In the following, *top* will refer to the element at the top of the operand stack, and *nexttop* will refer to the element next to it.

- **aaload**:

  Format: 0x32

  Loads an element of an array of addresses (i.e. from a multidimensional array). Treats *nexttop* as a pointer to an array (i.e. a variable declared of array type), and *top* is an index into the array. The instruction loads the array element and pushes it onto the operand stack. In other words, *nexttop* and *top* are popped, and *nexttop*[*top*] is fetched and pushed onto the operand stack.

- **aastore**:

  Format: 0x53

  Does the opposite of **aaload**, popping the operand stack first to get *value*, then to get *index*, the finally to get *address*. It then stores *value* into element *index* of the array starting at *address*.

- **aload**:

  Format: 0x19 *8bitindex*

  Pops operand stack, treats the popped value as an address, loads it into slot *8bitindex*.

- **aload_0**, **aload_1**, **aload_2**, **aload_3**:

  Formats: 0x2a, 0x2b, 0x2c

  The instruction **aload_0**, is the same as **aload**, but specifically for slot 0. The others are analogous.

- **astore**:

  Format: 0x3a, *8bitindex*

  Opposite of **aload**, popping the operand stack and placing the popped value (presumed to be an address) into the specified slot.

- **astore_0**, **astore_1**, **astore_2**, **astore_3**:

  Formats: 0x4b, 0x4c, 0x4d, 0x4e

  Same as **astore**, but specifically for slot 0, slot 1 etc..

- **bipush**:

  Format: 0x10 *8bitinteger*

  Pushes the given 8-bit integer onto the operand stack.

- **dup**:

  Format: 0x59

  Duplicates the top word on the operand stack, so the operand stack now has two copies of that word instead of one.

- **getstatic**:

  Format: 0xb2 *16bitindex*

  Opposite of **putstatic**. Gets value of the given static item, then pushes it on the operand stack.

- **iadd**:

  Format: 0x60

  Pops *nexttop* and *top*, and pushes the sum *nexttop* + *top*.

- **iaload**:

  Format: 0x2e

  Load an element of an integer array. See **aaload** above.

- **iastore**:

  Format: 0x4f

  Store to an element of an integer array. See **aastore** above.

- **iconst_0**, **iconst_1**, **iconst_2**, **iconst_3**, **iconst_4**, **iconst_5**:

  Format: 0x3, 0x4, 0x5, 0x6, 0x7, 0x8

  Pushes the integer constant 0, 1, 2 etc. onto the operand stack.

- **idiv**:

  Format: 0x6c

  Pops *nexttop* and *top*, and pushes the quotient *nexttop* / *top*.

- **if_icmpeq**:

  Format: 0x9f *jumpdistance*

  If *top* = *nexttop*, jumps the given distance to the branch target. The quantity *jumpdistance* is a 2-byte, 2s complement signed number, measured from the jump instruction. Both *top* and *nexttop* must be integers; a runtime error occurs if no.

- **if_icmpge**:

  Format: 0xa2 *jumpdistance*

  Same as **if_icmpeq**, but jumps if *top* $\geq$ *nexttop*.

- **if_icmple**:

  Format: 0xa4 *jumpdistance*

  Same as **if_icmpeq**, but jumps if *top* $\leq$ nexttop.

- **if_icmplt**:

  Format: 0xa1 *jumpdistance*

  Same as **if_icmpeq**, but jumps if *top* $<$ *nexttop*.

- **if_icmpne**:

  Format: 0xa0 *jumpdistance*

  Same as **if_icmpeq**, but jumps if *top* $\neq$ *nexttop*.

- **iinc**:

  Format: 0x84 *8bitindex 8bitinteger*

  Increments slot *8bitindex* by the amount *8bitinteger*.

- **iload**:

  Format: 0x15 *8bitindex*

  Same as **aload** but for integers instead of addresses.

- **iload_0**, **iload_1**, **iload_2**, **iload_3**:

  Formats: 0x1a, 0x1b, 0x1c, 0x1d

  Same as **aload_0** etc. but for integers instead of addresses.

- **imul**:

  Format: 0x68

  Pops *nexttop* and *top*, and pushes the product *nexttop* $\times$ *top*.

- **invokespecial**:

  Format: 0xb7 *16bitindex*

  Like **invokevirtual**, but for constructors, superclass and other special situations.

- **invokestatic**:

  Format: 0xb8 *16bitindex*

  Method call. The quantity *16bitindex* serves as an index into the Constant Pool, pointing to the given method. Creates a new stack frame for the method. Pops the method's arguments from the caller's operand stack and places them into the method's local variables section. Points the **frame** register to the method's stack frame, and jumps to the method.

- **invokevirtual**:

  Format: 0xb6 *16bitindex*

  Same as **invokestatic**, but the arguments include the "hidden" argument **this**, i.e. a pointer to the object this method is being invoked on.

- **ireturn**:

  Format: 0xac

  Return from method with return value. Pops integer from current operand stack, places it on the caller's operand stack, restores the **frame** register to point to the caller's stack frame, and jumps back to the caller.

- **istore**:

  Format: 0x36 *8bitindex*

  Same as **astore** but for integers instead of addresses.

- **istore_0**, **istore_1**, **istore_2**, **istore_3**:

  Formats: 0x3b, 0x3c, 0x3d, 0x3e

  Same as **astore_0** etc. but for integers instead of addresses.

- **isub**:

  Format: 0x64

  Pops *nexttop* and *top*, and pushes the difference *nexttop* - *top*.

- **ldc**:

  Format: 0x12 *8bitindex*

  Gets an item from the Constant Pool and pushes it onto the operand stack.

- **new**:

  Format: 0xbb *16bitindex*

  Performs the Java **new** operation, with *16bitindex* being the index into the Constant Pool, pointing to the given class. Creates the given object using memory from the Heap, and then pushes the address of the new object on the operand stack.

- **putstatic**:

  Format: 0xb3 *16bitindex*

  Pops the operand stack, and assigns the popped value to the static item given by *16bitindex*.

- **return**:

  Format: 0xb1

  Same as **ireturn**, but for methods without a return value.