

Der eilige Graal

Michael Wiedeking
MATHEMA Software GmbH

Die (nicht vorhandene) ideale Maschine

```
class Simple {
    int half(int x) {
        return x / 2;
    }
    int f(int x, int y) {
        int r = x + y;
        r = half(r);
        if (r < 50) {
            return 50;
        }
        return r;
    }
}

public static void main(String[] a) {
    Simple s = new Simple();
    System.out.println(s.f(37, 44));
}
```

```
int half(int x) {
    return x / 2;
}

int half(int);
0: iload_1
1: iconst_2
2: idiv
3: ireturn
```

```
int f(int x, int y) {
    int r = x + y;
    r = half(r);
    if (r < 50) {
        return 50;
    }
    return r;
}

int f(int, int);
0: iload_1
1: iload_2
2: iadd
3: istore_3
4: aload_0
5: iload_3
6: invokevirtual #20; //Method half:(I)I
9: istore_3
10: iload_3
11: bipush 50
13: if_icmpge 19
16: bipush 50
18: ireturn
19: iload_3
20: ireturn
```

```
public static void main(java.lang.String[]);
0: new #1; //class Simple
3: dup
4: invokespecial #27; //Method <init>:()V
7: astore_1
8: getstatic #28; //Field java/lang/System.out:L java/io/PrintStream;
11: aload_1
12: bipush 37
14: bipush 44
16: invokevirtual #34; //Method f:(II)I
19: invokevirtual #36; //Method java/io/PrintStream.println:(I)V
22: return
```

```
public static void main(java.lang.String[]);
0: new #1; //class Simple
3: dup
4: invokespecial #27; //Method <init>:()V
7: astore_1
8: getstatic #28; //Field java/lang/System.out:L java/io/PrintStream;
11: aload_1
12: bipush 37
14: bipush 44
16: invokevirtual #34; //Method f:(II)I
19: invokevirtual #36; //Method java/io/PrintStream.println:(I)V
22: return
```

```
11: aload_1
12: bipush 37
14: bipush 44
16: invokevirtual #34; //Method f:(II)I
19: invokevirtual #36; //Method java/io/PrintStream.println:(I)V
```

*2023B6

```
11: aload_1
12: bipush 37
14: bipush 44
16: invokevirtual #34; //Method f:(II)I
19: invokevirtual #36; //Method java/io/PrintStream.println:(I)V
```

37
*2023B6

```
11: aload_1
12: bipush 37
14: bipush 44
16: invokevirtual #34; //Method f:(II)I
19: invokevirtual #36; //Method java/io/PrintStream.println:(I)V
```

44
37
*2023B6

```
11: aload_1
12: bipush 37
14: bipush 44
16: invokevirtual #34; //Method f:(II)I
19: invokevirtual #36; //Method java/io/PrintStream.println:(I)V
```

44
37
*2023B6

```
int f(int, int);
0: iload_1
1: iload_2
2: iadd
3: istore_3
4: aload_0
5: iload_3
6: invokevirtual #20; //Method half:(I)I
9: istore_3
10: ...
```

3 (r):	
2 (y):	44
1 (x):	37
0 (hi):	*2023B6

```
int f(int, int);
0: iload_1
1: iload_2
2: iadd
3: istore_3
4: aload_0
5: iload_3
6: invokevirtual #20; //Method half:(I)I
9: istore_3
10: ...
```

37	
3 (r):	
2 (y):	44
1 (x):	37
0 (hi):	*2023B6

```
int f(int, int);
0: iload_1
1: iload_2
2: iadd
3: istore_3
4: aload_0
5: iload_3
6: invokevirtual #20; //Method half:(I)I
9: istore_3
10: ...
```

44	
37	
3 (r):	
2 (y):	44
1 (x):	37
0 (hi):	*2023B6

```
int f(int, int);
0: iload_1
1: iload_2
2: iadd
3: istore_3
4: aload_0
5: iload_3
6: invokevirtual #20; //Method half:(I)I
9: istore_3
10: ...
```

81	
3 (r):	
2 (y):	44
1 (x):	37
0 (hi):	*2023B6

```
int f(int, int);
0: iload_1
1: iload_2
2: iadd
3: istore_3
4: aload_0
5: iload_3
6: invokevirtual #20; //Method half:(I)I
9: istore_3
10: ...
```

*2023B6	
81	
3 (r):	
2 (y):	44
1 (x):	37
0 (hi):	*2023B6

```
int f(int, int):
0: iload_1
1: iload_2
2: iadd
3: istore_3
4: aload_0
5: iload_3
6: invokevirtual #20; //Method half:(I)I
9: istore_3
10: ...
```

3 (r):	81
2 (y):	44
1 (x):	37
0 (this):	*2023B6

```
int f(int, int):
0: iload_1
1: iload_2
2: iadd
3: istore_3
4: aload_0
5: iload_3
6: invokevirtual #20; //Method half:(I)I
9: istore_3
10: ...
```

3 (r):	81
2 (y):	44
1 (x):	37
0 (this):	*2023B6

```
int f(int, int):
0: iload_1
1: iload_2
2: iadd
3: istore_3
4: aload_0
5: iload_3
6: invokevirtual #20; //Method half:(I)I
9: istore_3
10: ...
```

3 (r):	40
2 (y):	44
1 (x):	37
0 (this):	*2023B6

```
int f(int, int):
6: ...
9: istore_3
10: iload_3
11: bipush 50
13: if_icmpge 19
16: bipush 50
18: return
19: iload_3
20: return
```

3 (r):	40
2 (y):	44
1 (x):	37
0 (this):	*2023B6

```
int f(int, int):
6: ...
9: istore_3
10: iload_3
11: bipush 50
13: if_icmpge 19
16: bipush 50
18: return
19: iload_3
20: return
```

3 (r):	40
2 (y):	44
1 (x):	37
0 (this):	*2023B6

```
int f(int, int):
6: ...
9: istore_3
10: iload_3
11: bipush 50
13: if_icmpge 19
16: bipush 50
18: return
19: iload_3
20: return
```

3 (r):	40
2 (y):	44
1 (x):	37
0 (this):	*2023B6

```
int f(int, int):
6: ...
9: istore_3
10: iload_3
11: bipush 50
13: if_icmpge 19
16: bipush 50
18: return
19: iload_3
20: return
```

3 (r):	40
2 (y):	44
1 (x):	37
0 (this):	*2023B6

```
int f(int, int):
6: ...
9: istore_3
10: iload_3
11: bipush 50
13: if_icmpge 19
16: bipush 50
18: return
19: iload_3
20: return
```

3 (r):	40
2 (y):	44
1 (x):	37
0 (this):	*2023B6

```
int f(int, int):
6: ...
9: istore_3
10: iload_3
11: bipush 50
13: if_icmpge 19
16: bipush 50
18: return
19: iload_3
20: return
```

50

```
11: aload_1
12: bipush 37
14: bipush 44
16: invokevirtual #34; //Method f:(I)I
19: invokevirtual #36; //Method java/io/PrintStream.println:(I)V
```

50
44
37
*2023B6

```
11: aload_1
12: bipush 37
14: bipush 44
16: invokevirtual #34; //Method f:(I)I
19: invokevirtual #36; //Method java/io/PrintStream.println:(I)V
```

50

Hot Spot

Hot Spot: Nachtlokal
—Duden

Hot Spot: Krisenherd
—Duden

Hot Spot: Spitzenlastpunkt
—LEO

Spot: Abk. für **Spotlight** das; -s, -s <engl.>:
Beleuchtung od. Scheinwerfer, der auf einen Punkt
gerichtet ist u. dabei die Umgebung im Dunkeln lässt
—Duden

```

int half(int);
0: iload_1
1: iconst_2
2: idiv
3: ireturn

while (code.hasNext()) {
  switch (code.next()) {
    case iload_1: ... break;
    case iconst_2: ... break;
    case ireturn: ... break;
    case idiv: ... break;
    ...
  }
}

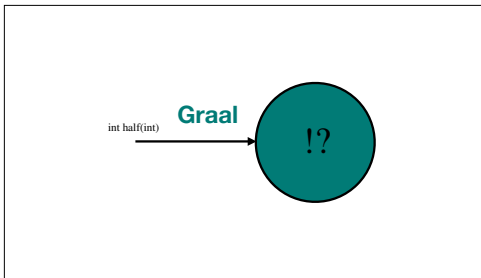
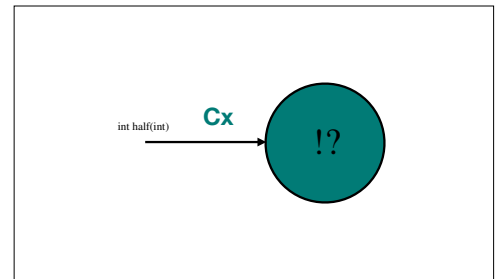
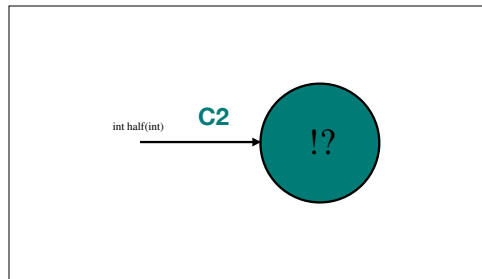
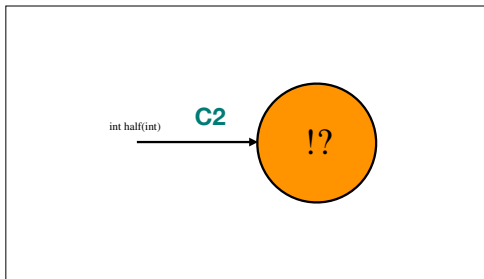
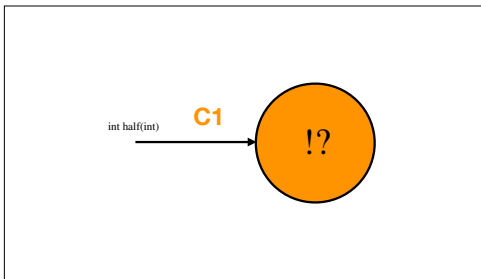
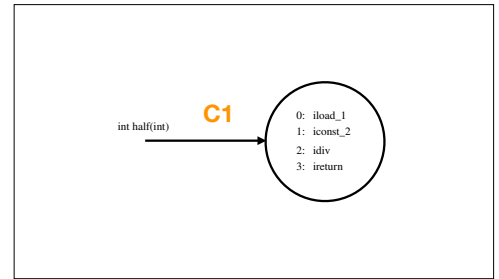
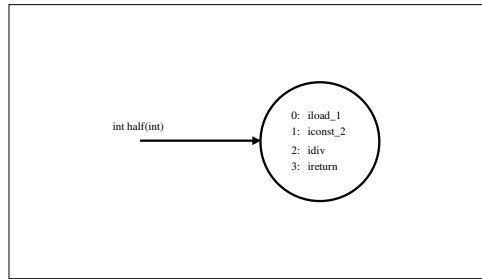
```

```

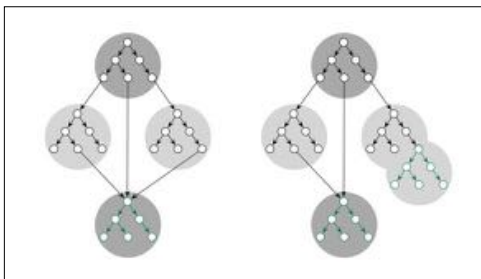
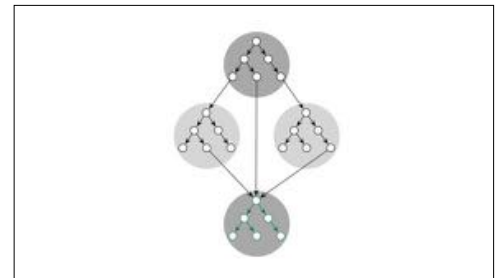
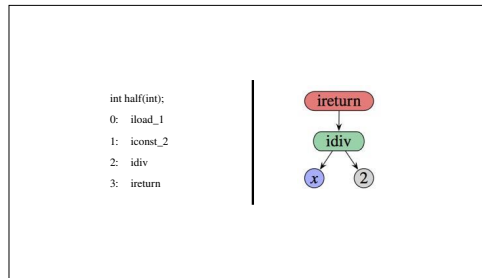
int f(int, int);
0: iload_1
1: iload_2
2: iadd
3: istore_3
4: aload_0
5: iload_3
6: invokevirtual #20 // half:()I
9: istore_3
...

while (code.hasNext()) {
  switch (code.next()) {
    case iload_1: ... break;
    case iconst_2: ... break;
    case ireturn: ... break;
    case invokevirtual: ... break;
    ...
  }
}

```



Der eilige Graal



```

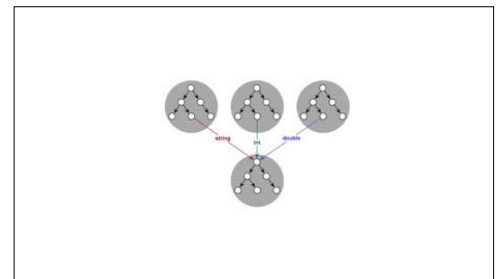
fun add(a, b) {
  return a + b
}

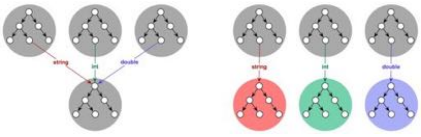
```

```

fun add(a, b) {
  assert add(1, 2) = 3
  assert add(1/2, 1/4) = 3/4
  assert add(1/2, -1) = -1/2
  assert add("a", "b") = "ab"
  assert add("a", 1) = "a1"
  assert add([1, 2], [3, 4]) = [1, 2, 3, 4]
}

```



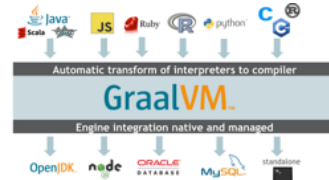


```
-XX:+UnlockExperimentalVMOptions
-XX:+UseJVMCICompiler
```

"That library is not available in my language. I need to rewrite it."

"That language would be the perfect fit for my problem, but we cannot run it in our environment."

"That problem is already solved in my language, but the language is too slow."



```
var ArrayList = Java.type("java.util.ArrayList");
var list = new ArrayList();
list.add("Apfel");
list.add("Birne");
list.add("Zitron");
print(list);
```

```
String script = "console.log('Hello from the project');";
```

```
Context jsContext = Context.create("js");
jsContext.eval("js", script);
```

```
var array = Polyglot.eval("R", "c(1, 2, 42, 4)")
console.log(array[2]);
```

```
$ js --polyglot --jvm polyglot.js
42
$ node --polyglot --jvm polyglot.js
42
```

```
array <- eval.polyglot("js", "[1, 2, 42, 4]")
print(array[3L])
```

```
$ Rscript --polyglot --jvm polyglot.R
[1] 42
```

```
array = Polyglot.eval("js", "[1, 2, 42, 4]")
puts array[2]
```

```
$ ruby --polyglot --jvm polyglot.rb
42
```

```
import polyglot
array = polyglot.eval(language="js", string="[1, 2, 42, 4]")
print(array[2])
```

```
$ graalpython --polyglot --jvm polyglot.py
42
```

```
import org.graalvm.polyglot.*;
class Polyglot {
    public static void main(String[] args) {
        Context polyglot = Context.create();
        Value array = polyglot.eval("js", "[1, 2, 42, 4]");
        int result = array.getElement(2).asInt();
        System.out.println(result);
    }
}

$ javac Polyglot.java
$ java Polyglot
42
```

```
#include <stdio.h>
#include <polyglot.h>
int main() {
    void* array = polyglot_eval("js", "[1, 2, 42, 4]");
    int element = polyglot_as_i32(polyglot_get_array_element(array, 2));
    printf("%d\n", element);
    return element;
}

$ clang -g -O1 -c -emit-llvm -I$GRAALVM_HOME/jre/languages/llvm/polyglot.c
$ lli --polyglot --jvm polyglot.bc
42
```

Indy
(und Condy)

```
void f(List<Integer> list) {
    ...
    ... list.stream().filter(x -> x % 2 == 0) ...
    ...
}
```

```

0: aload_1
1: invokeinterface #18 // java.util.List.stream:
    ()Ljava/util/stream/Stream;
6: invokedynamic #27 // #0:test:
    ()Ljava/util/function/Predicate;
11: invokeinterface #28 // java/util/stream/Stream.filter:
    (Ljava/util/function/Predicate;)Ljava/util/stream/Stream;

```

```

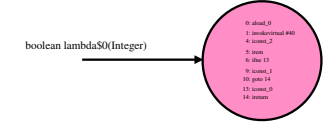
BootstrapMethods:
0: #58 REF_invokeStatic
    java/lang/invoke/LambdaMetafactory.metafactory:()
    Ljava/lang/invoke/MethodHandlesLookup;
    Ljava/lang/String;
    Ljava/lang/invoke/MethodType;
    Ljava/lang/invoke/MethodType;
    Ljava/lang/invoke/MethodHandle;
    Ljava/lang/invoke/MethodType;
    Ljava/lang/invoke/CallSite;
Method arguments:
#60 (Ljava/lang/Object;)Z
#63 REF_invokeStatic simple/S.Lambda$0:(Ljava/lang/Integer;)Z
#64 (Ljava/lang/Integer;)Z

```

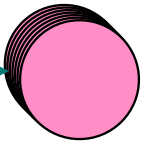
```

private static boolean lambda$0(java.lang.Integer);
0: aload_0
1: invokevirtual #40 // java/lang/Integer.intValue:()I
4: iconst_2
5: irem
6: ifne 13
9: iconst_1
10: goto 14
13: iconst_0
14: return

```



boolean lambda\$0(Integer)



```
int[] a = {1000, 1001, 1002};
```

```

0: iconst_3
1: newarray int
3: dup
4: iconst_0
5: sipush 1000
8: iastore
9: dup
10: iconst_1
11: sipush 1001
14: iastore
15: dup
16: iconst_2
17: sipush 1002
20: iastore
21: astore_1

```

```
int[] a = {1000, 1001, 1002, 1003, 1004, 1005, 1006};
```

```

0: sipush 7
2: invokevirtual
4: dup
5: iconst_0
6: sipush 3000
9: iastore
10: dup
11: iconst_1
12: sipush 1000
15: iastore
16: dup
17: iconst_2
18: sipush 1002
21: iastore
22: iconst_3
23: iconst_3
24: sipush 1003
27: iastore
28: dup
29: iconst_4
30: sipush 1004
33: iastore
34: dup
35: iconst_5
36: sipush 1005
39: iastore
40: dup
41: iconst_6
42: sipush 1006
45: iastore
46: iconst_1
47: astore_1

```

```

int[] a = {
1000, 1001, 1002, 1003, 1004, 1005, 1006, 1007, 1008, 1009,
1010, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019,
1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029,
1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039,
1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049,
1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059,
1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069,
1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079,
1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089,
1090, 1091, 1092, 1093, 1094, 1095, 1096, 1097, 1098, 1099
};

```

```

0: sipush 100
1: invokevirtual
2: sipush 100
3: invokevirtual
4: sipush 100
5: invokevirtual
6: sipush 100
7: invokevirtual
8: sipush 100
9: invokevirtual
10: sipush 100
11: invokevirtual
12: sipush 100
13: invokevirtual
14: sipush 100
15: invokevirtual
16: sipush 100
17: invokevirtual
18: sipush 100
19: invokevirtual
20: sipush 100
21: invokevirtual
22: sipush 100
23: invokevirtual
24: sipush 100
25: invokevirtual
26: sipush 100
27: invokevirtual
28: sipush 100
29: invokevirtual
30: sipush 100
31: invokevirtual
32: sipush 100
33: invokevirtual
34: sipush 100
35: invokevirtual
36: sipush 100
37: invokevirtual
38: sipush 100
39: invokevirtual
40: sipush 100
41: invokevirtual
42: sipush 100
43: invokevirtual
44: sipush 100
45: invokevirtual
46: sipush 100
47: invokevirtual
48: sipush 100
49: invokevirtual
50: sipush 100
51: invokevirtual
52: sipush 100
53: invokevirtual
54: sipush 100
55: invokevirtual
56: sipush 100
57: invokevirtual
58: sipush 100
59: invokevirtual
60: sipush 100
61: invokevirtual
62: sipush 100
63: invokevirtual
64: sipush 100
65: invokevirtual
66: sipush 100
67: invokevirtual
68: sipush 100
69: invokevirtual
70: sipush 100
71: invokevirtual
72: sipush 100
73: invokevirtual
74: sipush 100
75: invokevirtual
76: sipush 100
77: invokevirtual
78: sipush 100
79: invokevirtual
80: sipush 100
81: invokevirtual
82: sipush 100
83: invokevirtual
84: sipush 100
85: invokevirtual
86: sipush 100
87: invokevirtual
88: sipush 100
89: invokevirtual
90: sipush 100
91: invokevirtual
92: sipush 100
93: invokevirtual
94: sipush 100
95: invokevirtual
96: sipush 100
97: invokevirtual
98: sipush 100
99: invokevirtual

```

Fazit

(Unbedingt) ausprobieren

(Unbedingt) ausprobieren
Verzögerungen berücksichtigen

(Unbedingt) ausprobieren
Verzögerungen berücksichtigen
Sicherstellen, dass man das hat, was man will

(Unbedingt) ausprobieren
Verzögerungen berücksichtigen
Sicherstellen, dass man das hat, was man will
Eventuell Ahead of Time Compilation (AOT)

(Unbedingt) ausprobieren
Verzögerungen berücksichtigen
Sicherstellen, dass man das hat, was man will
Eventuell *Ahead of Time Compilation (AOT)*
Und gegebenenfalls polyglottes Programmieren

Fragen?

Vielen Dank!