Message in a Bottle

Sending Messages, Caches, PICs, and More

Pablo Tesone - 02/12/2020

Sending Messages

A bytecode view

```
sendingAMessage

    ^ 42 aMessage
```

Source Code

```
41 <20> pushConstant: 42
42 <81> send: aMessage
43 <5C> returnTop
```

Bytecode

Sending Messages

A bytecode view

```
sendingAMessage

    ^ 42 aMessage
```

Source Code

```
41 <20> pushConstant: 42
42 <81> send: aMessage
43 <5C> returnTop
```

Bytecode

Sending a message activates a method on a given receiver with the required parameters.

Sending Messages

A bytecode view

```
sendingAMessage

    ^ 42 aMessage
```

Source Code

```
41 <20> pushConstant: 42
42 <81> send: aMessage
43 <5C> returnTop
```

Bytecode

Sending a message activates a method on a given receiver with the required parameters.

Sending Messages

A bytecode view

```
sendingAMessage

    ^ 42 aMessage
```

Source Code

```
41 <20> pushConstant: 42
42 <81> send: aMessage
43 <5C> returnTop
```

Bytecode

Sending a message activates a method on a given receiver with the required parameters.

Gotta Catch'Em All!

Sending Messages

Parameters & Receiver

```
41 <20> pushConstant: 42
42 <81> send: aMessage
43 <5C> returnTop
```

Bytecode

Sending Messages

Parameters & Receiver

```
41 <20> pushConstant: 42
42 <81> send: aMessage
43 <5C> returnTop
```

Bytecode

Receiver & Parameters are pushed to the Stack, just before the send message.

Sending Messages

Let’s find the method

```
41 <20> pushConstant: 42
42 <81> send: aMessage
43 <5C> returnTop
```

Bytecode

To get the method to activate, we need to do the lookup.

Sending Messages

Everything starts with a Selector

```
41 <20> pushConstant: 42
42 <81> send: aMessage
43 <5C> returnTop
```

Bytecode

We need to get the selector. How we know is this one?

Sending Messages

Everything starts with a Selector

```
41 <20> pushConstant: 42
42 <81> send: aMessage
43 <5C> returnTop
```

Bytecode

We need to get the selector. How we know is this one?

```
42 <81> send: aMessage
```

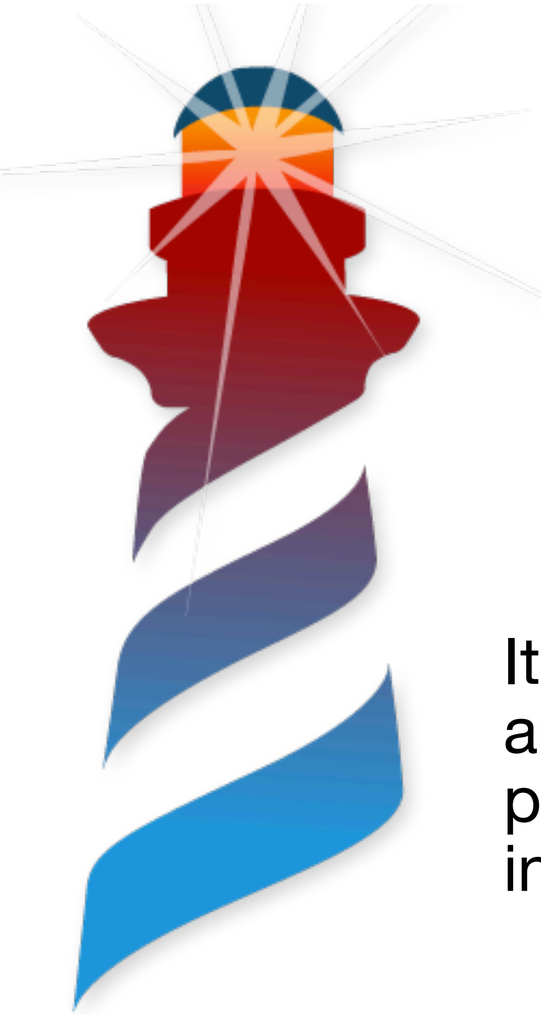
It is encoded (or kind of) in the bytecode

Sending Messages

The Bytecode is your friend

```
42 <81> send: aMessage
```

Our Bytecode



Sending Messages

The Bytecode is your friend

```
42 <81> send: aMessage
```

Our Bytecode

It encodes the # of arguments and the position of the selector in the literals

Sending Messages

The Bytecode is your friend

```
42 <81> send: aMessage
```

Our Bytecode

It encodes the # of arguments and the position of the selector in the literals

```
42
#aMessage
#sendingAMessage
#VMJitMethodTest->VMJitMethodTest
```

Our Literals (zero-based)

Sending Messages

The Bytecode is your friend

```
42 <81> send: aMessage
```

Our Bytecode

It encodes the # of arguments and the position of the selector in the literals

```
42
#aMessage
#sendingAMessage
#VMJitMethodTest->VMJitMethodTest
```

Our Literals (zero-based)

Sending Messages

The Bytecode is your friend

```
42 <81> send: aMessage
```

Our Bytecode

It encodes the # of arguments and the position of the selector in the literals

```
42
#aMessage
#sendingAMessage
#VMJitMethodTest->VMJitMethodTest
```

Our Literals (zero-based)

All Bytecodes

16r80	16r81	16r82	16r83	…
Send 0 args Literal 0	Send 0 args Literal 1	Send 0 args Literal 2	Send 0 args Literal 3	…
16r90	16r91	16r92	16r93	…
Send 1 args Literal 0	Send 1 args Literal 1	Send 1 args Literal 2	Send 1 args Literal 3	…
16rA0	16rA1	16rA2	16rA3	…
Send 2 args Literal 0	Send 2 args Literal 1	Send 2 args Literal 2	Send 2 args Literal 3	…

Sending Messages

The Bytecode is your friend

```
42 <81> send: aMessage
```

Our Bytecode

It encodes the # of arguments and the position of the selector in the literals

```
42
#aMessage
#sendingAMessage
#VMJitMethodTest->VMJitMethodTest
```

Our Literals (zero-based)

All Bytecodes

16r80	16r81	16r82	16r83	…
Send 0 args Literal 0	Send 0 args Literal 1	Send 0 args Literal 2	Send 0 args Literal 3	…
16r90	16r91	16r92	16r93	…
Send 1 args Literal 0	Send 1 args Literal 1	Send 1 args Literal 2	Send 1 args Literal 3	…
16rA0	16rA1	16rA2	16rA3	…
Send 2 args Literal 0	Send 2 args Literal 1	Send 2 args Literal 2	Send 2 args Literal 3	…

Sending Messages

The Bytecode is your friend

```
42 <81> send: aMessage
```

Our Bytecode

It encodes the # of arguments and the position of the selector in the literals

```
42
#aMessage
#sendingAMessage
#VMJitMethodTest->VMJitMethodTest
```

Our Literals (zero-based)

All Bytecodes

16r80	16r81	16r82	16r83	…
Send 0 args Literal 0	Send 0 args Literal 1	Send 0 args Literal 2	Send 0 args Literal 3	…
16r90	16r91	16r92	16r93	…
Send 1 args Literal 0	Send 1 args Literal 1	Send 1 args Literal 2	Send 1 args Literal 3	…
16rA0	16rA1	16rA2	16rA3	…
Send 2 args Literal 0	Send 2 args Literal 1	Send 2 args Literal 2	Send 2 args Literal 3	…

Not all possible combinations…. We have an extended send message bytecode (16rEA)

Method Lookup

This one… I hope you know.

- We have:
  - The Receiver
  - The Selector

Method Lookup

This one… I hope you know.

- We have:
  - The Receiver
  - The Selector

We need to get the Method.

Method Lookup

This one… I hope you know.

- We have:
  - The Receiver
  - The Selector

We need to get the Method.

Here the VM do the method lookup, as we now it.

Method Lookup

But… there is always a but

- Doing the lookup every time is expensive.
- The hierarchies can be deep!!!
- We need an improvement.

Method Lookup

But… there is always a but

- Doing the lookup every time is expensive.
- The hierarchies can be deep!!!
- We need an improvement.

Let’s add a cache!!

Method Lookup

But… there is always a but

- Let’s add a Global Method Cache
- We have a Hashed Map
- Key = ClassTag + Selector
- Value = CompiledMethod

Global Method Cache

ClassTag	Selector	CompiledMethod
ClassTag(X)	#aMessage	X >> #aMessage

Hash(X,aMessage)

1024 Entries

A Hash map, with the class and selector.

Global Method Cache

Handling collisions

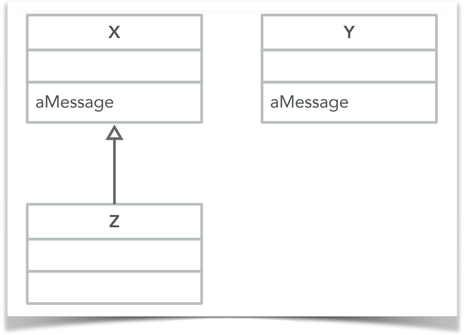
- We calculate a hash of ClassTag and Selector
- Collisions occurs (we only have 1024 slots)
- We do three probes (multiplying by 2 the hash)

“You have 3 questions”

What happens if there is Polymorphism?

	ClassTag	Selector	CompiledMethod
Hash(Y,aMessage)	ClassTag(Y)	#aMessage	Y >> #aMessage
Hash(X,aMessage)	ClassTag(X)	#aMessage	X >> #aMessage

1024 Entries



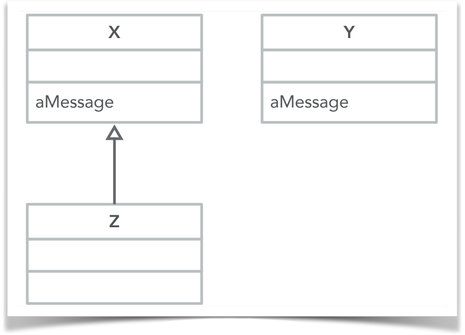
Our Classes

A new entry is added with the correct hash

What happens if there is inheritance?

	ClassTag	Selector	CompiledMethod
Hash(Y,aMessage)	ClassTag(Y)	#aMessage	Y >> #aMessage
Hash(X,aMessage)	ClassTag(X)	#aMessage	X >> #aMessage

1024 Entries

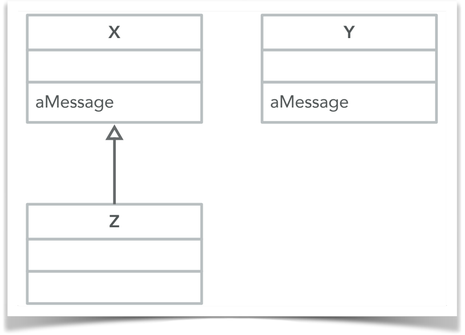


Our Classes

What happens if there is inheritance?

	ClassTag	Selector	CompiledMethod
Hash(Y,aMessage)	ClassTag(Y)	#aMessage	Y >> #aMessage
Hash(X,aMessage)	ClassTag(X)	#aMessage	X >> #aMessage
Hash(Z,aMessage)	ClassTag(Z)	#aMessage	X >> #aMessage

1024 Entries



Our Classes

The Hierarchy is flatten, but the compiled methods are shared.

A message is sent

Finally….

- With
  - Receiver
  - Arguments
  - Method

A message is sent

Finally….

- With
  - Receiver
  - Arguments
  - Method

We can finally activate the method!!

A message is sent

Finally….

- With
  - Receiver
  - Arguments
  - Method

We can finally activate the method!!

And… if previously found in the cache, compile it to machine code

How is it compiled!!!

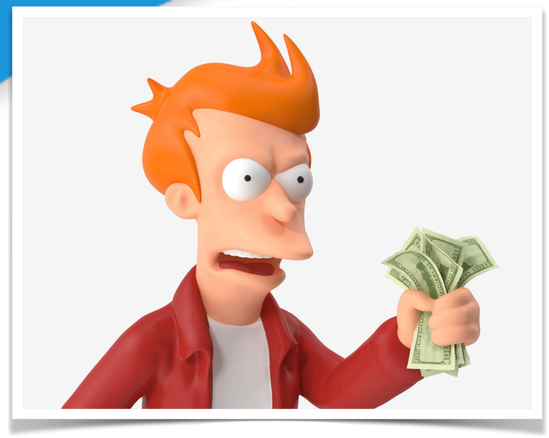
Shut-up and show me code!!!

41 <20> pushConstant: 42

42 <81> send: aMessage

43 <5C> returnTop

Bytecode



```
Abort:
                    MoveCqR 0 ReceiverResultReg
                    PushR LinkReg
                    Call abortTrampoline
                    AlignmentNops 8
Entry:
                    AndCqRR 7 ReceiverResultReg
                    JumpNonZero Label2
                    MoveMwrR 0 ReceiverResultReg TempReg
                    AndCqR 16r3FFFFF TempReg
                    Nop
                    Nop
                    Nop
Label 2:
                    CmpRR ClassReg TempReg
                    JumpNonZero Abort
                    PushR ReceiverResultReg
Label 3:
                    PushR LinkReg
                    PushR FPReg
                    MoveRR SPReg FPReg
                    PushCw 16r1000430
                    MoveCqR 16r1016400 SendNumArgsReg
                    PushR SendNumArgsReg
                    PushR ReceiverResultReg

                    MoveAwR 16r7FFFFFFFFFFFFC88 TempReg
                    CmpRR TempReg SPReg
                    JumpBelow Abort

                    MoveCqR 16r151 ReceiverResultReg
                    MovePatcheableC32R 1 ClassReg
                    Call send0ArgTrampoline

                    MoveRR FPReg SPReg
                    PopR FPReg
                    PopR LinkReg
                    RetN 8
```

How is it compiled!!!

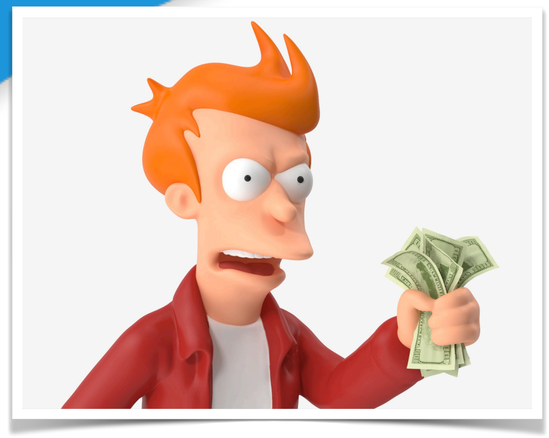
Let it goes slowly

41 <20> pushConstant: 42

42 <81> send: aMessage

43 <5C> returnTop

Bytecode



```
Abort:
                MoveCqR 0 ReceiverResultReg
                PushR LinkReg
                Call abortTrampoline
                AlignmentNops 8
Entry:
                AndCqRR 7 ReceiverResultReg TempReg
                JumpNonZero Label2
                MoveMwrR 0 ReceiverResultReg TempReg
                AndCqR 16r3FFFFF TempReg
                Nop
                Nop
                Nop

Label 2:
                CmpRR ClassReg TempReg
                JumpNonZero Abort
                PushR ReceiverResultReg
Label 3:
                PushR LinkReg
                PushR FPReg
                MoveRR SPReg FPReg
                PushCw 16r1000430
                MoveCqR 16r1016400 SendNumArgsReg
                PushR SendNumArgsReg
                PushR ReceiverResultReg

                MoveAwR 16r7FFFFFFFFFFFC88 TempReg
                CmpRR TempReg SPReg
                JumpBelow Abort

                MoveCqR 16r151 ReceiverResultReg
                MovePatcheableC32R 1 ClassReg
                Call send0ArgTrampoline

                MoveRR FPReg SPReg
                PopR FPReg
                PopR LinkReg
                RetN 8
```

How is it compiled!!!

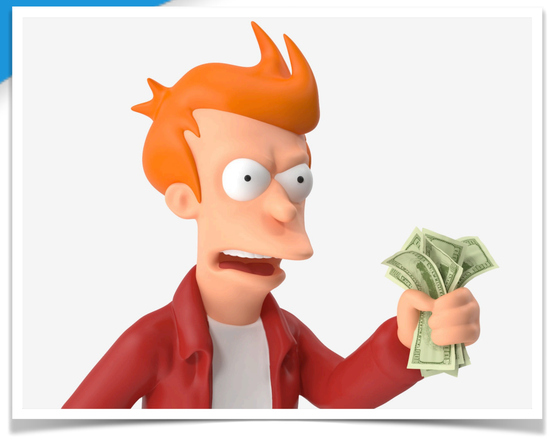
Let it goes slowly

41 <20> pushConstant: 42

42 <81> send: aMessage

43 <5C> returnTop

Bytecode



```
Abort:
                MoveCqR 0 ReceiverResultReg
                PushR LinkReg
                Call abortTrampoline
                AlignmentNops 8
Entry:
                AndCqRR 7 ReceiverResultReg TempReg
                JumpNonZero Label2
                MoveMwrR 0 ReceiverResultReg TempReg
                AndCqR 16r3FFFFF TempReg
                Nop
                Nop
                Nop

Label 2:
                CmpRR ClassReg TempReg
                JumpNonZero Abort
                PushR ReceiverResultReg
Label 3:
                PushR LinkReg
                PushR FPReg
                MoveRR SPReg FPReg
                PushCw 16r1000430
                MoveCqR 16r1016400 SendNumArgsReg
                PushR SendNumArgsReg
                PushR ReceiverResultReg

                MoveAwR 16r7FFFFFFFFFFFFC88 TempReg
                CmpRR TempReg SPReg
                JumpBelow Abort

                MoveCqR 16r151 ReceiverResultReg
                MovePatcheableC32R 1 ClassReg
                Call send0ArgTrampoline

                MoveRR FPReg SPReg
                PopR FPReg
                PopR LinkReg
                RetN 8
```

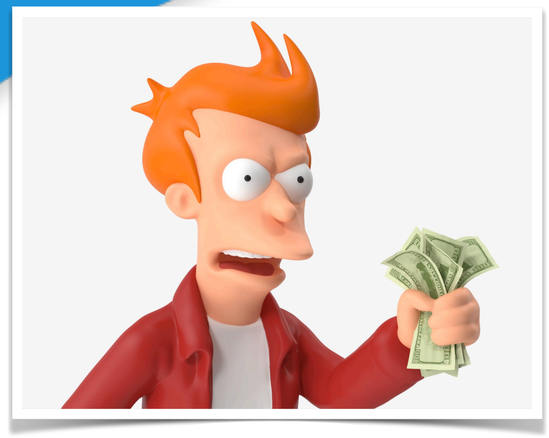
Entry point & Type check

How is it compiled!!!

Let it goes slowly

```
41 <20> pushConstant: 42
42 <81> send: aMessage
43 <5C> returnTop
```

Bytecode



```
Abort:
                MoveCqR 0 ReceiverResultReg
                PushR LinkReg
                Call abortTrampoline
                AlignmentNops 8
Entry:
                AndCqRR 7 ReceiverResultReg TempReg
                JumpNonZero Label2
                MoveMwrR 0 ReceiverResultReg TempReg
                AndCqR 16r3FFFFF TempReg
                Nop
                Nop
                Nop

Label 2:
                CmpRR ClassReg TempReg
                JumpNonZero Abort
                PushR ReceiverResultReg

Label 3:
                PushR LinkReg
                PushR FPReg
                MoveRR SPReg FPReg
                PushCw 16r1000430
                MoveCqR 16r1016400 SendNumArgsReg
                PushR SendNumArgsReg
                PushR ReceiverResultReg

                MoveAwR 16r7FFFFFFFFFFFFC88 TempReg
                CmpRR TempReg SPReg
                JumpBelow Abort

                MoveCqR 16r151 ReceiverResultReg
                MovePatcheableC32R 1 ClassReg
                Call send0ArgTrampoline

                MoveRR FPReg SPReg
                PopR FPReg
                PopR LinkReg
                RetN 8
```

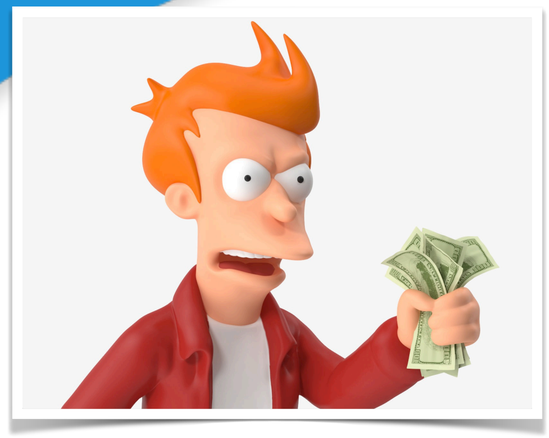
Calling the Abort trampoline

How is it compiled!!!

Let it goes slowly

```
41 <20> pushConstant: 42
42 <81> send: aMessage
43 <5C> returnTop
```

Bytecode



```
Abort:
                MoveCqR 0 ReceiverResultReg
                PushR LinkReg
                Call abortTrampoline
                AlignmentNops 8
Entry:
                AndCqRR 7 ReceiverResultReg TempReg
                JumpNonZero Label2
                MoveMwrR 0 ReceiverResultReg TempReg
                AndCqR 16r3FFFFF TempReg
                Nop
                Nop
                Nop

Label 2:
                CmpRR ClassReg TempReg
                JumpNonZero Abort
                PushR ReceiverResultReg
Label 3:
                PushR LinkReg
                PushR FPReg
                MoveRR SPReg FPReg
                PushCw 16r1000430
                MoveCqR 16r1016400 SendNumArgsReg
                PushR SendNumArgsReg
                PushR ReceiverResultReg

                MoveAwR 16r7FFFFFFFFFFFFC88 TempReg
                CmpRR TempReg SPReg
                JumpBelow Abort

                MoveCqR 16r151 ReceiverResultReg
                MovePatcheableC32R 1 ClassReg
                Call send0ArgTrampoline

                MoveRR FPReg SPReg
                PopR FPReg
                PopR LinkReg
                RetN 8
```

Frame building

How is it compiled!!!

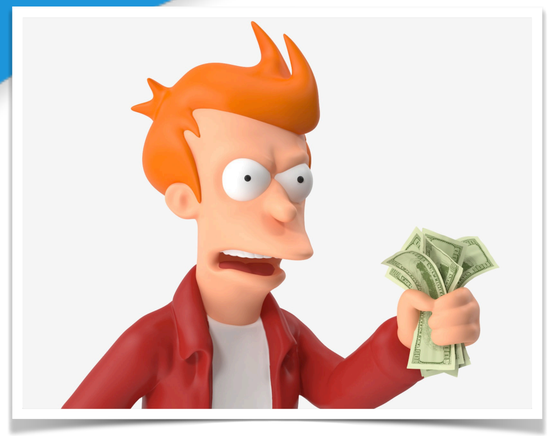
Let it goes slowly

41 <20> pushConstant: 42

42 <81> send: aMessage

43 <5C> returnTop

Bytecode



```
Abort:
                MoveCqR 0 ReceiverResultReg
                PushR LinkReg
                Call abortTrampoline
                AlignmentNops 8
Entry:
                AndCqRR 7 ReceiverResultReg TempReg
                JumpNonZero Label2
                MoveMwrR 0 ReceiverResultReg TempReg
                AndCqR 16r3FFFFF TempReg
                Nop
                Nop
                Nop
Label 2:
                CmpRR ClassReg TempReg
                JumpNonZero Abort
                PushR ReceiverResultReg
Label 3:
                PushR LinkReg
                PushR FPReg
                MoveRR SPReg FPReg
                PushCw 16r1000430
                MoveCqR 16r1016400 SendNumArgsReg
                PushR SendNumArgsReg
                PushR ReceiverResultReg

                MoveAwR 16r7FFFFFFFFFFFC88 TempReg
                CmpRR TempReg SPReg
                JumpBelow Abort

                MoveCqR 16r151 ReceiverResultReg
                MovePatcheableC32R 1 ClassReg
                Call send0ArgTrampoline

                MoveRR FPReg SPReg
                PopR FPReg
                PopR LinkReg
                RetN 8
```

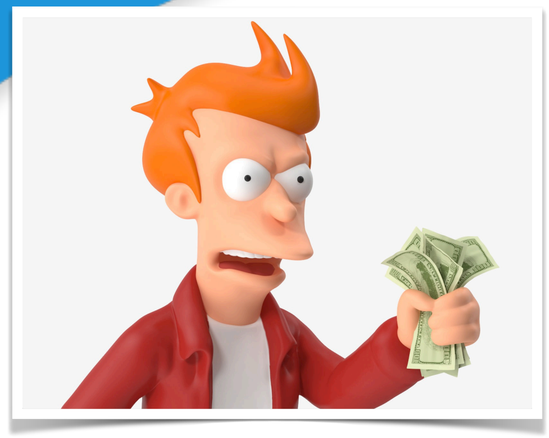
Frame un-building & return

How is it compiled!!!

Let it goes slowly

```
41 <20> pushConstant: 42
42 <81> send: aMessage
43 <5C> returnTop
```

Bytecode



```
Abort:
                MoveCqR 0 ReceiverResultReg
                PushR LinkReg
                Call abortTrampoline
                AlignmentNops 8
Entry:
                AndCqRR 7 ReceiverResultReg TempReg
                JumpNonZero Label2
                MoveMwrR 0 ReceiverResultReg TempReg
                AndCqR 16r3FFFFF TempReg
                Nop
                Nop
                Nop

Label 2:
                CmpRR ClassReg TempReg
                JumpNonZero Abort
                PushR ReceiverResultReg
Label 3:
                PushR LinkReg
                PushR FPReg
                MoveRR SPReg FPReg
                PushCw 16r1000430
                MoveCqR 16r1016400 SendNumArgsReg
                PushR SendNumArgsReg
                PushR ReceiverResultReg

                MoveAwR 16r7FFFFFFFFFFFFC88 TempReg
                CmpRR TempReg SPReg
                JumpBelow Abort

                MoveCqR 16r151 ReceiverResultReg
                MovePatcheableC32R 1 ClassReg
                Call send0ArgTrampoline

                MoveRR FPReg SPReg
                PopR FPReg
                PopR LinkReg
                RetN 8
```

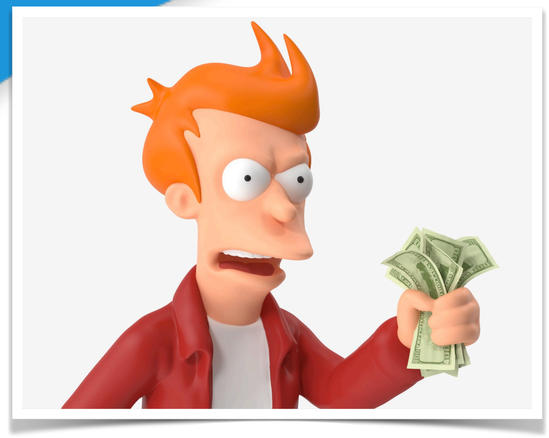
Stack overflow check & maybe context switch

How is it compiled!!!

Let it goes slowly

```
41 <20> pushConstant: 42
42 <81> send: aMessage
43 <5C> returnTop
```

Bytecode



```
Abort:
                MoveCqR 0 ReceiverResultReg
                PushR LinkReg
                Call abortTrampoline
                AlignmentNops 8
Entry:
                AndCqRR 7 ReceiverResultReg TempReg
                JumpNonZero Label2
                MoveMwrR 0 ReceiverResultReg TempReg
                AndCqR 16r3FFFFF TempReg
                Nop
                Nop
                Nop
Label 2:
                CmpRR ClassReg TempReg
                JumpNonZero Abort
                PushR ReceiverResultReg
Label 3:
                PushR LinkReg
                PushR FPReg
                MoveRR SPReg FPReg
                PushCw 16r1000430
                MoveCqR 16r1016400 SendNumArgsReg
                PushR SendNumArgsReg
                PushR ReceiverResultReg

                MoveAwR 16r7FFFFFFFFFFFFC88 TempReg
                CmpRR TempReg SPReg
                JumpBelow Abort

                MoveCqR 16r151 ReceiverResultReg
                MovePatcheableC32R 1 ClassReg
                Call send0ArgTrampoline

                MoveRR FPReg SPReg
                PopR FPReg
                PopR LinkReg
                RetN 8
```

Sending Message

I want a Zoom
Let’s Analyse it

```
MoveCqR 16r151 ReceiverResultReg
MovePatcheableC32R 1 ClassReg
Call send0ArgTrampoline
```

41 <20> pushConstant: 42

42 <81> send: aMessage

43 <5C> returnTop

Bytecode (friendly reminder)

I want a Zoom
Let’s Analyse it

```
MoveCqR 16r151 ReceiverResultReg
MovePatcheableC32R 1 ClassReg
Call send0ArgTrampoline
```

41 <20> pushConstant: 42

42 <81> send: aMessage

43 <5C> returnTop

Bytecode (friendly reminder)

I want a Zoom
Let's Analyse it

```
MoveCqR 16r151 ReceiverResultReg
MovePatcheableC32R 1 ClassReg
Call send0ArgTrampoline
```

Move Receiver (42) to the ReceiverResultReg (42 as SmallInteger => 16r151)

41 <20> pushConstant: 42

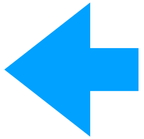
42 <81> send: aMessage

43 <5C> returnTop

Bytecode (friendly reminder)

I want a Zoom
Let’s Analyse it

```
MoveCqR 16r151 ReceiverResultReg
MovePatcheableC32R 1 ClassReg
Call send0ArgTrampoline
```



41 <20> pushConstant: 42

42 <81> send: aMessage

43 <5C> returnTop

Bytecode (friendly reminder)

I want a Zoom
Let's Analyse it

```
MoveCqR 16r151 ReceiverResultReg
MovePatcheableC32R 1 ClassReg
Call send0ArgTrampoline
```

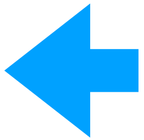
Move Selector Index to ClassReg

```
41 <20> pushConstant: 42
42 <81> send: aMessage
43 <5C> returnTop
```

Bytecode (friendly reminder)

I want a Zoom
Let’s Analyse it

```
MoveCqR 16r151 ReceiverResultReg
MovePatcheableC32R 1 ClassReg
Call send0ArgTrampoline
```



41 <20> pushConstant: 42

42 <81> send: aMessage

43 <5C> returnTop

Bytecode (friendly reminder)

I want a Zoom
Let's Analyse it

```
MoveCqR 16r151 ReceiverResultReg
MovePatcheableC32R 1 ClassReg
Call send0ArgTrampoline
```

We call the trampoline to do the send.

```
41 <20> pushConstant: 42
42 <81> send: aMessage
43 <5C> returnTop
```

Bytecode (friendly reminder)

I want a Zoom
Let's Analyse it

```
MoveCqR 16r151 ReceiverResultReg
MovePatcheableC32R 1 ClassReg
Call send0ArgTrampoline
```

41 <20> pushConstant: 42

42 <81> send: aMessage

43 <5C> returnTop

Bytecode (friendly reminder)

We call the trampoline to do the send.

We have specialised cases for 0, 1 & 2 arguments.

For 3 or more…. easy add a parameter

I want a Zoom
More arguments

- If the number of arguments is 1 or 2, they go in registers (Arg0Reg & Arg1Reg).
- If there are more arguments they go in the stack.
- If there are 3 or more arguments, the number of arguments go in the SendNumArgsReg register.

```
… Receiver & Args already in the stack …
MovePatcheableC32R 3 SendNumArgsReg
MovePatcheableC32R 1 ClassReg
Call sendNArgTrampoline
```

We can do it better…

Monomorphic Calls

- This solution is good but:
  - We are jumping to the interpreter in each message send.
  - We are doing a cache lookup every time, and maybe a full lookup.
  - The most probable case is that we are always activating the same compiled method.
  - If the target method is already compiled in machine code… why not call it directly?

What we want!

Monomorphic Calls

- Once we have found the method to execute, why not patch the message send to call the correct method directly.

```
MoveCqR 16r151 ReceiverResultReg
MovePatcheableC32R 1 ClassReg
Call send0ArgTrampoline
```

```
MoveCqR 16r151 ReceiverResultReg
MovePatcheableC32R 16r5588 ClassReg
Call 16rFF001122
```

```
41 <20> pushConstant: 42
42 <81> send: aMessage
43 <5C> returnTop
```

Bytecode (friendly reminder)

What we want!

Monomorphic Calls

- Once we have found the method to execute, why not patch the message send to call the correct method directly.

```
MoveCqR 16r151 ReceiverResultReg
MovePatcheableC32R 1 ClassReg
Call send0ArgTrampoline
```

```
MoveCqR 16r151 ReceiverResultReg
MovePatcheableC32R 16r5588 ClassReg
Call 16rFF001122
```

Entry address of the JITed method of `SmallInteger >> #aMessage`

```
41 <20> pushConstant: 42
42 <81> send: aMessage
43 <5C> returnTop
```

Bytecode (friendly reminder)

What we want!

Monomorphic Calls

- Once we have found the method to execute, why not patch the message send to call the correct method directly.

```
MoveCqR 16r151 ReceiverResultReg
MovePatcheableC32R 1 ClassReg
Call send0ArgTrampoline
```

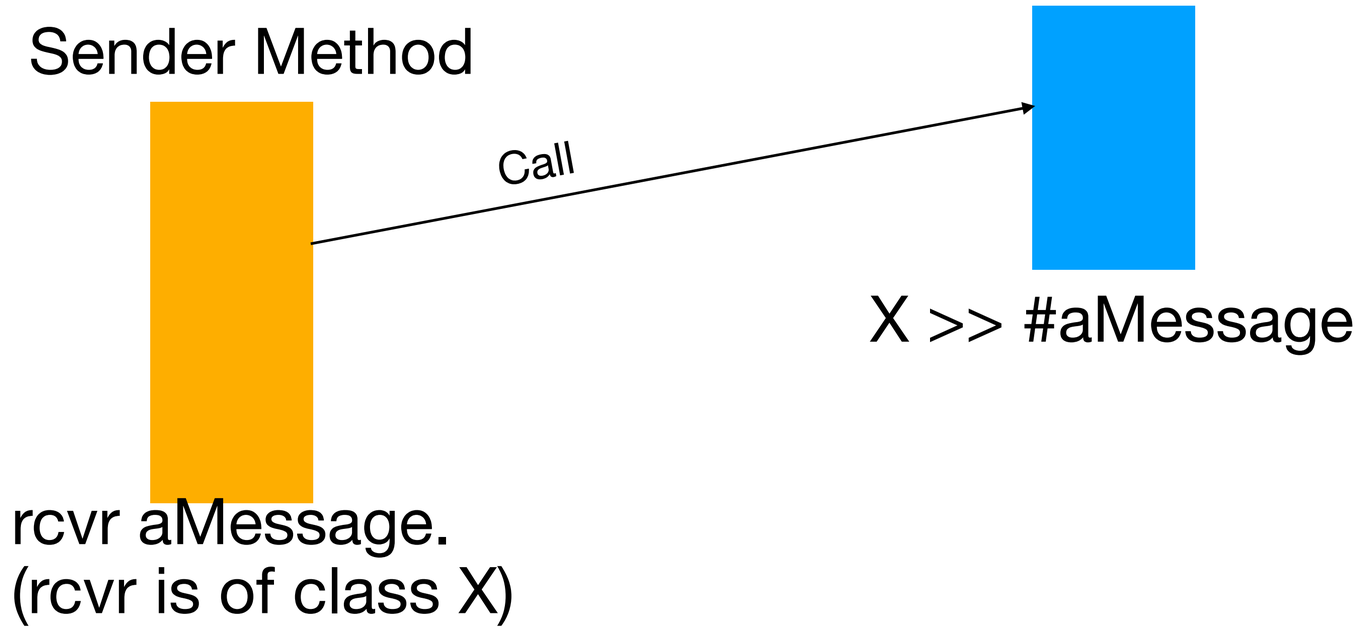
```
MoveCqR 16r151 ReceiverResultReg
MovePatcheableC32R 16r5588 ClassReg
Call 16rFF001122
```

Expected Class Tag of the Receiver

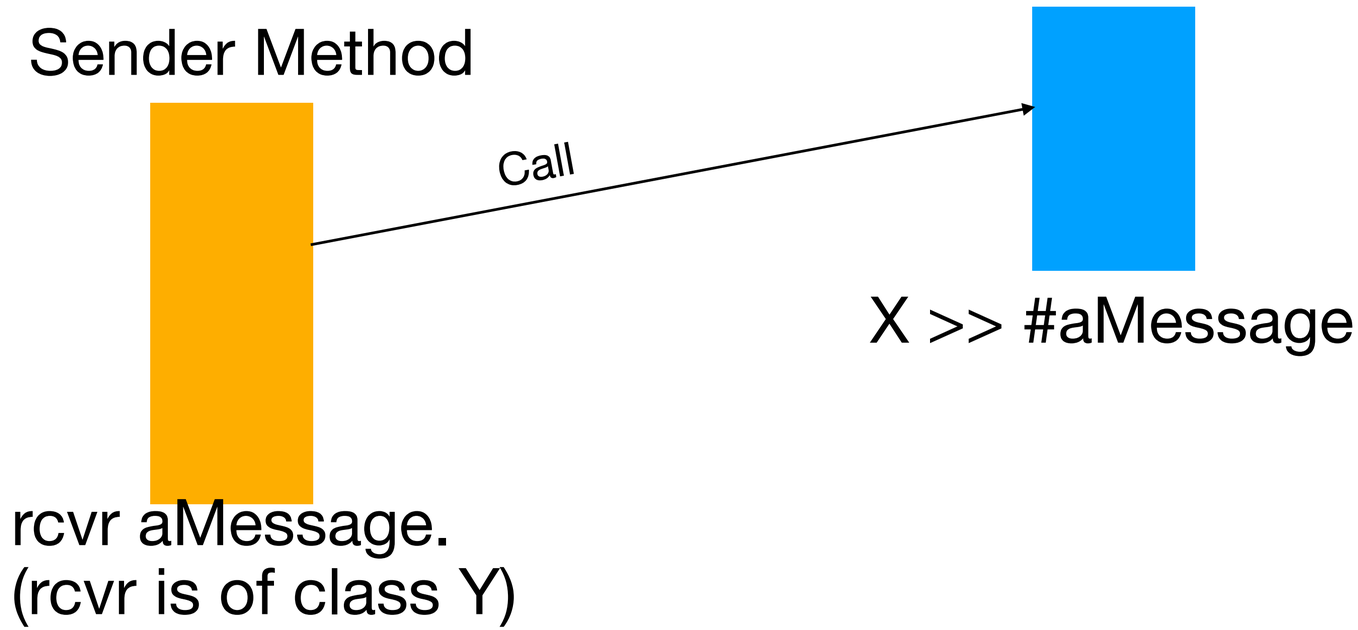
```
41 <20> pushConstant: 42
42 <81> send: aMessage
43 <5C> returnTop
```

Bytecode (friendly reminder)

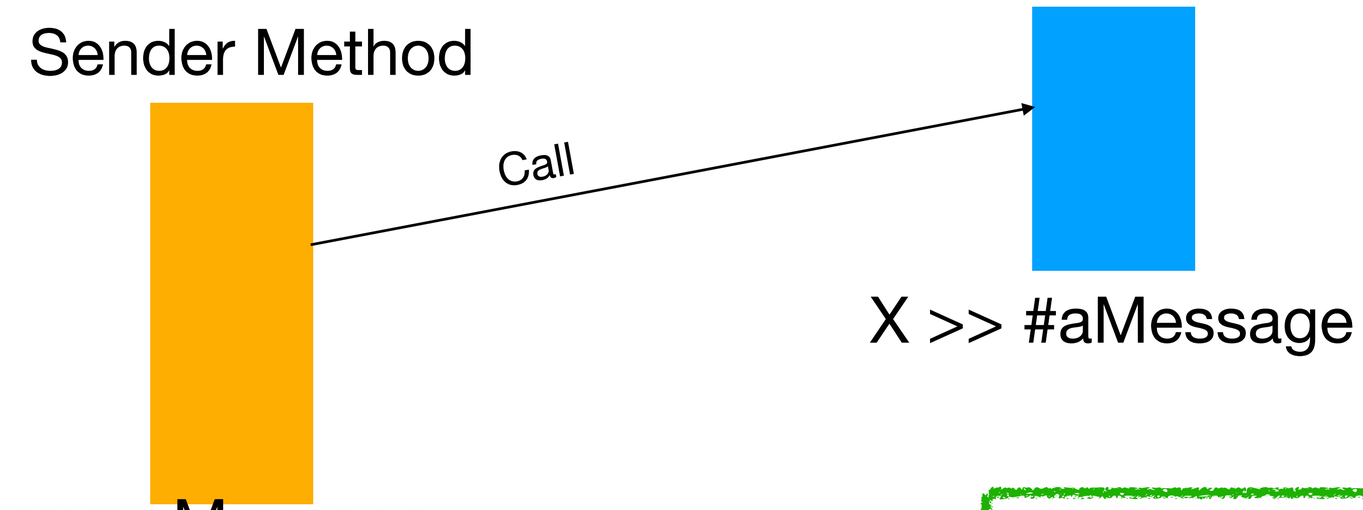
Executing the call



If the receiver is not the expected of the class?

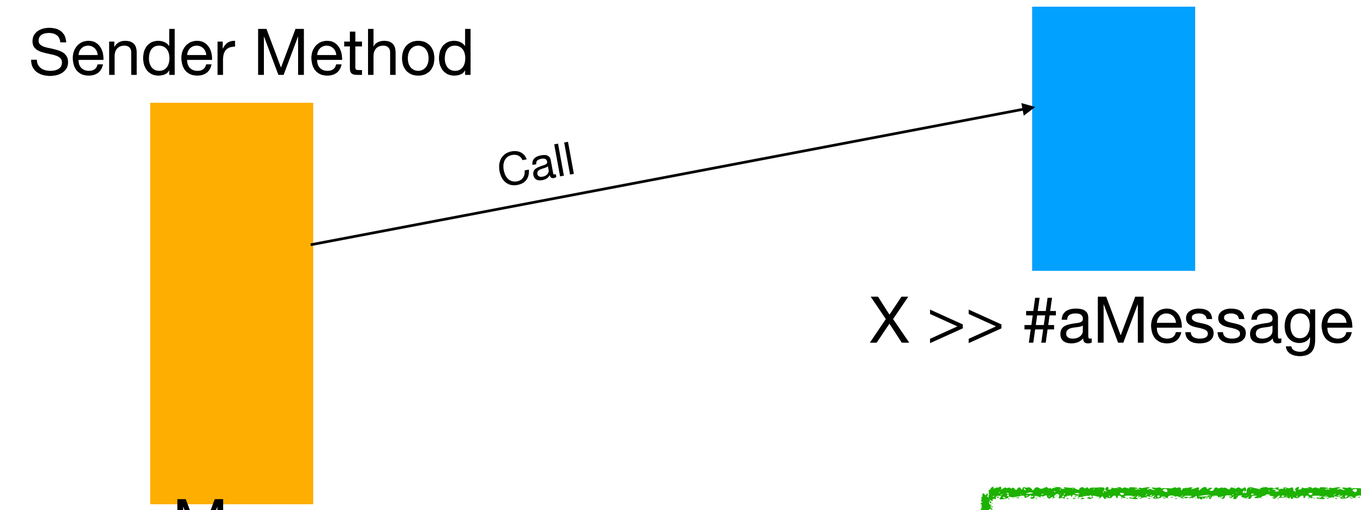


If the receiver is not the expected of the class?

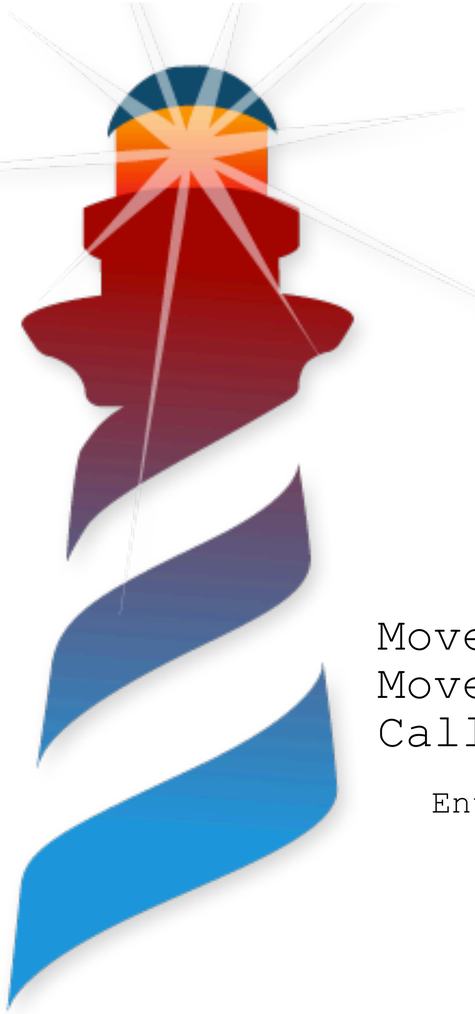


We should not execute
X >> #aMessage

If the receiver is not the expected of the class?



We need to validate this condition



Why Sending the Expected Class Tag?

Monomorphic Calls

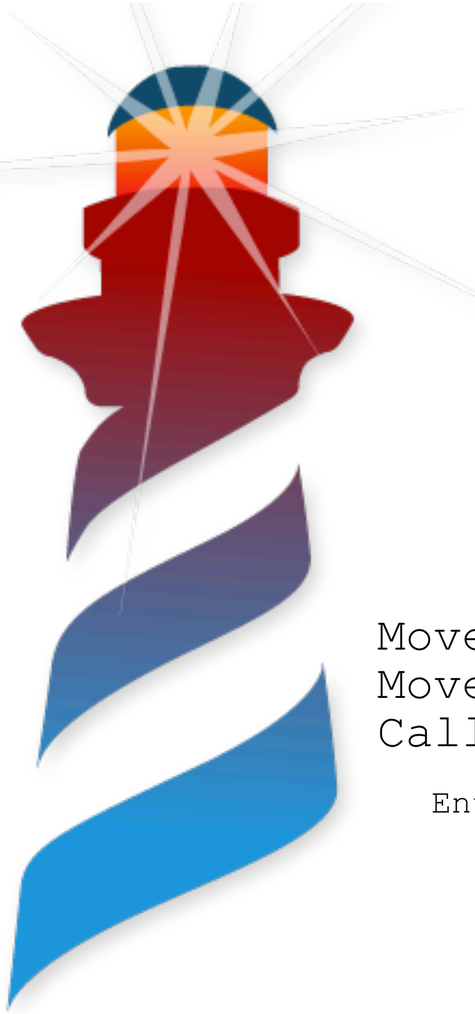
- So we can test if we are in the correct receiver. If not the abort trampoline is call.

```
MoveCqR 16r151 ReceiverResultReg
MovePatcheableC32R 1 ClassReg
Call send0ArgTrampoline
```

```
MoveCqR 16r151 ReceiverResultReg
MovePatcheableC32R 16r5588 ClassReg
Call 16rFF001122
```

```
Entry:
        AndCqRR 7 ReceiverResultReg TempReg
        JumpNonZero Label2
        MoveMwrR 0 ReceiverResultReg TempReg
        AndCqR 16r3FFFFF TempReg
        Nop
        Nop
        Nop

Label 2:
        CmpRR ClassReg TempReg
        JumpNonZero Abort
        PushR ReceiverResultReg
```



Why Sending the Expected Class Tag?

Monomorphic Calls

- So we can test if we are in the correct receiver. If not the abort trampoline is call.

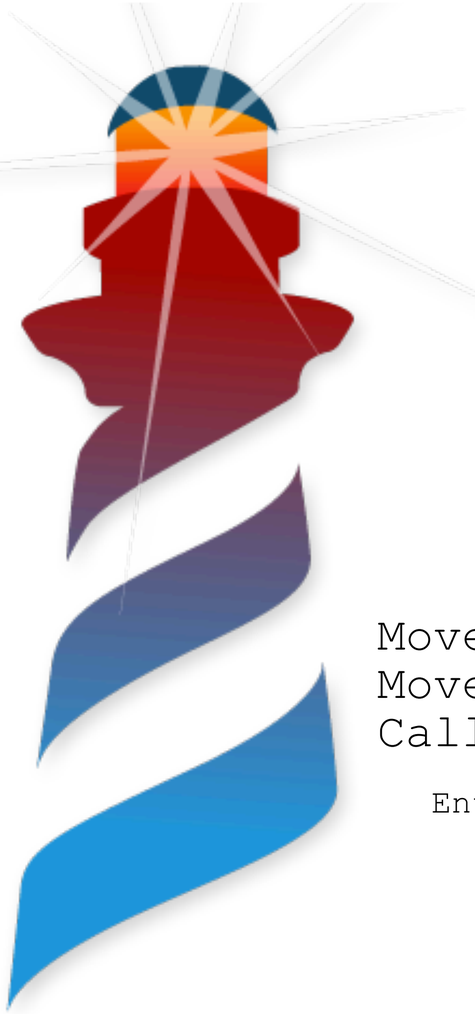
```
MoveCqR 16r151 ReceiverResultReg
MovePatcheableC32R 1 ClassReg
Call send0ArgTrampoline
```

```
MoveCqR 16r151 ReceiverResultReg
MovePatcheableC32R 16r5588 ClassReg
Call 16rFF001122
```

```
Entry:
        AndCqRR 7 ReceiverResultReg TempReg
        JumpNonZero Label2
        MoveMwrR 0 ReceiverResultReg TempReg
        AndCqR 16r3FFFFF TempReg
        Nop
        Nop
        Nop

Label 2:
        CmpRR ClassReg TempReg
        JumpNonZero Abort
        PushR ReceiverResultReg
```

Extract the Class tag if it is an immediate



Why Sending the Expected Class Tag?

Monomorphic Calls

- So we can test if we are in the correct receiver. If not the abort trampoline is call.

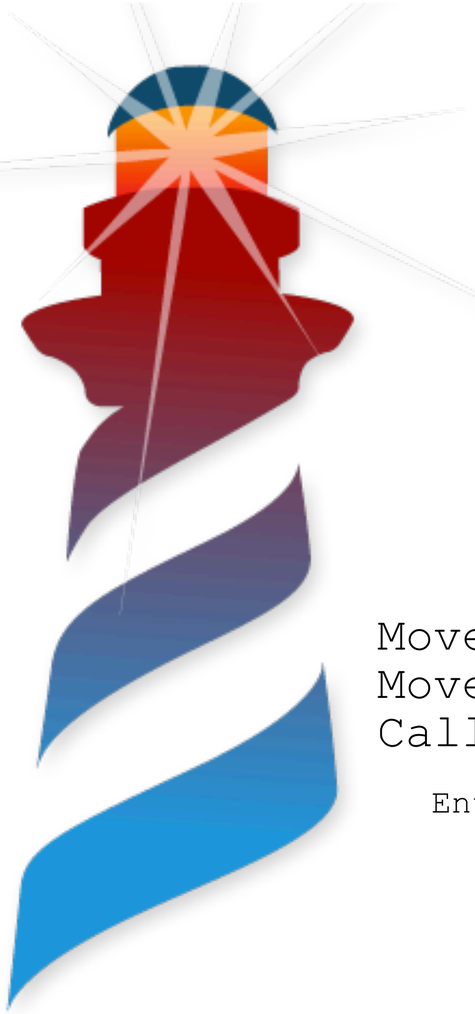
```
MoveCqR 16r151 ReceiverResultReg
MovePatcheableC32R 1 ClassReg
Call send0ArgTrampoline
```

```
MoveCqR 16r151 ReceiverResultReg
MovePatcheableC32R 16r5588 ClassReg
Call 16rFF001122
```

```
Entry:
        AndCqRR 7 ReceiverResultReg TempReg
        JumpNonZero Label2
        MoveMwrR 0 ReceiverResultReg TempReg
        AndCqR 16r3FFFFF TempReg
        Nop
        Nop
        Nop

Label 2:
        CmpRR ClassReg TempReg
        JumpNonZero Abort
        PushR ReceiverResultReg
```

Extract the Class tag if non immediate.



Why Sending the Expected Class Tag?

Monomorphic Calls

- So we can test if we are in the correct receiver. If not the abort trampoline is call.

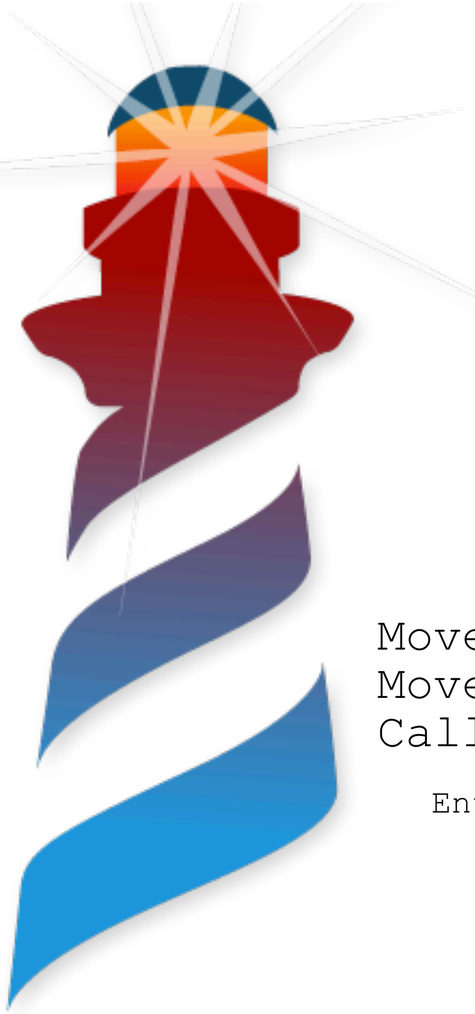
```
MoveCqR 16r151 ReceiverResultReg
MovePatcheableC32R 1 ClassReg
Call send0ArgTrampoline
```

```
MoveCqR 16r151 ReceiverResultReg
MovePatcheableC32R 16r5588 ClassReg
Call 16rFF001122
```

```
Entry:
        AndCqRR 7 ReceiverResultReg TempReg
        JumpNonZero Label2
        MoveMwrR 0 ReceiverResultReg TempReg
        AndCqR 16r3FFFFF TempReg
        Nop
        Nop
        Nop

Label 2:
        CmpRR ClassReg TempReg
        JumpNonZero Abort
        PushR ReceiverResultReg
```

Compare the expected and the actual class tag. If it is the same… continue the method



Why Sending the Expected Class Tag?

Monomorphic Calls

- So we can test if we are in the correct receiver. If not the abort trampoline is call.

```
MoveCqR 16r151 ReceiverResultReg
MovePatcheableC32R 1 ClassReg
Call send0ArgTrampoline
```

```
MoveCqR 16r151 ReceiverResultReg
MovePatcheableC32R 16r5588 ClassReg
Call 16rFF001122
```

```
Entry:
        AndCqRR 7 ReceiverResultReg TempReg
        JumpNonZero Label2
        MoveMwrR 0 ReceiverResultReg TempReg
        AndCqR 16r3FFFFF TempReg
        Nop
        Nop
        Nop

Label 2:
        CmpRR ClassReg TempReg
        JumpNonZero Abort
        PushR ReceiverResultReg
```

If not the same… jump to the call to the abort trampoline

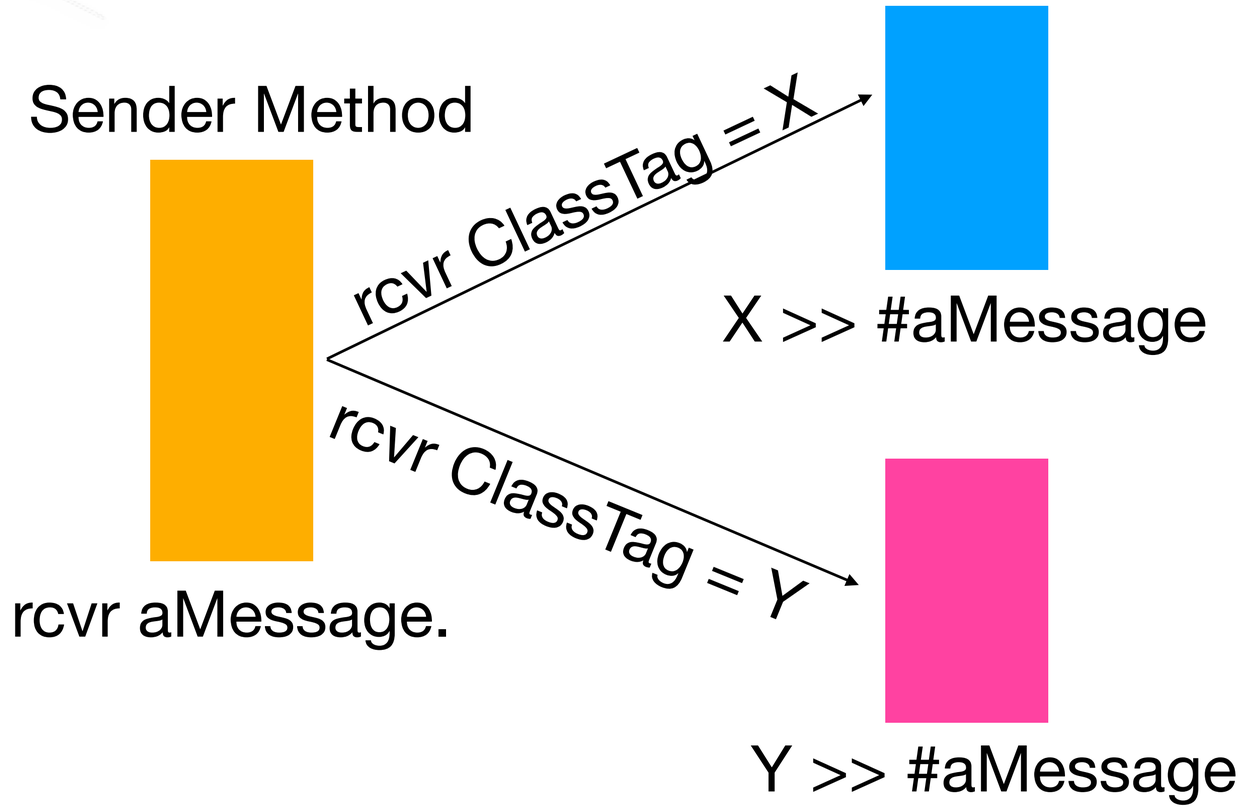
What the Abort Trampoline Do?

a.k.a Single-In-line Cache Miss

- Call #ceSICMiss: in the VM C Runtime with the receiver.
- We are expecting a single method to be activated here, but another appears. So we need to fix the patched sender.
- We have to extend the single-in-line cache with a polymorphic cache (or even a Megamorphic).

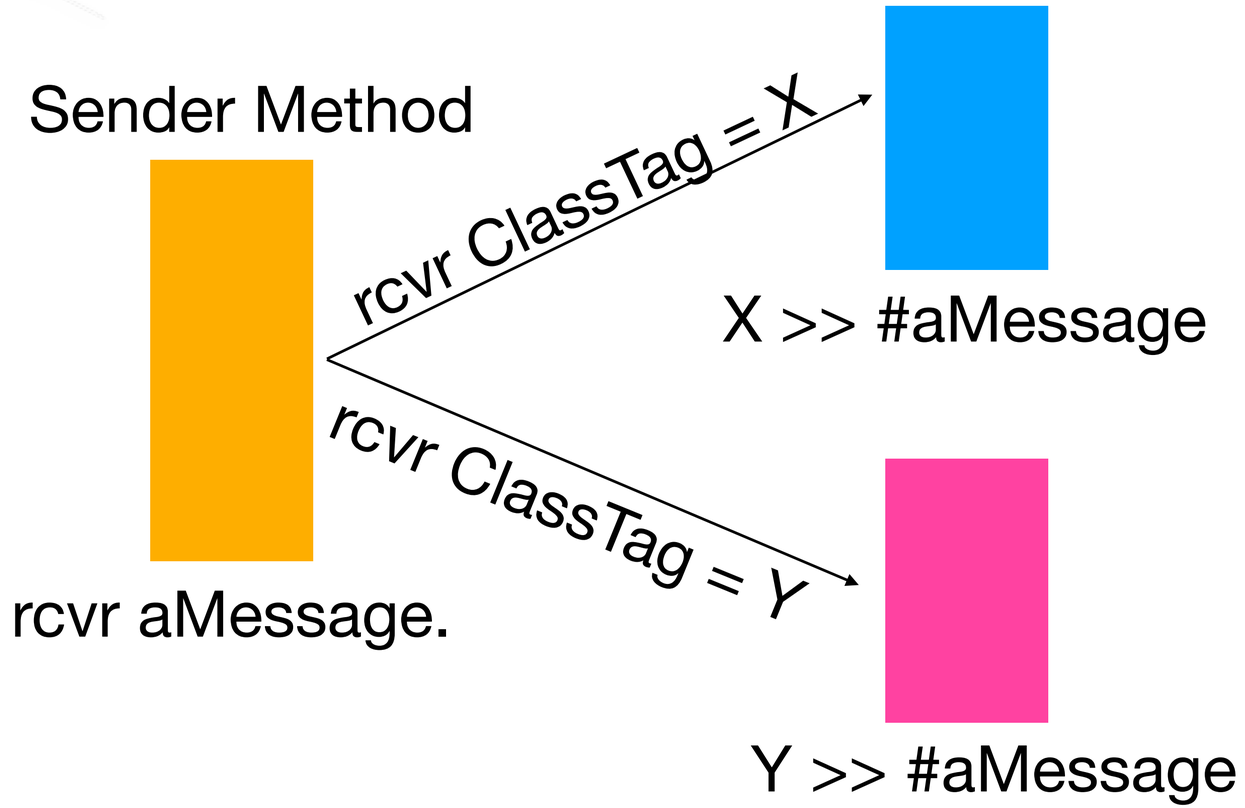
Polymorphic Calls

Welcome Polymorphism!



Polymorphic Calls

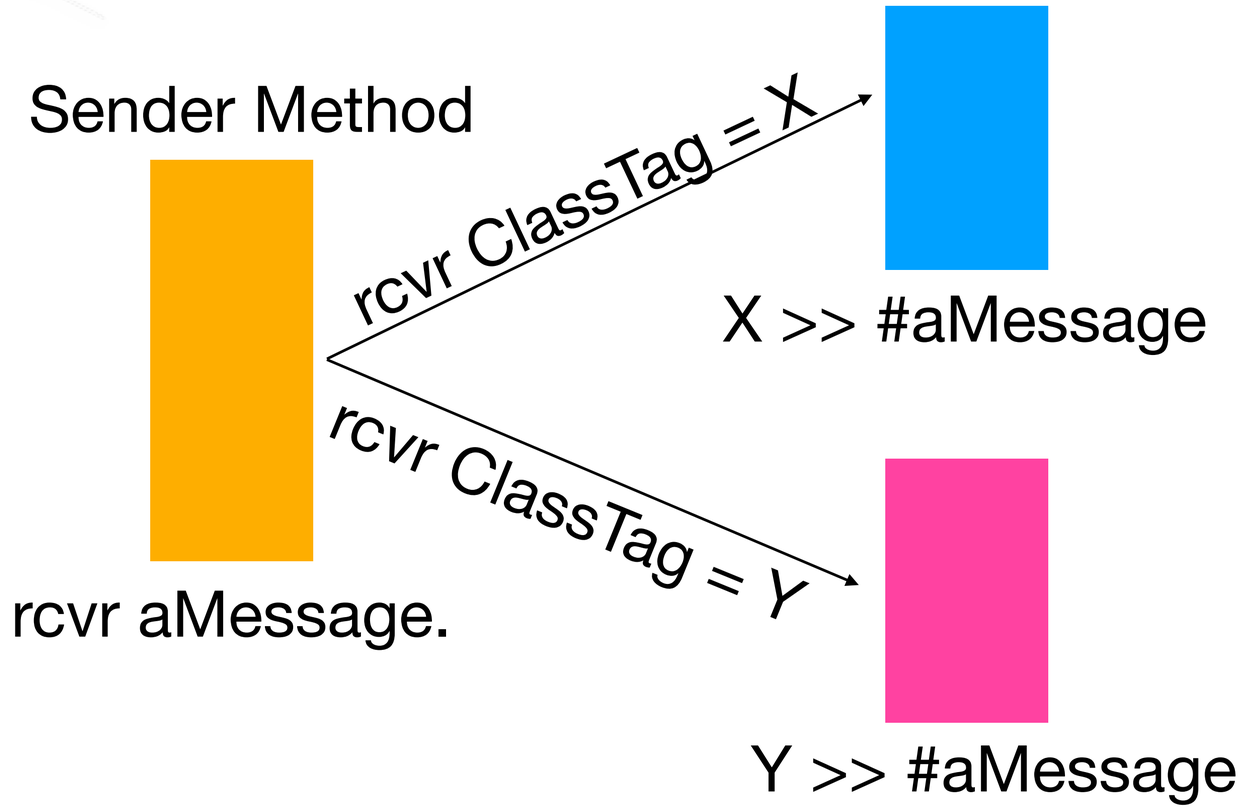
Welcome Polymorphism!



We want to put this decision logic somewhere!!!

Polymorphic Calls

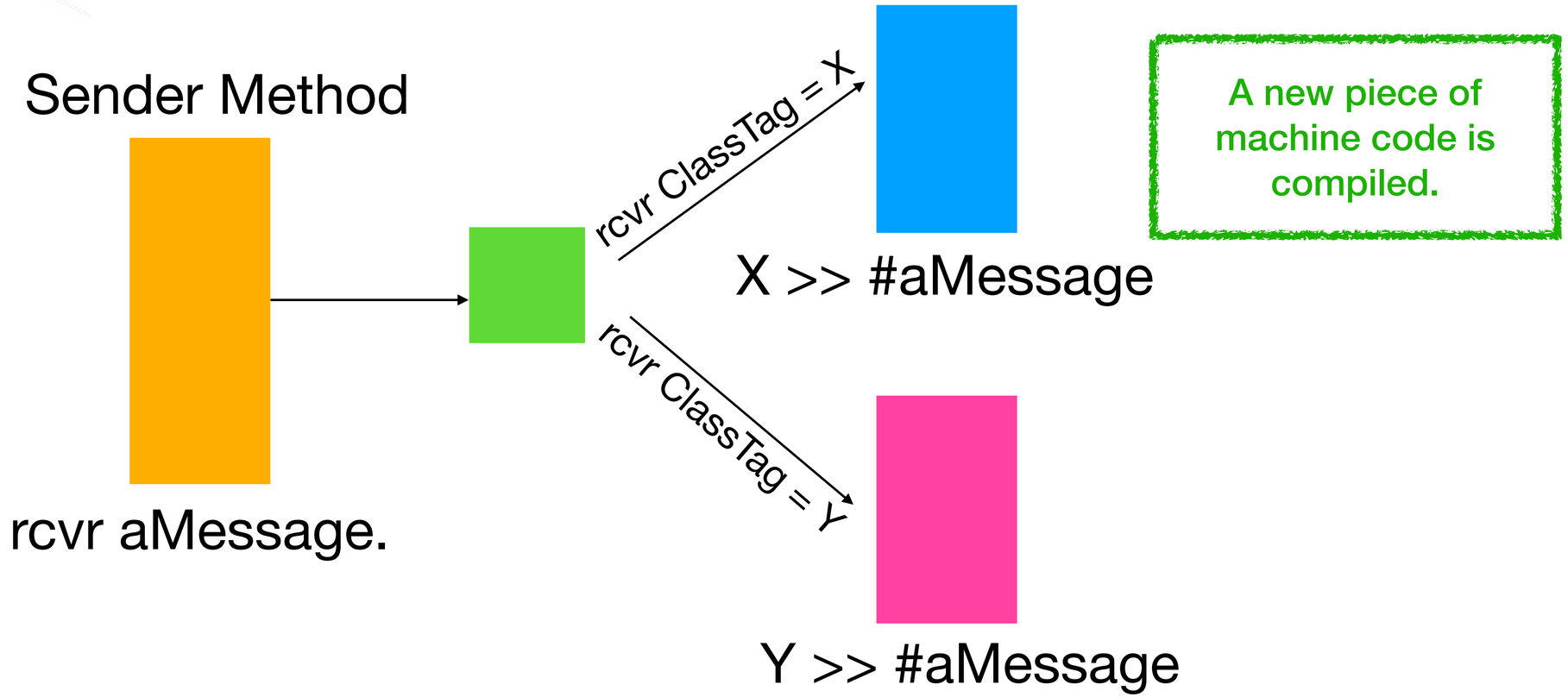
Welcome Polymorphism!



We cannot modify the existing method. So create something in the middle

Polymorphic Calls

Polymorphic-In-line-Cache, I choose you!



Polymorphic Calls

Polymorphic-In-line-Cache, I choose you!

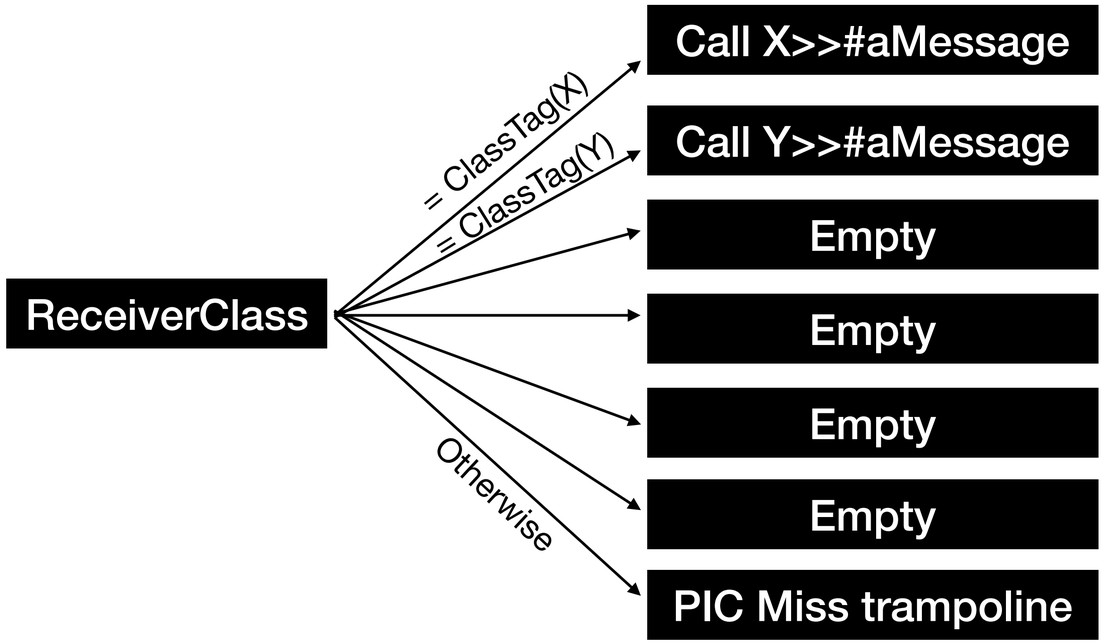
```
MoveCqR 16r151 ReceiverResultReg
MovePatcheableC32R 16r5588 ClassReg
Call 16rFF001122
```

```
MoveCqR 16r151 ReceiverResultReg
MovePatcheableC32R 16r5588 ClassReg
Call 16rFF998877
```

The sender is now patched to call the PIC.

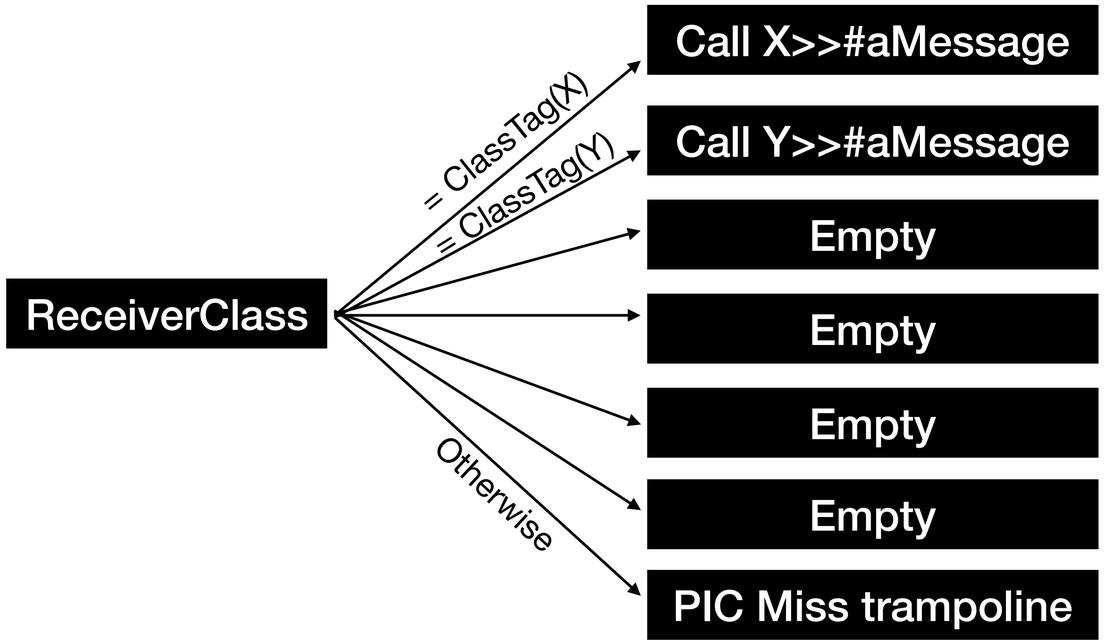
Polymorphic Calls

PICs a.k.a a Nice Switch Case



Polymorphic Calls

PICs a.k.a a Nice Switch Case



Starts with 2 Cases

A PIC is created by call site if needed.

6 Slots in total.

If one of the existing does not fit. PIC Miss Trampoline

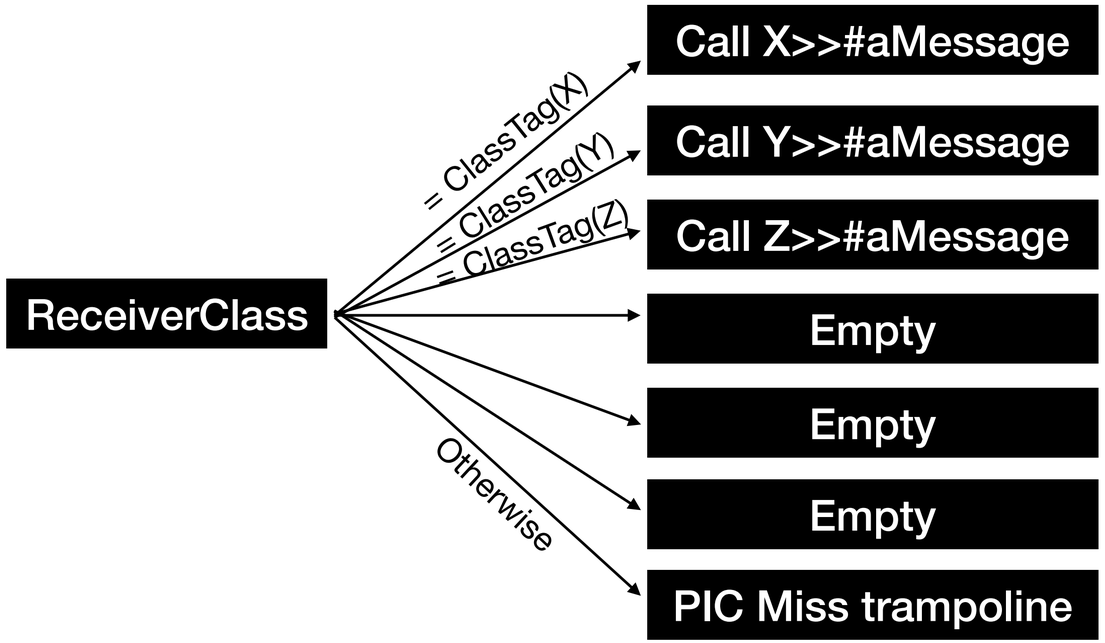
Polymorphic Calls

PIC Miss Trampoline

- This VM C runtime function is executed if a new method appear and it is not in the PIC.
- If there is a machine code method. Patch the PIC to include it.
- A new case is added to the PIC, and the new method is executed.

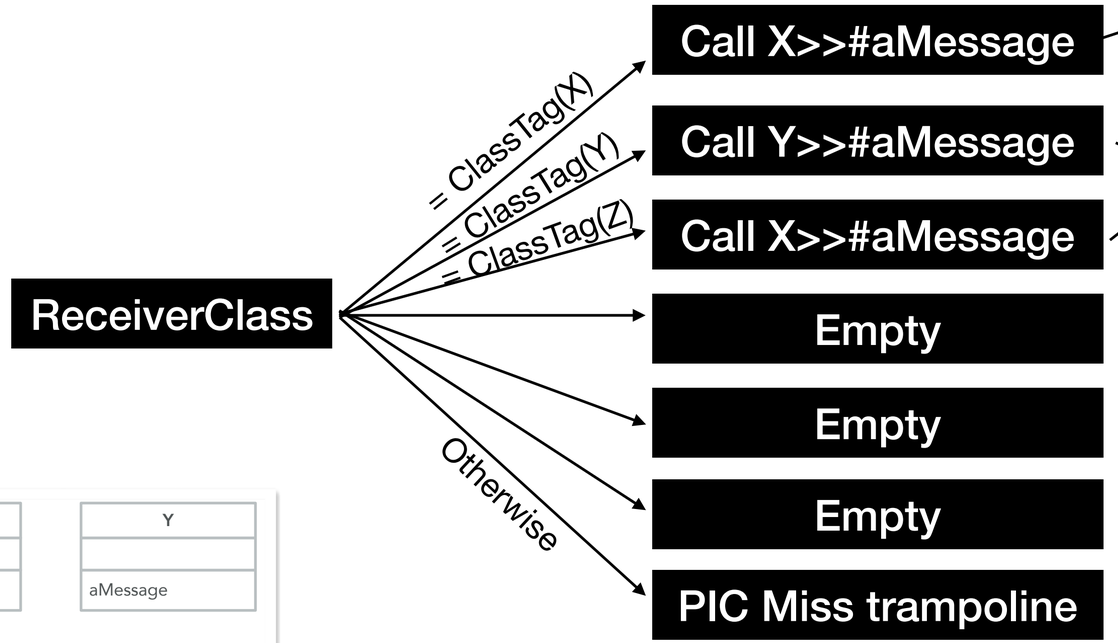
Polymorphic Calls

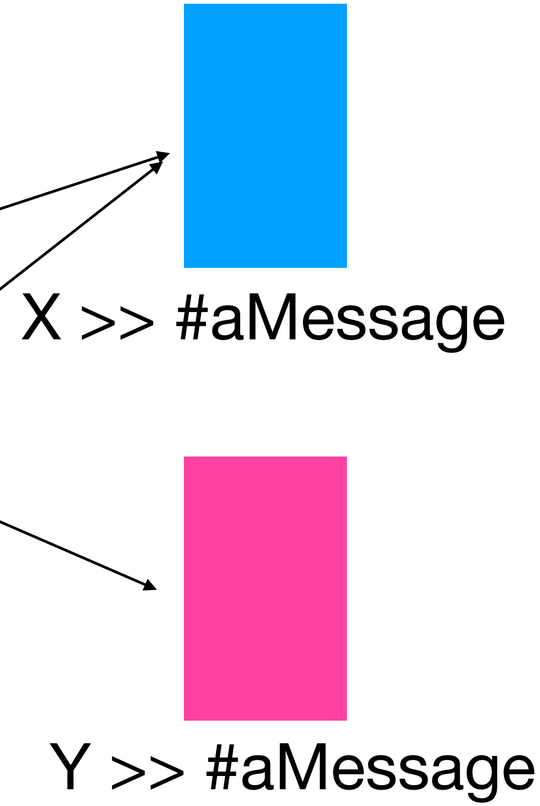
PICs a.k.a a Nice Switch Case

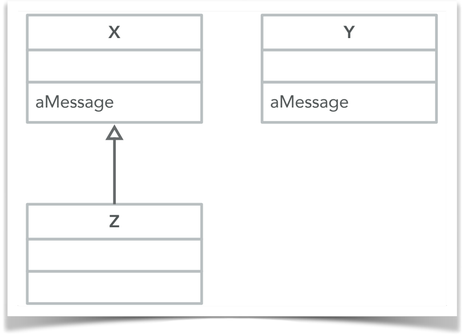


Polymorphic Calls

What with Inheritance



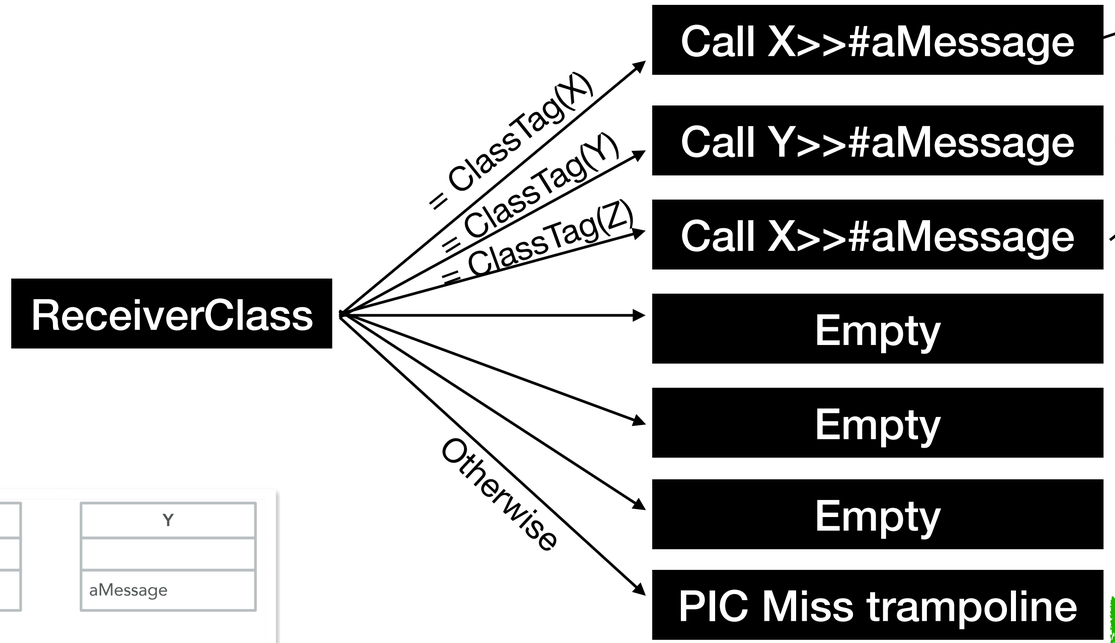


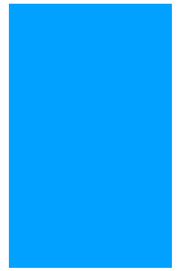


Our Classes

Polymorphic Calls

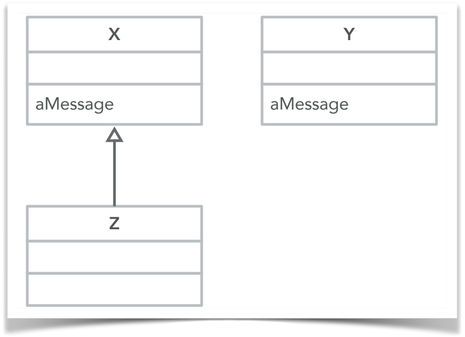
What with Inheritance





X >> #aMessage

Y >> #aMessage



Our Classes

If there is inheritance, and the message is sent. We have two entries: with different class tags, but same method

Megamorphic Calls

When a PIC is not enough

- If we have more a PIC that is full (all its slots are used by methods) and a new possible method is found for that class site.
- We have a Megamorphic call site.

Megamorphic Calls

What a Megamorphic-in-line-cache do?

- It is compiled in the method zone.
- There is one by selector (different call-sites uses the same one).
- It is compiled to do:
  - Search the pair <selector,classTag> in the method cache.
  - Search it 3 times.
  - If found, call it
  - If not found… call a trampoline to do the slow lookup and method activation.

What do we miss?

There is always more to learn…

- Super message sends
- MessageNotUnderstood handling
- Special selectors
- Young methods and early megamorphic promotion
- Optimizations / Implementation details
- ...

What do we miss?

There is always more to learn…

- Super message sends
- MessageNotUnderstood handling
- Special selectors
- Young methods and early megamorphic promotion
- Optimizations / Implementation details
- ...

Thanks!!!!

We are going to catch them all!!!