You are given the following symbol table, and a piece of instruction.

index	identifier	type
1	foo	function
2	bar	int
3	v	int
4	x	int
5	y	int
6	z	int

The instruction: y = (x - y)/v * foo(z, bar(5))/ bar(v);

Now your task is to do the following operations step by step on the instruction.

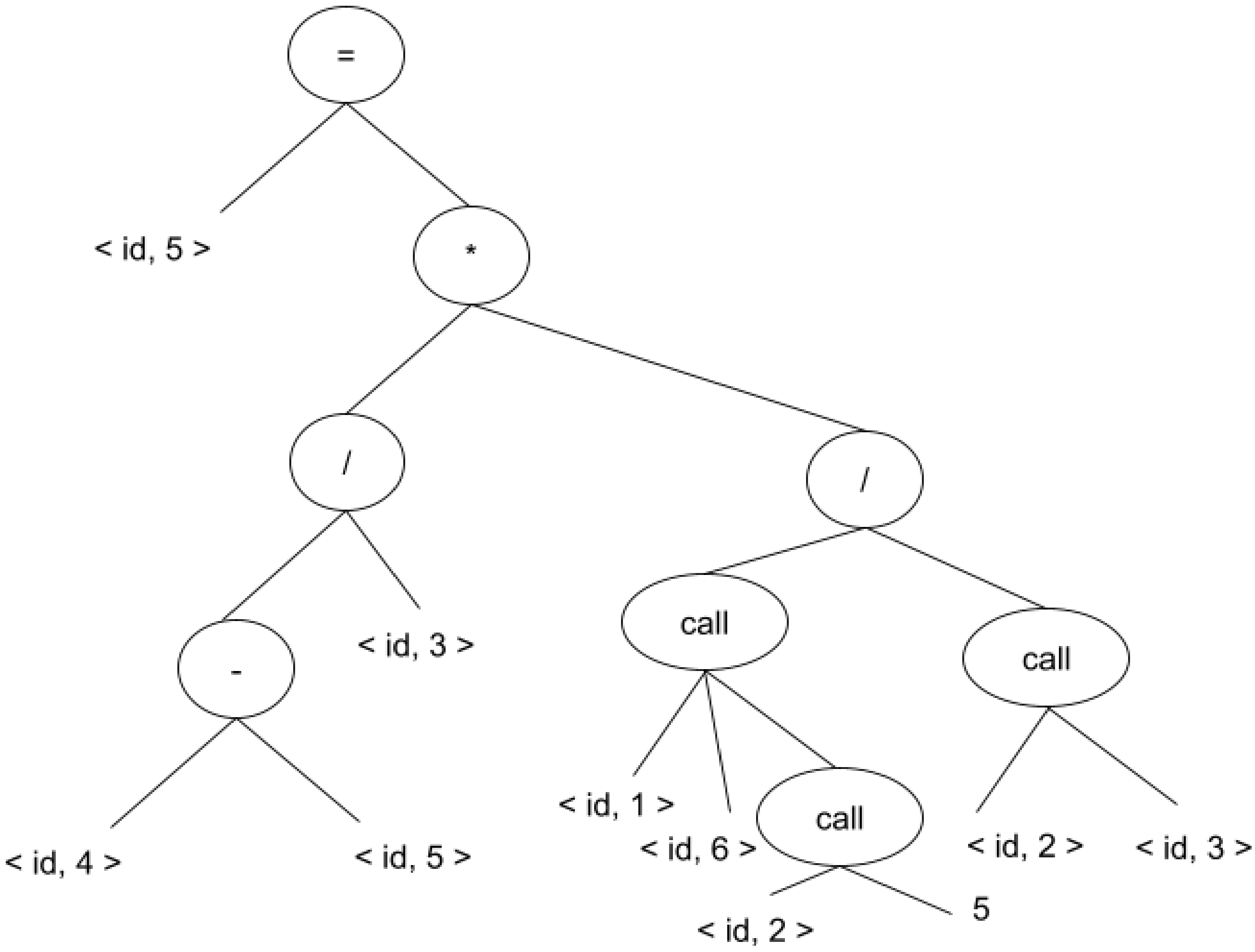
- Tokenize the instruction based on the symbol table
- Construct the syntax tree from obtained tokens
- Construct the semantic tree from prepared syntax tree
- Generate the x86 assembly code using the semantic tree

Solution:

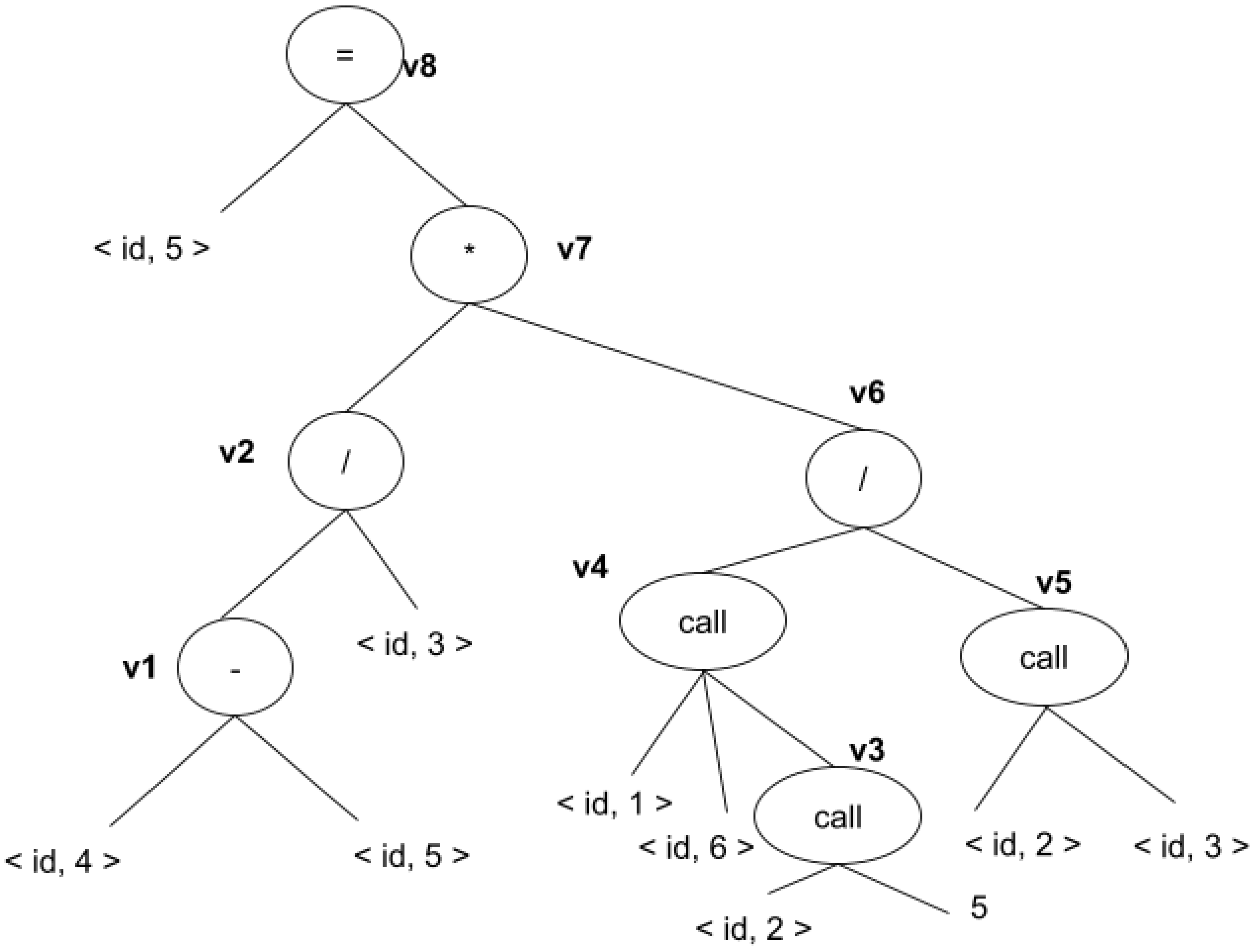
Step 1:

< id, 5 >	< = >	< (>	< id, 4 >	< - >	< id, 5 >	<) >	< / >
< id, 3 >	< * >	< id, 1 >	< (>	< id, 6 >	< , >	< id, 2 >	< (>
< 5 >	<) >	<) >	< / >	< id, 2 >	< (>	< id, 3 >	<) >
< ; >							

Step 2:



Step 3:


=
v8
< id, 5 >
*
v7
v6
v2
/
/
v4
call
v5
call
v1
-
< id, 3 >
v3
< id, 1 >
call
< id, 6 >
< id, 2 >
< id, 3 >
< id, 4 >
< id, 5 >
< id, 2 >
5

Step 4:

Intermediate code →

```
t1 = id4 - id5
t2 = t1/id3
param 5
t3 = call id2, 1
param id6
param t3
t4 = call id1, 2
param id3
t5 = call id2, 1
t6 = t4/t5
t7 = t2 * t6
id5 = t7
```

Optimized intermediate code →

```
t1 = id4 - id5
t2 = t1/id3
param 5
t1 = call id2, 1
param id6
param t1
t3 = call id1, 2
param id3
t4 = call id2, 1
t1 = t3/t4
id5 = t2 * t1
```

x86 Assembly code →

```
MOV AX, [id4]
SUB AX, [id5]
MOV BX, [id3]
XOR DX, DX
DIV BX
MOV BX, AX
MOV AX, 5
PUSH AX
CALL id2
ADD SP, 2
PUSH AX
MOV AX, [id6]
PUSH AX
CALL id1
ADD SP, 4
MOV CX, AX
MOV AX, [id3]
PUSH AX
CALL id2
ADD SP, 2
MOV DX, AX
MOV AX, CX
MOV CX, DX
XOR DX, DX
DIV CX
MUL BX
MOV [id5], AX
```