Runtime Stack

COMP201 Lab Session

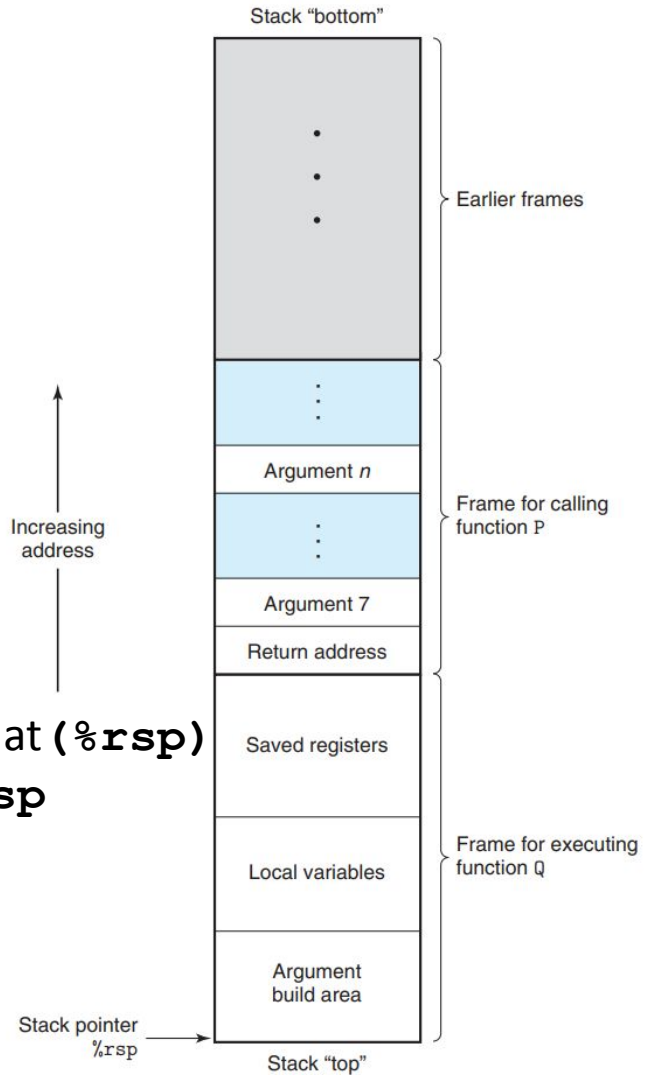
Fall 2025

KOÇ UNIVERSITY

Enes Sanli

Recap: x86-64 Stack

- Grows **downward** towards **lower** memory addresses
- **`%rsp`** points to **top** of the stack



- **`push %reg:`** subtract 8 from **`%rsp`**, put val in **`%reg`** at **`(%rsp)`**
- **`pop %reg:`** put val at **`(%rsp)`** in **`%reg`**, add 8 to **`%rsp`**

Content adapted from: Randal E. Bryant and David R. O'Hallaron, Computer Systems: A Programmer's Perspective, Third Edition, Pearson, 2016

Recap: x86-64 Register Conventions

- **Arguments passed in registers:**
  - `%rdi, %rsi, %rdx, %rcx, %r8, %r9`
- **Return value:** `%rax`

- **Callee-saved:**
  - `%rbx, %r12, %r13, %r14, %rbp, %rsp`
- **Caller-saved:**
  - `%rdi, %rsi, %rdx, %rcx, %r8, %r9, %r10, %r11, %rax`
- **Stack pointer:** `%rsp`

- **Instruction pointer:** `%rip`

Recap: x86-64 Function Call Setup

Caller:

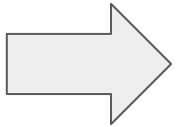
- Allocates stack frame large enough for saved registers, optional arguments
- Save any caller-saved registers in stack frame
- Save any optional arguments (in **reverse order**) in frame
- `call foo:` **push** `%rip` to stack, **jump** to label `foo`

Callee:

- Push any callee-saved registers, decrease %rsp to make room for new frame
- Increase `%rsp`, pop any callee-saved registers (in **reverse order**)
- `ret: pop %rip`

Example Code

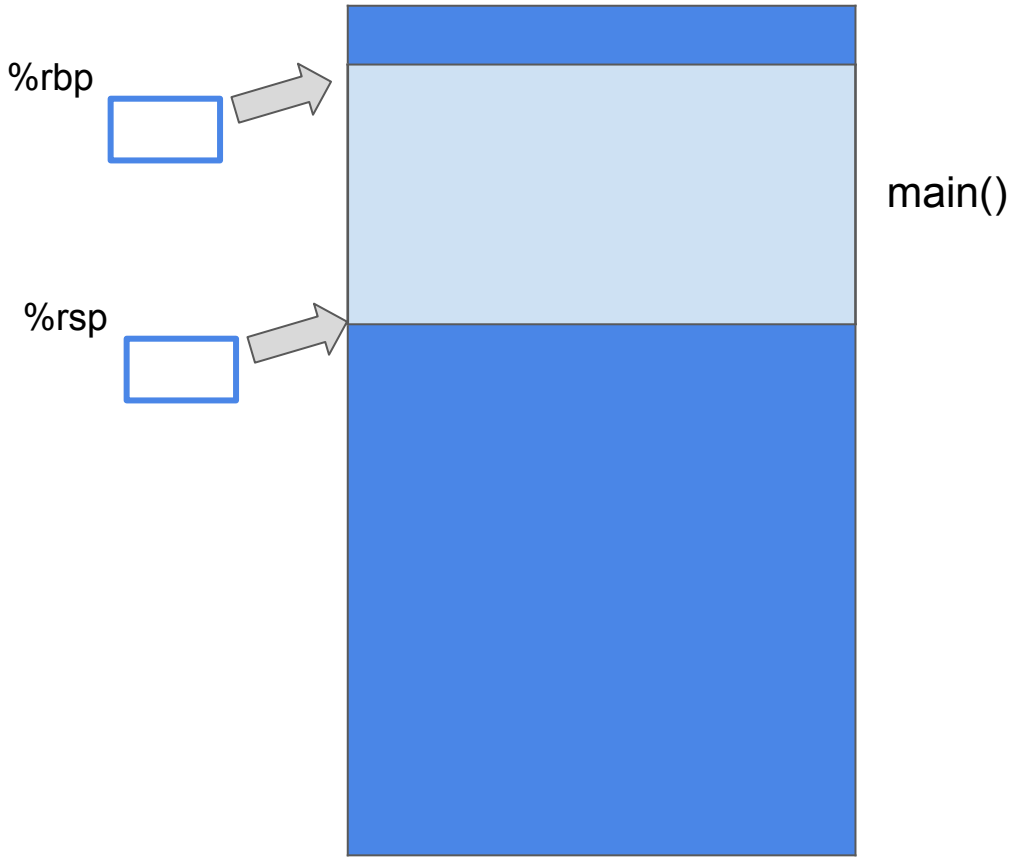
```
int foo1()
{
     int i = 2;
     return i;
}

int foo()
{
     int i = 5;
     return foo1();
}
```



```
0x0000000000400546 <foo1>:
     push rbp
     movq rsp, rbp
     sub 16, rsp
     movl $2, -0x4(rbp)
     movl -0x4(rbp), eax
     movq rbp, rsp
     pop rbp
     ret

0x0000000000400626 <foo>:
     push rbp
     movq rsp, rbp
     sub 16, rsp
     movl $5, -0x4(rbp)
     call 0x400546 <foo1>
     movq rbp, rsp
     pop rbp
     ret
```

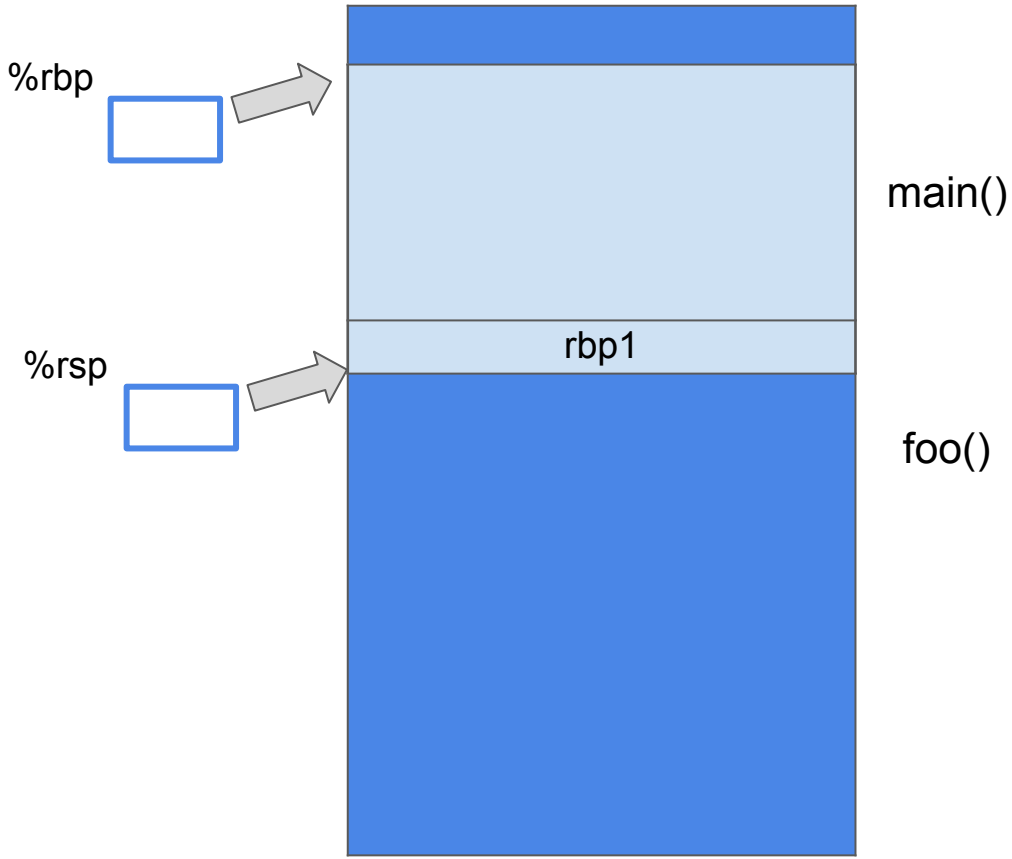
Execution Flow

```
0x0000000000400546 <foo1>:
      push rbp
      movq rsp, rbp
      sub 16, rsp
      movl $2, -0x4(rbp)
      movl -0x4(rbp), eax
      movq rbp, rsp
      pop rbp
      ret

0x0000000000400626 <foo>:
      push rbp
      movq rsp, rbp
      sub 16, rsp
      movl $5, -0x4(rbp)
      call 0x400546 <foo1>
      movq rbp, rsp
      pop rbp
      ret
```



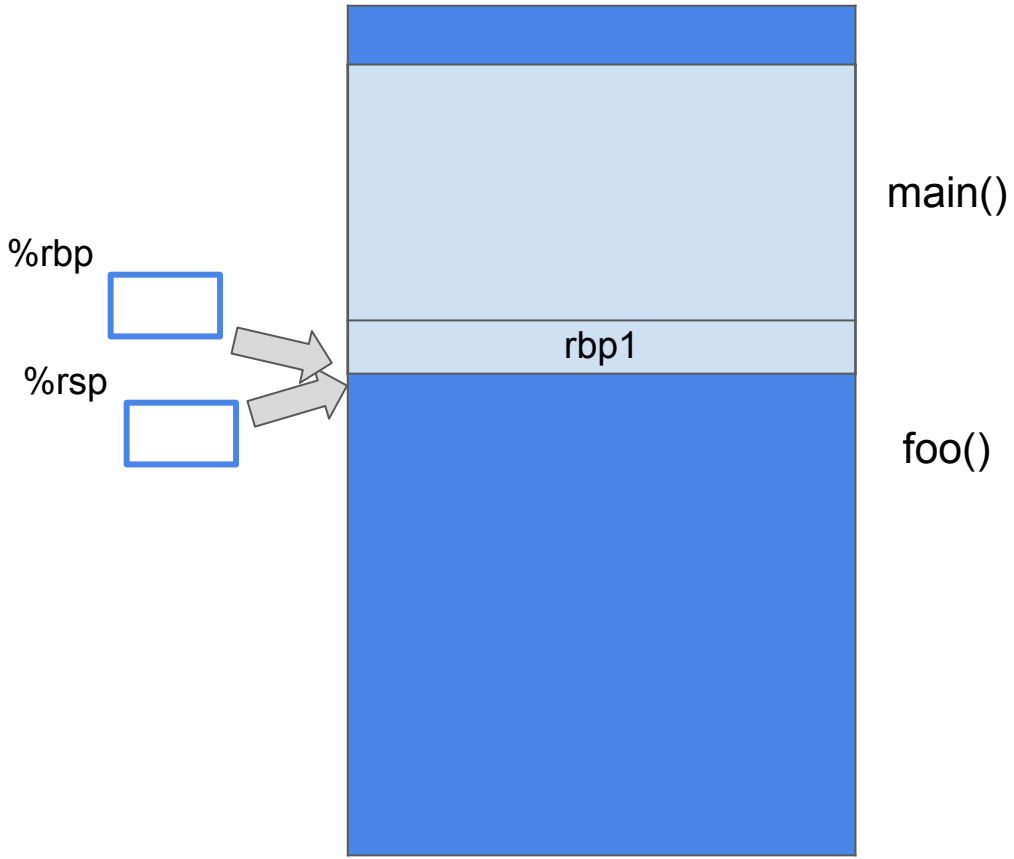
Execution Flow

```
0x0000000000400546 <foo1>:
      push rbp
      movq rsp, rbp
      sub 16, rsp
      movl $2, -0x4(rbp)
      movl -0x4(rbp), eax
      movq rbp, rsp
      pop rbp
      ret

0x0000000000400626 <foo>:
      push rbp
      movq rsp, rbp
      sub 16, rsp
      movl $5, -0x4(rbp)
      call 0x400546 <foo1>
      movq rbp, rsp
      pop rbp
      ret
```



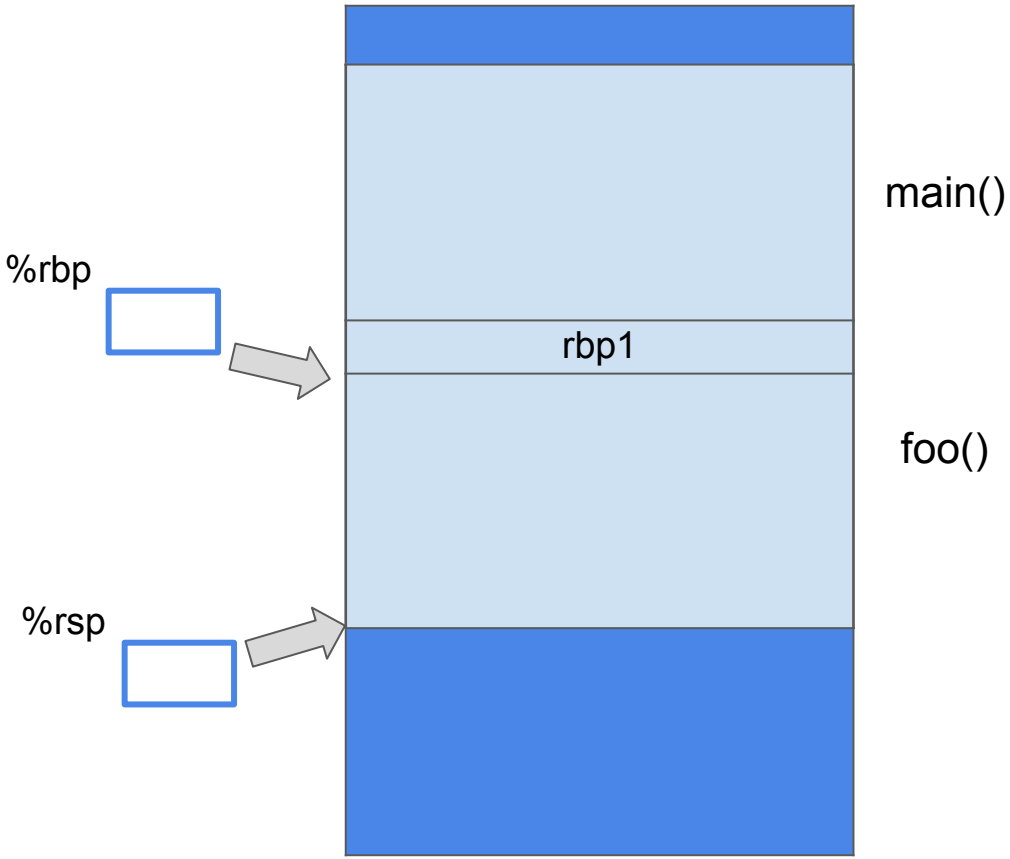
Execution Flow

```
0x0000000000400546 <foo1>:
      push rbp
      movq rsp, rbp
      sub 16, rsp
      movl $2, -0x4(rbp)
      movl -0x4(rbp), eax
      movq rbp, rsp
      pop rbp
      ret

0x0000000000400626 <foo>:
      push rbp
      movq rsp, rbp
      sub 16, rsp
      movl $5, -0x4(rbp)
      call 0x400546 <foo1>
      movq rbp, rsp
      pop rbp
      ret
```



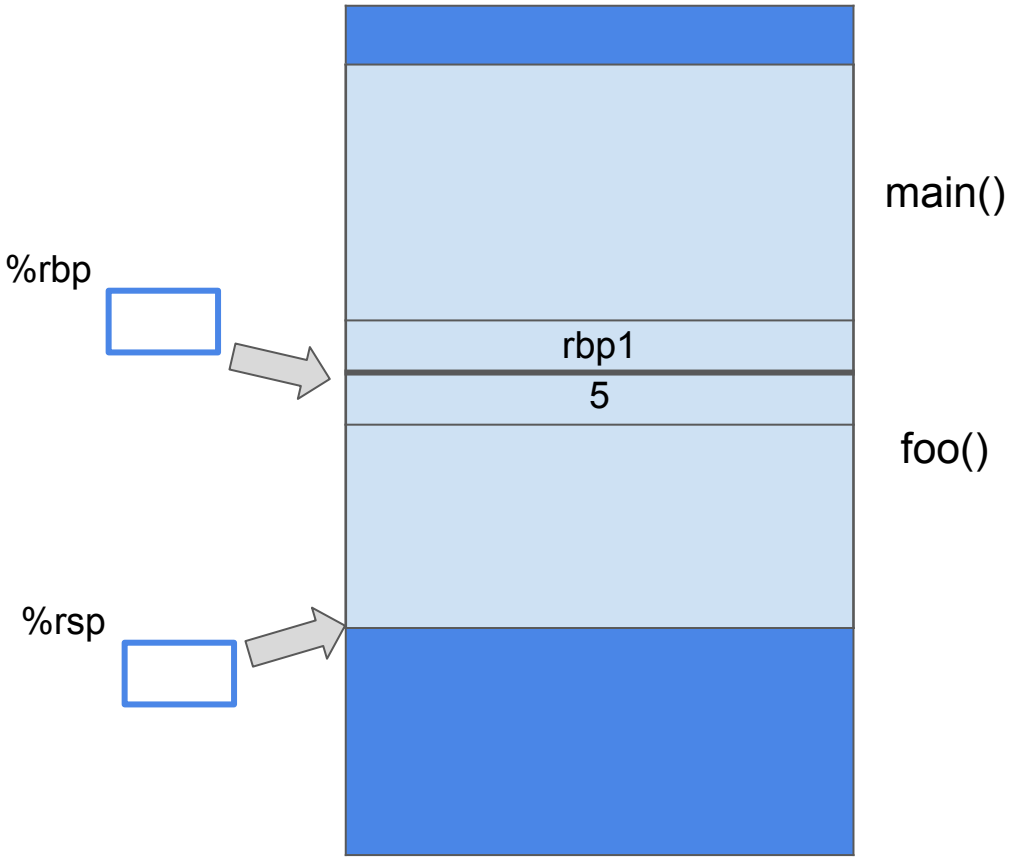
Execution Flow

```
0x0000000000400546 <foo1>:
      push rbp
      movq rsp, rbp
      sub 16, rsp
      movl $2, -0x4(rbp)
      movl -0x4(rbp), eax
      movq rbp, rsp
      pop rbp
      ret

0x0000000000400626 <foo>:
      push rbp
      movq rsp, rbp
      sub 16, rsp
      movl $5, -0x4(rbp)
      call 0x400546 <foo1>
      movq rbp, rsp
      pop rbp
      ret
```



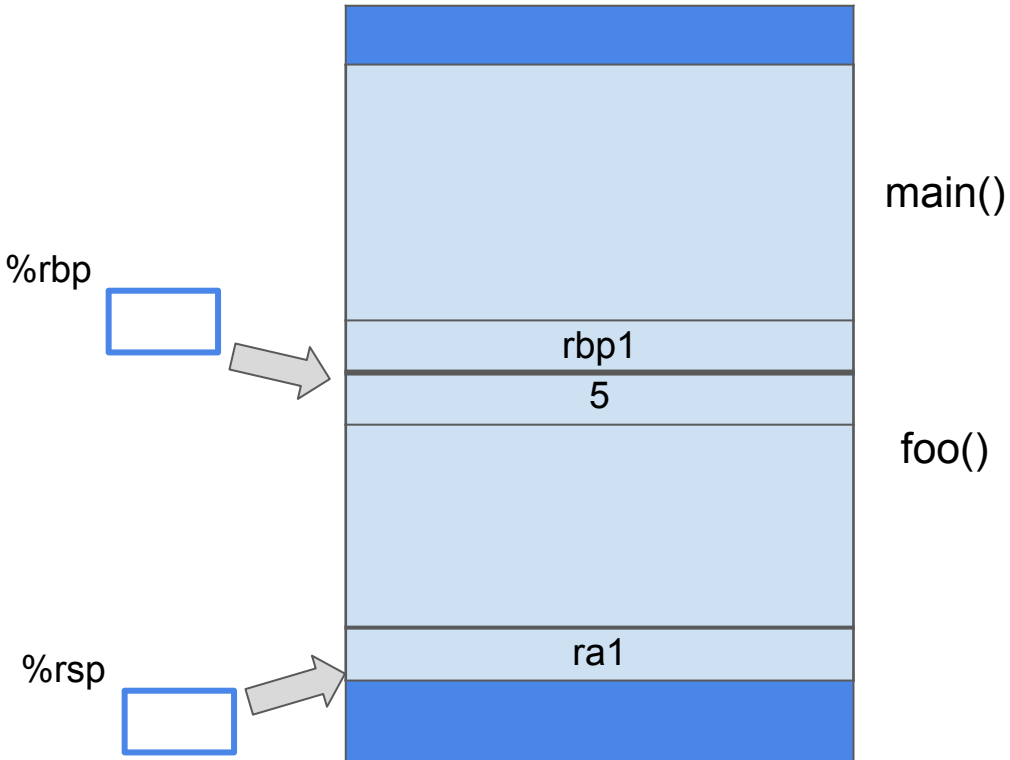
Execution Flow

```
0x0000000000400546 <foo1>:
      push rbp
      movq rsp, rbp
      sub 16, rsp
      movl $2, -0x4(rbp)
      movl -0x4(rbp), eax
      movq rbp, rsp
      pop rbp
      ret

0x0000000000400626 <foo>:
      push rbp
      movq rsp, rbp
      sub 16, rsp
      movl $5, -0x4(rbp)
      call 0x400546 <foo1>
      movq rbp, rsp
      pop rbp
      ret
```



Execution Flow

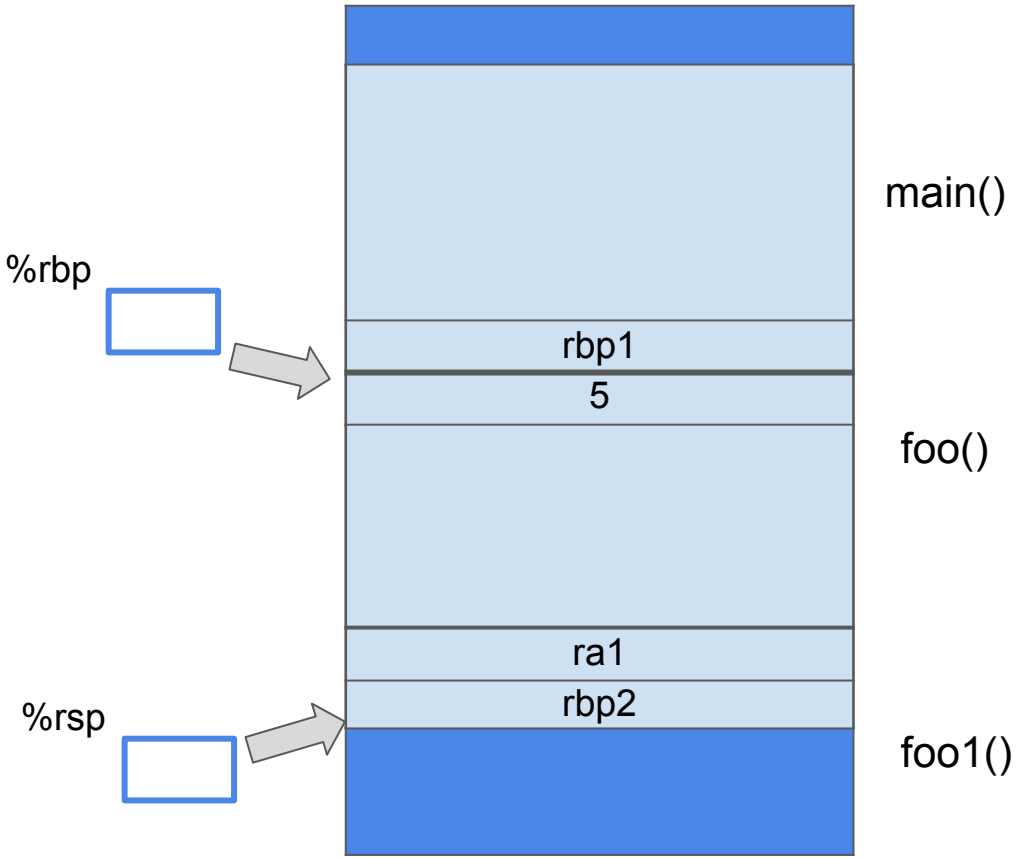
```
0x0000000000400546 <foo1>:
      push rbp
      movq rsp, rbp
      sub 16, rsp
      movl $2, -0x4(rbp)
      movl -0x4(rbp), eax
      movq rbp, rsp
      pop rbp
      ret

0x0000000000400626 <foo>:
      push rbp
      movq rsp, rbp
      sub 16, rsp
      movl $5, -0x4(rbp)
      call 0x400546 <foo1>
      movq rbp, rsp        ra1
      pop rbp
      ret
```



call pushes the address of the next instruction (ra1) to the stack and puts the address of foo1 label to the program counter (%rip)

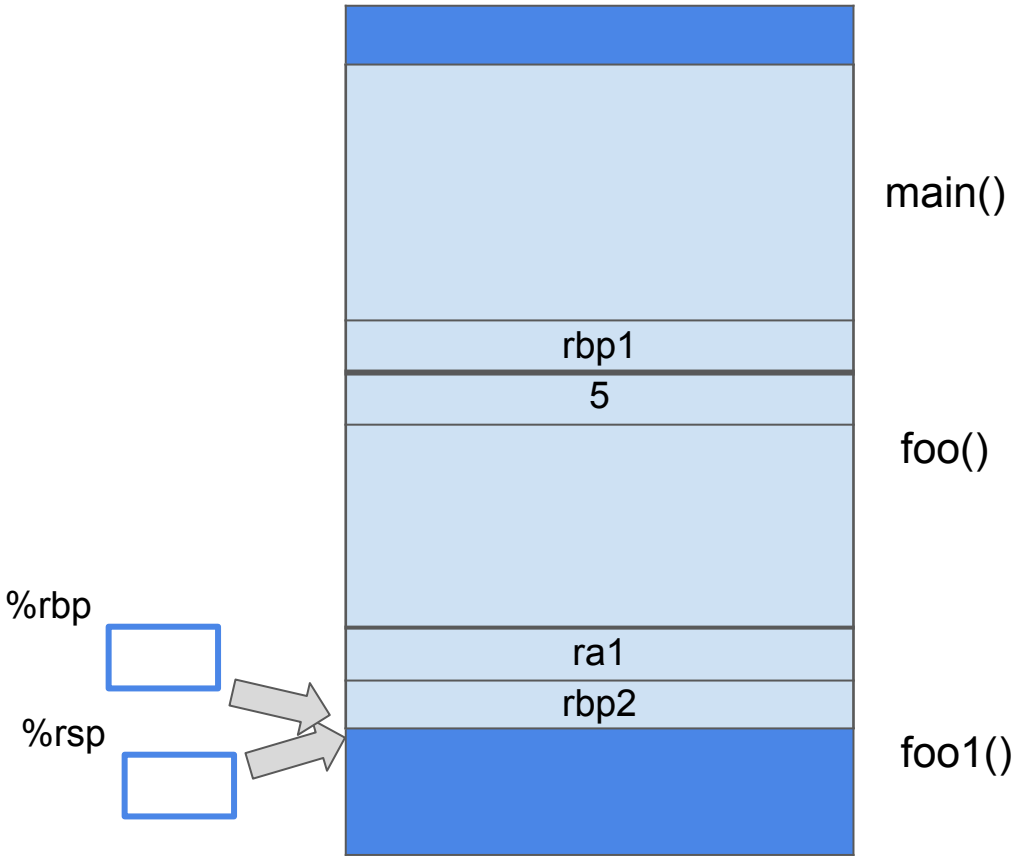
Execution Flow

```
0x0000000000400546 <foo1>:
      push rbp
      movq rsp, rbp
      sub 16, rsp
      movl $2, -0x4(rbp)
      movl -0x4(rbp), eax
      movq rbp, rsp
      pop rbp
      ret

0x0000000000400626 <foo>:
      push rbp
      movq rsp, rbp
      sub 16, rsp
      movl $5, -0x4(rbp)
      call 0x400546 <foo1>
      movq rbp, rsp
      pop rbp
      ret
```



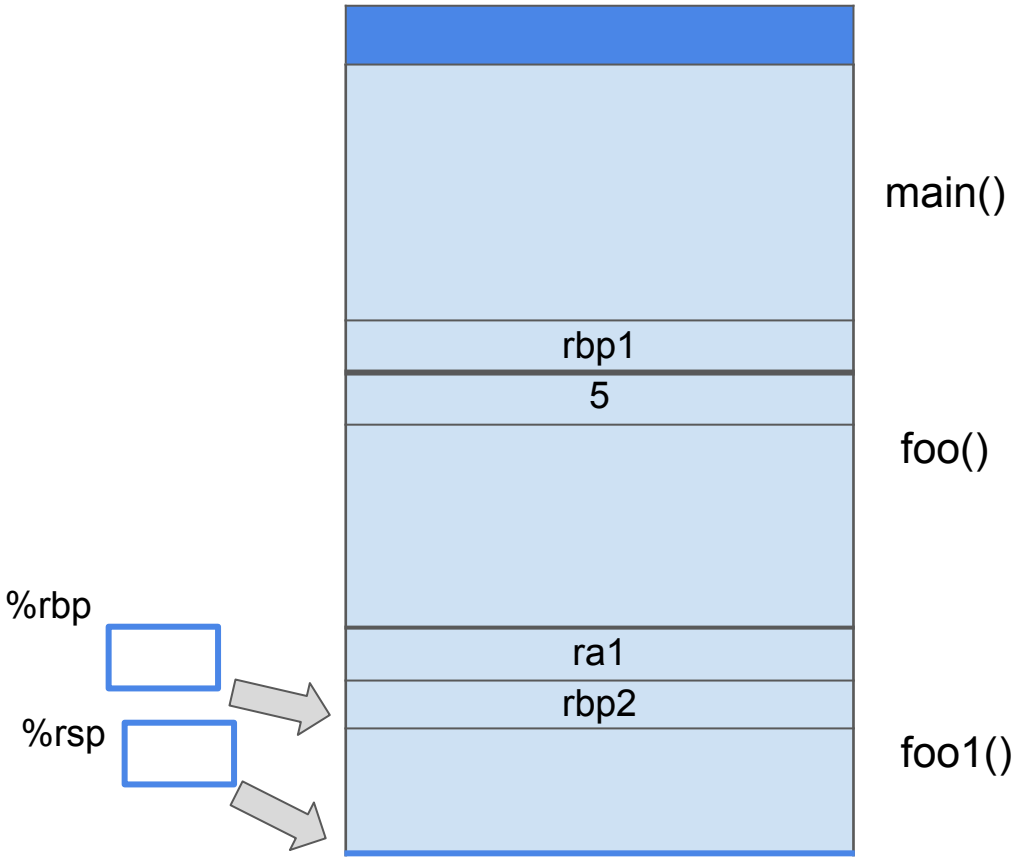
Execution Flow

```
0x0000000000400546 <foo1>:
      push rbp
      movq rsp, rbp
      sub 16, rsp
      movl $2, -0x4(rbp)
      movl -0x4(rbp), eax
      movq rbp, rsp
      pop rbp
      ret

0x0000000000400626 <foo>:
      push rbp
      movq rsp, rbp
      sub 16, rsp
      movl $5, -0x4(rbp)
      call 0x400546 <foo1>
      movq rbp, rsp
      pop rbp
      ret
```



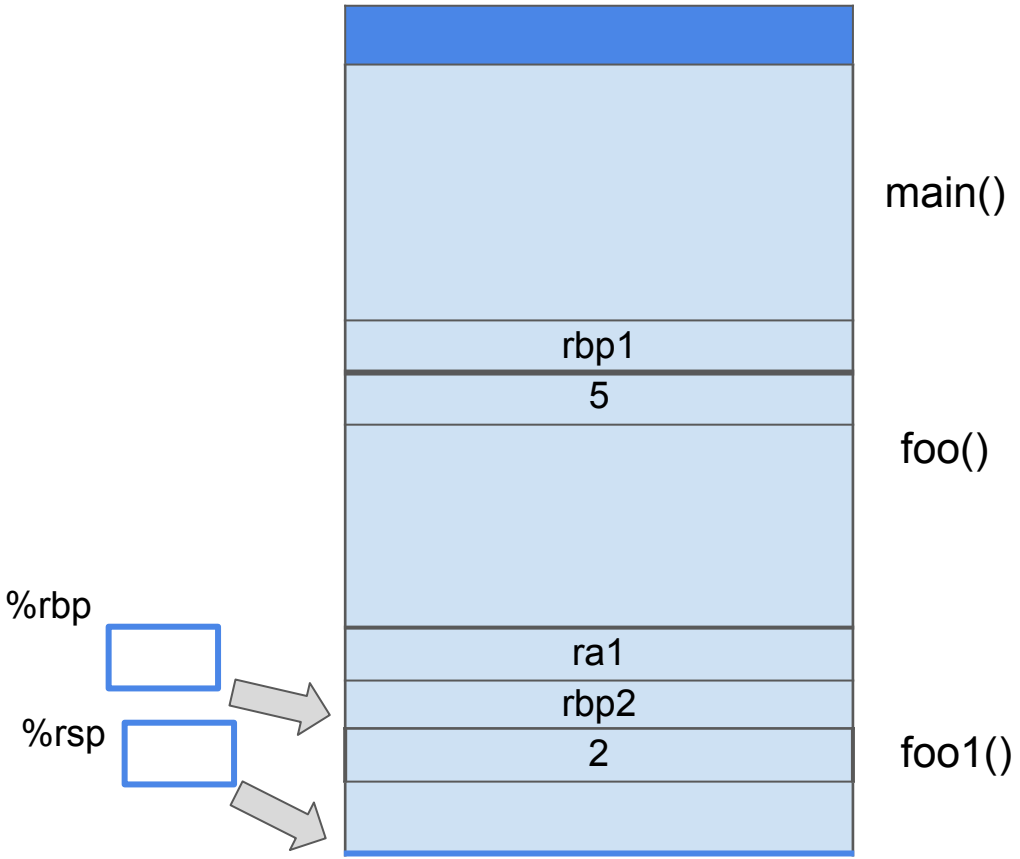
Execution Flow

```
0x0000000000400546 <foo1>:
      push rbp
      movq rsp, rbp
      sub 16, rsp
      movl $2, -0x4(rbp)
      movl -0x4(rbp), eax
      movq rbp, rsp
      pop rbp
      ret

0x0000000000400626 <foo>:
      push rbp
      movq rsp, rbp
      sub 16, rsp
      movl $5, -0x4(rbp)
      call 0x400546 <foo1>
      movq rbp, rsp
      pop rbp
      ret
```



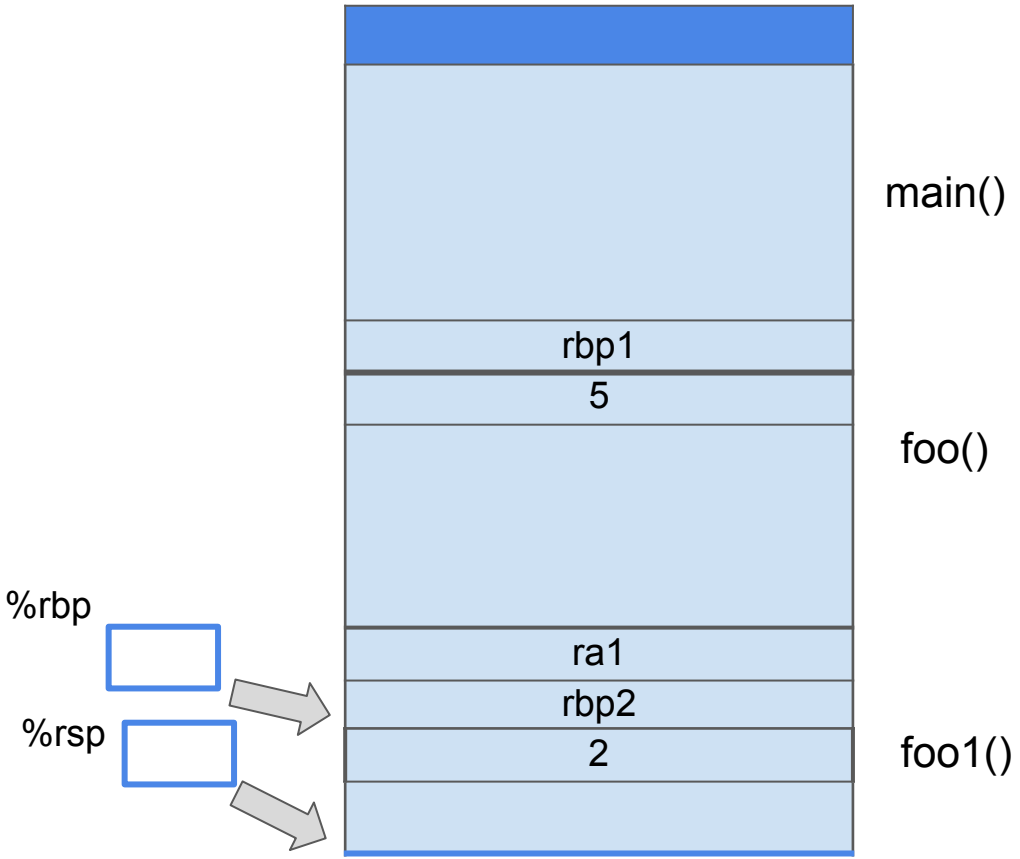
Execution Flow

```
0x0000000000400546 <foo1>:
      push rbp
      movq rsp, rbp
      sub 16, rsp
      movl $2, -0x4(rbp)
      movl -0x4(rbp), eax
      movq rbp, rsp
      pop rbp
      ret

0x0000000000400626 <foo>:
      push rbp
      movq rsp, rbp
      sub 16, rsp
      movl $5, -0x4(rbp)
      call 0x400546 <foo1>
      movq rbp, rsp
      pop rbp
      ret
```



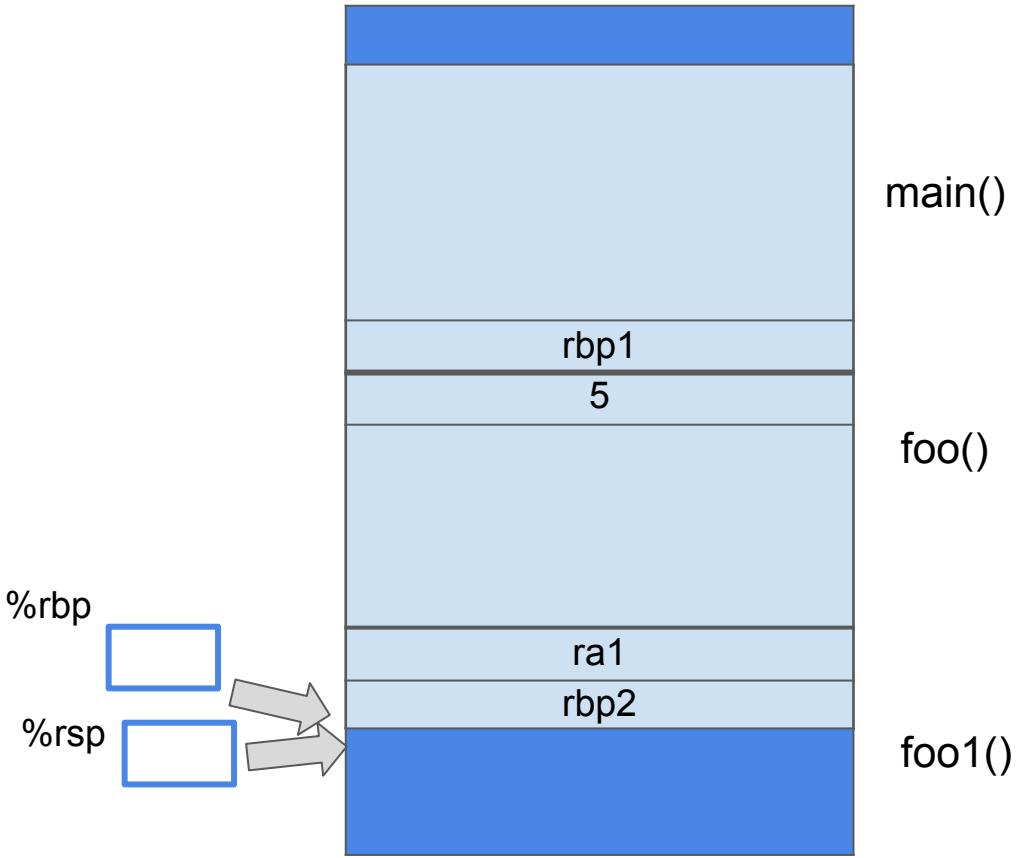
Execution Flow

```
0x0000000000400546 <foo1>:
      push rbp
      movq rsp, rbp
      sub 16, rsp
      movl $2, -0x4(rbp)
      movl -0x4(rbp), eax
      movq rbp, rsp
      pop rbp
      ret

0x0000000000400626 <foo>:
      push rbp
      movq rsp, rbp
      sub 16, rsp
      movl $5, -0x4(rbp)
      call 0x400546 <foo1>
      movq rbp, rsp
      pop rbp
      ret
```



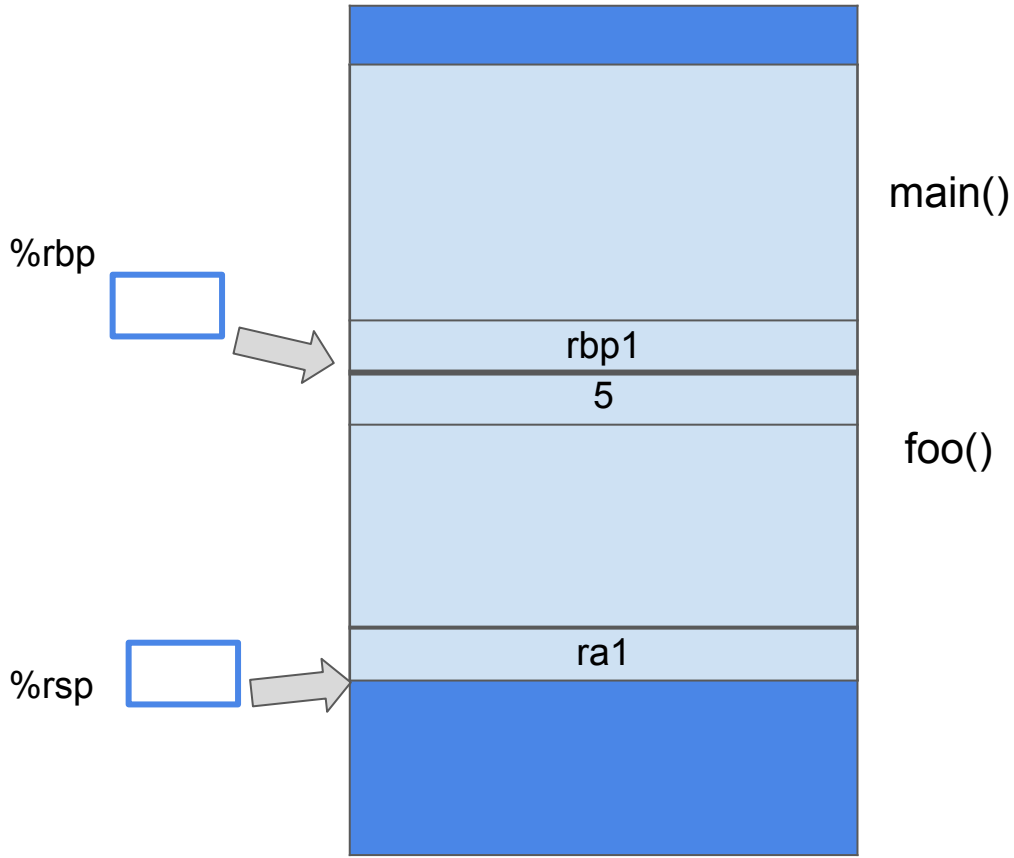
Execution Flow

```
0x0000000000400546 <foo1>:
      push rbp
      movq rsp, rbp
      sub 16, rsp
      movl $2, -0x4(rbp)
      movl -0x4(rbp), eax
      movq rbp, rsp
      pop rbp
      ret

0x0000000000400626 <foo>:
      push rbp
      movq rsp, rbp
      sub 16, rsp
      movl $5, -0x4(rbp)
      call 0x400546 <foo1>
      movq rbp, rsp
      pop rbp
      ret
```



Execution Flow

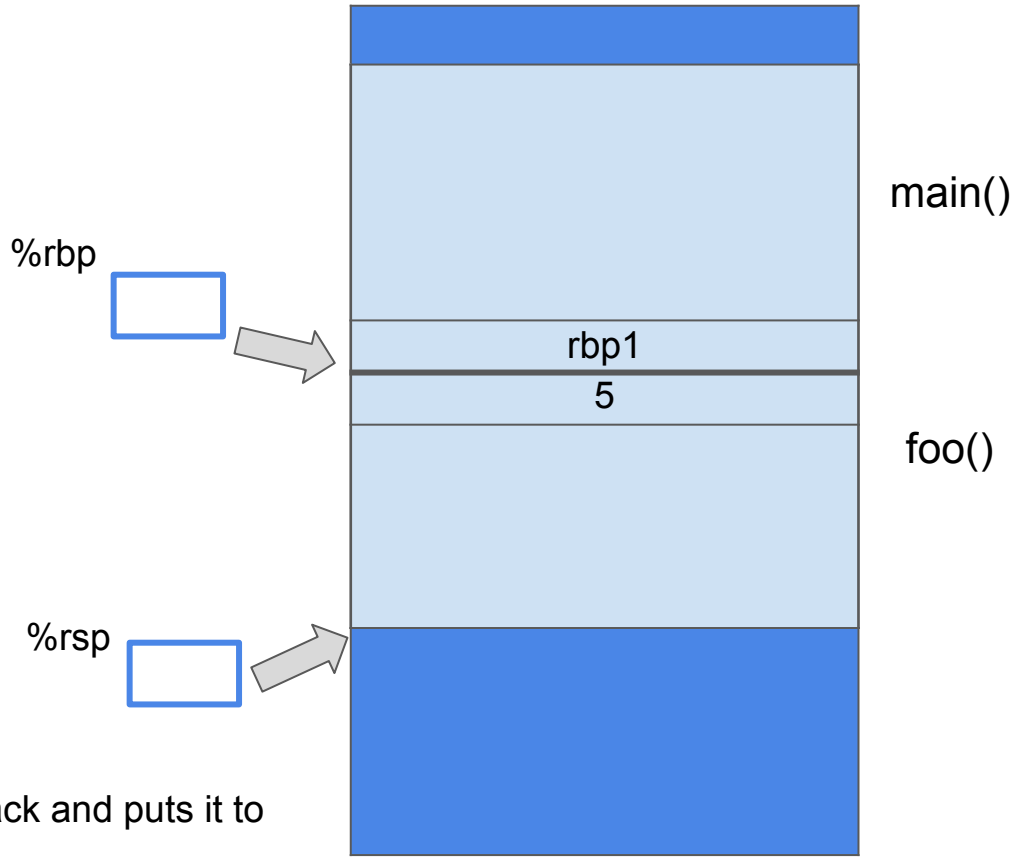
```
0x0000000000400546 <foo1>:
      push rbp
      movq rsp, rbp
      sub 16, rsp
      movl $2, -0x4(rbp)
      movl -0x4(rbp), eax
      movq rbp, rsp
      pop rbp
      ret

0x0000000000400626 <foo>:
      push rbp
      movq rsp, rbp
      sub 16, rsp
      movl $5, -0x4(rbp)
      call 0x400546 <foo1>
      movq rbp, rsp
      pop rbp
      ret
```



Execution Flow

```
0x0000000000400546 <foo1>:
      push rbp
      movq rsp, rbp
      sub 16, rsp
      movl $2, -0x4(rbp)
      movl -0x4(rbp), eax
      movq rbp, rsp
      pop rbp
      ret

0x0000000000400626 <foo>:
      push rbp
      movq rsp, rbp
      sub 16, rsp
      movl $5, -0x4(rbp)
      call 0x400546 <foo1>
      movq rbp, rsp
      pop rbp
      ret
```

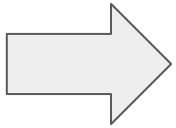
ra1



ret pops the return address (ra1) from the stack and puts it to %rip.

How to pass parameters to a called function??

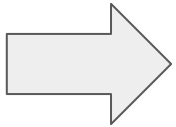
```
int foo1(int a, int b, int c)
{
     return a+b+c;
}

int foo()
{
     return foo1(1,2,3);
}
```



```
0x0000000000400546 <foo1>:
     push %rbp
     movq %rsp, %rbp
     movl edi, -0x4(%rbp)
     movl esi, -0x8(%rbp)
     movl edx, -0xc(%rbp)
     movl -0x4(rbp), %edx
     movl -0x8(rbp), %eax
     addl %eax, %edx
     movl -0xc(%rbp), %eax
     addl edx, %eax
     pop %rbp
     ret
0x0000000000400626 <foo>:
     push %rbp
     movq %rsp, %rbp
     movl $3, %edx
     movl $2, %esi
     movl $1, %edi
     call 0x400546 <foo1>
     pop rbp
     ret
```

How to pass parameters to a called function??

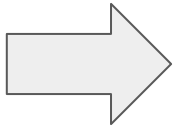
```
int foo1(int a, int b, int c,int d, int e, int f)
{
     // Some statement here;
}

int foo()
{
     return foo1(1,2,3,4,5,6);
}
```



```
0x0000000000400546 <foo1>:
     # Some statement here;
0x0000000000400626 <foo>:
     push %rbp
     movq %rsp, %rbp
     movl $1, %edi
     movl $2, %esi
     movl $3, %edx
     movl $4, %ecx
     movl $5, %r8
     movl $6, %r9
     call 0x400546 <foo1>:
     pop %rbp
     ret
```

How to pass parameters to a called function??

```
int foo1(int a, int b, int c,
         int d, int e, int f,
         int g, int h)
{
     // Some statement here;
}

int foo()
{
     return foo1(1,2,3,4,5,6,7,8);
}
```



```
0x0000000000400546 <foo1>:
     # Some statement here;
0x0000000000400626 <foo>:
     push %rbp
     movq %rsp, %rbp
     subl $16, %rsp
     movl $1, %edi
     movl $2, %esi
     movl $3, %edx
     movl $4, %ecx
     movl $5, %r8d
     movl $6, %r9d
     push $8
     push $7
     call 0x400546 <foo1>
     addl $16, %rsp
     ret
```

Key Points

- Stack grows downward
- %rbp
- First 6 arguments
- Return value
- Caller saved
- Calle saved
- Stack 16-byte aligned

Additional Resources

- https://users.ece.utexas.edu/~adnan/gdb-refcard.pdf
- https://www.geeksforgeeks.org/linux-unix/gdb-command-in-linux-with-examples/