

冰凌天 Lv2

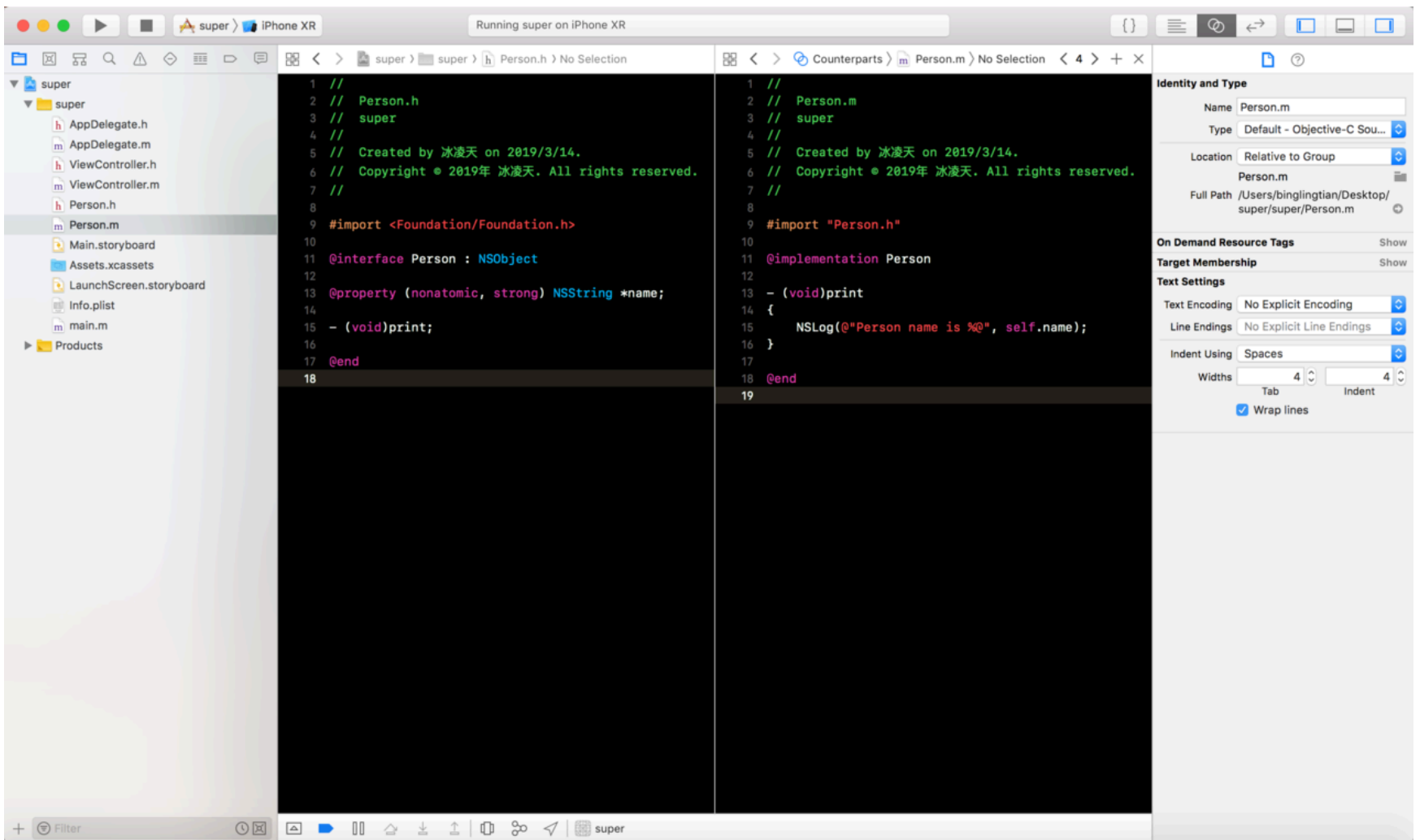
2019年03月14日 阅读 71

[关注](#)

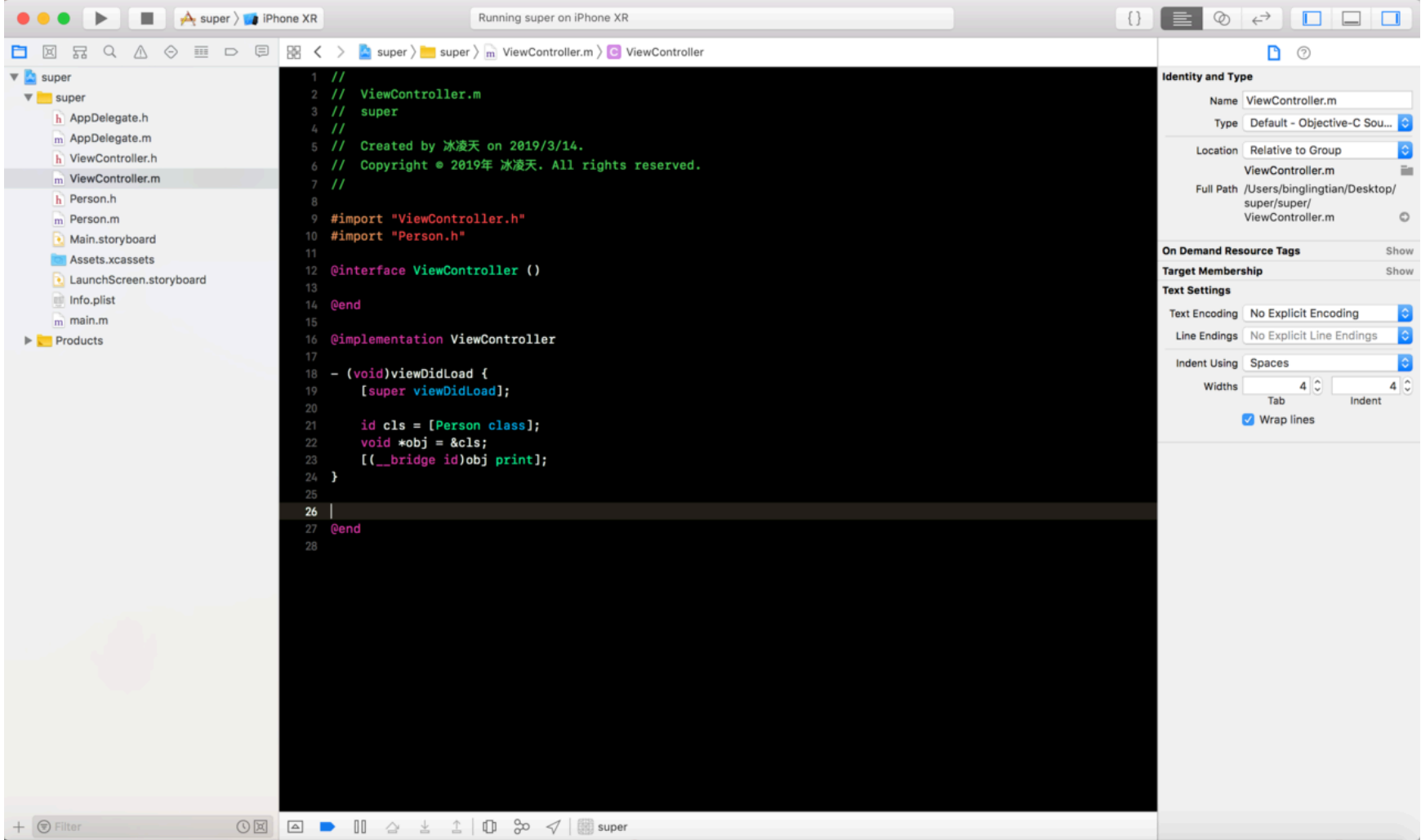
小码哥iOS学习笔记第十五天: super面试题

一、面试题

- 创建 OC 项目, 定义 `Person` 类, 添加 `name` 属性和实现 `print` 方法, 如下图所示

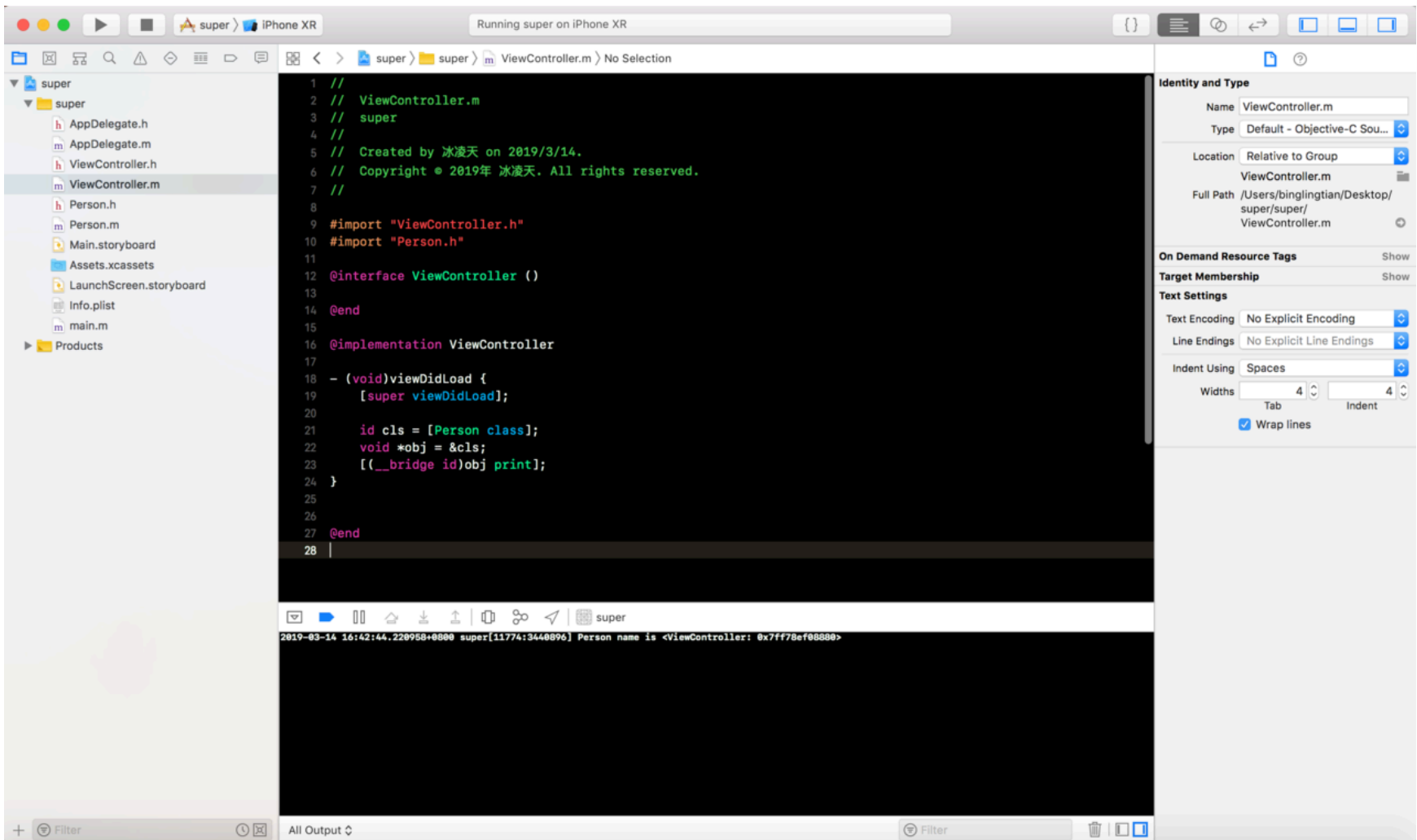


- 在 `ViewController` 中的 `-viewDidLoad` 方法里实现下面的代码



问: 代码是否会报错? 是否能够顺利运行? 是否能打印?, 如果能, 会打印什么?

- 运行程序, 可以看到程序没有报错, 且正常运行, 并有如下打印

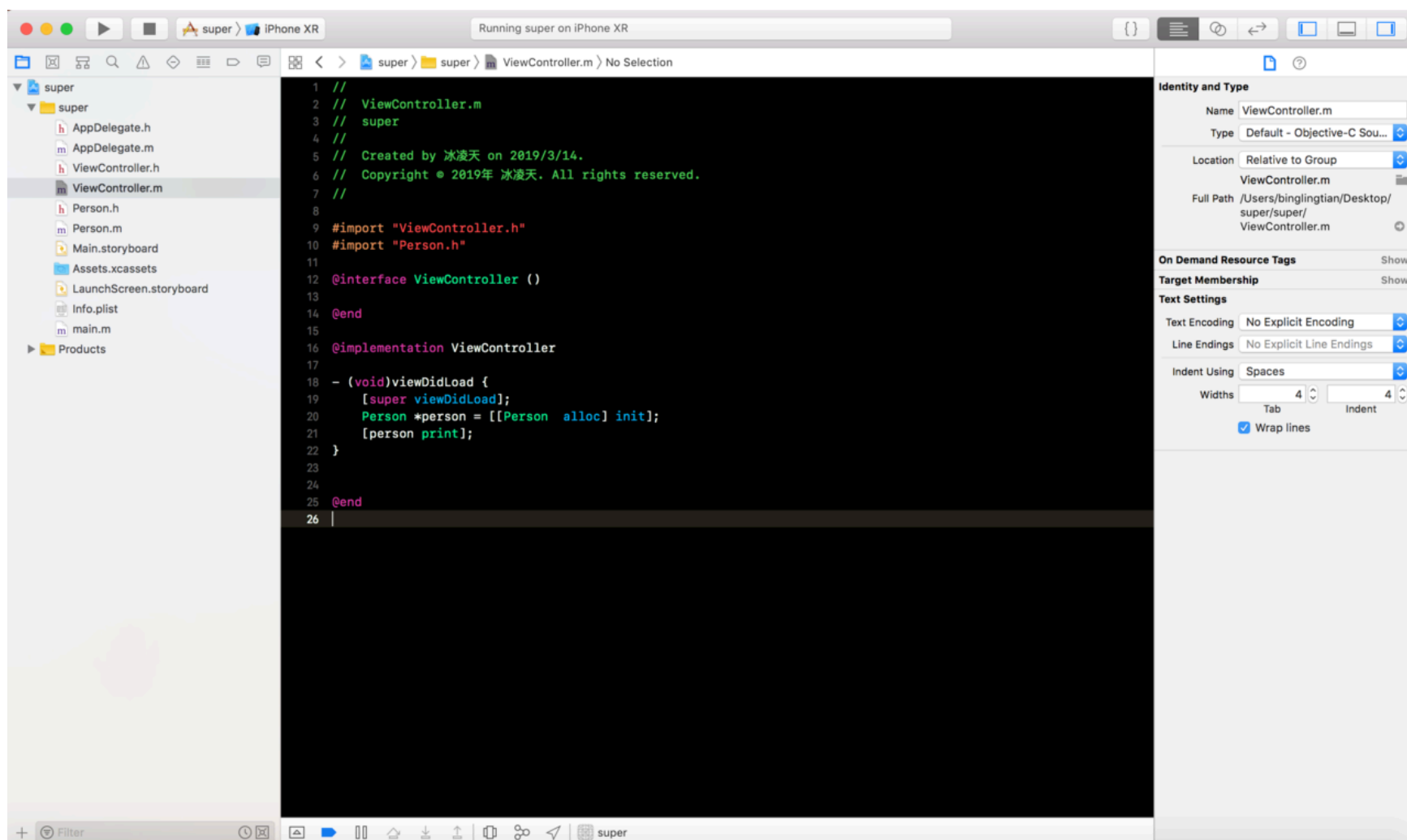


Person name is <ViewController: 0x7ff78ef08880>

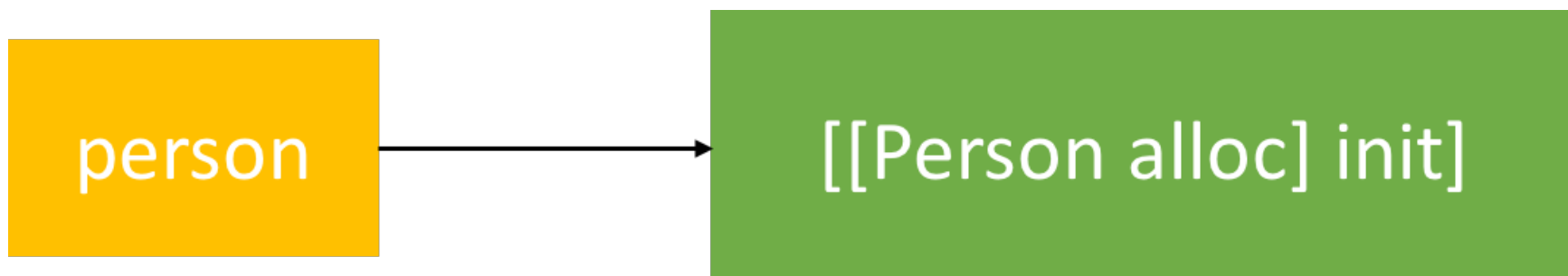
问: 为什么这么打印?

二、解析

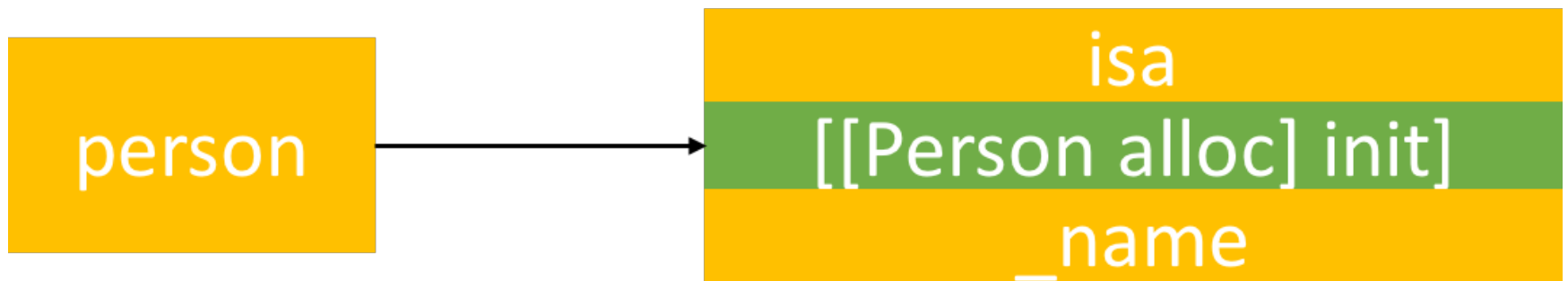
- 现有如下代码



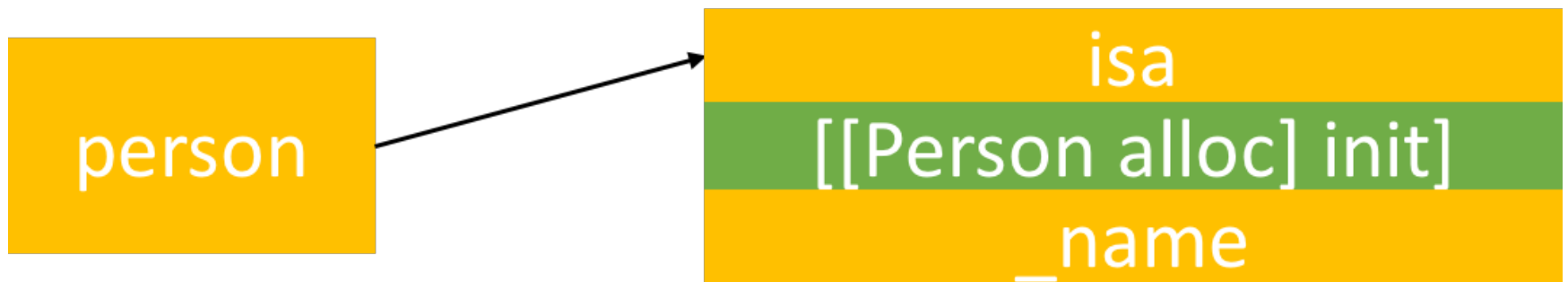
- `person` 是一个指针, 存储着 `[[Person alloc] init]` 的地址, 所以 `person` 指向刚创建的 `Person` 实例对象



- 我们知道, 一个对象在底层就是一个结构体, 它的第一个成员变量是 `isa`, 所以 `[[Person alloc] init]` 在底层就是下面的样子



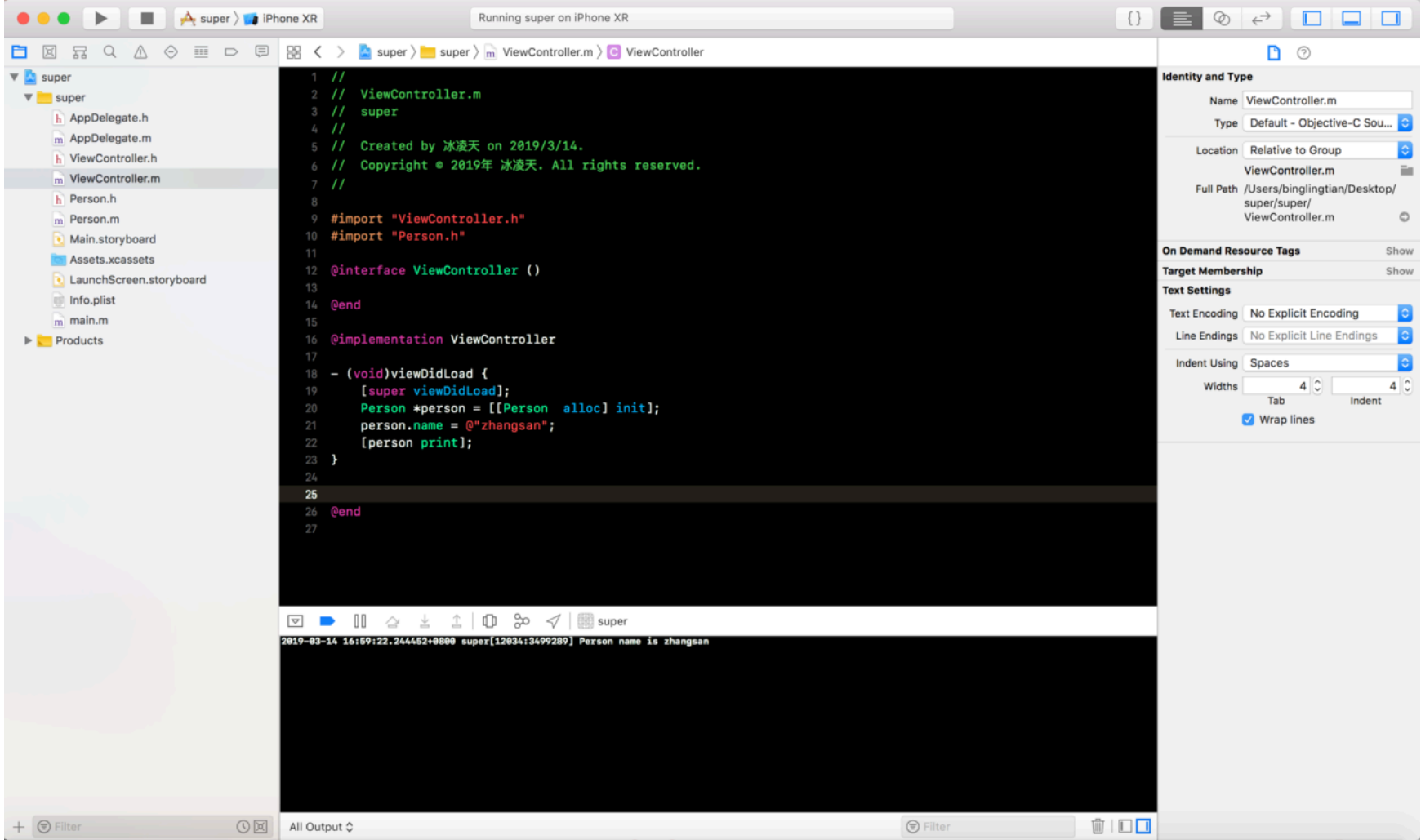
- 因为 `isa` 是 `Person` 对象中的第一个成员变量, 所以 `isa` 的地址与 `[[Person alloc] init]` 的地址相同



- 当 `person` 调用 `-print` 方法时, 本质是通过 `isa` 找到 `[Person class]` 对象, 然后找到 `-print` 方法来调用, 所以指针关系图如下



- 在 `-print` 方法中, 有打印成员变量 `_name`, 所以需要通过 `[[Person alloc] init]` 找到 `_name` 存储的值
- 在底层是通过 `isa` 的地址 + 8 找到 `_name` 的地址, 然后访问 `_name` 存储的内容
- 这样就可以顺利打印了



obj 能调用 print 方法的原因

- 接着我们看回面试题中的代码

```
id cls = [Person class];
```

- 这一句代码表示 cls 指针指向 Person 的类对象



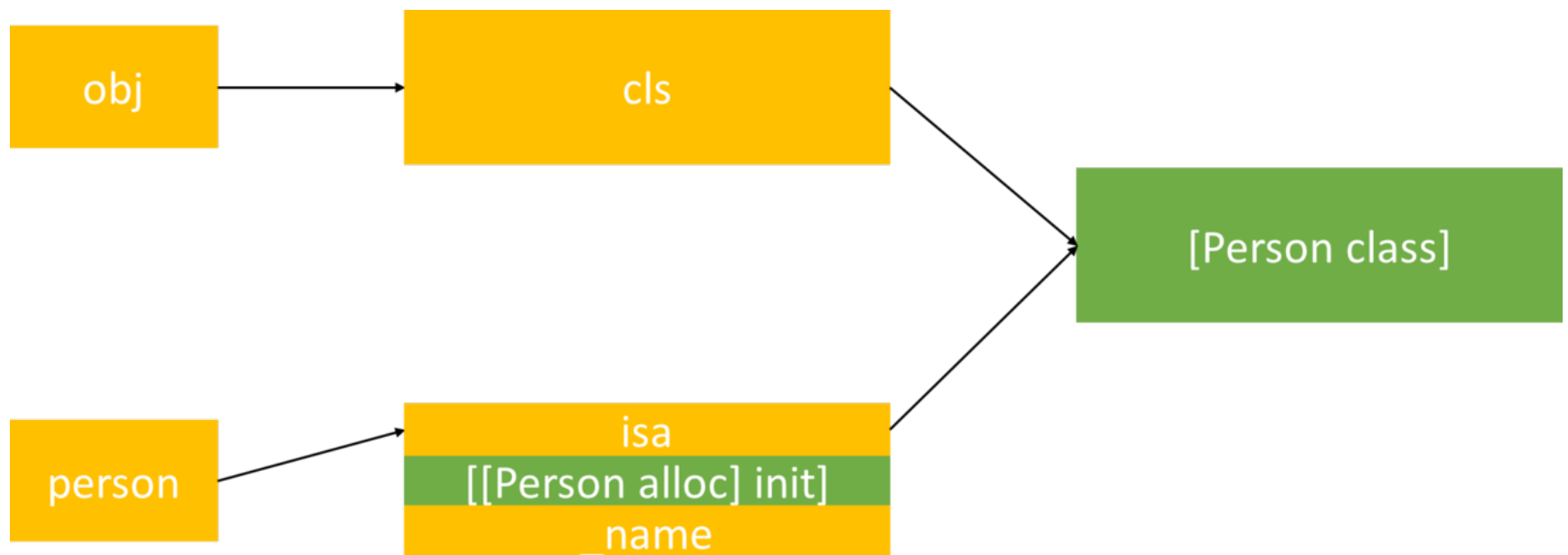
- 接着下一句，obj 存储着 cls 的地址

```
void *obj = &cls;
```

- 也就是说 obj 指向 cls



- 这个指向和上面的 `person` 指向类似, 有下面这种指针结构



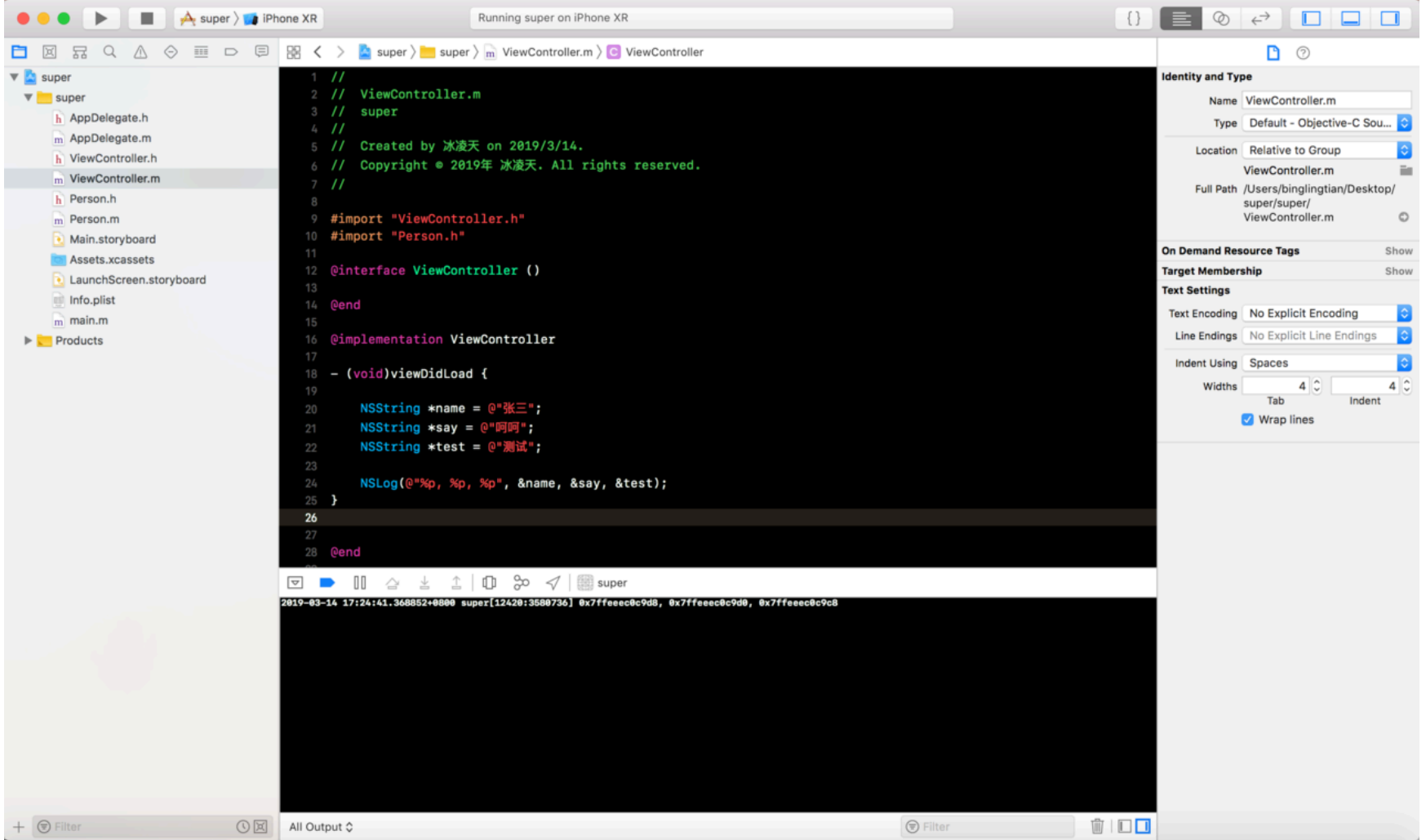
- 最后看调用方法这一句

```
[(__bridge id)obj print];
```

- 在结构上
 - `person` 调用方法的流程是: `person->isa->[Person class]`
 - `obj` 调用的流程与 `person` 类似. `obj->cls->[Person class]`
- 这就是 `obj` 能够调用 `-print` 方法的原因

局部变量在栈中的排列顺序

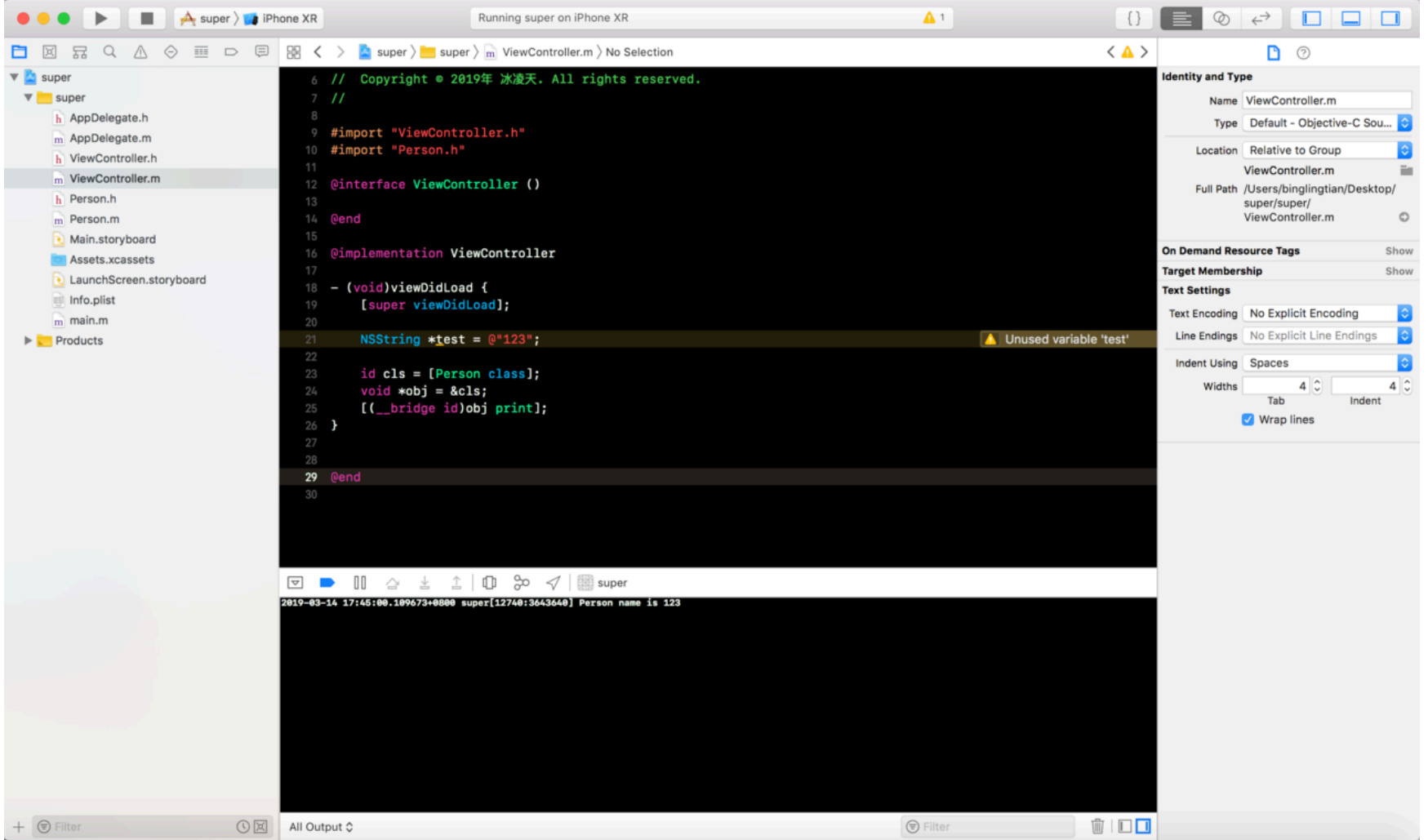
- 定义三个局部变量, 查看他们的地址



- 三个局部变量的地址

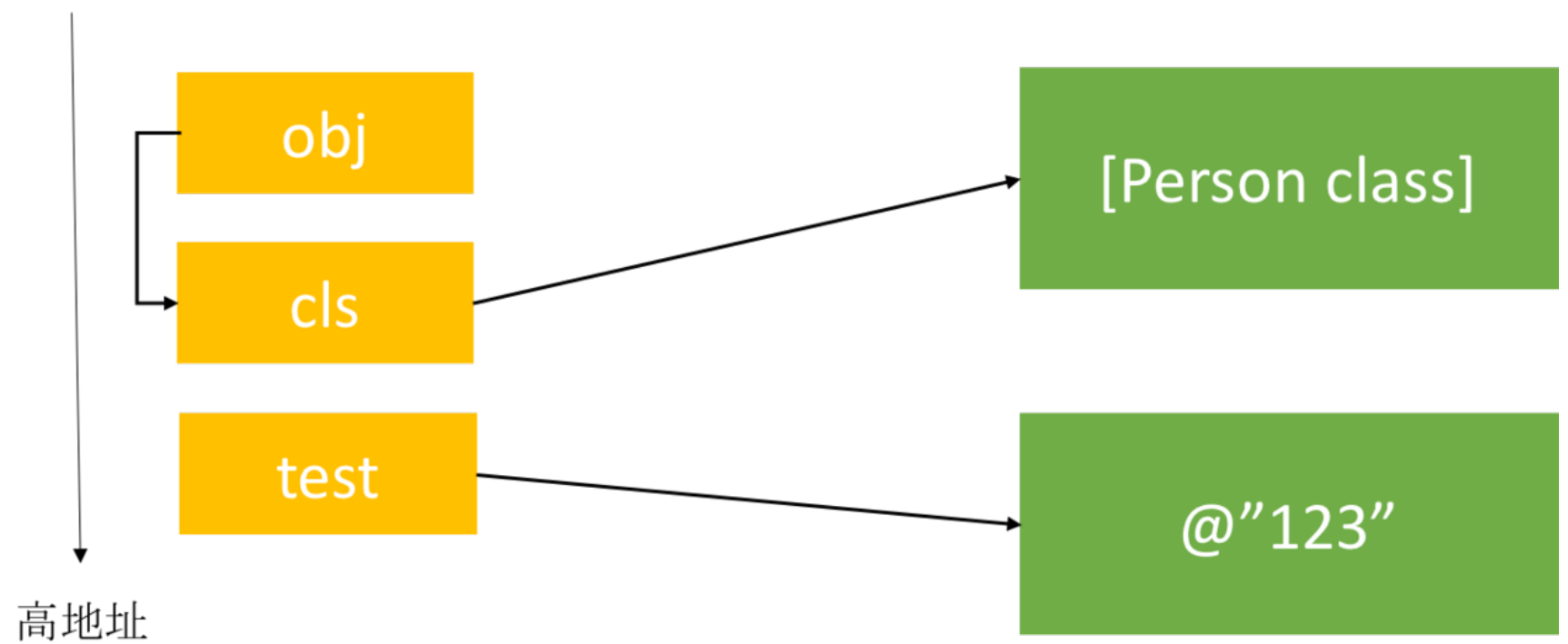
0x7ffeeec0c9d8
0x7ffeeec0c9d0
0x7ffeeec0c9c8

- 可以看到, 连续的三个局部变量, 在栈中的存储顺序是连续的
- 接着我们看下面的这种情况, 在 `cls` 前面添加 `test` 变量, 运行看一下打印



- 此时在内存中的排序如下

低地址

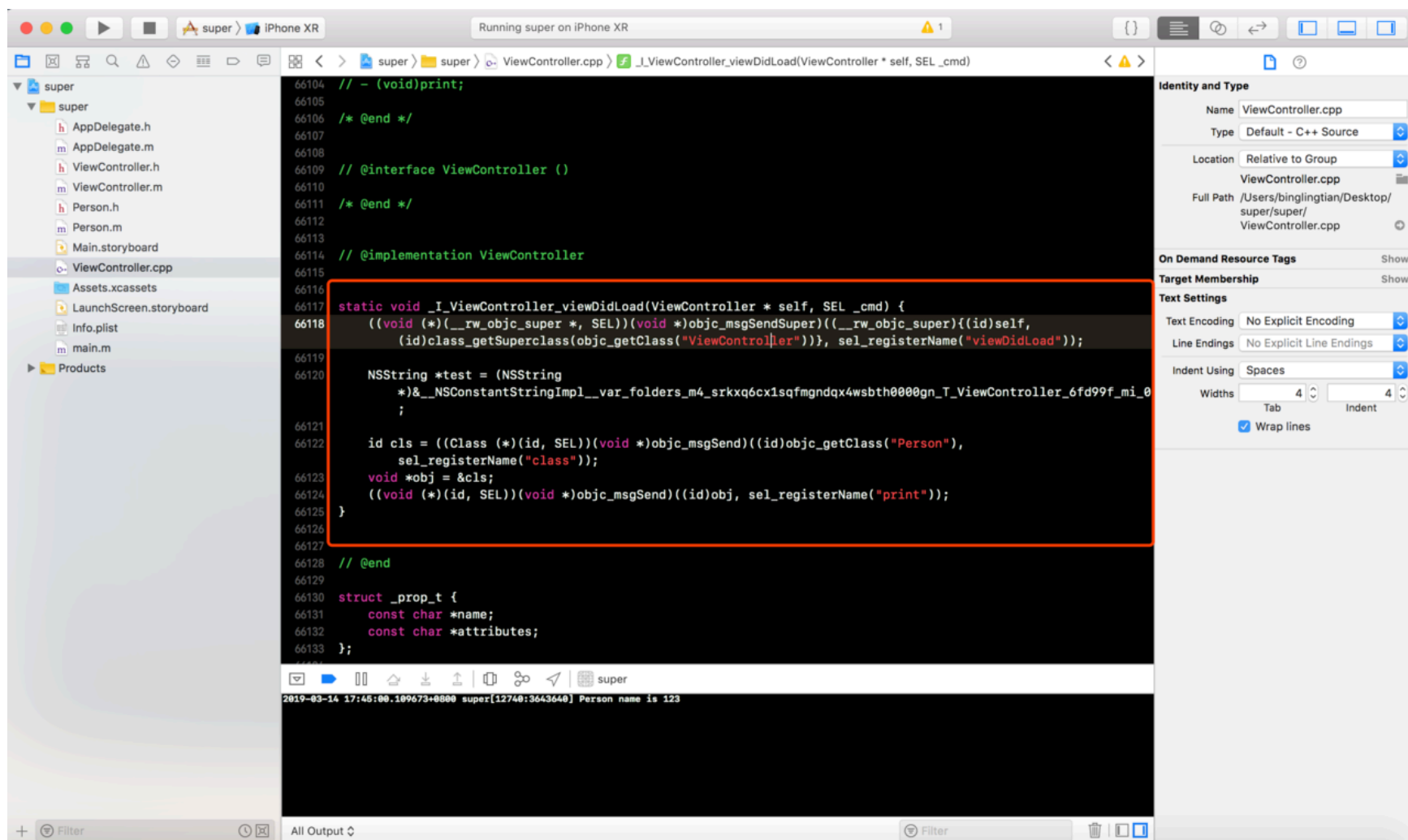


- `obj` 通过找到 `cls` , 然后用 `cls`的地址+8 , 找到的就是 `test` 的地址, 所以打印时, 找到的 `_name` 就是 `test` 的值 123

那么面试题中的打印为什么是 `ViewController` 的实例对象呢?

super的底层结构

- 查看 `ViewController.cpp` 底层结构



- 其中 `[super viewDidLoad];` 这一句在底层代码是下面这一句

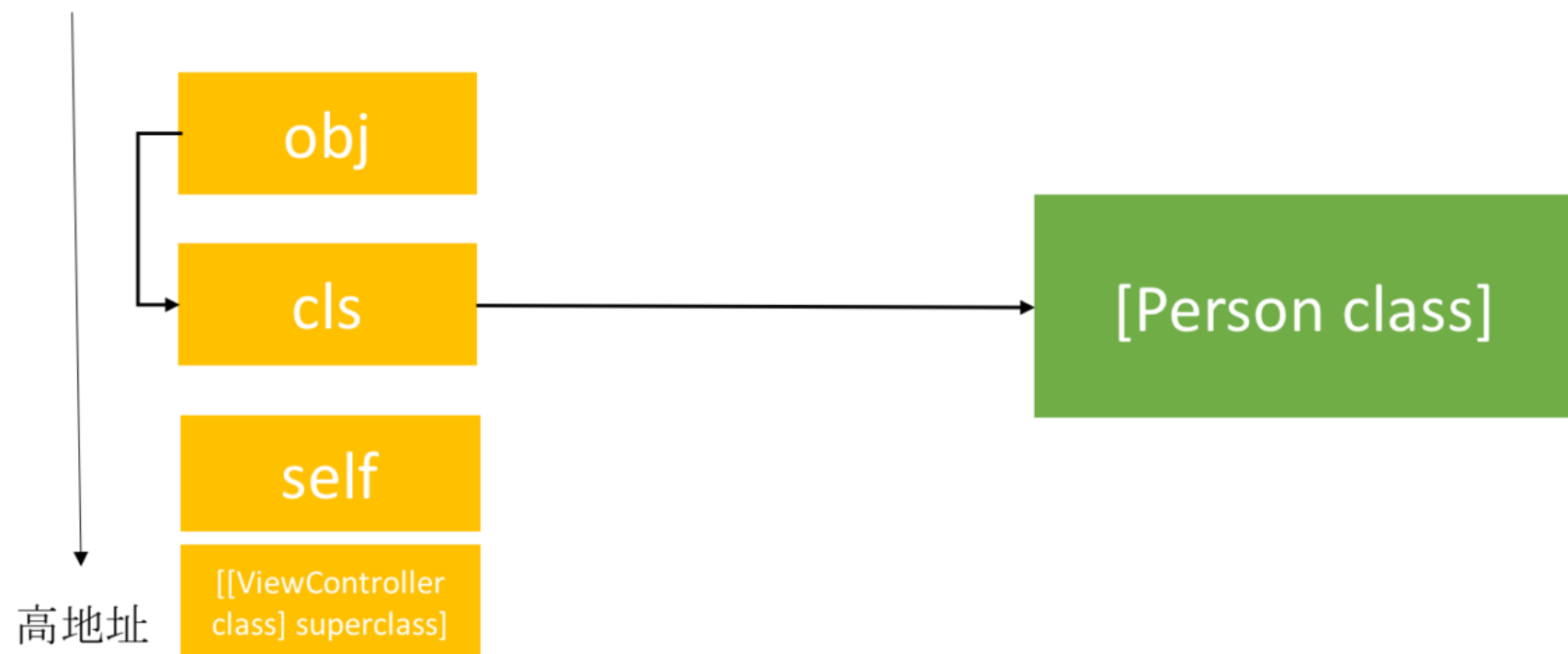
```
((void *) (__rw_objc_super *, SEL))(void *)objc_msgSendSuper((__rw_objc_super){(id)self
```

- 整理后, 代码如下

```
objc_msgSendSuper({self, class_getSuperclass(objc_getClass("ViewController"))}, sel_regi
```

- 也就是说, 在栈中有个结构体 `{self, class_getSuperclass(objc_getClass("ViewController"))}`
- 此时, 栈中的结构如下图所示

低地址



- 所以此时通过 `cls`的地址+8，找到的就是 `self`，也就是 当前控制器对象

关注下面的标签，发现更多相似文章

iOS



冰凌天

Lv2

iOS

获得点赞 189 · 获得阅读 9,940

关注

安装掘金浏览器插件

打开新标签页发现好内容，掘金、GitHub、Dribbble、ProductHunt 等站点内容轻松获取。快来安装掘金浏览器插件获取高质量内容吧！

评论

输入评论...

相关推荐

专栏知识小集 · 1天前 · iOS

值得收藏的 5 个 iOS 库

 44

 4

专栏_森宇_ · 4天前 · iOS

iOS 13 适配要点总结

 72

 16

专栏知识小集 · 7天前 · iOS

Xcode 11 的那些新东西

 50



荐 · 知识小集 · 10天前 · iOS

专栏

iOS 13 正式发布，来看看有哪些 API 变动

 68

 7

专栏by在水一方 · 3天前 · iOS

iOS 端 h5 页面秒开优化实践

 17



专栏GSC · 1天前 · iOS


iOS开发：集成的SDK不支持模拟器调试怎么办？


 3

 4

专栏iOS桃子 · 2天前 · iOS

iOS | 面试知识整理 - 数据持久化(八)

 2



专栏宋卓 · 21小时前 · iOS

编程范式 --- 函数式编程（Funtional Programming，简称FP）

 1

 1

专栏QiShare · 6天前 · iOS

iOS13 DarkMode适配（一）

 25

 16

iOS -- 问题杂记

 16



