No.	Instruction	Translation Rules
1	aaload	load reference [ps] from array [ps]
2	aastore	store [ps] into reference array [ps]
3	aconst_null	push null
4	aload	load reference [ps] from loacl variable [pv]
5	aload_<n>	load reference [ps] from loacl variable [pv]
6	anewarray	create new array of reference [pv]
7	areturn	return reference [ps] from method
8	arraylength	get length of array [pv]
9	astore	store reference [ps] into local variable [pv]
10	astore_<n>	store reference [ps] into local variable [pv]
11	athrow	throw exception or error from [pv]
12	baload	load int [ps] from [pv]
13	bastore	store [ps] into int array [ps]
14	bipush	push int [pc]
15	caload	load char [ps] from [ps]
16	castore	store [ps] into char array [ps]
17	checkcast	check whether [ps] is of given type
18	d2f	convert double [ps] to float
19	d2i	convert double [ps] to int
20	d2l	convert double [ps] to long
21	dadd	double result is double [ps] add double [ps]; push double result
22	daload	load double [ps] from array [ps]
23	dastore	store [ps] into double array [ps]
24	dcmp<op>	compare double [ps] and [ps]
25	dconst_<d>	push double constant [pc]
26	ddiv	double result is double [ps] divided by double [ps]; push double result
27	dload	load double [ps] from local variable [pv]
28	dload_<n>	load double [ps] from local variable [pv]
29	dmul	double result is double [ps] multiply double [ps]; push double result
30	dneg	double result is negate double [ps]; push double result
31	drem	double result is double [ps] remainder double [ps]; push double result
32	dreturn	return double [ps] from method
33	dstore	store double [ps] into local variable [pv]
34	dstore_<n>	store double [ps] into local variable [pv]
35	dsub	double [ps] substract double [ps]; push double result
36	dup	duplicate [px]
37	dup_x1	duplicate [px]
38	dup_x2	duplicate [px]
39	dup2	duplicate [ps] and [ps]/duplicate [ps]

40	dup2_x1	duplicate [ps] and [ps]/duplicate [ps]
41	dup2_x2	duplicate [ps] and [ps]/duplicate [ps]
42	f2d	convert float [ps] to double
43	f2i	convert float [ps] to int
44	f2l	convert float [ps] to long
45	fadd	float result is float [ps] add float [ps]; push float result
46	faload	load float [ps] from array [ps]
47	fastore	store [ps] into float array [ps]
48	fcmp<op>	compare float [ps] and [ps]
49	fconst_<f>	push float constant [pc]
50	fdiv	float result is float [ps] divided by float [ps]; push float result
51	fload	load float [ps] from local variable [pv]
52	fload_<n>	load float [ps] from local variable [pv]
53	fmul	float result is float [ps] multiply float [ps]; push float result
54	fneg	float result is negate float [ps]; push float result
55	frem	float result is float [ps] remainder float [ps]; push float result
56	freturn	return float [ps] from method
57	fstore	store float [ps] into local variable [pv]
58	fstore_<n>	store float [ps] into local variable [pv]
59	fsub	float result is float [ps] subtract float [ps]; push float result
60	getfield	fetch field from object [ps]
61	getstatic	get static field from class [ps]
62	goto	go to [pi]
63	goto_w	go to [pi]
64	i2b	convert int [ps] to byte
65	i2c	convert int [ps] to char
66	i2d	convert int [ps] to double
67	i2f	convert int [ps] to float
68	i2l	convert int [ps] to long
69	i2s	convert int [ps] to short
70	iadd	int result is int [ps] add int [ps]; push int result
71	iaload	load int [ps] from array [ps]
72	iand	take bitwise AND of int [ps] and int [ps]
73	iastore	store into [ps] int array [ps]
74	iconst_<i>	push int constant [pc]
75	idiv	int result is int [ps] divided by int [ps]; push int result
76	if_acmpeq	if int [ps] is equal to int [ps] then go to [pi]
77	if_acmpne	if int [ps] is not equal to int [ps] then go to [pi]
78	if_icmpeq	if int [ps] is equal to int [ps] then go to [pi]
79	if_icmpne	if int [ps] is not equal to int [ps] then go to [pi]
80	if_icmplt	if int [ps] is less then int [ps] then go to [pi]
81	if_icmple	if int [ps] is less or equal to int [ps] then go to [pi]
82	if_icmpgt	if int [ps] is greater than int [ps] then go to [pi]

83	if_icmpge	if int [ps] is greater or equal to int [ps] then go to [pi]
84	ifeq	if int [ps] is equal to 0 then go to [pi]
85	ifne	if int [ps] is not equal to 0 then go to [pi]
86	iflt	if int [ps] is less then 0 then go to [pi]
87	ifle	if int [ps] is less or equal to 0 then go to [pi]
88	ifgt	if int [ps] is greater than 0 then go to [pi]
89	ifge	if int [ps] is greater or equal to 0 then go to [pi]
90	ifnonnull	if reference [ps] is not null then go to [pi]
91	ifnull	if reference [ps] is null then go to [pi]
92	iinc	increment local variable [pv] by constant [pc]
93	iload	load int [ps] from local variable [pv]
94	iload_<n>	load int [ps] from local variable [pv]
95	imul	int result is int [ps] multiply int [ps]; push int result
96	ineg	int result is negate int [ps]; push int result
97	instanceof	object [ps] is of given type [ps]
98	invokedynamic	invoke dynamic method [ps]
99	invokeinterface	invoke interface method [ps]
100	invokespecial	invoke instance superclass, private or instance initialization method [ps]
101	invokestatic	invoke class static method [ps]
102	invokevirtual	invoke instance method [ps]
103	ior	take bitwise inclusive OR of int [ps] and [ps]
104	irem	int result is int [ps] remainder int [ps]; push int result
105	ireturn	return int [ps] from method
106	ishl	int result is int [ps] shift [ps] left; push int result
107	ishr	int result is int [ps] arithmetic shift [ps] right; push int result
108	istore	store int [ps] into local variable [pv]
109	istore_<n>	store int [ps] into local variable [pv]
110	isub	int result is int [ps] subtract int [ps]; push int result
111	iushr	int result is int [ps] logical shift [ps] right; push int result
112	ixor	take bitwise exclusive OR of int [ps] and [ps]
113	jsr	jump subroutine [pi]
114	jsr_w	jump subroutine [pi]
115	l2d	convert long [ps] to double
116	l2f	convert long [ps] to float
117	l2i	convert long [ps] to int
118	ladd	long result is long [ps] add long [ps]; push long result.
119	laload	load long [ps] from array [ps]
120	land	take bitwise AND of long [ps] and [ps]
121	lastore	store [ps] into long array [ps]
122	lcmp	compare long [ps] and [ps]
123	lconst_<l>	push long constant [pc]
124	ldc	push [ps] from run-time constant pool

125	ldc_w	push [ps] from run-time constant pool
126	ldc2_w	push long or double [ps] from run-time constant pool
127	ldiv	long result is long [ps] divided by long [ps]; push long result
128	lload	load long [ps] from local variable [pv]
129	lload_<n>	load long [ps] from local variable [pv]
130	lmul	long result is long [ps] multiply long [ps]; push long result
131	lneg	long result is negate long [ps]; push long result
132	lookupswitch	if [ps] is equal [pc] then go to [pi] ...
133	lor	take bitwise inclusive OR of long [ps] and [ps]
134	lrem	long result is long [ps] remainder long [ps]; push long result
135	lreturn	return long [ps] from method
136	lshl	long result is long [ps] shift [ps] left; push long result
137	lshr	long result is long [ps] arithmetic shift [ps] right; push long result
138	lstore	store long [ps] into local variable [pv]
139	lstore_<n>	store long [ps] into local variable [pv]
140	lsub	long result is long [ps] subtract long [ps]; push long result
141	lushr	long result is long [ps] logical shift [ps] right; push long result
142	lxor	take bitwise exclusive OR of long [ps] and [ps]
143	monitorenter	enter monitor for object [ps]
144	monitorexit	exit monitor for object [ps]
145	multianewarray	create new [ps] multidimensional array
146	new	create new object [ps]
147	newarray	create new [ps] array
148	nop	do nothing
149	pop	pop [ps]
150	pop2	pop [ps] and [ps]
151	putfield	set field [ps] in object [pv]
152	putstatic	set static [ps] in class [pv]
153	ret	return from subroutine [pi]
154	return	return void from method
155	saload	load short [ps] from array [ps]
156	sastore	store [ps] into short array [ps]
157	sipush	push int [pc]
158	swap	swap [ps] and [ps]
159	tableswitch	if [ps] is equal [pc] then go to [pi] ...
160	wide	extend local variable index by additional bytes