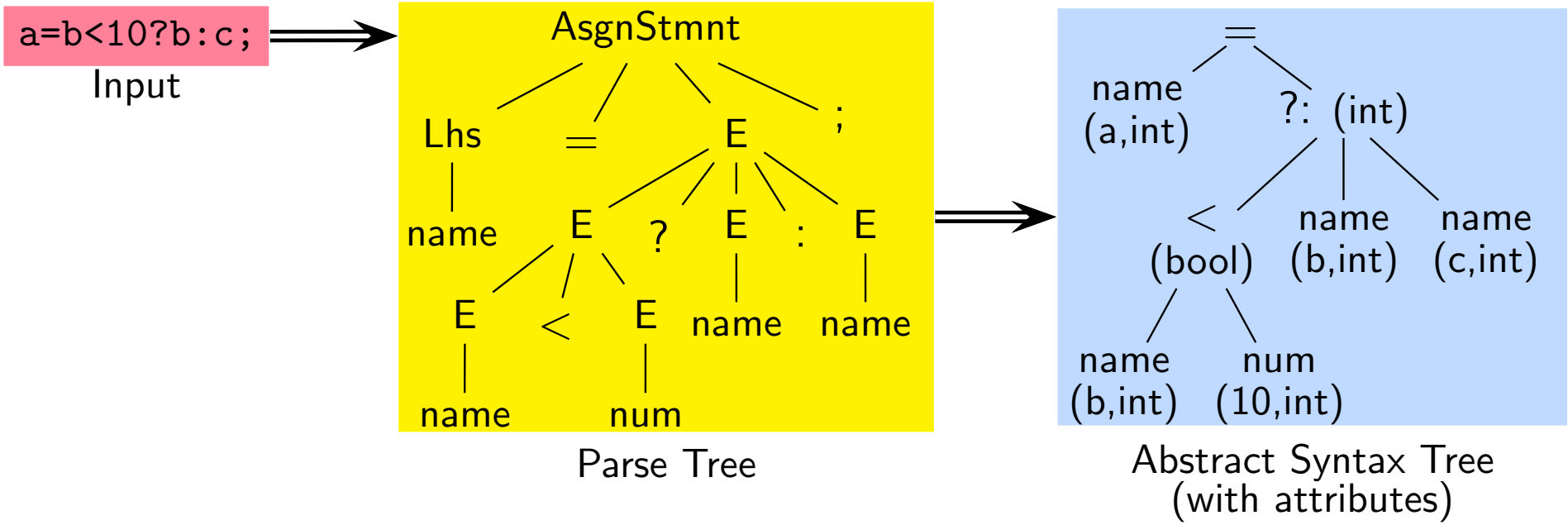


Issues:

- Symbol tables

  Have variables been declared? What are their types? What is their scope?

Translation Sequence in Our Example: Semantic Analysis



Issues:

- Symbol tables

  Have variables been declared? What are their types? What is their scope?

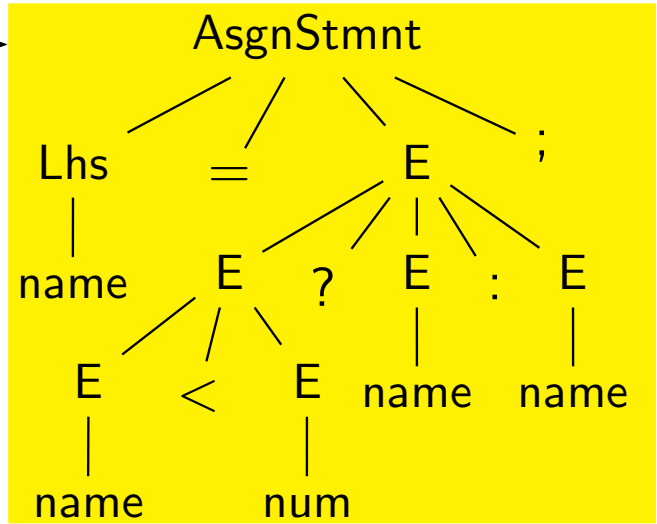
- Type consistency of operators and operands

  The result of computing b<10? is bool and not int

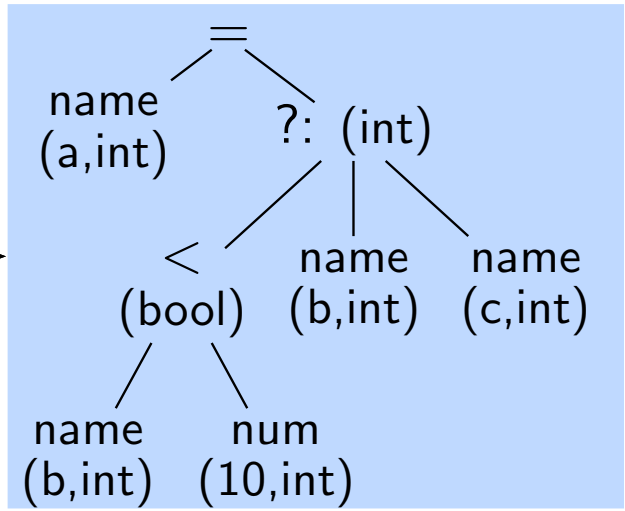
Translation Sequence in Our Example: IR Generation

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

Input

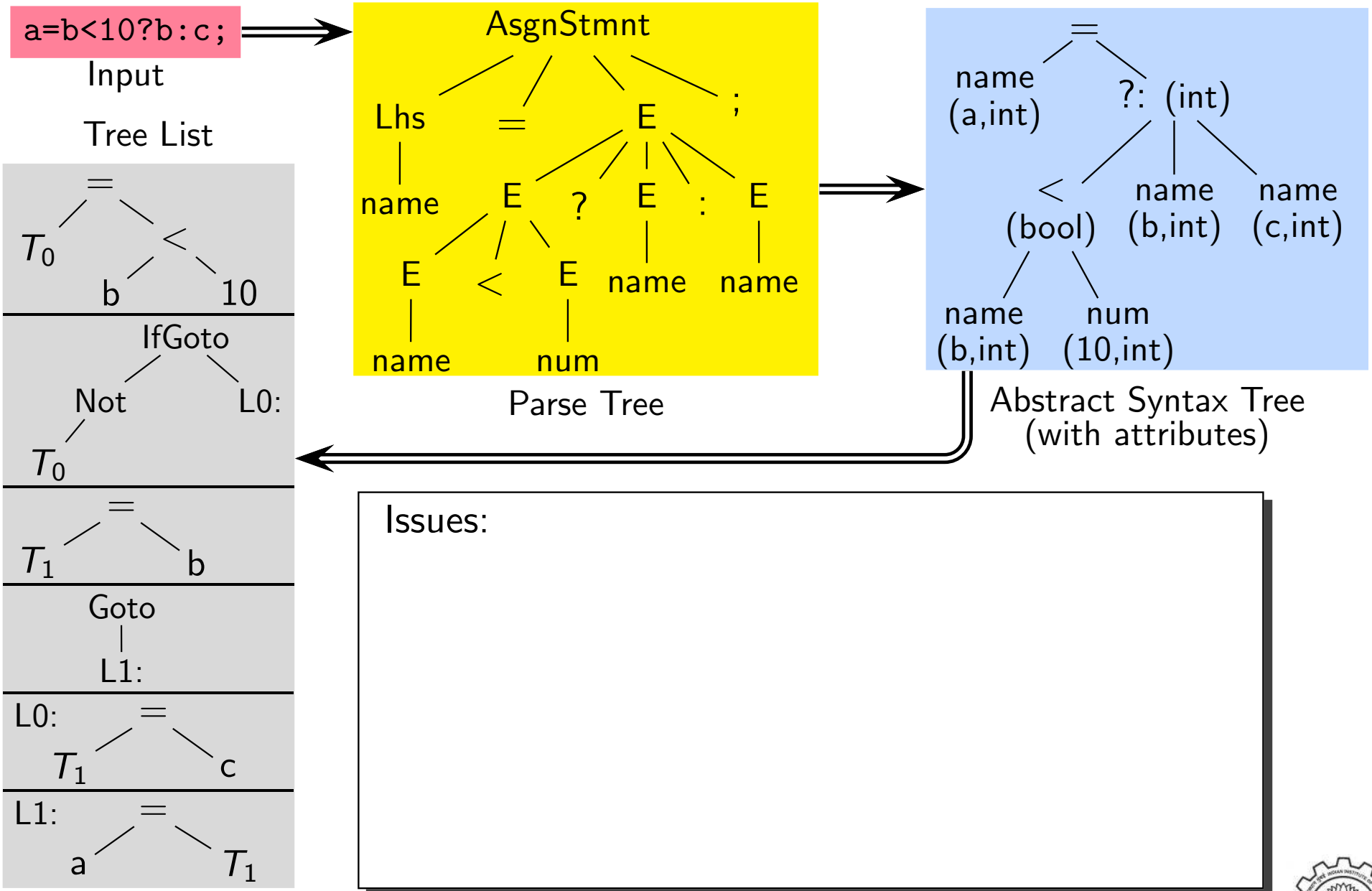


Parse Tree

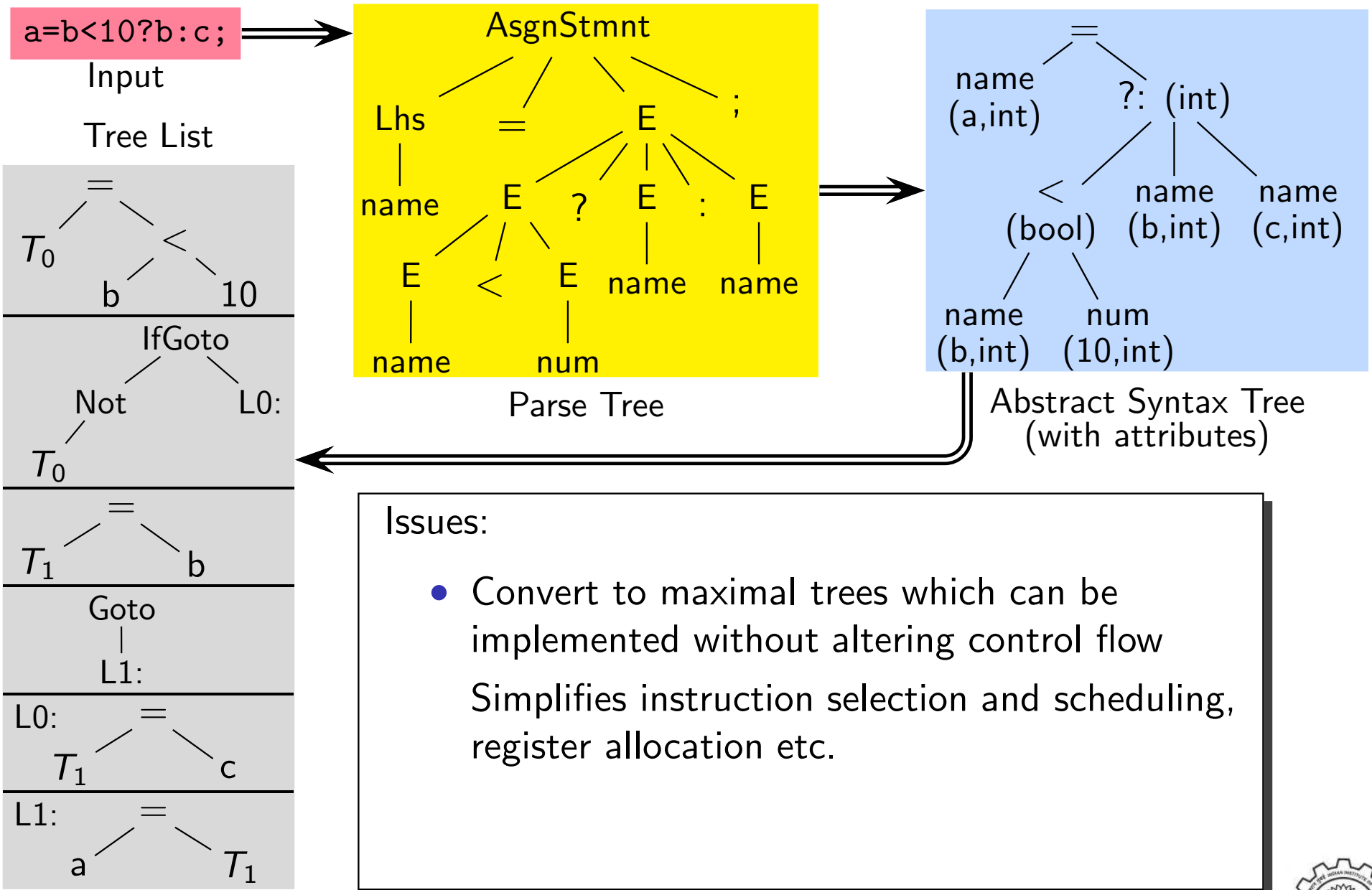


Abstract Syntax Tree
(with attributes)

Translation Sequence in Our Example: IR Generation



Translation Sequence in Our Example: IR Generation

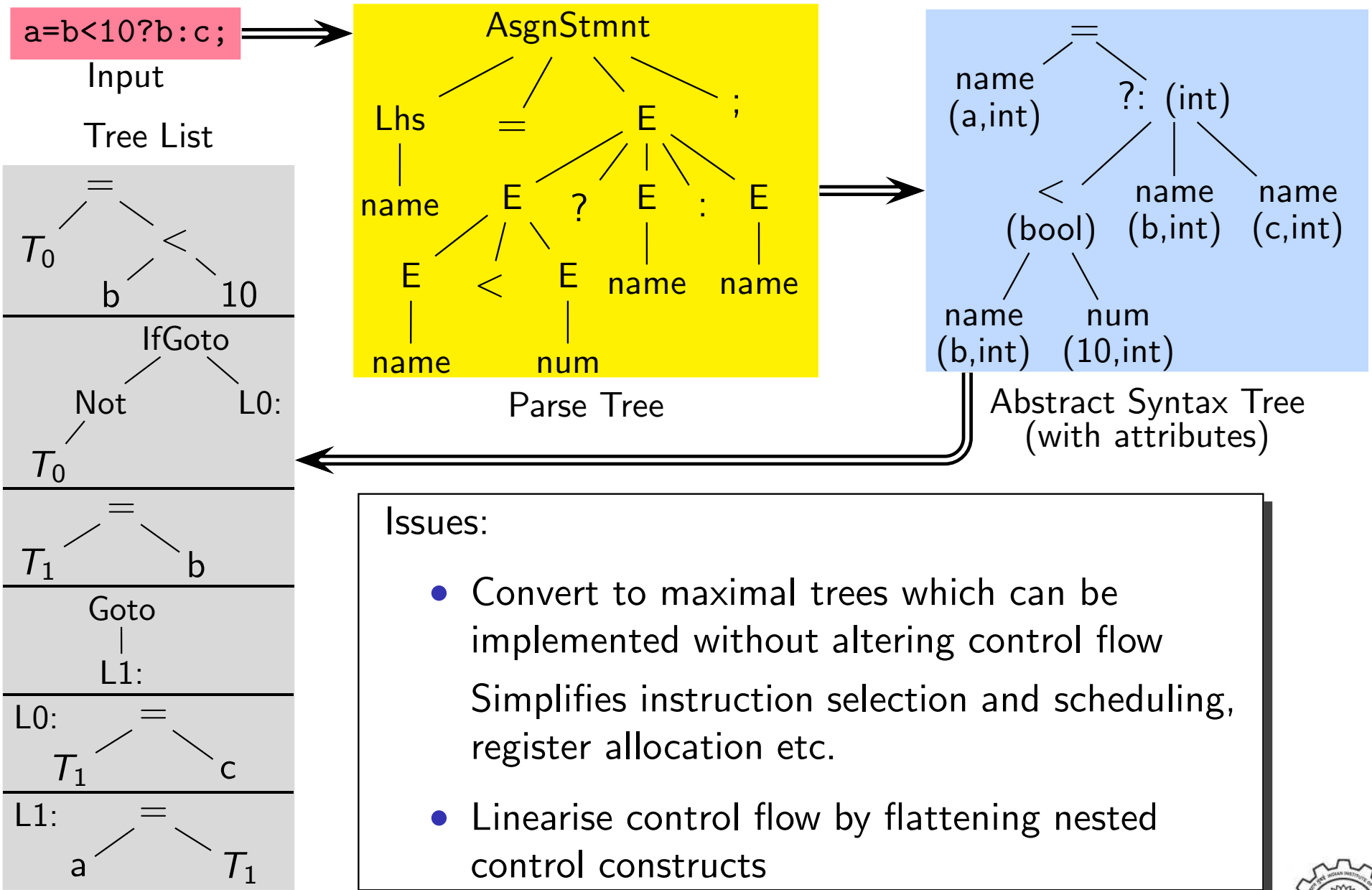


Issues:

- Convert to maximal trees which can be implemented without altering control flow

  Simplifies instruction selection and scheduling, register allocation etc.

Translation Sequence in Our Example: IR Generation



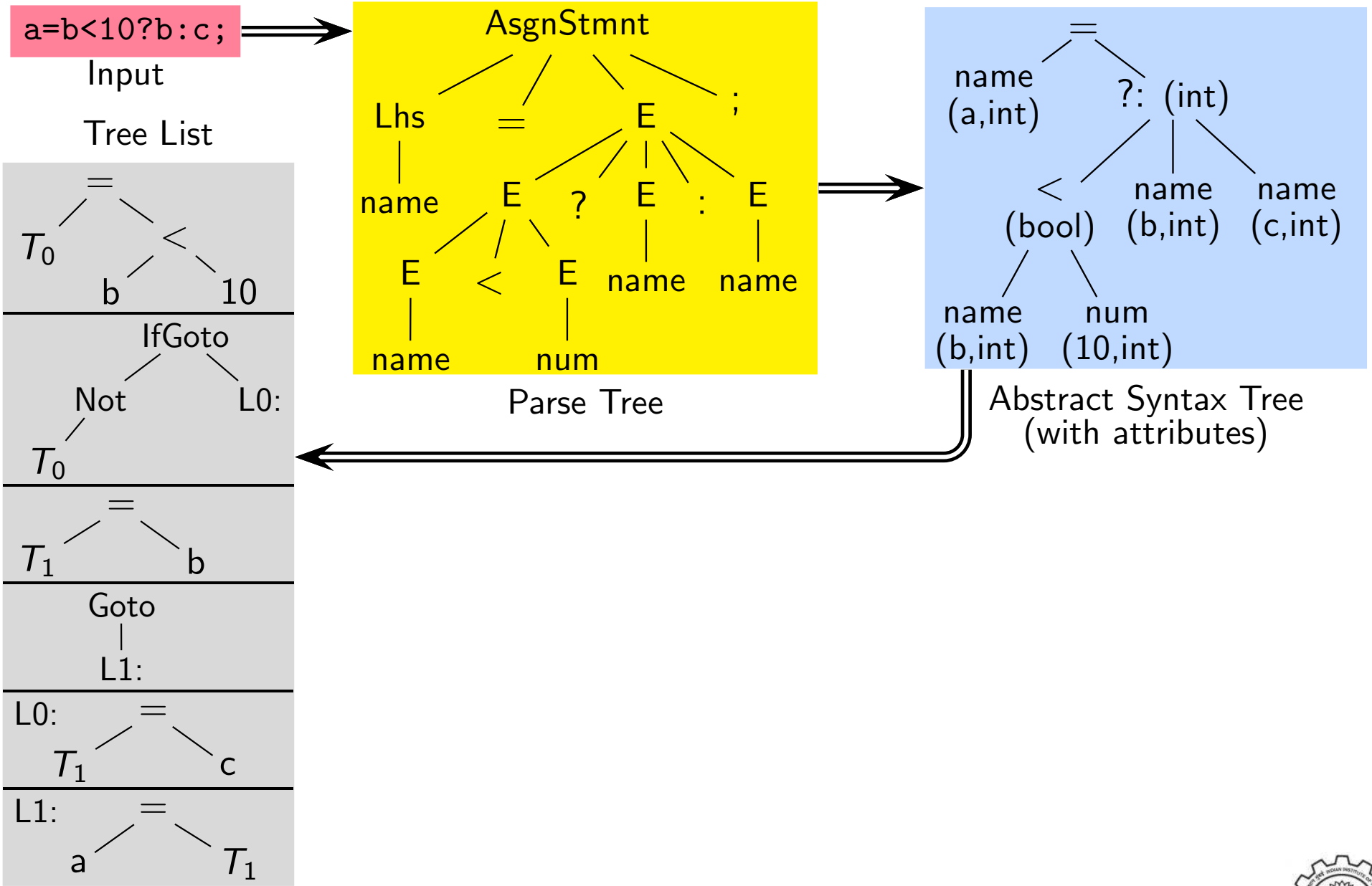
Issues:

- Convert to maximal trees which can be implemented without altering control flow

  Simplifies instruction selection and scheduling, register allocation etc.

- Linearise control flow by flattening nested control constructs

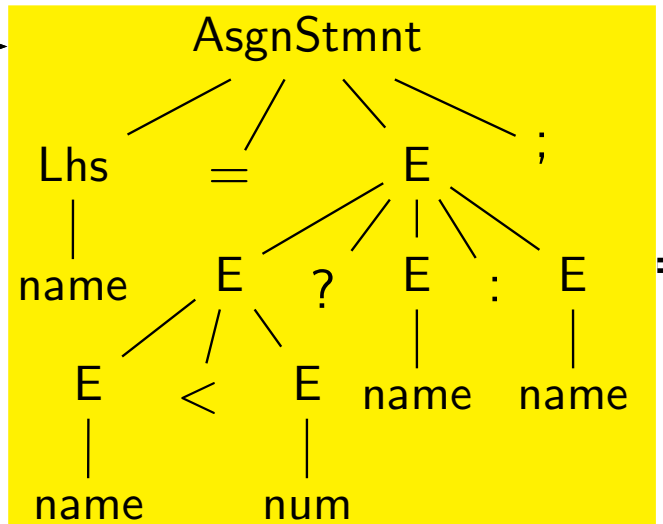
Translation Sequence in Our Example: Instruction Selection



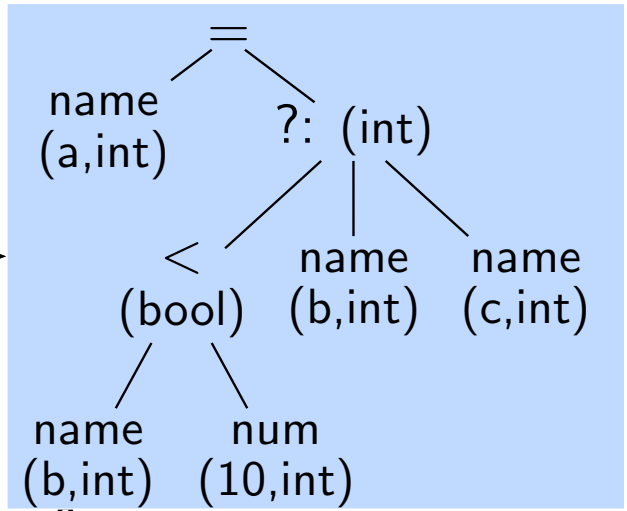
Translation Sequence in Our Example: Instruction Selection

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

Input

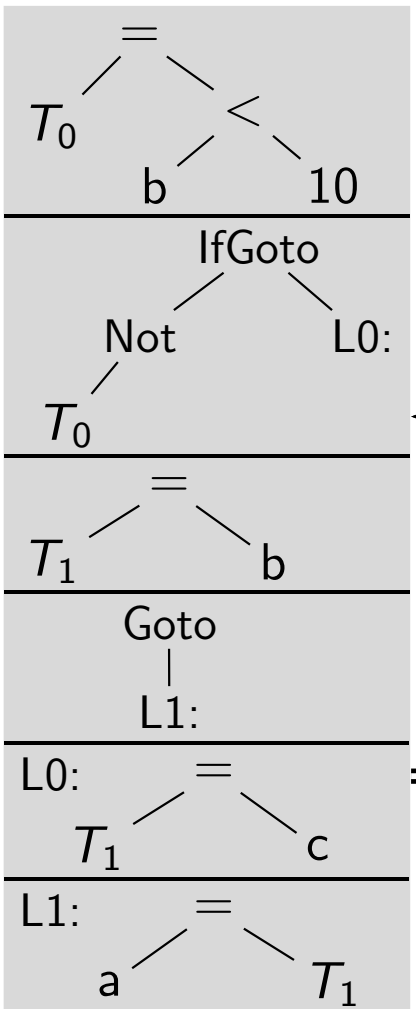


Parse Tree



Abstract Syntax Tree
(with attributes)

Tree List

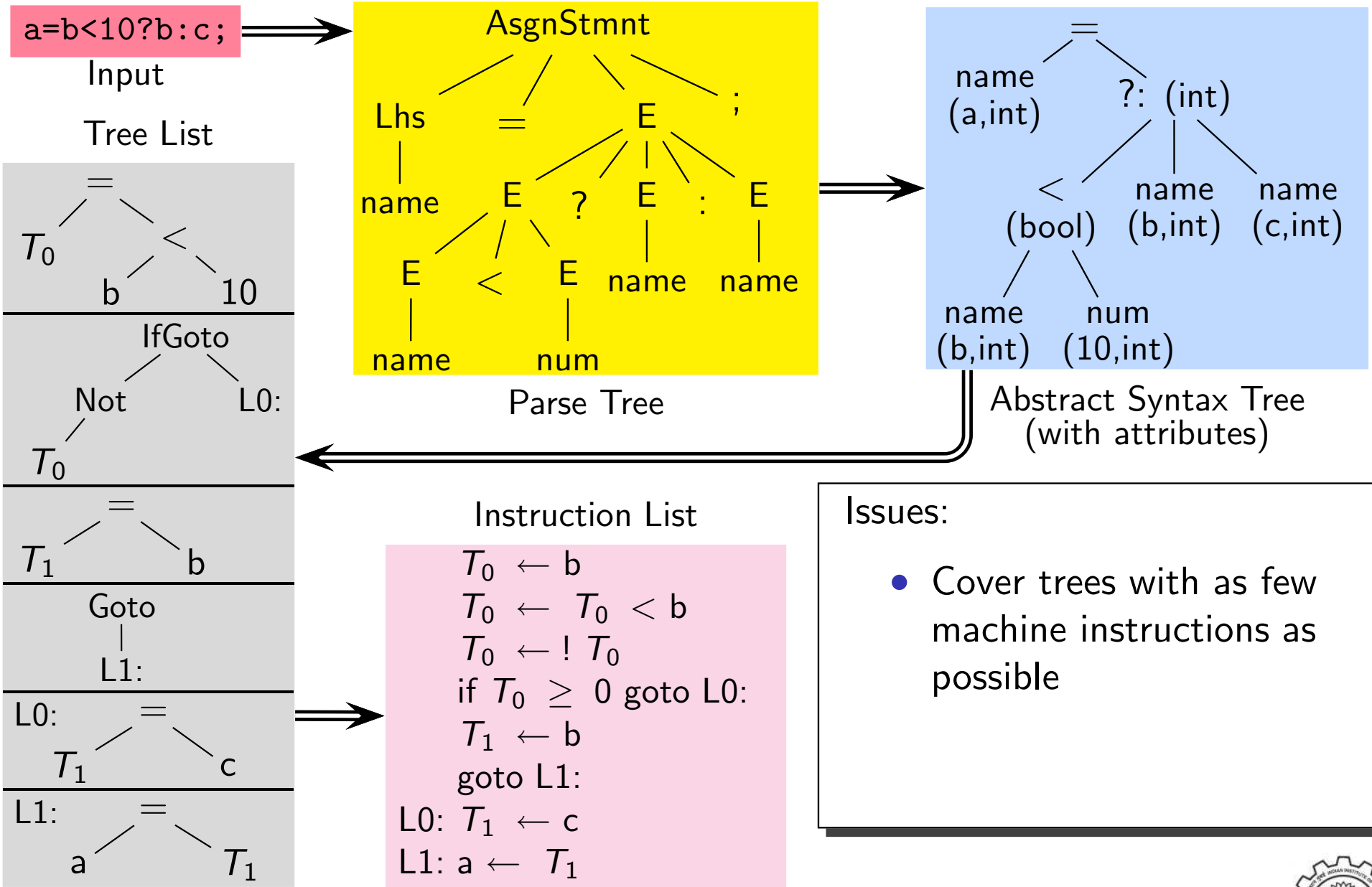


Instruction List

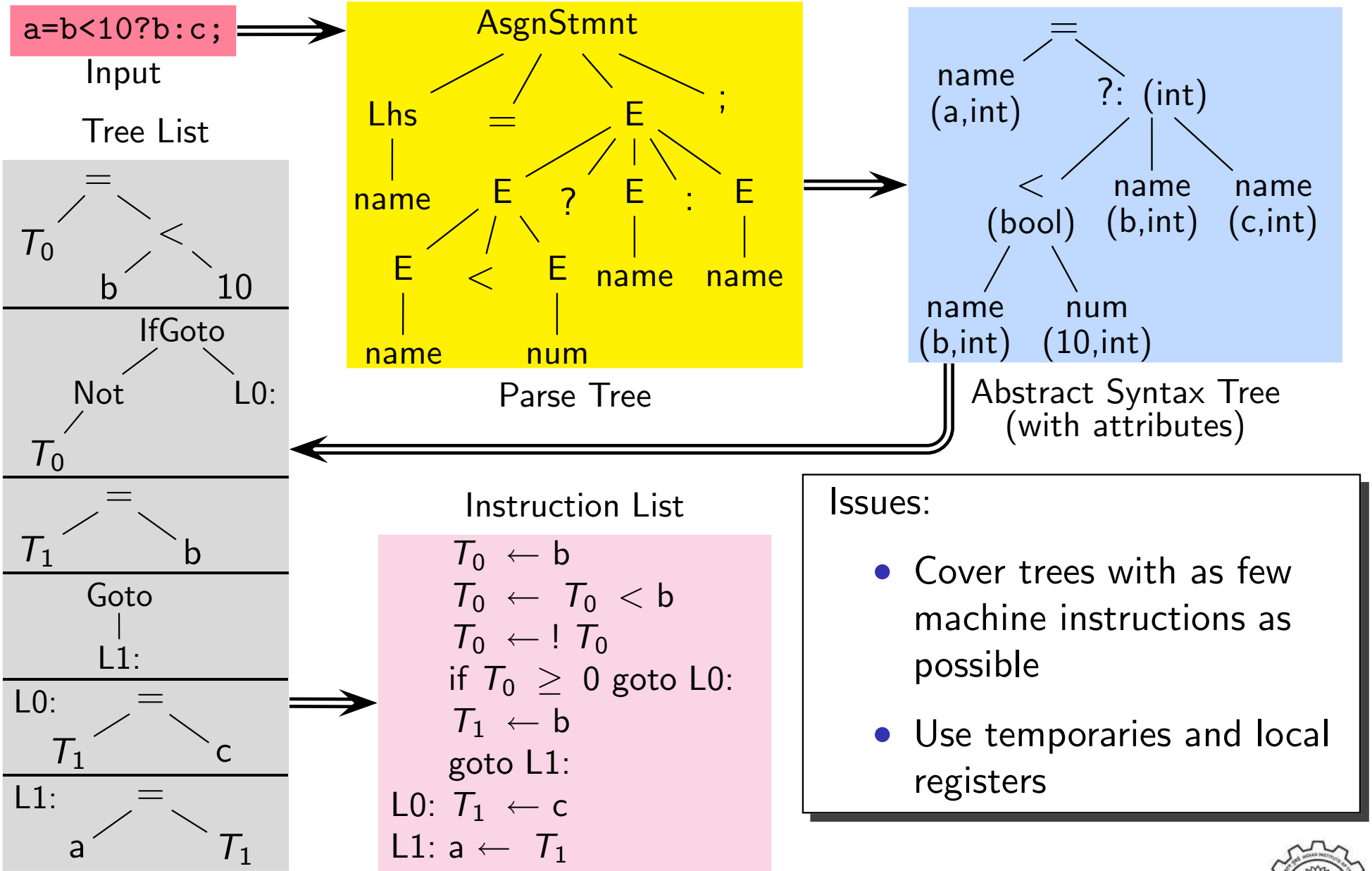
$T_0 \leftarrow$ b
$T_0 \leftarrow T_0 <$ b
$T_0 \leftarrow !\ T_0$
if $T_0 \geq 0$ goto L0:
$T_1 \leftarrow$ b
goto L1:
L0: $T_1 \leftarrow$ c
L1: a $\leftarrow T_1$

Issues:

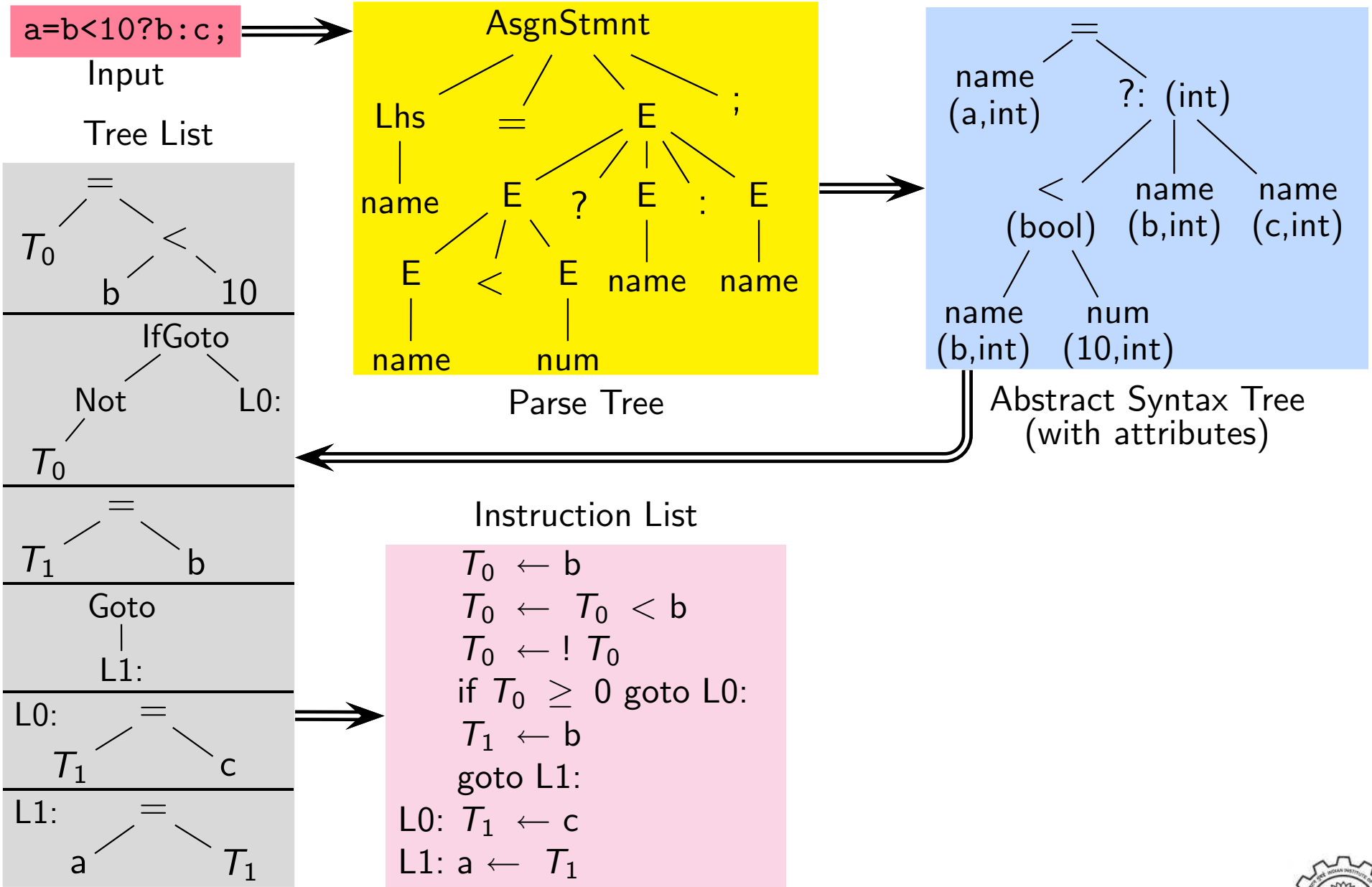
Translation Sequence in Our Example: Instruction Selection



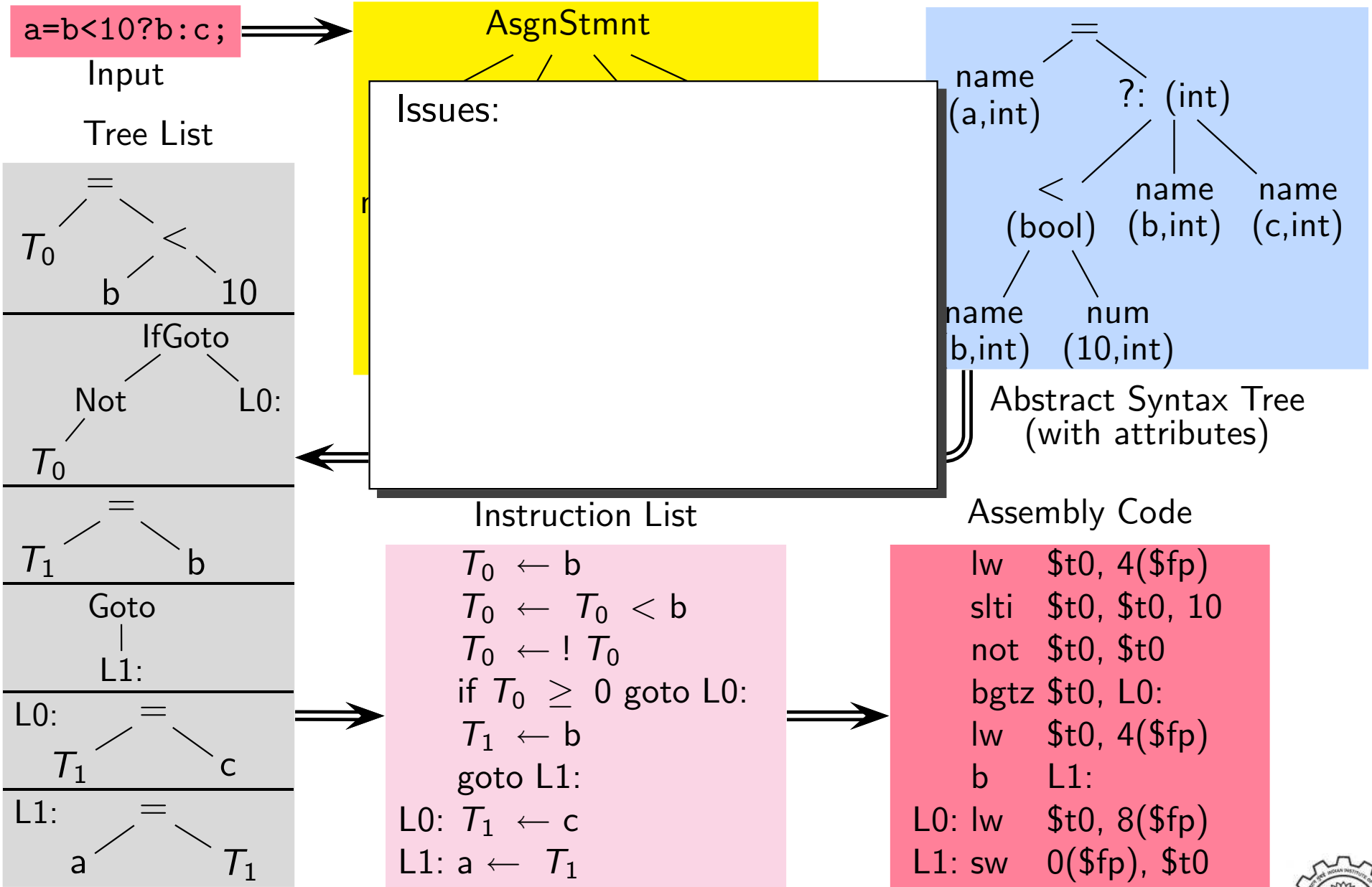
Translation Sequence in Our Example: Instruction Selection



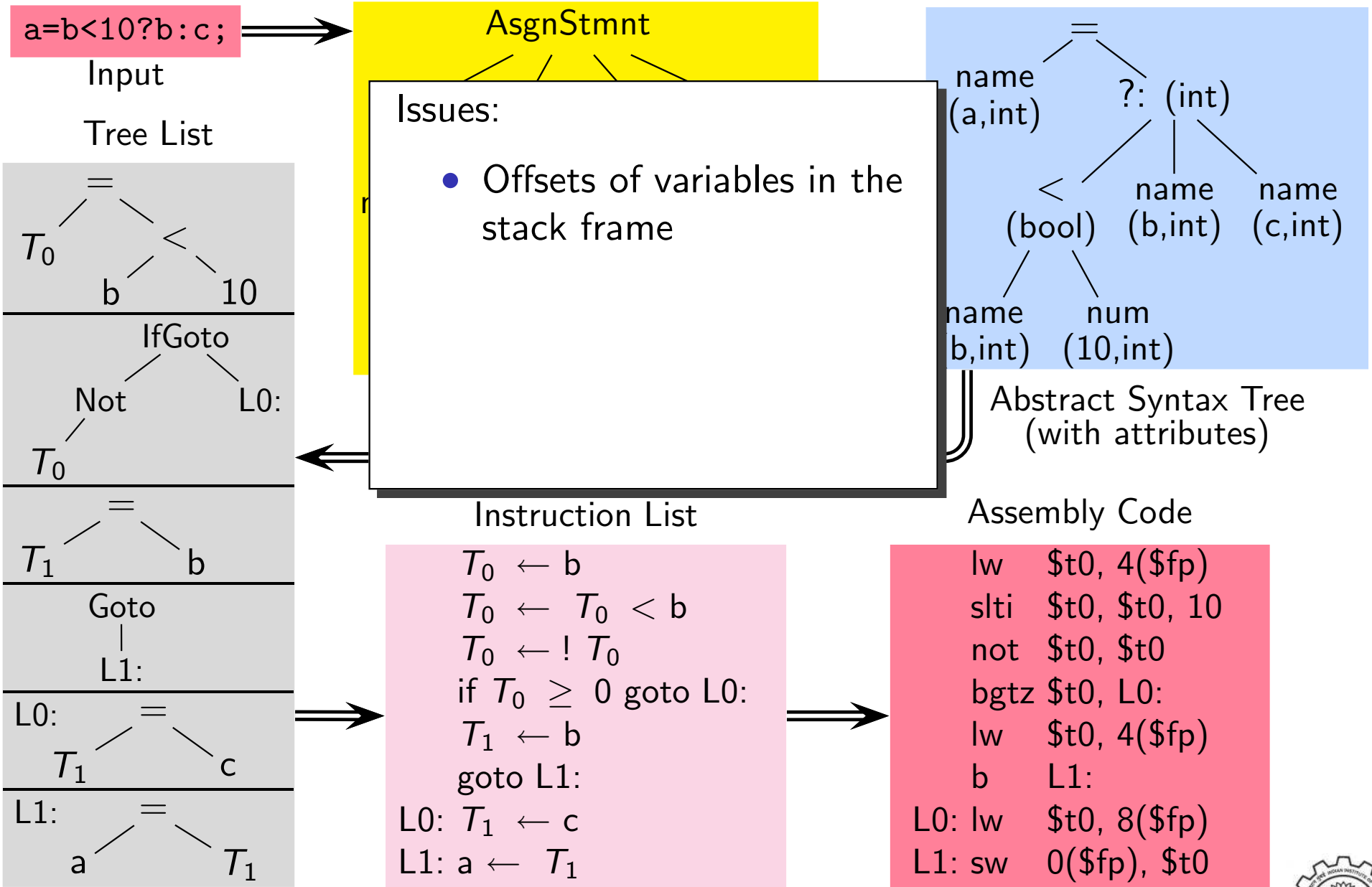
Translation Sequence in Our Example: Emitting Instructions



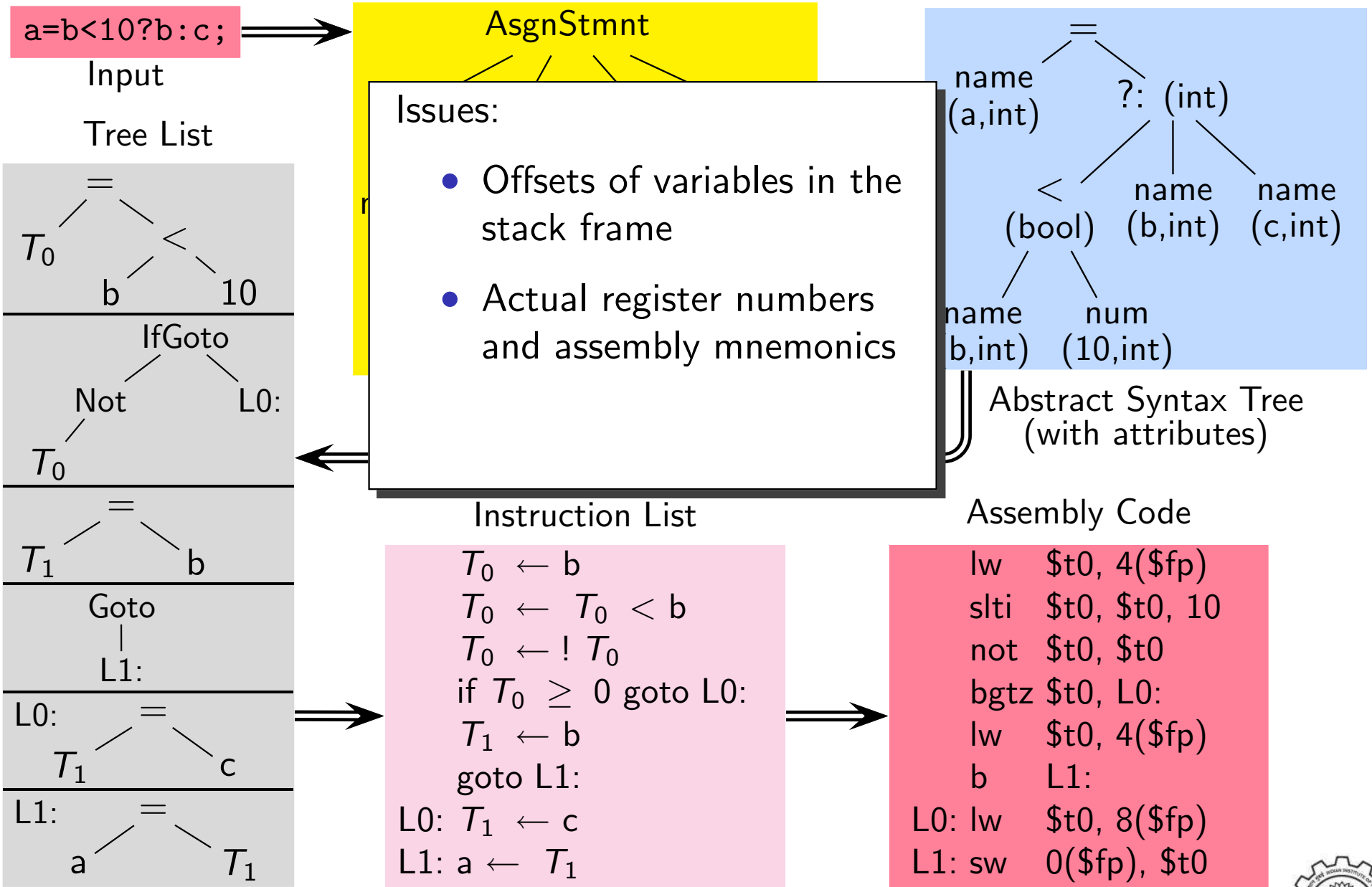
Translation Sequence in Our Example: Emitting Instructions



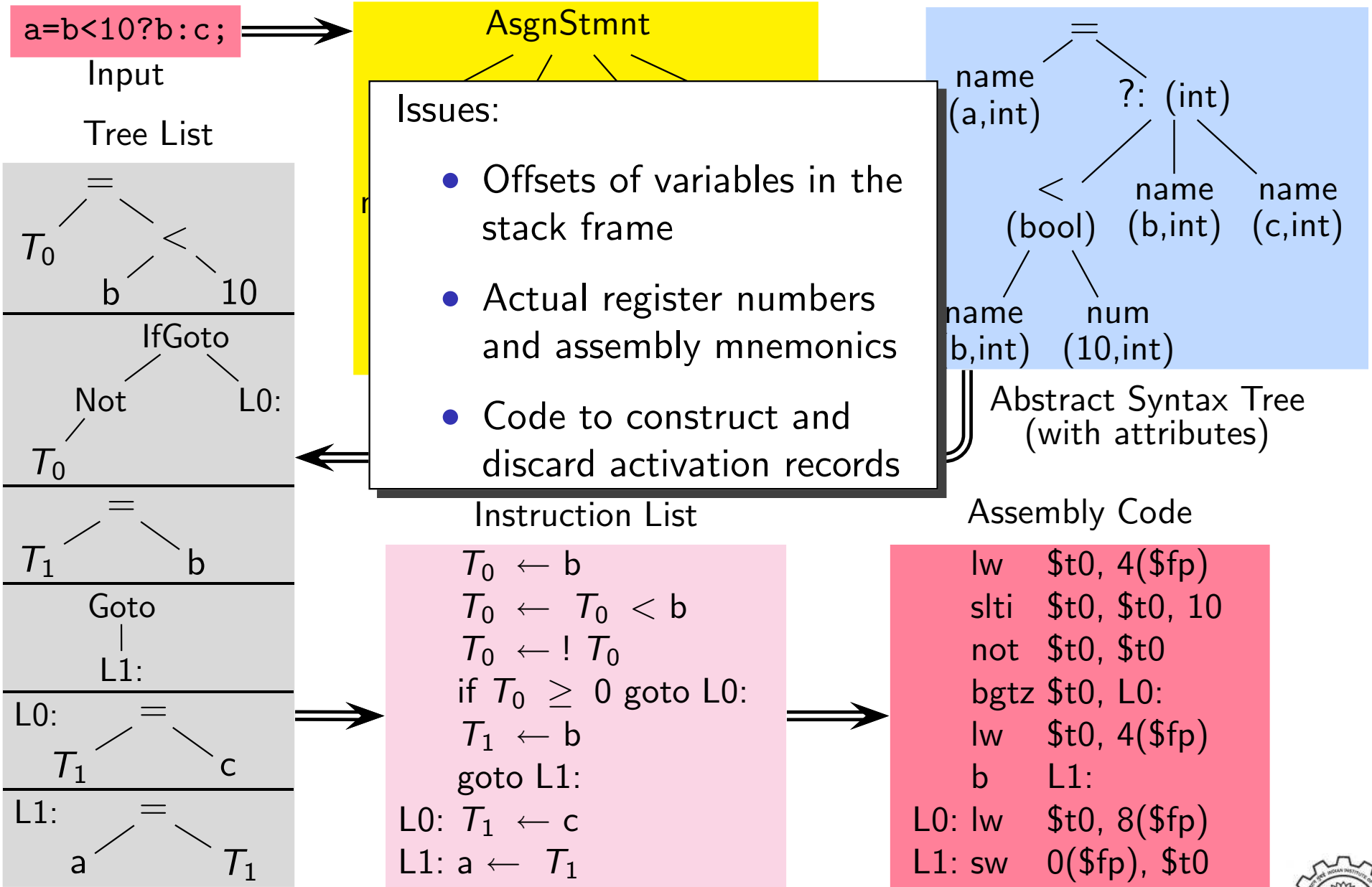
Translation Sequence in Our Example: Emitting Instructions



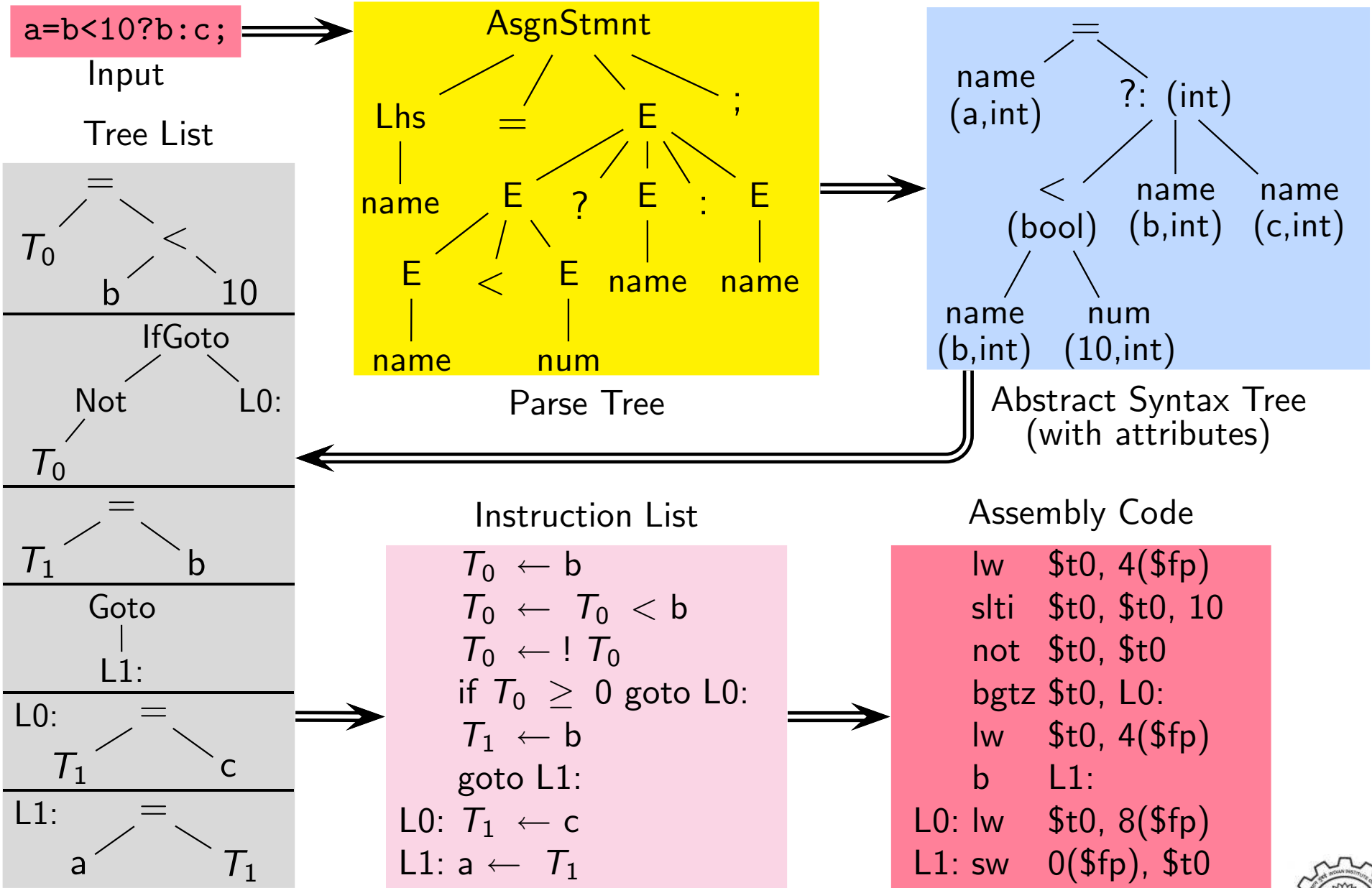
Translation Sequence in Our Example: Emitting Instructions



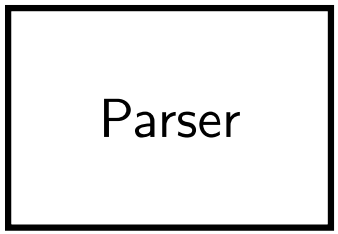
Translation Sequence in Our Example: Emitting Instructions



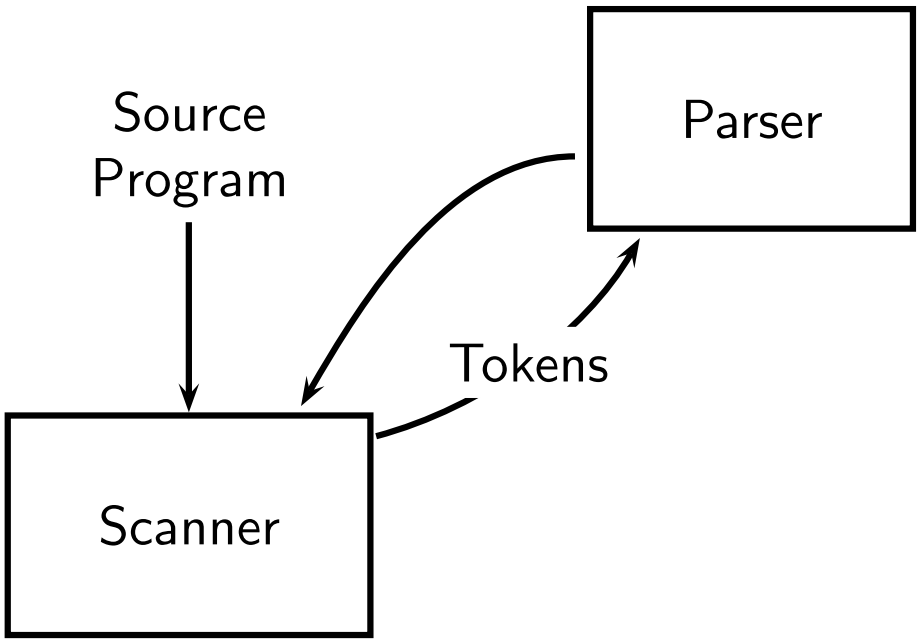
Translation Sequence in Our Example: Emitting Instructions



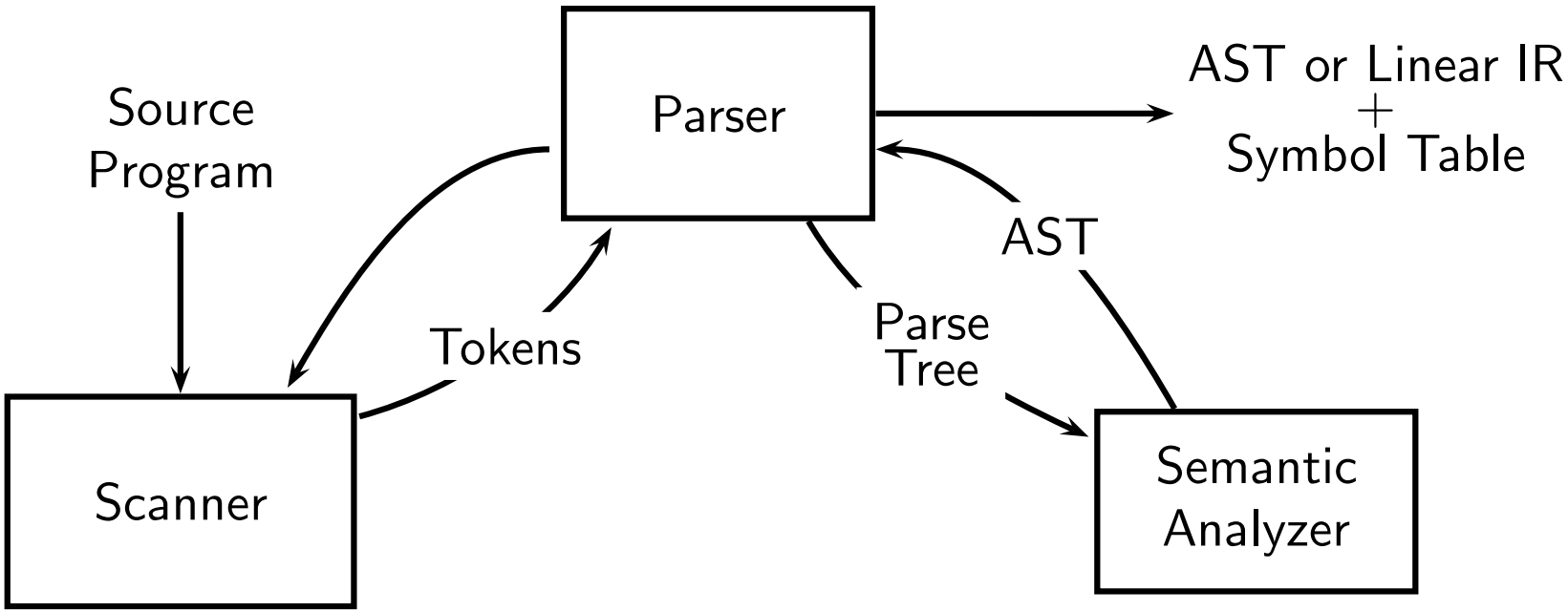
Typical Front Ends



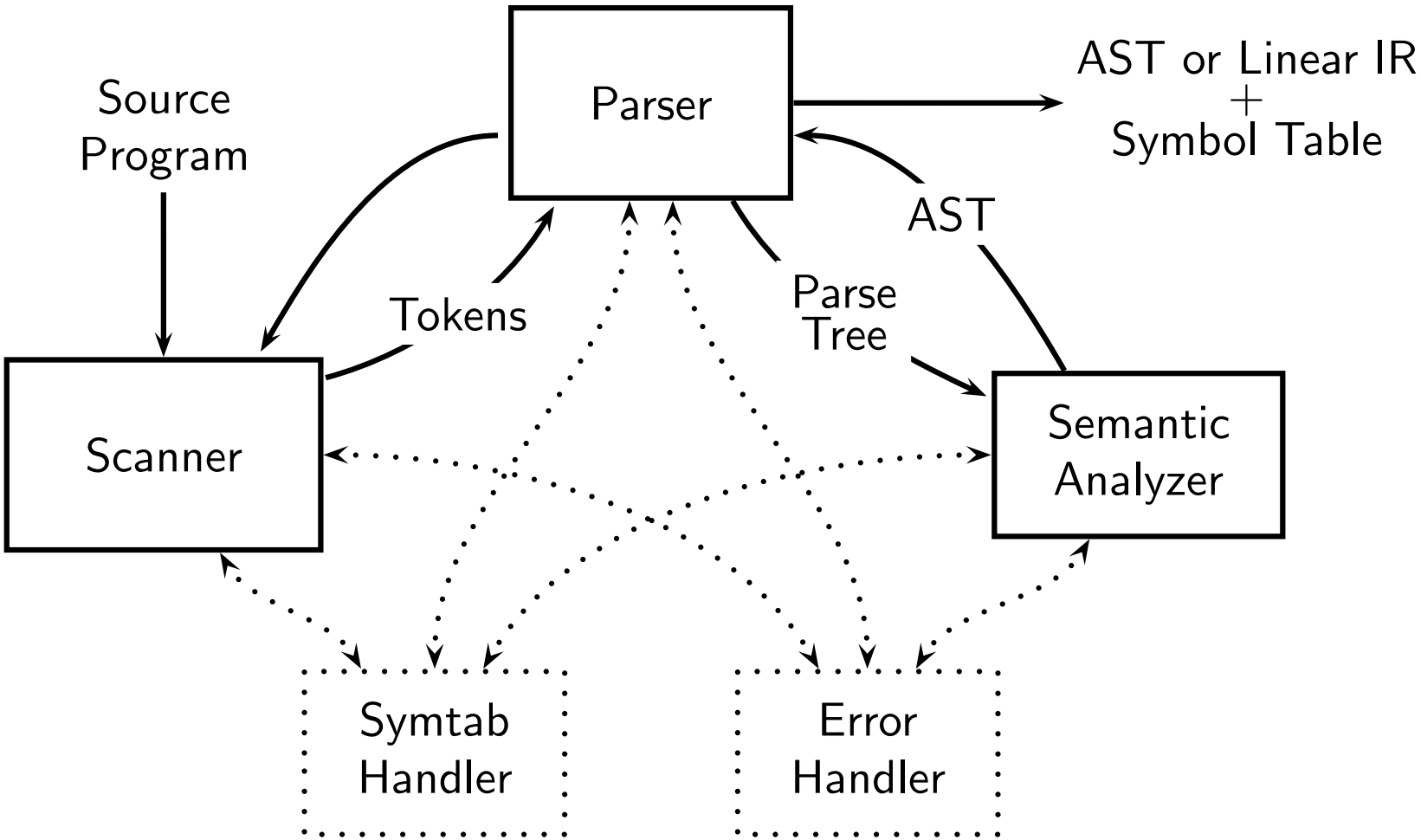
Typical Front Ends



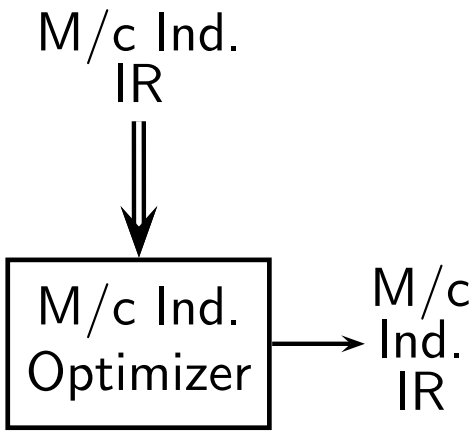
Typical Front Ends



Typical Front Ends

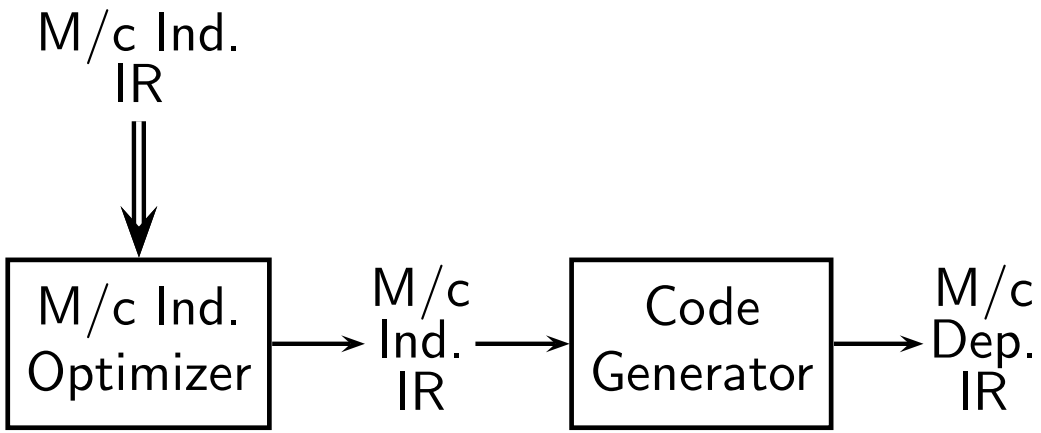


Typical Back Ends



- Compile time evaluations
- Eliminating redundant computations

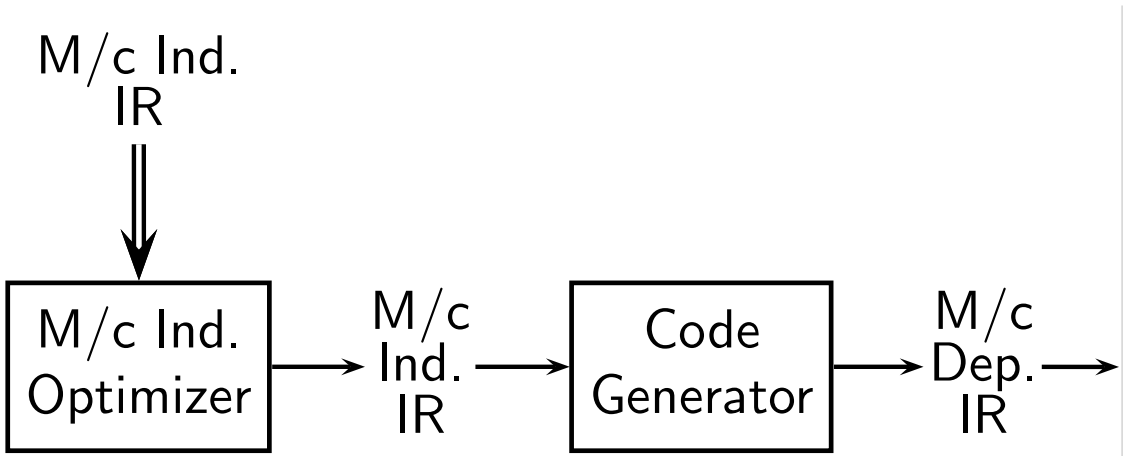
Typical Back Ends



- Compile time evaluations
- Eliminating redundant computations

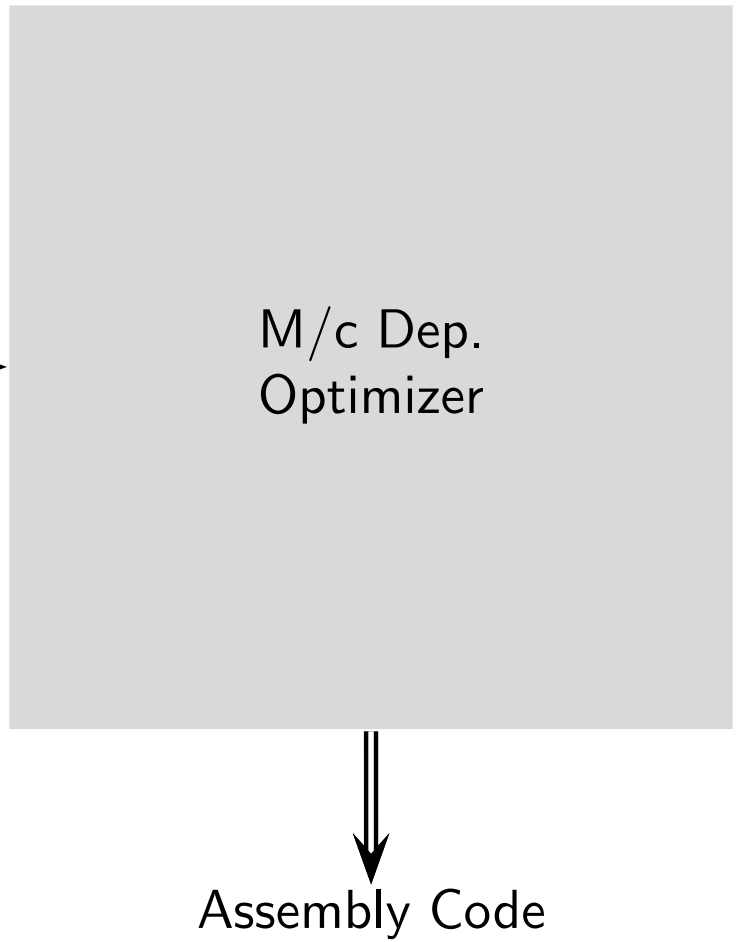
- Instruction Selection
- Local Reg Allocation
- Choice of Order of Evaluation

Typical Back Ends

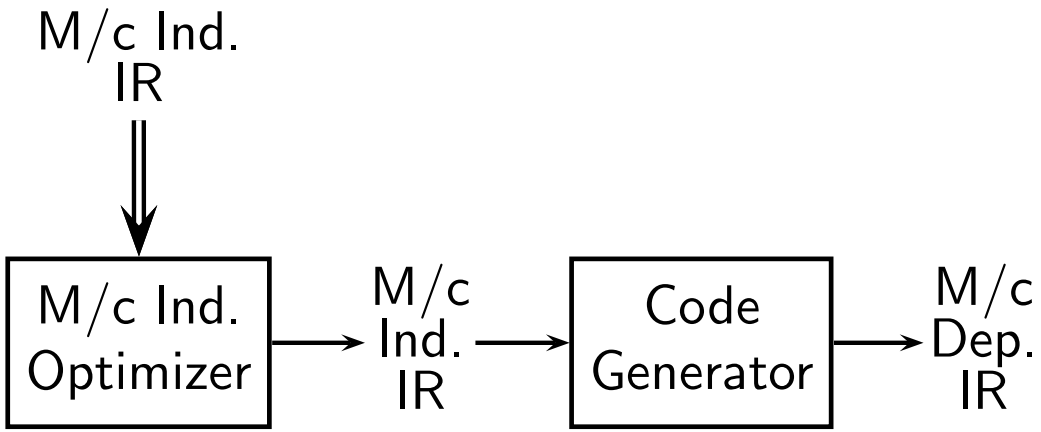


- Compile time evaluations
- Eliminating redundant computations

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

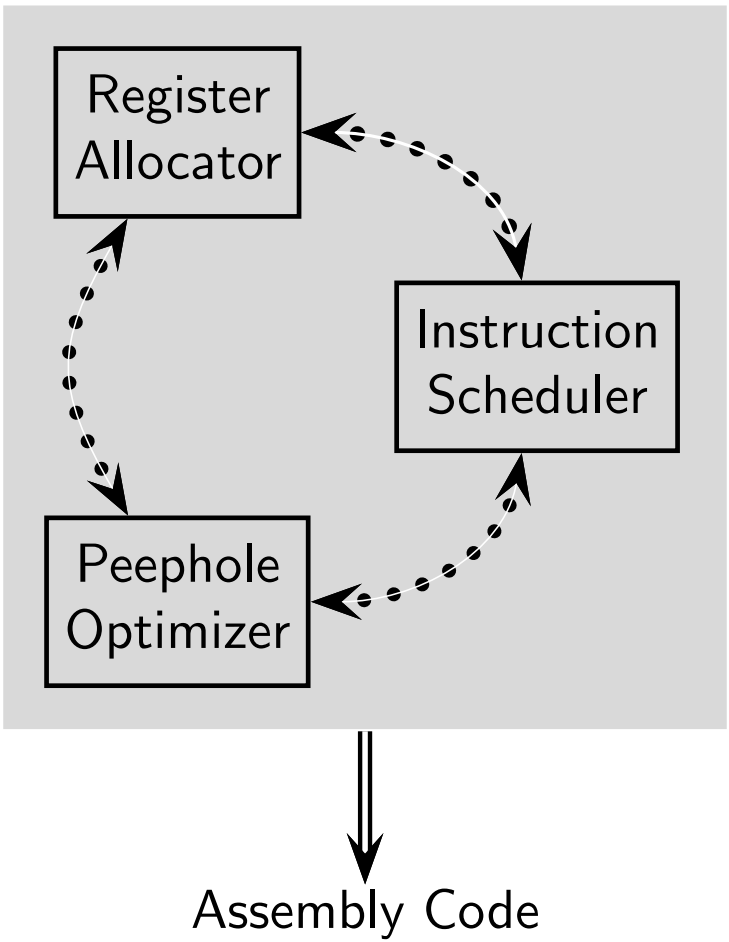


Typical Back Ends



- Compile time evaluations
- Eliminating redundant computations

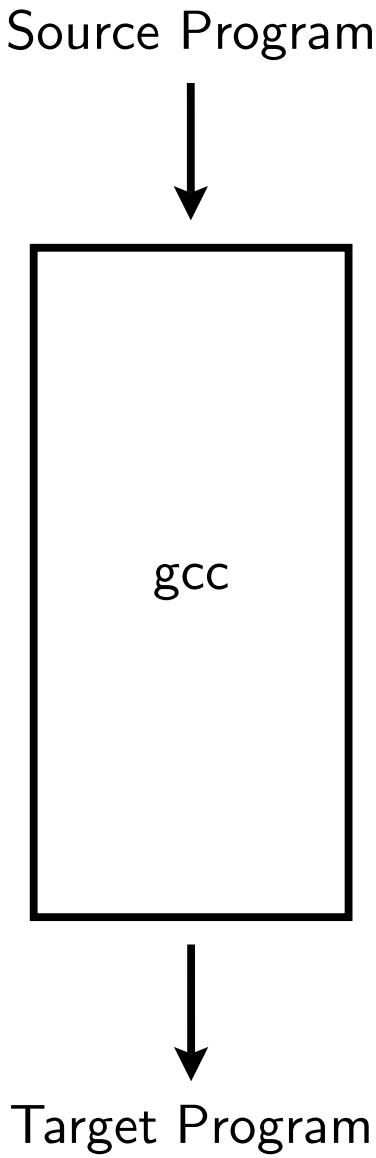
- Instruction Selection
- Local Reg Allocation
- Choice of Order of Evaluation



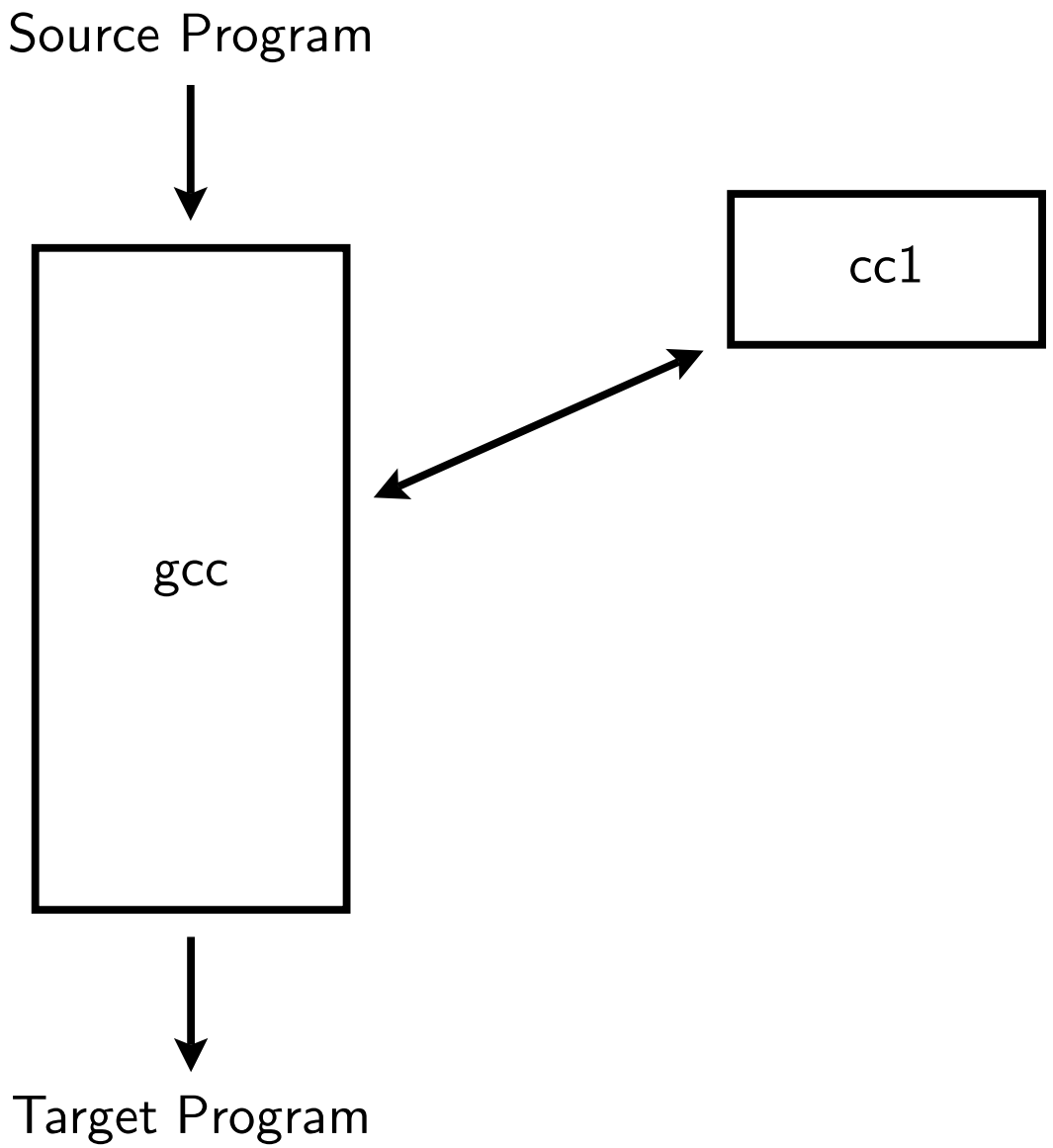
Part 4

Introduction to GCC

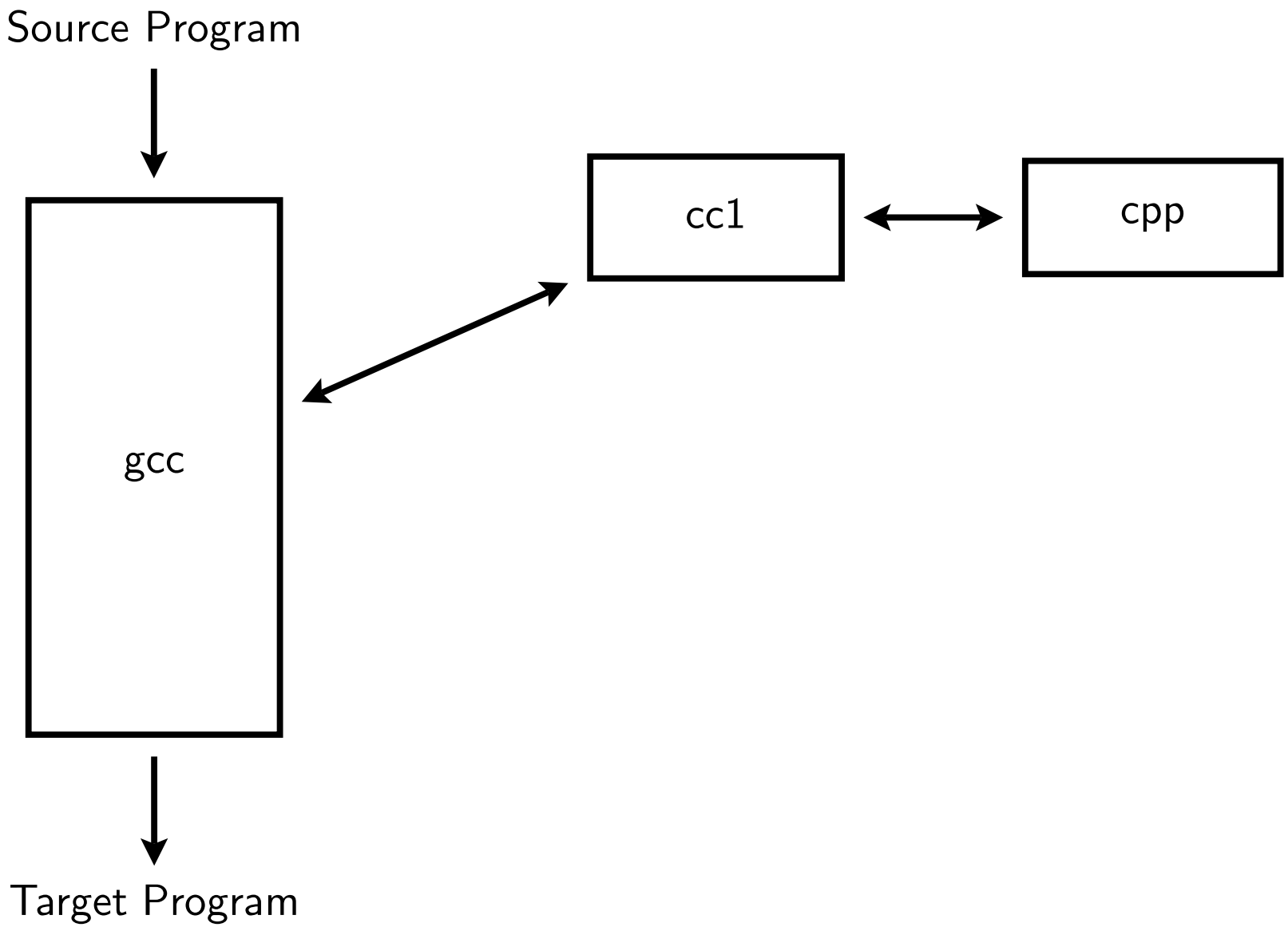
The GNU Tool Chain



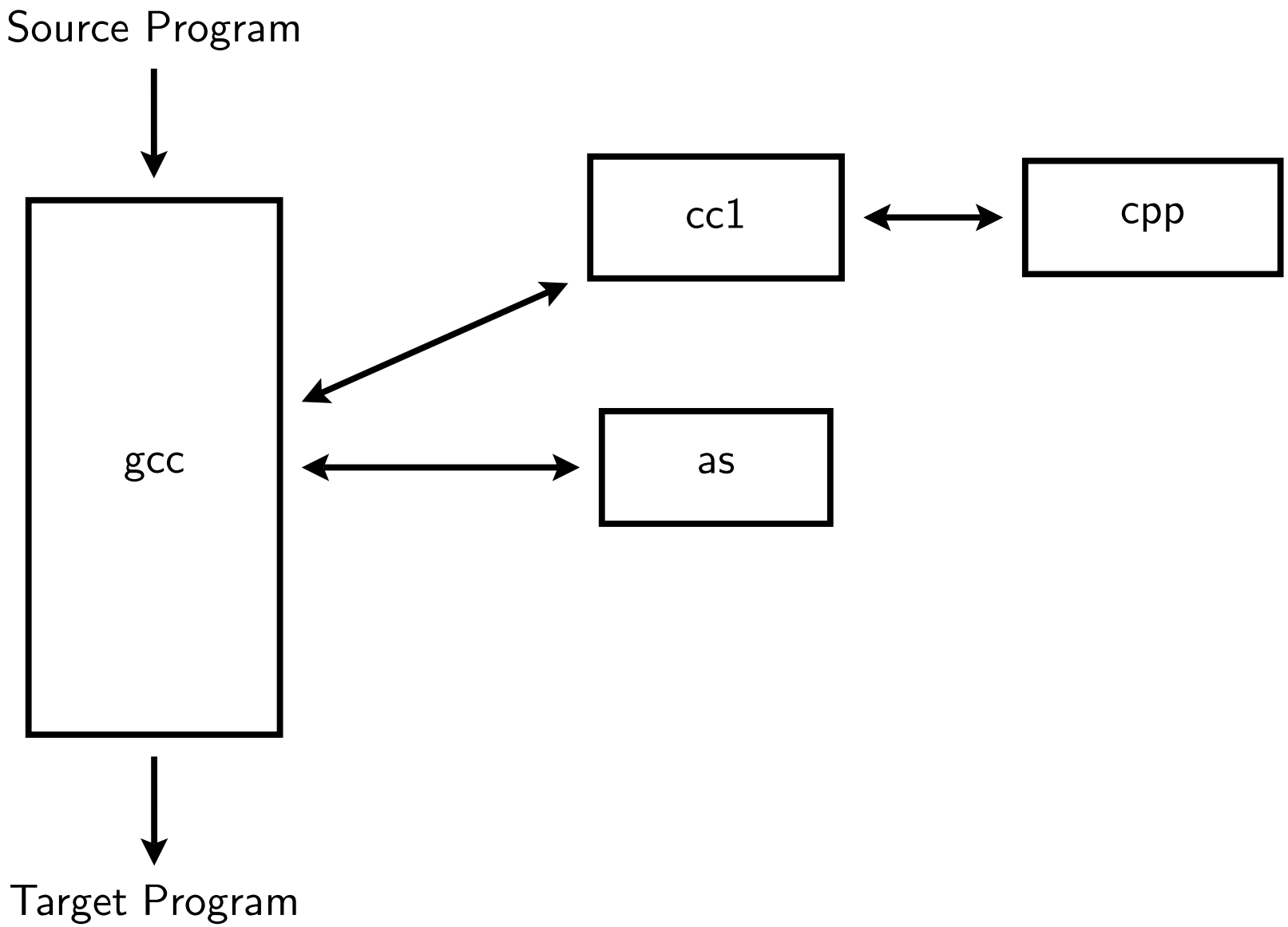
The GNU Tool Chain



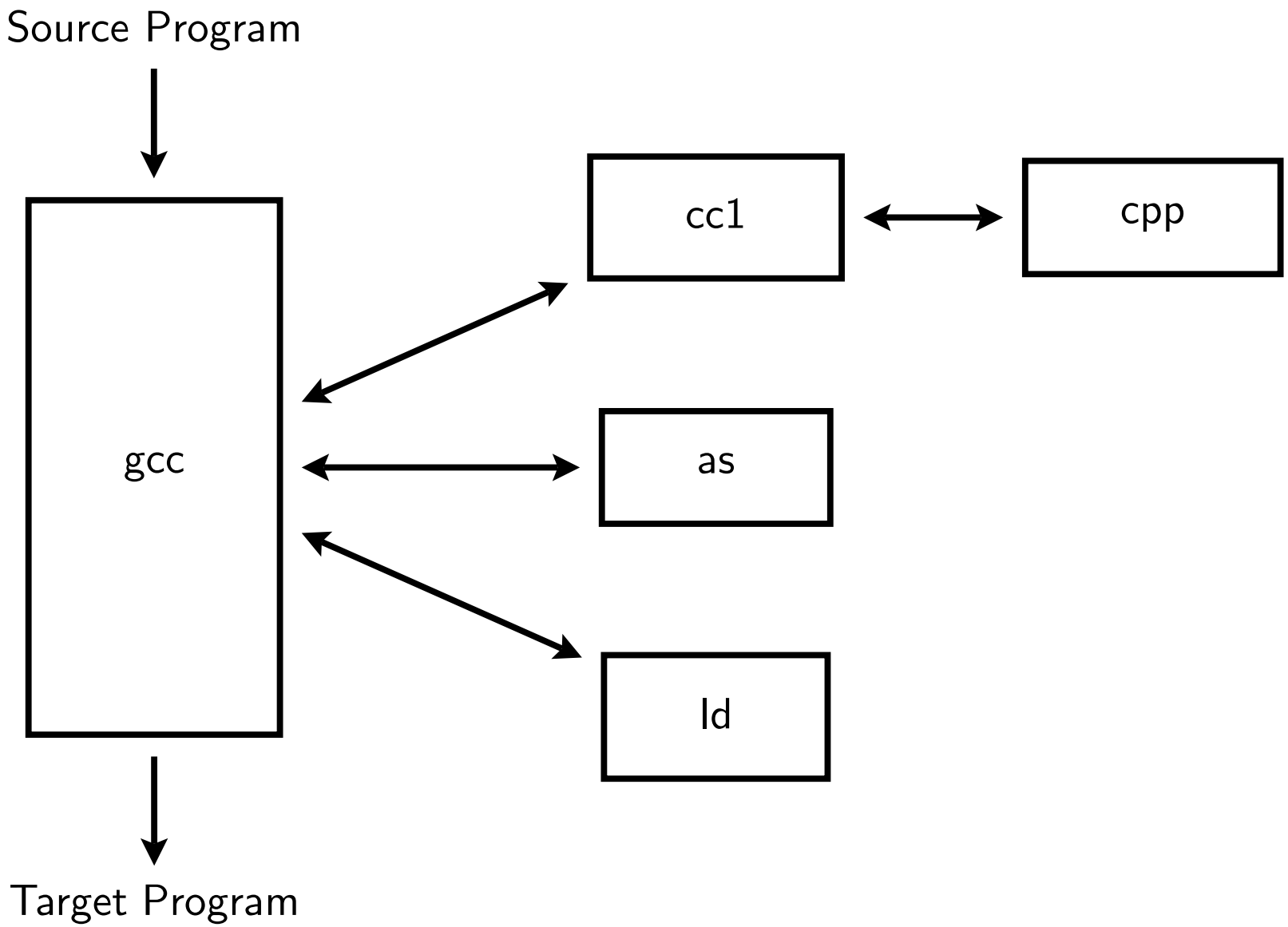
The GNU Tool Chain



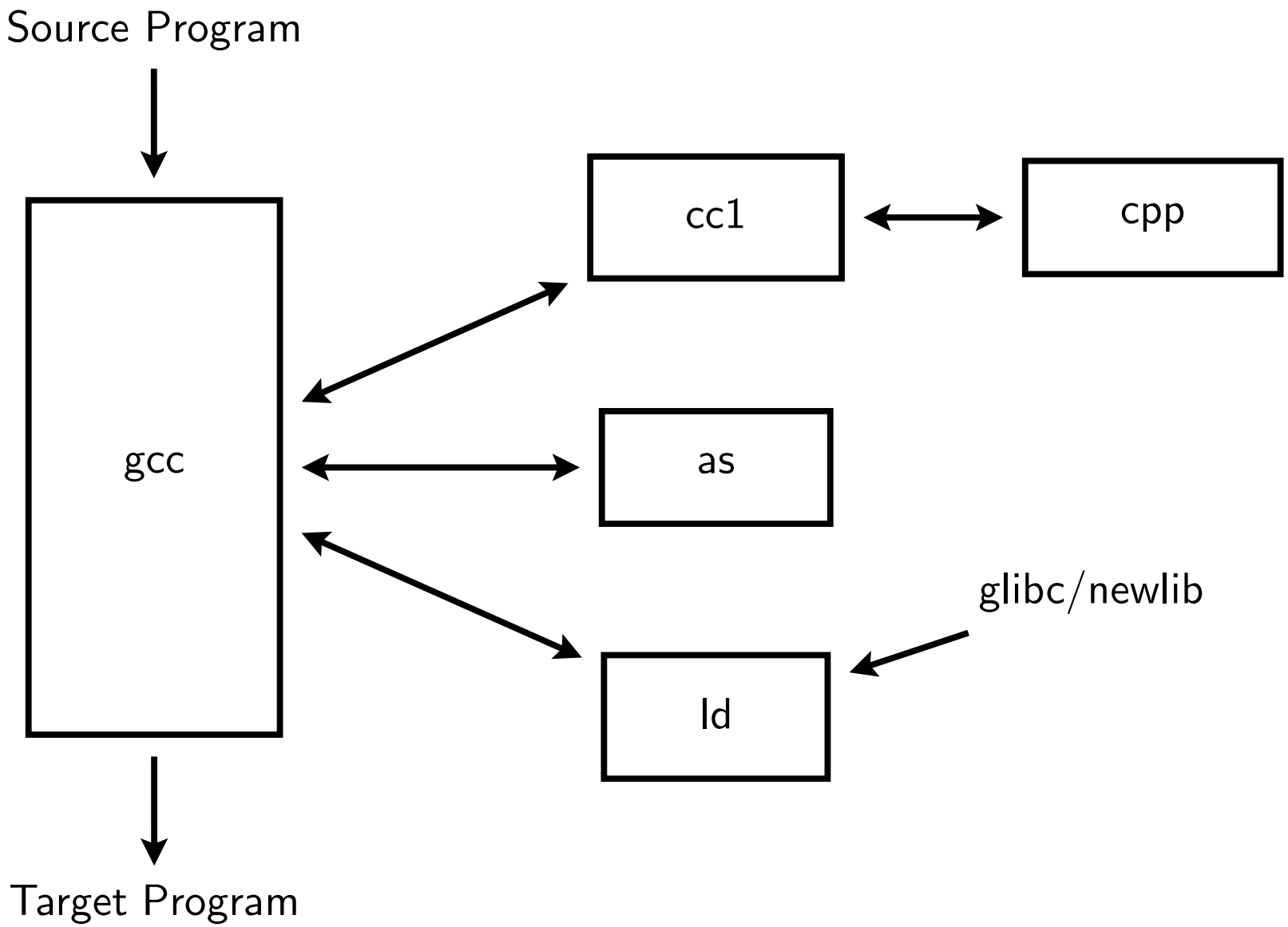
The GNU Tool Chain



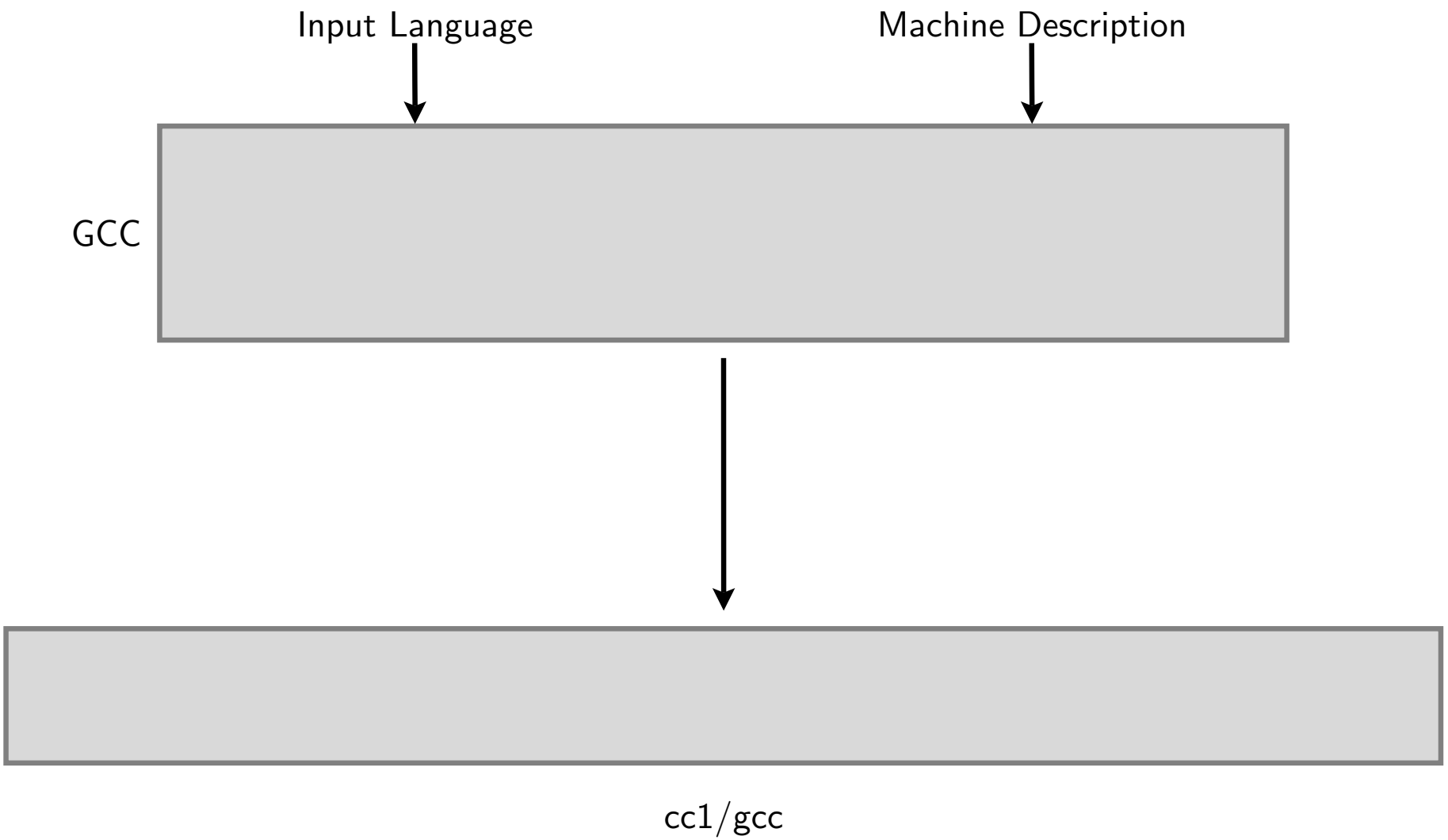
The GNU Tool Chain



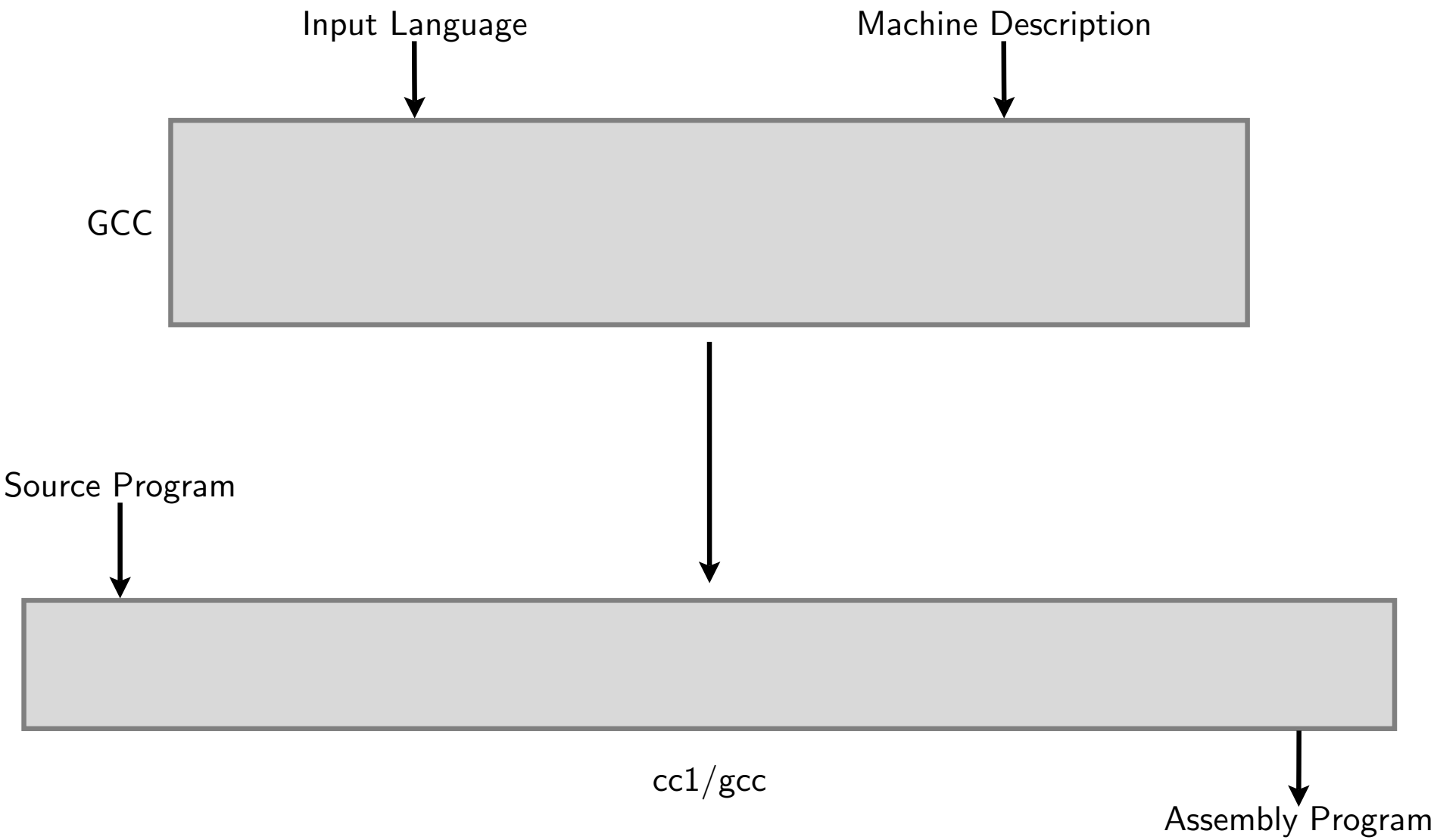
The GNU Tool Chain



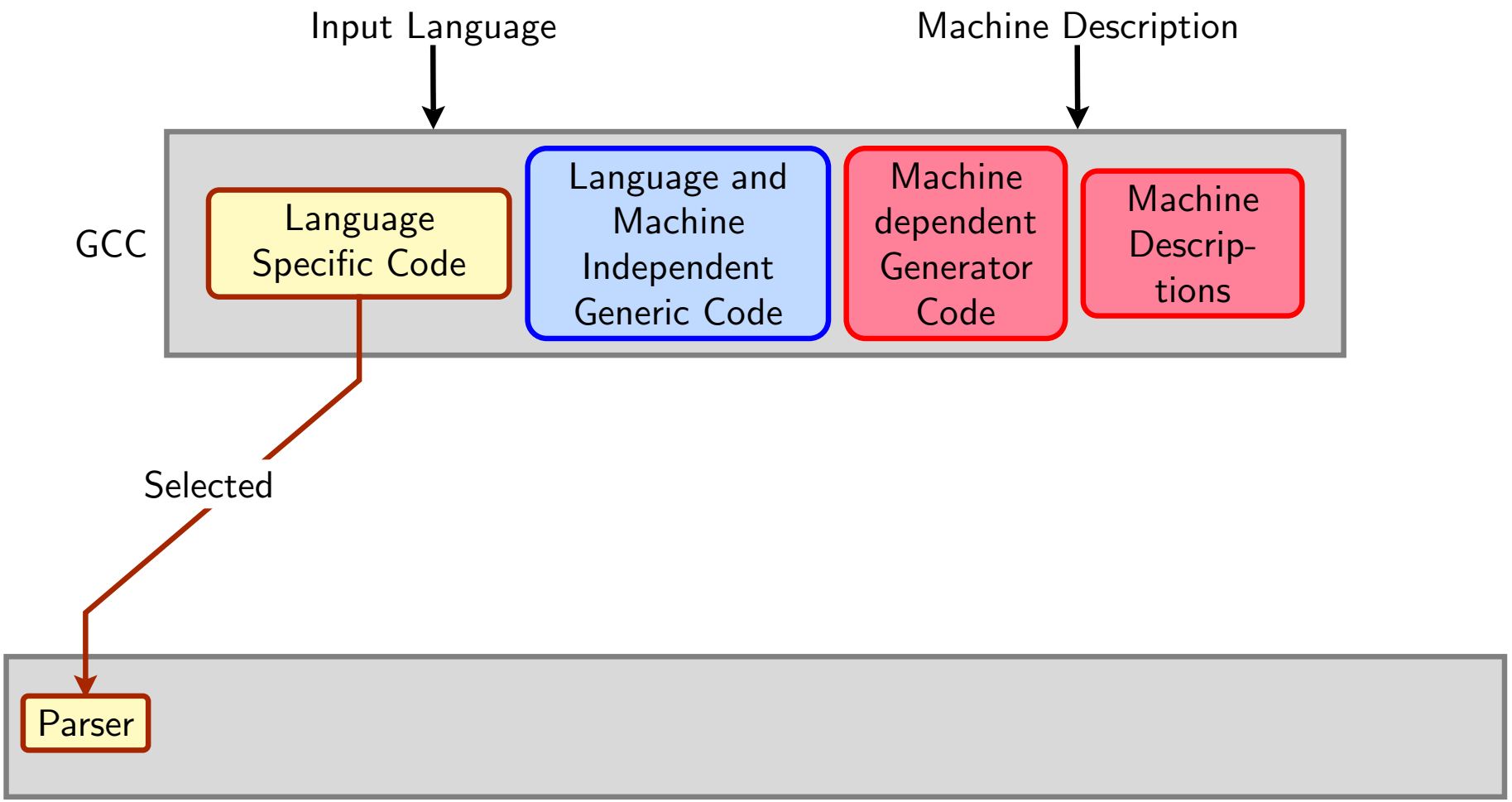
The GCC Framework



The GCC Framework

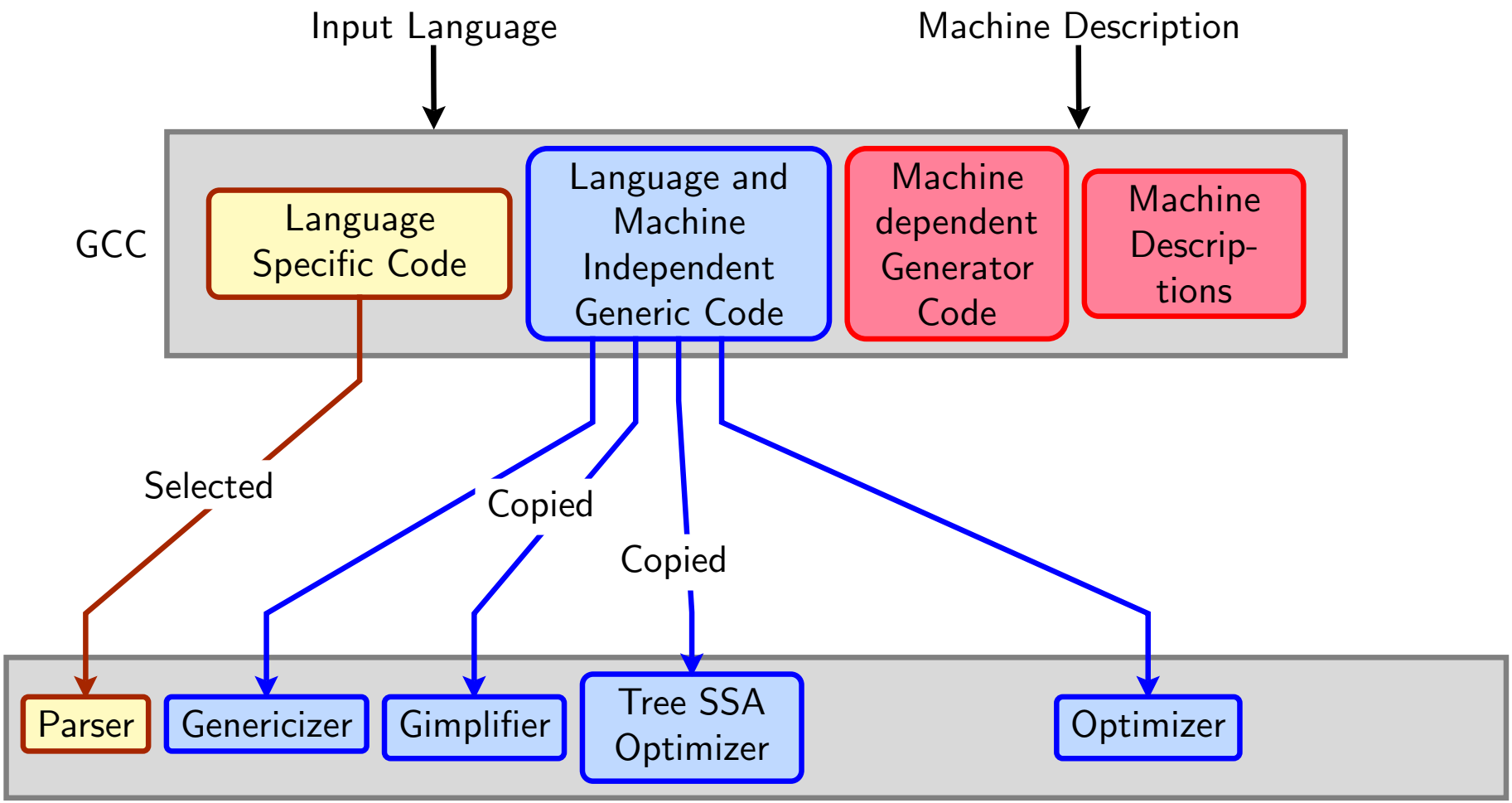


The GCC Framework



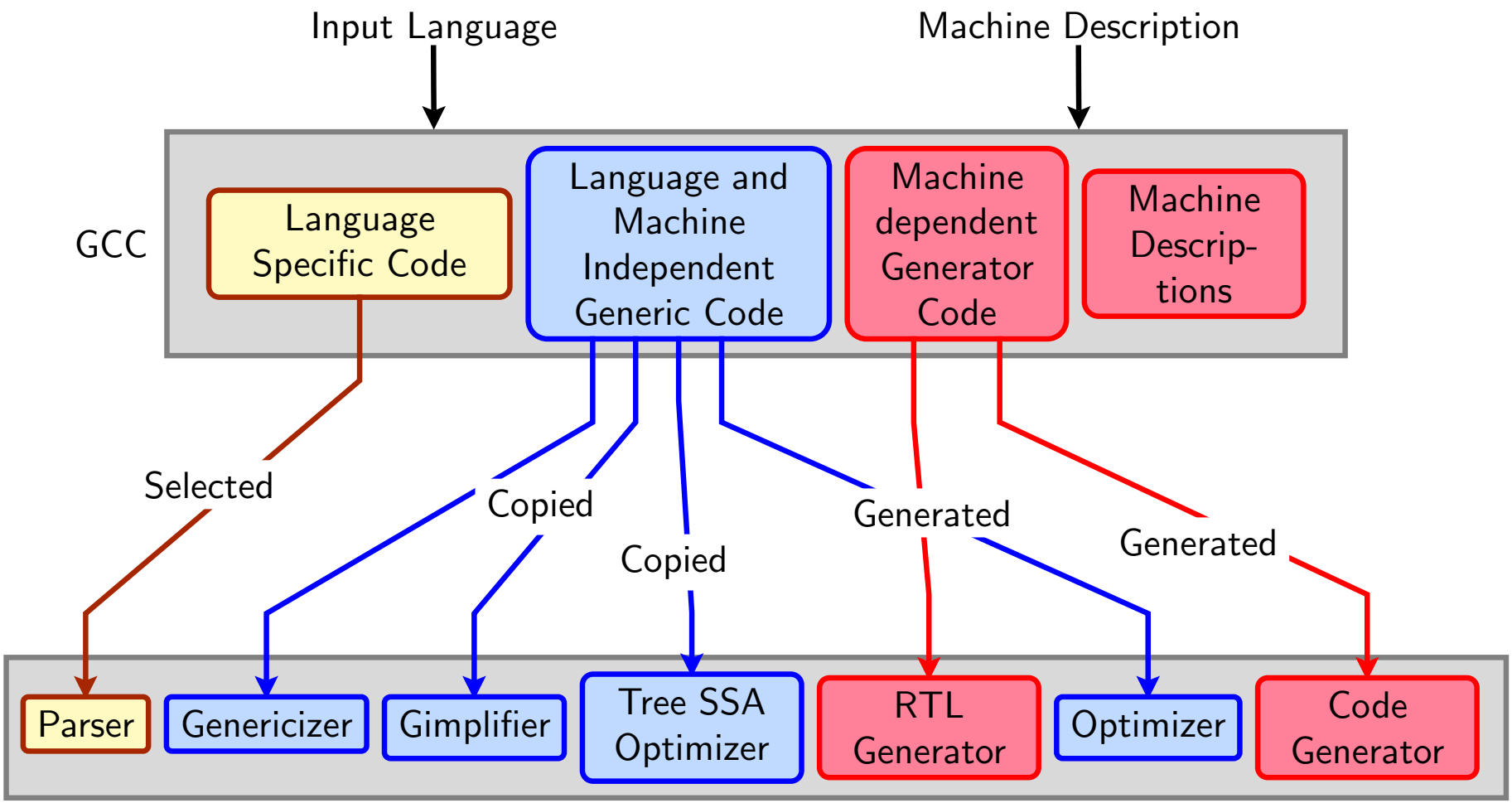
cc1/gcc

The GCC Framework



cc1/gcc

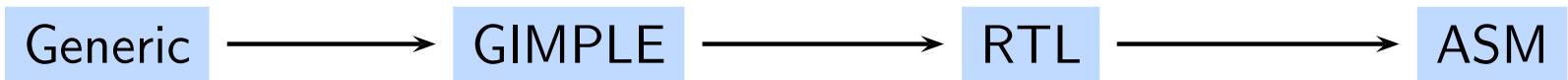
The GCC Framework



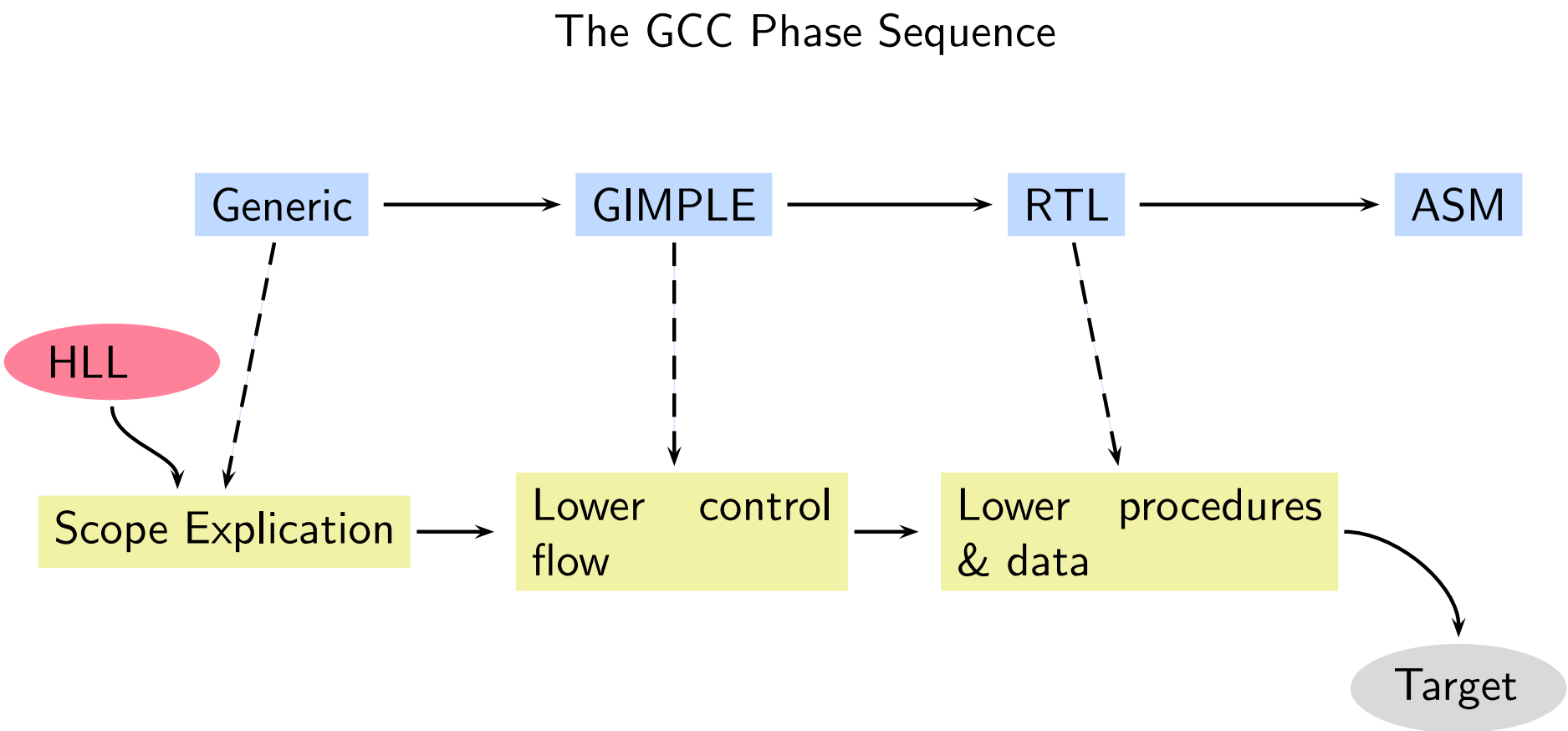
cc1/gcc

The `cc1` Phase Sequence as IR Chain

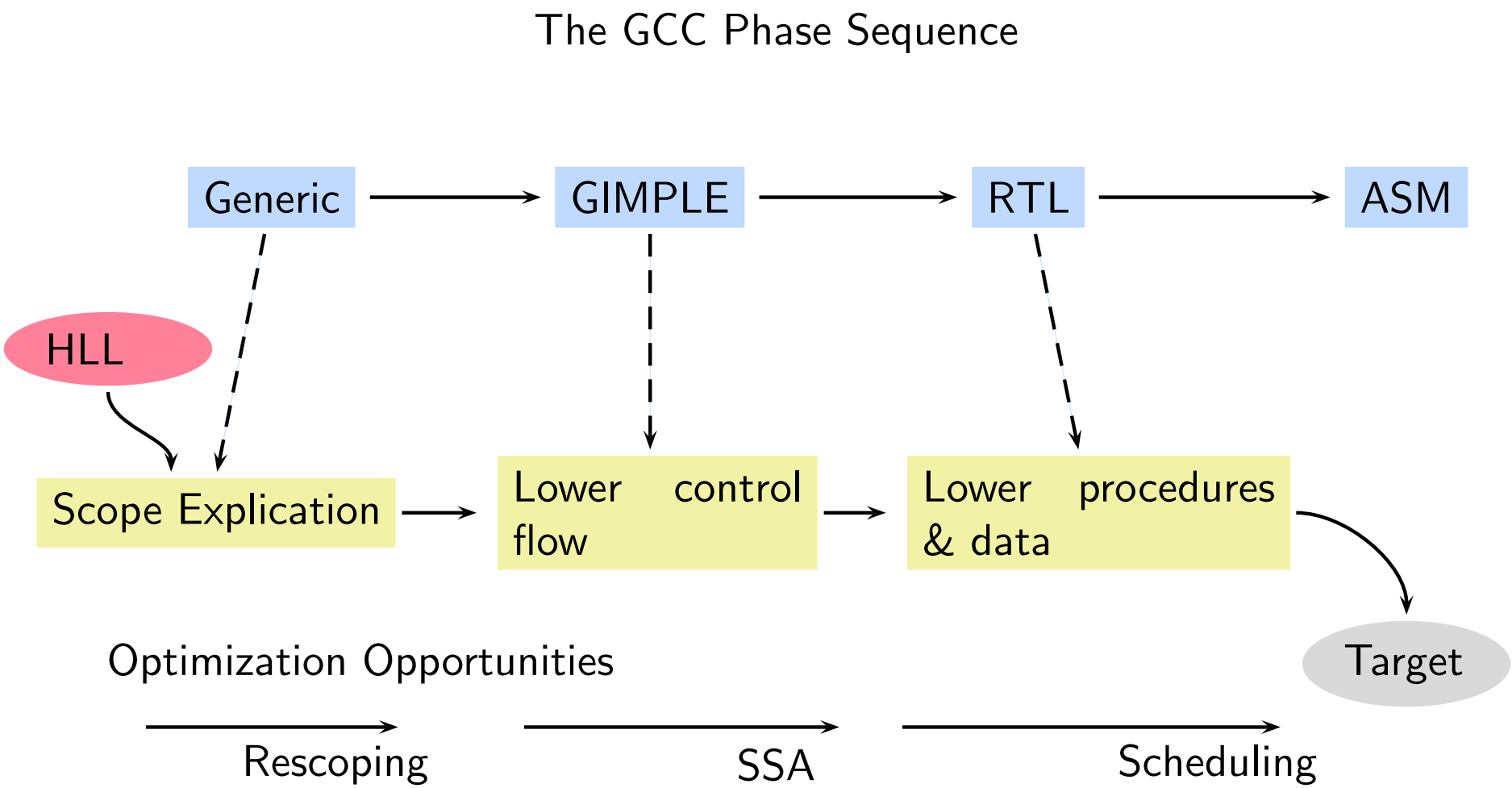
The GCC Phase Sequence



The `cc1` Phase Sequence as IR Chain



The `cc1` Phase Sequence as IR Chain



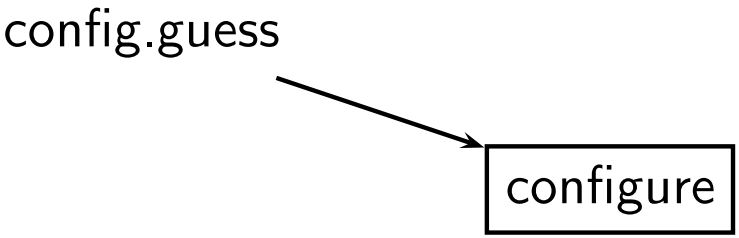
Part 5

Configuration and Building

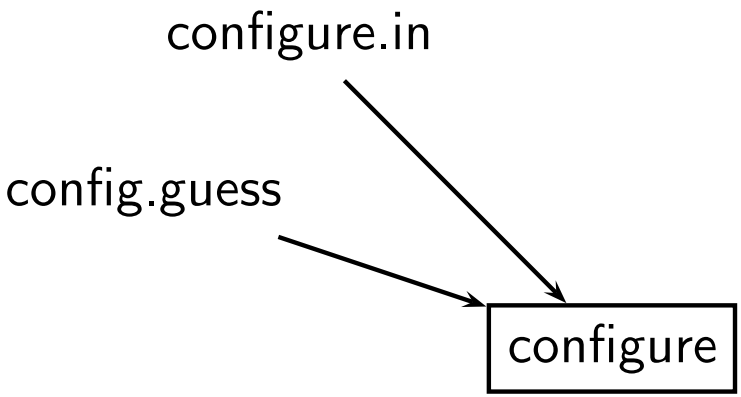
Configuring GCC

configure

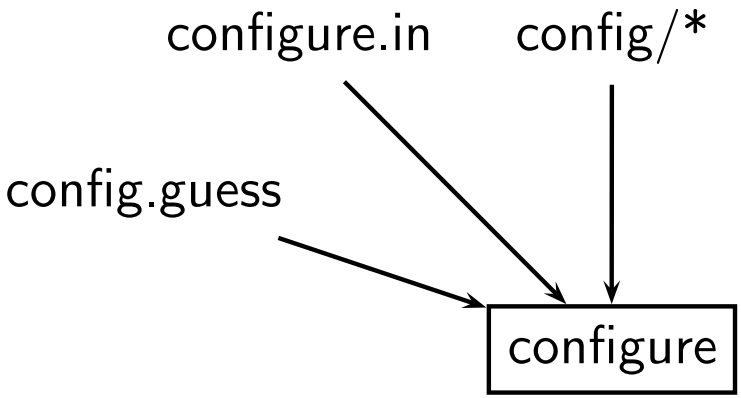
Configuring GCC



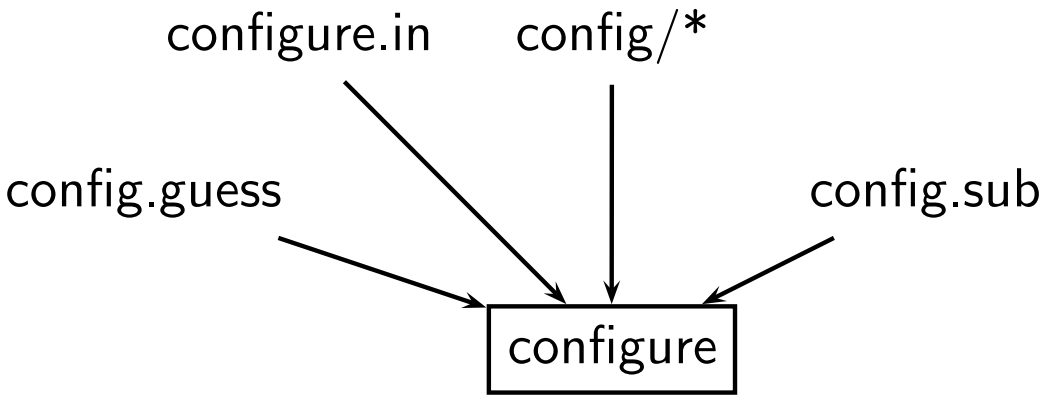
Configuring GCC



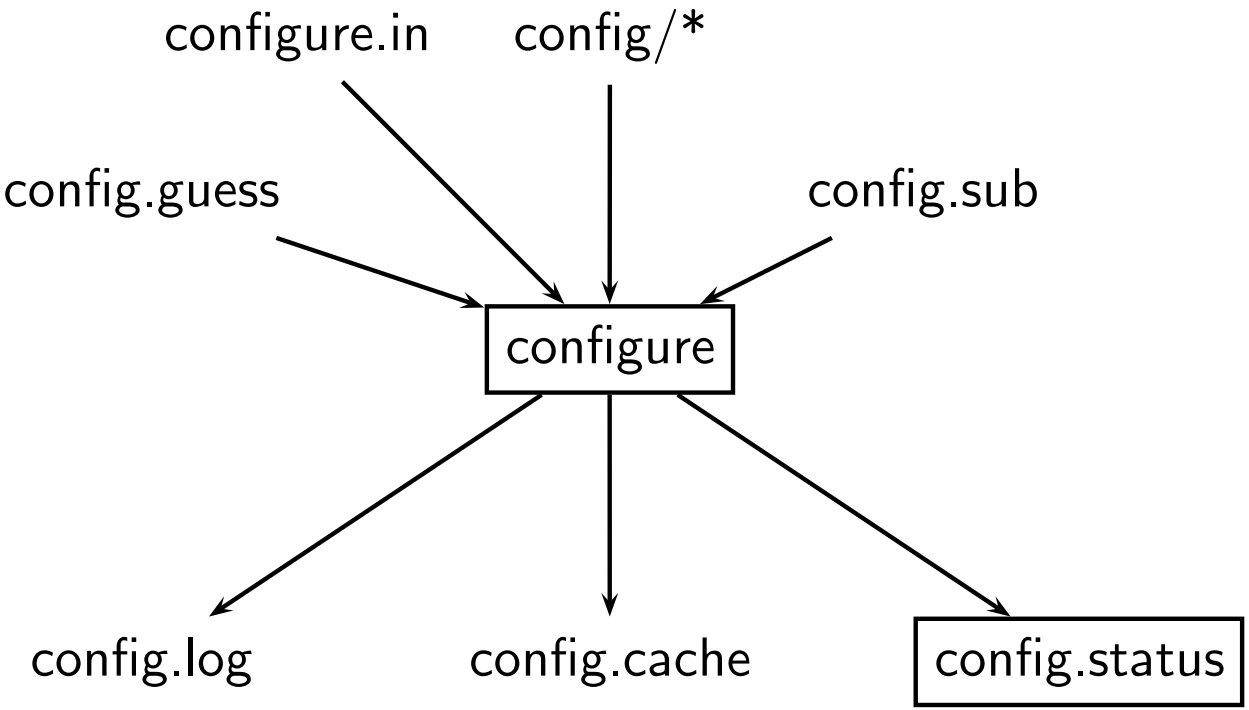
Configuring GCC



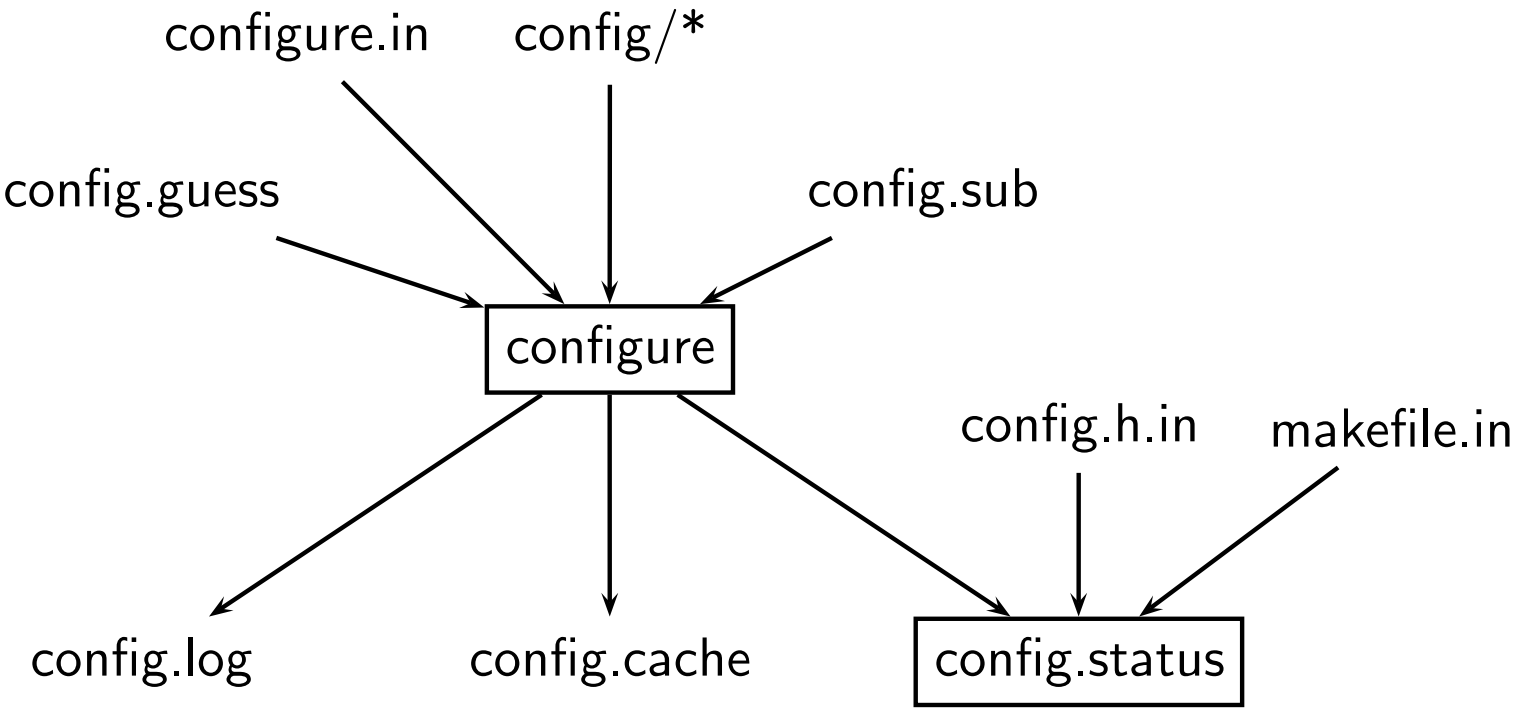
Configuring GCC



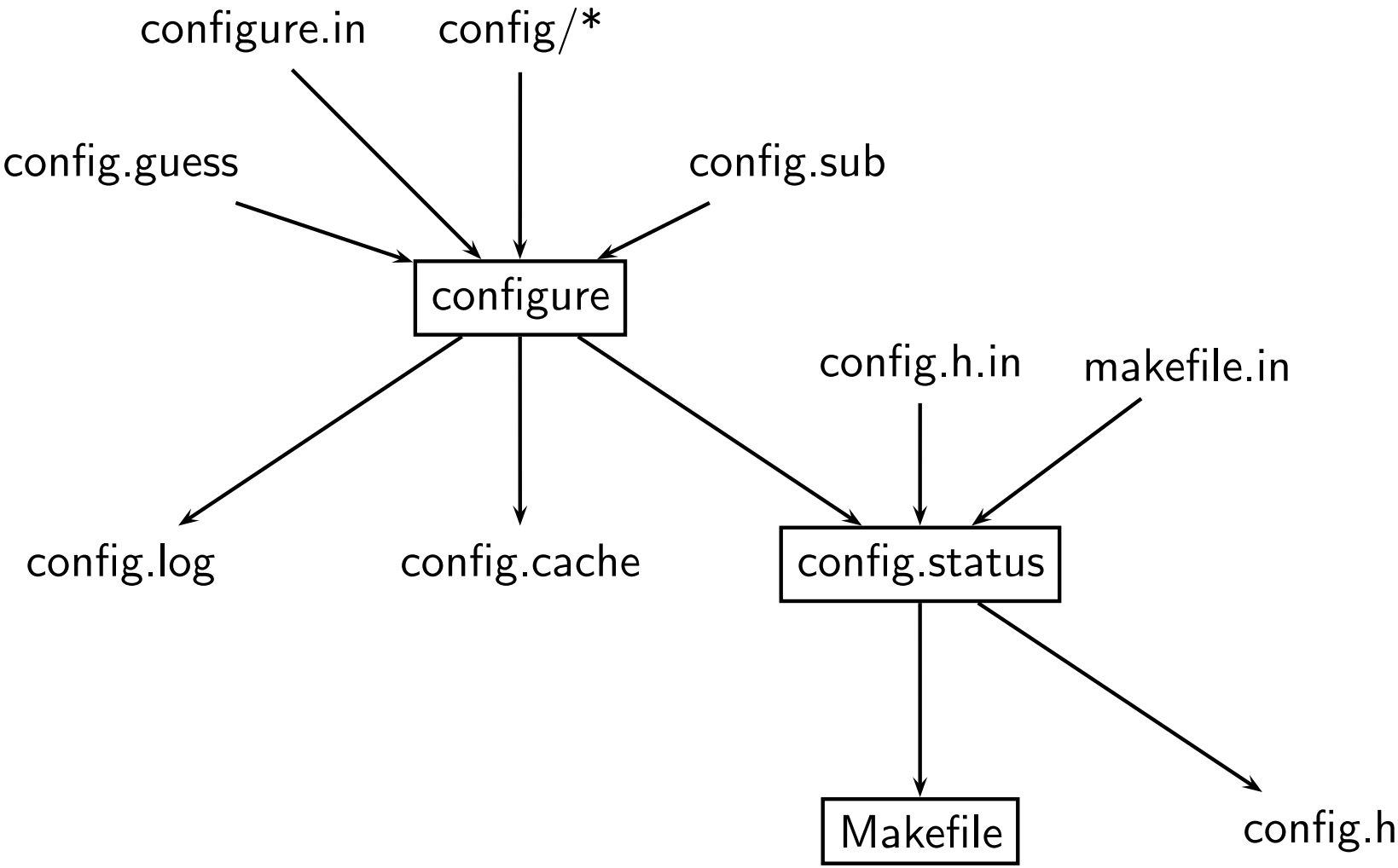
Configuring GCC



Configuring GCC



Configuring GCC



Part 6

About The Course