Table 7.3: Key benchmark characteristics, using optimised source code

	B.sort	H.sort	Bin.Search	XXTEA	MD5	RC5	FFT	Outlier	LEC	CoreMark	MoteTrack	HeatCalib	HeatDetect	average
CODE SIZE (BYTES)														
Bytecode	74	134	83	379	2983	453	441	287	334	2788	2552	310	2661	
Native C	118	298	146	1442	9458	910	1292	380	560	6128	3906	1944	5294	
AOT original	418	1012	412	3792	29502	4090	2576	1402	1628	13982	12784	2454	17248	
AOT optimised	258	596	310	2236	14654	2018	1324	800	1056	8990	8478	1610	10346	
EXECUTED BYTECODE INSTRUCTIONS (% of total executed bytecode instructions before optimisation)														
Load/Store	79.8	71.7	58.1	44.9	43.3	41.1	61.1	69.0	59.5	54.1	70.3	51.8	48.0	57.9
Constant load	0.2	8.1	11.0	12.5	19.1	17.6	6.4	0.6	7.9	10.0	5.4	10.1	16.6	9.7
Processing	8.0	7.8	14.8	32.4	28.9	36.6	18.0	13.0	12.7	14.0	5.9	17.9	10.3	16.9
math	8.0	5.5	10.3	10.1	12.5	10.7	11.6	13.0	7.1	8.2	5.9	3.7	9.4	8.9
bit shift	0.0	2.2	4.5	8.1	5.4	8.0	6.1	0.0	3.8	2.2	0.0	8.5	0.9	3.8
bit logic	0.0	0.0	0.0	14.2	11.0	17.9	0.3	0.0	1.9	3.6	0.0	5.7	0.0	4.2
Branches	12.0	10.9	15.5	4.0	5.8	2.3	5.1	17.4	10.5	16.0	13.6	14.7	19.2	11.3
Invoke	0.0	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.9	0.3	0.0	0.1
Others	0.0	1.0	0.6	0.2	2.5	2.4	9.4	0.0	7.1	4.7	2.2	4.2	5.9	3.1
Total	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
STACK (bytes)														
Max. stack	6	8	4	24	20	14	10	6	18	16	12	22	16	13.5
Avg. stack	2.08	2.37	2.14	11.76	6.30	6.77	3.36	1.89	2.73	3.15	2.19	4.83	3.08	4.1
	B.sort	H.sort	Bin.Search	XXTEA	MD5	RC5	FFT	Outlier	LEC	CoreMark	MoteTrack	HeatCalib	HeatDetect	average

Table 7.4: Performance data per benchmark

	B.sort	H.sort	Bin.Search	XXTEA	MD5	RC5	FFT	Outlier	LEC	CoreMark	MoteTrack	HeatCalib	HeatDetect	average
PERFORMANCE OVERHEAD USING ORIGINAL SOURCE (% of native C)														
Total	1277.1	1927.2	1319.4	714.5	470.6	409.9	437.8	549.0	885.3	809.7	1018.7	210.2	203.9	787.2
push/pop	640.1	356.7	233.7	197.2	115.7	70.1	66.6	207.2	106.6	220.4	166.5	80.9	78.8	195.4
load/store	360.1	197.4	175.3	67.0	46.7	33.2	29.3	190.3	110.7	136.8	218.2	67.6	43.8	129.0
mov(w)	10.0	41.1	8.4	6.6	3.6	0.1	5.2	21.5	5.1	5.5	38.6	-3.0	9.5	11.7
other	266.9	331.4	902.1	82.8	104.0	67.8	76.8	130.1	370.6	234.2	220.0	37.4	65.6	222.3
vm	0.0	1000.6	0.0	361.1	200.4	238.7	260.0	-0.1	292.2	212.9	375.4	27.3	6.2	228.8
OVERHEAD REDUCTION FROM SOURCE CODE OPTIMISATION (% of native C)														
Source optimisation	-613.2	-1234.0	-843.6	-464.1	-244.2	-285.6	-315.0	-56.5	-612.7	-433.7	-227.9	0.0	1.7	-409.9
PERFORMANCE OVERHEAD BEFORE COMPILER OPTIMISATIONS (% of native C)														
Total	663.9	693.2	475.8	250.4	226.4	124.3	122.8	492.5	272.6	376.0	790.8	210.2	205.6	377.3
push/pop	266.9	200.8	202.2	166.4	105.3	61.9	57.2	205.5	105.6	123.8	137.7	80.9	77.5	137.8
load/store	240.3	177.5	191.0	42.5	43.9	28.5	25.2	190.4	111.7	89.2	165.3	67.6	47.6	109.3
mov(w)	23.3	14.8	4.5	3.9	2.6	-1.2	4.2	8.0	5.1	5.3	17.6	-3.0	10.9	7.4
other	133.5	118.4	78.1	37.7	74.6	35.1	36.2	88.8	49.0	97.7	94.8	37.4	63.4	72.7
vm	0.0	181.7	0.0	0.0	0.0	0.0	0.0	-0.1	1.1	60.0	375.4	27.3	6.2	50.1
OVERHEAD REDUCTION PER COMPILER OPTIMISATION (% of native C)														
Impr. peephole	-233.5	-157.7	-149.4	-60.3	-48.2	-23.1	-36.5	-186.9	-54.2	-58.8	-60.2	-35.2	-54.5	-89.1
Stack caching	-40.0	-56.0	-57.3	-98.4	-58.0	-39.8	-16.2	-27.8	-67.7	-40.7	-63.1	-41.4	-24.2	-48.6
Pop. val. caching	-133.1	-84.9	-67.4	-6.8	-12.9	-8.8	-10.7	-51.0	-28.8	-24.5	-41.5	-15.4	-15.5	-38.5
Mark loops	-102.9	-46.8	-85.4	+5.0	-10.9	-8.0	-7.9	-114.9	-18.0	-40.0	-54.3	-38.2	-28.6	-42.4
Const shift	0.0	-17.1	-35.4	-18.4	-45.2	-20.9	-3.8	0.0	-9.6	-10.1	0.0	-17.2	-3.3	-13.9
16-bit array index	-53.2	-34.9	-15.7	-13.9	-5.5	-4.2	-2.8	-36.2	-9.7	-38.9	-19.7	-1.7	-9.0	-18.9
SIMUL	0.0	0.0	0.0	0.0	0.0	0.0	-27.2	0.0	0.0	-36.6	0.0	0.0	0.0	-4.9
Lightw. methods	0.0	-207.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-67.5	-395.7	-30.6	-0.3	-54.0
PERFORMANCE OVERHEAD AFTER COMPILER OPTIMISATIONS (% of native C)														
Total	101.2	88.5	65.2	57.6	45.7	19.5	17.7	75.7	84.6	58.9	156.3	30.5	70.2	67.0
push/pop	0.0	-2.8	0.0	37.4	0.1	2.9	2.0	-0.2	-13.7	2.5	20.4	5.6	1.7	4.3
load/store	1.0	29.3	27.0	-2.3	20.3	4.3	2.4	4.5	54.3	17.1	72.0	2.7	13.5	18.9
mov(w)	10.0	9.4	11.8	5.6	1.5	0.1	2.9	6.8	7.4	9.6	14.9	5.1	4.4	6.9
other	90.2	52.5	26.4	16.9	23.8	12.2	10.4	64.7	35.5	28.8	35.7	17.0	46.1	35.4
vm	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-0.1	1.1	0.8	13.2	0.0	4.4	1.5
	B.sort	H.sort	Bin.Search	XXTEA	MD5	RC5	FFT	Outlier	LEC	CoreMark	MoteTrack	HeatCalib	HeatDetect	average

Table 7.5: Code size data per benchmark

	B.sort	H.sort	Bin.Search	XXTEA	MD5	RC5	FFT	Outlier	LEC	CoreMark	MoteTrack	HeatCalib	HeatDetect	average
CODE SIZE OVERHEAD USING ORIGINAL SOURCE (% of native C)														
Total	449.2	298.0	208.2	287.2	166.0	239.3	94.9	316.3	186.4	159.4	255.0	26.2	238.5	225.0
push/pop	159.3	99.3	71.2	140.6	110.7	108.6	47.7	92.6	60.7	69.6	78.1	31.7	93.9	89.5
load/store	128.8	65.8	76.7	68.9	40.8	56.5	20.3	103.2	71.4	51.6	75.9	22.6	56.4	64.5
mov(w)	1.7	17.4	9.6	10.1	-3.6	0.0	2.5	14.7	5.7	-3.1	24.1	-14.3	15.1	6.1
other	159.3	115.4	50.7	67.6	18.0	74.3	24.5	105.8	48.6	41.2	76.9	-13.8	73.1	64.7
OVERHEAD REDUCTION FROM SOURCE CODE OPTIMISATION (% of native C)														
Source optimisation	-195.0	-58.4	-26.0	-124.2	+45.9	+110.2	+4.5	-47.4	+4.3	-31.2	-27.7	0.0	-12.7	-27.5
CODE SIZE OVERHEAD BEFORE COMPILER OPTIMISATIONS (% of native C)														
Total	254.2	239.6	182.2	163.0	211.9	349.5	99.4	268.9	190.7	128.2	227.3	26.2	225.8	197.5
push/pop	71.2	80.5	60.3	103.7	133.3	165.3	52.6	86.3	63.6	55.2	72.8	31.7	83.3	81.5
load/store	88.1	73.8	74.0	28.4	56.7	67.9	19.7	101.1	72.9	45.8	68.2	22.6	60.1	59.9
mov(w)	10.2	9.4	4.1	2.6	-1.0	2.2	4.3	4.7	5.7	-3.4	19.6	-14.3	16.2	4.6
other	84.7	75.8	43.8	28.2	22.9	114.1	22.8	76.8	48.6	30.5	66.7	-13.8	66.2	51.3
OVERHEAD REDUCTION PER COMPILER OPTIMISATION (% of native C)														
Impr. peephole	-67.8	-53.0	-45.2	-38.3	-49.4	-62.5	-32.2	-77.8	-33.9	-24.7	-27.4	-13.6	-49.8	-44.3
Stack caching	-25.4	-26.2	-24.7	-59.4	-85.4	-111.2	-20.9	-30.6	-39.7	-27.6	-26.7	-12.6	-38.3	-40.7
Pop. val. caching	-16.9	-29.5	-6.8	-6.2	-18.7	-18.7	-13.5	-5.2	-18.5	-9.9	-26.7	-8.1	-20.7	-15.3
Mark loops	+1.7	0.0	+21.9	+5.9	-1.2	-2.6	-4.2	-16.4	+2.5	-1.5	-8.7	-1.3	-11.4	-1.2
Const shift	0.0	-6.1	-6.9	+1.7	+2.8	-16.0	-4.6	-2.6	-1.8	-1.5	0.0	-1.7	-0.1	-2.8
16-bit array index	-27.2	-22.8	-8.2	-11.6	-5.1	-16.7	-11.6	-25.8	-10.7	-7.4	-16.9	-2.2	-10.7	-13.6
SIMUL	0.0	0.0	0.0	0.0	0.0	0.0	-9.9	0.0	0.0	-3.4	0.0	0.0	0.0	-1.1
Lightw. methods	0.0	-2.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-5.5	-3.8	-3.9	+0.6	-1.1
CODE SIZE OVERHEAD AFTER COMPILER OPTIMISATIONS (% of native C)														
Total	118.6	100.0	112.3	55.1	54.9	121.8	2.5	110.5	88.6	46.7	117.1	-17.2	95.4	77.4
push/pop	23.7	16.1	27.4	13.3	0.0	6.2	1.9	-2.1	-5.0	1.7	16.3	3.9	-3.1	7.7
load/store	33.9	41.6	49.3	14.8	37.2	25.3	-2.6	57.9	45.0	30.1	40.9	8.0	37.6	32.2
mov(w)	1.7	6.7	6.8	2.5	-2.4	11.9	-0.8	1.1	7.1	-0.2	15.4	-10.7	13.3	4.0
other	59.3	35.6	28.8	24.4	20.1	78.5	4.0	53.7	41.4	15.1	44.5	-18.4	47.6	33.4
	B.sort	H.sort	Bin.Search	XXTEA	MD5	RC5	FFT	Outlier	LEC	CoreMark	MoteTrack	HeatCalib	HeatDetect	average

Table 7.9: Cost of safety guarantees

	B.sort	H.sort	Bin.Search	XXTEA	MD5	RC5	FFT	Outlier	LEC	CoreMark	MoteTrack	HeatCalib	HeatDetect	average
EXECUTED BYTECODE INSTRUCTIONS (% of total executed bytecode instructions after optimisation)														
Array element/object field STORES	18.0	7.8	0.0	2.9	4.5	1.5	6.1	5.8	3.7	2.6	10.0	1.4	4.7	5.3
Array element/object field LOADS	18.0	15.9	7.1	8.6	6.2	6.4	7.0	10.7	7.9	11.7	21.4	4.1	8.8	10.3
PERFORMANCE OVERHEAD VS NATIVE C (% of native C)														
unsafe	101.2	88.5	65.2	57.6	45.7	19.5	17.7	75.7	84.6	58.9	156.3	30.5	70.2	67.0
safe writes	247.5	153.9	65.2	68.2	60.3	22.2	30.3	128.4	118.4	76.7	266.1	33.9	88.2	104.6
safe reads and writes	393.9	287.8	151.7	100.0	80.3	33.4	43.0	226.6	179.8	155.0	445.1	43.9	120.8	173.9
PERFORMANCE OVERHEAD VS UNSAFE VM (% of unsafe AOT)														
safe writes	72.7	34.7	0.0	6.7	10.0	2.3	10.7	30.0	18.3	11.2	42.8	2.6	10.6	22.5
safe reads and writes	145.5	105.7	52.4	26.9	23.7	11.6	21.5	85.9	51.6	60.5	112.7	10.3	29.7	64.0
CODE SIZE OVERHEAD VS NATIVE C (% of native C)														
unsafe	118.6	100.0	112.3	55.1	54.9	121.8	2.5	110.5	88.6	46.7	117.1	-17.2	95.4	77.4
safe writes	125.4	105.4	112.3	56.2	55.7	125.3	5.0	118.9	94.3	50.5	125.4	-16.4	102.6	81.6
safe reads and writes	132.2	113.4	117.8	60.1	59.1	132.3	8.0	123.2	102.9	58.2	145.3	-13.9	106.2	88.1
CODE SIZE OVERHEAD VS UNSAFE VM (% of unsafe AOT)														
safe writes	3.1	2.7	0.0	0.7	0.5	1.6	2.4	4.0	3.0	2.6	3.8	1.0	3.7	2.4
safe reads and writes	6.2	6.7	2.6	3.2	2.7	4.7	5.4	6.0	7.6	7.8	13.0	4.0	5.5	6.0

Table 8.2: Quantitative impact of Java/JVM issues

Section	Measure [a]	B.sort	H.sort	Bin.Search	XXTEA	MD5	RC5	FFT	Outlier	LEC	CoreMark	MoteTrack	HeatCalib	HeatDetect
8.2	Size of constant data						200	2,048		51		20,560		
	Const array RAM overhead						208	2,056		67		too big		
	Const array flash overhead						1,998	26,714		930		too big		
8.3	Size of main data structures in C	512	512	200	144	174	256	256	860	1024	1633 [b]	606	644	1088
	Size of main data structures in Java	520	520	208	160	214	288	272	884	1058	1996	1387 [c]	676	1158
	Size increase	1.6%	1.6%	4.0%	11.1%	23.0%	12.5%	6.3%	2.8%	3.3%	22.2%	128.9%	5.0%	6.4%
8.4	Casts	1	6	5	8	8	8	16	3	10	70	33	4	64
	Lines of code [d]	11	24	16	38	165	27	73	44	77	849	475	51	266
	Casts per 100 LOC	9	25	31	21	5	30	22	7	13	8	7	8	24
8.6	Slowdown non-inlined version		69%		57%	25%	37%	20%			13%			
	Size difference non-inlined version		+42		-224	-1502	-94	-20			+48			
8.7	Slowdown w/o optimisations	91%	52%	544%	3%			3%	23%		117%	76%		2%
8.8	Slowdown from heap allocations									330%	6%	65%		

[a] A blank entry indicates the benchmark was not affected. Highlights indicate a significant impact.

[b] Actual amount of memory used. CoreMark's C version allocates 2047 bytes, but the remaining space is not used.

[c] After replacing Motetrack's 2-byte RSSI array with two variables.

[d] Counted as the number of actual code lines, excluding blanks lines, comments, and single brackets.