SNS COLLEGE OF TECHNOLOGY

(Autonomous)

COIMBATORE-35

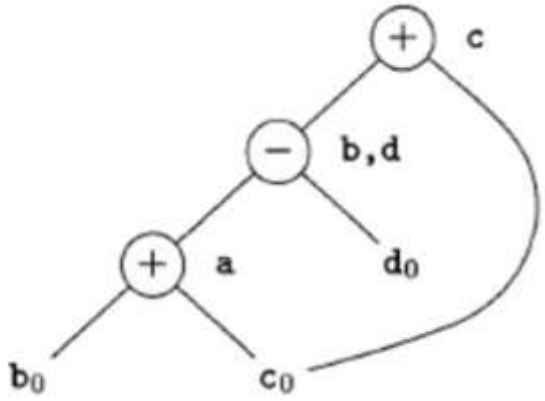
The DAG representation of basic blocks &Generating Code from DAG

The DAG representation of basic blocks

A DAG for a basic block is a directed acyclic graph with the following labels on nodes:

1. Leaves are labeled by unique identifiers, either variable names or constants.
2. Interior nodes are labeled by an operator symbol.
3. Nodes are also optionally given a sequence of identifiers for labels to store the computed values.

```
a = b + c
b = a - d
c = b + c
d = a - d
```



The DAG representation of basic blocks

Algorithm for construction of DAG

Input: A basic block

Output: A DAG for the basic block containing the following information:

1. A label for each node. For leaves, the label is an identifier. For interior nodes, an operator symbol.
2. For each node a list of attached identifiers to hold the computed values.

Case (i) x : = y OP z

Case (ii) x : = OP y

Case (iii) x : = y

The DAG representation of basic blocks

Method:

Step 1:

If y is undefined then create node(y).

If z is undefined, create node(z) for case(i).

Step 2:

For the case(i), create a node(OP) whose left child is node(y) and right child is node(z). (Checking for common sub expression). Let n be this node.

For case(ii), determine whether there is node(OP) with one child node(y). If not create such a node.

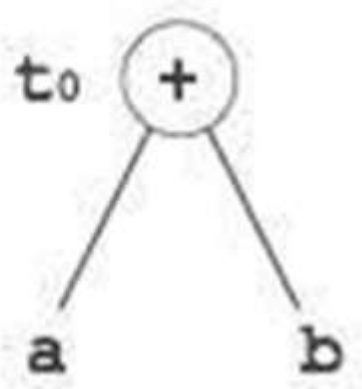
For case(iii), node n will be node(y).

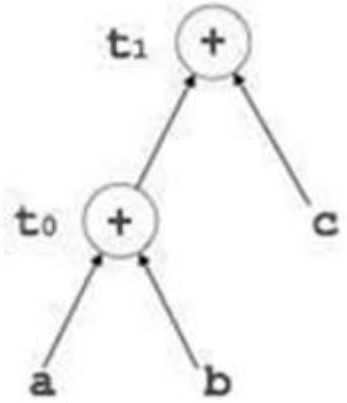
Step 3:

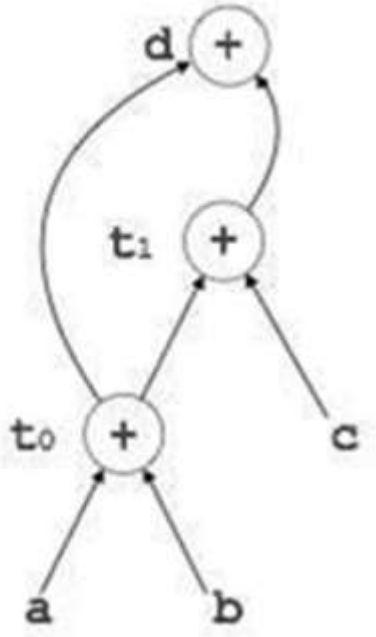
Delete x from the list of identifiers for node(x). Append x to the list of attached identifiers for the node n found in step 2 and set node(x) to n.

The DAG representation of basic blocks

$t_0 = a + b\ t_1 = t_0 + c\ d = t_0 + t_1$



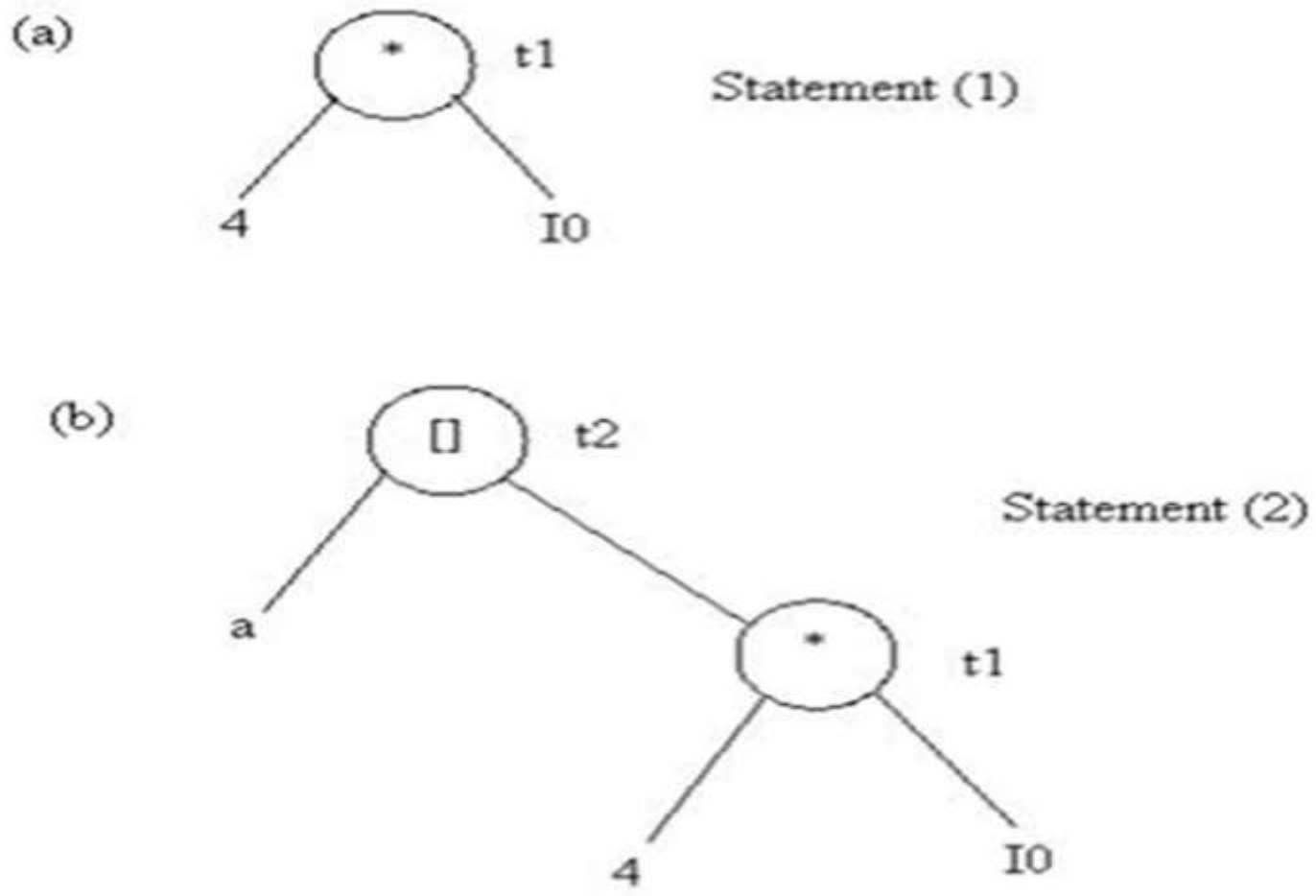




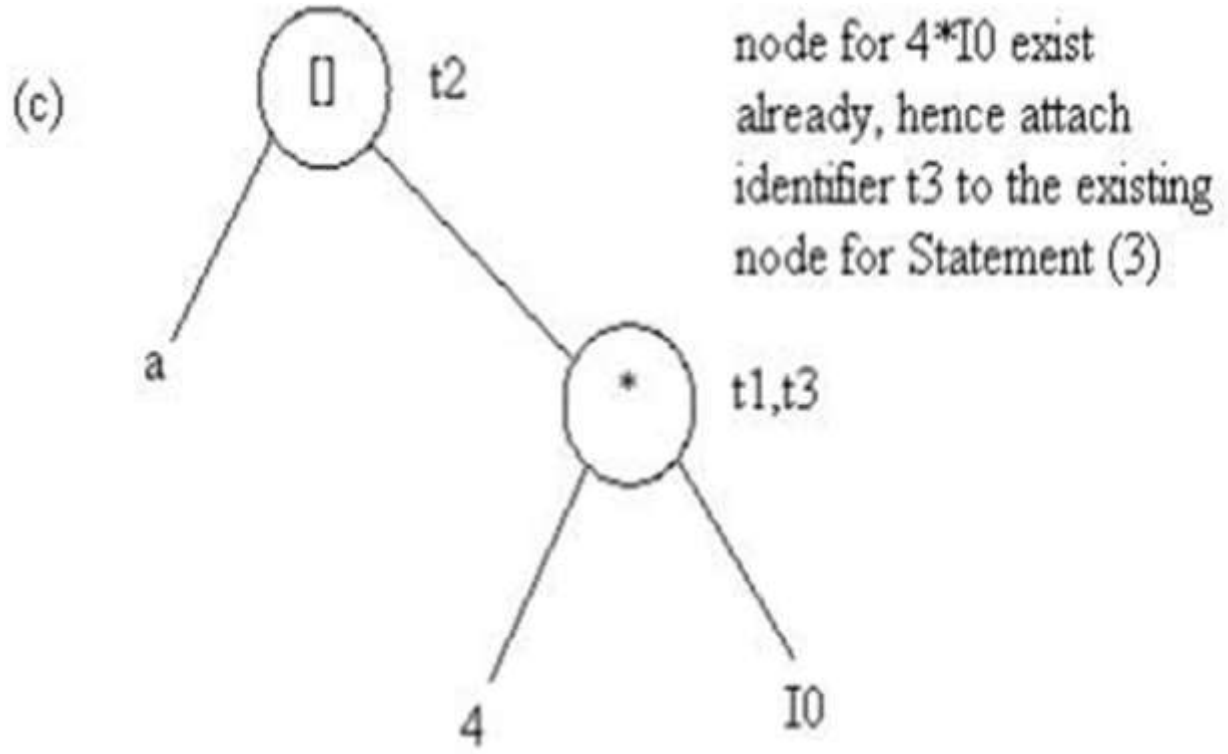
The DAG representation of basic blocks

```
1. t1 := 4* i
2. t2 := a[t1]
3. t3 := 4* i
4. t4 := b[t3]
5. t5 := t2*t4
6. t6 := prod+t5
7. prod := t6
8. t7 := i+1
9. i := t7
10. if i<=20 goto (1)
```

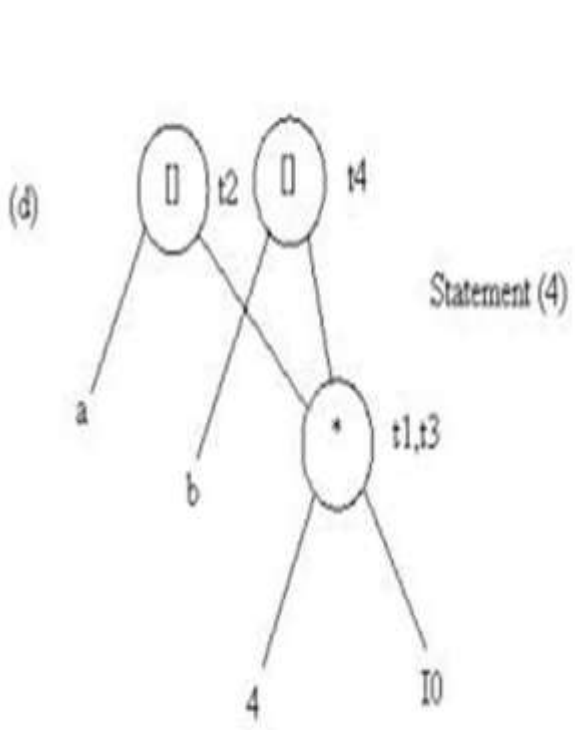
The DAG representation of basic blocks

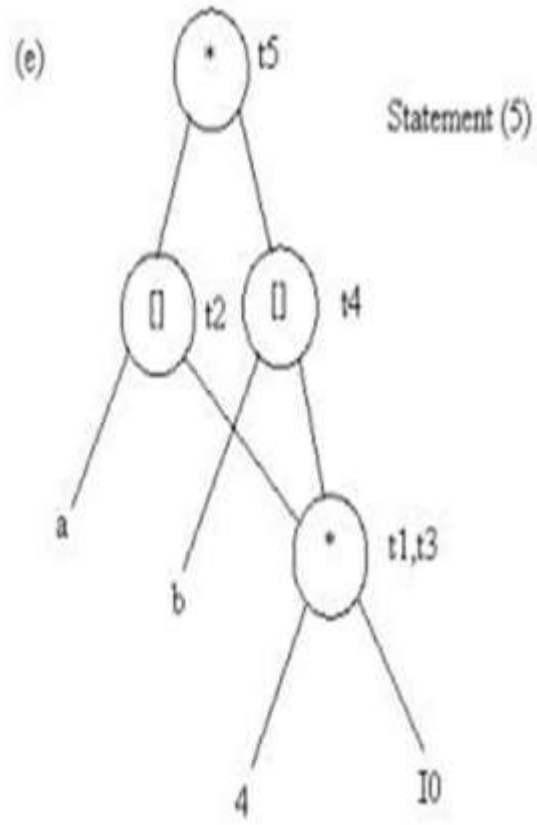


The DAG representation of basic blocks

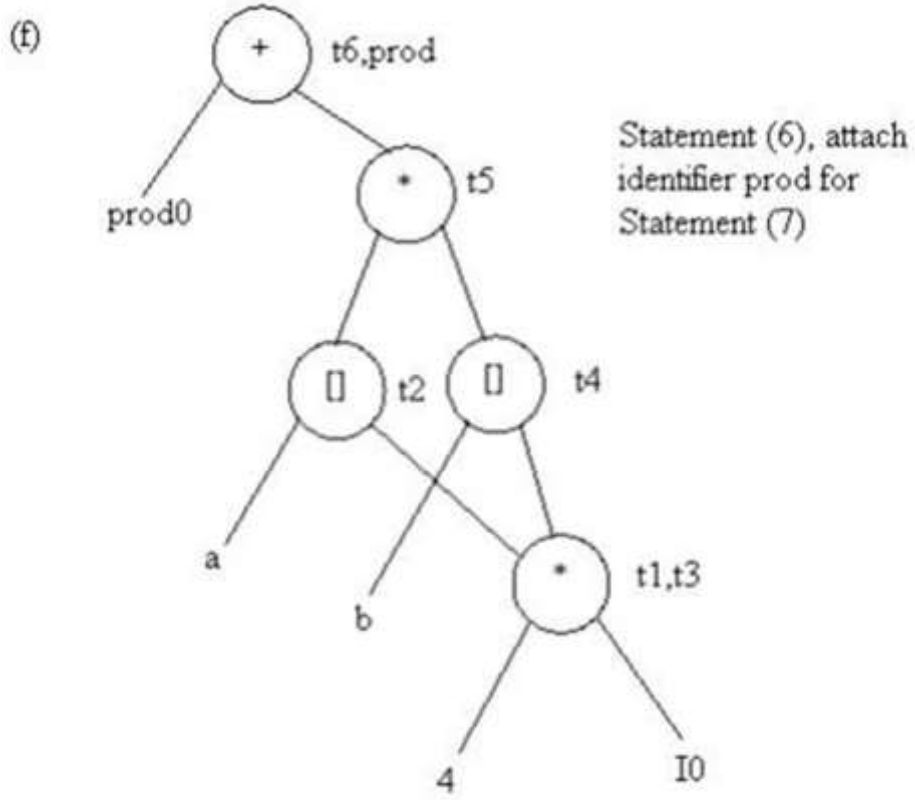


The DAG representation of basic blocks

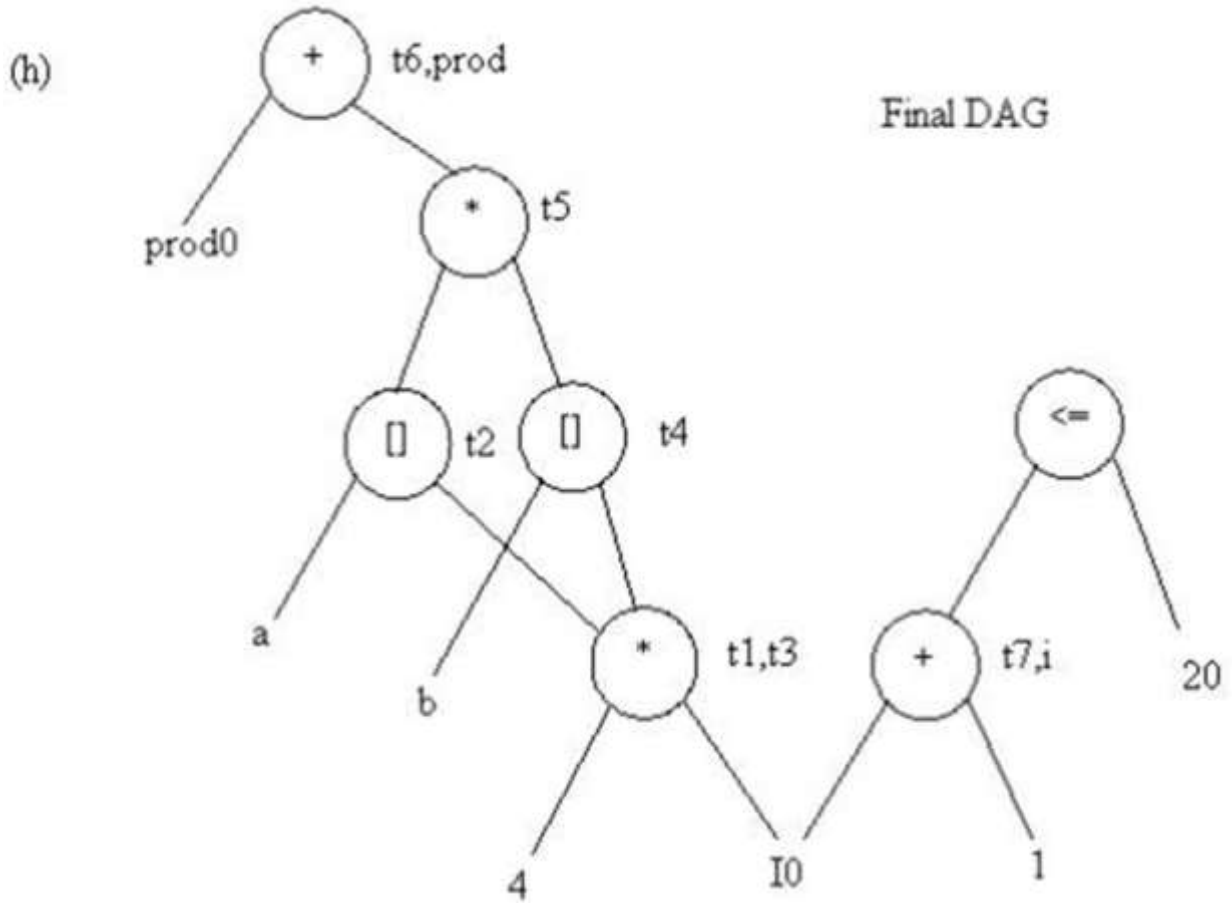




The DAG representation of basic blocks



The DAG representation of basic blocks



Generating code from DAG

Generating code from DAG

The advantage of generating code for a basic block from its dag representation is that from a dag we can easily see how to rearrange the order of the final computation sequence than we can start from a linear sequence of three-address statements or quadruples.

Rearranging the order

The order in which computations are done can affect the cost of resulting object code. For example, consider the following basic block:

t1 : = a + b

t2 : = c + d

t3 : = e - t2

t4 : = t1 - t3

Generating code from DAG

Generated code sequence for basic block:

MOV a , R0

ADD b , R0

MOV c , R1

ADD d , R1

MOV R0 , t1

MOV e , R0

SUB R1 , R0

MOV t1 , R1

SUB R0 , R1

MOV R1 , t4

Generating code from DAG

- **Rearranged basic block:**
- Now t1 occurs immediately before t4.
- t2 : = c + d
- t3 : = e - t2
- t1 : = a + b
- t4 : = t1 - t3

- **Revised code sequence:**
- MOV c , R0
- ADD d , R0
- MOV a , R0
- SUB R0 , R1
- MOV a , R0
- ADD b , R0
- SUB R1 , R0
- MOV R0 , t4

In this order, two instructions **MOV R0 , t1 and MOV t1 , R1** have been saved.

Generating code from DAG

A Heuristic ordering for Dags

The heuristic ordering algorithm attempts to make the evaluation of a nod the evaluation of its leftmost argument. The algorithm shown below produces the ordering in reverse.

Algorithm:

1) while unlisted interior nodes remain do begin
2) select an unlisted node n, all of whose parents have been listed;
3) list n;
4) while the leftmost child m of n has no unlisted parents and is not a

leaf do
begin
5) list m;
6) n = m
end
end

Generating code from DAG

- **Example: Consider the DAG shown below**
- Initially, the only node with no unlisted parents is 1 so set n=1 at line (2) and list 1 at line (3).
- Now, the left argument of 1, which is 2, has its parents listed, so we list 2 and set n=2 at line (6).
- Now, at line (4) we find the leftmost child of 2, which is 6, has an unlisted parent 5. Thus we select a new n at line (2), and node 3 is the only candidate.
- list 3 and proceed down its left chain, listing 4, 5 and 6. This leaves only 8 among the interior nodes so list that. The resulting list is 1234568 and the order of evaluation is 8654321.

Generating code from DAG

Code sequence:

t8 : = d + e

t6 : = a + b

t5 : = t6 - c

t4 : = t5 * t8

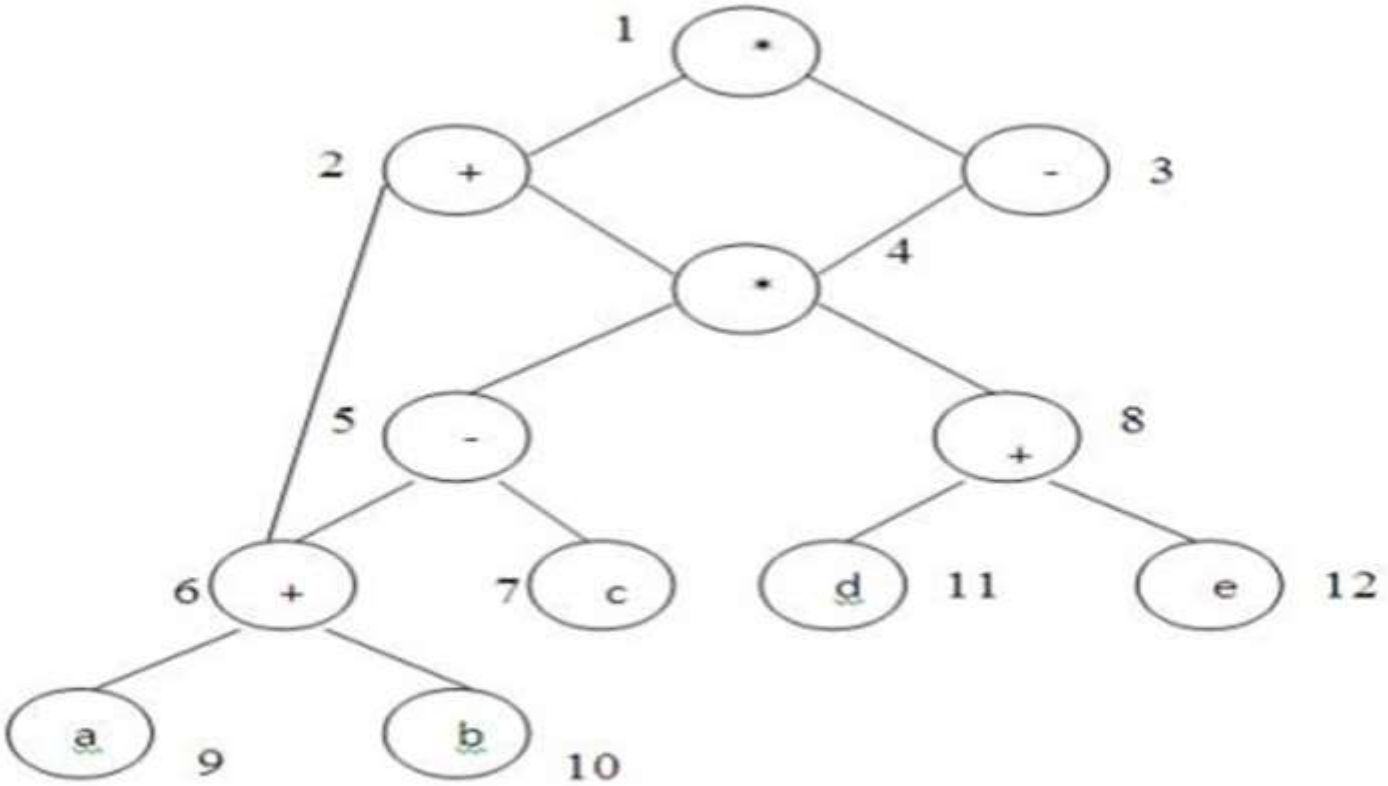
t3 : = t4 - e

t2 : = t6 + t4

t1 : = t2 * t3

This will yield an optimal code for the DAG on machine whatever be the number of registers.

Generating code from DAG



A DAG

Summarization