A Tour of the Objective-C Runtime API

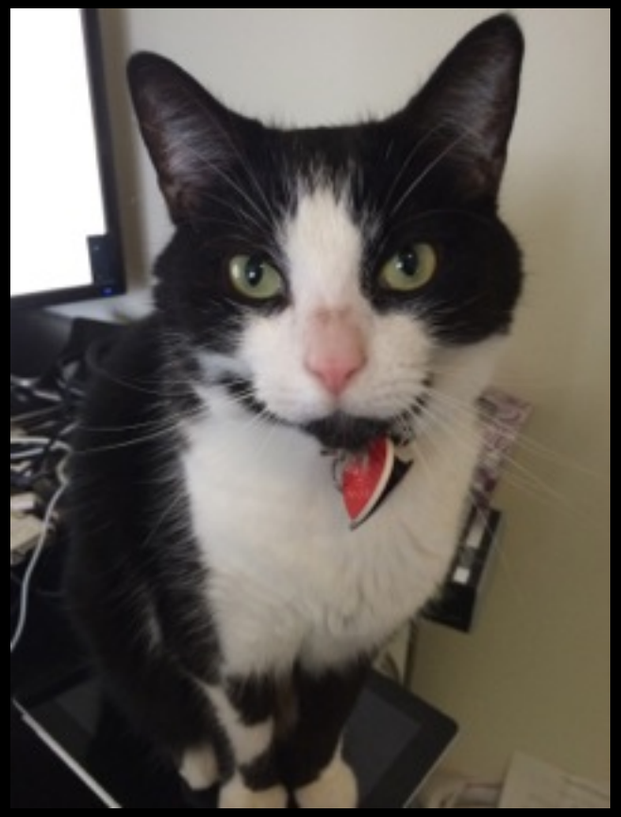
by mikeash

and Plausible Labs

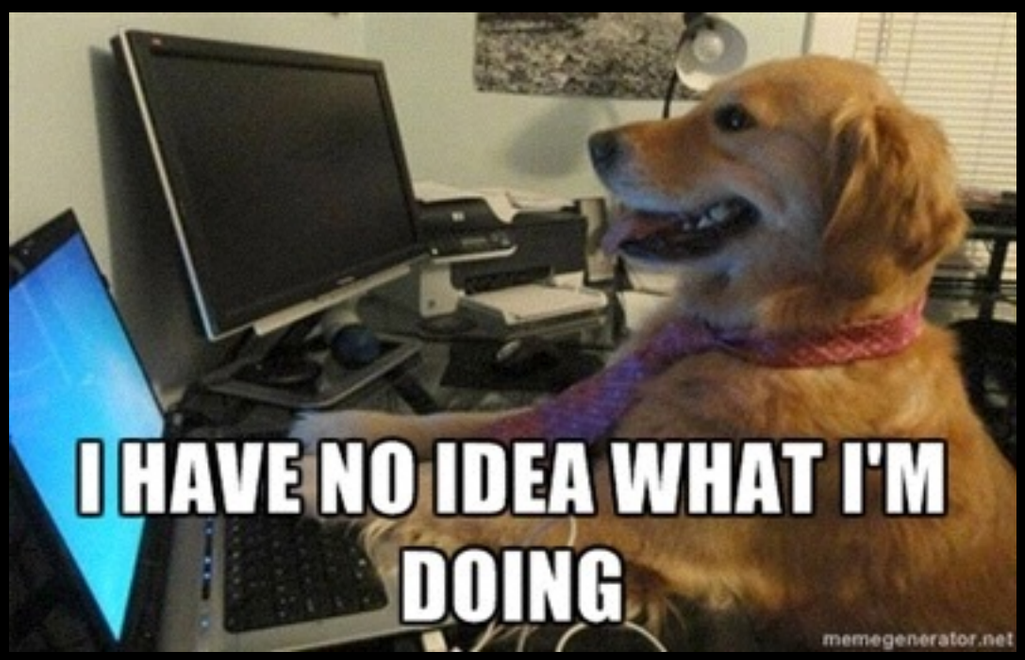
http://lmgtfy.com

About Me

Friday Q&A - mikeash.com/pyblog



About You



Languages at Runtime

Compiler Output

CPU

C
C++

Languages at Runtime

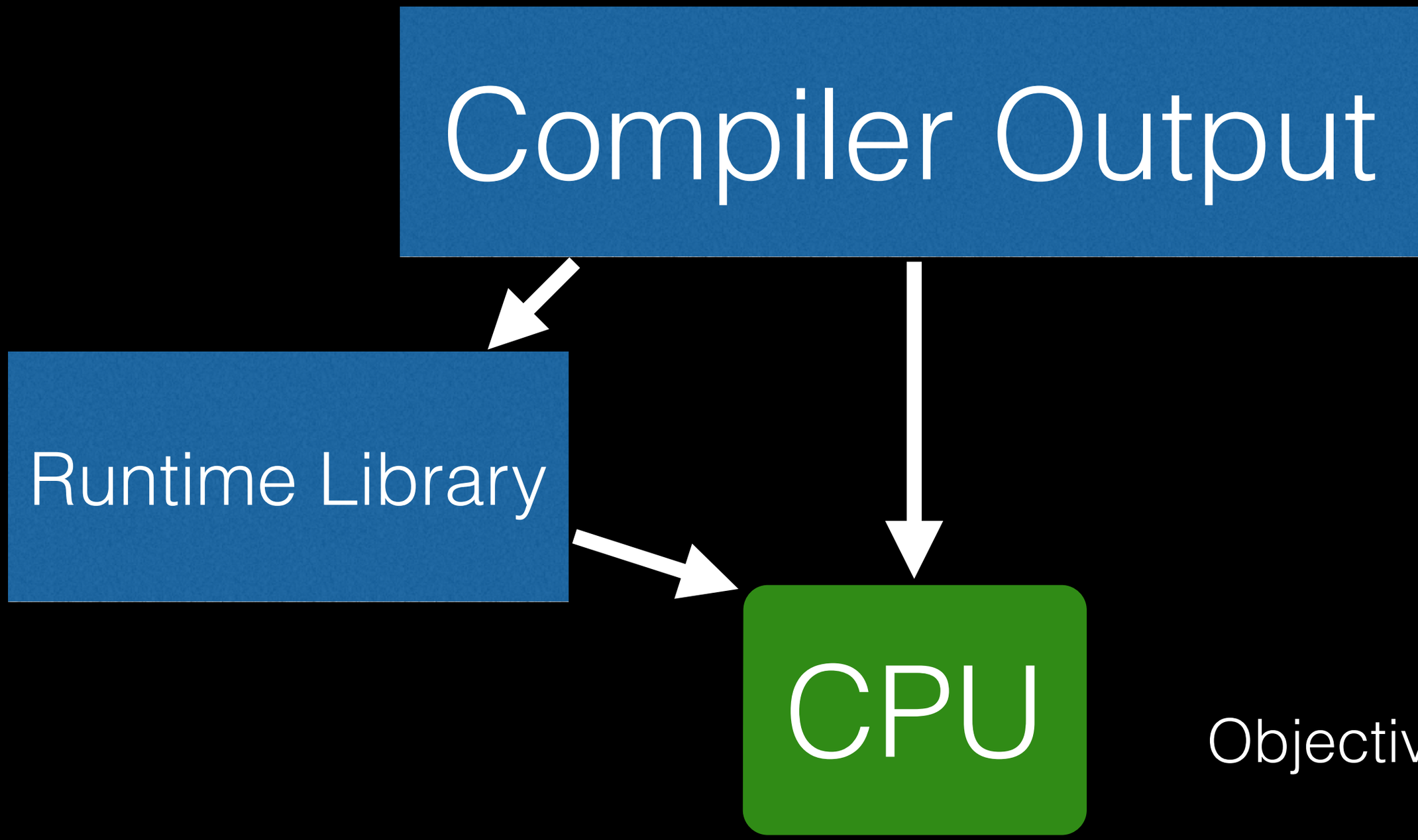
Compiler Output

Runtime Environment

CPU

Java
C#
Python
Ruby
PHP

Languages at Runtime



Objective-C

Objective-C Runtime

Compiler emits calls to the runtime library to:

- Send messages
- Retain and release
- Manage autorelease pools

Why?

- Complex: does not belong in compiler
- Can be updated over time
  (Ex: tagged pointer introduction, autorelease pool implementation changes)

Why?

- API
- Introspection
- Manually invoke runtime behavior
- Make new compilers for existing runtime (Nu, RubyMotion)

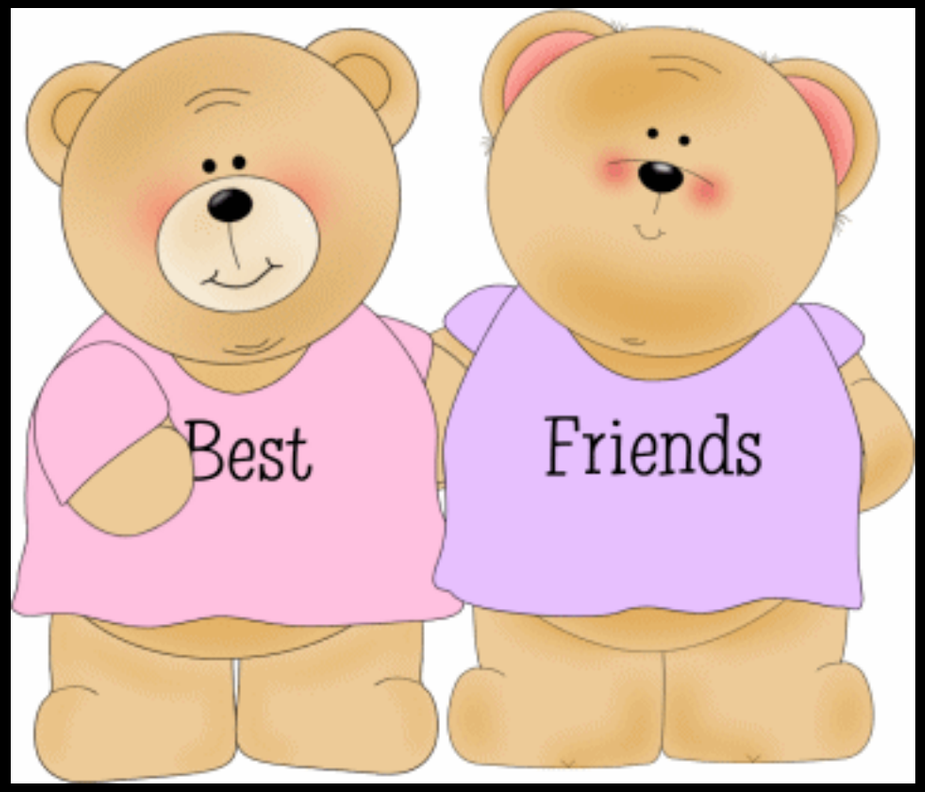
How?

How?

- Use simpler language to implement
- Objective-C runtime is written in….
- C++!

C++ vs Objective-C

C++ and Objective-C



What Can You Do?

- Send messages
- Inspect classes
- Make classes
- Make methods
- Create objects
- …

What Can You Do?

- Send messages
- Inspect classes
- Make classes
- Make methods
- Create objects
- …

More Stuff

- NSInteger/NSUinteger
- NSObject
- BOOL/YES/NO
- nil
- Class
- id
- …

Where?

- /usr/include/objc/*.h
- #import <objc/header.h>
- /usr/lib/libobjc.dylib

Complicated BOOL

```
typedef signed char BOOL;

#if __has_feature(objc_bool)
#define YES                 __objc_yes
#define NO                  __objc_no
#else
#define YES                 ((BOOL)1)
#define NO                  ((BOOL)0)
#endif
```

Complicated nil

```
#ifndef nil
# if __has_feature(cxx_nullptr)
#   define nil nullptr
# else
#   define nil __DARWIN_NULL
# endif
#endif
```

Complicateder nil

```
#ifdef __cplusplus
#ifdef __GNUG__
#define __DARWIN_NULL __null
#else /* ! __GNUG__ */
#ifdef __LP64__
#define __DARWIN_NULL (0L)
#else /* !__LP64__ */
#define __DARWIN_NULL 0
#endif /* __LP64__ */
#endif /* __GNUG__ */
#else /* ! __cplusplus */
#define __DARWIN_NULL ((void *)0)
#endif /* __cplusplus */
```

`sel_getName`

C string from selector

```
const char *selname = sel_getName(selector);
```

```
NSString *selname = NSStringFromSelector(selector);
```

`sel_registerName`

Selector from C string

a.k.a. `sel_getUid`

```
SEL sel = sel_getUid("retain");

SEL sel = NSSelectorFromString(@"retain");
```

`sel_isMapped`

Useless!

"On some platforms, an invalid reference (to invalid memory addresses) can cause a crash."

`object_getIndexedIvars`

Obtains pointer to extra storage at the end of an object

Use with `class_createInstance`

```
id obj = class_createInstance([MyClass class], 42);
char *extra = object_getIndexedIvars(obj);
strcpy(extra, "I am a pretty princess.");
```

`objc_sync_enter`
`objc_sync_exit`

Used for `@synchronized`

```
objc_sync_enter(obj);
/* Critical section here. */
objc_sync_exit(obj);
```

`objc_exception_throw`

Used for `@throw`

```
objc_exception_throw(exception);
```

Also:

`objc_exception_rethrow`
`objc_begin_catch`
`objc_end_catch`
`objc_terminate`
`objc_setExceptionPreprocessor`
`objc_setExceptionMatcher`
`objc_setUncaughtExceptionHandler`
`objc_addExceptionHandler`
`objc_removeExceptionHandler`

Garbage Collection

```
objc_collect
objc_collectingEnabled
objc_collectableZone
objc_setCollectionThreshold
objc_setCollectionRatio
objc_atomicCompareAndSwapPtr
objc_atomicCompareAndSwapPtrBarrier
objc_atomicCompareAndSwapGlobal
objc_atomicCompareAndSwapGlobalBarrier
objc_atomicCompareAndSwapInstanceVariable
objc_atomicCompareAndSwapInstanceVariableBarrier
objc_assign_strongCast
objc_assign_global
objc_assign_threadlocal
objc_assign_ivar
objc_memmove_collectable
objc_read_weak
objc_assign_weak
objc_registerThreadWithCollector
objc_unregisterThreadWithCollector
objc_assertRegisteredThreadWithCollecto
objc_clear_stack
objc_is_finalized
objc_finalizeOnMainThread
objc_collecting_enabled
objc_set_collection_threshold
objc_set_collection_ratio
objc_start_collector_thread
objc_startCollectorThread
objc_allocate_object
```

When to use: never

`object_getClass`

Get the class of an object.

```
Class c = object_getClass(obj);
```

```
Class c = obj->isa;
```

```
Class c = [obj class];
```

Works in the face of tagged pointers, non-pointer isa.

Also: `object_getClassName`

`object_setClass`

Set an object's class. Really! Dangerous.

Really dangerous.

Requires exact same ivar layout.

Use with dynamic subclassing:
instance-specific overrides.

Examples: KVO, MAZeroingWeakRef

objc_copyClassList

`objc_copyClassList`

Returns nil-terminated C array of Class objects
Also returns count, if you want it!

```
Class *classes = objc_copyClassList(NULL);
for(Class *cursor = classes; *cursor != nil; cursor++) {
    NSLog(@"%s", class_getName(*cursor));
}
free(classes);
```

General Pattern

- Make the call
- Ignore the count
- Iterate until you hit `nil`
- `free()` the returned array

objc_copyClassList

~~if([class isSubclassOfClass:[NSObject class]])~~

```
BOOL GetRootClass(Class c) {
    Class superclass = class_getSuperclass(c);
    if(superclass == nil)
        return c;
    else
        return GetRootClass(superclass);
}

BOOL IsNSObjectSubclass(Class c) {
    return GetRootClass(c) == [NSObject class];
}
```

`class_copyMethodList`

Iterate all methods
Use the pattern!

```
Method *methods = class_copyMethodList(class, NULL);
for(Method *cursor = methods; *cursor != NULL; cursor++) {
    NSLog(@"%s", sel_getName(method_getName(*cursor));
}
free(methods);
```

Do stuff with methods

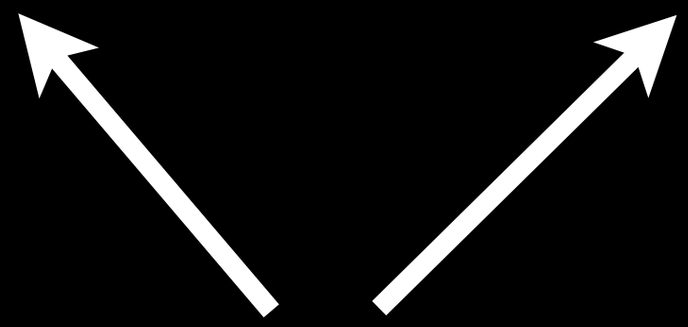
```
if(method_getNumberOfArguments(method) == 3) {
    const char *argType = method_copyArgumentType(method, 2);
    if(strcmp(argType, @encode(id)) == 0) {
        IMP implementation = method_getImplementation(method);
        ((void (*)(id, SEL, id))implementation)(object, selector, argument);
    }
    free(argType);
}
```

What?

```
((void (*)(id, SEL, id))implementation)(object, selector, argument)
```

No Really, What?

```
((void (*)(id, SEL, id))implementation)(object, selector, argument)
```



Cast to function pointer type

Returns `void`

Takes `id`, `SEL`, `id`

But...

Methods are *functions*

Two hidden parameters: `self` and `_cmd`

Look up function
Call function

That's a method call!

C's type system makes us sad. But we carry on.

class_copyIvarList

The pattern!

```
Ivar *ivars = class_copyIvarList(class, NULL);
for(Ivar *cursor = ivars; *cursor != NULL; cursor++) {
    NSLog(@"%s", ivar_getName(*cursor));
}
free(ivars);
```

Read Ivars

```
if(strcmp(ivar_getTypeEncoding(ivar), @encode(id)) == 0) {
    ptrdiff_t offset = ivar_getOffset(ivar);
    char *rawObject = (__bridge void *)object;
    char *ivarPtr = rawObject + offset;
    __unsafe_unretained id value;
    memcpy(&value, ivarPtr, sizeof(value));
}
```

`class_copyPropertyList`

Guess what

```
objc_property_t *properties = class_copyPropertyList(class, NULL);
for(objc_property_t *cursor = properties; *cursor != NULL; cursor++) {
    NSLog(@"%s", property_getName(*cursor));
}
free(properties);
```

property_copyAttributeValue

What's the type?

```
char *type = property_copyAttributeValue(*cursor, "T");
NSLog(@"type: %s", type);
free(type);
```

Is it weak?

```
char *weak = property_copyAttributeValue(*cursor, "W");
NSLog(@"weak: %s", weak != NULL ? "YES" : "NO");
free(weak);
```

Read-only? Atomic? Copy? Dynamic? Custom getter/setter?

imp_implementationWithBlock

Create LIFE!

```
IMP imp = imp_implementationWithBlock(^NSUInteger (id self) {
    return 0;
});

Method isEqual = class_getInstanceMethod([NSObject class],
                                         @selector(hash));

method_setImplementation(isEqual, imp);
```

Did somebody say swizzle?

```
SEL sel = @selector(description);
Method description = class_getInstanceMethod([NSObject class], sel);

IMP oldIMP = method_getImplementation(description);

IMP newIMP = imp_implementationWithBlock(^(id self) {
    NSString *orig = ((NSString *(*)(id, SEL))oldIMP)(self, sel);
    return [orig stringByAppendingString: @" and a duck"];
});
```

Ducks

http://richardwiseman.files.wordpress.com/2011/09/ll-final-report.pdf

Google: laughlab report

Build a Class

```
@interface Person : NSObject

- (id)initWithFirstName: (NSString *)firstName
        lastName: (NSString *)lastName age: (NSUInteger)age;

@property (readonly) NSString *firstName;
@property (readonly) NSString *lastName;
@property (readonly) NSUInteger age;

@end
```

https://gist.github.com/mikeash/9847262

objc_msgSend

```
00003b20       b1e8   cbz r0, 0x3b5e
00003b22   f8d09000   ldr.w  r9, [r0]
00003b26   f8b9c00c   ldrh.w r12, [r9, #12]
00003b2a   f8d99008   ldr.w  r9, [r9, #8]
00003b2e   ea0c0c01   and.w  r12, r12, r1
00003b32   eb0909cc   add.w  r9, r9, r12, lsl #3
00003b36   f8d9c000   ldr.w  r12, [r9]
00003b3a   ea9c0f01   teq.w  r12, r1
00003b3e       d102   bne 0x3b46
00003b40   f8d9c004   ldr.w  r12, [r9, #4]
00003b44       4760   bx  r12
00003b46   f1bc0f01   cmp.w  r12, #0x1
00003b4a       d305   blo 0x3b58
00003b4c       bf08   it  eq
00003b4e   f8d99004   ldreq.w r9, [r9, #4]
00003b52   f859cf08   ldr r12, [r9, #8]!
00003b56       e7f0   b   0x3b3a
00003b58   f8d09000   ldr.w  r9, [r0]
00003b5c       e120   b   __objc_msgSend_uncached
00003b5e   f04f0100   mov.w  r1, #0x0
00003b62       4770   bx  lr
```

objc_msgSend

```
00003b20      b1e8   cbz r0, 0x3b5e
```

- `r0` contains `self` pointer
- Compare `self` with `0` a.k.a. `nil`
- If `nil`, go to `0x3b5e`

objc_msgSend

```
00003b5e    f04f0100    mov.w   r1, #0x0
00003b62        4770    bx  lr
```

- `r0` contains `nil`
- Put `0` a.k.a. `nil` into `r1`
- `r0` and `r1` contain the return value
- Return `0`
- `nil` short circuit implementation

objc_msgSend

```
00003b22   f8d09000   ldr.w  r9, [r0]
```

- `r0` contains `self`
- Load what `self` points at
- Load the object's class

objc_msgSend

```
00003b26   f8b9c00c   ldrh.w r12, [r9, #12]
```

- `r9` contains the class
- Load offset `12` within the class
- This is the cache mask
- Basically, the cache size
- `r12` now contains the cache mask

objc_msgSend

```
00003b2a   f8d99008   ldr.w  r9, [r9, #8]
```

- `r9` contains the class
- Load offset `8` within the class
- This is the method cache
- `r9` now contains the method cache

objc_msgSend

```
00003b2e   ea0c0c01   and.w  r12, r12, r1
```

- `r1` contains the selector being sent (`_cmd`)
- `r12` contains the cache mask
- `AND` them together to produce a hash table index
- `r12` now contains the hash table index in the cache

objc_msgSend

```
00003b32   eb0909cc   add.w  r9, r9, r12, lsl #3
```

- `r9` contains the cache pointer
- `r12` contains the hash table index
- `lsl #3` means shift the index left by `3`
- Means multiply by `8`
- `r9` now points to the exact cache bucket for `_cmd`

objc_msgSend

```
00003b36   f8d9c000   ldr.w  r12, [r9]
```

- `r9` points to the cache entry
- Load the cache entry's selector
- `r12` now contains the cache entry's selector

objc_msgSend

```
00003b3a   ea9c0f01   teq.w  r12, r1
```

- Test `r12` and `r1` for equality
- `r12` contains the cache entry's selector
- `r1` contains the selector being sent

objc_msgSend

```
00003b3e        d102    bne 0x3b46
```

- If they're not equal, go to `0x3b46`
- Cache miss

objc_msgSend

```
00003b40   f8d9c004   ldr.w  r12, [r9, #4]
```

- Cache hit path
- `r9` points to the matching cache entry
- Load offset `4` into the entry
- This is the `IMP` of the matched method
- `r12` now contains the `IMP`

objc_msgSend

```
00003b44      4760   bx  r12
```

- `r12` contains the `IMP` of the target method
- Jump to it
- DONE!

objc_msgSend

```
00003b46   f1bc0f01   cmp.w   r12, #0x1
```

- Cache miss path
- `r12` contains the cache entry's selector
- Compare it with `1`

objc_msgSend

```
00003b4a        d305    blo 0x3b58
```

- Is the cache entry selector < 1?
- In other words, is it zero?
- Zero means a cache miss
- If it's a cache miss, go to `0x3b58`
- `0x3b58` just calls into C code
- Slow path

objc_msgSend

```
00003b4c       bf08   it  eq
```

- Check if `r12` contains `1`
- `1` is a sentinel for the end of the cache
- Signal to wrap to the beginning

objc_msgSend

```
00003b4e   f8d99004   ldreq.w r9, [r9, #4]
```

- Conditional execution
- If `r12` was `1`, then load `r9` from the offset `4` of `r9`
- If `1`, `r9` points to the sentinel cache entry
- Offset `4` of the sentinel entry points to the beginning
- `r9` now points to the beginning cache entry
- (Actually one prior)

objc_msgSend

```
00003b52   f859cf08   ldr r12, [r9, #8]!
```

- WTF?
- Don't panic
- Remain calm
- But really: WTF?

objc_msgSend

```
00003b52   f859cf08   ldr r12, [r9, #8]!
```

- Load offset `8` from r9 into `r12`
- `r9` points to the current cache entry
- Offset `8` is the selector of the NEXT cache entry

- What's the `!` do? It's not for emphasis
- Means to ALSO set `r9 = r9 + 8`
- Update `r9` to point to the next cache entry
- `r9` now points to the next cache entry

objc_msgSend

```
00003b56        e7f0    b   0x3b3a

00003b3a    ea9c0f01    teq.w   r12, r1
```

- Go back to `0x3b3a`
- What's at `0x3b3a`?
- It's the cache entry comparison again
- Closing brace of a `while()` loop

`objc_msgSend`

- In short:
- Check for `nil` and short circuit
- Load the class cache
- Linear chained hash table
- If found, jump to the method
- If not found, abort to C code to handle it

objc_msgSend

objc_msgSend

Hire me

mike@mikeash.com
http://plausible.coop