Stack Frame Creation Shared by Caller and Callee

Who chooses parameter values?
(Caller or callee?)

Caller pushes the parameters onto the stack.

For example, **`main`** pushes the input to **`find_abs`**.

Callee creates the remainder of the stack frame.

When JSR Returns, Return Value is on Top of Stack

Why is the return value next on the stack?

Local variables
Address of caller's stack frame
Return address (R7 in LC-3)
Outputs (return value)
Inputs (parameters, arguments)

these form the **linkage**

Return value remains on stack on return.

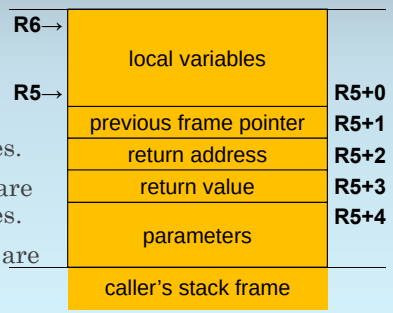
Local Variables and Parameters Accessed Using R5

R6 points to top of stack.

R5 points to bottom of local variables.

R5+0, -1, … are local variables.

R5+4, +5, … are parameters.



Compilers Use Symbol Tables to Locate Variables

How does a compiler generate instructions?

First, it builds a **symbol table** (like an assembler's, but with more information).

Here's an example for **`translate.c`**:

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	…

