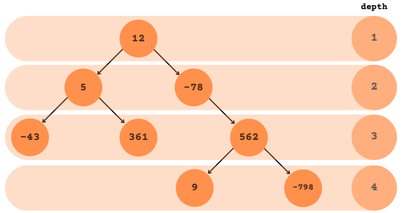
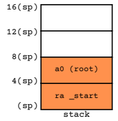
Binary Tree



This approach worked, now will try will fp

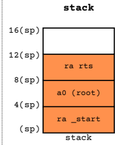
```
First steps, before calling recursion
...
mv s0, sp
sw ra, (sp) # store return address for _start
addi sp, sp, -4
sw a0, 0(sp) # store root of the tree
li a2, 0 # depth
...
```



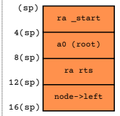
(Example test)
value to be search is **5**

First call of dfs

```
...
depth_first_search:
addi sp, sp, -4
sw ra, (sp) # store return address for recursive_tree_search
lw a0, 4(sp) # load root of the tree, a0 <- a0 (root)
addi a2, a2, 1 # depth++

lw t0, 0(a0) # load node->val
beq a1, t0, found # if node->val == val, found
...
```

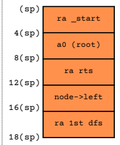



```
In case you didn't find, search left

...
lw t0, 4(a0) # load node->left
beq t0, zero, search_right # if root->left == NULL, search_right
                           # in this code root->left != NULL
# search left
addi sp, sp, -4
sw t0, (sp) # store node->left
jal ra, depth_first_search
...
```

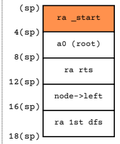


Second call of dfs

```
...
depth_first_search:
addi sp, sp, -4
sw ra, (sp) # store return address for recursive_tree_search
lw a0, 4(sp) # load root of the tree, a0 <- node->left
addi a2, a2, 1 # depth++

lw t0, 0(a0) # load node->val
beq a1, t0, found # if node->val == val, found
# Now it found 5
...
```




```
It found the number 5
...
found:
addi sp, sp, 16             # we sum 8 to sp, to reach ra rst
lw ra, (sp)                 # we have to reach ra _start
mv a0, a2
ret
```



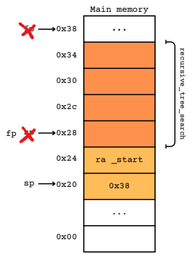
(Example test)
value to be search is **5**

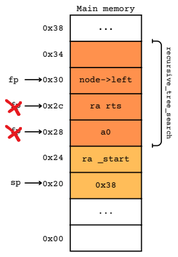
Worked only when fp was empty!!!

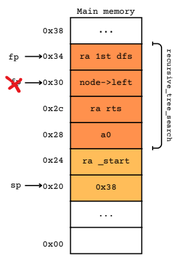
```
recursive_tree_search:

addi sp, sp, -8
sw ra, 4(sp)
sw fp, 0(sp)
addi fp, sp, 8
sw a0, (fp)
```

```
depth_first_search:
lw a0, (fp)             # current node
addi fp, fp, 4
sw ra, (fp)             # ra rts

addi a2, a2, 1          #depth++

lw t0, 0(a0)            # node->val
beq a1, t0, found       # node -> val == val, found

lw t0, 4(a0)            # node->left
beq t0, zero, search_right

# search left
addi fp, fp, 4
sw t0, (fp)
jal depth_first_search
```

```
depth_first_search:
lw a0, (fp)             # current node
addi fp, fp, 4
sw ra, (fp)             # ra 1st dfs

addi a2, a2, 1          #depth++

lw t0, 0(a0)            # node->val
beq a1, t0, found       # node -> val == val, found
...
found:
lw ra, (sp)
lw fp, 4(sp)
addi sp, sp, 8
mv a0, a2
ret
```



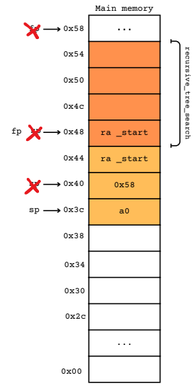




(Example test)
value to be search is **361**

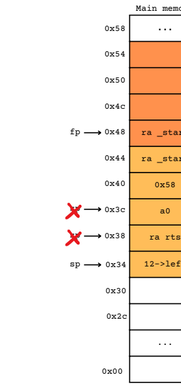
For 5 it worked!

```
recursive_tree_search:

addi sp, sp, -8
sw ra, 4(sp)
sw fp, 0(sp)
addi fp, sp, 8
sw ra, (fp)
li a2, 0

addi sp, sp, -4
sw a0, (sp)
```



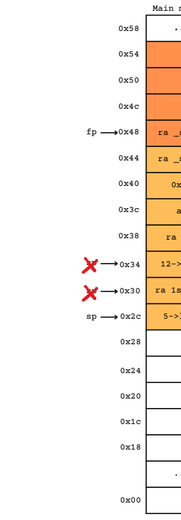
12

```
depth_first_search:
lw a0, (sp)             # current node
addi sp, sp, -4
sw ra, (sp)             # ra rts

addi a2, a2, 1          #depth++

lw t0, 0(a0)            # node->val
beq a1, t0, found       # node -> val == val, found

lw t0, 4(a0)            # node->left
beq t0, zero, search_right

# search left
addi sp, sp, -4
sw t0, (sp)
jal depth_first_search
```



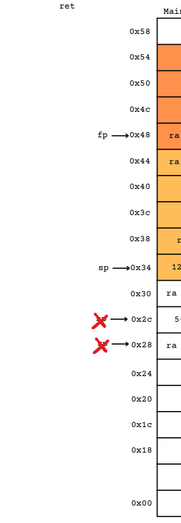
5

```
depth_first_search:
lw a0, (sp)             # node->left
addi sp, sp, -4
sw ra, (sp)

addi a2, a2, 1          #depth++

lw t0, 0(a0)            # node->val
beq a1, t0, found       # node -> val == val, found

lw t0, 4(a0) # node->left
beq t0, zero, search_right

# search left
addi sp, sp, -4
sw t0, (sp)
jal depth_first_search
```



-43

```
depth_first_search:
lw a0, (sp)             # 5->left
addi sp, sp, -4
sw ra, (sp)

addi a2, a2, 1          #depth++

lw t0, 0(a0)            # node->val
beq a1, t0, found       # node -> val == val, found

lw t0, 4(a0) # -43->left
beq t0, zero, search_right
...
search_right:
lw a0, 4(sp) # node 5->left (-43)
lw t0, 8(a0) # load 5->left->right (-43->right)

beq t0, zero, not_found
...
not_found:
lw ra, (sp)             # ra 2nd dfs
addi sp, sp, 12
addi a2, a2 -1
ret
```



5

```
depth_first_search:
addi sp, sp, -4
sw ra, (sp)
lw a0, (sp) # 12->left

addi a2, a2, 1          #depth++

lw t0, 0(a0)            # node->val
beq a1, t0, found       # node->val == val, found

lw t0, 4(a0) # 12->left
beq t0, zero, search_right
...
search_right:
lw a0, 4(sp) # node 5->left (-43)
lw t0, 8(a0) # load 5->left->right (-43->right)

beq t0, zero, not_found
...
not_found:
lw ra, (sp)             # ra 2nd dfs
```