

```
1
2 -----
3 Interpreter
4
5 code size      =    126K bytes
6 total space    =    895K bytes
7 wasted space   =    769K bytes
8
9 # of codelets  =    266
10 avg codelet size =   487 bytes
11
12 -----
13
14 slow signature handler [0xa7558ae0, 0xa7558ba0] 192 bytes
15
16 Loaded disassembler from /home/zbhuang/OpenJDK/jdk8u/build/linux-x86-normal-server-slowdebug/jdk/lib/i386/hsdis-i386.so
17
18 -----
19 method entry point (kind = zerolocals) [0xa7564380, 0xa75648c0] 1344 bytes
20
21 0xa7564380: mov     0x8(%ebx),%edx
22 0xa7564383: movzwl 0x24(%edx),%ecx
23 0xa7564387: movzwl 0x22(%edx),%edx
24 0xa756438b: sub     %ecx,%edx
25 0xa756438d: cmp     $0x3f6,%edx
26 0xa7564393: jbe     0xa7564414
27 0xa7564399: push    %esi
28 0xa756439a: mov     %esp,%esi
29 0xa756439c: shr     $0xc,%esi
30 0xa756439f: mov     -0x4871aba0(,%esi,4),%esi
31 0xa75643a6: lea     0x28(%edx,4),%eax
32 0xa75643ad: cmpl    $0x0,0xd8(%esi)
33 0xa75643b7: jne     0xa75643ce
34 0xa75643bd: push    $0xb756221c
35 0xa75643c2: call    0xa75643c7
36 0xa75643c7: pusha
37 0xa75643c8: call    0xb6ff5f32
38 0xa75643cd: hlt
39 0xa75643ce: cmpl    $0x0,0xdc(%esi)
40 0xa75643d8: jne     0xa75643ef
41 0xa75643de: push    $0xb756222f
42 0xa75643e3: call    0xa75643e8
43 0xa75643e8: pusha
44 0xa75643e9: call    0xb6ff5f32
45 0xa75643ee: hlt
46 0xa75643ef: add     0xd8(%esi),%eax
47 0xa75643f5: sub     0xdc(%esi),%eax
48 0xa75643fb: add     $0x9000,%eax
49 0xa7564401: cmp     %eax,%esp
50 0xa7564403: ja      0xa7564413
51 0xa7564409: pop     %esi
52 0xa756440a: pop     %eax
53 0xa756440b: mov     %esi,%esp
54 0xa756440d: push    %eax
55 0xa756440e: jmp     0xa75589e0
56 0xa7564413: pop     %esi
57 0xa7564414: pop     %eax
58 0xa7564415: lea     -0x4(%esp,%ecx,4),%edi
59 0xa7564419: test    %edx,%edx
60 0xa756441b: jle     0xa7564429
61 0xa7564421: push    $0x0
62 0xa7564426: dec     %edx
63 0xa7564427: jg      0xa7564421
64 0xa7564429: push    %eax
65 0xa756442a: push    %ebp
66 0xa756442b: mov     %esp,%ebp
67 0xa756442d: push    %esi
68 0xa756442e: push    $0x0
69 0xa7564433: mov     0x8(%ebx),%esi
70 0xa7564436: lea     0x28(%esi),%esi
71 0xa7564439: push    %ebx
72 0xa756443a: mov     0xc(%ebx),%edx
73 0xa756443d: test    %edx,%edx
74 0xa756443f: je      0xa756444b
75 0xa7564445: add     $0xe0,%edx
76 0xa756444b: push    %edx
77 0xa756444c: mov     0x8(%ebx),%edx
78 0xa756444f: mov     0x8(%edx),%edx
79 0xa7564452: mov     0xc(%edx),%edx
80 0xa7564455: push    %edx
81 0xa7564456: push    %edi
82 0xa7564457: push    %esi
83 0xa7564458: push    $0x0
84 0xa756445d: mov     %esp,(%esp)
85 0xa7564460: mov     0x14(%ebx),%eax
86 0xa7564463: test    $0x100,%eax
```

```
87 0xa7564468: je 0xa756447f
88 0xa756446e: push $0xb7562538
89 0xa7564473: call 0xa7564478
90 0xa7564478: pusha
91 0xa7564479: call 0xb6ff5f32
92 0xa756447e: hlt
93 0xa756447f: test $0x400,%eax
94 0xa7564484: je 0xa756449b
95 0xa756448a: push $0xb75623a4
96 0xa756448f: call 0xa7564494
97 0xa7564494: pusha
98 0xa7564495: call 0xb6ff5f32
99 0xa756449a: hlt
100 0xa756449b: mov %esp,%eax
101 0xa756449d: shr $0xc,%eax
102 0xa75644a0: mov -0x4871aba0(,%eax,4),%eax
103 0xa75644a7: movb $0x1,0x1a5(%eax)
104 0xa75644ae: mov -0x10(%ebp),%eax
105 0xa75644b1: test %eax,%eax
106 0xa75644b3: je 0xa7564527
107 0xa75644b9: mov -0x4(%eax),%ecx
108 0xa75644bc: test %ecx,%ecx
109 0xa75644be: js 0xa7564527
110 0xa75644c4: add %ecx,%eax
111 0xa75644c6: mov 0x4(%eax),%ecx
112 0xa75644c9: sub $0x2,%ecx
113 0xa75644cc: mov 0x8(%eax,%ecx,4),%edx
114 0xa75644d0: neg %edx
115 0xa75644d2: mov (%edi,%edx,4),%edx
116 0xa75644d5: test %edx,%edx
117 0xa75644d7: jne 0xa75644e0
118 0xa75644d9: orl $0x1,0xc(%eax,%ecx,4)
119 0xa75644de: jmp 0xa7564522
120 0xa75644e0: mov 0x4(%edx),%edx
121 0xa75644e3: xor 0xc(%eax,%ecx,4),%edx
122 0xa75644e7: test $0xffffffffc,%edx
123 0xa75644ed: je 0xa7564522
124 0xa75644ef: test $0x2,%edx
125 0xa75644f5: jne 0xa7564522
126 0xa75644f7: cmpl $0x0,0xc(%eax,%ecx,4)
127 0xa75644ff: je 0xa756451e
128 0xa7564501: cmpl $0x1,0xc(%eax,%ecx,4)
129 0xa7564509: je 0xa756451e
130 0xa756450b: xor 0xc(%eax,%ecx,4),%edx
131 0xa756450f: test $0xffffffffc,%edx
132 0xa7564515: je 0xa7564522
133 0xa7564517: orl $0x2,0xc(%eax,%ecx,4)
134 0xa756451c: jmp 0xa7564522
135 0xa756451e: mov %edx,0xc(%eax,%ecx,4)
136 0xa7564522: sub $0x2,%ecx
137 0xa7564525: jns 0xa75644cc
138 0xa7564527: mov 0xc(%ebx),%eax
139 0xa756452a: test %eax,%eax
140 0xa756452c: je 0xa756454e
141 0xa756452e: mov 0xbc(%eax),%ecx
142 0xa7564534: add $0x8,%ecx
143 0xa7564537: mov %ecx,0xbc(%eax)
144 0xa756453d: and $0x3f8,%ecx
145 0xa7564543: je 0xa75647f7
146 0xa7564549: jmp 0xa7564623
147 0xa756454e: mov 0x10(%ebx),%eax
148 0xa7564551: test %eax,%eax
149 0xa7564553: jne 0xa756460e
150 0xa7564559: call 0xa7564563
151 0xa756455e: jmp 0xa7564603
152 0xa7564563: push %ebx
153 0xa7564564: lea 0x8(%esp),%eax
154 0xa7564568: cmpl $0x0,-0x8(%ebp)
155 0xa756456f: je 0xa7564586
156 0xa7564575: push $0xb7449b58
157 0xa756457a: call 0xa756457f
158 0xa756457f: pusha
159 0xa7564580: call 0xb6ff5f32
160 0xa7564585: hlt
161 0xa7564586: mov %esi,-0x1c(%ebp)
162 0xa7564589: mov %esp,%edi
163 0xa756458b: shr $0xc,%edi
164 0xa756458e: mov -0x4871aba0(,%edi,4),%edi
165 0xa7564595: push %edi
166 0xa7564596: mov %ebp,0x148(%edi)
167 0xa756459c: mov %eax,0x140(%edi)
168 0xa75645a2: call 0xb6e61c92
169 0xa75645a7: add $0x8,%esp
170 0xa75645aa: push %eax
171 0xa75645ab: mov %esp,%eax
172 0xa75645ad: shr $0xc,%eax
```

```
173 0xa75645b0: mov    -0x4871aba0(,%eax,4),%eax
174 0xa75645b7: cmp    %eax,%edi
175 0xa75645b9: je     0xa75645d0
176 ;; MacroAssembler::call_VM_base: rdi not callee saved?
177 0xa75645bf: push  $0xb74acf8c
178 0xa75645c4: call  0xa75645c9
179 0xa75645c9: pusha
180 0xa75645ca: call  0xb6ff5f32
181 0xa75645cf: hlt
182 0xa75645d0: pop    %eax
183 0xa75645d1: movl   $0x0,0x140(%edi)
184 0xa75645db: movl   $0x0,0x148(%edi)
185 0xa75645e5: movl   $0x0,0x144(%edi)
186 0xa75645ef: cmpl   $0x0,0x4(%edi)
187 0xa75645f6: jne    0xa756400
188 0xa75645fc: mov    -0x1c(%ebp),%esi
189 0xa75645ff: mov    -0x18(%ebp),%edi
190 0xa7564602: ret
191 0xa7564603: mov    0x10(%ebx),%eax
192 0xa7564606: test   %eax,%eax
193 0xa7564608: je     0xa7564623
194 0xa756460e: mov    0x8(%eax),%ecx
195 0xa7564611: add    $0x8,%ecx
196 0xa7564614: mov    %ecx,0x8(%eax)
197 0xa7564617: and    $0x3f8,%ecx
198 0xa756461d: je     0xa75647f7
199 0xa7564623: mov    %eax,-0x1000(%esp)
200 0xa756462a: mov    %eax,-0x2000(%esp)
201 0xa7564631: mov    %eax,-0x3000(%esp)
202 0xa7564638: mov    %eax,-0x4000(%esp)
203 0xa756463f: mov    %eax,-0x5000(%esp)
204 0xa7564646: mov    %eax,-0x6000(%esp)
205 0xa756464d: mov    %eax,-0x7000(%esp)
206 0xa7564654: mov    %eax,-0x8000(%esp)
207 0xa756465b: mov    %eax,-0x9000(%esp)
208 0xa7564662: mov    %esp,%eax
209 0xa7564664: shr    $0xc,%eax
210 0xa7564667: mov    -0x4871aba0(,%eax,4),%eax
211 0xa756466e: movb   $0x0,0x1a5(%eax)
212 0xa7564675: mov    0x14(%ebx),%eax
213 0xa7564678: test   $0x20,%eax
214 0xa756467d: je     0xa7564694
215 0xa7564683: push   $0xb75623d4
216 0xa7564688: call   0xa756468d
217 0xa756468d: pusha
218 0xa756468e: call   0xb6ff5f32
219 0xa7564693: hlt
220 0xa7564694: mov    -0x20(%ebp),%eax
221 0xa7564697: cmp    %esp,%eax
222 0xa7564699: je     0xa75646b0
223 0xa756469f: push   $0xb75623f4
224 0xa75646a4: call   0xa75646a9
225 0xa75646a9: pusha
226 0xa75646aa: call   0xb6ff5f32
227 0xa75646af: hlt
228 0xa75646b0: cmpb   $0x0,0xb78c1e7e
229 0xa75646b7: je     0xa75646f4
230 0xa75646bd: mov    %esp,%ecx
231 0xa75646bf: shr    $0xc,%ecx
232 0xa75646c2: mov    -0x4871aba0(,%ecx,4),%ecx
233 0xa75646c9: mov    -0xc(%ebp),%ebx
234 0xa75646cc: push   %ebx
235 0xa75646cd: push   %ecx
236 0xa75646ce: cmpl   $0x0,-0x8(%ebp)
237 0xa75646d5: je     0xa75646ec
238 0xa75646db: push   $0xb7449b18
239 0xa75646e0: call   0xa75646e5
240 0xa75646e5: pusha
241 0xa75646e6: call   0xb6ff5f32
242 0xa75646eb: hlt
243 0xa75646ec: call   0xb71757d0
244 0xa75646f1: add    $0x8,%esp
245 0xa75646f4: movzbl (%esi),%ebx
246 0xa75646f7: jmp    *-0x48722ae0(,%ebx,4)
247 0xa75646fe: call   0xa7564708
248 0xa7564703: jmp    0xa75647a7
249 0xa7564708: lea    0x4(%esp),%eax
250 0xa756470c: cmpl   $0x0,-0x8(%ebp)
251 0xa7564713: je     0xa756472a
252 0xa7564719: push   $0xb7449b58
253 0xa756471e: call   0xa7564723
254 0xa7564723: pusha
255 0xa7564724: call   0xb6ff5f32
256 0xa7564729: hlt
257 0xa756472a: mov    %esi,-0x1c(%ebp)
258 0xa756472d: mov    %esp,%edi
```

```

259 0xa756472f: shr    $0xc,%edi
260 0xa7564732: mov    -0x4871aba0(,%edi,4),%edi
261 0xa7564739: push   %edi
262 0xa756473a: mov    %ebp,0x148(%edi)
263 0xa7564740: mov    %eax,0x140(%edi)
264 0xa7564746: call   0xb6e61328
265 0xa756474b: add    $0x4,%esp
266 0xa756474e: push   %eax
267 0xa756474f: mov    %esp,%eax
268 0xa7564751: shr    $0xc,%eax
269 0xa7564754: mov    -0x4871aba0(,%eax,4),%eax
270 0xa756475b: cmp    %eax,%edi
271 0xa756475d: je     0xa7564774
272 ;; MacroAssembler::call_VM_base: rdi not callee saved?
273 0xa7564763: push   $0xb74acf8c
274 0xa7564768: call   0xa756476d
275 0xa756476d: pusha
276 0xa756476e: call   0xb6ff5f32
277 0xa7564773: hlt
278 0xa7564774: pop    %eax
279 0xa7564775: movl   $0x0,0x140(%edi)
280 0xa756477f: movl   $0x0,0x148(%edi)
281 0xa7564789: movl   $0x0,0x144(%edi)
282 0xa7564793: cmpl   $0x0,0x4(%edi)
283 0xa756479a: jne    0xa756400
284 0xa75647a0: mov    -0x1c(%ebp),%esi
285 0xa75647a3: mov    -0x18(%ebp),%edi
286 0xa75647a6: ret
287 0xa75647a7: push   %eax
288 0xa75647a8: push   %ebx
289 0xa75647a9: mov    -0xc(%ebp),%ebx
290 0xa75647ac: mov    0xc(%ebx),%eax
291 0xa75647af: test   %eax,%eax
292 0xa75647b1: je     0xa75647ea
293 0xa75647b7: push   %esi
294 0xa75647b8: push   %ebx
295 0xa75647b9: cmpl   $0x0,-0x8(%ebp)
296 0xa75647c0: je     0xa75647d7
297 0xa75647c6: push   $0xb7449b18
298 0xa75647cb: call   0xa75647d0
299 0xa75647d0: pusha
300 0xa75647d1: call   0xb6ff5f32
301 0xa75647d6: hlt
302 0xa75647d7: call   0xb6e61172
303 0xa75647dc: add    $0x8,%esp
304 0xa75647df: mov    0xc(%ebx),%ebx
305 0xa75647e2: add    $0xe0,%ebx
306 0xa75647e8: add    %ebx,%eax
307 0xa75647ea: mov    %eax,-0x10(%ebp)
308 0xa75647ed: pop    %ebx
309 0xa75647ee: pop    %eax
310 0xa75647ef: mov    -0xc(%ebp),%ebx
311 0xa75647f2: jmp    0xa7564623
312 0xa75647f7: mov    $0x0,%eax
313 0xa75647fc: call   0xa7564806
314 0xa7564801: jmp    0xa75648a6
315 0xa7564806: push   %eax
316 0xa7564807: lea    0x8(%esp),%eax
317 0xa756480b: cmpl   $0x0,-0x8(%ebp)
318 0xa7564812: je     0xa7564829
319 0xa7564818: push   $0xb7449b58
320 0xa756481d: call   0xa7564822
321 0xa7564822: pusha
322 0xa7564823: call   0xb6ff5f32
323 0xa7564828: hlt
324 0xa7564829: mov    %esi,-0x1c(%ebp)
325 0xa756482c: mov    %esp,%edi
326 0xa756482e: shr    $0xc,%edi
327 0xa7564831: mov    -0x4871aba0(,%edi,4),%edi
328 0xa7564838: push   %edi
329 0xa7564839: mov    %ebp,0x148(%edi)
330 0xa756483f: mov    %eax,0x140(%edi)
331 0xa7564845: call   0xb6e60bc8
332 0xa756484a: add    $0x8,%esp
333 0xa756484d: push   %eax
334 0xa756484e: mov    %esp,%eax
335 0xa7564850: shr    $0xc,%eax
336 0xa7564853: mov    -0x4871aba0(,%eax,4),%eax
337 0xa756485a: cmp    %eax,%edi
338 0xa756485c: je     0xa7564873
339 ;; MacroAssembler::call_VM_base: rdi not callee saved?
340 0xa7564862: push   $0xb74acf8c
341 0xa7564867: call   0xa756486c
342 0xa756486c: pusha
343 0xa756486d: call   0xb6ff5f32
344 0xa7564872: hlt

```

```
345 0xa7564873: pop    %eax
346 0xa7564874: movl   $0x0,0x140(%edi)
347 0xa756487e: movl   $0x0,0x148(%edi)
348 0xa7564888: movl   $0x0,0x144(%edi)
349 0xa7564892: cmpl   $0x0,0x4(%edi)
350 0xa7564899: jne     0xa756400
351 0xa756489f: mov     -0x1c(%ebp),%esi
352 0xa75648a2: mov     -0x18(%ebp),%edi
353 0xa75648a5: ret
354 0xa75648a6: mov     -0xc(%ebp),%ebx
355 0xa75648a9: jmp     0xa7564623
356 0xa75648ae: nop
357 0xa75648af: nop
358 0xa75648b0: int3
359 0xa75648b1: int3
360 0xa75648b2: int3
361 0xa75648b3: int3
362 0xa75648b4: int3
363 0xa75648b5: int3
364 0xa75648b6: int3
365 0xa75648b7: int3
366 0xa75648b8: int3
367 0xa75648b9: int3
368 0xa75648ba: int3
369 0xa75648bb: int3
370 0xa75648bc: int3
371 0xa75648bd: int3
372 0xa75648be: int3
373 0xa75648bf: int3
374
375 -----
376
377 hello world!
378
379 RUN FINISHED; exit value 0; real time: 13s; user: 430ms; system: 970ms
380
```