Three address code

Consists of a sequence of instructions, each instruction may have up to three addresses, prototypically

t1 = t2 op t3

- **Addresses** may be one of:

– A **name**. Each name is a symbol table index. For convenience, we write the names as the identifier.

– A **constant.**

– A **compiler-generated temporary**. Each time a temporary address is needed, the compiler generates another name from the stream t1, t2, t3, etc.

Three-Address Code

Temporary names allow for code optimization to easily move instructions

At target-code generation time, these names will be allocated to registers or to memory

- At most one operator in an instruction

  x+y*z

  – t1=y*z
    t2=x+t1

Example of 3-address code
a:=b*-c + b*-c

t_1:=- c
t_2:=b * t_1
t_3:=- c
t_4:=b * t_3
t_5:=t_2 + t_4
a:=t_5

t_1:=- c
t_2:=b * t_1
t_5:=t_2 + t_2
a:=t_5

Types of Three-Address Statements.

Assignment Statement:	**x:=y op z**
Assignment Statement:	**x:=op z**
Copy Statement:	**x:=z**
Unconditional Jump:	**goto L**
Conditional Jump:	**if x relop y goto L**
Stack Operations:	**Push/pop**

Procedure call:

param x_1
param x_2
…
param x_n
call p,n

P(x1,x2,.....xn)

Index Assignments:

x:=y[i]
x[i]:=y

Address and Pointer Assignments:

x:=&y
x:=*y
*x:=y