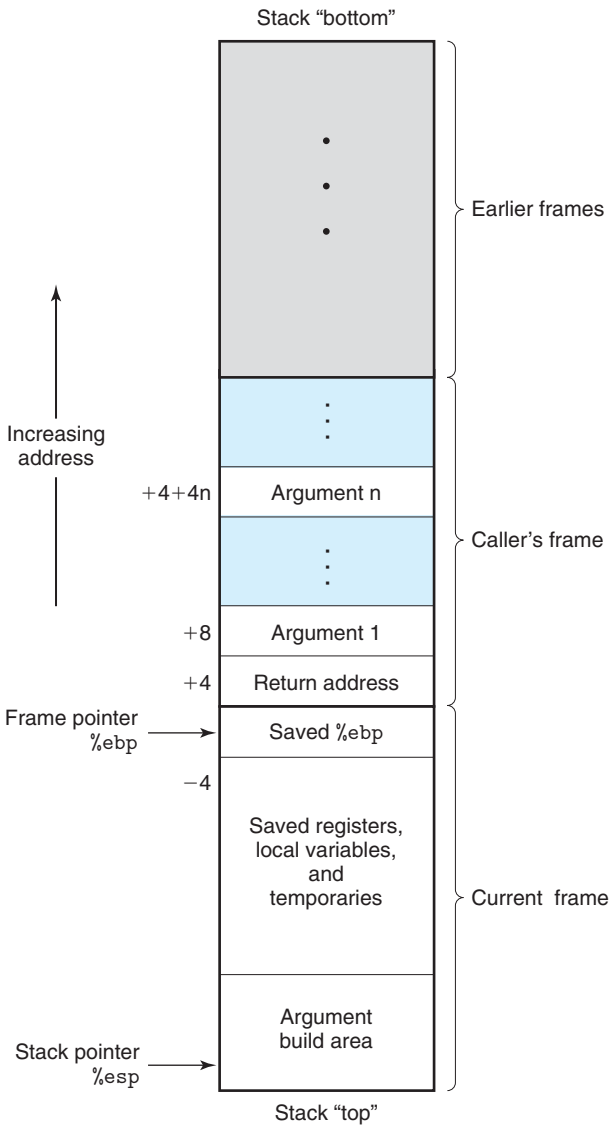

Figure 3.21
Stack frame structure. The stack is used for passing arguments, for storing return information, for saving registers, and for local storage.

serving as the *stack pointer*. The stack pointer can move while the procedure is executing, and hence most information is accessed relative to the frame pointer.

Suppose procedure `P` (the *caller*) calls procedure `Q` (the *callee*). The arguments to `Q` are contained within the stack frame for `P`. In addition, when `P` calls `Q`, the *return address* within `P` where the program should resume execution when it returns from `Q` is pushed onto the stack, forming the end of `P`'s stack frame. The stack frame for `Q` starts with the saved value of the frame pointer (a copy of register `%ebp`), followed by copies of any other saved register values.