Opcodes'

tables

Complete

This file is free to modify and re-use, with no restrictions, even commercially. it's an OpenOffice document. 1 grab its content via: *svn export https://corkami.googlecode.com/svn/trunk/oOo/opcodes_tables_complete* 2 rezip with subdirectories as opcodes_tables_complete.ods

opcodes tables are available as compact and complete form.

Feb 21, 2012 Ange Albertini 2012 http://corkami.com

inspired by the work of Daniel Plohmann Creative Commons Attribution 3.0 Unported License

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	хA	хB	xC	хD	хE	xF
0x	nop	aconst_null	iconst_m1	iconst_0	iconst_1	iconst_2	iconst_3	iconst_4	iconst_5	lconst_0	lconst_1	fconst_0	fconst_1	fconst_2	dconst_0	dconst_1
0.	Do nothing	Push null				Push int consta				Push long			Push float			double
1x	bipush	sipush	ldc	ldc_w	ldc2_w	iload	lload	fload	dload	aload 1	iload_0	iload_1	iload_2	iload_3	lload_0	lload_1
1	Push byte 1	Tush short -	Push item from runtime constant pool 1	Push item from runtime constant pool (wide index) 2	Push long or double from runtime constant pool (wide index) 2			uble, reference} fro				Load int from local			U	m local variable
2v	lload_2	lload_3	fload_0	fload_1	fload_2	fload_3	dload_0	dload_1	dload_2	dload_3	aload_0	aload_1	aload_2	aload_3	iaload	laload
2^	v	m local variable		Load float fro	m local variabl	e			om local variable		Lo	ad reference from lo	cal variable		Load {int,lor	ng} from array
3x	faload	daload	aaload	baload	caload	saload	istore	Istore	fstore	dstore	astore 1	istore_0	istore_1	istore_2	istore_3	Istore_0
57		Load (float, dou	ble, reference, byt	te or boolean, ch	ar, short} from		Store {int, float, double, reference} into local variable Store int into I							ocal variable		Store long into local variable
4x	lstore_1	lstore_2	lstore_3	fstore_0	fstore_1	fstore_2	fstore_3	dstore_0	dstore_1	dstore_2	dstore_3	astore_0	astore_1	astore_2	astore_3	iastore
		re long into local v	variable		Store flo	at into local variable			Store double	into local variable				to local variable		Store into int array
5x	lastore	fastore	dastore	aastore	bastore	castore	sastore	рор	pop2	dup	dup_x1	dup_x2	dup2	dup2_x1	dup2_x2	swap
57			<u> </u>			char, short} array		Pop the top operand stack value			Duplicate the top operand stack value and insert two values down	Duplicate the top operand stack value and insert two or three values down	Duplicate the top one or two operand stack values or	plicate the top one or two operand stack values and insert two or three values down	Digitals fields are or less spectral viacit values and inset law, lives, or fair values down	Swap the top two operand stack values
6x	iadd	ladd	fadd	dadd	isub	lsub	fsub	dsub	imul	Imul	fmul	dmul	idiv	ldiv	fdiv	ddiv
07			, float, double}				ong, float, double}				ng, float, double}			Divide {int, long		•
7x	irem	Irem	frem	drem	ineg	Ineg	fneg	dneg	ishl	Ishl	ishr	lshr	iushr	lushr	iand	land
17			ong, float, double	•			ng, float, double}			ft {int, long}	Arithmetic shift		<u> </u>	right {int, long}		ND {int, long}
8x	ior	lor	ixor	lxor	iinc 2	i2l	i2f	i2d	l2i	l2f	l2d	f2i	f2l	f2d	d2i	d2l
UN		DR {int, long}	Boolean XOF				-		-	t Convert long to float			Convert float to long (Convert float to double	Convert double to int	Convert double to long
9x	d2f	i2b	i2c	i2s	lcmp	fcmpl	fcmpg	dcmpl	dcmpg	ifeq	ifne	iflt	ifge	ifgt	ifle 2	if_icmpeq
UN	Convert double to float	Convert int to byte	Convert int to chai		Compare long			Compare				t comparison with z				Branch if int comparison succeeds 2
Ax	if_icmpne	if_icmplt	if_icmpge	if_icmpgt	if_icmple 2	if_acmpeq	if_acmpne 2	goto 2	jsr 2	ret 1	tableswitch	lookupswitch _v	ireturn	Ireturn	freturn	dreturn
, , ,			int comparison su			Branch if reference co		Branch always	Jump subroutine		Access jump table by index and jump	Access jump table by key match and jump		n {int, long, float	-	
Bx	areturn	return	getstatic	putstatic 2	getfield	putfield 2	invokevirtual	invokespecial	invokestatic	invokeinterface	xxunusedxxx1	new	newarray	anewarray	arraylength	athrow
			Get static field from class				Invoke instance method; dispatch based on class 2	Indefendence method, special landing for superviser, pinale, and reduces relativation method to markets. 2		Invoke interface method 4		Create new object 2	Create new array 1 C	reate new array of reference 2	Get length of array	Throw exception or error
Сх	checkcast	instance of	monitorenter	monitorexit	wide	multianewarray	ifnull	ifnonnull 2	goto_w 4	jsr_w 4	breakpoint					
	Check whether object is of given type 2	Determine if object is of given type 2	Enter monitor for objec	t Exit monitor for object	Extend local variable index by additional bytes	Create new multidimensional array 3	Branch if reference is null	Branch if reference not null	Branch always (wide index)	Jump subroutine (wide index)						

Fx

impdep1 impdep2

misc	arithmetic	flow
constants	logical	method
immediates	conversion	object
locals	comparison	system
stack	conditional	undefined

JVM (Java)

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	хA	хB	xC	хD	хE	xF
0x	nop Do nothing	break breakpoint instruction	ldarg.0	Idarg.1	Idarg.2 I,2,3} onto the stack	ldarg.3	Idloc.0	Idloc.1	Idloc.2 {0, 1, 2, 3} onto st	Idloc.3	stloc.0	stloc.1 from stack into I	stloc.2	stloc.3	Idarg.s	Idarga.s
1x	starg.s	Idloc.s	Idloca.s	stlock.s	Idnull	ldc.i4.m1	Idc.i4.0	Idc.i4.1	Idc.i4.2	Idc.i4.3	Idc.i4.4	Idc.i4.5	Idc.i4.6	Idc.i4.7	ldc.i4.8	Idc.i4.s
17	Store value to the argument numbered num.	load local variable of index indx onto stack		pop value from stack to local variable	Push a null reference on the stack					3,4,5,6,7,8} onto the						Push num onto the stack as int32
2x	Idc.i4	Idc.i8	Idc.r4 Push num of type float32 onto the stack as F	Idc.r8 Push num of type float64 onto the stack as F		dup Duplicate the value on the top of the stack	pop remove the top element of the stack	jmp Exit current method and jump to the specified method	call call a method	calli indirect method call	ret Return from method, possibly with a value	br.s	brfalse.s	brtrue.s	beq.s branch on equal	bge.s
3x	bgt.s	ble.s	blt.s	bne.un.s	bge.un.s	bgt.un.s	ble.un.s	blt.un.s	br	brfalse	brtrue	beq	bge	bgt	ble	blt
	branch on greater than	branch on less than or equal to	branch on less than bqt.un	branch on not equal or unordered	branch on greater than or equal to, unsigned or unordered	branch on greater than, unsigned or unordered Switch	Idind.i1	branch on less than, unsigned or unordered	Idind.i2	branch on false, null, or zero	branch on non-false or non-null	branch on equa	branch on greater than or equal to	branch on greater thar Idind.i	branch on less than or equal to	branch on less than Idind.r8
4x	branch on not equal or unordered	branch on greater than or equal to, unsigned or unordered	branch on greater than, unsigned or unordered		branch on less than, unsigned or unordered		Indirect load value of type int8 as int32 on the stack	ndirect load value of type unsigned int8 as int32 on the stack	Indirect load value of type int16 as int32 on the stack	Indirect load value of type unsigned int16 as int32 on the stack	Indirect load value of type int32 as int32 on the stack	Indirect load value of type unsigned int32 as int32 on the sta	x Indirect load value of type int54 as int54 on the stack	Indirect load value of type native int as native int on the stad	x Indirect load value of type float32 as F on the stack	x Indirect load value of type foat64 as F on the stack
5x	Idind.ref	stind.ref	stind.i1	stind.i2	stind.i4	stind.i8	stind.r4	stind.r8	add	sub	mul	div	div.un	rem	rem.un	and
	Indirect load value of type object ref as 0 on the stack	Store value of type object ref (type O) into memory at address XOI	Store value of type int8 into memory at address	Store value of type int16 into memory at address	Store value of type int32 into memory at address		Store value of type float32 into memory at address	Store value of type float64 into memory at address	add numeric values	subtract numeric values conv.i4	Multiply values	Divide two values to return a quotient or floating-point res	Divide two values, unsigned, returning a quotient.	Remainder when dividing one value by anothe	r Remainder when staiding one unaigned value by another	bitwise AND
6x	bitwise OR	bitwise XOR	shift integer left			neg Negate value	Bitwise complement	Convert to int8, pushing int32 on stack		Convert to int32, pushing int32 on stack		CONVIT Convert to float32, pushing F on stact	COTTVITO	Convert to unsigned int32, pushing int32 on stack	CONV.CO	Call VIIL
7x	cpobj	ldobj	Idstr Push a string object for the literal string	newobj	castclass cast an object to a class	isinst	conv.r.un			unbox	throw Throw an exception	ldfld	ldflda	stfld	ldsfld	ldsflda
8x	stsfld Replace the value of field with val	Store a value of type typeTok at an address	·	conv.ovf.i2.un	conv.ovf.i4.un		conv.ovf.u1.un	conv.ovf.u2.un	conv.ovf.u4.un	conv.ovf.u8.un	conv.ovf.i.un	conv.ovf.u.un	box convert a boxable value to its boxed form	Newarr Create a new array with elements of type etype	Idlen	Idelema Iso to assess of event at hold with the basis
9x	Idelem.i1	ldelem.u1	Idelem.i2	Idelem.u2	Idelem.i4	Idelem.u4	Idelem.i8	Idelem.i	Idelem.r4	Idelem.r8	Idelem.ref	stelem.i	stelem.i1	stelem.i2	stelem.i4	stelem.i8
Ax	stelem.r4	stelem.r8	stelem.ref	Idelem load element from array	stelem	unbox.any										
Вx				conv.ovf.i1	Replace ettag sectores at index with the value of the same	CONV.OVF.i2	conv.ovf.u2	conv.ovf.i4	conv.ovf.u4	conv.ovf.i8	conv.ovf.u8					
Сх			refanyval Push the address stored in a typed reference	ckfinite check for a finite real number			mkrefany Push a typed reference to ptr of type class onto the stack					•				
Dx	Idtoken	conv.u2	Conv.u1	CONV.I	conv.ovf.i	conv.ovf.u	add.ovf add {signed, unsigned} integer	add.ovf.un	mul.ovf multiply {signed, unsigned} inf	mul.ovf.un teger values with overflow check	sub.ovf	sub.ovf.un ger values, checking for overflov	endfinally end the finally of fault classes of an exception block	leave Exit a protected	leave.s	stind.i
Ex	CONV.U															
FE	arglist	ceq	cgt	cgt.un	cit	cit.un	ldftn	ldvirtftn		ldarg	Idarga	starg	Idloc	Idloca	stloc	localloc
FE	get argument list	compare equal endfilter	unaligned.	compare greater than, unsigned or unordered volatile.	compare less than tail.	compare less than, unsigned or unordered	Push a pointer to a method referenced by method, on the stack I constrained	Push address of virtual method method on the stack	initblk	load numbered argument onto the stack	load an argument address	Store value to the argument numbered num	load local variable of index indx onto stack	Load address of local variable with index ind	readonly.	Allocate space from the local memory pool
г <u>с</u> 1х			-		call terminates current method		invoke a member on a value of a variable type	Copy data from memory to memory	Set all bytes in a block of memory to a given byte value	possibly skip a fault check	Rethrow the current exception		Push the size, in bytes, of a type as an unsigned int32	Push the type token stored in a typed reference		

misc	immediates	arithmetic	fields	prefix
system	stack	logical	array	
args	method	conversion	references	
locals	conditional	object	flow	
constants	indirects	exception	comparison	undefined

Common Intermediate Language (.Net)

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	хA	хB	xC	хD	хE	xF
	nop	move	move/from16	move/16	move-wide	move-wide/from16	move-wide/16	move-object	move-object/from16	move-object/16	move-result	move-result-wide	move-result-object	move-exception	return-void	return
0x	Waste cycles	Move the conte	nts of one non-objec	ct register to another	Move the c	ontents of one register-pair to	another	Move the co	ontents of one object-bearin	g register to another	Move the single-word non-object result of the most recent <i>invoke-kind</i> into the indicated register		Move the object result of the most recent invoke-kind into the indicated register	Save a just-caught exception into the given register	Return from a void method	Return from a single-width (32-bit) non-object value-returning method
1x	return-wide	return-object	Const/4 Nove the given iteral value (sign-extended to 32 bits) into the specified register.	Const/16	Const Move the given literal value into the specified register	const/high16	const-wide/16	const-wide/32	const-wide	const-wide/high16	const-string	const-string-jumbo	Const-class	monitorenter	monitorexit Release the monitor for the indicated object	checkcast
	instanceof	arraylength	newinstance	newarray	filled-new-array	filled-new-array-range	fill-array-data	throw	goto	goto/16	goto/32	packed-switch	sparse-switch	cmpl-float	cmpg-float	cmpl-double
2x																
	tore in the given destination register 1 if the indicated referen- is an instance of the given type, or 0 if not	CC Store in the given destination register the length of the indicated array, in entries	Construct a new instance of the indicated type, storing a reference to it in the destination.	 Construct a new array of the indicated type and size 	Construct an array of the given type a	and size, filling it with the supplied contents	Fill the given array with the indicated data	Throw the indicated exception	Uncond	itionally jump to the indicated in	nstruction	using a table of offsets corresponding to ease of the value in a particular integral range, or fail through to the next instruction if there is no match	ounp to a new instruction based of the value in the great register, using an ordered table of value of state. or fail through to the next instruction if there is no match.	Perform the indicated having part or long comparison, storing 3.1 the two arguments are report, 1.1 the second ang	nert is larger, or -1.0 the first argument is larger. The "back label for the flasting part operations indicates how hold or	represent are sensed. "Of law" non-close man, 1 for half companions, and "I black" instructions when -1.
3x	cmpg-double	cmp-long	if-eq	if-ne	if-lt	if-ge	if-gt	if-le	if-eqz	if-nez	if-ltz	if-gez	if-gtz	if-lez		
SX	المتعار معارضها المراجعة والمراجع والمعار والمحال والمحال والمحال والمحال والمحال والمحال والمحال والمحال المحال			Branch to the	e given destination if the g	jiven two registers' values cor				<u> </u>	estination if the given registe	er's value compares with 0 as s	specified.			
4x					aget	aget-wide	aget-object	aget-bool	aget-byte	aget-char	aget-short	aput	aput-wide	aput-object	aput-bool	aput-byte
										tion at the identified index of the					<u> </u>	
5x	aput-char	aput-short	iget	iget-wide	iget-object	iget-bool	iget-byte	iget-char	iget-short	iput	iput-wide	iput-object	iput-bool	iput-byte	iput-char	iput-short
-		iven array, loading or storing into the value register		a wat h a al	a wat huita			_		with the identified field, loading	<u> </u>		anut abau	a work a la a vit	in the states of	invelse enner
6x	sget	sget-wide	sget-object	sget-bool	sget-byte	sget-char	sget-short	sput	sput-wide	sput-object	sput-bool	sput-byte	sput-char	sput-short	invoke-virtual	invoke-super
	invoke-direct	invoke-static	invoke-interface		invoke-virtual/range	invoke-super/range	· ·		atic field, loading or storing invoke-interface/range	Into the value register		neg-int	not-int	neg-long	Call the indic	neg-float
7x		he indicated meth			invoke-virtuai/range		If the indicated method	-	invoke-interface/range				unary operation on the s		not-long the result in the destir	-
	neg-double	int-to-long	int-to-float	int-to-double	long-to-int	long-to-float	long-to-double	float-to-int	float-to-long	float-to-double	double-to-int	double-to-long	double-to-float	int-to-byte	int-to-char	int-to-short
8x	nog dodbio	int to long	int to notit		long-to-int	iong-to-noat	long to double		U U	g the result in the destination re						
0	add-int	sub-int	mul-int	div-int	rem-int	and-int	or-int	xor-int	shl-int	shr-int	ushr-int	add-long	sub-long	mul-long	div-long	rem-long
9x							Perform the identified	ed binary operation on th	ne two source registers, stor	ring the result in the first source	e register			C	, i i i i i i i i i i i i i i i i i i i	Ŭ
	and-long	or-long	xor-long	shl-long	shr-long	ushr-long	add-float	sub-float	mul-float	div-float	rem-float	add-double	sub-double	mul-double	div-double	rem-double
AX							Perform the identifie	ed binary operation on th	ne two source registers, stor	ring the result in the first source	e register					
Вx	add-int/2addr	sub-int/2addr	mul-int/2addr	div-int/2addr	rem-int/2addr	and-int/2addr	or-int/2addr	xor-int/2addr	shl-int/2addr	shr-int/2addr	ushr-int/2addr	add-long/2addr	sub-long/2addr	mul-long/2addr	div-long/2addr	rem-long/2addr
							Perform the identifie	V 1	v	ring the result in the first source						
Сх	and-long/2addr	or-long/2addr	xor-long/2addr	shl-long/2addr	shr-long/2addr	ushr-long/2addr	add-float/2addr	sub-float/2addr	mul-float/2addr	div-float/2addr	rem-float/2addr	add-double/2addr	sub-double/2addr	mul-double/2addr	div-double/2addr	rem-double/2addr
ΟΛ								· · · · · · · · · · · · · · · · · · ·		ring the result in the first source	<u> </u>					
Dx	add-int/lit16	sub-int/lit16	mul-int/lit16	div-int/lit16	rem-int/lit16	and-int/lit16	or-int/lit16	xor-int/lit16	add-int/lit8	sub-int/lit8	mul-int/lit8	div-int/lit8	rem-int/lit8	and-int/lit8	or-int/lit8	xor-int/lit8
				ed register (first argu	ument) and literal value (s	econd argument), storing the	result in the destination	n register	H	Perform the indicated binary op	on the indicated register (fir	st argument) and literal value	(second argument), stor	ng the result in the dea	V	
Ex	shl-int/lit8	shr-int/lit8	ushr-int/lit8												execute-inline Executes the inline method identified by inline ID	
	enominate molicated binary op on the indicated register ((inst argument) and interar value (Second argi	amenty, storing the result in the destination register												Executes the milline method identified by inline ID	
	invoke-direct-empty		iget-quick	iget-wide-quick	iget-object-quick	iput-quick	iput-wide-quick	iput-object-quick	invoke-virtual-quick	invoke- virtual/range-quick	invoke-super-quick	invoke-super/range-quick				

	invoke-direct-empty	iget-quick	iget-wide-quick	iget-object-quick	iput-quick	iput-wide-quick	iput-object-quick	invoke-vi
Fx	Stands as a placeholder for pruned empty methods like Object <init>. This acts as nop-during normal execution</init>	Gets the value stored at offset in vy instance's data area to vx	Gats the object reference value stored at offset in vy instance's data area to vo.ve+1	Gets the object reference value stored at offset in vy instance's data area to vx	Puts the value stored in vx to offset in vy instance's data area	Puts the value stored in vx.vx+1 to offset in vy instance's data area	Puts the object reference value stored in vx to offset in vy instance's data area to vx	Invokes a virtual method us

misc	object	conversion
moves	flow	arithmetic
method	conditional	
literals	transfer	
system	logical	undefined

standard	4 => 4	
/from16	16 => 8	
/16	16 => 16	

ng the vtable of the target object Invokes a virtual method using the vtable of the target object invokes a virtual

Dalvik Virtual Machine (android)

Android	1 http://source.android.com/tech/dalvik/dalvik-bytecode.html
	2 http://pallergabor.uw.hu/androidblog/dalvik_opcodes.html
.Net	1 http://www.ecma-international.org/publications/standards/Ecma-335.htm p355-469
	2 http://www.asukaze.net/etc/cil/opcode.html
Java	http://java.sun.com/docs/books/jvms/second_edition/html/Instructions.doc.html