Write Return Value Back into `the_number`

```
the_number = find_abs (the_number);
```

R0 now holds the value of the right side.

So we need to store into `the_number`.

Where is `the_number` again?

Let's look it up in the symbol table!

scope	identifier	type	from	offset	...
translate.c	the_number	int32_t	R4	0	...
find_abs	abs_value	int32_t	R5	0	...
find_abs	num	int32_t	R5	4	...

Store R0 into `the_number`

```
LDR R0,R4,#0
ADD R6,R6,#-1
STR R0,R6,#0
JSR FIND_ABS
LDR R0,R6,#0
ADD R6,R6,#2
STR R0,R4,#0
```

Write R0 into `the_number`.

Is there an LC-3 instruction for that?

scope	identifier	type	from	offset	...
translate.c	the_number	int32_t	R4	0	...

Reference Version of Function Call and Assignment

```
LDR R0,R4,#0
ADD R6,R6,#-1
STR R0,R6,#0
JSR FIND_ABS
LDR R0,R6,#0
ADD R6,R6,#2
STR R0,R4,#0
```

That's it for the function call!

You can find both the C code and the LC-3 code on the web page.

```
the_number = find_abs (the_number);
```


Code for a Function Consists of Four Parts

Now we're ready to translate `find_abs`.

A function's code consists of four parts:

1. set up the stack frame,
2. execute the statements,
3. tear down the stack frame (leaving the return address on the stack with LC-3),
4. and return (RET).

