11/14/2001

Bytecode Table

b = byte
w = word (aligned)
o = oop (aligned)
grey: not intercepted on sst
yellow: not implemented

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	push temp #0	push temp #1	push temp #2	push temp #3	push temp #4	push temp #5		push temp #255-b [b]	push arg #n-1	push arg #n-2	push arg #n-3	push arg #n-1-b [b]	allocate 1 temp	allocate 2 temps	allocate 3 temps	allocate b temps (0 is 256) [b]
1	store & pop temp #0	store & pop temp #1	store & pop temp #2	store & pop temp #3	store & pop temp #4	store & pop temp #5	store temp #255-b [b]	store & pop temp #255-b [b]	push -b [b]	push b+1 [b]	push lit [o]	push tos	push self	push nil	push true	push false
2									^ instVar name [o]	push classVar [w]	store & pop classVar [w]	store classVar [w]	^ instVar [w]	push instVar [w]	store & pop instVar [w]	store instVar [w]
3	allocate floats (0 is 256) [b]	floatify [b]	move float [bb]	set float [bww]	nullary float op [bb]	unary float op [bb]	binary float op [bb]	unary float op > oop [bb]	binary float op > oop [bb]					push instVar name [o]	store & pop instVar name [o]	store instVar name [o]
4	push temp #0 via context 0	push temp #1 via context 0	push temp #2 via context 0	push temp #b via context 0 [b]	store & pop temp #0 via context 0	store & pop temp #1 via context 0	store & pop temp #2 via context 0	store & pop temp #b via context 0 [b]	push new closure with context (0 arg block) [o]	push new closure with context (1 arg block) [o]	push new closure with context (2 arg block) [o]	push new closure with context (b arg block) [bo]	install new context for method with 0 vars	install new context for method with 1 var	install new context for method with 2 vars	install new context for method with b vars [b]
5	push temp #0 via context 1	push temp #1 via context 1	push temp #2 via context 1	push temp #b via context 1 [b]	store & pop temp #0 via context 1	store & pop temp #1 via context 1	store & pop temp #2 via context 1	store & pop temp #b via context 1 [b]	push new closure with tos (0 arg block) [o]	push new closure with tos (1 arg block) [o]	push new closure with tos (2 arg block) [o]	push new closure with tos (b arg block) [bo]	pop tos	install new context for block with 1 var	install new context for block with 2 vars	install new context for block with b vars [b]
6	push temp #0 via context b (0 is 256) [b]	push temp #1 via context b (0 is 256) [b]	push temp #2 via context b (0 is 256) [b]	push temp #b1 via context b2 (0 is 256) [bb]	store & pop temp #0 via context b (0 is 256) [b]	store & pop temp #1 via context b (0 is 256) [b]	store & pop temp #2 via context b (0 is 256) [b]	store & pop temp #b1 via context b2 (0 is 256) [bb]	set self via context	copy 1 arg into context [b]	copy 2 args into context [bb]	copy b (0 is 256) args into context [b...]	copy self into context	copy self & 1 arg into context [b]	copy self & 2 args into context [bb]	copy self & b (0 is 256) args into context [b...]
7	ifTrue [bb]	ifFalse [bb]	and [b]	or [b]	whileTrue [b]	whileFalse [b]	jump [b]	loop start & jump [bb]	ifTrue [bw]	ifFalse [bw]	and [w]	or [w]	whileTrue [w]	whileFalse [w]	jump [w]	loop start & jump [ww]
8	interpreted send, 0 args [oo]	interpreted send, 1 arg [oo]	interpreted send, 2 args [oo]	interpreted send, n args [boo]	interpreted send & pop, 0 args [oo]	interpreted send & pop, 1 arg [oo]	interpreted send & pop, 2 args [oo]	interpreted send & pop, n args [boo]	interpreted send, self [oo]	interpreted send & pop, self [oo]	interpreted send, super [oo]	interpreted send & pop, super [oo]	return tos & pop 0 args	return tos & pop 1 arg	return tos & pop 2 args	return tos & pop n args [b]
9	polymorphic send, 0 args [oo]	polymorphic send, 1 arg [oo]	polymorphic send, 2 args [oo]	polymorphic send, n args [boo]	polymorphic send & pop, 0 args [oo]	polymorphic send & pop, 1 arg [oo]	polymorphic send & pop, 2 args [oo]	polymorphic send & pop, n args [boo]	polymorphic send, self [oo]	polymorphic send & pop, self [oo]	polymorphic send, super [oo]	polymorphic send & pop, super [oo]	return self & pop 0 args	return self & pop 1 arg	return self & pop 2 args	return self & pop n args [b]
A	compiled send, 0 args [wo]	compiled send, 1 arg [wo]	compiled send, 2 args [wo]	compiled send, n args [bwo]	compiled send & pop, 0 args [wo]	compiled send & pop, 1 arg [wo]	compiled send & pop, 2 args [wo]	compiled send & pop, n args [bwo]	compiled send, self [wo]	compiled send & pop, self [wo]	compiled send, super [wo]	compiled send & pop, super [wo]	zap, return tos & pop n args [b]	zap, return self & pop n args [b]	non-local return tos & pop n args [b]	non-local return self & pop n args [b]
B	primitive call [w]	predict ^ primitive call [w]	primitive call, failure block [ww]	predict ^ primitive call, failure block [ww]	sync. DLL call, n args [oowb]				self send, access method [oo]	primitive send, 0 args [oo]	primitive send, super [oo]	primitive send & pop, super [oo]		primitive send, 1 arg [oo]	primitive send, 2 args [oo]	primitive send, n args [boo]
C	primitive call lookup & patch [o]	primitive call lookup & patch [o]	primitive call lookup & patch [ow]	primitive call lookup & patch [ow]	async. DLL call, n args [oowb]	primitive call lookup & patch [o]	primitive call lookup & patch [ow]		send, access method [oo]	primitive send & pop, 0 args [oo]	primitive send, self [oo]	primitive send & pop, self [oo]		primitive send & pop, 1 arg [oo]	primitive send pop, 2 args [oo]	primitive send & pop, n args [boo]
D	megamorphic send, 0 args [oo]	megamorphic send, 1 arg [oo]	megamorphic send, 2 args [oo]	megamorphic send, n args [boo]	megamorphic send & pop, 0 args [oo]	megamorphic send & pop, 1 arg [oo]	megamorphic send & pop, 2 args [oo]	megamorphic send & pop, n args [boo]	megamorphic send, self [oo]	megamorphic send & pop, self [oo]	megamorphic send, super [oo]	megamorphic send & pop, super [oo]		special primitive call hint, 1 arg [b]		
E	smi +, predicted [oo]	smi -, predicted [oo]	smi *, predicted [oo]	smi //, predicted [oo]	smi \\, predicted [oo]	#@ [oo]	smi =, predicted [oo]	smi ~=, predicted [oo]	smi <, predicted [oo]	smi <=, predicted [oo]	smi >, predicted [oo]	smi >=, predicted [oo]	at:, predicted [oo]	at:put:, predicted [oo]	#==	~~
F	push globalVar [o]	store & pop globalVar [o]	store globalVar [o]	push classVar name [o]	store & pop classVar name [o]	store classVar name [o]	smi and, predicted [oo]	smi or, predicted [oo]	smi xor, predicted [oo]	smi shift predicted [oo]						halt