



Java Bytecode

Eike Robert
Freie Universität Berlin

Softwareprojekt Übersetzerbau, 2013

Bytecode Allgemein

Architektur

Aufbau

Instruktionsgruppen

Beispiel



Robert Fehrmann

rob19xx



Eike Cochu

eikecochu

Bytecode Allgemein

Architektur

Aufbau

Instruktionsgruppen

Beispiel

Architektur

- ▶ 1 Byte pro Befehl → $2^8 = 256$ Befehle
- ▶ 51 davon z.Zt. unbenutzt
- ▶ 3 gesperrt
 - ▶ 0xCA → Breakpoint-Marker
 - ▶ 0xFE, 0xFF → Reserviert für spezielle Debuggerbefehle
- ▶ kompakt
- ▶ JVM Stackorientiert → kompatibel zu registerarmen Plattformen (z.B. Intel 80486: 9 Register)



Bytecode Allgemein

Architektur

Aufbau

Instruktionsgruppen

Beispiel

`<offset> <opcode> [<arg1>, <arg2>]`

- ▶ `offset` → aktuelle Bytezahl, Sprungmarker
- ▶ `opcode`, `args` → Befehl und Argumente

```
<offset> <opcode> [<arg1>, <arg2>]
```

- ▶ `offset` → aktuelle Bytezahl, Sprungmarker
- ▶ `opcode`, `args` → Befehl und Argumente

```
0 iinc 0, 1
```

- ▶ Befehl `iinc`: Inkrementieren
- ▶ Prefix `i`: → integer
- ▶ Argument 1 (0): oberstes Stackelement
- ▶ Argument 2 (1): um 1 erhöhen

► Prefixe/Suffixe: Datentypen

i integer, l long, s short, b byte, c character

f float, d double, z boolean, a reference

```
0 fcml
```


▶ Prefixe/Suffixe: Datentypen

i integer, l long, s short, b byte, c character
f float, d double, z boolean, a reference

```
0 fcmpl
```

▶ Suffixe (speziell): const, load, store + `_n`

```
0 iconst_0    // 03                push int 0 to stack
0 iconst_m1   // 02                push int -1 to stack
1 sipush 999  // 11 03 E7            push signed int 999 to stack
```

- ▶ An Bytegrenzen ausgerichtet
- ▶ JVM-Stack Slotgröße: 4 byte
 - ▶ integer, float, byte, short: 4 byte/1 Slot
 - ▶ long, double: 8 byte/2 Slots

Bytecode Allgemein

Architektur

Aufbau

Instruktionsgruppen

Beispiel

► Laden/Speichern

aload_0, istore

- ▶ Laden/Speichern
- ▶ Arithmetische/logische Operationen

`aload_0, istore`

`ladd, fcmpl`

- ▶ Laden/Speichern
- ▶ Arithmetische/logische Operationen
- ▶ Typkonversion

```
aload_0, istore  
ladd, fcmpl  
i2b, d2i
```

- ▶ Laden/Speichern
- ▶ Arithmetische/logische Operationen
- ▶ Typkonversion
- ▶ Objekterzeugung und -manipulierung

```
aload_0, istore  
ladd, fcmpl  
i2b, d2i  
new, putfield
```

- ▶ Laden/Speichern
- ▶ Arithmetische/logische Operationen
- ▶ Typkonversion
- ▶ Objekterzeugung und -manipulierung
- ▶ Operandenstapelmanagement

```
aload_0, istore  
ladd, fcmpl  
i2b, d2i  
new, putfield  
swap, dup2
```


- ▶ Laden/Speichern
- ▶ Arithmetische/logische Operationen
- ▶ Typkonversion
- ▶ Objekterzeugung und -manipulierung
- ▶ Operandenstapelmanagement
- ▶ Kontrollübertragung

```
aload_0, istore  
ladd, fcmpl  
i2b, d2i  
new, putfield  
swap, dup2  
ifeq, goto
```

- ▶ Laden/Speichern `aload_0, istore`
- ▶ Arithmetische/logische Operationen `ladd, fcmpl`
- ▶ Typkonversion `i2b, d2i`
- ▶ Objekterzeugung und -manipulierung `new, putfield`
- ▶ Operandenstapelmanagement `swap, dup2`
- ▶ Kontrollübertragung `ifeq, goto`
- ▶ Methodenausführung `invokespecial, areturn`

Bytecode Allgemein

- Architektur

- Aufbau

- Instruktionsgruppen

Beispiel

```
$ javac Example.java
```

```
$ javac Example.java  
$ javap -c -s -v -l Example
```

```
1 Constant pool:
2   #1= Methodref   #5.#15 //java/lang/Object."<init>":()V
3   #2= Fieldref    #16.#17 //java/lang/System.out:Ljava/io/PrintStream;
4   #3= Methodref   #18.#19 //java/io/PrintStream.println:(I)V
5
6 class example.Example {
7
8   example.Example();
9     0: aload_0
10    1: invokespecial #1
11    4: return
12
13   public static void main(java.lang.String[]);
14     0: iconst_0
15     1: istore_1
16     2: iload_1
17     3: iconst_5
18     4: if_icmpge    20
19     7: getstatic    #2
20    10: iload_1
21    11: invokevirtual #3
22    14: iinc         1, 1
23    17: goto        2
24    20: return
25 }
```

```
1 class Example {
2
3     public static void main(String[] args) {
4         for(int i = 0; i < 5; i++) {
5             System.out.println(i);
6         }
7     }
8
9 }
```

```
1 Classfile /data/Repositories/FUBerlin/Semester/ss13/ss13-com/Vortrag/example/Example.class
2 Last modified 17.04.2013; size 446 bytes
3 MD5 checksum 1f4fd95d93cfdb73b348333b97ca87d2
4 Compiled from "Example.java"
5 class example.Example
6   SourceFile: "Example.java"
7   minor version: 0
8   major version: 51
9   flags: ACC_SUPER
10  Constant pool:
11     #1 = Methodref          #5.#15          // java/lang/Object.<init>:()V
12     #2 = Fieldref          #16.#17          // java/lang/System.out:Ljava/io/PrintStream;
13     #3 = Methodref          #18.#19          // java/io/PrintStream.println:(I)V
14     #4 = Class             #20                  // example/Example
15     #5 = Class             #21                  // java/lang/Object
16     #6 = Utf8              <init>
17     #7 = Utf8              ()V
18     #8 = Utf8              Code
19     #9 = Utf8              LineNumberTable
20     #10 = Utf8             main
21     #11 = Utf8             ([Ljava/lang/String;)V
22     #12 = Utf8             StackMapTable
23     #13 = Utf8             SourceFile
24     #14 = Utf8             Example.java
25     #15 = NameAndType      #6:#7            // "<init>":()V
26     #16 = Class            #22                  // java/lang/System
27     #17 = NameAndType      #23:#24          // out:Ljava/io/PrintStream;
28     #18 = Class            #25                  // java/io/PrintStream
29     #19 = NameAndType      #26:#27          // println:(I)V
30     #20 = Utf8             example/Example
31     #21 = Utf8             java/lang/Object
32     #22 = Utf8             java/lang/System
33     #23 = Utf8             out
34     #24 = Utf8             Ljava/io/PrintStream;
35     #25 = Utf8             java/io/PrintStream
36     #26 = Utf8             println
37     #27 = Utf8             (I)V
38  {
39     example.Example();
40     Signature: ()V
41     flags:
42     LineNumberTable:
43         line 3: 0
```



```

44 Code:
45     stack=1, locals=1, args_size=1
46     0: aload_0
47     1: invokespecial #1           // Method java/lang/Object.<init>:()V
48     4: return
49     LineNumberTable:
50         line 3: 0
51
52 public static void main(java.lang.String[]);
53 Signature: ([Ljava/lang/String;)V
54 flags: ACC_PUBLIC, ACC_STATIC
55     LineNumberTable:
56         line 6: 0
57         line 7: 7
58         line 6: 14
59         line 9: 20
60 Code:
61     stack=2, locals=2, args_size=1
62     0: iconst_0
63     1: istore_1
64     2: iload_1
65     3: iconst_5
66     4: if_icmpge     20
67     7: getstatic     #2           // Field java/lang/System.out:Ljava/io/PrintStream;
68    10: iload_1
69    11: invokevirtual #3           // Method java/io/PrintStream.println:(I)V
70    14: iinc          1, 1
71    17: goto         2
72    20: return
73     LineNumberTable:
74         line 6: 0
75         line 7: 7
76         line 6: 14
77         line 9: 20
78     StackMapTable: number_of_entries = 2
79         frame_type = 252 /* append */
80         offset_delta = 2
81         locals = [ int ]
82         frame_type = 250 /* chop */
83         offset_delta = 17
84
85 }

```

- ▶ The Java Virtual Machine Specification, Java SE 7 Edition
<http://docs.oracle.com/javase/specs/jvms/se7/html/>
insbesondere Kapitel 6: Java Virtual Machine Instruction Set
- ▶ Java Class File Disassembler Documentation
<http://docs.oracle.com/javase/7/docs/technotes/tools/windows/javap.html>
- ▶ http://en.wikipedia.org/wiki/Java_bytecode
- ▶ <http://www.javaworld.com/jw-09-1996/jw-09-bytecodes.html>