一、未归类系列A

此系列暂未归类。

指令码	助记符	说明
0x00	nop	什么都不做
0x01	aconst_null	将null推送至栈顶

二、const系列

该系列命令主要负责把简单的数值类型送到栈顶。该系列命令不带参数。注意只把简单的数值类型送到栈顶时，才使用如下的命令。

比如对应int型才该方式只能把-1,0,1,2,3,4,5（分别采用iconst_m1,iconst_0, iconst_1, iconst_2, iconst_3, iconst_4, iconst_5）

送到栈顶。对于int型，其他的数值请使用push系列命令（比如bipush）。

指令码	助记符	说明
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

0x0bfconst_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

0x0edconst_0 将double型(0)推送至栈顶

0x0f dconst_1 将double型(1)推送至栈顶

三、push系列

该系列命令负责把一个整形数字（长度比较小）送到到栈顶。该系列命令有一个参数，用于指定要送到栈顶的数字。

注意该系列命令只能操作一定范围内的整形数值，超出该范围的使用将使用ldc命令系列。

指令码 助记符 说明

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

四、ldc系列

该系列命令负责把数值常量或String常量值从常量池中推送至栈顶。该命令后面需要给一个表示常量在常量池中位置(编号)的参数，

哪些常量是放在常量池呢？比如：final static int id=32768;final static float double=6.5。

对于const系列命令和push系列命令操作范围之外的数值类型常量，都放在常量池中.

另外，所有不是通过new创建的String都是放在常量池中的。

指令码	助记符	说明
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）

五、load系列

5.1、load系列A

该系列命令负责把本地变量的送到栈顶。这里的本地变量不仅可以是数值类型，还可以是引用类型。

对于前四个本地变量可以采用iload_0,iload_1,iload_2,iload_3(它们分别表示第0,1,2,3个整形变量)这种不到参数的简化命令形式。

对于第4以上的本地变量将使用iload命令这种形式，在它后面给一参数，以表示是对第几个(从0开始)本类型的本地变量进行操作。

对本地变量所进行的编号，是对所有类型的本地变量进行的（并不按照类型分类）。

对于非静态函数，第一变量是this,即其对于的操作是aload_0.

还有函数传入参数也算本地变量，在进行编号时，它是先于函数体的本地变量的。

指令码	助记符	说明
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
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

5.2、load系列B

该系列命令负责把数组的某项送到栈顶。该命令根据栈里内容来确定对哪个数组的哪项进行操作。

比如，如果有成员变量：final String names[]={"robin","hb"};

那么这句话：String str=names[0];对应的指令为

17:aload_0 //将this引用推送至栈顶，即压入栈。

18:getfield#5; //Field names:[Ljava/lang/String;//将栈顶的指定的对象的第5个实例域（Field）的值（这个值可能是引用，这里就是引用）压入栈顶

21:iconst_0 //数组的索引值（下标）推至栈顶，即压入栈

22:aaload //根据栈里内容来把name数组的第一项的值推至栈顶

23:astore5 //把栈顶的值存到str变量里。因为str在我的程序中是其所在非静态函数的第5个变量(从0开始计数)，

指令码	助记符	说明
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

六、store系列

6.1、store系列A

该系列命令负责把栈顶的值存入本地变量。这里的本地变量不仅可以是数值类型，还可以是引用类型。

如果是把栈顶的值存入到前四个本地变量的话，采用的是istore_0,istore_1，istore_2，istore_3(它们分别表示第0,1,2,3个本地整形变量)这种不到参数的简化命令形式。如果是把栈顶的值存入到第四个以上本地变量的话，将使用istore命令这种形式，在它后面给一参数，以表示是把栈顶的值存入到第几个(从0开始)本地变量中。

对本地变量所进行的编号，是对所有类型的本地变量进行的（并不按照类型分类）。

对于非静态函数，第一变量是this,它是只读的.

还有函数传入参数也算本地变量，在进行编号时，它是先于函数体的本地变量的。

指令码	助记符	说明

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

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

6.2、store系列B

该系列命令负责把栈顶项的值存到数组里。该命令根据栈里内容来确定对哪个数组的哪项进行操作。

比如，如下代码:

```
int moneys[]=new int[5];

moneys[1]=100;
```

其对应的指令为:

```
  49:iconst_5
```

50:newarray int

52:astore11

54:aload11

56:iconst_1

57:bipush100

59:iastore

60:lload6　　//因为str在我的程序中是其所非静态在函数的第6个变量(从0开始计数).

指令码	助记符	说明
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

七、pop系列

该系列命令似乎只是简单对栈顶进行操作，更多详情待补充。

指令码	助记符	说明
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（long或double类型的)或两个数值弹出（其它）
0x59	dup	复制栈顶数值(数值不能是long或double类型的)并将复制值压入栈顶
0x5a	dup_x1	复制栈顶数值(数值不能是long或double类型的)并将两个复制值压入栈顶
0x5b	dup_x2	复制栈顶数值(数值不能是long或double类型的)并将三个（或两个）复制值压入栈顶
0x5c	dup2	复制栈顶一个（long或double类型的)或两个（其它）数值并将复制值压入栈顶
0x5d	dup2_x1	复制栈顶数值(long或double类型的)并将两个复制值压入栈顶
0x5e	dup2_x2	复制栈顶数值(long或double类型的)并将三个（或两个）复制值压入栈顶

八、栈顶元素数学操作及移位操作系列

该系列命令用于对栈顶元素行数学操作，和对数值进行移位操作。**移位操作的操作数和要移位的数都是从栈里取得。**

比如对于代码：int k=100;k=k>>1;其对应的JVM指令为：

60:bipush100

62:istore12//因为k在我的程序中是其所在非静态函数的第12个变量(从0开始计数).

64:iload12

66:iconst_1

67:ishr

68:istore12

指令码	助记符	说明
0x5f	swap	将栈最顶端的两个数值互换(数值不能是long或double类型的)
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
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（符号）移位指定位数并将结果压入栈顶
0x7b	lshr	将long型数值右（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

九、自增减指令

该指令用于对本地(局部)变量进行自增减操作。该指令第一参数为本地变量的编号，第二个参数为自增减的数量。

比如对于代码：

```
int d=10;
d++;
d+=2;
d--;
```

其指令为：

```
2: bipush10
4: istore_2//在我的程序中是其所在非静态函数的第2个变量(从0开始计数).
5: iinc2, 1//在我的程序中是其所在非静态函数的第2个变量(从0开始计数).
```

8: iinc2, 2

11: iinc2, -1

对本地变量所进行的编号，是对所有类型的本地变量进行的（并不按照类型分类）。

对于非静态函数，第一变量是this,它是只读的.还有函数传入参数也算本地变量，在进行编号时，它是先于函数体的本地变量的。

指令码	助记符	说明
0x84	iinc	将指定int型变量增加指定值（i++, i--, i+=2）

十、类型转化系列

该系列指令负责对栈顶数值类型进行类型转化，并把结果压入栈顶。

指令码	助记符	说明
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

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

十二、比较指令系列A

该系列指令用于对栈顶非int型元素进行比较，并把结果压入栈顶。

比如，代码：

```
void test()

{

long a=11;

long b=10;

boolean result=(a>b);

}
```

其指令为：

```
void test();

  Code:

   0:ldc2_w#16; //long 11l

   3:lstore_1

   4:ldc2_w#18; //long 10l

   7:lstore_3
```

8:lload_1

9:lload_3

10:lcmp

11:ifle18

14:iconst_1

15:goto19

18:iconst_0

19:istore5

21:return

指令码	助记符	说明
0x94	lcmp	比较栈顶两long型数值大小，并将结果（1，0，-1）压入栈顶
0x95	fcmpl	比较栈顶两float型数值大小，并将结果（1，0，-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型数值大小，并将结果（1，0，-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型数值大小，并将结果（1，0，-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型数值大小，并将结果（1，0，-1）压入栈顶；当其中一个数值为NaN时，将1压入栈顶

十三、有条件跳转指令系列A

该系列指令用于对栈顶int型元素进行比较，根据结果进行跳转。第一个参数为要跳转到的代码的地址（这里的地址是指其指令在函数内是第几个指令）。注意对于boolean型，其实是把它当做int型来处理的。另外对于引用比较使用的时，其实是对存储的对象的地址进行比较。

比如代码：

```
void test()

{

int a=11;

int b=10;

boolean result=(a>b);

if(result)

a+=2;

if(!result)

a+=2;

if(a>0)
```

a--;

}

其对应的指令为：

void test();

Code:

0:bipush11

2:istore_1

3:bipush10

5:istore_2

6:iload_1

7:iload_2

8:if_icmple15//如果比较结果小于0，就跳到第15个指令继续执行

11:iconst_1

12:goto16

15:iconst_0

16:istore_3

17:iload_3

18:ifeq24//如果结果为0时(即为false)，就跳转到第24个指令继续执行

21:iinc1, 2

24:iload_3

25:ifne31//如果结果不为0时(即为true)，就跳转到第31个指令继续执行

28:iinc1, 2

31:iload_1

32:ifle38

35:iinc1, -1//如果结果小于0时，就跳转到第38个指令继续执行

38:return

指令码	助记符	说明
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型数值小于等于0时跳转

0x9f　　if_icmpeq　比较栈顶两int型数值大小，当结果等于0时跳转

0xa0　　if_icmpne　比较栈顶两int型数值大小，当结果不等于0时跳转

0xa1　　if_icmplt　比较栈顶两int型数值大小，当结果小于0时跳转

0xa2　　if_icmpge　比较栈顶两int型数值大小，当结果大于等于0时跳转

0xa3　　if_icmpgt　比较栈顶两int型数值大小，当结果大于0时跳转

0xa4　　if_icmple　比较栈顶两int型数值大小，当结果小于等于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

十四、无条件跳转指令系列A

该系列指令用于指令的跳转。

指令码	助记符	说明
0xa7	goto	无条件跳转
0xa8	jsr	跳转至指定16位offset位置，并将jsr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, 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）

十五、返回指令系列

该系列指令用于从函数中返回。如果有返回值的话，都把函数的返回值放在栈道中，以便它的调用方法取得它。

return 10;这个语句其实对应的指令是两条：

9:bipush10

11:ireturn

指令码	助记符	说明
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

十六、域操作指令系列

该系列指令用于对静态域和非静态域进行读写。该系列命令需要跟一个表明域编号的参数，

比如,在函数中对成员变量m进行;m++

其指令为：

0:aload_0

1:dup

2:getfield#2; //Field m:I

5:iconst_1

6:iadd

7:putfield#2; //Field m:I

指令码	助记符	说明
0xb2	getstatic	获取指定类的静态域，并将其值压入栈顶
0xb3	putstatic	用栈顶的值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	用栈顶的值为指定的类的实例域赋值

十七、方法操作命令系列

该系列指令用于对静态方法和非静方法进行调用。该系列命令需要跟一个表明方法编号的参数。

如果方法有传入参数的话，则需要先压栈到栈顶。另外，方法的返回参数是保存到栈顶的，因此我们可以通过栈道值取得方法的返回值。

比如对于代码：

```
void test(){int k=add(12,45);}
```

其指令为：

```
void test();
  Code:
   0:aload_0
   1:bipush12
   3:bipush45
   5:invokevirtual#2; //Method add:(II)I
   8:istore_1
   9:return
```

指令码	助记符	说明
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

十八、未归类系列B

此系列暂未归类。

指令码 助记符 说明

0xba --

十九、new及数组系列

该系列用于创建一个对象和数组。

比如代码：

```
void test()

{

int ids[]=new int[5];

Object objs[]=new Object[5];

Object obj=new Object();

Hello hello=new Hello();
```

```
int len=objs.length;

}
```

其指令为：

```
void test();

  Code:

   0:iconst_5

   1:newarray int

   3:astore_1

   4:iconst_5

   5:anewarray#2; //class java/lang/Object

   8:astore_2

  9:new#2; //class java/lang/Object

   12:dup

   13:invokespecial#1; //Method java/lang/Object."":()V

   16:astore_3

17:new#3; //class Hello
```

20:dup

21:invokespecial#4; //Method "":()V

24:astore4

26:aload_2

27:arraylength

28:istore5

30:return

指令码	助记符	说明
0xbb	new	创建一个对象，并将其引用值压入栈顶
0xbc	newarray	创建一个指定原始类型（如int, float, char...）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

二十、异常抛出指令

用于抛出异常。

指令码	助记符	说明
0xbf	athrow	将栈顶的异常抛出

二十一、对象操作指令

该系列指令用于操作对象。

指令码	助记符	说明
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

二十二、未归类系列C

此系列暂未归类。

指令码	助记符	说明
0xc4	wide	<待补充>

二十三、new多维数组系列

指令码	助记符	说明
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

二十四、有条件跳转指令系列B

该系列用于根据引用是否为空，来进行相应的指令跳转。

比如代码：

```
void test()

{

int i=0;

Object obj=new Object();

if(obj==null){i=0;}

if(obj!=null){i=1;}

}
```

其对应的指令为：

```
void test();

  Code:

    0:iconst_0

    1:istore_1

    2:new#2; //class java/lang/Object

    5:dup
```

6:invokespecial#1; //Method java/lang/Object."":()V

9:astore_2

10:aload_2

11:ifnonnull16

14:iconst_0

15:istore_1

16:aload_2

17:ifnull22

20:iconst_1

21:istore_1

22:return

指令码	助记符	说明
0xc6	ifnull	为null时跳转
0xc7	ifnonnull	不为null时跳转

二十五、无条件跳转指令系列B

该系列指令用于进行无条件指令跳转。

指令码	助记符	说明

0xc8	goto_w	无条件跳转（宽索引）
0xc9	jsr_w	跳转至指定32位offset位置，并将jsr_w下一条指令地址压入栈顶