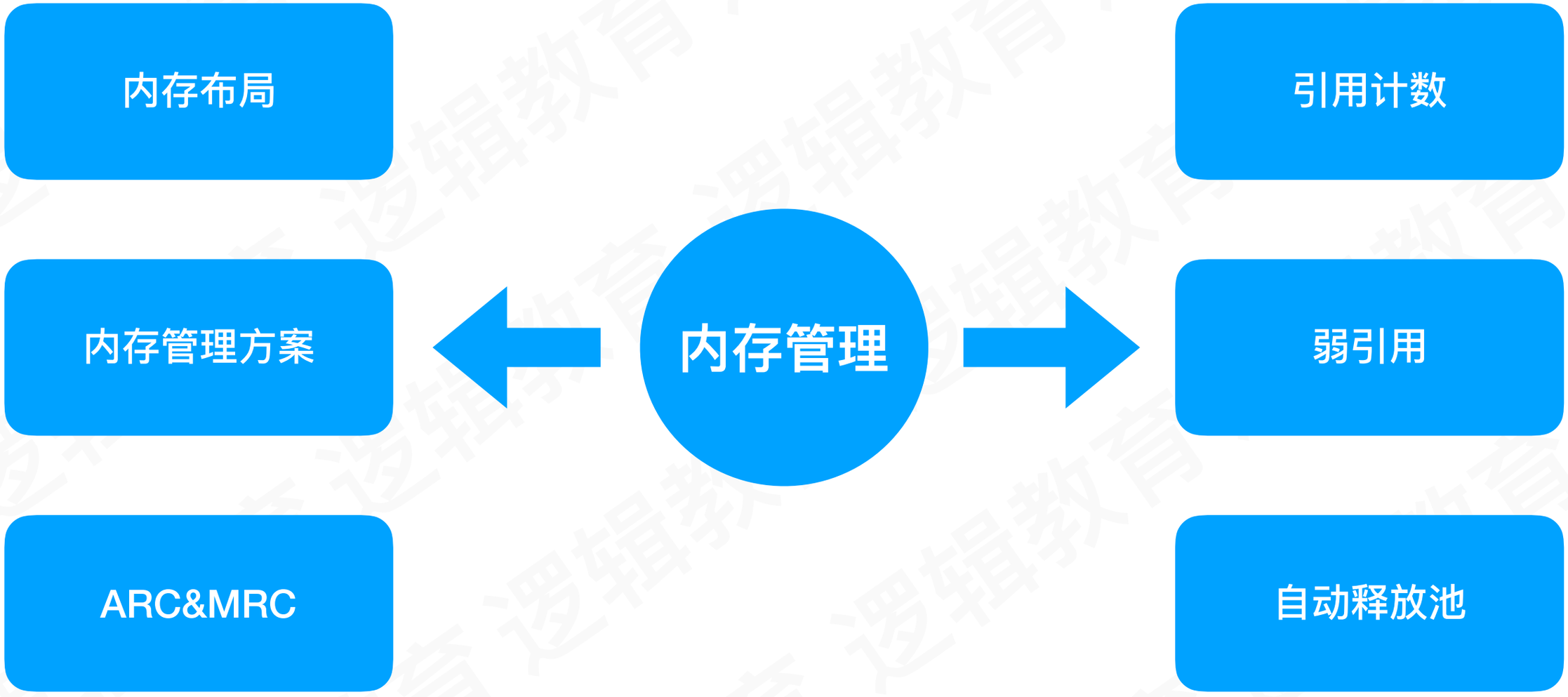
逻辑教育
Logic education

大师班第28天

和谐学习，不急不躁

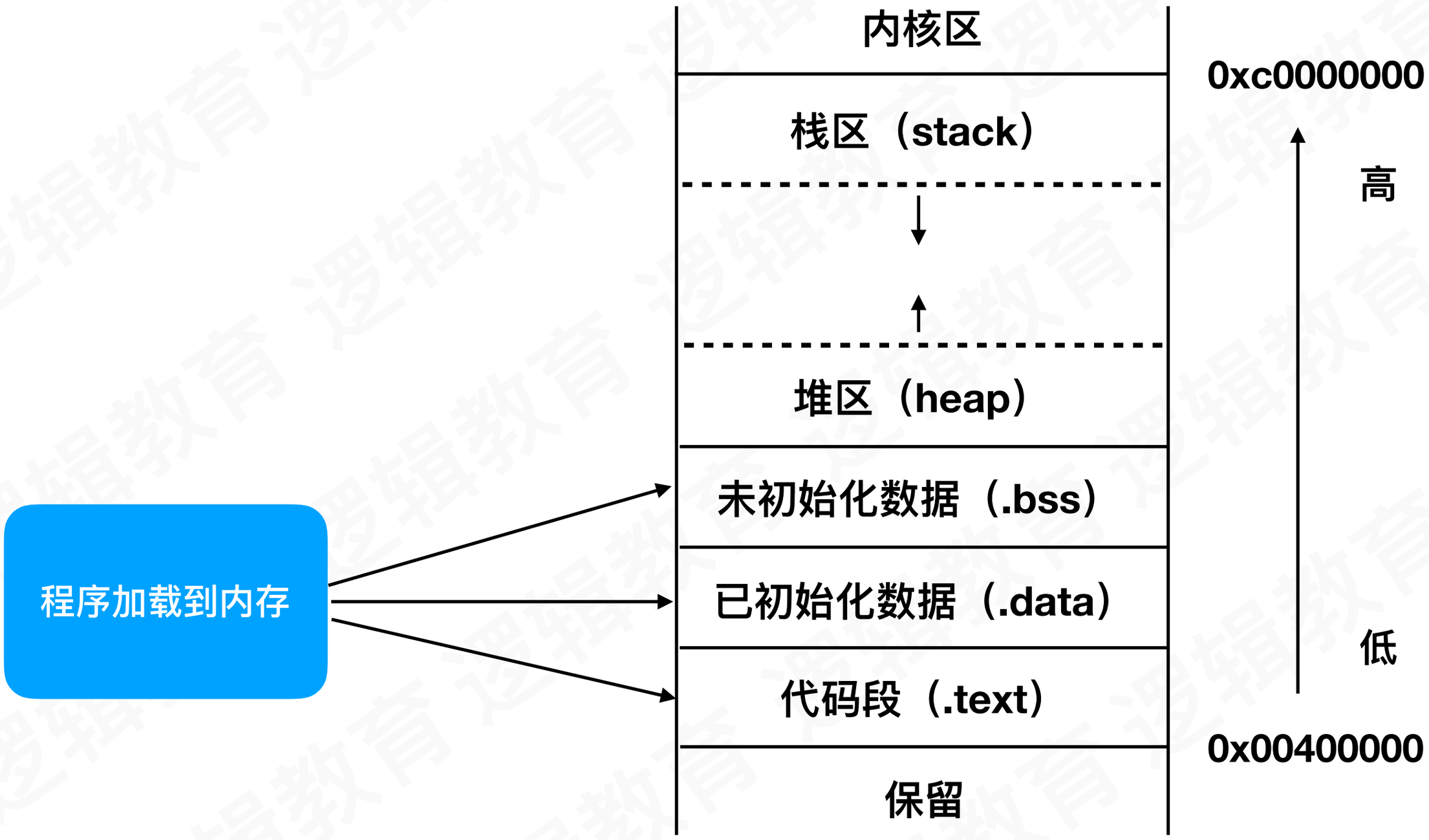
LG_Cooci

课程研发:Cooci老师
课程授课:Cooci老师



课程研发:Cooci老师
课程授课:Cooci老师

内存布局



课程研发:Cooci老师
课程授课:Cooci老师

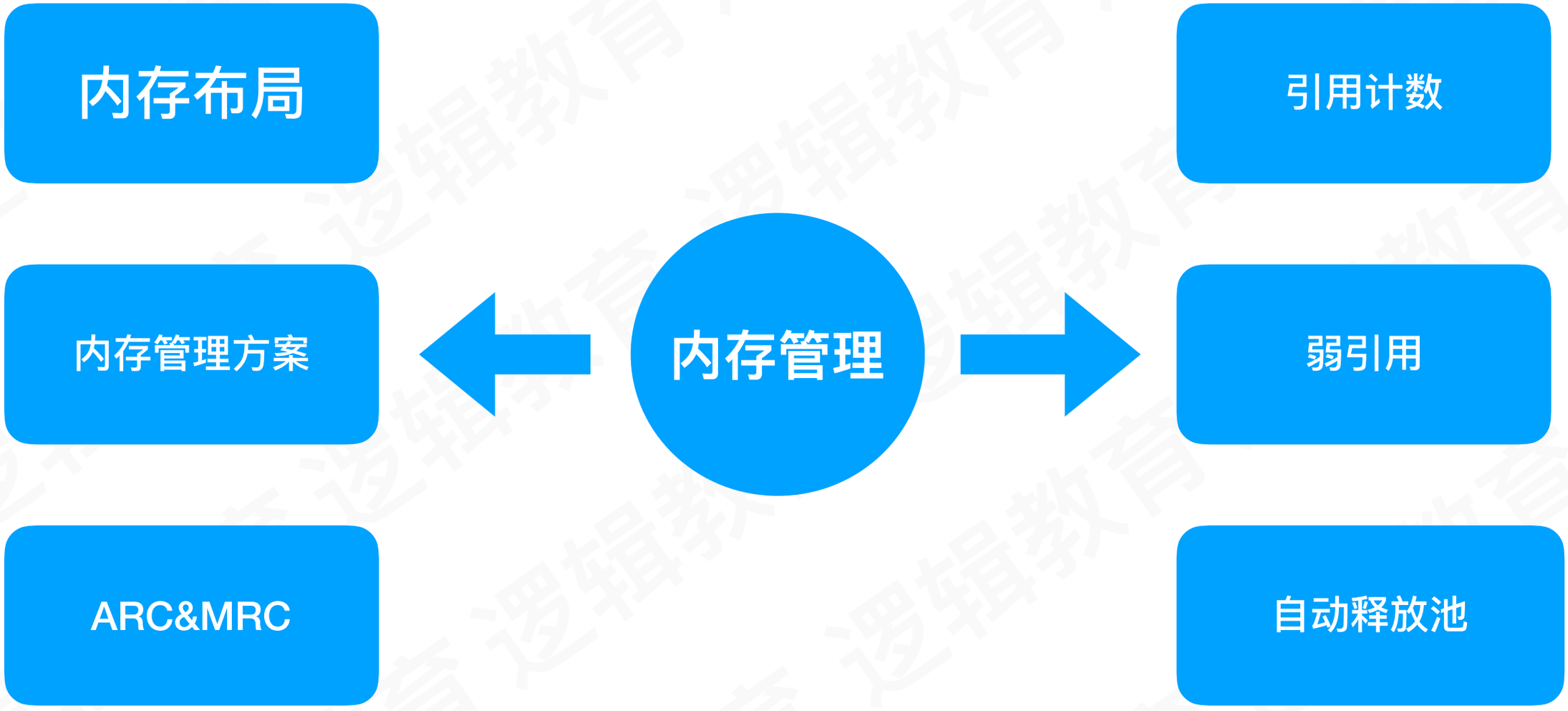
栈区：函数，方法
堆区：通过alloc分配的对象，block copy
BSS段：未初始化的全局变量，静态变量
数据段：初始化的全局变量，静态变量
text：程序代码，加载到内存中

内存布局

栈区内存地址：一般为：0x7开头

堆区内存地址：一般为：0x6开头

数据段，BSS内存地址：一般为：0x1开头



课程研发:Cooci老师
课程授课:Cooci老师

内存管理方案

TaggedPointer：小对象-NSNumber,NSDate

NONPOINTER_ISA：非指针型isa

散列表：引用计数表，弱引用表

课程研发:Cooci老师
课程授课:Cooci老师

TaggedPointer

```
static inline void * _Nonnull
_objc_encodeTaggedPointer(uintptr_t ptr)
{
    return (void *)(objc_debug_taggedpointer_obfuscator ^ ptr);
}
```

```
static inline uintptr_t
_objc_decodeTaggedPointer(const void * _Nullable ptr)
{
    return (uintptr_t)ptr ^ objc_debug_taggedpointer_obfuscator;
}
```

TaggedPointer

```
static void
initializeTaggedPointerObfuscator(void)
{
    if (sdkIsOlderThan(10_14, 12_0, 12_0, 5_0, 3_0) ||
        // Set the obfuscator to zero for apps linked against older SDKs,
        // in case they're relying on the tagged pointer representation.
        DisableTaggedPointerObfuscation) {
        objc_debug_taggedpointer_obfuscator = 0;
    } else {
        // Pull random data into the variable, then shift away all non-payload
            bits.
        arc4random_buf(&objc_debug_taggedpointer_obfuscator,
                       sizeof(objc_debug_taggedpointer_obfuscator));
        objc_debug_taggedpointer_obfuscator &= ~_OBJC_TAG_MASK;
    }
}
```

TaggedPointer

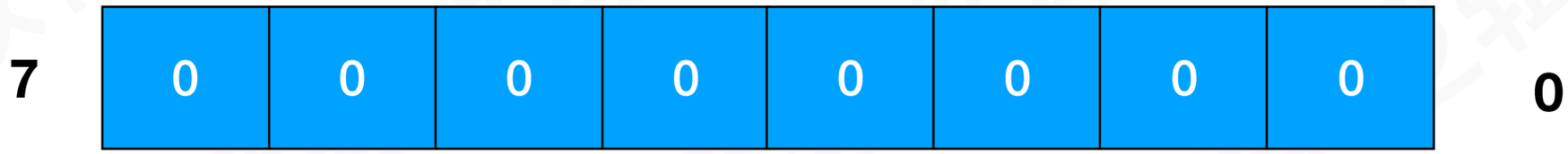
1：Tagged Pointer专门用来存储小的对象，例如NSNumber和NSDate

2：Tagged Pointer指针的值不再是地址了，而是真正的值。所以，实际上它不再是一个对象了，它只是一个披着对象皮的普通变量而已。所以，它的内存并不存储在堆中，也不需要malloc和free

3.在内存读取上有着3倍的效率，创建时比以前快106倍。

课程研发:Cooci老师
课程授课:Cooci老师

NONPOINTER_ISA



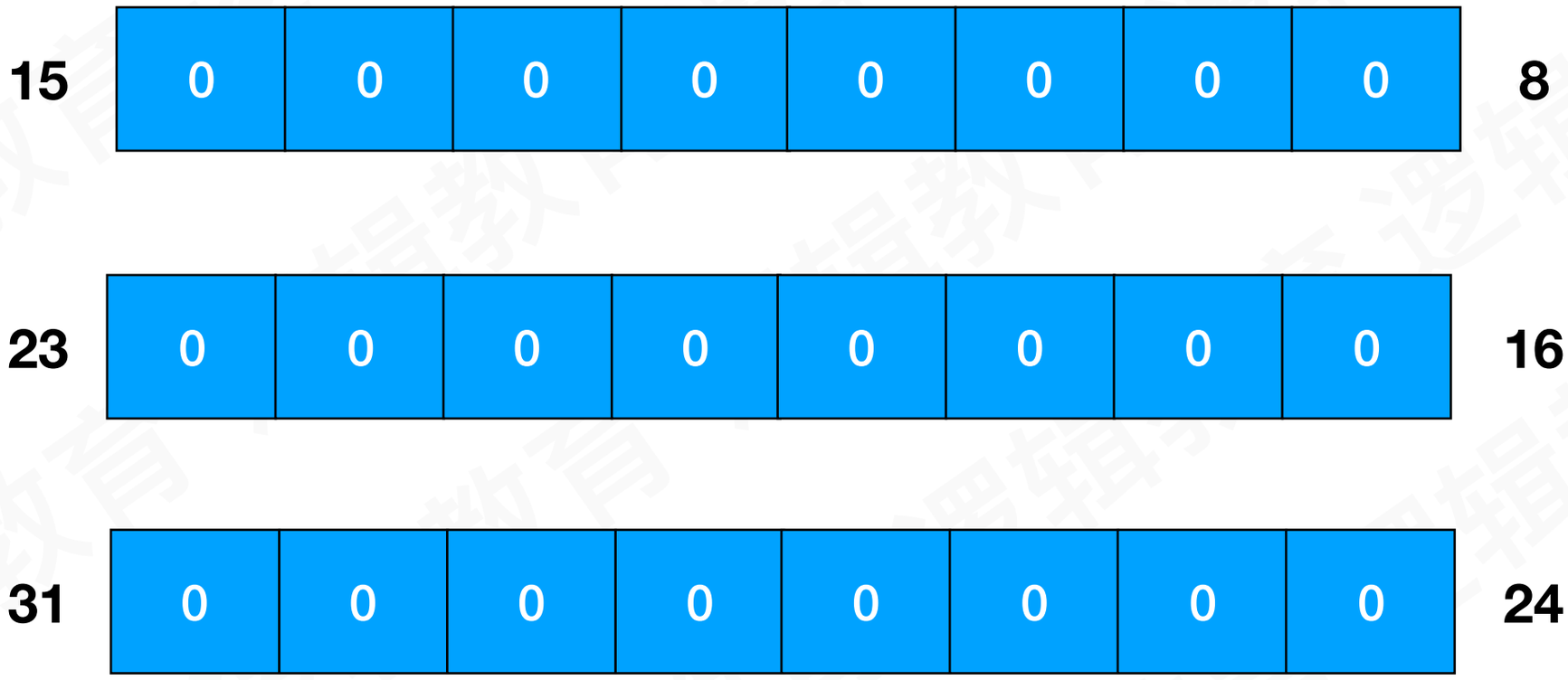
nonpointer：表示是否对 isa 指针开启指针优化

0：纯isa指针，1：不止是类对象地址,isa 中包含了类信息、对象的引用计数等

has_assoc：关联对象标志位，0没有，1存在

has_cxx_dtor：该对象是否有 C++ 或者 Objc 的析构器,如果有析构函数,则需要做析构逻辑,如果没有,则可以更快的释放对象

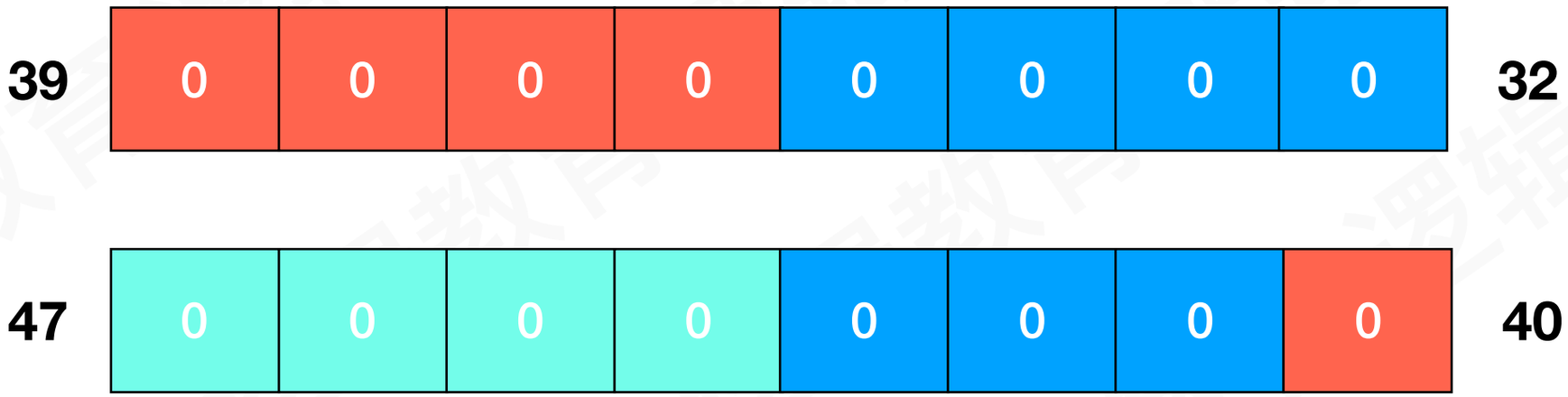
NONPOINTER_ISA



shiftcls:

存储类指针的值。开启指针优化的情况下，在 arm64 架构中有 33 位用来存储类指针。

NONPOINTER_ISA



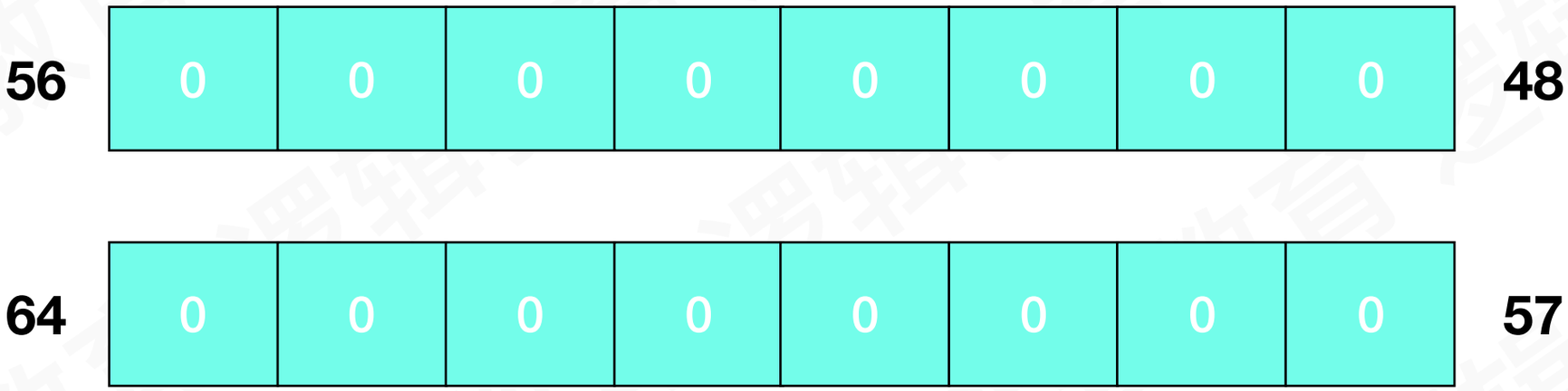
magic：用于调试器判断当前对象是真的对象还是没有初始化的空间

weakly_referenced：志对象是否被指向或者曾经指向一个 ARC 的弱变量，没有弱引用的对象可以更快释放。

deallocating：标志对象是否正在释放内存

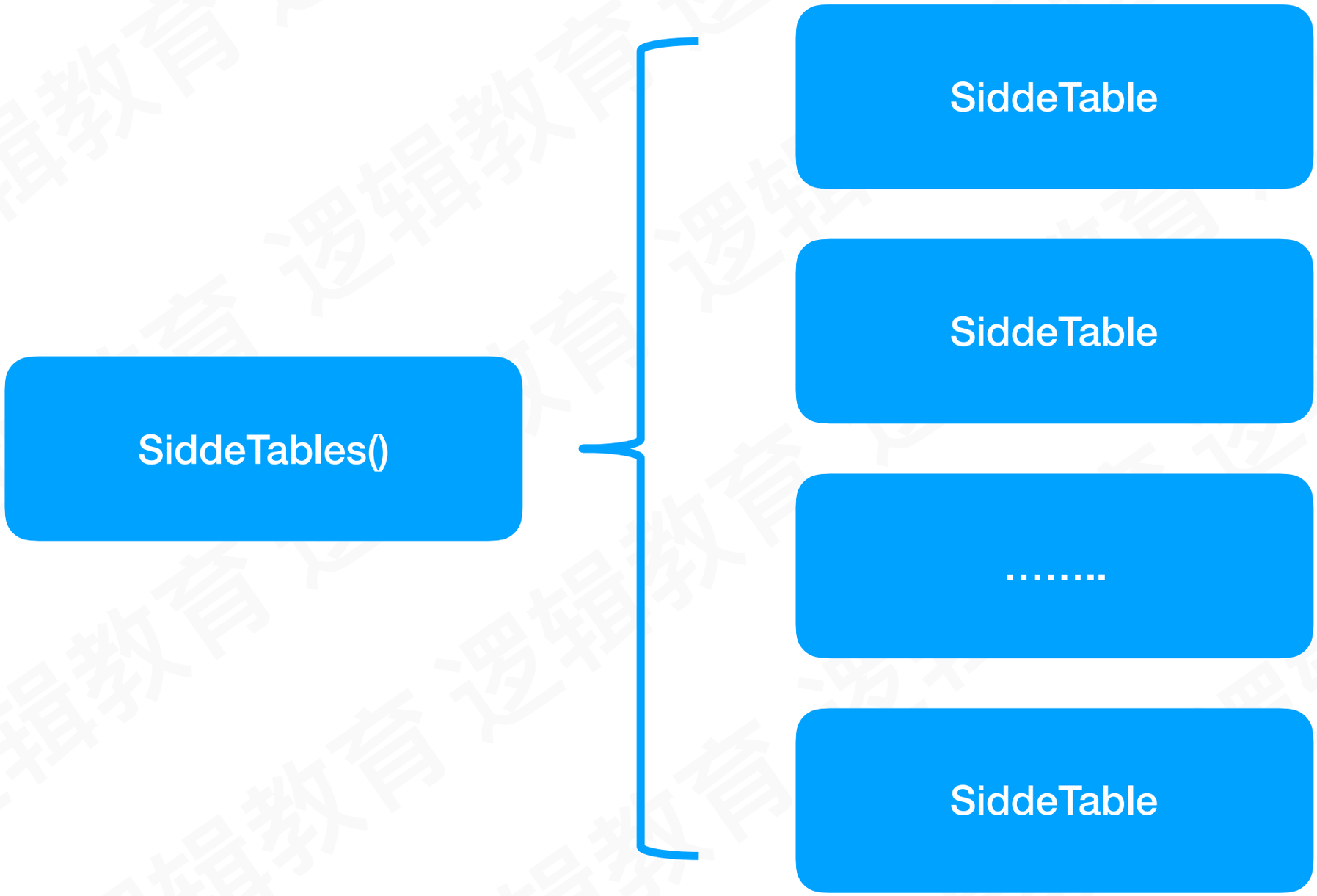
has_sidetable_rc：当对象引用技术大于 10 时，则需要借用该变量存储进位

NONPOINTER_ISA



extra_rc：当表示该对象的引用计数值，实际上是引用计数值减 1，
例如，如果对象的引用计数为 10，那么 extra_rc 为 9。如果引用计数大于 10，
则需要使用到下面的 has_sidetable_rc。

散列表



课程研发:Cooci老师
课程授课:Cooci老师

散列表

- Spinlock_t:自旋锁
- SiddeTable
- RefcountMap引用技术
- weak_table_t 弱引用表

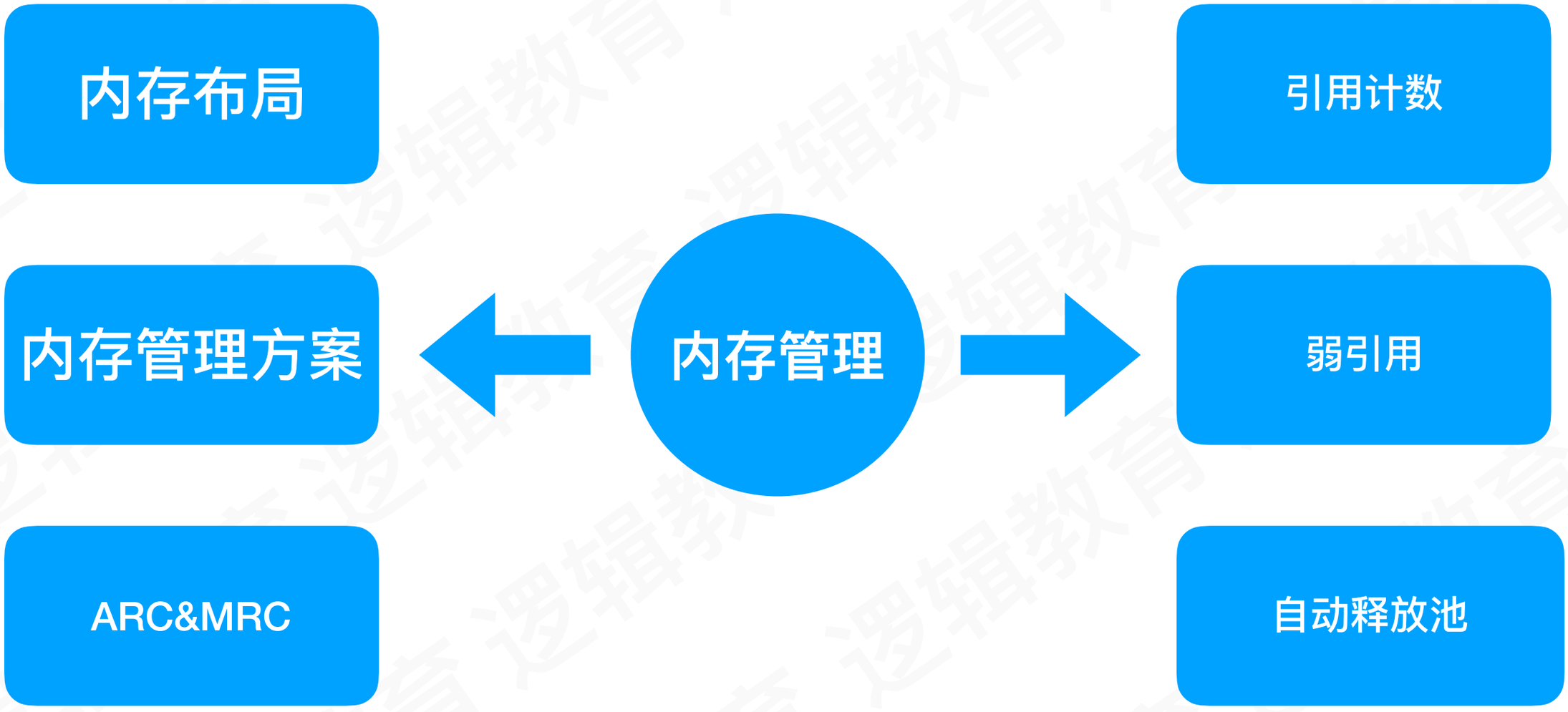
课程研发:Cooci老师
课程授课:Cooci老师

散列表

为什么是多张sideTable，而不是一张表?

Person	2
Student	5
Teacher	1
iOSer	0

课程研发:Cooci老师
课程授课:Cooci老师



课程研发:Cooci老师
课程授课:Cooci老师

逻辑教育
Logic education

MRC&ARC

alloc

retain

realese

retainCount

autorealese

dealloc

课程研发:Cooci老师
课程授课:Cooci老师

MRC&ARC

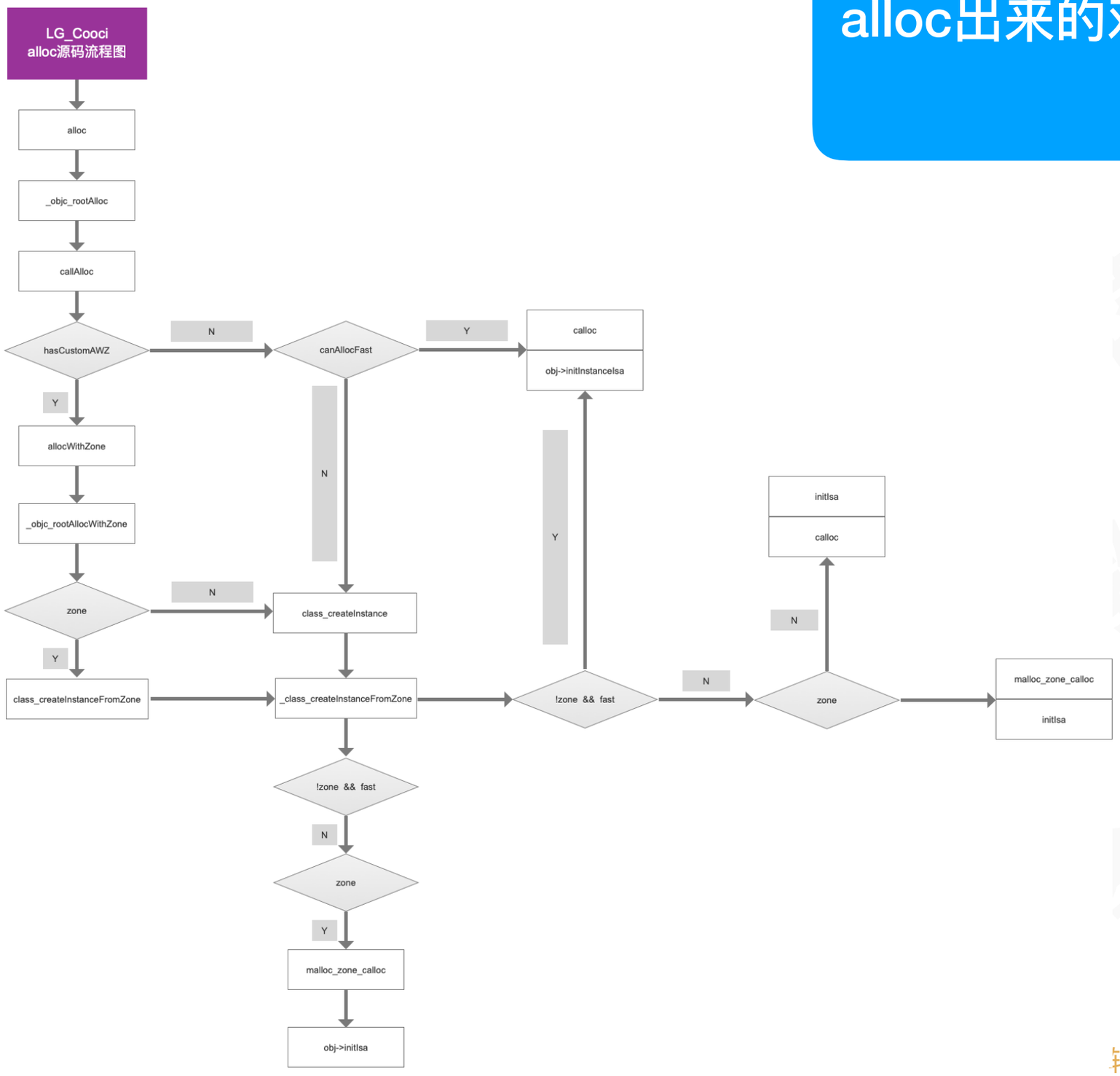
ARC是LLVM和Runtime配合的结果

ARC中禁止手动调用retain/release/retainCount/dealloc

ARC新加了weak、strong属性关键字

课程研发:Cooci老师
课程授课:Cooci老师

alloc



alloc出来的对象retainCount是多少?

retainCount

```
inline uintptr_t
objc_object::rootRetainCount()
{
    if (isTaggedPointer()) return (uintptr_t)this;

    sidetable_lock();
    isa_t bits = LoadExclusive(&isa.bits);
    ClearExclusive(&isa.bits);
    if (bits.nonpointer) {
        uintptr_t rc = 1 + bits.extra_rc;
        if (bits.has_sidetable_rc) {
            rc += sidetable_getExtraRC_nolock();
        }
        sidetable_unlock();
        return rc;
    }

    sidetable_unlock();
    return sidetable_retainCount();
}
```

retain&release

```
newisa.bits = addc(newisa.bits, RC_ONE, 0, &carry);  // extra_rc++
```

针对相应引用计数位加1

如果引用计数出现上溢出，那么我们开始分开存储，一半存到散列表

retain&release

```
newisa.bits = subc(newisa.bits, RC_ONE, 0, &carry);  // extra_rc--
```

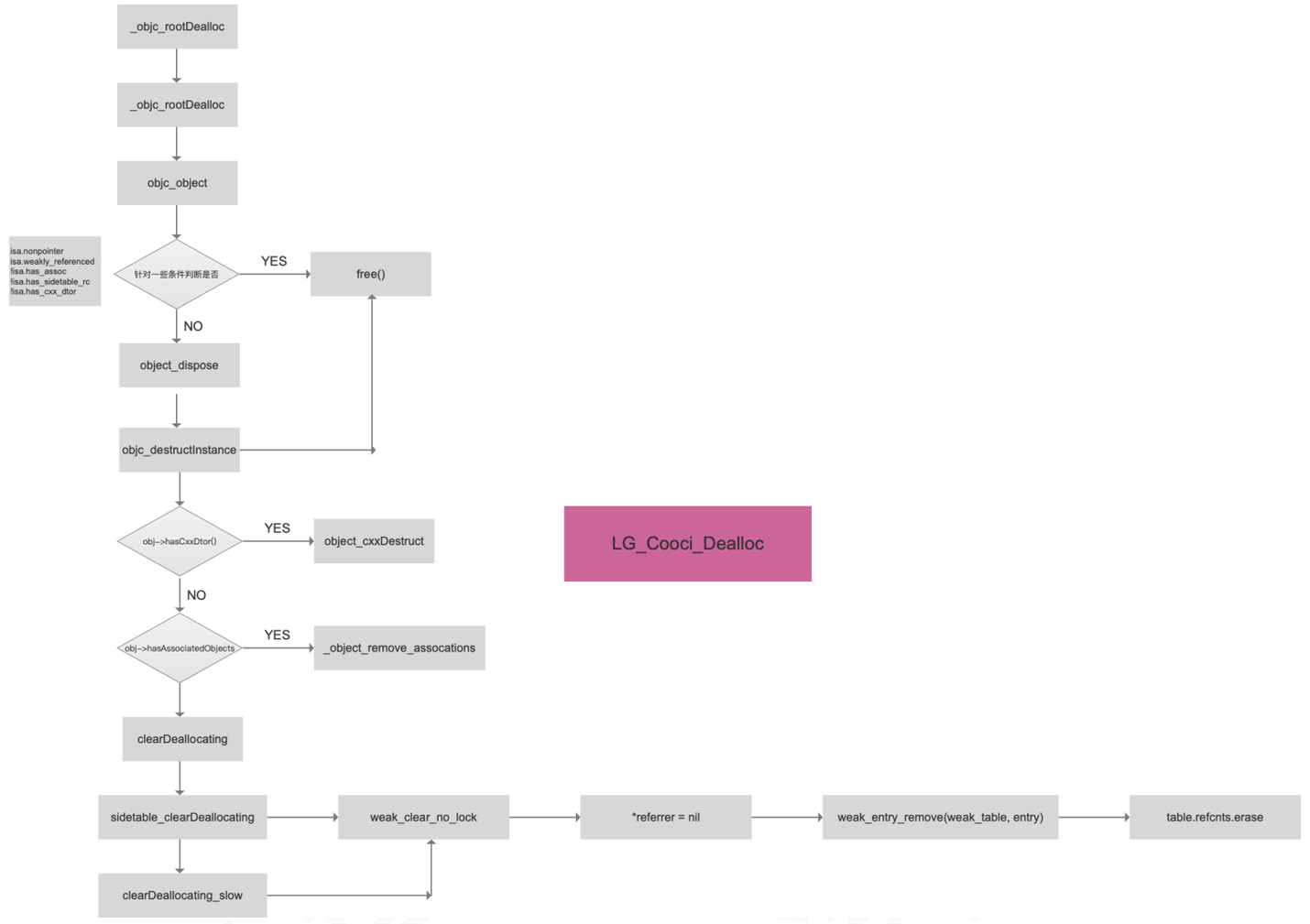
针对相应引用计数位减1

如果引用计数出现下溢出，就去散列表借来的引用计数 - 1 存到extra_rc

release 就算借散列表的引用计数过来，还是下溢出，那么就调用dealloc

```
__sync_synchronize();
if (performDealloc) {
    ((void(*)(objc_object *, SEL))objc_msgSend)(this, SEL_dealloc);
}
return true;
```

dealloc



课程研发:Cooci老师
课程授课:Cooci老师

dealloc

1：根据当前对象的状态是否直接调用free（）释放

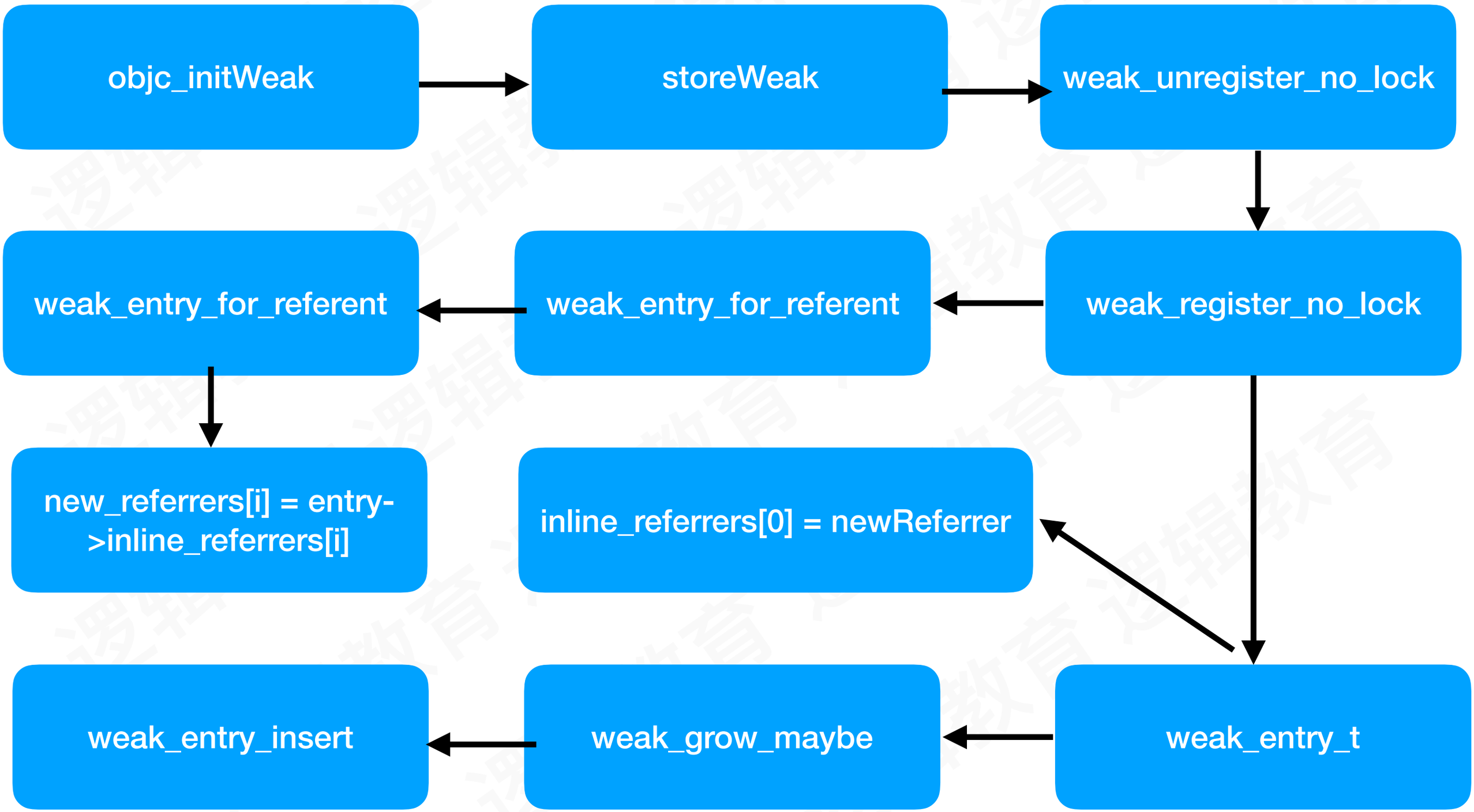
2：是否存在C++的析构函数、移除这个对象的关联属性

3：将指向该对象的弱引用指针置为nil

4：从弱引用表中擦除对该对象的引用计数

课程研发:Cooci老师
课程授课:Cooci老师

weak



weak

1：首先我们知道有一个非常牛逼的家伙-sideTable

2：得到sideTable的weakTable 弱引用表

3：创建一个weak_entry_t

4：把referent加入到weak_entry_t的数组inline_referrers

5：把weak_table扩容一下

5：把new_entry加入到weak_table中

课程研发:Cooci老师
课程授课:Cooci老师

strong&unsafe_unretain

变量修饰符有以下几种情况

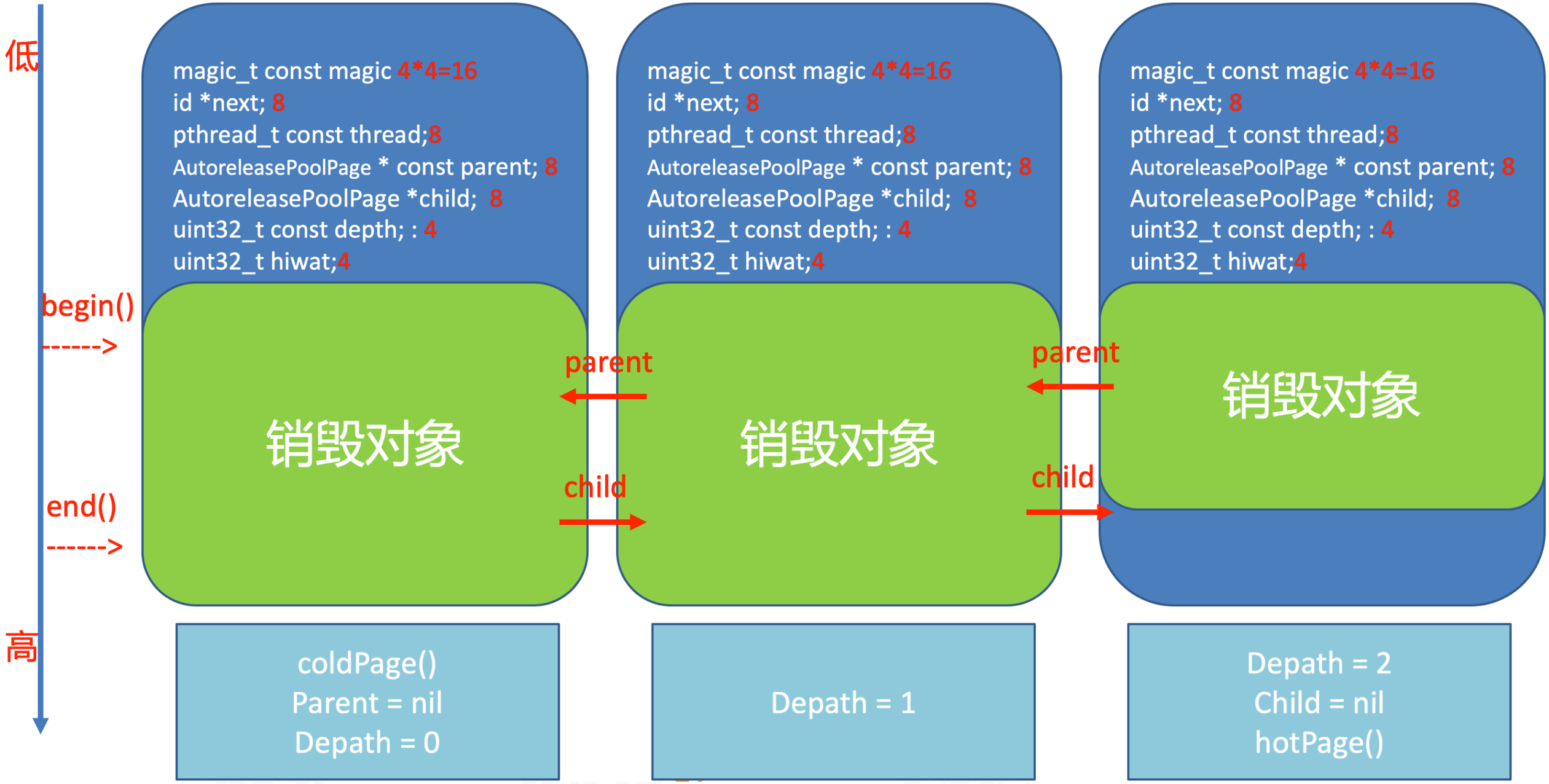
```
switch (memoryManagement) {
case objc_ivar_memoryWeak:       objc_storeWeak(location, value); break;
case objc_ivar_memoryStrong:     objc_storeStrong(location, value); break;
case objc_ivar_memoryUnretained: *location = value; break;
case objc_ivar_memoryUnknown:    _objc_fatal("impossible");
}
```

strong&unsafe_unretain

retain新值，release旧值

```
void
objc_storeStrong(id *location, id obj)
{
    id prev = *location;
    if (obj == prev) {
        return;
    }
    objc_retain(obj);
    *location = obj;
    objc_release(prev);
}
```

自动释放池



课程研发:Cooci老师
课程授课:Cooci老师

自动释放池

```
* magic 用来校验 AutoreleasePoolPage 的结构是否完整;
* next 指向最新添加的 autoreleased 对象的下一个位置，初始化时指向
  begin() ;
* thread 指向当前线程;
* parent 指向父结点，第一个结点的 parent 值为 nil ;
* child 指向子结点，最后一个结点的 child 值为 nil ;
* depth 代表深度，从 0 开始，往后递增 1;
* hiwat 代表 high water mark 。
```

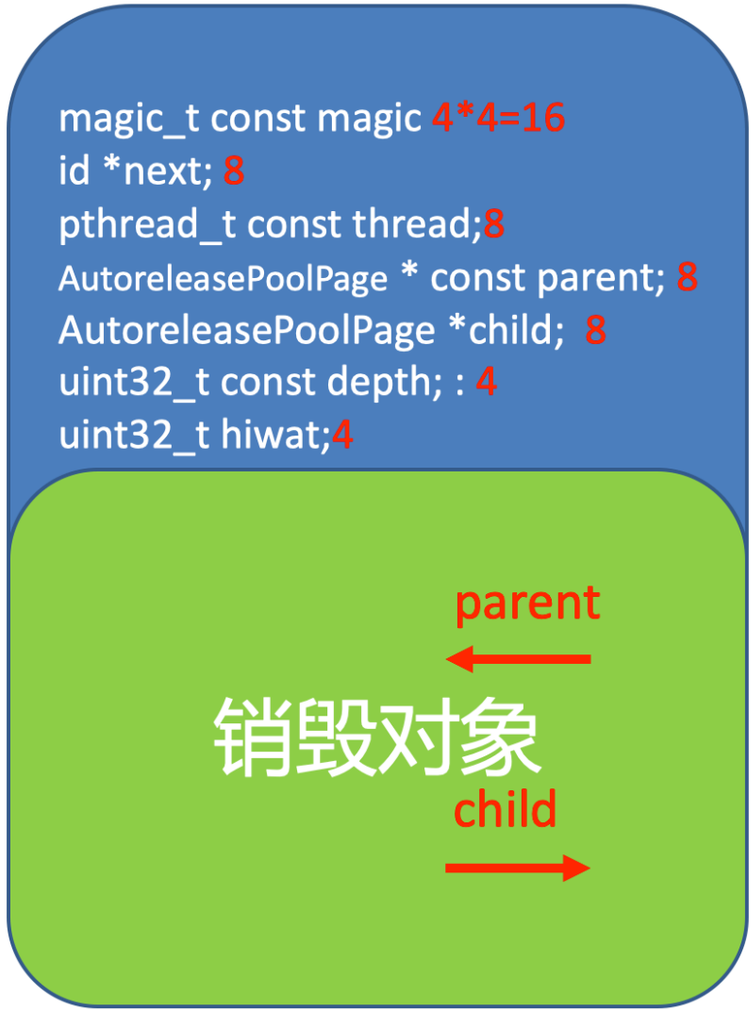
自动释放池

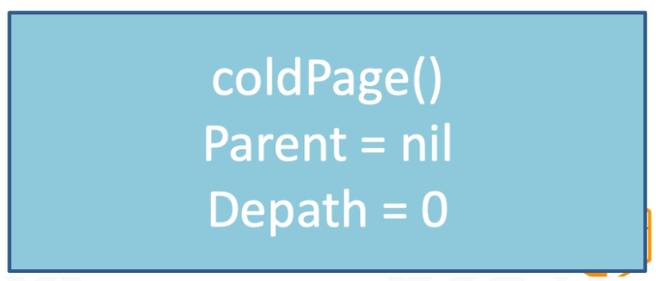
低

高

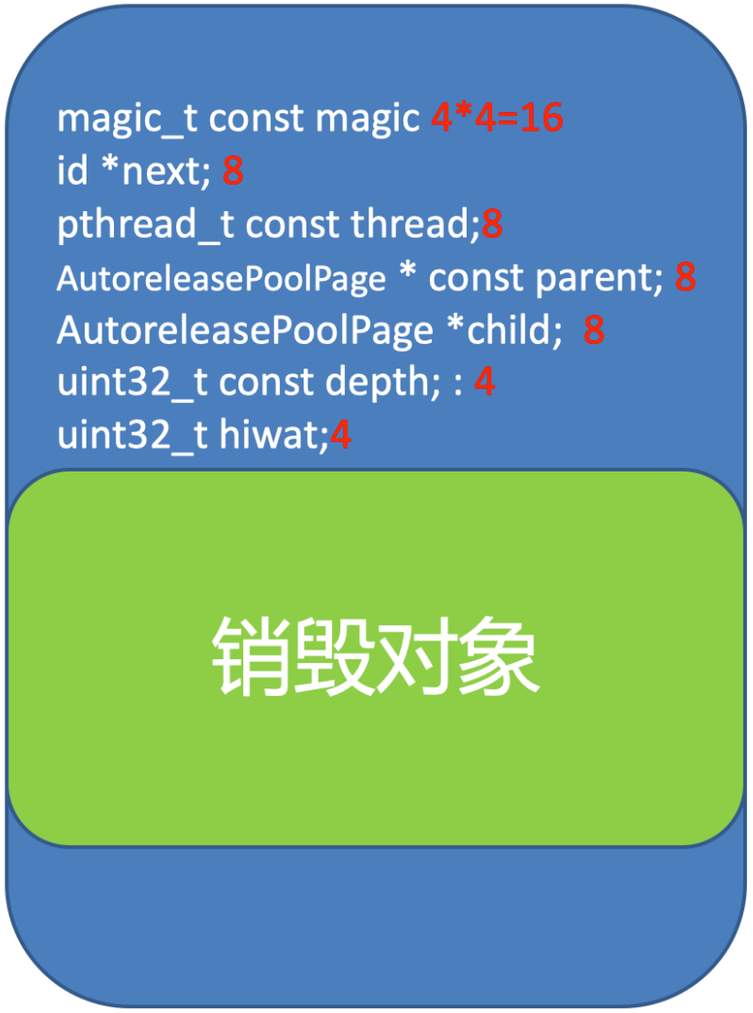
begin()
------>

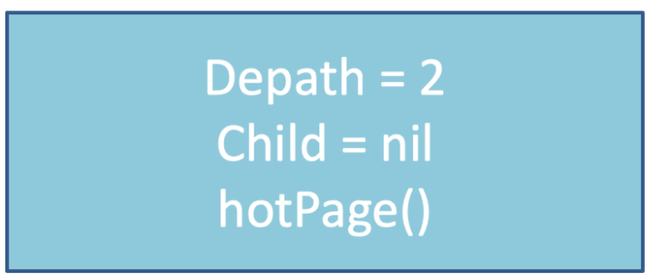
end()
------>





Next()
------>
Objc
release





逻辑教育
Logic education

Hello Cooci

我就是我，颜色不一样的烟火

课程研发:Cooci老师
课程授课:Cooci老师