

Efficient JavaVM Just-in-Time

at run time. Translating the stack operations to native code is done by translating the operations back to expressions represented as directed acyclic graphs as an intermediate step. In [6] he translates Forth to native code using C as an intermediate language. In this system the stack slots

The second pass of the compiler translates each JavaVM `load` or `store` instruction into a corresponding intermediate code `MOVE` instruction using a new register as the destination register in the case of a `load`.

stack depth	0	1	2	3	4	5	6	>6
occurrences	7930	258	136	112	36	8	3	0

Table 1. distribution of stack depth at block boundary

value. To compute the information the run time stack is simulated at compile time. Instead of values the compile time stack contains the type of the value, if a local variable was loaded to a constant and similar information. Adl-Tabatabai [1] used a dynamic if

chain length	1	2	3	4	5	6	7	8	9	>9
occurrences	1892	62	23	62	30	11	41	9	7	65

Table 3. distribution of creator-store distances

	sieve	JavaLex	javac	espresso	Toba	java_cup
run time on 21164A 600MHz						

	sieve	JavaLex	javac	espresso	Toba	java_cup
run time on 21064A 300MHz (in seconds)						
JDK	83.2	29.8	18.5	8.7	32.1	3.5
Digital JIT kaf	6.27	84.4	47.6	14.1	-	9.8

java.io.ByteArrayOutputStream.write (int)void

Local Table:

0: (addr) S15
1: (int) S14
2: (int) S13
3: (int) S12 (addr) S12

Interface Table:

0: (int) T24

[L00]	0	ALOAD	0		
[T23]	1	GETFIELD	16		
[L02]	2	IADDCONST	1		
[L02]	3	NOP			
		[]	4	ISTORE	2
		[L02]	5	ILOAD	2
		[L00 L02]	6	ALOAD	0
		[T23 L02]	7	GETFIELD	8
		[T23 L02]	8	ARRAYLENGTH	

[] 9 IF_ICMPLE L005
.....

07.63 25.51 45TdRETURN 47.87 0 Td (18)Tj 23.86 0 Td (IF_I

Figure 4 Example: intermediate instructions and