

JASMIN INSTRUCTIONS Jonathan Meyer, July 1996

Introduction

This document shows the syntax and the types of parameters required by each Java VM instruction in Jasmin. It also shows brief illustrative examples.

See <u>The Jasmin User Guide</u> for a description of other aspects of the Jasmin syntax.

Local variable instructions

```
The following instructions use local variables:
```

```
ret <var-num>
aload <var-num>
astore <var-num>
dload <var-num>
dstore <var-num>
```

fload <var-num> fstore <var-num> iload <var-num> istore <var-num>

lload <var-num>

1store <var-num> for example: ; push local variable 1 onto the stack aload 1 ; return to the address held in local variable 2 ret 2

The bipush, sipush and iinc instructions The bipush and sipush instructions take an integer as a parameter:

bipush <int> sipush <int>

for example: ; push 100 onto the stack bipush 100

The iinc instruction takes two integer parameters: iinc <var-num> <amount>

for example:

; subtract 10 from local variable 3 iinc 3 -10

Branch instructions

The following instructions take a label as a parameter:

goto <label> goto_w <label> if_acmpeq <label>

if_acmpne <label> if_icmpeq <label> if_icmpge <label> if_icmpgt <label> if_icmple <label> if_icmplt <label> if_icmpne <label>

ifeq <label>

ifge <label> ifgt <label> ifle <label> iflt <label> ifne <label> ifnonnull <label> ifnull <label> jsr <label> jsr_w <label>

For example:

Label1:

new

For example:

Class and object operations The following instructions take a class name as a parameter: anewarray <class> checkcast <class>

invokestatic

invokevirtual

class

instanceof <class>

goto Label1

new java/lang/String ; create a new String object Method invokation

The following instructions are used to invoke methods:

invokenonvirtual <method-spec>

<method-spec>

<method-spec>

<class>

for example:

; invokes java.io.PrintStream.println(String); invokevirtual java/io/PrintStream/println(Ljava/lang/String;)V

foo/baz/Myclass/myMethod(Ljava/lang/String;)V

method

A special case is invokeinterface, which takes a <method-spec> and an integer indicating how many arguments the method takes:

descriptor

A method specification is formed of three parts: the characters before the last '/' form the class name. The characters between the last '/' and '(' are the method name. The rest of the string is the descriptor.

; jump to the code at Label1

; (an infinite loop!)

for example: invokeinterface foo/Baz/myMethod(I)V 1 Field manipulation instructions

The four instructions getfield, getstatic, putfield and putstatic have the form:

invokeinterface <method-spec> <num-args>

getfield <field-spec> <descriptor> getstatic <field-spec> <descriptor> putfield <field-spec> <descriptor>

putstatic <field-spec> <descriptor>

for example: ; get java.lang.System.out, which is a PrintStream getstatic java/lang/System/out Ljava/io/PrintStream;

after the last '/'. For example: foo/baz/AnotherClass/anotherFunField -- class name ----- --field name --

<descriptor> is the Java type descriptor of the field. For example:

Ljava/io/PrintStream; The newarray instruction

<field-spec> is composed of two parts, a classname and a fieldname. The classname is all of the characters in the <field-spec> up to the last '/' character, and the fieldname is the rest of the characters

newarray <array-type> for example:

The newarray instruction is followed by the type of the array,

newarray int newarray short newarray float

The multianewarray instruction takes two parameters, the type descriptor for the array and the number of dimensions to allocate:

The multianewarray instruction

multianewarray [[[I 2

ldc 1.2

ldc "Hello World"

ldc_w 3.141592654

<lookupswitch> ::= lookupswitch

ldc 10

The ldc and ldc_w instructions The ldc and ldc w instructions are followed by a constant:

ldc <constant> ldc_w <constant> <constant> is either an integer, a floating point number, or a quoted string. For example:

; push a float

; push an int

; push a String

; push PI as a double

multianewarray <array-descriptor> <num-dimensions>

The lookupswitch instruction

The lookupswitch instruction has the syntax:

<int1> : <label1> <int2> : <label2>

default : <default-label> For example: ; If the int on the stack is 3, jump to Label1.

; If it is 5, jump to Label2. ; Otherwise jump to DefaultLabel.

lookupswitch 3 : Label1 5 : Label2 default : DefaultLabel

Label1: ... got 3 Label2: ... got 5

... got something else

The tableswitch instruction

The tableswitch instruction has the syntax:

<tableswitch> ::= tableswitch <low> <label1> <label2> default : <default-label>

DefaultLabel:

For example:

; If it is 1, jump to Label2. ; Otherwise jump to DefaultLabel.

tableswitch 0 Label1 Label2 default : DefaultLabel Label1:

... got 0

No parameter

Label2: ... got 1 DefaultLabel: ... got something else

; If the int on the stack is 0, jump to Label1.

daload dastore dcmpg dcmpl dconst_0 dconst_1 ddiv dload_0 dload_1 dload_2 dload_3 dmul dneg drem dreturn dstore_0 dstore_1 dstore_2 dstore_3 dsub dup dup2 dup2_x1 dup2_x2 dup_x1 dup_x2 f2d f2i f2l fadd faload fastore fcmpg fcmpl fconst_0 fconst_1 fconst_2 fdiv

aaload aastore aconst_null aload_0 aload_1 aload_2 aload_3 areturn arraylength astore_0 astore 1 astore 2 astore 3 athrow baload bastore breakpoint caload castore d2f d2i d2l dadd

The following instructions (the majority) take no parameters:

lshr lstore_0 lstore_1 lstore_2 lstore_3 lsub lushr lxor monitorenter monitorexit nop pop pop2 return saload sastore swap for example: ; remove the top item from the stack ; push 1 onto the stack iconst 1

; swap the top two items on the stack swap Copyright (c) Jonathan Meyer, July 1996 Jasmin Home | Jon Meyer's Home

fload_0 fload_1 fload_2 fload_3 fmul fneg frem freturn fstore_0 fstore_1 fstore_2 fstore_3 fsub i2d i2f i2l iadd iaload iand iastore iconst_0 iconst_1 iconst_2 iconst_3 iconst_4 iconst_5 iconst_m1 idiv iload_0 iload_1 iload_2 iload_3 imul ineg int2byte int2char int2short ior irem ireturn ishl ishr istore_0 istore_1 istore_2 istore_3 isub iushr ixor l2d l2f l2i ladd laload land lastore lcmp lconst_0 lconst_1 ldiv lload_0 lload_1 lload_2 lload_3 lmul lneg lor lrem lreturn lshl