# Recitation3

# Outline

Bomblab

x86 Review

Stack

Buffer Overflow

Datalab

# Schedule

Datalab handed back

Bomblab done!

Buflab out last Thursday and due in two weeks

# Bomblab

# Results

241/274 who submitted got full credit

Very nice!

Now onto the next lab!

# x86 Review

# x86 Review Question

What is the difference between `MOV 4(%eax), %eax` and `LEA 4(%eax), %eax`?

# x86 Review Question

What occurs when executing

`CMP %eax, %edx`?

# x86 Review Question

What occurs when executing `TEST %eax, %eax`?

Questions?

# Stack

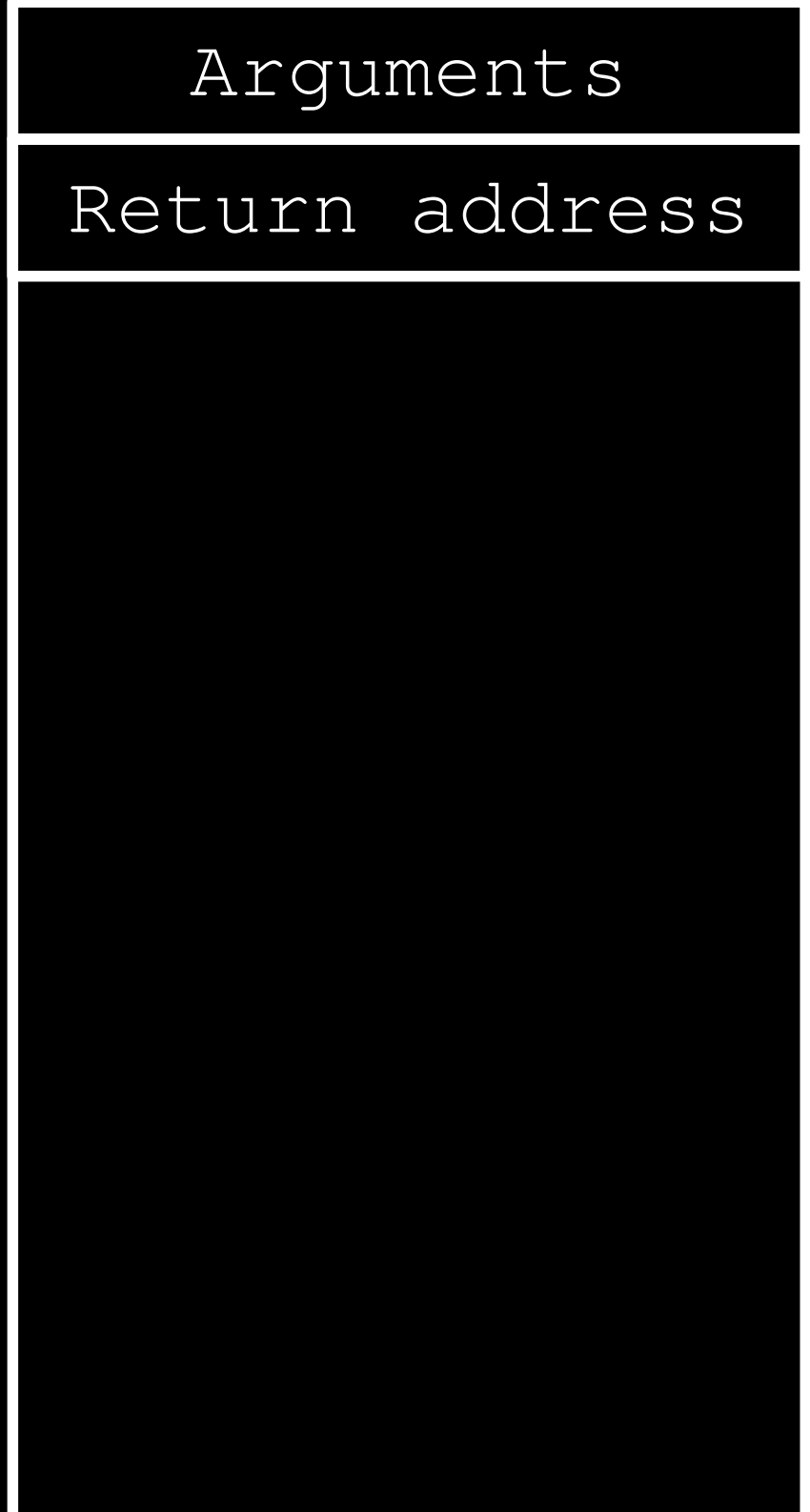
# Review Question

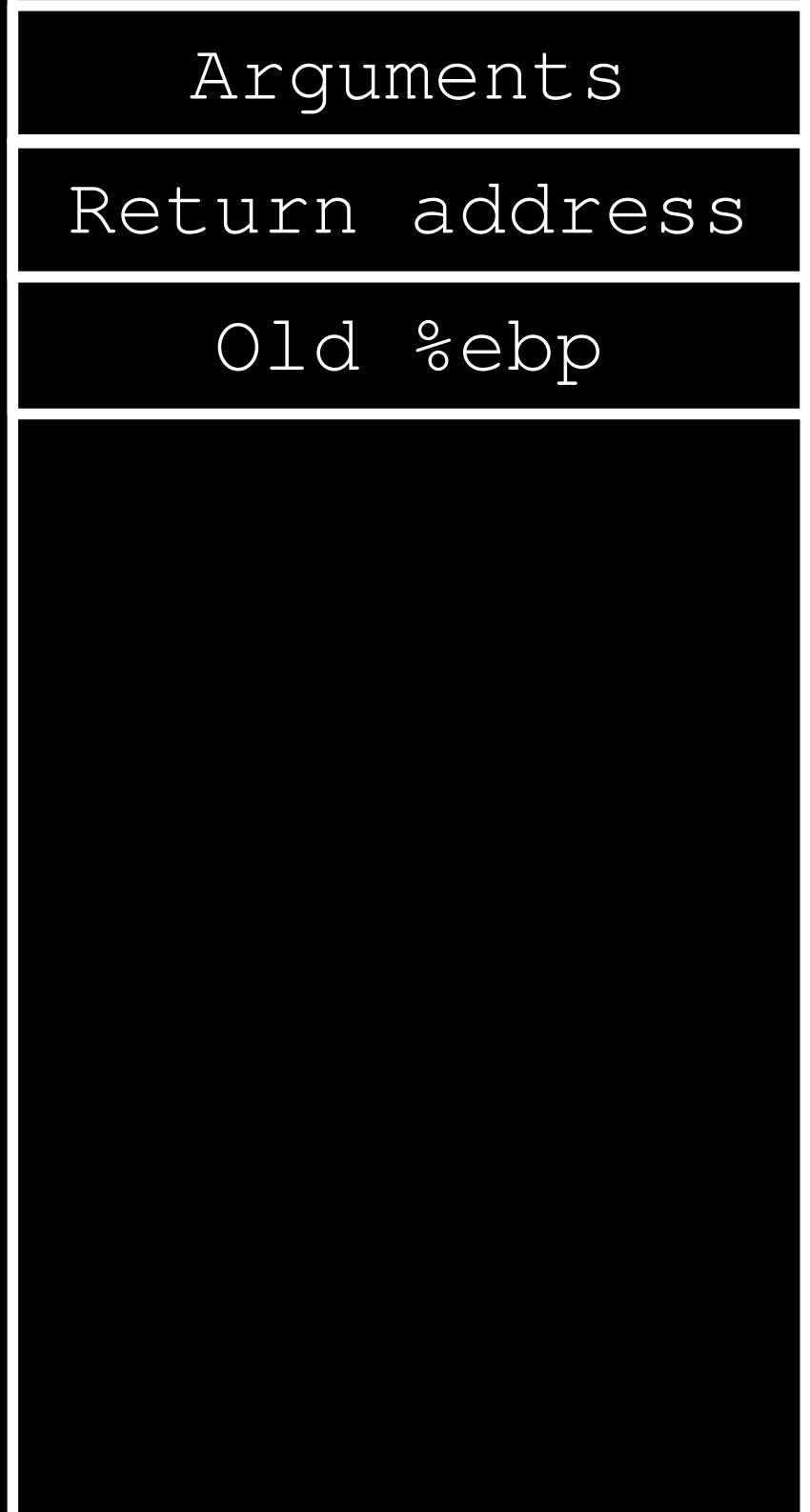
Where on the stack is the return value of a function stored?

# Review Question

What occurs when the x86 instruction `CALL func` is executed?

# Arguments


Arguments
Return address

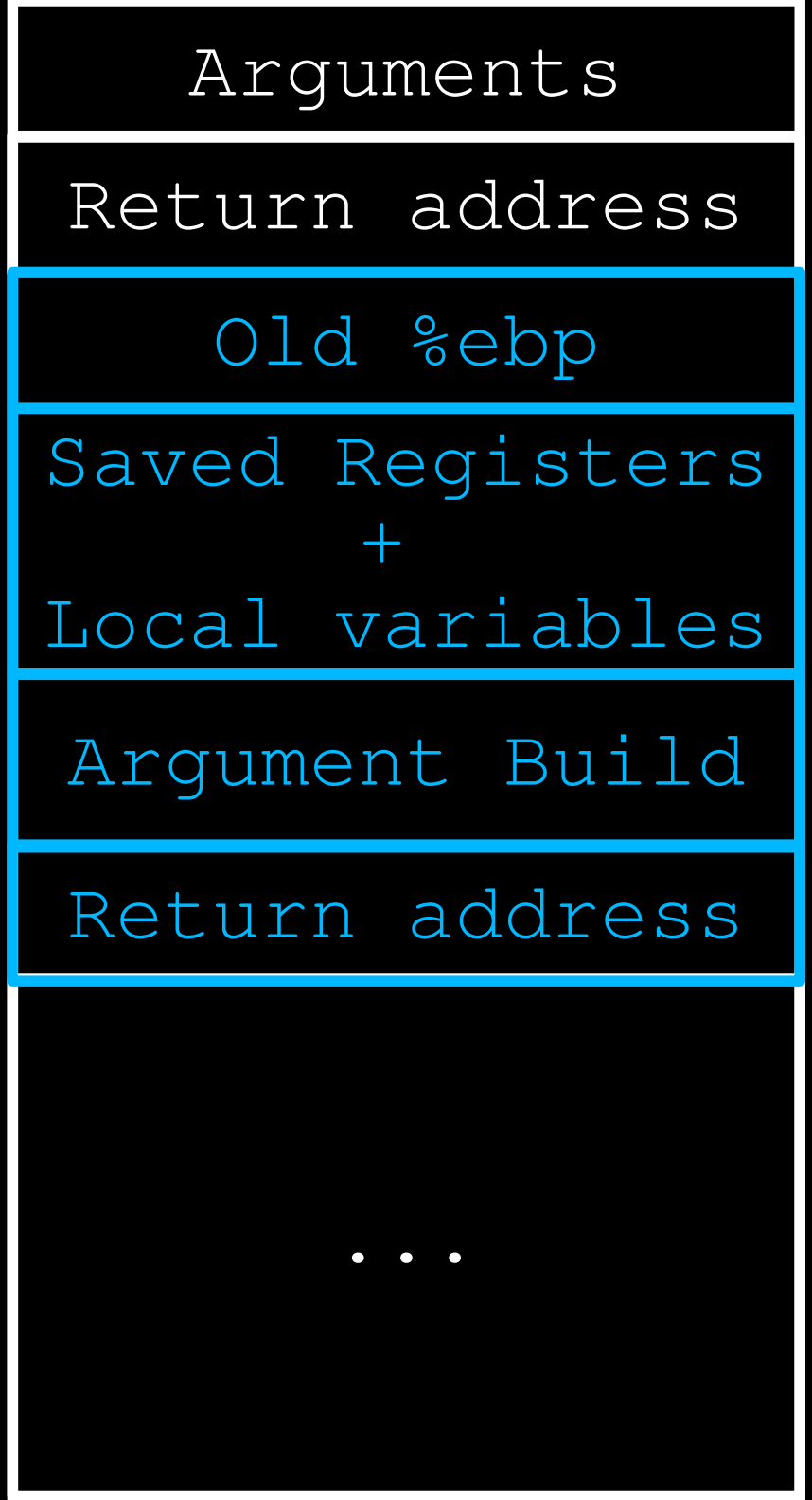

Arguments
Return address
Old %ebp

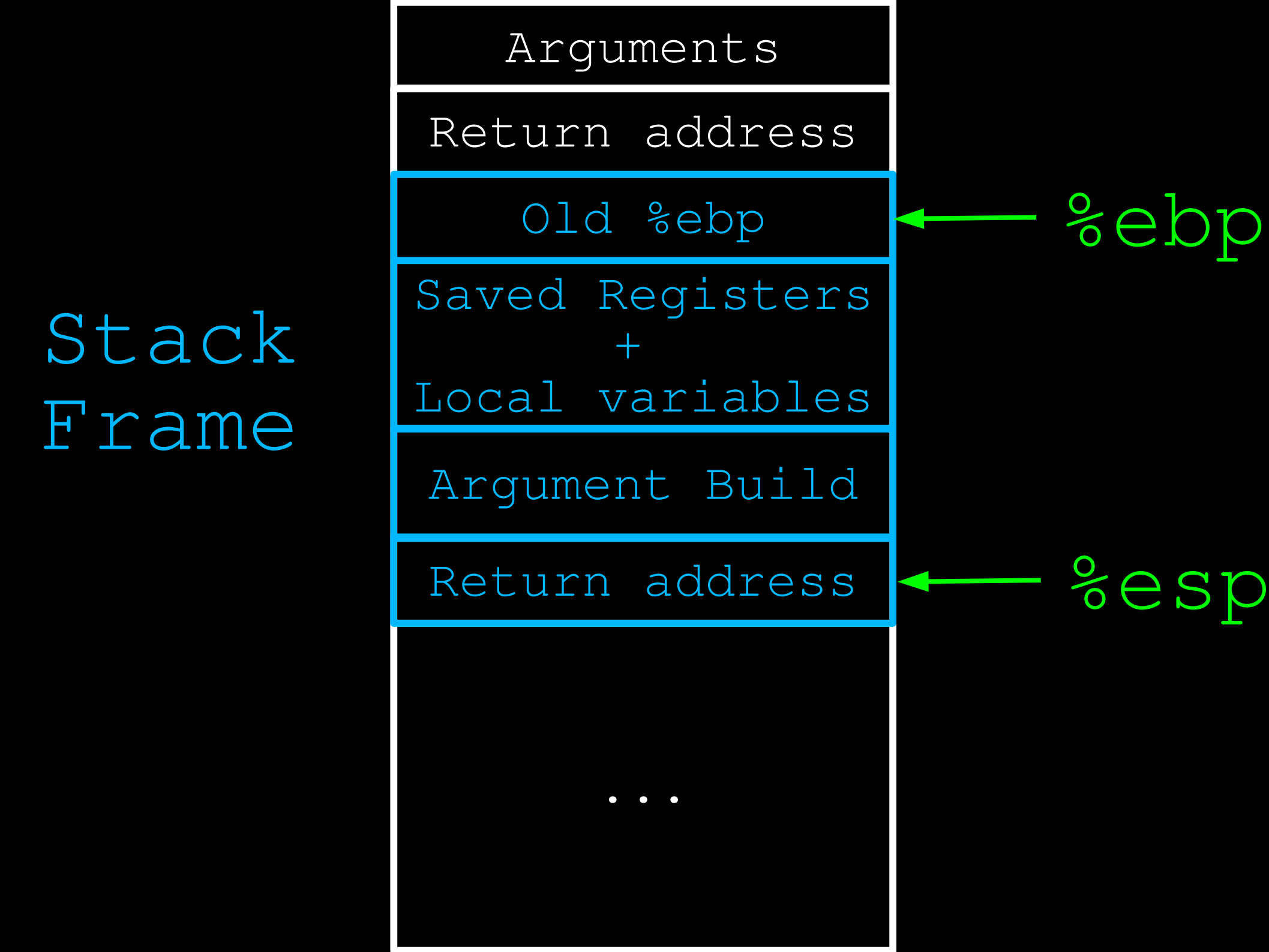

Arguments
Return address
Old %ebp
Saved Registers
+
Local variables


Arguments
Return address
Old %ebp
Saved Registers
+
Local variables
Argument Build


Arguments
Return address
Old %ebp
Saved Registers
+
Local variables
Argument Build
Return address
...

Stack Frame


Arguments
Return address
Old %ebp
Saved Registers
+
Local variables
Argument Build
Return address
...

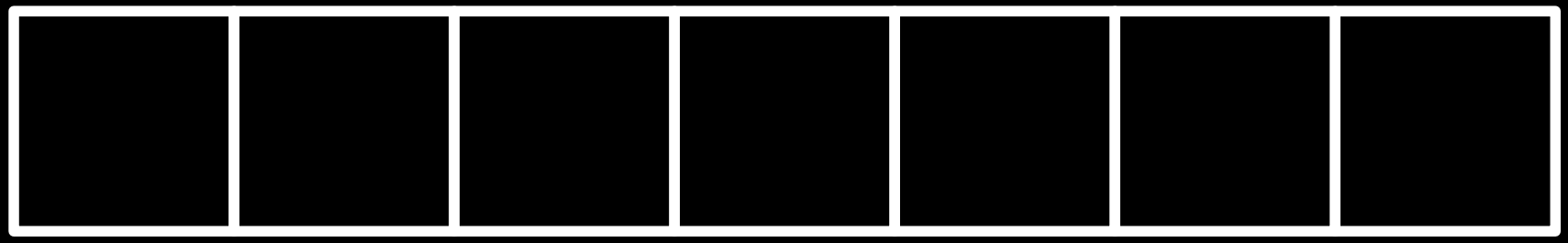

Arguments
Return address
Old %ebp
Saved Registers
+
Local variables
Argument Build
Return address
...
%ebp
%esp
Stack
Frame

# Flash Demo

# Buffer Overflows

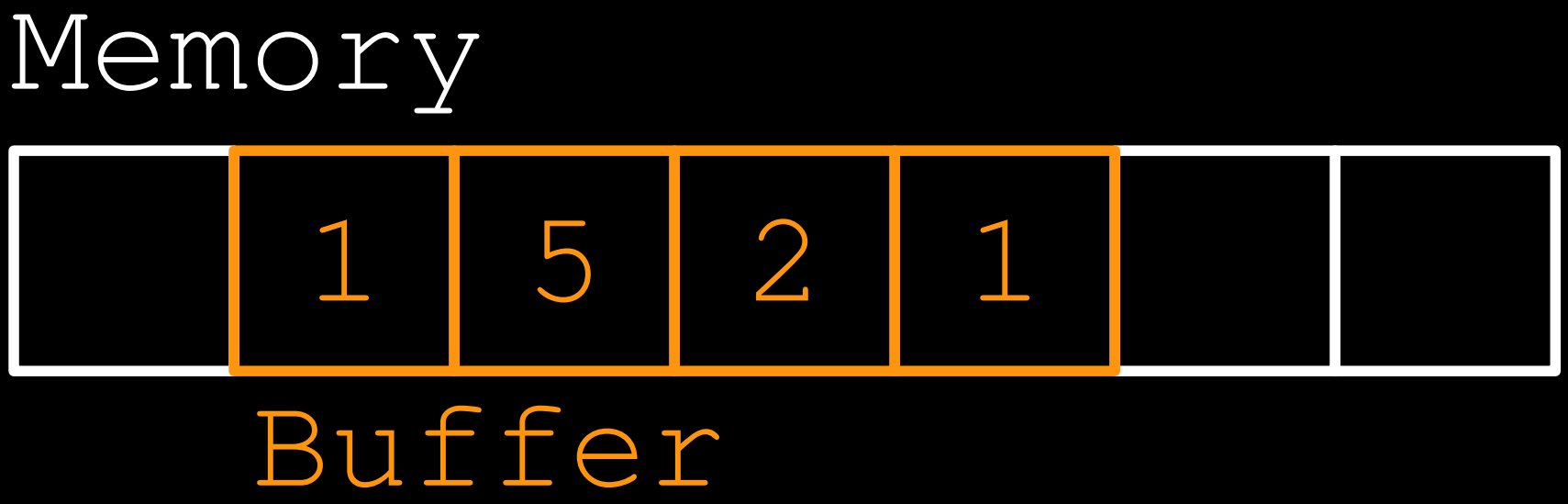
# How it works

Memory

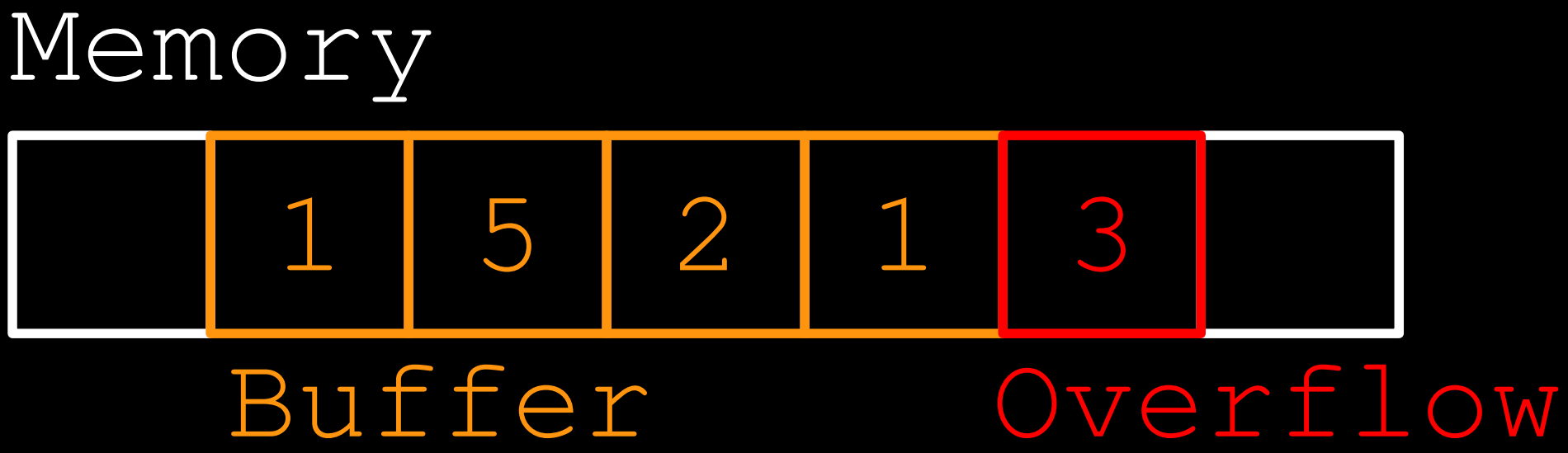


# Allocate some buffer

# Use Buffer



# Buffer Overflow



# Buffer Overflow

Memory


Local variable?

Return address?

# Stack Buffer Overflow

`gets(void *buf)`

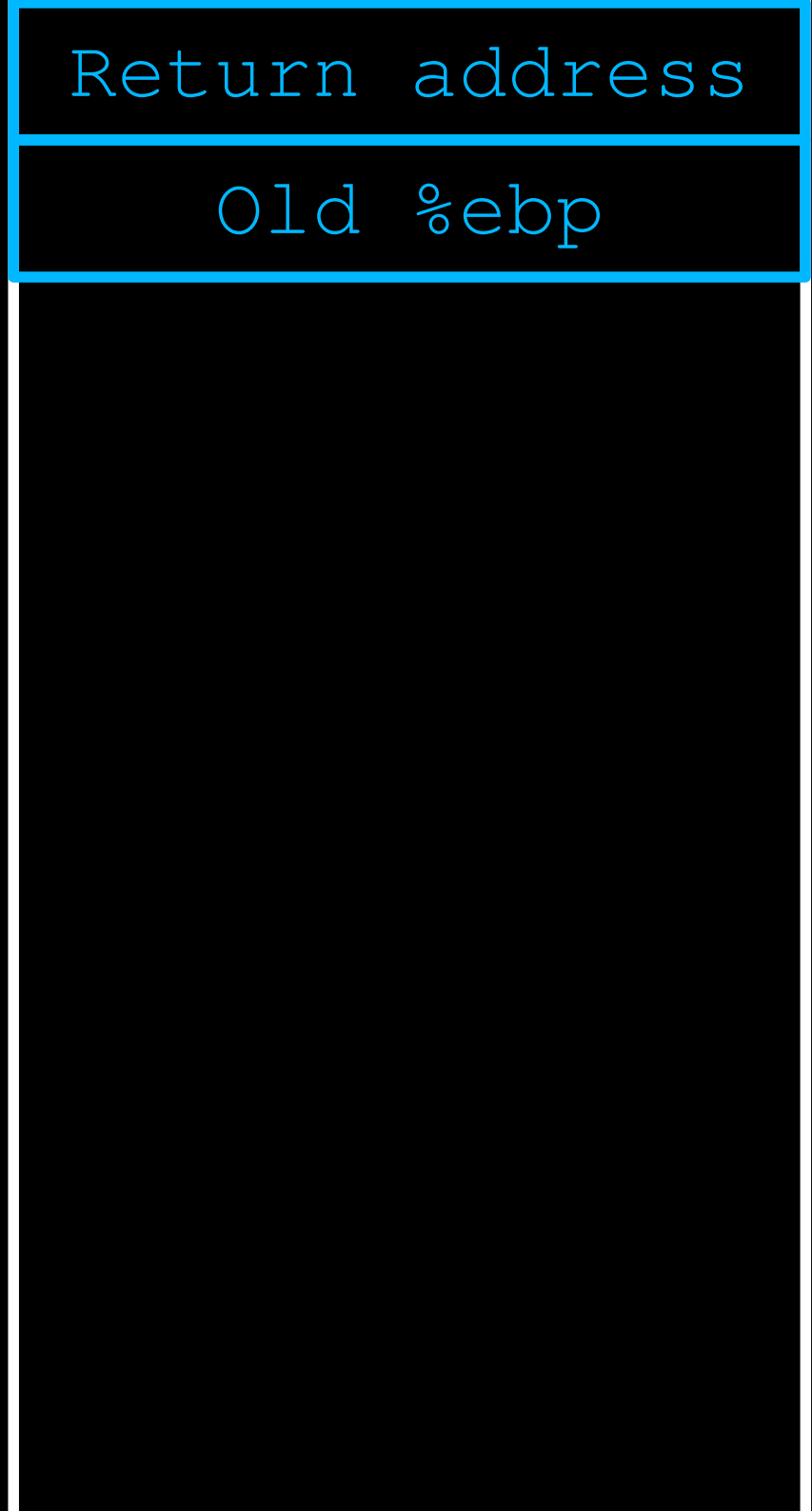
A simple function

Retrieves some user input until it sees '\n'

Then stores the input into “buf”

```
int checkPwd()
{
    char ok = 'F';
    char buf[15];

    puts("Enter password:");
    gets(buf);

    if (!strcmp(buf, "15213"))
    {
        ok = 'T';
    }

    return ok == 'T';
}
```

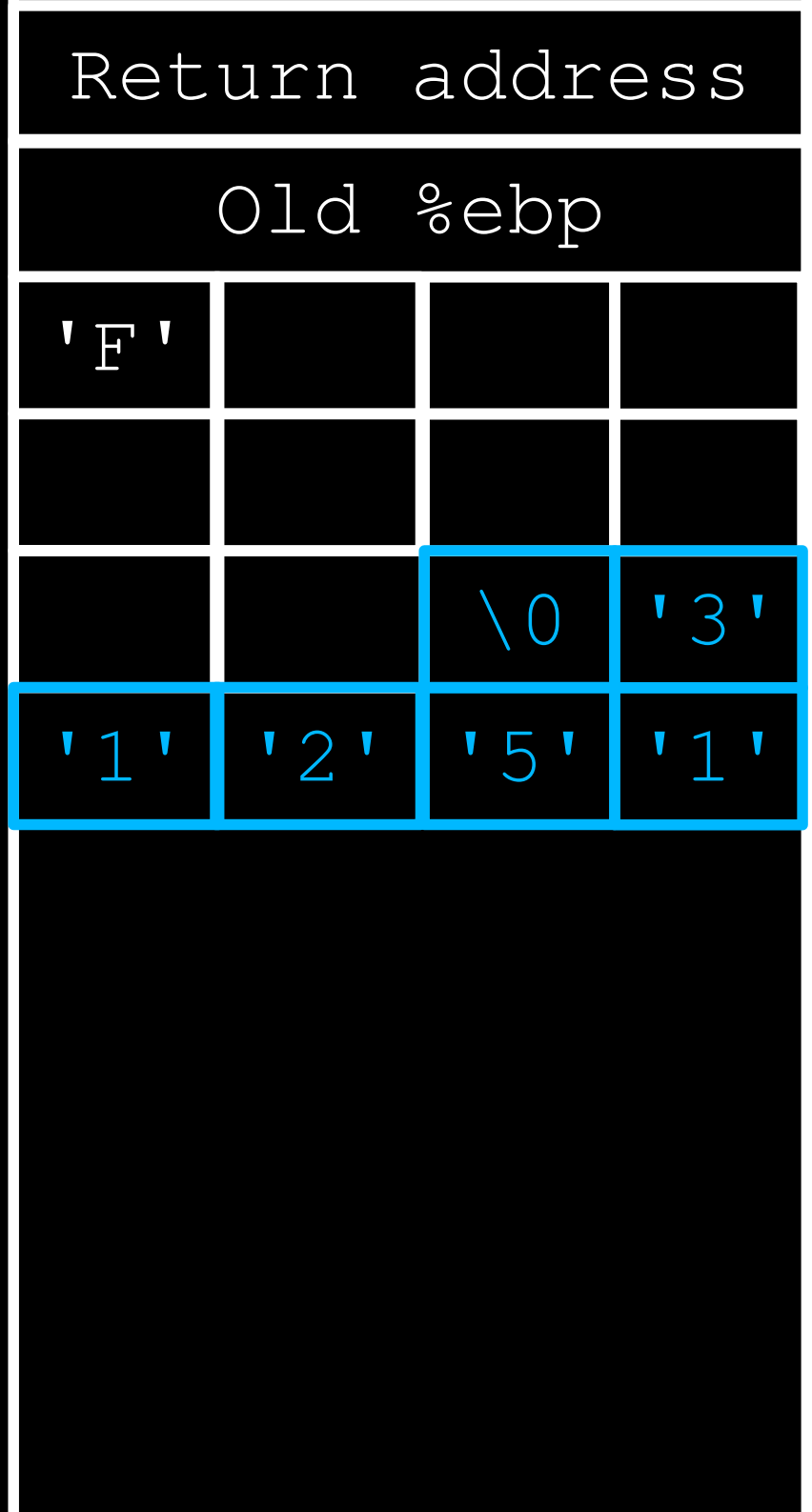
```
int checkPwd()
{
  char ok = 'F';
  char buf[15];

  puts("Enter password:");
  gets(buf);

  if (!strcmp(buf, "15213"))
  {
    ok = 'T';
  }

  return ok == 'T';
}
```



```
int checkPwd(
{
  char ok = 'F';
  char buf[15];

  puts("Enter password:");
  gets(buf);

  if (!strcmp(buf, "15213"))
  {
    ok = 'T';
  }

  return ok == 'T';
}
```

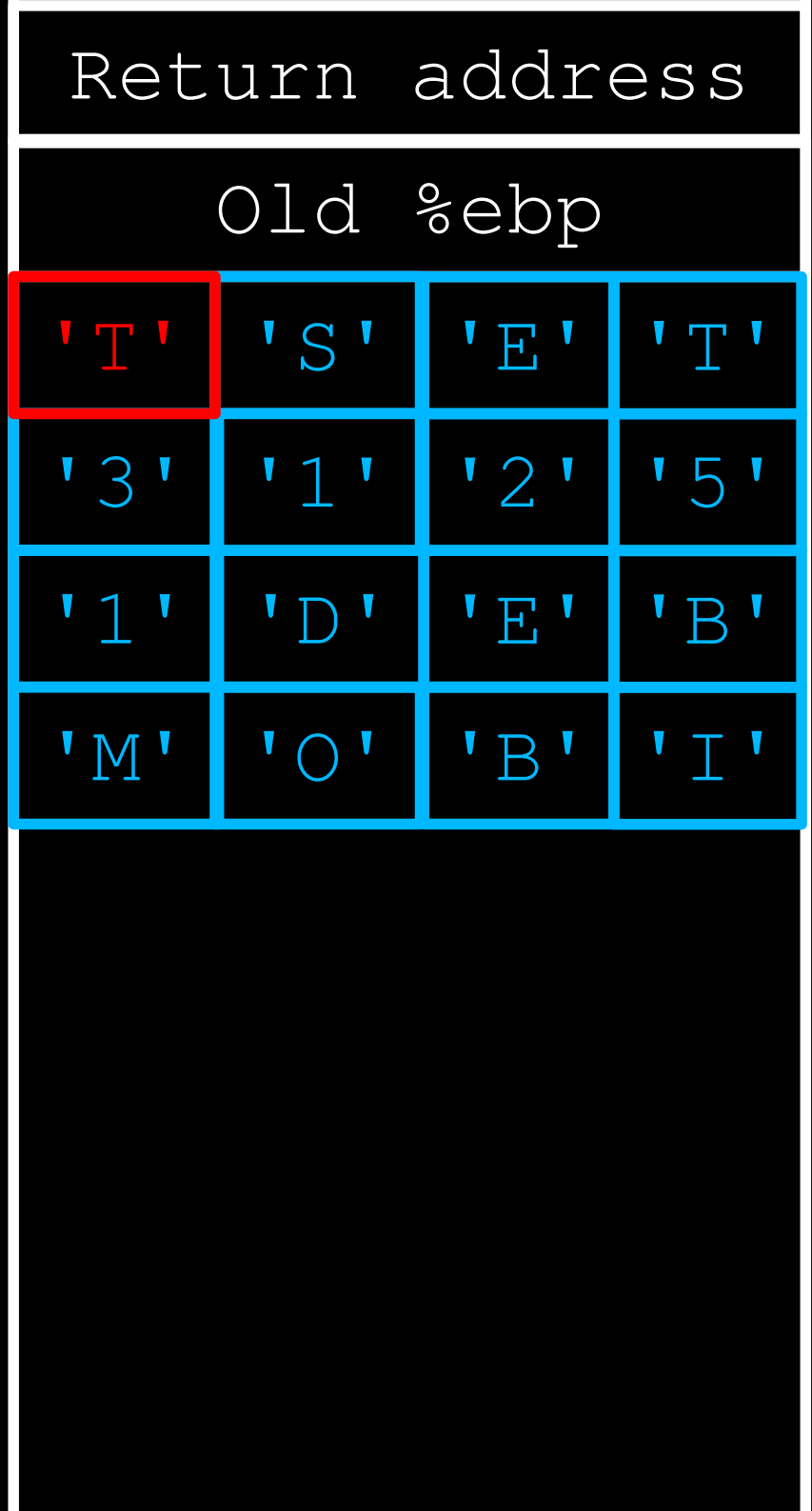
```
int checkPwd()
{
  char ok = 'F';
  char buf[15];

  puts("Enter password:");
  gets(buf);

  if (!strcmp(buf, "15213"))
  {
    ok = 'T';
  }

  return ok == 'T';
}
```

```
Enter password:15213
```



```
int checkPwd()
{
  char ok = 'F';
  char buf[15];

  puts("Enter password:");
  gets(buf);

  if (!strcmp(buf, "15213"))
  {
    ok = 'T';
  }

  return ok == 'T';
}
```

Enter password:
IBOMBED15213TEST



# Demo

# Buflab

# TODO

Apply a series of five stack overflow attacks

No more bombs!

# Disclaimer

The purpose of this lab is to help you learn about the runtime operations of programs and understand the nature of this form of security so that you can avoid it in your code...

...There are criminal statutes against using any form of attack to gain unauthorized access to any system resources.

(Don't be this guy)

# Real World Buffer Overflows

# Morris worm (1988)

UNIX vulnerabilities

Infected ~6000 out of 60000 computers connected to Internet

Robert Morris sentenced 3 yrs probation, 400 hours of community service, $10000 fine

# SQL Slammer Worm (2003)

Microsoft's SQL Server

90% of vulnerable machines infected within 10 minutes

This week in Google News:
"buffer overflow"

"Two of the vulnerabilities are buffer overflow issues... impacting the iPhone and the iPod Touch"

"One of the critical vulnerabilities is a buffer overflow issue within Real Player..."

"A buffer overflow vulnerability in the Java Runtime Environment..."