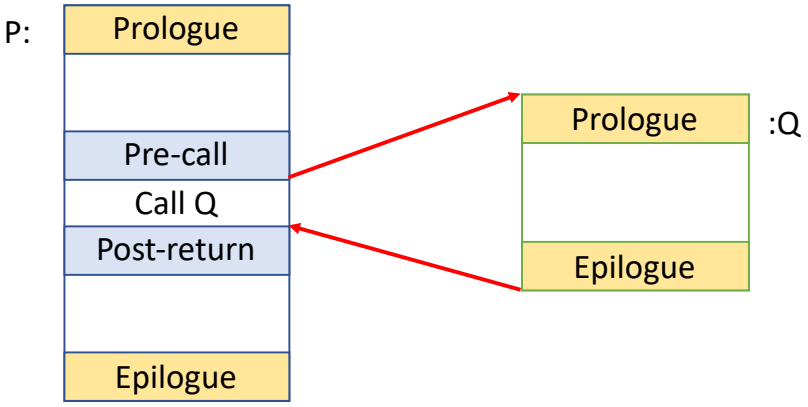
# Code Generation for Methods

- Four pieces of linking code
  - Pre-call and post-return sequences around the call site.
  - Prologue and epilogue sequences in each method definition.
- Stack frame layout
  - Use both `%rbp` and `%rsp`. Don't use red zone.
  - Use `%rbp` to access any data from caller's stack frame.
  - Use either `%rbp` or `%rsp` to access data from callee's stack frame.
  - Remember to follow data alignment rules.




| Position | Contents | Frame |
|---|---|---|
| 8n+16(%rbp) | memory argument eightbyte $n$ | Previous |
| | ... | |
| 16(%rbp) | memory argument eightbyte 0 | |
| 8(%rbp) | return address | Current |
| 0(%rbp) | previous %rbp value | |
| -8(%rbp) | unspecified | |
| | ... | |
| 0(%rsp) | variable size | |
| -128(%rsp) | red zone | |

# Code Generation for Methods: Example

- Note how P behaves both as a caller (of Q) and as a callee (of its caller).
- The lines in green are the prologue and the epilogue, and show P's callee-saved actions. Note how the pushq and popq actions happen in opposite order.
- The lines in yellow show P's caller-saved actions.
- The lines in red are the normal actions of P.

```
long P(long x, long y) {
  long u = Q(y);
  long v = Q(x);
  return u+v;
}

P:
  pushq %rbp
  pushq %rbx
  subq $8, %rsp
  movq %rdi, %rbp
  movq %rsi, %rdi
  call Q
  movq %rax, %rbx
  movq %rbp, %rdi
  call Q
  addq %rbx, %rax
  addq $8, %rsp
  popq %rbx
  popq %rbp
  retq
```