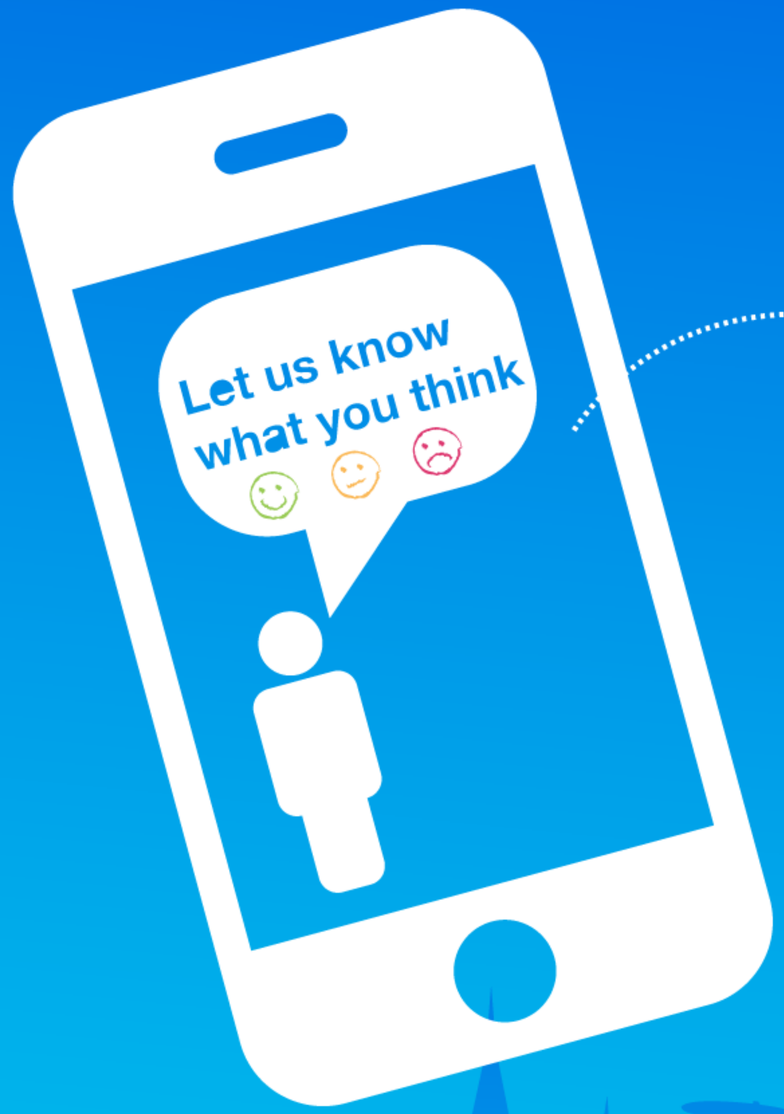
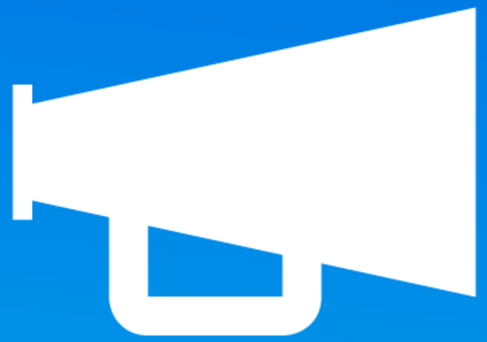
goto; conference

Join the conversation #gotocph

goto; conference





Click ‘engage’
to rate sessions
and ask questions

Join the conversation #gotocph

Swift Memory Layout

Mike Ash

GOTO Copenhagen 2016

💤👌🎃🌈🎤👛💫

About Me

mikeash.com
plausible.coop
NSBlog
github.com/mikeash
@mikeash

PHOTO

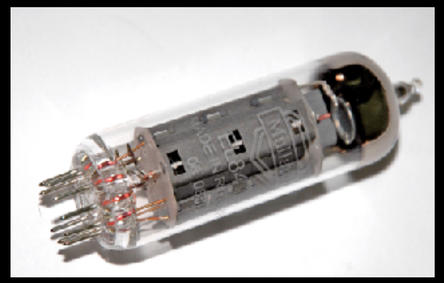
- What even is memory?
- Memory dumper program
- How Swift lays out data

Memory

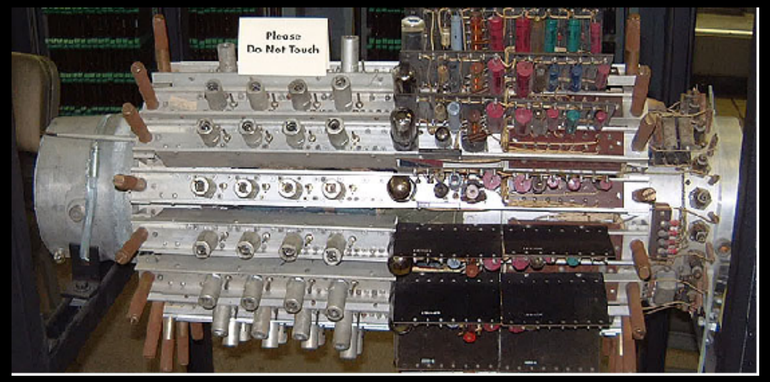
Memory

Memory

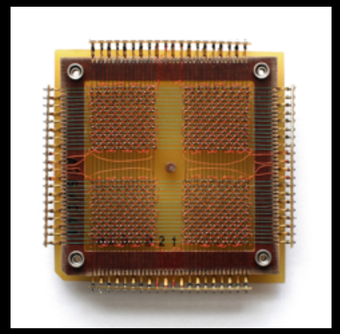
Vacuum tubes



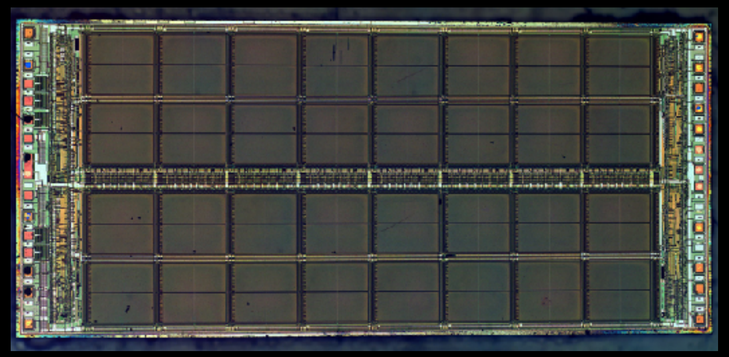
Mercury (or gin) delay line



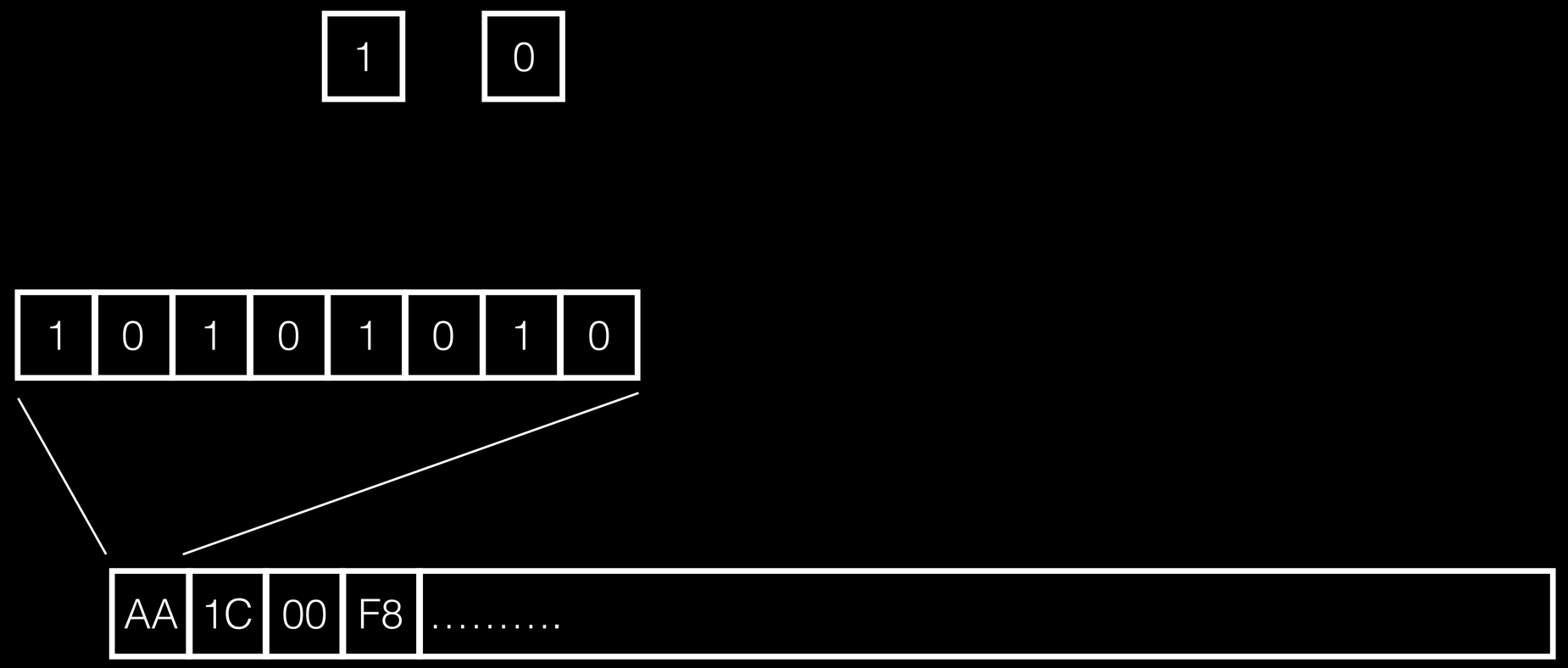
Magnetic core



DRAM



Memory



Memory

0	1	2	3	……
AA	1C	00	F8	…….

Memory

0	01AA2C5EFF001101
8	0000000000000000
16	00000000000000FF
24	A0F31C228A177013
…	⋮

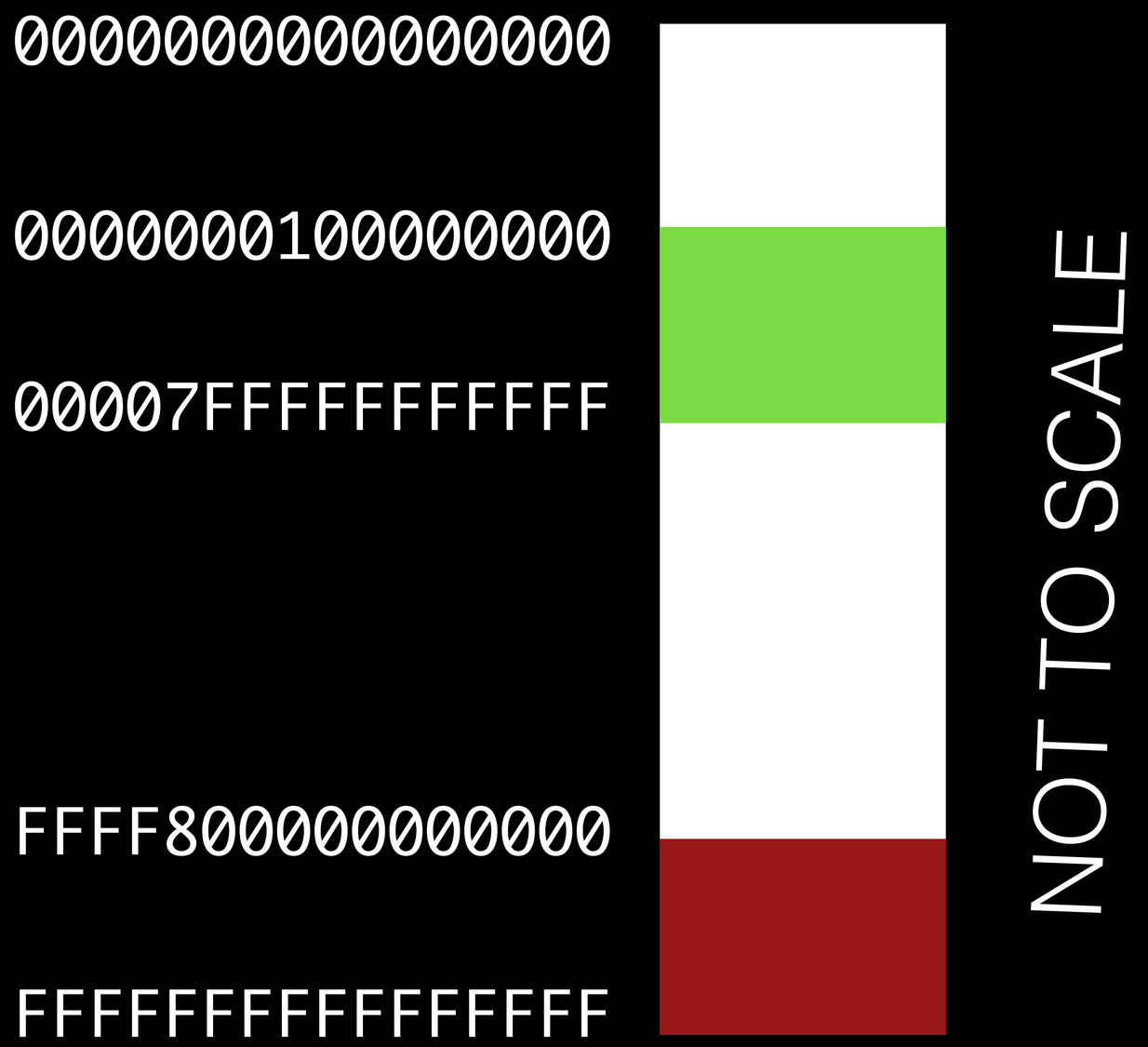
Memory

0000000000000000	01AA2C5EFF001101
0000000000000008	0000000000000000
0000000000000010	00000000000000FF
0000000000000018	A0F31C228A177013
…	⋮

Memory

…	⋮
0000000100000000	01AA2C5EFF001101
0000000100000008	0000000000000000
0000000100000010	00000000000000FF
0000000100000018	A0F31C228A177013
…	⋮

Memory - Big Picture



Memory

…	⋮
0000000100000000	01AA2C5EFF001101
0000000100000008	0000000000000000
0000000100000010	00000000000000FF
0000000100000018	00000001000AE780
…	⋮
00000001000AE780	A0F31C228A177013
	001101FFAE738000

Memory

…	⋮
0000000100000000	01AA2C5EFF001101
0000000100000008	0000000000000000
0000000100000010	00000000000000FF
0000000100000018	00000001000AE780
…	⋮
00000001000AE780	A0F31C228A177013
	001101FFAE738000

Memory

…	⋮
0000000100000000	01AA2C5EFF001101
0000000100000008	0000000000000000
0000000100000010	00000000000000FF
0000000100000018	80E70A0001000000
…	⋮
000000001000AE780	A0F31C228A177013
	001101FFAE738000

Memory

Stack

```
var x = …
var y = …
var z = x + y
let string = view.text
let text = view.string
let count = array.count
```

Heap

```
UIView()
NSObject()
MyClass()
YourClass()
TheirClass()

malloc/free
```

Global Data

```
"string constants"
"more string constants"
class MyClass {}
struct MyStruct {}
protocol MyProtocol {}
```

Dumping Memory

```
var x = ...
bytes(of: &x)
```

```
func bytes<T>(of value: T) -> [UInt8] {
    ...
}
```

Dumping Memory

https://github.com/mikeash/memorydumper2

http://tinyurl.com/swmem

http://www.www.reallyhugeurl.com/index.php/freak=no5zyn3o&ego.y=0p0iyjmf&lol=td2g2qxx&oed=gojsz0bh&oed=fheq2iqt&ego.x=g1c2s5daxsjkjhf&ssn=7kegc1kllfo1r0a&eat=qe4zk8hgmzvl827&oedeldritch=9qtni82cz8omnzk1x13twrw1qohhuhkrbuzr06q8ya1evomdpsaglggcyhde4ksr5

Dumping Memory

Xcode 8

Swift 3

Dumping Memory

```
var x = ...
bytes(of: &x)
```

```
func bytes<T>(of value: T) -> [UInt8] {
    ...
}
```

Dumping Memory

```
func bytes<T>(of value: T) -> [UInt8] {
    var value = value
    let size = MemoryLayout<T>.size
    return withUnsafePointer(to: &value, {
        $0.withMemoryRebound(
            to: UInt8.self,
            capacity: size,
        {
            Array(UnsafeBufferPointer(
                  start: $0, count: size))
        })
    })
}
```

Dumping Memory

```
let x = 0x0102030405060708
print(bytes(of: x))
print(bytes(of: 42))

[8, 7, 6, 5, 4, 3, 2, 1]
[42, 0, 0, 0, 0, 0, 0, 0]
```

Dumping Memory

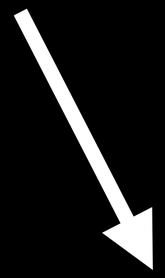
```
func hexString<Seq: Sequence>
    (bytes: Seq, limit: Int? = nil, separator: String = " ")
    -> String
where Seq.Iterator.Element == UInt8 {
    let spacesInterval = 8
    var result = ""
    for (index, byte) in bytes.enumerated() {
        if let limit = limit, index >= limit {
            result.append("...")
            break
        }
        if index > 0 && index % spacesInterval == 0 {
            result.append(separator)
        }
        result.append(String(format: "%02x", byte))
    }
    return result
}
```

Dumping Memory

```
let x = 0x0102030405060708
print(hexString(bytes: bytes(of: x)))
print(hexString(bytes: bytes(of: 42)))

0807060504030201
2a00000000000000
```

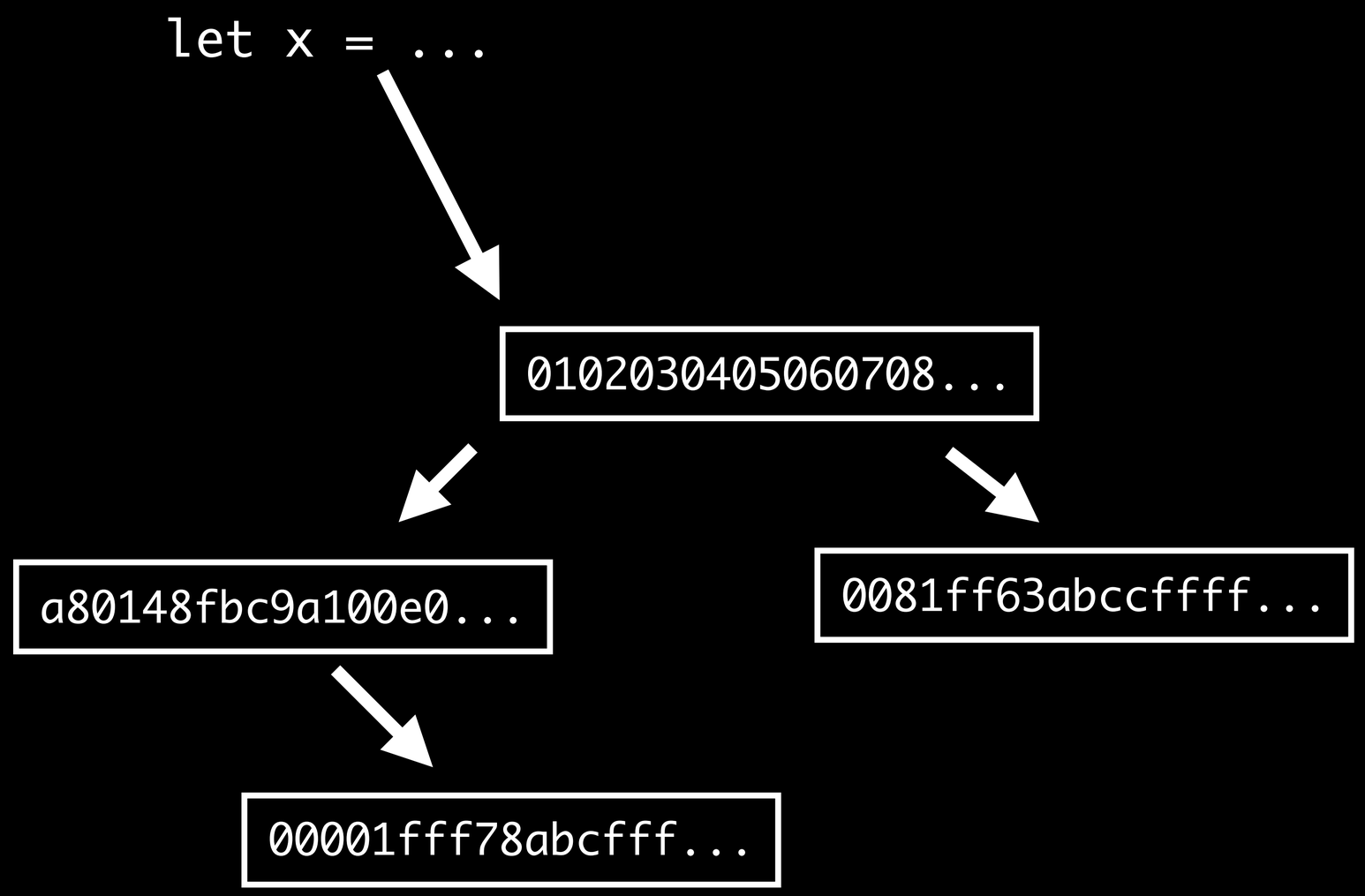
Dumping Memory

```
let x = ...
```



```
0102030405060708...
```

Dumping Memory



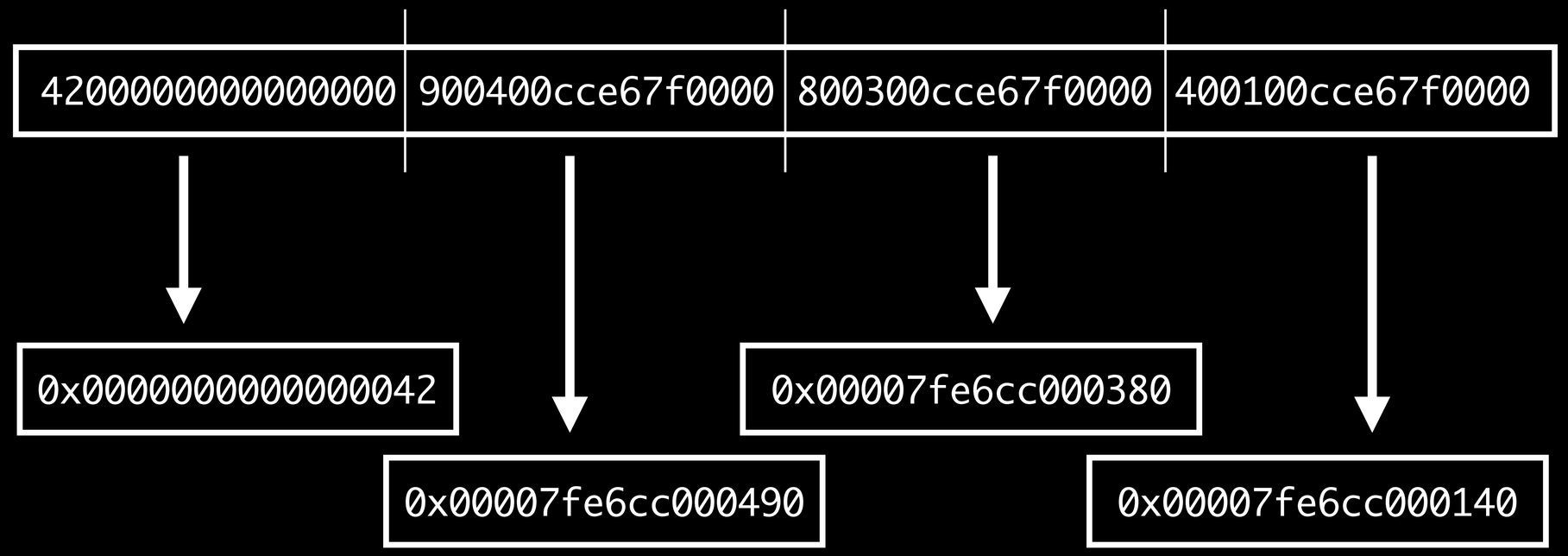
Pointers are Integers

```
struct Pointer {
    var address: UInt
}
```

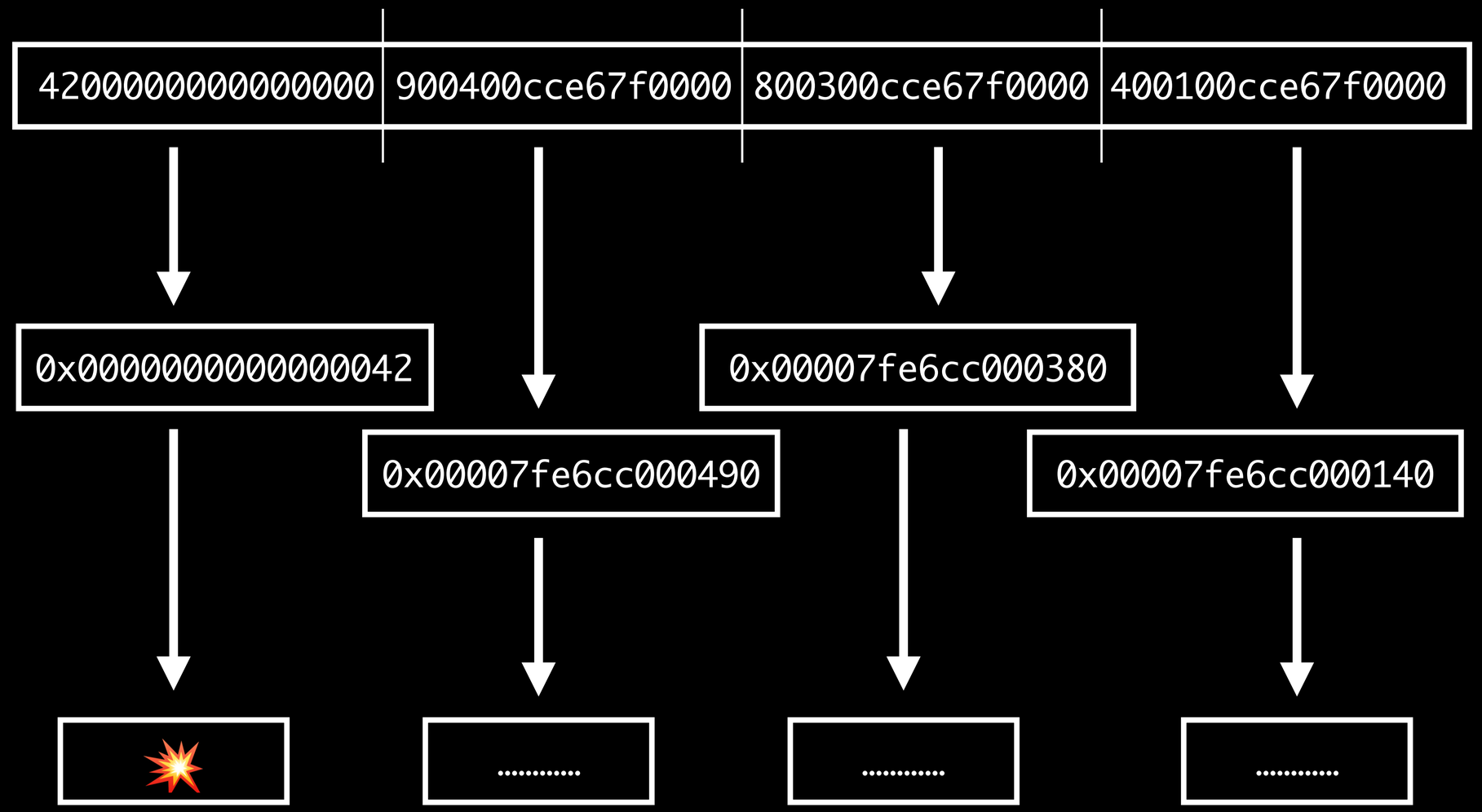
Pointers are Integers

```
buffer.withUnsafeBufferPointer({ bufferPointer in
    return bufferPointer.baseAddress?.withMemoryRebound(
        to: Pointer.self,
        capacity: bufferPointer.count / MemoryLayout<Pointer>.size,
    {
        let castBufferPointer = UnsafeBufferPointer(
            start: $0,
            count: bufferPointer.count / MemoryLayout<Pointer>.size)
        return Array(castBufferPointer)
    }) ?? []
})
```

Dumping Memory



Bad Pointers



Bad Pointers

`mach_vm_read_overwrite`

Copy N bytes from X to Y

Similar to `memcpy`

Returns an error on invalid pointers

Bad Pointers

```
public func mach_vm_read_overwrite(
    _ target_task: vm_map_t,
    _ address: mach_vm_address_t,
    _ size: mach_vm_size_t,
    _ data: mach_vm_address_t,
    _ outsize: UnsafeMutablePointer<mach_vm_size_t>!)
 -> kern_return_t
```

Safe Reads

```
func safeRead(ptr: Pointer, into: inout [UInt8]) -> Bool {
    let result =
        into.withUnsafeMutableBufferPointer(
    { bufferPointer -> kern_return_t in
        var outSize: mach_vm_size_t = 0
        return mach_vm_read_overwrite(
            mach_task_self_,
            mach_vm_address_t(ptr),
            mach_vm_size_t(bufferPointer.count),
            mach_vm_address_t(bufferPointer.baseAddress),
            &outSize)
    })
    return result == KERN_SUCCESS
}
```

How Much to Read?

Initial value: `MemoryLayout<T>.size`

Heap allocations: `malloc_size`

Code and Globals: scan with `dladdr`

Bonus: dladdr also gives names

Name Mangling

```
_TFCs23_ContiguousArrayStorage32_getNonV
erbatimBridgedHeapBufferfT_GVs11_HeapBuf
ferSiPs9AnyObject__
```

➜ swift-demangle ➜

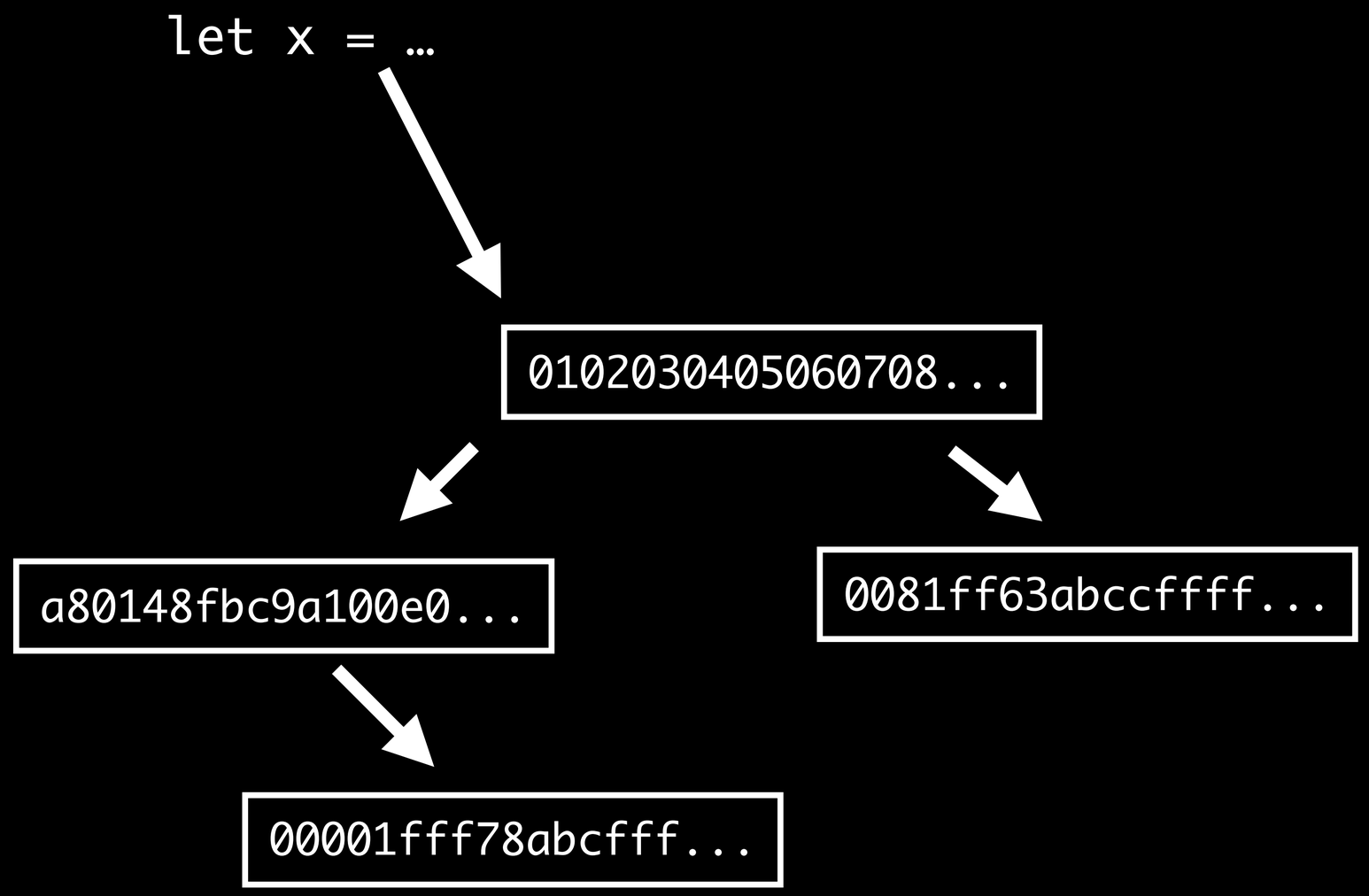
```
Swift._ContiguousArrayStorage
._getNonVerbatimBridgedHeapBuffer () ->
Swift._HeapBuffer<Swift.Int,
Swift.AnyObject>
```

Strings

Search for printable ASCII at least 4 characters long

```
let lowerBound: UInt8 = 32
let upperBound: UInt8 = 126

let pieces = buffer.split(whereSeparator:
     { !(lowerBound ... upperBound ~= $0) })
let sufficientlyLongPieces = pieces.filter(
     { $0.count >= 4 })
return sufficientlyLongPieces.map(
     { String(bytes: $0, encoding: .utf8)! })
```

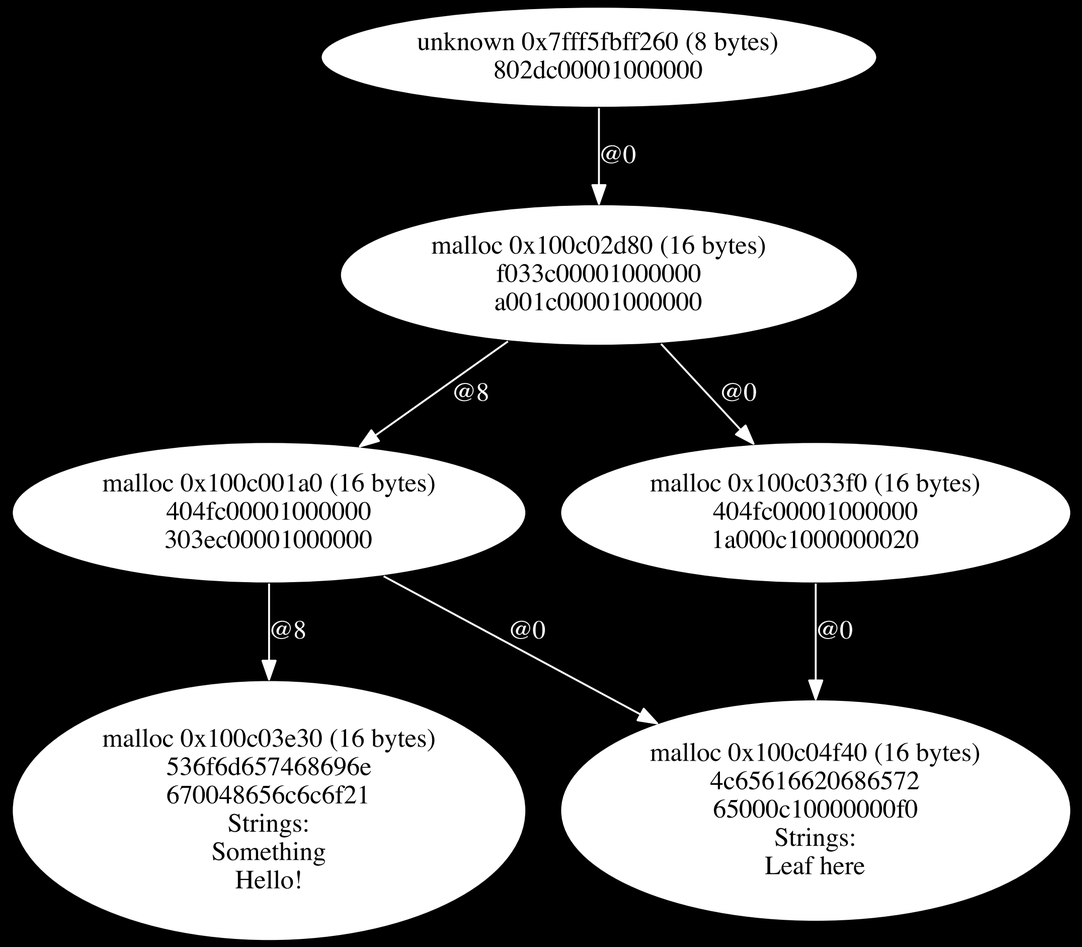
Output


let x = …
0102030405060708...
a80148fbc9a100e0...
0081ff63abccffff...
00001fff78abcfff...

Graphviz

```
_100c001a0 [label="malloc 0x100c001a0 (16 bytes)
404fc00001000000
303ec00001000000"]
_100c001a0 -> _100c04f40 [label="@0"]
_100c001a0 -> _100c03e30 [label="@8"]
_100c033f0 [label="malloc 0x100c033f0 (16 bytes)
404fc00001000000
1a000c1000000020"]
_100c033f0 -> _100c04f40 [label="@0"]
_100c03e30 [label="malloc 0x100c03e30 (16 bytes)
536f6d657468696e
670048656c6c6f21
Strings:
Something
Hello!"]
```

Output


unknown 0x7fff5fbff260 (8 bytes)
802dc00001000000
@0
malloc 0x100c02d80 (16 bytes)
f033c00001000000
a001c00001000000
@8
@0
malloc 0x100c001a0 (16 bytes)
404fc00001000000
303ec00001000000
malloc 0x100c033f0 (16 bytes)
404fc00001000000
1a000c1000000020
@8
@0
@0
malloc 0x100c03e30 (16 bytes)
536f6d657468696e
670048656c6c6f21
Strings:
Something
Hello!
malloc 0x100c04f40 (16 bytes)
4c65616620686572
65000c10000000f0
Strings:
Leaf here

Memory Layouts

- Arch-specific (these are x86-64)
- Swift stuff depends on the compiler version
- Offsets, sizes, contents, meaning subject to change
- Still useful for debugging, general knowledge of how things work

C structs

```
struct S {
    long x;
    long y;
    long z;
};
S s = { 1, 2, 3 };
```

unknown 0x7fff5fbff2a0 (24 bytes)
0100000000000000
0200000000000000
0300000000000000

C structs

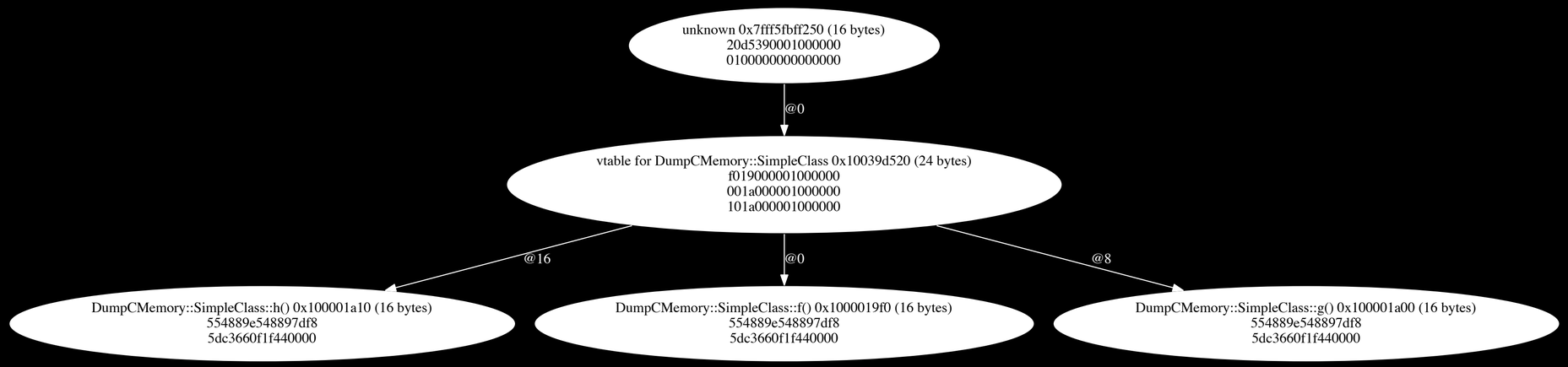
```
struct WithPadding {
    char a;
    char b;
    char c;
    short d;
    char e;
    int f;
    char g;
    long h;
};
WithPadding withPadding =
    { 1, 2, 3, 4, 5, 6, 7, 8 };
```

unknown 0x7fff5fbff288 (24 bytes)
0102030004000500
0600000007000000
0800000000000000

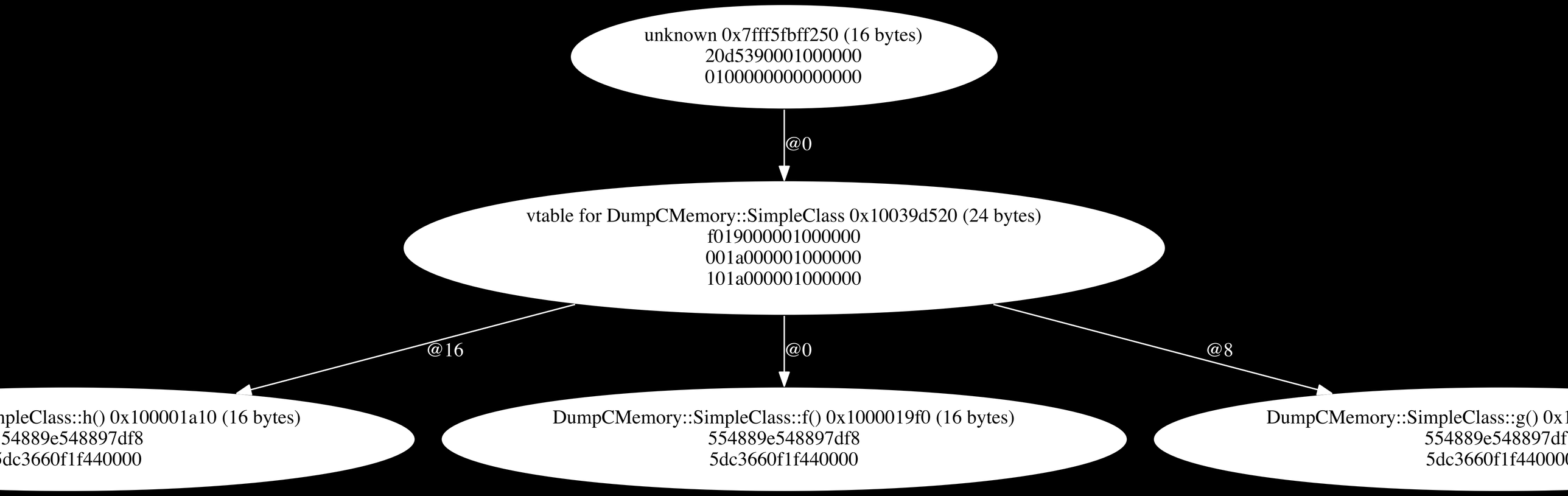
C++ classes

```
class SimpleClass {
public:
    long x;

    virtual void f() {}
    virtual void g() {}
    virtual void h() {}
};

SimpleClass simpleClass;
simpleClass.x = 1;
```

C++ classes


unknown 0x7fff5fbff250 (16 bytes)
20d5390001000000
0100000000000000
@0
vtable for DumpCMemory::SimpleClass 0x10039d520 (24 bytes)
f019000001000000
001a000001000000
101a000001000000
@16
@0
@8
DumpCMemory::SimpleClass::h() 0x100001a10 (16 bytes)
554889e548897df8
5dc3660f1f440000
DumpCMemory::SimpleClass::f() 0x1000019f0 (16 bytes)
554889e548897df8
5dc3660f1f440000
DumpCMemory::SimpleClass::g() 0x100001a00 (16 bytes)
554889e548897df8
5dc3660f1f440000

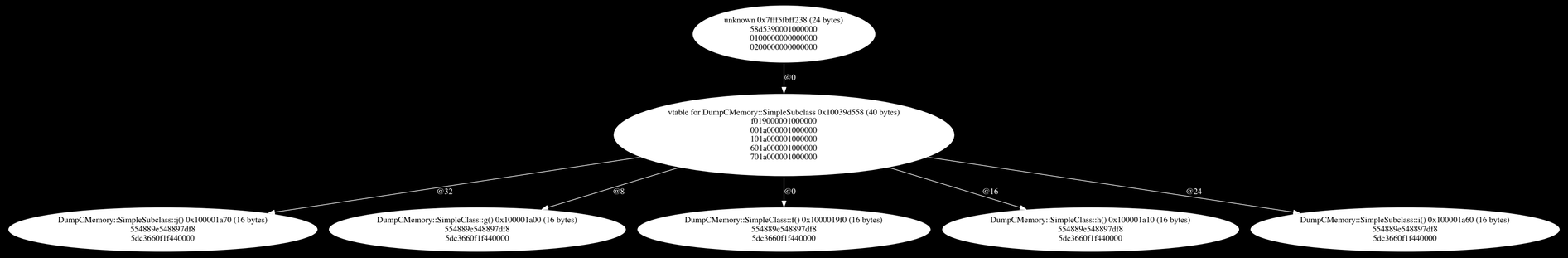
C++ classes


unknown 0x7fff5fbff250 (16 bytes)
20d5390001000000
0100000000000000
@0
vtable for DumpCMemory::SimpleClass 0x10039d520 (24 bytes)
f019000001000000
001a000001000000
101a000001000000
@16
@0
@8
mpleClass::h() 0x100001a10 (16 bytes)
54889e548897df8
5dc3660f1f440000
DumpCMemory::SimpleClass::f() 0x1000019f0 (16 bytes)
554889e548897df8
5dc3660f1f440000
DumpCMemory::SimpleClass::g() 0x1
554889e548897df
5dc3660f1f44000

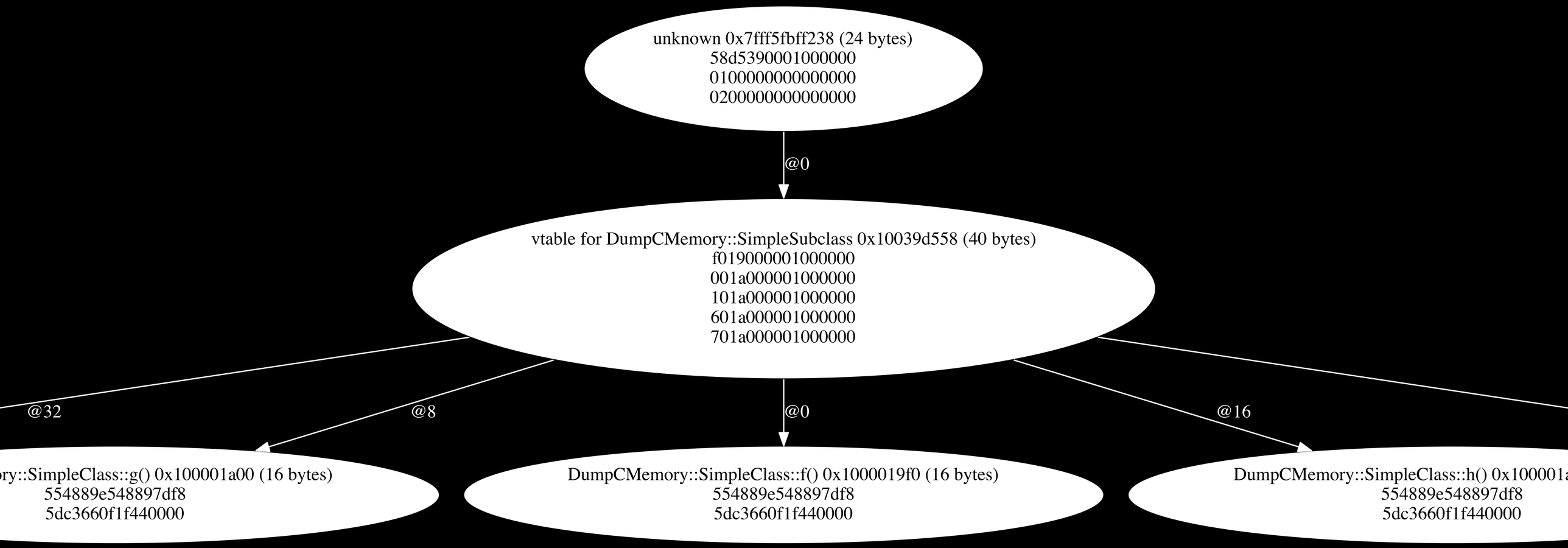
C++ classes

```
class SimpleSubclass: public SimpleClass {
public:
    long y;

    virtual void i() {}
    virtual void j() {}
};

SimpleSubclass simpleSubclass;
simpleSubclass.x = 1;
simpleSubclass.y = 2;
```

C++ classes


unknown 0x7fff5fbff238 (24 bytes)
58d53590001000000
01000000000000000
02000000000000000
@0
vtable for DumpCMemory::SimpleSubclass 0x10039d558 (40 bytes)
f019000001000000
001a000001000000
101a000001000000
601a000001000000
701a000001000000
@32
@8
@0
@16
@24
DumpCMemory::SimpleSubclass::j() 0x100001a70 (16 bytes)
554889e548897df8
5dc3660f1f440000
DumpCMemory::SimpleClass::g() 0x100001a00 (16 bytes)
554889e548897df8
5dc3660f1f440000
DumpCMemory::SimpleClass::f() 0x1000019f0 (16 bytes)
554889e548897df8
5dc3660f1f440000
DumpCMemory::SimpleClass::h() 0x100001a10 (16 bytes)
554889e548897df8
5dc3660f1f440000
DumpCMemory::SimpleSubclass::i() 0x100001a60 (16 bytes)
554889e548897df8
5dc3660f1f440000

C++ classes


unknown 0x7fff5fbff238 (24 bytes)
58d5390001000000
0100000000000000
0200000000000000
@0
vtable for DumpCMemory::SimpleSubclass 0x10039d558 (40 bytes)
f019000001000000
001a000001000000
101a000001000000
601a000001000000
701a000001000000
@32
@8
@0
@16
ory::SimpleClass::g() 0x100001a00 (16 bytes)
554889e548897df8
5dc3660f1f440000
DumpCMemory::SimpleClass::f() 0x1000019f0 (16 bytes)
554889e548897df8
5dc3660f1f440000
DumpCMemory::SimpleClass::h() 0x100001a
554889e548897df8
5dc3660f1f440000

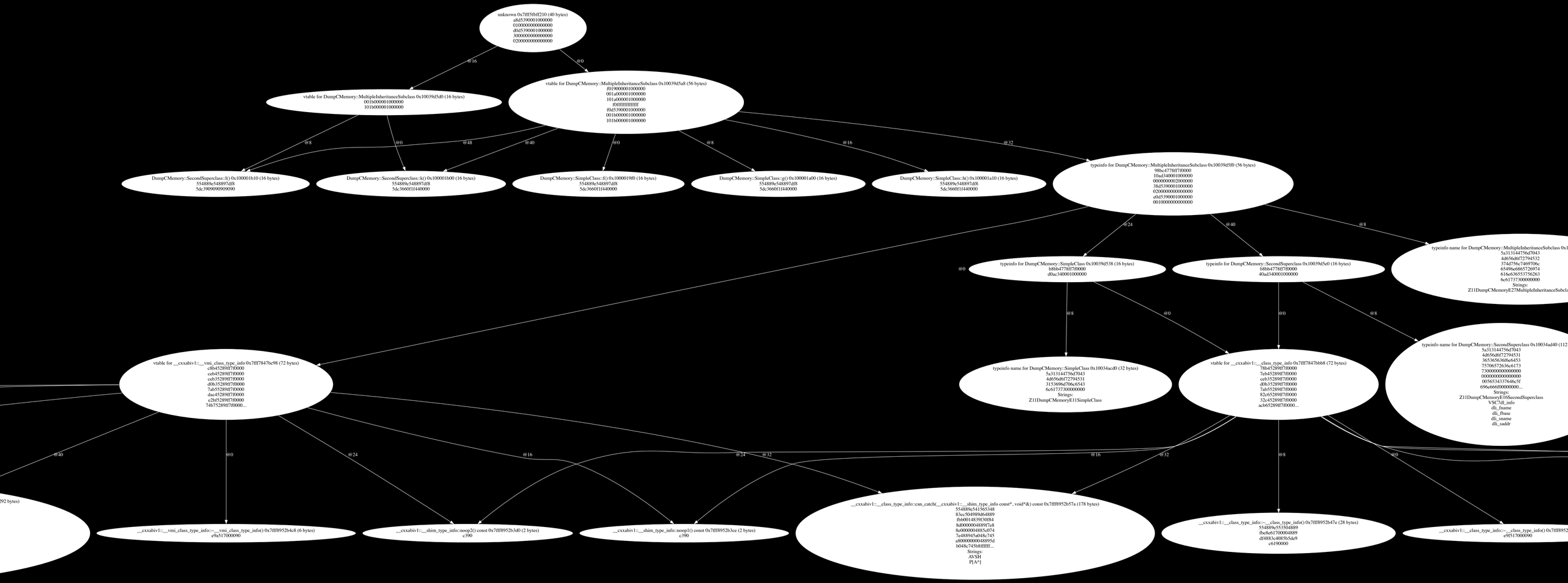
C++ classes

```
class SecondSuperclass {
public:
    long y;

    virtual void k() {}
    virtual void l() {}
};
```

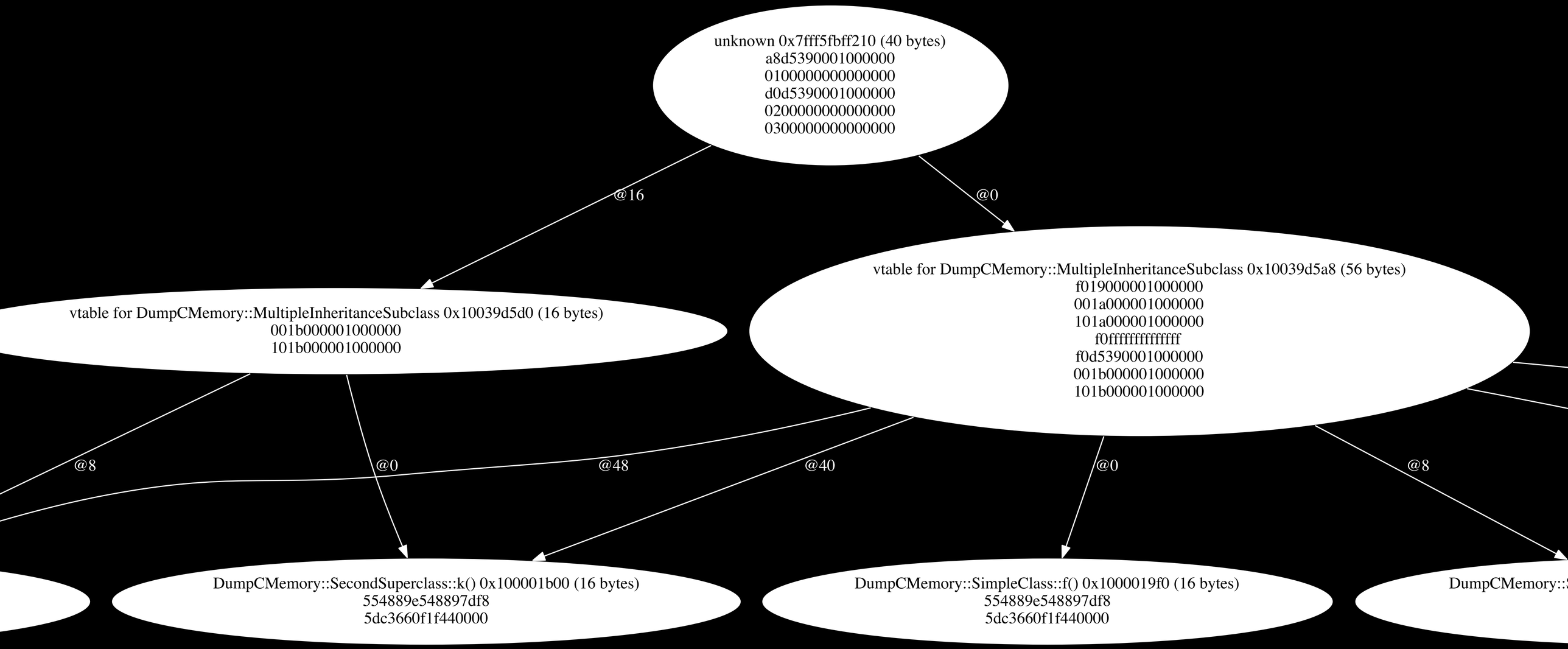
C++ classes

```
class MultipleInheritanceSubclass:
    public SimpleClass, SecondSuperclass {
 public:
     long z;
 };

 MultipleInheritanceSubclass
     multipleInheritanceSubclass;
 multipleInheritanceSubclass.x = 1;
 multipleInheritanceSubclass.y = 2;
 multipleInheritanceSubclass.z = 3;
```

C++ classes


unknown 0x7fff5fbff210 (40 bytes)
a8d5390001000000
0100000000000000
d0d5390001000000
3000000000000000
0200000000000000
@16
@0
vtable for DumpCMemory::MultipleInheritanceSubclass 0x10039d5d0 (16 bytes)
001b000001000000
101b000001000000
vtable for DumpCMemory::MultipleInheritanceSubclass 0x10039d5a8 (56 bytes)
f0190000010000000
001a000001000000
101a000001000000
ffffffffffffffff
f0d5390001000000
001b000001000000
101b000001000000
@8
@0
@48
@40
@0
@8
@16
@32
DumpCMemory::SecondSuperclass::l() 0x100001b10 (16 bytes)
554889c548897df8
5dc390909090909090
DumpCMemory::SecondSuperclass::k() 0x100001b00 (16 bytes)
554889c548897df8
5dc3660f1f1440000
DumpCMemory::SimpleClass::f() 0x100001910 (16 bytes)
554889c548897df8
5dc3660f1f1440000
DumpCMemory::SimpleClass::g() 0x100001a00 (16 bytes)
554889c548897df8
5dc3660f1f1440000
DumpCMemory::SimpleClass::h() 0x100001a10 (16 bytes)
554889c548897df8
5dc3660f1f1440000
typeinfo for DumpCMemory::MultipleInheritanceSubclass 0x10039d5f0 (56 bytes)
98bc4778ff7f0000
10ad340001000000
0000000002000000
38d5390001000000
0200000000000000
e0d5390001000000
001000000000000
@24
@40
@8
typeinfo for DumpCMemory::SimpleClass 0x10039d538 (16 bytes)
b8b64778ff7f0000
d0ac340001000000
typeinfo for DumpCMemory::SecondSuperclass 0x10039d5e0 (16 bytes)
b8b64778ff7f0000
40ad340001000000
typeinfo name for DumpCMemory::MultipleInheritanceSubclass 0x1
5a313144756d7043
4d656d6f72794532
374d756c74697064c
65496e686572697
616e636553756263
6c61737300000000
Strings:
Z11DumpCMemoryE27MultipleInheritanceSubcl
@0
@8
@0
@8
typeinfo name for DumpCMemory::SimpleClass 0x10034acd0 (32 bytes)
5a313144756d7043
4d656d6f72794531
3153696d706c6543
6c617373000000000
Strings:
Z11DumpCMemoryE11SimpleClass
vtable for __cxxabiv1::__class_type_info 0x7fff7847bbb8 (72 bytes)
78b45289ff7f0000
7db45289ff7f0000
ecb35289ff7f0000
d0b35289ff7f0000
7ab55289ff7f0000
82c65289ff7f0000
32c45289ff7f0000
ace65289ff7f0000...
typeinfo name for DumpCMemory::SecondSuperclass 0x10034ad40 (112
5a313144756d7043
4d656d6f72794531
3653656366f6e6453
7570657263fc6173
7300000000000000
0000000000000000
00565343337646c5f
696e666f00000000...
Strings:
Z11DumpCMemoryE16SecondSuperclass
VSC7dl_info
dli_fname
dli_fbase
dli_sname
dli_saddr
vtable for __cxxabiv1::__vmi_class_type_info 0x7fff7847bc98 (72 bytes)
e8b45289ff7f0000
ceb45289ff7f0000
ceb35289ff7f0000
d0b35289ff7f0000
7ab55289ff7f0000
dac45289ff7f0000
e2b45289ff7f0000
74b75289ff7f0000...
@40
@0
@24
@16
@24
@32
@16
@32
@8
@0
292 bytes)
__cxxabiv1::__vmi_class_type_info::~__vmi_class_type_info() 0x7fff8952b4c8 (6 bytes)
e9a517000090
__cxxabiv1::__shim_type_info::noop2() const 0x7fff8952b3d0 (2 bytes)
c390
__cxxabiv1::__shim_type_info::noop1() const 0x7fff8952b3ce (2 bytes)
c390
__cxxabiv1::__class_type_info::can_catch(__cxxabiv1::__shim_type_info const*, void*&) const 0x7fff8952b57a (178 bytes)
554889e54156534
83ec5049489d64889
fbb0014839f30f84
8d000000488997c8
8e0000004885c074
7e488945a048c745
a80000000048895d
b048c745b8ffffff...
Strings:
AVSH
P[A^]
__cxxabiv1::__class_type_info::~__class_type_info() 0x7fff8952b47e (28 bytes)
554889e553504889
fbe8e617000048b9
df4883c4085b5de9
c6190000
__cxxabiv1::__class_type_info::~__class_type_info() 0x7fff8952
e9517000090

C++ classes


unknown 0x7fff5fbff210 (40 bytes)
a8d5390001000000
0100000000000000
d0d5390001000000
0200000000000000
0300000000000000
@16
@0
vtable for DumpCMemory::MultipleInheritanceSubclass 0x10039d5d0 (16 bytes)
001b000001000000
101b000001000000
vtable for DumpCMemory::MultipleInheritanceSubclass 0x10039d5a8 (56 bytes)
f019000001000000
001a000001000000
101a000001000000
f0ffffffffffffff
f0d5390001000000
001b000001000000
101b000001000000
@8
@0
@48
@40
@0
@8
DumpCMemory::SecondSuperclass::k() 0x100001b00 (16 bytes)
554889e548897df8
5dc3660f1f440000
DumpCMemory::SimpleClass::f() 0x1000019f0 (16 bytes)
554889e548897df8
5dc3660f1f440000

Swift Types

```
struct EmptyStruct {}
```

unknown 0x7fff5fbff2a0 (0 bytes)

Swift Types

```
struct SimpleStruct {
    var x: Int = 1
    var y: Int = 2
    var z: Int = 3
}
```

unknown 0x7fff5fbff2a0 (24 bytes)
0100000000000000
0200000000000000
0300000000000000

Swift Types

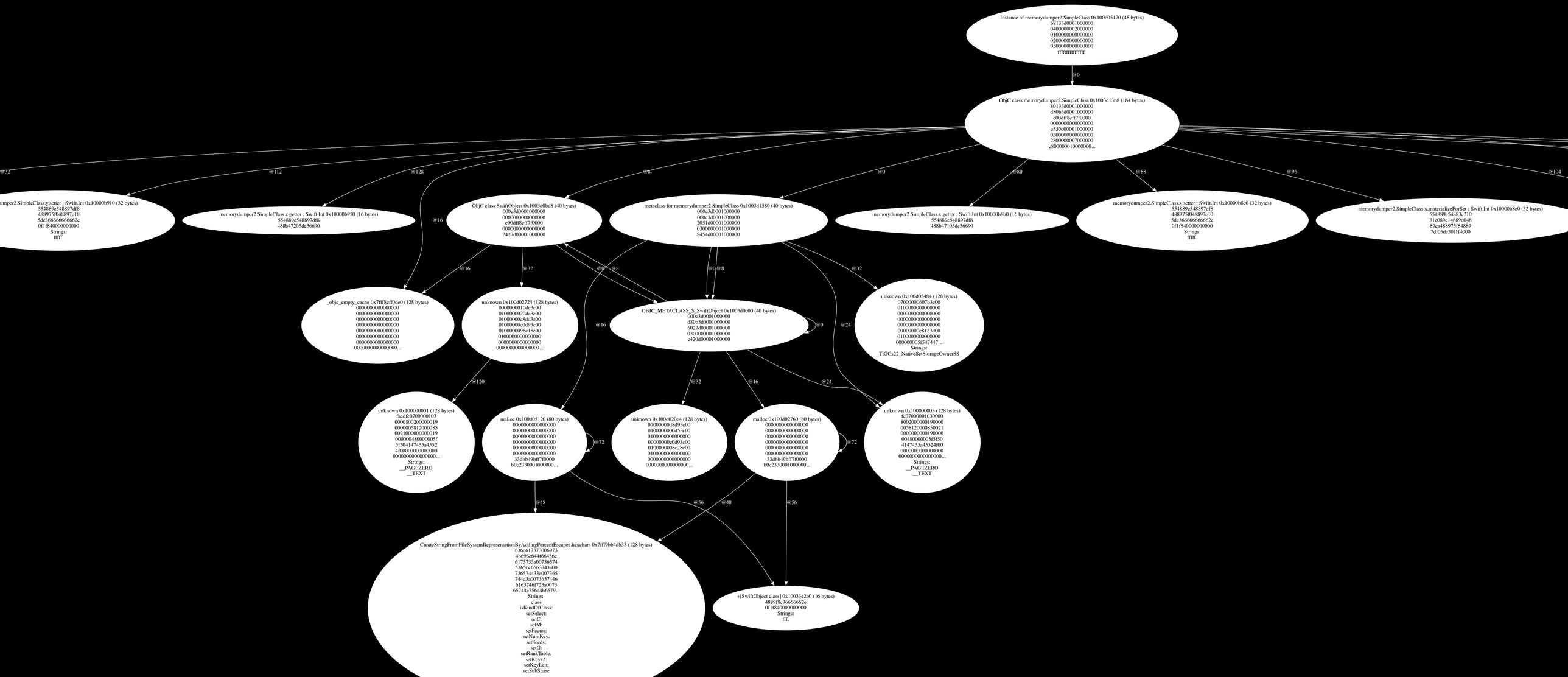
```
struct StructWithPadding {
    var a: UInt8 = 1
    var b: UInt8 = 2
    var c: UInt8 = 3
    var d: UInt16 = 4
    var e: UInt8 = 5
    var f: UInt32 = 6
    var g: UInt8 = 7
    var h: UInt64 = 8
}
```

unknown 0x7fff5fbff2a0 (24 bytes)
0102035f04000500
06000000077f0000
0800000000000000

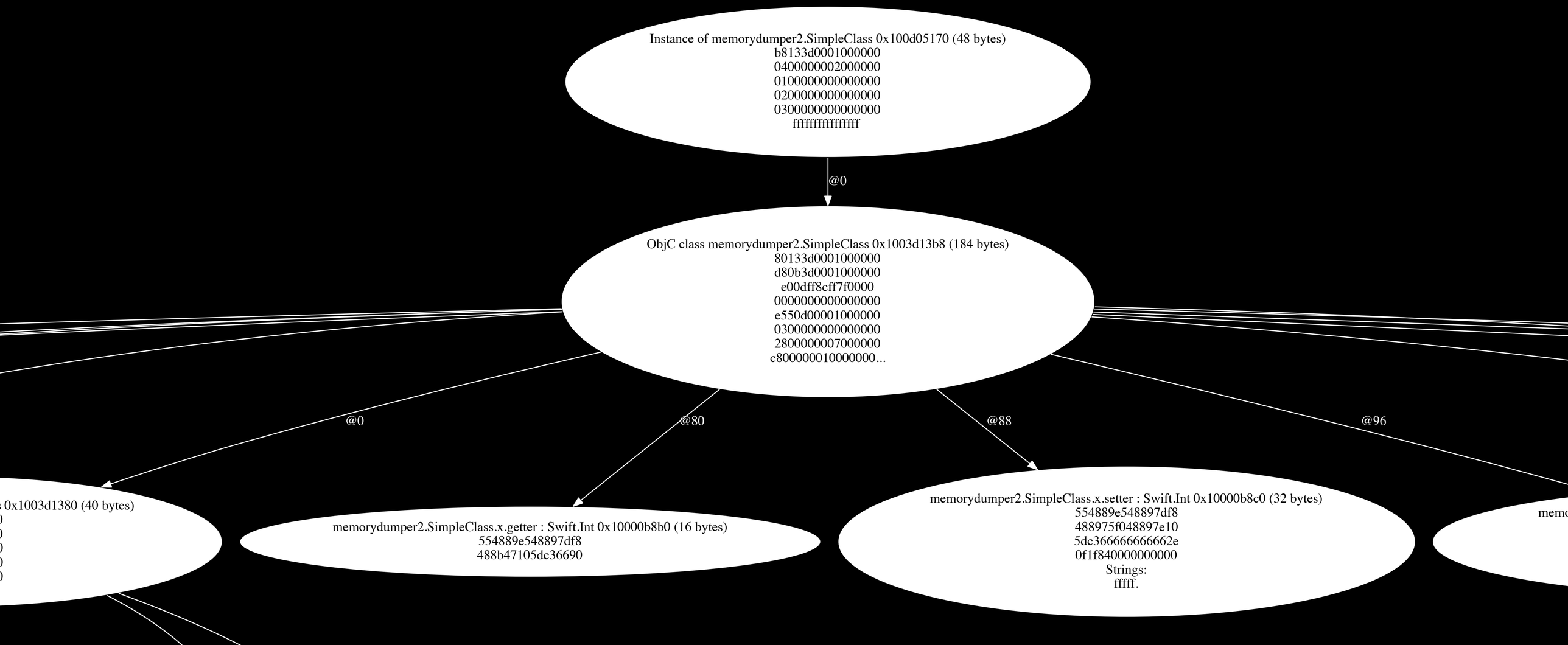
Swift Types

```
class SimpleClass {
    var x: Int = 1
    var y: Int = 2
    var z: Int = 3
}
```

Swift Types

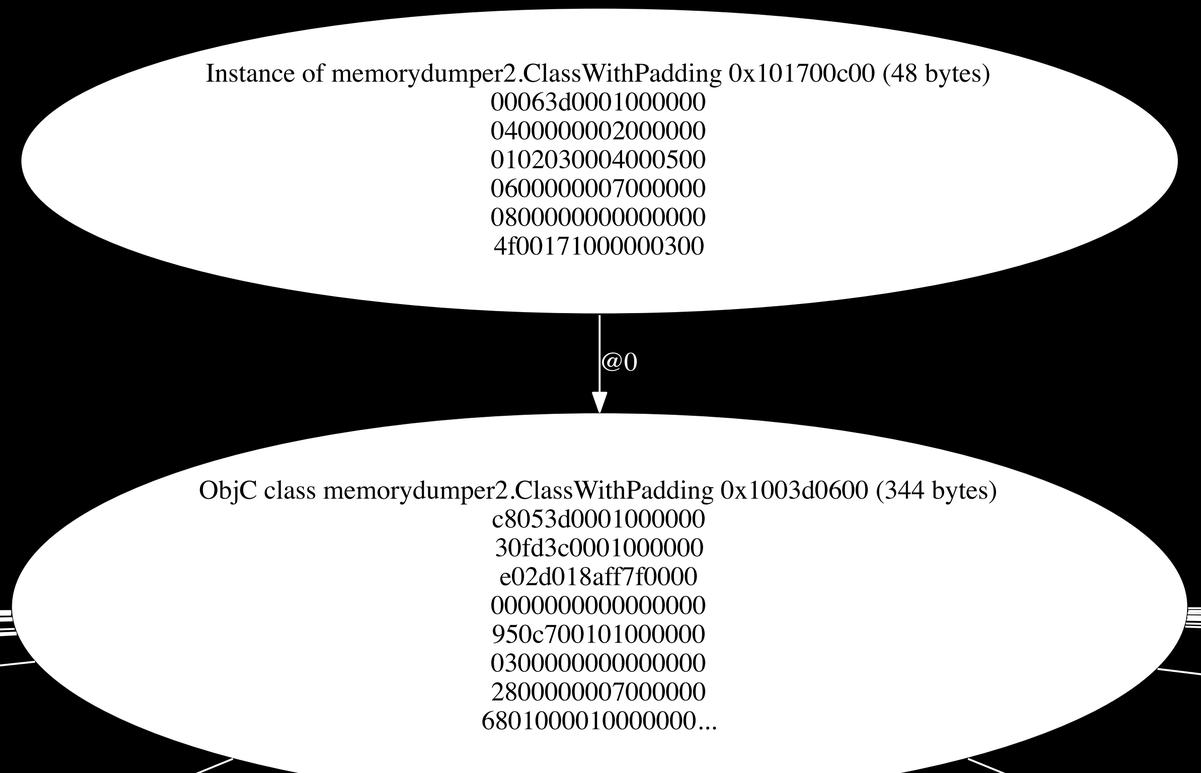

Instance of memorydumper2.SimpleClass 0x100d05170 (48 bytes)
b8133d0001000000
0400000002000000
0100000000000000
0200000000000000
0300000000000000
ffffffffffffffff
@0
ObjC class memorydumper2.SimpleClass 0x1003d13b8 (184 bytes)
80133d0001000000
d80b3d0001000000
e00df8fcff7f0000
0000000000000000
e550d00001000000
0300000000000000
2800000007000000
e800000010000000...
@32
@112
@128
@8
@0
@80
@88
@96
@104
@16
@120
@72
@48
@56
@24
memorydumper2.SimpleClass.y.setter : Swift.Int 0x10000b910 (32 bytes)
554889e548897df8
488975f0488897e18
5dc366666666662e
0f1f840000000000
Strings:
fffff.
memorydumper2.SimpleClass.z.getter : Swift.Int 0x10000b950 (16 bytes)
554889e548897df8
488b47205dc36690
ObjC class SwiftObject 0x1003d0bd8 (40 bytes)
000c3d0001000000
0000000000000000
e00df8fcff7f0000
0000000000000000
2427d00001000000
metaclass for memorydumper2.SimpleClass 0x1003d1380 (40 bytes)
000c3d0001000000
000c3d0001000000
2051d0000100000
0300000001000000
8454d00001000000
memorydumper2.SimpleClass.x.getter : Swift.Int 0x100000b8b0 (16 bytes)
554889e548897df8
488b47105dc36690
memorydumper2.SimpleClass.x.setter : Swift.Int 0x10000b8c0 (32 bytes)
554889e548897df8
488975f0488897e10
5dc366666666662e
0f1f840000000000
Strings:
fffff.
memorydumper2.SimpleClass.x.materializeForSet : Swift.Int 0x10000b8e0 (32 bytes)
554889e54883c210
31c088c148880d48
89ca488975f84889
7df05dc30f1f4000
_objc_empty_cache 0x7fff8cff0de0 (128 bytes)
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000...
unknown 0x100d02724 (128 bytes)
0000000010de3c00
0100000020da3c00
01000000c8dd3c00
01000000c0d93c00
0100000098c18c00
0100000000000000
0000000000000000
0000000000000000...
OBJC_METACLASS_$_SwiftObject 0x1003d0c00 (40 bytes)
000c3d0001000000
d80b3d0001000000
6027d00001000000
0300000001000000
c420d00001000000
unknown 0x100d05484 (128 bytes)
070000006071b3c00
0100000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
00000000c8123d00
0100000000000000
000000005f547447...
Strings:
_TtGCs22_NativeSetStorageOwnerSS_
unknown 0x100000001 (128 bytes)
facdfc07000000103
0000800200000019
000000581200085
0021000000000019
0000004800000005f
5f504147455a4552
4f000000000000000
0000000000000000...
Strings:
__PAGEZERO
__TEXT
malloc 0x100d05120 (80 bytes)
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
33dbb49bff7f0000
b0e2330001000000...
unknown 0x100d020c4 (128 bytes)
070000000d8d93c00
0100000000d53c00
0100000000000000
0000000000c0d93c00
0100000008c28e00
0100000000000000
0000000000000000
0000000000000000...
malloc 0x100d02760 (80 bytes)
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
33dbb49bff7f0000
b0e2330001000000...
unknown 0x100000003 (128 bytes)
fs070000001030000
8002000000190000
005812000085002 1
0000000000190000
0048000005f5f5f50
4147455a4552 4f00
0000000000000000
0000000000000000...
Strings:
__PAGEZERO
__TEXT
CreateStringFromFileSystemRepresentationBy AddingPercentEscapes.hexchars 0x7fff9bb4db33 (128 bytes)
636c617373006973
4b696e6644f66436c
6173733a00736574
5365e656374a00
73657443 3a007365
744d3a007365 7446
6163746f723a0073
65744e756d4b6579...
Strings:
class
isKindOfClass:
setSelect:
setC:
setM:
setFactor:
setNumKey:
setSeeds:
setG:
setRankTable:
setKeys2:
setKeyLen:
setSubShare
+[SwiftObject class] 0x10033e2b0 (16 bytes)
48898c366666662e
0f1f840000000000
Strings:
fff.

Swift Types


Instance of memorydumper2.SimpleClass 0x100d05170 (48 bytes)
b8133d0001000000
0400000002000000
0100000000000000
0200000000000000
0300000000000000
ffffffffffffffff
@0
ObjC class memorydumper2.SimpleClass 0x1003d13b8 (184 bytes)
80133d0001000000
d80b3d0001000000
e00dff8cff7f0000
0000000000000000
e550d00001000000
0300000000000000
2800000007000000
c800000010000000...
@0
@80
@88
@96
0x1003d1380 (40 bytes)
memorydumper2.SimpleClass.x.getter : Swift.Int 0x10000b8b0 (16 bytes)
554889e548897df8
488b47105dc36690
memorydumper2.SimpleClass.x.setter : Swift.Int 0x10000b8c0 (32 bytes)
554889e548897df8
488975f048897e10
5dc366666666662e
0f1f840000000000
Strings:
fffff.

Swift Types

```
class ClassWithPadding {
    var a: UInt8 = 1
    var b: UInt8 = 2
    var c: UInt8 = 3
    var d: UInt16 = 4
    var e: UInt8 = 5
    var f: UInt32 = 6
    var g: UInt8 = 7
    var h: UInt64 = 8
}
```



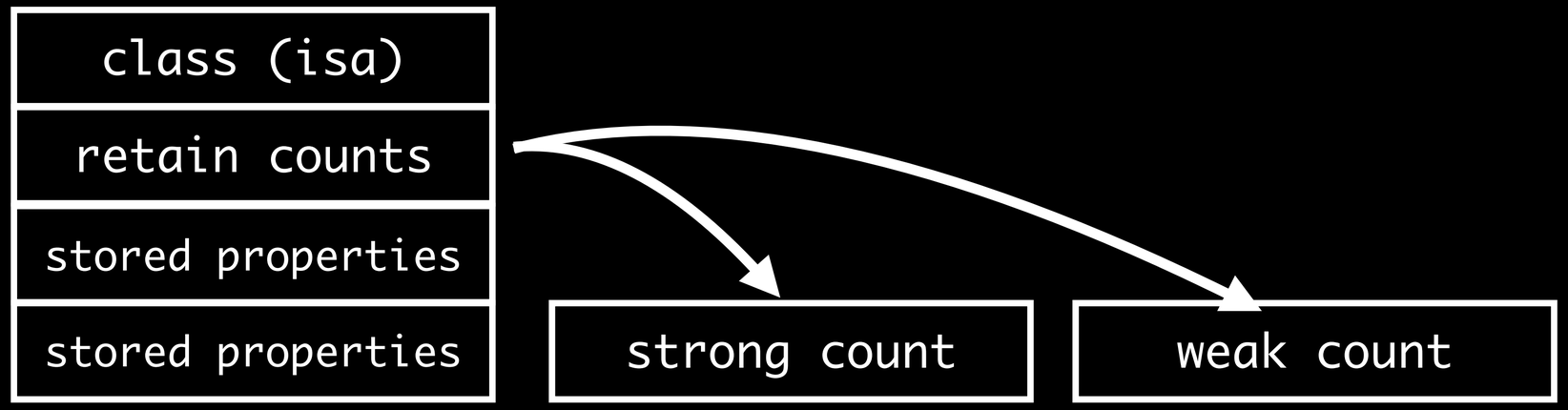
Swift Types

Object

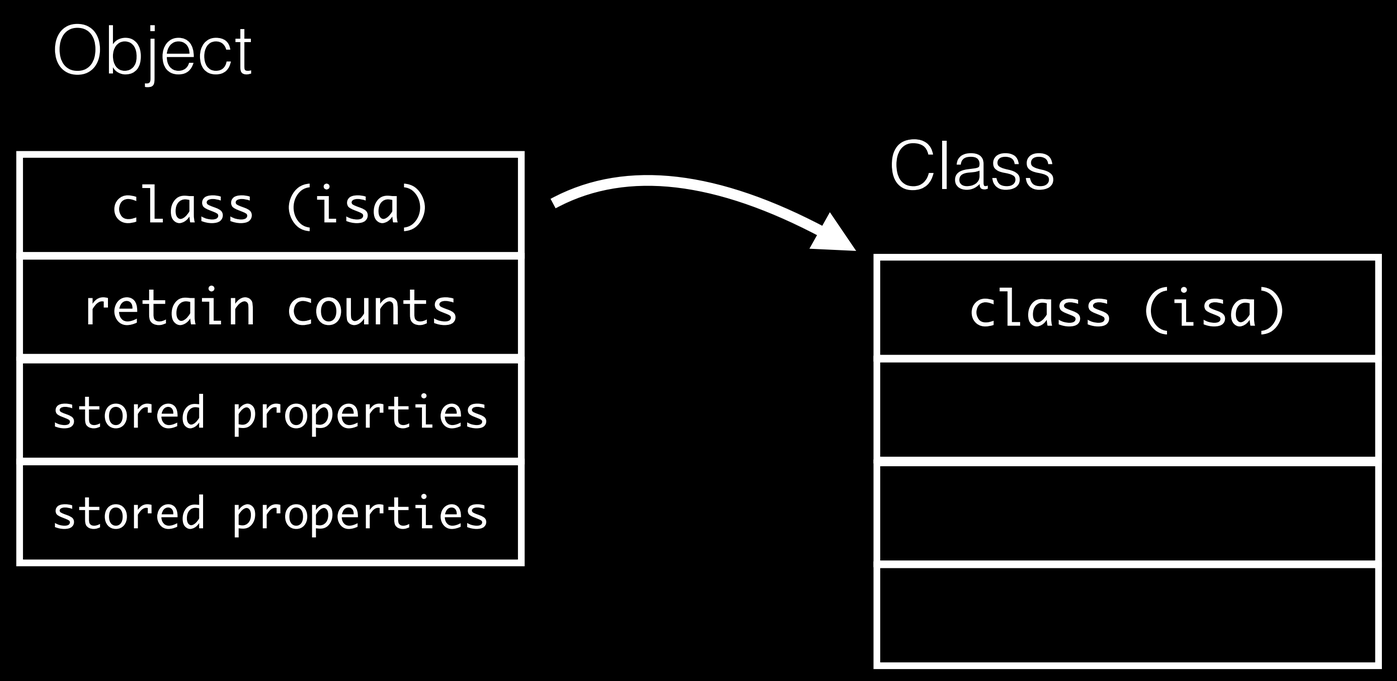
class (isa)
retain counts
stored properties
stored properties

Swift Types

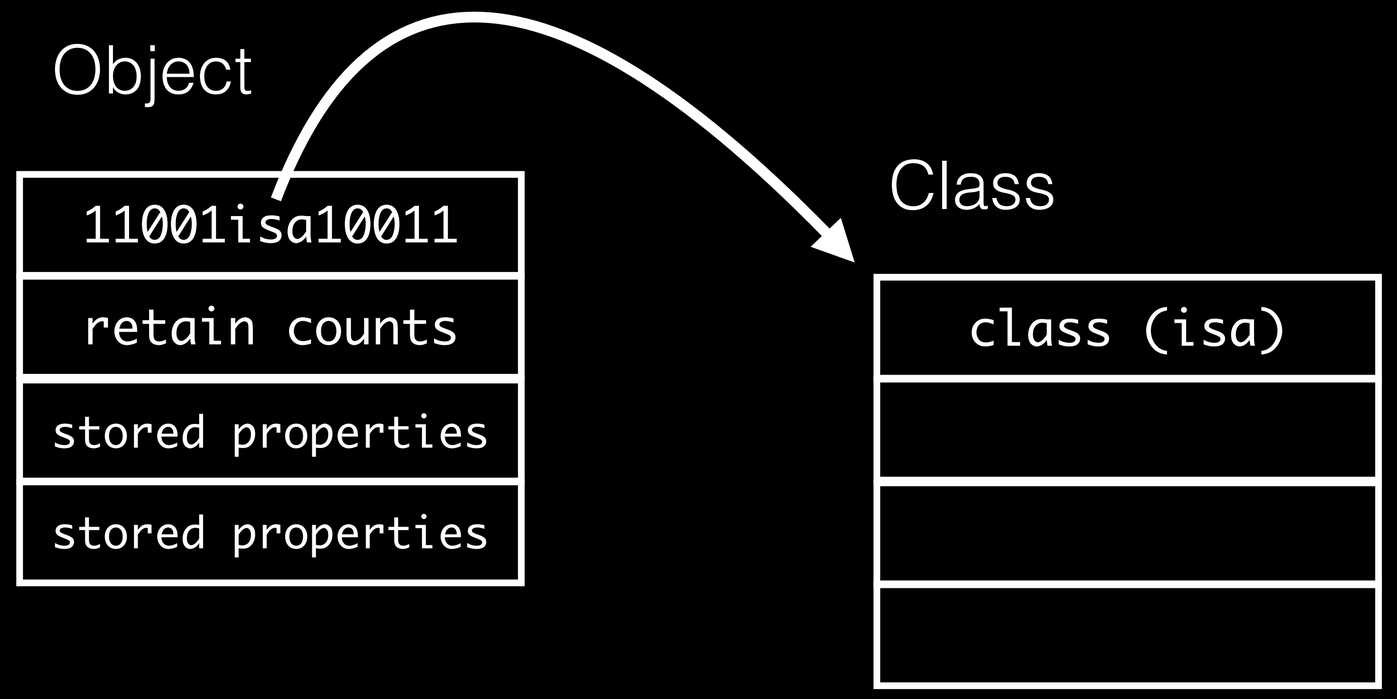
Object



Swift Types


Object
class (isa)
retain counts
stored properties
stored properties
Class
class (isa)

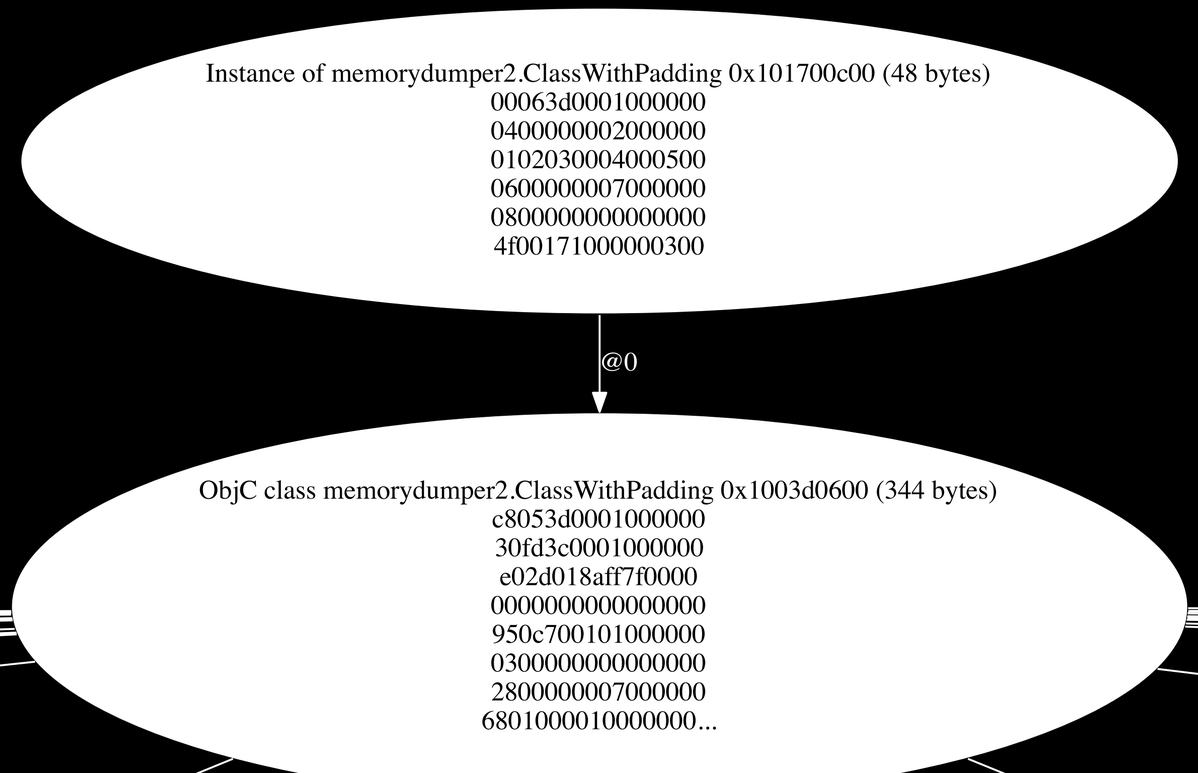
Swift Types


Object
11001isa10011
retain counts
stored properties
stored properties
Class
class (isa)

Objective-C Classes

http://opensource.apple.com/source/objc4/

`runtime.h`



Objective-C Classes

```
Class isa
Class super_class
const char *name
long version
long info
long instance_size
struct objc_ivar_list *ivars
struct objc_method_list **methodLists
struct objc_cache *cache
struct objc_protocol_list *protocols
```

Swift Classes

```
uint32_t flags;
uint32_t instanceAddressOffset;
uint32_t instanceSize;
uint16_t instanceAlignMask;
uint16_t reserved;

uint32_t classSize;
uint32_t classAddressOffset;
void *description;
```

Swift Classes

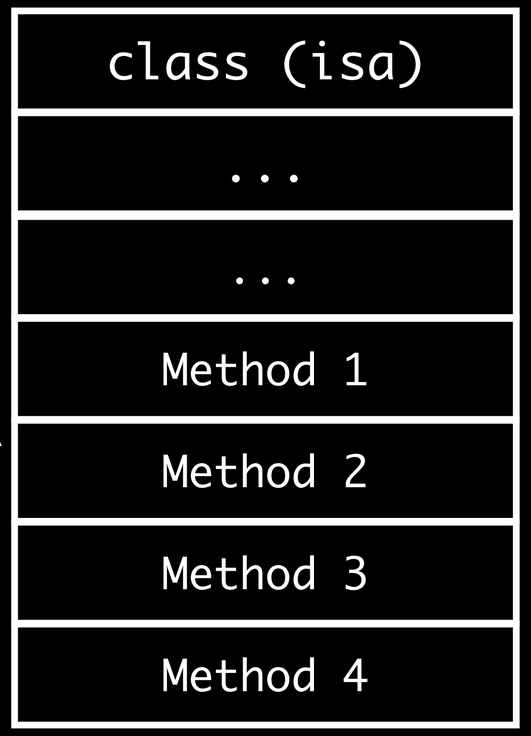
Class

class (isa)
...
...
Method 1
Method 2
Method 3
Method 4

Method Calls

```
obj.method()
```

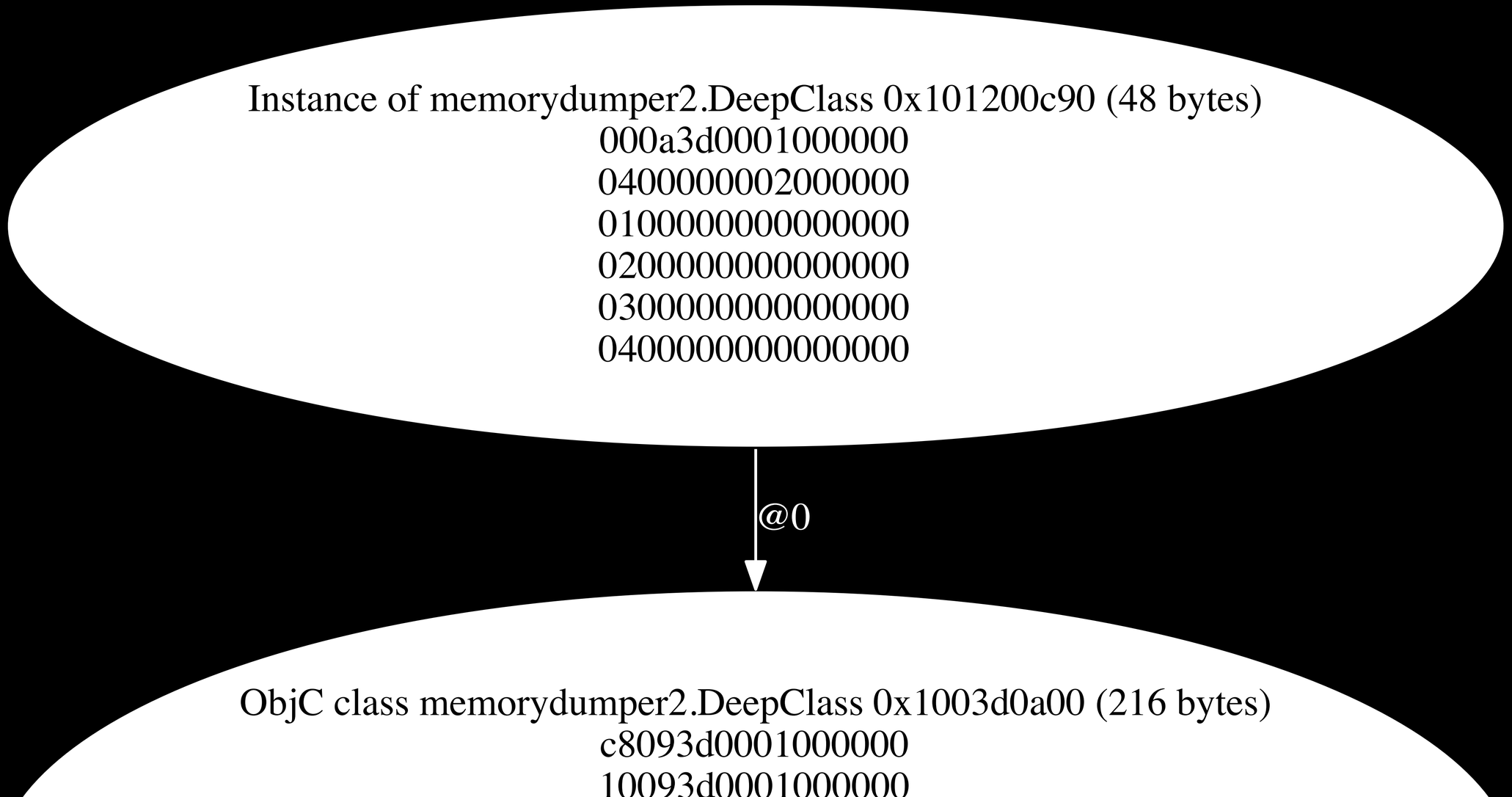
```
class = obj->isa
methodf = class[method_offset]
methodf(obj)
```



Subclasses

```
class DeepClassSuper1 {
    var a = 1
}
class DeepClassSuper2: DeepClassSuper1 {
    var b = 2
}
class DeepClassSuper3: DeepClassSuper2 {
    var c = 3
}
class DeepClass: DeepClassSuper3 {
    var d = 4
}
```

Subclasses

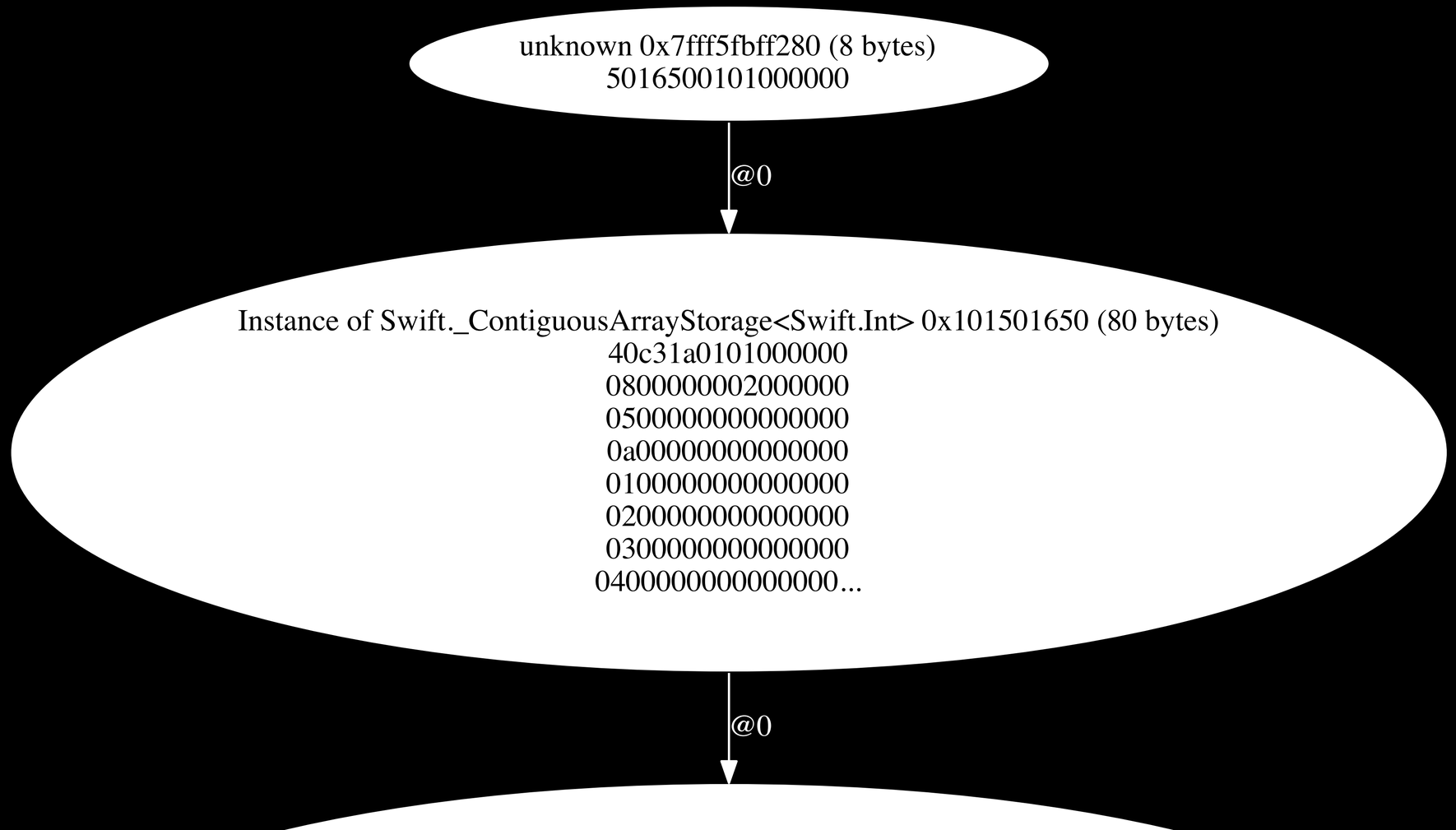

Instance of memorydumper2.DeepClass 0x101200c90 (48 bytes)
000a3d0001000000
0400000002000000
0100000000000000
0200000000000000
0300000000000000
0400000000000000
@0
ObjC class memorydumper2.DeepClass 0x1003d0a00 (216 bytes)
c8093d0001000000
10093d0001000000

Arrays

```
[1, 2, 3, 4, 5]
```

Arrays

```
[1, 2, 3, 4, 5]
```



Protocols

```
protocol P {
    func f()
    func g()
    func h()
}
```

```
struct ProtocolHolder {
    var a: P
    var b: P
    var c: P
}
```

Protocols

```
struct StructSmallP: P {
    func f() {}
    func g() {}
    func h() {}
    var a = 0x6c6c616d73 // "small"
}
```

Protocols

```
struct StructBigP: P {
    func f() {}
    func g() {}
    func h() {}
    var a = 0x656772616c // "large"
    var b = 0x1010101010101010
    var c = 0x2020202020202020
    var d = 0x3030303030303030
}
```

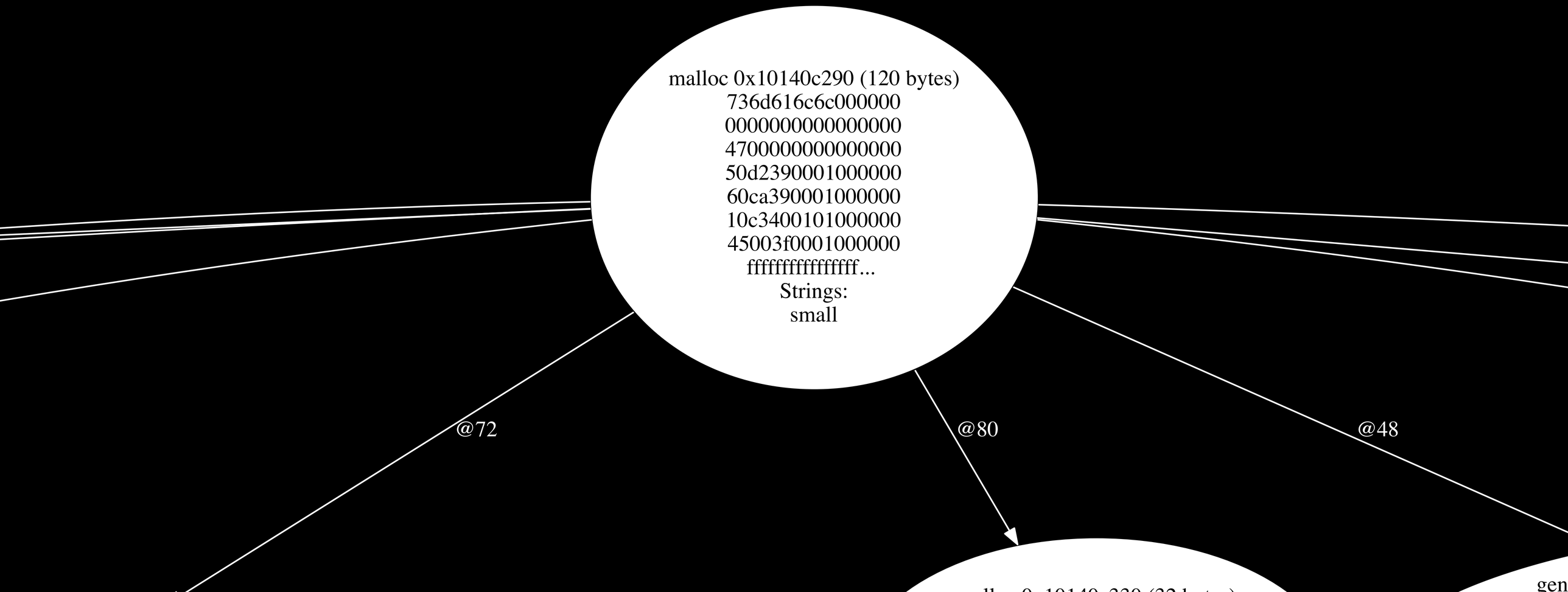
Protocols

```
struct ClassP: P {
    func f() {}
    func g() {}
    func h() {}
    var a = 0x7373616c63 // "class"
    var b = 0x4040404040404040
    var c = 0x5050505050505050
    var d = 0x6060606060606060
}
```

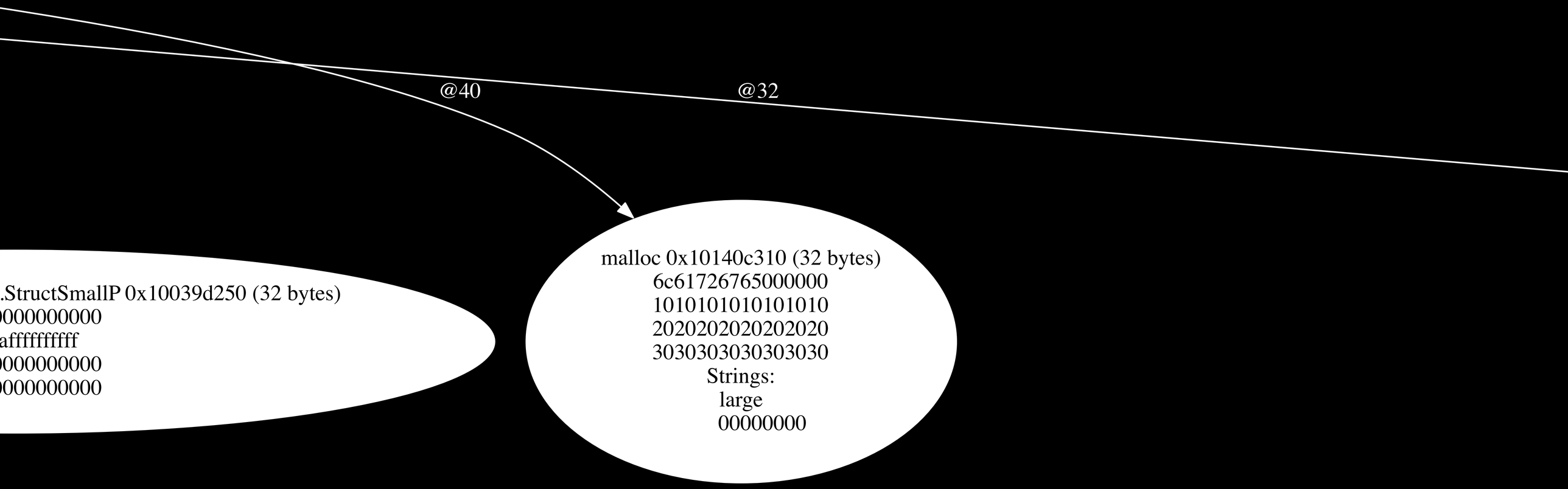
Protocols

```
let holder = ProtocolHolder(
    a: StructSmallP(),
    b: StructBigP(),
    c: ClassP())
```

Protocols


malloc 0x10140c290 (120 bytes)
736d616c6c000000
0000000000000000
4700000000000000
50d2390001000000
60ca390001000000
10c3400101000000
45003f0001000000
ffffffffffffffff...
Strings:
small
@72
@80
@48

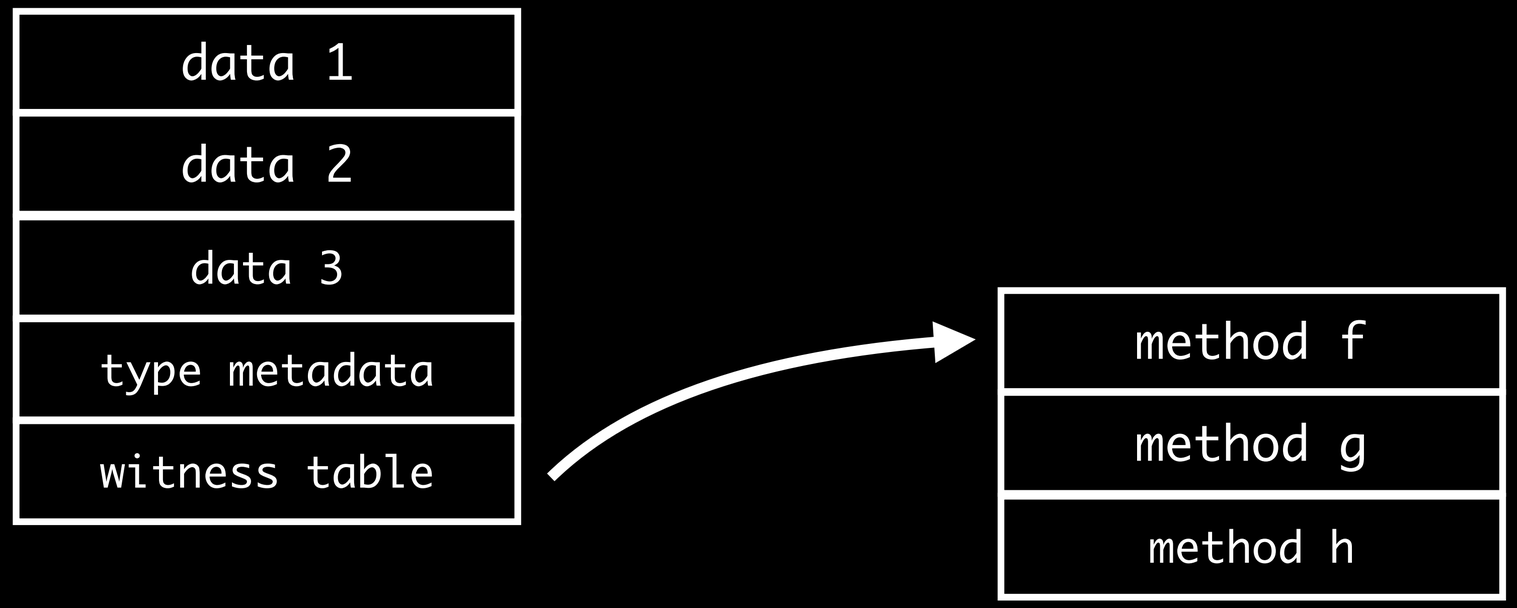
Protocols


@40
@32
.StructSmallP 0x10039d250 (32 bytes)
0000000000
affffffffff
0000000000
0000000000
malloc 0x10140c310 (32 bytes)
6c61726765000000
1010101010101010
2020202020202020
3030303030303030
Strings:
large
00000000

Protocols

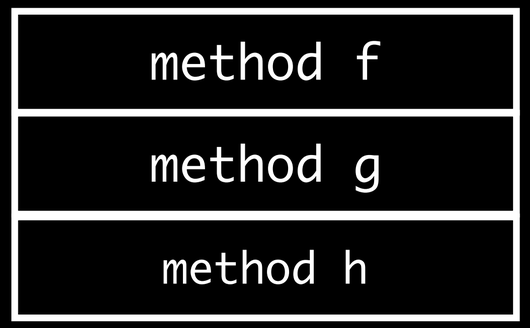
data 1
data 2
data 3
type metadata
witness table

Protocols

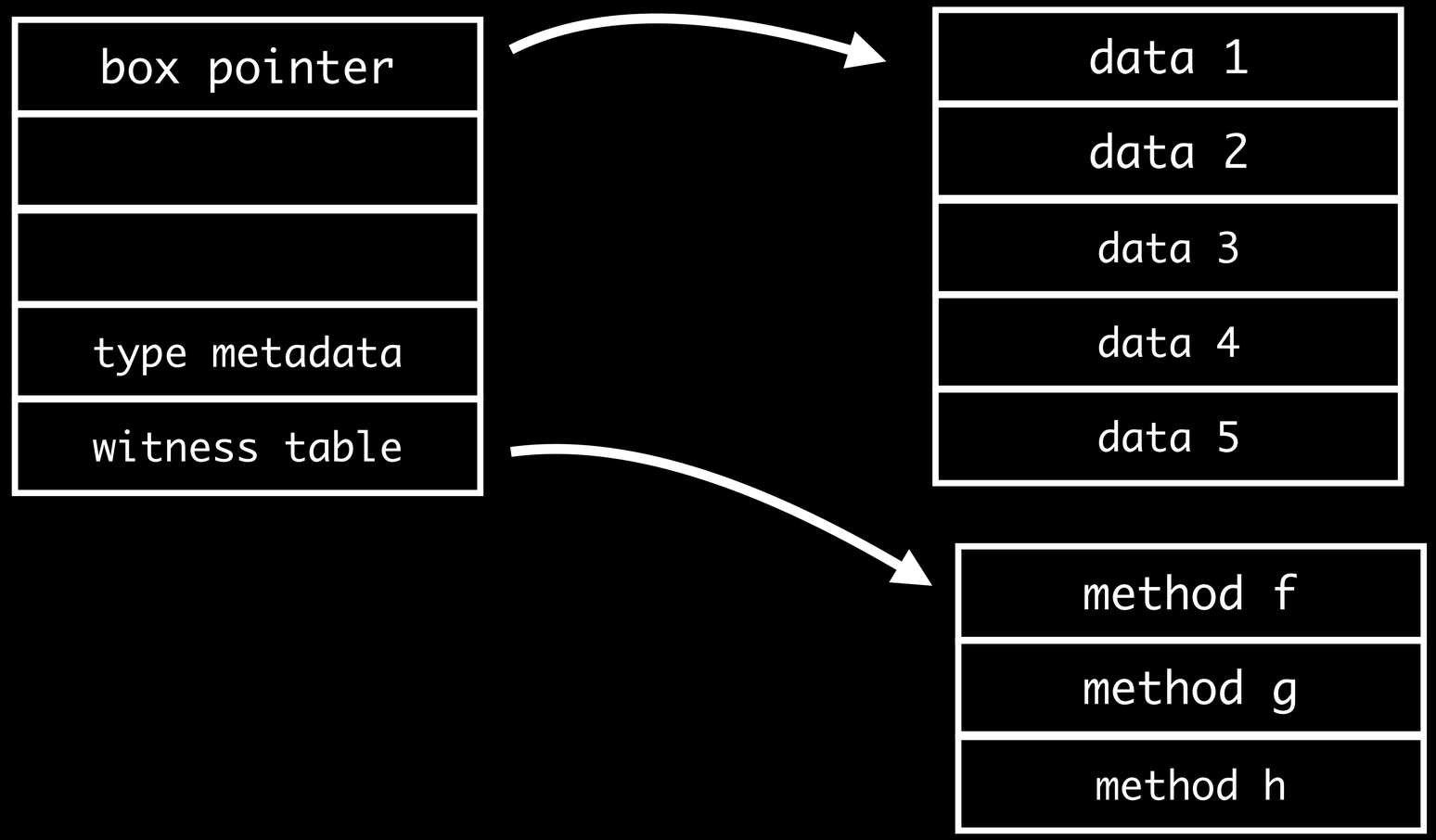

data 1
data 2
data 3
type metadata
witness table
method f
method g
method h

Protocol Method Call

```
let p: Protocol = …
p.g()

table = p[4]
methodf = table[offset]
methodf(p)
```



Protocols


box pointer
type metadata
witness table
data 1
data 2
data 3
data 4
data 5
method f
method g
method h

Enums

```
enum SimpleEnum {
    case A, B, C, D, E
}
struct SimpleEnumHolder {
    var a: SimpleEnum
    var b: SimpleEnum
    var c: SimpleEnum
    var d: SimpleEnum
    var e: SimpleEnum
}
SimpleEnumHolder(a: .A, b: .B,
                 c: .C, d: .D, e: .E)
```

Enums

```
enum SimpleEnum {
    case A, B, C, D, E
}
struct SimpleEnumHolder {
    var a: SimpleEnum
    var b: SimpleEnum
    var c: SimpleEnum
    var d: SimpleEnum
    var e: SimpleEnum
}
SimpleEnumHolder(a: .A, b: .B,
                 c: .C, d: .D, e: .E)
```

unknown 0x7fff5fbff250 (5 bytes)
0001020304

Enums

```
enum IntRawValueEnum: Int {
    case A = 1, B, C, D, E
}
struct IntRawValueEnumHolder {
    var a: IntRawValueEnum
    var b: IntRawValueEnum
    var c: IntRawValueEnum
    var d: IntRawValueEnum
    var e: IntRawValueEnum
}
IntRawValueEnumHolder(a: .A, b: .B,
                      c: .C, d: .D, e: .E)
```

Enums

```
enum IntRawValueEnum: Int {
    case A = 1, B, C, D, E
}
struct IntRawValueEnumHolder {
    var a: IntRawValueEnum
    var b: IntRawValueEnum
    var c: IntRawValueEnum
    var d: IntRawValueEnum
    var e: IntRawValueEnum
}
IntRawValueEnumHolder(a: .A, b: .B,
                      c: .C, d: .D, e: .E)
```

unknown 0x7fff5fbff210 (5 bytes)
0001020304

Enums

```
enum StringRawValueEnum: String {
    case A = "whatever", B, C, D, E
}
struct StringRawValueEnumHolder {
    var a: StringRawValueEnum
    var b: StringRawValueEnum
    var c: StringRawValueEnum
    var d: StringRawValueEnum
    var e: StringRawValueEnum
}
StringRawValueEnumHolder(a: .A, b: .B,
                         c: .C, d: .D, e: .E)
```

Enums

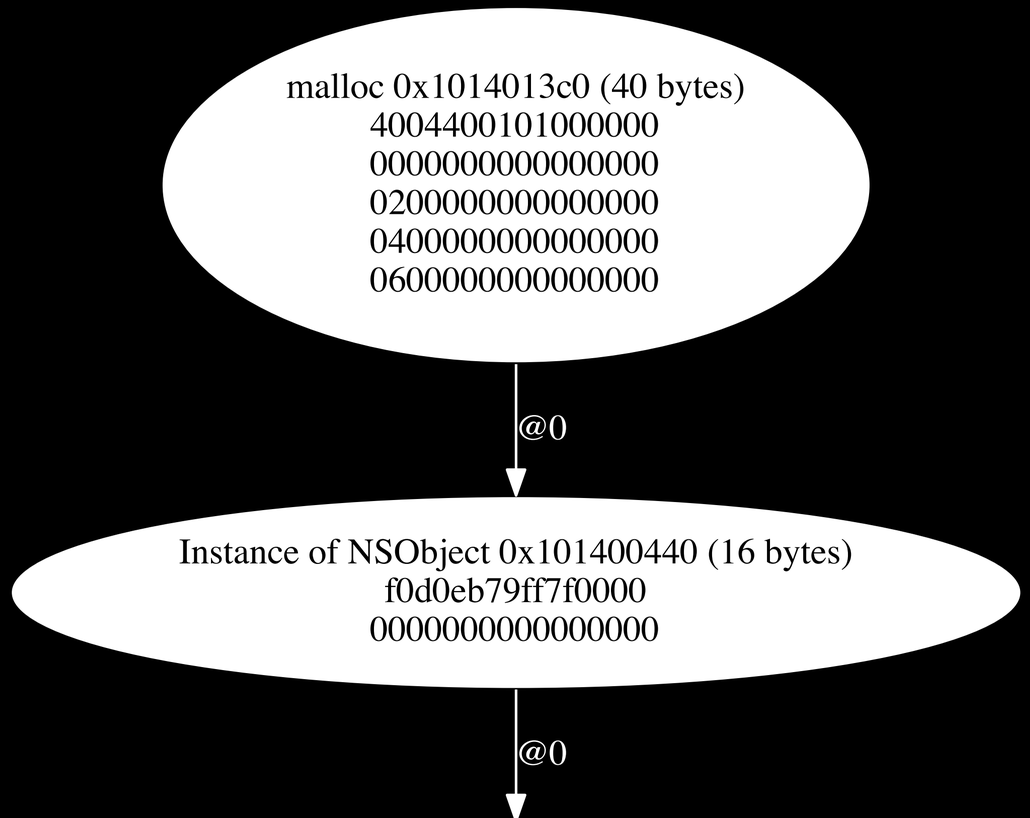
```
enum StringRawValueEnum: String {
    case A = "whatever", B, C, D, E
}
struct StringRawValueEnumHolder {
    var a: StringRawValueEnum
    var b: StringRawValueEnum
    var c: StringRawValueEnum
    var d: StringRawValueEnum
    var e: StringRawValueEnum
}
StringRawValueEnumHolder(a: .A, b: .B,
                         c: .C, d: .D, e: .E)
```

unknown 0x7fff5fbff1f0 (5 bytes)
0001020304

Enums

```
enum OneAssociatedObjectEnum {
    case A(AnyObject)
    case B, C, D, E
}
struct OneAssociatedObjectEnumHolder {
    var a: OneAssociatedObjectEnum
    var b: OneAssociatedObjectEnum
    var c: OneAssociatedObjectEnum
    var d: OneAssociatedObjectEnum
    var e: OneAssociatedObjectEnum
}
OneAssociatedObjectEnumHolder(
    a: .A(NSObject()),
    b: .B, c: .C, d: .D, e: .E)
```

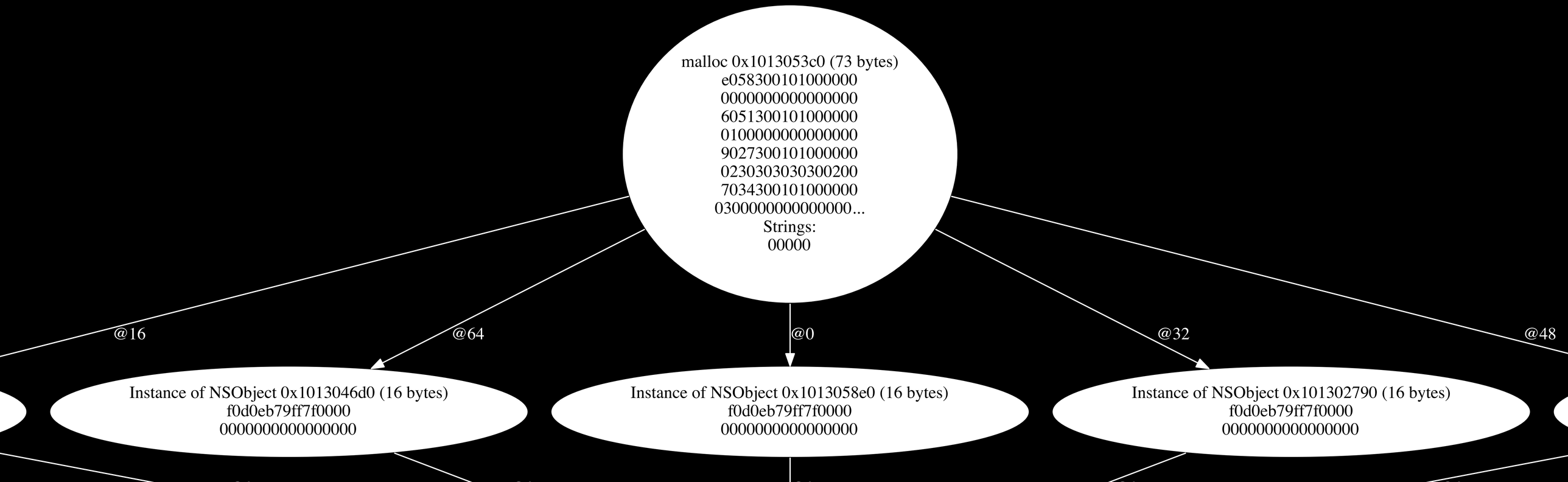
Enums


malloc 0x1014013c0 (40 bytes)
4004400101000000
0000000000000000
0200000000000000
0400000000000000
0600000000000000
@0
Instance of NSObject 0x101400440 (16 bytes)
f0d0eb79ff7f0000
0000000000000000
@0

Enums

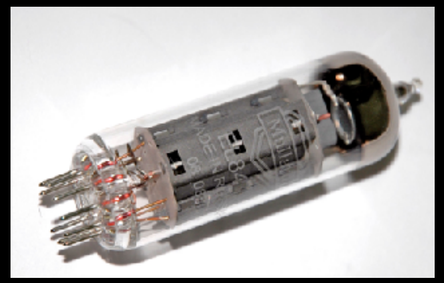
```
enum ManyAssociatedObjectsEnum {
    case A(AnyObject)
    case B(AnyObject)
    case C(AnyObject)
    case D(AnyObject)
    case E(AnyObject)
}
struct ManyAssociatedObjectsEnumHolder {
    var a: ManyAssociatedObjectsEnum
    var b: ManyAssociatedObjectsEnum
    var c: ManyAssociatedObjectsEnum
    var d: ManyAssociatedObjectsEnum
    var e: ManyAssociatedObjectsEnum
}
ManyAssociatedObjectsEnumHolder(
    a: .A(NSObject()), b: .B(NSObject()),
    c: .C(NSObject()), d: .D(NSObject()),
    e: .E(NSObject()))
```

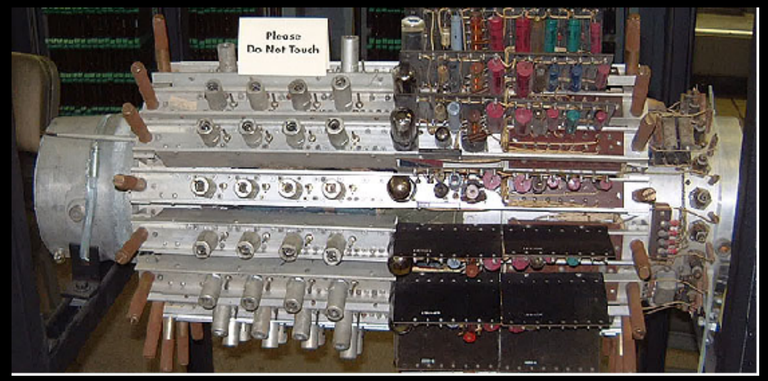
Enums

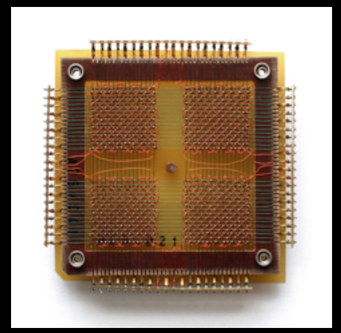

malloc 0x1013053c0 (73 bytes)
e058300101000000
0000000000000000
6051300101000000
0100000000000000
9027300101000000
0230303030300200
7034300101000000
0300000000000000...
Strings:
00000
@16
@64
@0
@32
@48
Instance of NSObject 0x1013046d0 (16 bytes)
f0d0eb79ff7f0000
0000000000000000
Instance of NSObject 0x1013058e0 (16 bytes)
f0d0eb79ff7f0000
0000000000000000
Instance of NSObject 0x101302790 (16 bytes)
f0d0eb79ff7f0000
0000000000000000

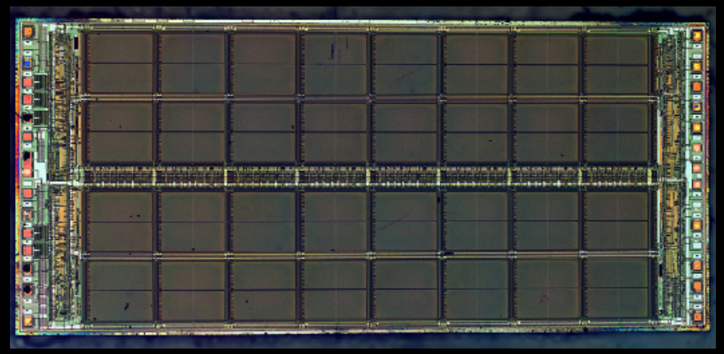
Wrapping Up

Wrapping Up





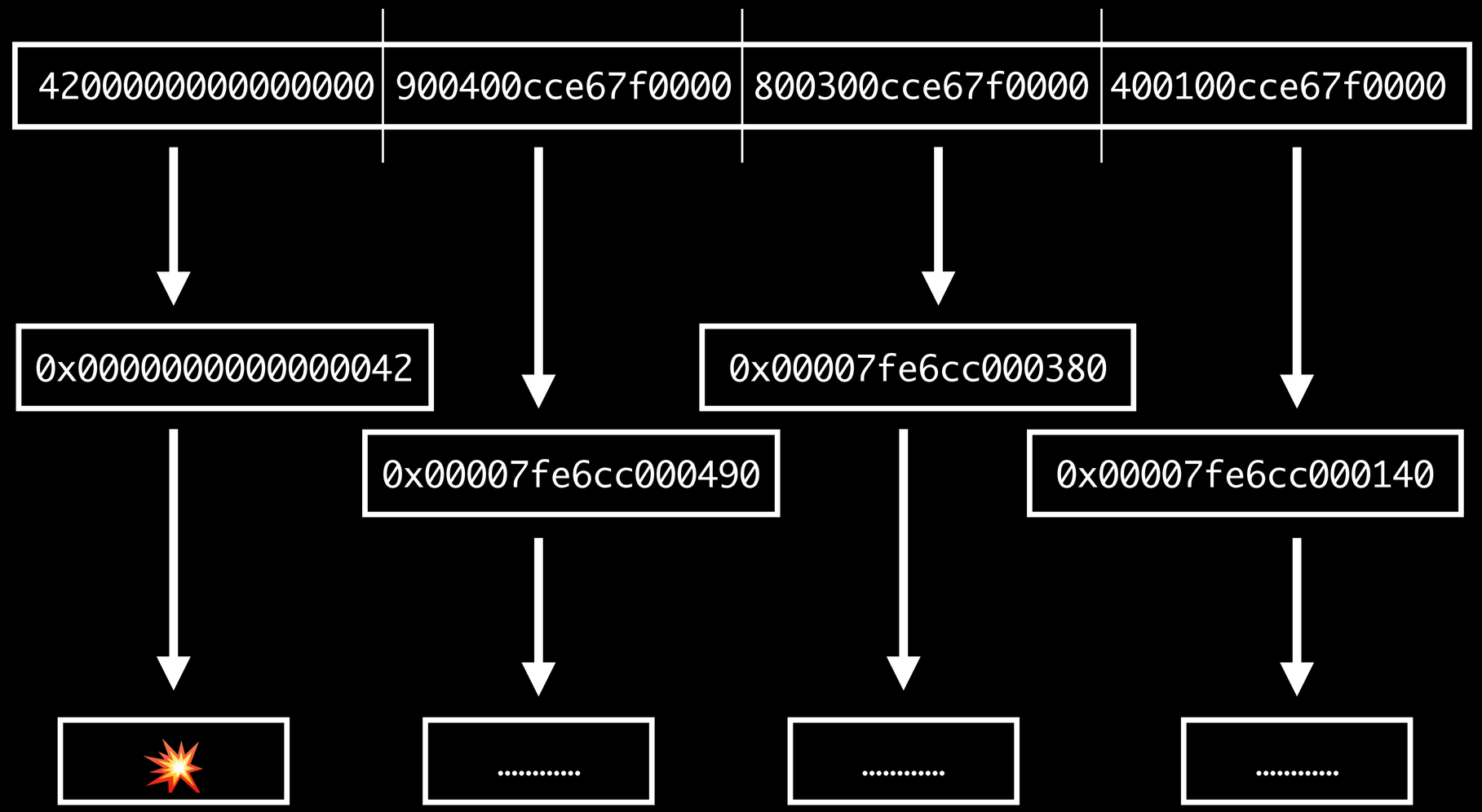




Wrapping Up

…	⋮
0000000100000000	01AA2C5EFF001101
0000000100000008	0000000000000000
0000000100000010	00000000000000FF
0000000100000018	A0F31C228A177013
…	⋮

Wrapping Up

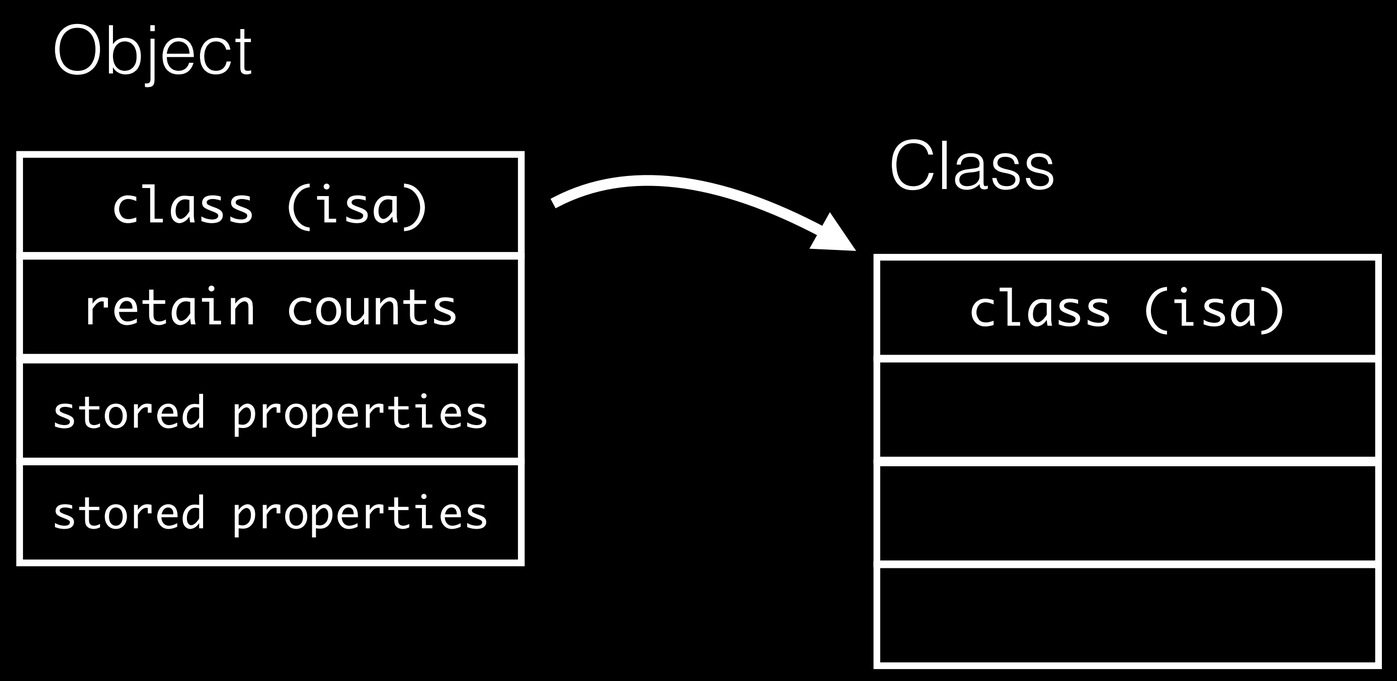

4200000000000000
900400cce67f0000
800300cce67f0000
400100cce67f0000
0x0000000000000042
0x00007fe6cc000380
0x00007fe6cc000490
0x00007fe6cc000140
💥
.....
.....
.....

Wrapping Up

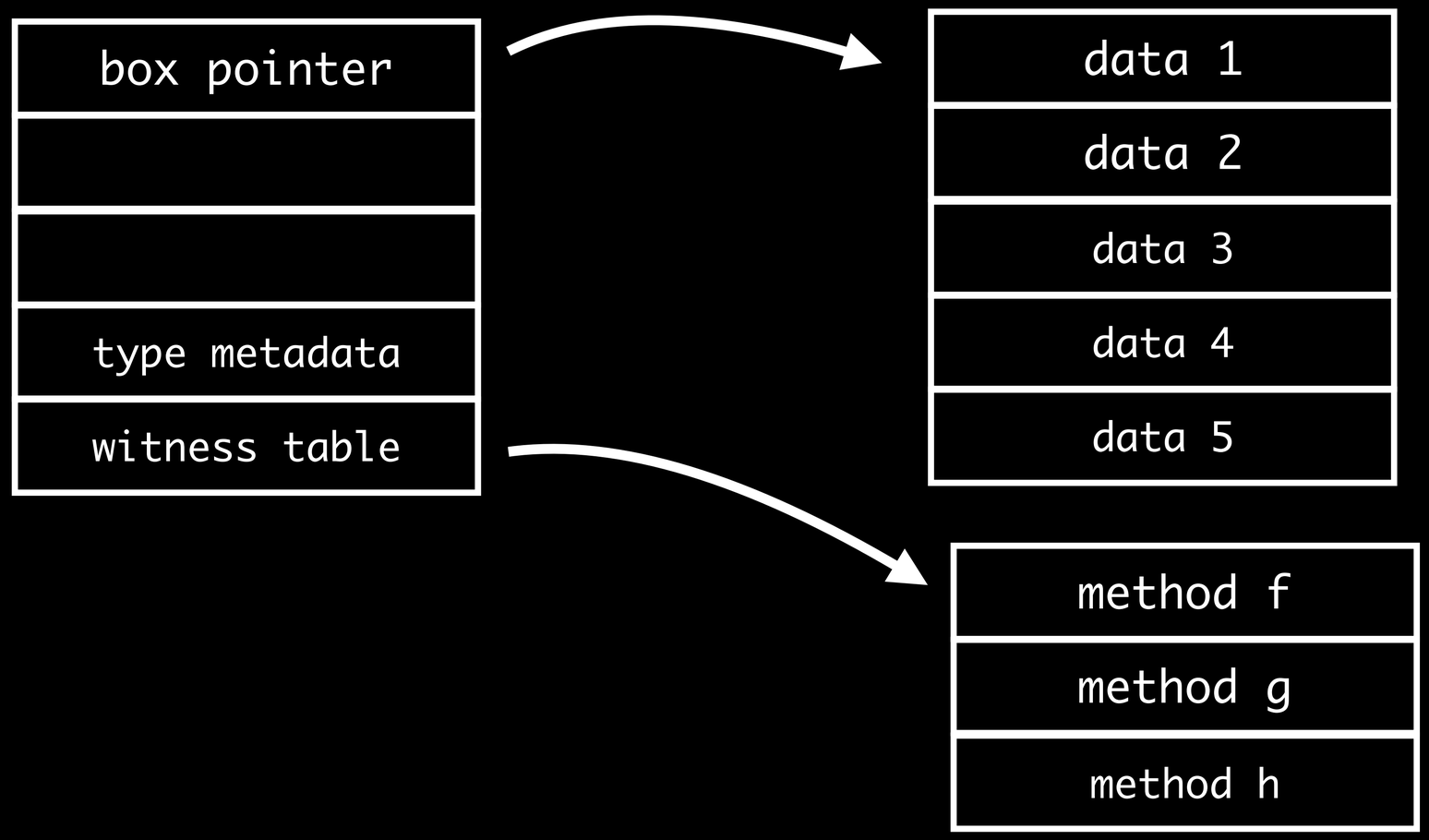
```
struct WithPadding {
    char a;
    char b;
    char c;
    short d;
    char e;
    int f;
    char g;
    long h;
};
WithPadding withPadding =
    { 1, 2, 3, 4, 5, 6, 7, 8 };
```

unknown 0x7fff5fbff288 (24 bytes)
0102030004000500
0600000007000000
0800000000000000

Wrapping Up


Object
class (isa)
retain counts
stored properties
stored properties
Class
class (isa)

Wrapping Up

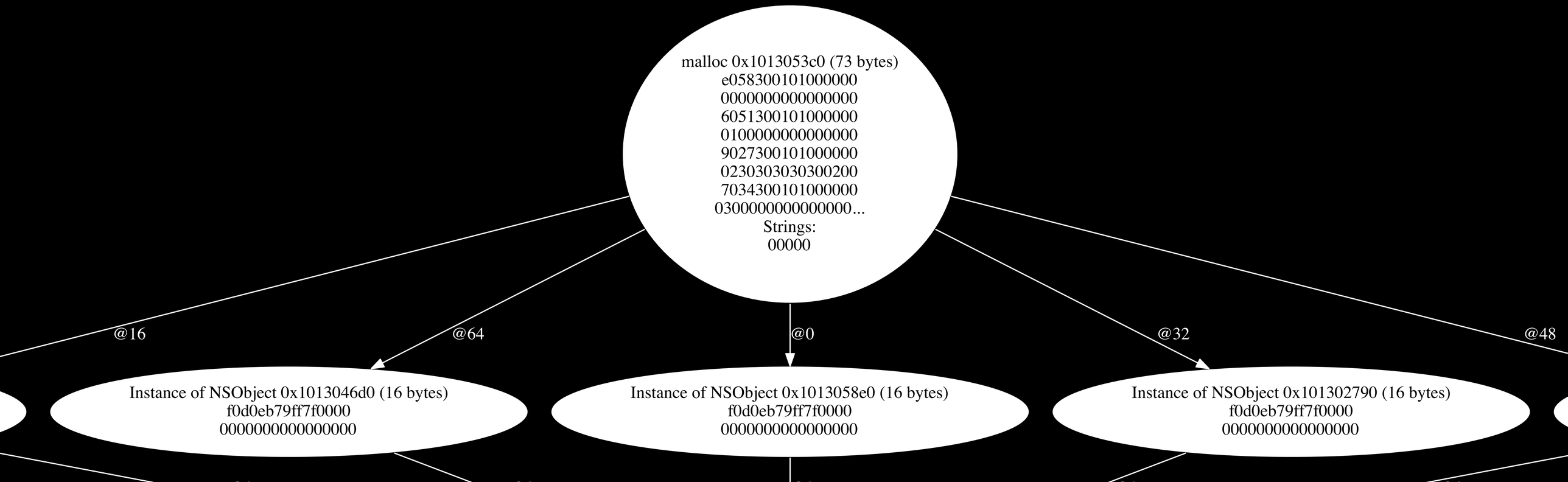

box pointer
type metadata
witness table
data 1
data 2
data 3
data 4
data 5
method f
method g
method h

Wrapping Up

```
enum SimpleEnum {
    case A, B, C, D, E
}
struct SimpleEnumHolder {
    var a: SimpleEnum
    var b: SimpleEnum
    var c: SimpleEnum
    var d: SimpleEnum
    var e: SimpleEnum
}
SimpleEnumHolder(a: .A, b: .B,
                 c: .C, d: .D, e: .E)
```

unknown 0x7fff5fbff250 (5 bytes)
0001020304

Wrapping Up


malloc 0x1013053c0 (73 bytes)
e058300101000000
0000000000000000
6051300101000000
0100000000000000
9027300101000000
0230303030300200
7034300101000000
0300000000000000...
Strings:
00000
@16
@64
@0
@32
@48
Instance of NSObject 0x1013046d0 (16 bytes)
f0d0eb79ff7f0000
0000000000000000
Instance of NSObject 0x1013058e0 (16 bytes)
f0d0eb79ff7f0000
0000000000000000
Instance of NSObject 0x101302790 (16 bytes)
f0d0eb79ff7f0000
0000000000000000

Wrapping Up

- Data laid out linearly
- Padded for alignment
- Class instances have isa and refcounts first
- Protocol values have 3 words of inline data
- Larger data is boxed
- Dynamic method dispatch uses vtables
- Swift is powerful: all C-ish evil stuff available
- Can learn a lot by poking around

Wrapping Up

goto; conference

Please

Remember to rate this session

Thank you!

Join the conversation #gotocph