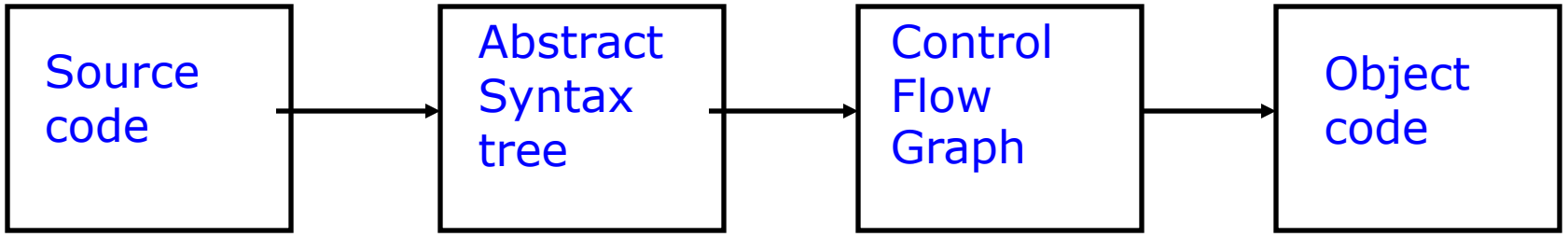
B.S. Abdur Rahman
Crescent
Institute of Science & Technology
Deemed to be University u/s 3 of the UGC Act, 1956
GST Road, Vandalur, Chennai 600 048

CSD 3202-COMPILER DESIGN

MODULE IV

FLOW GRAPHS

DATA FLOW ANALYSIS - Compiler Structure



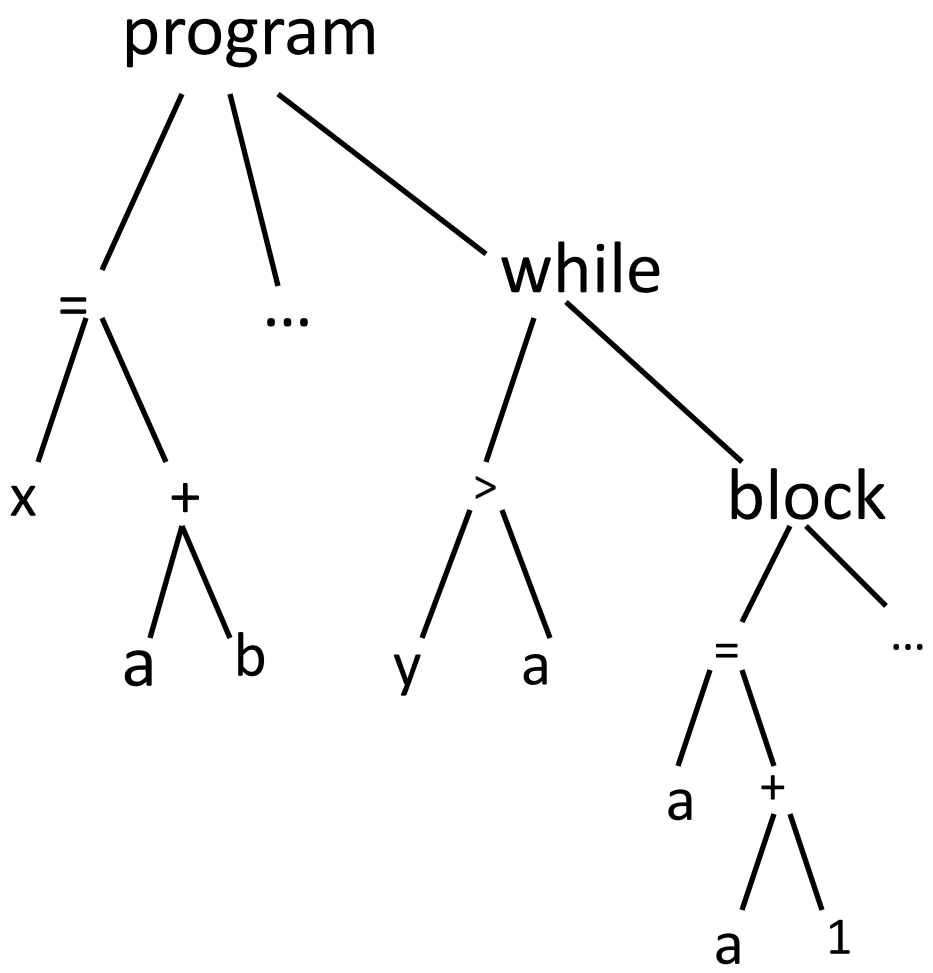
- Source code parsed to produce abstract syntax tree.
- Abstract syntax tree transformed to *control flow graph*.
- *Data flow analysis* operates on the control flow graph (and other intermediate representations).

Abstract Syntax Tree (AST)

- Programs are written in text
  - as sequences of characters
  - may be awkward to work with.
- First step: Convert to structured representation.
  - Use lexer (like lex) to recognize tokens
  - Use parser (like yacc) to group tokens structurally
    - produce AST

Abstract Syntax Tree Example

```
x := a + b;
y := a * b
While (y > a)
{
   a := a +1;
   x := a + b
}
```



ASTs

- ASTs are abstract
  - don't contain all information in the program
    - e.g., spacing, comments, brackets, parenthesis.
  - Any ambiguity has been resolved

    e.g., a + b + c produces the same AST as

    (a +b) + c.

Disadvantages of ASTs

- ASTs have many similar forms
  - e.g., for while, repeat , until, etc
  - e.g., if, ?, switch
- Expressions in AST may be complex, nested
- (42 * y) + (z > 5 ? 12 * z : z +20)
- Want simpler representation for analysis
  - … at least for dataflow analysis.

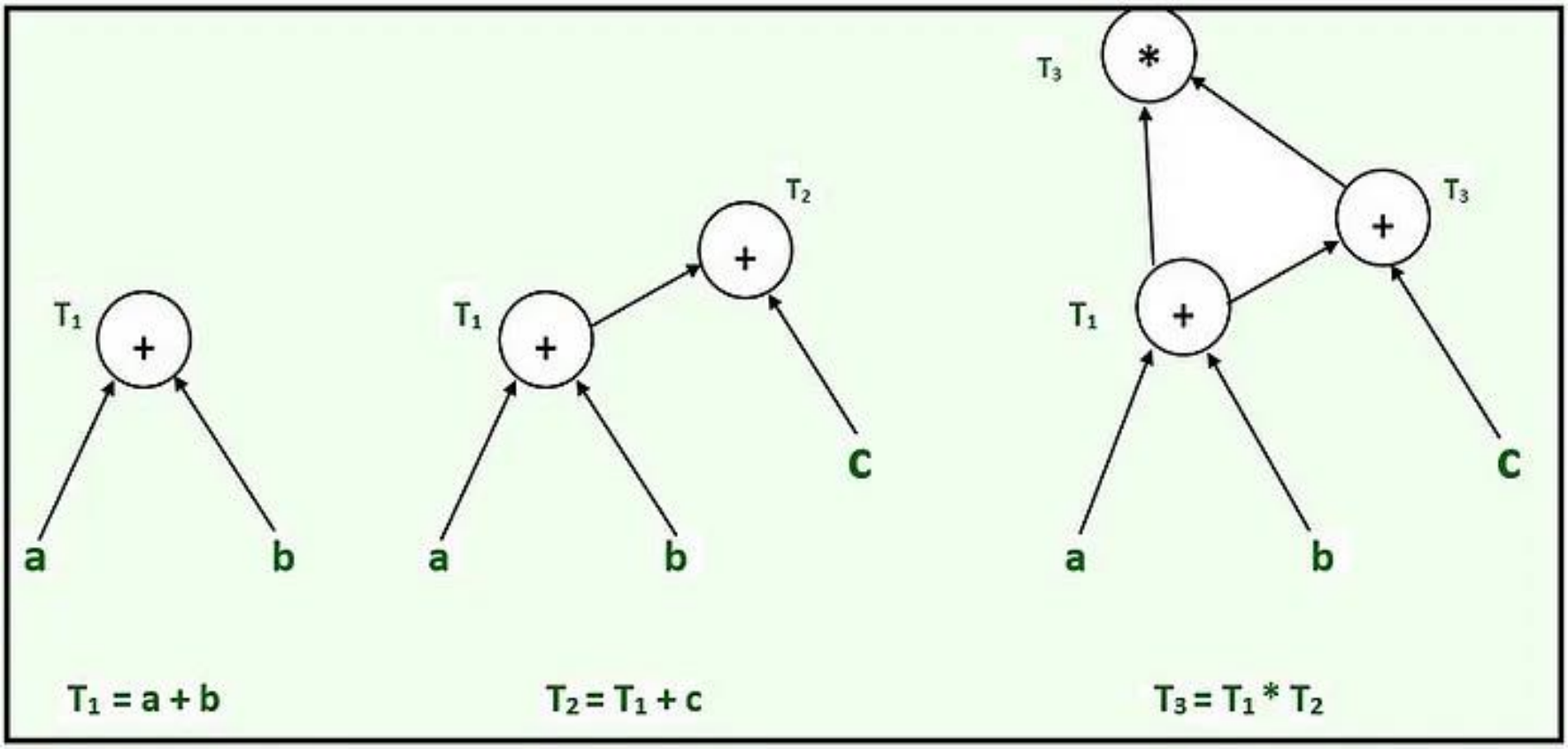
Directed Acyclic Graphs (DAG)

- A Directed Acyclic Graph (DAG) is a directed graph that contains nodes connected by edges, with the property that the graph contains no directed cycles.
- It is commonly used to represent the control flow and data dependencies of a program. This representation is often used as an intermediate representation (IR) that facilitates program optimization and transformation.
- When a program is compiled, it goes through several stages, including lexical analysis, parsing, semantic analysis, and code generation.
- During these stages, the compiler constructs a DAG representation of the program that captures the control flow and data dependencies between its components.

Directed Acyclic Graphs (DAG)

- Use of DAGs in this context enables the compiler to perform optimizations such as dead code elimination and loop unrolling, common sub-expression elimination and constant folding, which can significantly improve the performance of the generated code.
- DAGs in compiler design is to represent the control flow of a program in which each node represents a basic block of code, and each edge represents a control flow transfer between basic blocks.

Directed Acyclic Graphs (DAG)



Control-Flow Graph (CFG)

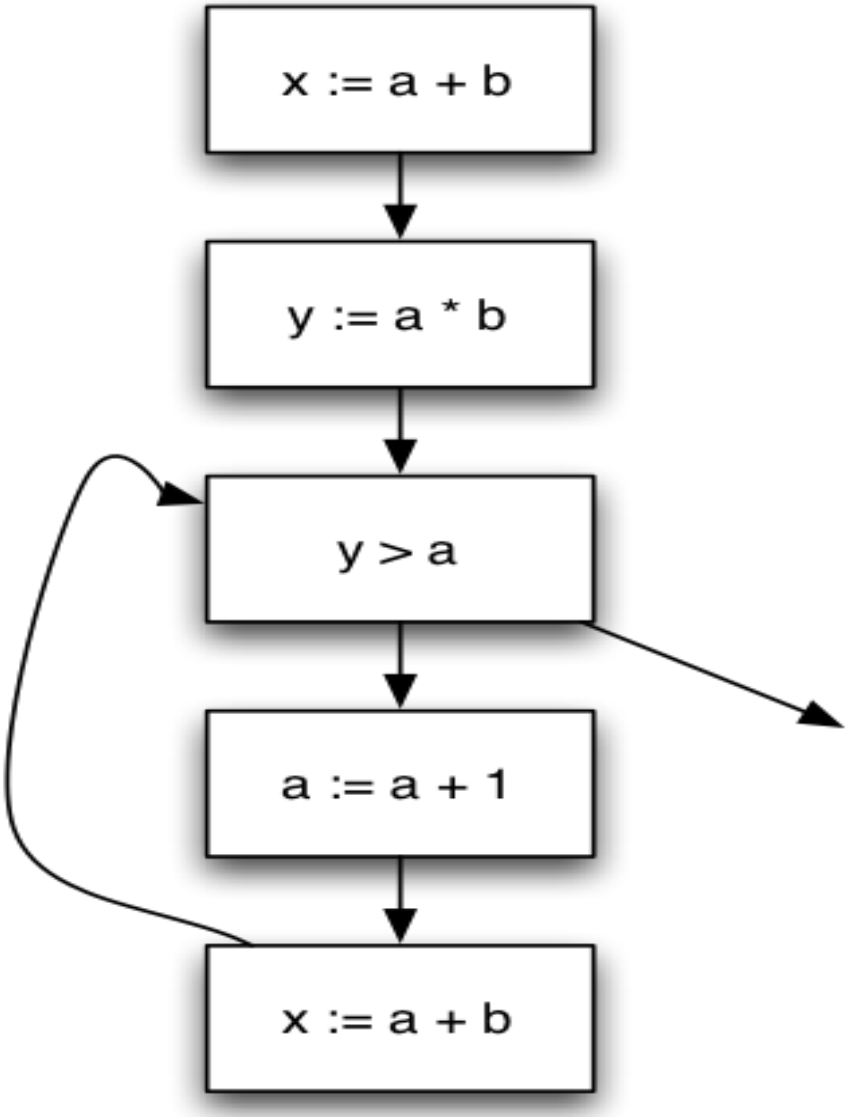
- A directed graph where
  - Each node represents a statement
  - Edges represent control flow
- Statements may be
  - Assignments x = y op z or x = op z
  - Copy statements x = y
  - Branches goto L or if relop y goto L
  - etc
- Partition the intermediate code into basic blocks.
- The basic blocks become the nodes of a flow graph.

Control-Flow Graph (CFG)

- Flow graph is a directed graph.
- It contains the flow of control information for the set of basic block.
- A control flow graph is used to depict that how the program control is being parsed among the blocks.

Control-flow Graph Example

```
x := a + b;
y := a * b
While (y > a)
{
   a := a +1;
   x := a + b
}
```

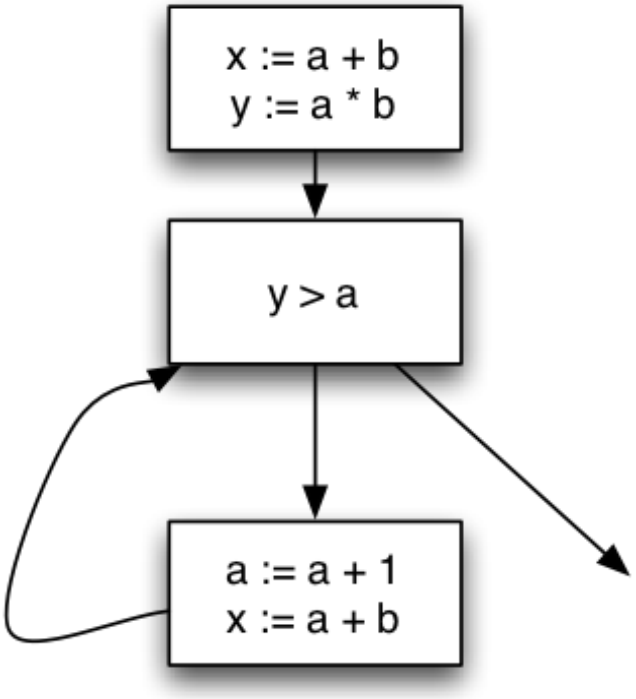


Variations on CFGs

- Usually don't include declarations (e.g. int x;).
- May want a unique entry and exit point.
- May group statements into *basic blocks.*
  - A *basic block* is a sequence of instructions with no branches into or out of the block.

Control-Flow Graph with Basic Blocks

```
X := a + b;
Y := a * b
While (y > a)
{
   a := a +1;
   x := a + b
}
```



- Can lead to more efficient implementations
- But more complicated to explain so…
  - We will use single-statement blocks in lecture

CFG vs. AST

- CFGs are much simpler than ASTs
  - Fewer forms, less redundancy, only simple expressions
- But, ASTs are a more faithful representation
  - CFGs introduce temporaries
  - Lose block structure of program