Welcome to CSE 310
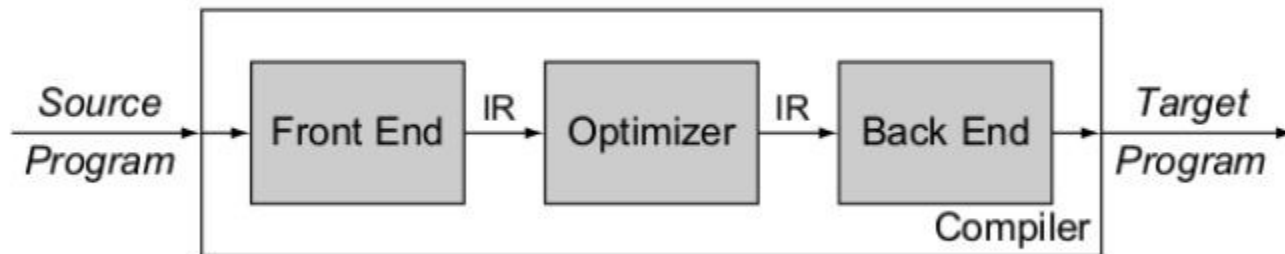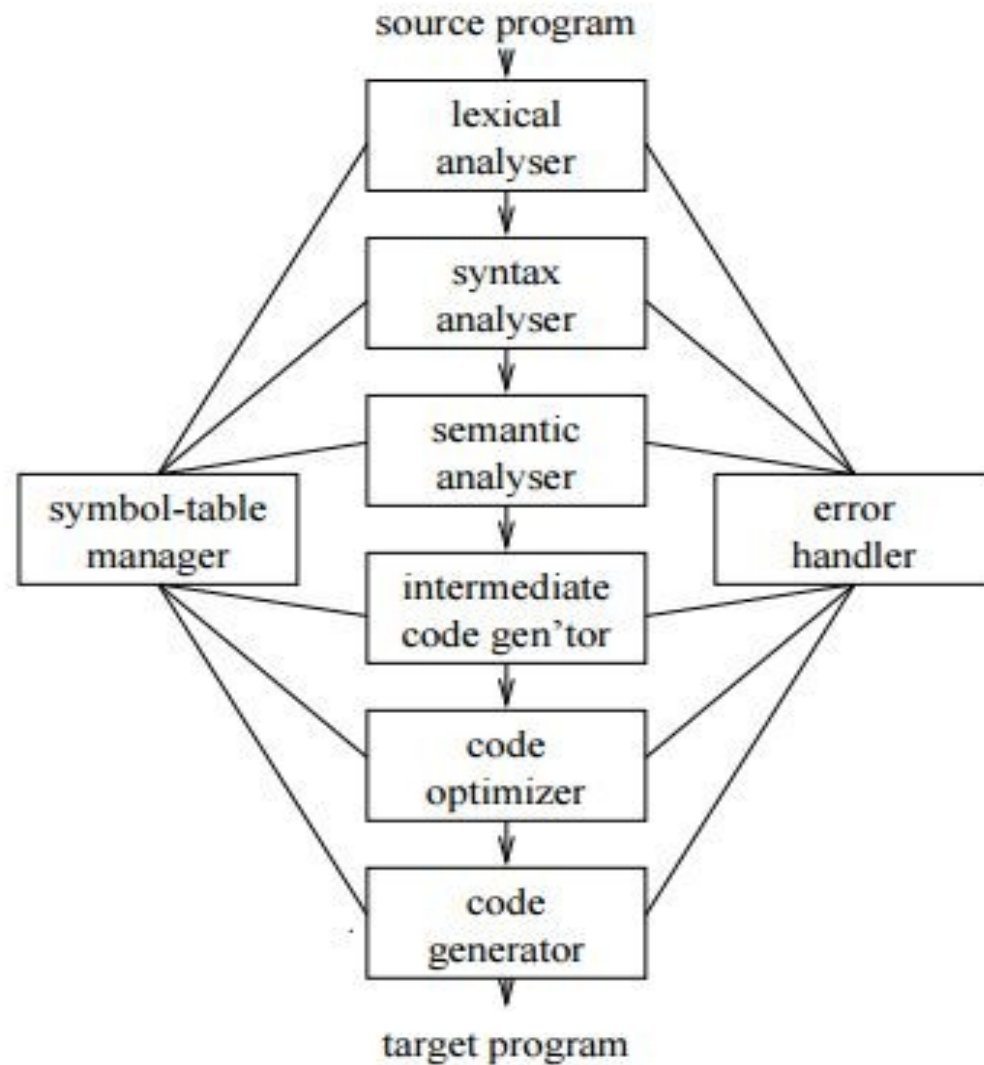
Compiler

- Convert one source program to a target program

- The compilation process usually divided int

Compiler

Dept of CSE, BUET

What will we do in this course?

- Construct and manage symbol table

- Perform lexical analysis using flex

- Perform syntax analysis, semantic analysis and intermediate code generation using bison

- May be some code optimization too

- So… We are going to build a COMPILER!

Some Info

- Linux platform

- No plagiarism

Symbol Table

- A table storing information of occurrence of various entities in the source program

- Information are:
  - Symbol Name
  - Type
  - Scope
  - Value

- Used in almost all phases of a compiler

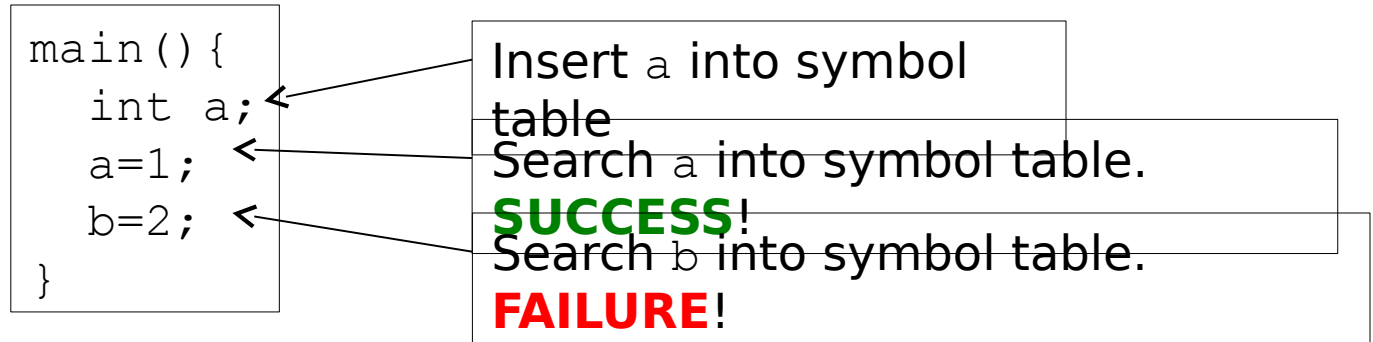# Offline 1: Symbol Table Management

- Implement a simple symbol table

- Hash based (Chaining)

- Each entry is a two tuple <Symbol Name, Symbol Type>

- Use Symbol Name as key of hash table

Offline 1: Symbol Table Management
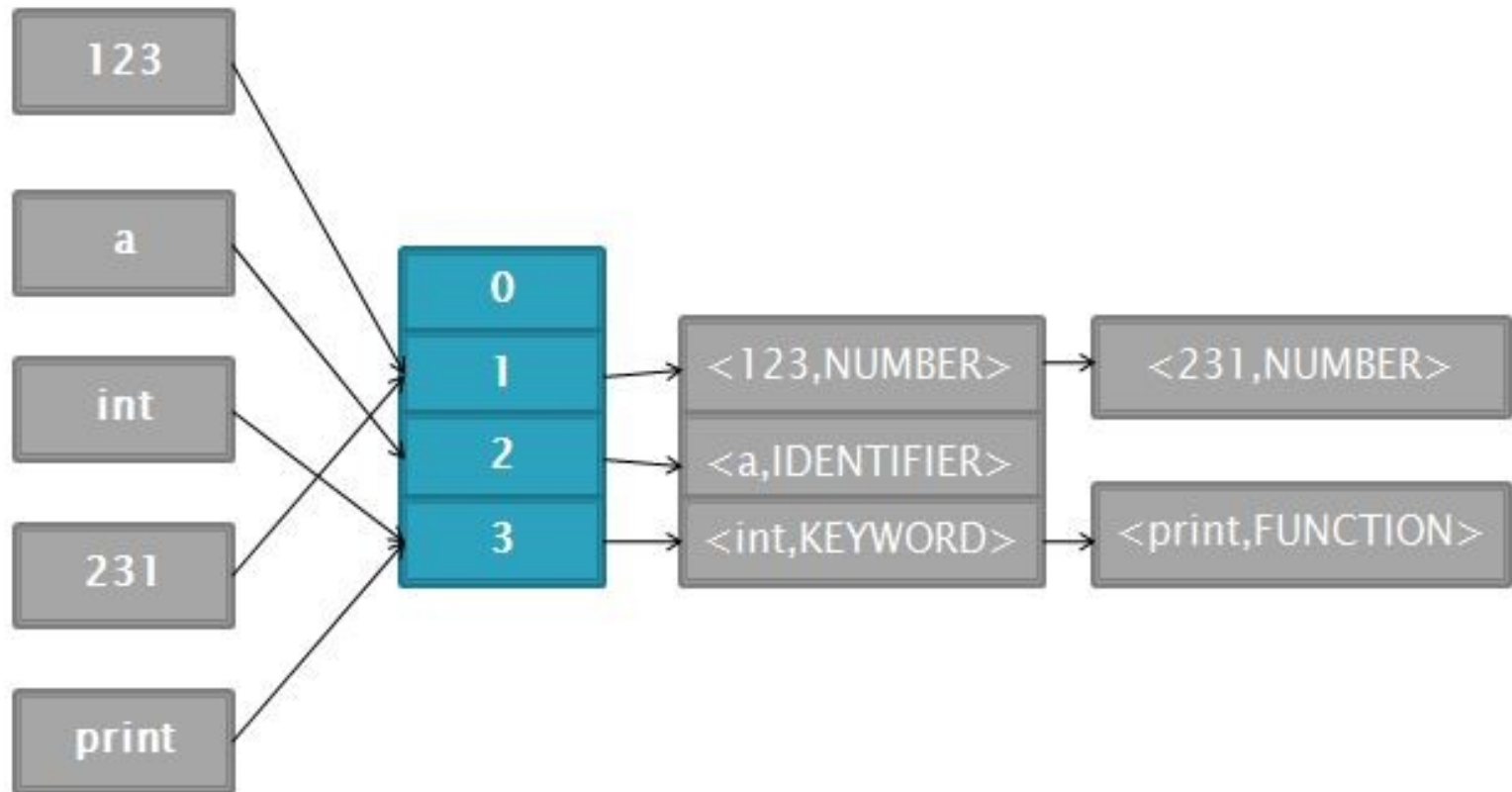
How Symbol Table Helps?

- How can this type of Symbol Table help?
  - Detect undeclared variable

```
main(){
   int a;
   a=1;
   b=2;
}
```

Insert `a` into symbol table

Search `a` into symbol table. **SUCCESS**!

Search `b` into symbol table. **FAILURE**!

  - Type checking
    - Add an extra field for each symbol named **datatype**
    - During an assignment operation check datatype field of RHS and LHS

How Symbol Table Helps?

- How can this type of Symbol Table help?
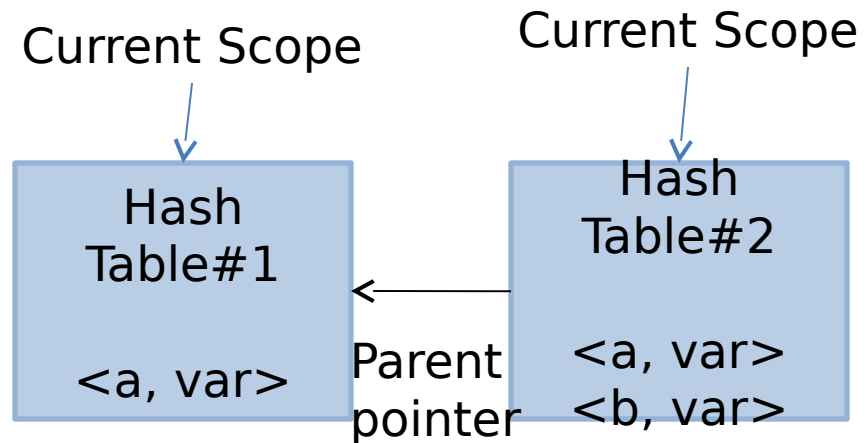 - Scope Management

```
main(){
    int a;
    {
        int a,b;
    }
    b=2;
}
```

- Need to allow duplicate entry in symbol table
- Also delete some entries when a block exits
- How to accommodate this??

Symbol Table for Scope Management

- List of Hash Tables

```
main(){
  int a;
  {
      int a,b;
  }
  b=2;
}
```

Current Scope

Current Scope

Hash Table#1

<a, var>

Hash Table#2

<a, var>
<b, var>

Parent pointer

Offline 1: Symbol Table Management

- Three Classes
  1. SymbolInfo
     - Each entry of symbol table is an instance of SymbolInfo. (Remember two tuples!)

Offline 1: Symbol Table Management

- Three Classes
  2. ScopeTable
     - This class is the implementation of a hash table.
     - Represents each scope
     - Implement four operations
       » Insert
       » Lookup
       » Delete
       » Print

Offline 1: Symbol Table Management

- Three Classes
  3. SymbolTable
     - Maintain a list of ScopeTables
     - Implement four operations
       » Enter Scope
       » Exit Scope
       » Insert
       » Delete
       » Look up
       » Print All Tables
       » Print Current Table