# Dalvik VM Instruction Formats

Copyright © 2007 The Android Open Source Project

## Introduction and Overview

This document lists the instruction formats used by Dalvik bytecodeand is meant to be used in conjunction with thebytecode reference document.

### Bitwise descriptions

The first column in the format table lists the bitwise layout ofthe format. It consists of one or more space-separated "words" each ofwhich describes a 16-bit code unit. Each character in a wordrepresents four bits, read from high bits to low, with vertical bars("|") interspersed to aid in reading. Uppercase lettersin sequence from "A" are used to indicate fields withinthe format (which then get defined further by the syntax column). The term"op" is used to indicate the position of an eight-bitopcode within the format, and similarly "exop" is usedto indicate an extended sixteen-bit opcode. A slashed zero("Ø") is used to indicate that all bits must bezero in the indicated position.

For the most part, lettering proceeds from earlier code units tolater code units, and low-order to high-order within a code unit.However, there are a few exceptions to this general rule, which aredone in order to make the naming of similar-meaning parts be the sameacross different instruction formats. These cases are noted explicitlyin the format descriptions.

For example, the format "B|A|op CCCC" indicatesthat the format consists of two 16-bit code units. The first wordconsists of the opcode in the low eight bits and a pair of four-bitvalues in the high eight bits; and the second word consists of a single16-bit value.

### Format IDs

The second column in the format table indicates the short identifierfor the format, which is used in other documents and in code to identifythe format.

Most format IDs consist of three characters, two digits followed by aletter. The first digit indicates the number of 16-bit code units in theformat. The second digit indicates the maximum number of registers that theformat contains (maximum, since some formats can accomodate a variablenumber of registers), with the special designation "r" indicatingthat a range of registers is encoded. The final letter semi-mnemonicallyindicates the type of any extra data encoded by the format. For example,format "21t" is of length two, contains one register reference,and additionally contains a branch target.

Suggested static linking formats have an additional"s" suffix, making them four characters total. Similarly,suggested "inline" linking formats have an additional "i"suffix. (In this context, inline linking is like static linking,except with more direct ties into a virtual machine's implementation.) Finally, a couple oddball suggested formats (e.g.,"20bc") include two pieces of data which are bothrepresented in its format ID.

The full list of typecode letters are as follows. Note that someforms have different sizes, depending on the format:

| **Mnemonic** | **Bit Sizes** | **Meaning** |
| --- | --- | --- |
| b | 8 | immediate signed **b**yte |
| c | 16, 32 | **c**onstant pool index |
| f | 16 | inter**f**ace constants (only used in statically linked formats) |
| h | 16 | immediate signed **h**at (high-order bits of a 32- or 64-bit    value; low-order bits are all 0) |
| i | 32 | immediate signed **i**nt, or 32-bit float |
| l | 64 | immediate signed **l**ong, or 64-bit double |
| m | 16 | **m**ethod constants (only used in statically linked formats) |
| n | 4 | immediate signed **n**ibble |
| s | 16 | immediate signed **s**hort |
| t | 8, 16, 32 | branch **t**arget |
| x | 0 | no additional data |

### Syntax

The third column of the format table indicates the human-orientedsyntax for instructions which use the indicated format. Each instructionstarts with the named opcode and is optionally followed by one ormore arguments, themselves separated with commas.

Wherever an argument refers to a field from the first column, theletter for that field is indicated in the syntax, repeated once foreach four bits of the field. For example, an eight-bit field labeled"BB" in the first column would also be labeled"BB" in the syntax column.

Arguments which name a register have the form "vX".The prefix "v" was chosen instead of the more common"r" exactly to avoid conflicting with (non-virtual) architectureson which a Dalvik virtual machine might be implemented which themselvesuse the prefix "r" for their registers. (That is, thisdecision makes it possible to talk about both virtual and real registerstogether without the need for circumlocution.)

Arguments which indicate a literal value have the form"#+X". Some formats indicate literals that onlyhave non-zero bits in their high-order bits; for these, the zeroesare represented explicitly in the syntax, even though they do notappear in the bitwise representation.

Arguments which indicate a relative instruction address offset have theform "+X".

Arguments which indicate a literal constant pool index have the form"kind@X", where "kind"indicates which constant pool is being referred to. Each opcode thatuses such a format explicitly allows only one kind of constant; seethe opcode reference to figure out the correspondence. The fourkinds of constant pool are "string" (string pool index),"type" (type pool index), "field" (fieldpool index), and "meth" (method pool index).

Similar to the representation of constant pool indices, there arealso suggested (optional) forms that indicate prelinked offsets orindices. There are two types of suggested prelinked value: vtable offsets(indicated as "vtaboff") and field offsets (indicated as"fieldoff").

In the cases where a format value isn't explictly part of the syntaxbut instead picks a variant, each variant is listed with the prefix"[X=N]" (e.g., "[A=2]") to indicatethe correspondence.

**The Formats**

| **Format** | **ID** | **Syntax** | **Notable Opcodes Covered** |
| --- | --- | --- | --- |
| *N/A* | 00x | *N/A* | *pseudo-format used for unused opcodes; suggested for use as the    nominal format for a breakpoint opcode* |
| ØØ|*op* | 10x | *op* |  |
| B|A|*op* | 12x | *op* vA, vB |  |
| 11n | *op* vA, #+B |  |
| AA|*op* | 11x | *op* vAA |  |
| 10t | *op* +AA | goto |
| ØØ|*op* AAAA | 20t | *op* +AAAA | goto/16 |
| AA|*op* BBBB | 20bc | *op* AA, kind@BBBB | *suggested format for statically determined verification errors;    A is the type of error and B is an index into a type-appropriate    table (e.g. method references for a no-such-method error)* |
| AA|*op* BBBB | 22x | *op* vAA, vBBBB |  |
| 21t | *op* vAA, +BBBB |  |
| 21s | *op* vAA, #+BBBB |  |
| 21h | *op* vAA, #+BBBB0000 *op* vAA, #+BBBB000000000000 |  |
| 21c | *op* vAA, type@BBBB *op* vAA, field@BBBB *op* vAA, string@BBBB | check-cast    const-class    const-string |
| AA|*op* CC|BB | 23x | *op* vAA, vBB, vCC |  |
| 22b | *op* vAA, vBB, #+CC |  |
| B|A|*op* CCCC | 22t | *op* vA, vB, +CCCC |  |
| 22s | *op* vA, vB, #+CCCC |  |
| 22c | *op* vA, vB, type@CCCC *op* vA, vB, field@CCCC | instance-of |
| 22cs | *op* vA, vB, fieldoff@CCCC | *suggested format for statically linked field access instructions of    format 22c* |
| ØØ|*op* AAAAlo AAAAhi | 30t | *op* +AAAAAAAA | goto/32 |
| ØØ|*op* AAAA BBBB | 32x | *op* vAAAA, vBBBB |  |
| AA|*op* BBBBlo BBBBhi | 31i | *op* vAA, #+BBBBBBBB |  |
| 31t | *op* vAA, +BBBBBBBB |  |
| 31c | *op* vAA, string@BBBBBBBB | const-string/jumbo |
| A|G|*op* BBBB F|E|D|C | 35c | *[A=5] op* {vC, vD, vE, vF, vG},    meth@BBBB *[A=5] op* {vC, vD, vE, vF, vG},    type@BBBB *[A=4] op* {vC, vD, vE, vF},    *kind*@BBBB *[A=3] op* {vC, vD, vE},    *kind*@BBBB *[A=2] op* {vC, vD},    *kind*@BBBB *[A=1] op* {vC},    *kind*@BBBB *[A=0] op* {},    *kind*@BBBB  *The unusual choice in lettering here reflects a desire to make    the count and the reference index have the same label as in format    3rc.* |  |
| 35ms | *[A=5] op* {vC, vD, vE, vF, vG},    vtaboff@BBBB *[A=4] op* {vC, vD, vE, vF},    vtaboff@BBBB *[A=3] op* {vC, vD, vE},    vtaboff@BBBB *[A=2] op* {vC, vD},    vtaboff@BBBB *[A=1] op* {vC},    vtaboff@BBBB  *The unusual choice in lettering here reflects a desire to make    the count and the reference index have the same label as in format    3rms.* | *suggested format for statically linked invoke-virtual    and invoke-super instructions of format 35c* |
| 35mi | *[A=5] op* {vC, vD, vE, vF, vG},    inline@BBBB *[A=4] op* {vC, vD, vE, vF},    inline@BBBB *[A=3] op* {vC, vD, vE},    inline@BBBB *[A=2] op* {vC, vD},    inline@BBBB *[A=1] op* {vC},    inline@BBBB  *The unusual choice in lettering here reflects a desire to make    the count and the reference index have the same label as in format    3rmi.* | *suggested format for inline linked invoke-static    and invoke-virtual instructions of format 35c* |
| AA|*op* BBBB CCCC | 3rc | *op* {vCCCC .. vNNNN}, meth@BBBB *op* {vCCCC .. vNNNN}, type@BBBB  *where NNNN = CCCC+AA-1, that is A    determines the count 0..255, and C    determines the first register* |  |
| 3rms | *op* {vCCCC .. vNNNN}, vtaboff@BBBB  *where NNNN = CCCC+AA-1, that is A    determines the count 0..255, and C    determines the first register* | *suggested format for statically linked invoke-virtual    and invoke-super instructions of format 3rc* |
| 3rmi | *op* {vCCCC .. vNNNN}, inline@BBBB  *where NNNN = CCCC+AA-1, that is A    determines the count 0..255, and C    determines the first register* | *suggested format for inline linked invoke-static    and invoke-virtual instructions of format 3rc* |
| AA|*op* BBBBlo BBBB BBBB BBBBhi | 51l | *op* vAA, #+BBBBBBBBBBBBBBBB | const-wide |
| *exop* BB|AA CCCC | 33x | *exop* vAA, vBB, vCCCC |  |
| 32s | *exop* vAA, vBB, #+CCCC |  |
| *exop* BBBBlo BBBBhi AAAA | 40sc | *exop* AAAA, kind@BBBBBBBB | *suggested format for statically determined verification errors;    see 20bc, above* |
| *exop* BBBBlo BBBBhi AAAA | 41c | *exop* vAAAA, field@BBBBBBBB *exop* vAAAA, type@BBBBBBBB  *The unusual choice in lettering here reflects a desire to make    the letters match their use in related formats 21c and 31c.* |  |
| *exop* CCCClo CCCChi    AAAA BBBB | 52c | *exop* vAAAA, vBBBB, field@CCCCCCCC *exop* vAAAA, vBBBB, type@CCCCCCCC  *The unusual choice in lettering here reflects a desire to make    the letters match their use in related formats 22c and 22cs.* |  |
| *exop* BBBBlo BBBBhi    AAAA CCCC | 5rc | *exop* {vCCCC .. vNNNN}, meth@BBBBBBBB *exop* {vCCCC .. vNNNN}, type@BBBBBBBB  *where NNNN = CCCC+AAAA-1, that is A    determines the count 0..65535, and C    determines the first register*  *The unusual choice in lettering here reflects a desire to make    the letters match their use in related formats 3rc, 3rms, and 3rmi.* |  |