THREE ADDRESS CODE

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

x := y op z

where x, y, 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.

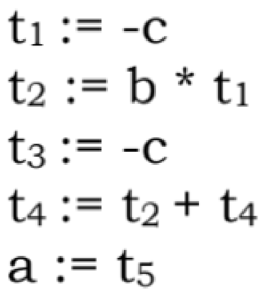
In three-address code, there is at most one operator on the right side of an instruction; that is, no built-up arithmetic expressions are permitted. Thus a source-language expression like x+y*z might be translated into the sequence of three-address instructions

t1 := y * z

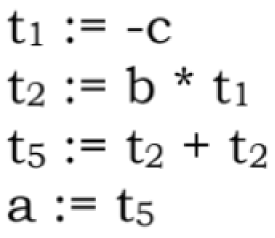
t2 := x + t1

where t1 and t2 are compiler-generated temporary names. The use of names for the intermediate values computed by a program allows three- address code to be easily rearranged unlike postfix notation.

Three-address code is a linearized representation of a syntax tree or a dag in which explicit names correspond to the interior nodes of the graph. The syntax tree and dag



```
t1 := -c
t2 := b * t1
t3 := -c
t4 := t2 + t4
a := t5
```



```
t1 := -c
t2 := b * t1
t5 := t2 + t2
a := t5
```

Addresses and Instructions:

- A name: For convenience, we allow source-program names to appear as addresses in three-address code
- A constant: In practice, a compiler must deal with many different types of constants and variables
- A compiler-generated temporary:It is useful, especially in optimizing compilers, to create a distinct name each time a temporary is needed.

Implementations of three-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.