Structure of Machine Code Methods

An introduction

Pablo Tesone - 22/10/2020

Pharo

RMod

Inria
INVENTEURS DU MONDE NUMÉRIQUE

Previously….

We have seen how we generate the code…

```
<76> push 1
<20> push 17
<B0> send #+
<7C> returnTop
```

bytecode

```
move r1 #1
move r2 #17
checkSmallInt r1
checkSmallInt r2
add r3 r1 r2
checkSmallInt r3
move r1 r3
ret
```

CogRTL - IR

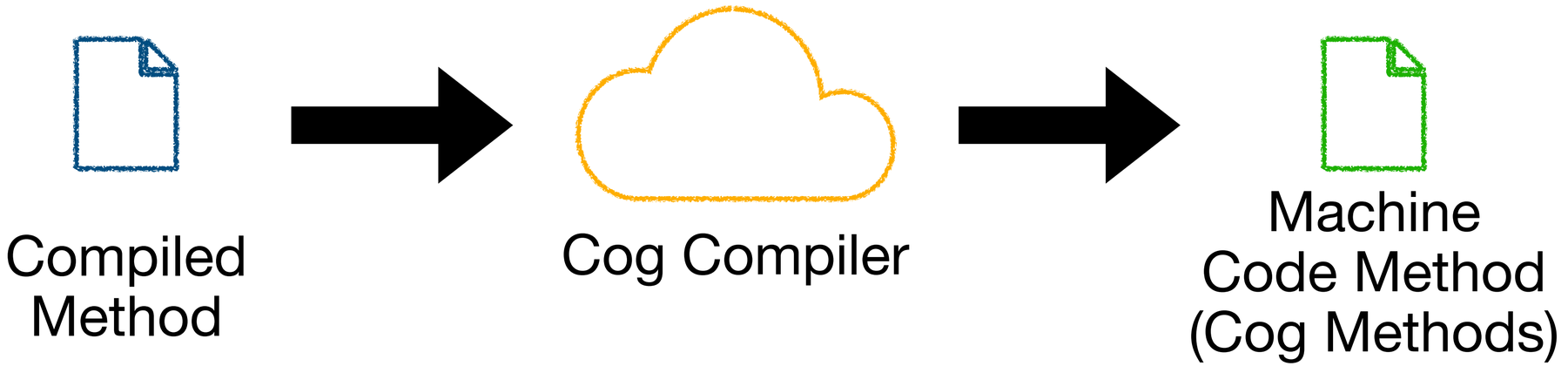
```
move X5 #1
move X3 #17
test X5 #0x1
je 0x40
test X3 #0x1
je 0x32
add X0 X3 X5
test X0 #0x1
je 0x24
move X5 X0
ret
```

Machine Code

IR Generation
“gen*”

Machine Code Generation
“Concretize”

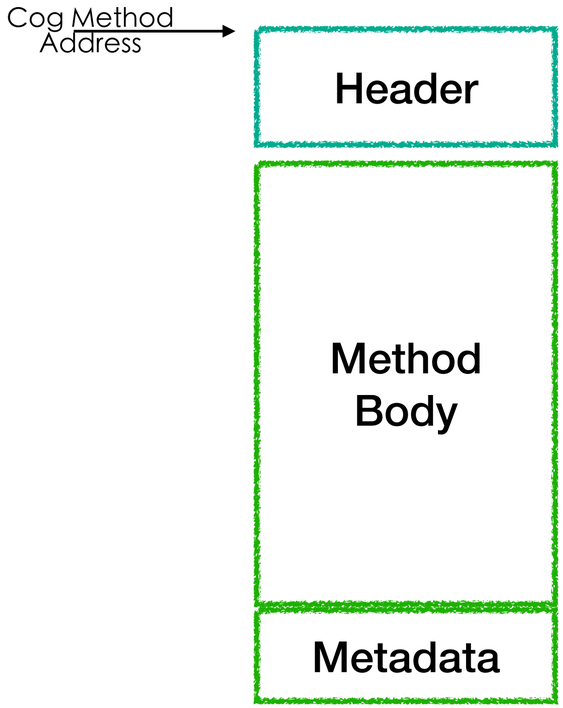
Machine Code



Method Zone (Cog Zone)

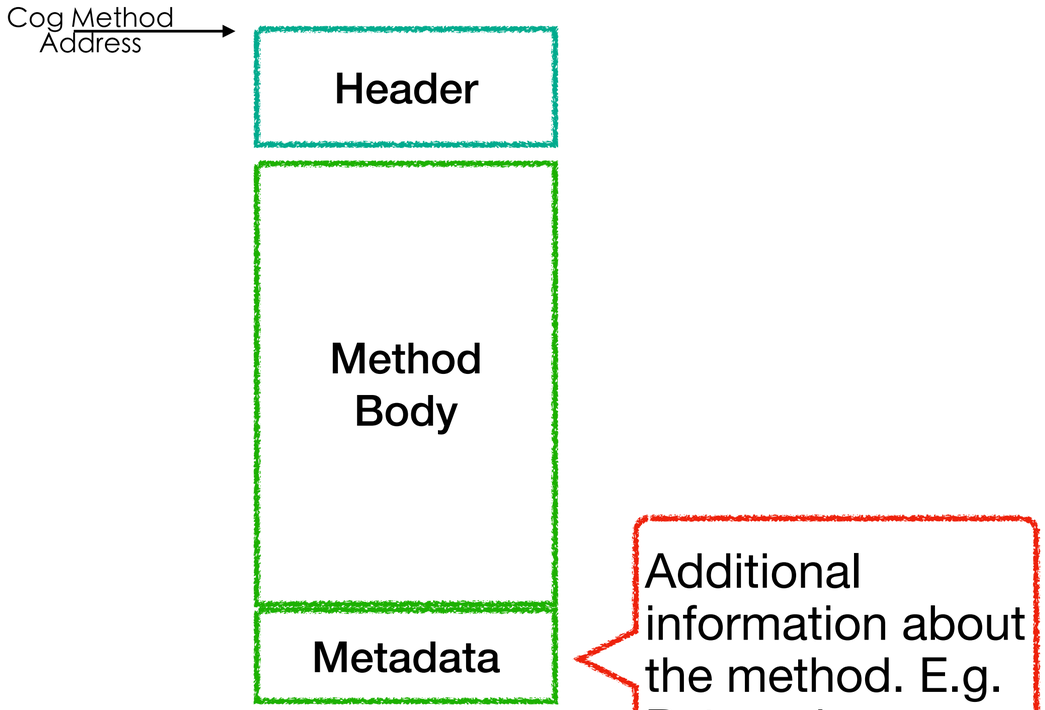
- Cog Methods are created in a particular region of memory.
- ~ 1.4Mb in a Pharo VM
- Used to store all machine code methods
- Garbage collected and reused

Cog Method Structure



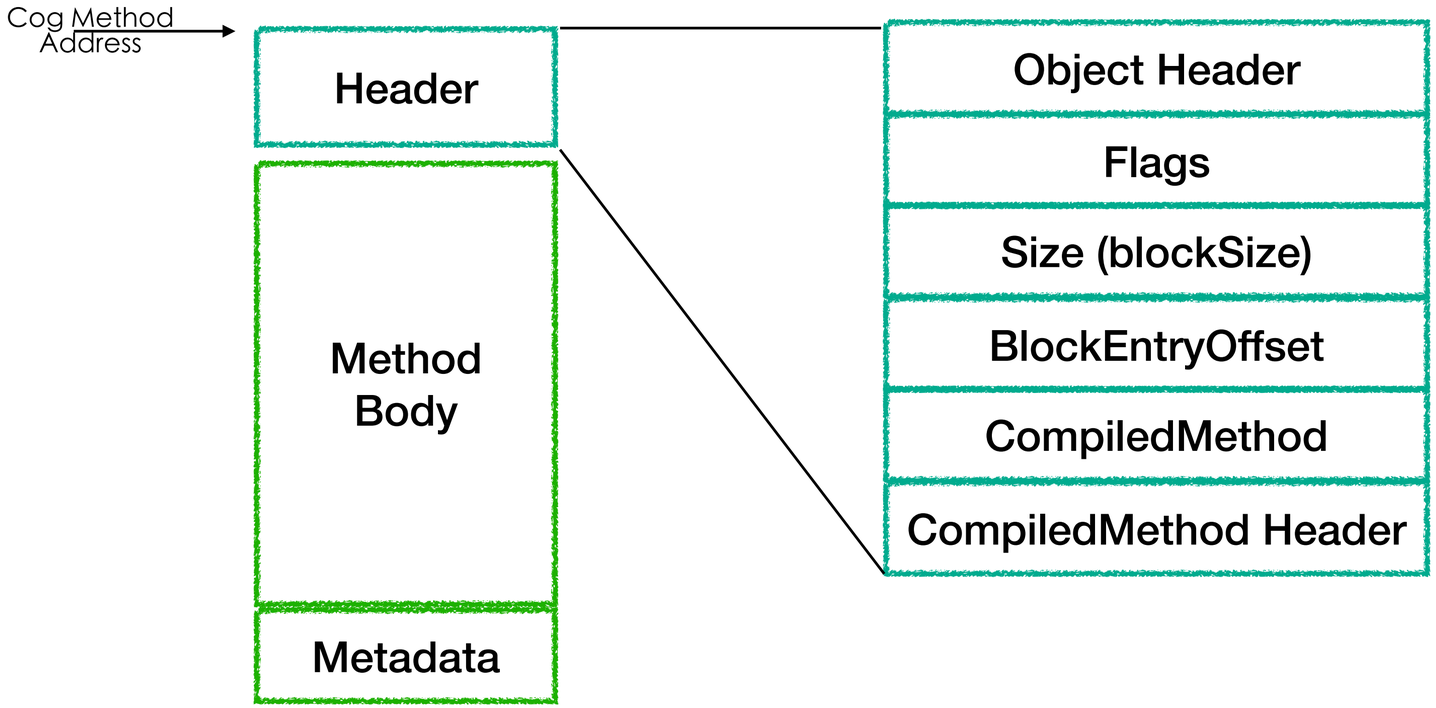
Cog Method Structure

Metadata



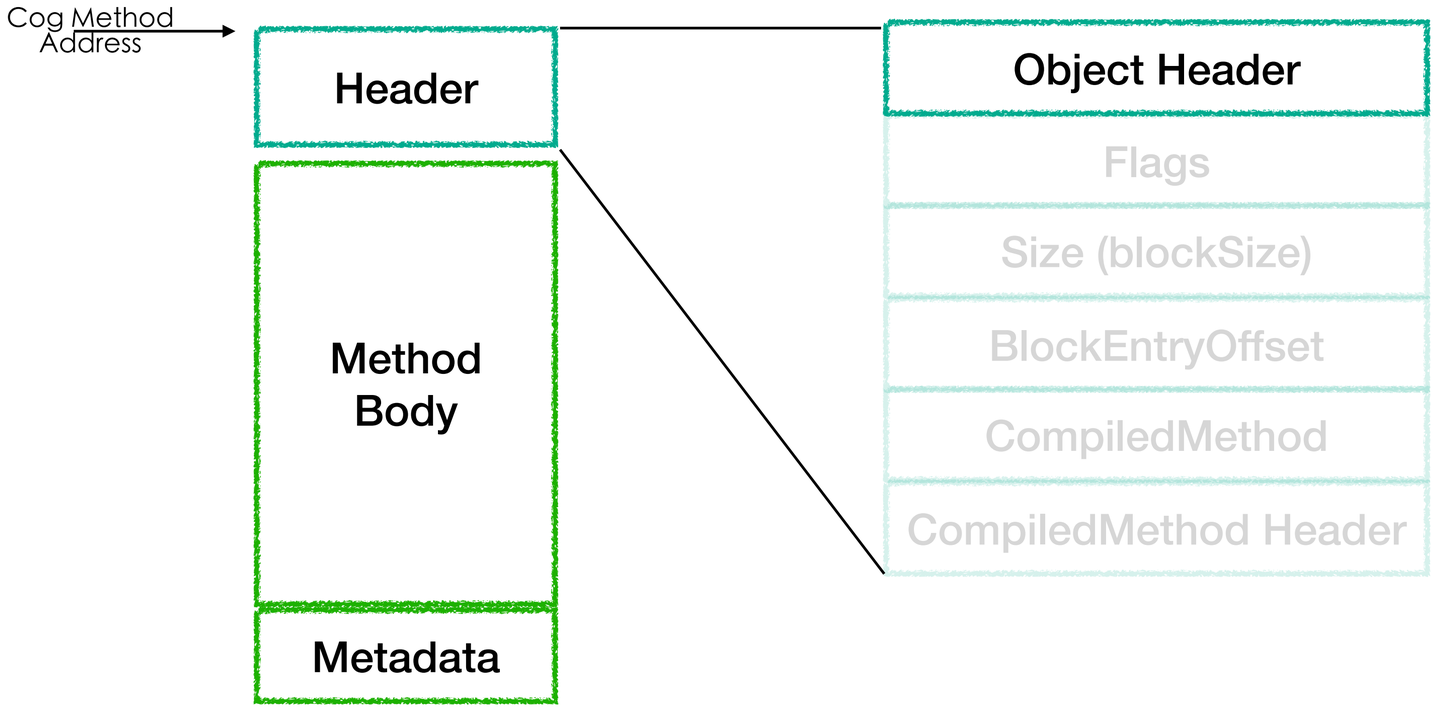
Cog Method Structure

Header



Cog Method Structure

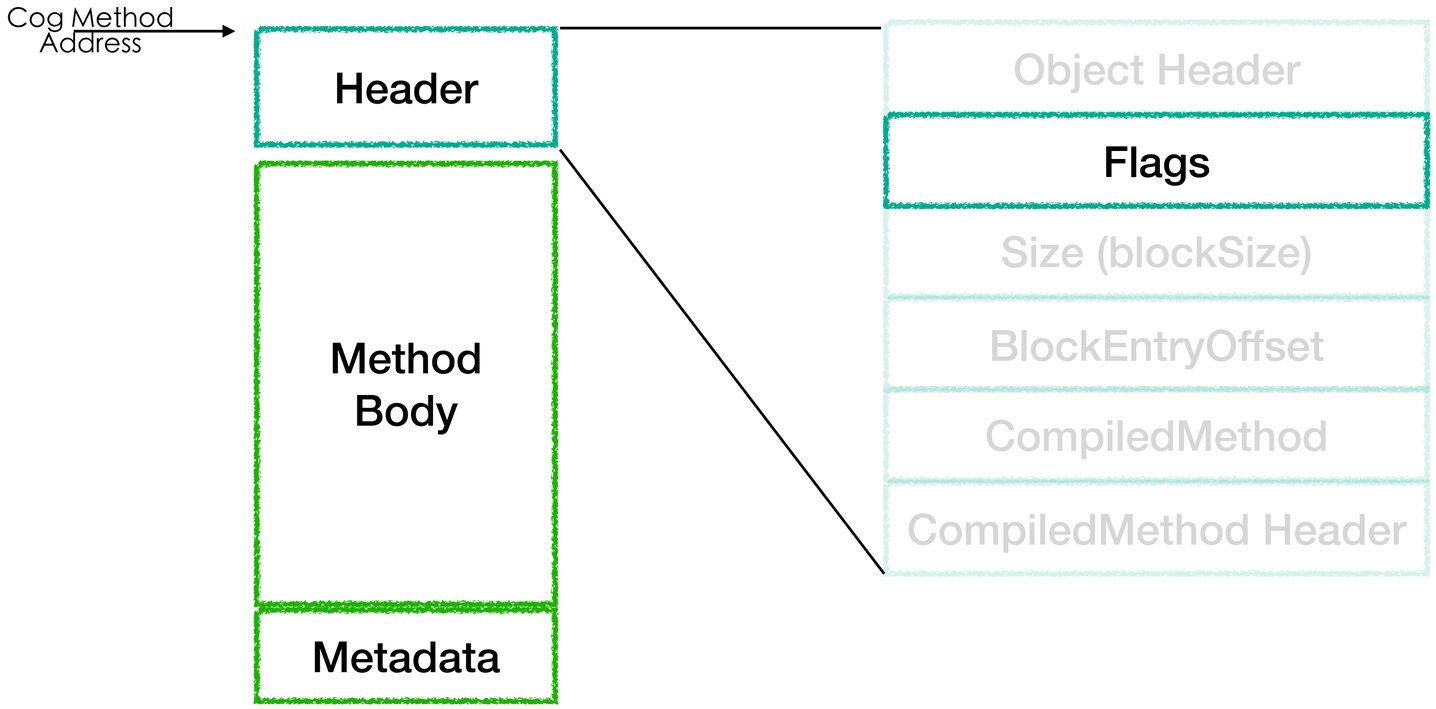
Header



Fake Object Header to be “similar” to objects

Cog Method Structure

Header



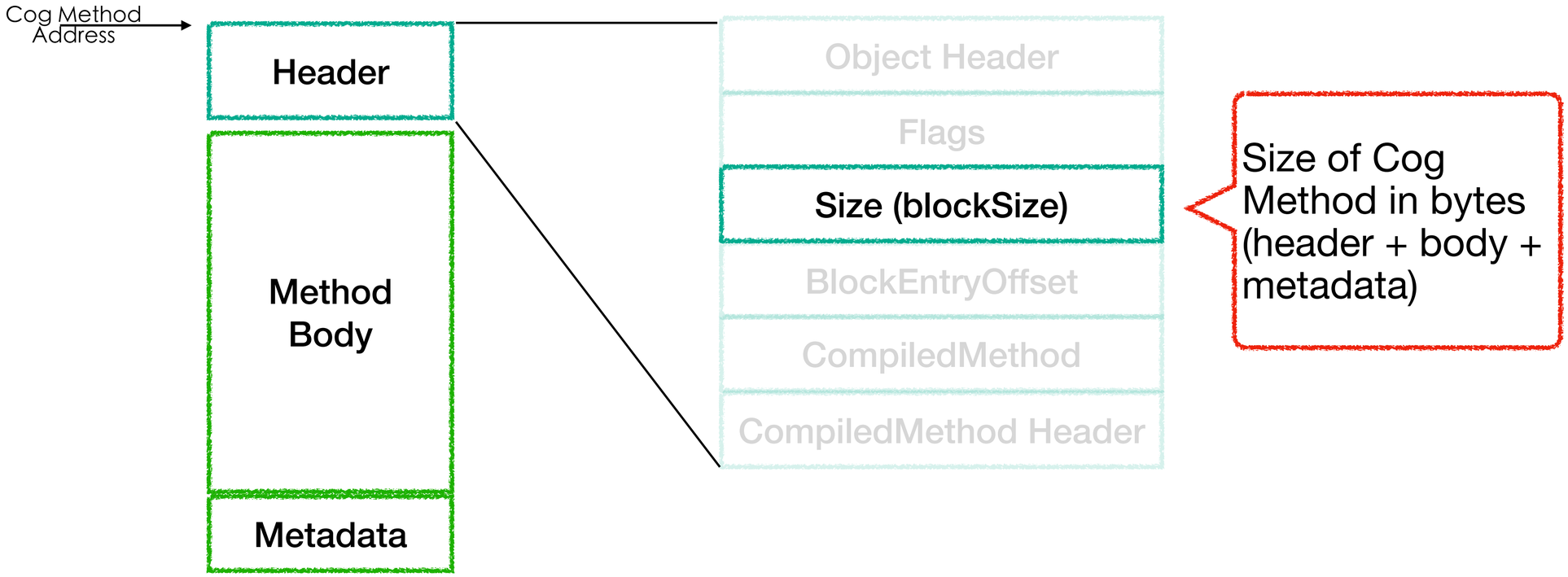
Cog Method Structure

Header - Flags

- Between other fields:
  - number of arguments
  - type (method, block, Polymorphic-Inline-Cache)
  - usage count

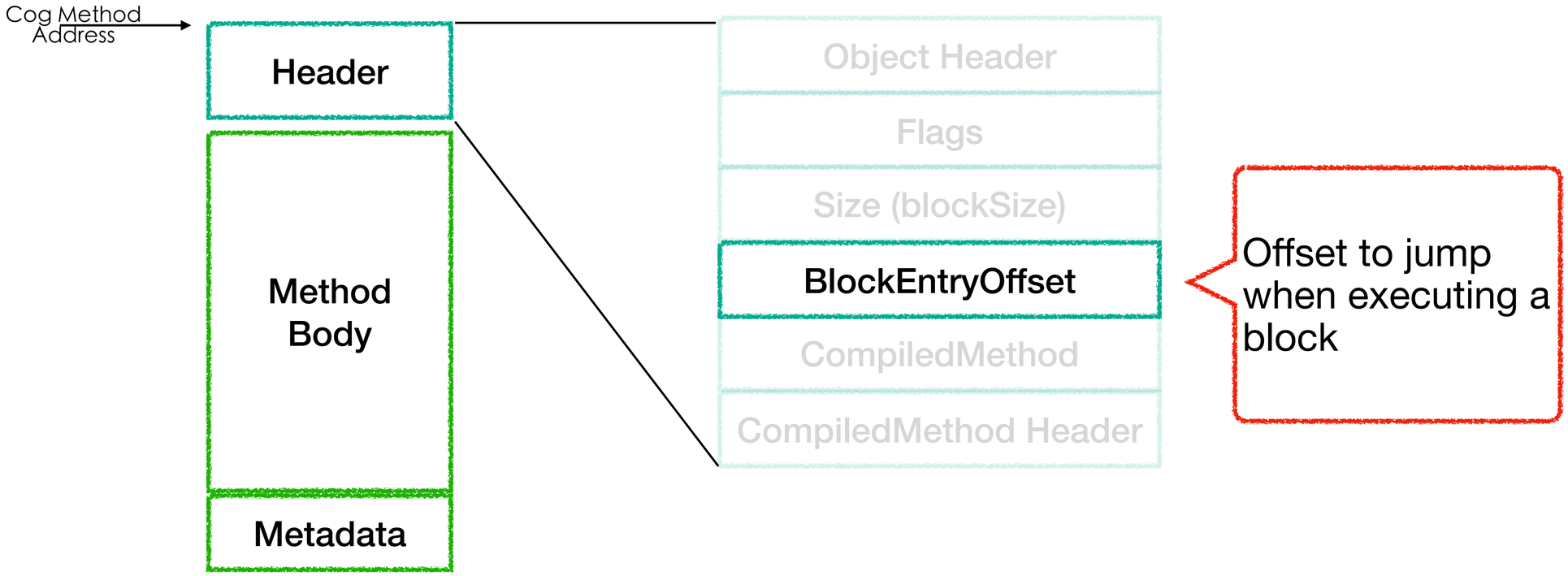
Cog Method Structure

Header



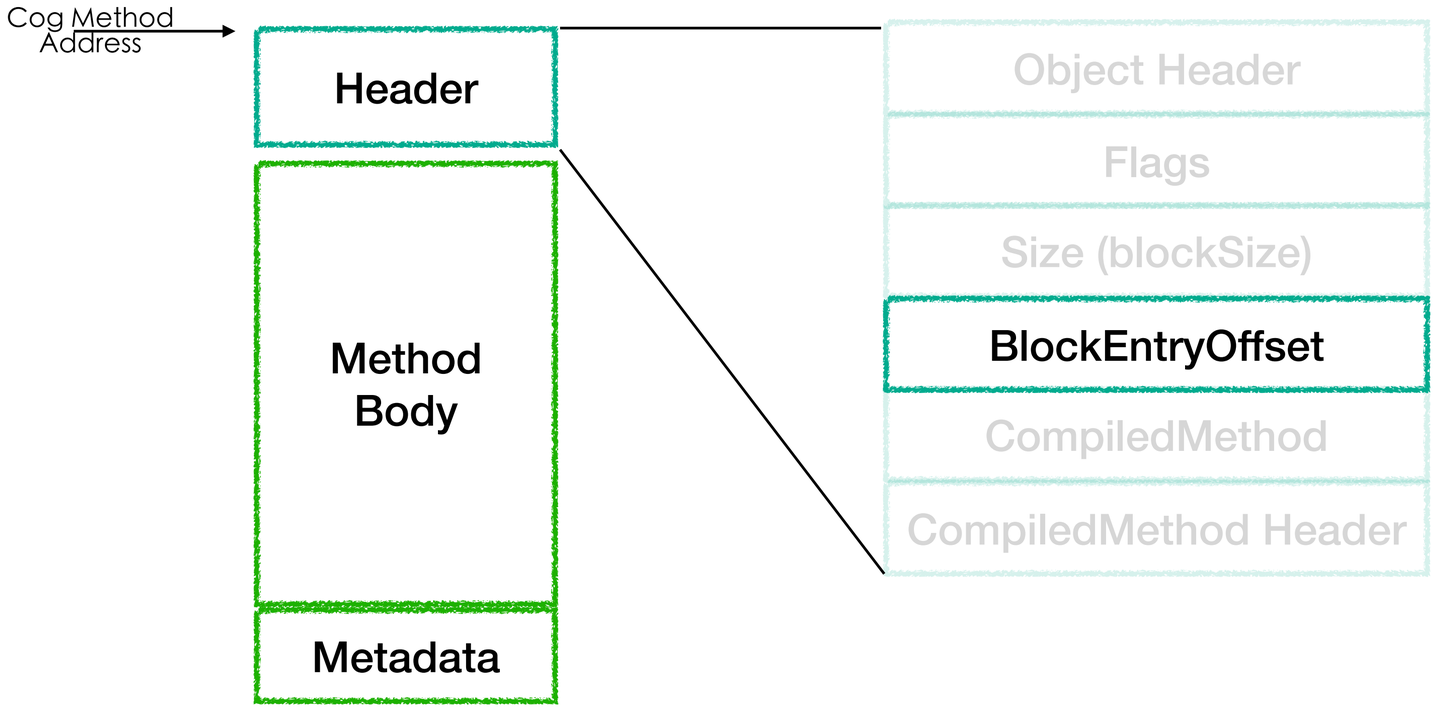
Cog Method Structure

Header



Cog Method Structure

Header

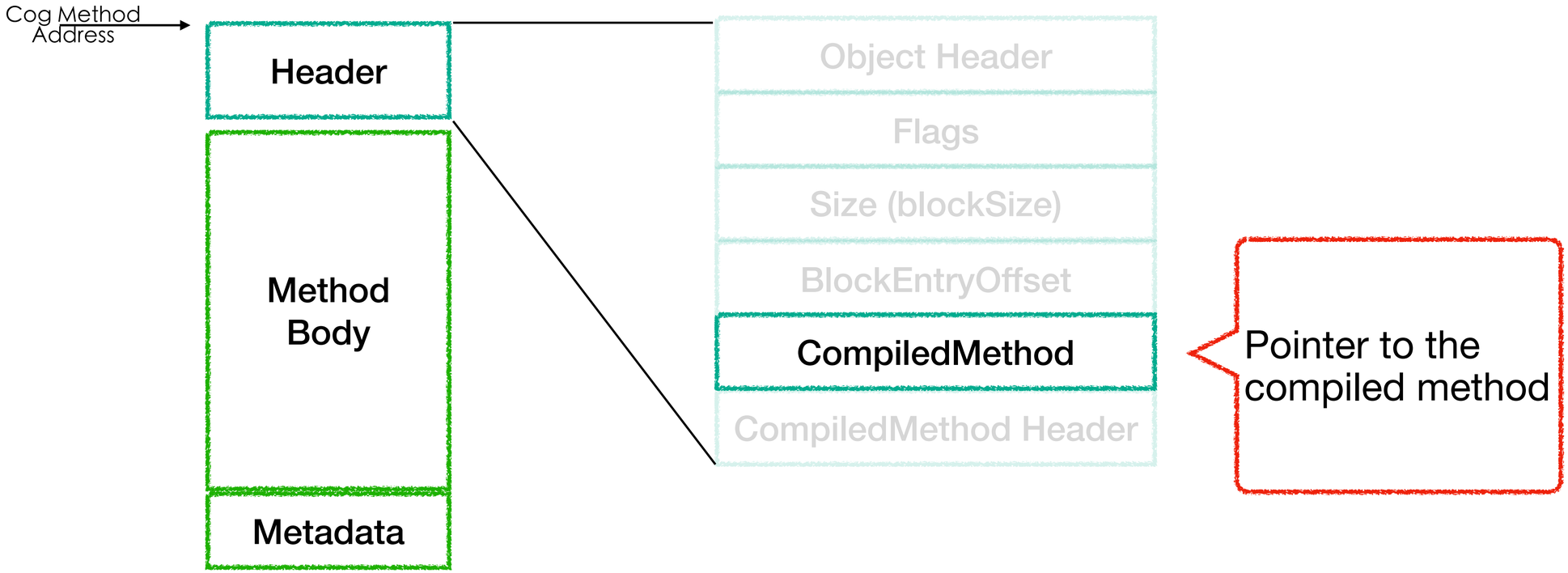


Offset to jump when executing a block

Outside the Scope of this presentation… wait for the block presentation :)

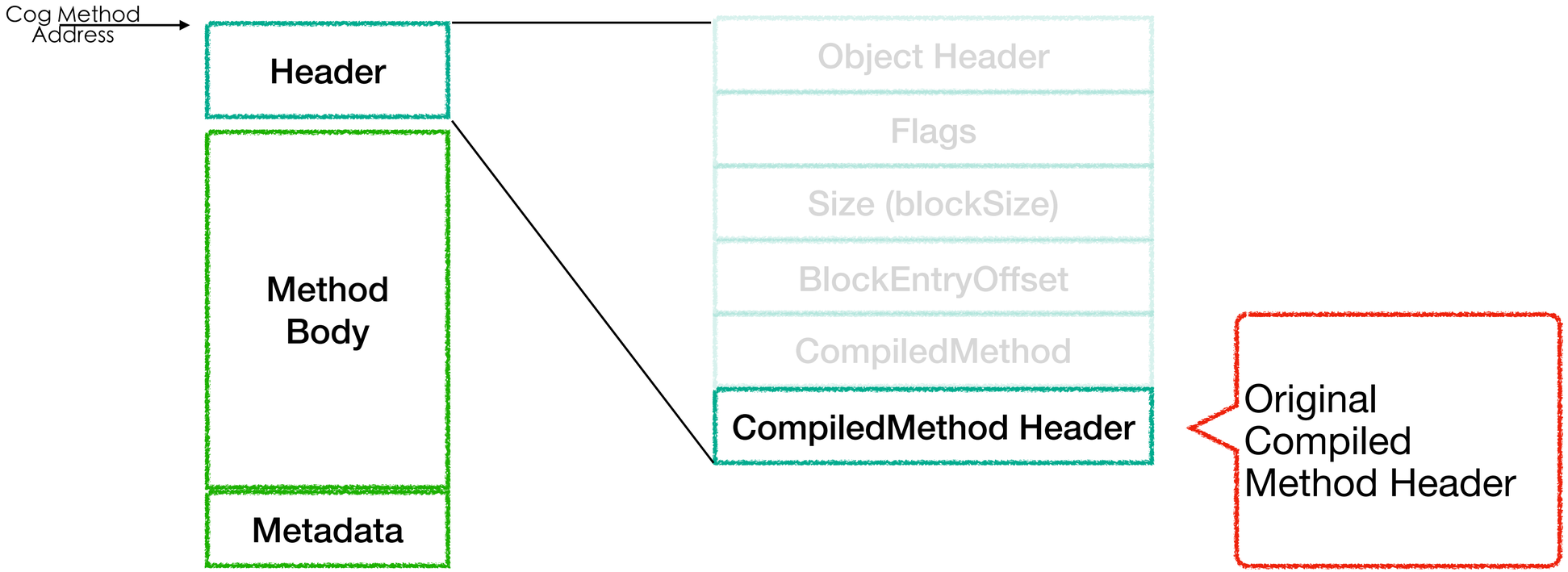
Cog Method Structure

Header



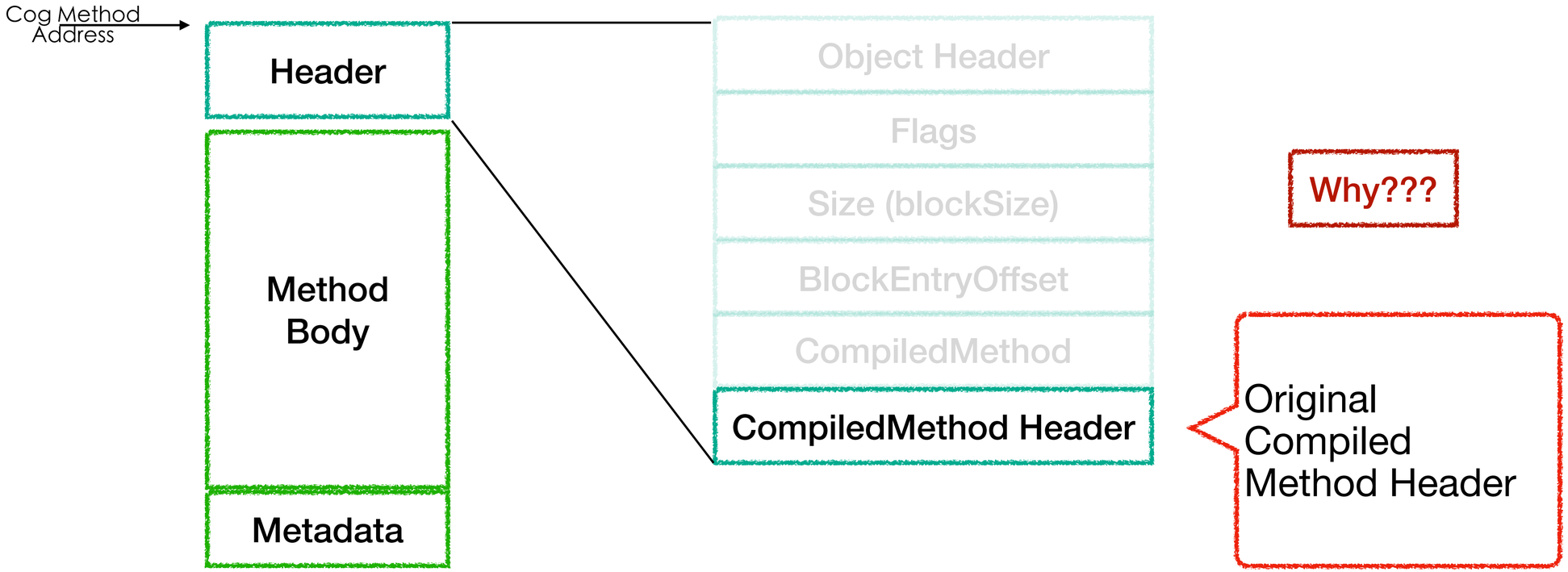
Cog Method Structure

Header



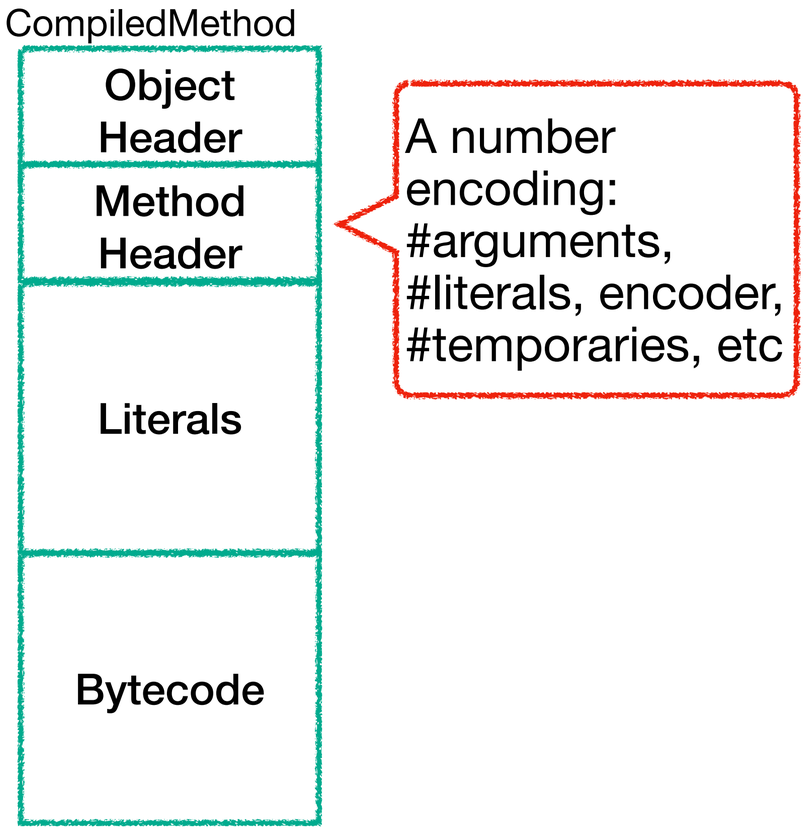
Cog Method Structure

Header



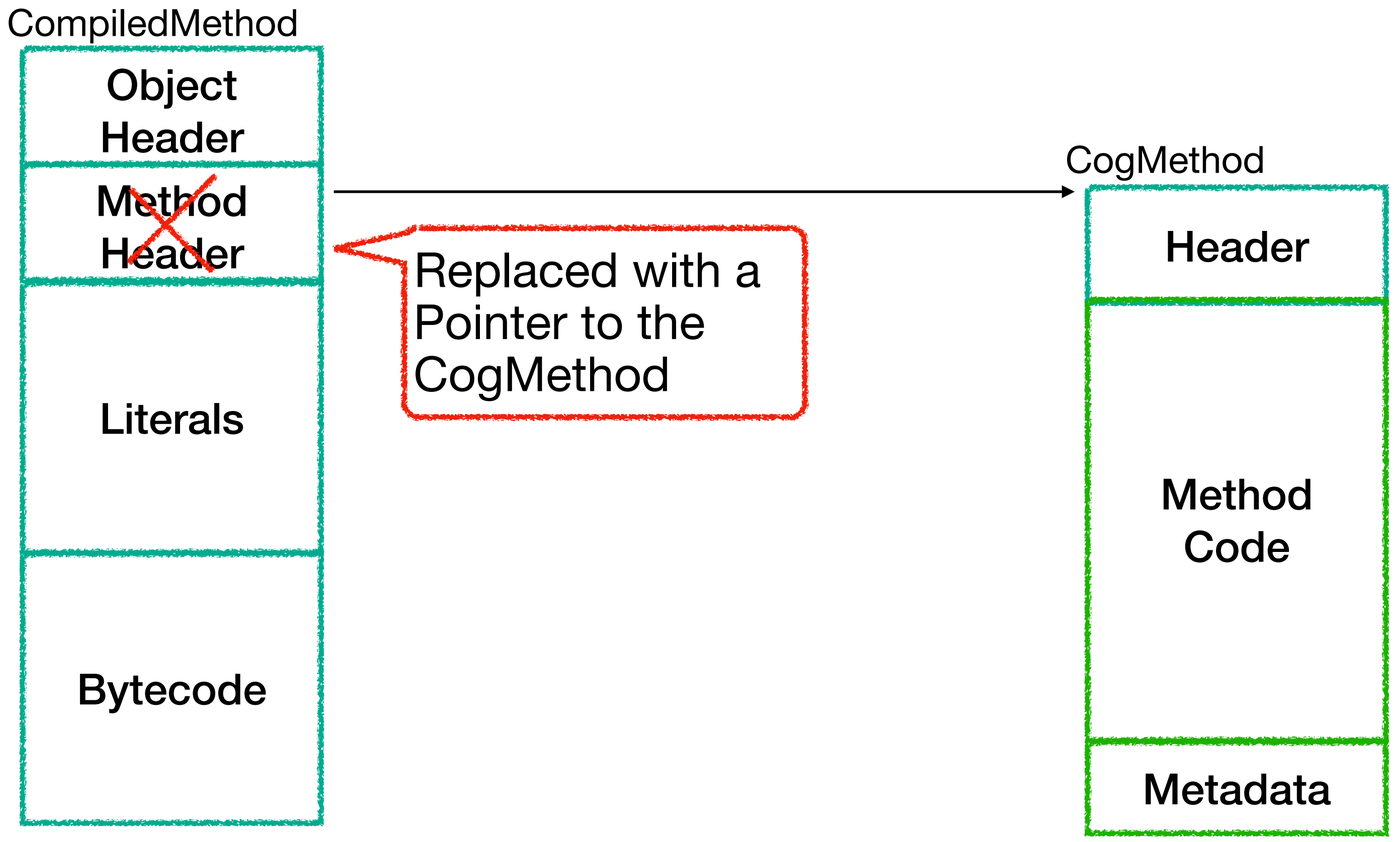
Compiled Method and CogMethods

Without corresponding CogMethod



Compiled Method and CogMethods

With Corresponding CogMethod



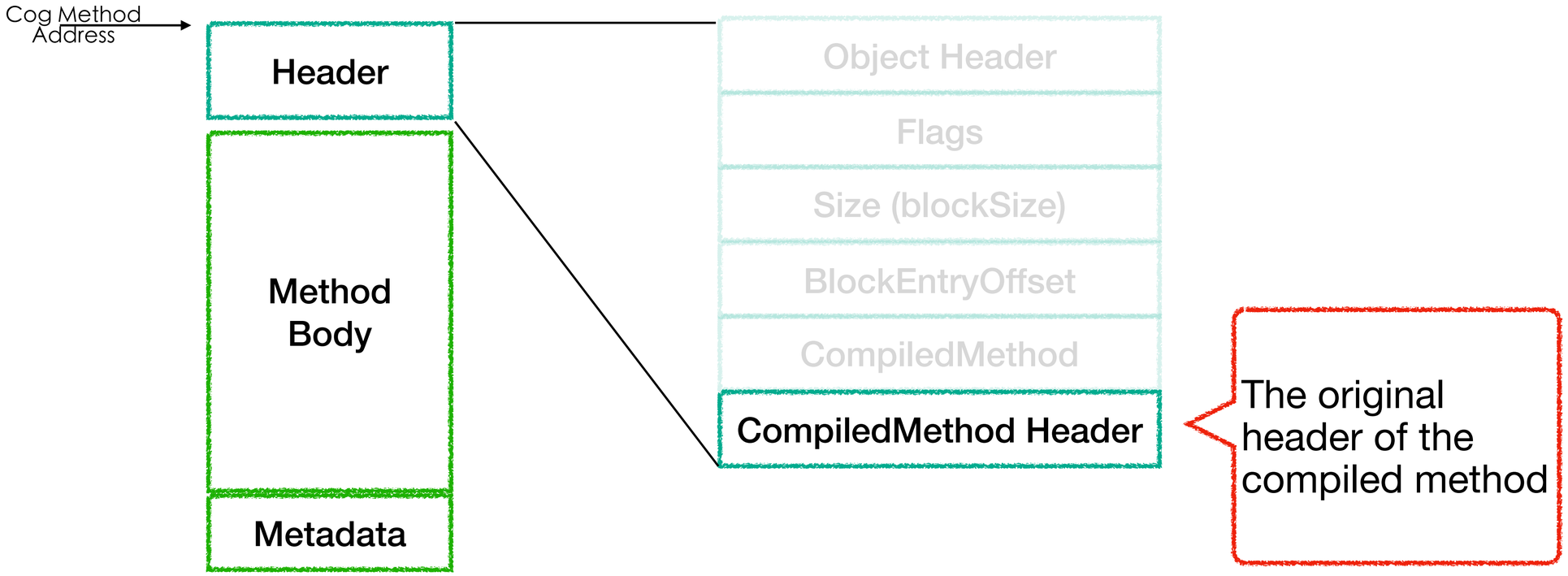
Compiled Method and CogMethods

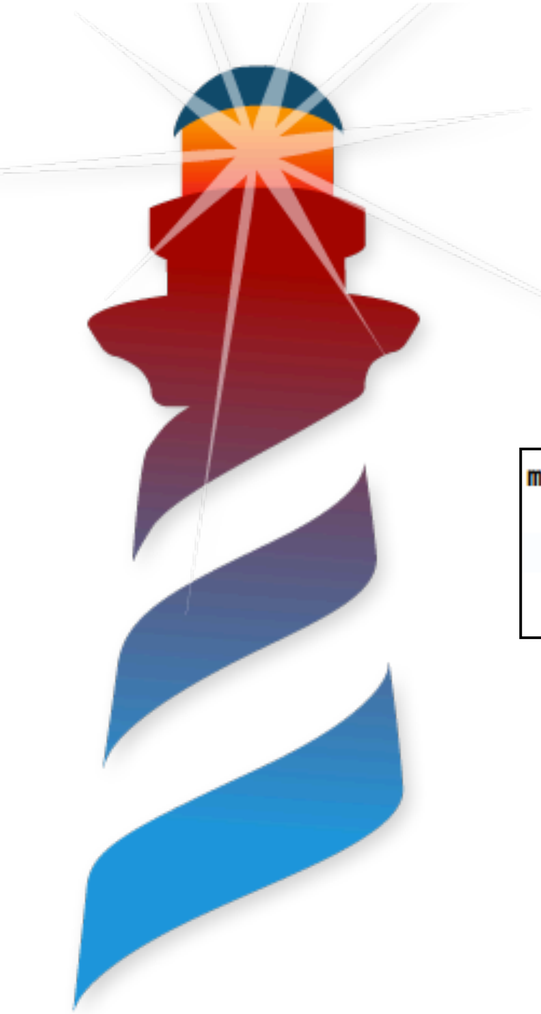
With Corresponding CogMethod

- So…
  - If a Compiled Method has a SmallInteger, it is the header and the method is not compiled in machine code.
  - If the Compiled Method has a Pointer as header, it has a corresponding compiled method in Machine code.
  - And the header??

Compiled Method and CogMethods

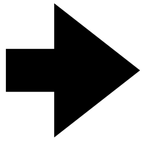
With Corresponding CogMethod



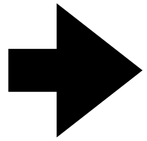


Compiling a Simple Method

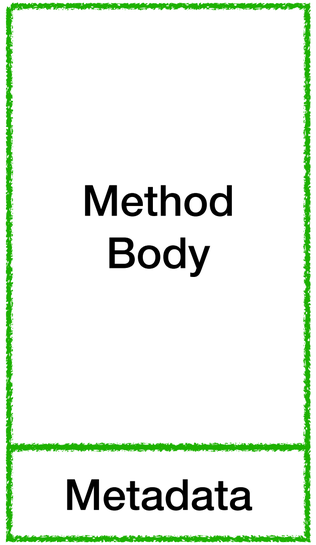
```
methodToCompile1
    ^ 42
```



```
33 <20> pushConstant: 42
34 <5C> returnTop
```



Object Header	16r8000000A000035
#Args / type	0 / 2 (Method)
Size	120
Block Entry	0
Method Object	16r10218C0
Method Header	16r19



Compiling a Simple Method

Body Content

```
33 <20> pushConstant: 42
34 <5C> returnTop
```

Method Entry →

```
        MoveCqR 0 ReceiverResultReg
Label 0:
        PushR LinkReg
        Call 16777688/10001D8
        AlignmentNops 8
Label 1:
        AndCqRR 7 ReceiverResultReg TempReg
        JumpNonZero Label 2
        MoveMwrR 0 ReceiverResultReg TempReg
        AndCqR 4194303/3FFFFF TempReg
        Nop
        Nop
        Nop
Label 2:
        CmpRR ClassReg TempReg
        JumpNonZero Label 0
```

```
        PushCq 337/151
        PopR ReceiverResultReg
        RetN 8
```

Compiling a Simple Method

Body Content

```
33 <20> pushConstant: 42
34 <5C> returnTop
```

Method Entry →

```
        MoveCqR 0 ReceiverResultReg
Label 0:
        PushR LinkReg
        Call 16777688/10001D8
        AlignmentNops 8
Label 1:
        AndCqRR 7 ReceiverResultReg TempReg
        JumpNonZero Label 2
        MoveMwrR 0 ReceiverResultReg TempReg
        AndCqR 4194303/3FFFFF TempReg
        Nop
        Nop
        Nop
Label 2:
        CmpRR ClassReg TempReg
        JumpNonZero Label 0
```

Type Guard & Abort routine call

```
        PushCq 337/151
        PopR ReceiverResultReg
        RetN 8
```

Compiling a Simple Method

Body Content

```
33 <20> pushConstant: 42
34 <5C> returnTop
```

Method Entry →

```
        MoveCqR 0 ReceiverResultReg
Label 0:
        PushR LinkReg
        Call 16777688/10001D8
        AlignmentNops 8
Label 1:
        AndCqRR 7 ReceiverResultReg TempReg
        JumpNonZero Label 2
        MoveMwrR 0 ReceiverResultReg TempReg
        AndCqR 4194303/3FFFFF TempReg
        Nop
        Nop
        Nop
Label 2:
        CmpRR ClassReg TempReg
        JumpNonZero Label 0
```

```
        PushCq 337/151
        PopR ReceiverResultReg
        RetN 8
```

Method Body

Checking the Type on Method Activation

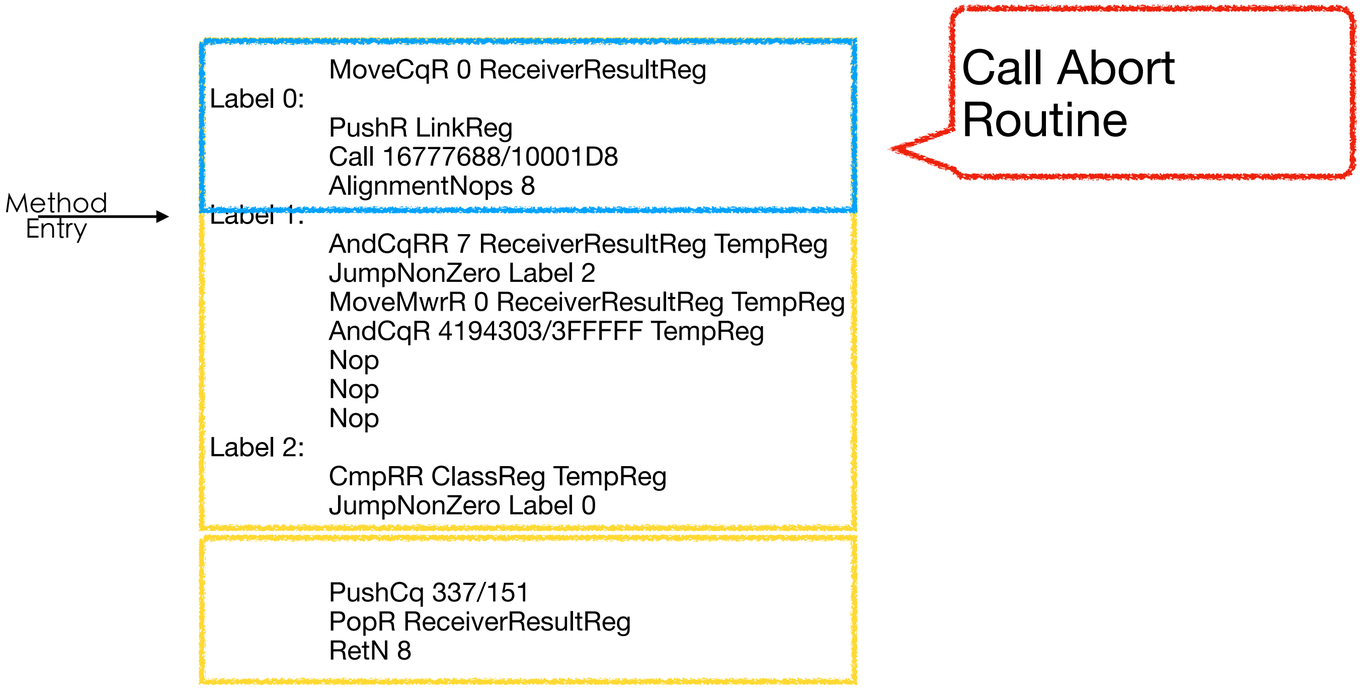
- When a Cog method is activated:
  - We need to check if the type is correct!
  - We need to compare the expected class with the class of the receiver.
    - In a simple scenario:
      - We have the class tag in a register (ClassReg)
      - We have the receiver in a Register (ReceiverResultReg)
  - If the method is not the expected, we need to abort and find the correct method (or MNU)

Checking the Type on Method Activation

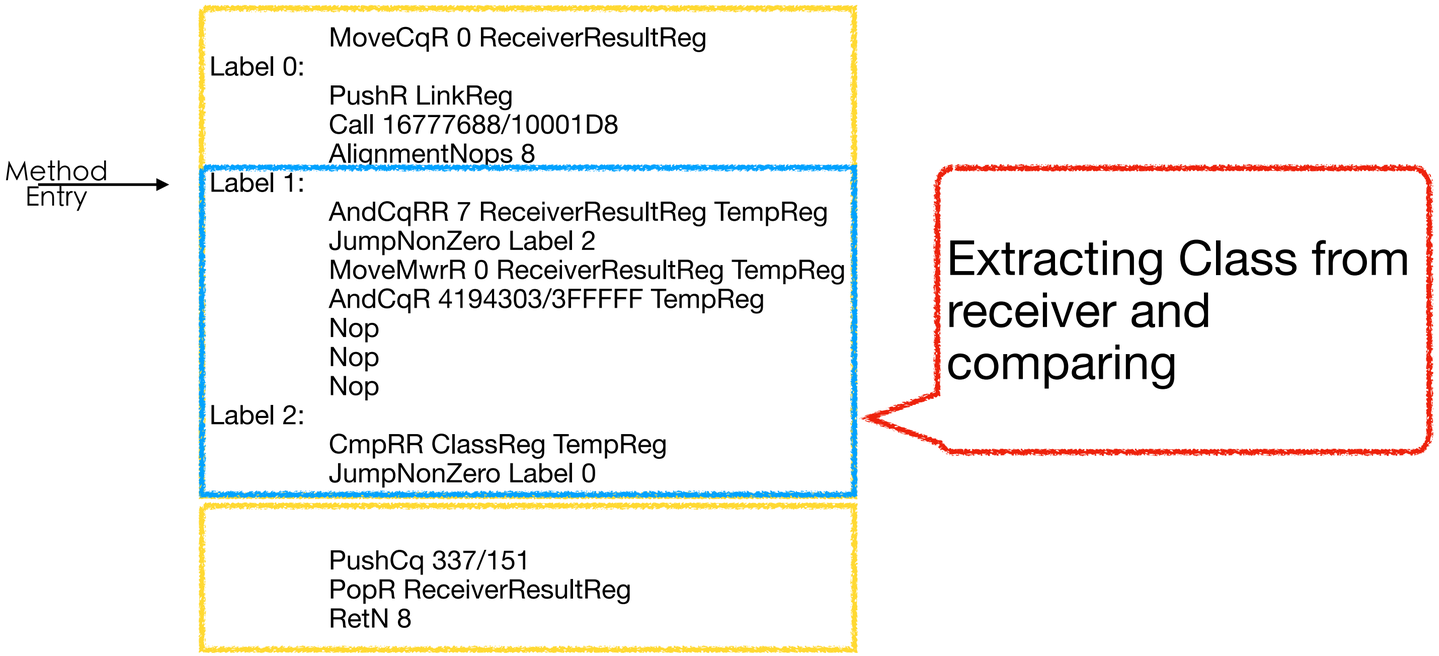
- When a Cog method is activated:
  - We need to check if the type is correct!
  - We need to compare the expected class with the class of the receiver.
    - In a simple scenario:
      - We have the class tag in a register (ClassReg)
      - We have the receiver in a Register (ReceiverResultReg)
  - If the method is not the expected, we need to abort and find the correct method (or MNU)

More when talk about Message Sends & Polymorphic Inline Caches :)

Checking the Type on Method Activation



Checking the Type on Method Activation



Frame Building

- The creation of the frame is done in the activated method.
  - We have two types of methods
    - Frameless
    - Frameful
  - Depends if they are Leaves (if they don’t send messages)

Frame Building

A Frameless Example

```
33 <20> pushConstant: 42
34 <5C> returnTop
```

Method Entry →

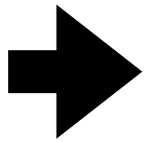
```
MoveCqR 0 ReceiverResultReg
Label 0:
PushR LinkReg
Call 16777688/10001D8
AlignmentNops 8
Label 1:
AndCqRR 7 ReceiverResultReg TempReg
JumpNonZero Label 2
MoveMwrR 0 ReceiverResultReg TempReg
AndCqR 4194303/3FFFFF TempReg
Nop
Nop
Nop
Label 2:
CmpRR ClassReg TempReg
JumpNonZero Label 0

PushCq 337/151
PopR ReceiverResultReg
RetN 8
```

Method Body without creating frame

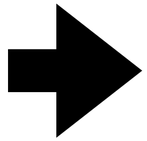
Frame Building

A Framefull Example

```
methodToCompile1

    ^ 42 bitInvert
```



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



Method Entry →

```
Label 0:
        MoveCqR 0 ReceiverResultReg
        PushR LinkReg
        Call 16777688/10001D8
        AlignmentNops 8

Label 1:
        AndCqRR 7 ReceiverResultReg TempReg
        JumpNonZero Label 2
        MoveMwrR 0 ReceiverResultReg TempReg
        AndCqR 4194303/3FFFFF TempReg
        Nop
        Nop
        Nop

Label 2:
        CmpRR ClassReg TempReg
        JumpNonZero Label 0

        PushR LinkReg
        PushR FPReg
        MoveRR SPReg FPReg
        PushCw CogMethodAddress
        MoveCqR (#bitInvert oop) SendNumArgsReg
        PushR SendNumArgsReg
        PushR ReceiverResultReg

        PushCq 337/151
        MoveMwrR 0 SPReg ReceiverResultReg
        MovePatcheableC32R 1 ClassReg
        Call SendMessageRoutine

        PushR ReceiverResultReg
        PopR ReceiverResultReg
        MoveRR FPReg SPReg@
        PopR FPReg
        PopR LinkReg
        RetN 8
```

Frame Building

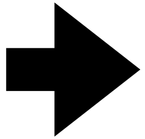
A Framefull Example

Method Entry →

```
Label 0:
        MoveCqR 0 ReceiverResultReg
        PushR LinkReg
        Call 16777688/10001D8
        AlignmentNops 8

Label 1:
        AndCqRR 7 ReceiverResultReg TempReg
        JumpNonZero Label 2
        MoveMwrR 0 ReceiverResultReg TempReg
        AndCqR 4194303/3FFFFF TempReg
        Nop
        Nop
        Nop
```

Type Guard & Abort routine call

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



```
Label 2:
        CmpRR ClassReg TempReg
        JumpNonZero Label 0

        PushR LinkReg
        PushR FPReg
        MoveRR SPReg FPReg
        PushCw CogMethodAddress
        MoveCqR (#bitInvert oop) SendNumArgsReg
        PushR SendNumArgsReg
        PushR ReceiverResultReg

        PushCq 337/151
        MoveMwrR 0 SPReg ReceiverResultReg
        MovePatcheableC32R 1 ClassReg
        Call SendMessageRoutine

        PushR ReceiverResultReg
        PopR ReceiverResultReg
        MoveRR FPReg SPReg@
        PopR FPReg
        PopR LinkReg
        RetN 8
```

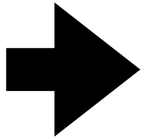
Frame Building

A Framefull Example

Method Entry →

```
Label 0:
        MoveCqR 0 ReceiverResultReg
        PushR LinkReg
        Call 16777688/10001D8
        AlignmentNops 8
Label 1:
        AndCqRR 7 ReceiverResultReg TempReg
        JumpNonZero Label 2
        MoveMwrR 0 ReceiverResultReg TempReg
        AndCqR 4194303/3FFFFF TempReg
        Nop
        Nop
        Nop
```

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



```
Label 2:
        CmpRR ClassReg TempReg
        JumpNonZero Label 0

        PushR LinkReg
        PushR FPReg
        MoveRR SPReg FPReg
        PushCw CogMethodAddress
        MoveCqR (#bitInvert oop) SendNumArgsReg
        PushR SendNumArgsReg
        PushR ReceiverResultReg

        PushCq 337/151
        MoveMwrR 0 SPReg ReceiverResultReg
        MovePatcheableC32R 1 ClassReg
        Call SendMessageRoutine

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

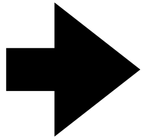
Frame Creation

Frame Building

A Framefull Example

Method Entry →

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



```
Label 0:
        MoveCqR 0 ReceiverResultReg
        PushR LinkReg
        Call 16777688/10001D8
        AlignmentNops 8
Label 1:
        AndCqRR 7 ReceiverResultReg TempReg
        JumpNonZero Label 2
        MoveMwrR 0 ReceiverResultReg TempReg
        AndCqR 4194303/3FFFFF TempReg
        Nop
        Nop
        Nop
Label 2:
        CmpRR ClassReg TempReg
        JumpNonZero Label 0

        PushR LinkReg
        PushR FPReg
        MoveRR SPReg FPReg
        PushCw CogMethodAddress
        MoveCqR CogMethodFlags SendNumArgsReg
        PushR SendNumArgsReg
        PushR ReceiverResultReg

        PushCq 337/151
        MoveMwrR 0 SPReg ReceiverResultReg
        MovePatcheableC32R 1 ClassReg
        Call SendMessageRoutine

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

Message Send

Frame Building

A Framefull Example

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

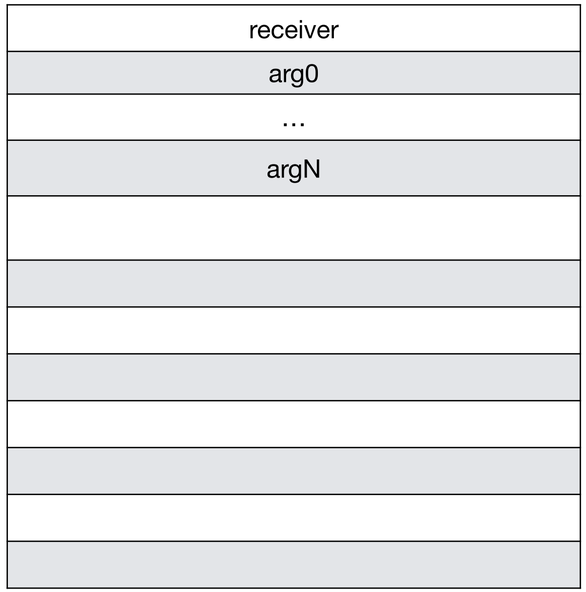
Method Entry →

```
Label 0:
        MoveCqR 0 ReceiverResultReg
        PushR LinkReg
        Call 16777688/10001D8
        AlignmentNops 8
Label 1:
        AndCqRR 7 ReceiverResultReg TempReg
        JumpNonZero Label 2
        MoveMwrR 0 ReceiverResultReg TempReg
        AndCqR 4194303/3FFFFF TempReg
        Nop
        Nop
        Nop
Label 2:
        CmpRR ClassReg TempReg
        JumpNonZero Label 0

        PushR LinkReg
        PushR FPReg
        MoveRR SPReg FPReg
        PushCw CogMethodAddress
        MoveCqR NilOop SendNumArgsReg
        PushR SendNumArgsReg
        PushR ReceiverResultReg

        PushCq 337/151
        MoveMwrR 0 SPReg ReceiverResultReg
        MovePatcheableC32R 1 ClassReg
        Call SendMessageRoutine

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

Frame Pop & Return

Frame Building

Machine Code Frame Structure



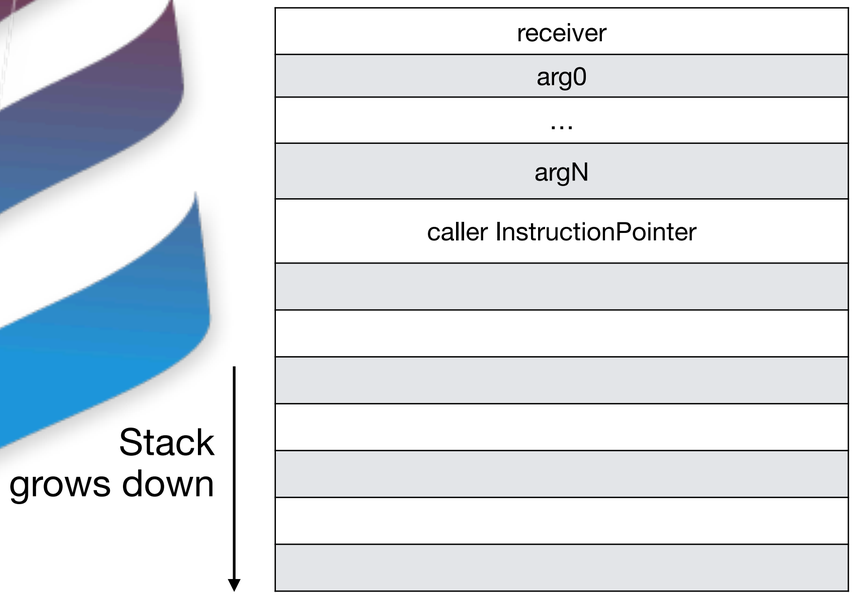
Pushed by Caller

```
Label 0:
        MoveCqR 0 ReceiverResultReg
        PushR LinkReg
        Call 16777688/10001D8
        AlignmentNops 8
Label 1:
        AndCqRR 7 ReceiverResultReg TempReg
        JumpNonZero Label 2
        MoveMwrR 0 ReceiverResultReg TempReg
        AndCqR 4194303/3FFFFF TempReg
        Nop
        Nop
        Nop
Label 2:
        CmpRR ClassReg TempReg
        JumpNonZero Label 0

        PushR LinkReg
        PushR FPReg
        MoveRR SPReg FPReg
        PushCw CogMethodAddress
        MoveCqR (#bitInvert oop) SendNumArgsReg
        PushR SendNumArgsReg
        PushR ReceiverResultReg

        PushCq 337/151
        MoveMwrR 0 SPReg ReceiverResultReg
        MovePatcheableC32R 1 ClassReg
        Call SendMessageRoutine

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

Frame Creation

Frame Building

Machine Code Frame Structure



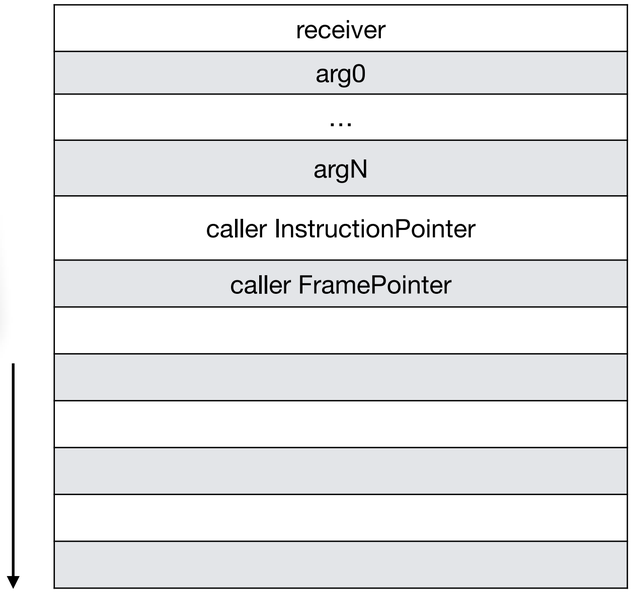
Where to return

```
Label 0:
        MoveCqR 0 ReceiverResultReg
        PushR LinkReg
        Call 16777688/10001D8
        AlignmentNops 8
        ...
        AndCqRR 7 ReceiverResultReg TempReg
        JumpNonZero Label 2
        MoveMwrR 0 ReceiverResultReg TempReg
        AndCqR 4194303/3FFFFF TempReg
        Nop
        Nop
        Nop

Label 2:
        CmpRR ClassReg TempReg
        JumpNonZero Label 0

        PushR LinkReg
        PushR FPReg
        MoveRR SPReg FPReg
        PushCw CogMethodAddress
        MoveCqR (#bitInvert oop) SendNumArgsReg
        PushR SendNumArgsReg
        PushR ReceiverResultReg

        PushCq 337/151
        MoveMwrR 0 SPReg ReceiverResultReg
        MovePatcheableC32R 1 ClassReg
        Call SendMessageRoutine

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

Frame Creation

Frame Building

Machine Code Frame Structure



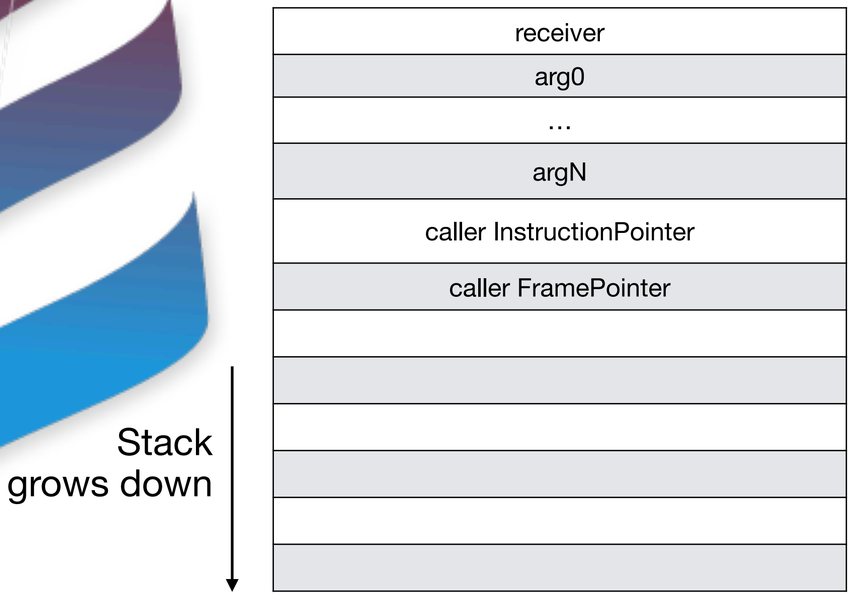
We keep a pointer to previous frame

```
Label 0:
        MoveCqR 0 ReceiverResultReg
        PushR LinkReg
        Call 16777688/10001D8
        AlignmentNops 8
entry:
        AndCqRR 7 ReceiverResultReg TempReg
        JumpNonZero Label 2
        MoveMwrR 0 ReceiverResultReg TempReg
        AndCqR 4194303/3FFFFF TempReg
        Nop
        Nop
        Nop

Label 2:
        CmpRR ClassReg TempReg
        JumpNonZero Label 0

        PushR LinkReg
        PushR FPReg
        MoveRR SPReg FPReg
        PushCw CogMethodAddress
        MoveCqR (#bitInvert oop) SendNumArgsReg
        PushR SendNumArgsReg
        PushR ReceiverResultReg

        PushCq 337/151
        MoveMwrR 0 SPReg ReceiverResultReg
        MovePatcheableC32R 1 ClassReg
        Call SendMessageRoutine

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

Frame Creation

Frame Building

Machine Code Frame Structure



Change the register to our frame

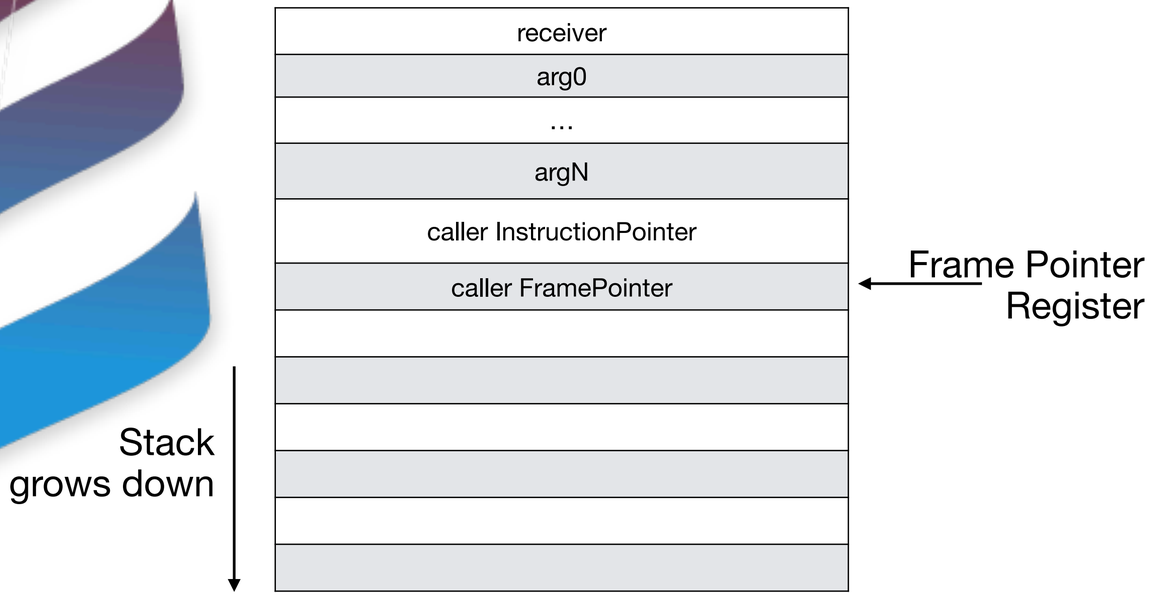
Frame Pointer Register

```
Label 0:
        MoveCqR 0 ReceiverResultReg
        PushR LinkReg
        Call 16777688/10001D8
        AlignmentNops 8
        ...
        AndCqRR 7 ReceiverResultReg TempReg
        JumpNonZero Label 2
        MoveMwrR 0 ReceiverResultReg TempReg
        AndCqR 4194303/3FFFFF TempReg
        Nop
        Nop
        Nop

Label 2:
        CmpRR ClassReg TempReg
        JumpNonZero Label 0

        PushR LinkReg
        PushR FPReg
        MoveRR SPReg FPReg
        PushCw CogMethodAddress
        MoveCqR (#bitInvert oop) SendNumArgsReg
        PushR SendNumArgsReg
        PushR ReceiverResultReg

        PushCq 337/151
        MoveMwrR 0 SPReg ReceiverResultReg
        MovePatcheableC32R 1 ClassReg
        Call SendMessageRoutine

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

Frame Creation

Frame Building

Machine Code Frame Structure

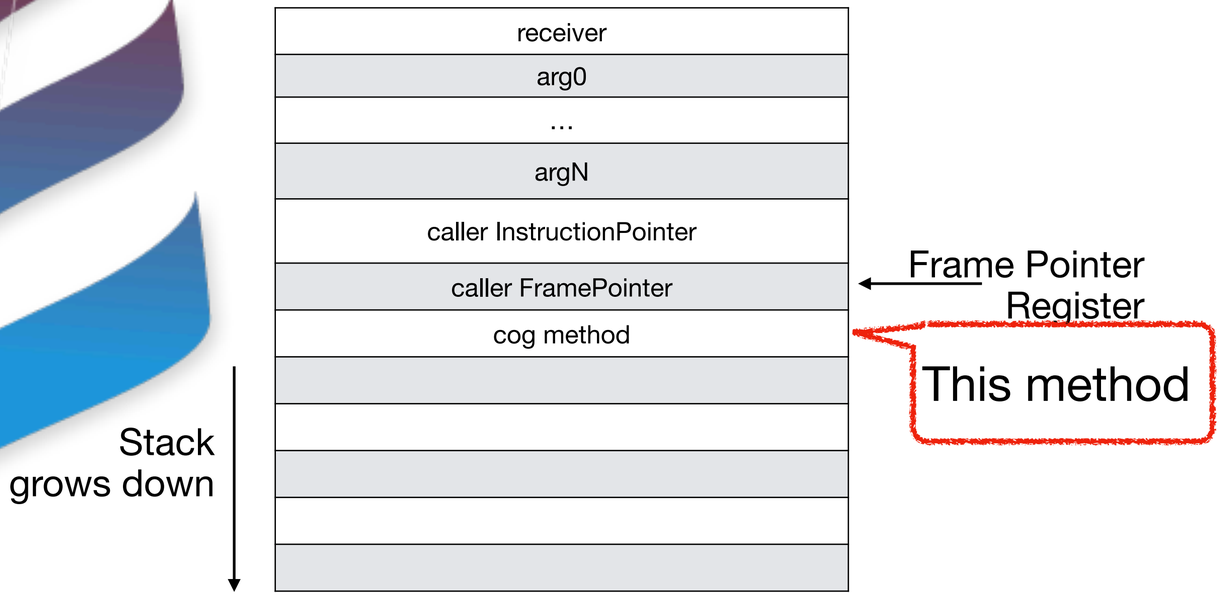



```
Label 0:
        MoveCqR 0 ReceiverResultReg
        PushR LinkReg
        Call 16777688/10001D8
        AlignmentNops 8

        AndCqRR 7 ReceiverResultReg TempReg
        JumpNonZero Label 2
        MoveMwrR 0 ReceiverResultReg TempReg
        AndCqR 4194303/3FFFFF TempReg
        Nop
        Nop
        Nop

Label 2:
        CmpRR ClassReg TempReg
        JumpNonZero Label 0

        PushR LinkReg
        PushR FPReg
        MoveRR SPReg FPReg
        PushCw CogMethodAddress
        MoveCqR (#bitInvert oop) SendNumArgsReg
        PushR SendNumArgsReg
        PushR ReceiverResultReg

        PushCq 337/151
        MoveMwrR 0 SPReg ReceiverResultReg
        MovePatcheableC32R 1 ClassReg
        Call SendMessageRoutine

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

Frame Creation

Frame Building

Machine Code Frame Structure




```
Label 0:
        MoveCqR 0 ReceiverResultReg
        PushR LinkReg
        Call 16777688/10001D8
        AlignmentNops 8
        ...
        AndCqRR 7 ReceiverResultReg TempReg
        JumpNonZero Label 2
        MoveMwrR 0 ReceiverResultReg TempReg
        AndCqR 4194303/3FFFFF TempReg
        Nop
        Nop
        Nop

Label 2:
        CmpRR ClassReg TempReg
        JumpNonZero Label 0

        PushR LinkReg
        PushR FPReg
        MoveRR SPReg FPReg
        PushCw CogMethodAddress
        MoveCqR (#bitInvert oop) SendNumArgsReg
        PushR SendNumArgsReg
        PushR ReceiverResultReg

        PushCq 337/151
        MoveMwrR 0 SPReg ReceiverResultReg
        MovePatcheableC32R 1 ClassReg
        Call SendMessageRoutine

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

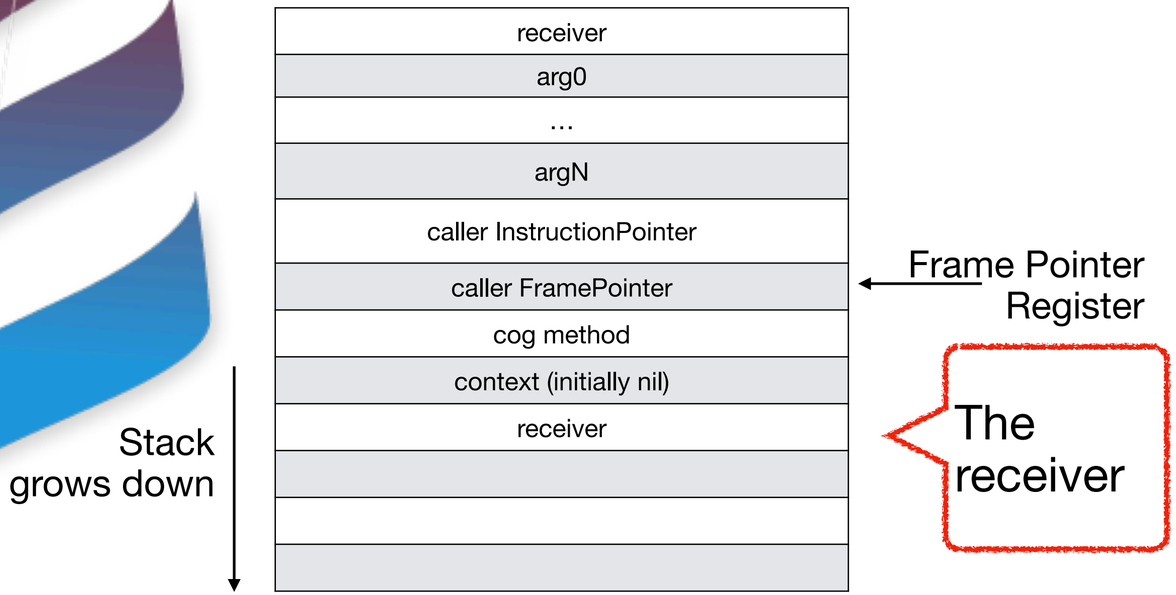
Frame Building

Machine Code Frame Structure


```
Label 0:
        MoveCqR 0 ReceiverResultReg
        PushR LinkReg
        Call 16777688/10001D8
        AlignmentNops 8
Label 1:
        AndCqRR 7 ReceiverResultReg TempReg
        JumpNonZero Label 2
        MoveMwrR 0 ReceiverResultReg TempReg
        AndCqR 4194303/3FFFFF TempReg
        Nop
        Nop
        Nop

Label 2:
        CmpRR ClassReg TempReg
        JumpNonZero Label 0

        PushR LinkReg
        PushR FPReg
        MoveRR SPReg FPReg
        PushCw CogMethodAddress
        MoveCqR (#bitInvert oop) SendNumArgsReg
        PushR SendNumArgsReg
        PushR ReceiverResultReg

        PushCq 337/151
        MoveMwrR 0 SPReg ReceiverResultReg
        MovePatcheableC32R 1 ClassReg
        Call SendMessageRoutine

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

Frame Building

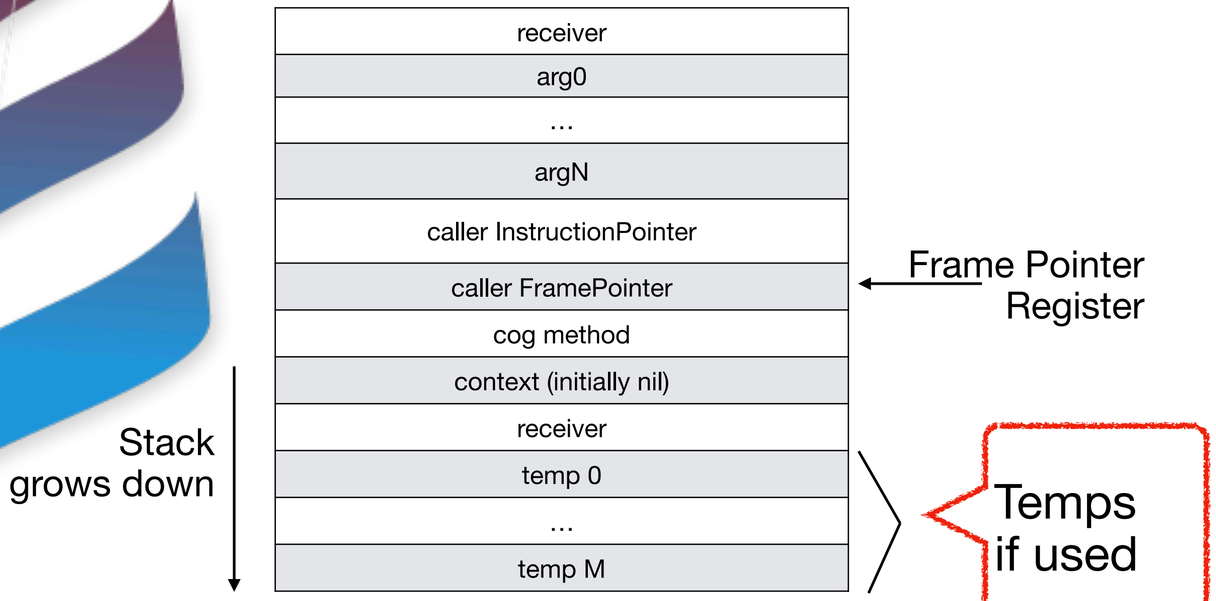
Machine Code Frame Structure




```
Label 0:
        MoveCqR 0 ReceiverResultReg
        PushR LinkReg
        Call 16777688/10001D8
        AlignmentNops 8
Label 1:
        AndCqRR 7 ReceiverResultReg TempReg
        JumpNonZero Label 2
        MoveMwrR 0 ReceiverResultReg TempReg
        AndCqR 4194303/3FFFFF TempReg
        Nop
        Nop
        Nop

Label 2:
        CmpRR ClassReg TempReg
        JumpNonZero Label 0

        PushR LinkReg
        PushR FPReg
        MoveRR SPReg FPReg
        PushCw CogMethodAddress
        MoveCqR (#bitInvert oop) SendNumArgsReg
        PushR SendNumArgsReg
        PushR ReceiverResultReg

        PushCq 337/151
        MoveMwrR 0 SPReg ReceiverResultReg
        MovePatcheableC32R 1 ClassReg
        Call SendMessageRoutine

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

Frame Building

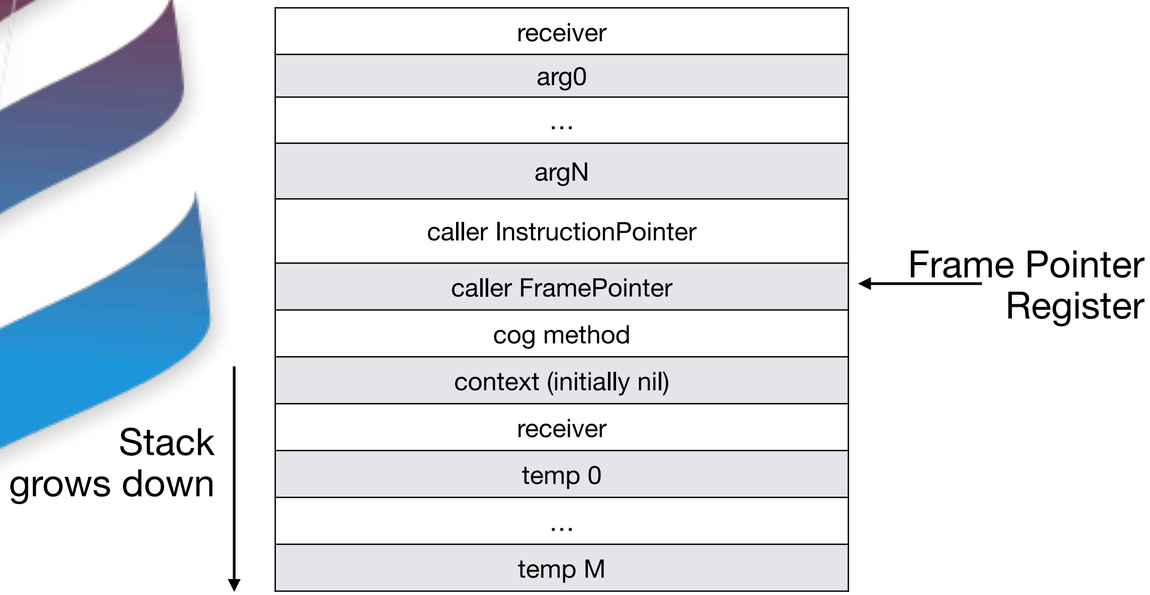
Machine Code Frame Structure




```
Label 0:
        MoveCqR 0 ReceiverResultReg
        PushR LinkReg
        Call 16777688/10001D8
        AlignmentNops 8

        AndCqRR 7 ReceiverResultReg TempReg
        JumpNonZero Label 2
        MoveMwrR 0 ReceiverResultReg TempReg
        AndCqR 4194303/3FFFFF TempReg
        Nop
        Nop
        Nop

Label 2:
        CmpRR ClassReg TempReg
        JumpNonZero Label 0

        PushR LinkReg
        PushR FPReg
        MoveRR SPReg FPReg
        PushCw CogMethodAddress
        MoveCqR (#bitInvert oop) SendNumArgsReg
        PushR SendNumArgsReg
        PushR ReceiverResultReg

        PushCq 337/151
        MoveMwrR 0 SPReg ReceiverResultReg
        MovePatcheableC32R 1 ClassReg
        Call SendMessageRoutine

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

Frame Building

Machine Code Frame Structure



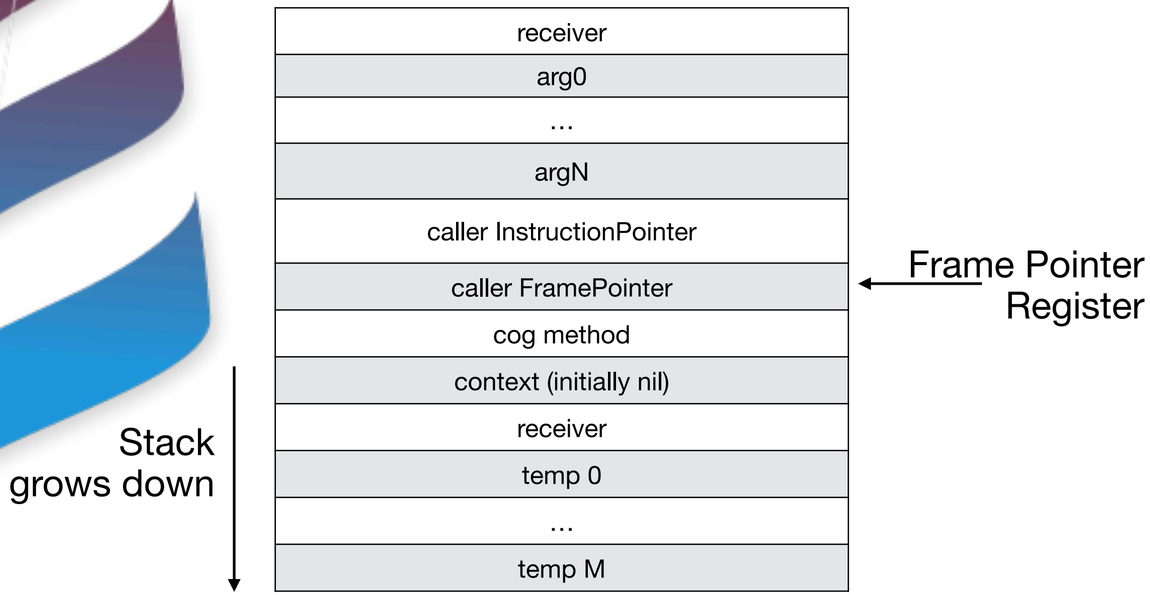

```
Label 0:
        MoveCqR 0 ReceiverResultReg
        PushR LinkReg
        Call 16777688/10001D8
        AlignmentNops 8
Label 1:
        AndCqRR 7 ReceiverResultReg TempReg
        JumpNonZero Label 2
        MoveMwrR 0 ReceiverResultReg TempReg
        AndCqR 4194303/3FFFFF TempReg
        Nop
        Nop
        Nop

Label 2:
        CmpRR ClassReg TempReg
        JumpNonZero Label 0

        PushR LinkReg
        PushR FPReg
        MoveRR SPReg FPReg
        PushCw CogMethodAddress
        MoveCqR (#bitInvert oop) SendNumArgsReg
        PushR SendNumArgsReg
        PushR ReceiverResultReg

        PushCq 337/151
        MoveMwrR 0 SPReg ReceiverResultReg
        MovePatcheableC32R 1 ClassReg
        Call SendMessageRoutine

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

Frame Creation

Attention, it is not the same in the interpreter

Frame Building

Machine Code Frame Structure



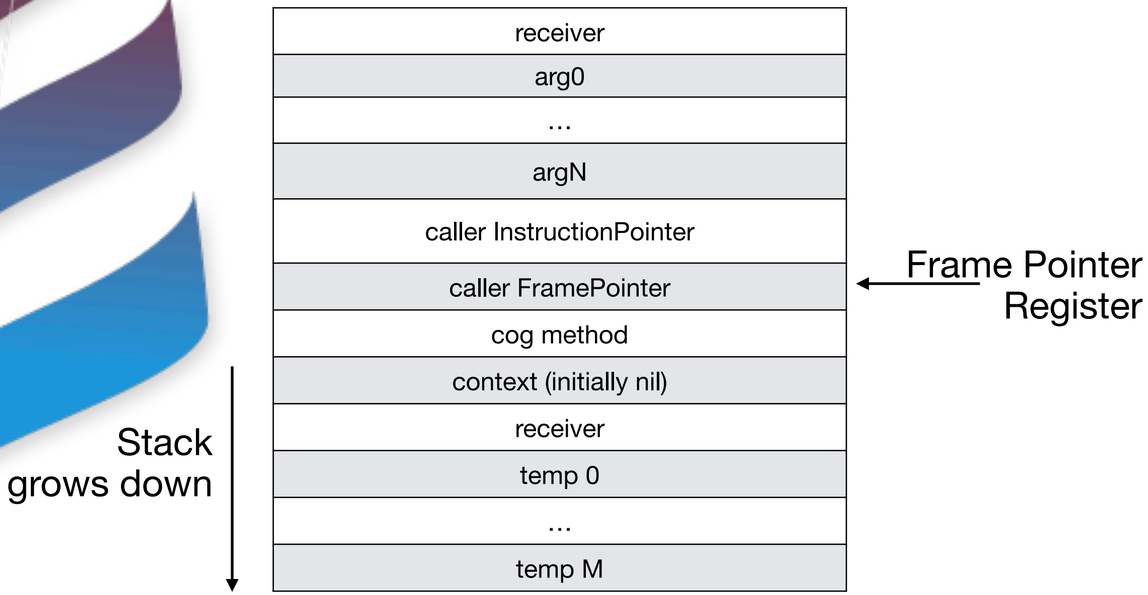

```
Label 0:
        MoveCqR 0 ReceiverResultReg
        PushR LinkReg
        Call 16777688/10001D8
        AlignmentNops 8
        …
        AndCqRR 7 ReceiverResultReg TempReg
        JumpNonZero Label 2
        MoveMwrR 0 ReceiverResultReg TempReg
        AndCqR 4194303/3FFFFF TempReg
        Nop
        Nop
        Nop

Label 2:
        CmpRR ClassReg TempReg
        JumpNonZero Label 0

        PushR LinkReg
        PushR FPReg
        MoveRR SPReg FPReg
        PushCw CogMethodAddress
        MoveCqR (#bitInvert oop) SendNumArgsReg
        PushR SendNumArgsReg
        PushR ReceiverResultReg

        PushCq 337/151
        MoveMwrR 0 SPReg ReceiverResultReg
        MovePatcheableC32R 1 ClassReg
        Call SendMessageRoutine

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

Frame Creation

Attention, it is not the same in the interpreter

Frame Pop

Machine Code Frame Structure

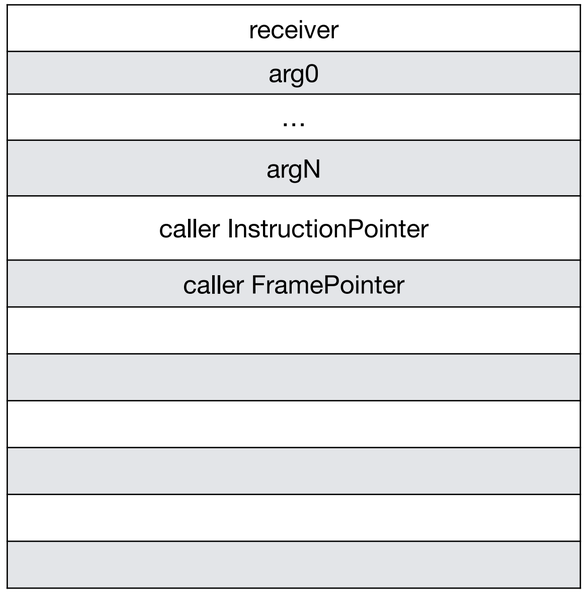



```
Label 0:
        MoveCqR 0 ReceiverResultReg
        PushR LinkReg
        Call 16777688/10001D8
        AlignmentNops 8
Label 1:
        AndCqRR 7 ReceiverResultReg TempReg
        JumpNonZero Label 2
        MoveMwrR 0 ReceiverResultReg TempReg
        AndCqR 4194303/3FFFFF TempReg
        Nop
        Nop
        Nop

Label 2:
        CmpRR ClassReg TempReg
        JumpNonZero Label 0

        PushR LinkReg
        PushR FPReg
        MoveRR SPReg FPReg
        PushCw CogMethodAddress
        MoveCqR (#bitInvert oop) SendNumArgsReg
        PushR SendNumArgsReg
        PushR ReceiverResultReg

        PushCq 337/151
        MoveMwrR 0 SPReg ReceiverResultReg
        MovePatcheableC32R 1 ClassReg
        Call SendMessageRoutine

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

Frame Pop & Return

Frame Pop

Machine Code Frame Structure

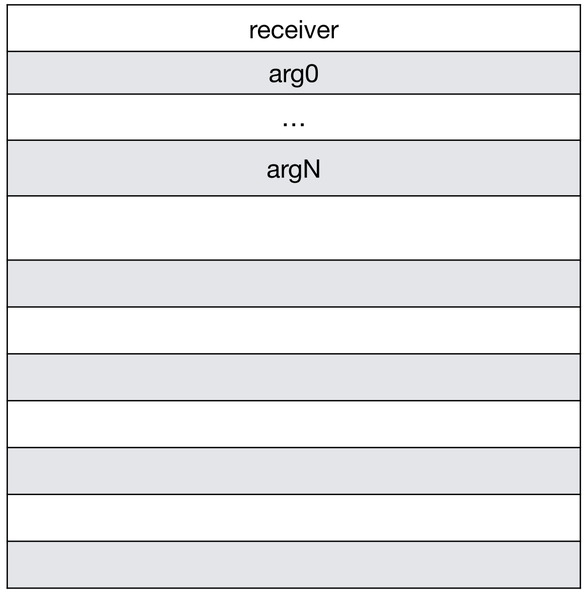



```
Label 0:
        MoveCqR 0 ReceiverResultReg
        PushR LinkReg
        Call 16777688/10001D8
        AlignmentNops 8
Label 1:
        AndCqRR 7 ReceiverResultReg TempReg
        JumpNonZero Label 2
        MoveMwrR 0 ReceiverResultReg TempReg
        AndCqR 4194303/3FFFFF TempReg
        Nop
        Nop
        Nop

Label 2:
        CmpRR ClassReg TempReg
        JumpNonZero Label 0

        PushR LinkReg
        PushR FPReg
        MoveRR SPReg FPReg
        PushCw CogMethodAddress
        MoveCqR (#bitInvert oop) SendNumArgsReg
        PushR SendNumArgsReg
        PushR ReceiverResultReg

        PushCq 337/151
        MoveMwrR 0 SPReg ReceiverResultReg
        MovePatcheableC32R 1 ClassReg
        Call SendMessageRoutine

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

Frame pop & Return

Frame Pop

Machine Code Frame Structure




```
Label 0:
        MoveCqR 0 ReceiverResultReg
        PushR LinkReg
        Call 16777688/10001D8
        AlignmentNops 8
        ...
        AndCqRR 7 ReceiverResultReg TempReg
        JumpNonZero Label 2
        MoveMwrR 0 ReceiverResultReg TempReg
        AndCqR 4194303/3FFFFF TempReg
        Nop
        Nop
        Nop

Label 2:
        CmpRR ClassReg TempReg
        JumpNonZero Label 0

        PushR LinkReg
        PushR FPReg
        MoveRR SPReg FPReg
        PushCw CogMethodAddress
        MoveCqR (#bitInvert oop) SendNumArgsReg
        PushR SendNumArgsReg
        PushR ReceiverResultReg

        PushCq 337/151
        MoveMwrR 0 SPReg ReceiverResultReg
        MovePatcheableC32R 1 ClassReg
        Call SendMessageRoutine

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

Frame pop & Return

Frame Pop

Machine Code Frame Structure


```
Label 0:
        MoveCqR 0 ReceiverResultReg
        PushR LinkReg
        Call 16777688/10001D8
        AlignmentNops 8
        [illegible]
        AndCqRR 7 ReceiverResultReg TempReg
        JumpNonZero Label 2
        MoveMwrR 0 ReceiverResultReg TempReg
        AndCqR 4194303/3FFFFF TempReg
        Nop
        Nop
        Nop

Label 2:
        CmpRR ClassReg TempReg
        JumpNonZero Label 0

        PushR LinkReg
        PushR FPReg
        MoveRR SPReg FPReg
        PushCw CogMethodAddress
        MoveCqR (#bitInvert oop) SendNumArgsReg
        PushR SendNumArgsReg
        PushR ReceiverResultReg

        PushCq 337/151
        MoveMwrR 0 SPReg ReceiverResultReg
        MovePatcheableC32R 1 ClassReg
        Call SendMessageRoutine

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

Frame Pop & Return

Primitives
Basics

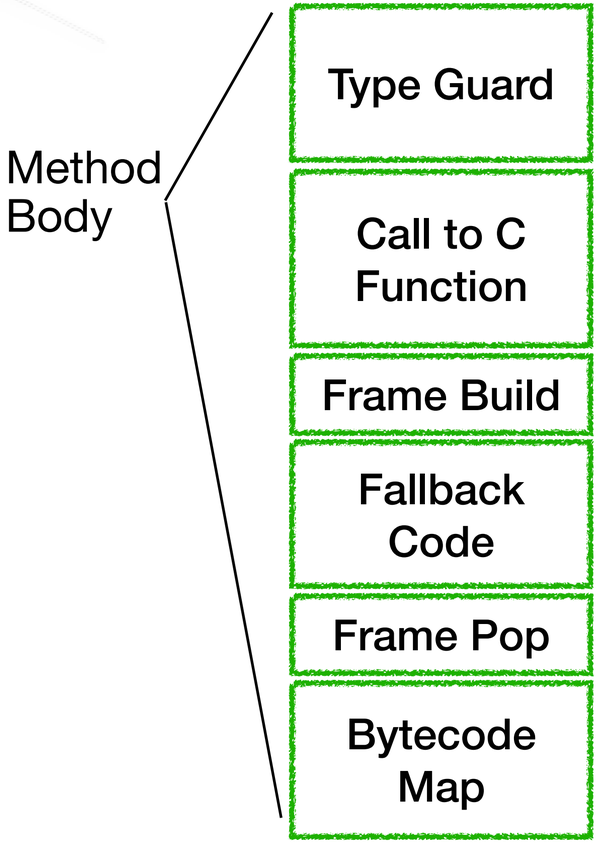
- A primitive is a special kind of method
- They have:
  - an implementation in C or Native Code
  - fallback code (in case the previous implementation) fails
- We need the same behaviour in a Cog Method

Primitives
Many Kinds

- Depending in the native part implementation
  - A Function written in C (interpreter / plugin)
  - A Complete Machine Code implementation
  - A partial Machine Code + delegating in C Function

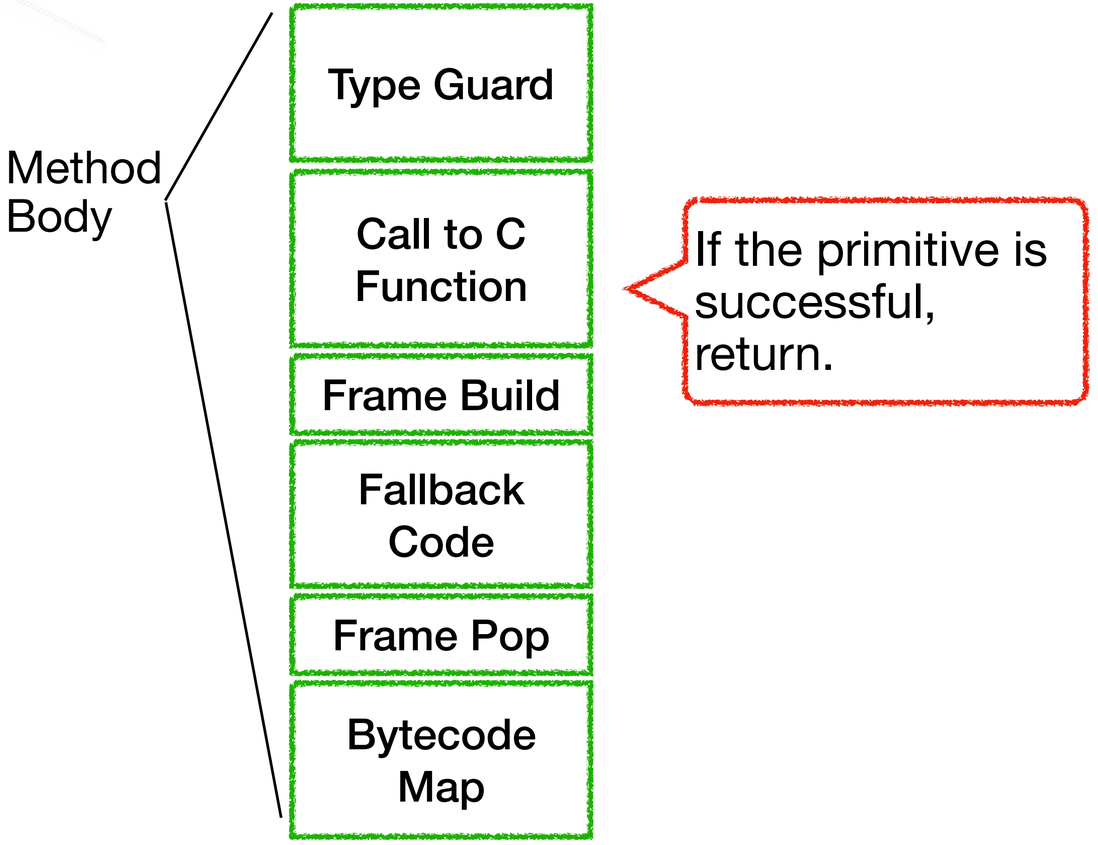
Primitives

Only C Function



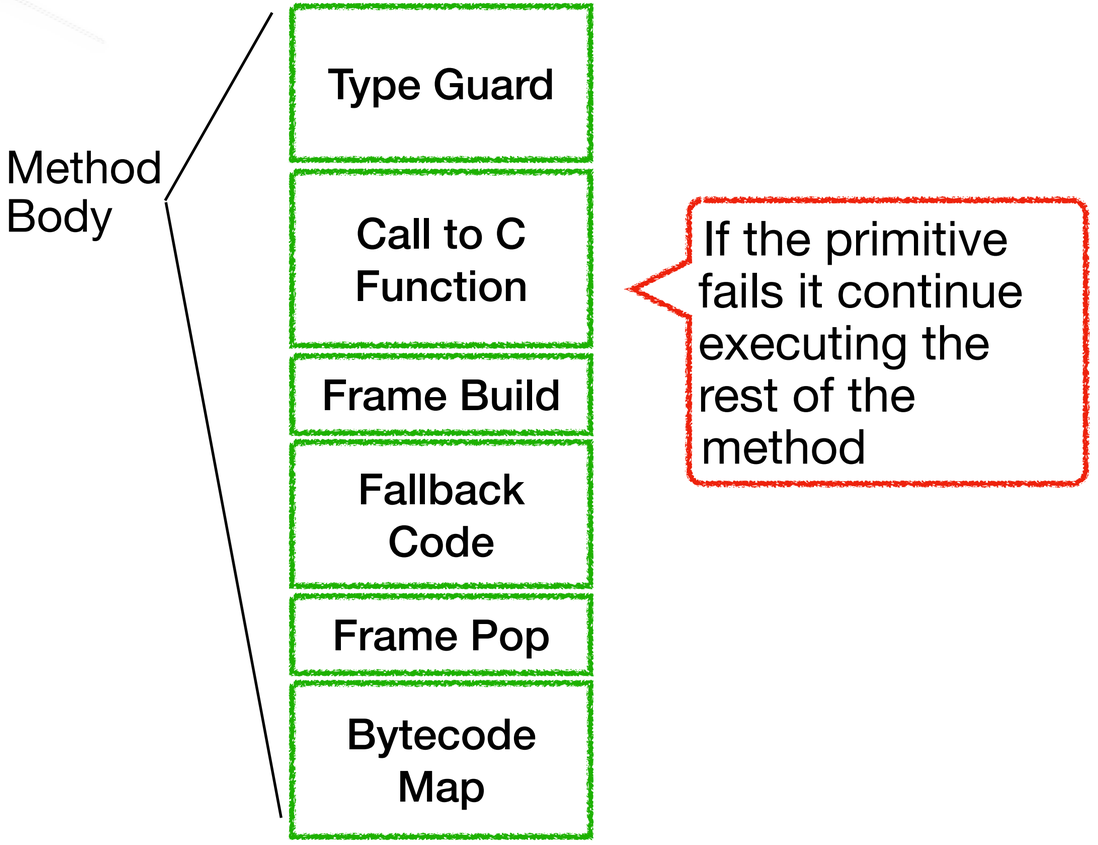
Primitives

Only C Function



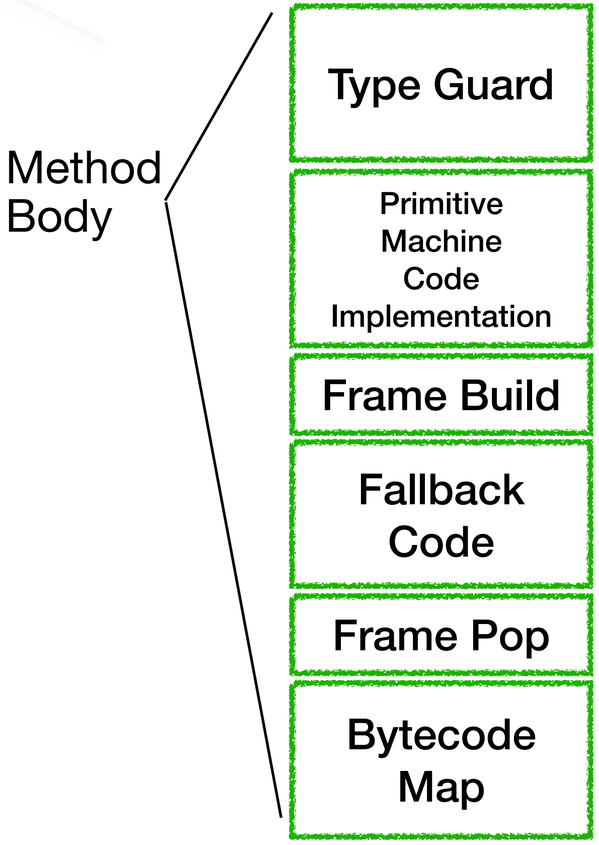
Primitives

Only C Function



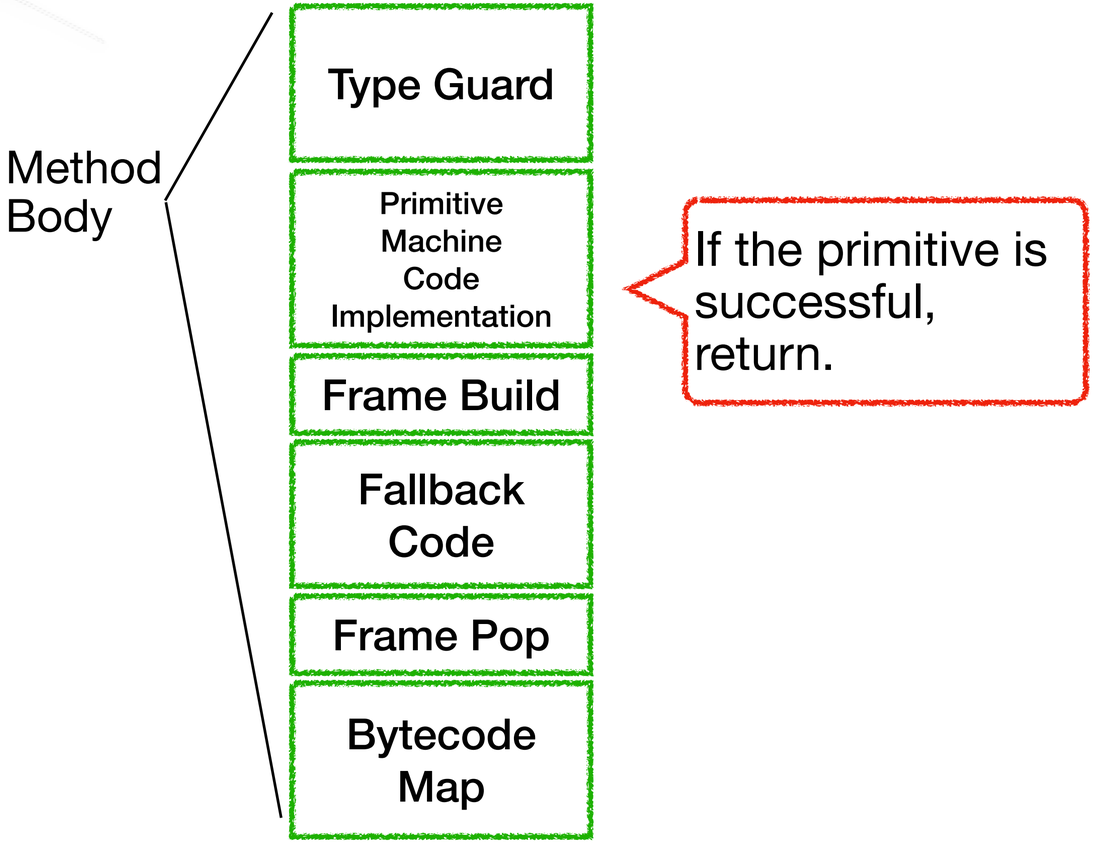
Primitives

Only Machine Code Implementation



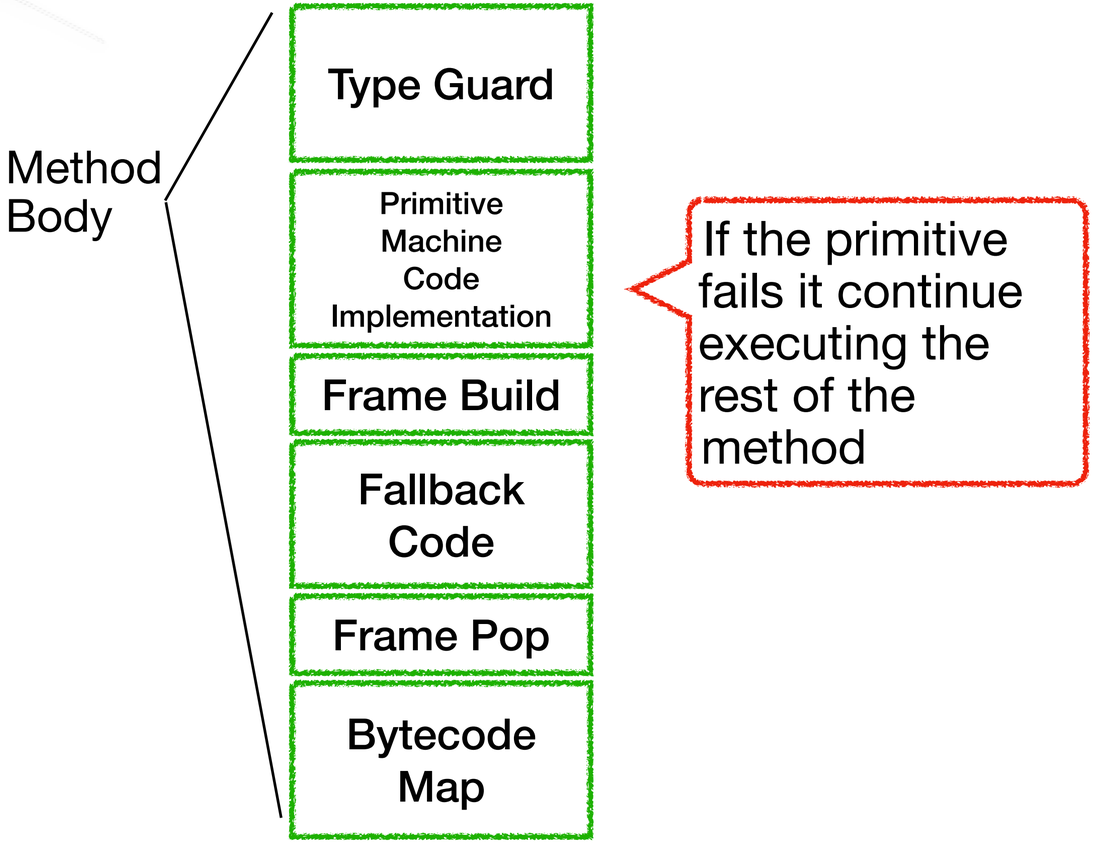
Primitives

Only Machine Code Implementation



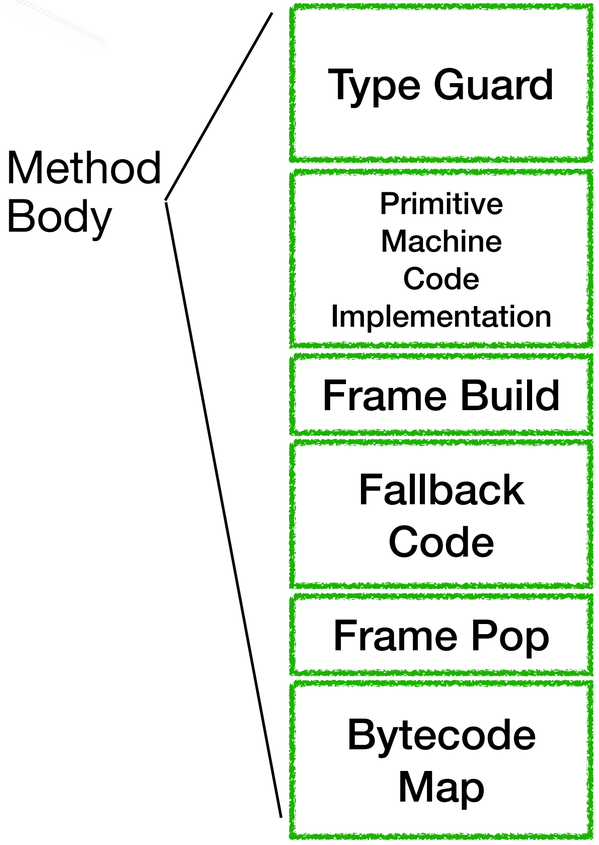
Primitives

Only Machine Code Implementation



Primitives

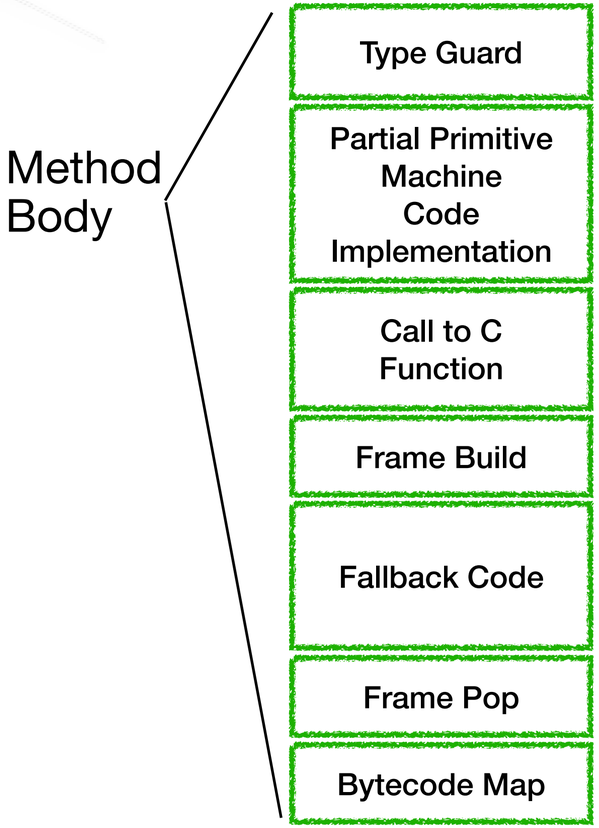
Only Machine Code Implementation



They are called
“Complete Primitives”

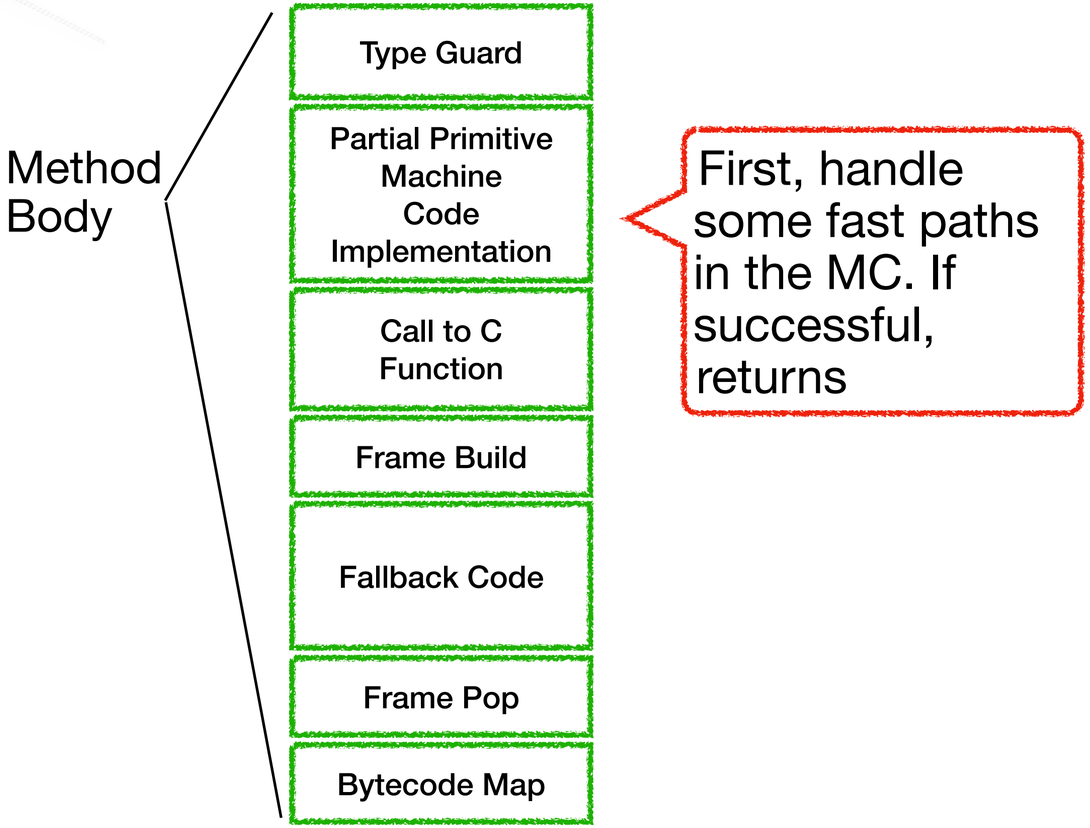
Primitives

A partial Machine Code + delegating in C Function



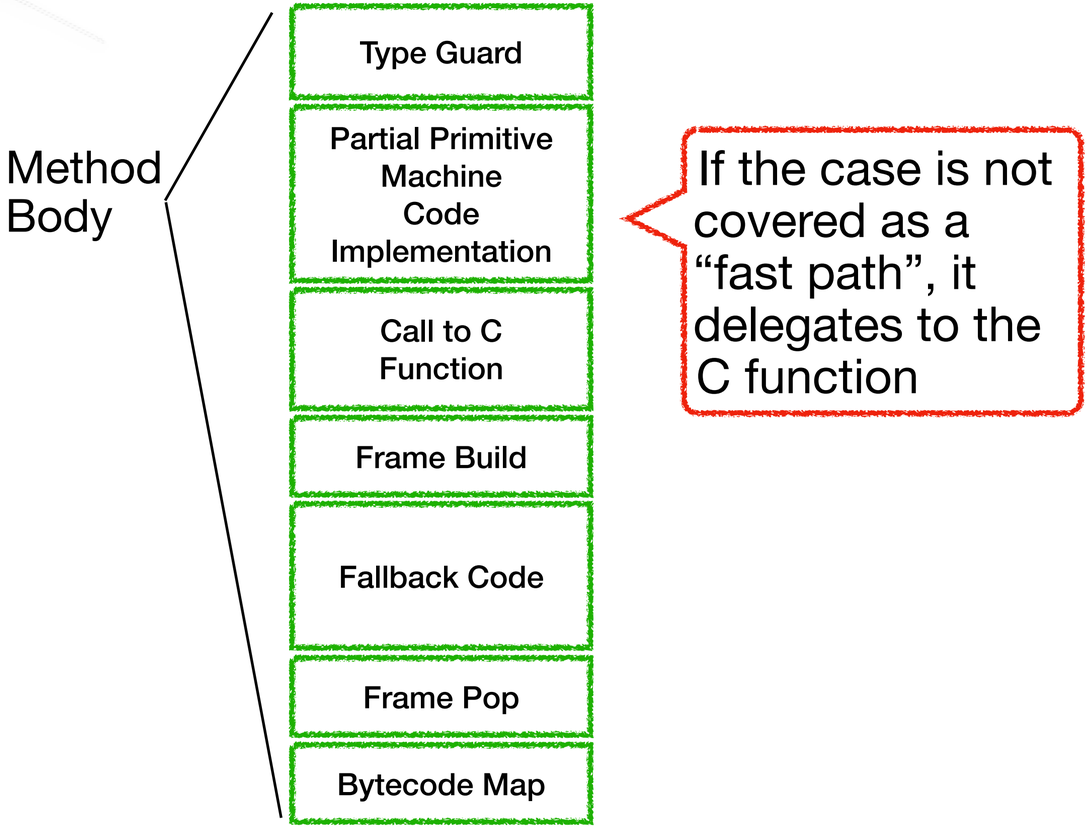
Primitives

A partial Machine Code + delegating in C Function



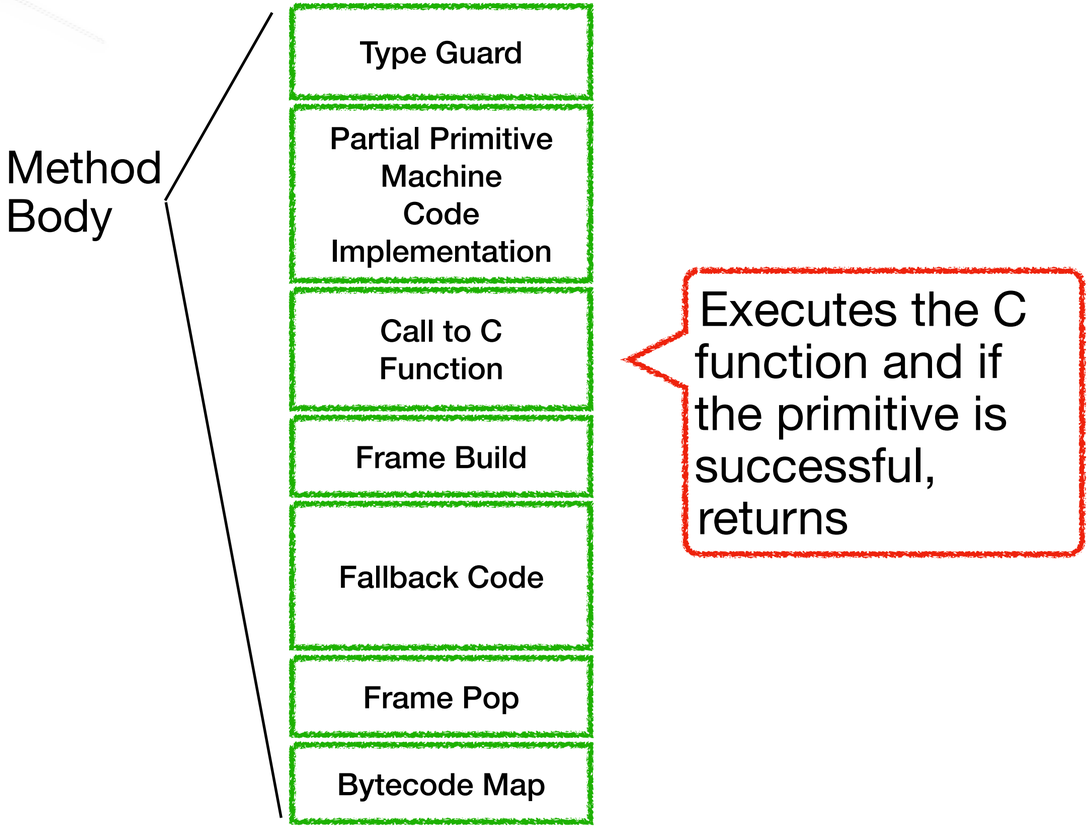
Primitives

A partial Machine Code + delegating in C Function



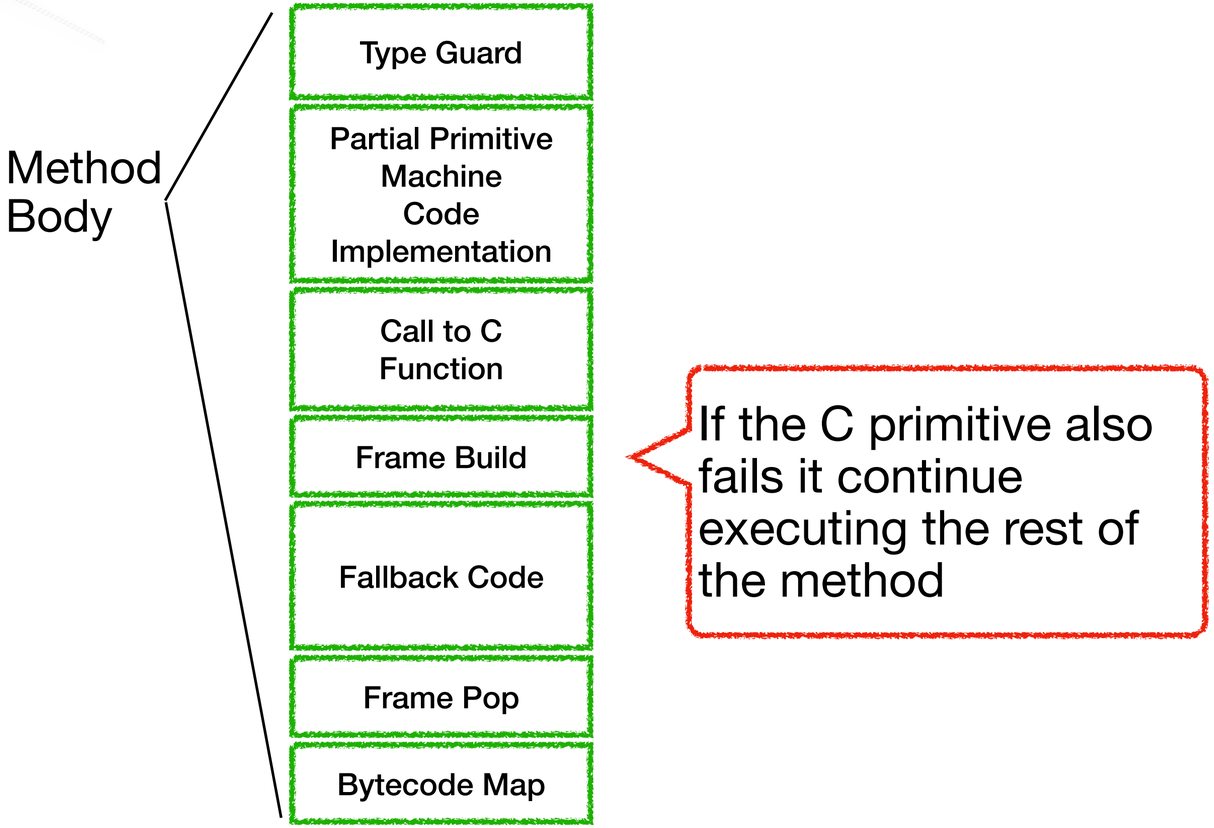
Primitives

A partial Machine Code + delegating in C Function



Primitives

A partial Machine Code + delegating in C Function



Cog Methods
Remaining Subjects

- Blocks (3 flavours)
- Message Sends & PICs
- Bytecode Mapping
- Smalltalk-Stack Primitives
- ….

Cog Methods

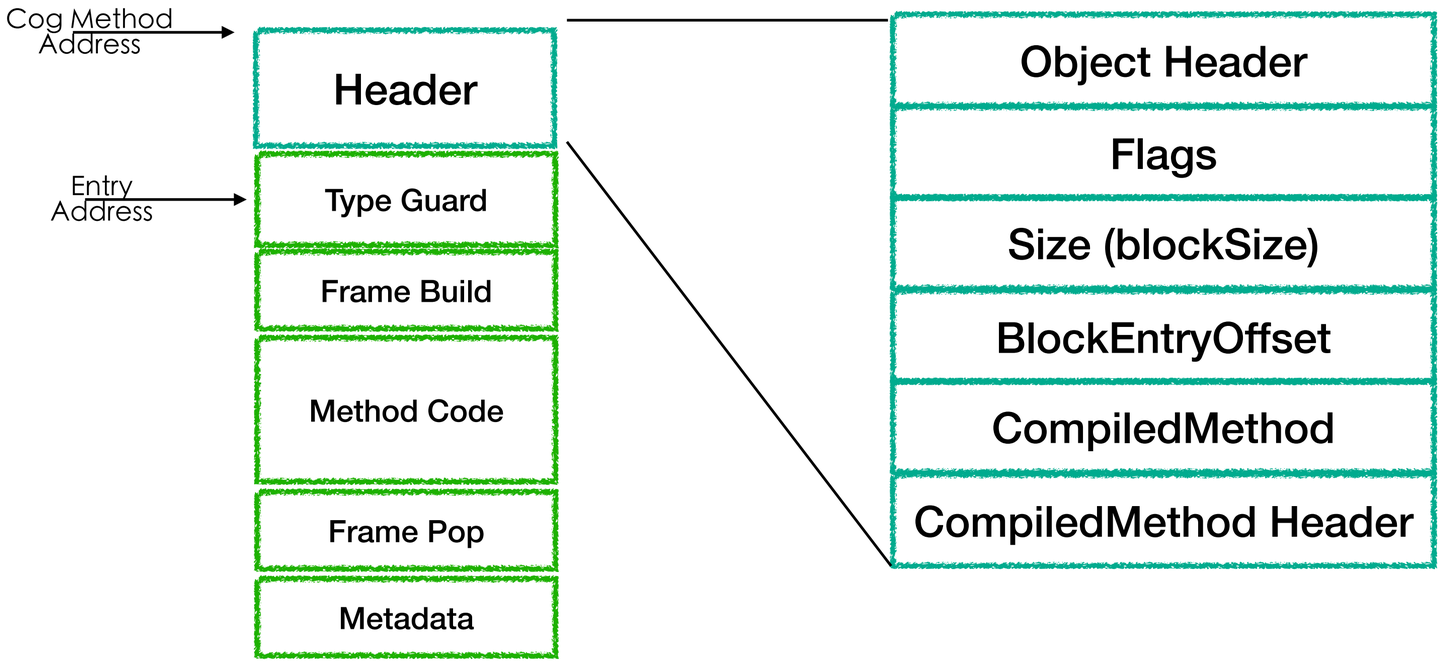
Remaining Subjects

- Blocks (3 flavours)
- Message Sends & PICs
- Bytecode Mapping
- Smalltalk-Stack Primitives
- Code Zone compaction
- ….

More to continue learning

Structure of Machine Code Methods

An introduction



Thanks!!!!