The Statement in Main Begins with a Function Call

Now we're ready to translate the statement:

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

Remember that for an assignment, the compiler generates instructions to...

1. evaluate the expression on the right, then
2. store the result into the address on the left.

So **first**, we must **call the function**.

Calling a Function Consists of Four Steps

Calling a function consists of four steps:

1. evaluate and push the parameters,
2. call the function (with JSR),
3. read the return value from the top of the stack, and
4. pop off the return value and the parameters.

Evaluate Expressions Used for Parameter Values

Step 1: Evaluate and push parameters.

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

The function is called with one parameter.

Where is it?

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	...

Read the Variable `the_number` into R0

LDR R0,R4,#0

We're finally ready to write code!

First, read `the_number` into R0.

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	...

