

What Lies Beneath?

A tour of the dark gritty underbelly
of OpenJDK

Andrew Dinn
Roman Kennke
Andrew Haley
Christine H. Flood

What Lies Beneath?

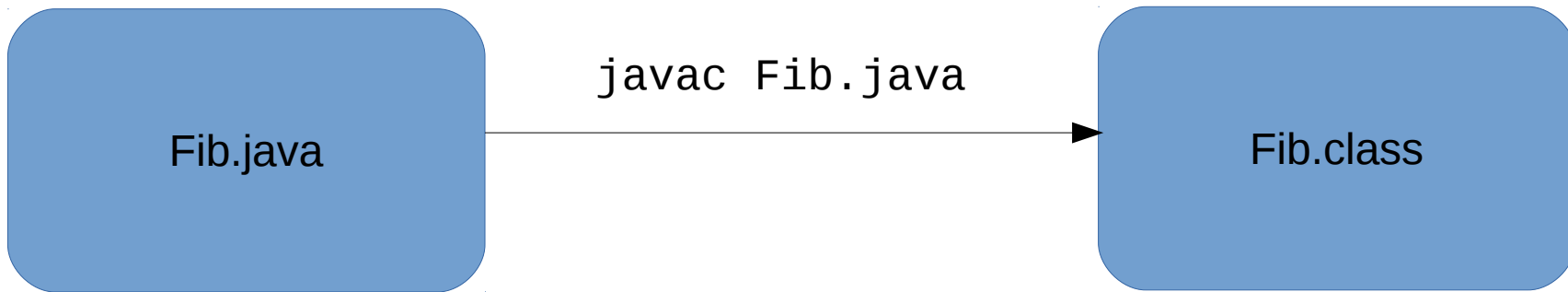
- Bytecode
- Template Interpreter
- C1 JIT Compiler
- C2 JIT Compiler
- Special Tricks
- Questions

We start with everyone's favorite Java program.

```
class Fib {
    static int fib(int x) {
        if ((x == 1) || (x == 2))
            return 1;
        else return (fib(x-1) + fib(x-2));
    }

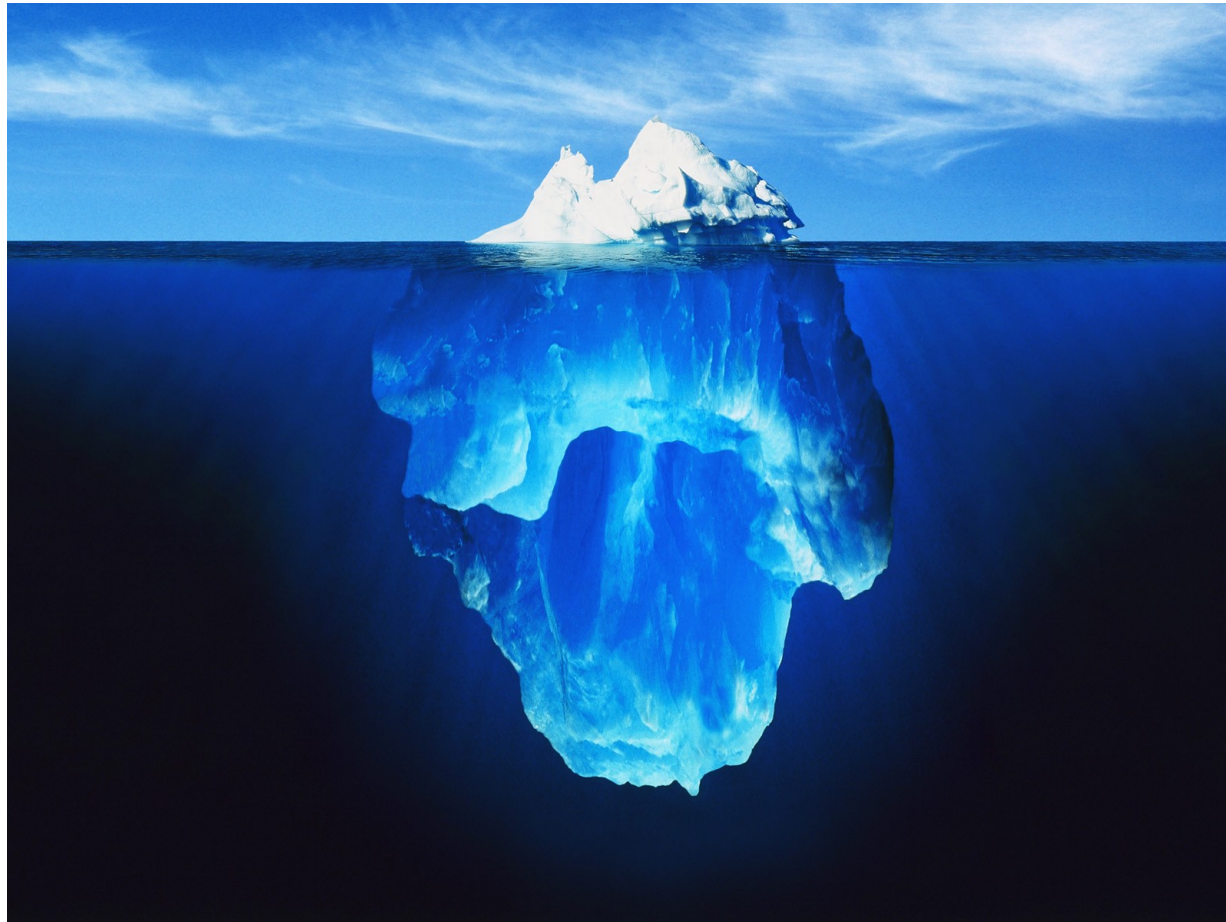
    public static void main(String args[]) {
        int arg = Integer.parseInt(args[0]);
        System.out.println("Fib of " + arg + " = " + fib(arg));
    }
}
```

What you see.



```
java Fib 17  
Fib of 17 = 1597
```

There's a lot happening below the surface.



- **Bytecode**
- Template Interpreter
- C1 JIT Compiler
- C2 JIT Compiler
- Special Tricks
- Questions

javac generates Java Bytecode

```
static int fib(int);
```

```
  flags: ACC_STATIC
```

```
Code:
```

```
  stack=3, locals=1, args_size=1
```

```
   0: iload_0                13: iconst_1
   1: iconst_1                14: isub
   2: if_icmpeq               10    15: invokestatic #2 // Method fib:(I)I
   5: iload_0                18: iload_0
   6: iconst_2                19: iconst_2
   7: if_icmpne               12    20: isub
  10: iconst_1                21: invokestatic #2 // Method fib:(I)I
  11: ireturn                 24: iadd
  12: iload_0                25: ireturn
```

Bytecode verification

- Abstract interpretation
 - Interpret the program except instead of values you are calculating the types of the stack and the locals at each instruction.
 - Merge points require merging types.

Bytecode Abstract Interpretation

```
static int fib(int);
```

```
  flags: ACC_STATIC
```

```
Code:
```

```
  stack=3, locals=1, args_size=1
```

```
    0: iload_0
```

```
    1: iconst_1
```

```
    2: if_icmpeq      10
```

```
    5: iload_0
```

```
    6: iconst_2
```

```
    7: if_icmpne      12
```

```
   10: iconst_1
```

```
   11: ireturn
```

```
    . . .
```

```
stack=[], locals = [int]
```

```
stack = [int], locals=[int]
```

```
stack = [1, int]
```

```
stack = []
```

```
stack = [int]
```

```
stack = [2, int]
```

```
stack = []
```

```
stack = [1]
```

```
. . .
```

What Lies Beneath?

- Bytecode
- **Template Interpreter**
- C1 JIT Compiler
- C2 JIT Compiler
- Special Tricks
- Questions

Template Interpreter

- Interpreter only execution

```
$ java -Xint -XX:+PrintInterpreter Fib 17
```

- Use of `PrintInterpreter` requires
 - `hsdis-amd64.so`
- For product release jvms must also unlock
 - `-XX:+UnlockDiagnosticVMOptions`

hsdis-amd64.so

- Download hsdis lib from
 - <https://kenai.com/projects/base-hsdis/downloads>
- e.g. for Linux
 - `$ wget https://kenai.com/projects/base-hsdis/downloads/download/linux-hsdis-amd64.so`
- use correct name
 - `$ mv linux-hsdis-amd64.so hsdis-amd64.so`
 - ensure it is in your `LD_LIBRARY_PATH`
 - or copy it to `${JAVA_HOME}/jre/lib/amd64`

Template Interpreter

```
static int fib(int);
```

```
    flags: ACC_STATIC
```

```
Code:
```

```
    stack=3, locals=1, args_size=1
```

```
    0: iload_0
```

```
    1: iconst_1
```

```
    2: if_icmpeq      10
```

```
    5: iload_0
```

```
    6: iconst_2
```

```
    7: if_icmpne     12
```

```
   10: iconst_1
```

```
   11: ireturn
```

```
    . . .
```

```
iload_0  26  iload_0  [0x7f077902b040,  
0x7f077902b0a0]  96 bytes
```

```
0x7f077902b040: push  %rax
```

```
0x7f077902b041: jmpq  0x7f077902b070
```

```
0x7f077902b046: sub   $0x8,%rsp
```

```
0x7f077902b04a: vmovss %xmm0,(%rsp)
```

```
0x7f077902b04f: jmpq  0x7f077902b070
```

```
0x7f077902b054: sub   $0x10,%rsp
```

```
0x7f077902b058: vmovsd %xmm0,(%rsp)
```

```
0x7f077902b05d: jmpq  0x7f077902b070
```

```
0x7f077902b062: sub   $0x10,%rsp
```

```
0x7f077902b066: mov   %rax,(%rsp)
```

```
0x7f077902b06a: jmpq  0x7f077902b070
```

```
0x7f077902b06f: push  %rax
```

```
0x7f077902b070: mov  (%r14),%eax
```

```
0x7f077902b073: movzbl 0x1(%r13),%ebx
```

```
0x7f077902b078: inc  %r13
```

```
0x7f077902b07b: mov  $0x7f078ff6af00,%r10
```

```
0x7f077902b085: jmpq  *(%r10,%rbx,8)
```

Template Interpreter

```
static int fib(int);
```

```
flags: ACC_STATIC
```

```
Code:
```

```
stack=3, locals=1, args_size=1
```

```
0: iload_0
```

```
1: iconst_1
```

```
2: if_icmpeq      10
```

```
5: iload_0
```

```
6: iconst_2
```

```
7: if_icmpne     12
```

```
10: iconst_1
```

```
11: ireturn
```

```
. . .
```

```
iconst_1 4 iconst_1 [0x7f0779029a60,  
0x7f0779029ac0] 96 bytes
```

```
0x7f0779029a60: push %rax
```

```
0x7f0779029a61: jmpq 0x7f0779029a90
```

```
0x7f0779029a66: sub $0x8,%rsp
```

```
0x7f0779029a6a: vmovss %xmm0,(%rsp)
```

```
0x7f0779029a6f: jmpq 0x7f0779029a90
```

```
0x7f0779029a74: sub $0x10,%rsp
```

```
0x7f0779029a78: vmovsd %xmm0,(%rsp)
```

```
0x7f0779029a7d: jmpq 0x7f0779029a90
```

```
0x7f0779029a82: sub $0x10,%rsp
```

```
0x7f0779029a86: mov %rax,(%rsp)
```

```
0x7f0779029a8a: jmpq 0x7f0779029a90
```

```
0x7f0779029a8f: push %rax
```

```
0x7f0779029a90: mov $0x1,%eax
```

```
0x7f0779029a95: movzbl 0x1(%r13),%ebx
```

```
0x7f0779029a9a: inc %r13
```

```
0x7f0779029a9d: mov $0x7f078ff6af00,%r10
```

```
0x7f0779029aa7: jmpq *(%r10,%rbx,8)
```

Template Interpreter

```
static int fib(int);
  flags: ACC_STATIC
Code:
  stack=3, locals=1, args_size=1
    0: iload_0
    1: iconst_1
    2: if_icmpeq      10
    5: iload_0
    6: iconst_2
    7: if_icmpne     12
   10: iconst_1
   11: ireturn
      . . .
```

- Profile which branch taken
 - MethodData holds profile counters

Template Interpreter

```
    . . .  
13: iconst_1  
14: isub  
15: invokestatic #2 // Method  
fib:(I)I  
18: iload_0  
19: iconst_2  
20: isub  
21: invokestatic #2 // Method  
fib:(I)I  
24: iadd  
25: ireturn
```

- Need to load class?
- Fetch new MethodData
- Build call frame
 - args become locals
 - push/reload locals reg
 - push/reload method reg
 - push/reload bcp reg
- Profile call

Interpreter Performance

```
$ time java -Xint Fib 42
```

```
Fib of 42 = 267914296
```

```
real 0m41.312s
```

```
user 0m41.143s
```

```
sys 0m0.152s
```

What Lies Beneath?

- Bytecode
- Template Interpreter
- **C1 JIT Compiler**
- C2 JIT Compiler
- Special Tricks
- Questions

C1 JIT Compiler

- Client compiler
 - for short running desktop applications
- Relatively Standard Optimising Compiler
 - see the Dragon Book (and the code :-)

```
$ java -XX:+PrintIR2 -XX:+PrintCFG2 -XX:  
+PrintAssembly -XX:CompileOnly=Fib -XX:  
+CommentedAssembly -XX:TieredStopAtLevel=2 -XX:  
+DebugNonSafepoints Fib 24
```

- n.b. most options are debug build only

```
PrintIR2 PrintCFG2 CommentedAssembly  
TieredStopAtLevel DebugNonSafepoints
```

How to build a debug jdk8

- Obtain forest

```
$ hg clone http://hg.openjdk.java.net/jdk8u/jdk8u
```

```
$ cd jdk8u
```

```
$ bash get_source.sh
```

- Configure build

```
$ ./configure --with-debug-level=slowdebug  
              --with-boot-jdk=/usr/lib/jvm/java-1.7.0
```

– you will need to install a lot of packages!

- Make the jvm images

```
$ make images
```

C1 CFG before code generation

CFG before code generation

B17 [0, 0] -> B18 sux: B18

B18 (S) [0, 0] -> B0 dom B17 sux: B0 pred:
B17

B0 (SV) [0, 2] -> B2 B1 dom B18 sux: B2 B1
pred: B18

B2 (V) [7, 9] -> B4 B3 dom B0 sux: B4 B3
pred: B0

B4 (V) [14, 2] -> B8 B7 dom B2 sux: B8 B7
pred: B2

B7 (V) [5, 20] -> B5 dom B4 sux: B5 pred: B4

B8 (V) [7, 9] -> B10 B9 dom B4 sux: B10 B9
pred: B4

B9 (V) [12, 20] -> B5 dom B8 sux: B5 pred:
B8

B10 (V) [14, 20] -> B5 dom B8 sux: B5 pred:
B8

B5 (V) [20, 2] -> B14 B13 dom B4 sux: B14
B13 pred: B7 B9 B10Stack:

0 i32 [i6 i6 i41]

B14 (V) [7, 9] -> B16 B15 dom B5 sux: B16 B15
pred: B5

B15 (V) [12, 26] -> B11 dom B14 sux: B11 pred:
B14

B16 (V) [14, 26] -> B11 dom B14 sux: B11 pred:
B14

B13 (V) [5, 26] -> B11 dom B5 sux: B11 pred:
B5

B11 (V) [26, 27] dom B5 pred: B13 B15
B16Stack:

0 i32

1 i61 [i6 i6 i70]

B3 (V) [12, 13] dom B2 pred: B2

B1 (V) [5, 6] dom B0 pred: B0

C1 IR B10: fib(x - 1) + fib(x-2)

IR before code generation

. . .
B10 (V) [14, 20] -> B5 dom B8 sux: B5 pred: B8

empty stack

inlining depth 1

<u>bci</u>	<u>use</u>	<u>tid</u>	<u>instr</u>
16	1	i34	i15 - i6
. 17	0	v35	profile NULL Fib.fib)
. 17	1	i36	invokestatic(i34) Fib.fib(I)I
22	1	i38	i15 - i10
. 23	0	v39	profile NULL Fib.fib)
. 23	1	i40	invokestatic(i38) Fib.fib(I)I stack [0:i36]
. 26	1	i41	i36 + i40
. 20	0	42	goto B5 stack [0:i41]
. . .			

C1 Assembly B10

```
;; block B10 [14, 20]
```

```
0x7f7db4dfc3bf: mov     %esi,0x44(%rsp)
0x7f7db4dfc3c3: mov     $0x7f7db21a8670,%rbx
; {metadata(method data for {method}
{0x7f7db21a83a8} 'fib' '(I)I' in 'Fib')}}
0x7f7db4dfc3cd: addq   $0x1,0x170(%rbx)
0x7f7db4dfc3d5: mov     %rdi,%rbx
0x7f7db4dfc3d8: dec     %ebx
0x7f7db4dfc3da: mov     %rbx,%rsi
; *invokestatic fib
; - Fib::fib@17 (line 8)
; - Fib::fib@17 (line 8)

0x7f7db4dfc3dd: mov     %edi,0x40(%rsp)
0x7f7db4dfc3e7: callq  0x7f7db4cd5300
; OopMap{off=428}
; *invokestatic fib
; - Fib::fib@17 (line 8)
; - Fib::fib@17 (line 8)
; {static_call}
0x7f7db4dfc3ec: mov     $0x7f7db21a8670,%rsi
; {metadata(method data for {method}
{0x7f7db21a83a8} 'fib' '(I)I' in 'Fib')}}

0x7f7db4dfc3f6: addq   $0x1,0x180(%rsi)
0x7f7db4dfc3fe: mov     0x40(%rsp),%edi
0x7f7db4dfc402: sub     $0x2,%edi
0x7f7db4dfc405: mov     %rdi,%rsi
; *invokestatic fib
; - Fib::fib@23 (line 8)
; - Fib::fib@17 (line 8)

0x7f7db4dfc408: mov     %eax,0x48(%rsp)
0x7f7db4dfc40f: callq  0x7f7db4cd5300
; OopMap{off=468}
; *invokestatic fib
; - Fib::fib@23 (line 8)
; - Fib::fib@17 (line 8)
; {static_call}
0x7f7db4dfc414: mov     0x48(%rsp),%esi
0x7f7db4dfc418: add     %eax,%esi
0x7f7db4dfc41a: mov     %rsi,%rdi
; *iload_0
; - Fib::fib@20 (line 8)
0x7f7db4dfc41d: mov     0x44(%rsp),%esi
```

C1 Performance

```
$ time java Fib 42  
Fib of 42 = 267914296
```

```
real 0m1.059s
```

```
user 0m0.944s
```

```
sys 0m0.131s
```


What Lies Beneath?

- Bytecode
- Template Interpreter
- C1 JIT Compiler
- **C2 JIT Compiler**
- Special Tricks
- Questions

C2 JIT Compiler

- Server compiler
 - for long running server applications
- Optimising Compiler Specially for JITting
 - see Global Code Motion Global Value Numbering, Click, PLDI95
 - (and especially the code ;-)

```
$ java -XX:+PrintCompilation -XX:+PrintIdeal -XX:  
+PrintOptoAssembly -XX:+PrintAssembly -XX:  
+TieredCompilation -XX:CompileOnly=Fib.fib Fib 24
```

- n.b. these options are debug build only
PrintOptoAssembly PrintIdeal

C2 JIT Compiler

Highly efficient in time and space

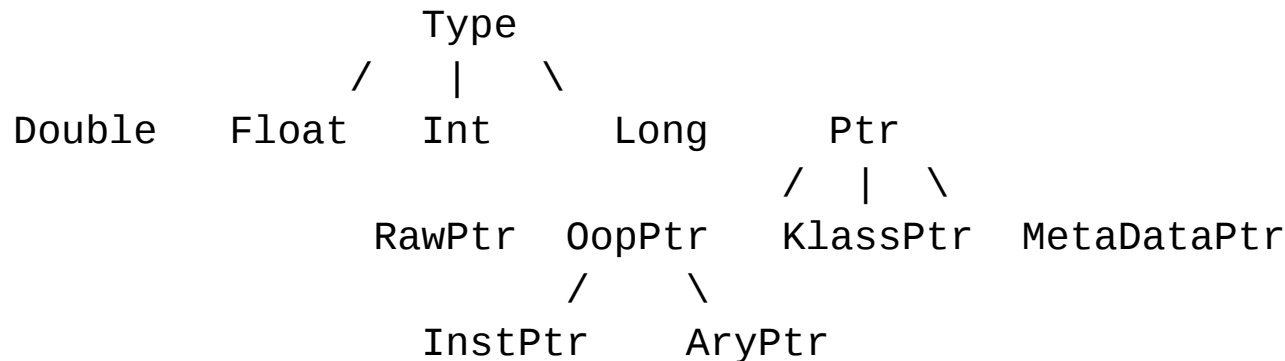
- Sea of Nodes for IR
 - 1 graph models data, control, memory, io dependencies
 - Dataflow graph equivalent to SSA form
 - Dependencies alone impose node order
- Bytecode to HIR – abstract computation graph
 - Many Ideal graph transforms in fixed phases
 - $O(N \log(N))$ per phase for N bytecodes (including inlined)
- HIR to MIR – transform to MachineNodes
 - ADLC : pattern based translation
- Deopt + Recompile adaptive compilation

C2 JIT Nodes

- Hierarchy of Node classes
 - All have id (int), ins & outs (int[]), opcode, type
 - Most ins/outs are typed dataflows
 - AddNode, IAddNode, ConstNode, MemNode
 - n.b. may be multiple outs (multiple uses of data)
 - A few nodes have Ctrl/Mem/AbIO type for some ins
 - IfNode, JmpNode, CatchNode, RegionNode, PhiNode
 - MemNode, LoadNode, MergeMemNode
- Notable methods
 - Ideal() - normal form e.g. (IAdd IConst(0), X) ==> X
 - Value() - node value's type

C2 JIT Types

- Class Type models Node value types
 - Type class hierarchy



- TypeFlow analysis => weaken/strengthen type
- occurrences model unconstrained (Bottom), constant, range or overconstrained (Top) values
 - see Cliff Click's blog for full details (including bug!)

PrintCompilation

```
$ time java -XX:+PrintCompilation -XX:CompileOnly=Fib.fib Fib 48
  454    1    3    Fib::fib (26 bytes)
  462    2    4    Fib::fib (26 bytes)
  468    1    3    Fib::fib (26 bytes)    made not entrant
Fib of 48 = 512559680
```

```
real    0m14.311s
user    0m14.251s
sys     0m0.108s
```

```
$ time java -XX:+PrintCompilation -XX:-TieredCompilation
-XX:CompileOnly=Fib.fib Fib 48
  539    1          Fib::fib (26 bytes)
Fib of 48 = 512559680
```

```
real    0m14.394s
user    0m14.311s
sys     0m0.114s
```

C2 PrintIdeal

```
36 IfTrue    === 35  [[ 56 ]] #1 !jvms: Fib::fib @ bci:7
54 CmpI      === _ 43 22  [[ 55 ]] !jvms: Fib::fib @ bci:2 Fib::fib @
bci:15
55 Bool      === _ 54  [[ 56 ]] [ne] !jvms: Fib::fib @ bci:2 Fib::fib @
bci:15
63 CmpI      === _ 43 32  [[ 64 ]] !jvms: Fib::fib @ bci:7 Fib::fib @
bci:15
56 If        === 36 55  [[ 57 58 ]] P=0.809012, C=224255.000000 !jvms:
Fib::fib @ bci:2 Fib::fib @ bci:15
32 ConI      === 0  [[ 33 63 136 ]] #int:2
127 CmpI     === _ 72 22  [[ 128 ]] !jvms: Fib::fib @ bci:2 Fib::fib
@ bci:21
136 CmpI     === _ 72 32  [[ 137 ]] !jvms: Fib::fib @ bci:7 Fib::fib
@ bci:21
64 Bool      === _ 63  [[ 65 ]] [ne] !jvms: Fib::fib @ bci:7 Fib::fib @
bci:15
57 IfTrue    === 56  [[ 65 ]] #1 !jvms: Fib::fib @ bci:2 Fib::fib @ bci:15
58 IfFalse   === 56  [[ 62 ]] #0 !jvms: Fib::fib @ bci:2 Fib::fib @ bci:15
67 IfFalse   === 65  [[ 62 ]] #0 !jvms: Fib::fib @ bci:7 Fib::fib @ bci:15
. . .
```

C2 PrintIdeal

```
62 Region === 62 67 58 [[ 62 46 ]] !jvms: Fib::fib @ bci:10
Fib::fib @ bci:15
. . .
46 Region === 46 100 62 [[ 46 50 47 48 129 ]] !jvms: Fib::fib @
bci:15
. . .
50 Phi === 46 111 22 [[ 164 186 146 ]] #int !jvms: Fib::fib @
bci:15
48 Phi === 46 95 7 [[ 120 146 ]] #memory Memory: @BotPTR *+bot,
idx=Bot; !jvms: Fib::fib @ bci:15
47 Phi === 46 94 6 [[ 146 119 ]] #abIO !jvms: Fib::fib @ bci:15
. . .
129 If === 46 128 [[ 130 131 ]] P=0.809011, C=247266.000000 !
jvms: Fib::fib @ bci:2 Fib::fib @ bci:21
. . .
146 CallStaticJava === 139 47 48 8 1 ( 90 1 50 72 ) [[ 147
148 149 151 ]] # Static Fib::fib int ( int ) Fib::fib @ bci:15 Fib::fib
@ bci:21 !jvms: Fib::fib @ bci:15 Fib::fib @ bci:21
. . .
188 Rethrow === 107 108 109 8 9 exception 110 [[ 0 ]]
187 Return === 13 14 15 8 9 returns 17 [[ 0 ]]
0 Root === 0 187 188 [[ 0 1 3 22 32 42 73 91 163 ]] inner
```


C2 PrintOptoAssembly

```
{method}
- this oop:          0x7f1087c003a8
- method holder:    'Fib'
- constants:        0x7f1087c00070 constant pool [56] {0x7f1087c00070}
for 'Fib' cache=0x7f1087c004e8
- access:           0x81000008  static
- name:             'fib'
- signature:        '(I)I'
- max stack:        4
- max locals:       1
- size of params:   1
- method size:      12
- vtable index:     -2
- i2i entry:        0x7f108d01eb00
- adapters:         AHE@0x7f10900eb270: 0xa0000000 i2c: 0x7f108d148420
c2i: 0x7f108d148559 c2iUV: 0x7f108d14852c
- compiled entry    0x7f108d148559
- code size:        26
- code start:       0x7f1087c00388
- code end (excl): 0x7f1087c003a2
- method data:      0x7f1087c00670
- checked ex length: 0
- linenumbers start: 0x7f1087c003a2
- localvar length: 0
```

C2 PrintOptoAssembly

```
#
# int ( int )
#
#r018 rsi    : parm 0: int
# -- Old rsp -- Framesize: 48 --
#r191 rsp+44: in_preserve
#r190 rsp+40: return address
#r189 rsp+36: in_preserve
#r188 rsp+32: saved fp register
#r187 rsp+28: pad2, stack alignment
#r186 rsp+24: pad2, stack alignment
#r185 rsp+20: Fixed slot 1
#r184 rsp+16: Fixed slot 0
#r195 rsp+12: spill
#r194 rsp+ 8: spill
#r193 rsp+ 4: spill
#r192 rsp+ 0: spill
#
abababab    N1: #      B1 <- B12 B22  Freq: 1
abababab
```

C2 PrintOptoAssembly

```
000 B1: # B3 B2 <- BLOCK HEAD IS JUNK   Freq: 1
000   # stack bang (216 bytes)
      pushq   rbp # Save rbp
      subq    rsp, #32   # Create frame

00c   movl     [rsp + #0], RSI # spill
00f   cmpl     RSI, #1
012   je,s    B3   P=0.191038 C=7878.000000
012
014 B2: # B13 B3 <- B1   Freq: 0.808962
014   cmpl     RSI, #2
017   jne,s   B13  P=0.617919 C=6373.000000
017
019 B3: # B12 <- B2 B1  Freq: 0.500127
019   movl     RAX, #1 # int
01e   jmp,s   B12
01e
020 B4: # B11 <- B7 B6  Freq: 0.249995
020   movl     RAX, #1 # int
025   jmp,s   B11
025
027 B5: # B6 <- B14 B13 Freq: 0.25
027   movl     R11, #1 # int
      . . .
```

C2 PrintOptoAssembly

```
. . .
037 B8: # B20 B9 <- B7 Freq: 0.249868
037   movl    [rsp + #8], R11 # spill
03c   movl    RSI, [rsp + #4] # spill
      nop    # 3 bytes pad for loops and calls
043   call,static Fib::fib
      # Fib::fib @ bci:15  L[0]=RBP
      # Fib::fib @ bci:21  L[0]=_ STK[0]=rsp + #8
      # OopMap{off=72}

048
048 B9: # B21 B10 <- B8 Freq: 0.249863
      # Block is sole successor of call
048   movl    RBP, RAX    # spill
04a   movl    RSI, [rsp + #0] # spill
04d   addl    RSI, #-4    # int
      nop    # 3 bytes pad for loops and calls
053   call,static Fib::fib
      # Fib::fib @ bci:21  L[0]=_ STK[0]=RBP
      # Fib::fib @ bci:21  L[0]=_ STK[0]=rsp + #8
      # OopMap{off=88}

058
058 B10: #      B11 <- B9 Freq: 0.249858
      # Block is sole successor of call
058   addl    RAX, RBP    # int
05a   movl    R11, [rsp + #8] # spill
. . .
```

C2 PrintOptoAssembly

```
. . .
05f B11: # B12 <- B10 B4 Freq: 0.499853
05f addl RAX, R11 # int
062
062 B12: # N1 <- B11 B3 Freq: 0.99998
062 addq rsp, 32 # Destroy frame
    popq rbp
    testl rax, [rip + #offset_to_poll_page] # Safepoint: poll for GC

06d ret

. . .
0c9 B20: # B22 <- B8 Freq: 2.49975e-06
0c9 # exception oop is in rax; no code emitted
0c9 movq RSI, RAX # spill
0cc jmp,s B22
0cc
0ce B21: # B22 <- B9 Freq: 2.49863e-06
0ce # exception oop is in rax; no code emitted
0ce movq RSI, RAX # spill
0ce
0d1 B22: # N1 <- B18 B19 B20 B21 Freq: 9.99472e-06
0d1 addq rsp, 32 # Destroy frame
    popq rbp

0d6 jmp rethrow_stub
```

C2 PrintAssembly

Decoding compiled method 0x00007f4e791fec90:

Code:

[Entry Point]

[Verified Entry Point]

[Constants]

{method} {0x00007f4e73c003a8} 'fib' '(I)I' in 'Fib'

parm0: rsi = int

[sp+0x30] (sp of caller)

;; N1: # B1 <- B12 B22 Freq: 1

;; B1: # B3 B2 <- BLOCK HEAD IS JUNK Freq: 1

0x00007f4e791fee00: mov %eax, -0x16000(%rsp)

0x00007f4e791fee07: push %rbp

0x00007f4e791fee08: sub \$0x20,%rsp ;*synchronization entry
; - Fib::fib@-1 (line 3)

0x00007f4e791fee0c: mov %esi, (%rsp)

0x00007f4e791fee0f: cmp \$0x1,%esi

0x00007f4e791fee12: je 0x00007f4e791fee19 ;*if_icmpeq
; - Fib::fib@2 (line 3)

. . .

C2 PrintAssembly

```
;; B8: # B20 B9 <- B7 Freq: 0.249975
```

```
0x00007f4e791fee37: mov    %r11d,0x8(%rsp)
0x00007f4e791fee3c: mov    0x4(%rsp),%esi
0x00007f4e791fee40: nop
0x00007f4e791fee41: nop
0x00007f4e791fee42: nop
0x00007f4e791fee43: callq 0x00007f4e79106300 ; OopMap{off=72}
                                ;*invokestatic fib
                                ; - Fib::fib@15 (line 5)
                                ; - Fib::fib@21 (line 5)
                                ; {static_call}
```

```
;; B9: # B21 B10 <- B8 Freq: 0.24997
```

```
0x00007f4e791fee48: mov    %eax,%ebp
0x00007f4e791fee4a: mov    (%rsp),%esi
0x00007f4e791fee4d: add    $0xffffffffffffffffc,%esi ;*isub
                                ; - Fib::fib@20 (line 5)
                                ; - Fib::fib@21 (line 5)
```

```
0x00007f4e791fee50: nop
0x00007f4e791fee51: nop
0x00007f4e791fee52: nop
0x00007f4e791fee53: callq 0x00007f4e79106300 ; OopMap{off=88}
                                ;*invokestatic fib
```

```
. . .
```

C2 PrintAssembly

```
;; B22: # N1 <- B18 B19 B20 B21 Freq: 9.999e-06
```

```
0x00007f4e791feed1: add    $0x20,%rsp
0x00007f4e791feed5: pop    %rbp
0x00007f4e791feed6: jmpq   0x00007f4e791faea0 ; {runtime_call}
0x00007f4e791feedb: hlt
0x00007f4e791feedc: hlt
0x00007f4e791feedd: hlt
0x00007f4e791feede: hlt
0x00007f4e791feedf: hlt
```

[Stub Code]

```
0x00007f4e791feee0: mov    $0x0,%rbx ; {no_reloc}
0x00007f4e791feeea: jmpq   0x00007f4e791feeea ; {runtime_call}
0x00007f4e791feeeef: mov    $0x0,%rbx ; {static_stub}
0x00007f4e791feef9: jmpq   0x00007f4e791feef9 ; {runtime_call}
0x00007f4e791feefe: mov    $0x0,%rbx ; {static_stub}
0x00007f4e791fef08: jmpq   0x00007f4e791fef08 ; {runtime_call}
0x00007f4e791fef0d: mov    $0x0,%rbx ; {static_stub}
0x00007f4e791fef17: jmpq   0x00007f4e791fef17 ; {runtime_call}
```

[Exception Handler]

```
0x00007f4e791fef1c: jmpq   0x00007f4e79004ee0 ; {runtime_call}
```

[Deopt Handler Code]

```
0x00007f4e791fef21: callq  0x00007f4e791fef26
0x00007f4e791fef26: subq   $0x5,(%rsp)
0x00007f4e791fef2b: jmpq   0x00007f4e791072e0 ; {runtime_call}
```

```
. . .
```


What Lies Beneath?

- Bytecode
- Template Interpreter
- C1 JIT Compiler
- C2 JIT Compiler
- **Special Tricks**
- Questions

On Stack Replacement

- Compile and jump into a hot method
 - e.g. a big loop

```
for (idx = 0; idx < limit; idx++) {  
    // do something complex lots of times  
}
```

```
while (condition) {  
    // do something complex lots of times  
}
```

- Compile from OSR point
 - with current locals as inputs (e.g. `idx`, `limit`)
- Swap interpreted frame for compiled frame!

Deoptimisation

- Jump out into the interpreter and recompile
 - e.g. a cold path is entered
 - C1/C2 generate deopt trap on cold paths
 - saves on code size and compile time
 - n.b. -Xcomp with -Tiered == deopt-a-lot (all paths are cold)
 - e.g. access to not yet-loaded class
 - C1/C2 generate deopt trap for this case
 - bet on never if it was not loaded by interpreter
- Swap compiled frame for interpreted frame!
 - several frames if deopt is for inlined code!
 - locals need to be restored to local area(s)!

Safepoint Checks

- Do we need to stop the world?
 - for a GC/GC phase? or other VM housekeeping?
 - e.g. clear out deoptimised methods
- Checks happen at strategic points
 - after callouts to VM stubs/helpers
 - before return
 - at loop back edges

Safepoint Checks

```
062  B12: #      N1 <- B11 B3  Freq: 0.99998
062  addq  rsp, 32 # Destroy frame
      popq  rbp
      testl rax, [rip + #offset_to_poll_page] # Safepoint: poll for GC

06d  ret
```

- Check by reading well known address
 - Poll page is mprotected -r when Safepoint needed
 - Signal handler identifies SEGV address
 - returns from signal into stub code
 - Stub code pushes frame for callout to VM
 - return address of frame is instruction after testl

What Lies Beneath?

- Bytecode
- Template Interpreter
- C1 JIT Compiler
- C2 JIT Compiler
- Special Tricks
- **Questions**