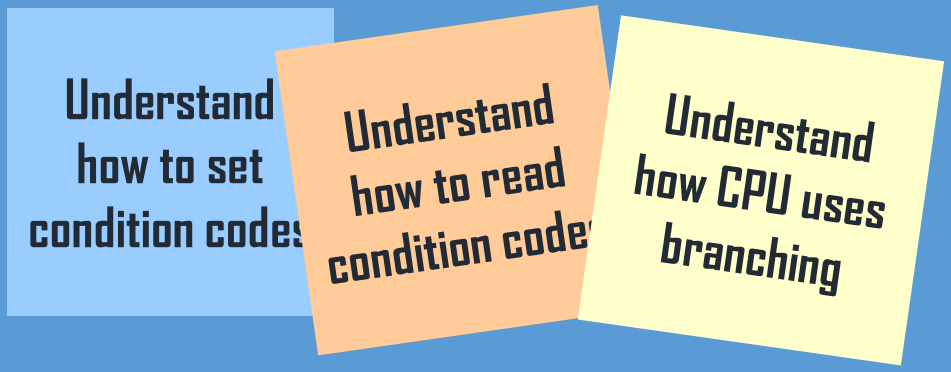


TRUONG DAI HOC KHOA HOC TU NHIEN
KHTN
TP. HO CHI MINH

x86 Procedures

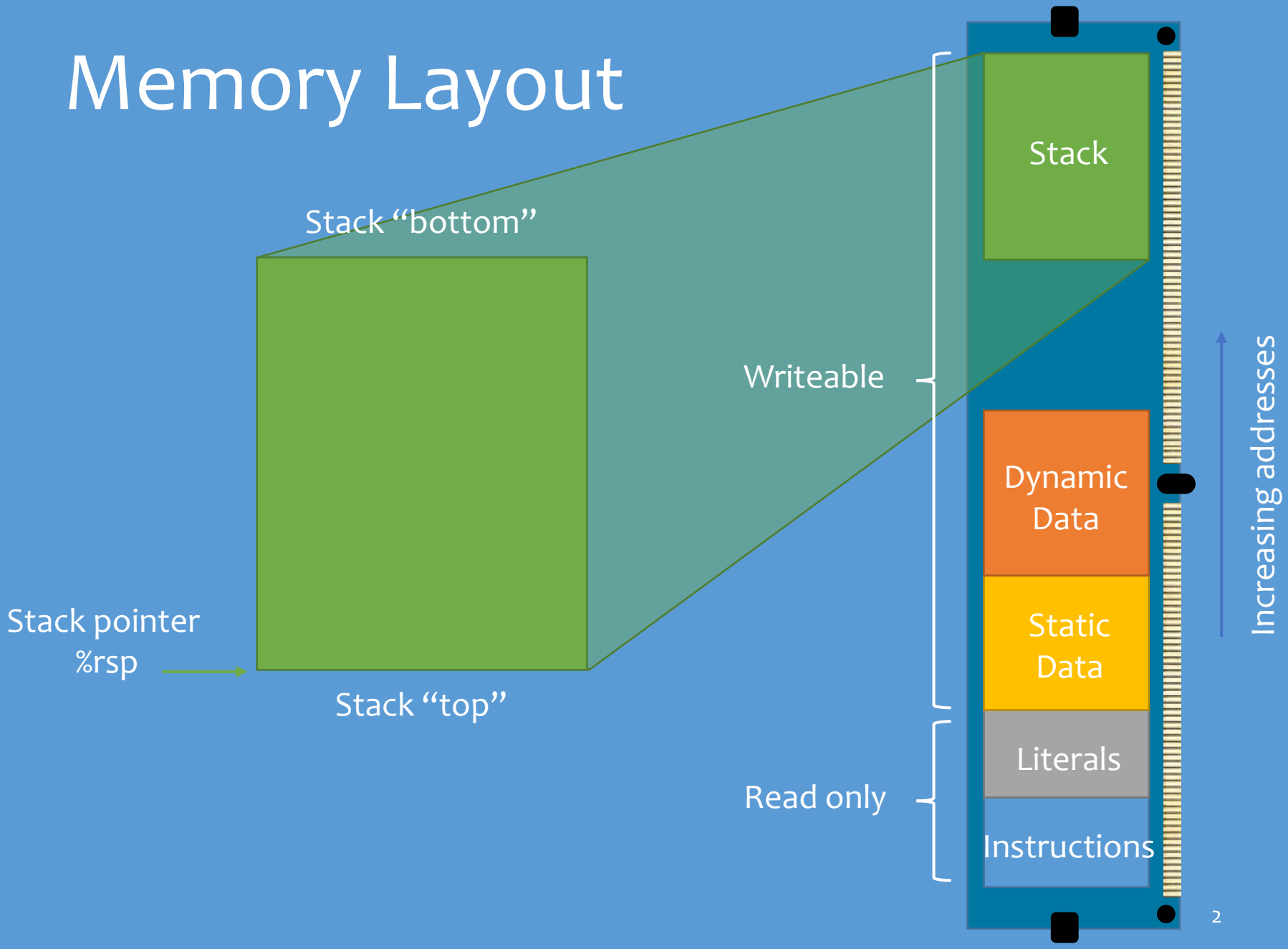
① Stack

② Control/Data Transfer

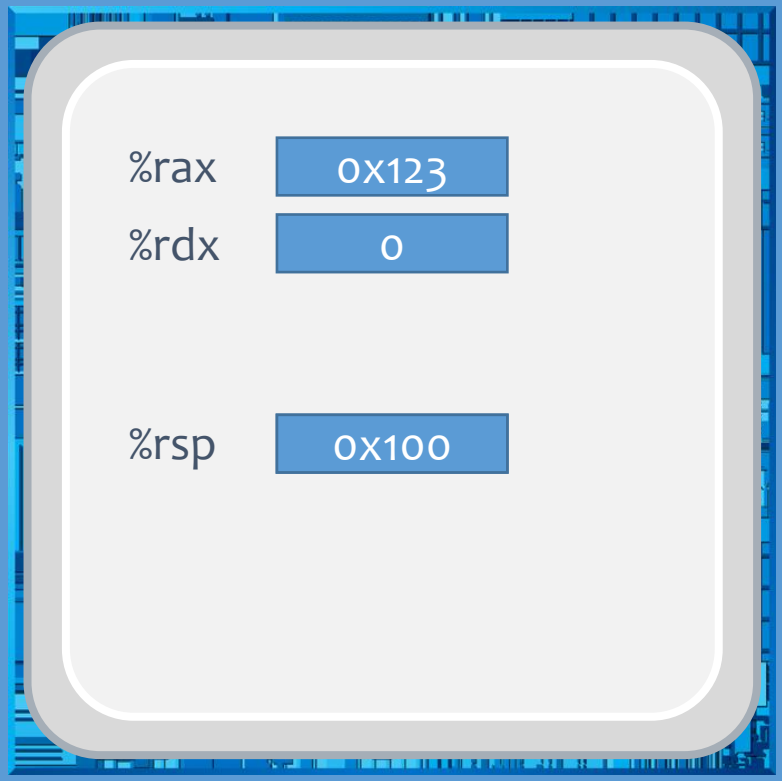
③ Local Storage

④ Recursive Procedures

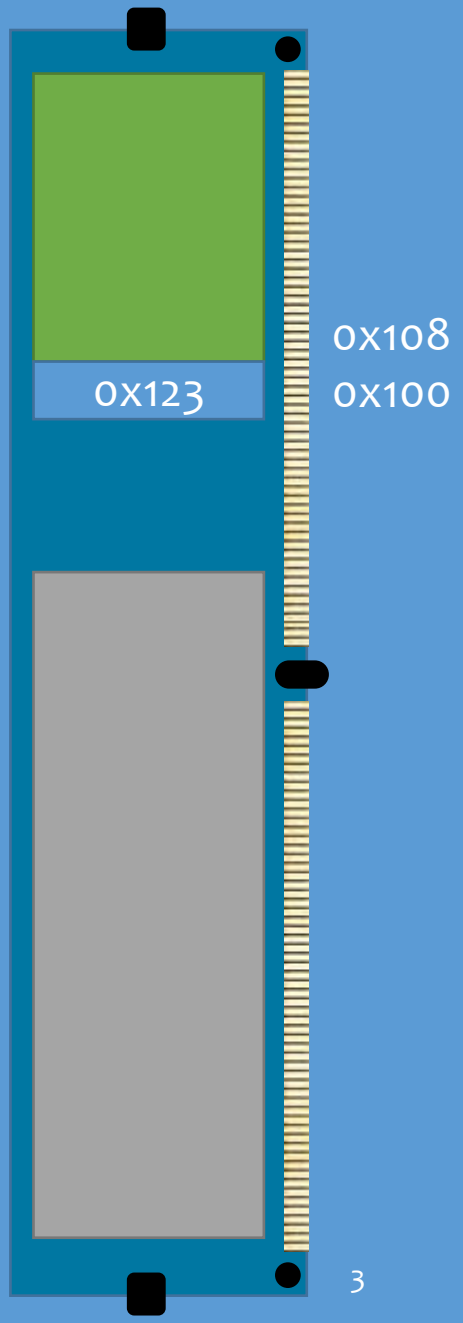
Memory Layout



Stack Push

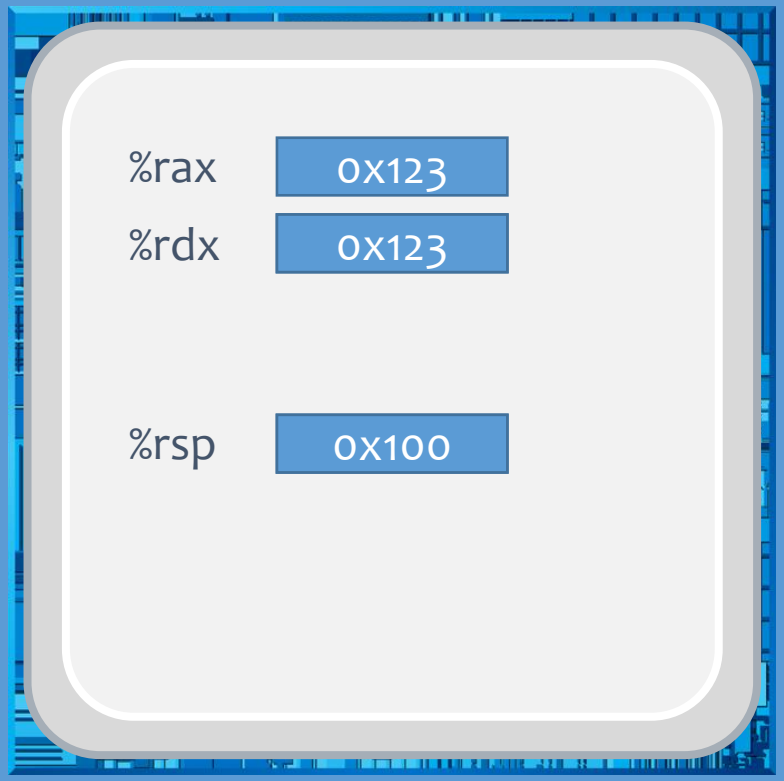


PUSH S

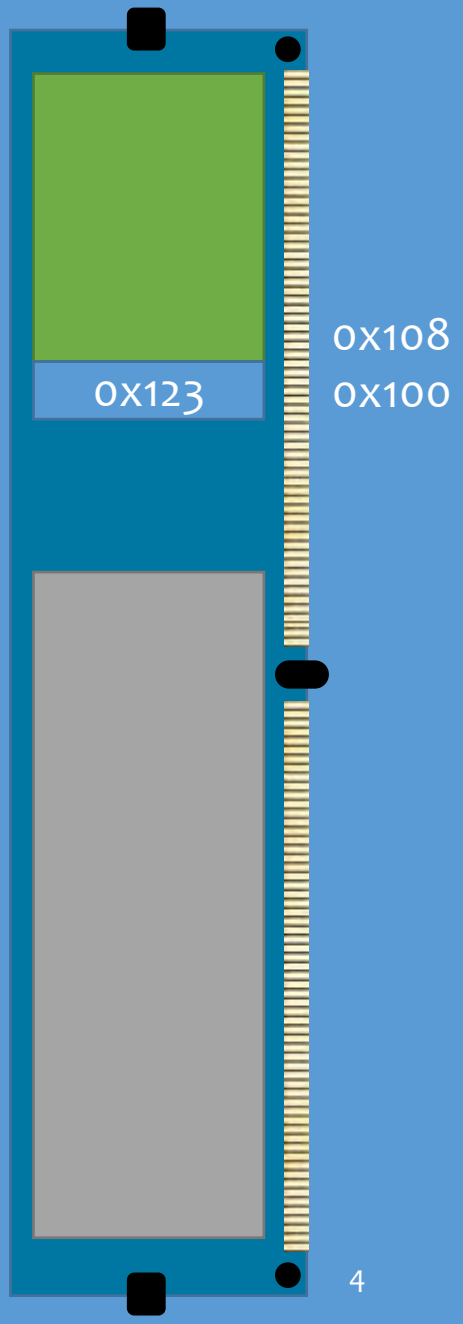



```
pushq   %rax
```

Stack Pop

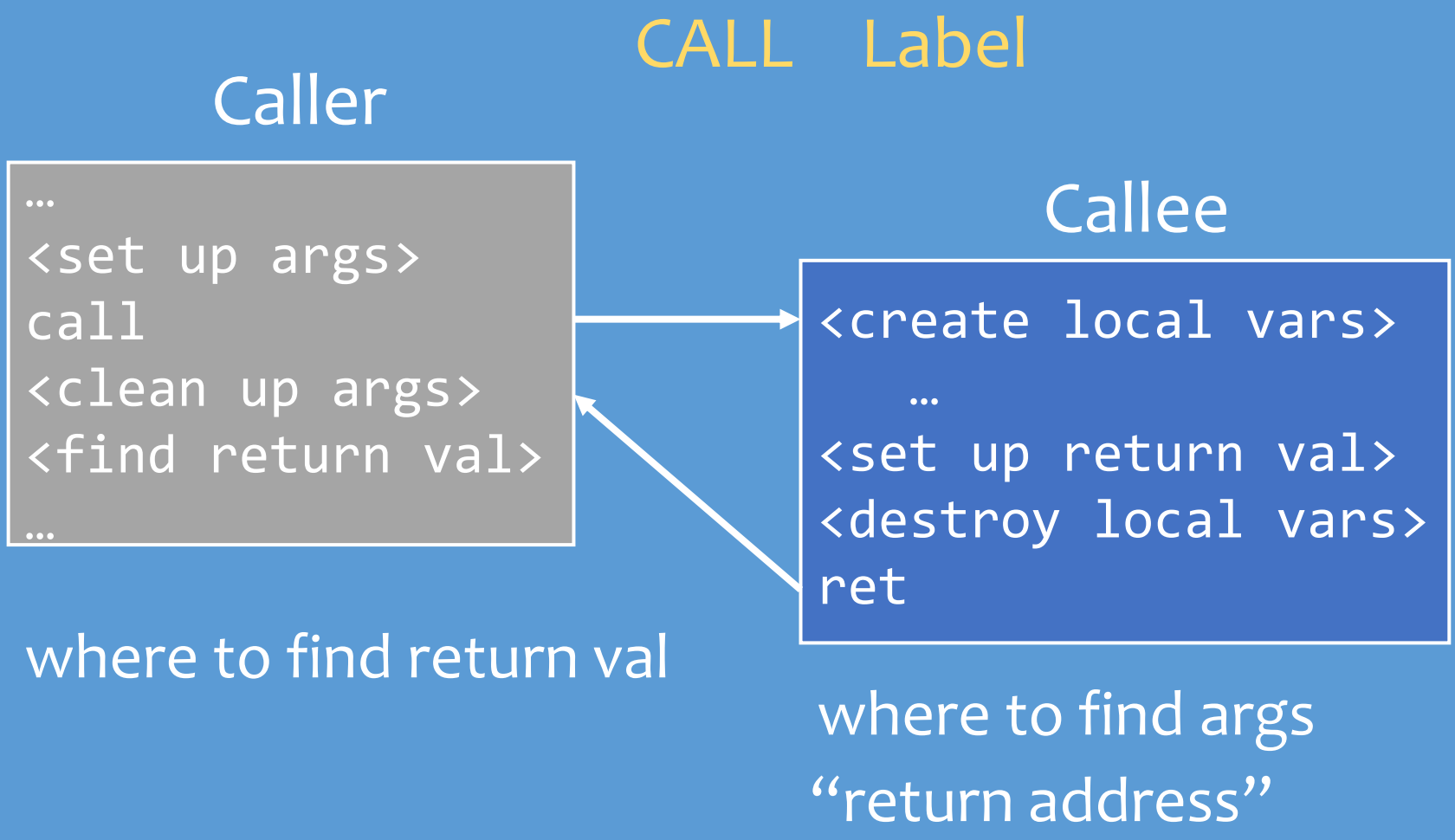


POP D



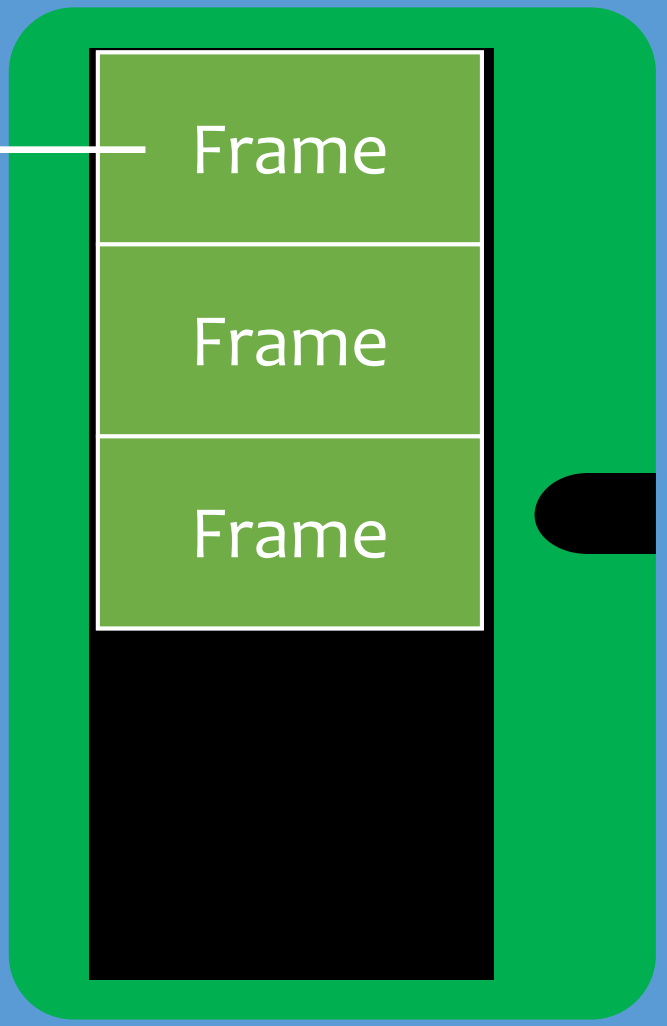

```
popq    %rdx
```

Procedure Call



Stack Frame

Local variables
Function arguments
Return information
Temporary space



Base Pointer: %rbp →

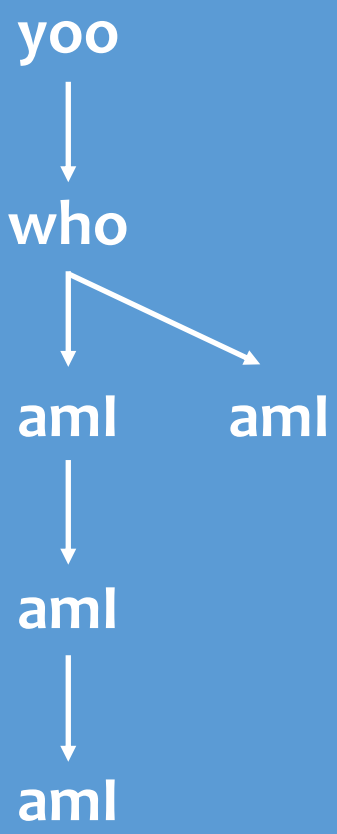
Stack Pointer: %rsp →

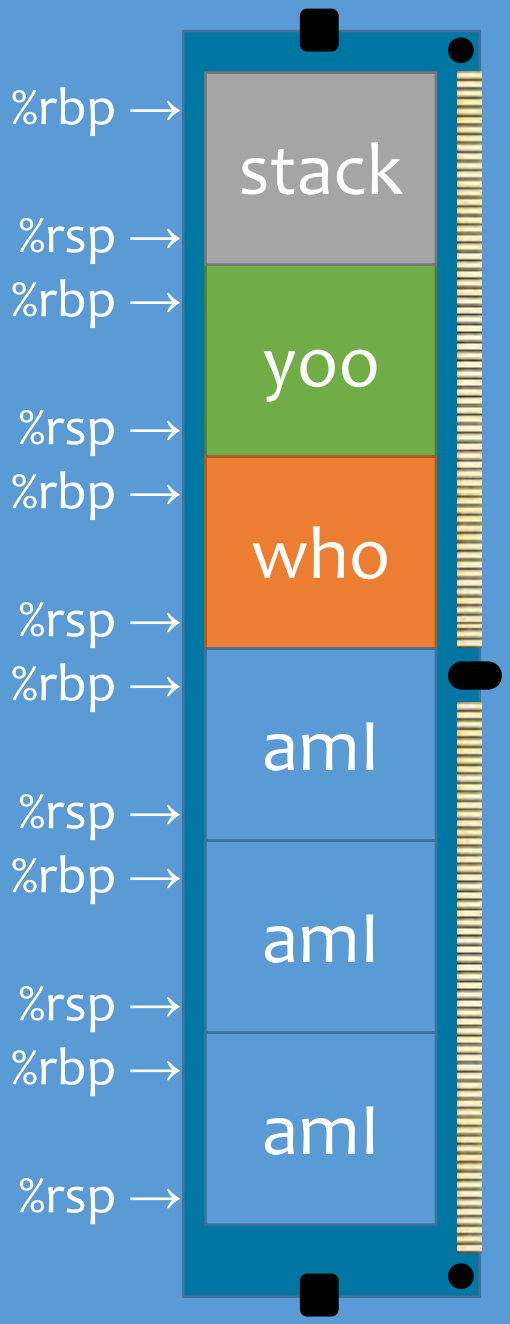
Call Chain Example

```
yoo(…)
{
   …
   who();
   …
}
```

```
who(…)
{
   …
   amI();
   …
   amI();
   …
}
```

```
amI(…)
{
   …
   amI();
   …
}
```



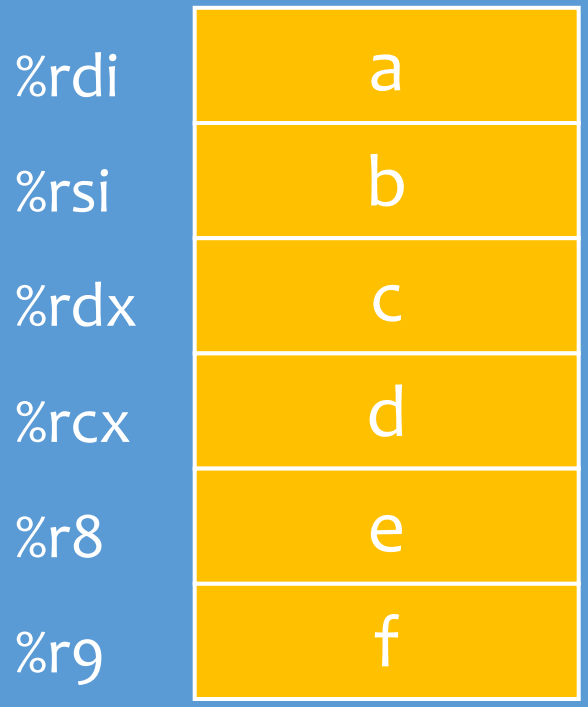


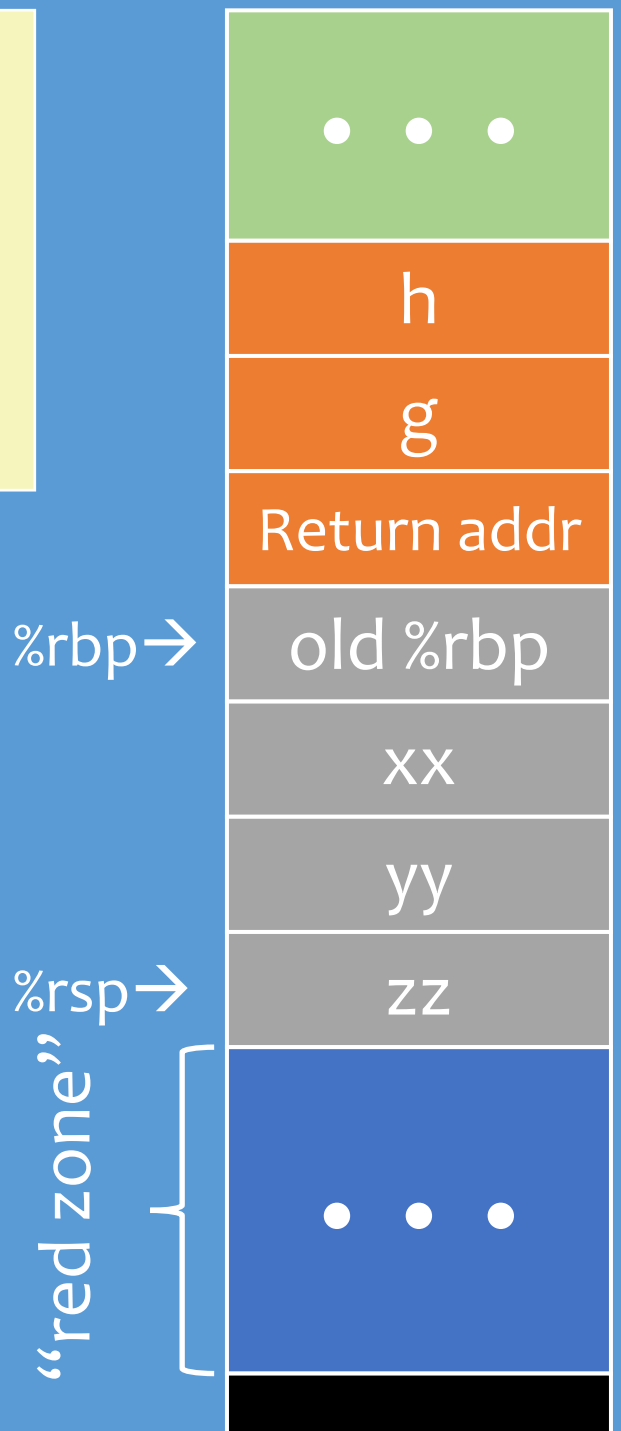
Register Saving Convention

%rax	Return value
%rbx	Callee saved
%rcx	Argument #4
%rdx	Argument #3
%rsi	Argument #2
%rdi	Argument #1
%rsp	Stack pointer
%rbp	Callee saved

%r8	Argument #5
%r9	Argument #6
%r10	Caller saved
%r11	Caller saved
%r12	Callee saved
%r13	Callee saved
%r14	Callee saved
%r15	Callee saved

```
long myfunc(long a, long b, long c, long d,
            long e, long f, long g, long h)
{
    long xx = a * b * c * d * e * f * g * h;
    long yy = a + b + c + d + e + f + g + h;
    long zz = utilfunc(xx, yy, xx % yy);
    return zz + 20;
}
```



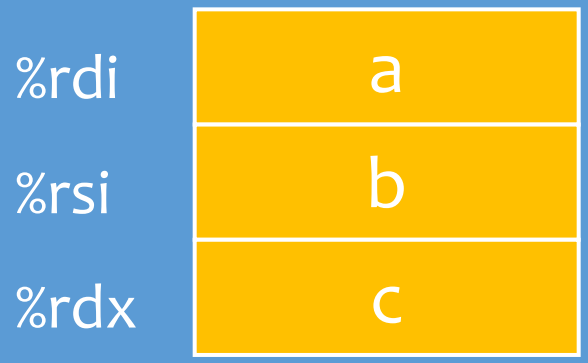


Functions can access memory up to 128 bytes

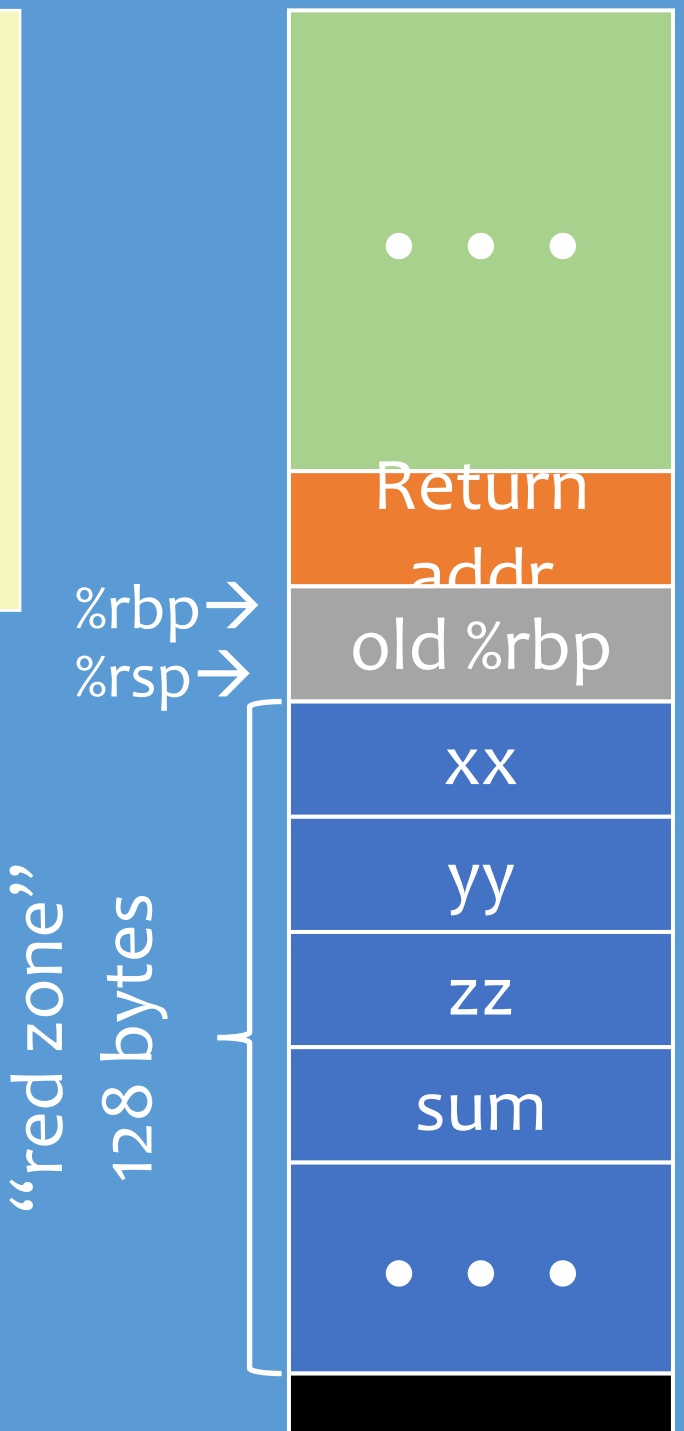
x86-64

```
long utilfunc(long a, long b, long c)
{
    long xx = a + 2;
    long yy = b + 3;
    long zz = c + 4;
    long sum = xx + yy + zz;

    return xx * yy * zz + sum;
}
```



References to stack frame → %rsp
%rbp now available for general-purpose use



x86-64

Summary

- Memory Layout
- Stack
  - PUSH
  - POP
- Stack frame
- Procedure Call
  - CALL
  - RET
- Register Saving Convention
- Red zone

Charles Petzold

American programmer, Microsoft MVP

> Programming in machine code is like eating with a toothpick.