

# Reverse Engineering

2024/8/6 國立清華大學資訊安全研究所暑訓

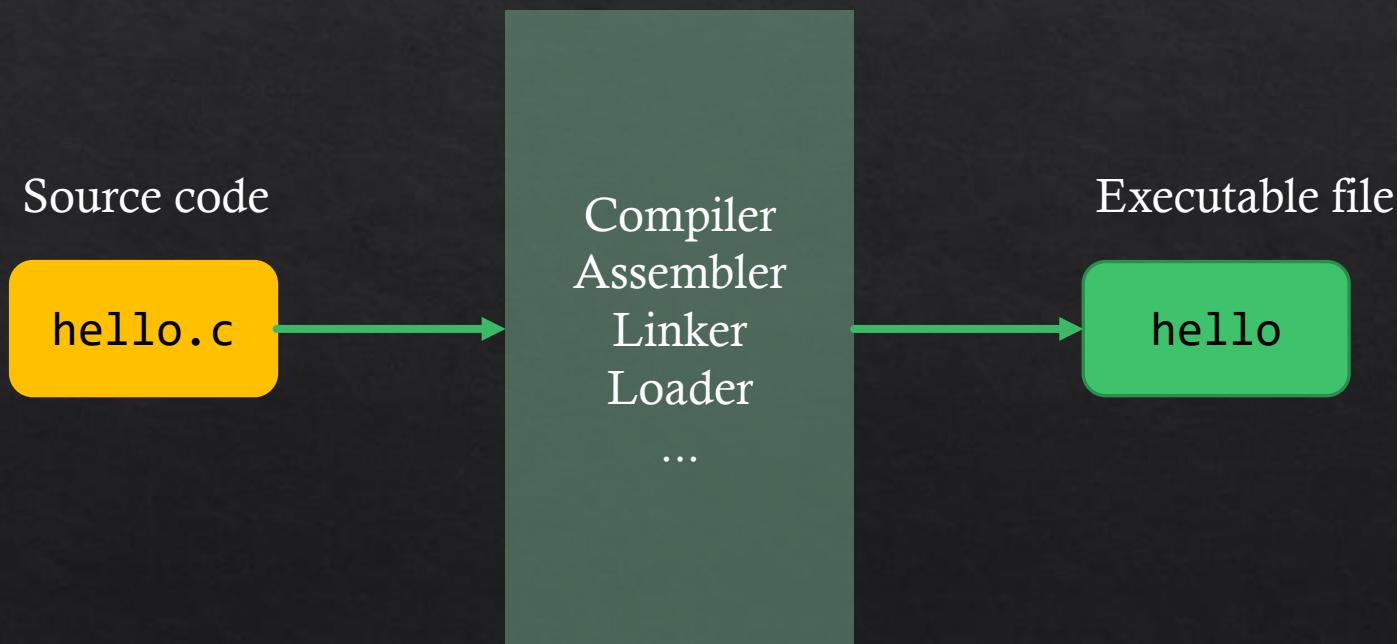
潘甫翰 Sharkcode

# Outline

- ❖ Reverse Engineering Overview
- ❖ Analysis
- ❖ x64 Assembly
- ❖ Memory Layout
- ❖ Demo Learning Website
- ❖ Lab
- ❖ Tools
  - ❖ GHIDRA
  - ❖ GDB
  - ❖ pwntools
- ❖ Lab

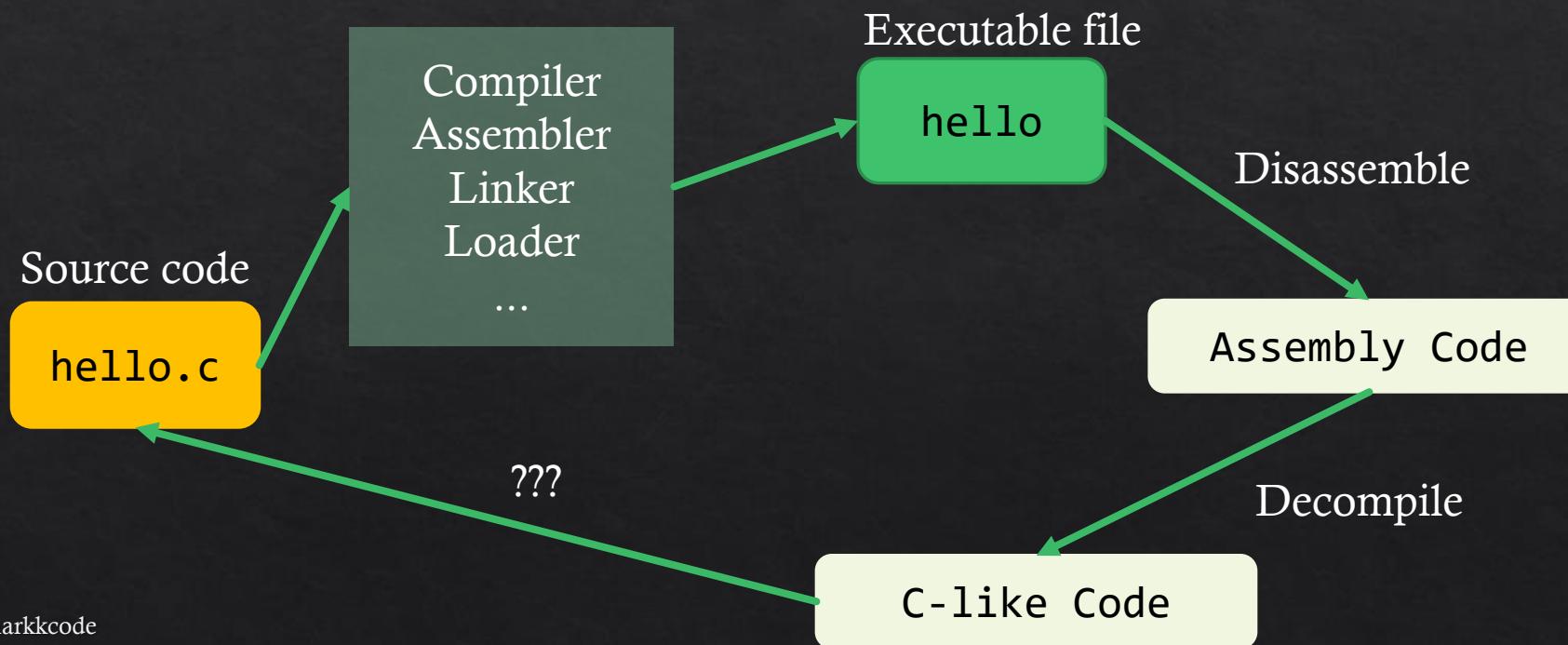
# Reverse Engineering Overview

- ❖ When we finish writing a program `hello.c`, we might want to use Compiler, Assembler, Linker, Loader to generate an executable file `hello` to execute.



# Reverse Engineering Overview

- ❖ So what exactly is Reverse Engineering doing?
  - ❖ Analyzing the files and information you have to reconstruct the behavior of a program or project, which can facilitate further actions such as writing a cracking program, etc.



# Bigger Overview?

- ❖ Figure out **what is going on** ...

通靈

# Analysis

- ❖ Statically
  - ❖ Without running...
- ❖ Dynamically
  - ❖ When running...

# x64 Assembly

- ❖ <https://www.intel.com/content/dam/develop/external/us/en/documents/introduction-to-x64-assembly-181178.pdf>

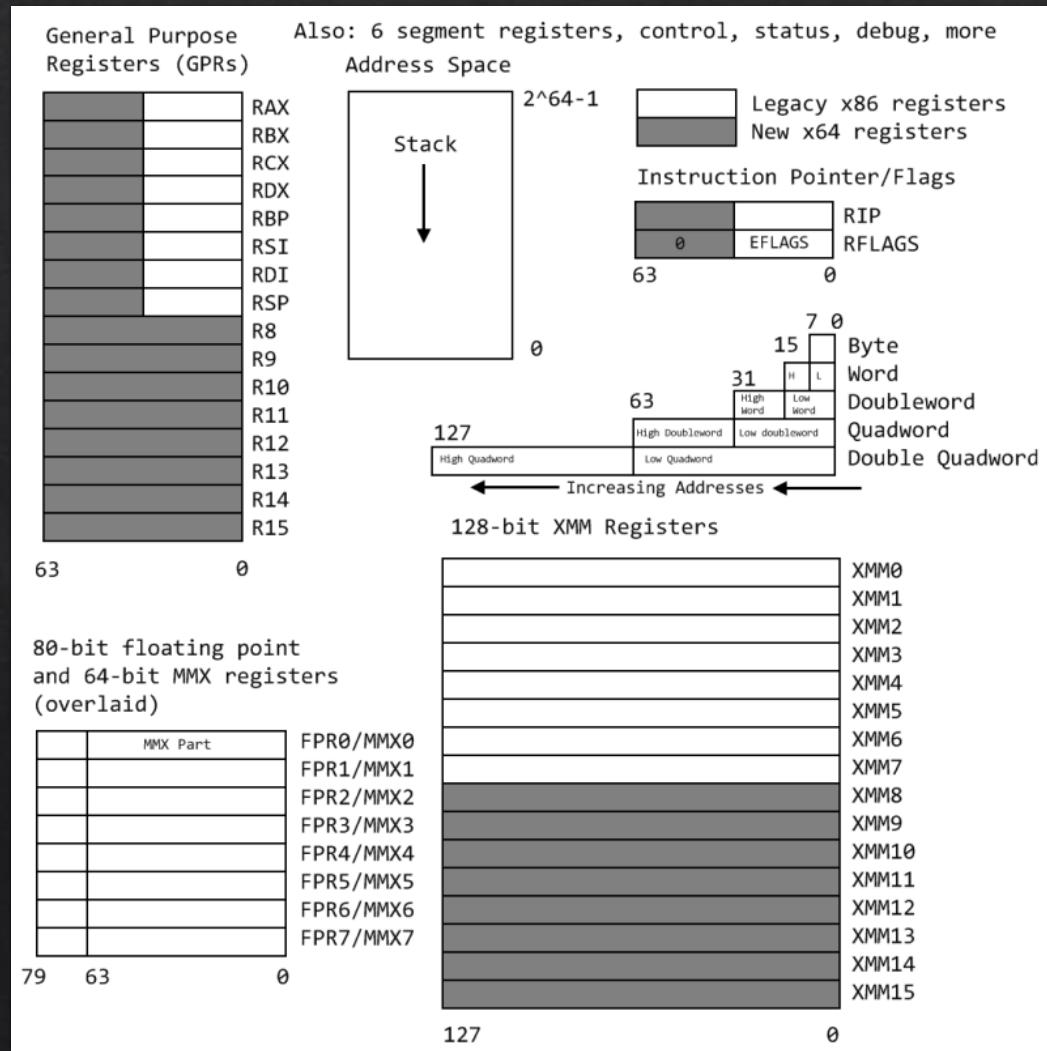


# x64 Assembly -- Cheatsheet

- ❖ [https://cs.brown.edu/courses/cs033/docs/guides/x64\\_cheatsheet.pdf](https://cs.brown.edu/courses/cs033/docs/guides/x64_cheatsheet.pdf)



# x64 Assembly -- General Architecture



# x64 Assembly -- Registers

## 4.3 Register Usage

There are sixteen 64-bit registers in x86-64: `%rax`, `%rbx`, `%rcx`, `%rdx`, `%rdi`, `%rsi`, `%rbp`, `%rsp`, and `%r8-r15`. Of these, `%rax`, `%rcx`, `%rdx`, `%rdi`, `%rsi`, `%rsp`, and `%r8-r11` are considered caller-save registers, meaning that they are not necessarily saved across function calls. By convention, `%rax` is used to store a function's return value, if it exists and is no more than 64 bits long. (Larger return types like structs are returned using the stack.) Registers `%rbx`, `%rbp`, and `%r12-r15` are callee-save registers, meaning that they are saved across function calls. Register `%rsp` is used as the *stack pointer*, a pointer to the topmost element in the stack.

Additionally, `%rdi`, `%rsi`, `%rdx`, `%rcx`, `%r8`, and `%r9` are used to pass the first six integer or pointer parameters to called functions. Additional parameters (or large parameters such as structs passed by value) are passed on the stack.

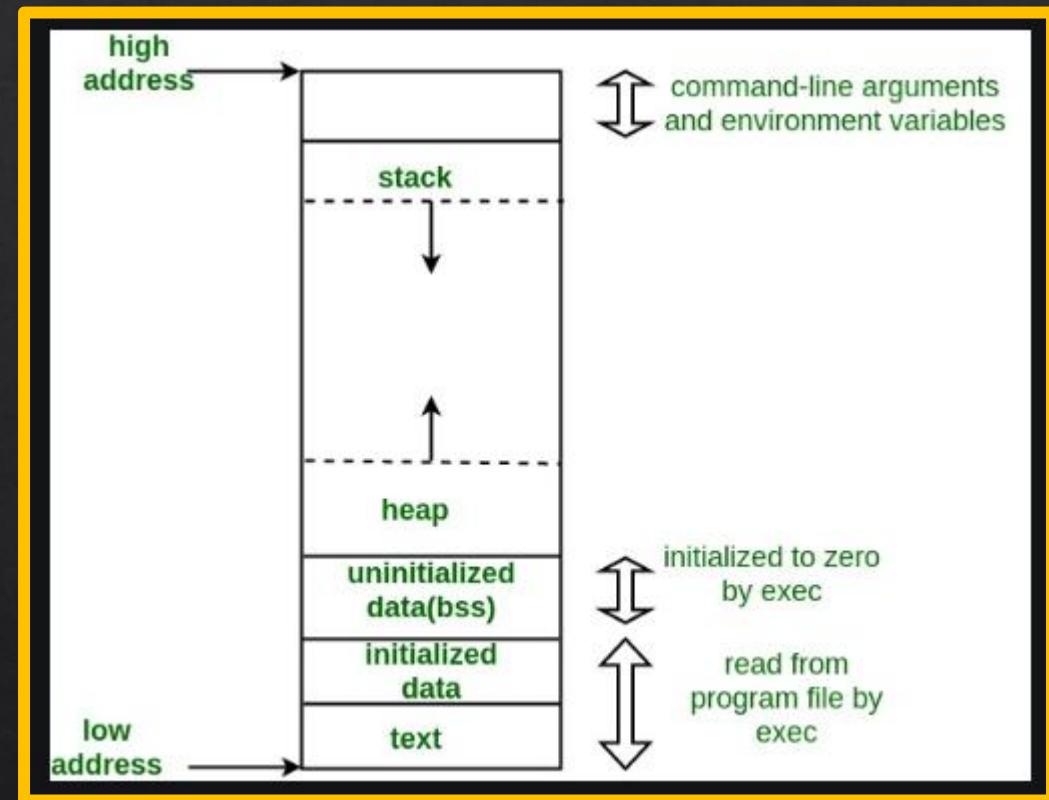
In 32-bit x86, the *base pointer* (formerly `%ebp`, now `%rbp`) was used to keep track of the base of the current stack frame, and a called function would save the base pointer of its caller prior to updating the base pointer to its own stack frame. With the advent of the 64-bit architecture, this has been mostly eliminated, save for a few special cases when the compiler cannot determine ahead of time how much stack space needs to be allocated for a particular function (see Dynamic stack allocation).

# x64 Assembly -- Instructions

- ❖ mov
- ❖ jmp
- ❖ call
- ❖ ret
- ❖ nop
- ❖ ...

# Memory Layout

- ❖ <https://www.geeksforgeeks.org/memory-layout-of-c-program/>



# Demo Learning Website

<https://godbolt.org/>

# Website Overview

The screenshot shows the Compiler Explorer interface with two tabs: "C++ source #1" and "x86-64 gcc 12.2 (Editor #1)".

**C++ source #1:**

```
1 // Type your code here, or load an example.
2 int square(int num) {
3     return num * num;
4 }
```

**x86-64 gcc 12.2 (Editor #1):**

```
1 square(int):
2     push    rbp
3     mov     rbp,  rsp
4     mov     DWORD PTR [rbp-4], edi
5     mov     eax,  DWORD PTR [rbp-4]
6     imul   eax,  eax
7     pop    rbp
8     ret
```

At the bottom, there is an "Output" section showing:

```
C Output (0/0) x86-64 gcc 12.2 i - 739ms (2811B) ~170 lines filtered
```

Page footer: By 潘甫翰 Sharkcode (4, 2) Compiler License 14

# Website Overview

The screenshot shows the Compiler Explorer interface. At the top left is the logo "COMPILER EXPLORER". The top navigation bar includes "Add...", "More", "Templates", "Backtrace intel SolidSands", "Share", "Policies", and "Other". The main area has two code editors. The left editor contains C++ code for a square function:

```
// Type your code here, or load an example.  
int square(int num) {  
    return num * num;  
}
```

A red box highlights the "C++" language dropdown, with a yellow callout "Choose Language" pointing to it. Another red box highlights the "x86-64 gcc 12.2" compiler dropdown, with a yellow callout "Choose Compiler" pointing to it. A third red box highlights the "Compiler options..." dropdown, with a yellow callout "Choose Compiler Options" pointing to it. The right editor shows the generated assembly code:square(int):  
 .cfi\_startproc  
 mov eax, DWORD PTR [rbp-4]  
 imul eax, eax  
 pop rbp  
 ret

At the bottom, there is an "Output" section showing "Output (0/0) x86-64 gcc 12.2" with a duration of "739ms (2811B) ~170 lines filtered", a "Compiler License" link, and page numbers "(4, 2)" and "15".

# Hello World DEMO

```
// SOURCE CODE
#include <stdio.h>

int main(int argc, char **argv) {
    printf("Hello World!\n");
    return 0;
}
```

# Hello World DEMO

The diagram illustrates the compilation process from C source code to assembly code. It shows two main windows: Compiler Explorer on the left and Backtrace intel on the right.

**Compiler Explorer (Left):**

- A C++ source code editor window titled "C++ source #1" containing the following code:

```
1 #include <stdio.h>
2
3 int main(int argc, char **argv) {
4
5     printf("Hello World!\n");
6
7     return 0;
8 }
```

- An "Output" tab below the code editor.

**Backtrace intel (Right):**

- An assembly code editor window titled "x86-64 gcc 12.2 (Editor #1)" showing the generated assembly code:

```
1 .LC0:
2     .string "Hello World!"
3
4 main:
5     push    rbp
6     mov     rbp, rsp
7     sub    rsp, 16
8     mov    DWORD PTR [rbp-4], edi
9     mov    QWORD PTR [rbp-16], rsi
10    mov    edi, OFFSET FLAT:.LC0
11    call   puts
12    mov    eax, 0
13    leave
14    ret
```

**Flow Diagram (Bottom):**

```
graph TD
    A[Source code  
hello.c] --> B[Compiler Assembler  
Linker Loader  
...]
    B --> C[hello]
    C -- Executable file --> D[Assembly Code]
    C -- Disassemble --> E[c-like Code]
    E -- Decompile --> F[Assembly Code]
    F -- Disassemble --> G[hello]
    G -- Executable file --> H[Assembly Code]
```

The flow starts with "Source code (hello.c)", which is processed by the "Compiler Assembler Linker Loader". This produces an "Executable file (hello)". From the executable file, the flow can lead to either "Assembly Code" or "c-like Code". "c-like Code" can be converted back to "Assembly Code" via "Decompile". "Assembly Code" can also be converted back to "Executable file (hello)" via "Disassemble".

Annotations in the diagram include:

- "which can facilitate further actions such as writing a cracking program, etc."
- "By 潘甫翰 Sharkcode"
- "rkkcode" (in the bottom left corner of the diagram area)
- "17" (in the bottom right corner of the diagram area)

# Hello World DEMO

The screenshot shows the Compiler Explorer interface with two panes. The left pane displays the C++ source code for a "Hello World" application. The right pane shows the generated assembly code for the x86-64 architecture using gcc 12.2.

**C++ Source Code (Left):**

```
1 #include <stdio.h>
2
3 int main(int argc, char **argv) {
4
5     printf("Hello World!\n");
6
7     return 0;
8 }
```

**Assembly Output (Right):**

```
1 .LC0:
2     .string "Hello World!"
3
4 main:
5     push    rbp
6     mov     rbp, rsp
7     sub     rsp, 16
8     mov     DWORD PTR [rbp-4], edi
9     mov     QWORD PTR [rbp-16], rsi
10    mov     edi, OFFSET FLAT:.LC0
11    call    puts
12    mov     eax, 0
13    leave
14    ret
```

A red arrow points from the highlighted line in the source code ("printf("Hello World!\n");") to the corresponding assembly instruction (.string "Hello World!") in the assembly output.

By 潘甫翰 Sharkcode

18

C Output (0/0) x86-64 gcc 12.2 i - 764ms (3942B) ~252 lines filtered

(10, 1)

Compiler License

# Hello World DEMO

The image shows two side-by-side code editors from the Compiler Explorer interface. The left editor is titled "C++ source #1" and contains the following C++ code:

```
1 #include <stdio.h>
2
3 int main(int argc, char **argv) {
4
5     printf("Hello World!\n");
6
7     return 0;
8 }
```

The right editor is titled "x86-64 gcc 12.2 (Editor #1)" and displays the generated assembly code:

```
1 .LC0:
2     .string "Hello World!"
3
4 main:
5     push    rbp
6     mov     rbp, rsp
7     sub     rsp, 16
8     mov     DWORD PTR [rbp-4], edi
9     mov     QWORD PTR [rbp-16], rsi
10    mov    edi, OFFSET FLAT:.LC0
11    call   puts
12    mov    eax, 0
13
14    leave
15    ret
```

A red box highlights the assembly code from line 4 to 7, which corresponds to the function prologue. A yellow callout bubble points to this box with the text "Function Prologue". Another red box highlights the assembly code from line 14 to 15, which corresponds to the function epilogue. A second yellow callout bubble points to this box with the text "Function Epilogue".

At the bottom of the interface, there is an "Output" tab showing "Output (0/0) x86-64 gcc 12.2" and some performance metrics.

Page footer: By 潘甫翰 Sharkcode (10, 1) 19 Compiler License

# Hello World DEMO

The screenshot shows the Compiler Explorer interface with two panes. The left pane displays the C++ source code:

```
1 #include <stdio.h>
2
3 int main(int argc, char **argv) {
4
5     printf("Hello World!\n");
6
7     return 0;
8 }
```

The right pane shows the generated assembly code for x86-64 gcc 12.2:

```
1 .LC0:
2     .string "Hello World!"
3
4 main:
5     push    rbp
6     mov     rbp, rsp
7     sub    rsp, 16
8     mov     DWORD PTR [rbp-4], edi
9     mov     QWORD PTR [rbp-16], rsi
10    mov    edi, OFFSET FLAT:.LC0
11    call   puts
12    mov    eax, 0
13    leave
14    ret
```

A red box highlights the instruction `sub rsp, 16`, and a yellow callout bubble points to it with the text "Stack grows".

At the bottom, the output panel shows:

By 潘甫翰 Sharkcode 20

C Output (0/0) x86-64 gcc 12.2 i - 764ms (3942B) ~252 lines filtered E

(10, 1) Compiler License

# Hello World DEMO

The screenshot shows two panes in the Compiler Explorer interface. The left pane displays the C++ source code:

```
1 #include <stdio.h>
2
3 int main(int argc, char **argv) {
4
5     printf("Hello World!\n");
6
7     return 0;
8 }
```

The line `int main(int argc, char **argv)` is highlighted with a red box. The right pane shows the generated assembly code for x86-64 gcc 12.2:

```
1 .LC0:
2
3 main:
4     push    rbp
5     mov     rbp, rsp
6     sub    rsp, 16
7     mov     DWORD PTR [rbp-4], edi
8     mov     QWORD PTR [rbp-16], rsi
9     mov     edi, OFFSET FLAT:.LC0
10    call    puts
11    mov     eax, 0
12    leave
13    ret
```

A callout box points from the highlighted line in the source code to the corresponding assembly instructions. Another callout box highlights the assembly instruction `mov DWORD PTR [rbp-4], edi` and contains the text "Store arguments on stack". A purple box highlights the note about registers `%rdi, %rsi, %rdx, %rcx, %r8, and %r9` used for parameter passing.

By 潘甫翰 Sharkcode

21

C Output (0/0) x86-64 gcc 12.2 i - 764ms (3942B) ~252 lines filtered E

(10, 1)

Compiler License

# Hello World DEMO

The screenshot shows the Compiler Explorer interface with two panes. The left pane displays the C++ source code:

```
1 #include <stdio.h>
2
3 int main(int argc, char **argv) {
4
5     printf("Hello World!\n");
6
7     return 0;
8 }
```

The line `printf("Hello World!\n");` is highlighted with a red box. The right pane shows the generated assembly code for x86-64 gcc 12.2:

```
1 .LC0:
2     .string "Hello World!"
3 main:
4
5     mov    DWORD PTR [rbp-4], edi
6     mov    QWORD PTR [rbp-16], rsi
7     mov    edi, OFFSET FLAT:.LC0
8     call   puts
9     mov    eax, 0
10    leave 
11    ret
```

A callout box highlights the assembly instruction `mov edi, OFFSET FLAT:.LC0`, which corresponds to the printf call in the C code. A yellow arrow points from this callout to the original printf call in the C source code.

At the bottom, the status bar shows: Output (0/0) x86-64 gcc 12.2 - 764ms (3942B) ~252 lines filtered.

Page footer: By 潘甫翰 Sharkcode (10, 1) Compiler License 22

# Hello World DEMO

The screenshot shows the Compiler Explorer interface with two panes. The left pane displays the C++ source code for a "Hello World" program, and the right pane shows the generated assembly code.

**C++ Source Code:**

```
1 #include <stdio.h>
2
3 int main(int argc, char **argv) {
4
5     printf("Hello World!\n");
6
7     return 0;
8 }
9
10
```

The line `printf("Hello World!\n");` is highlighted with a red box.

**Assembly Output:**

```
1 .LC0:
2     .string "Hello World!"
3
4 main:
5     push    rbp
6     mov     rbp, rsp
7     sub     rsp, 16
8     mov     DWORD PTR [rbp-4], edi
9     mov     QWORD PTR [rbp-16], rsi
10    mov     edi, OFFSET FLAT:.LC0
11    call    puts
12    mov     eax, 0
13    leave
14    ret
```

The lines `mov edi, OFFSET FLAT:.LC0` and `call puts` are highlighted with a red box.

At the bottom, there is an "Output" tab showing "0/0" results, and a "Compiler License" link.

# Hello World DEMO

The screenshot shows the Compiler Explorer interface with two tabs: 'C++ source #1' and 'x86-64 gcc 12.2 (Editor #1)'. The C++ source code is:

```
1 #include <stdio.h>
2
3 int main(int argc, char **argv) {
4
5     printf("Hello World!\n");
6
7     return 0;
8 }
```

A tooltip box highlights the line 'return 0;' with a red border. The tooltip contains the following text:

There are sixteen 64-bit registers in x86-64: %rax, %rbx, %rcx, %rdx, %rdi, %rsi, %rbp, %rsp, and %r8-r15. Of these, %rax, %rcx, %rdx, %rdi, %rsi, %rsp, and %r8-r11 are considered caller-save registers, meaning that they are not necessarily saved across function calls. By convention, %rax is used to store a function's return value if it exists and is no more than 64 bits long. (Larger return types like structs are returned using the stack.) Registers %rbx, %rbp, and %r12-r15 are callee-save registers, meaning that they are saved across function calls. Register %rsp is used as the *stack pointer*, a pointer to the topmost element in the stack.

The assembly output is:

```
8     mov    QWORD PTR [rbp-16], rsi
9     mov    edi, OFFSET FLAT:.LC0
10    call   puts
11    mov    eax, 0
12    leave
13    ret
```

The instruction 'mov eax, 0' is highlighted with a yellow border. A yellow arrow points from the tooltip to this instruction.

At the bottom, the status bar shows: Output (0/0) x86-64 gcc 12.2 - 764ms (3942B) ~252 lines filtered. The page number '(10, 1)' is also visible.

omeD gninraeL etisbeW

<https://dogbolt.org/>

# Compare with several decompilers

 **Decompiler Explorer** What is this?

**Upload File**  
Your file must be less than 2MB in size. Uploaded binaries [are retained](#).

angr     BinaryNinja     Boomerang     dewolf     Ghidra     Hex-Rays     RecStudio     Reko     Relyze     RetDec     Snowman

**Samples**  
Or check out one of these samples we've provided:

A CTF Challenge on x86 Linux

**angr**

```
9.2.34
1 int __init()
2 {
3     return;
4     if (false)
5     {
6         0();
7         return;
8     }
9 }
10 long long sub_401020()
11 {
12     unsigned long long v0; // [bp-0x8]
13     v0 = 0;
14     goto *(4210576);
15 }
16 long long sub_401030()
17 {
18     int64_t var_8 = 0;
19     /* jump -> nullptr */
20 }
21 unsigned long long v0; // [bp-0x8]
22 v0 = 0;
23 return sub_401020();
24 }
25 }
26 long long sub_401040()
27 {
28     unsigned long long v0; // [bp-0x8]
29     v0 = 1;
30     return sub_401020();
31 }
32 }
```

By 潘雨翰 Sharkcode

**BinaryNinja**

```
3.3.3996 (e34a955e)
1 void __init()
2 {
3     if (__gmon_start__ != 0)
4     {
5         __gmon_start__();
6     }
7 }
8 int64_t sub_1020()
9 {
10     int64_t var_8 = 0;
11     /* jump -> nullptr */
12 }
13 int64_t sub_1030()
14 {
15     int64_t var_8 = 0;
16     /* tailcall */
17     return sub_1020();
18 }
19 unsigned long long v0; // [bp-0x8]
20 v0 = 0;
21 return sub_401020();
22 }
23 }
```

**Ghidra**

```
10.2.2 (9813cde2)
1 #include "out.h"
2
3
4
5 int __init(EVP_PKEY_CTX *ctx)
6
7 {
8     int iVar1;
9
10    iVar1 = __gmon_start__();
11    return iVar1;
12 }
13
14
15 void FUN_00101020(void)
16
17
18 {
19     // WARNING: Treating indirect jump
20     (*(code *)undefined *)0x0();
21     return;
22 }
23
24
25 void __cxa_finalize(void)
26
27
28 {
29     __cxa_finalize();
30     return;
31 }
32
33
34 }
```

**Hex-Rays**

```
8.2.0.221215
1
2
3
4
5 void __fastcall __libc_csu_init(unsigned int a1, _int
6
7 {
8     signed __int64 v3; // rbp
9     __int64 i; // rbx
10
11     init_proc();
12     v3 = &do_global_dtors_aux_fini_array_entry - &fra
13     if ( v3 )
14     {
15         for ( i = 0LL; i != v3; ++i )
16             (*(&frame_dummy_init_array_entry + i))();
17     }
18
19     // 3D80: using guessed type __int64 (__fastcall *fra
20     // 3D88: using guessed type __int64 (__fastcall *do_
21
22     // -----
23     void __libc_csu_fini(void)
24
25     ;
26
27     // -----
28     void term_proc()
29
30     ;
31
32     // -----
33     // nfuncs=41 queued=21 decompiled=21 lumina nreq=0 wo
34     // ALL OK, 21 function(s) have been successfully deco
35
36 }
```

# Lab

- ❖ <https://play.picoctf.org/practice>
  - ❖ ARMAssembly 1

# ARMssembly 1

ARMssembly 1 

Tags: [picoCTF 2021](#) [Reverse Engineering](#)

AUTHOR: PRANAY GARG

Description

For what argument does this program print `win` with variables `68`, `2` and `3`? File: [chall\\_1.S](#) Flag format: picoCTF{XXXXXXXX} -> (hex, lowercase, no 0x, and 32 bits. ex. 5614267 would be picoCTF{0055aabb})

Hints 

1 Shifts

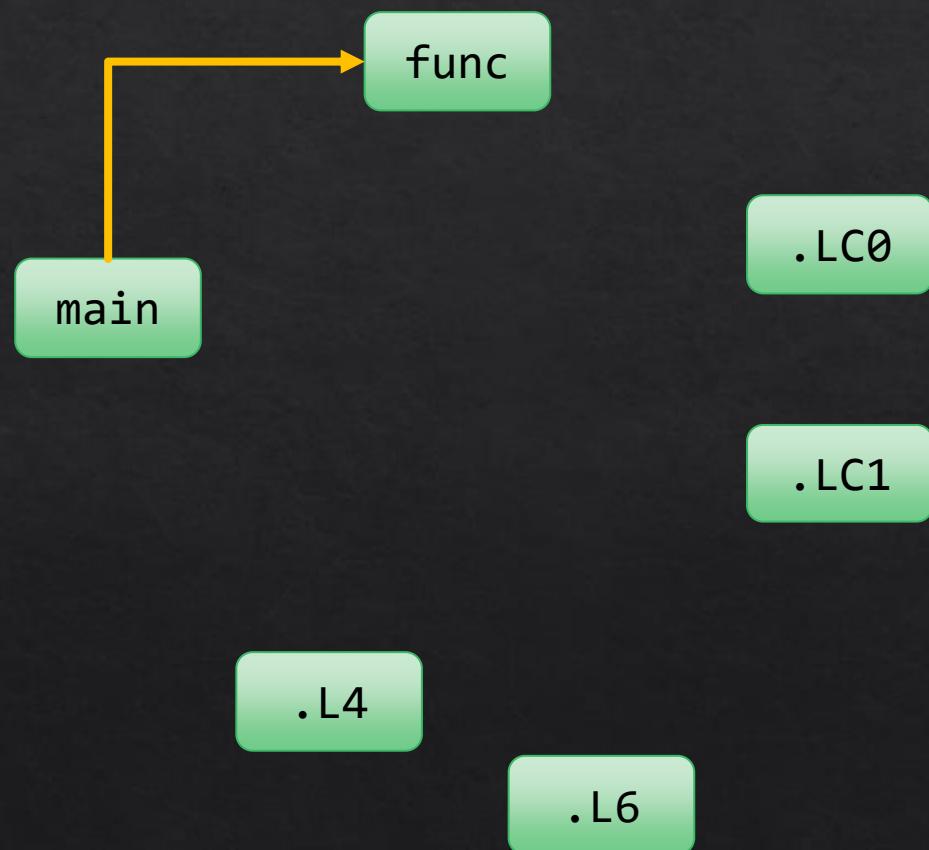
# Chall\_1.S

❖ armv8

```
.arch armv8-a
.file  "chall_1.c"
.text
.align 2
.global func
.type  func, %function
```

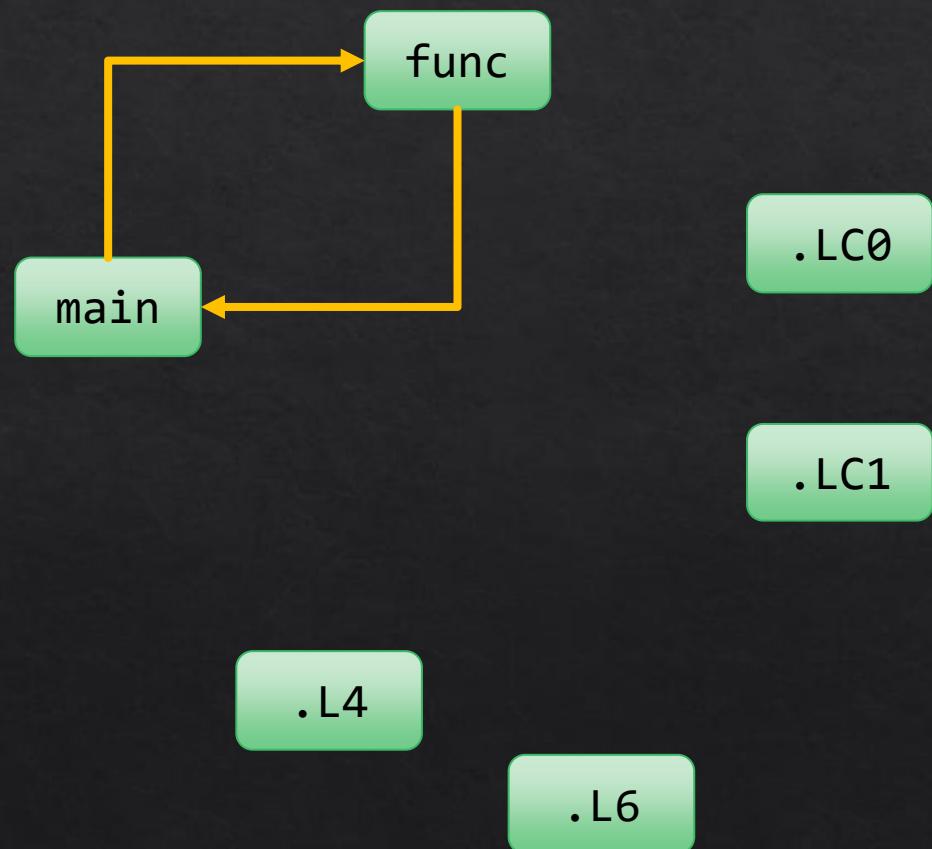
# Chall\_1.S

## ❖ Program Flow



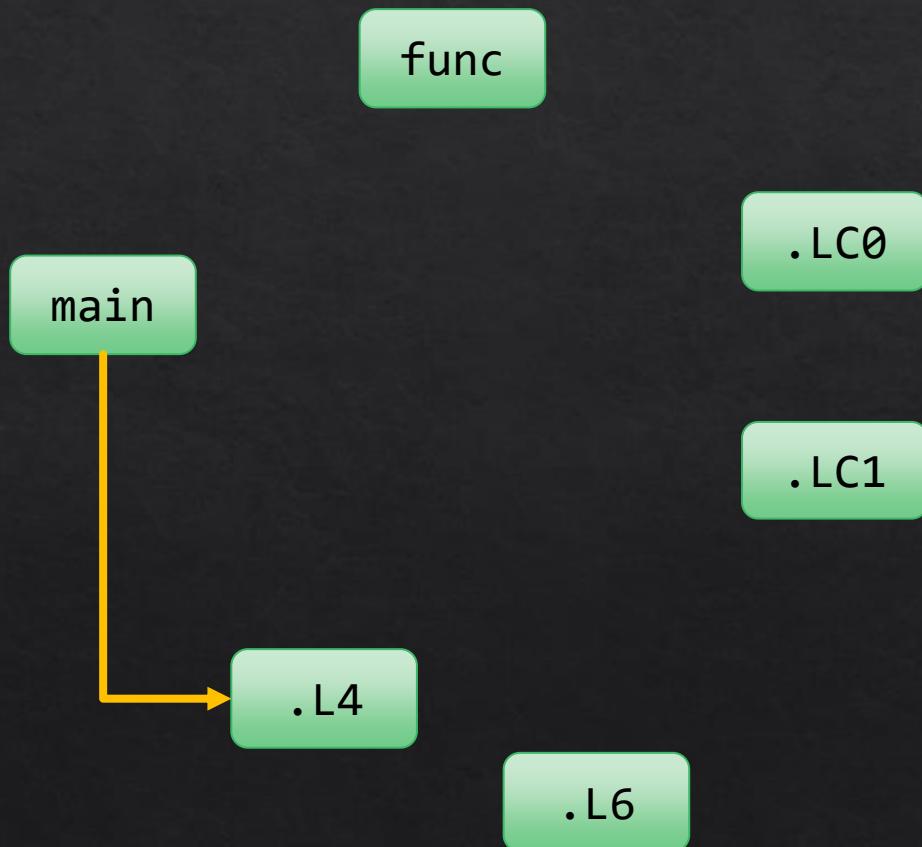
# Chall\_1.S

## ❖ Program Flow



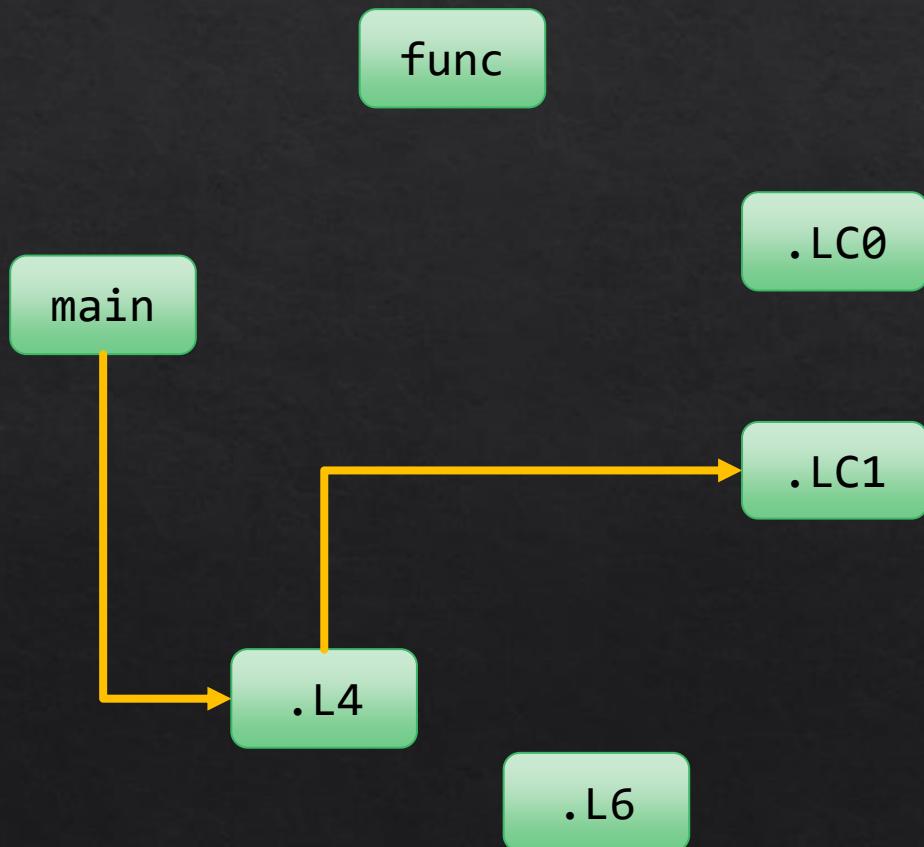
# Chall\_1.S

## ❖ Program Flow



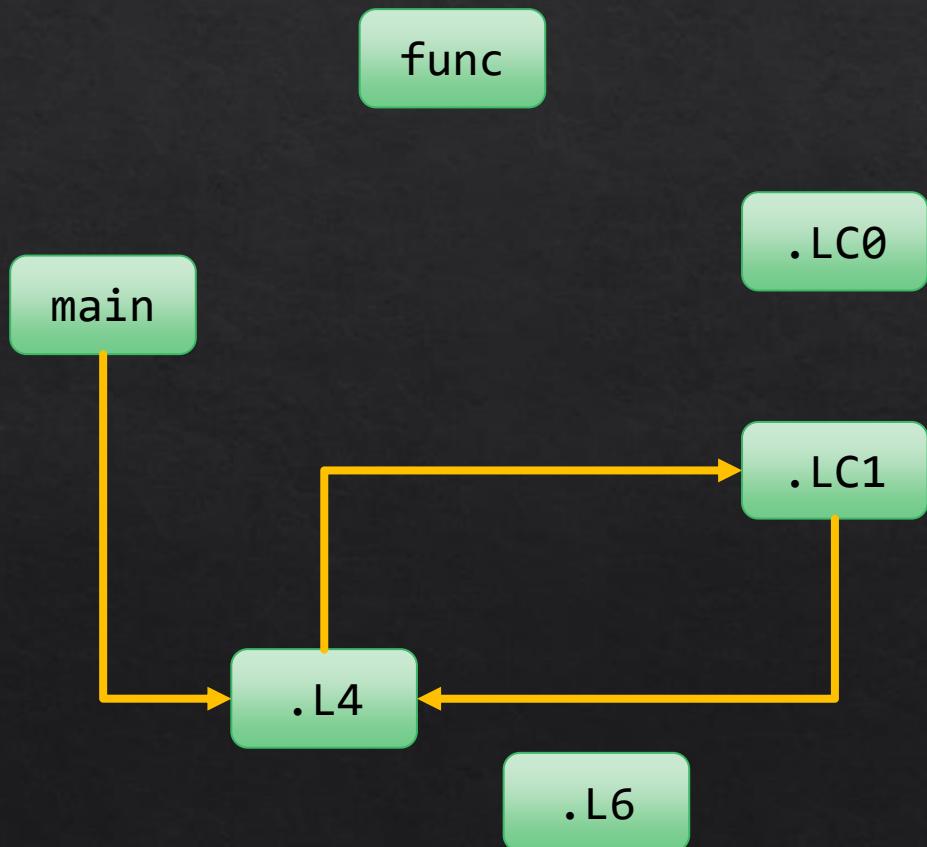
# Chall\_1.S

## ❖ Program Flow



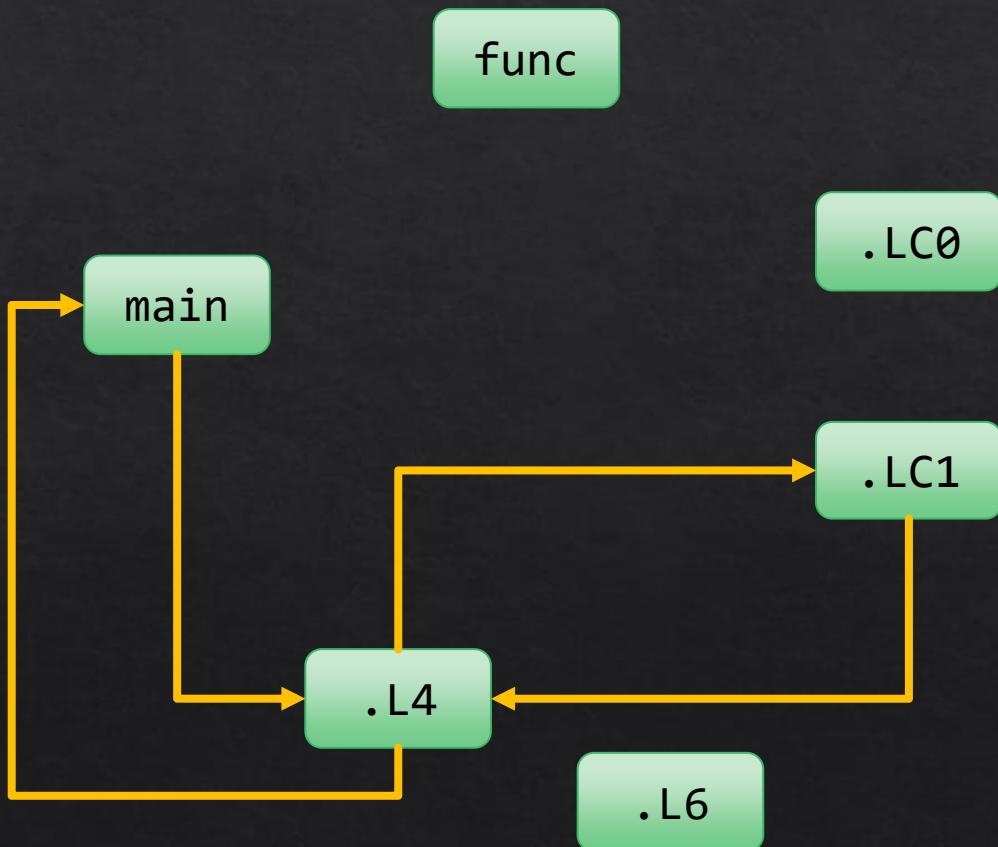
# Chall\_1.S

## ❖ Program Flow



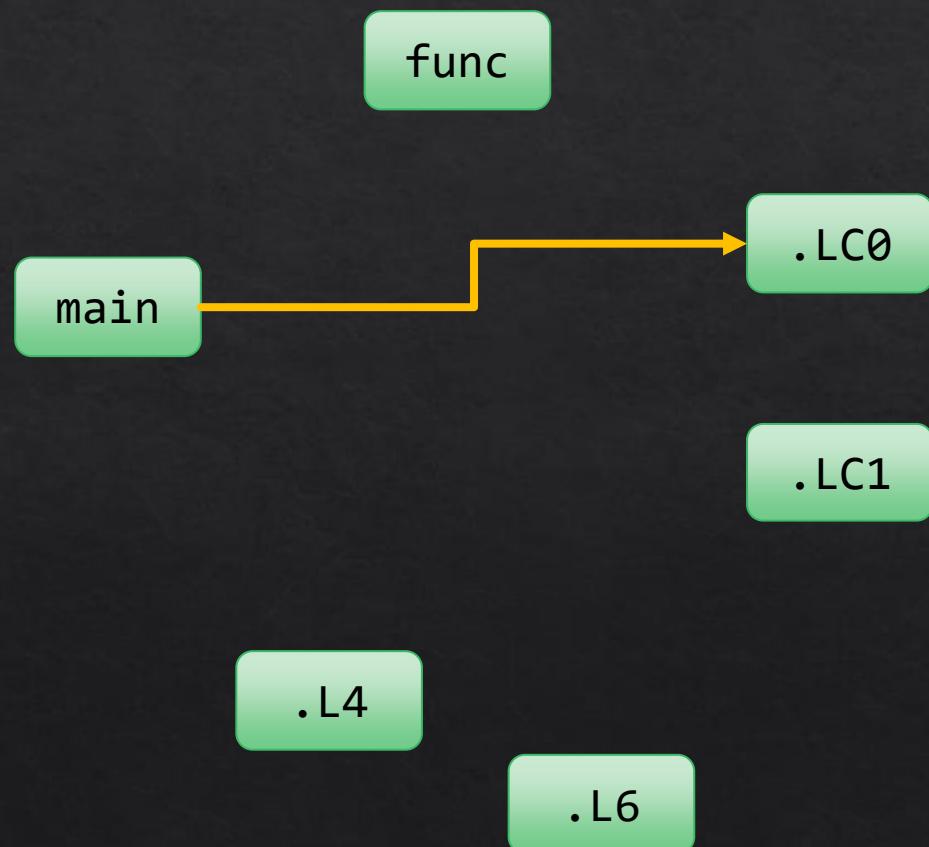
# Chall\_1.S

## ❖ Program Flow



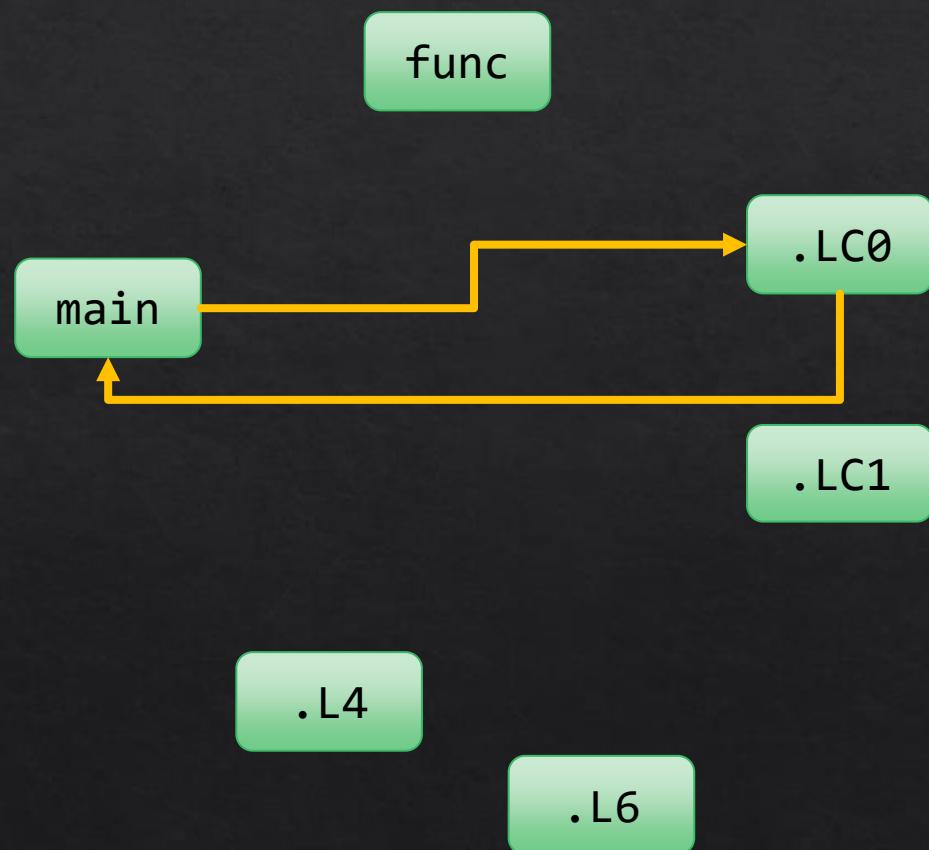
# Chall\_1.S

## ❖ Program Flow



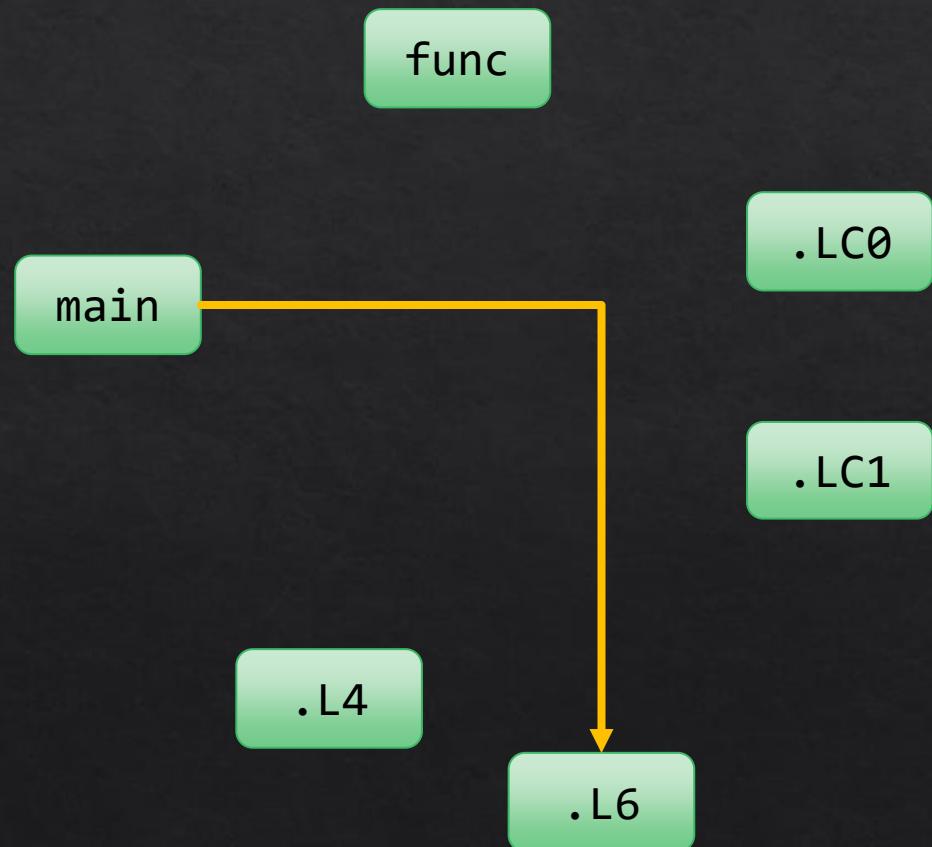
# Chall\_1.S

## ❖ Program Flow



# Chall\_1.S

## ❖ Program Flow



# Chall\_1.S

- ❖ How to win ?
  - ❖ When `func` returns 0 (`w0 == 0`) .

```
bl      atoi
str    w0, [x29, 44]
ldr    w0, [x29, 44]
bl      func
cmp    w0, 0
bne   .L4
adrp  x0, .LC0
add   x0, x0, :lo12:.LC0
bl      puts
b     .L6
```

.string "You win!"

# Chall\_1.S

◆ func

Register : Value

SP	
PC	
w0	x
w1	y



func:

```
sub    sp, sp, #32
str   w0, [sp, 12]
mov   w0, 68
str   w0, [sp, 16]
mov   w0, 2
str   w0, [sp, 20]
mov   w0, 3
str   w0, [sp, 24]
ldr   w0, [sp, 20]
ldr   w1, [sp, 16]
lsl   w0, w1, w0
str   w0, [sp, 28]
ldr   w1, [sp, 28]
ldr   w0, [sp, 24]
sdiv  w0, w1, w0
str   w0, [sp, 28]
ldr   w1, [sp, 28]
ldr   w0, [sp, 12]
sub   w0, w1, w0
str   w0, [sp, 28]
ldr   w0, [sp, 28]
add   sp, sp, 32
ret
.size  func, .-func^40
.section .rodata
.align 3
```

# Chall\_1.S

◆ func

Register : Value

SP	
PC	
w0	x
w1	y

Stack

???

???

???

???

???

???

???

func:

```
sub    sp, sp, #32
str   w0, [sp, 12]
mov   w0, 68
str   w0, [sp, 16]
mov   w0, 2
str   w0, [sp, 20]
mov   w0, 3
str   w0, [sp, 24]
ldr   w0, [sp, 20]
ldr   w1, [sp, 16]
lsl   w0, w1, w0
str   w0, [sp, 28]
ldr   w1, [sp, 28]
ldr   w0, [sp, 24]
sdiv  w0, w1, w0
str   w0, [sp, 28]
ldr   w1, [sp, 28]
ldr   w0, [sp, 12]
sub   w0, w1, w0
str   w0, [sp, 28]
ldr   w0, [sp, 28]
add   sp, sp, 32
ret
.size  func, .-func41
.section .rodata
.align 3
```

# Chall\_1.S

## func

Register : Value

SP	
PC	
w0	x
w1	y

By 潘甫翰 Sharkcode

Stack



???

x

???

???

???

???

???

func:

```
sub    sp, sp, #32
str   w0, [sp, 12]
mov   w0, 68
str   w0, [sp, 16]
mov   w0, 2
str   w0, [sp, 20]
mov   w0, 3
str   w0, [sp, 24]
ldr   w0, [sp, 20]
ldr   w1, [sp, 16]
lsl   w0, w1, w0
str   w0, [sp, 28]
ldr   w1, [sp, 28]
ldr   w0, [sp, 24]
sdiv  w0, w1, w0
str   w0, [sp, 28]
ldr   w1, [sp, 28]
ldr   w0, [sp, 12]
sub   w0, w1, w0
str   w0, [sp, 28]
ldr   w0, [sp, 28]
add   sp, sp, 32
ret
.size  func, .-func42
.section .rodata
.align 3
```

# Chall\_1.S

◆ func

Register : Value

SP	
PC	
w0	68
w1	y

Stack

???

x

???

???

???

???

???

func:

```
sub    sp, sp, #32
str   w0, [sp, 12]
mov   w0, 68
str   w0, [sp, 16]
mov   w0, 2
str   w0, [sp, 20]
mov   w0, 3
str   w0, [sp, 24]
ldr   w0, [sp, 20]
ldr   w1, [sp, 16]
lsl   w0, w1, w0
str   w0, [sp, 28]
ldr   w1, [sp, 28]
ldr   w0, [sp, 24]
sdiv  w0, w1, w0
str   w0, [sp, 28]
ldr   w1, [sp, 28]
ldr   w0, [sp, 12]
sub   w0, w1, w0
str   w0, [sp, 28]
ldr   w0, [sp, 28]
add   sp, sp, 32
ret
.size  func, .-func43
.section .rodata
.align 3
```

# Chall\_1.S

◆ func

Register : Value

SP	
PC	
w0	68
w1	y

By 潘甫翰 Sharkcode

Stack



func:

```
sub    sp, sp, #32
str   w0, [sp, 12]
mov   w0, 68
str   w0, [sp, 16]
mov   w0, 2
str   w0, [sp, 20]
mov   w0, 3
str   w0, [sp, 24]
ldr   w0, [sp, 20]
ldr   w1, [sp, 16]
lsl   w0, w1, w0
str   w0, [sp, 28]
ldr   w1, [sp, 28]
ldr   w0, [sp, 24]
sdiv  w0, w1, w0
str   w0, [sp, 28]
ldr   w1, [sp, 28]
ldr   w0, [sp, 12]
sub   w0, w1, w0
str   w0, [sp, 28]
ldr   w0, [sp, 28]
add   sp, sp, 32
ret
.size  func, .-func44
.section .rodata
.align 3
```

# Chall\_1.S

◆ func

Register : Value

SP	
PC	
w0	2
w1	y

By 潘甫翰 Sharkcode

Stack



func:

```
sub    sp, sp, #32
str   w0, [sp, 12]
mov   w0, 68
str   w0, [sp, 16]
mov   w0, 2
str   w0, [sp, 20]
mov   w0, 3
str   w0, [sp, 24]
ldr   w0, [sp, 20]
ldr   w1, [sp, 16]
lsl   w0, w1, w0
str   w0, [sp, 28]
ldr   w1, [sp, 28]
ldr   w0, [sp, 24]
sdiv  w0, w1, w0
str   w0, [sp, 28]
ldr   w1, [sp, 28]
ldr   w0, [sp, 12]
sub   w0, w1, w0
str   w0, [sp, 28]
ldr   w0, [sp, 28]
add   sp, sp, 32
ret
.size  func, .-func45
.section .rodata
.align 3
```

# Chall\_1.S

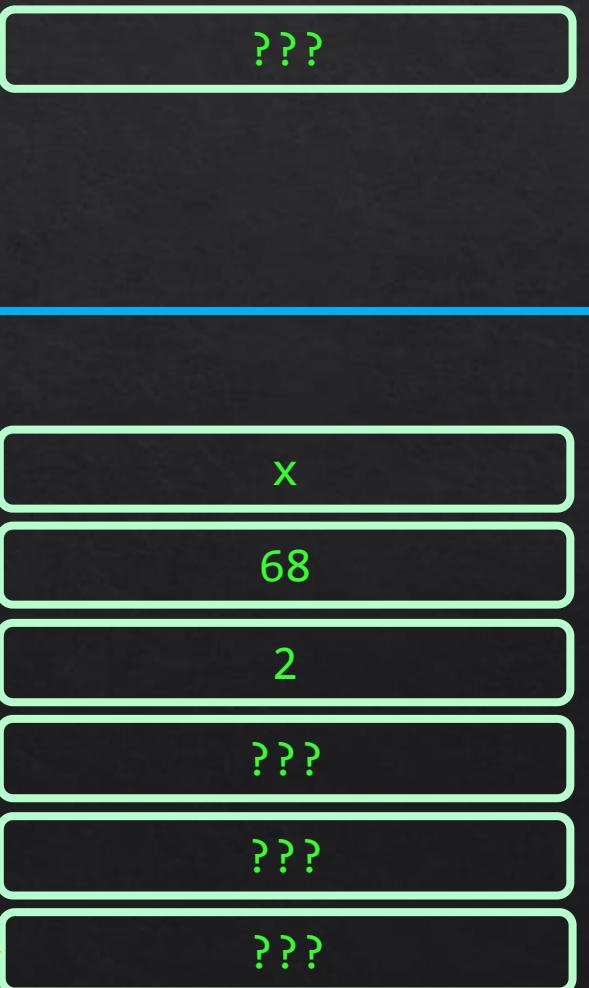
◆ func

Register : Value

SP	
PC	
w0	2
w1	y

By 潘甫翰 Sharkcode

Stack



func:

```
sub    sp, sp, #32
str   w0, [sp, 12]
mov   w0, 68
str   w0, [sp, 16]
mov   w0, 2
str   w0, [sp, 20]
mov   w0, 3
str   w0, [sp, 24]
ldr   w0, [sp, 20]
ldr   w1, [sp, 16]
lsl   w0, w1, w0
str   w0, [sp, 28]
ldr   w1, [sp, 28]
ldr   w0, [sp, 24]
sdiv  w0, w1, w0
str   w0, [sp, 28]
ldr   w1, [sp, 28]
ldr   w0, [sp, 12]
sub   w0, w1, w0
str   w0, [sp, 28]
ldr   w0, [sp, 28]
add   sp, sp, 32
ret
.size  func, .-func46
.section .rodata
.align 3
```

# Chall\_1.S

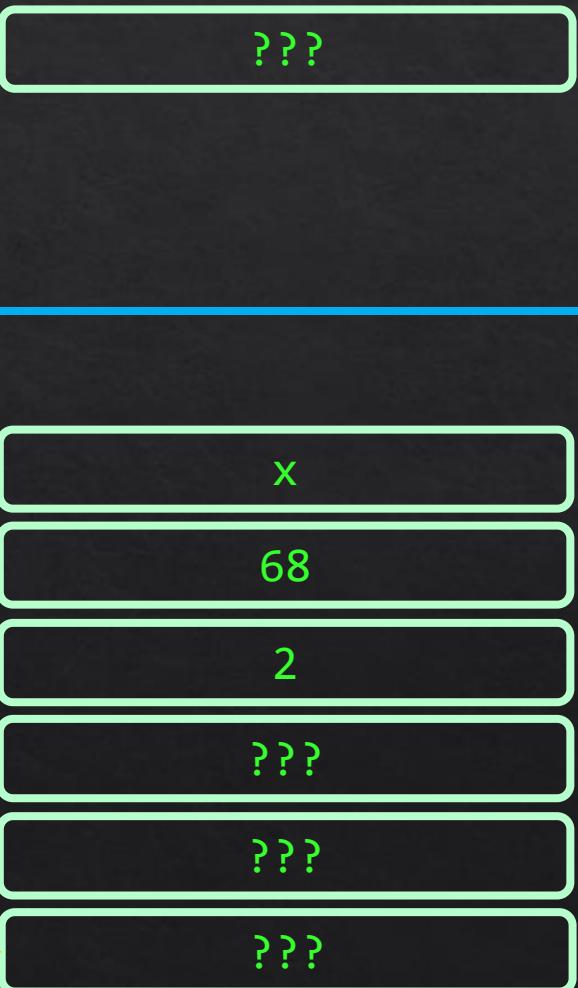
◆ func

Register : Value

SP	
PC	
w0	3
w1	y

By 潘甫翰 Sharkcode

Stack



func:

```
sub    sp, sp, #32
str   w0, [sp, 12]
mov   w0, 68
str   w0, [sp, 16]
mov   w0, 2
str   w0, [sp, 20]
mov   w0, 3
str   w0, [sp, 24]
ldr   w0, [sp, 20]
ldr   w1, [sp, 16]
lsl   w0, w1, w0
str   w0, [sp, 28]
ldr   w1, [sp, 28]
ldr   w0, [sp, 24]
sdiv  w0, w1, w0
str   w0, [sp, 28]
ldr   w1, [sp, 28]
ldr   w0, [sp, 12]
sub   w0, w1, w0
str   w0, [sp, 28]
ldr   w0, [sp, 28]
add   sp, sp, 32
ret
.size  func, .-func47
.section .rodata
.align 3
```

# Chall\_1.S

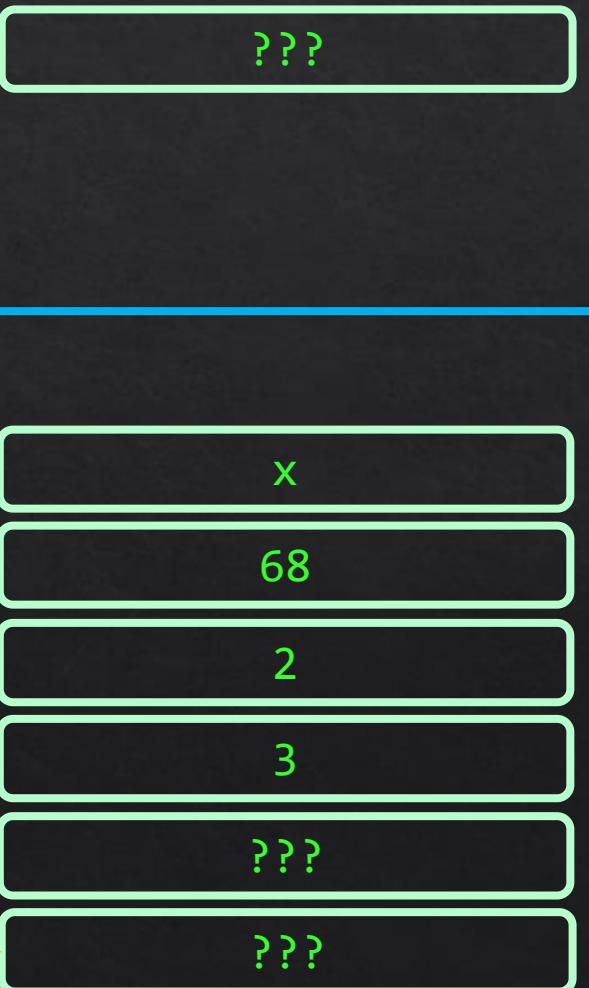
## func

Register : Value

SP	
PC	
w0	3
w1	y

By 潘甫翰 Sharkcode

Stack



func:

```
sub    sp, sp, #32
str   w0, [sp, 12]
mov   w0, 68
str   w0, [sp, 16]
mov   w0, 2
str   w0, [sp, 20]
mov   w0, 3
str   w0, [sp, 24]
ldr   w0, [sp, 20]
ldr   w1, [sp, 16]
lsl   w0, w1, w0
str   w0, [sp, 28]
ldr   w1, [sp, 28]
ldr   w0, [sp, 24]
sdiv  w0, w1, w0
str   w0, [sp, 28]
ldr   w1, [sp, 28]
ldr   w0, [sp, 12]
sub   w0, w1, w0
str   w0, [sp, 28]
ldr   w0, [sp, 28]
add   sp, sp, 32
ret
.size  func, .-func48
.section .rodata
.align 3
```

# Chall\_1.S

◆ func

Register : Value

SP	
PC	
w0	2
w1	y

By 潘甫翰 Sharkcode

Stack



func:

```
sub    sp, sp, #32
str   w0, [sp, 12]
mov   w0, 68
str   w0, [sp, 16]
mov   w0, 2
str   w0, [sp, 20]
mov   w0, 3
str   w0, [sp, 24]
ldr   w0, [sp, 20]
ldr   w1, [sp, 16]
lsl   w0, w1, w0
str   w0, [sp, 28]
ldr   w1, [sp, 28]
ldr   w0, [sp, 24]
sdiv  w0, w1, w0
str   w0, [sp, 28]
ldr   w1, [sp, 28]
ldr   w0, [sp, 12]
sub   w0, w1, w0
str   w0, [sp, 28]
ldr   w0, [sp, 28]
add   sp, sp, 32
ret
.size  func, .-func49
.section .rodata
.align 3
```

# Chall\_1.S

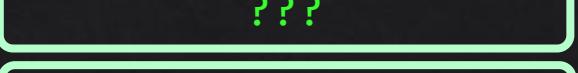
◆ func

Register : Value

SP	
PC	
w0	2
w1	68

By 潘甫翰 Sharkcode

Stack



func:

```
sub    sp, sp, #32
str   w0, [sp, 12]
mov   w0, 68
str   w0, [sp, 16]
mov   w0, 2
str   w0, [sp, 20]
mov   w0, 3
str   w0, [sp, 24]
ldr   w0, [sp, 20]
ldr   w1, [sp, 16]
lsl   w0, w1, w0
str   w0, [sp, 28]
ldr   w1, [sp, 28]
ldr   w0, [sp, 24]
sdiv  w0, w1, w0
str   w0, [sp, 28]
ldr   w1, [sp, 28]
ldr   w0, [sp, 12]
sub   w0, w1, w0
str   w0, [sp, 28]
ldr   w0, [sp, 28]
add   sp, sp, 32
ret
.size  func, .-func50
.section .rodata
.align 3
```

w0 = 68 << 2 = 272

# Chall\_1.S

◆ func

Register : Value

SP	
PC	
w0	272
w1	68

By 潘甫翰 Sharkcode

Stack



func:

```
sub    sp, sp, #32
str   w0, [sp, 12]
mov   w0, 68
str   w0, [sp, 16]
mov   w0, 2
str   w0, [sp, 20]
mov   w0, 3
str   w0, [sp, 24]
ldr   w0, [sp, 20]
ldr   w1, [sp, 16]
lsl   w0, w1, w0
str   w0, [sp, 28]
ldr   w1, [sp, 28]
ldr   w0, [sp, 24]
sdiv  w0, w1, w0
str   w0, [sp, 28]
ldr   w1, [sp, 28]
ldr   w0, [sp, 12]
sub   w0, w1, w0
str   w0, [sp, 28]
ldr   w0, [sp, 28]
add   sp, sp, 32
ret
.size  func, .-func51
.section .rodata
.align 3
```

# Chall\_1.S

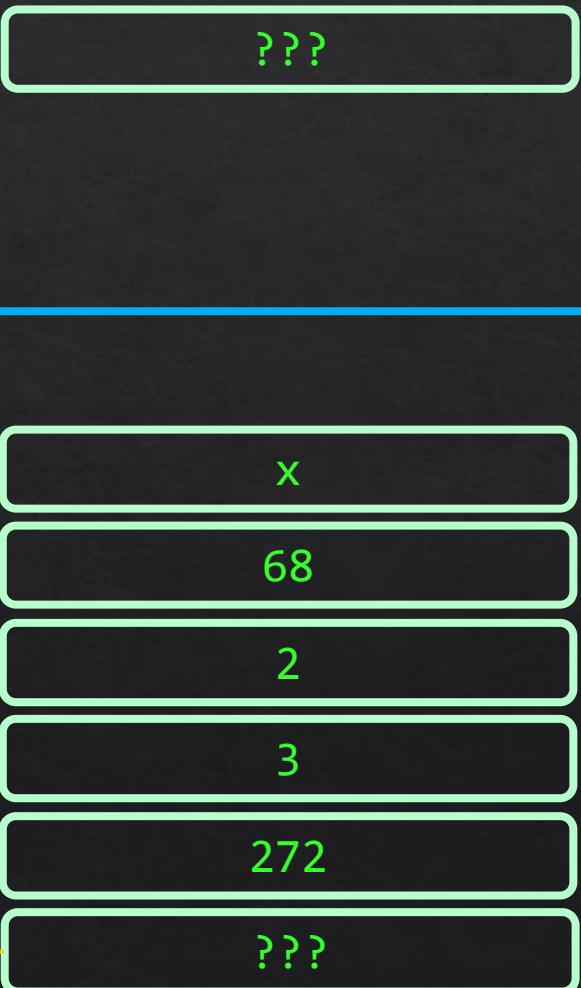
◆ func

Register : Value

SP	
PC	
w0	272
w1	68

By 潘甫翰 Sharkcode

Stack



func:

```
sub    sp, sp, #32
str   w0, [sp, 12]
mov   w0, 68
str   w0, [sp, 16]
mov   w0, 2
str   w0, [sp, 20]
mov   w0, 3
str   w0, [sp, 24]
ldr   w0, [sp, 20]
ldr   w1, [sp, 16]
lsl   w0, w1, w0
str   w0, [sp, 28]
ldr   w1, [sp, 28]
ldr   w0, [sp, 24]
sdiv  w0, w1, w0
str   w0, [sp, 28]
ldr   w1, [sp, 28]
ldr   w0, [sp, 12]
sub   w0, w1, w0
str   w0, [sp, 28]
ldr   w0, [sp, 28]
add   sp, sp, 32
ret
.size  func, .-func52
.section .rodata
.align 3
```

# Chall\_1.S

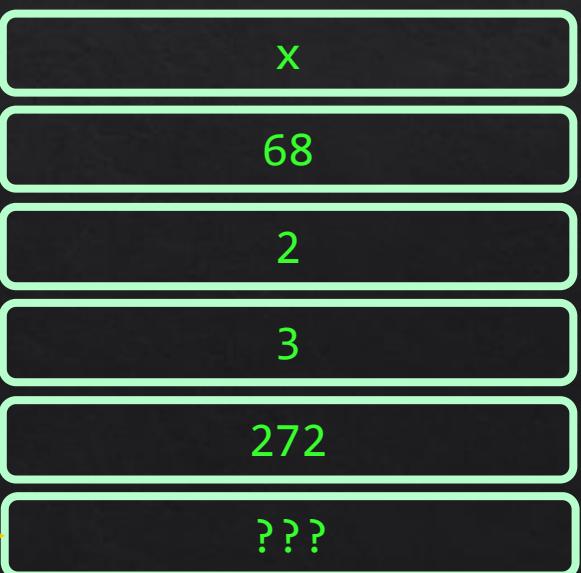
◆ func

Register : Value

SP	
PC	
w0	272
w1	272

By 潘甫翰 Sharkcode

Stack



func:

```
sub    sp, sp, #32
str   w0, [sp, 12]
mov   w0, 68
str   w0, [sp, 16]
mov   w0, 2
str   w0, [sp, 20]
mov   w0, 3
str   w0, [sp, 24]
ldr   w0, [sp, 20]
ldr   w1, [sp, 16]
lsl   w0, w1, w0
str   w0, [sp, 28]
ldr   w1, [sp, 28]
ldr   w0, [sp, 24]
sdiv  w0, w1, w0
str   w0, [sp, 28]
ldr   w1, [sp, 28]
ldr   w0, [sp, 12]
sub   w0, w1, w0
str   w0, [sp, 28]
ldr   w0, [sp, 28]
add   sp, sp, 32
ret
.size  func, .-func53
.section .rodata
.align 3
```

# Chall\_1.S

## func

Register : Value

SP	
PC	
w0	3
w1	272

By 潘甫翰 Sharkcode

Stack



func:

```
sub    sp, sp, #32
str   w0, [sp, 12]
mov   w0, 68
str   w0, [sp, 16]
mov   w0, 2
str   w0, [sp, 20]
mov   w0, 3
str   w0, [sp, 24]
ldr   w0, [sp, 20]
ldr   w1, [sp, 16]
lsl   w0, w1, w0
str   w0, [sp, 28]
ldr   w1, [sp, 28]
ldr   w0, [sp, 24]
sdiv  w0, w1, w0
str   w0, [sp, 28]
ldr   w1, [sp, 28]
ldr   w0, [sp, 12]
sub   w0, w1, w0
str   w0, [sp, 28]
ldr   w0, [sp, 28]
add   sp, sp, 32
ret
.size  func, .-func54
.section .rodata
.align 3
```

$$w0 = 272 / 3 = 90$$

# Chall\_1.S

◆ func

Register : Value

SP	
PC	
w0	90
w1	272

By 潘甫翰 Sharkcode



func:

```

sub    sp, sp, #32
str   w0, [sp, 12]
mov   w0, 68
str   w0, [sp, 16]
mov   w0, 2
str   w0, [sp, 20]
mov   w0, 3
str   w0, [sp, 24]
ldr   w0, [sp, 20]
ldr   w1, [sp, 16]
lsl   w0, w1, w0
str   w0, [sp, 28]
ldr   w1, [sp, 28]
ldr   w0, [sp, 24]
sdiv  w0, w1, w0
str   w0, [sp, 28]
ldr   w1, [sp, 28]
ldr   w0, [sp, 12]
sub   w0, w1, w0
str   w0, [sp, 28]
ldr   w0, [sp, 28]
add   sp, sp, 32
ret
.size  func, .-func55
.section .rodata
.align 3

```

# Chall\_1.S

## func

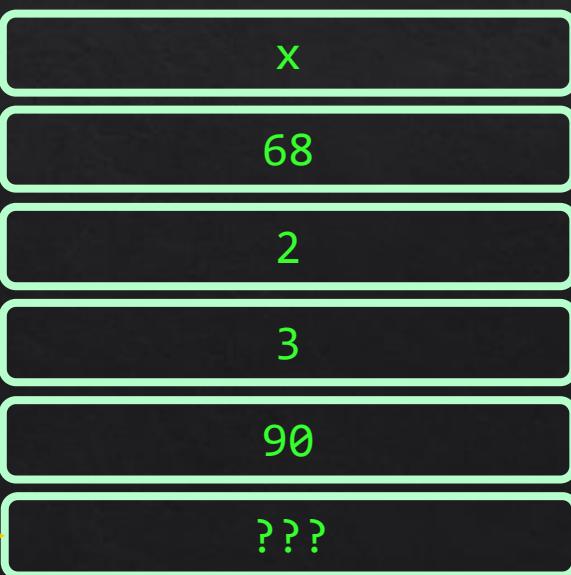
Register : Value

SP	
PC	
w0	90
w1	272

By 潘甫翰 Sharkcode

Stack

???



func:

```
sub    sp, sp, #32
str   w0, [sp, 12]
mov   w0, 68
str   w0, [sp, 16]
mov   w0, 2
str   w0, [sp, 20]
mov   w0, 3
str   w0, [sp, 24]
ldr   w0, [sp, 20]
ldr   w1, [sp, 16]
lsl   w0, w1, w0
str   w0, [sp, 28]
ldr   w1, [sp, 28]
ldr   w0, [sp, 24]
sdiv  w0, w1, w0
str   w0, [sp, 28]
ldr   w1, [sp, 28]
ldr   w0, [sp, 12]
sub   w0, w1, w0
str   w0, [sp, 28]
ldr   w0, [sp, 28]
add   sp, sp, 32
ret
.size  func, .-func56
.section .rodata
.align 3
```

# Chall\_1.S

◆ func

Register : Value

SP	
PC	
w0	90
w1	90

By 潘甫翰 Sharkcode

Stack

???

x

68

2

3

90

???

func:

```
sub    sp, sp, #32
str   w0, [sp, 12]
mov   w0, 68
str   w0, [sp, 16]
mov   w0, 2
str   w0, [sp, 20]
mov   w0, 3
str   w0, [sp, 24]
ldr   w0, [sp, 20]
ldr   w1, [sp, 16]
lsl   w0, w1, w0
str   w0, [sp, 28]
ldr   w1, [sp, 28]
ldr   w0, [sp, 24]
sdiv  w0, w1, w0
str   w0, [sp, 28]
ldr   w1, [sp, 28]
ldr   w0, [sp, 12]
sub   w0, w1, w0
str   w0, [sp, 28]
ldr   w0, [sp, 28]
add   sp, sp, 32
ret
.size  func, .-func57
.section .rodata
.align 3
```

# Chall\_1.S

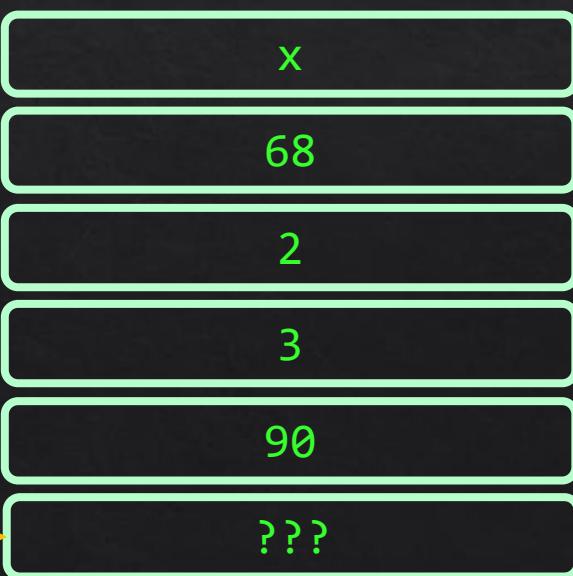
◆ func

Register : Value

SP	
PC	
w0	x
w1	90

Stack

???



func:

```
sub    sp, sp, #32
str   w0, [sp, 12]
mov   w0, 68
str   w0, [sp, 16]
mov   w0, 2
str   w0, [sp, 20]
mov   w0, 3
str   w0, [sp, 24]
ldr   w0, [sp, 20]
ldr   w1, [sp, 16]
lsl   w0, w1, w0
str   w0, [sp, 28]
ldr   w1, [sp, 28]
ldr   w0, [sp, 24]
sdiv  w0, w1, w0
str   w0, [sp, 28]
ldr   w1, [sp, 28]
ldr   w0, [sp, 12]
sub   w0, w1, w0
str   w0, [sp, 28]
ldr   w0, [sp, 28]
add   sp, sp, 32
ret
.size  func, .-func58
.section .rodata
.align 3
```

# Chall\_1.S

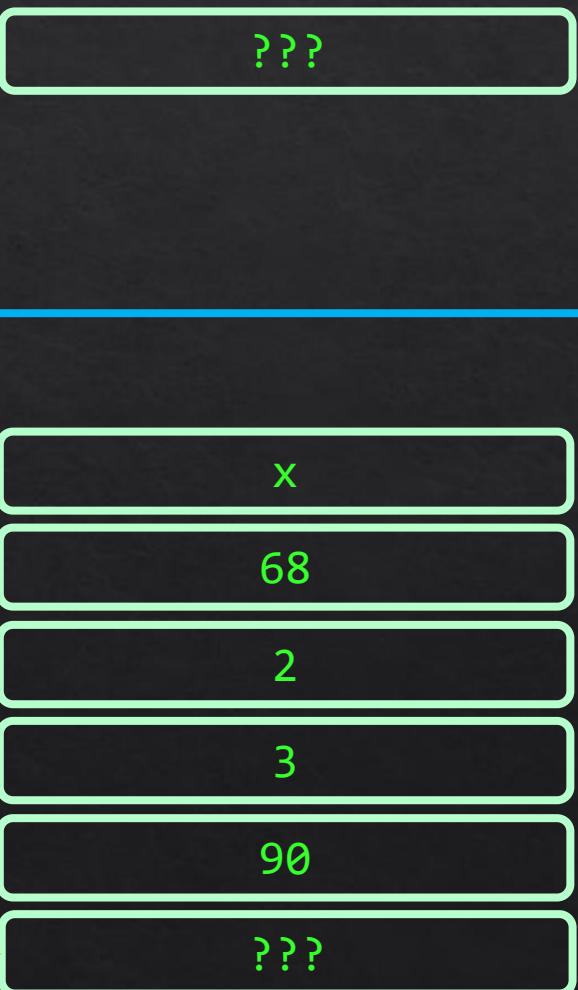
◆ func

Register : Value

SP	
PC	
w0	90-x
w1	90

By 潘甫翰 Sharkcode

Stack



func:

```
sub    sp, sp, #32
str   w0, [sp, 12]
mov   w0, 68
str   w0, [sp, 16]
mov   w0, 2
str   w0, [sp, 20]
mov   w0, 3
str   w0, [sp, 24]
ldr   w0, [sp, 20]
ldr   w1, [sp, 16]
lsl   w0, w1, w0
str   w0, [sp, 28]
ldr   w1, [sp, 28]
ldr   w0, [sp, 24]
sdiv  w0, w1, w0
str   w0, [sp, 28]
ldr   w1, [sp, 28]
ldr   w0, [sp, 12]
sub   w0, w1, w0
str   w0, [sp, 28]
ldr   w0, [sp, 28]
add   sp, sp, 32
ret
.size  func, .-func59
.section .rodata
.align 3
```

# Chall\_1.S

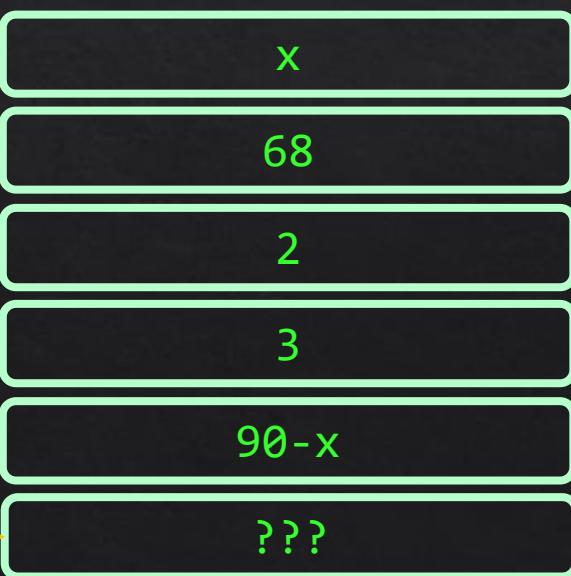
◆ func

Register : Value

SP	
PC	
w0	90-x
w1	90

Stack

???



By 潘甫翰 Sharkcode

func:

```
sub    sp, sp, #32
str   w0, [sp, 12]
mov   w0, 68
str   w0, [sp, 16]
mov   w0, 2
str   w0, [sp, 20]
mov   w0, 3
str   w0, [sp, 24]
ldr   w0, [sp, 20]
ldr   w1, [sp, 16]
lsl   w0, w1, w0
str   w0, [sp, 28]
ldr   w1, [sp, 28]
ldr   w0, [sp, 24]
sdiv  w0, w1, w0
str   w0, [sp, 28]
ldr   w1, [sp, 28]
ldr   w0, [sp, 12]
sub   w0, w1, w0
str   w0, [sp, 28]
ldr   w0, [sp, 28]
add   sp, sp, 32
ret
.size  func, .-func^60
.section .rodata
.align 3
```

# Chall\_1.S

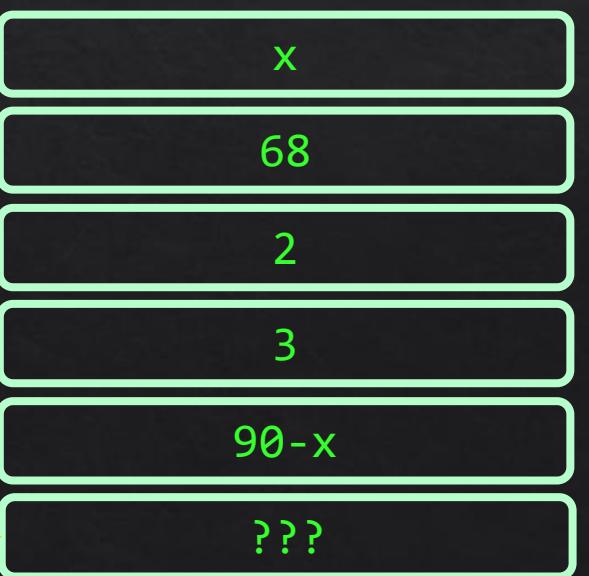
◆ func

Register : Value

SP	
PC	
w0	90-x
w1	90

By 潘甫翰 Sharkcode

Stack



func:

```
sub    sp, sp, #32
str   w0, [sp, 12]
mov   w0, 68
str   w0, [sp, 16]
mov   w0, 2
str   w0, [sp, 20]
mov   w0, 3
str   w0, [sp, 24]
ldr   w0, [sp, 20]
ldr   w1, [sp, 16]
lsl   w0, w1, w0
str   w0, [sp, 28]
ldr   w1, [sp, 28]
ldr   w0, [sp, 24]
sdiv  w0, w1, w0
str   w0, [sp, 28]
ldr   w1, [sp, 28]
ldr   w0, [sp, 12]
sub   w0, w1, w0
str   w0, [sp, 28]
ldr   w0, [sp, 28]
add   sp, sp, 32
ret
.size  func, .-func61
.section .rodata
.align 3
```

# Chall\_1.S

◆ func

Register : Value

SP	
PC	
w0	90-x
w1	90



func:

```
sub    sp, sp, #32
str   w0, [sp, 12]
mov   w0, 68
str   w0, [sp, 16]
mov   w0, 2
str   w0, [sp, 20]
mov   w0, 3
str   w0, [sp, 24]
ldr   w0, [sp, 20]
ldr   w1, [sp, 16]
lsl   w0, w1, w0
str   w0, [sp, 28]
ldr   w1, [sp, 28]
ldr   w0, [sp, 24]
sdiv  w0, w1, w0
str   w0, [sp, 28]
ldr   w1, [sp, 28]
ldr   w0, [sp, 12]
sub   w0, w1, w0
str   w0, [sp, 28]
ldr   w0, [sp, 28]
add   sp, sp, 32
ret
.size  func, .-func
.section .rodata
.align 3
```

ret

◆ pop rip

# Chall\_1.S

## ◆ Main

◆ We need  $x=90$  to win.

Register : Value

SP	
PC	
w0	90-x
w1	90

main:

```
    stp    x29, x30, [sp, -48]!
    add    x29, sp, 0
    str    w0, [x29, 28]
    str    x1, [x29, 16]
    ldr    x0, [x29, 16]
    add    x0, x0, 8
    ldr    x0, [x0]
    bl    atoi
    str    w0, [x29, 44]
    ldr    w0, [x29, 44]
    bl    func
    cmp    w0, 0
    bne    .L4
    adrp   x0, .LC0
    add    x0, x0, :lo12:.LC0
    bl    puts
    b     .L6
```

cmp 90-x == 0

x=90=5a

Flag

picoCTF{0000005a}

# Tools

GDB, GHIDRA, pwntools

# Tools

## ❖ demo.c

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 typedef struct {
5     int id;
6     char name[50];
7     float grade;
8 } Student;
9
10 int main() {
11
12     setvbuf(stdout,0,2,0);
13     setvbuf(stdin,0,2,0);
14     setvbuf(stderr,0,2,0);
15
16     int n;
17     printf("Enter the number of students: ");
18     scanf("%d", &n);
19     Student *students = (Student*)malloc(n * sizeof(Student));
20     for (int i = 0; i < n; i++) {
21         printf("Enter student %d's ID: ", i+1);
22         scanf("%d", &(students[i].id));
23         printf("Enter student %d's name: ", i+1);
24         scanf("%s", students[i].name);
25         printf("Enter student %d's grade: ", i+1);
26         scanf("%f", &(students[i].grade));
27     }
28     printf("Student information:\n");
29     for (int i = 0; i < n; i++) {
30         printf("ID: %d, Name: %s, Grade: %.2f\n", students[i].id, students[i].name, students[i].grade);
31     }
32     free(students);
33     return 0;
34 }
```

# Tools -- GHIDRA

◆ 22e

The screenshot shows the GHIDRA interface with three main panes:

- Program Trees** pane on the left, showing the file structure of the demo executable.
- Listing** pane in the center, displaying assembly code for the main function. A specific instruction at address 0010122e is highlighted with a red box: `0010122e e8 bd CALL <EXTERNAL>::__isoc99_scanf`.
- Decompiler** pane on the right, showing the decompiled C code for the main function. The highlighted line is: `__isoc99_scanf(&DAT_00102027,&local_24);`

**Symbol Tree** pane details:

- Imports: `<EXTERNAL>`, `libc_start_main`
- Exports: `main`
- Functions: `main`

**Data Type Manager** pane details:

- Data Types: `BuiltinTypes`, `demo`, `generic_clib_64`

**Filter**: `main`

```
1 undefined8 main(void)
2 {
3     long in_FS_OFFSET;
4     int local_24;
5     int local_20;
6     int local_1c;
7     void *local_18;
8     long local_10;
9
10    local_10 = *(long *)(in_FS_OFFSET + 0x28);
11    printf("Enter the number of students: ");
12    __isoc99_scanf(&DAT_00102027,&local_24);
13    local_18 = malloc((long)local_24 * 0x3C);
14    for (local_20 = 0; local_20 < local_24; local_20++)
15        printf("Enter student %d's ID: ",(ulong)(local_18 +
16        __isoc99_scanf(&DAT_0010205C,(long)local_18));
17    __isoc99_scanf(&DAT_0010205C,(long)local_18);
18    printf("Enter student %d's name: ",(ulong)(local_18 +
19        __isoc99_scanf(&DAT_0010205C,(long)local_18));
20    __isoc99_scanf(&DAT_0010205C,(long)local_18);
21    printf("Enter student %d's grade: ",(ulong)(local_18 +
22        __isoc99_scanf(&DAT_0010207A));
23    }
24    puts("Student information:");
25    for (local_1c = 0; local_1c < local_24; local_1c++)
26        printf((char *) (double *) (float *) ((long)local_18 +
27            "ID: %d, Name: %s, Grade: %.2f\n",
28            (ulong) * (uint *) ((long)local_18 + (long)local_18 +
29            (long)local_18 + (long)local_1c * 0x3C));
30    }
31    free(local_18);
32    if (local_10 != *(long *) (in_FS_OFFSET + 0x28))
33        /* WARNING: Subroutine does not return */
34        __stack_chk_fail();
35    }
36    return 0;
37 }
```

# Tools -- GDB

## ◆ 22e

```
0x000055555555200 <+3>:    mov    rax,rax
0x00005555555520e <+37>:   mov    eax,0x0
0x000055555555213 <+42>:   call   0x555555550d0 <printf@plt>
0x000055555555218 <+47>:   lea    rax,[rbp-0x1c]
0x00005555555521c <+51>:   mov    rsi,rax
0x00005555555521f <+54>:   lea    rax,[rip+0xe01]      # 0x555555556027
0x000055555555226 <+61>:   mov    rdi,rax
0x000055555555229 <+64>:   mov    eax,0x0
0x00005555555522e <+69>:   call   0x555555550f0 <__isoc99_scanf@plt>
0x000055555555233 <+74>:   mov    eax,DWORD PTR [rbp-0x1c]
0x000055555555236 <+77>:   movsxd rdx,eax
0x000055555555239 <+80>:   mov    rax,rdx
0x00005555555523c <+83>:   shl    rax,0x4
0x000055555555240 <+87>:   sub    rax,rdx
0x000055555555243 <+90>:   shl    rax,0x2
0x000055555555247 <+94>:   mov    rdi,rax
0x00005555555524a <+97>:   call   0x555555550e0 <malloc@plt>
0x00005555555524f <+102>:  mov    QWORD PTR [rbp-0x10],rax
0x000055555555253 <+106>:  mov    DWORD PTR [rbp-0x18],0x0
```

# Tools -- pwntools

## ❖ demo\_py.py

```
1 from pwn import *
2
3 io = process("./demo")
4
5 io.sendafter(b": ", b"1\n")
6
7 io.sendafter(b": ", b"2\n")
8 io.sendafter(b": ", b"3\n")
9 io.sendafter(b": ", b"4\n")
10
11 io.interactive()
12
```

# Tools -- pwntools

- ❖ `python3 demo_py.py`

```
~/rev_pwn python3 demo_py.py
[+] Starting local process './demo': pid 1741811
[*] Switching to interactive mode
[*] Process './demo' stopped with exit code 0 (pid 1741811)
Student information:
ID: 2, Name: 3, Grade: 4.00
[*] Got EOF while reading in interactive
$ █
```

# Lab

./reverse\_demo\_s

Author Sharkcode

Your goal is to find out the right key!

# Lab Solution

./reverse\_demo\_s

Author Sharkcode

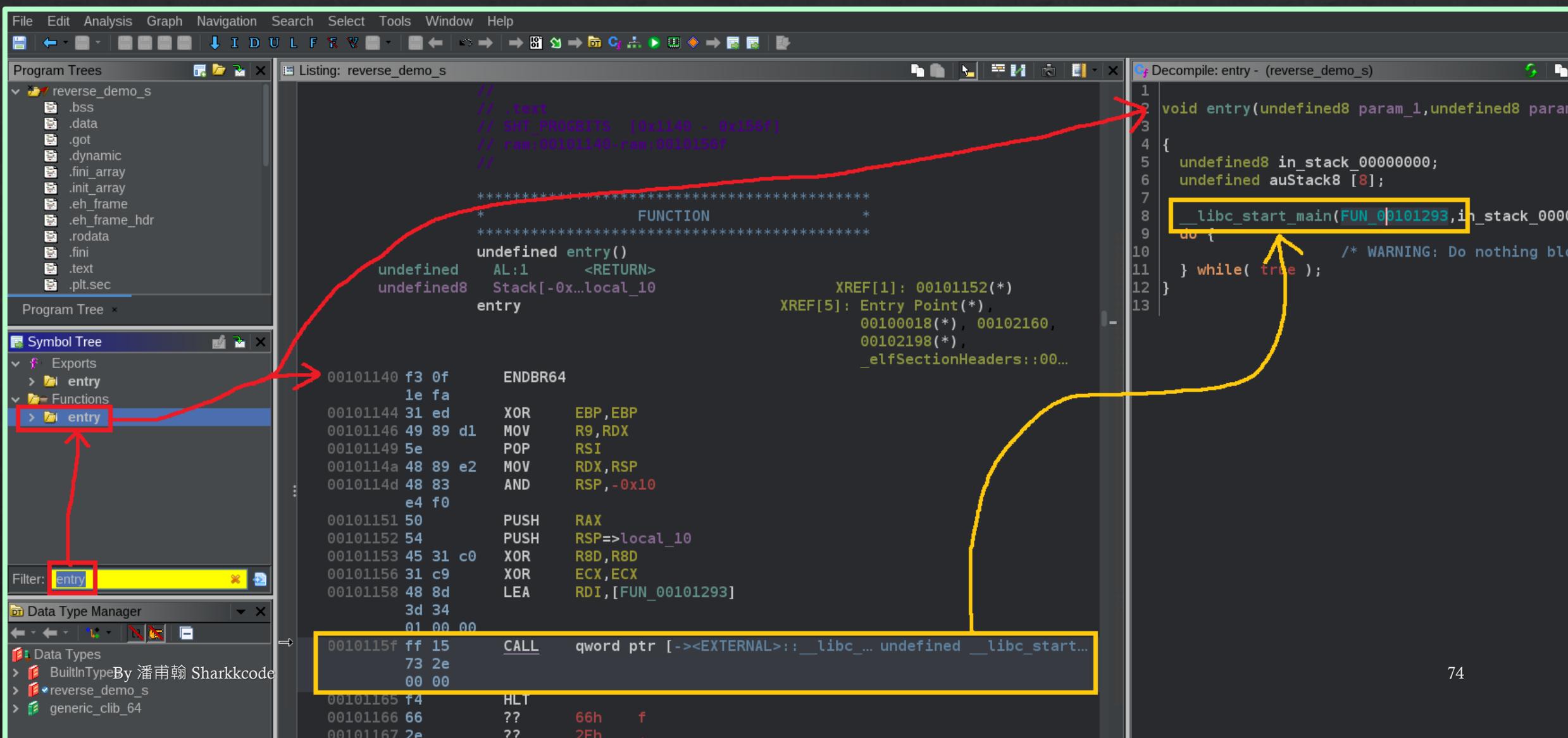
Your goal is to find out the right key!

```
./reverse_demo_s
```

```
~/reverse_ccuisc  
~/reverse_ccuisc ./reverse_demo_s  
Welcome to Super BIRD Software!!!  
You NEED to pay for an access token to open this VALUABLE software~~~  
aaa  
a  
a  
a  
a  
a  
  
aaaa  
a  
aaaaaa  
aaaa  
  
aaa  
Enter Access Key: Nahhh! Wrong Key...  
Please pay in order to get the right key!!!  
Enter Access Key: Nahhh! Wrong Key...  
Please pay in order to get the right key!!!
```

What is going on?

## Find main → FUN\_00101293



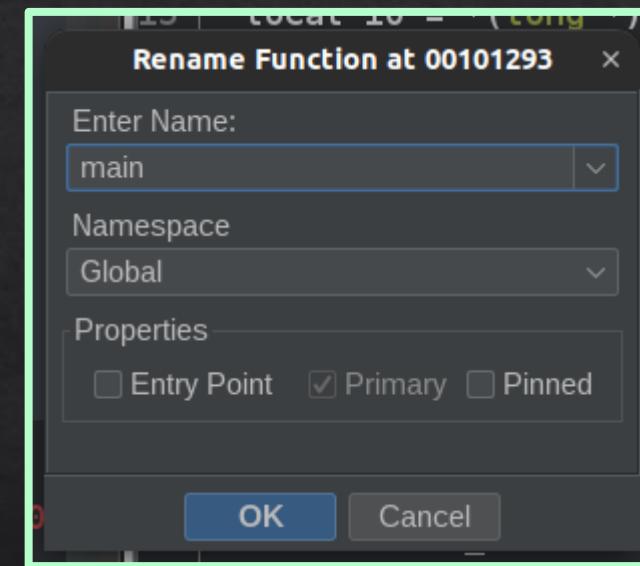
Rename `FUN_00101293` to `main`.

Decompile: `FUN_00101293 - (reverse_demo_s)`

```
1
2 undefined8 FUN_00101293()
3 {
4     int iVar1;
5     char *pcVar2;
6     char *_s2;
7     long in_FS_OFFSET;
8     uint local_44;
9     uint local_40;
10    int local_3c;
11    char local_28 [24];
12    long local_10;
13
14    local_10 = *(long *)
15    puts("Welcome to S");
16    puts("You NEED to pay for an access token to op");
17
18    local_44 = 0;
```

Context menu open over the function name `FUN_00101293`:

- Edit Function Signature
- Rename Function**
- Commit Params/Return
- Commit Local Names
- Secondary Highlight
- Copy
- Comments
- Find...
- References
- Properties



Sleep 15s...

The screenshot shows a debugger interface with two panes. The left pane displays assembly code, and the right pane displays the corresponding C code.

**Assembly Code:**

```
c0 13  
001014b5 c6 00 7d    MOV     byte ptr [RAX],0x7d  
LAB 001014b8  
001014b8 bf 0f        MOV     EDI,0xf  
00 00 00  
001014bd e8 6e        CALL    <EXTERNAL>::sleep  
fc ff ff  
001014c2 48 8d        LEA     RAX,[s_Enter_Access_Key:_00  
05 c5  
0b 00 00  
001014c9 48 89 c7        MOV     RDI=>s_Enter_Access_Key:_00  
001014cc b8 00        MOV     EAX,0x0
```

**C Code:**

```
52  
53     s2[8] = 'T';  
54     s2[7] = 'O';  
55     s2[6] = 'G';  
56     s2[5] = 'U';  
57     s2[0x13] = '}';  
58     while( true ) {  
59         sleep(0xf);  
60         printf("Enter Access Key: ");  
61         _isoc99_scanf(&DAT_001020a1,lo  
62         iVar1 = strncmp(local_28,__s2,0  
63         if (iVar1 == 0) break;  
64         puts("Nahhh! Wrong Key...");  
65         puts("Please pay in order to ge
```

The assembly code block from address 001014bd to 001014cc is highlighted with a red rectangle. The C code block from line 59 to 65 is also highlighted with a red rectangle.

## Replace sleep

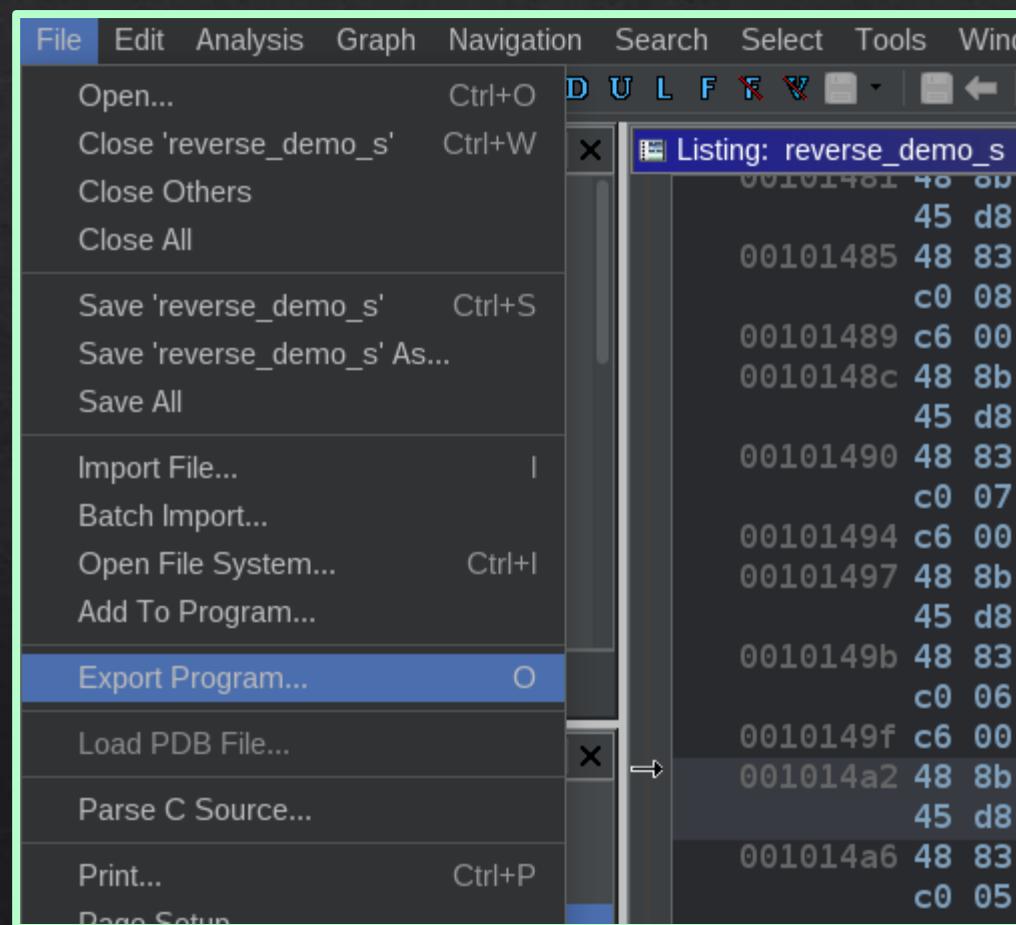
The screenshot shows a debugger interface displaying assembly code. The instruction at address 001014bd is highlighted in blue. A context menu is open over this instruction, listing various options such as MOV, byte ptr [RAX], 0x47; MOV RAX, word ptr [RBP + loc], and several clearing and modification options. The 'Patch Instruction' option is selected, highlighted with a blue background.

Address	OpCode	Hex Value	Description
0010149f	c6 00 47		MOV byte ptr [RAX], 0x47
001014a2	48 8b		MOV RAX, word ptr [RBP + loc]
	45 d8		
001014a6	48 83		
	c0 05		
001014aa	c6 00 55		
001014ad	48 8b		
	45 d8		
001014b1	48 83		
	c0 13		
001014b5	c6 00 7d		
	LA		
001014b8	bf 0f		
	00 00 00		
001014bd	e8 6e		
	fc ff ff		
001014c2	48 8d		
	05 c5		
	0b 00 00		
001014c9	48 89 c7		
001014cc	b8 00		
	00 00 00		
001014d1	e8 2a		
	fc ff ff		
001014d6	48 8d		
	45 e0		
001014da	48 89 c6		
001014dd	48 8d		

## Replace sleep

```
001014b1 48 83      ADD    RAX,0x13  
:          c0 13  
001014b5 c6 00 7d    MOV    byte ptr [RAX],0x7d  
  
LAB_001014b8  
001014b8 bf 0f      MOV    EDI,0xf  
:          00 00 00  
001014bd 90          NOP  
001014be 90          NOP  
001014bf 90          NOP  
001014c0 90          NOP  
001014c1 90          NOP  
001014c2 48 8d      LEA    RAX,[s_Enter_Access]  
:          05 c5  
:          0b 00 00  
001014c9 48 89 c7    MOV    RDI=>s_Enter_Access  
001014cc b8 00      MOV    EAX,0x0
```

## Replace sleep



## Replace sleep

```
~/reverse_ccuisC  
~/reverse_ccuisC ./reverse_demo_s_nop  
Welcome to Super BIRD Software!!!  
You NEED to pay for an access token to open this VALUABLE software~~~  
Enter Access Key: aaa  
Nahhh! Wrong Key...  
Please pay in order to get the right key!!!  
Enter Access Key: █
```

What happens when we enter the right key?

```
58 | while( true ) {
59 |     sleep(0xf);
60 |     printf("Enter Access Key: ");
61 |     __isoc99_scanf(&DAT_001020a1,local_28);
62 |     iVar1 = strncmp(local_28,__s2,0x14);
63 |     if (iVar1 == 0) break;
64 |     puts("Nahhh! Wrong Key...");  
65 |     puts("Please pay in order to get the right key!!!");  
66 |
67 |     system("curl parrot.live");
68 |     printf("That is our secret parrot for that will make your wish come true!!!!");
69 |     if (local_10 != *(long *)(in_FS_OFFSET + 0x28)) {
70 |         /* WARNING: Subroutine does not return */
71 |         __stack_chk_fail();
72 |     }
73 |     return 0;
74 |
75 }
```

## Simple Comparison

```
58 |     while( true ) {
59 |         sleep(0xf);
60 |         printf("Enter Access Key: ");
61 |         __isoc99_scanf(&DAT_001020a1,local_28);
62 |         iVar1 = strncmp(local_28,__s2,0x14);
63 |         if (iVar1 == 0) break;
64 |         puts("Nahhh! Wrong Key...");  
65 |         puts("Please pay in order to get the right key!!!");  
66 |     }  
67 |     system("curl parrot.live");  
68 |     printf("That is our secret parrot for that will make your wish come true!!!");  
69 |     if (local_10 != *(long *)(in_FS_OFFSET + 0x28)) {  
70 |         /* WARNING: Subroutine does not return */  
71 |         __stack_chk_fail();  
72 |     }  
73 |     return 0;  
74 | }  
75 |
```

local\_28 is user input.

```
__s2[0x12] = *pcVar2;
__s2[0x11] = pcVar2[1];
__s2[0x10] = pcVar2[2];
__s2[0xf] = pcVar2[3];
__s2[0xe] = '!';
__s2[0xd] = 'D';
__s2[0xc] = 'R';
__s2[0xb] = 'I';
__s2[10] = 'B';
__s2[4] = '{';
__s2[3] = 'G';
__s2[2] = 'A';
__s2[1] = 'L';
*__s2 = 'F';
__s2[9] = 'A';
__s2[8] = 'T';
__s2[7] = 'O';
__s2[6] = 'G';
__s2[5] = 'U';
__s2[0x13] = '}';
```

```
__s2 = "FLAG{UGOTABIRD!" + pcVar2[3] + pcVar2[2] + pcVar2[1] +
pcVar2[0] + "}"
```

local\_44 → pcVar2

Line 18 ~ Line 31

```
18 local_44 = 0;
19 for (local_40 = 0; (int)local_40 < 0x14; local_40 = local_40 + 1) {
20     if ((int)local_40 % 3 == 0) {
21         local_44 = local_44 + 1;
22     }
23     else {
24         if ((int)local_40 % 3 == 1) {
25             local_44 = local_44 ^ local_40;
26         }
27         else {
28             local_44 = (int)(local_44 * local_40 + -1) / 3;
29         }
30     }
31 }
32 malloc(0x50);
33 pcVar2 = (char *)FUN_00101229(local_44,0x10);
```

Calculate local\_44

```
C exp.c
1 #include <stdio.h>
2
3 int main(int argc, char **argv) {
4
5     int local_44 = 0;
6     for(int local_40 = 0; local_40 < 0x14; local_40++) {
7         if(local_40 % 3 == 0) {
8             local_44++;
9         } else {
10            if(local_40 % 3 == 1) {
11                local_44 ^= local_40;
12            } else {
13                local_44 = (local_44 * local_40 - 1) / 3;
14            }
15        }
16    }
17
18    printf("local_44: %d\n", local_44);
19
20    return 0;
21 }
22
23
```

local\_44 → pcVar2

```
~/reverse_ccuisC  
~/reverse_ccuisC ./exp  
local_44: 4520  
~/reverse_ccuisC █
```

(char \*)FUN\_00101129(local\_44, 0x10) → pcVar2

Line 33

```
18 local_44 = 0;
19 for (local_40 = 0; (int)local_40 < 0x14; local_40 = local_40 + 1) {
20     if ((int)local_40 % 3 == 0) {
21         local_44 = local_44 + 1;
22     }
23     else {
24         if ((int)local_40 % 3 == 1) {
25             local_44 = local_44 ^ local_40;
26         }
27         else {
28             local_44 = (int)(local_44 * local_40 + -1) / 3;
29         }
30     }
31 }
32 malloc(0x50);
33 pcVar2 = (char *)FUN_00101229(local_44,0x10);
```

(char \*)FUN\_00101129(local\_44, 0x10) → pcVar2

The screenshot shows the Immunity Debugger interface with the title bar "Cf Decompile: FUN\_00101129 - (reverse\_demo\_s)". The main window displays the following C-like pseudocode:

```
1
2 undefined * FUN_00101129(int param_1,int param_2)
3
4 {
5     int local_1c;
6     int local_c;
7
8     local_1c = param_1;
9     for (local_c = 0x1e; (local_1c != 0 && (local_c != 0)); local_c = local_c + -1) {
10        (&DAT_00104040)[local_c] = "0123456789abcdef"[local_1c % param_2];
11        local_1c = local_1c / param_2;
12    }
13    return &DAT_00104040 + (local_c + 1);
14}
15
```

The code defines a function FUN\_00101129 that takes two integer parameters. It initializes local variables local\_1c and local\_c. It then enters a loop where it iterates from 0x1e down to 0. In each iteration, it updates a memory location at address &DAT\_00104040 + local\_c with a character from the string "0123456789abcdef" based on the remainder of local\_1c divided by param\_2. Finally, it returns the address of the DAT variable plus the current value of local\_c.

# Guess???

0x10 = 16  
0123456789abcdef  Hexadecimal?

```
>>>  
>>> print(hex(4520))  
0x11a8  
>>> █
```

Try...

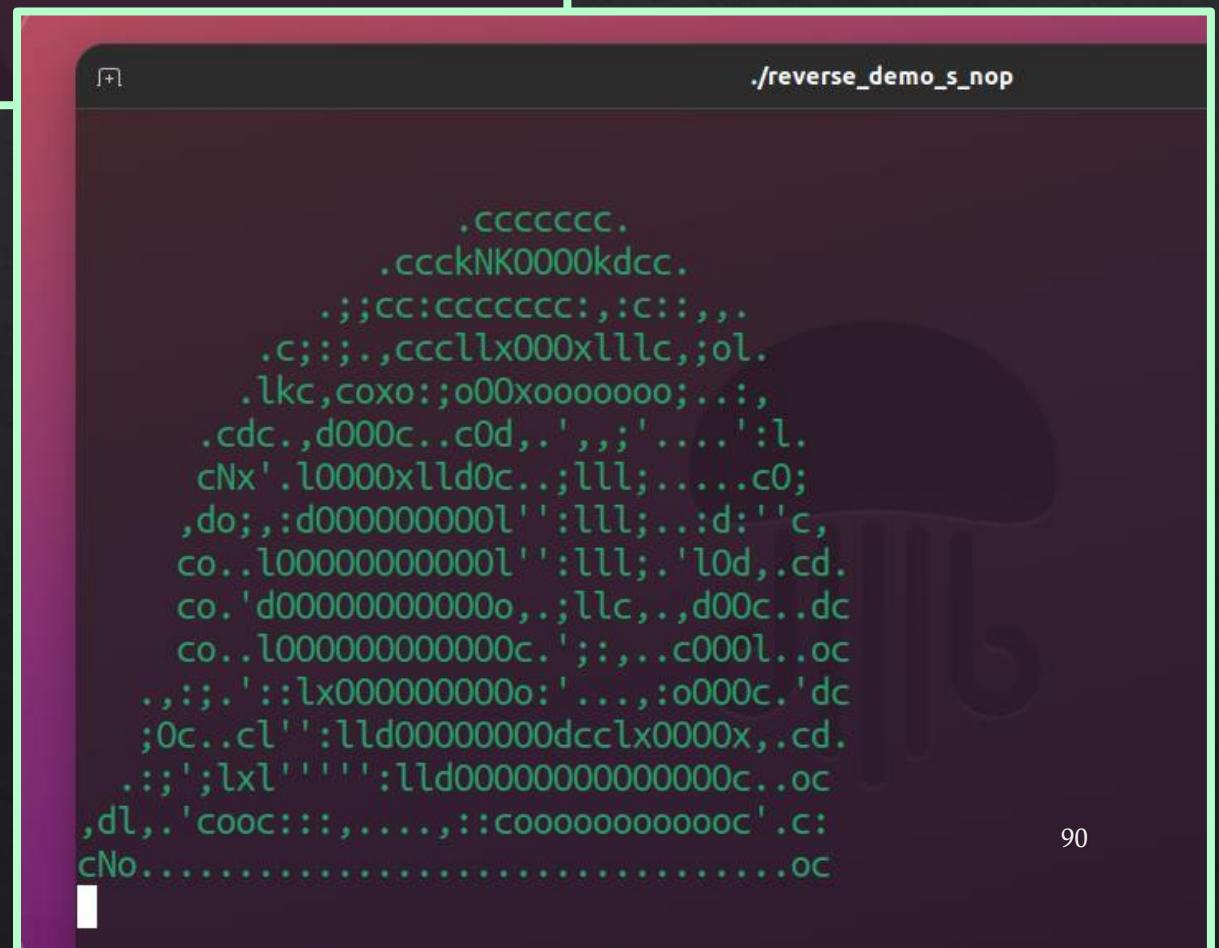
```
pcVar2 = "11a8"  
__s2 = "FLAG{UGOTABIRD!" + pcVar2[3] + pcVar2[2] + pcVar2[1] + pcVar2[0] + "}"
```



```
__s2 = "FLAG{UGOTABIRD!8a11}"
```

# BIRD!!!

```
~/reverse_ccuisC  
~/reverse_ccuisC ./reverse_demo_s_nop  
Welcome to Super BIRD Software!!!  
You NEED to pay for an access token to open this VALUABLE software~~~  
Enter Access Key: FLAG{UGOTABIRD!8a11}
```



```
.ccccccc.  
. , , ;cooolccoo; , , .  
.d0x; .. ;lllll; .. ;x0d.  
.cdo; ',lo0XXXXXkll; ' ;odc.  
,ol:;c, ' :oko:cccccc, ... ckl.  
;c.;kXo..:::;c::'.....oc  
,dc..oXX0kk0o.' :lll; .. cxxc.,ld,  
kNo.'oXXXXXXo', :lll; .. oXX0o;c0d.  
K0c;o0XXXXXXo.' :lol; .. dXXXXl'; xc  
0l,:k0XXXXXX0c.,clc': :0XXXXx,.oc  
K0c;d0XXXXXXl..';' .. lXXXXXo..oc  
dNo..oXXXXXXXo:... 'lx0XXXXk,..; ..  
cNo..lXXXXXXXXX0ol kXXXXXXXXXk l,..;:'.  
. , ' ..dkkkkk0XXXXXXXXXXXXX0xxl;,; ;l:.  
;c.;: ''': do0XXXXXXXXXXXXXXXXXXXX0do; ' ;clc.  
;c.l0dood:''' oXXXXXXXXXXXXXXXXXXXXk,..;ol.  
' ; ..:xxxxxocccxxxxxXXXXXXXXXXXXl:'. ';;.  
' ; ..:.....; ..:; l'
```

^CThat is our secret parrot for that will make your wish come true!!!%  
~/reverse\_ccuisC

# Lab

./tea

From SHELL CTF 2022

Your goal is to find out the FLAG!

# Lab Solution

./tea

From SHELL CTF 2022

Your goal is to find out the FLAG!

# tea

- ❖ main
  - ❖ len == 20
  - ❖ Some functions.

```
1  undefined8 main(void)
2  {
3      size_t sVar1;
4
5      boilWater();
6      sVar1 = strlen(pwd);
7      if (sVar1 == 0x20) {
8          addSugar();
9          addTea();
10         addMilk();
11         strainAndServe();
12     }
13    else {
14        puts("wrong length");
15    }
16    return 0;
17 }
```

# tea

- ❖ boilWater
  - ❖ Enter flag.

```
1 void boilWater(void)
2 {
3     printf("Enter the flag: ");
4     __isoc99_scanf(&DAT_00102019, pwd);
5     return;
6 }
```

# tea

- ❖ strainAndServe
  - ❖ Compare flag.

```
1 void strainAndServe(void)
2 {
3     int iVar1;
4
5     iVar1 = strcmp("R;crc75ihl`cNYe` ]m%50gYhugow~34i",pwd);
6     if (iVar1 == 0) {
7         puts("yep, that's right");
8     }
9     else {
10        puts("nope, that's not it");
11    }
12    return;
13 }
```

# tea

- ❖ How the flag is modified?

```
flag ---- [addSugar] ---> flag2
```

```
flag2 --- [addTea] -----> flag3
```

```
flag3 --- [addMilk] ----> flag4
```

```
flag4 is R;crc75ihl`cNYe` ]m%50gYhugow~34i
```

# tea

- ❖ How the flag is modified? (more detailed)

```
part1: String with no '5'.
```

```
part2: String starts with '5' and with no 'R'
```

```
part3: String starts with 'R'
```

```
flag4 = ( part3 + part1 + part2 )
```

# tea

- ❖ How to generate the initial flag?

Notice that **flag4** is just simply move part3 to the front. The length of part2 is knowned. If we know the length of part3, we can calculate the length of part1 immediately. **So we can brute force part3's length.**

Why the length of part2 is knowned? Because  $(\text{part1} + \text{part2})$  is not modified. If we found a '5' in  $(\text{part1} + \text{part2})$ , that '5' must be the start of part2. A the end of part2 is the end of **flag4** , obviously.

# THANKS

## Q&A