# Overview of Compilation

Daqing Hou

# Outline

- Definition of symbols
  variables
  types
  functions
- Reference to symbols & expressions
  scopes & scoping rules
  symbol tables
- Structure of a compiler
  lexical analysis: tokens
  syntactical analysis: ASTs (Abstract Syntax Tree)
- What can be done with lexical and syntactical information?

# Abstract Syntax Trees

- Concrete syntax / grammars are structural / syntactical rules a valid program follows
- Parsers use grammars to `split' a program into `parts'
- In additional to grammars, there are other rules a valid program must follow, e.g.,
  definition before use
  type rules, a.k.a., semantics rules
  initialization before use of a variable
- Different from grammars, ASTs capture the `essence' of program structures

# Program units, scopes, symbol tables

- A program is organized as a set of, possibly nested, program units
- A program unit may define new symbols
- A symbol can be either a type, a function, or a variable
- The scope of a symbol is the program text where it can be used
- Scoping rules govern how a particular kind of symbols are used
- Symbol tables are used to enforce scoping rules
- A stack versus a tree view of symbol tables

# Example: find()

```
int find(int a[], int N, int f)
{
int m=0; int n=N-1;
while (m<n) {
  int i=m; int j=n; int r = a[f];
   while (i<=j) {
     while (a[i]<r) ++i;
     while (a[j]>r) --j;
     if (i<=j) {
         swap(a[i], a[j]); ++i; --j;}
   }
   if (f<=j) n=j;
   else if (f>=i) m=i;
   else break;
 }
 return a[f];
}
```

# `optimization'

```
100    int a, b, c, d;
101    c = a + b;
102    d = c+1;
```

C code

```
100    ldw a, r1
101    ldw b, r2
102    add r1, r2, r3
103    stw r3, c

104    ldw c, r3
105    add r3, 1, r4
106    stw r4, d
```

Naïve SPARC code

```
100    add r1, r2, r3
101    add r3, 1, r4
```

Optimized code

# Tools that use lexical and syntax

- Checkstyle
  http://checkstyle.sourceforge.net/
- PMD
  http://pmd.sourceforge.net/

Install & play with either or both tools and discuss what info would be needed in order to support the checking offered.

# Reference texts (optional)

Any of the following has chapters on optimization

- Compilers: Principles, techniques, and tools by Aho, Sethi, Ullman
- Advanced compiler design & implementation by Steven S. Muchnick, ISBN 1-55860-320-4