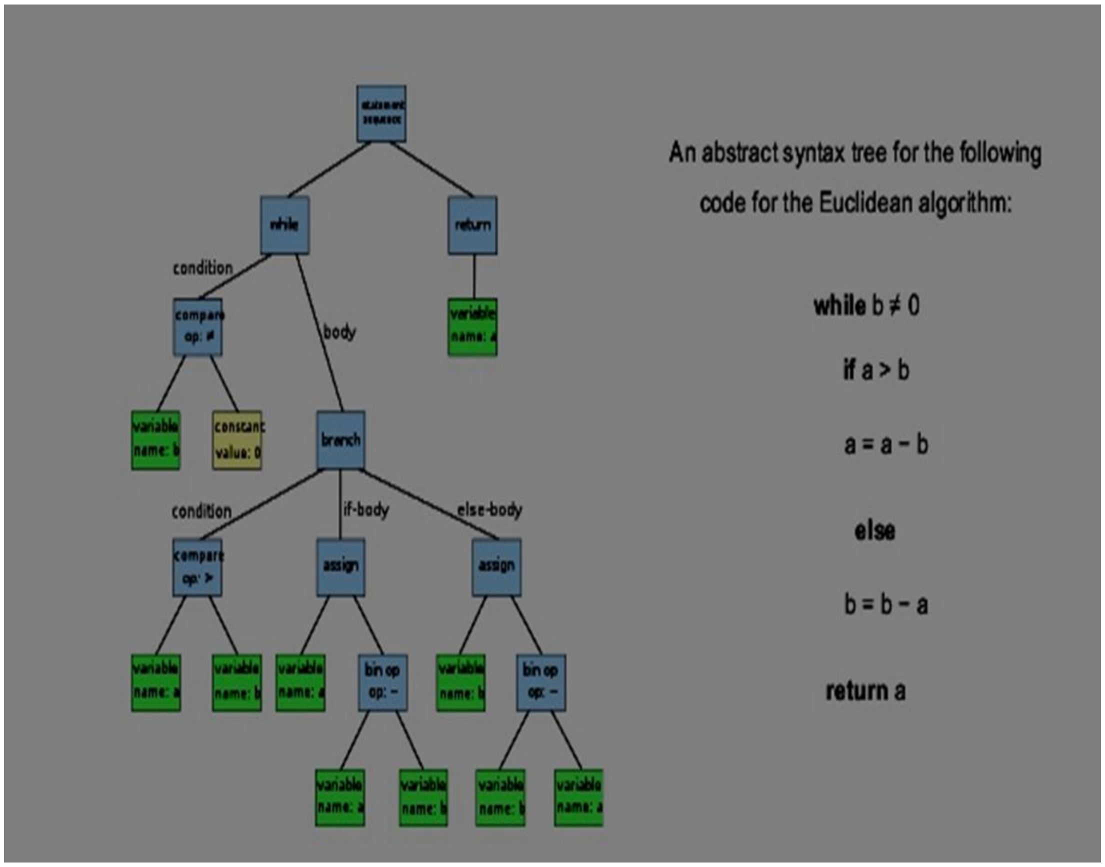


An abstract syntax tree for the following code for the Euclidean algorithm:

```
while b ≠ 0
if a > b
a = a − b
else
b = b − a
return a
```

Syntax Trees in Compilers

CO322

E/13/244 E/13/377 E/13/132 | May 12, 2017

An abstract syntax tree is a tree representation of the abstract syntactic structure of source code written in a programming language. Each node of the tree denotes a construct occurring in the source code. The syntax is "abstract" in not representing every detail appearing in the real syntax. For instance, grouping parentheses are implicit in the tree structure, and a syntactic construct like an if-condition-then expression may be denoted by means of a single node with three branches.

This abstract syntax trees from concrete syntax trees, traditionally designated parse trees, which are often built by a parser during the source code translation and compiling process. Once built, additional information is added to the abstract syntax tree by means of subsequent processing.

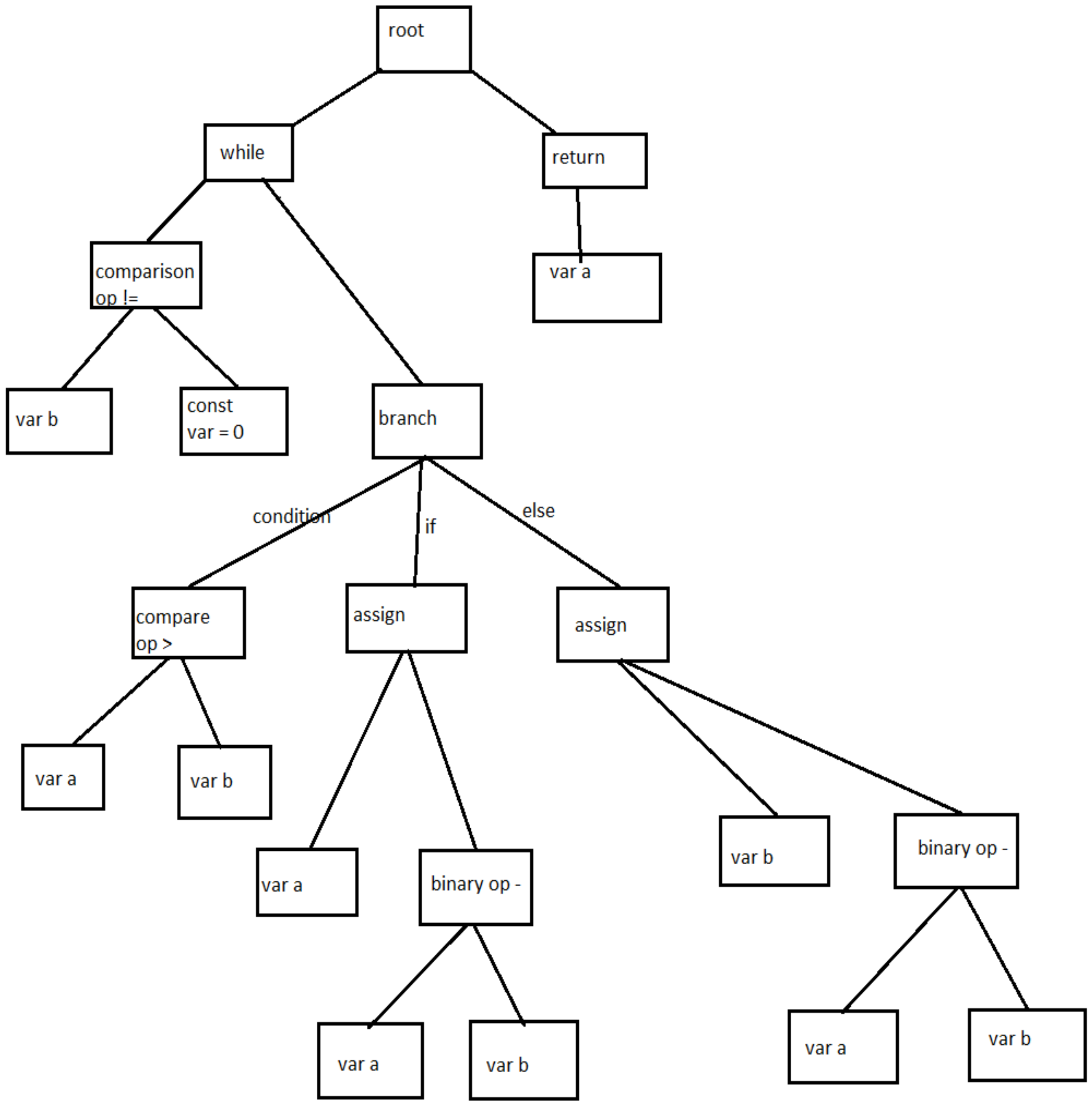
Design

This is way to convert a code into a syntax tree. We can use a simple code for to do this.

e.g:1

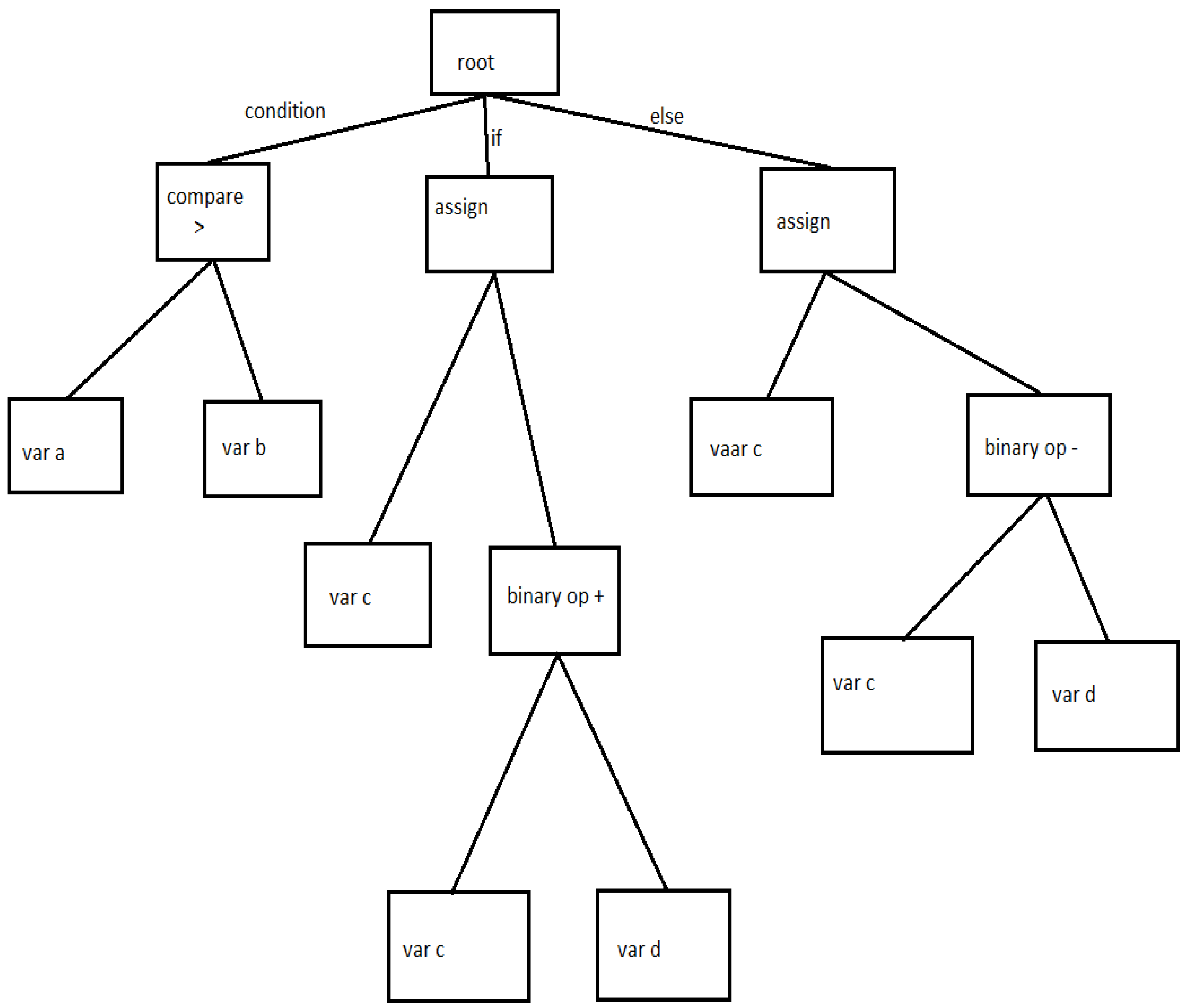
```
 while b != 0
     if a > b
         a = a – b;
    else
        b = b-a;
return a;
```

When this is converted into a tree it is like this.


root
while
return
comparison op !=
var a
var b
const var = 0
branch
condition
if
else
compare op >
assign
assign
var a
var b
var a
binary op -
var b
binary op -
var a
var b
var a
var b

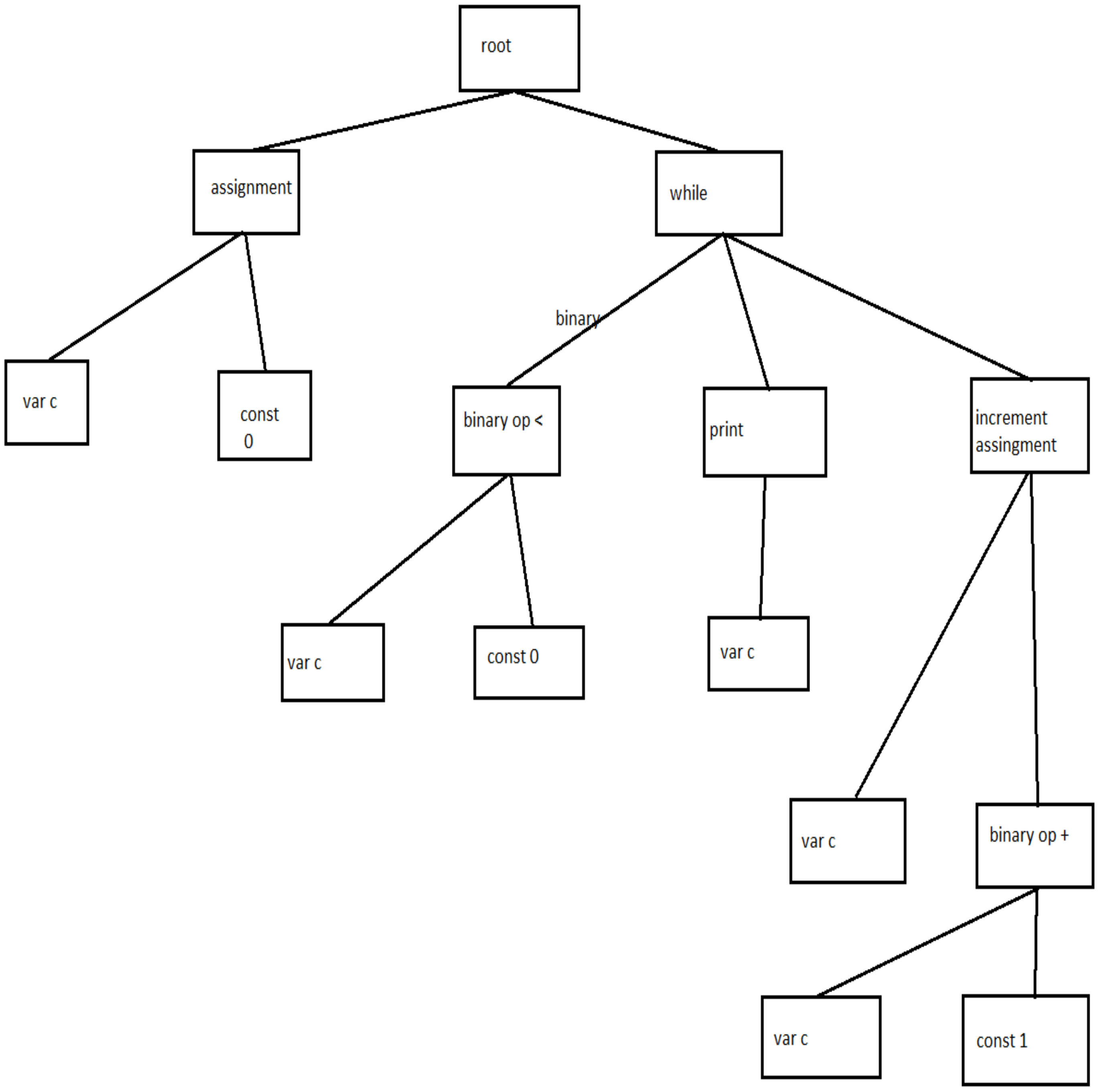
e.g. 2:

```
If a > b:
        c = c + d;
else
        c = c – d;
```



e.g: 3

```
c = 0
while c < 10
        print c ;
c = c + 1;
```


root
assignment
while
binary
var c
const 0
binary op <
print
increment assingment
var c
const 0
var c
var c
binary op +
var c
const 1