CS536
ST and IR

A Sahu

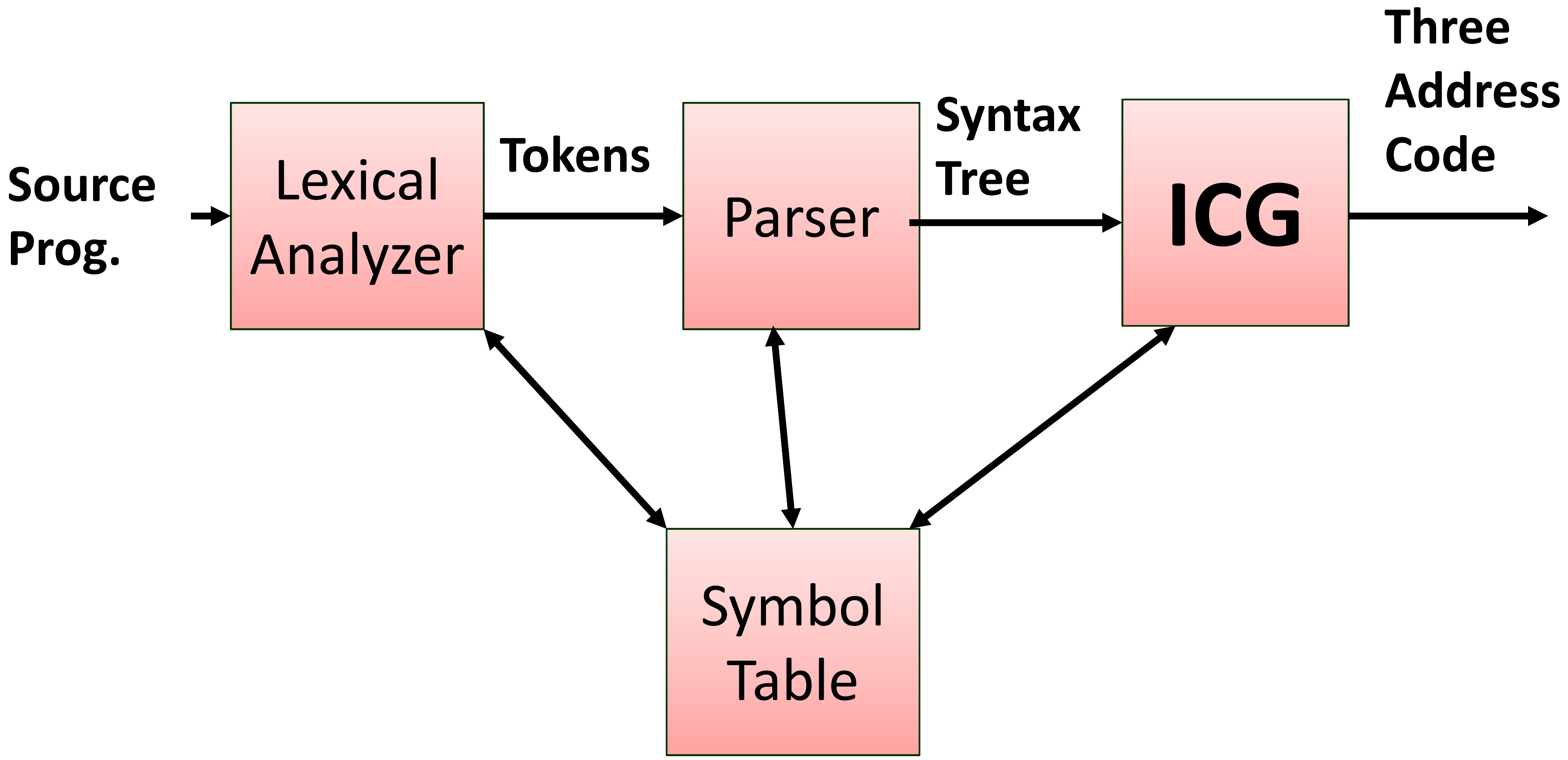
CSE, IIT Guwahati

Outline

- Symbol Table
- Intermediate Representation
  - Variants of Syntax Tree: DAG, ST
  - Tree address codes
  - Address and Instructions

Symbol Table

A model of Compiler Front end



ICG: Intermediate
Code Generator

Symbol Table

- It is Data Structure used by compiler to hold information about source program construct
- The information collected incrementally by analysis phases and used in synthesis phase to generate IR code
- ST have information about identifiers : type, position in storage,
- ST: needs support multiple declaration of the same name/identifiers with in a program

Symbol Table

- Separate symbol table for each scope
  - A program block with declaration will have its own ST with entry for each declaration in the block
  - Similarly A Class have it own ST.
- Symbol Table: uses hash DS
  - Number read are higher
  - Modify, add, delete to the ST are less

Use of Symbol Tables

- Use of ST: Role of ST is to pass information from declaration to uses
  - A semantic action “puts : information about ID x into ST, when declaring of x is analyzed
  - A semantic action associated with production

  Such as ***factor →id*** gets info about id from ST.
  - A
- During translation class Env can be used

  top=new Env(top);

Example : ST for translating for blocks

program→ block	top=null

block →’{; decls stmts ’}’	saved=top;
	Top=new Env(top);
	print(“{“); top=saved;
	Print(“}”);

decls → decl | ϵ
decl→type id;
	s=new Symbol()
	s.type=type.lexme
	Top.put(id.lexme.s);

stmts→stmts stmt | ϵ stmt→block | factor;	Print(“; “);

factor →id	s=top.get(id.lexme)
	Print(id.lexme);print(“:”);
	print(s.type);

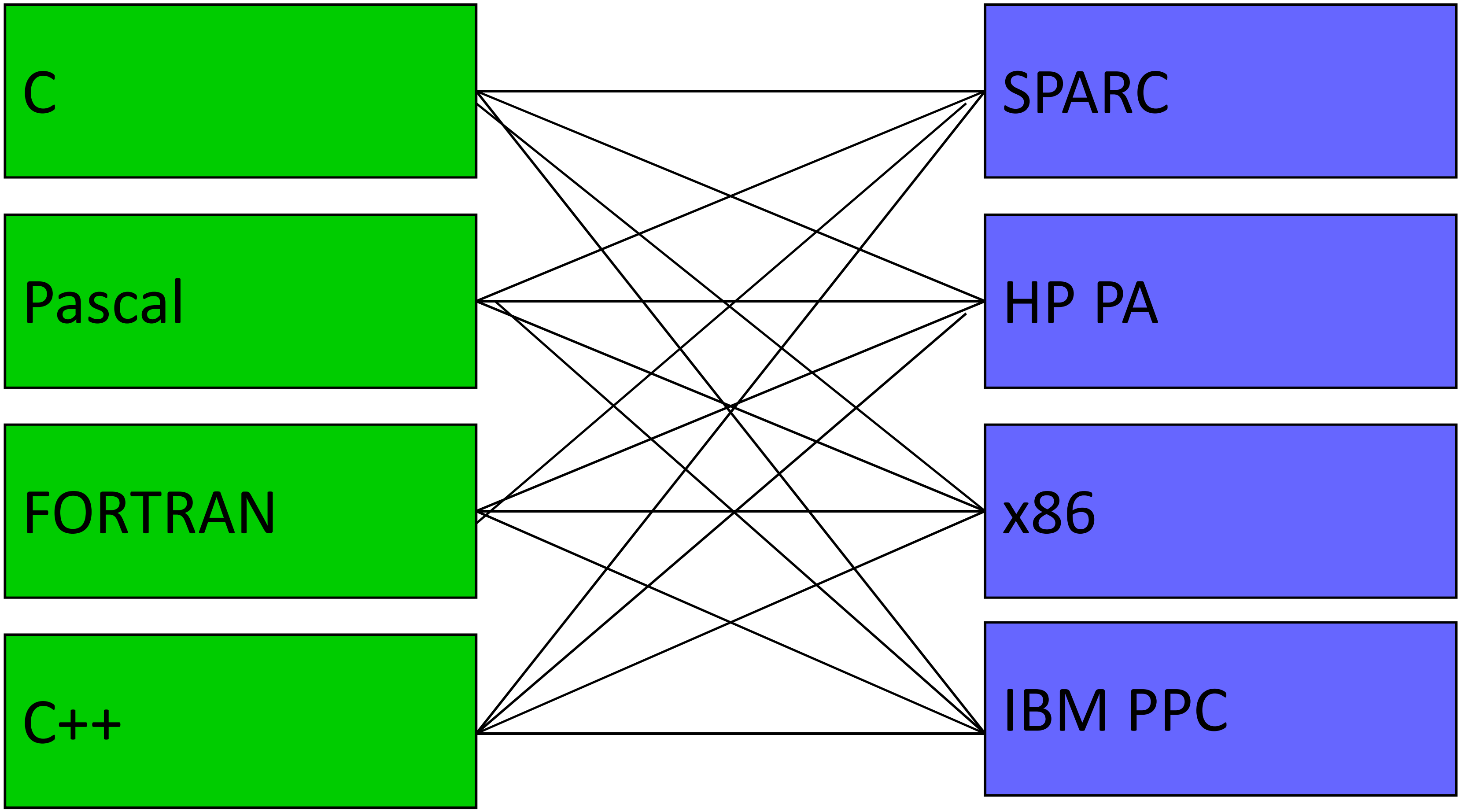
Intermediate Representation (IR)

Intermediate Representation (IR)

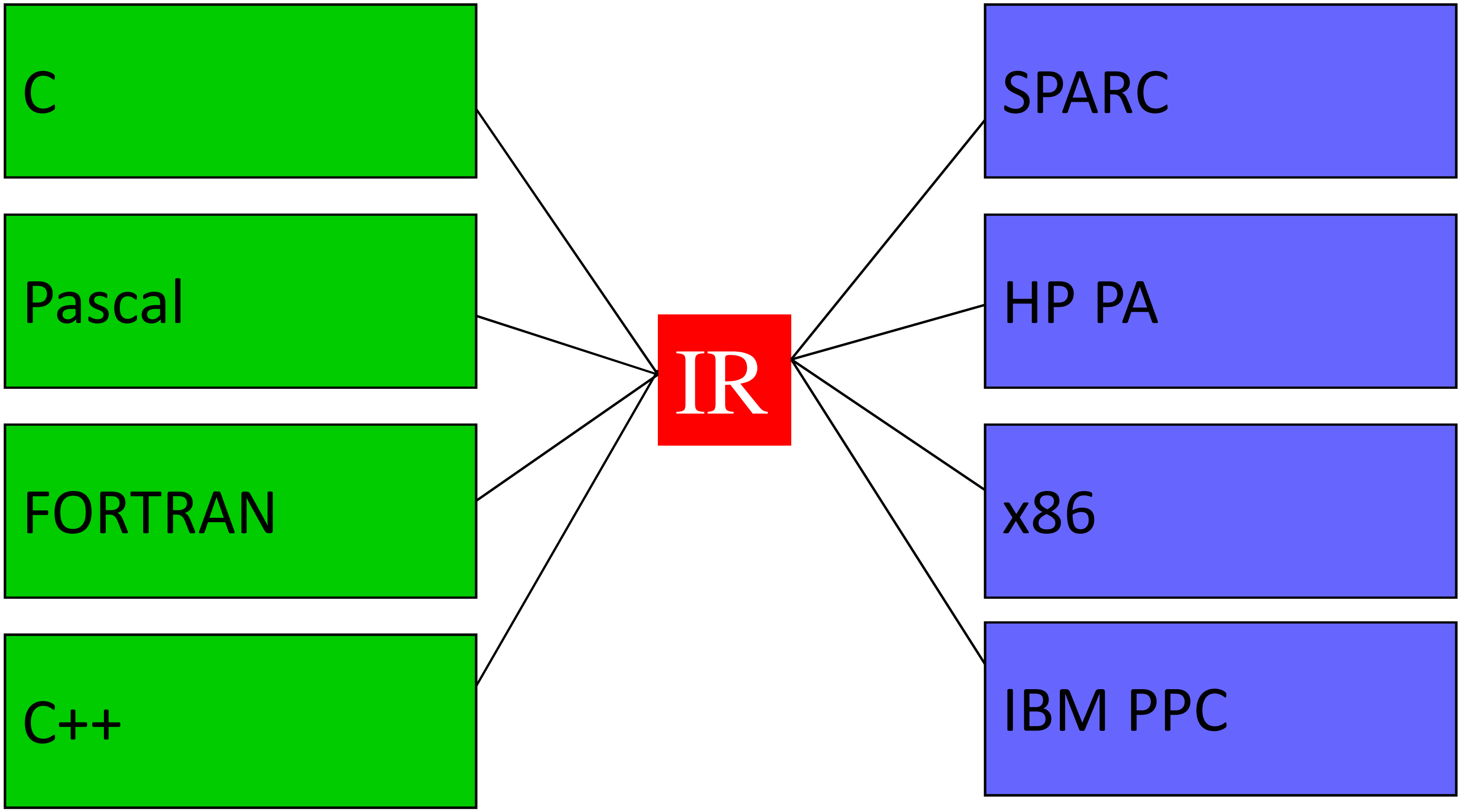
- A kind of abstract machine language
- that can express the target machine operations
- without committing to too much machine details.

Why IR ?

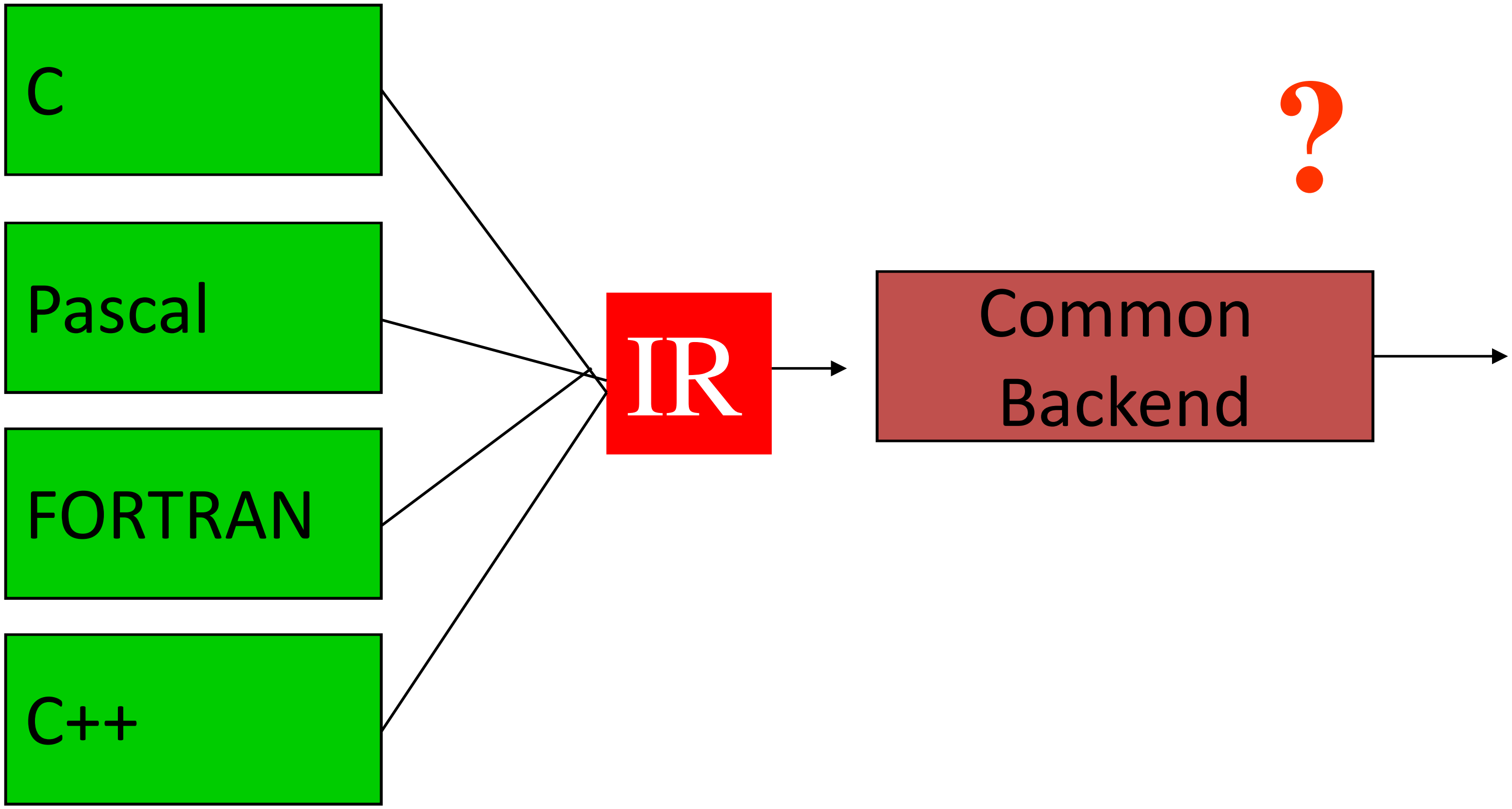
Without IR



With IR



With IR



Intermediate Representations

- Intermediate representations span the gap between the source and target languages:
  - closer to target language;
  - (more or less) machine independent;
  - allows many optimizations to be done in a machine-independent way.
- Implementable via syntax directed translation, so can be folded into the parsing process.

Types of Intermediate Languages

- **<u>High Level IR </u>(**e.g., AST):
  - closer to the source language
  - easy to generate from an input program
  - code optimizations may not be straightforward.
- Low Level IR (e.g., 3-address code, RTL):
  - closer to the target machine;
  - easier for optimizations, final code generation;

Advantages of Using an Intermediate Language

- ***Retargeting*** –
  - Build a compiler for a new machine
  - By attaching a new code generator to an existing front-end.
- ***Optimization*** –
  - Reuse intermediate code optimizers in compilers
  - for different languages and different machines.

Note: the terms "intermediate code", "intermediate language", and "intermediate representation" are all used interchangeably.

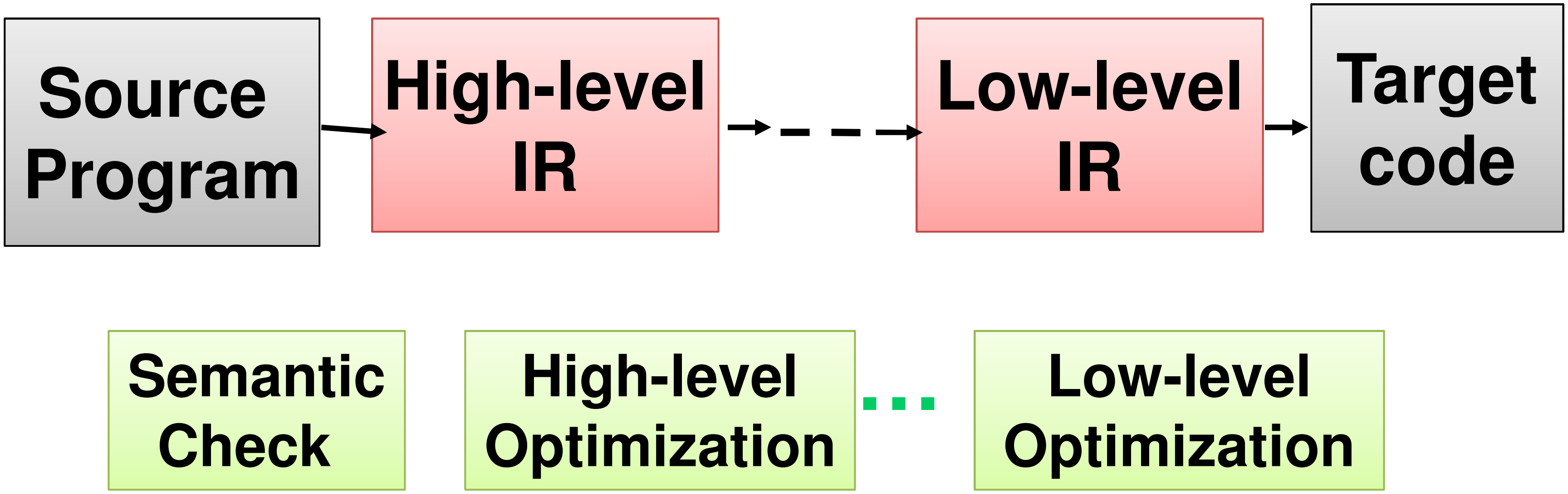
Issues in Designing an IR

- **Whether to use an existing IR**
  - if target machine architecture is similar
  - if the new language is similar
- **Whether the IR is appropriate for the kind of optimizations to be performed**
  - e.g. speculation and predication
  - some transformations may take much longer than they would on a different IR

Issues in Designing an IR

- **Designing a new IR needs to consider**
  - Level (how machine dependent it is)
  - Structure
  - Expressiveness
  - Appropriateness for general and special optimizations
  - Appropriateness for code generation
  - Whether multiple IRs should be used

Multiple-Level IR



Using Multiple-level IR

- Translating from one level to another in the compilation process
  - Preserving an existing technology investment
  - Some representations may be more appropriate for a particular task.

Commonly Used IR

- Possible IR forms
  - Graphical representations: such as syntax trees, AST (Abstract Syntax Trees), DAG
  - Postfix notation
  - Three address code
  - SSA (Static Single Assignment) form
- IR should have individual components that describe simple things

Thanks