JIT Compilation with Graal

Journey of a Java Program

Swapnil Gaikwad, Alan Hayward
Arm, Manchester
@SwapnilGaikwad

Agenda

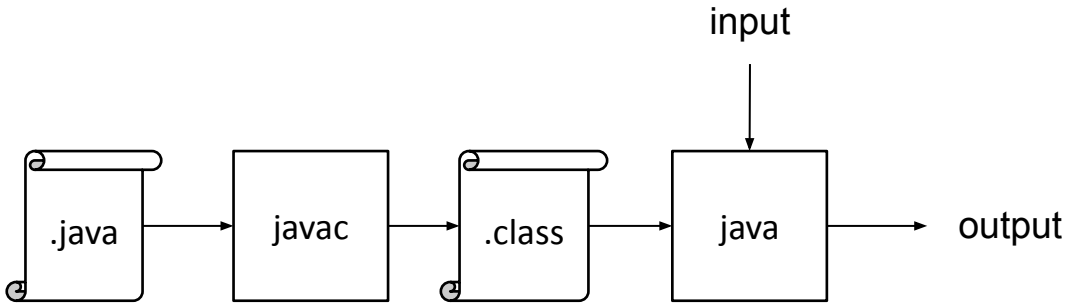
- Lifecycle of a Java program
- Interpreter
- JIT Compilation
  - Graal Compiler
- Compiler Optimisations
- Summary

https://github.com/SwapnilGaikwad/jmanc-graal

Lifecycle of a Java Program

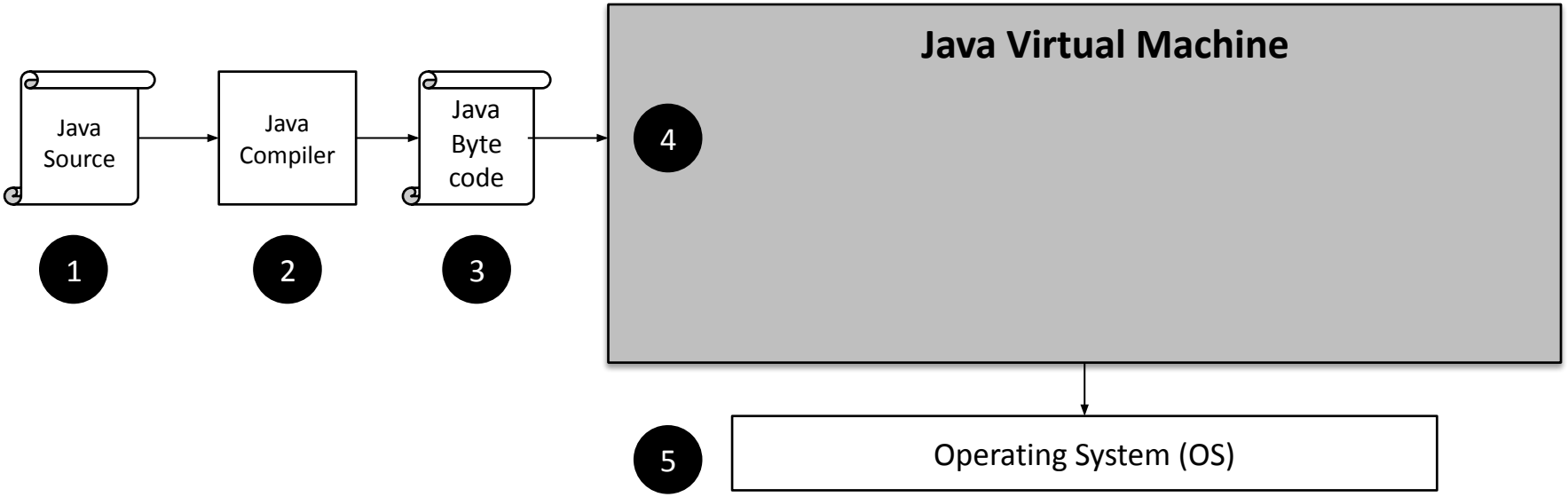
```
$ javac MyTest.java
$ java MyTest
```

Lifecycle of a Java Program



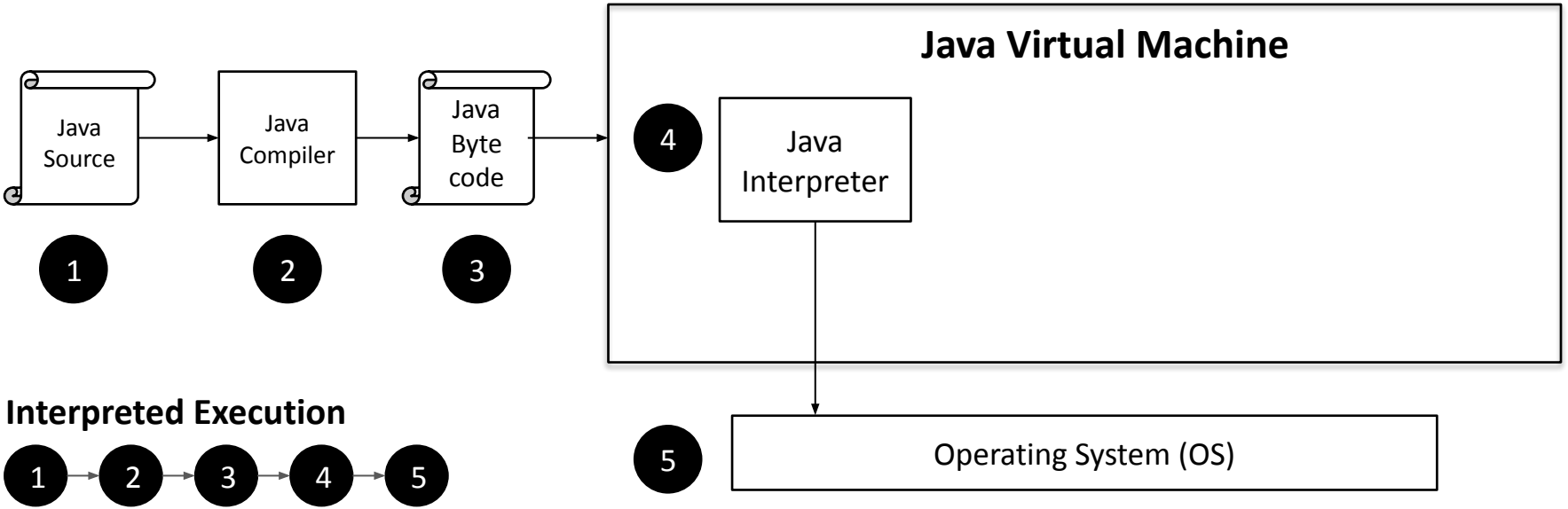

```
$ javac MyTest.java
$ java MyTest
```

Java Compilation Approach

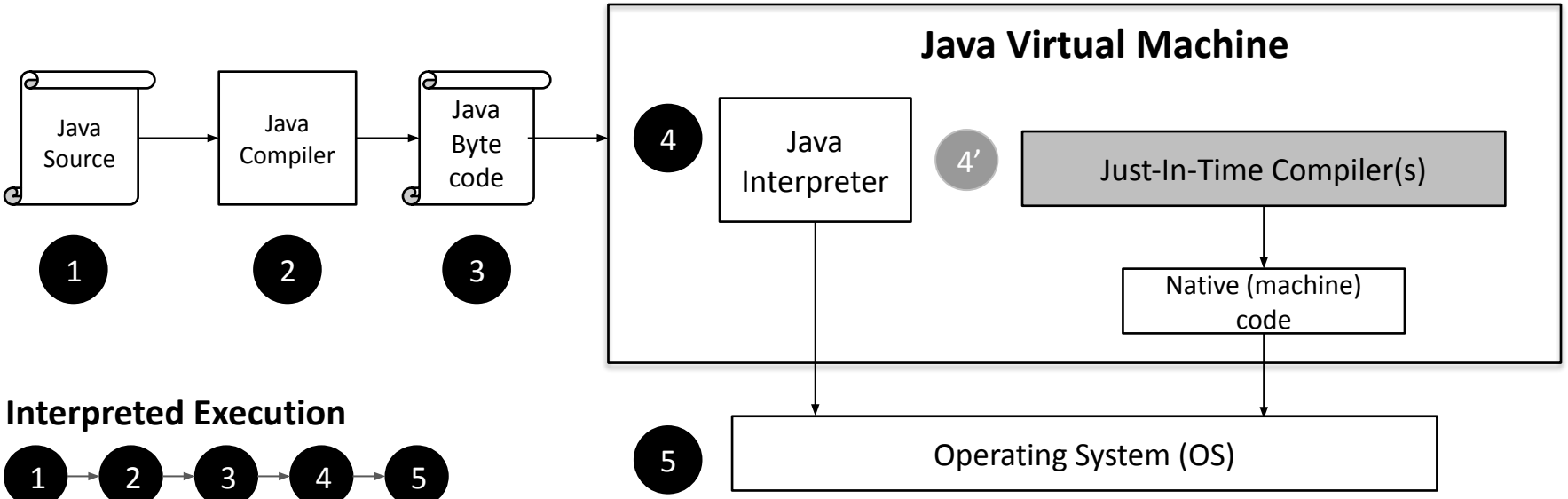


- Java uses hybrid approach of both compiler and interpreter

Java JVM Overview



Java JVM Overview



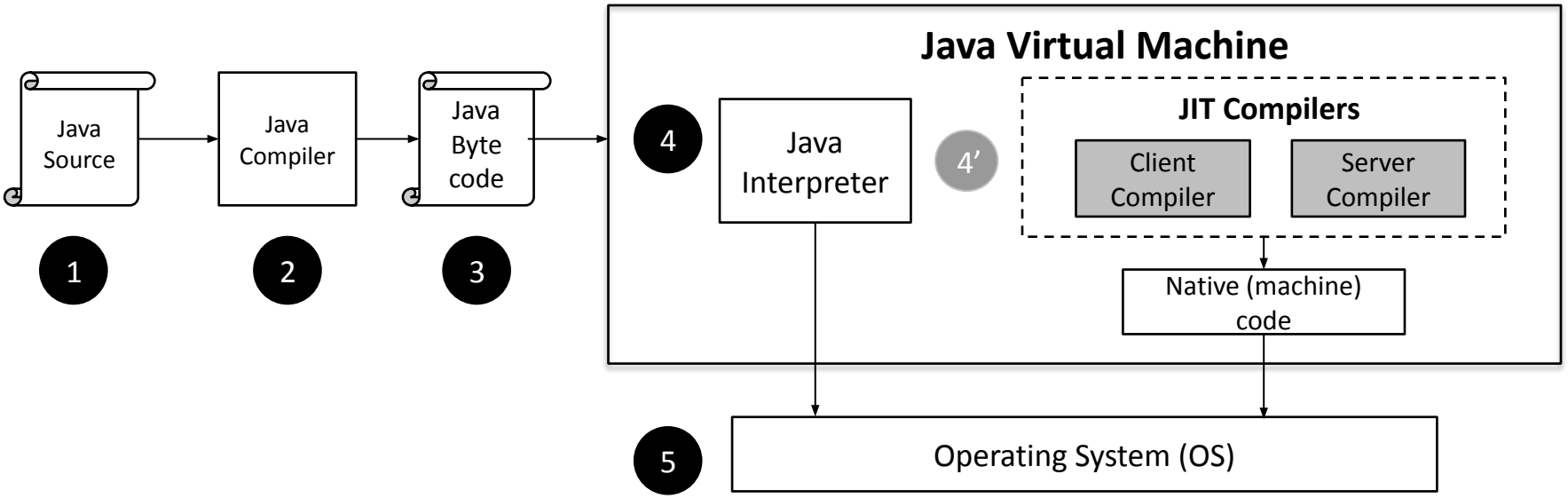
Interpreted Execution

1 → 2 → 3 → 4 → 5

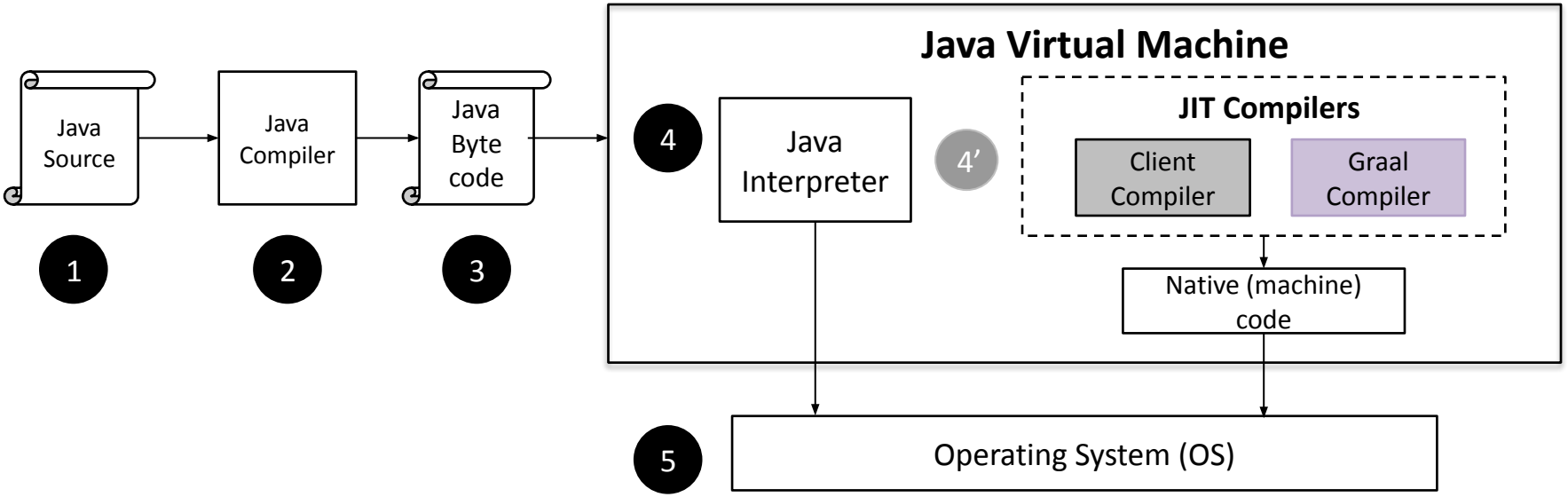
JIT Compiled Execution

1 → 2 → 3 → 4 → 4' → 5

HotSpot JVM Overview



Graal VM Overview



A Simple Java Program

```
public class MyTest {
    public static int myInc(int a) {
        return a + 1;
    }

    public static void main(String[] args) {
        MyTest.myInc(5);
    }
}
```

Interpreter: Bytecodes

```
...

public static int myInc(int);
  Code:
     0: iload_0
     1: iconst_1
     2: iadd
     3: ireturn

...
```

```
$ javap -c MyTest.class
```

Interpreter: Bytecodes

```
public static int myInc(int);
  Code:
     0: iload_0
     1: iconst_1
     2: iadd
     3: ireturn
```

```
Call myInt(5)
```

Interpreter: Bytecodes

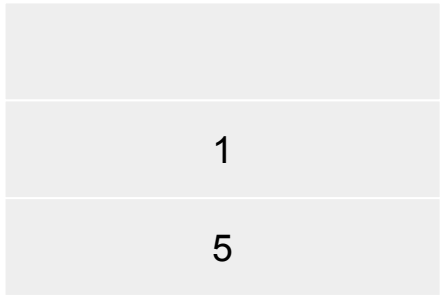
```
public static int myInc(int);
  Code:
     0: iload_0
     1: iconst_1
     2: iadd
     3: ireturn
```

```
Load the first local variable
```

Interpreter: Bytecodes

```
public static int myInc(int);
  Code:
     0: iload_0
     1: iconst_1
     2: iadd
     3: ireturn
```

Push constant 1



Interpreter: Bytecodes

```
public static int myInc(int);
  Code:
     0: iload_0
     1: iconst_1
     2: iadd
     3: ireturn
```

```
Pop top two values, add them and
push result on the stack
```

Interpreter: Bytecodes

```
public static int myInc(int);
  Code:
     0: iload_0
     1: iconst_1
     2: iadd
     3: ireturn
```

```
Return int from method
```

Performance

- Interpreted version
- JIT Compiled with C1 compiler
- JIT Compiled with Graal compiler

Benchmark	Mode	Cnt	Score	Error	Units
MyBenchmark.interpreter	avgt	3	41469.329 ±	1390.876	ns/op
MyBenchmark.tier1Only	avgt	3	614.950 ±	16.311	ns/op
MyBenchmark.tierAll	avgt	3	389.108 ±	8.780	ns/op

Interpreter: Assembly for a bytecode

```
public static int myInc(int);
  Code:
     0: iload_0
     1: iconst_1
     2: iadd
     3: ireturn
```

```
----------------------------------------------------------------------
iload_0  26 iload_0  [0x0000ffffd2ea6200, 0x0000ffffd2ea6240]  64 bytes

  0x0000ffffd2ea6200: str       x0, [x20,#-8]!
  0x0000ffffd2ea6204: b 0x0000ffffd2ea6228
  0x0000ffffd2ea6208: str       s0, [x20,#-8]!
  0x0000ffffd2ea620c: b 0x0000ffffd2ea6228
  0x0000ffffd2ea6210: str       d0, [x20,#-16]!
  0x0000ffffd2ea6214: b 0x0000ffffd2ea6228
  0x0000ffffd2ea6218: str       xzr, [x20,#-8]!
  0x0000ffffd2ea621c: str       x0, [x20,#-8]!
  0x0000ffffd2ea6220: b 0x0000ffffd2ea6228
  0x0000ffffd2ea6224: str       x0, [x20,#-8]!
  0x0000ffffd2ea6228: ldr       x0, [x24]
  0x0000ffffd2ea622c: ldrb      w8, [x22,#1]!
  0x0000ffffd2ea6230: add       w9, w8, #0x400
  0x0000ffffd2ea6234: ldr       x9, [x21,w9,uxtw #3]
  0x0000ffffd2ea6238: br        x9
  0x0000ffffd2ea623c: nop
```

```
$ java -XX:+UnlockDiagnosticVMOptions -XX:+PrintInterpreter MyTest
```

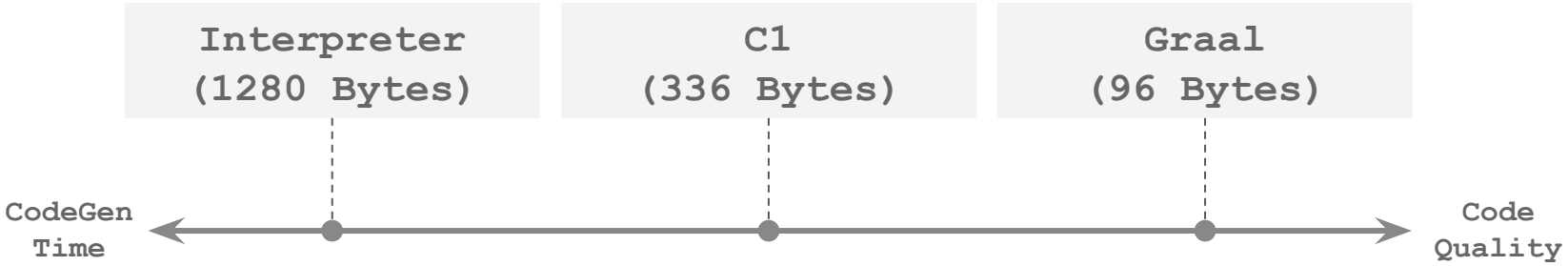
Performance

Benchmark	Mode	Cnt	Score	Error	Units
MyBenchmark.interpreter	avgt	3	41469.329 ±	1390.876	ns/op
MyBenchmark.tier1Only	avgt	3	614.950 ±	16.311	ns/op
MyBenchmark.tierAll	avgt	3	389.108 ±	8.780	ns/op

Interpreter	C1	Graal
(1280 Bytes)	(336 Bytes)	(96 Bytes)

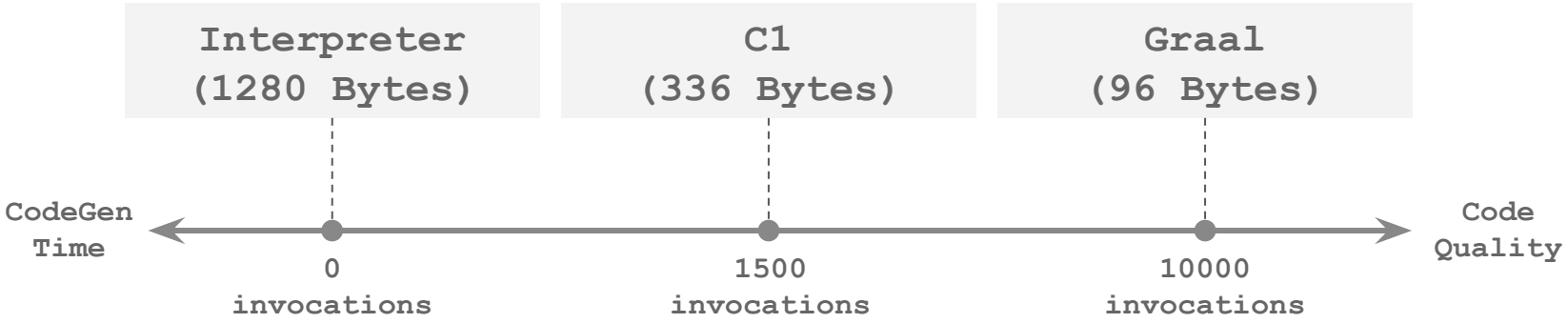
Performance

Benchmark	Mode	Cnt	Score	Error	Units
MyBenchmark.interpreter	avgt	3	41469.329 ±	1390.876	ns/op
MyBenchmark.tier1Only	avgt	3	614.950 ±	16.311	ns/op
MyBenchmark.tierAll	avgt	3	389.108 ±	8.780	ns/op

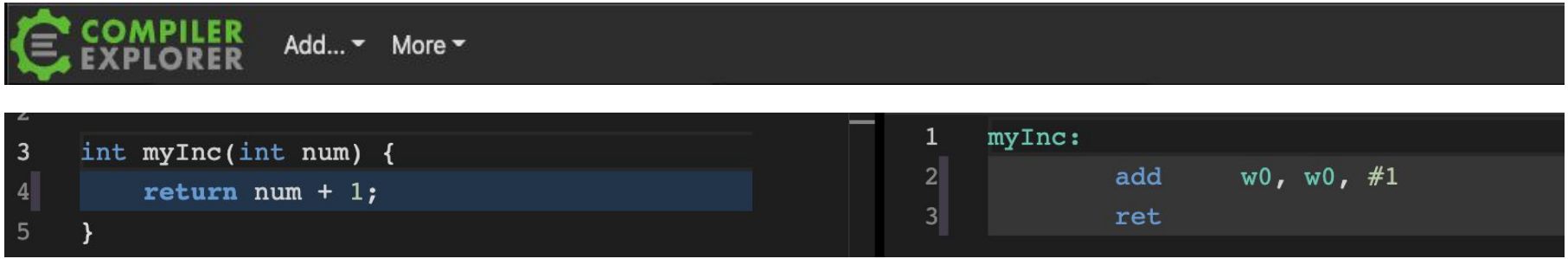


Performance

Benchmark	Mode	Cnt	Score	Error	Units
MyBenchmark.interpreter	avgt	3	41469.329 ±	1390.876	ns/op
MyBenchmark.tier1Only	avgt	3	614.950 ±	16.311	ns/op
MyBenchmark.tierAll	avgt	3	389.108 ±	8.780	ns/op

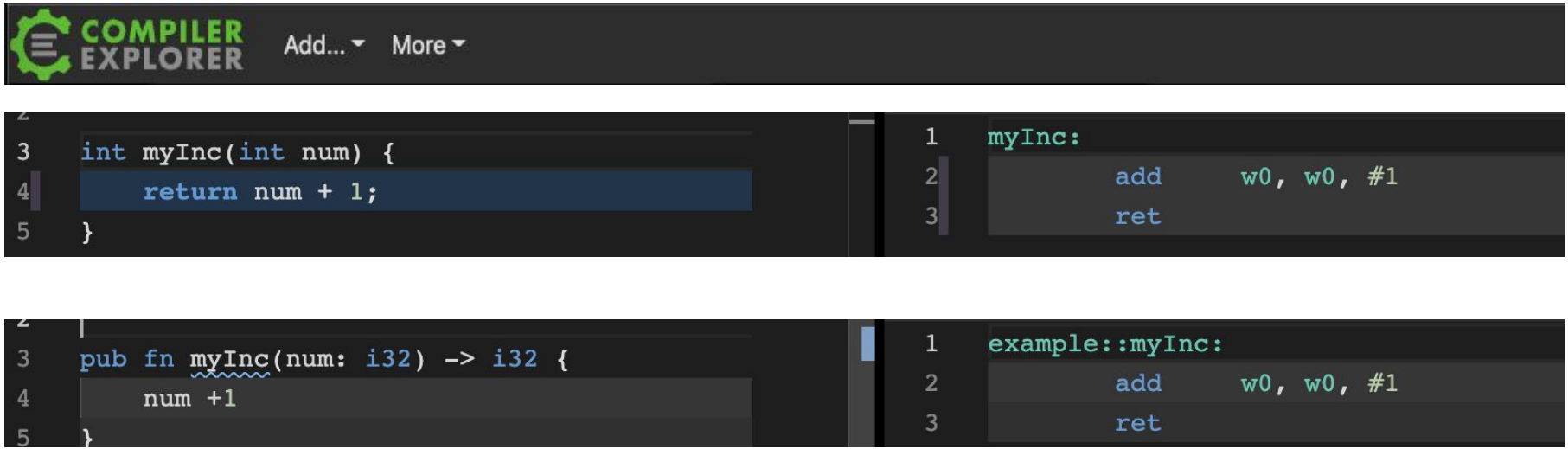


Assembly in C



Source: https://godbolt.org/

Assembly in C



Source: https://godbolt.org/

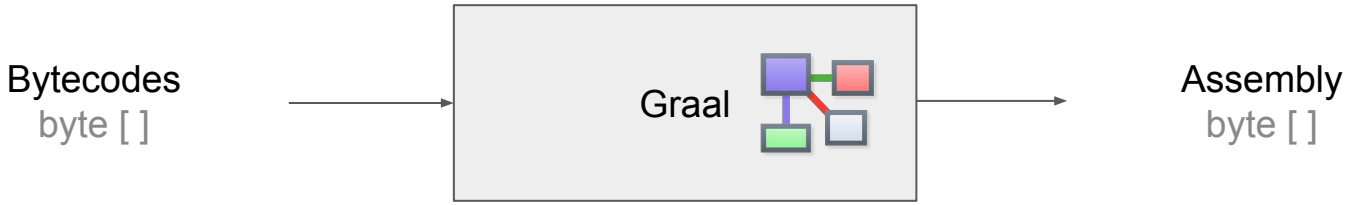
JIT Compilation

```
java -XX:+PrintCompilation MyTest
  72    1       3       java.lang.Object::<init> (1 bytes)
  73    2       3       java.lang.StringLatin1::hashCode (42 bytes)
  73    4       3       java.lang.String::hashCode (49 bytes)
  74    3       3       java.lang.String::isLatin1 (19 bytes)
  74    5       3       java.util.ImmutableCollections$SetN::probe (56 bytes)
  75    8       3       java.lang.String::coder (15 bytes)
  75    6       3       java.lang.Math::floorMod (10 bytes)
  75    7       3       java.lang.Math::floorDiv (22 bytes)
  77   10       3       java.lang.StringLatin1::equals (36 bytes)
  77   11       3       java.util.ImmutableCollections$SetN::hashCode (46 bytes)
  78    9       3       java.lang.String::equals (65 bytes)
  79   13       3       java.util.ImmutableCollections::emptySet (4 bytes)
  80   12       3       java.util.Objects::equals (23 bytes)
  80   14       3       java.util.Set::of (4 bytes)
  81   15       3       java.util.AbstractCollection::<init> (5 bytes)
  81   16       3       java.util.ImmutableCollections$AbstractImmutableCollection::<init> (5 bytes)
  81   17       3       java.util.Objects::requireNonNull (14 bytes)
  82   18       3       java.util.ImmutableCollections$AbstractImmutableSet::<init> (5 bytes)
  82   19       4       java.lang.Object::<init> (1 bytes)
  82   20       3       java.util.Set::of (66 bytes)
  83    1       3       java.lang.Object::<init> (1 bytes)   made not entrant
  83   21       1       java.lang.module.ModuleDescriptor::name (5 bytes)
  84   22       1       java.lang.module.ModuleReference::descriptor (5 bytes)
  89   23       3       java.lang.String::charAt (25 bytes)
  89   25       3       java.util.concurrent.ConcurrentHashMap::tabAt (22 bytes)
  90   27       3       jdk.internal.misc.Unsafe::getObjectAcquire (7 bytes)
  90   26       3       java.util.ImmutableCollections$SetN$SetNIterator::hasNext (13 bytes)
```

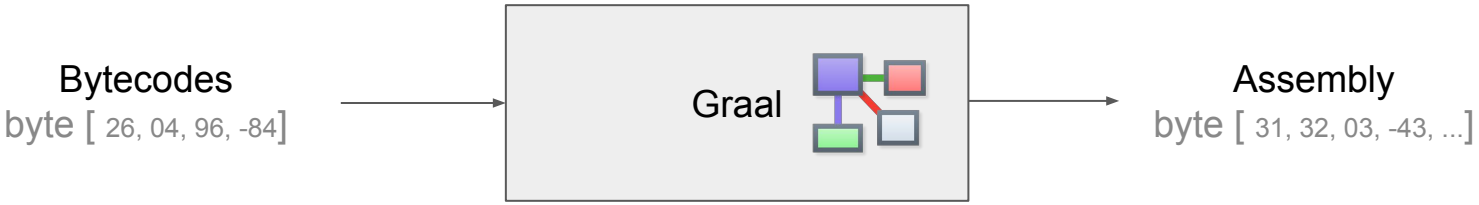
```
$ java -XX:+PrintCompilation MyTest
```

https://www.graalvm.org/

Graal Compiler

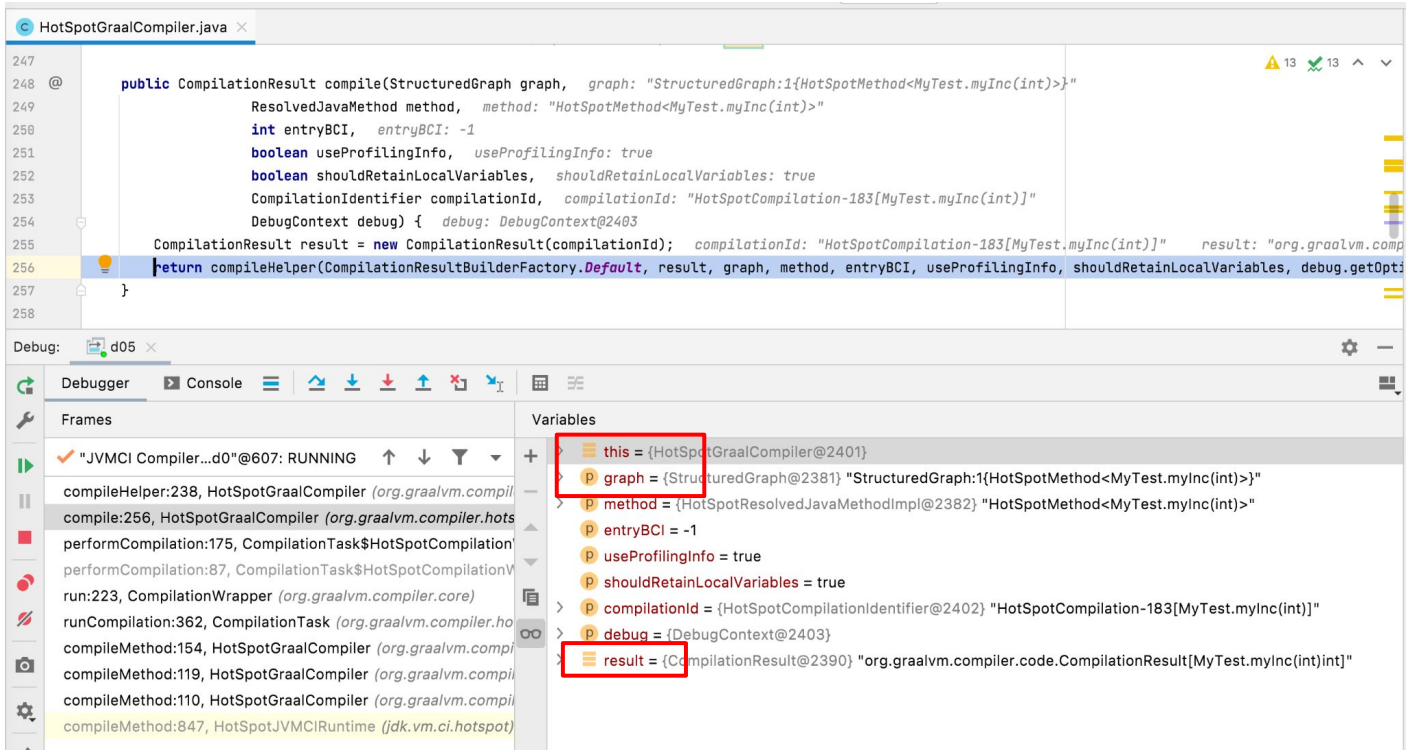


Graal Compiler



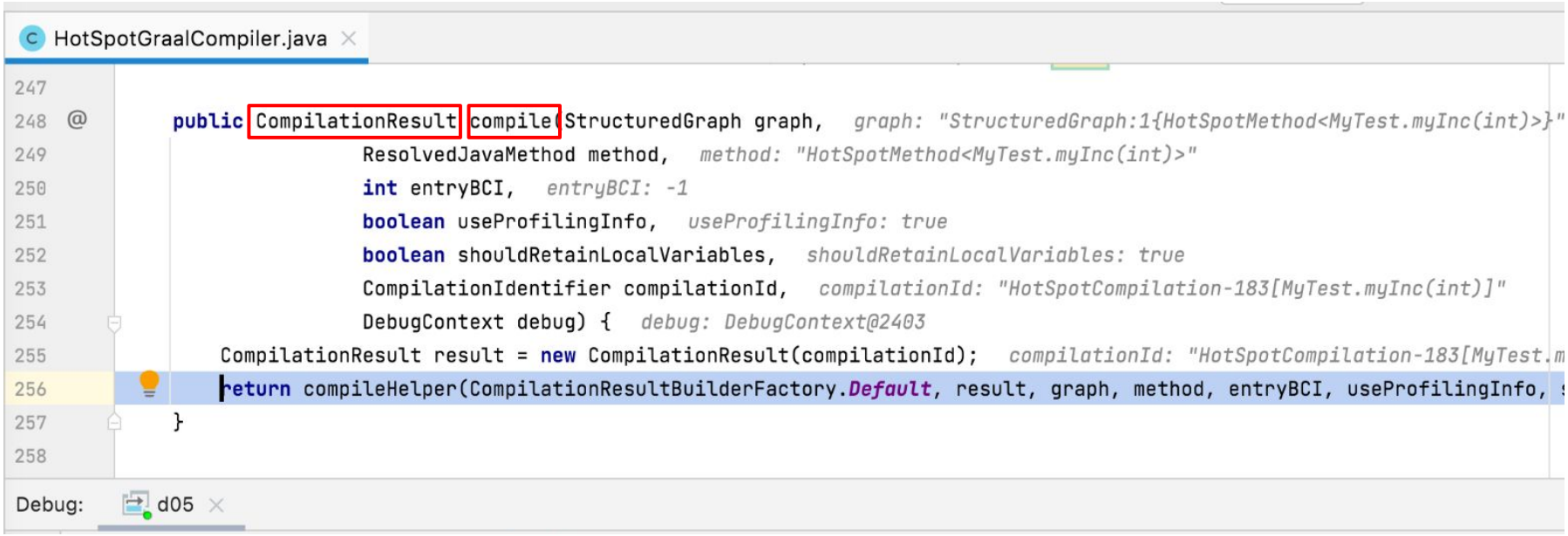

```
class Graal {
   byte[ ] compile (byte method[ ]) {
     ...
   }
}
```

Graal Compiler



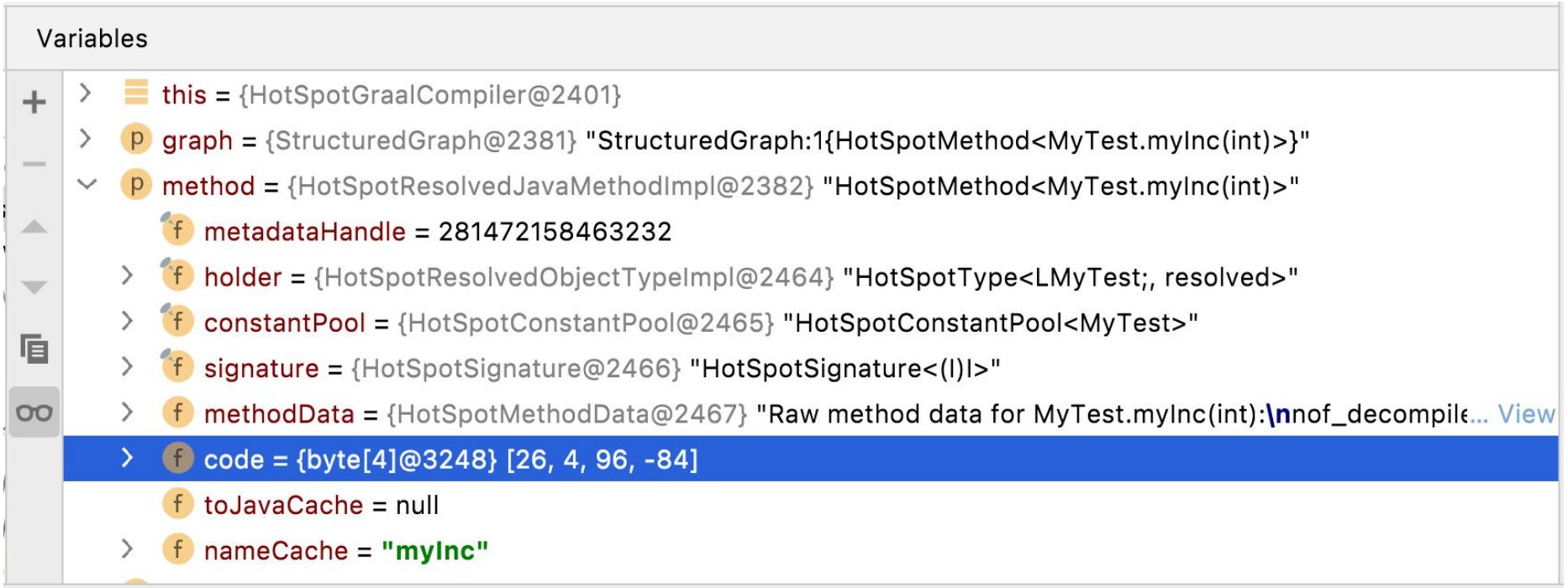

```
mx --dbg=*:9090 vm -XX:-BackgroundCompilation -XX:+UnlockDiagnosticVMOptions -XX:CompileCommand="compileonly, MyTest.myInc" -XX:CompileCommand="print, MyTest.myInc"
MyTest
```

Graal Compiler

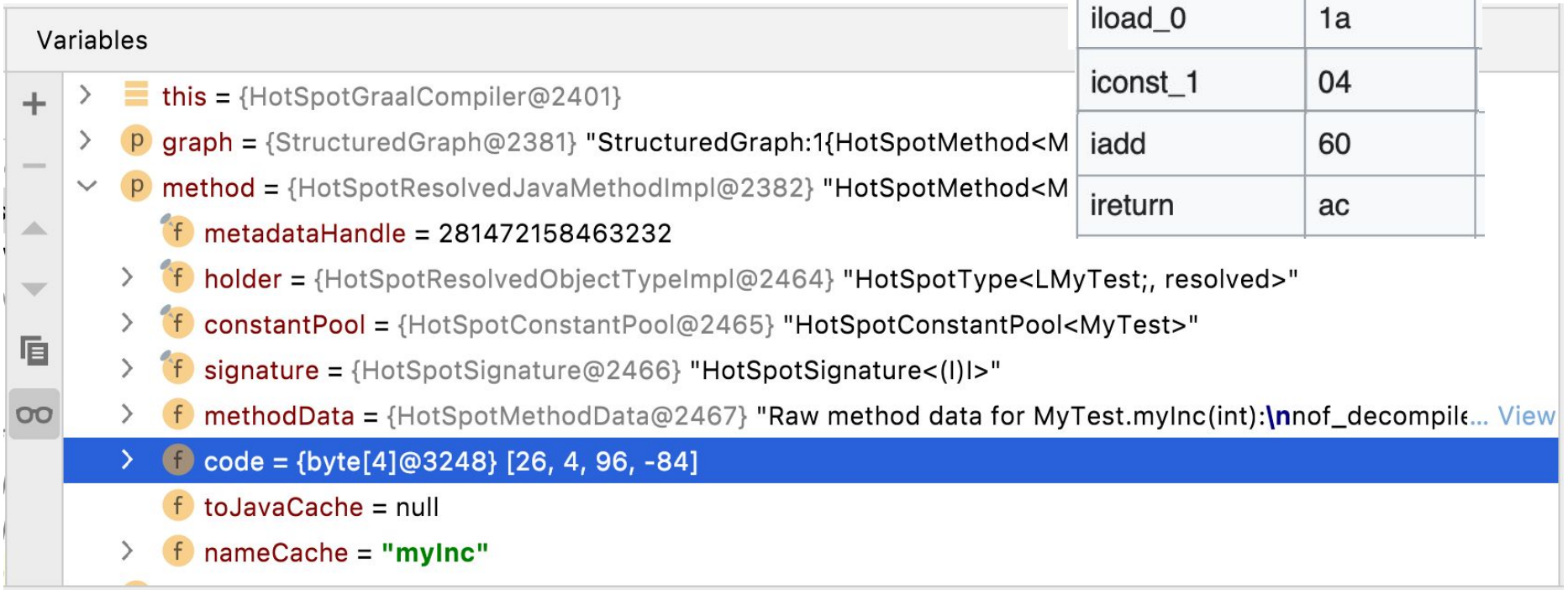


Graal Compiler

Graal Compiler



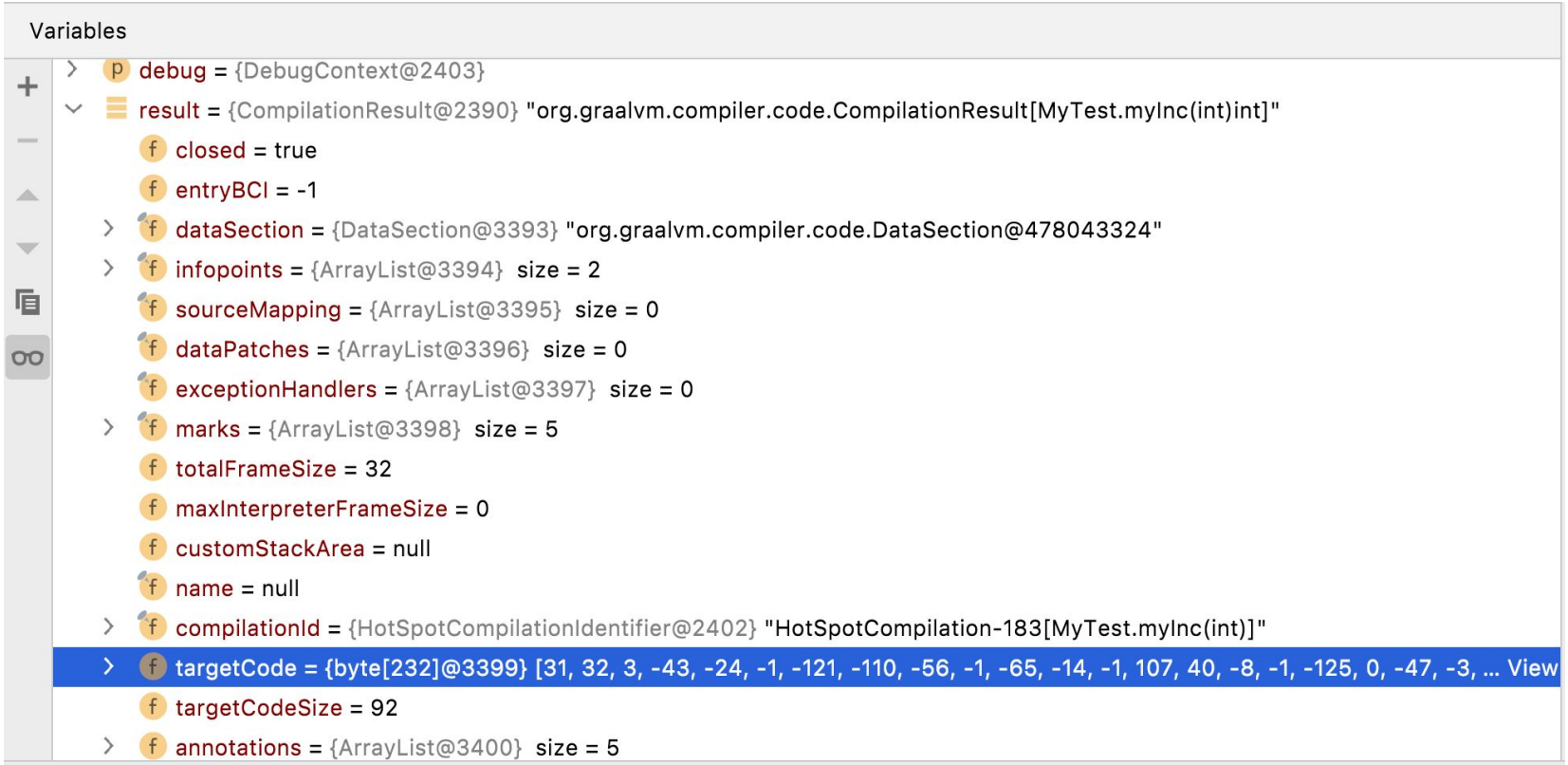
Graal Compiler



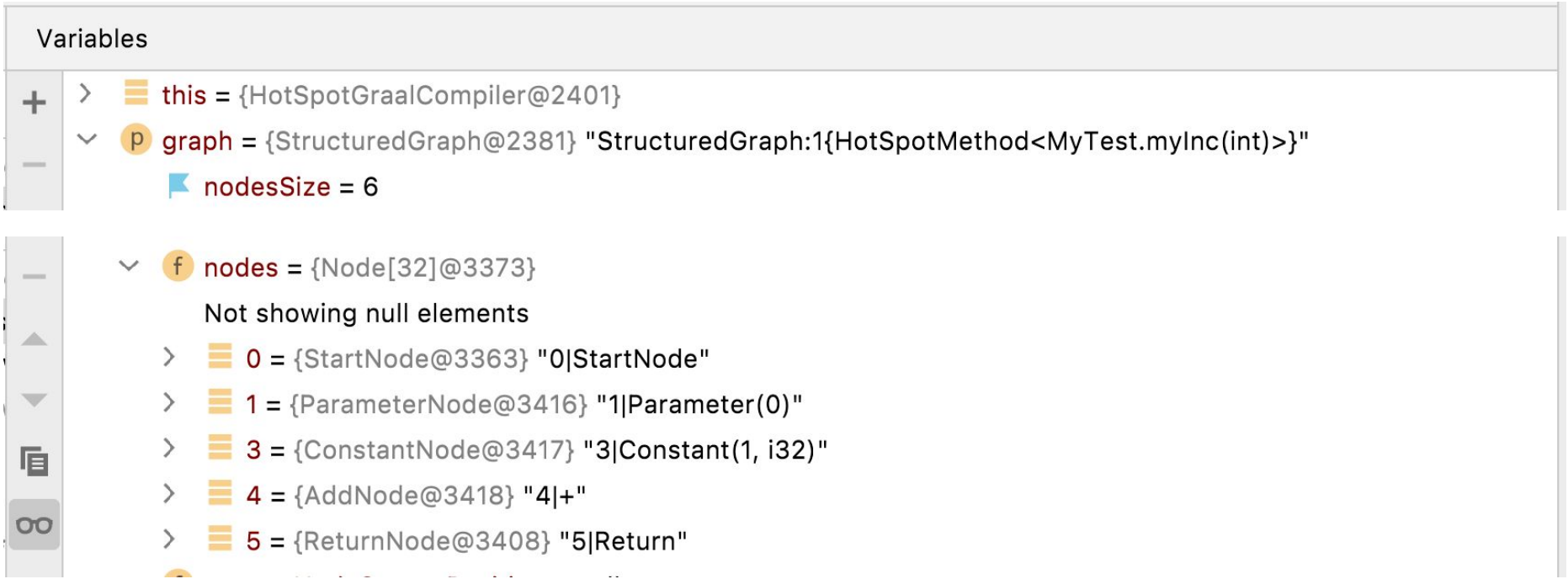
iload_0	1a
iconst_1	04
iadd	60
ireturn	ac

Source: https://en.wikipedia.org/wiki/List_of_Java_bytecode_instructions

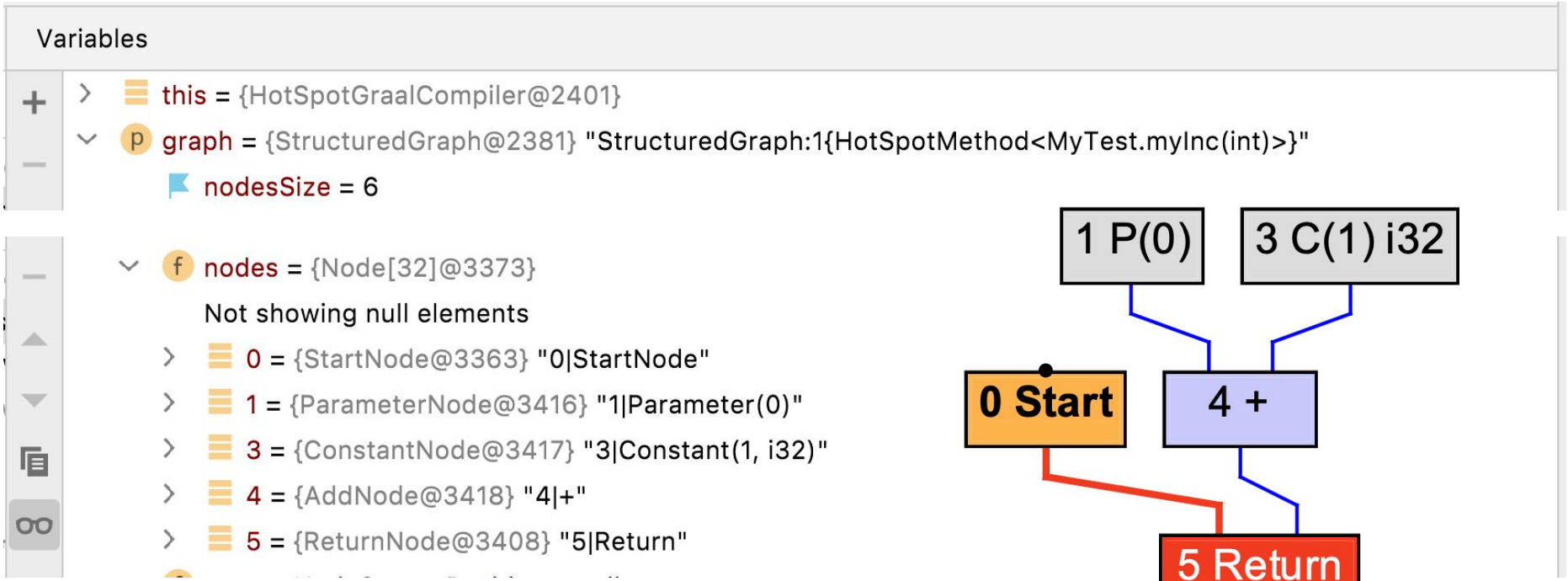
Graal Compiler



Graal Compiler



Graal Compiler



https://www.graalvm.org/tools/igv/

Emit Code

AddNode.java

```
192     @Override
193     public void generate(NodeLIRBuilderTool nodeValueMap, ArithmeticLIRGeneratorTool gen) {
194         Value op1 = nodeValueMap.operand(getX());
195         assert op1 != null : getX() + ", this=" + this;
196         Value op2 = nodeValueMap.operand(getY());
197         if (shouldSwapInputs(nodeValueMap)) {...}
202         nodeValueMap.setResult( node: this, gen.emitAdd(op1, op2,  setFlags: false));
203     }
```

graal/compiler/src/org.graalvm.compiler.nodes/src/org/graalvm/compiler/nodes/calc/AddNode.java

Emit Code

AddNode.java

```
192     @Override
193     public void generate(NodeLIRBuilderTool nodeValueMap, ArithmeticLIRGeneratorTool gen) {
194         Value op1 = nodeValueMap.operand(getX());
195         assert op1 != null : getX() + ", this=" + this;
196         Value op2 = nodeValueMap.operand(getY());
197         if (shouldSwapInputs(nodeValueMap)) {...}
202         nodeValueMap.setResult( node: this, gen.emitAdd(op1, op2,  setFlags: false));
203     }
```

ArithmeticLIRGenerator.java

```
56
57      protected abstract Variable emitAdd(LIRKind resultKind, Value a, Value b, boolean setFlags);
58
59
60
61
```

Choose Implementation of **emitAdd** (2 methods found)

- AArch64ArithmeticLIRGenerator (org.graalvm.compiler.core.aarch64) — org.graalvm.compiler.core.aarch64
- AMD64ArithmeticLIRGenerator (org.graalvm.compiler.core.amd64) — org.graalvm.compiler.core.amd64

graal/compiler/src/org.graalvm.compiler.lir/src/org/graalvm/compiler/lir/gen/ArithmeticLIRGenerator.java

Emit Code

AddNode.java

```
192     @Override
193     public void generate(NodeLIRBuilderTool nodeValueMap, ArithmeticLIRGeneratorTool gen) {
194         Value op1 = nodeValueMap.operand(getX());
195         assert op1 != null : getX() + ", this=" + this;
196         Value op2 = nodeValueMap.operand(getY());
197         if (shouldSwapInputs(nodeValueMap)) {...}
```

AArch64ArithmeticLIRGenerator.java

```
97      @Override
98      protected Variable emitAdd(LIRKind resultKind, Value a, Value b, boolean setFlags) {
99          if (isNumericInteger(a.getPlatformKind())) {
100             AArch64ArithmeticOp op = setFlags ? AArch64ArithmeticOp.ADDS : AArch64ArithmeticOp.ADD;
101             return emitBinary(resultKind, op, commutative: true, a, b);
102         } else {
103             assert !setFlags : "Cannot set flags on floating point arithmetic";
104             return emitBinary(resultKind, AArch64ArithmeticOp.FADD, commutative: true, a, b);
105         }
106     }
```

graal/compiler/src/org.graalvm.compiler.core.aarch64/src/org/graalvm/compiler/core/aarch64/AArch64ArithmeticLIRGenerator.java

Code Generation

```
MyTest.myInc(I)I  [0x0000ffffdd2d6a80, 0x0000ffffdd2d6ae0]  96 bytes
[Entry Point]
[Verified Entry Point]
[Constants]
  # {method} {0x0000ffff7ce2a7d8} 'myInc' '(I)I' in 'MyTest'
  # parm0:    c_rarg1   = int
  #           [sp+0x20]  (sp of caller)
  0x0000ffffdd2d6a80: nop
  0x0000ffffdd2d6a84: mov       x8, #0xffffffffffffc000         // #-16384
  0x0000ffffdd2d6a88: movk      x8, #0xfffe, lsl #16
  0x0000ffffdd2d6a8c: str       xzr, [sp,x8]
  0x0000ffffdd2d6a90: sub       sp, sp, #0x20
  0x0000ffffdd2d6a94: stp       x29, x30, [sp,#16]
  0x0000ffffdd2d6a98: add       w0, w1, #0x1      W0 = W1 + 1
  0x0000ffffdd2d6a9c: ldp       x29, x30, [sp,#16]
  0x0000ffffdd2d6aa0: add       sp, sp, #0x20
  0x0000ffffdd2d6aa4: ldr       x8, [x28,#264]
  0x0000ffffdd2d6aa8: ldr       wzr, [x8]       ;   {poll_return}
  0x0000ffffdd2d6aac: ret
```

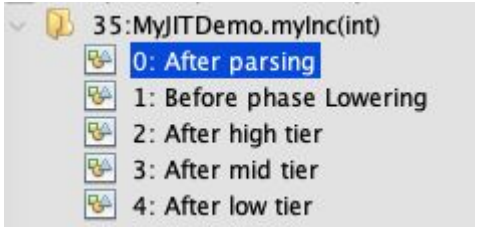
```
$GRAAL_HOME/bin/java -XX:+UnlockExperimentalVMOptions -XX:+EnableJVMCI -XX:+UseJVMCICompiler \
-XX:-BackgroundCompilation -XX:CompileCommand=print,MyTest.myInc' MyTest
```

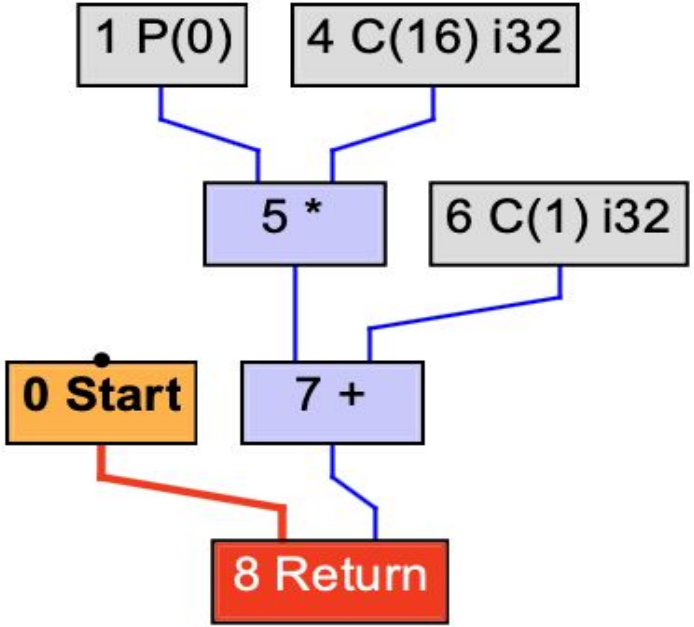
Compiler Optimisations

```
public class MyJITDemo {
    private static int myMult(final int a) {
        return a * 16;
    }
    public static int myInc(int a) {
        a = myMult(a);
        return a + 1;
    }
    public static void main(String[] args) {
        for(int i = 0; i < 1_000_000; i++)
            MyJITDemo.myInc(i);
    }
}
```

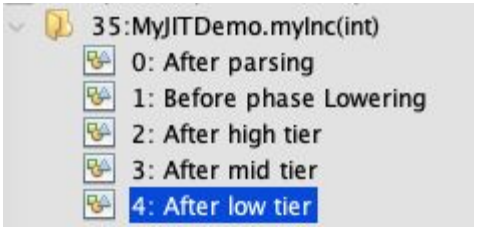
```
$GRAAL_HOME/bin/java -server -XX:+UnlockExperimentalVMOptions -XX:+EnableJVMCI -XX:+UseJVMCICompiler -Dgraal.Dump=: \
-XX:-BackgroundCompilation '-XX:CompileCommand=compileonly, MyJITDemo.myInc' '-XX:CompileCommand=print, MyJITDemo.myInc' MyJITDemo
```

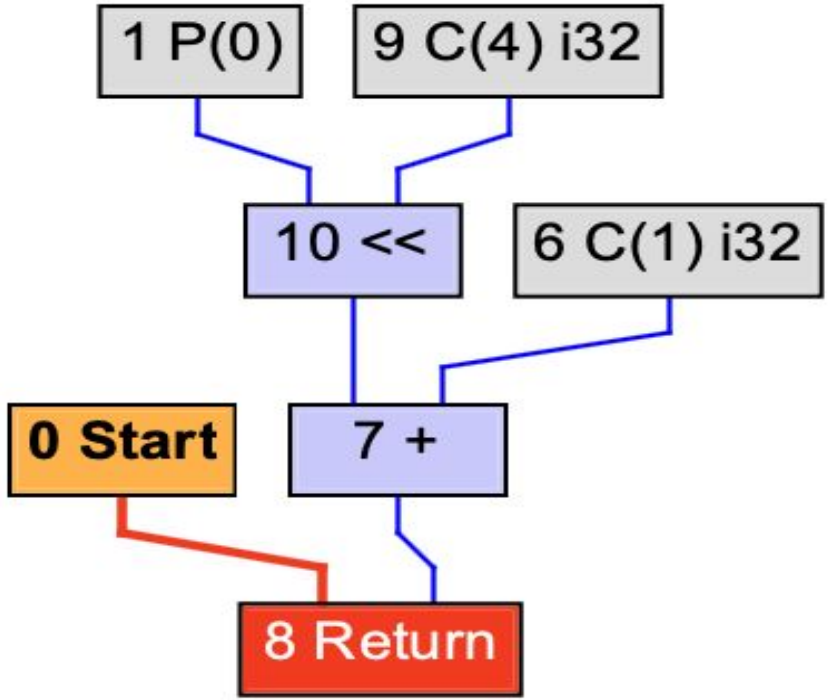
Graal Compiler: Before Compilation





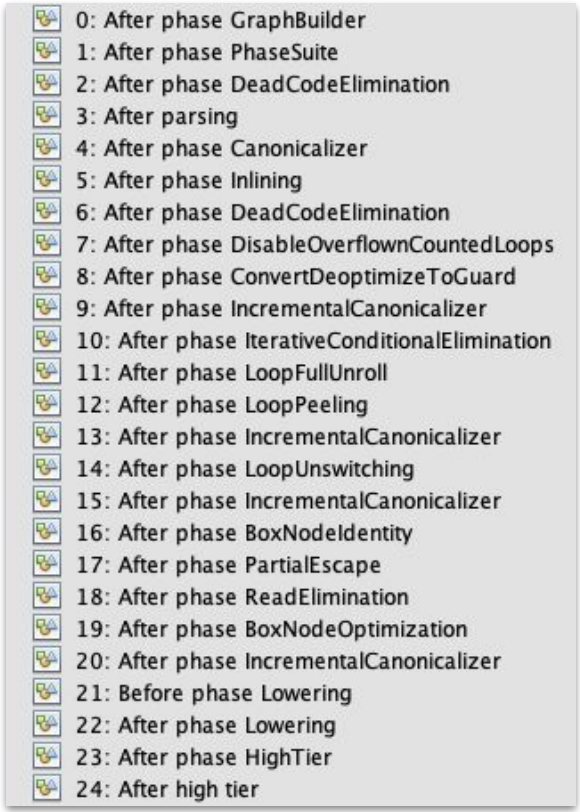
Graal Compiler: After Compilation

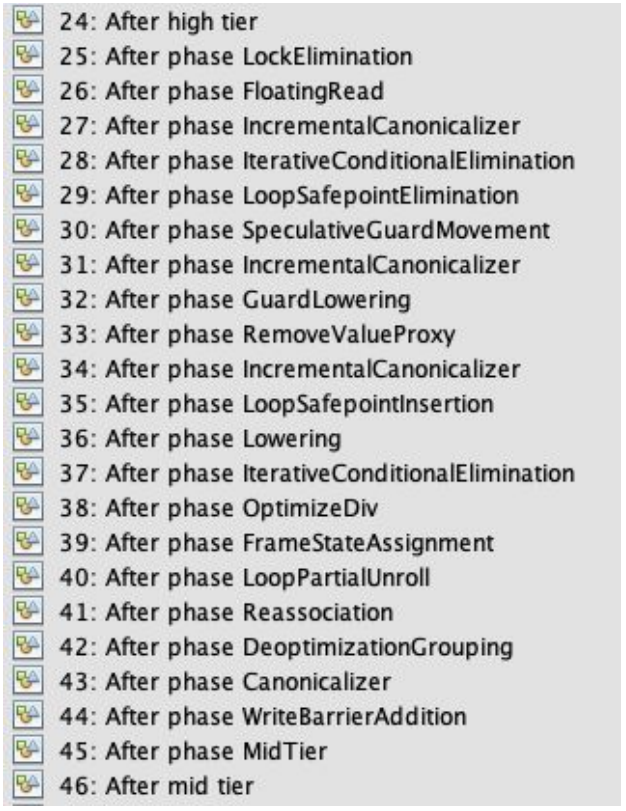




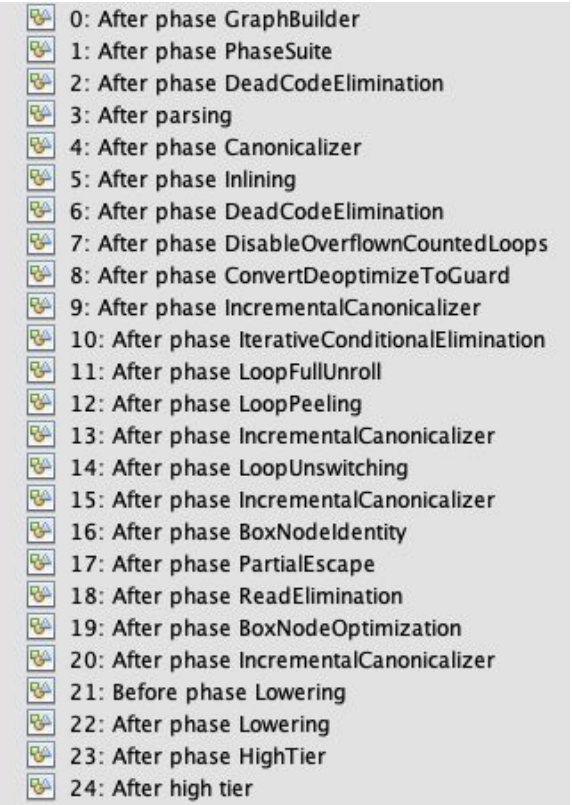
Compilation Pipeline (High Tier)

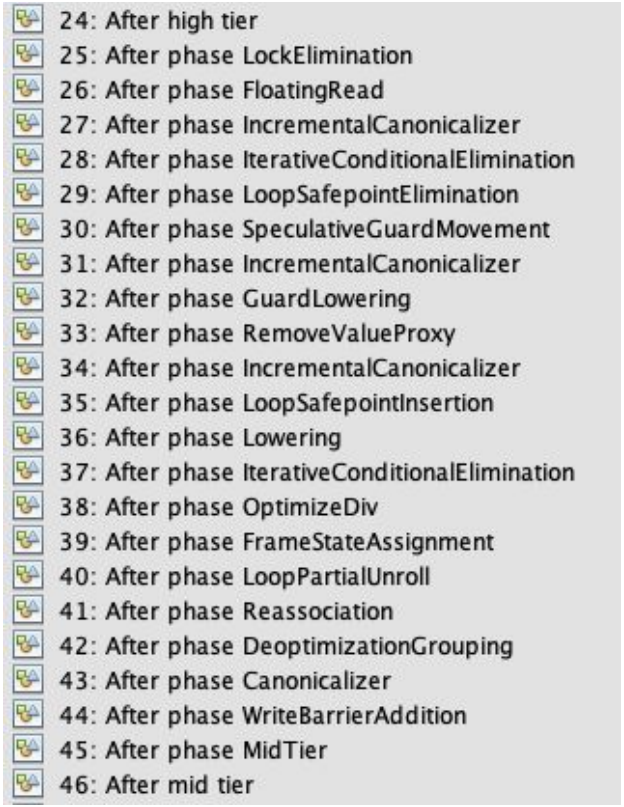
Compilation Pipeline (Mid Tier)

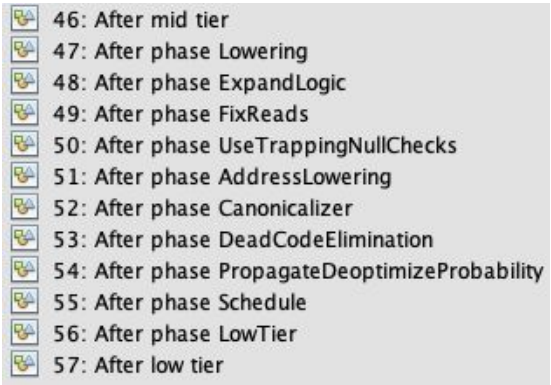




Compilation Pipeline (Low Tier)

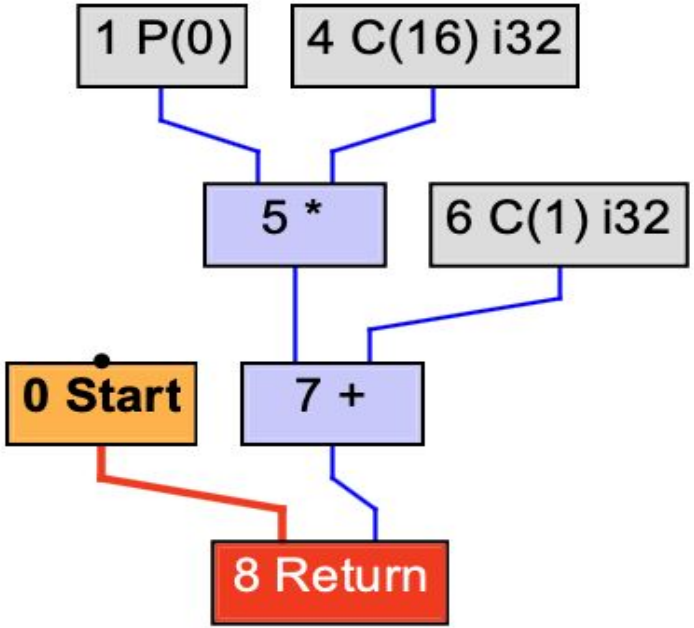






Optimising a graph

- Visitor pattern
  - `canonicalise()`
  - `generate()`



Simplify multiplication

MulNode.java

```
128
129    @    public static ValueNode canonical(Stamp stamp, ValueNode forX, long i, NodeView view) {
130             if (i == 0) {
131                 return ConstantNode.forIntegerStamp(stamp,  value: 0);
132             } else if (i == 1) {
133                 return forX;
134             } else if (i == -1) {
135                 return NegateNode.create(forX, view);
136             } else if (i > 0) {
137                 if (CodeUtil.isPowerOf2(i)) {
138                     return new LeftShiftNode(forX, ConstantNode.forInt(CodeUtil.log2(i)));
```

graal/compiler/src/org.graalvm.compiler.nodes/src/org/graalvm/compiler/nodes/calc/MulNode.java

Optimised Assembly

```
MyJITDemo.myInc(I)I  [0x0000ffffdd2d6d80, 0x0000ffffdd2d6de0]  96 bytes
[Entry Point]
[Verified Entry Point]
[Constants]
  # {method} {0x0000ffff7ce2a898} 'myInc' '(I)I' in 'MyJITDemo'
  # parm0:    c_rarg1   = int
  #           [sp+0x20]  (sp of caller)
  0x0000ffffdd2d6d80: nop
  0x0000ffffdd2d6d84: mov       x8, #0xffffffffffffc000         // #-16384
  0x0000ffffdd2d6d88: movk      x8, #0xfffe, lsl #16
  0x0000ffffdd2d6d8c: str       xzr, [sp,x8]
  0x0000ffffdd2d6d90: sub       sp, sp, #0x20
  0x0000ffffdd2d6d94: stp       x29, x30, [sp,#16]
  0x0000ffffdd2d6d98: orr       w0, wzr, #0x1
  0x0000ffffdd2d6d9c: add       w0, w0, w1, lsl #4
                                                ; -

  0x0000ffffdd2d6da0: ldp       x29, x30, [sp,#16]
  0x0000ffffdd2d6da4: add       sp, sp, #0x20
  0x0000ffffdd2d6da8: ldr       x8, [x28,#264]
  0x0000ffffdd2d6dac: ldr       wzr, [x8]       ;   {poll_return}
  0x0000ffffdd2d6db0: ret
```

W0 = 1
W0 = W0 + (W1 << 4)

Platform specific instructions

AArch64NodeMatchRules.java

```
316     @MatchRule("(Add=op x (LeftShift=lshift (SignExtend=ext y) Constant))")
317     public ComplexMatchResult mergeSignExtendByShiftIntoAddSub(BinaryNode op, LeftShiftNode lshift,
318                                                                ValueNode ext, ValueNode x, ValueNode y) {
319         assert isNumericInteger(lshift);
320         int shiftAmt = getClampedShiftAmt(lshift);
321         if (!isSupportedExtendedAddSubShift((IntegerConvertNode<?, ?>) ext, shiftAmt)) {...}
324         ExtendType extType;
325         if (ext instanceof SignExtendNode) {...} else {...}
330         return emitExtendedAddSubShift(op, x, y, extType, shiftAmt);
331     }
```

Summary

- Graal as a Just-in-Time (JIT) compiler
- Graph representation of a Java program
- Compiler Optimisations
  - Platform specific
  - Platform agnostic
- Generate assembly instructions

Challenges

- Benchmarking
  - Prove the optimisation improves performance
  - Analyse behaviour on different platforms
- Prototyping
  - Test a given sequence of instructions
  - Identify suitable instructions

We didn’t look at

- Instruction Scheduling
- Register Allocation
- Graal Intrinsics
- Graal Native Image
- Graal for running other languages
- Other JVM Components

Thank You!

Questions?

Resources

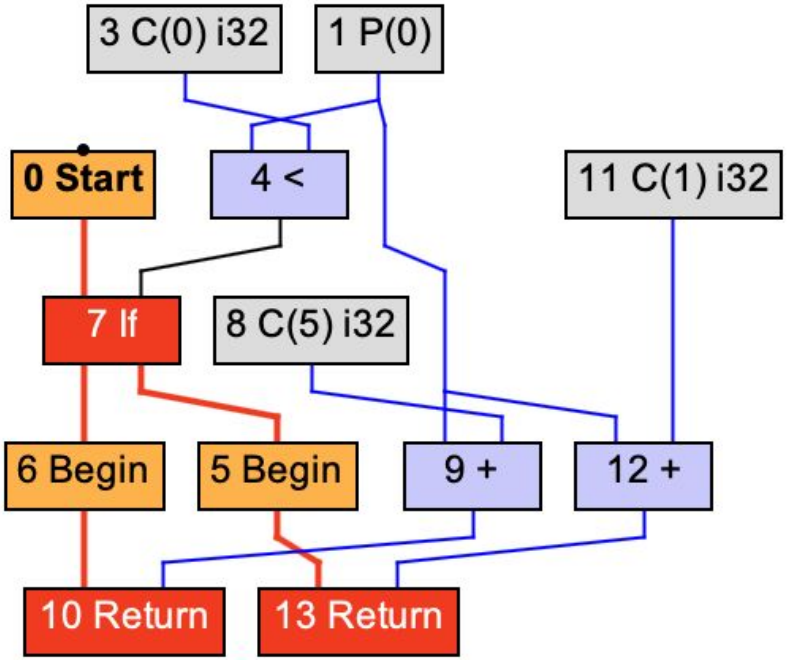
- Graal Repo: https://github.com/oracle/graal
- Graal Publications & Videos: https://github.com/oracle/graal/blob/master/docs/Publications.md
- Graal Compiler Docs: https://github.com/oracle/graal/tree/master/compiler/docs
- Top 10 Things To Do With GraalVM: https://medium.com/graalvm/graalvm-ten-things-12d9111f307d
- IdealGraphVisualizer: https://github.com/oracle/graal/blob/master/docs/tools/ideal-graph-visualizer.md

https://github.com/SwapnilGaikwad/jmanc-graal

Graal Compiler

```
public class MyTest {
  public static int myInc(int a) {
    if(a < 0){
      return a + 5;
    }
    return a + 1;
  }

  public static void main(String[] args)
{
    int i = -3;
    while(true) {
      MyTest.myInc(i++);
      if(i > 100)
        i = -10;
    }
  }
}
```



Optimisation Phase

OptimizeDivPhase.java

```
48  public class OptimizeDivPhase extends Phase {
49
50      @Override
51  @   protected void run(StructuredGraph graph) {
52          for (IntegerDivRemNode rem : graph.getNodes(IntegerDivRemNode.TYPE)) {
53              if (rem instanceof SignedRemNode && divByNonZeroConstant(rem)) {
54                  optimizeRem(rem);
55              }
56          }
57          for (IntegerDivRemNode div : graph.getNodes(IntegerDivRemNode.TYPE)) {...}
62      }
```

Optimisation Phase

OptimizeDivPhase.java

```
73  @   protected final void optimizeRem(IntegerDivRemNode rem) {
74          assert rem.getOp() == IntegerDivRemNode.Op.REM;
75          // Java spec 15.17.3.: (a/b)*b+(a%b) == a
76          // so a%b == a-(a/b)*b
77          StructuredGraph graph = rem.graph();
78          ValueNode div = findDivForRem(rem);
79          ValueNode mul = BinaryArithmeticNode.mul(graph, div, rem.getY(), NodeView.DEFAULT);
80          ValueNode result = BinaryArithmeticNode.sub(graph, rem.getX(), mul, NodeView.DEFAULT);
81          graph.replaceFixedWithFloating(rem, result);
82      }
```