A. Appendix

A.1. Bytecode instructions

This section gives a short description of the bytecode instructions. For a complete description, see the Java Virtual Machine Specification.

Conventions: a and b represent `int`, `float`, `long` or `double` values (*e.g.*, they mean `int` for `IADD` but `long` for `LADD`), o and p represent objet references, v represents any value (or, for stack instructions, a value of size 1), w represents a `long` or `double`, and i, j and n represent `int` values.

Local variables

Instruction	Stack before	Stack after
`ILOAD`, `LLOAD`, `FLOAD`, `DLOAD` *var*	...	... , a
`ALOAD` *var*	...	... , o
`ISTORE`, `LSTORE`, `FSTORE`, `DSTORE` *var*	... , a	...
`ASTORE` *var*	... , o	...
`IINC` *var incr*	...	...

Stack

`POP`	... , v	...
`POP2`	... , v_1 , v_2	...
	... , w	...
`DUP`	... , v	... , v , **v**
`DUP2`	... , v_1 , v_2	... , v_1 , v_2 , $\mathbf{v_1}$, $\mathbf{v_2}$
	... , w	... , w, **w**
`SWAP`	... , v_1 , v_2	... , $v_{\mathbf{2}}$, $v_{\mathbf{1}}$
`DUP_X1`	... , v_1 , v_2	... , $\mathbf{v_2}$, v_1 , v_2

DUP_X2	… , v_1 , v_2 , v_3	… , $\mathbf{v_3}$, v_1 , v_2 , v_3
	… , w , v	… , $\mathbf{v}$, w , v
DUP2_X1	… , v_1 , v_2 , v_3	… , $\mathbf{v_2}$, $\mathbf{v_3}$, v_1 , v_2 , v_3
	… , v , w	… , $\mathbf{w}$, v , w
DUP2_X2	… , v_1 , v_2 , v_3 , v_4	… , $\mathbf{v_3}$, $\mathbf{v_4}$, v_1 , v_2 , v_3 , v_4
	… , w , v_1 , v_2	… , $\mathbf{v_1}$, $\mathbf{v_2}$, w , v_1 , v_2
	…. , v_1 , v_2 , w	… , $\mathbf{w}$, v_1 , v_2 , w
	… , w_1 , w_2	… , $\mathbf{w_2}$, w_1 , w_2

Constants

ICONST_*n* $(-1 \leq n \leq 5)$	…	… , n
LCONST_*n* $(0 \leq n \leq 1)$	…	… , nL
FCONST_*n* $(0 \leq n \leq 2)$	…	… , nF
DCONST_*n* $(0 \leq n \leq 1)$	…	… , nD
BIPUSH b, $-128 \leq b < 127$	…	… , b
SIPUSH s, $-32768 \leq s < 32767$	…	… , s
LDC *cst* (int, float, long, double, String or Type)	…	… , *cst*
ACONST_NULL	…	… , null

Arithmetic and logic

IADD, LADD, FADD, DADD	… , a , b	… , a + b
ISUB, LSUB, FSUB, DSUB	… , a , b	… , a - b
IMUL, LMUL, FMUL, DMUL	… , a , b	… , a * b
IDIV, LDIV, FDIV, DDIV	… , a , b	… , a / b
IREM, LREM, FREM, DREM	… , a , b	… , a % b
INEG, LNEG, FNEG, DNEG	… , a	… , -a
ISHL, LSHL	… , a , n	… , a << n
ISHR, LSHR	… , a , n	… , a >> n
IUSHR, LUSHR	… , a , n	… , a >>> n
IAND, LAND	… , a , b	… , a & b
IOR, LOR	… , a , b	… , a | b
IXOR, LXOR	… , a , b	… , a ^ b
LCMP	… , a , b	… , a == b ? 0 : (a < b ? -1 : 1)

`FCMPL`, `FCMPG`	... , a , b	... , a == b ? 0 : (a < b ? -1 : 1)
`DCMPL`, `DCMPG`	... , a , b	... , a == b ? 0 : (a < b ? -1 : 1)

Casts

`I2B`	... , i	... , `(byte)` i
`I2C`	... , i	... , `(char)` i
`I2S`	... , i	... , `(short)` i
`L2I`, `F2I`, `D2I`	... , a	... , `(int)` a
`I2L`, `F2L`, `D2L`	... , a	... , `(long)` a
`I2F`, `L2F`, `D2F`	... , a	... , `(float)` a
`I2D`, `L2D`, `F2D`	... , a	... , `(double)` a
`CHECKCAST` *class*	... , o	... , (*class*) o

Objects, fields and methods

`NEW` *class*	...	... , `new` *class*
`GETFIELD` *c f t*	... , o	... , o.*f*
`PUTFIELD` *c f t*	... , o , v	...
`GETSTATIC` *c f t*	...	... , *c*.*f*
`PUTSTATIC` *c f t*	... , v	...
`INVOKEVIRTUAL` *c m t*	... , o , v_1 , ... , v_n	... , o.*m*(v_1, ... v_n)
`INVOKESPECIAL` *c m t*	... , o , v_1 , ... , v_n	... , o.*m*(v_1, ... v_n)
`INVOKESTATIC` *c m t*	... , v_1 , ... , v_n	... , *c*.*m*(v_1, ... v_n)
`INVOKEINTERFACE` *c m t*	... , o , v_1 , ... , v_n	... , o.*m*(v_1, ... v_n)
`INVOKEDYNAMIC` *m t bsm*	... , o , v_1 , ... , v_n	... , o.*m*(v_1, ... v_n)
`INSTANCEOF` *class*	... , o	... , o `instanceof` *class*
`MONITORENTER`	... , o	...
`MONITOREXIT`	... , o	...

Arrays

`NEWARRAY` *type* (for any primitive type)	... , n	... , `new` *type*[n]
`ANEWARRAY` *class*	... , n	... , `new` *class*[n]
`MULTIANEWARRAY` *[...[t n*	... , i_1 , ... , i_n	... , `new` *t*[i_1]...[i_n]...

BALOAD, CALOAD, SALOAD	... , o , i	... , o[i]
IALOAD, LALOAD, FALOAD, DALOAD	... , o , i	... , o[i]
AALOAD	... , o , i	... , o[i]
BASTORE, CASTORE, SASTORE	... , o , i , j	...
IASTORE, LASTORE, FASTORE, DASTORE	... , o , i , a	...
AASTORE	... , o , i , p	...
ARRAYLENGTH	... , o	... , o.length

Jumps

IFEQ	... , i	...	jump if i == 0
IFNE	... , i	...	jump if i != 0
IFLT	... , i	...	jump if i < 0
IFGE	... , i	...	jump if i >= 0
IFGT	... , i	...	jump if i > 0
IFLE	... , i	...	jump if i <= 0
IF_ICMPEQ	... , i , j	...	jump if i == j
IF_ICMPNE	... , i , j	...	jump if i != j
IF_ICMPLT	... , i , j	...	jump if i < j
IF_ICMPGE	... , i , j	...	jump if i >= j
IF_ICMPGT	... , i , j	...	jump if i > j
IF_ICMPLE	... , i , j	...	jump if i <= j
IF_ACMPEQ	... , o , p	...	jump if o == p
IF_ACMPNE	... , o , p	...	jump if o != p
IFNULL	... , o	...	jump if o == null
IFNONNULL	... , o	...	jump if o != null
GOTO	...	...	jump always
TABLESWITCH	... , i	...	jump always
LOOKUPSWITCH	... , i	...	jump always

Return

IRETURN, LRETURN, FRETURN, DRETURN	... , a	
ARETURN	... , o	
RETURN	...	
ATHROW	... , o	