Summary of Dalvik bytecode set

| **Op** | **Mnemonic / Syntax** | **Arguments** | **Description** |
| --- | --- | --- | --- |
| 00 10x | nop |  | Waste cycles. **Note:** Data-bearing pseudo-instructions are tagged with this opcode, in which case the high-order byte of the opcode unit indicates the nature of the data. See "packed-switch-payload Format", "sparse-switch-payload Format", and "fill-array-data-payload Format" below. |
| 01 12x | move vA, vB | A: destination register (4 bits) B: source register (4 bits) | Move the contents of one non-object register to another. |
| 02 22x | move/from16 vAA, vBBBB | A: destination register (8 bits) B: source register (16 bits) | Move the contents of one non-object register to another. |
| 03 32x | move/16 vAAAA, vBBBB | A: destination register (16 bits) B: source register (16 bits) | Move the contents of one non-object register to another. |
| 04 12x | move-wide vA, vB | A: destination register pair (4 bits) B: source register pair (4 bits) | Move the contents of one register-pair to another.  **Note:** It is legal to move from v*N* to eitherv*N-1* or v*N+1*, so implementations must arrange for both halves of a register pair to be read before anything is written. |
| 05 22x | move-wide/from16 vAA, vBBBB | A: destination register pair (8 bits) B: source register pair (16 bits) | Move the contents of one register-pair to another.  **Note:** Implementation considerations are the same as move-wide, above. |
| 06 32x | move-wide/16 vAAAA, vBBBB | A: destination register pair (16 bits) B: source register pair (16 bits) | Move the contents of one register-pair to another.  **Note:** Implementation considerations are the same as move-wide, above. |
| 07 12x | move-object vA, vB | A: destination register (4 bits) B: source register (4 bits) | Move the contents of one object-bearing register to another. |
| 08 22x | move-object/from16 vAA, vBBBB | A: destination register (8 bits) B: source register (16 bits) | Move the contents of one object-bearing register to another. |
| 09 32x | move-object/16 vAAAA, vBBBB | A: destination register (16 bits) B: source register (16 bits) | Move the contents of one object-bearing register to another. |
| 0a 11x | move-result vAA | A: destination register (8 bits) | Move the single-word non-object result of the most recent invoke-*kind* into the indicated register. This must be done as the instruction immediately after an invoke-*kind* whose (single-word, non-object) result is not to be ignored; anywhere else is invalid. |
| 0b 11x | move-result-wide vAA | A: destination register pair (8 bits) | Move the double-word result of the most recent invoke-*kind* into the indicated register pair. This must be done as the instruction immediately after an invoke-*kind* whose (double-word) result is not to be ignored; anywhere else is invalid. |
| 0c 11x | move-result-object vAA | A: destination register (8 bits) | Move the object result of the most recentinvoke-*kind* into the indicated register. This must be done as the instruction immediately after an invoke-*kind* or filled-new-array whose (object) result is not to be ignored; anywhere else is invalid. |
| 0d 11x | move-exception vAA | A: destination register (8 bits) | Save a just-caught exception into the given register. This must be the first instruction of any exception handler whose caught exception is not to be ignored, and this instruction must *only* ever occur as the first instruction of an exception handler; anywhere else is invalid. |
| 0e 10x | return-void |  | Return from a void method. |
| 0f 11x | return vAA | A: return value register (8 bits) | Return from a single-width (32-bit) non-object value-returning method. |
| 10 11x | return-wide vAA | A: return value register-pair (8 bits) | Return from a double-width (64-bit) value-returning method. |
| 11 11x | return-object vAA | A: return value register (8 bits) | Return from an object-returning method. |
| 12 11n | const/4 vA, #+B | A: destination register (4 bits) B: signed int (4 bits) | Move the given literal value (sign-extended to 32 bits) into the specified register. |
| 13 21s | const/16 vAA, #+BBBB | A: destination register (8 bits) B: signed int (16 bits) | Move the given literal value (sign-extended to 32 bits) into the specified register. |
| 14 31i | const vAA, #+BBBBBBBB | A: destination register (8 bits) B: arbitrary 32-bit constant | Move the given literal value into the specified register. |
| 15 21h | const/high16 vAA, #+BBBB0000 | A: destination register (8 bits) B: signed int (16 bits) | Move the given literal value (right-zero-extended to 32 bits) into the specified register. |
| 16 21s | const-wide/16 vAA, #+BBBB | A: destination register (8 bits) B: signed int (16 bits) | Move the given literal value (sign-extended to 64 bits) into the specified register-pair. |
| 17 31i | const-wide/32 vAA, #+BBBBBBBB | A: destination register (8 bits) B: signed int (32 bits) | Move the given literal value (sign-extended to 64 bits) into the specified register-pair. |
| 18 51l | const-wide vAA, #+BBBBBBBBBBBBBBBB | A: destination register (8 bits) B: arbitrary double-width (64-bit) constant | Move the given literal value into the specified register-pair. |
| 19 21h | const-wide/high16 vAA, #+BBBB000000000000 | A: destination register (8 bits) B: signed int (16 bits) | Move the given literal value (right-zero-extended to 64 bits) into the specified register-pair. |
| 1a 21c | const-string vAA, string@BBBB | A: destination register (8 bits) B: string index | Move a reference to the string specified by the given index into the specified register. |
| 1b 31c | const-string/jumbo vAA, string@BBBBBBBB | A: destination register (8 bits) B: string index | Move a reference to the string specified by the given index into the specified register. |
| 1c 21c | const-class vAA, type@BBBB | A: destination register (8 bits) B: type index | Move a reference to the class specified by the given index into the specified register. In the case where the indicated type is primitive, this will store a reference to the primitive type's degenerate class. |
| 1d 11x | monitor-enter vAA | A: reference-bearing register (8 bits) | Acquire the monitor for the indicated object. |
| 1e 11x | monitor-exit vAA | A: reference-bearing register (8 bits) | Release the monitor for the indicated object.  **Note:** If this instruction needs to throw an exception, it must do so as if the pc has already advanced past the instruction. It may be useful to think of this as the instruction successfully executing (in a sense), and the exception getting thrown*after* the instruction but *before* the next one gets a chance to run. This definition makes it possible for a method to use a monitor cleanup catch-all (e.g., finally) block as the monitor cleanup for that block itself, as a way to handle the arbitrary exceptions that might get thrown due to the historical implementation of Thread.stop(), while still managing to have proper monitor hygiene. |
| 1f 21c | check-cast vAA, type@BBBB | A: reference-bearing register (8 bits) B: type index (16 bits) | Throw a ClassCastException if the reference in the given register cannot be cast to the indicated type.  **Note:** Since A must always be a reference (and not a primitive value), this will necessarily fail at runtime (that is, it will throw an exception) if B refers to a primitive type. |
| 20 22c | instance-of vA, vB, type@CCCC | A: destination register (4 bits) B: reference-bearing register (4 bits) C: type index (16 bits) | Store in the given destination register 1 if the indicated reference is an instance of the given type, or 0 if not.  **Note:** Since B must always be a reference (and not a primitive value), this will always result in 0 being stored if C refers to a primitive type. |
| 21 12x | array-length vA, vB | A: destination register (4 bits) B: array reference-bearing register (4 bits) | Store in the given destination register the length of the indicated array, in entries |
| 22 21c | new-instance vAA, type@BBBB | A: destination register (8 bits) B: type index | Construct a new instance of the indicated type, storing a reference to it in the destination. The type must refer to a non-array class. |
| 23 22c | new-array vA, vB, type@CCCC | A: destination register (4 bits) B: size register C: type index | Construct a new array of the indicated type and size. The type must be an array type. |
| 24 35c | filled-new-array {vC, vD, vE, vF, vG}, type@BBBB | A: array size and argument word count (4 bits) B: type index (16 bits) C..G: argument registers (4 bits each) | Construct an array of the given type and size, filling it with the supplied contents. The type must be an array type. The array's contents must be single-word (that is, no arrays of long or double, but reference types are acceptable). The constructed instance is stored as a "result" in the same way that the method invocation instructions store their results, so the constructed instance must be moved to a register with an immediately subsequent move-result-object instruction (if it is to be used). |
| 25 3rc | filled-new-array/range {vCCCC .. vNNNN}, type@BBBB | A: array size and argument word count (8 bits) B: type index (16 bits) C: first argument register (16 bits) N = A + C - 1 | Construct an array of the given type and size, filling it with the supplied contents. Clarifications and restrictions are the same as filled-new-array, described above. |
| 26 31t | fill-array-data vAA, +BBBBBBBB *(with supplemental data as specified below in "*fill-array-data-payload*Format")* | A: array reference (8 bits) B: signed "branch" offset to table data pseudo-instruction (32 bits) | Fill the given array with the indicated data. The reference must be to an array of primitives, and the data table must match it in type and must contain no more elements than will fit in the array. That is, the array may be larger than the table, and if so, only the initial elements of the array are set, leaving the remainder alone. |
| 27 11x | throw vAA | A: exception-bearing register (8 bits) | Throw the indicated exception. |
| 28 10t | goto +AA | A: signed branch offset (8 bits) | Unconditionally jump to the indicated instruction.  **Note:** The branch offset must not be 0. (A spin loop may be legally constructed either with goto/32 or by including a nopas a target before the branch.) |
| 29 20t | goto/16 +AAAA | A: signed branch offset (16 bits) | Unconditionally jump to the indicated instruction.  **Note:** The branch offset must not be 0. (A spin loop may be legally constructed either with goto/32 or by including a nopas a target before the branch.) |
| 2a 30t | goto/32 +AAAAAAAA | A: signed branch offset (32 bits) | Unconditionally jump to the indicated instruction. |
| 2b 31t | packed-switch vAA, +BBBBBBBB *(with supplemental data as specified below in "*packed-switch-payload*Format")* | A: register to test B: signed "branch" offset to table data pseudo-instruction (32 bits) | Jump to a new instruction based on the value in the given register, using a table of offsets corresponding to each value in a particular integral range, or fall through to the next instruction if there is no match. |
| 2c 31t | sparse-switch vAA, +BBBBBBBB *(with supplemental data as specified below in "*sparse-switch-payload*Format")* | A: register to test B: signed "branch" offset to table data pseudo-instruction (32 bits) | Jump to a new instruction based on the value in the given register, using an ordered table of value-offset pairs, or fall through to the next instruction if there is no match. |
| 2d..31 23x | cmp*kind* vAA, vBB, vCC 2d: cmpl-float *(lt bias)* 2e: cmpg-float *(gt bias)* 2f: cmpl-double *(lt bias)* 30: cmpg-double *(gt bias)* 31: cmp-long | A: destination register (8 bits) B: first source register or pair C: second source register or pair | Perform the indicated floating point or longcomparison, setting a to 0 if b == c, 1 if b > c, or -1 if b < c. The "bias" listed for the floating point operations indicates how NaNcomparisons are treated: "gt bias" instructions return 1 for NaN comparisons, and "lt bias" instructions return -1.  For example, to check to see if floating point x < y it is advisable to use cmpg-float; a result of -1 indicates that the test was true, and the other values indicate it was false either due to a valid comparison or because one of the values was NaN. |
| 32..37 22t | if-*test* vA, vB, +CCCC 32: if-eq 33: if-ne 34: if-lt 35: if-ge 36: if-gt 37: if-le | A: first register to test (4 bits) B: second register to test (4 bits) C: signed branch offset (16 bits) | Branch to the given destination if the given two registers' values compare as specified.  **Note:** The branch offset must not be 0. (A spin loop may be legally constructed either by branching around a backwardgoto or by including a nop as a target before the branch.) |
| 38..3d 21t | if-*test*z vAA, +BBBB 38: if-eqz 39: if-nez 3a: if-ltz 3b: if-gez 3c: if-gtz 3d: if-lez | A: register to test (8 bits) B: signed branch offset (16 bits) | Branch to the given destination if the given register's value compares with 0 as specified.  **Note:** The branch offset must not be 0. (A spin loop may be legally constructed either by branching around a backwardgoto or by including a nop as a target before the branch.) |
| 3e..43 10x | *(unused)* |  | *(unused)* |
| 44..51 23x | *arrayop* vAA, vBB, vCC 44: aget 45: aget-wide 46: aget-object 47: aget-boolean 48: aget-byte 49: aget-char 4a: aget-short 4b: aput 4c: aput-wide 4d: aput-object 4e: aput-boolean 4f: aput-byte 50: aput-char 51: aput-short | A: value register or pair; may be source or dest (8 bits) B: array register (8 bits) C: index register (8 bits) | Perform the identified array operation at the identified index of the given array, loading or storing into the value register. |
| 52..5f 22c | i*instanceop* vA, vB, field@CCCC 52: iget 53: iget-wide 54: iget-object 55: iget-boolean 56: iget-byte 57: iget-char 58: iget-short 59: iput 5a: iput-wide 5b: iput-object 5c: iput-boolean 5d: iput-byte 5e: iput-char 5f: iput-short | A: value register or pair; may be source or dest (4 bits) B: object register (4 bits) C: instance field reference index (16 bits) | Perform the identified object instance field operation with the identified field, loading or storing into the value register.  **Note:** These opcodes are reasonable candidates for static linking, altering the field argument to be a more direct offset. |
| 60..6d 21c | s*staticop* vAA, field@BBBB 60: sget 61: sget-wide 62: sget-object 63: sget-boolean 64: sget-byte 65: sget-char 66: sget-short 67: sput 68: sput-wide 69: sput-object 6a: sput-boolean 6b: sput-byte 6c: sput-char 6d: sput-short | A: value register or pair; may be source or dest (8 bits) B: static field reference index (16 bits) | Perform the identified object static field operation with the identified static field, loading or storing into the value register.  **Note:** These opcodes are reasonable candidates for static linking, altering the field argument to be a more direct offset. |
| 6e..72 35c | invoke-*kind* {vC, vD, vE, vF, vG}, meth@BBBB 6e: invoke-virtual 6f: invoke-super 70: invoke-direct 71: invoke-static 72: invoke-interface | A: argument word count (4 bits) B: method reference index (16 bits) C..G: argument registers (4 bits each) | Call the indicated method. The result (if any) may be stored with an appropriatemove-result\* variant as the immediately subsequent instruction.  invoke-virtual is used to invoke a normal virtual method (a method that is notprivate, static, or final, and is also not a constructor).  When the method\_id references a method of a non-interface class, invoke-super is used to invoke the closest superclass's virtual method (as opposed to the one with the same method\_id in the calling class). The same method restrictions hold as forinvoke-virtual.  In Dex files version 037 or later, if themethod\_id refers to an interface method,invoke-super is used to invoke the most specific, non-overridden version of that method defined on that interface. The same method restrictions hold as forinvoke-virtual. In Dex files prior to version037, having an interface method\_id is illegal and undefined.  invoke-direct is used to invoke a non-static direct method (that is, an instance method that is by its nature non-overridable, namely either a privateinstance method or a constructor).  invoke-static is used to invoke a staticmethod (which is always considered a direct method).  invoke-interface is used to invoke aninterface method, that is, on an object whose concrete class isn't known, using amethod\_id that refers to an interface.  **Note:** These opcodes are reasonable candidates for static linking, altering the method argument to be a more direct offset (or pair thereof). |
| 73 10x | *(unused)* |  | *(unused)* |
| 74..78 3rc | invoke-*kind*/range {vCCCC .. vNNNN}, meth@BBBB 74: invoke-virtual/range 75: invoke-super/range 76: invoke-direct/range 77: invoke-static/range 78: invoke-interface/range | A: argument word count (8 bits) B: method reference index (16 bits) C: first argument register (16 bits) N = A + C - 1 | Call the indicated method. See first invoke-*kind* description above for details, caveats, and suggestions. |
| 79..7a 10x | *(unused)* |  | *(unused)* |
| 7b..8f 12x | *unop* vA, vB 7b: neg-int 7c: not-int 7d: neg-long 7e: not-long 7f: neg-float 80: neg-double 81: int-to-long 82: int-to-float 83: int-to-double 84: long-to-int 85: long-to-float 86: long-to-double 87: float-to-int 88: float-to-long 89: float-to-double 8a: double-to-int 8b: double-to-long 8c: double-to-float 8d: int-to-byte 8e: int-to-char 8f: int-to-short | A: destination register or pair (4 bits) B: source register or pair (4 bits) | Perform the identified unary operation on the source register, storing the result in the destination register. |
| 90..af 23x | *binop* vAA, vBB, vCC 90: add-int 91: sub-int 92: mul-int 93: div-int 94: rem-int 95: and-int 96: or-int 97: xor-int 98: shl-int 99: shr-int 9a: ushr-int 9b: add-long 9c: sub-long 9d: mul-long 9e: div-long 9f: rem-long a0: and-long a1: or-long a2: xor-long a3: shl-long a4: shr-long a5: ushr-long a6: add-float a7: sub-float a8: mul-float a9: div-float aa: rem-float ab: add-double ac: sub-double ad: mul-double ae: div-double af: rem-double | A: destination register or pair (8 bits) B: first source register or pair (8 bits) C: second source register or pair (8 bits) | Perform the identified binary operation on the two source registers, storing the result in the destination register.  **Note:** Contrary to other -longmathematical operations (which take register pairs for both their first and their second source), shl-long, shr-long, andushr-long take a register pair for their first source (the value to be shifted), but a single register for their second source (the shifting distance). |
| b0..cf 12x | *binop*/2addr vA, vB b0: add-int/2addr b1: sub-int/2addr b2: mul-int/2addr b3: div-int/2addr b4: rem-int/2addr b5: and-int/2addr b6: or-int/2addr b7: xor-int/2addr b8: shl-int/2addr b9: shr-int/2addr ba: ushr-int/2addr bb: add-long/2addr bc: sub-long/2addr bd: mul-long/2addr be: div-long/2addr bf: rem-long/2addr c0: and-long/2addr c1: or-long/2addr c2: xor-long/2addr c3: shl-long/2addr c4: shr-long/2addr c5: ushr-long/2addr c6: add-float/2addr c7: sub-float/2addr c8: mul-float/2addr c9: div-float/2addr ca: rem-float/2addr cb: add-double/2addr cc: sub-double/2addr cd: mul-double/2addr ce: div-double/2addr cf: rem-double/2addr | A: destination and first source register or pair (4 bits) B: second source register or pair (4 bits) | Perform the identified binary operation on the two source registers, storing the result in the first source register.  **Note:** Contrary to other -long/2addrmathematical operations (which take register pairs for both their destination/first source and their second source), shl-long/2addr, shr-long/2addr, and ushr-long/2addr take a register pair for their destination/first source (the value to be shifted), but a single register for their second source (the shifting distance). |
| d0..d7 22s | *binop*/lit16 vA, vB, #+CCCC d0: add-int/lit16 d1: rsub-int (reverse subtract) d2: mul-int/lit16 d3: div-int/lit16 d4: rem-int/lit16 d5: and-int/lit16 d6: or-int/lit16 d7: xor-int/lit16 | A: destination register (4 bits) B: source register (4 bits) C: signed int constant (16 bits) | Perform the indicated binary op on the indicated register (first argument) and literal value (second argument), storing the result in the destination register.  **Note:** rsub-int does not have a suffix since this version is the main opcode of its family. Also, see below for details on its semantics. |
| d8..e2 22b | *binop*/lit8 vAA, vBB, #+CC d8: add-int/lit8 d9: rsub-int/lit8 da: mul-int/lit8 db: div-int/lit8 dc: rem-int/lit8 dd: and-int/lit8 de: or-int/lit8 df: xor-int/lit8 e0: shl-int/lit8 e1: shr-int/lit8 e2: ushr-int/lit8 | A: destination register (8 bits) B: source register (8 bits) C: signed int constant (8 bits) | Perform the indicated binary op on the indicated register (first argument) and literal value (second argument), storing the result in the destination register.  **Note:** See below for details on the semantics of rsub-int. |
| e3..f9 10x | *(unused)* |  | *(unused)* |
| fa 45cc | invoke-polymorphic {vC, vD, vE, vF, vG}, meth@BBBB, proto@HHHH | A: argument word count (4 bits)  B: method reference index (16 bits)  C: method handle reference to invoke (16 bits)  D..G: argument registers (4 bits each)  H: prototype reference index (16 bits) | Invoke the indicated method handle. The result (if any) may be stored with an appropriate move-result\* variant as the immediately subsequent instruction.  The method reference must be tojava.lang.invoke.MethodHandle.invoke orjava.lang.invoke.MethodHandle.invokeExact.  The prototype reference describes the argument types provided and the expected return type.  The invoke-polymorphic bytecode may raise exceptions when it executes. The exceptions are described in the API documentation forjava.lang.invoke.MethodHandle.invoke andjava.lang.invoke.MethodHandle.invokeExact.  Present in Dex files from version 038onwards. |
| fb 4rcc | invoke-polymorphic/range {vCCCC .. vNNNN}, meth@BBBB, proto@HHHH | A: argument word count (8 bits)  B: method reference index (16 bits)  C: method handle reference to invoke (16 bits)  H: prototype reference index (16 bits)  N = A + C - 1 | Invoke the indicated method handle. See the invoke-polymorphic description above for details.  Present in Dex files from version 038onwards. |
| fc 35c | invoke-custom {vC, vD, vE, vF, vG}, call\_site@BBBB | A: argument word count (4 bits)  B: call site reference index (16 bits)  C..G: argument registers (4 bits each) | Resolves and invokes the indicated call site. The result from the invocation (if any) may be stored with an appropriate move-result\* variant as the immediately subsequent instruction.  This instruction executes in two phases: call site resolution and call site invocation.  Call site resolution checks whether the indicated call site has an associatedjava.lang.invoke.CallSite instance. If not, the bootstrap linker method for the indicated call site is invoked using arguments present in the DEX file (see[call\_site\_item](https://source.android.com/devices/tech/dalvik/dex-format.html#call-site-item)). The bootstrap linker method returns ajava.lang.invoke.CallSite instance that will then be associated with the indicated call site if no association exists. Another thread may have already made the association first, and if so execution of the instruction continues with the first associated java.lang.invoke.CallSiteinstance.  Call site invocation is made on thejava.lang.invoke.MethodHandle target of the resolved java.lang.invoke.CallSiteinstance. The target is invoked as if executing invoke-polymorphic (described above) using the method handle and arguments to the invoke-custom instruction as the arguments to an exact method handle invocation.  Exceptions raised by the bootstrap linker method are wrapped in ajava.lang.BootstrapMethodError. ABootstrapMethodError is also raised if:   * the bootstrap linker method fails to return a java.lang.invoke.CallSiteinstance. * the returned java.lang.invoke.CallSitehas a null method handle target. * the method handle target is not of the requested type.   Present in Dex files from version 038onwards. |
| fd 3rc | invoke-custom/range {vCCCC .. vNNNN}, call\_site@BBBB | A: argument word count (8 bits)  B: call site reference index (16 bits)  C: first argument register (16-bits)  N = A + C - 1 | Resolve and invoke a call site. See theinvoke-custom description above for details.  Present in Dex files from version 038onwards. |
| fe..ff 10x | *(unused)* |  | *(unused)* |