Freie Universität Berlin

Java Bytecode

Eike Robert
Freie Universität Berlin

Softwareprojekt Übersetzerbau, 2013

Inhalt

Inhalt

Architektur

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

Inhalt

Aufbau

```
<offset> <opcode> [<args1> <, args2>]
```

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

Aufbau

```
<offset> <opcode> [<args1> <, args2>]
```

- `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

Aufbau

- Prefixe/Suffixe: Datentypen

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

```
0 fcmpl
```

Aufbau

- 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
1 sipush 999  // 11 03 E7
```

Aufbau

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

Inhalt

Instruktionsgruppen

- Laden/Speichern `aload_0, istore`

Instruktionsgruppen

- Laden/Speichern `aload_0, istore`
- Arithmentische/logische Operationen `ladd, fcmpl`

Instruktionsgruppen

- Laden/Speichern `aload_0, istore`
- Arithmentische/logische Operationen `ladd, fcmpl`
- Typkonversion `i2b, d2i`

Instruktionsgruppen

- Laden/Speichern `aload_0, istore`
- Arithmentische/logische Operationen `ladd, fcmpl`
- Typkonversion `i2b, d2i`
- Objekterzeugung und -manipulierung `new, putfield`

Instruktionsgruppen

- Laden/Speichern `aload_0, istore`
- Arithmentische/logische Operationen `ladd, fcmpl`
- Typkonversion `i2b, d2i`
- Objekterzeugung und -manipulierung `new, putfield`
- Operandenstapelmanagement `swap, dup2`

Instruktionsgruppen

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

Instruktionsgruppen

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

Inhalt

Hello World! Code


```
1  package example;
2
3  public class Example {
4
5    public static void main(String[] args) {
6      System.out.println("Hello, World!");
7    }
8
9  }
```

HelloWorld! Bytecode-Ausschnitt

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

Hello World! Bytecode-Ausschnitt

```
1  Classfile Example.class
2
3  Constant pool:
4    #1  = Methodref  #6.#15   // Object."<init>":()V
5    #2  = Fieldref   #16.#17  // System.out:PrintStream
6    #3  = String     #18      // Hello, World!
7    #4  = Methodref  #19.#20  // PrintStream.println:(String;)V
8    #18 = Utf8       Hello, World!
9
10 public class example.Example {
11
12   public example.Example();
13     0: aload_0
14     1: invokespecial #1 // Method Object."<init>":()V
15     4: return
16
17   public static void main(j.l.String[]);
18     0: getstatic     #2 // Field System.out:PrintStream
19     3: ldc           #3 // String Hello, World!
20     5: invokevirtual #4 // Method PrintStream.println:(String;)V
21     8: return
22 }
```

HelloWorld! Classfile 1

```
Classfile /data/Repositories/FUBerlin/Semester/ss13/ss13-com/Vortrag/example/Example.class
  Last modified 12.04.2013; size 429 bytes
  MD5 checksum fe91c51c057ecd33db3dc365459fb92d
  Compiled from "Example.java"
public class example.Example
  SourceFile: "Example.java"
  minor version: 0
  major version: 51
  flags: ACC_PUBLIC, ACC_SUPER
Constant pool:
   #1 = Methodref          #6.#15         //  java/lang/Object."<init>":()V
   #2 = Fieldref           #16.#17        //  java/lang/System.out:Ljava/io/PrintStream;
   #3 = String             #18            //  Hello, World!
   #4 = Methodref          #19.#20        //  java/io/PrintStream.println:(Ljava/lang/String;)V
   #5 = Class              #21            //  example/Example
   #6 = Class              #22            //  java/lang/Object
   #7 = Utf8               <init>
   #8 = Utf8               ()V
   #9 = Utf8               Code
  #10 = Utf8               LineNumberTable
  #11 = Utf8               main
  #12 = Utf8               ([Ljava/lang/String;)V
  #13 = Utf8               SourceFile
  #14 = Utf8               Example.java
  #15 = NameAndType        #7:#8          //  "<init>":()V
  #16 = Class              #23            //  java/lang/System
  #17 = NameAndType        #24:#25        //  out:Ljava/io/PrintStream;
  #18 = Utf8               Hello, World!
  #19 = Class              #26            //  java/io/PrintStream
  #20 = NameAndType        #27:#28        //  println:(Ljava/lang/String;)V
  #21 = Utf8               example/Example
  #22 = Utf8               java/lang/Object
  #23 = Utf8               java/lang/System
  #24 = Utf8               out
  #25 = Utf8               Ljava/io/PrintStream;
```

HelloWorld! Classfile 2

```
  #26 = Utf8               java/io/PrintStream
  #27 = Utf8               println
  #28 = Utf8               (Ljava/lang/String;)V
{
  public example.Example();
    Signature: ()V
    flags: ACC_PUBLIC
    LineNumberTable:
      line 3: 0
    Code:
      stack=1, locals=1, args_size=1
         0: aload_0
         1: invokespecial #1                  // Method java/lang/Object."<init>":()V
         4: return
      LineNumberTable:
        line 3: 0

  public static void main(java.lang.String[]);
    Signature: ([Ljava/lang/String;)V
    flags: ACC_PUBLIC, ACC_STATIC
    LineNumberTable:
      line 6: 0
      line 7: 8
    Code:
      stack=2, locals=1, args_size=1
         0: getstatic     #2                  // Field java/lang/System.out:Ljava/io/PrintStream;
         3: ldc           #3                  // String Hello, World!
         5: invokevirtual #4                  // Method java/io/PrintStream.println:(Ljava/lang/String;)V
         8: return
      LineNumberTable:
        line 6: 0
        line 7: 8
}
```

Quellen

- 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`