CS536
Abstract Syntax Tree (AST) and Syntax Directed Translation (SDT)

A Sahu

CSE, IIT Guwahati

http://jatinga.iitg.ac.in/~asahu/cs536/

Outline

- Basic of AST
- **Basic of Syntax Directed Translation**
- Intermediate Representation

http://jatinga.iitg.ac.in/~asahu/cs536/

Doubt of Last Class

```
main(){
 char *A=“H” ;
 printf(“%s”,A);
}
```

Text	Data	Bss
1578	600	8

```
main(){
 char *A=“Hello I am Mr Aryabartta Sahu, of Dept CSE,
              IIT Guwahati, Assam, India, Pin-768037 ” ;
 printf(“%s”,A);
}
```

Text	Data	Bss
1673	600	8

LLVM Installation

- $sudo apt-get install cmake clang swapsapce
- Download llvm-project-llvmorg-12.0.0.tar.gz (or recent version) from LLVM official website "Source code (tar.gz)"
- $mkdir ~/LLVM; cd ~/LLVM; mkdir install; mv ~/Download/ll*tgz . ; tar –xvzf ll*tgz
- $cd llvm-project-llvmorg-12.0.0/
- $mkdir build; cd build
- $cmake --clean-first -G "Unix Makefiles" DLLVM_PARALLEL_LINK_JOBS=1 -DCMAKE_INSTALL_PREFIX=**$(HOME)**/LLVM/install/ ../llvm
- $make –j1 **//This may take some hours**
- $make install

Test Installation of LLVM

- How test a in built pass/transform
- cd $(HOME)/LLVM/llvm-project-llvmorg-12.0.0/llvm/test/Transforms/HelloNew

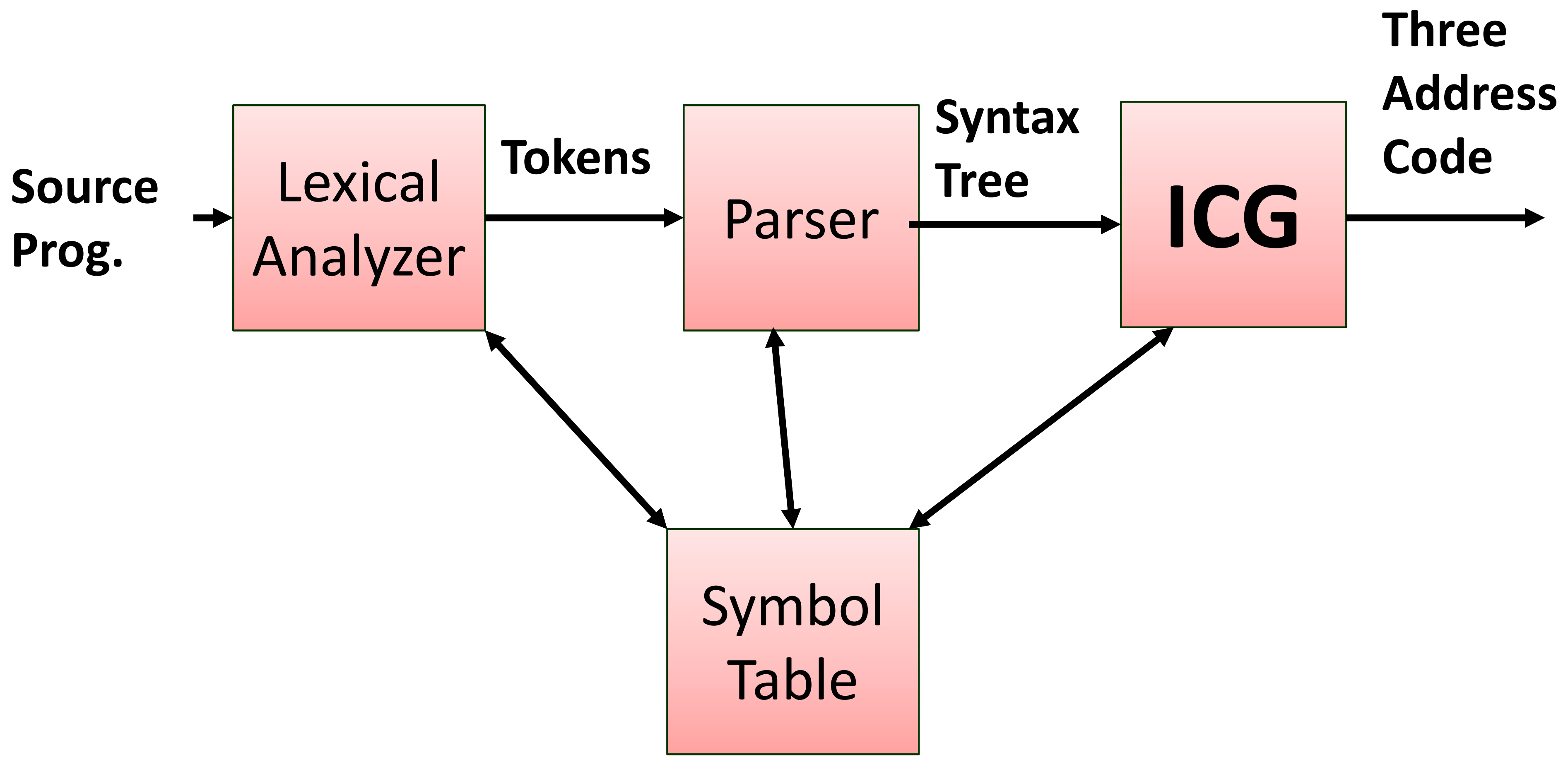
- $$(HOME)/LLVM/install/bin/opt -disable-output -passes=helloworld helloworld.ll

or

- $$(HOME)/LLVM/llvm-project-llvmorg-12.0.0/build/bin/opt -disable-output -passes=helloworld helloworld.ll

Basic of
Syntax Directed Translation

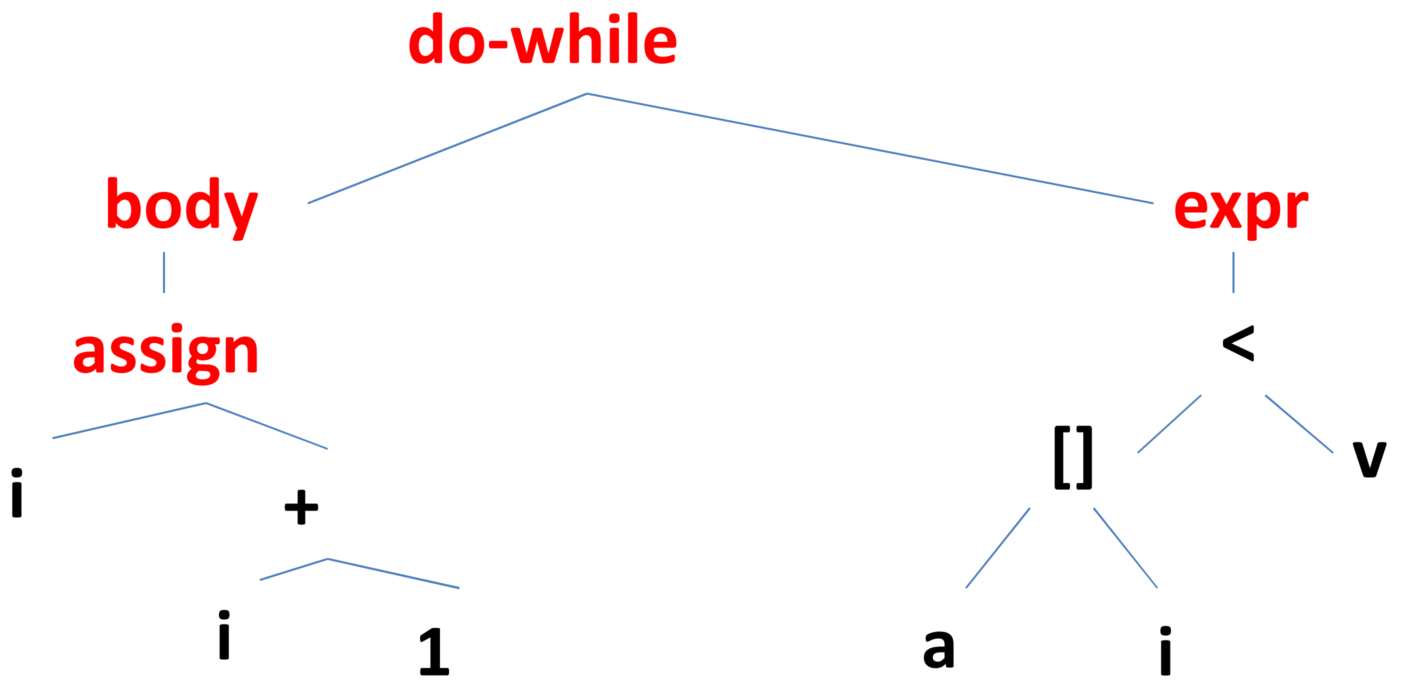
A model of Compiler Front end



ICG: Intermediate Code Generator

Abstract Syntax Tree: AST

- Syntax tree: hierarchical syntactic structure of the source program
- AST for : `do i=i+1; while (a[i]<v);`



Syntax Definition

- A grammar naturally describe the hierarchical structure of the most program

if (expression) statement **else** statement

- Production rule : Can have the form

stmt $\rightarrow$ **if** (expr) stmt **else** stmt

- In a production: if, else, (,) are terminals
  - The term expr, stmt are non-terminal
  - Can have the form (again)

Definition of Context Free Grammar

- **CFG has four components**
- A set of terminal symbols (referred as token)
  - Elementary Symbols of the Grammar
- S set of non-terminals (NT/syntactic variables)
- A set of production rules
  - Each production of NT called head/left side
  - Arrow and a sequence of T and/or NT called body/right side production
- A designation of one NT as ***Start symbol***

Production Example

Expression : list of digits separated by plus and minus signs

list → list + digit
list→ list – digit
list→digit
digit→0|1|2|3|4|5|6|7|8|9

list → list + digit | list – digit | digit
digit→0|1|2|3|4|5|6|7|8|9

Derivation

- A grammar derives strings by beginning with start symbols
- And repeated replacing a non terminals by body of the production for that non terminals
- The terminal strings can be derived from the start symbol

```
9-5+2 is list, Can be derived as follows
list → list +2          // list →list + digit
     → list -5 + 2    // list →list  - digit
     → 9 – 5 -2       // list → digit
```

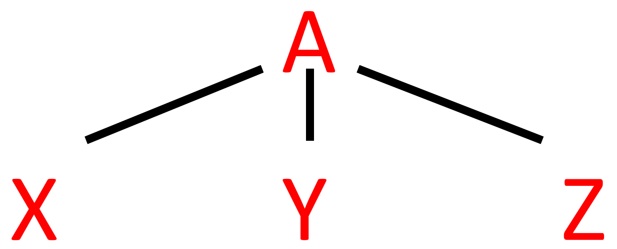
Another Production Example

call → id (optparams)
optparams→ params | ϵ
params → params, param | param

- The term ϵ specifies the empty string
- This analogous/similar to earlier production

Parsing Trees

- A parse tree pictorially shows: How the start symbol of a grammar derives strings in language



1. The root is labeled by start symbol
2. Each leaf is labeled by a terminal or by ϵ
3. Each interior node is labeled by a non-terminals
4. If NT A $X_{1,}$ $X_{2,}$… , X_n are labeled children of A from left to right then there must be production A→$X_{1,}$ $X_{2,}$ …,$X_{n,}$ where X1, X2, ..Xn are eithe NT or T,
5. If A → ϵ , then A may have single child ϵ

Parsing Trees: Properties

- A tree consists of one or more nodes
- **Exactly one root node in a Tree**
  - Root have no-parent, it is top node
  - Other node have exactly one parent
- Leaf: node with no children
- N is parent of M, M is child of N, Children of one node is Siblings, Ordered from left to right
- Descendent (self, child*), Ancestor (self, parent*)

Thanks