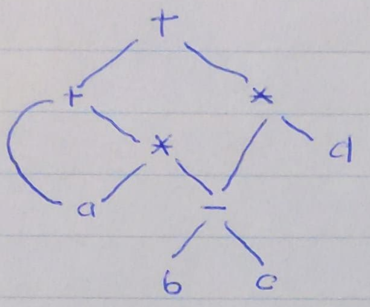
- Example
  - Three-address code is a linearized representation of a syntax tree or a DAG in which explicit names correspond to the interior nodes of the graph.
  - A DAG together with the corresponding three-address code sequence is given below:



$t_1 = b - c$

$t_2 = a * t_1$

$t_3 = a + t_2$

$t_4 = t_1 * d$

$t_5 = t_3 + t_4$

(a) DAG (b) Three-address code

- The reason for the term "three-address code" is that each statement usually contains three addresses, two for the operands and one for the result.

Types of Three Address Statements

- For convenience, we allow source program names to appear as addresses in three-address code. In an implementation, a source name is replaced by a pointer to its symbol table entry, where all information about the name is kept.
- Symbolic labels will be used by instructions that alter the flow of control. A symbolic label represents the index of a three-address instruction in the sequence of instructions. Actual indexes can be substituted for the labels, either by making a separate pass or by backpatching.

- Here is a list of the common three-address instruction forms:

1. Assignment instructions of the form $x = y \text{ op } z$, where op is a binary arithmetic or logical operation, and x, y, and z are addresses.
2. Assignments of the form $x = \text{op } y$, where op is a unary operation. Essential unary operations include unary minus, logical negation, shift operators, and conversion operators that, for example, convert an integer to a floating-point number.
3. Copy instructions of the form $x = y$, where x is assigned the value of y.
4. The unconditional jump 'goto L'. The three-address instruction with label L is the next to be executed.
5. Conditional jumps of the form 'if x goto L' and 'ifFalse x goto L'. These instructions execute the instruction with label L next if x is true and false respectively. Otherwise, the following three-address instruction in sequence is executed next, as usual.
6. Conditional jumps such as 'if x relop y goto L', which apply a relational operator to x and y, and execute the instruction with label L next if x stands in relation relop y. If not, the three-address instruction following 'if x relop y goto L' is executed next, in sequence.
7. Procedure calls and returns are implemented using the following instructions:
   - param x for parameters;
   - call p,n for procedure and function calls, and
   - return y where y is optional.

   Their typical use is as the sequence of three-address instructions

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

generated as part of a call of the procedure $p(x_1, x_2, \cdots, x_n)$.

8. Indexed copy instructions of the form $x = y[i]$ and $x[i] = y$. The instruction $x = y[i]$ sets x to the value in the location i memory units beyond location y. The instruction $x[i] = y$ sets the contents of the location i units beyond x to the value of y.
9. Address and pointer assignments of the form $x = \&y$, $x = *y$, and $*x = y$. The instruction $x = \&y$ sets the r-value of x to be the location (l-value) of y. y is a name that denotes an expression with an l-value, and x is a pointer name. In the instruction $x = *y$, y is a pointer whose r-value is a location. The r-value of x is made equal to the contents of that location. Finally, $*x = y$ sets the r-value of the object pointed to by x to the r-value of y.

- Example : Consider the statement

  do i = i+1 ; while (a[i] < v) ;

  - Two possible translations of this statement are shown below. The translation in (a) uses a symbolic label L, attached to the first instruction. The translation in (b) shows position numbers for the instructions, starting arbitrarily at position 100.

(a) Symbolic labels		(b) Position numbers	
L:	$t_1 = i + 1$	100 :	$t_1 = i + 1$
	$i = t_1$	101 :	$i = t_1$
	$t_2 = i * 8$	102 :	$t_2 = i * 8$
	$t_3 = a[t_2]$	103 :	$t_3 = a[t_2]$
	if $t_3 < v$ goto L	104 :	if $t_3 < v$ goto 100

- The choice of allowable operators is an important issue in the design of an intermediate form. The operator set clearly must be rich enough to implement the operations in the source language.