Dalvik Operation Codes

Opcode (hex)	Opcode name	Explanation	Example
00	nop	No operation	0000 - nop
01	move vx,vy	Moves the content of vy into vx. Both registers must be in the first 256 register range.	0110 - move v0, v1 Moves v1 into v0.
02	move/from16 vx,vy	Moves the content of vy into vx. vy may be in the 64k register range while vx is one of the first 256 registers.	0200 1900 - move/from16 v0, v25 Moves v25 into v0.
03	move/16		
04	move-wide		
05	move-wide/from16 vx,vy	Moves a long/double value from vy to vx. vy may be in the 64k register range while wx is one of the first 256 registers.	0516 0000 - move-wide/from16 v... Moves v0 into v22.
06	move-wide/16		
07	move-object vx,vy	Moves the object reference from vy to vx.	0781 - move-object v1, v8 Moves the object reference in v8
08	move-object/from16 vx,vy	Moves the object reference from vy to vx, vy can address 64k registers and vx can address 256 registers.	0801 1500 - move-object/from16 Move the object reference in v21
09	move-object/16		
0A	move-result vx	Move the result value of the previous method invocation into vx.	0A00 - move-result v0 Move the return value of a previo invocation into v0.
0B	move-result-wide vx	Move the long/double result value of the previous method	0B02 - move-result-wide v2 Move the long/double result value method invocation into v2,v3.

		invocation into vx,vx+1.	
0C	move-result-object vx	Move the result object reference of the previous method invocation into vx.	0C00 - move-result-object v0
0D	move-exception vx	Move the exception object reference thrown during a method invocation into vx.	0D19 - move-exception v25
0E	return-void	Return without a return value	0E00 - return-void
0F	return vx	Return with vx return value	0F00 - return v0 Returns with return value in v0.
10	return-wide vx	Return with double/long result in vx,vx+1.	1000 - return-wide v0 Returns with a double/long value
11	return-object vx	Return with vx object reference value.	1100 - return-object v0 Returns with object reference valu
12	const/4 vx,lit4	Puts the 4 bit constant into vx	1221 - const/4 v1, #int2 Moves literal 2 into v1. The destir in the lower 4 bit in the second by in the higher 4 bit.
13	const/16 vx,lit16	Puts the 16 bit constant into vx	1300 0A00 - const/16 v0, #int 10 Puts the literal constant of 10 into
14	const vx, lit32	Puts the integer constant into vx	1400 4E61 BC00 - const v0, #123 #00BC614E Moves literal 12345678 into v0.
15	const/high16 v0, lit16	Puts the 16 bit constant into the topmost bits of the register. Used to initialize float values.	1500 2041 - const/high16 v0, #flo #41200000 Moves the floating literal of 10.0 i bit literal in the instruction carries the floating point number.
16	const-wide/16 vx, lit16	Puts the integer constant into vx and vx+1 registers, expanding the integer constant into a long constant..	1600 0A00 - const-wide/16 v0, #l Moves literal 10 into v0 and v1 re
17	const-wide/32 vx, lit32	Puts the 32 bit constant into vx and vx+1 registers, expanding the	1702 4e61 bc00 - const-wide/32 v 12345678 // #00bc614e Puts #12345678 into v2 and v3 re

		integer constant into a long constant.	
18	const-wide vx, lit64	Puts the 64 bit constant into vx and vx+1 registers.	1802 874b 6b5d 54dc 2b00- cons 12345678901234567 // #002bdc5 Puts #12345678901234567 into registers.
19	const-wide/high16 vx,lit16	Puts the 16 bit constant into the highest 16 bit of vx and vx+1 registers. Used to initialize double values.	1900 2440 - const-wide/high16 v(#402400000 Puts the double constant of 10.0
1A	const-string vx,string_id	Puts reference to a string constant identified by string_id into vx.	1A08 0000 - const-string v8, "" // Puts reference to string@0000 (e string table) into v8.
1B	const-string-jumbo		
1C	const-class vx,type_id	Moves the class object of a class identified by type_id (e.g. Object.class) into vx.	1C00 0100 - const-class v0, Test Moves reference to Test3.class (e type id table) into
1D	monitor-enter vx	Obtains the monitor of the object referenced by vx.	1D03 - monitor-enter v3 Obtains the monitor of the object v3.
1E	monitor-exit	Releases the monitor of the object referenced by vx.	1E03 - monitor-exit v3 Releases the monitor of the objec v3.
1F	check-cast vx, type_id	Checks whether the object reference in vx can be cast to an instance of a class referenced by type_id. Throws ClassCastException if the cast is not possible, continues execution otherwise.	1F04 0100 - check-cast v4, Test3 Checks whether the object refere cast to type@0001 (entry #1 in th
20	instance-of vx,vy,type_id	Checks whether vy is instance of a class identified by type_id. Sets vx non-zero if it is, 0 otherwise.	2040 0100 - instance-of v0, v4, Te type@0001 Checks whether the object refere instance of type@0001 (entry #1 table). Sets v0 to non-zero if v4 is Test3, 0 otherwise.
21	array-length vx,vy	Calculates the number of elements of the array referenced by vy	2111 - array-length v1, v1 Calculates the number of elemen referenced by v1 and puts the res

		and puts the length value into vx.	
22	new-instance vx,type	Instantiates an object type and puts the reference of the newly created instance into vx.	2200 1500 - new-instance v0, java.io.FileInputStream // type@0 Instantiates type@0015 (entry #1 table) and puts its reference into
23	new-array vx,vy,type_id	Generates a new array of type_id type and vy element size and puts the reference to the array into vx.	2312 2500 - new-array v2, v1, ch type@0025 Generates a new array of type@0 size and puts the reference to the v2.
24	filled-new-array {parameters},type_id	Generates a new array of type_id and fills it with the parameters[5]. Reference to the newly generated array can be obtained by a move-result-object instruction, immediately following the filled-new-array instruction.	2420 530D 0000 - filled-new-array type@0D53 Generates a new array of type@0 array's size will be 2 and both ele filled with the contents of v0 regis
25	filled-new-array-range {vx..vy},type_id	Generates a new array of type_id and fills it with a range of parameters. Reference to the newly generated array can be obtained by a move-result-object instruction, immediately following the filled-new-array instruction.	2503 0600 1300 - filled-new-array {v19..v21}, [B // type@0006 Generates a new array of type@0 array's size will be 3 and the elem filled using the v19,v20 and v21 r
26	fill-array-data vx,array_data_offset	Fills the array referenced by vx with the static data. The location of the static data is the sum of the position of the current instruction and the offset	2606 2500 0000 - fill-array-data v Fills the array referenced by v0 w data at current instruction+25H w The offset is expressed as a 32-b static data is stored in the followin 0003 // Table type: static array da 0400 // Byte per array element (in byte integers) 0300 0000 // Number of elements 0100 0000 // Element #0: integer 0200 0000 // Element #1: integer 0300 0000 // Element #2: integer3

27	throw vx	Throws an exception object. The reference of the exception object is in vx.	2700 - throw v0 Throws an exception. The except reference is in v0.
28	goto target	Unconditional jump by short offset[2].	28F0 - goto 0005 // -0010 Jumps to current position-16 word 0005 is the label of the target inst
29	goto/16 target	Unconditional jump by 16 bit offset[2].	2900 0FFE - goto/16 002f // -01f1 Jumps to the current position-1F1 is the label of the target instructio
2A	goto/32 target		
2B	packed-switch vx,table	Implements a switch statement where the case constants are close to each other. The instruction uses an index table. vx indexes into this table to find the offset of the instruction for a particular case. If vx falls out of the index table, the execution continues on the next instruction (default case).	2B02 0C00 0000 - packed-switch +000c Execute a packed switch accordin argument in v2. The position of th at current instruction+0CH words like the following: 0001 // Table type: packed switch 0300 // number of elements 0000 0000 // element base 0500 0000 0: 00000005 // case 0 0700 0000 1: 00000007 // case 1 0900 0000 2: 00000009 // case 2
2C	sparse-switch vx,table	Implements a switch statement with sparse case table. The instruction uses a lookup table with case constants and offsets for each case constant. If there is no match in the table, execution continues on the next instruction (default case).	2C02 0c00 0000 - sparse-switch +000c Execute a sparse switch accordin argument in v2. The position of th at current instruction+0CH words like the following. 0002 // Table type: sparse switch 0300 // number of elements 9cff ffff // first case: -100 fa00 0000 // second case constar e803 0000 // third case constant: 0500 0000 // offset for the first ca 0700 0000 // offset for the second +7 0900 0000 // offset for the third ca
2D	cmpl-float	Compares the float values in vy and vz and sets the integer value in vx accordingly[3]	2D00 0607 - cmpl-float v0, v6, v7 Compares the float values in v6 a v0 accordingly. NaN bias is less-t instruction will return -1 if any of t NaN.
2E	cmpg-float vx, vy, vz	Compares the float values in vy and vz and sets the integer	2E00 0607 - cmpg-float v0, v6, v7 Compares the float values in v6 a v0 accordingly. NaN bias is greate

		value in vx accordingly[3].	instruction will return 1 if any of th NaN.
2F	cmpl-double vx,vy,vz	Compares the double values in vy and vz[2] and sets the integer value in vx accordingly[3].	2F19 0608 - cmpl-double v25, v6 Compares the double values in v(and sets v25 accordingly. NaN bia the instruction will return -1 if any parameters is NaN.
30	cmpg-double vx, vy, vz	Compares the double values in vy and vz[2] and sets the integer value in vx accordingly[3].	3000 080A - cmpg-double v0, v8, Compares the double values in va then sets v0 accordingly. NaN bia than, the instruction will return 1 i parameters is NaN.
31	cmp-long vx, vy, vz	Compares the long values in vy and vz and sets the integer value in vx accordingly[3].	3100 0204 - cmp-long v0, v2, v4 Compares the long values in v2 a v0 accordingly.
32	if-eq vx,vy,target	Jumps to target if vx==vy[2]. vx and vy are integer values.	32b3 6600 - if-eq v3, v11, 0080 // Jumps to the current position+66I v3==v11. 0080 is the label of the instruction.
33	if-ne vx,vy,target	Jumps to target if vx!=vy[2]. vx and vy are integer values.	33A3 1000 - if-ne v3, v10, 002c // Jumps to the current position+10I v3!=v10. 002c is the label of the t
34	if-lt vx,vy,target	Jumps to target is vx<vy[2]. vx and vy are integer values.	3432 CBFF - if-lt v2, v3, 0023 // -(Jumps to the current position-35I- 0023 is the label of the target inst
35	if-ge vx, vy,target	Jumps to target if vx>=vy[2]. vx and vy are integer values.	3510 1B00 - if-ge v0, v1, 002b // Jumps to the current position+1B v0>=v1. 002b is the label of the ta
36	if-gt vx,vy,target	Jumps to target if vx>vy[2]. vx and vy are integer values.	3610 1B00 - if-ge v0, v1, 002b // Jumps to the current position+1B v0>v1. 002b is the label of the tal
37	if-le vx,vy,target	Jumps to target if vx<=vy[2]. vx and vy are integer values.	3756 0B00 - if-le v6, v5, 0144 // + Jumps to the current position+0B v6<=v5. 0144 is the label of the ta
38	if-eqz vx,target	Jumps to target if vx==0[2]. vx is an integer value.	3802 1900 - if-eqz v2, 0038 // +0(Jumps to the current position+19I v2==0. 0038 is the label of the tar
39	if-nez vx,target	Checks vx and jumps if vx is nonzero[2].	3902 1200 - if-nez v2, 0014 // +0(Jumps to current position+18 wor is nonzero. 0014 is the label of th instruction.
3A	if-ltz vx,target	Checks vx and jumps if vx<0[2].	3A00 1600 - if-ltz v0, 002d // +001 Jumps to the current position+16I 002d is the label of the target inst

3B	if-gez vx,target	Checks vx and jumps if vx>=0^2.	3B00 1600 - if-gez v0, 002d // +00 Jumps to the current position+16l >=0. 002d is the label of the targe
3C	if-gtz vx,target	Checks vx and jumps if vx>0^2.	3C00 1D00 - if-gtz v0, 004a // +00 Jumps to the current position+1D 004A is the label of the target inst
3D	if-lez vx,target	Checks vx and jumps if vx<=0^2.	3D00 1D00 - if-lez v0, 004a // +00 Jumps to the current position+1D v0<=0. 004A is the label of the ta
3E	unused_3E		
3F	unused_3F		
40	unused_40		
41	unused_41		
42	unused_42		
43	unused_43		
44	aget vx,vy,vz	Gets an integer value of an object reference array into vx. The array is referenced by vy and is indexed by vz.	4407 0306 - aget v7, v3, v6 Gets an integer array element. Th referenced by v3 and the elemen v6. The element will be put into v
45	aget-wide vx,vy,vz	Gets a long/double value of long/double array into vx,vx+1. The array is referenced by vy and is indexed by vz.	4505 0104 - aget-wide v5, v1, v4 Gets a long/double array element referenced by v1 and the elemen v4. The element will be put into v
46	aget-object vx,vy,vz	Gets an object reference value of an object reference array into vx. The array is referenced by vy and is indexed by vz.	4602 0200 - aget-object v2, v2, v(Gets an object reference array el is referenced by v2 and the eleme v0. The element will be put into v
47	aget-boolean vx,vy,vz	Gets a boolean value of a boolean array into vx. The array is referenced by vy and is indexed by vz.	4700 0001 - aget-boolean v0, v0, Gets a boolean array element. Th referenced by v0 and the elemen v1. The element will be put into v(

48	aget-byte vx,vy,vz	Gets a byte value of a byte array into vx. The array is referenced by vy and is indexed by vz.	4800 0001 - aget-byte v0, v0, v1 Gets a byte array element. The a referenced by v0 and the elemen v1. The element will be put into v(
49	aget-char vx, vy,vz	Gets a char value of a character array into vx. The element is indexed by vz, the array object is referenced by vy	4905 0003 - aget-char v5, v0, v3 Gets a character array element. T referenced by v0 and the elemen v3. The element will be put into v!
4A	aget-short vx,vy,vz	Gets a short value of a short array into vx. The element is indexed by vz, the array object is referenced by vy.	4A00 0001 - aget-short v0, v0, v1 Gets a short array element. The a referenced by v0 and the elemen v1. The element will be put into v(
4B	aput vx,vy,vz	Puts the integer value in vx into an element of an integer array. The element is indexed by vz, the array object is referenced by vy.	4B00 0305 - aput v0, v3, v5 Puts the integer value in v2 into a referenced by v0. The target array indexed by v1.
4C	aput-wide vx,vy,vz	Puts the double/long value in vx,vx+1 into a double/long array. The array is referenced by vy, the element is indexed by vz.	4C05 0104 - aput-wide v5, v1, v4 Puts the double/long value in v5,v double/long array referenced by v array element is indexed by v4.
4D	aput-object vx,vy,vz	Puts the object reference value in vx into an element of an object reference array. The element is indexed by vz, the array object is referenced by vy.	4D02 0100 - aput-object v2, v1, v Puts the object reference value in object reference array referenced target array element is indexed by
4E	aput-boolean vx,vy,vz	Puts the boolean value in vx into an element of a boolean array. The element is indexed by vz, the array object is referenced by vy.	4E01 0002 - aput-boolean v1, v0, Puts the boolean value in v1 into reference array referenced by v0. element is indexed by v2.

4F	aput-byte vx,vy,vz	Puts the byte value in vx into an element of a byte array. The element is indexed by vz, the array object is referenced by vy.	4F02 0001 - aput-byte v2, v0, v1 Puts the boolean value in v2 into referenced by v0. The target arra indexed by v1.
50	aput-char vx,vy,vz	Puts the char value in vx into an element of a character array. The element is indexed by vz, the array object is referenced by vy.	5003 0001 - aput-char v3, v0, v1 Puts the character value in v3 into array referenced by v0. The targe is indexed by v1.
51	aput-short vx,vy,vz	Puts the short value in vx into an element of a short array. The element is indexed by vz, the array object is referenced by vy.	5102 0001 - aput-short v2, v0, v1 Puts the short value in v2 into a c referenced by v0. The target arra indexed by v1.
52	iget vx, vy, field_id	Reads an instance field into vx. The instance is referenced by vy.	5210 0300 - iget v0, v1, Test2.i6:I Reads field@0003 into v0 (entry table). The instance is referenced
53	iget-wide vx,vy,field_id	Reads an instance field into vx[1]. The instance is referenced by vy.	5320 0400 - iget-wide v0, v2, Tes field@0004 Reads field@0004 into v0 and v1 #4 in the field id table). The instar by v2.
54	iget-object vx,vy,field_id	Reads an object reference instance field into vx. The instance is referenced by vy.	iget-object v1, v2, LineReader.fis:Ljava/io/FileInputS field@0002 Reads field@0002 into v1 (entry table). The instance is referenced
55	iget-boolean vx,vy,field_id	Reads a boolean instance field into vx. The instance is referenced by vy.	55FC 0000 - iget-boolean v12, v1 field@0000 Reads the boolean field@0000 in (entry #0 in the field id table). The referenced by v15.
56	iget-byte vx,vy,field_id	Reads a byte instance field into vx. The instance is referenced by vy.	5632 0100 - iget-byte v2, v3, Test field@0001 Reads the char field@0001 into v #1 in the field id table). The instar by v3.
57	iget-char vx,vy,field_id	Reads a char instance field into vx. The instance is referenced by vy.	5720 0300 - iget-char v0, v2, Test field@0003 Reads the char field@0003 into v #3 in the field id table). The instar by v2.

58	iget-short vx,vy,field_id	Reads a short instance field into vx. The instance is referenced by vy.	5830 0800 - iget-short v0, v3, Tes\nfield@0008\nReads the short field@0008 into v\n#8 in the field id table). The instar\nby v3.
59	iput vx,vy, field_id	Puts vx into an instance field. The instance is referenced by vy.	5920 0200 - iput v0,v2, Test2.i6:I\nStores v0 into field@0002 (entry \ntable). The instance is referenced
5A	iput-wide vx,vy, field_id	Puts the wide value located in vx and vx+1 registers into an instance field. The instance is referenced by vy.	5A20 0000 - iput-wide v0,v2, Test\nfield@0000\nStores the wide value in v0, v1 re\nfield@0000 (entry #0 in the field i\ninstance is referenced by v2.
5B	iput-object vx,vy,field_id	Puts the object reference in vx into an instance field. The instance is referenced by vy.	5B20 0000 - iput-object v0, v2,\nLineReader.bis:Ljava/io/BufferedI\nfield@0000\nStores the object reference in v0 \n(entry #0 in the field table). The ir\nreferenced by v2.
5C	iput-boolean vx,vy, field_id	Puts the boolean value located in vx into an instance field. The instance is referenced by vy.	5C30 0000 - iput-boolean v0, v3,\nfield@0000\nPuts the boolean value in v0 into \n(entry #0 in the field id table). The\nreferenced by v3.
5D	iput-byte vx,vy,field_id	Puts the byte value located in vx into an instance field. The instance is referenced by vy.	5D20 0100 - iput-byte v0, v2, Tes\nfield@0001\nPuts the boolean value in v0 into \n(entry #1 in the field id table). The\nreferenced by v2.
5E	iput-char vx,vy,field_id	Puts the char value located in vx into an instance field. The instance is referenced by vy.	5E20 0300 - iput-char v0, v2, Tes\nfield@0003\nPuts the char value in v0 into fiel\n#3 in the field id table). The instar\nby v2.
5F	iput-short vx,vy,field_id	Puts the short value located in vx into an instance field. The instance is referenced by vy.	5F21 0800 - iput-short v1, v2, Tes\nfield@0008\nPuts the short value in v1 into fiel\n#8 in the field id table). The instar\nby v2.
60	sget vx,field_id	Reads the integer field identified by the field_id into vx.	6000 0700 - sget v0, Test3.is1:I //\nReads field@0007 (entry #7 in the\ninto v0.
61	sget-wide vx, field_id	Reads the static field identified by the field_id into vx and vx+1 registers.	6100 0500 - sget-wide v0, Test2.I\nReads field@0005 (entry #5 in the\ninto v0 and v1 registers.
62	sget-object vx,field_id	Reads the object reference field	6201 0C00 - sget-object v1,\nTest3.os1:Ljava/lang/Object; // fie

		identified by the field_id into vx.	Reads field@000c (entry #CH in into v1.
63	sget-boolean vx,field_id	Reads the boolean static field identified by the field_id into vx.	6300 0C00 - sget-boolean v0, Tes field@000c Reads boolean field@000c (entry id table) into v0.
64	sget-byte vx,field_id	Reads the byte static field identified by the field_id into vx.	6400 0200 - sget-byte v0, Test3.b field@0002 Reads byte field@0002 (entry #2 table) into v0.
65	sget-char vx,field_id	Reads the char static field identified by the field_id into vx.	6500 0700 - sget-char v0, Test3.c field@0007 Reads byte field@0007 (entry #7 table) into v0.
66	sget-short vx,field_id	Reads the short static field identified by the field_id into vx.	6600 0B00 - sget-short v0, Test3. field@000b Reads short field@000b (entry #E table) into v0.
67	sput vx, field_id	Puts vx into a static field.	6700 0100 - sput v0, Test2.i5:I // f Stores v0 into field@0001 (entry table).
68	sput-wide vx, field_id	Puts vx and vx+1 into a static field.	6800 0500 - sput-wide v0, Test2.l Puts the long value in v0 and v1 i field@0005 static field (entry #5 i table).

69	sput-object vx,field_id	Puts object reference in vx into a static field.	6900 0c00 - sput-object v0,
			Test3.os1:Ljava/lang/Object; // fie
			Puts the object reference value ir field@000c static field (entry #CH table).

6A	sput-boolean vx,field_id	Puts boolean value in vx into a static field.	6A00 0300 - sput-boolean v0, Tes field@0003 Puts the byte value in v0 into the static field (entry #3 in the field id
6B	sput-byte vx,field_id	Puts byte value in vx into a static field.	6B00 0200 - sput-byte v0, Test3.b field@0002 Puts the byte value in v0 into the static field (entry #2 in the field id
6C	sput-char vx,field_id	Puts char value in vx into a static field.	6C01 0700 - sput-char v1, Test3.c field@0007 Puts the char value in v1 into the static field (entry #7 in the field id
6D	sput-short vx,field_id	Puts short value in vx into a static field.	6D00 0B00 - sput-short v0, Test3 field@000b Puts the short value in v0 into the static field (entry #BH in the field
6E	invoke-virtual { parameters }, methodtocall	Invokes a virtual method with parameters.	6E53 0600 0421 - invoke-virtual { v3}, Test2.method5:(IIII)V // metho Invokes the 6th method in the me

			the following arguments: v4 is the v0, v1, v2, and v3 are the method The method has 5 arguments (4 second byte)[5].
6F	invoke-super {parameter},methodtocall	Invokes the virtual method of the immediate parent class.	6F10 A601 0100 invoke-super {v1},java.io.FilterOutputStream.cl method@01a6 Invokes method@01a6 with one
70	invoke-direct { parameters }, methodtocall	Invokes a method with parameters without the virtual method resolution.	7010 0800 0100 - invoke-direct {v java.lang.Object.<init>:()V // meth Invokes the 8th method in the me just one parameter, v1 is the "this
71	invoke-static {parameters}, methodtocall	Invokes a static method with parameters.	7110 3400 0400 - invoke-static {v java.lang.Integer.parseInt:(Ljava/ method@0034 Invokes method@34 static metho is called with one parameter, v4[5].
72	invoke-interface {parameters},methodtocall	Invokes an interface method.	7240 2102 3154 invoke-interface mwfw.IReceivingProtocolAdapter. (ILjava/lang/String;Ljava/io/InputSt method@0221 Invokes method@221 interface m parameters in v1,v3,v4 and v5[5].
73	unused_73		
74	invoke-virtual/range {vx..vy},methodtocall	Invokes virtual method with a range of registers. The instruction specifies the first register and the number of registers to be passed to the method.	7403 0600 1300 - invoke-virtual {' Test2.method5:(IIII)V // method@ Invokes the 6th method in the me the following arguments: v19 is th v20 and v21 are the method para
75	invoke-super/range	Invokes the virtual method of the immediate parent class. The instruction specifies the first register and the number of registers to be passed to the method.	7501 A601 0100 invoke-super {v1},java.io.FilterOutputStream.cl method@01a6 Invokes method@01a6 with one
76	invoke-direct/range {vx..vy},methodtocall	Invokes direct method with a range of registers. The instruction specifies the first register and the number of registers	7603 3A00 1300 - invoke-direct/ra {v19..21},java.lang.Object.<init>:(method@003a Invokes method@3A with 1 param byte of the instruction=03). The p stored in v19 (5th,6th bytes of the

		to be passed to the method.	
77	invoke-static/range {vx..vy},methodtocall	Invokes static method with a range of registers. The instruction specifies the first register and the number of registers to be passed to the method.	7703 3A00 1300 - invoke-static/ra {v19..21},java.lang.Integer.parseI Ljava/lang/String;)I // method@00 Invokes method@3A with 1 param byte of the instruction=03). The p stored in v19 (5th,6th bytes of the
78	invoke-interface-range	Invokes an interface method with a range of registers. The instruction specifies the first register and the number of registers to be passed to the method.	7840 2102 0100 invoke-interface mwfw.IReceivingProtocolAdapter. (ILjava/lang/String;Ljava/io/InputSi method@0221 Invokes method@221 interface m parameters in v1..v4.
79	unused_79		
7A	unused_7A		
7B	neg-int vx,vy	Calculates vx=-vy.	7B01 - neg-int v1,v0 Calculates -v0 and stores the res
7C	not-int vx,vy		
7D	neg-long vx,vy	Calculates vx,vx+1=-(vy,vy+1)	7D02 - neg-long v2,v0 Calculates -(v0,v1) and stores the (v2,v3)
7E	not-long vx,vy		
7F	neg-float vx,vy	Calculates vx=-vy	7F01 - neg-float v1,v0 Calculates -v0 and stores the res
80	neg-double vx,vy	Calculates vx,vx+1=-(vy,vy+1)	8002 - neg-double v2,v0 Calculates -(v0,v1) and stores the (v2,v3)
81	int-to-long vx, vy	Converts the integer in vy into a long in vx,vx+1.	8106 - int-to-long v6, v0 Converts an integer in v0 into a lo
82	int-to-float vx, vy	Converts the integer in vx into a float in vx.	8206 - int-to-float v6, v0 Converts the integer in v0 into a f
83	int-to-double vx, vy	Converts the integer in vy into the double in vx,vx+1.	8306 - int-to-double v6, v0 Converts the integer in v0 into a c
84	long-to-int vx,vy	Converts the long value in vy,vy+1 into an integer in vx.	8424 - long-to-int v4, v2 Converts the long value in v2,v3 i value in v4.

85	long-to-float vx, vy	Converts the long value in vy,vy+1 into a float in vx.	8510 - long-to-float v0, v1 Convcerts the long value in v1,v2 value in v0.
86	long-to-double vx, vy	Converts the long value in vy,vy+1 into a double value in vx,vx+1.	8610 - long-to-double v0, v1 Converts the long value in v1,v2 i value in v0,v1.
87	float-to-int vx, vy	Converts the float value in vy into an integer value in vx.	8730 - float-to-int v0, v3 Converts the float value in v3 into in v0.
88	float-to-long vx,vy	Converts the float value in vy into a long value in vx.	8830 - float-to-long v0, v3 Converts the float value in v3 into v0,v1.
89	float-to-double vx, vy	Converts the float value in vy into a double value in vx,vx+1.	8930 - float-to-double v0, v3 Converts the float value in v3 into in v0,v1.
8A	double-to-int vx, vy	Converts the double value in vy,vy+1 into an integer value in vx.	8A40 - double-to-int v0, v4 Converts the double value in v4,v value in v0.
8B	double-to-long vx, vy	Converts the double value in vy,vy+1 into a long value in vx,vx+1.	8B40 - double-to-long v0, v4 Converts the double value in v4,v value in v0,v1.
8C	double-to-float vx, vy	Converts the double value in vy,vy+1 into a float value in vx.	8C40 - double-to-float v0, v4 Converts the double value in v4,v value in v0,v1.
8D	int-to-byte vx,vy	Converts the int value in vy to a byte value and stores it in vx.	8D00 - int-to-byte v0, v0 Converts the integer in v0 into a b byte value into v0.
8E	int-to-char vx,vy	Converts the int value in vy to a char value and stores it in vx.	8E33 - int-to-char v3, v3 Converts the integer in v3 into a c char value into v3.
8F	int-to-short vx,vy	Converts the int value in vy to a short value and stores it in vx.	8F00 - int-to-short v0, v0 Converts the integer in v0 into a s the short value into v3.
90	add-int vx,vy,vz	Calculates vy+vz and puts the result into vx.	9000 0203 - add-int v0, v2, v3 Adds v3 to v2 and puts the result
91	sub-int vx,vy,vz	Calculates vy-vz and puts the result into vx.	9100 0203 - sub-int v0, v2, v3 Subtracts v3 from v2 and puts the

92	mul-int vx, vy, vz	Multiplies vz with wy and puts the result int vx.	9200 0203 - mul-int v0,v2,v3 Multiplies v2 with w3 and puts the
93	div-int vx,vy,vz	Divides vy with vz and puts the result into vx.	9303 0001 - div-int v3, v0, v1 Divides v0 with v1 and puts the re
94	rem-int vx,vy,vz	Calculates vy % vz and puts the result into vx.	9400 0203 - rem-int v0, v2, v3 Calculates v3 % v2 and puts the
95	and-int vx, vy, vz	Calculates vy AND vz and puts the result into vx.	9503 0001 - and-int v3, v0, v1 Calculates v0 AND v1 and puts th
96	or-int vx, vy, vz	Calculates vy OR vz and puts the result into vx.	9603 0001 - or-int v3, v0, v1 Calculates v0 OR v1 and puts the
97	xor-int vx, vy, vz	Calculates vy XOR vz and puts the result into vx.	9703 0001 - xor-int v3, v0, v1 Calculates v0 XOR v1 and puts th
98	shl-int vx, vy, vz	Shift vy left by the positions specified by vz and store the result into vx.	9802 0001 - shl-int v2, v0, v1 Shift v0 left by the positions speci store the result in v2.
99	shr-int vx, vy, vz	Shift vy right by the positions specified by vz and store the result into vx.	9902 0001 - shr-int v2, v0, v1 Shift v0 right by the positions spe store the result in v2.
9A	ushr-int vx, vy, vz	Unsigned shift right (>>>) vy by the positions specified by vz and store the result into vx.	9A02 0001 - ushr-int v2, v0, v1 Unsigned shift v0 right by the pos by v1 and store the result in v2.
9B	add-long vx, vy, vz	Adds vy to vz and puts the result into vx[1].	9B00 0305 - add-long v0, v3, v5 The long value in v3,v4 is added v5,v6 and the result is stored in v
9C	sub-long vx,vy,vz	Calculates vy-vz and puts the result into vx[1].	9C00 0305 - sub-long v0, v3, v5 Subtracts the long value in v5,v6 value in v3,v4 and puts the result
9D	mul-long vx,vy,vz	Calculates vy*vz and puts the result into vx[1].	9D00 0305 - mul-long v0, v3, v5 Multiplies the long value in v5,v6 value in v3,v4 and puts the result
9E	div-long vx, vy, vz	Calculates vy/vz and puts the result into vx[1].	9E06 0002 - div-long v6, v0, v2 Divides the long value in v0,v1 wi in v2,v3 and pust the result into v
9F	rem-long vx,vy,vz	Calculates vy % vz and puts the result into vx[1].	9F06 0002 - rem-long v6, v0, v2 Calculates v0,v1 % v2,v3 and put v6,v7.

A0	and-long vx, vy, vz	Calculates the vy AND vz and puts the result into vx[1].	A006 0002 - and-long v6, v0, v2 Calculates v0,v1 AND v2,v3 and into v6,v7.
A1	or-long vx, vy, vz	Calculates the vy OR vz and puts the result into vx[1].	A106 0002 - or-long v6, v0, v2 Calculates v0,v1 OR v2,v3 and pu v6,v7.
A2	xor-long vx, vy, vz	Calculates the vy XOR vz and puts the result into vx[1].	A206 0002 - xor-long v6, v0, v2 Calculates v0,v1 XOR v2,v3 and into v6,v7.
A3	shl-long vx, vy, vz	Shifts left vy by vz positions and stores the result in vx[1].	A302 0004 - shl-long v2, v0, v4 Shift v0,v1 by postions specified the result into v2,v3.
A4	shr-long vx,vy,vz	Shifts right vy by vz positions and stores the result in vx[1].	A402 0004 - shr-long v2, v0, v4 Shift v0,v1 by postions specified the result into v2,v3.
A5	ushr-long vx, vy, vz	Unsigned shifts right vy by vz positions and stores the result in vx[1].	A502 0004 - ushr-long v2, v0, v4 Unsigned shift v0,v1 by postions and puts the result into v2,v3.
A6	add-float vx,vy,vz	Adds vy to vz and puts the result into vx.	A600 0203 - add-float v0, v2, v3 Adds the floating point numbers i puts the result into v0.
A7	sub-float vx,vy,vz	Calculates vy-vz and puts the result into vx.	A700 0203 - sub-float v0, v2, v3 Calculates v2-v3 and puts the res
A8	mul-float vx, vy, vz	Multiplies vy with vz and puts the result into vx.	A803 0001 - mul-float v3, v0, v1 Multiplies v0 with v1 and puts the
A9	div-float vx, vy, vz	Calculates vy/vz and puts the result into vx.	A903 0001 - div-float v3, v0, v1 Divides v0 with v1 and puts the re
AA	rem-float vx,vy,vz	Calculates vy % vz and puts the result into vx.	AA03 0001 - rem-float v3, v0, v1 Calculates v0 % v1 and puts the
AB	add-double vx,vy,vz	Adds vy to vz and puts the result into vx[1].	AB00 0305 - add-double v0, v3, v Adds the double value in v5,v6 re double value in v3,v4 registers ar result in v0,v1 registers.
AC	sub-double vx,vy,vz	Calculates vy-vz and puts the result into vx[1].	AC00 0305 - sub-double v0, v3, v Subtracts the value in v5,v6 from v3,v4 and puts the result into v0,v
AD	mul-double vx, vy, vz	Multiplies vy with vz and puts the result into vx[1].	AD06 0002 - mul-double v6, v0, v Multiplies the double value in v0,v double value in v2,v3 and puts the v6,v7.

AE	div-double vx, vy, vz	Calculates vy/vz and puts the result into vx[1].	AE06 0002 - div-double v6, v0, v2 Divides the double value in v0,v1 value in v2,v3 and puts the result
AF	rem-double vx,vy,vz	Calculates vy % vz and puts the result into vx[1].	AF06 0002 - rem-double v6, v0, v Calculates v0,v1 % v2,v3 and put v6,v7.
B0	add-int/2addr vx,vy	Adds vy to vx.	B010 - add-int/2addr v0,v1 Adds v1 to v0.
B1	sub-int/2addr vx,vy	Calculates vx-vy and puts the result into vx.	B140 - sub-int/2addr v0, v4 Subtracts v4 from v0 and puts the
B2	mul-int/2addr vx,vy	Multiplies vx with vy.	B210 - mul-int/2addr v0, v1 Multiples v0 with v1 and puts the
B3	div-int/2addr vx,vy	Divides vx with vy and puts the result into vx.	B310 - div-int/2addr v0, v1 Divides v0 with v1 and puts the re
B4	rem-int/2addr vx,vy	Calculates vx % vy and puts the result into vx	B410 - rem-int/2addr v0, v1 Calculates v0 % v1 and puts the
B5	and-int/2addr vx, vy	Calculates vx AND vy and puts the result into vx.	B510 - and-int/2addr v0, v1 Calculates v0 AND v1 and puts th
B6	or-int/2addr vx, vy	Calculates vx OR vy and puts the result into vx.	B610 - or-int/2addr v0, v1 Calculates v0 OR v1 and puts the
B7	xor-int/2addr vx, vy	Calculates vx XOR vy and puts the result into vx.	B710 - xor-int/2addr v0, v1 Calculates v0 XOR v1 and puts th
B8	shl-int/2addr vx, vy	Shifts vx left by vy positions.	B810 - shl-int/2addr v0, v1 Shift v0 left by v1 positions.
B9	shr-int/2addr vx, vy	Shifts vx right by vy positions.	B910 - shr-int/2addr v0, v1 Shift v0 right by v1 positions.
BA	ushr-int/2addr vx, vy	Unsigned shift right (>>>) vx by the positions specified by vy.	BA10 - ushr-int/2addr v0, v1 Unsigned shift v0 by the positions
BB	add-long/2addr vx,vy	Adds vy to vx[1].	BB20 - add-long/2addr v0, v2 Adds the long value in v2,v3 regis value in v0,v1 registers.
BC	sub-long/2addr vx,vy	Calculates vx-vy and puts the result into vx[1].	BC70 - sub-long/2addr v0, v7 Subtracts the long value in v7,v8 value in v0,v1 and puts the result
BD	mul-long/2addr vx,vy	Calculates vx*vy and puts the result into vx[1].	BD70 - mul-long/2addr v0, v7 Multiplies the long value in v7,v8 value in v0,v1 and puts the result

BE	div-long/2addr vx, vy	Calculates vx/vy and puts the result into vx[1].	BE20 - div-long/2addr v0, v2 Divides the long value in v0,v1 wi in v2,v3 and puts the result into v
BF	rem-long/2addr vx,vy	Calculates vx % vy and puts the result into vx[1].	BF20 - rem-long/2addr v0, v2 Calculates v0,v1 % v2,v3 and put v0,v1
C0	and-long/2addr vx, vy	Calculates vx AND vy and puts the result into vx[1].	C020 - and-long/2addr v0, v2 Calculates v0,v1 OR v2,v3 and pu v0,v1.
C1	or-long/2addr vx, vy	Calculates vx OR vy and puts the result into vx[1].	C120 - or-long/2addr v0, v2 Calculates v0,v1 OR v2,v3 and pu v0,v1.
C2	xor-long/2addr vx, vy	Calculates vx XOR vy and puts the result into vx[1].	C220 - xor-long/2addr v0, v2 Calculates v0,v1 XOR v2,v3 and into v0,v1.
C3	shl-long/2addr vx, vy	Shifts left the value in vx,vx+1 by the positions specified by vy and stores the result in vx,vx+1.	C320 - shl-long/2addr v0, v2 Shifts left v0,v1 by the positions s
C4	shr-long/2addr vx, vy	Shifts right the value in vx,vx+1 by the positions specified by vy and stores the result in vx,vx+1.	C420 - shr-long/2addr v0, v2 Shifts right v0,v1 by the positions
C5	ushr-long/2addr vx, vy	Unsigned shifts right the value in vx,vx+1 by the positions specified by vy and stores the result in vx,vx+1.	C520 - ushr-long/2addr v0, v2 Unsigned shifts right v0,v1 by the specified by v2.
C6	add-float/2addr vx,vy	Adds vy to vx.	C640 - add-float/2addr v0,v4 Adds v4 to v0.
C7	sub-float/2addr vx,vy	Calculates vx-vy and stores the result in vx.	C740 - sub-float/2addr v0,v4 Adds v4 to v0.
C8	mul-float/2addr vx, vy	Multiplies vx with vy.	C810 - mul-float/2addr v0, v1 Multiplies v0 with v1.
C9	div-float/2addr vx, vy	Calculates vx/vy and puts the result into vx.	C910 - div-float/2addr v0, v1 Divides v0 with v1 and puts the re
CA	rem-float/2addr vx,vy	Calculates vx/vy and puts the result into vx.	CA10 - rem-float/2addr v0, v1 Calculates v0 % v1 and puts the

CB	add-double/2addr vx, vy	Adds vy to vx[1].	CB70 - add-double/2addr v0, v7 Adds v7 to v0.
CC	sub-double/2addr vx, vy	Calculates vx-vy and puts the result into vx[1].	CC70 - sub-double/2addr v0, v7 Subtracts the value in v7,v8 from v0,v1 and puts the result into v0,v
CD	mul-double/2addr vx, vy	Multiplies vx with vy[1].	CD20 - mul-double/2addr v0, v2 Multiplies the double value in v0,v double value in v2,v3 and puts the v0,v1.
CE	div-double/2addr vx, vy	Calculates vx/vy and puts the result into vx[1].	CE20 - div-double/2addr v0, v2 Divides the double value in v0,v1 value in v2,v3 and puts the value
CF	rem-double/2addr vx,vy	Calculates vx % vy and puts the result into vx[1].	CF20 - rem-double/2addr v0, v2 Calculates v0,v1 % v2,v3 and put v0,v1.
D0	add-int/lit16 vx,vy,lit16	Adds vy to lit16 and stores the result into vx.	D001 D204 - add-int/lit16 v1, v0, #04d2 Adds v0 to literal 1234 and stores v1.
D1	sub-int/lit16 vx,vy,lit16	Calculates vy - lit16 and stores the result into vx.	D101 D204 - sub-int/lit16 v1, v0, #04d2 Calculates v0 - literal 1234 and st into v1.
D2	mul-int/lit16 vx,vy,lit16	Calculates vy * lit16 and stores the result into vx.	D201 D204 - mul-int/lit16 v1, v0, #04d2 Calculates v0 * literal 1234 and st into v1.
D3	div-int/lit16 vx,vy,lit16	Calculates vy / lit16 and stores the result into vx.	D301 D204 - div-int/lit16 v1, v0, # #04d2 Calculates v0 / literal 1234 and st into v1.
D4	rem-int/lit16 vx,vy,lit16	Calculates vy % lit16 and stores the result into vx.	D401 D204 - rem-int/lit16 v1, v0, #04d2 Calculates v0 % literal 1234 and s into v1.
D5	and-int/lit16 vx,vy,lit16	Calculates vy AND lit16 and stores the result into vx.	D501 D204 - and-int/lit16 v1, v0, #04d2 Calculates v0 AND literal 1234 an result into v1.
D6	or-int/lit16 vx,vy,lit16	Calculates vy OR lit16 and stores the result into vx.	D601 D204 - or-int/lit16 v1, v0, #i Calculates v0 OR literal 1234 and result into v1.
D7	xor-int/lit16 vx,vy,lit16	Calculates vy XOR lit16 and stores the result into vx.	D701 D204 - xor-int/lit16 v1, v0, # #04d2 Calculates v0 XOR literal 1234 an result into v1.

D8	add-int/lit8 vx,vy,lit8	Adds vy to lit8 and stores the result into vx.	D800 0201 - add-int/lit8 v0,v2, #in Adds literal 1 to v2 and stores the
D9	sub-int/lit8 vx,vy,lit8	Calculates vy-lit8 and stores the result into vx.	D900 0201 - sub-int/lit8 v0,v2, #in Calculates v2-1 and stores the re
DA	mul-int/lit8 vx,vy,lit8	Multiplies vy with lit8 8-bit literal constant and puts the result into vx.	DA00 0002 - mul-int/lit8 v0,v0, #in Multiplies v0 with literal 2 and put v0.
DB	div-int/lit8 vx,vy,lit8	Calculates vy/lit8 and stores the result into vx.	DB00 0203 - mul-int/lit8 v0,v2, #in Calculates v2/3 and stores the re
DC	rem-int/lit8 vx,vy,lit8	Calculates vy % lit8 and stores the result into vx.	DC00 0203 - rem-int/lit8 v0,v2, #i Calculates v2 % 3 and stores the
DD	and-int/lit8 vx,vy,lit8	Calculates vy AND lit8 and stores the result into vx.	DD00 0203 - and-int/lit8 v0,v2, #in Calculates v2 AND 3 and stores t
DE	or-int/lit8 vx, vy, lit8	Calculates vy OR lit8 and puts the result into vx.	DE00 0203 - or-int/lit8 v0, v2, #int Calculates v2 OR literal 3 and pu v0.
DF	xor-int/lit8 vx, vy, lit8	Calculates vy XOR lit8 and puts the result into vx.	DF00 0203	0008: xor-int/lit8 v0, Calculates v2 XOR literal 3 and p into v0.
E0	shl-int/lit8 vx, vy, lit8	Shift v0 left by the bit positions specified by the literal constant and put the result into vx.	E001 0001 - shl-int/lit8 v1, v0, #in Shift v0 left by 1 position and put v1.
E1	shr-int/lit8 vx, vy, lit8	Shift v0 right by the bit positions specified by the literal constant and put the result into vx.	E101 0001 - shr-int/lit8 v1, v0, #in Shift v0 right by 1 position and pu v1.
E2	ushr-int/lit8 vx, vy, lit8	Unsigned right shift of v0 (>>>) by the bit positions specified by the literal constant and put the result into vx.	E201 0001 - ushr-int/lit8 v1, v0, # Unsigned shift v0 right by 1 positi result into v1.
E3	unused_E3		
E4	unused_E4		
E5	unused_E5		

E6	unused_E6		
E7	unused_E7		
E8	unused_E8		
E9	unused_E9		
EA	unused_EA		
EB	unused_EB		
EC	unused_EC		
ED	unused_ED		
EE	execute-inline {parameters},inline ID	Executes the inline method identified by inline ID[6].	EE20 0300 0100 - execute-inline #0003 Executes inline method #3 using passing one parameter in v0.
EF	unused_EF		
F0	invoke-direct-empty	Stands as a placeholder for pruned empty methods like Object.<init>. This acts as nop during normal execution[6].	F010 F608 0000 - invoke-direct-e Ljava/lang/Object;.<init>:()V // me Replacement for the empty metho java/lang/Object;<init>.
F1	unused_F1		
F2	iget-quick vx,vy,offset	Gets the value stored at offset in vy instance's data area to vx[6].	F221 1000 - iget-quick v1, v2, [ob Gets the value at offset 0CH of th pointed by v2 and stores the obje v1.
F3	iget-wide-quick vx,vy,offset	Gets the object reference value stored at offset in vy instance's data area to vx,vx+1[6].	F364 3001 - iget-wide-quick v4, v Gets the value at offset 130H of tl pointed by v6 and stores the obje v4,v5.
F4	iget-object-quick vx,vy,offset	Gets the object reference value stored at offset in vy instance's data area to vx[6].	F431 0C00 - iget-object-quick v1, Gets the object reference value a the instance pointed by v3 and ste reference in v1.
F5	iput-quick vx,vy,offset	Puts the value stored in vx to offset in vy instance's data area[6].	F521 1000 - iput-quick v1, v2, [ob Puts the object reference value in of the instance pointed by v2.

F6	iput-wide-quick vx,vy,offset	Puts the value stored in vx,vx+1 to offset in vy instance's data area[6].	F652 7001 - iput-wide-quick v2, v Puts the value in v2,v3 to offset 1 instance pointed by v5.
F7	iput-object-quick vx,vy,offset	Puts the object reference value stored in vx to offset in vy instance's data area to vx[6].	F701 4C00 - iput-object-quick v1, Puts the object reference value in 0CH of the instance pointed by v3
F8	invoke-virtual-quick {parameters},vtable offset	Invokes a virtual method using the vtable of the target object[6].	F820 B800 CF00 - invoke-virtual- vtable #00b8 Invokes a virtual method. The targ instance is pointed by v15 and vta points to the method to be called. parameter to the method call.
F9	invoke-virtual-quick/range {parameter range},vtable offset	Invokes a virtual method using the vtable of the target object[6]	F906 1800 0000 - invoke-virtual-q {v0..v5},vtable #0018 Invokes a method using the vtable pointed by v0. v1..v5 registers are the method call.
FA	invoke-super-quick {parameters},vtable offset	Invokes a virtual method in the target object's immediate parent class using the vtable of that parent class[6].	FA40 8100 3254 - invoke-super-q v5}, vtable #0081 Invokes a method using the vtable immediate parent class of instanc v3, v4 and v5 registers are param method call.
FB	invoke-super-quick/range {register range},vtable offset	Invokes a virtual method in the target object's immediate parent class using the vtable of that parent class[6].	F906 1B00 0000 - invoke-super-q {v0..v5},vtable #001b Invokes a method using the vtable immediate parent class of instanc v1..v5 registers are parameters to call.
FC	unused_FC		
FD	unused_FD		
FE	unused_FE		
FF	unused_FF		