附录 B　虚拟机字节码指令表

字节码	助记符	指令含义
0x00	nop	什么都不做
0x01	aconst_null	将 null 推送至栈顶
0x02	iconst_m1	将 int 型 -1 推送至栈顶
0x03	iconst_0	将 int 型 0 推送至栈顶
0x04	iconst_1	将 int 型 1 推送至栈顶
0x05	iconst_2	将 int 型 2 推送至栈顶
0x06	iconst_3	将 int 型 3 推送至栈顶
0x07	iconst_4	将 int 型 4 推送至栈顶
0x08	iconst_5	将 int 型 5 推送至栈顶
0x09	lconst_0	将 long 型 0 推送至栈顶
0x0a	lconst_1	将 long 型 1 推送至栈顶
0x0b	fconst_0	将 float 型 0 推送至栈顶
0x0c	fconst_1	将 float 型 1 推送至栈顶
0x0d	fconst_2	将 float 型 2 推送至栈顶
0x0e	dconst_0	将 double 型 0 推送至栈顶
0x0f	dconst_1	将 double 型 1 推送至栈顶
0x10	bipush	将单字节的常量值（-128~127）推送至栈顶
0x11	sipush	将一个短整型常量值（-32768~32767）推送至栈顶
0x12	ldc	将 int、float 或 String 型常量值从常量池中推送至栈顶
0x13	ldc_w	将 int、float 或 String 型常量值从常量池中推送至栈顶（宽索引）
0x14	ldc2_w	将 long 或 double 型常量值从常量池中推送至栈顶（宽索引）
0x15	iload	将指定的 int 型本地变量推送至栈顶
0x16	lload	将指定的 long 型本地变量推送至栈顶
0x17	fload	将指定的 float 型本地变量推送至栈顶
0x18	dload	将指定的 double 型本地变量推送至栈顶
0x19	aload	将指定的引用类型本地变量推送至栈顶
0x1a	iload_0	将第一个 int 型本地变量推送至栈顶
0x1b	iload_1	将第二个 int 型本地变量推送至栈顶
0x1c	iload_2	将第三个 int 型本地变量推送至栈顶
0x1d	iload_3	将第四个 int 型本地变量推送至栈顶
0x1e	lload_0	将第一个 long 型本地变量推送至栈顶
0x1f	lload_1	将第二个 long 型本地变量推送至栈顶

（续）

字节码	助记符	指令含义
0x20	lload_2	将第三个 long 型本地变量推送至栈顶
0x21	lload_3	将第四个 long 型本地变量推送至栈顶
0x22	fload_0	将第一个 float 型本地变量推送至栈顶
0x23	fload_1	将第二个 float 型本地变量推送至栈顶
0x24	fload_2	将第三个 float 型本地变量推送至栈顶
0x25	fload_3	将第四个 float 型本地变量推送至栈顶
0x26	dload_0	将第一个 double 型本地变量推送至栈顶
0x27	dload_1	将第二个 double 型本地变量推送至栈顶
0x28	dload_2	将第三个 double 型本地变量推送至栈顶
0x29	dload_3	将第四个 double 型本地变量推送至栈顶
0x2a	aload_0	将第一个引用类型本地变量推送至栈顶
0x2b	aload_1	将第二个引用类型本地变量推送至栈顶
0x2c	aload_2	将第三个引用类型本地变量推送至栈顶
0x2d	aload_3	将第四个引用类型本地变量推送至栈顶
0x2e	iaload	将 int 型数组指定索引的值推送至栈顶
0x2f	laload	将 long 型数组指定索引的值推送至栈顶
0x30	faload	将 float 型数组指定索引的值推送至栈顶
0x31	daload	将 double 型数组指定索引的值推送至栈顶
0x32	aaload	将引用型数组指定索引的值推送至栈顶
0x33	baload	将 boolean 或 byte 型数组指定索引的值推送至栈顶
0x34	caload	将 char 型数组指定索引的值推送至栈顶
0x35	saload	将 short 型数组指定索引的值推送至栈顶
0x36	istore	将栈顶 int 型数值存入指定本地变量
0x37	lstore	将栈顶 long 型数值存入指定本地变量
0x38	fstore	将栈顶 float 型数值存入指定本地变量
0x39	dstore	将栈顶 double 型数值存入指定本地变量
0x3a	astore	将栈顶引用型数值存入指定本地变量
0x3b	istore_0	将栈顶 int 型数值存入第一个本地变量
0x3c	istore_1	将栈顶 int 型数值存入第二个本地变量
0x3d	istore_2	将栈顶 int 型数值存入第三个本地变量
0x3e	istore_3	将栈顶 int 型数值存入第四个本地变量
0x3f	lstore_0	将栈顶 long 型数值存入第一个本地变量
0x40	lstore_1	将栈顶 long 型数值存入第二个本地变量
0x41	lstore_2	将栈顶 long 型数值存入第三个本地变量
0x42	lstore_3	将栈顶 long 型数值存入第四个本地变量
0x43	fstore_0	将栈顶 float 型数值存入第一个本地变量
0x44	fstore_1	将栈顶 float 型数值存入第二个本地变量
0x45	fstore_2	将栈顶 float 型数值存入第三个本地变量

（续）

字节码	助记符	指令含义
0x46	fstore_3	将栈顶 float 型数值存入第四个本地变量
0x47	dstore_0	将栈顶 double 型数值存入第一个本地变量
0x48	dstore_1	将栈顶 double 型数值存入第二个本地变量
0x49	dstore_2	将栈顶 double 型数值存入第三个本地变量
0x4a	dstore_3	将栈顶 double 型数值存入第四个本地变量
0x4b	astore_0	将栈顶引用型数值存入第一个本地变量
0x4c	astore_1	将栈顶引用型数值存入第二个本地变量
0x4d	astore_2	将栈顶引用型数值存入第三个本地变量
0x4e	astore_3	将栈顶引用型数值存入第四个本地变量
0x4f	iastore	将栈顶 int 型数值存入指定数组的指定索引位置
0x50	lastore	将栈顶 long 型数值存入指定数组的指定索引位置
0x51	fastore	将栈顶 float 型数值存入指定数组的指定索引位置
0x52	dastore	将栈顶 double 型数值存入指定数组的指定索引位置
0x53	aastore	将栈顶引用型数值存入指定数组的指定索引位置
0x54	bastore	将栈顶 boolean 或 byte 型数值存入指定数组的指定索引位置
0x55	castore	将栈顶 char 型数值存入指定数组的指定索引位置
0x56	sastore	将栈顶 short 型数值存入指定数组的指定索引位置
0x57	pop	将栈顶数值弹出（数值不能是 long 或 double 类型的）
0x58	pop2	将栈顶的一个（对于 long 或 double 类型）或两个数值（对于非 long 或 double 的其他类型）弹出
0x59	dup	复制栈顶数值并将复制值压入栈顶
0x5a	dup_x1	复制栈顶数值并将两个复制值压入栈顶
0x5b	dup_x2	复制栈顶数值并将三个（或两个）复制值压入栈顶
0x5c	dup2	复制栈顶一个（对于 long 或 double 类型）或两个（对于非 long 或 double 的其他类型）数值并将复制值压入栈顶
0x5d	dup2_x1	dup_x1 指令的双倍版本
0x5e	dup2_x2	dup_x2 指令的双倍版本
0x5f	swap	将栈最顶端的两个数值互换（数值不能是 long 或 double 类型）
0x60	iadd	将栈顶两 int 型数值相加并将结果压入栈顶
0x61	ladd	将栈顶两 long 型数值相加并将结果压入栈顶
0x62	fadd	将栈顶两 float 型数值相加并将结果压入栈顶
0x63	dadd	将栈顶两 double 型数值相加并将结果压入栈顶
0x64	isub	将栈顶两 int 型数值相减并将结果压入栈顶
0x65	lsub	将栈顶两 long 型数值相减并将结果压入栈顶
0x66	fsub	将栈顶两 float 型数值相减并将结果压入栈顶
0x67	dsub	将栈顶两 double 型数值相减并将结果压入栈顶
0x68	imul	将栈顶两 int 型数值相乘并将结果压入栈顶
0x69	lmul	将栈顶两 long 型数值相乘并将结果压入栈顶

（续）

字节码	助记符	指令含义
0x6a	fmul	将栈顶两 float 型数值相乘并将结果压入栈顶
0x6b	dmul	将栈顶两 double 型数值相乘并将结果压入栈顶
0x6c	idiv	将栈顶两 int 型数值相除并将结果压入栈顶
0x6d	ldiv	将栈顶两 long 型数值相除并将结果压入栈顶
0x6e	fdiv	将栈顶两 float 型数值相除并将结果压入栈顶
0x6f	ddiv	将栈顶两 double 型数值相除并将结果压入栈顶
0x70	irem	将栈顶两 int 型数值作取模运算并将结果压入栈顶
0x71	lrem	将栈顶两 long 型数值作取模运算并将结果压入栈顶
0x72	frem	将栈顶两 float 型数值作取模运算并将结果压入栈顶
0x73	drem	将栈顶两 double 型数值作取模运算并将结果压入栈顶
0x74	ineg	将栈顶 int 型数值取负并将结果压入栈顶
0x75	lneg	将栈顶 long 型数值取负并将结果压入栈顶
0x76	fneg	将栈顶 float 型数值取负并将结果压入栈顶
0x77	dneg	将栈顶 double 型数值取负并将结果压入栈顶
0x78	ishl	将 int 型数值左移位指定位数并将结果压入栈顶
0x79	lshl	将 long 型数值左移位指定位数并将结果压入栈顶
0x7a	ishr	将 int 型数值右（带符号）移位指定位数并将结果压入栈顶
0x7b	lshr	将 long 型数值右（带符号）移位指定位数并将结果压入栈顶
0x7c	iushr	将 int 型数值右（无符号）移位指定位数并将结果压入栈顶
0x7d	lushr	将 long 型数值右（无符号）移位指定位数并将结果压入栈顶
0x7e	iand	将栈顶两 int 型数值作“按位与”并将结果压入栈顶
0x7f	land	将栈顶两 long 型数值作“按位与”并将结果压入栈顶
0x80	ior	将栈顶两 int 型数值作“按位或”并将结果压入栈顶
0x81	lor	将栈顶两 long 型数值作“按位或”并将结果压入栈顶
0x82	ixor	将栈顶两 int 型数值作“按位异或”并将结果压入栈顶
0x83	lxor	将栈顶两 long 型数值作“按位异或”并将结果压入栈顶
0x84	iinc	将指定 int 型变量增加指定值（如 i++、i--、i+=2 等）
0x85	i2l	将栈顶 int 型数值强制转换成 long 型数值并将结果压入栈顶
0x86	i2f	将栈顶 int 型数值强制转换成 float 型数值并将结果压入栈顶
0x87	i2d	将栈顶 int 型数值强制转换成 double 型数值并将结果压入栈顶
0x88	l2i	将栈顶 long 型数值强制转换成 int 型数值并将结果压入栈顶
0x89	l2f	将栈顶 long 型数值强制转换成 float 型数值并将结果压入栈顶
0x8a	l2d	将栈顶 long 型数值强制转换成 double 型数值并将结果压入栈顶
0x8b	f2i	将栈顶 float 型数值强制转换成 int 型数值并将结果压入栈顶
0x8c	f2l	将栈顶 float 型数值强制转换成 long 型数值并将结果压入栈顶
0x8d	f2d	将栈顶 float 型数值强制转换成 double 型数值并将结果压入栈顶
0x8e	d2i	将栈顶 double 型数值强制转换成 int 型数值并将结果压入栈顶
0x8f	d2l	将栈顶 double 型数值强制转换成 long 型数值并将结果压入栈顶

（续）

字节码	助记符	指令含义
0x90	d2f	将栈顶 double 型数值强制转换成 float 型数值并将结果压入栈顶
0x91	i2b	将栈顶 int 型数值强制转换成 byte 型数值并将结果压入栈顶
0x92	i2c	将栈顶 int 型数值强制转换成 char 型数值并将结果压入栈顶
0x93	i2s	将栈顶 int 型数值强制转换成 short 型数值并将结果压入栈顶
0x94	lcmp	比较栈顶两 long 型数值的大小，并将结果（1、0 或 -1）压入栈顶
0x95	fcmpl	比较栈顶两 float 型数值的大小，并将结果（1、0 或 -1）压入栈顶；当其中一个数值为“NaN”时，将 -1 压入栈顶
0x96	fcmpg	比较栈顶两 float 型数值的大小，并将结果（1、0 或 -1）压入栈顶；当其中一个数值为“NaN”时，将 1 压入栈顶
0x97	dcmpl	比较栈顶两 double 型数值的大小，并将结果（1、0 或 -1）压入栈顶；当其中一个数值为“NaN”时，将 -1 压入栈顶
0x98	dcmpg	比较栈顶两 double 型数值的大小，并将结果（1、0 或 -1）压入栈顶；当其中一个数值为“NaN”时，将 1 压入栈顶
0x99	ifeq	当栈顶 int 型数值等于 0 时跳转
0x9a	ifne	当栈顶 int 型数值不等于 0 时跳转
0x9b	iflt	当栈顶 int 型数值小于 0 时跳转
0x9c	ifge	当栈顶 int 型数值大于或等于 0 时跳转
0x9d	ifgt	当栈顶 int 型数值大于 0 时跳转
0x9e	ifle	当栈顶 int 型数值小于或等于 0 时跳转
0x9f	if_icmpeq	比较栈顶两 int 型数值的大小，当结果等于 0 时跳转
0xa0	if_icmpne	比较栈顶两 int 型数值的大小，当结果不等于 0 时跳转
0xa1	if_icmplt	比较栈顶两 int 型数值的大小，当结果小于 0 时跳转
0xa2	if_icmpge	比较栈顶两 int 型数值的大小，当结果大于或等于 0 时跳转
0xa3	if_icmpgt	比较栈顶两 int 型数值的大小，当结果大于 0 时跳转
0xa4	if_icmple	比较栈顶两 int 型数值的大小，当结果小于或等于 0 时跳转
0xa5	if_acmpeq	比较栈顶两引用型数值，当结果相等时跳转
0xa6	if_acmpne	比较栈顶两引用型数值，当结果不相等时跳转
0xa7	goto	无条件跳转
0xa8	jsr	跳转至指定的 16 位 offset 位置，并将 jsr 的下一条指令地址压入栈顶
0xa9	ret	返回至本地变量指定的 index 的指令位置（一般与 jsr 或 jsr_w 联合使用）
0xaa	tableswitch	用于 switch 条件跳转，case 值连续（可变长度指令）
0xab	lookupswitch	用于 switch 条件跳转，case 值不连续（可变长度指令）
0xac	ireturn	从当前方法返回 int
0xad	lreturn	从当前方法返回 long
0xae	freturn	从当前方法返回 float
0xaf	dreturn	从当前方法返回 double
0xb0	areturn	从当前方法返回对象引用
0xb1	return	从当前方法返回 void
0xb2	getstatic	获取指定类的静态域，并将其值压入栈顶

（续）

字节码	助记符	指令含义
0xb3	putstatic	为指定的类的静态域赋值
0xb4	getfield	获取指定类的实例域，并将其值压入栈顶
0xb5	putfield	为指定的类的实例域赋值
0xb6	invokevirtual	调用实例方法
0xb7	invokespecial	调用超类构造方法，实例初始化方法，私有方法
0xb8	invokestatic	调用静态方法
0xb9	invokeinterface	调用接口方法
0xba	invokedynamic	调用动态方法
0xbb	new	创建一个对象，并将其引用值压入栈顶
0xbc	newarray	创建一个指定的原始类型（如 int、float、char 等）的数组，并将其引用值压入栈顶
0xbd	anewarray	创建一个引用型（如类、接口、数组）的数组，并将其引用值压入栈顶
0xbe	arraylength	获得数组的长度值并压入栈顶
0xbf	athrow	将栈顶的异常抛出
0xc0	checkcast	检验类型转换，检验未通过将抛出 ClassCastException
0xc1	instanceof	检验对象是否是指定的类的实例，如果是，则将 1 压入栈顶，否则将 0 压入栈顶
0xc2	monitorenter	获得对象的锁，用于同步方法或同步块
0xc3	monitorexit	释放对象的锁，用于同步方法或同步块
0xc4	wide	扩展本地变量的宽度
0xc5	multianewarray	创建指定类型和指定维度的多维数组（执行该指令时，操作栈中必须包含各维度的长度值），并将其引用值压入栈顶
0xc6	ifnull	为 null 时跳转
0xc7	ifnonnull	不为 null 时跳转
0xc8	goto_w	无条件跳转（宽索引）
0xc9	jsr_w	跳转至指定的 32 位 offset 位置，并将 jsr_w 的下一条指令地址压入栈顶