objective-c RUNTIME

jano@jano.com.es

What is it?

Objective-C is object oriented and dynamic.

objects are C structs

```
@interface Person @end
@implementation Person @end
```

↓

```
struct Person {
  Class isa;
}
```

↓

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

reflection

```
[NSString alloc]
```

↓

```
objc_msgSend(objc_getClass("NSString"),
    sel_registerName("alloc"))
```

↓

alloc	0x00002710

“Objective-c has message passing, not method calls”

A message call

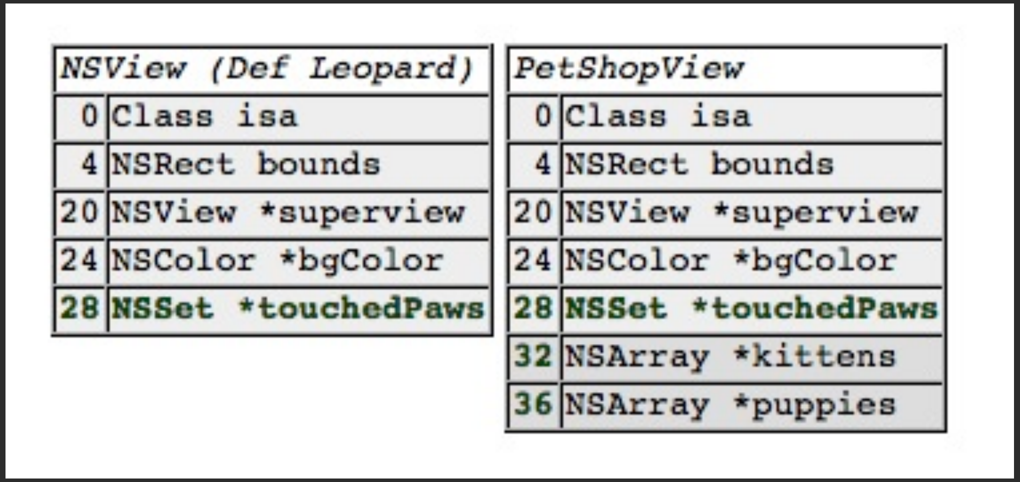
```
// method signature
+ (NSString*)randomStringWithLength:(NSUInteger)length {
    // ... method implementation
}

// message
[NSString randomStringWithLength:10]

// selector
randomStringWithLength:

// receiver
NSString

// parameters
10
```

Runtime features

Introspection
Dynamic typing, binding, linking
Categories
Code creation and swizzling
Forwarding proxy support
Non-fragile instance variables

Fragile base class

```
#import <Foundation/Foundation.h>

@interface Person : NSObject
@property NSString *_name;
@end

@implementation Person {
    NSInteger age;
}
@end
```



NSView (Def Leopard)	
0	Class isa
4	NSRect bounds
20	NSView *superview
24	NSColor *bgColor
28	**NSSet *touchedPaws**

PetShopView	
0	Class isa
4	NSRect bounds
20	NSView *superview
24	NSColor *bgColor
28	**NSSet *touchedPaws**
32	NSArray *kittens
36	NSArray *puppies

http://www.sealiesoftware.com/blog

@dynamic

@dynamic example 1/2

```objc
#import <objc/runtime.h>
#import <Foundation/Foundation.h>

@interface Person : NSObject
@property (copy) NSString *name;
@end

@interface Person()
@property (nonatomic,strong) NSMutableDictionary *dic;
@end

@implementation Person

@dynamic name;

-(id) init {
    if (self = [super init]){
        _dic = [NSMutableDictionary dictionary];
    }
    return self;
}

@end

int main(int argc, char *argv[]){
    @autoreleasepool {
        Person *p = [Person new];
        p.name = @"Dolores";
        NSLog(@"%@",p.name);
    }
}
```

@dynamic example 2/2

```
// generic getter
id propertyIMP(id self, SEL _cmd) {
    return [((Person*)self).dic objectForKey:NSStringFromSelector(_cmd)];
}


// generic setter
void setPropertyIMP(id self, SEL _cmd, id value) {

    // setName --> name
    NSMutableString *key = [NSStringFromSelector(_cmd) mutableCopy];
    [key deleteCharactersInRange:NSMakeRange(0, 3)];
    [key deleteCharactersInRange:NSMakeRange([key length]-1, 1)];
    [key replaceCharactersInRange:NSMakeRange(0, 1) withString:[[key substringToIndex:1]
lowercaseString]];

    [((Person*)self).dic setObject:value forKey:key];
}


+ (BOOL)resolveInstanceMethod:(SEL)aSEL {
    if ([NSStringFromSelector(aSEL) hasPrefix:@"set"]) {
        class_addMethod([self class], aSEL, (IMP)setPropertyIMP, "v@:@");
    } else {
        class_addMethod([self class], aSEL,(IMP)propertyIMP, "@@:");
    }
    return YES;
}
```

Swizzling

KVO trickery: Dynamic Subclass

```
- (Class) dynamicallySubclass:(id)instance {

     const char * prefix = "DynamicSubclass_";
     Class instanceClass = [instance class];
     NSString * className = NSStringFromClass(instanceClass);

     BOOL isDynamicSubclass = strncmp(prefix, [className UTF8String], strlen(prefix)) == 0;
     if (isDynamicSubclass) { return [instance class]; }

     NSString *subclassName = [NSString stringWithFormat:@"%s%@", prefix, className];
     Class subclass = NSClassFromString(subclassName);

     BOOL classExists = subclass!=nil;

     if (classExists) {
         object_setClass(instance, subclass);

     } else {
         subclass = objc_allocateClassPair(instanceClass, [subclassName UTF8String], 0);
         if (subclass != nil) {

             // subclass created, now change it:
             //   1. create the a kvoProperty method for each property
             //   2. replace the imp of each property with the imp of the kvoProperty method

             // method swizzling would be:
             //   IMP newImp = class_getMethodImplementation([self class], @selector(kvoProperty));
             //   class_addMethod(subclass, @selector(name), newImp, "v@:");

             objc_registerClassPair(subclass);
         }
     }

     return subclass;
 }
```

KVO trickery: ISA Swizzling

```
#import <objc/runtime.h>
 #import <Foundation/Foundation.h>

 @interface NSObject (IsaSwizzling)
 - (void)setClass:(Class)aClass;
 @end

 @implementation NSObject (IsaSwizzling)

 - (void)setClass:(Class)aClass {
     NSAssert(class_getInstanceSize([self class]) == class_getInstanceSize(aClass),
              @"Classes must be the same size to swizzle. Did you add ivars?");
     object_setClass(self, aClass);
 }

 @end
```

Implementation

~~NeXT~~

~~Apple's Legacy runtime (32bit)~~

Apple's Objective-C 2.1

~~Étoilé Runtime~~

GNUStep

Compilers

GCC

~~Apple's GCC fork 4.2.1~~

~~LLVM-GCC~~

Clang

the object struct

What is an object?

```
// Foundation.framework/NSObject.h
@interface NSObject <NSObject> {
    Class isa;
}
// ... bunch of methods
@end

struct NSObject {
    Class isa;
}

// /usr/include/objc/objc.h
typedef struct objc_class *Class;

struct NSObject {
    objc_class *isa;
}

typedef struct objc_object {
  Class isa;
} *id;
```

Legacy class struct

```
// /usr/include/objc/runtime.h
struct objc_class {
    Class isa;
#if !__OBJC2__
    Class super_class                         OBJC2_UNAVAILABLE;
    const char *name                          OBJC2_UNAVAILABLE;
    long version                              OBJC2_UNAVAILABLE;
    long info                                 OBJC2_UNAVAILABLE;
    long instance_size                        OBJC2_UNAVAILABLE;
    struct objc_ivar_list *ivars              OBJC2_UNAVAILABLE;
    struct objc_method_list **methodLists     OBJC2_UNAVAILABLE;
    struct objc_cache *cache                  OBJC2_UNAVAILABLE;
    struct objc_protocol_list *protocols      OBJC2_UNAVAILABLE;
#endif
} OBJC2_UNAVAILABLE;
```

Modern class struct 1/3

```
typedef struct class_ro_t {
    uint32_t flags;
    uint32_t instanceStart;
    uint32_t instanceSize;
#ifdef __LP64__
    uint32_t reserved;
#endif
    const uint8_t * ivarLayout;
    const char * name;
    const method_list_t * baseMethods;
    const protocol_list_t * baseProtocols;
    const ivar_list_t * ivars;
    const uint8_t * weakIvarLayout;
    const property_list_t *baseProperties;
} class_ro_t;
```

Modern class struct 2/3

```
typedef struct class_rw_t {
    uint32_t flags;
    uint32_t version;
    const class_ro_t *ro;
    union {
        method_list_t **method_lists;   // RW_METHOD_ARRAY == 1
        method_list_t *method_list;     // RW_METHOD_ARRAY == 0
    };
    struct chained_property_list *properties;
    const protocol_list_t ** protocols;
    struct class_t *firstSubclass;
    struct class_t *nextSiblingClass;
} class_rw_t;
```

Modern class struct 3/3

```
typedef struct class_t {
    struct class_t *isa;
    struct class_t *superclass;
    Cache cache;
    IMP *vtable;
    // class_rw_t * plus custom rr/alloc flags
    uintptr_t data_NEVER_USE;
    class_rw_t *data() const {
        return (class_rw_t *)(data_NEVER_USE & ~(uintptr_t)3);
    }
    // ...
} class_t;
```

Objective-C: a thin layer on top of C

Example: [Person new]

```
#import <objc/runtime.h>

@interface Person
@end

@implementation Person
+(id)new {
    Class cls = objc_getClass("Person");
    id obj = class_createInstance(cls, class_getInstanceSize(cls));
    return obj;
}
@end


int main(int argc, char *argv[]){
    @autoreleasepool {
        [Person new];
    }
}
```

Person.cpp 1/3

```
// ...

#define __OFFSETOFIVAR__(TYPE, MEMBER) ((long long) &((TYPE *)0)->MEMBER)
#include <objc/runtime.h>

#ifndef _REWRITER_typedef_Person
#define _REWRITER_typedef_Person
typedef struct objc_object Person;
#endif

/* @end */

// @implementation Person

static id _C_Person_new(Class self, SEL _cmd) {
    Class cls = objc_getClass("Person");
    id obj = class_createInstance(cls, class_getInstanceSize(cls));
    return obj;
}
// @end


int main(int argc, char *argv[]){
    @autoreleasepool {
        ((id (*)(id, SEL))(void *)objc_msgSend)(objc_getClass("Person"), sel_registerName("new"));
    }
}

struct _objc_method {
     SEL _cmd;
     char *method_types;
     void *_imp;
};
```

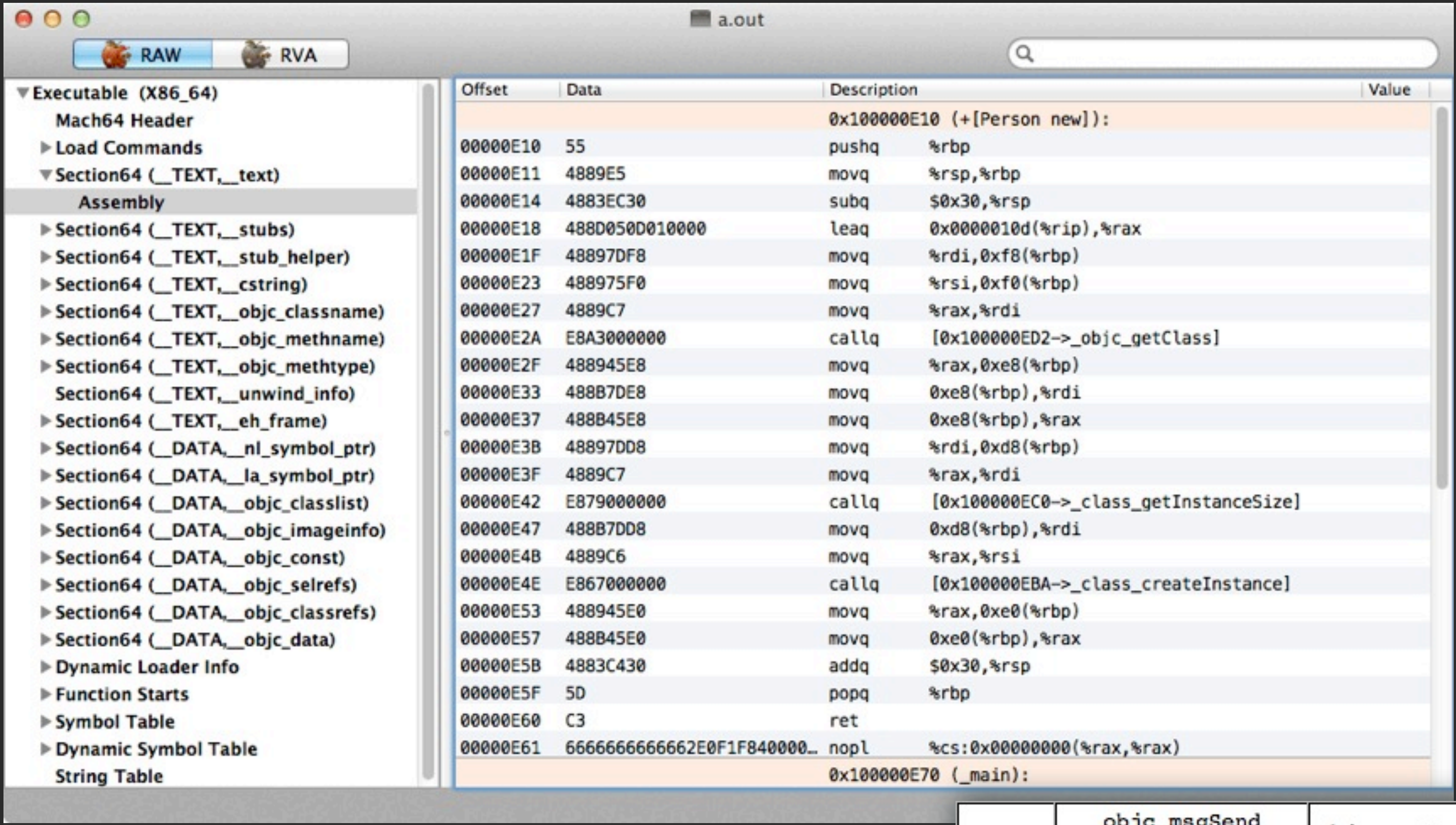
Person.cpp 2/3

```
static struct {
     struct _objc_method_list *next_method;
     int method_count;
     struct _objc_method method_list[1];
} _OBJC_CLASS_METHODS_Person __attribute__ ((used, section ("__OBJC, __cls_meth")))= {
     0, 1
     ,{{(SEL)"new", "@16@0:8", (void *)_C_Person_new}
      }
};

struct _objc_class {
     struct _objc_class *isa;
     const char *super_class_name;
     char *name;
     long version;
     long info;
     long instance_size;
     struct _objc_ivar_list *ivars;
     struct _objc_method_list *methods;
     struct objc_cache *cache;
     struct _objc_protocol_list *protocols;
     const char *ivar_layout;
     struct _objc_class_ext  *ext;
};

static struct _objc_class _OBJC_METACLASS_Person __attribute__ ((used, section ("__OBJC, __meta_class")))=
{
     (struct _objc_class *)"Person", 0, "Person", 0,2, sizeof(struct _objc_class), 0
     , (struct _objc_method_list *)&_OBJC_CLASS_METHODS_Person
     ,0,0,0,0
};

static struct _objc_class _OBJC_CLASS_Person __attribute__ ((used, section ("__OBJC, __class")))= {
     &_OBJC_METACLASS_Person, 0, "Person", 0,1,0,0,0,0,0,0,0
};
```

Person.cpp 3/3

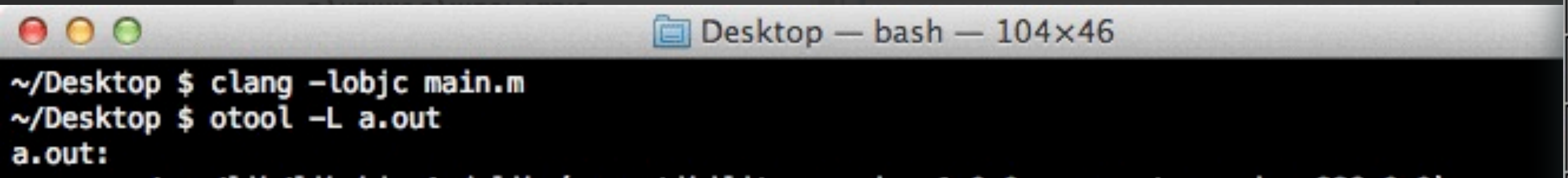
```
struct _objc_symtab {
     long sel_ref_cnt;
     SEL *refs;
     short cls_def_cnt;
     short cat_def_cnt;
     void *defs[1];
};

static struct _objc_symtab _OBJC_SYMBOLS __attribute__((used, section ("__OBJC, __symbols")))= {
     0, 0, 1, 0
     ,&_OBJC_CLASS_Person
};


struct _objc_module {
     long version;
     long size;
     const char *name;
     struct _objc_symtab *symtab;
};

static struct _objc_module _OBJC_MODULES __attribute__ ((used, section ("__OBJC, __module_info")))= {
     7, sizeof(struct _objc_module), "", &_OBJC_SYMBOLS
};
```

MachOView

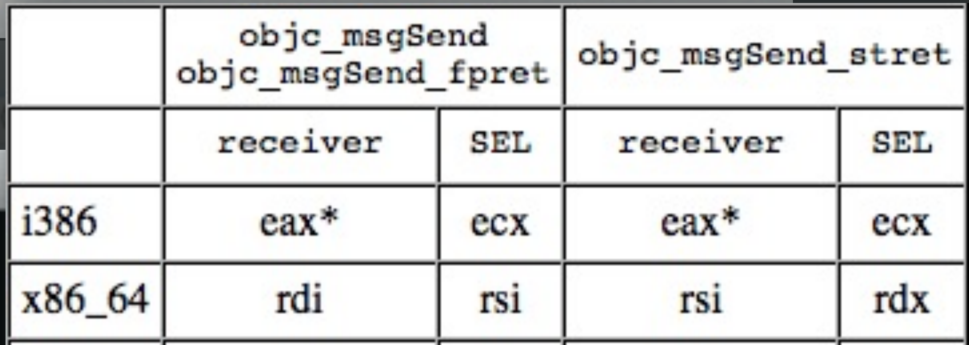


```
▼Executable (X86_64)
    Mach64 Header
  ▶ Load Commands
  ▼ Section64 (__TEXT,__text)
      Assembly
  ▶ Section64 (__TEXT,__stubs)
  ▶ Section64 (__TEXT,__stub_helper)
  ▶ Section64 (__TEXT,__cstring)
  ▶ Section64 (__TEXT,__objc_classname)
  ▶ Section64 (__TEXT,__objc_methname)
  ▶ Section64 (__TEXT,__objc_methtype)
    Section64 (__TEXT,__unwind_info)
  ▶ Section64 (__TEXT,__eh_frame)
  ▶ Section64 (__DATA,__nl_symbol_ptr)
  ▶ Section64 (__DATA,__la_symbol_ptr)
  ▶ Section64 (__DATA,__objc_classlist)
  ▶ Section64 (__DATA,__objc_imageinfo)
  ▶ Section64 (__DATA,__objc_const)
  ▶ Section64 (__DATA,__objc_selrefs)
  ▶ Section64 (__DATA,__objc_classrefs)
  ▶ Section64 (__DATA,__objc_data)
  ▶ Dynamic Loader Info
  ▶ Function Starts
  ▶ Symbol Table
  ▶ Dynamic Symbol Table
    String Table
```

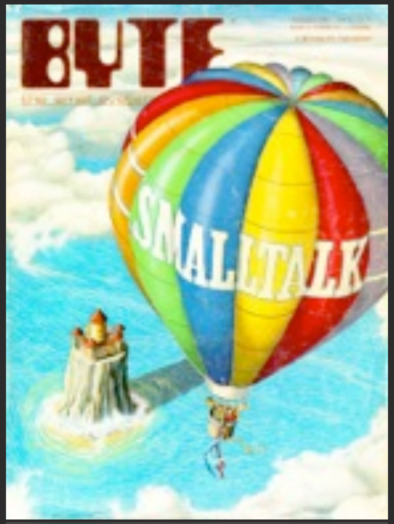
```
Offset    Data                    Description                                      Value
                                  0x100000E10 (+[Person new]):
00000E10  55                      pushq    %rbp
00000E11  4889E5                  movq     %rsp,%rbp
00000E14  4883EC30                subq     $0x30,%rsp
00000E18  488D050D010000          leaq     0x0000010d(%rip),%rax
00000E1F  48897DF8                movq     %rdi,0xf8(%rbp)
00000E23  488975F0                movq     %rsi,0xf0(%rbp)
00000E27  4889C7                  movq     %rax,%rdi
00000E2A  E8A3000000              callq    [0x100000ED2->_objc_getClass]
00000E2F  488945E8                movq     %rax,0xe8(%rbp)
00000E33  488B7DE8                movq     0xe8(%rbp),%rdi
00000E37  488B45E8                movq     0xe8(%rbp),%rax
00000E3B  48897DD8                movq     %rdi,0xd8(%rbp)
00000E3F  4889C7                  movq     %rax,%rdi
00000E42  E879000000              callq    [0x100000EC0->_class_getInstanceSize]
00000E47  488B7DD8                movq     0xd8(%rbp),%rdi
00000E4B  4889C6                  movq     %rax,%rsi
00000E4E  E867000000              callq    [0x100000EBA->_class_createInstance]
00000E53  488945E0                movq     %rax,0xe0(%rbp)
00000E57  488B45E0                movq     0xe0(%rbp),%rax
00000E5B  4883C430                addq     $0x30,%rsp
00000E5F  5D                      popq     %rbp
00000E60  C3                      ret
00000E61  6666666666662E0F1F840000…  nopl  %cs:0x00000000(%rax,%rax)
                                  0x100000E70 (_main):
```

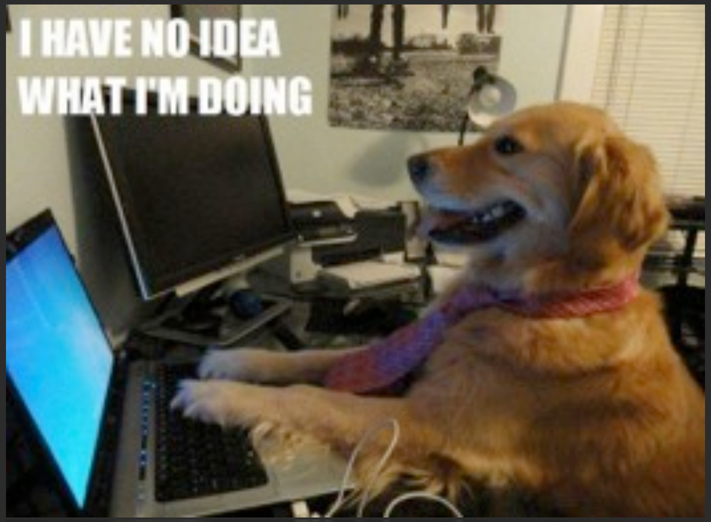


```
~/Desktop $ clang -lobjc main.m
~/Desktop $ otool -L a.out
a.out:
```

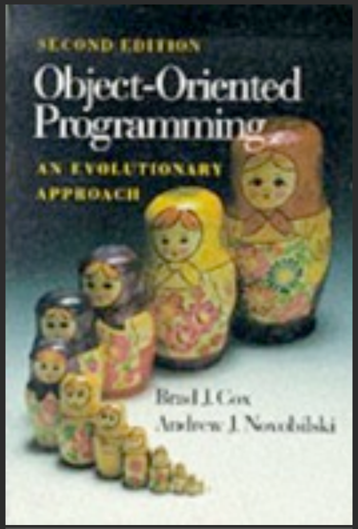


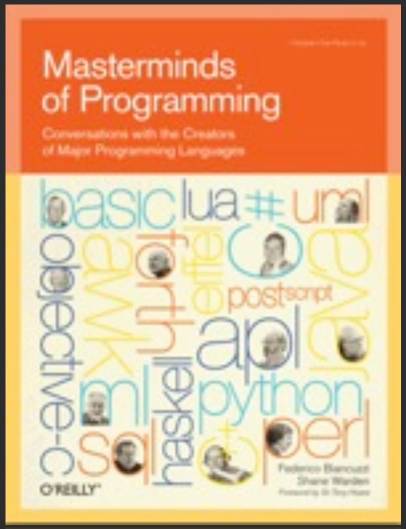
	objc_msgSend objc_msgSend_fpret		objc_msgSend_stret	
	receiver	SEL	receiver	SEL
i386	eax*	ecx	eax*	ecx
x86_64	rdi	rsi	rsi	rdx





ninjasandrobots.com/you-need-some-experience



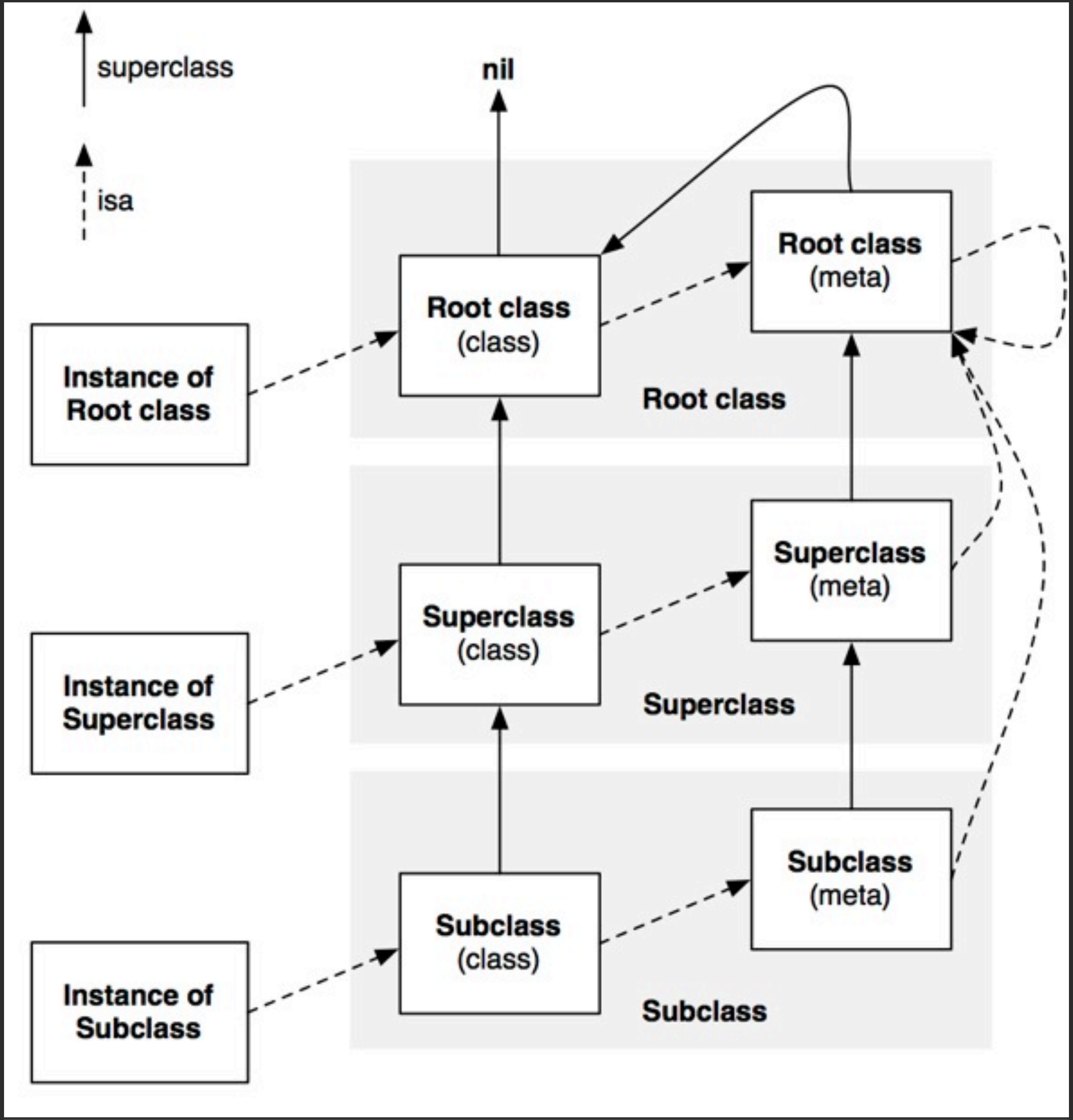

Masterminds of Programming
Conversations with the Creators of Major Programming Languages
basic
lua
uml
objective-c
awk
forth
eiffel
postscript
apl
haskell
ml
python
sql
perl
O'REILLY

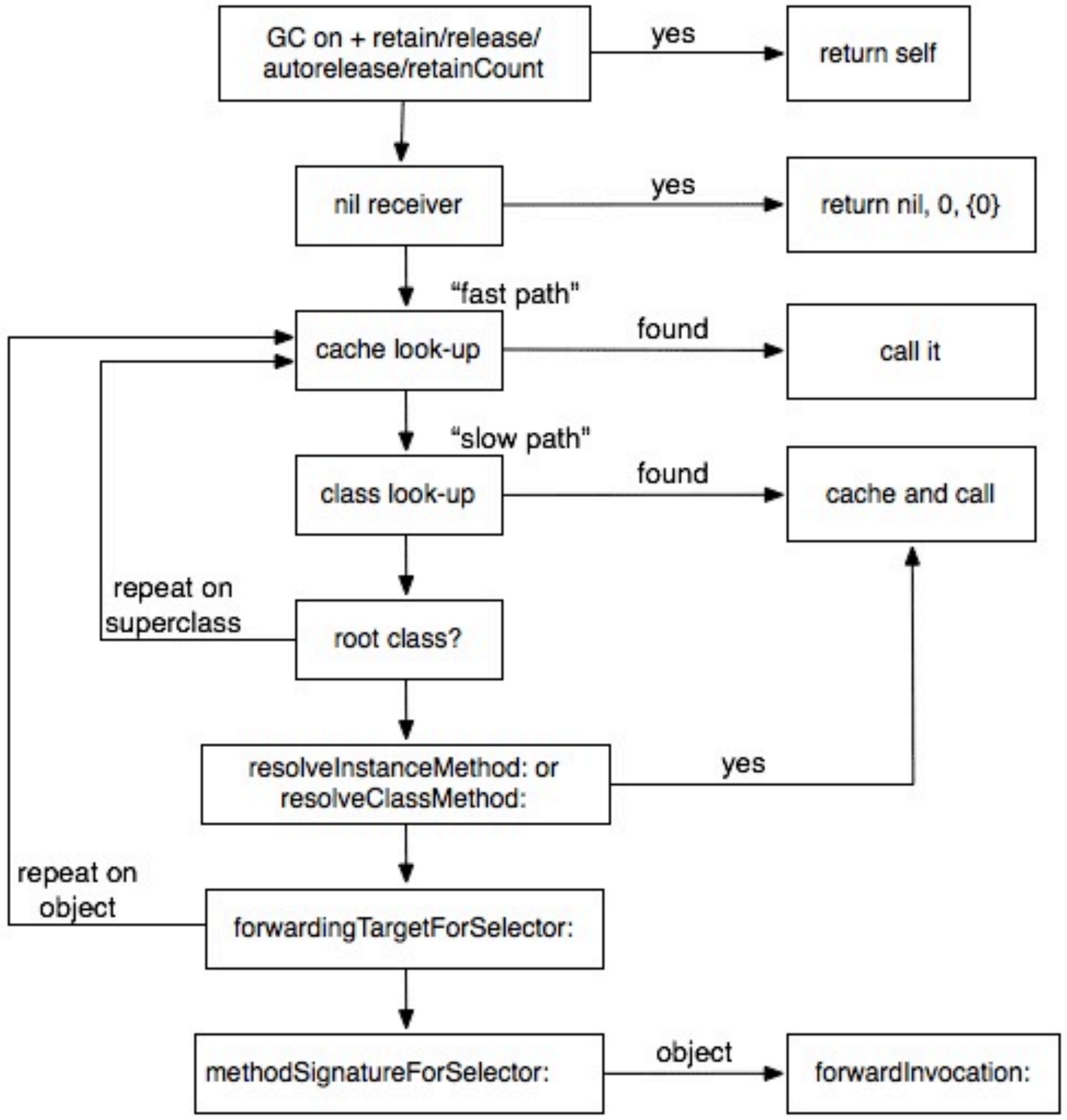
objc_sendMsg

objc msgSend

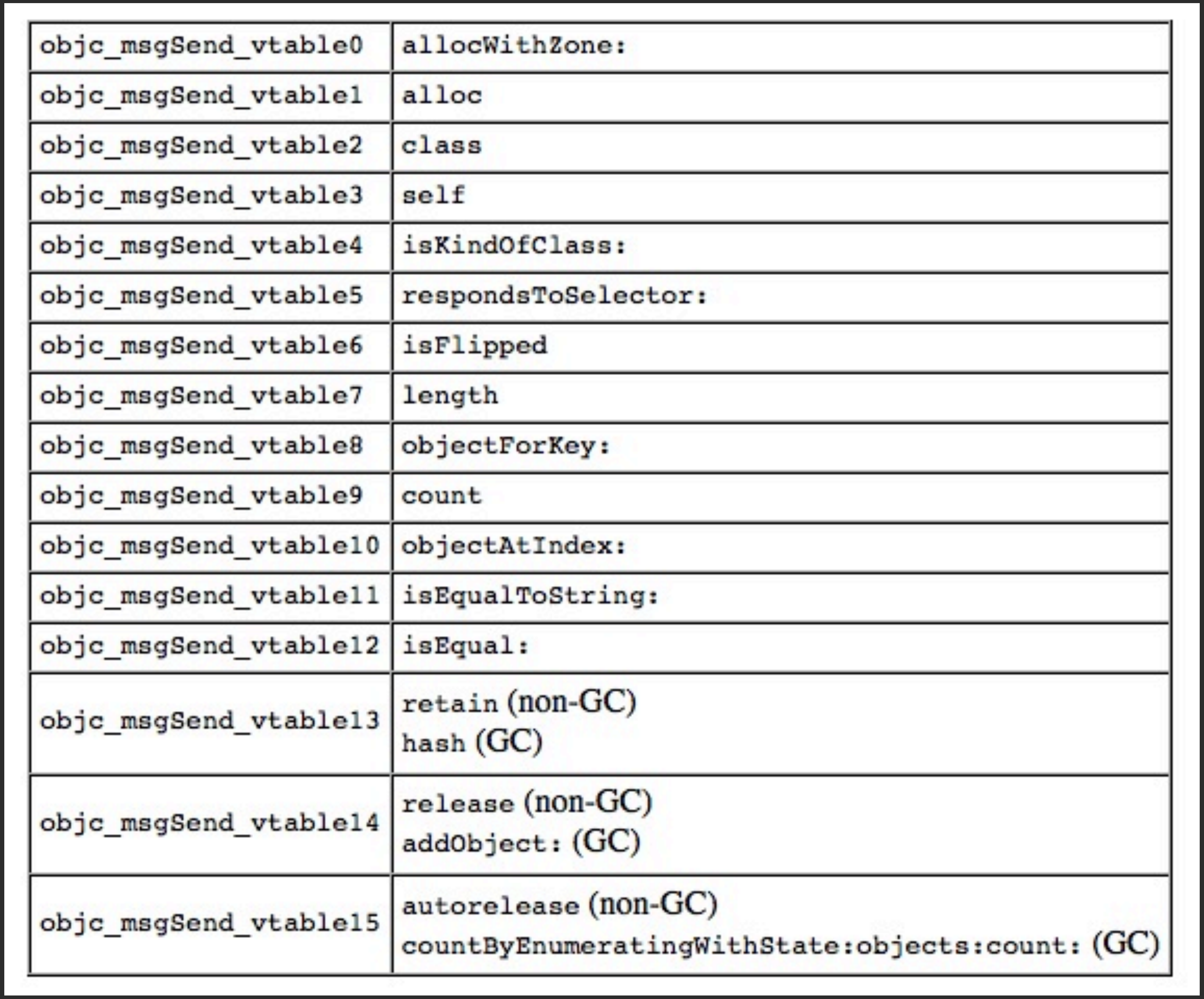
```
id objc_msgSend(id receiver, SEL name, arguments...) {
    IMP function =
        class_getMethodImplementation(receiver->isa, name);
    return function(arguments);
}
```

http://www.sealiesoftware.com/blog




GC on + retain/release/ autorelease/retainCount
yes
return self
nil receiver
yes
return nil, 0, {0}
"fast path"
cache look-up
found
call it
"slow path"
class look-up
found
cache and call
repeat on superclass
root class?
resolveInstanceMethod: or resolveClassMethod:
yes
repeat on object
forwardingTargetForSelector:
methodSignatureForSelector:
object
forwardInvocation:

Virtual table



objc_msgSend_vtable0	allocWithZone:
objc_msgSend_vtable1	alloc
objc_msgSend_vtable2	class
objc_msgSend_vtable3	self
objc_msgSend_vtable4	isKindOfClass:
objc_msgSend_vtable5	respondsToSelector:
objc_msgSend_vtable6	isFlipped
objc_msgSend_vtable7	length
objc_msgSend_vtable8	objectForKey:
objc_msgSend_vtable9	count
objc_msgSend_vtable10	objectAtIndex:
objc_msgSend_vtable11	isEqualToString:
objc_msgSend_vtable12	isEqual:
objc_msgSend_vtable13	retain (non-GC) hash (GC)
objc_msgSend_vtable14	release (non-GC) addObject: (GC)
objc_msgSend_vtable15	autorelease (non-GC) countByEnumeratingWithState:objects:count: (GC)

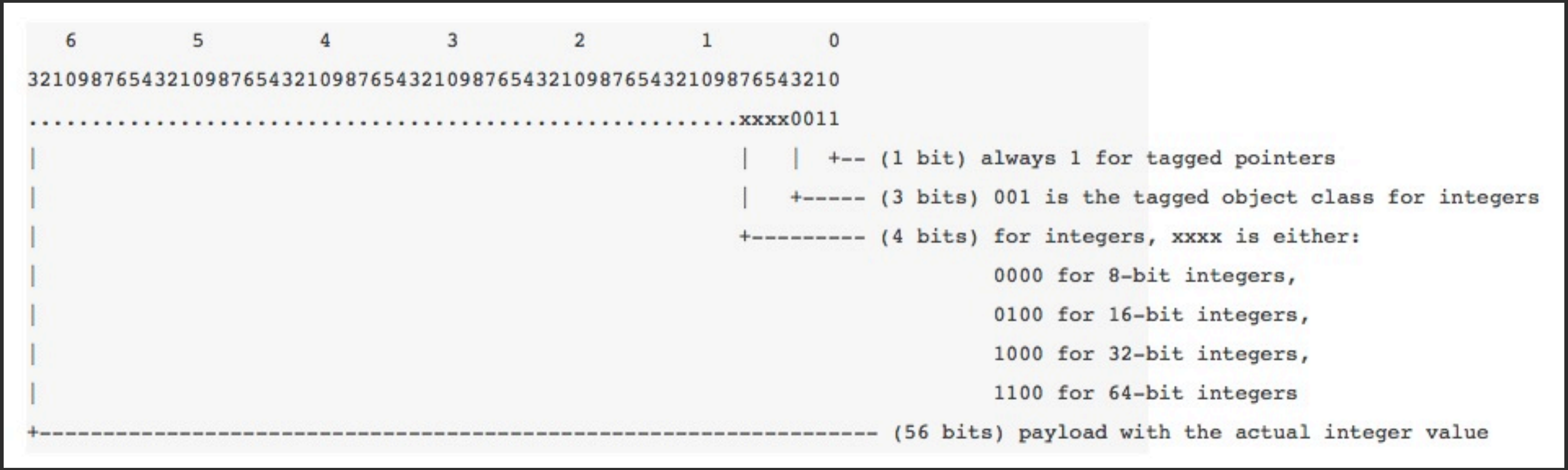
http://www.sealiesoftware.com/blog

Calling the implementation directly

```
#import <objc/runtime.h>

@interface Person
@end

@implementation Person
+(id)new {
    Class cls = objc_getClass("Person");
    id obj = class_createInstance(cls, class_getInstanceSize(cls));
    return obj;
}
@end

int main(int argc, char *argv[]){
    @autoreleasepool {

        // [Person new]
        Person *person;
        SEL newSel = sel_registerName("new");
        Class personClass = objc_getClass("Person");
        Method method = class_getClassMethod(personClass, newSel);
        IMP newImp = method_getImplementation(method);
        id (*new)(id,SEL) = (id (*)(id,SEL)) newImp;
        person = new(personClass,newSel);

    }
}
```

Tagged Pointers

tagged pointers



```
  6         5         4         3         2         1         0
3210987654321098765432109876543210987654321098765432109876543210
........................................................xxxx0011
|                                                       |   |  +-- (1 bit) always 1 for tagged pointers
|                                                       |   +----- (3 bits) 001 is the tagged object class for integers
|                                                       +--------- (4 bits) for integers, xxxx is either:
|                                                                           0000 for 8-bit integers,
|                                                                           0100 for 16-bit integers,
|                                                                           1000 for 32-bit integers,
|                                                                           1100 for 64-bit integers
+------------------------------------------------------------------ (56 bits) payload with the actual integer value
```

objectivistc.tumblr.com/post/7872364181/tagged-pointers-and-fast-pathed-cfnumber-integers-in

Toll free bridge

Blocks

Blocks

http://www.opensource.apple.com/source/libclosure/

```
struct Block_literal_1 {
     void *isa; // &_NSConcreteStackBlock or &_NSConcreteGlobalBlock
     int flags;
     int reserved;

     // reference to the C function that implements this block
     void (*invoke)(void *, ...);

     struct Block_descriptor_1 {
         unsigned long int reserved;      // NULL
         unsigned long int size;          // sizeof(struct Block_literal_1)

         // optional helper functions
         void (*copy_helper)(void *dst, void *src);      // IFF (1<<25)
         void (*dispose_helper)(void *src);              // IFF (1<<25)

         // required ABI.2010.3.16
         const char *signature;                          // IFF (1<<30)
     } *descriptor;
     // ... imported variables
 };
```

Game over man! that was my last slide