Dec
121
123
55
63
[illegible]
[illegible]
66
101
[illegible]
131
194
195
197
187
188
0
87
88
181
179
169
[illegible]7
53
86
[illegible]
170
196

A.3 Instructions by Category

This section contains a listing of the instructions and what they do, grouped by category. Here is a key to the descriptions:

a	The top slot of the stack. May be an `int`, `float`, or `reference`.
b	The second stack slot. May be an `int`, `float`, or `reference`.
c	The third stack slot. May be an `int`, `float`, or `reference`.
d	The fourth stack slot. May be an `int`, `float`, or `reference`.
ab	The `long` or `double` on top of the stack, made up of slots `a` and `b`.
cd	The `long` or `double` second on the stack, made up of slots `c` and `d`.

A.3.1 Arithmetic

Mnemonic	Arguments	Description
`dadd`		Add `double` (ab+cd)
`dcmpg`		Compare `double`
`dcmpl`		Compare `double`
`ddiv`		Divide `double` (ab/cd)
`dmul`		Multiply `double` (ab*cd)
`dneg`		Negate `double` (-ab)
`drem`		Remainder `double` (ab%cd)
`dsub`		Subtract `double` (ab-cd)
`fadd`		Add `float` (a+b)
`fcmpg`		Compare `float`
`fcmpl`		Compare `float`
`fdiv`		Divide `float` (a/b)
`fmul`		Multiply `float` (a*b)
`fneg`		Negate `float` (-a)
`frem`		Remainder `float` (a%b)
`fsub`		Subtract `float` (a-b)
`i2b`		Convert `int` to `byte`
`i2c`		Convert `int` to `char`
`i2s`		Convert `int` to `short`

Mnemonic	Arguments	Description
iadd		Add int (a+b)
iand		Bitwise and ints (a & b)
idiv		Divide ints (a/b)
imul		Multiply ints (a*b)
ineg		Negate int (-a)
ior		Bitwise or ints (a | b)
irem		Remainder int (a%b)
ishl		Shift int left (a << b)
ishr		Shift int right (a >> c)
isub		Subtract int (a-b)
iushr		Unsigned shift int right (a >>> c)
ixor		Bitwise xor ints (a ^ b)
ladd		Add long (ab+cd)
land		Bitwise and longs (ab & cd)
lcmp		Compare longs
ldiv		Divide long (ab/cd)
lmul		Multiply long (ab*cd)
lneg		Negate long (-ab)
lor		Bitwise or longs (ab | cd)
lrem		Remainder longs (ab%cd)
lshl		Shift long left (bc << 9)
lshr		Shift long right (bc >> 9)
lsub		Subtract long (ab-cd)
lushr		Unsigned shift long right (bc >>> 9)
lxor		Bitwise x or longs (ab ^ cd)

A.3.2 Array

Mnemonic	Arguments	Description
aaload		Push array element a from array b
aastore		Store a in array element b of array c
anewarray	class	Create array of class, length a
arraylength		Length of array a
athrow		Throw exception a

Mnemonic	Arguments	Description
baload		Push array element a from array b
bastore		Store a in array element b of array c
caload		Push array element a from array b
castore		Store a in array element
daload		Push array element a from array b
dastore		Store ab in array element
faload		Push array element a from array b
fastore		Store a in array element b of array c
iaload		Push array element a from array b
iastore		Store a in array element
laload		Push array element a from array b
lastore		Store ab in array element c of array d
multianewarray	class n	Create multidimensional array with first n dimensions initialized to lengths a, b, c. . .
newarray	type	Create array of type, length a
saload		Push array element a from array b
sastore		Store a in array element b of array c

A.3.3 Constant

Mnemonic	Arguments	Description
aconst_null		Push null reference
bipush	n	Push int between –128 and 127
dconst_0		Push 0 (double)
dconst_1		Push 1 (double)
fconst_0		Push 0 (float)
fconst_1		Push 1 (float)
fconst_2		Push 2 (float)
iconst_0		Push 0 (int)
iconst_1		Push 1 (int)
iconst_2		Push 2 (int)
iconst_3		Push 3 (int)
iconst_4		Push 4 (int)
iconst_5		Push 5 (int)

Mnemonic	Arguments	Description
iconst_m1		Push –1(int)
lconst_0		Push 0 (long)
lconst_1		Push 1 (long)
ldc	x	Push x (a constant int, float, or String)
ldc_w	x	Push x (a constant int, float, or String)
ldc2_w	x	Push x (a constant long or double)
sipush	n	Push int between –32,768 and 32767

A.3.4 Control

Mnemonic	Arguments	Description
areturn		Return reference from method
dreturn		Return double from method
freturn		Return float from method
goto	label	Branch always
goto_w	label	Branch always to label
if_acmpeq	label	Branch if a == b
if_acmpne	label	Branch if a != b
if_icmpeq	label	Branch if a > b
if_icmpge	label	Branch if a >= b
if_icmpgt	label	Branch if a > b
if_icmple	label	Branch if a <= b
if_icmplt	label	Branch if a < b
if_icmpne	label	Branch if a != b
ifeq	label	Branch if a == 0
ifge	label	Branch if a >= 0
ifgt	label	Branch if a > 0
ifle	label	Branch if a <= 0
iflt	label	Branch if a < 0
ifne	label	Branch if a != 0
ifnonnull	label	Branch if a is not null
ifnull	label	Branch if a is null

Mnemonic	Arguments	Description
ireturn		Return int from method
jsr	label	Branch to label; push return location
jsr_w	label	Jump to label
lookupswitch	tag1: label1 tag2: label2... default: label*n*	Branch to label1 on tag1, label2 on tag2, . . . , label*n* otherwise
lreturn		Return long from method
nop		Do nothing
ret	n	Branch to location in variable n
return		Return from method
tableswitch	n label1 label2 . . . default: label*n*	Branch to label1 on n, label2 on n+1, . . . , label*n* otherwise

A.3.5 Data Type Conversion

Mnemonic	Arguments	Description
d2f		Convert double ab to float
d2i		Convert double ab to int
d2l		Convert double ab to long
f2d		Convert float a to double
f2i		Convert float a to int
f2l		Convert float a to long
i2d		Convert int a to double
i2f		Convert int a to float
i2l		Convert int a to long
l2d		Convert long ab to double
l2f		Convert long ab to float
l2i		Convert long ab to int

A.3.6 Object

Mnemonic	Arguments	Description
checkcast	class	Throw exception if a is not an instance of class
getfield	class/field desc	Push object field from object a
getstatic	class/field desc	Push static field
instanceof	class	Push 1 if a is class, 0 otherwise
invokeinterface	class/method desc n	Invoke method through interface with n argument words
invokespecial	class/method desc	Invoke method directly
invokestatic	class/method desc	Invoke static method
invokevirtual	class/method desc	Invoke method virtually
new	class	Create new object of class
putfield	class/field desc	Store a in object field
putstatic	class/field desc	Store a in static field

A.3.7 Stack Manipulation

Mnemonic	Arguments	Description
dup		Duplicate a
dup_x1		Duplicate a, insert under b
dup_x2		Duplicate a, insert under c
dup2		Duplicate ab
dup2_x1		Duplicate ab, insert under c
dup2_x2		Duplicate ab, insert under d
pop		Remove a
pop2		Remove ab
swap		Swap a and b

A.3.8 Synchronization

Mnemonic	Arguments	Description
monitorenter		Gain control of monitor of a
monitorexit		Release monitor of a

A.3.9 Variable

Mnemonic	Arguments	Description
aload	n	Push local variable n
aload_0		Push local variable 0
aload_1		Push local variable 1
aload_2		Push local variable 2
aload_3		Push local variable 3
astore	n	Store a in local variable n
astore_0		Store a in local variable 0
astore_1		Store a in local variable 1
astore_2		Store a in local variable 2
astore_3		Store a in local variable 3
dload	n	Push local variable n
dload_0		Push local variable 0
dload_1		Push local variable 1
dload_2		Push local variable 2
dload_3		Push local variable 3
dstore	n	Store ab in local variable n
dstore_0		Store ab in local variable 0
dstore_1		Store ab in local variable 1
dstore_2		Store ab in local variable 2
dstore_3		Store ab in local variable 3
fload	n	Push local variable n
fload_0		Push local variable 0
fload_1		Push local variable 1
fload_2		Push local variable 2
fload_3		Push local variable 3
fstore	n	Store a in local variable n
fstore_0		Store a in local variable 0
fstore_1		Store a in local variable 1
fstore_2		Store a in local variable 2
fstore_3		Store a in local variable 3
iinc	n increment	Increment local variable
iload	n	Push local variable n
iload_0		Push local variable 0
iload_1		Push local variable 1
iload_2		Push local variable 2

Mnemonic	Arguments	Description
iload_3		Push local variable 3
istore	n	Store a in local variable n
istore_0		Store a in local variable 0
istore_1		Store a in local variable 1
istore_2		Store a in local variable 2
istore_3		Store a in local variable 3
lload	n	Push local variable n
lload_0		Push local variable 0
lload_1		Push local variable 1
lload_2		Push local variable 2
lload_3		Push local variable 3
lstore	n	Store ab in local variable n
lstore_0		Store ab in local variable 0
lstore_1		Store ab in local variable 1
lstore_2		Store ab in local variable 2
lstore_3		Store ab in local variable 3
wide	instruction arguments	Like instruction, except using wider range of values