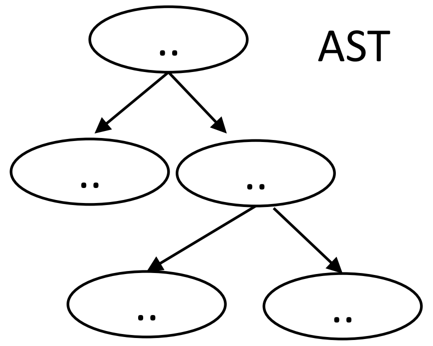
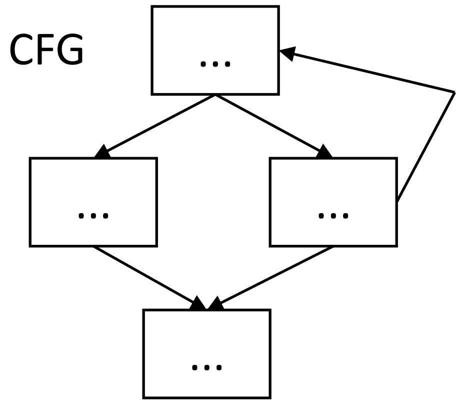
CSE110A: Compilers

MIDTERM REVIEW

PostOrder Traversal Example





3 address code

```
store i32 0, ptr %2
%3 = load i32, ptr %1
%4 = add nsw i32 %3, 1,
store i32 %4, ptr %1
%5 = load i32, ptr %2
```

Postfix Order Traversal : An Example

When compilers parse arithmetic or logical expressions, they often build a **parse tree** or **abstract syntax tree (AST)** representing the expression's structure.

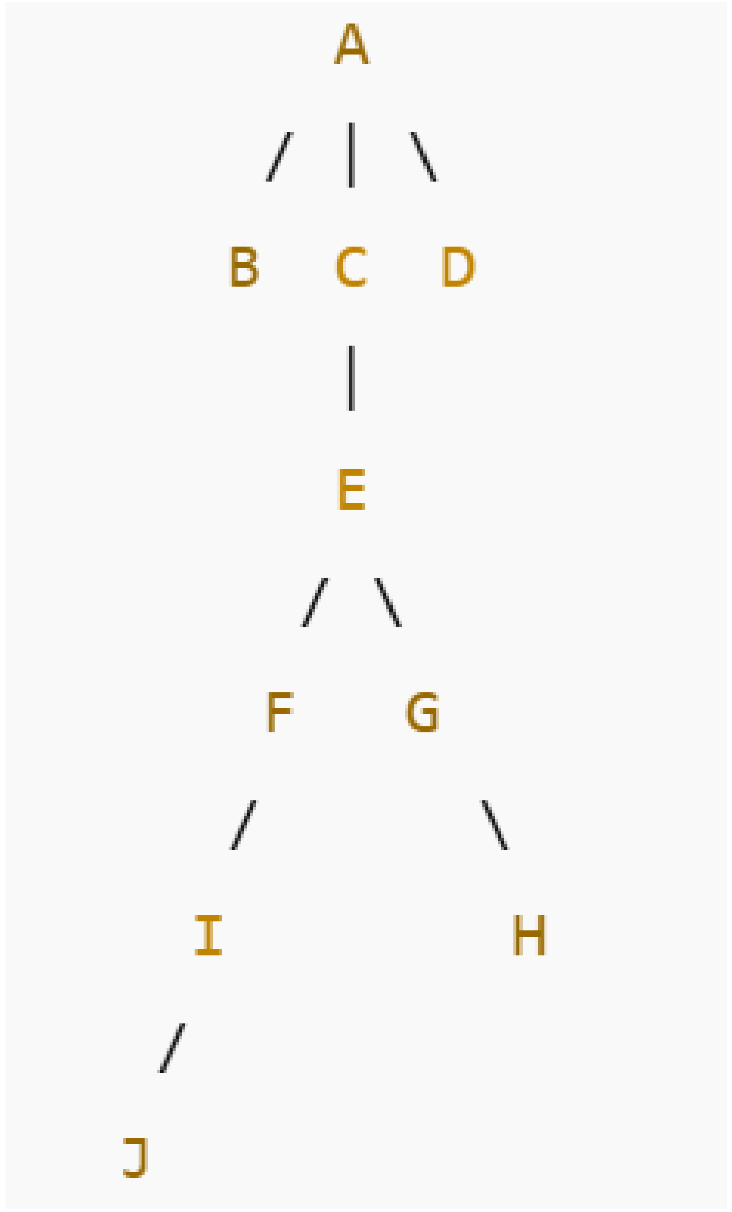
Postfix order traversal of this tree naturally matches how expressions can be **evaluated using a stack-based approach** — like in a **stack machine** or when generating **Reverse Polish Notation (RPN)**.

Postfix Order Traversal : An Example

We will be making extensive use of Postfix Order Traversal.

Here is an example which should help give you get a good feeling for how Postfix Order Traversal works.

Postfix Order Traversal : An Example



Visit children
bottom-up
left to right,
then parent)

- ✅ **"Visit children"** → recursive descent into child nodes
- ✅ **"bottom-up"** → children are fully visited before their parent
- ✅ **"left to right“** → siblings are visited in left-to-right order
- ✅ **"then parent"** → the parent node is visited last

B → J → I → F → H → G → E → C → D → A

Postfix Order Traversal : An Example

```
      A
    / | \
   B  C  D
      |
      E
     / \
    F   G
   /      \
  I        H
 /
J
```

Traversal:

B →

J → I → F →

H → G →

E → C →

D →

A

Visit children

bottom-up

left to right,

then parent)