Assignment No.12

AIM:

Generation of 3- address code for language.

THEORY:

3 address code with example

Three address code is a sequence of statements of the general form

x := y op z

where x, y, and z are names, constants, or compiler-generated temporaries; op stands for any operator, such as a fixed or floating-point arithmetic operator, or a logical operator on boolean-valued data. No built-up arithmetic expressions are permitted, as there is only one operator on the right side of a statement. Thus a source language expression like x+ y * z might be translated into a sequence

t1 := y * z

t2 :=x + t1

where tl and t2 are compiler-generated temporary names.

Types of 3 address code

The common three-address statements are:

1. Assignment statements of the form *x : = y op z*, where op is a binary arithmetic or logical operation.
2. Assignment instructions of the form *x : = op y*, where op is a unary operation. Essential unary operations include unary minus, logical negation, shift operators, and conversion operators that, for example, convert a fixed-point number to a floating-point number.
3. Copy statements of the form *x : = y* where the value of y is assigned to x.
4. The unconditional jump *goto L*. The three-address statement with label L is the next to be executed.

5. Conditional jumps such as *if x relop y goto L*. This instruction applies a relational operator (<, =, >=, etc.) to x and y, and executes the statement with label L next if x stands in relation relop to y. If not, the three-address statement following if x relop y goto L is executed next, as in the usual sequence.
6. *param x* and *call p, n* for procedure calls and *return y*, where y representing a returned value is optional. For example,

```
param x1
param x2
. . .
param xn
call p,n
```

   generated as part of a call of the procedure p(x1, x2, …. ,xn).

7. Indexed assignments of the form *x : = y[i]* and *x[i] : = y*. The statement *x : = y[i]* sets x to the value in the location i memory units beyond location y. The statement *x[i] : = y* sets the content of location i units beyond the x to the value y.
8. Address and pointer assignments of the form *x : = &y , x : = *y,* and **x : = y*. The statement *x : = &y* sets the value of x to be the location of y. In the statement *x : = *y,* presumably y is a pointer or a temporary whose r-value is a location. The statement **x : = y* sets the r-value of the oject pointed to be by x to the r-value of y.

Implementation of 3-address statements

- A three-address statement is an abstract form of intermediate code. In a compiler, these statements can be implemented as records with fields for the operator and the operands.

- Three such representations are:
  - ✓ Quadruples
  - ✓ Triples
  - ✓ Indirect triples

Indirect triples

1. Quadruples

- A quadruple is a record structure with four fields, which are, op, arg1, arg2 and result.
- The op field contains an internal code for the operator. The three-address statement x : = y op z is represented by placing y in arg1, z in arg2 and x in result.
- The contents of fields arg1, arg2 and result are normally pointers to the symbol-table entries for the names represented by these fields. If so, temporary names must be entered into the symbol table as they are created.
- Example - a : = b * - c + b * - c

	op	arg1	arg2	result
(0)	uminus	c		t1
(1)	*	b	t1	t2
(2)	uminus	c		t3
(3)	*	b	t3	t4
(4)	+	t2	t4	t5
(5)	: =	t3		a

2. Triples

- To avoid entering temporary names into the symbol table, we might refer to a temporary value by the position of the statement that computes it.
- If we do so, three-address statements can be represented by records with only three fields: op, arg1 and arg2.
- The fields arg1 and arg2, for the arguments of op, are either pointers to the symbol table or pointers into the triple structure (for temporary values).
- Since three fields are used, this intermediate code format is known as triples.
- Example - a : = b * - c + b * - c

	op	arg1	arg2
(0)	uminus	c	
(1)	*	b	(0)
(2)	uminus	c	
(3)	*	b	(2)
(4)	+	(1)	(3)
(5)	assign	a	(4)

3. Indirect triples

- Another implementation of three-address code is that of listing pointers to triples, rather than listing the triples themselves. This implementation is called indirect triples.
- For example, let us use an array statement to list pointers to triples in the desired order. Then the triples shown above might be represented as follows:

  a : = b * - c + b * - c

	statement		op	arg1	arg2
(0)	(14)	(14)	uminus	c	
(1)	(15)	(15)	*	b	(14)
(2)	(16)	(16)	uminus	c	
(3)	(17)	(17)	*	b	(16)
(4)	(18)	(18)	+	(15)	(17)
(5)	(19)	(19)	assign	a	(18)

INPUT:

Input is:

a-b+c*(d+c)

OUTPUT:

Output is:

t1=a-b

t2= d+c

t3= t1+t2

CONCLUSION:

- A three-address statement is an abstract form of intermediate code. In a compiler, these statements can be implemented as records with fields for the operator and the operands.
- Three address codes for the given expression is implemented.

REFERENCES:

- Compilers - Principles, Techniques and Tools - A.V. Aho, R. Shethi and J. D. Ullman (Pearson Education)