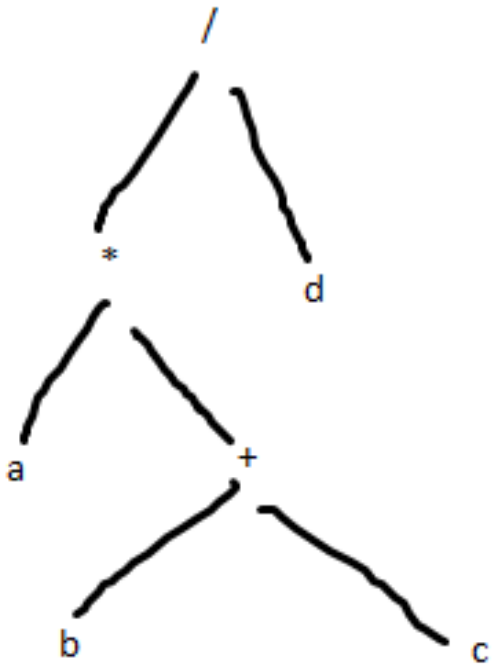
Implementation of SDT

Parse trees and Syntax Trees

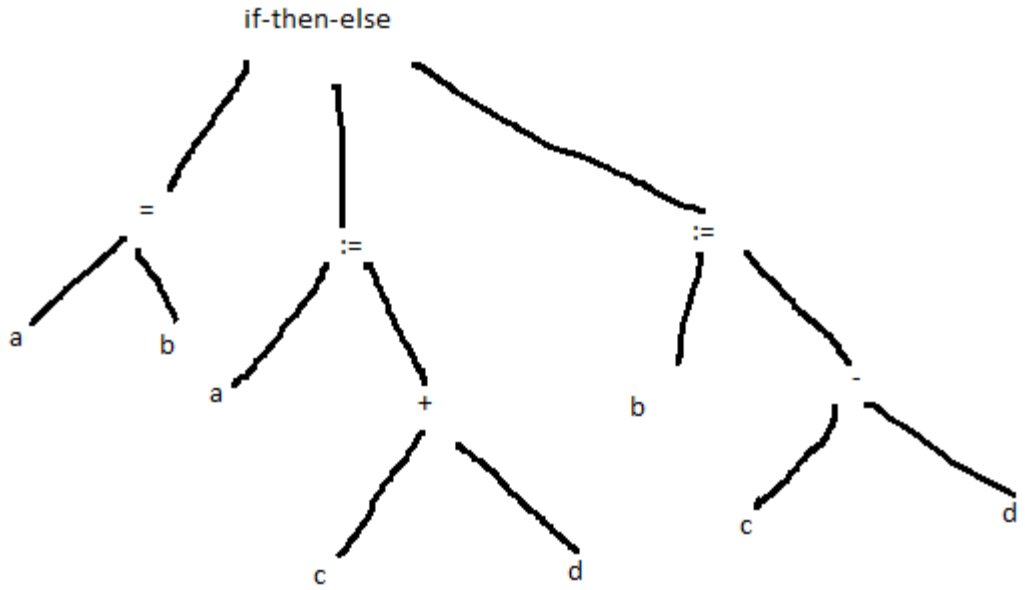
- The parse tree itself is a useful intermediate-language representation for a source program
- Syntax tree
  - A parse tree in which each leaf represents an operand and each interior node an operator

Example of syntax trees

a*(b+c)/d



If a=b **then** a:=c+d **else** b:=c-d



SDT scheme to construct syntax trees

Production	Semantic Action
E → E1 op E2	{ E.VAL := NODE(op,E1.VAL,E2.VAL) }
E → (E1)	{ E.VAL := E1.VAL }
E → -E1	{ E.VAL := UNARY(-,E1.VAL) }
E→ id	{ E.VAL := LEAF(id) }

The NODE function takes three arguments

NODE(OP, LEFT, RIGHT) : OP name of operator, LEFT : pointer to root of left subtree, RIGHT : pointer to root of right subtree

UNARY(OP, CHILD) : OP name of operator, CHILD : pointer to root of the child node

LEAF(id) : creates a new node labeled by id.

All methods returns a pointer to new node created.

Three-Address Code

- Sequence of statements of the form A:= B op C where A, B, C are either programmer defined names or compiler generated temporary names.
- Contains three addresses – one for result and two for operands.
- Example

  X + Y * Z can be expressed in three address statement as

  T1 := Y * Z

  T2 := X + T1

Common types of Three Address Statements

1. Assignment
   - A := B op C, where op is a binary operator.
   - A := op B, where op is a unary operator
   - A := B
2. Unconditional jump
   - Goto L, execute the Lth three address instruction.
3. Conditional Jumps
   - If A relop B goto L, relop is a relational operator

Common types of Three Address Statements

4. Subroutine call
   - **param** A and **call** P, n, equivalent to call P(A1,A2,…An)

     Param A1
     Param A2
     …..
     Param n
     Call P, n
5. Indexed assignment
   - A:=B[i]
   - A[i]:=B
6. Address and Pointer assignment
   - A := addr B, A=*B, *A=B

Quadruples

- Quadruples- A record structure of four fields is used as a representation of three-address statement
- Four field structure – OP, ARG1, ARG2, RESULT
- OP contains an internal code for operator
- Can have different codes for fixed or floating point operation or some other form of the operator.
- Example
  - A := -B *(C+D)
  - Three Address

    T1:=-B

    T2:=C+D

    T3:=T1*T2

    A:=T3

Quadruple

	OP	**ARG1**	**ARG2**	**RESULT**
(0)	uminus	B		T1
(1)	+	C	D	T2
(2)	*	T1	T2	T3
(3)	:=	T3		A

Triples

- Statement computing temporary value represents the temporary value.
- Three field structure – OP, ARG1, ARG2
- ARG1 and ARG2 can be pointers to symbol table entry or to the structure itself.
- Triple

	OP	ARG1	ARG2
(0)	uminus	B	
(1)	+	C	D
(2)	*	(0)	(1)
(3)	:=	A	(2)

Indirect Triples

- Another implementation of three address code is that of listing pointers to triples instead of triples themselves –Indirect triples
- Indirect triples representation of three-address statements

	OP	ARG1	ARG2
(14)	uminus	B	
(15)	+	C	D
(16)	*	(14)	(15)
(17)	:=	A	(16)

Triple

	STATEMENTS
(0)	(14)
(1)	(15)
(2)	(16)
(3)	(17)

Indirect Triple

Comparison of the three methods

- In quadruples, the temporary variables are stored in the symbol table while in triple they are not stored.
- Assignment of memory location in triples deferred till code generation.
- Quadruples more useful in optimizing compilers where interchanging of instruction is done.
- Change in the position of instructions in triples causes change in all fields which refer to that instruction.
- No such problem in indirect triple.

problems

- Convert the following into three address code and implement using quadruples and triples
  1. A:=(B+C) * (D+E)
  2. B:=(A+C) + F