6
Referências Bibliográficas

ANTONIOLI D.N.; PILZ, M. **Analysis of the Java Class File Format**. Technical Report ifi-98.04, Department of Computer Science, University of Zurich, 1998.

BELL, J. R. **Threaded code**. Commun ACM, 16(6): 370-372, 2000.

CRAIG, I. D. **Virtual Machines**, Londres: Springer-Verlag, 2005, p. 157.

DALY, C. et al. **Platform Independent Dynamic Java Virtual Machine Analysis**: the Java Grande Forum Benchmark Suite. In: Joint ACM Java Grande - ISCOPE 2001 Conference, Stanford University, USA, 2001.

DAVIS, B. et al. **The case for virtual register machines**, In IVME '03: Proceedings of the 2003 workshop on Interpreters, Virtual machines and emulators, San Diego: ACM Press, p. 41-49, 2003.

DONOGHUE, D. et al. **Bigram analysis of Java bytecode sequences**, In PPPJ '02/IRE '02: Proceedings of the inaugural conference on the Principles and Practice of programming, 2002 and Proceedings of the second workshop on Intermediate representation engineering for virtual machines, National University of Ireland, p. 187-192, 2002.

DONOGHUE, D.; POWER, J. F. **Identifying and Evaluating a Generic Set of Superinstructions for Embedded Java Programs**, In ESA/VLSI, CSREA Press, p. 192-198, 2004.

DOWLING, T.; POWER, J. F.; WALDRON, J. **Relating Static and Dynamic Measurements for the Java Virtual Machine Instruction Set**. In Symposium on Mathematical Methods and Computational Techniques in Electronic Engineering, Athens, Greece, 2001.

ERTL, M. A.; GREGG, D. **The behaviour of efficient virtual machine interpreters on modern architectures**. In Euro-Par 2001, pages 403–412. Springer LNCS 2150, 2001.

ERTL, M. A.; GREGG, D. **Combining stack caching with dynamic superinstructions**, In IVME '04: Proceedings of the 2004 workshop on Interpreters, virtual machines and emulators, Washington, D.C.: ACM Press, p. 7-14, 2004.

ERTL, A. et al. **Virtual machine showdown**: stack versus registers, in Proceedings of the ACM/SIGPLAN Conference of Virtual Execution Environments (VEE 05), p. 153-163, Chicago Illinois, 2005.

HSIEH, C. A.; GYLLENHAAL, J. C.; HWU, W. **Java bytecode to native code translation**: the caffeine prototype and preliminary results, In MICRO 29: Proceedings of the 29th annual ACM/IEEE international symposium on Microarchitecture, IEEE Computer Society, p. 90-99, Paris, France, 1996.

PORTHOUSE, C. **Jazelle**: High performance Java on embedded devices, ARM Ltd.

PUGH, W. **Compressing Java class files**. In PLDI '99: Proceedings of the ACM SIGPLAN 1999 conference on Programming language design and implementation, ACM Press, p. 247-258, Atlanta, GE, 1999.

RAYSIDE, D.; MAMAS, E.; HONS E. **Compact Java binaries for embedded systems**, In CASCON '99: Proceedings of the 1999 conference of the Centre for Advanced Studies on Collaborative research, IBM Press, p. 9-22, Mississauga, Canada, 1999.

STEPHENSON, B.; HOLST W. **A quantitative analysis of Java bytecode sequences**, In PPPJ '04: Proceedings of the 3rd international symposium on Principles and practice of programming in Java, Trinity College Dublin, p. 15-20, Las Vegas, NV, 2004.

SULLIVAN, G. T. et al. **Dynamic native optimization of interpreters**, In IVME '03: Proceedings of the 2003 workshop on Interpreters, virtual machines and emulators, ACM Press, p. 50-57, San Diego, CA, 2003.

WALDRON, J. **Analysis of Virtual Machine Stack Frame Usage by Java Methods**. Internet, Multimedia Systems and Applications, p. 271-274, 1999.

WALDRON, J. et al. **Comparison of factors influencing bytecode usage in the Java Virtual Machine**. In Second International Conference and Exhibition on the Practical Application of Java, p. 315-327, Manchester, UK, 2000.

YIHUEY, LI **Runrime Performance Evaluation of Junst-In-Time Compiler Enabled J9 Virtual Machine**, Arizona State University East, AZ, 2004.

IGOR PAVLOV, **7Zip Compression Tool**, http://www.sevenzip.org, 1999

BCEL - **Byte Code Engineering Library**, http://jakarta.apache.org/bcel.

SUPERWABA - **Virtual Machine for PDAs**, http://www.superwaba.com.br

IBM J9 - **WebSphere Everyplace Micro Environment**

http://www.ibm.com/developerworks/websphere/zones/wireless/weme_eval_runtimes.html

7
Apêndice

Neste apêndice colocamos tabelas que complementam as informações dos capítulos anteriores. Na seção 7.1 temos a lista dos bytecodes Java. Na 7.2, o mapeamento desses bytecodes nas instruções que criamos. Por fim, na seção 7.3, listamos o código gerado pelos compiladores VERA e JavaC do JDK 1.2.2.

7.1.
Lista dos bytecodes Java

A tabela 32 exibe os bytecodes Java na ordem definida pelo JCP, seguindo uma categorização que criamos.

CATEGORIA	Movimentação e Conversão	Aritméticos e Lógicos	Desvios	Outros

	0	35	70	105	140	175
0	NOP	FLOAD_1	FSTORE_3	LMUL	F2L	DRETURN
1	ACONST_NULL	FLOAD_2	DSTORE_0	FMUL	F2D	ARETURN
2	ICONST_M1	FLOAD_3	DSTORE_1	DMUL	D2I	RETURN
3	ICONST_0	DLOAD_0	DSTORE_2	IDIV	D2L	GETSTATIC
4	ICONST_1	DLOAD_1	DSTORE_3	LDIV	D2F	PUTSTATIC
5	ICONST_2	DLOAD_2	ASTORE_0	FDIV	I2B	GETFIELD
6	ICONST_3	DLOAD_3	ASTORE_1	DDIV	I2C	PUTFIELD
7	ICONST_4	ALOAD_0	ASTORE_2	IREM	I2S	INVOKEVIRTUAL
8	ICONST_5	ALOAD_1	ASTORE_3	LREM	LCMP	INVOKESPECIAL
9	LCONST_0	ALOAD_2	IASTORE	FREM	FCMPL	INVOKESTATIC
10	LCONST_1	ALOAD_3	LASTORE	DREM	FCMPG	INVOKEINTERFACE
11	FCONST_0	IALOAD	FASTORE	INEG	DCMPL	
12	FCONST_1	LALOAD	DASTORE	LNEG	DCMPG	NEW
13	FCONST_2	FALOAD	AASTORE	FNEG	IFEQ	NEWARRAY
14	DCONST_0	DALOAD	BASTORE	DNEG	IFNE	ANEWARRAY
15	DCONST_1	AALOAD	CASTORE	ISHL	IFLT	ARRAYLENGTH
16	BIPUSH	BALOAD	SASTORE	LSHL	IFGE	ATHROW
17	SIPUSH	CALOAD	POP	ISHR	IFGT	CHECKCAST
18	LDC	SALOAD	POP2	LSHR	IFLE	INSTANCEOF
19	LDC_W	ISTORE	DUP	IUSHR	IF_ICMPEQ	MONITORENTER
20	LDC2_W	LSTORE	DUP_X1	LUSHR	IF_ICMPNE	MONITOREXIT
21	ILOAD	FSTORE	DUP_X2	IAND	IF_ICMPLT	WIDE
22	LLOAD	DSTORE	DUP2	LAND	IF_ICMPGE	MULTIANEWARRAY
23	FLOAD	ASTORE	DUP2_X1	IOR	IF_ICMPGT	IFNULL
24	DLOAD	ISTORE_0	DUP2_X2	LOR	IF_ICMPLE	IFNONNULL
25	ALOAD	ISTORE_1	SWAP	IXOR	IF_ACMPEQ	GOTO_W
26	ILOAD_0	ISTORE_2	IADD	LXOR	IF_ACMPNE	JSR_W
27	ILOAD_1	ISTORE_3	LADD	IINC	GOTO	BREAKPOINT
28	ILOAD_2	LSTORE_0	FADD	I2L	JSR	
29	ILOAD_3	LSTORE_1	DADD	I2F	RET	
30	LLOAD_0	LSTORE_2	ISUB	I2D	TABLESWITCH	
31	LLOAD_1	LSTORE_3	LSUB	L2I	LOOKUPSWITCH	
32	LLOAD_2	FSTORE_0	FSUB	L2F	IRETURN	
33	LLOAD_3	FSTORE_1	DSUB	L2D	LRETURN	
34	FLOAD_0	FSTORE_2	IMUL	F2I	FRETURN	

Tabela 32: Bytecodes Java

7.2. Mapeamento entre os bytecodes Java e as novas instruções

Na tabela 33 exibimos um mapeamento de todos os bytecodes da linguagem Java com o equivalente na nova especificação. Desta forma, garantiremos que ela cobre a linguagem Java.

De maneira geral, as conversões são feitas de forma direta: `load` e `store` são convertidas em `mov`, as operações aritméticas e lógicas são convertidas para a operação correspondente, usando registradores como operandos. As operações com float foram ignoradas, pois o tipo float foi eliminado, como já visto.

Instruções que apenas manipulam a pilha foram marcadas como n/a (não aplicável). A instrução `nop` não foi implementada.

As instruções `jsr/ret` não foram implementadas, pois o suporte a `finally` será implementado de outra forma, colocando o conteúdo *inlined*. Muitas vezes o `finally` é usado de forma incorreta. Por exemplo: `try {a} catch {b} finally {c}` equivale a `try {a} catch {b} {c}`, que não requer o `jsr/ret` para ser implementado. O único caso em que ele é necessário é quando usado sem o `catch`: `try {a} finally {c}`.

Bytecode Java	Novas instruções
AALOAD	mov regO, regO[regI]
AASTORE	mov regO[regI], regO
ACONST_NULL	mov regO, null
ALOAD	mov regO, regO
ALOAD_0	mov regO, regO
ALOAD_1	mov regO, regO
ALOAD_2	mov regO, regO
ALOAD_3	mov regO, regO
ANEWARRAY	newArray
ARETURN	return reg
ARRAYLENGTH	mov regI, regO.tamvet
ASTORE	mov regO, regO
ASTORE_0	mov regO, regO
ASTORE_1	mov regO, regO
ASTORE_2	mov regO, regO
ASTORE_3	mov regO, regO

Bytecode Java	Novas instruções
ICONST_4	mov regI, 4
ICONST_5	mov regI, 5
ICONST_M1	mov regI, -1
IDIV	div regI, regI, regI
IF_ACMPEQ	jeq regO, regO, delta
IF_ACMPNE	jne regO, regO, delta
IF_ICMPEQ	jeq regI, regI, delta
IF_ICMPGE	jge regI, regI, delta
IF_ICMPGT	jgt regI, regI, delta
IF_ICMPLE	jle regI, regI, delta
IF_ICMPLT	jlt regI, regI, delta
IF_ICMPNE	jne regI, regI, delta
IFEQ	jeq reg?, 0, delta
IFGE	jge reg?, 0, delta
IFGT	jgt reg?, 0, delta
IFLE	jle reg?, 0, delta

Bytecode Java	Novas instruções
ATHROW	throw
BALOAD	mov regIb, regO[regI]
BASTORE	mov regO[regI], regIb
BIPUSH	mov regI, valor
BREAKPOINT	break
CALOAD	mov regIc, regO[regI]
CASTORE	mov regO[regI], regIc
CHECKCAST	checkcast
D2F	n/a
D2I	D2I regI, regD
D2L	D2L regL, regD
DADD	add regD, regD, regD
DALOAD	mov regO[regI], regD
DASTORE	mov regD, regO[regI]
DCMPG	JGT regD, regD, delta
DCMPL	JLT regD, regD, delta
DCONST_0	mov regD, 0
DCONST_1	mov regD, 1
DDIV	div regD, regD, regD
DLOAD	mov regD, regD
DLOAD_0	mov regD, regD
DLOAD_1	mov regD, regD
DLOAD_2	mov regD, regD
DLOAD_3	mov regD, regD
DMUL	mul regD, regD, regD
DNEG	mov regD,0; sub regD, regD(=0), regD
DREM	mod regD, regD, regD
DRETURN	ret regD
DSTORE	mov regD, regD
DSTORE_0	mov regD, regD
DSTORE_1	mov regD, regD
DSTORE_2	mov regD, regD
DSTORE_3	mov regD, regD
DSUB	sub regD, regD, regD
DUP	n/a

Bytecode Java	Novas instruções
IFLT	jlt reg?, 0, delta
IFNE	jne reg?, 0, delta
IFNONNULL	jne regO, 0
IFNULL	jeq regO, 0
IINC	add regI, valor
ILOAD	mov regI, regI
ILOAD_0	mov regI, regI
ILOAD_1	mov regI, regI
ILOAD_2	mov regI, regI
ILOAD_3	mov regI, regI
IMUL	mul regI, regI, regI
INEG	sub regI, 0, regI
INSTANCEOF	instanceof
INVOKEINTERFACE	CallExternal
INVOKESPECIAL	CallInternal
INVOKESTATIC	CallInternal/CallExternal
INVOKEVIRTUAL	CallInternal/CallExternal
IOR	or regI, regI, regI
IREM	mod regI, regI, regI
IRETURN	ret regI
ISHL	shl regI, regI, regI
ISHR	sshr regI, regI, regI
ISTORE	mov regI, regI
ISTORE_0	mov regI, regI
ISTORE_1	mov regI, regI
ISTORE_2	mov regI, regI
ISTORE_3	mov regI, regI
ISUB	sub regI, regI, regI
IUSHR	ushr regI, regI, regI
IXOR	xor regI, regI, regI
JSR	***
JSR_W	***
L2D	L2D regD, regL
L2F	n/a
L2I	L2I regL, regI

Bytecode Java	Novas instruções
DUP_X1	n/a
DUP_X2	n/a
DUP2	n/a
DUP2_X1	n/a
DUP2_X2	n/a
F2D	n/a
F2I	n/a
F2L	n/a
FADD	n/a
FALOAD	n/a
FASTORE	n/a
FCMPG	n/a
FCMPL	n/a
FCONST_0	n/a
FCONST_1	n/a
FCONST_2	n/a
FDIV	n/a
FLOAD	n/a
FLOAD_0	n/a
FLOAD_1	n/a
FLOAD_2	n/a
FLOAD_3	n/a
FMUL	n/a
FNEG	n/a
FREM	n/a
FRETURN	n/a
FSTORE	n/a
FSTORE_0	n/a
FSTORE_1	n/a
FSTORE_2	n/a
FSTORE_3	n/a
FSUB	n/a
GETFIELD	mov reg?, [cpIntr,cpExtr]
GETSTATIC	mov reg?, [estInt,estExtr]

Bytecode Java	Novas instruções
LADD	add regL, regL, regL
LALOAD	mov regL, regO[regI]
LAND	and regL, regL, regL
LASTORE	mov regO[regI], regL
LCMP	j<cond> regL, regL, delta
LCONST_0	mov regL, 0
LCONST_1	mov regL, 1
LDC	mov regI, simb
LDC_W	mov regI, simb
LDC2_W	mov regL, simb mov regD, simb
LDIV	div regL, regL, regL
LLOAD	mov regL, regL
LLOAD_0	mov regL, regL
LLOAD_1	mov regL, regL
LLOAD_2	mov regL, regL
LLOAD_3	mov regL, regL
LMUL	mul regL, regL, regL
LNEG	mov regL,0; sub regL, regL(0), regL
LOOKUPSWITCH	switch
LOR	or regL, regL, regL
LREM	mod regL, regL, regL
LRETURN	ret regL
LSHL	shl regL, regL, regL
LSHR	sshr regL, regL, regL
LSTORE	mov regL, regL
LSTORE_0	mov regL, regL
LSTORE_1	mov regL, regL
LSTORE_2	mov regL, regL
LSTORE_3	mov regL, regL
LSUB	sub regL, regL, regL
LUSHR	ushr regL, regL, regL
LXOR	xor regL, regL, regL
MONITORENTER	syncStart
MONITOREXIT	syncEnd

Bytecode Java	Novas instruções
GOTO	jmp s24
GOTO_W	jmp s24
I2B	I2B regI, regI
I2C	I2C regI, regI
I2D	I2D regD, regI
I2F	n/a
I2L	I2L regL, regI
I2S	I2S regI, regI
IADD	add regI, regI, regI
IALOAD	mov regI, regO[regI]
IAND	and regI, regI, regI
IASTORE	mov regO[regI], regI
ICONST_0	mov regI, 0
ICONST_1	mov regI, 1
ICONST_2	mov regI, 2
ICONST_3	mov regI, 3

Bytecode Java	Novas instruções
MULTIANEWARRAY	vários new array
NEW	new obj
NEWARRAY	new array
NOP	***
POP	n/a
POP2	n/a
PUTFIELD	mov [cpIntr,cpExtr], reg?
PUTSTATIC	mov [estIntr,estExtr], reg?
RET	***
RETURN	return
SALOAD	mov regIs, regO[regI]
SASTORE	mov regO[regI], regIs
SIPUSH	mov regI, valor
SWAP	n/a
TABLESWITCH	switch
WIDE	n/a

Tabela 33: Conversão entre bytecodes Java e as novas instruções.

***: não existe instrução correspondente n/a: não aplicável

7.3.
Listagem do código gerado pelos compiladores

A tabela a seguir mostra a listagem do código gerado pelo compilador VERA e pelo compilador do JDK 1.2.2, excluindo o construtor e os métodos `print` e `getTimeStamp`. Em primeiro lugar, são exibidos os campos da classe, em seguida as informações dos métodos com os códigos, e por fim, os dados contidos nas tabelas de constantes.

VERA	**JDK 1.2.2**

Class Name : AllTests
Source File: ./AllTests
AccessFlag : Public InfoIsHere
Instance Fields (I32):
  Name: 4097 - fieldI
  Type: 6
  AccessFlag: Private
  Value: 0
 	
  Name: 4101 - age
  Type: 6
  AccessFlag: Private
  Value: 0
 	
Instance Fields (I64):
  Name: 4100 - fieldL
  AccessFlag: Private
  Value: 0
 	
Instance Fields (Dbl):
  Name: 4098 - fieldF
  AccessFlag: Private
  Value: 0.000000
 	
  Name: 4099 - fieldD
  AccessFlag: Private
  Value: 0.000000
 	
Methods:
	Decompiled by Decafe PRO
	Classes: 1
	Methods: 17
	Fields: 5
	
	private int fieldI;
	private float fieldF;
	private double fieldD;
	private long fieldL;
	private int age;

Name: 4102 - testLocal
AccessFlag: Public
Parameters Count: 0
Return Type: 0 -
How many registers:
iCount: 2
Code:
int j = 0;
1 MOV_regI_s18 : 0 <- #0
for (int i=10000000; i > 0; i--)
2 MOV_regI_sym : 1 <- sym#1
3 JUMP_s24 : -> 2
j += 2;
4 INC_regI : 0 <- 0 #2
5 DECJGTZ_regI : 1 -> -1
6 RETURN_void :
	int i = 0;
	1 0:iconst_0
	2 1:istore_1
	for(int j = 0x989680; j > 0; j--)
	3 2:ldc1 #1 <Int 0x989680>
	4 4:istore_2
	5 5:goto 14
	i += 2;
	6 8:iinc 1 2
	7 11:iinc 2 -1
	8 14:iload_2
	9 15:ifgt 8
	10 18:return

VERA	**JDK 1.2.2**

Name: 4103 - testFieldI
AccessFlag: Public
Parameters Count: 0
Return Type: 0 -
How many registers:
iCount: 3
Code:
int c = 2;
1 MOV_regI_s18 : 0 <- #2
for (int i=10000000; i > 0; i--)
2 MOV_regI_sym : 1 <- sym#1
3 JUMP_s24 : -> 3
fieldI += c;
4 MOV_regI_thisField : 2 <- 0
5 ADD_thisField_regI_regI :
6 DECJGTZ_regI : 1 -> -2
7 RETURN_void :
	byte byte0 = 2;
	1 0:iconst_2
	2 1:istore_1
	for(int i = 0x989680; i > 0; i--)
	3 2:ldc1 #1 <Int 0x989680>
	4 4:istore_2
	5 5:goto 21
	fieldI += byte0;
	6 8:aload_0
	7 9:dup
	8 10:getfield #8 <int fieldI>
	9 13:iload_1
	10 14:iadd
	11 15:putfield #8 <int fieldI>
	12 18:iinc 2 -1
	13 21:iload_2
	14 22:ifgt 8
	15 25:return

Name: 4104 - testFieldF
AccessFlag: Public
Parameters Count: 0
Return Type: 0 -
How many registers:
iCount: 1
dCount: 3
Code:
float c = 2;
1 MOV_regD_s18 : 0 <- #2
for (int i=10000000; i > 0; i--)
2 MOV_regI_sym : 0 <- sym#1
3 JUMP_s24 : -> 4
fieldF += c;
4 MOV_regD_thisField : 2 <- 1
5 ADD_regD_regD_regD : 1 <- 2 0
6 MOV_thisField_regD : 1 <- 1
7 DECJGTZ_regI : 0 -> -3
8 RETURN_void :
	float f = 2.0F;
	1 0:fconst_2
	2 1:fstore_1
	for(int i = 0x989680; i > 0; i--)
	3 2:ldc1 #1 <Int 0x989680>
	4 4:istore_2
	5 5:goto 21
	fieldF += f;
	6 8:aload_0
	7 9:dup
	8 10:getfield #7 <float fieldF>
	9 13:fload_1
	10 14:fadd
	11 15:putfield #7 <float fieldF>
	12 18:iinc 2 -1
	13 21:iload_2
	14 22:ifgt 8
	15 25:return

Name: 4105 - testFieldD
AccessFlag: Public
Parameters Count: 0
Return Type: 0 -
How many registers:
iCount: 1
dCount: 3
Code:
double c = 2;
1 MOV_regD_s18 : 0 <- #2
for (int i=10000000; i > 0; i--)
2 MOV_regI_sym : 0 <- sym#1
3 JUMP_s24 : -> 4
fieldD += c;
4 MOV_regD_thisField : 2 <- 2
5 ADD_regD_regD_regD : 1 <- 2 0
6 MOV_thisField_regD : 2 <- 1
7 DECJGTZ_regI : 0 -> -3
8 RETURN_void :
	double d = 2D;
	1 0:ldc2w #17 <Double 2D>
	2 3:dstore_1
	for(int i = 0x989680; i > 0; i--)
	3 4:ldc1 #1 <Int 0x989680>
	4 6:istore_3
	5 7:goto 23
	fieldD += d;
	6 10:aload_0
	7 11:dup
	8 12:getfield #6 <double fieldD>
	9 15:dload_1
	10 16:dadd
	11 17:putfield #6 <double fieldD>
	12 20:iinc 3 -1
	13 23:iload_3
	14 24:ifgt 10
	15 27:return

Name: 4106 - testFieldL
AccessFlag: Public
Parameters Count: 0
Return Type: 0 -
How many registers:
iCount: 1
lCount: 3
	long l = 2L;
	1 0:ldc2w #13 <Long 2L>
	2 3:lstore_1
	for(int i = 0x989680; i > 0; i--)
	3 4:ldc1 #1 <Int 0x989680>
	4 6:istore_3
	5 7:goto 23

VERA	**JDK 1.2.2**

Code:
long c = 2;
1 MOV_regL_s18 : 0 <- #2
for (int i=10000000; i > 0; i--)
2 MOV_regI_sym : 0 <- sym#1
3 JUMP_s24 : -> 4
fieldL += c;
4 MOV_regL_thisField : 2 <- 3
5 ADD_regL_regL_regL : 1 <- 2 0
6 MOV_thisField_regL : 3 <- 1
7 DECJGTZ_regI : 0 -> -3
8 RETURN_void :
	fieldL += l;
	6 10:aload_0
	7 11:dup
	8 12:getfield #9 <long fieldL>
	9 15:lload_1
	10 16:ladd
	11 17:putfield #9 <long fieldL>
	12 20:iinc 3 -1
	13 23:iload_3
	14 24:ifgt 10
	15 27:return

Name: 4107 - testMethod1
AccessFlag: Public
Parameters Count: 0
Return Type: 0 -
How many registers:
iCount: 1
Code:
for (int i=10000000; i > 0; i--)
1 MOV_regI_sym : 0 <- sym#1
2 JUMP_s24 : -> 3
method(50,30,150,true);
3 CALL_internal : 8-> 0 par: 51
4 DECJGTZ_regI : 0 -> -2
5 RETURN_void :
	for(int i = 0x989680; i > 0; i--)
	1 0:ldc1 #1 <Int 0x989680>
	2 2:istore_1
	3 3:goto 22
	method(50, 30D, 150L, true);
	4 6:aload_0
	5 7:bipush 50
	6 9:ldc2w #19 <Double 30D>
	7 12:ldc2w #15 <Long 150L>
	8 15:iconst_1
	9 16:invokevirtual #11 <void method(int, double, long, boolean)>
	10 19:iinc 1 -1
	11 22:iload_1
	12 23:ifgt 6
	13 26:return

Name: 4108 - testMethod2
AccessFlag: Public
Parameters Count: 0
Return Type: 0 -
How many registers:
iCount: 3
dCount: 1
lCount: 1
Code:
int i1 = 50;
1 MOV_regI_s18 : 0 <- #50
double d1 = 30;
2 MOV_regD_s18 : 0 <- #30
long l1 = 150;
3 MOV_regL_s18 : 0 <- #150
boolean b1 = true;
4 MOV_regI_s18 : 1 <- #1
for (int i=10000000; i > 0; i--)
5 MOV_regI_sym : 2 <- sym#1
6 JUMP_s24 : -> 3
method(i1,d1,l1,b1);
7 CALL_internal : 8 -> 0 par: 0
8 DECJGTZ_regI : 2 -> -2
9 RETURN_void :
	byte byte0 = 50;
	1 0:bipush 50
	2 2:istore_1
	double d = 30D;
	3 3:ldc2w #19 <Double 30D>
	4 6:dstore_2
	long l = 150L;
	5 7:ldc2w #15 <Long 150L>
	6 10:lstore 4
	boolean flag = true;
	7 12:iconst_1
	8 13:istore 6
	for(int i = 0x989680; i > 0; i--)
	9 15:ldc1 #1 <Int 0x989680>
	10 17:istore 7
	11 19:goto 35
	method(byte0, d, l, flag);
	12 22:aload_0
	13 23:iload_1
	14 24:dload_2
	15 25:lload 4
	16 27:iload 6
	17 29:invokevirtual #11 <void method(int,double,long, boolean)>
	18 32:iinc 7 -1
	19 35:iload 7
	20 37:ifgt 22
	21 40:return

Name: 4109 - testArray
AccessFlag: Public
Parameters Count: 0
Return Type: 0 -
How many registers:
iCount: 4
oCount: 1
	int ai[] = new int[10];
	1 0:bipush 10
	2 2:newarray int[]
	3 4:astore_1
	for(int i = 0x989680; i > 0; i--)
	4 5:ldc1 #1 <Int 0x989680>

VERA	**JDK 1.2.2**

Code:
int[] a = new int[10];
1 NEWARRAY_len : 0 = sym#16, #10
for (int i=10000000; i > 0; i--)
2 MOV_regI_sym : 0 <- sym#1
3 JUMP_s24 : -> 5
a[5] += i;
4 MOV_regI_s18 : 1 <- #5
5 MOV_regI_aru : 3 <- 0[1]
6 ADD_regI_regI_regI : 2 <- 3 0
7 MOV_aru_regI : 0[1] <- 2
8 DECJGTZ_regI : 0 -> -4
9 RETURN_void :
	5 7:istore_2
	6 8:goto 21
	ai[5] += i;
	7 11:aload_1
	8 12:iconst_5
	9 13:dup2
	10 14:iaload
	11 15:iload_2
	12 16:iadd
	13 17:iastore
	14 18:iinc 2 -1
	15 21:iload_2
	16 22:ifgt 11
	17 25:return

Name: 4110 - method AccessFlag: Public Parameters Count: 4 Return Type: 0 - How many registers: iCount: 2 dCount: 1 lCount: 1 Index of Parameters: 6 9 7 2 Register Types of Parameters: 0 2 3 0 Code: 1 [177] RETURN_void :	1 0:return

Name: 4111 - testSet
AccessFlag: Public
Parameters Count: 0
Return Type: 0 -
How many registers:
iCount: 1
Code:
for (int i=10000000; i > 0; i--)
1 MOV_regI_sym : 0 <- sym#1
2 JUMP_s24 : -> 2
setAge(10);
3 CALL_internal : 11-> 0 par: 11
4 DECJGTZ_regI : 0 -> -1
5 RETURN_void :
	for(int i = 0x989680; i > 0; i--)
	1 0:ldc1 #1 <Int 0x989680>
	2 2:istore_1
	3 3:goto 15
	setAge(10);
	4 6:aload_0
	5 7:bipush 10
	6 9:invokevirtual #12
	<Method void setAge(int)>
	7 12:iinc 1 -1
	8 15:iload_1
	9 16:ifgt 6
	10 19:return

Name: 4112 - testGet
AccessFlag: Public
Parameters Count: 0
Return Type: 0 -
How many registers:
iCount: 2
Code:
for (int i=10000000; i > 0; i--)
1 MOV_regI_sym : 1 <- sym#1
2 JUMP_s24 : -> 2
a = getAge();
3 CALL_internal : 12-> 0 ret: 0
4 DECJGTZ_regI : 1 -> -1
5 RETURN_void :
	for(int j = 0x989680; j > 0; j--
	)
	1 0:ldc1 #1 <Int 0x989680>
	2 2:istore_2
	3 3:goto 14
	{
	int i = getAge();
	4 6:aload_0
	5 7:invokevirtual #10
	<Method int getAge()>
	6 10:istore_1
	}
	7 11:iinc 2 -1
	8 14:iload_2
	9 15:ifgt 6
	10 18:return

Name: 4113 - setAge AccessFlag: Public Parameters Count: 1 Return Type: 0 - How many registers:	

VERA	**JDK 1.2.2**
iCount: 1 Index of Parameters: 6 Register Types of Parameters: 0 Code: age = a; 1 MOV_thisField_regI : 4 <- 0 2 RETURN_void :	age = i; 1 0:aload_0 2 1:iload_1 3 2:putfield #5 <int age> 4 5:return
Name: 4114 - getAge AccessFlag: Public Parameters Count: 0 Return Type: 6 - I How many registers: iCount: 1 Code: return age; 1 MOV_regI_thisField : 0 <- 4 2 RETURN_regI : 0	return age; 1 0:aload_0 2 1:getfield #5 <int age> 3 4:ireturn
ConstantPool: i32s: i32: 10000000 Class Idents: name: AllTests Method/Field Idents: name: fieldI name: fieldF name: fieldD name: fieldL name: age name: testLocal name: testFieldI name: testFieldF name: testFieldD name: testFieldL name: testMethod1 name: testMethod2 name: testArray name: method name: testSet name: testGet name: setAge name: getAge name: getTimeStamp name: print	1: INTEGER: int_value=10000000 2: CLASS: name=AllTests 3: CLASS: name=java/lang/Object 4: METHODREF: class=java/lang/Object, na- me=<init>, type=()V 5: FIELDREF: class=AllTests, name=age, type=I 6: FIELDREF: class=AllTests, name=fieldD, type=D 7: FIELDREF: class=AllTests, name=fieldF, type=F 8: FIELDREF: class=AllTests, name=fieldI, type=I 9: FIELDREF: class=AllTests, name=fieldL, type=J 10: METHODREF: class=AllTests, name=getAge, type=()I 11: METHODREF: class=AllTests, name=method, type=(IDJZ)V 12: METHODREF: class=AllTests, name=setAge, type=(I)V 13: LONG: long_value=2 15: LONG: long_value=150 17: DOUBLE: double_value=2.0 19: DOUBLE: double_value=30.0 21: NAMEANDTYPE: name=<init>, descriptor=()V 22: NAMEANDTYPE: name=age, descriptor=I 23: NAMEANDTYPE: name=fieldD, descriptor=D 24: NAMEANDTYPE: name=fieldF, descriptor=F 25: NAMEANDTYPE: name=fieldI, descriptor=I 26: NAMEANDTYPE: name=fieldL, descriptor=J 27: NAMEANDTYPE: name=getAge, descriptor=()I 28: NAMEANDTYPE: name=method, descriptor=(IDJZ)V 29: NAMEANDTYPE: name=setAge, descriptor=(I)V 30: UTF8: string=()I 31: UTF8: string=()V

VERA	**JDK 1.2.2**
	32: UTF8: string=(I)V 33: UTF8: string=(IDJZ)V 34: UTF8: string=<init> 35: UTF8: string=AllTests 36: UTF8: string=Code 37: UTF8: string=D 38: UTF8: string=F 39: UTF8: string=I 40: UTF8: string=J 41: UTF8: string=age 42: UTF8: string=fieldD 43: UTF8: string=fieldF 44: UTF8: string=fieldI 45: UTF8: string=fieldL 46: UTF8: string=getAge 47: UTF8: string=java/lang/Object 48: UTF8: string=method 49: UTF8: string=setAge 50: UTF8: string=testArray 51: UTF8: string=testFieldD 52: UTF8: string=testFieldF 53: UTF8: string=testFieldI 54: UTF8: string=testFieldL 55: UTF8: string=testGet 56: UTF8: string=testLocal 57: UTF8: string=testMethod1 58: UTF8: string=testMethod2 59: UTF8: string=testSet

Tabela 34: Listagem do código gerado pelos compiladores VERA e JDK 1.2.2.