# Runtime Stack

COMP201 Lab Session

Spring 2022

KOÇ UNIVERSITY

# Recap: x86-64 Register Conventions

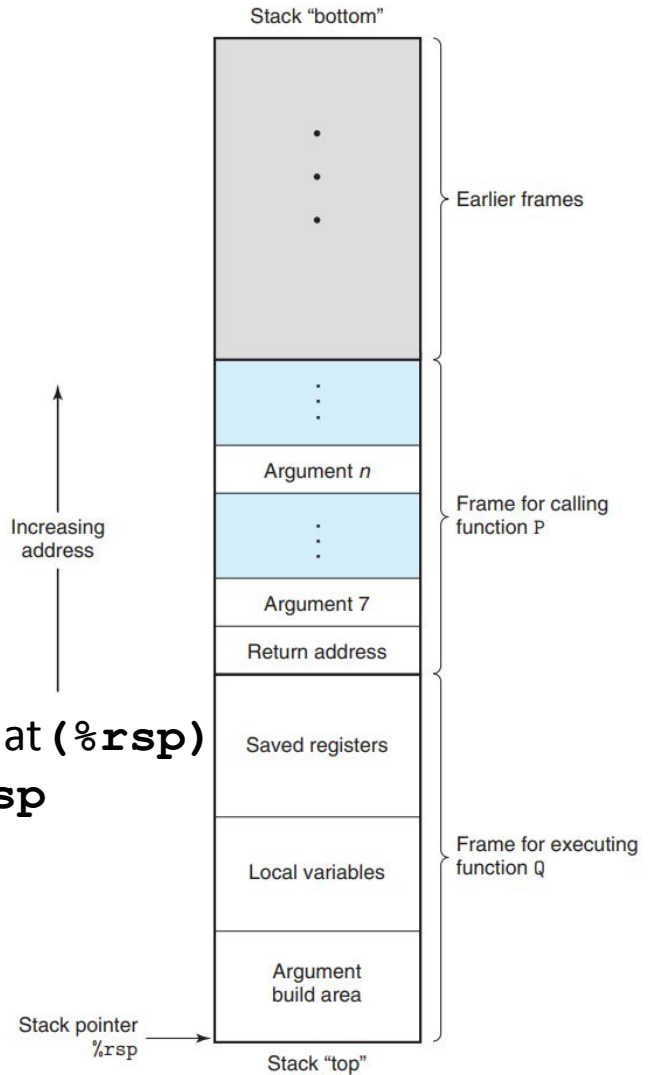
- **Arguments passed in registers:**
  - `%rdi, %rsi, %rdx, %rcx, %r8, %r9`
- **Return value:** `%rax`

- **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 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 Function Call Setup

Caller:

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

Callee:

- Push any callee-saved registers, decrease `%rsp` to make room for new frame

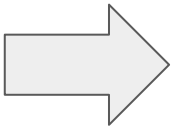
# Recap: x86-64 Function Call Return

Callee:

- Increase `%rsp`, pop any callee-saved registers (in **reverse order**),
- execute `ret: pop %rip`

# Example Code

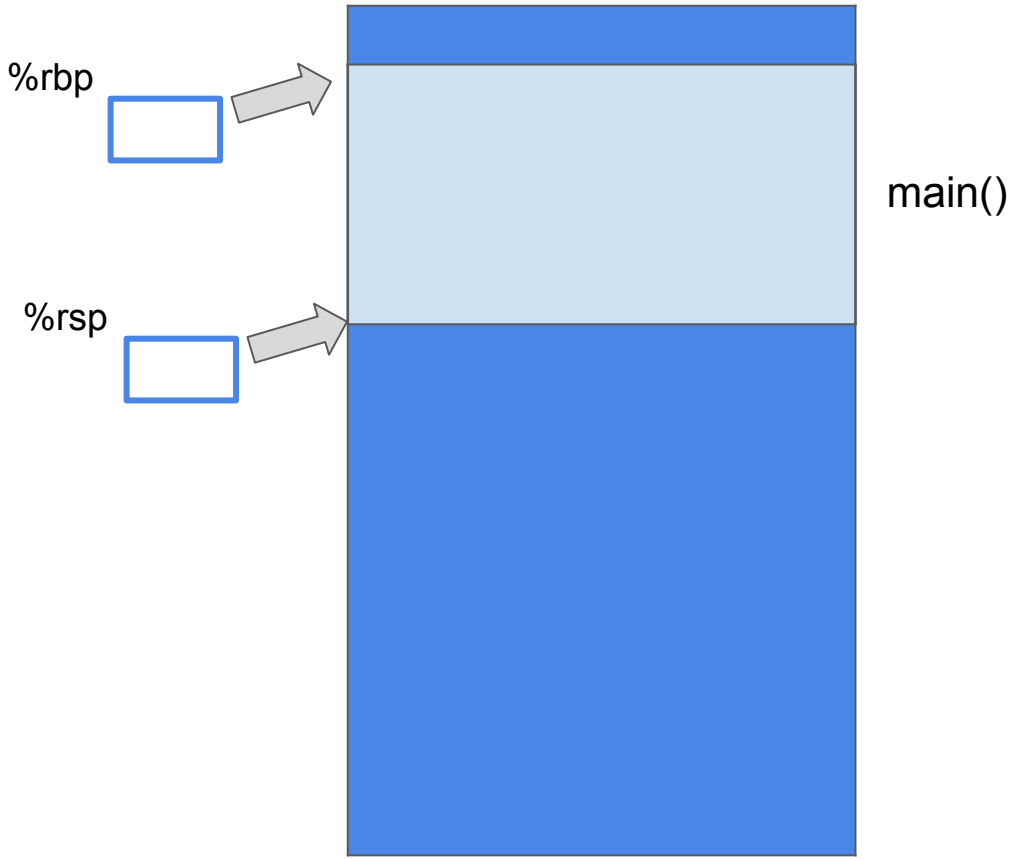
```
int foo1()
{
     int i = 2;
     return i;
}

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
```

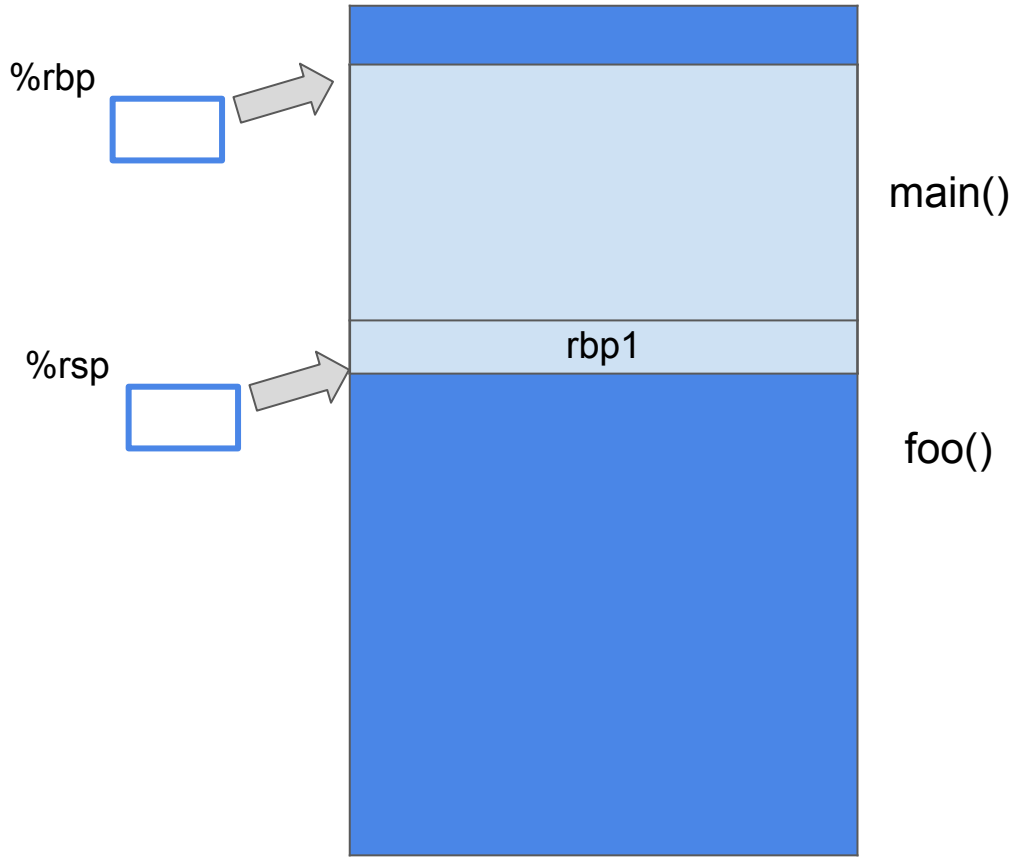
# Execution Flow

```
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
```



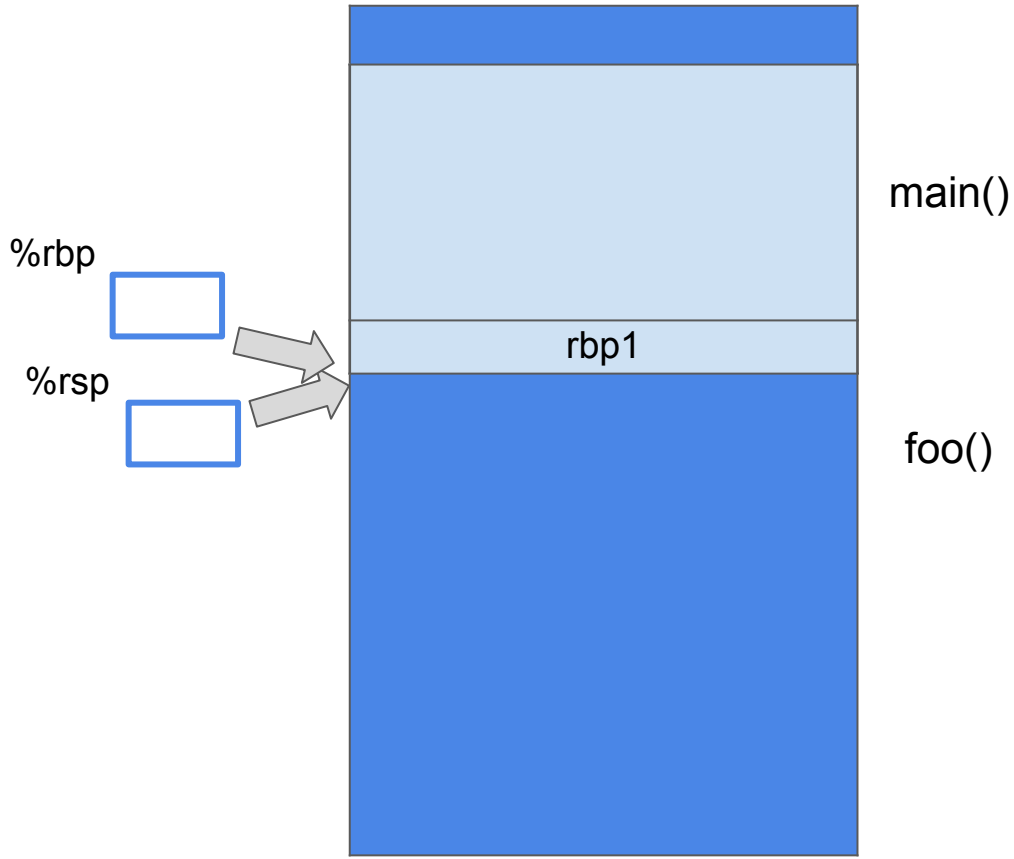
# Execution Flow

```
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
```



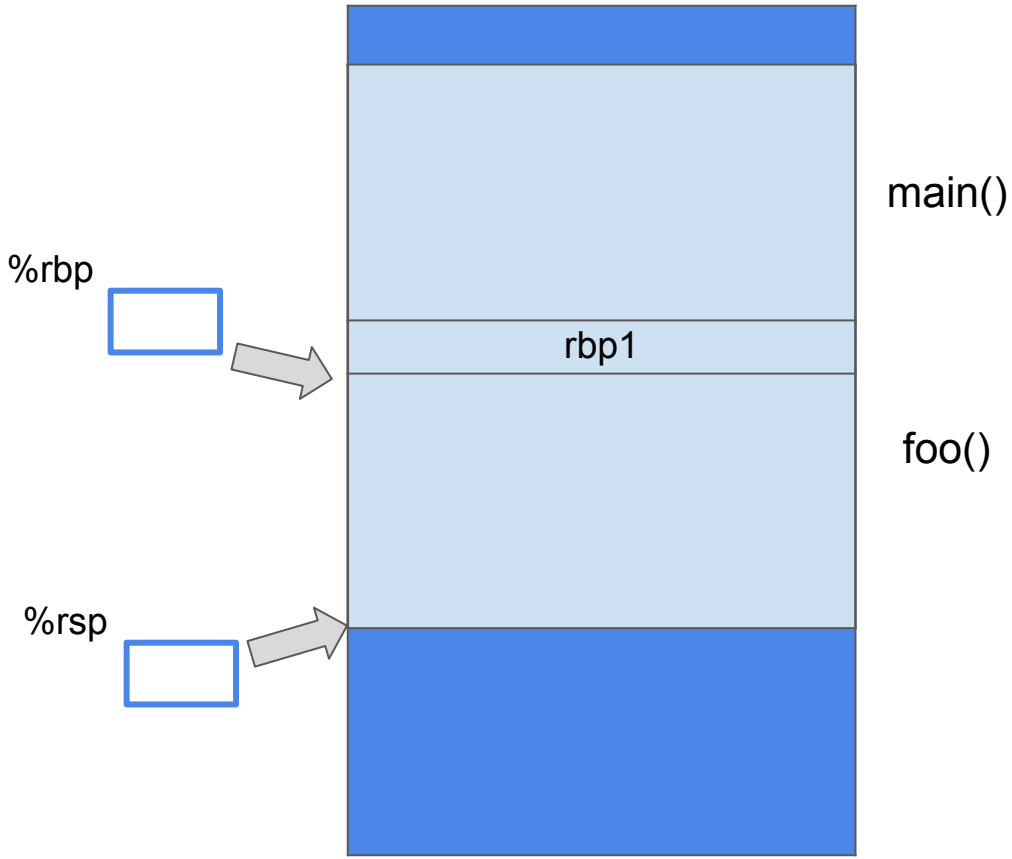
# Execution Flow

```
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
```



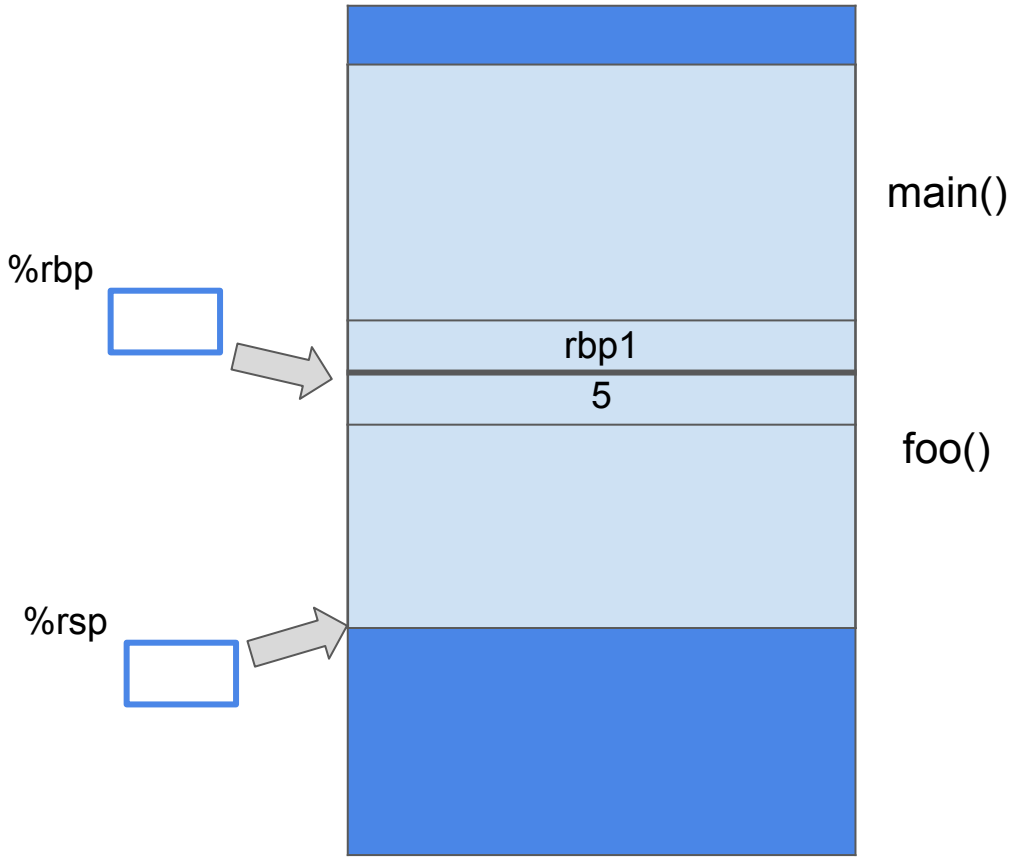
# Execution Flow

```
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

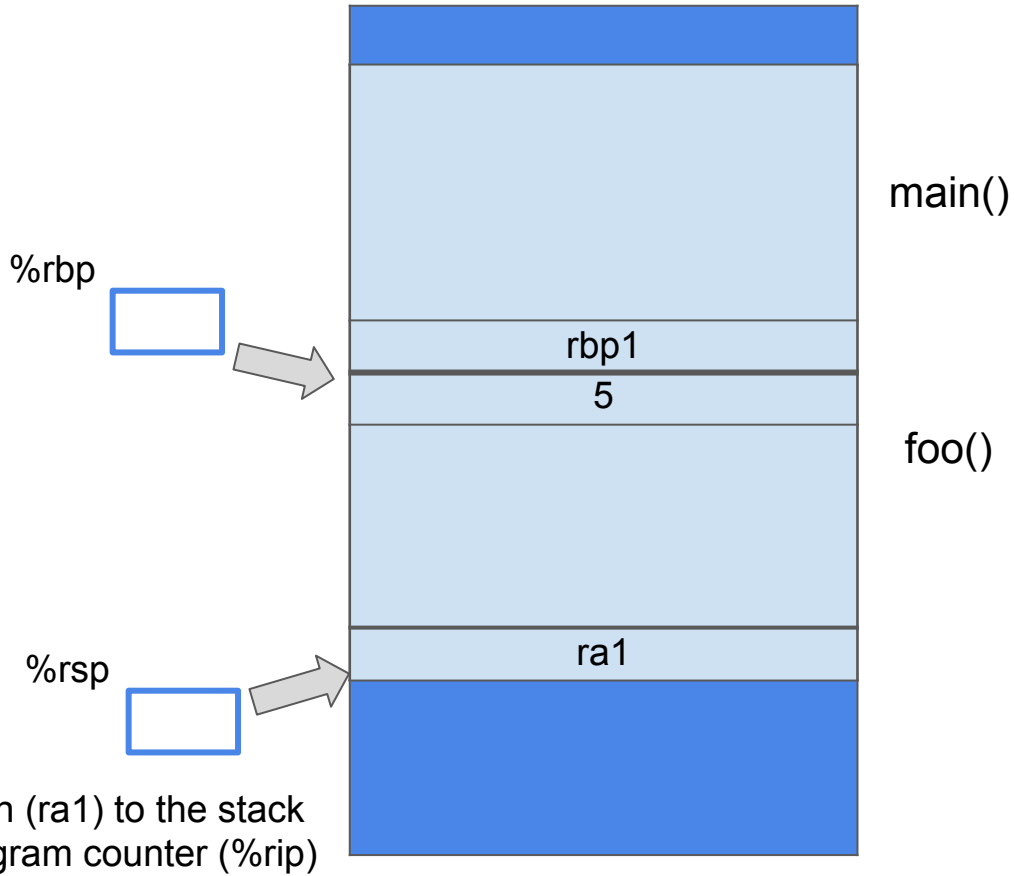
```
0x0000000000400546 <foo1>:
        push rbp
        movq rsp, rbp
        sub 16, rsp
        movl $2, -0x4(rbp)
        movl -0x4(rbp), eax
        movq rbp, rsp
        pop rbp
        ret
0x0000000000400626 <foo>:
        push rbp
        movq rsp, rbp
        sub 16, rsp
        movl $5, -0x4(rbp)
        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
        pop rbp
        ret
```

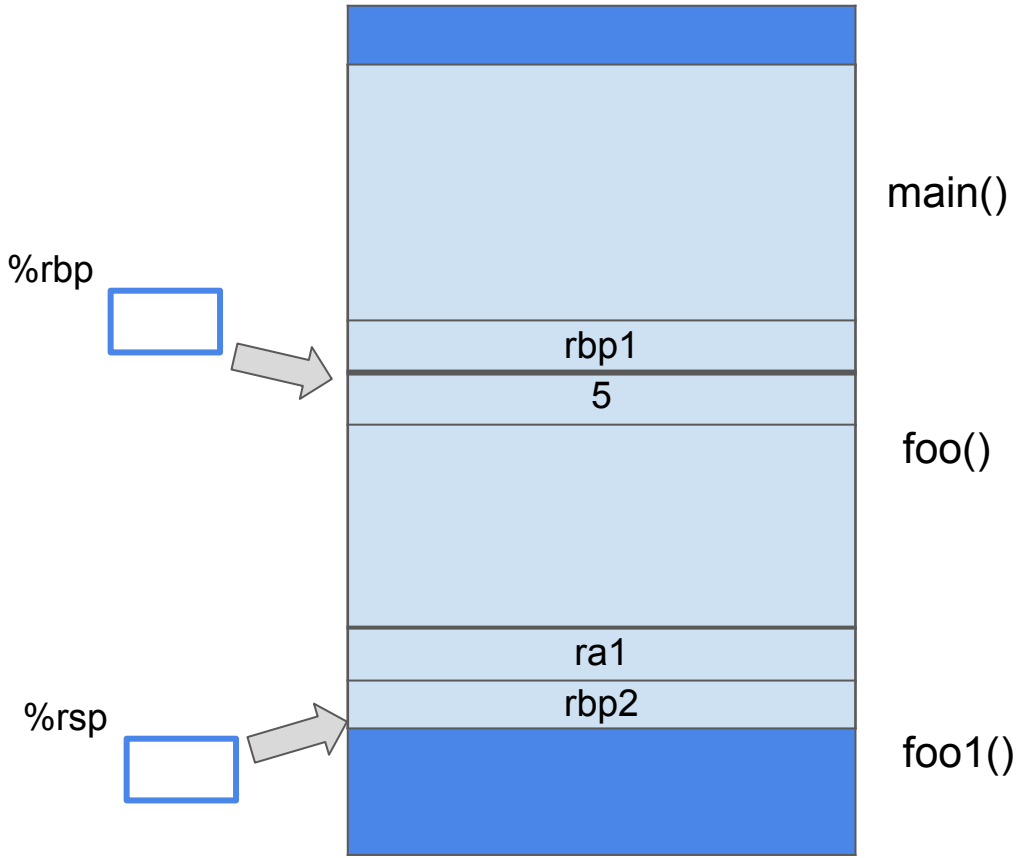
ra1




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

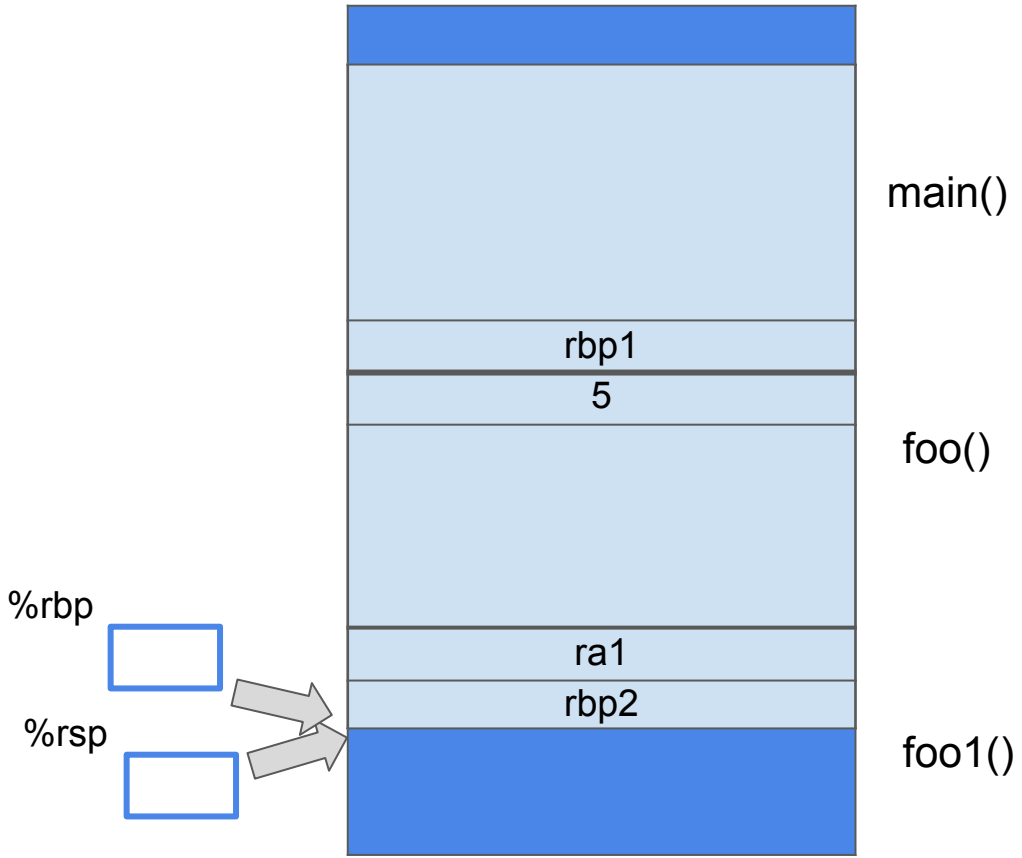
# Execution Flow

```
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
```



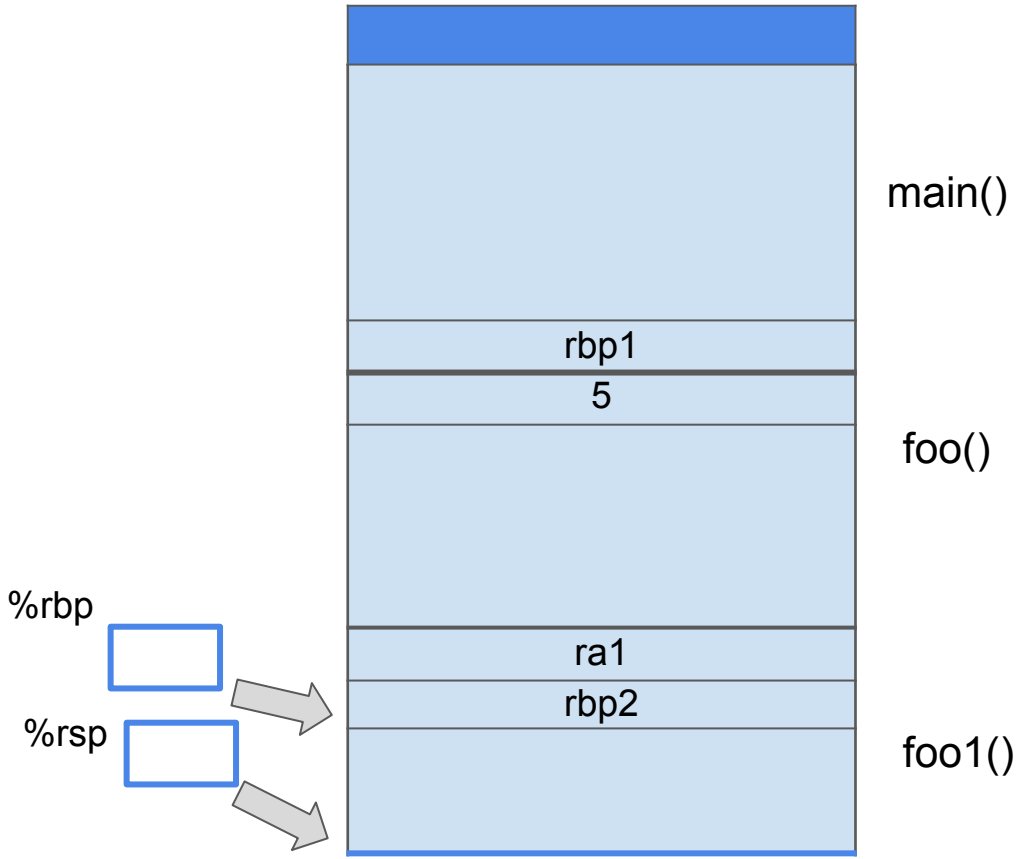
# Execution Flow

```
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
```



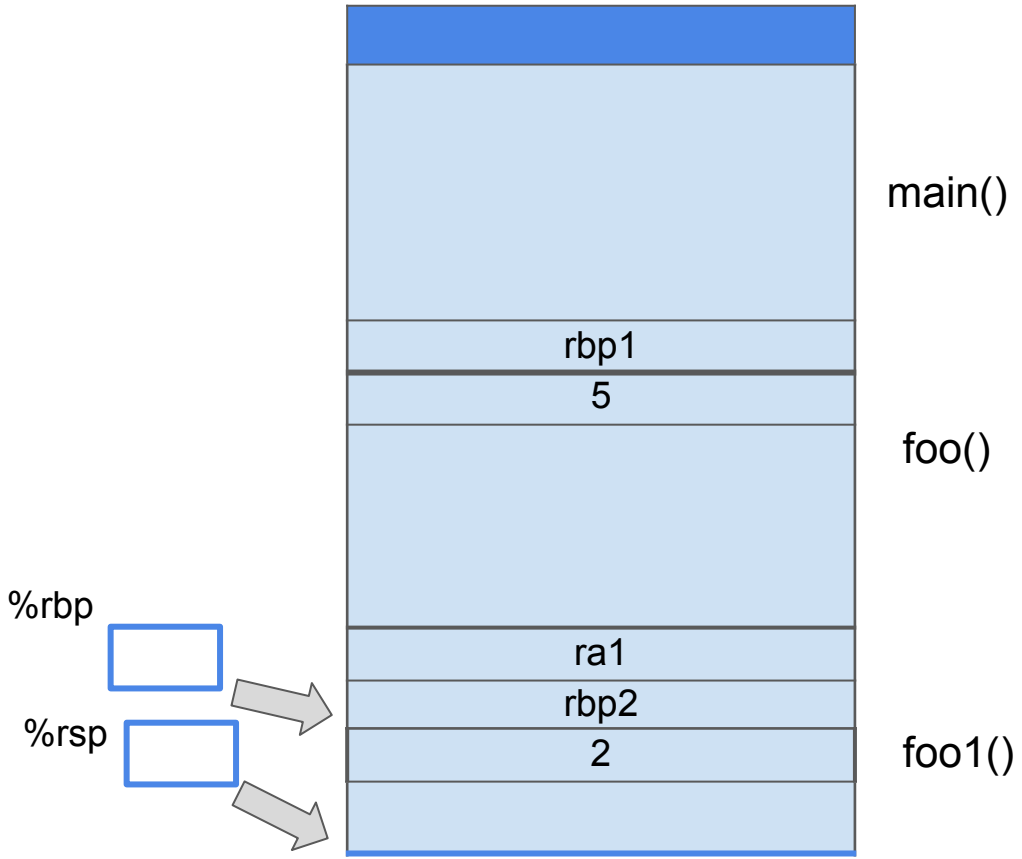
# Execution Flow

```
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
```



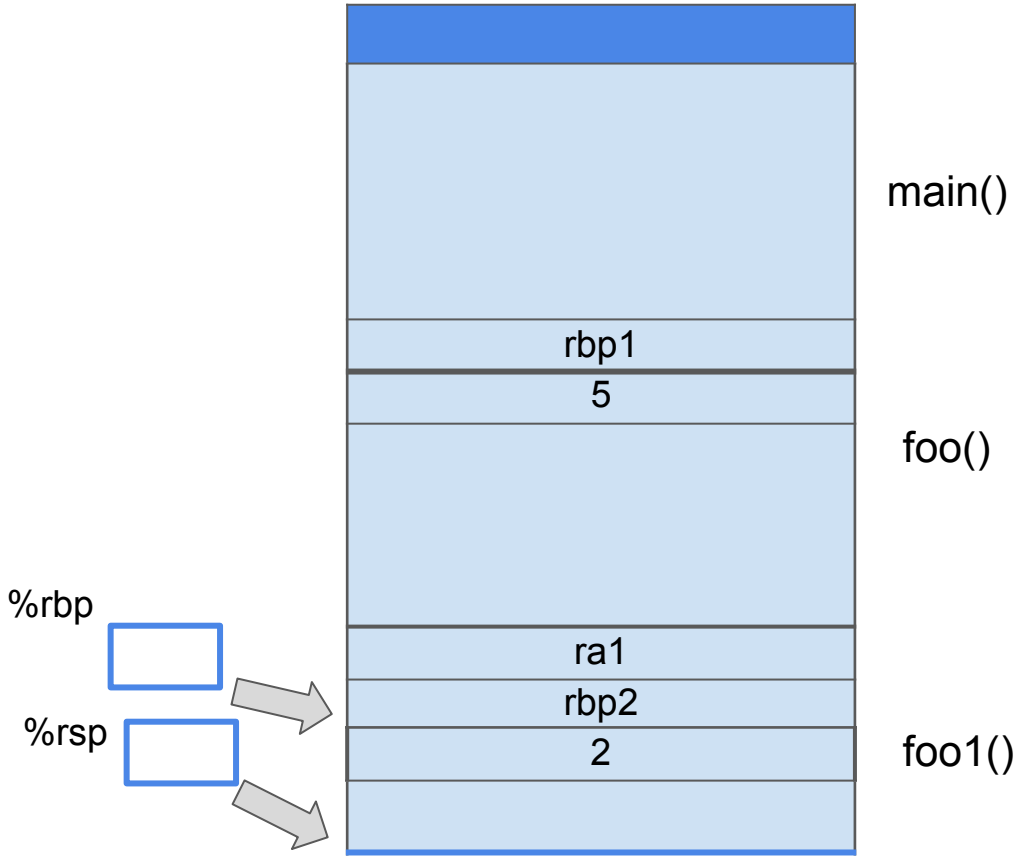
# Execution Flow

```
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
```



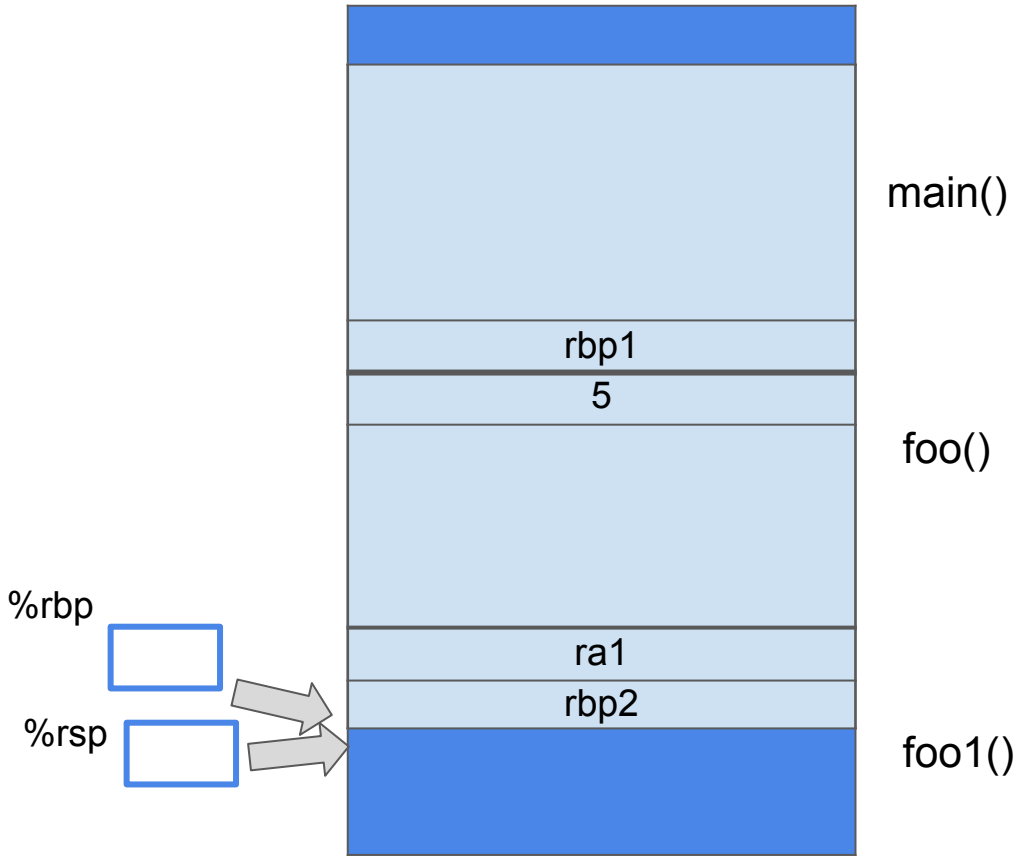
# Execution Flow

```
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
```



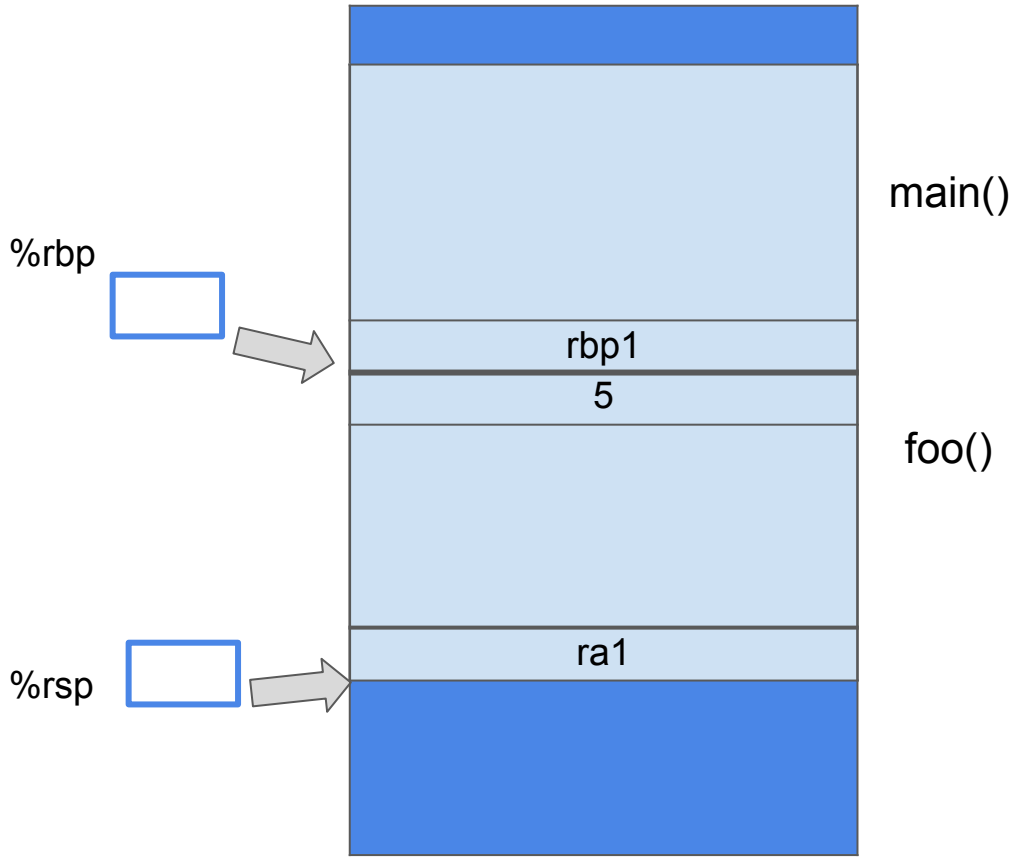
# Execution Flow

```
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
```



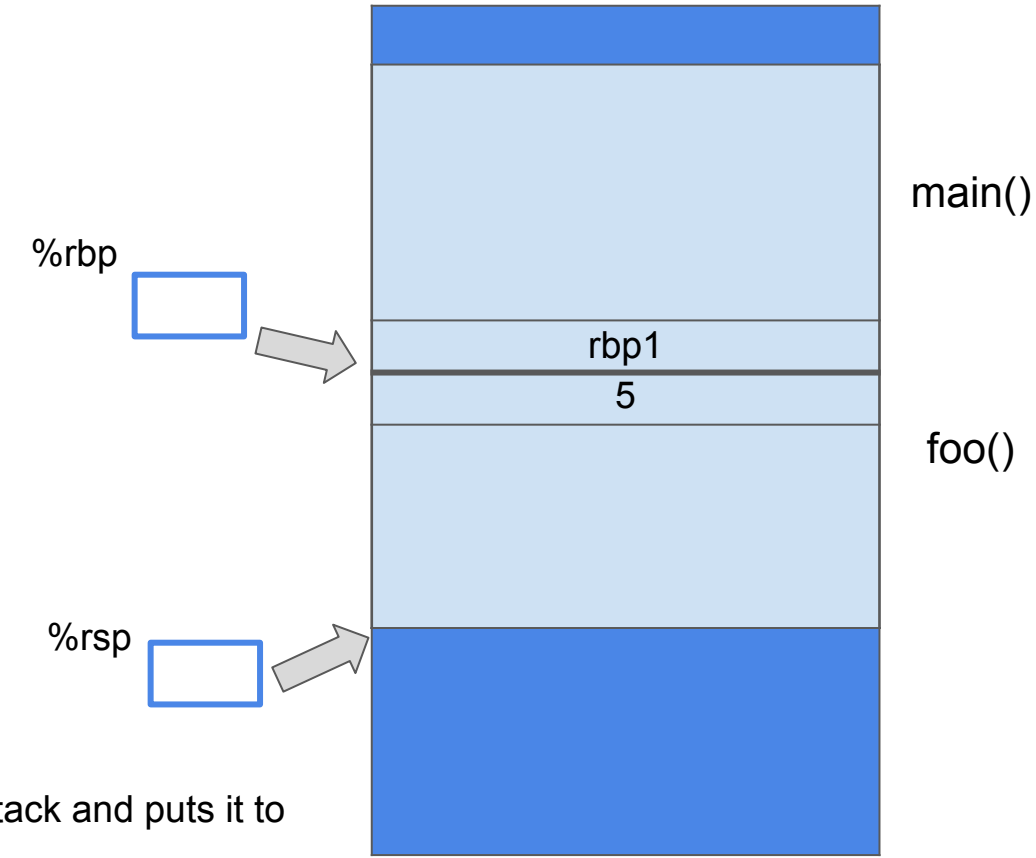
# Execution Flow

```
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

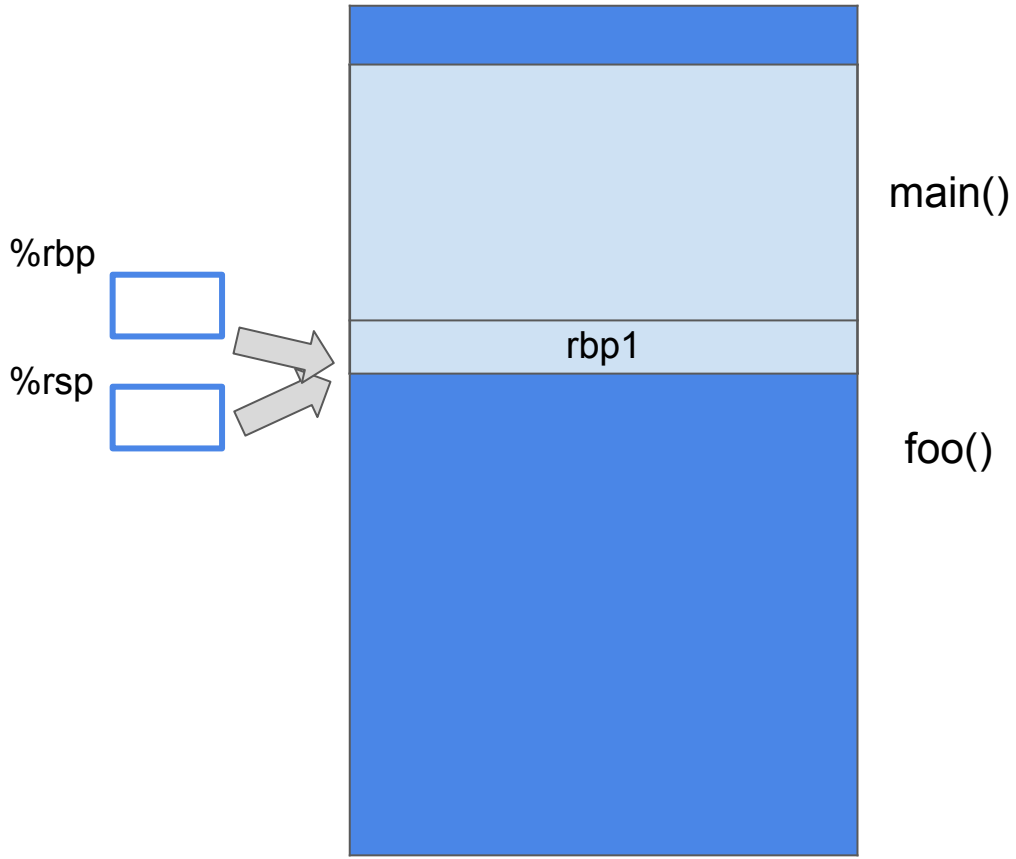
```
0x0000000000400546 <foo1>:
        push rbp
        movq rsp, rbp
        sub 16, rsp
        movl $2, -0x4(rbp)
        movl -0x4(rbp), eax
        movq rbp, rsp
        pop rbp
        ret
0x0000000000400626 <foo>:
        push rbp
        movq rsp, rbp
        sub 16, rsp
        movl $5, -0x4(rbp)
        call 0x400546 <foo1>
        movq rbp, rsp   <= ra1
        pop rbp
        ret
```




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

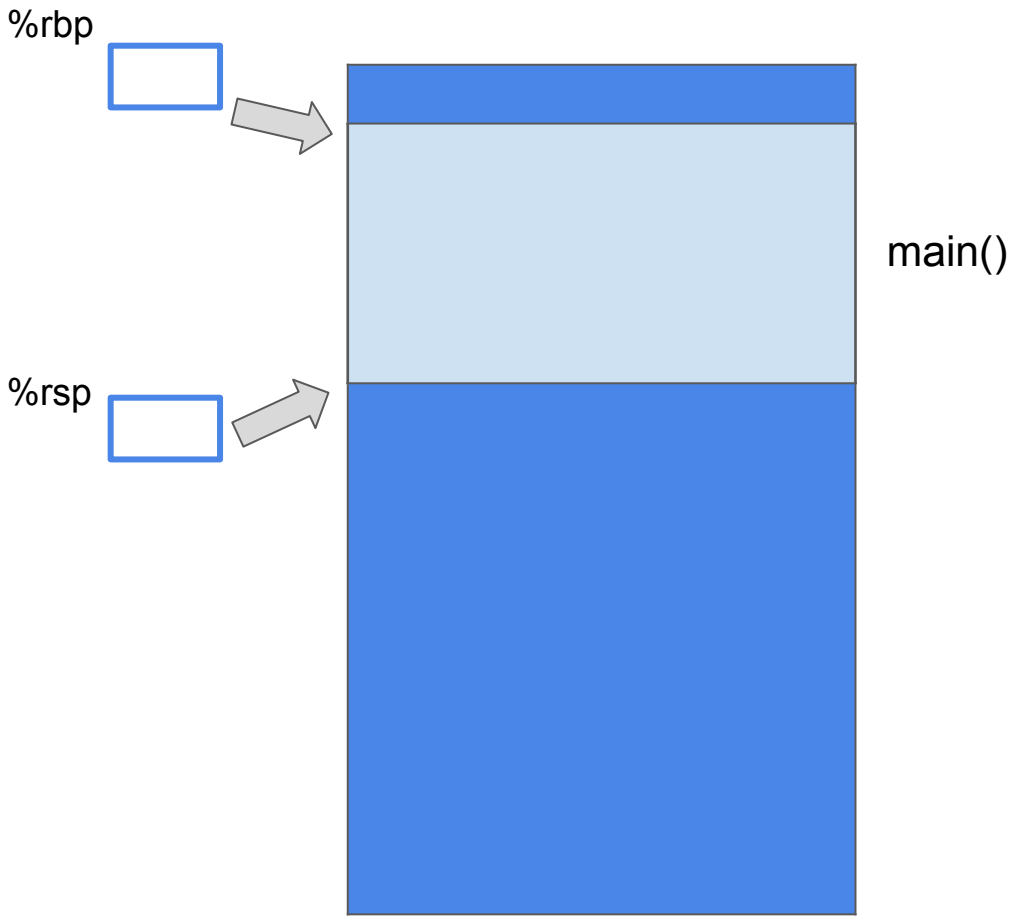
# Execution Flow

```
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
```



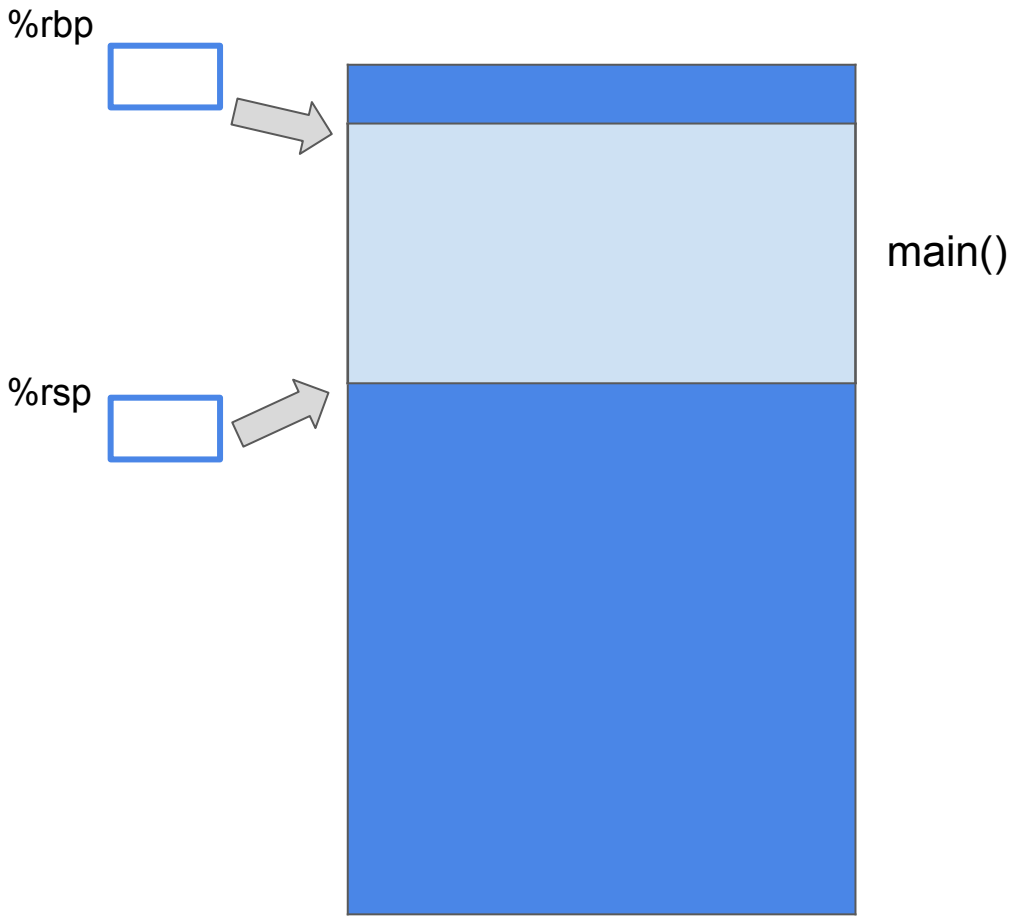
# Execution Flow

```
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
```



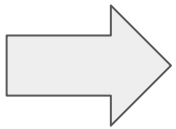
# Execution Flow

```
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
```



# How to pass parameters to a called function??

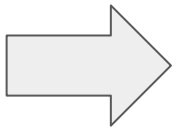
```
int foo1(int a, int b, int c)
{
        return a+b+c;
}
int foo()
{
        return foo1(1,2,3);
}
```



```
0x0000000000400546 <foo1>:
        push   rbp
        movq rsp, rbp
        movl edi, -0x4(rbp)
        movl esi, -0x8(rbp)
        movl edx, -0xc(rbp)
        movl -0x4(rbp), edx
        mov -0x8(rbp), eax
        add eax, edx
        mov -0xc(rbp), eax
        add 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??

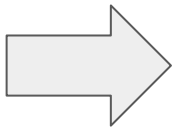
```
int foo1(int a, int b, int c, int d, int e,
int f)
{
        …….
}
int foo()
{
        return foo1(1,2,3,4,5,6);
}
```



```
0x0000000000400546 <foo1>:
        ……..
0x0000000000400626 <foo>:
        push rbp
        mov rsp, rbp
        movl $6, r9d
        movl $5, r8d
        movl $4, ecx
        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, int g, int h)
{
        ……..
}
int foo()
{
        return foo1(1,2,3,4,5,6,7,8);
}
```



```
0x0000000000400546 <foo1>:
        ……..
0x0000000000400626 <foo>:
        push rbp
        mov rsp, rbp
        sub 16, rsp
        push 8
        push 7
        mov $6, r9d
        mov $5, r8d
        mov $4, ecx
        mov $3, edx
        mov $2, esi
        mov $1, edi
        call 0x400546 <foo1>
        add 16, rsp
        leave
        ret
```

# Getting the assembly code

1. If you have the source code, you may use -S flag of GCC.

   *gcc -S -o asm_output.s helloworld.c*

2. If you have the executable, you may use objdump.

   *objdump --disassemble helloworld > helloworld.dump*

# Compiler Explorer (godbolt.org)