

<https://liuxiaofei.com.cn/blog/hotspot-jit%E7%BC%96%E8%AF%91C1/>

Hotspot JIT编译(C1)

Content:

1 字节码

2 Block List生成

生成高级中间表示(HIR)

生成LIR

分配物理寄存器

生成代码

上一篇文章介绍了JVM怎么用template interpreter来完成字节码到机器码的映射,但是频繁的取指,译指,然后执行,性能会有所损耗.

所以hotspot对频繁执行的方法进行编译优化能大大缩短代码执行时间.

这儿以编译器C1来说明JIT编译,过程主要包括生成BlockList,生成HIR,发射LIR,寄存器分配,生成机器码,代码安装.

对应的代码位置为jdk8\hotspot\src\share\vm\c1.

1 字节码

以String.indexOf(int,int)方法的编译为例:

```
//LineNo SourceCode
1545 public int indexOf(int ch, int fromIndex) {
1546     final int max = value.length;
1547     if (fromIndex < 0) {
1548         fromIndex = 0;
1549     } else if (fromIndex >= max) {
1550         // Note: fromIndex might be near -1>>>1.
1551         return -1;
1552     }
1553
1554     if (ch < Character.MIN_SUPPLEMENTARY_CODE_POINT) {
1555         // handle most cases here (ch is a BMP code point or a
1556         // negative value (invalid code point))
1557         final char[] value = this.value;
1558         for (int i = fromIndex; i < max; i++) {
1559             if (value[i] == ch) {
1560                 return i;
1561             }
1562         }
1563         return -1;
1564     } else {
1565         return indexOfSupplementary(ch, fromIndex);
1566     }
1567 }
```

//BCI(Byte Code Index): ByteCode

```
0: aload_0
1: getfield      #423 // Field value:[C
4: arraylength
5: istore_3
6: iload_2
7: ifge         15
10: iconst_0
11: istore_2
12: goto         22
15: iload_2
16: iload_3
17: if_icmplt    22
20: iconst_m1
21: ireturn
22: iload_1
23: ldc         #4 // int 65536
25: if_icmpge   63
28: aload_0
29: getfield      #423 // Field value:[C
32: astore      4
34: iload_2
35: istore        5
37: iload         5
39: iload_3
40: if_icmpge    61
43: aload         4
45: iload         5
47: caload
48: iload_1
49: if_icmpne    55
52: iload         5
54: ireturn
55: iinc         5, 1
58: goto         37
61: iconst_m1
62: ireturn
63: aload_0
64: iload_1
65: iload_2
66: invokespecial #465 // Method indexOfSupplementary:(II)I
69: ireturn
2 Block List生成
```

```

87 Call stack to build block list:
88
89 int Compilation::compile_java_method()
90     void Compilation::build_hir()
91         IR::IR(Compilation* compilation, ciMethod* method, int osr_bci)
92             IRScope::IRScope(Compilation* compilation, IRScope* caller, int caller_bci, ciMethod* method, int osr_bci, bool create_graph)
93                 BlockBegin* IRScope::build_graph(Compilation* compilation, int osr_bci)
94                     GraphBuilder::GraphBuilder(Compilation* compilation, IRScope* scope)
95                         BlockListBuilder::BlockListBuilder(Compilation* compilation, IRScope* scope, int osr_bci)
96                             BlockListBuilder::set_leaders()
97
98 //代码文件c1_GraphBuilder.cpp
99 //通过此方法对所有的字节码进行分析,然后划分并生成Block,从下面代码可以看到所有的跳转命令,比较命令,switch语句会生成新的block.
100 void BlockListBuilder::set_leaders() {
101     bool has_xhandlers = xhandlers()->has_handlers();
102     BlockBegin* current = NULL;
103
104     // The information which bci starts a new block simplifies the analysis
105     // Without it, backward branches could jump to a bci where no block was created
106     // during bytecode iteration. This would require the creation of a new block at the
107     // branch target and a modification of the successor lists.
108     BitMap bci_block_start = method()->bci_block_start();
109
110     ciBytecodeStream s(method());
111     while (s.next() != ciBytecodeStream::EOBC()) {
112         int cur_bci = s.cur_bci();
113
114         if (bci_block_start.at(cur_bci)) {
115             current = make_block_at(cur_bci, current);
116         }
117         assert(current != NULL, "must have current block");
118
119         if (has_xhandlers && GraphBuilder::can_trap(method(), s.cur_bc())) {
120             handle_exceptions(current, cur_bci);
121         }
122
123         switch (s.cur_bc()) {
124             // track stores to local variables for selective creation of phi functions
125             //在栈中标记局部变量
126             case Bytecodes::_iinc:      store_one(current, s.get_index()); break;
127             case Bytecodes::_istore:    store_one(current, s.get_index()); break;
128             case Bytecodes::_lstore:    store_two(current, s.get_index()); break;
129             case Bytecodes::_fstore:    store_one(current, s.get_index()); break;
130             case Bytecodes::_dstore:    store_two(current, s.get_index()); break;
131             case Bytecodes::_astore:    store_one(current, s.get_index()); break;
132             case Bytecodes::_istore_0:  store_one(current, 0); break;
133             case Bytecodes::_istore_1:  store_one(current, 1); break;
134             case Bytecodes::_istore_2:  store_one(current, 2); break;
135             case Bytecodes::_istore_3:  store_one(current, 3); break;
136             case Bytecodes::_lstore_0:  store_two(current, 0); break;
137             case Bytecodes::_lstore_1:  store_two(current, 1); break;
138             case Bytecodes::_lstore_2:  store_two(current, 2); break;
139             case Bytecodes::_lstore_3:  store_two(current, 3); break;
140             case Bytecodes::_fstore_0:  store_one(current, 0); break;
141             case Bytecodes::_fstore_1:  store_one(current, 1); break;
142             case Bytecodes::_fstore_2:  store_one(current, 2); break;
143             case Bytecodes::_fstore_3:  store_one(current, 3); break;
144             case Bytecodes::_dstore_0:  store_two(current, 0); break;
145             case Bytecodes::_dstore_1:  store_two(current, 1); break;
146             case Bytecodes::_dstore_2:  store_two(current, 2); break;
147             case Bytecodes::_dstore_3:  store_two(current, 3); break;
148             case Bytecodes::_astore_0:  store_one(current, 0); break;
149             case Bytecodes::_astore_1:  store_one(current, 1); break;
150             case Bytecodes::_astore_2:  store_one(current, 2); break;
151             case Bytecodes::_astore_3:  store_one(current, 3); break;
152
153             // track bytecodes that affect the control flow
154             case Bytecodes::_athrow:    // fall through
155             case Bytecodes::_ret:       // fall through
156             case Bytecodes::_ireturn:   // fall through
157             case Bytecodes::_lreturn:   // fall through
158             case Bytecodes::_freturn:   // fall through
159             case Bytecodes::_dreturn:   // fall through
160             case Bytecodes::_areturn:   // fall through
161             case Bytecodes::_return:
162                 current = NULL;
163                 break;
164
165             case Bytecodes::_ifeq:      // fall through
166             case Bytecodes::_ifne:      // fall through
167             case Bytecodes::_iflt:      // fall through
168             case Bytecodes::_ifge:      // fall through
169             case Bytecodes::_ifgt:      // fall through
170             case Bytecodes::_ifle:      // fall through
171             case Bytecodes::_if_icmpeq:  // fall through
172             case Bytecodes::_if_icmpne:  // fall through

```

```

173     case Bytecodes::_if_icmplt: // fall through
174     case Bytecodes::_if_icmpge: // fall through
175     case Bytecodes::_if_icmpgt: // fall through
176     case Bytecodes::_if_icmple: // fall through
177     case Bytecodes::_if_acmpeq: // fall through
178     case Bytecodes::_if_acmpne: // fall through
179     case Bytecodes::_ifnull:    // fall through
180     case Bytecodes::_ifnonnull:
181         make_block_at(s.next_bci(), current); //条件成功时,下一个字节码开始下一个block
182         make_block_at(s.get_dest(), current); //条件不成功时,跳转到的字节码地方开始下一个block
183         current = NULL;
184         break;
185
186     case Bytecodes::_goto:
187         make_block_at(s.get_dest(), current);
188         current = NULL;
189         break;
190
191     case Bytecodes::_goto_w:
192         make_block_at(s.get_far_dest(), current);
193         current = NULL;
194         break;
195
196     case Bytecodes::_jsr:
197         handle_jsr(current, s.get_dest(), s.next_bci());
198         current = NULL;
199         break;
200
201     case Bytecodes::_jsr_w:
202         handle_jsr(current, s.get_far_dest(), s.next_bci());
203         current = NULL;
204         break;
205
206     case Bytecodes::_tableswitch: {
207         // set block for each case
208         Bytecode_tableswitch sw(&s);
209         int l = sw.length();
210         for (int i = 0; i < l; i++) {
211             make_block_at(cur_bci + sw.dest_offset_at(i), current);
212         }
213         make_block_at(cur_bci + sw.default_offset(), current);
214         current = NULL;
215         break;
216     }
217
218     case Bytecodes::_lookupswitch: {
219         // set block for each case
220         Bytecode_lookupswitch sw(&s);
221         int l = sw.number_of_pairs();
222         for (int i = 0; i < l; i++) {
223             make_block_at(cur_bci + sw.pair_at(i).offset(), current);
224         }
225         make_block_at(cur_bci + sw.default_offset(), current);
226         current = NULL;
227         break;
228     }
229 }
230 }
231 }

```

最后,Block List生成如下:

```

234 begin_compilation
235     name " java.lang.String::indexOf"
236     method "virtual jint java.lang.String.indexOf(jint, jint)"
237     date 1527264031403
238 end_compilation
239 begin_cfg
240     name "BlockListBuilder virtual jint java.lang.String.indexOf(jint, jint)"
241     begin_block
242         name "B0" //Block名称
243         from_bci 0 //Block开始的字节码索引
244         to_bci -1 //Block的结束字节码索引
245         predecessors //前置Block
246         successors "B1" "B2" //此Block会跳转到的Block名称
247         xhandlers
248         flags "std"
249     end_block
250     begin_block
251         name "B1"
252         from_bci 10
253         to_bci -1
254         predecessors
255         successors "B3"
256         xhandlers
257         flags
258     end_block

```

```
259     begin_block
260         name "B2"
261         from_bci 15
262         to_bci -1
263         predecessors
264         successors "B4" "B3"
265         xhandlers
266         flags
267     end_block
268     begin_block
269         name "B4"
270         from_bci 20
271         to_bci -1
272         predecessors
273         successors
274         xhandlers
275         flags
276     end_block
277     begin_block
278         name "B3"
279         from_bci 22
280         to_bci -1
281         predecessors
282         successors "B5" "B6"
283         xhandlers
284         flags
285     end_block
286     begin_block
287         name "B5"
288         from_bci 28
289         to_bci -1
290         predecessors
291         successors "B7"
292         xhandlers
293         flags
294     end_block
295     begin_block
296         name "B7"
297         from_bci 37
298         to_bci -1
299         predecessors
300         successors "B8" "B9"
301         xhandlers
302         flags "plh"
303     end_block
304     begin_block
305         name "B8"
306         from_bci 43
307         to_bci -1
308         predecessors
309         successors "B10" "B11"
310         xhandlers
311         flags
312     end_block
313     begin_block
314         name "B10"
315         from_bci 52
316         to_bci -1
317         predecessors
318         successors
319         xhandlers
320         flags
321     end_block
322     begin_block
323         name "B11"
324         from_bci 55
325         to_bci -1
326         predecessors
327         successors "B7"
328         xhandlers
329         flags
330     end_block
331     begin_block
332         name "B9"
333         from_bci 61
334         to_bci -1
335         predecessors
336         successors
337         xhandlers
338         flags
339     end_block
340     begin_block
341         name "B6"
342         from_bci 63
343         to_bci -1
344         predecessors
```

```

345     successors
346     xhandlers
347     flags
348 end_block
349 end_cfg
350 生成高级中间表示(HIR)
351 Block List生成完后,接下来就需要为每个Block生成HIR了.
352 Call stack to create HIR:
353
354 int Compilation::compile_java_method()
355 void Compilation::build_hir()
356     IR::IR(Compilation* compilation, ciMethod* method, int osr_bci)
357     IRScope::IRScope(Compilation* compilation, IRScope* caller, int caller_bci, ciMethod* method, int osr_bci, bool create_graph)
358     BlockBegin* IRScope::build_graph(Compilation* compilation, int osr_bci)
359     GraphBuilder::GraphBuilder(Compilation* compilation, IRScope* scope)
360     void GraphBuilder::iterate_all_blocks(bool start_in_current_block_for_inlining)
361     void GraphBuilder::connect_to_end(BlockBegin* beg)
362     BlockEnd* GraphBuilder::iterate_bytecodes_for_block(int bci)
363
364 //代码文件c1_GraphBuilder.cpp
365 //循环每个字节码,生成对应的Instruction,所以的Instruction声明都在c1_Instruction.hpp中.
366 BlockEnd* GraphBuilder::iterate_bytecodes_for_block(int bci) {
367     .....
368     _skip_block = false;
369     assert(state() != NULL, "ValueStack missing!");
370     CompileLog* log = compilation()->log();
371     ciBytecodeStream s(method());
372     s.reset_to_bci(bci);
373     int prev_bci = bci;
374     scope_data()->set_stream(&s);
375     // iterate
376     Bytecodes::Code code = Bytecodes::_illegal;
377     bool push_exception = false;
378
379     if (block()->is_set(BlockBegin::exception_entry_flag) && block()->next() == NULL) {
380         // first thing in the exception entry block should be the exception object.
381         push_exception = true;
382     }
383
384     while (!bailed_out() && last()->as_BlockEnd() == NULL &&
385           (code = stream()->next()) != ciBytecodeStream::EOBC() &&
386           (block_at(s.cur_bci()) == NULL || block_at(s.cur_bci()) == block())) {
387     .....
388
389         // handle bytecode
390         switch (code) {
391             case Bytecodes::_nop : /* nothing to do */ break;
392         .....
393             case Bytecodes::_tableswitch : table_switch(); break;
394             case Bytecodes::_lookupswitch : lookup_switch(); break;
395             case Bytecodes::_ireturn : method_return(ipop()); break;
396             case Bytecodes::_lreturn : method_return(lpop()); break;
397             case Bytecodes::_freturn : method_return(fpop()); break;
398             case Bytecodes::_dreturn : method_return(dpop()); break;
399             case Bytecodes::_areturn : method_return(apop()); break;
400             case Bytecodes::_return : method_return(NULL ); break;
401             case Bytecodes::_getstatic : // fall through
402             case Bytecodes::_putstatic : // fall through
403             case Bytecodes::_getfield : // fall through
404             case Bytecodes::_putfield : access_field(code); break;
405             case Bytecodes::_invokevirtual : // fall through
406             case Bytecodes::_invokespecial : // fall through
407             case Bytecodes::_invokestatic : // fall through
408             case Bytecodes::_invokedynamic : // fall through
409             case Bytecodes::_invokeinterface : invoke(code); break;
410         .....
411             default : ShouldNotReachHere(); break;
412         }
413
414         if (log != NULL)
415             log->clear_context(); // skip marker if nothing was printed
416
417         // save current bci to setup Goto at the end
418         prev_bci = s.cur_bci();
419     }
420 }
421 .....
422
423 // done
424 return end;
425 }
426 以字节码“ 1: getfield #423 // Field value:[C”为例,
427 上面的代码就会生成对应的LoadField Instruction(LoadField* load = new LoadField(obj, offset, field, false, state_before, needs_patching)) .
428 LoadField的定义为:
429
430 //通过宏定义,为LoadField添加了as_LoadField(), LoadField()和visit(InstructionVisitor)方法.

```

```

431 //这儿的LoadField.visit方法会调用do_LoadField(this)方法,此方法会在两个类中进行实现.
432 LEAF(LoadField, AccessField)
433
434 BASE(AccessField, Instruction)
435
436 #define BASE(class_name, super_class_name) \
437     class class_name: public super_class_name { \
438     public: \
439         virtual class_name* as_##class_name() { return this; } \
440
441 #define LEAF(class_name, super_class_name) \
442     BASE(class_name, super_class_name) \
443     public: \
444         virtual const char* name() const { return #class_name; } \
445         virtual void visit(InstructionVisitor* v) { v->do_##class_name(this); } \
446
447 //代码在c1_InstructionPrinter.cpp中,在打印HIR的指令到output.cfg中时会调用.
448 void InstructionPrinter::do_LoadField(LoadField* x) {
449     print_field(x);
450     output()->print(" (%c", type2char(x->field()->type()->basic_type()));
451     output()->print(" %s", x->field()->name()->as_utf8());
452 }
453
454 //代码在c1_LIRGenerator.cpp中,在生成LIR代码时调用.
455 void LIRGenerator::do_LoadField(LoadField* x) {
456     bool needs_patching = x->needs_patching();
457     bool is_volatile = x->field()->is_volatile();
458     BasicType field_type = x->field_type();
459
460     CodeEmitInfo* info = NULL;
461     if (needs_patching) {
462         assert(x->explicit_null_check() == NULL, "can't fold null check into patching field access");
463         info = state_for(x, x->state_before());
464     } else if (x->needs_null_check()) {
465         NullCheck* nc = x->explicit_null_check();
466         if (nc == NULL) {
467             info = state_for(x);
468         } else {
469             info = state_for(nc);
470         }
471     }
472
473     LIRItem object(x->obj(), this);
474
475     object.load_item();
476
477 #ifndef PRODUCT
478     if (PrintNotLoaded && needs_patching) {
479         tty->print_cr("   ##class not loaded at load_%s bci %d",
480             x->is_static() ? "static" : "field", x->printable_bci());
481     }
482 #endif
483
484     bool stress_deopt = StressLoopInvariantCodeMotion && info && info->deoptimize_on_exception();
485     if (x->needs_null_check() &&
486         (needs_patching ||
487          MacroAssembler::needs_explicit_null_check(x->offset()) ||
488          stress_deopt)) {
489         LIR_Opr obj = object.result();
490         if (stress_deopt) {
491             obj = new_register(T_OBJECT);
492             __ move(LIR_OprFact::oopConst(NULL), obj);
493         }
494         // emit an explicit null check because the offset is too large
495         __ null_check(obj, new CodeEmitInfo(info));
496     }
497
498     LIR_Opr reg = rlock_result(x, field_type);
499     LIR_Address* address;
500     if (needs_patching) {
501         // we need to patch the offset in the instruction so don't allow
502         // generate_address to try to be smart about emitting the -1.
503         // Otherwise the patching code won't know how to find the
504         // instruction to patch.
505         address = new LIR_Address(object.result(), PATCHED_ADDR, field_type);
506     } else {
507         address = generate_address(object.result(), x->offset(), field_type);
508     }
509
510     if (is_volatile && !needs_patching) {
511         volatile_field_load(address, reg, info);
512     } else {
513         LIR_PatchCode patch_code = needs_patching ? lir_patch_normal : lir_patch_none;
514         __ load(address, reg, info, patch_code);
515     }
516 }

```

```

517         if (is_volatile && os::is_MP()) {
518             __ membar_acquire();
519         }
520     }
521 }
522 最后,生成的高级中间表示(HIR)为:
523
524 begin_cfg
525     name "After Generation of HIR"
526     begin_block
527         name "B12"
528         from_bci 0
529         to_bci 0
530         predecessors
531         successors "B13"
532         xhandlers
533         flags
534         begin_states
535             begin_locals
536                 size 6
537                 method "virtual jint java.lang.String.indexOf(jint, jint)"
538     0 a12
539     1 i13
540     2 i14
541         end_locals
542         end_states
543         begin_HIR
544     .0 0 47 std entry B13 <|@
545         end_HIR
546     end_block
547     begin_block
548         name "B13"
549         from_bci 0
550         to_bci 0
551         predecessors "B12"
552         successors "B0"
553         xhandlers
554         flags "std"
555         begin_states
556             begin_locals
557                 size 6
558                 method "virtual jint java.lang.String.indexOf(jint, jint)"
559     0 a12
560     1 i13
561     2 i14
562         end_locals
563         end_states
564         begin_HIR
565     .0 0 46 goto B0 <|@
566         end_HIR
567     end_block
568     begin_block
569         name "B0"
570         from_bci 0
571         to_bci 7
572         predecessors "B13"
573         successors "B2" "B1"
574         xhandlers
575         flags "std"
576         begin_states
577             begin_locals
578                 size 6
579                 method "virtual jint java.lang.String.indexOf(jint, jint)"
580 //格式为index value SSA(静态单一赋值,即每个变量只会赋值一次)
581     0 a12
582     1 i13
583     2 i14
584         end_locals
585         end_states
586         begin_HIR
587 //格式为pin.bci usecnt SSA type+id do_LoadField(obj.field type name)
588     .1 0 a15 a12_12 ([]) value <|@ // 1: getField对应的HIR
589     .4 0 i16 a15.length <|@
590 //bci usecnt type+id do_Constant(Int0)
591     7 0 i17 0 <|@
592 //pin.bci usecnt id do_If(if x cond y then block0 else block1)
593     .7 0 18 if i14 >= i17 then B2 else B1 <|@
594         end_HIR
595     end_block
596     begin_block
597         name "B1"
598         from_bci 10
599         to_bci 12
600         predecessors "B0"
601         successors "B3"
602         xhandlers

```

```

603     flags
604     begin_states
605     begin_locals
606     size 6
607     method "virtual jint java.lang.String.indexOf(jint, jint)"
608 0 a12
609 1 i13
610 3 i16
611     end_locals
612     end_states
613     begin_HIR
614 10 0 i19 0 <|@
615 .12 0 20 goto B3 <|@
616     end_HIR
617     end_block
618     begin_block
619     name "B3"
620     from_bci 22
621     to_bci 25
622     predecessors "B1" "B2"
623     successors "B6" "B5"
624     xhandlers
625     flags
626     begin_states
627     begin_locals
628     size 6
629     method "virtual jint java.lang.String.indexOf(jint, jint)"
630 0 a12
631 1 i13
632 2 i22 [ i19 i14]
633 3 i16
634     end_locals
635     end_states
636     begin_HIR
637 23 0 i25 65536 <|@
638 .25 0 26 if i13 >= i25 then B6 else B5 <|@
639     end_HIR
640     end_block
641     begin_block
642     name "B5"
643     from_bci 28
644     to_bci 37
645     predecessors "B3"
646     successors "B7"
647     xhandlers
648     flags
649     begin_states
650     begin_locals
651     size 6
652     method "virtual jint java.lang.String.indexOf(jint, jint)"
653 0 a12
654 1 i13
655 2 i22
656 3 i16
657     end_locals
658     end_states
659     begin_HIR
660 .29 0 a27 a12._12 ([) value <|@
661 .37 0 28 goto B7 <|@
662     end_HIR
663     end_block
664     begin_block
665     name "B7"
666     from_bci 37
667     to_bci 40
668     predecessors "B5" "B11"
669     successors "B9" "B8"
670     xhandlers
671     flags "plh"
672     begin_states
673     begin_locals
674     size 6
675     method "virtual jint java.lang.String.indexOf(jint, jint)"
676 1 i13
677 3 i16
678 4 a27
679 5 i29 [ i22 i35]
680     end_locals
681     end_states
682     begin_HIR
683 .40 0 30 if i29 >= i16 then B9 else B8 <|@
684     end_HIR
685     end_block
686     begin_block
687     name "B8"
688     from_bci 43

```



```

689     to_bci 49
690     predecessors "B7"
691     successors "B11" "B10"
692     xhandlers
693     flags
694     begin_states
695         begin_locals
696         size 6
697     method "virtual jint java.lang.String.indexOf(jint, jint)"
698 1 i13
699 3 i16
700 4 a27
701 5 i29
702     end_locals
703     end_states
704     begin_HIR
705 .47 0 i31 a27[i29] (C) [rc] <|@
706 .49 0 32 if i31 != i13 then B11 else B10 <|@
707     end_HIR
708     end_block
709     begin_block
710     name "B10"
711     from_bci 52
712     to_bci 54
713     predecessors "B8"
714     successors
715     xhandlers
716     flags
717     begin_states
718         begin_locals
719         size 6
720     method "virtual jint java.lang.String.indexOf(jint, jint)"
721 5 i29
722     end_locals
723     end_states
724     begin_HIR
725 .54 0 i33 ireturn i29 <|@
726     end_HIR
727     end_block
728     begin_block
729     name "B11"
730     from_bci 55
731     to_bci 58
732     predecessors "B8"
733     successors "B7"
734     xhandlers
735     flags
736     begin_states
737         begin_locals
738         size 6
739     method "virtual jint java.lang.String.indexOf(jint, jint)"
740 1 i13
741 3 i16
742 4 a27
743 5 i29
744     end_locals
745     end_states
746     begin_HIR
747 55 0 i34 1 <|@
748 55 0 i35 i29 + i34 <|@
749 .58 0 36 goto B7 (safepoint) <|@
750     end_HIR
751     end_block
752     begin_block
753     name "B9"
754     from_bci 61
755     to_bci 62
756     predecessors "B7"
757     successors
758     xhandlers
759     flags
760     begin_states
761         begin_locals
762         size 6
763     method "virtual jint java.lang.String.indexOf(jint, jint)"
764     end_locals
765     end_states
766     begin_HIR
767 61 0 i37 -1 <|@
768 .62 0 i38 ireturn i37 <|@
769     end_HIR
770     end_block
771     begin_block
772     name "B6"
773     from_bci 63
774     to_bci 69

```

```

775     predecessors "B3"
776     successors
777     xhandlers
778     flags
779     begin_states
780     begin_locals
781     size 6
782     method "virtual jint java.lang.String.indexOf(jint, jint)"
783 0 a12
784 1 i13
785 2 i22
786     end_locals
787     end_states
788     begin_HIR
789 .66 0 a39 null_check(a12) <|@
790 .66 0 v41 profile a12 java/lang/String.indexOf() <|@
791 .66 0 i42 a12.invokespecial(i13, i22)
792     java/lang/String.indexOfSupplementary(II)I <|@
793 .69 0 i43 ireturn i42 <|@
794     end_HIR
795     end_block
796     begin_block
797     name "B2"
798     from_bci 15
799     to_bci 17
800     predecessors "B0"
801     successors "B3" "B4"
802     xhandlers
803     flags
804     begin_states
805     begin_locals
806     size 6
807     method "virtual jint java.lang.String.indexOf(jint, jint)"
808 0 a12
809 1 i13
810 2 i14
811 3 i16
812     end_locals
813     end_states
814     begin_HIR
815 .17 0 21 if i14 < i16 then B3 else B4 <|@
816     end_HIR
817     end_block
818     begin_block
819     name "B4"
820     from_bci 20
821     to_bci 21
822     predecessors "B2"
823     successors
824     xhandlers
825     flags
826     begin_states
827     begin_locals
828     size 6
829     method "virtual jint java.lang.String.indexOf(jint, jint)"
830     end_locals
831     end_states
832     begin_HIR
833 20 0 i23 -1 <|@
834 .21 0 i24 ireturn i23 <|@
835     end_HIR
836     end_block
837 end_cfg
838 begin_cfg
839 name "Before RangeCheckElimination"
840 begin_block
841 name "B12"
842 from_bci 0
843 to_bci 0
844 predecessors
845 successors "B13"
846 xhandlers
847 flags
848 begin_states
849 begin_locals
850 size 6
851 method "virtual jint java.lang.String.indexOf(jint, jint)"
852 0 a12
853 1 i13
854 2 i14
855     end_locals
856     end_states
857     begin_HIR
858 .0 0 47 std entry B13 <|@
859     end_HIR
860     end_block

```

```

861     begin_block
862         name "B13"
863         from_bci 0
864         to_bci 0
865         predecessors "B12"
866         successors "B0"
867         xhandlers
868         flags "std"
869         dominator "B12"
870         begin_states
871             begin_locals
872                 size 6
873             method "virtual jint java.lang.String.indexOf(jint, jint)"
874     0 a12
875     1 i13
876     2 i14
877         end_locals
878         end_states
879         begin_HIR
880 .0 0 46 goto B0 <|@
881         end_HIR
882     end_block
883     begin_block
884         name "B0"
885         from_bci 0
886         to_bci 7
887         predecessors "B13"
888         successors "B2" "B1"
889         xhandlers
890         flags "std"
891         dominator "B13"
892         begin_states
893             begin_locals
894                 size 6
895             method "virtual jint java.lang.String.indexOf(jint, jint)"
896     0 a12
897     1 i13
898     2 i14
899         end_locals
900         end_states
901         begin_HIR
902 .1 0 a15 a12_12 ([]) value <|@
903 .4 0 i16 a15.length <|@
904 7 0 i17 0 <|@
905 .7 0 18 if i14 >= i17 then B2 else B1 <|@
906         end_HIR
907     end_block
908     begin_block
909         name "B1"
910         from_bci 10
911         to_bci 12
912         predecessors "B0"
913         successors "B3"
914         xhandlers
915         flags
916         dominator "B0"
917         begin_states
918             begin_locals
919                 size 6
920             method "virtual jint java.lang.String.indexOf(jint, jint)"
921     0 a12
922     1 i13
923     3 i16
924         end_locals
925         end_states
926         begin_HIR
927 .12 0 20 goto B3 <|@
928         end_HIR
929     end_block
930     begin_block
931         name "B3"
932         from_bci 22
933         to_bci 25
934         predecessors "B1" "B14"
935         successors "B6" "B5"
936         xhandlers
937         flags
938         dominator "B0"
939         begin_states
940             begin_locals
941                 size 6
942             method "virtual jint java.lang.String.indexOf(jint, jint)"
943     0 a12
944     1 i13
945     2 i22 [ i17 i14]
946     3 i16

```

```

947     end_locals
948     end_states
949     begin_HIR
950 23 0 i25 65536 <|@
951 .25 0 26 if i13 >= i25 then B6 else B5 <|@
952     end_HIR
953     end_block
954     begin_block
955         name "B5"
956         from_bci 28
957         to_bci 37
958         predecessors "B3"
959         successors "B7"
960         xhandlers
961         flags
962         dominator "B3"
963         begin_states
964         begin_locals
965             size 6
966             method "virtual jint java.lang.String.indexOf(jint, jint)"
967 0 a12
968 1 i13
969 2 i22
970 3 i16
971     end_locals
972     end_states
973     begin_HIR
974 .37 0 28 goto B7 <|@
975     end_HIR
976     end_block
977     begin_block
978         name "B7"
979         from_bci 37
980         to_bci 40
981         predecessors "B5" "B11"
982         successors "B9" "B8"
983         xhandlers
984         flags "bb" "plh" "llh"
985         dominator "B5"
986         loop_index 0
987         loop_depth 1
988         begin_states
989         begin_locals
990             size 6
991             method "virtual jint java.lang.String.indexOf(jint, jint)"
992 1 i13
993 3 i16
994 4 a15
995 5 i29 [ i22 i35]
996     end_locals
997     end_states
998     begin_HIR
999 .40 0 30 if i29 >= i16 then B9 else B8 <|@
1000     end_HIR
1001     end_block
1002     begin_block
1003         name "B8"
1004         from_bci 43
1005         to_bci 49
1006         predecessors "B7"
1007         successors "B11" "B10"
1008         xhandlers
1009         flags
1010         dominator "B7"
1011         loop_index 0
1012         loop_depth 1
1013         begin_states
1014         begin_locals
1015             size 6
1016             method "virtual jint java.lang.String.indexOf(jint, jint)"
1017 1 i13
1018 3 i16
1019 4 a15
1020 5 i29
1021     end_locals
1022     end_states
1023     begin_HIR
1024 .47 0 i31 a15[i29] (C) [rc] <|@
1025 .49 0 32 if i31 != i13 then B11 else B10 <|@
1026     end_HIR
1027     end_block
1028     begin_block
1029         name "B10"
1030         from_bci 52
1031         to_bci 54
1032         predecessors "B8"

```

```

1033     successors
1034     xhandlers
1035     flags
1036     dominator "B8"
1037     begin_states
1038     begin_locals
1039     size 6
1040     method "virtual jint java.lang.String.indexOf(jint, jint)"
1041 5 i29
1042     end_locals
1043     end_states
1044     begin_HIR
1045 .54 0 i33 ireturn i29 <|@
1046     end_HIR
1047     end_block
1048     begin_block
1049     name "B11"
1050     from_bci 55
1051     to_bci 58
1052     predecessors "B8"
1053     successors "B7"
1054     xhandlers
1055     flags "lle"
1056     dominator "B8"
1057     loop_index 0
1058     loop_depth 1
1059     begin_states
1060     begin_locals
1061     size 6
1062     method "virtual jint java.lang.String.indexOf(jint, jint)"
1063 1 i13
1064 3 i16
1065 4 a15
1066 5 i29
1067     end_locals
1068     end_states
1069     begin_HIR
1070 55 0 i34 1 <|@
1071 55 0 i35 i29 + i34 <|@
1072 .58 0 36 goto B7 (safepoint) <|@
1073     end_HIR
1074     end_block
1075     begin_block
1076     name "B9"
1077     from_bci 61
1078     to_bci 62
1079     predecessors "B7"
1080     successors
1081     xhandlers
1082     flags
1083     dominator "B7"
1084     begin_states
1085     begin_locals
1086     size 6
1087     method "virtual jint java.lang.String.indexOf(jint, jint)"
1088     end_locals
1089     end_states
1090     begin_HIR
1091 61 0 i37 -1 <|@
1092 .62 0 i38 ireturn i37 <|@
1093     end_HIR
1094     end_block
1095     begin_block
1096     name "B6"
1097     from_bci 63
1098     to_bci 69
1099     predecessors "B3"
1100     successors
1101     xhandlers
1102     flags
1103     dominator "B3"
1104     begin_states
1105     begin_locals
1106     size 6
1107     method "virtual jint java.lang.String.indexOf(jint, jint)"
1108 0 a12
1109 1 i13
1110 2 i22
1111     end_locals
1112     end_states
1113     begin_HIR
1114 .66 0 a39 null_check(a12) <|@
1115 .66 0 v41 profile a12 java/lang/String.indexOf) <|@
1116 .66 0 i42 a12.invokespecial(i13, i22)
1117     java/lang/String.indexOfSupplementary(II)I <|@
1118 .69 0 i43 ireturn i42 <|@

```

```

1119     end_HIR
1120   end_block
1121   begin_block
1122     name "B2"
1123     from_bci 15
1124     to_bci 17
1125     predecessors "B0"
1126     successors "B14" "B4"
1127     xhandlers
1128     flags
1129     dominator "B0"
1130     begin_states
1131       begin_locals
1132         size 6
1133         method "virtual jint java.lang.String.indexOf(jint, jint)"
1134   0 a12
1135   1 i13
1136   2 i14
1137   3 i16
1138     end_locals
1139     end_states
1140     begin_HIR
1141   .17 0 21 if i14 < i16 then B14 else B4 <|@
1142     end_HIR
1143   end_block
1144   begin_block
1145     name "B4"
1146     from_bci 20
1147     to_bci 21
1148     predecessors "B2"
1149     successors
1150     xhandlers
1151     flags
1152     dominator "B2"
1153     begin_states
1154       begin_locals
1155         size 6
1156         method "virtual jint java.lang.String.indexOf(jint, jint)"
1157     end_locals
1158     end_states
1159     begin_HIR
1160   20 0 i23 -1 <|@
1161   .21 0 i24 ireturn i23 <|@
1162     end_HIR
1163   end_block
1164   begin_block
1165     name "B14"
1166     from_bci 22
1167     to_bci 22
1168     predecessors "B2"
1169     successors "B3"
1170     xhandlers
1171     flags "ces"
1172     dominator "B2"
1173     begin_states
1174       begin_locals
1175         size 6
1176         method "virtual jint java.lang.String.indexOf(jint, jint)"
1177   0 a12
1178   1 i13
1179   2 i14
1180   3 i16
1181     end_locals
1182     end_states
1183     begin_HIR
1184   .22 0 49 goto B3 <|@
1185     end_HIR
1186   end_block
1187 end_cfg
1188 生成LIR
1189 现在,需要根据HIR生成对应的LIR.
1190 Call stack to create LIR:
1191
1192 int Compilation::compile_java_method()
1193   void Compilation::emit_lir()
1194     void IR::iterate_linear_scan_order(BlockClosure* closure) //closure为LIRGenerator gen(this, method())
1195     void LIRGenerator::block_do(BlockBegin* block)
1196
1197 //通过如下的方法,对每个block里的每个instruction调用visit.
1198 //前面生成HIR时,说到过的方法void LIRGenerator::do_LoadField(LoadField* x) 就在此调用并生成LIR.
1199 void LIRGenerator::block_do(BlockBegin* block) {
1200   CHECK_BAILOUT();
1201
1202   block_do_prolog(block);
1203   set_block(block);
1204

```

```

1205     for (Instruction* instr = block; instr != NULL; instr = instr->next()) {
1206         if (instr->is_pinned()) do_root(instr);
1207     }
1208
1209     set_block(NULL);
1210     block_do_epilog(block);
1211 }
1212
1213 // This is where the tree-walk starts; instr must be root;
1214 void LIRGenerator::do_root(Value instr) {
1215     CHECK_BAILOUT();
1216
1217     InstructionMark im(compilation(), instr);
1218
1219     assert(instr->is_pinned(), "use only with roots");
1220     assert(instr->subst() == instr, "shouldn't have missed substitution");
1221
1222     instr->visit(this);
1223
1224     assert(!instr->has_uses() || instr->operand()->is_valid() ||
1225           instr->as_Constant() != NULL || bailed_out(), "invalid item set");
1226 }
1227
1228 //代码在c1_LIR.cpp中
1229 //do_LoadField方法调用下面方法生成LIR_Op1: lir_move.
1230 void LIR_List::load(LIR_Address* addr, LIR_Opr src, CodeEmitInfo* info, LIR_PatchCode patch_code) {
1231     append(new LIR_Op1(
1232         lir_move,
1233         LIR_OprFact::address(addr),
1234         src,
1235         addr->type(),
1236         patch_code,
1237         info));
1238 }
1239
1240 //代码在c1_LIR.cpp中
1241 //此方法会在后面的线性扫描方法中调用,它对每个LIR_Op会做一些标记.
1242 //比如,lir_move它有输入和输出,所以有do_input和do_output.
1243 //在线性扫描中,如果有input,那么就会标记对应寄存器的使用区间(Interval),有output就会定义一个寄存器开始使用的位置.
1244 void LIR_OpVisitState::visit(LIR_Op* op) {
1245     // copy information from the LIR_Op
1246     reset();
1247     set_op(op);
1248
1249     switch (op->code()) {
1250
1251     .....
1252     case lir_move:           // input and result always valid, may have info
1253     case lir_pack64:         // input and result always valid
1254     case lir_unpack64:       // input and result always valid
1255     case lir_prefetchr:      // input always valid, result and info always invalid
1256     case lir_prefetchw:      // input always valid, result and info always invalid
1257     {
1258         assert(op->as_Op1() != NULL, "must be");
1259         LIR_Op1* op1 = (LIR_Op1*)op;
1260
1261         if (op1->_info)       do_info(op1->_info);
1262         if (op1->_opr->is_valid()) do_input(op1->_opr);
1263         if (op1->_result->is_valid()) do_output(op1->_result);
1264
1265         break;
1266     }
1267     .....
1268 }
1269 最后,生成的LIR为:
1270
1271 begin_cfg
1272     name "Before Register Allocation"
1273     begin_block
1274         name "B12"
1275         from_bci 0
1276         to_bci 0
1277         predecessors
1278         successors "B13"
1279         xhandlers
1280         flags
1281         first_lir_id 0
1282         last_lir_id 28
1283         begin_states
1284             begin_locals
1285                 size 6
1286                 method "virtual jint java.lang.String.indexOf(jint, jint)"
1287     0 a12 "[R177|L]"
1288     1 i13 "[R178|I]"
1289     2 i14 "[R179|I]"
1290         end_locals

```

```

1291     end_states
1292     begin_HIR
1293 .0 0 47 std entry B13 <|@
1294     end_HIR
1295     begin_LIR
1296     0 label [label:0xfc0043b8] <|@
1297     2 std_entry <|@
1298     4 move [rsi|L] [R177|L] <|@
1299     6 move [rdx|I] [R178|I] <|@
1300     8 move [rcx|I] [R179|I] <|@
1301     10 move [metadata:0x3301c100|M] [R180|M] <|@
1302     12 move [Base:[R180|M] Disp: 108|I] [R181|I] <|@
1303     14 add [R181|I] [int:8|I] [R181|I] <|@
1304     16 move [R181|I] [Base:[R180|M] Disp: 108|I] <|@
1305     18 move [metadata:0x32dd2238|M] [R182|M] <|@
1306     20 logic_and [R181|I] [int:8184|I] [R181|I] <|@
1307     22 cmp [R181|I] [int:0|I] <|@
1308     24 branch [EQ] [CounterOverflowStub: 0xf8025908] <|@
1309     26 label [label:0xf8025930] <|@
1310     28 branch [AL] [B13] <|@
1311     end_LIR
1312     end_block
1313     begin_block
1314     name "B13"
1315     from_bci 0
1316     to_bci 0
1317     predecessors "B12"
1318     successors "B0"
1319     xhandlers
1320     flags "std"
1321     dominator "B12"
1322     first_lir_id 30
1323     last_lir_id 32
1324     begin_states
1325     begin_locals
1326     size 6
1327     method "virtual jint java.lang.String.indexOf(jint, jint)"
1328     0 a12 "[R177|L]"
1329     1 i13 "[R178|I]"
1330     2 i14 "[R179|I]"
1331     end_locals
1332     end_states
1333     begin_HIR
1334 .0 0 46 goto B0 <|@
1335     end_HIR
1336     begin_LIR
1337     30 label [label:0xfc0046c8] <|@
1338     32 branch [AL] [B0] <|@
1339     end_LIR
1340     end_block
1341     begin_block
1342     name "B0"
1343     from_bci 0
1344     to_bci 7
1345     predecessors "B13"
1346     successors "B2" "B1"
1347     xhandlers
1348     flags "std"
1349     dominator "B13"
1350     first_lir_id 34
1351     last_lir_id 54
1352     begin_states
1353     begin_locals
1354     size 6
1355     method "virtual jint java.lang.String.indexOf(jint, jint)"
1356     #index value SSA
1357     0 a12 "[R177|L]"
1358     1 i13 "[R178|I]"
1359     2 i14 "[R179|I]"
1360     end_locals
1361     end_states
1362     begin_HIR
1363     #pin.bci usecnt SSA type+id do_LoadField(obj.field type name)
1364     .1 6 "[R183|L]" a15 a12._12 ([] value <|@
1365     .4 11 "[R184|I]" i16 a15.length <|@
1366     #bci usecnt type+id do_Constant(Int0)
1367     7 2 i17 0 <|@
1368     #pin.bci usecnt id do_If(if x cond y then block0 else block1)
1369     .7 0 18 if i14 >= i17 then B2 else B1 <|@
1370     end_HIR
1371     begin_LIR
1372     34 label [label:0xf801ee78] <|@
1373     36 move [Base:[R177|L] Disp: 12|L] [R183|L] <|@ //对应的HIR为 .1 6 "[R183|L]" a15 a12._12 ([] value <|@
1374     38 move [Base:[R183|L] Disp: 12|I] [R184|I] [bci:4] <|@ //对应的HIR为 .4 11 "[R184|I]" i16 a15.length <|@
1375     40 cmp [R179|I] [int:0|I] <|@
1376     42 move [metadata:0x3301c100|M] [R185|M] <|@

```



```

1377 44 cmove [GE] [lng:152|J] [lng:168|J] [R186|J] <|@
1378 46 move [Base:[R185|M] Index:[R186|J] Disp: 0|J] [R187|J] <|@
1379 48 leal [Base:[R187|J] Disp: 1|I] [R187|J] <|@
1380 50 move [R187|J] [Base:[R185|M] Index:[R186|J] Disp: 0|J] <|@
1381 52 branch [GE] [B2] <|@
1382 54 branch [AL] [B1] <|@
1383     end_LIR
1384 end_block
1385 begin_block
1386     name "B1"
1387     from_bci 10
1388     to_bci 12
1389     predecessors "B0"
1390     successors "B3"
1391     xhandlers
1392     flags
1393     dominator "B0"
1394     first_lir_id 56
1395     last_lir_id 64
1396     begin_states
1397     begin_locals
1398         size 6
1399     method "virtual jint java.lang.String.indexOf(jint, jint)"
1400 0 a12 "[R177|L]"
1401 1 i13 "[R178|I]"
1402 3 i16 "[R184|I]"
1403     end_locals
1404     end_states
1405     begin_HIR
1406 .12 0 20 goto B3 <|@
1407     end_HIR
1408     begin_LIR
1409 56 label [label:0xf801f198] <|@
1410 58 move [metadata:0x3301c100|M] [R188|M] <|@
1411 60 add [Base:[R188|M] Disp: 184|J] [int:1|I] [Base:[R188|M] Disp: 184|J] <|@
1412 62 move [int:0|I] [R189|I] <|@
1413 64 branch [AL] [B3] <|@
1414     end_LIR
1415 end_block
1416 begin_block
1417     name "B2"
1418     from_bci 15
1419     to_bci 17
1420     predecessors "B0"
1421     successors "B14" "B4"
1422     xhandlers
1423     flags
1424     dominator "B0"
1425     first_lir_id 66
1426     last_lir_id 82
1427     begin_states
1428     begin_locals
1429         size 6
1430     method "virtual jint java.lang.String.indexOf(jint, jint)"
1431 0 a12 "[R177|L]"
1432 1 i13 "[R178|I]"
1433 2 i14 "[R179|I]"
1434 3 i16 "[R184|I]"
1435     end_locals
1436     end_states
1437     begin_HIR
1438 .17 0 21 if i14 < i16 then B14 else B4 <|@
1439     end_HIR
1440     begin_LIR
1441 66 label [label:0xf801f4b8] <|@
1442 68 cmp [R179|I] [R184|I] <|@
1443 70 move [metadata:0x3301c100|M] [R190|M] <|@
1444 72 cmove [LT] [lng:208|J] [lng:224|J] [R191|J] <|@
1445 74 move [Base:[R190|M] Index:[R191|J] Disp: 0|J] [R192|J] <|@
1446 76 leal [Base:[R192|J] Disp: 1|I] [R192|J] <|@
1447 78 move [R192|J] [Base:[R190|M] Index:[R191|J] Disp: 0|J] <|@
1448 80 branch [LT] [B14] <|@
1449 82 branch [AL] [B4] <|@
1450     end_LIR
1451 end_block
1452 begin_block
1453     name "B14"
1454     from_bci 22
1455     to_bci 22
1456     predecessors "B2"
1457     successors "B3"
1458     xhandlers
1459     flags "ces"
1460     dominator "B2"
1461     first_lir_id 84
1462     last_lir_id 88

```

```

1463     begin_states
1464     begin_locals
1465     size 6
1466     method "virtual jint java.lang.String.indexOf(jint, jint)"
1467     0 a12 "[R177|L]"
1468     1 i13 "[R178|I]"
1469     2 i14 "[R179|I]"
1470     3 i16 "[R184|I]"
1471     end_locals
1472     end_states
1473     begin_HIR
1474     .22 0 49 goto B3 <|@
1475     end_HIR
1476     begin_LIR
1477     84 label [label:0xfc0062c8] <|@
1478     86 move [R179|I] [R189|I] <|@
1479     88 branch [AL] [B3] <|@
1480     end_LIR
1481     end_block
1482     begin_block
1483     name "B3"
1484     from_bci 22
1485     to_bci 25
1486     predecessors "B1" "B14"
1487     successors "B6" "B5"
1488     xhandlers
1489     flags
1490     dominator "B0"
1491     first_lir_id 90
1492     last_lir_id 106
1493     begin_states
1494     begin_locals
1495     size 6
1496     method "virtual jint java.lang.String.indexOf(jint, jint)"
1497     0 a12 "[R177|L]"
1498     1 i13 "[R178|I]"
1499     2 i22 [ i17 i14] "[R189|I]"
1500     3 i16 "[R184|I]"
1501     end_locals
1502     end_states
1503     begin_HIR
1504     23 1 i25 65536 <|@
1505     .25 0 26 if i13 >= i25 then B6 else B5 <|@
1506     end_HIR
1507     begin_LIR
1508     90 label [label:0xf801f7d8] <|@
1509     92 cmp [R178|I] [int:65536|I] <|@
1510     94 move [metadata:0x3301c100|M] [R193|M] <|@
1511     96 cmove [GE] [lng:240|J] [lng:256|J] [R194|J] <|@
1512     98 move [Base:[R193|M] Index:[R194|J] Disp: 0|J] [R195|J] <|@
1513     100 leal [Base:[R195|J] Disp: 1|I] [R195|J] <|@
1514     102 move [R195|J] [Base:[R193|M] Index:[R194|J] Disp: 0|J] <|@
1515     104 branch [GE] [B6] <|@
1516     106 branch [AL] [B5] <|@
1517     end_LIR
1518     end_block
1519     begin_block
1520     name "B5"
1521     from_bci 28
1522     to_bci 37
1523     predecessors "B3"
1524     successors "B7"
1525     xhandlers
1526     flags
1527     dominator "B3"
1528     first_lir_id 108
1529     last_lir_id 112
1530     begin_states
1531     begin_locals
1532     size 6
1533     method "virtual jint java.lang.String.indexOf(jint, jint)"
1534     0 a12 "[R177|L]"
1535     1 i13 "[R178|I]"
1536     2 i22 "[R189|I]"
1537     3 i16 "[R184|I]"
1538     end_locals
1539     end_states
1540     begin_HIR
1541     .37 0 28 goto B7 <|@
1542     end_HIR
1543     begin_LIR
1544     108 label [label:0xf801fe18] <|@
1545     110 move [R189|I] [R196|I] <|@
1546     112 branch [AL] [B7] <|@
1547     end_LIR
1548     end_block

```

```

1549     begin_block
1550     name "B7"
1551     from_bci 37
1552     to_bci 40
1553     predecessors "B5" "B11"
1554     successors "B9" "B8"
1555     xhandlers
1556     flags "bb" "plh" "llh"
1557     dominator "B5"
1558     loop_index 0
1559     loop_depth 1
1560     first_lir_id 114
1561     last_lir_id 130
1562     begin_states
1563     begin_locals
1564     size 6
1565     method "virtual jint java.lang.String.indexOf(jint, jint)"
1566 1 i13 "[R178|I]"
1567 3 i16 "[R184|I]"
1568 4 a15 "[R183|L]"
1569 5 i29 [ i22 i35] "[R196|I]"
1570     end_locals
1571     end_states
1572     begin_HIR
1573 .40 0 30 if i29 >= i16 then B9 else B8 <|@
1574     end_HIR
1575     begin_LIR
1576 114 label [label:0xf8020458] <|@
1577 116 cmp [R196|I] [R184|I] <|@
1578 118 move [metadata:0x3301c100|M] [R197|M] <|@
1579 120 cmove [GE] [lng:272|J] [lng:288|J] [R198|J] <|@
1580 122 move [Base:[R197|M] Index:[R198|J] Disp: 0|J] [R199|J] <|@
1581 124 leal [Base:[R199|J] Disp: 1|I] [R199|J] <|@
1582 126 move [R199|J] [Base:[R197|M] Index:[R198|J] Disp: 0|J] <|@
1583 128 branch [GE] [B9] <|@
1584 130 branch [AL] [B8] <|@
1585     end_LIR
1586     end_block
1587     begin_block
1588     name "B8"
1589     from_bci 43
1590     to_bci 49
1591     predecessors "B7"
1592     successors "B11" "B10"
1593     xhandlers
1594     flags
1595     dominator "B7"
1596     loop_index 0
1597     loop_depth 1
1598     first_lir_id 132
1599     last_lir_id 156
1600     begin_states
1601     begin_locals
1602     size 6
1603     method "virtual jint java.lang.String.indexOf(jint, jint)"
1604 1 i13 "[R178|I]"
1605 3 i16 "[R184|I]"
1606 4 a15 "[R183|L]"
1607 5 i29 "[R196|I]"
1608     end_locals
1609     end_states
1610     begin_HIR
1611 .47 1 "[R201|I]" i31 a15[i29] (C) [rc] <|@
1612 .49 0 32 if i31 != i13 then B11 else B10 <|@
1613     end_HIR
1614     begin_LIR
1615 132 label [label:0xf8020778] <|@
1616 134 convert [i21] [R196|I] [R200|J] <|@
1617 136 cmp [R196|I] [Base:[R183|L] Disp: 12|I] <|@
1618 138 branch [AE] [RangeCheckStub: 0xf80296d8] [bci:47] <|@
1619 140 move [Base:[R183|L] Index:[R200|J] * 2 Disp: 16|C] [R201|I] <|@
1620 142 cmp [R201|I] [R178|I] <|@
1621 144 move [metadata:0x3301c100|M] [R202|M] <|@
1622 146 cmove [NE] [lng:304|J] [lng:320|J] [R203|J] <|@
1623 148 move [Base:[R202|M] Index:[R203|J] Disp: 0|J] [R204|J] <|@
1624 150 leal [Base:[R204|J] Disp: 1|I] [R204|J] <|@
1625 152 move [R204|J] [Base:[R202|M] Index:[R203|J] Disp: 0|J] <|@
1626 154 branch [NE] [B11] <|@
1627 156 branch [AL] [B10] <|@
1628     end_LIR
1629     end_block
1630     begin_block
1631     name "B11"
1632     from_bci 55
1633     to_bci 58
1634     predecessors "B8"

```

```

1635     successors "B7"
1636     xhandlers
1637     flags "lle"
1638     dominator "B8"
1639     loop_index 0
1640     loop_depth 1
1641     first_lir_id 158
1642     last_lir_id 190
1643     begin_states
1644     begin_locals
1645     size 6
1646     method "virtual jint java.lang.String.indexOf(jint, jint)"
1647 1 i13 "[R178|I]"
1648 3 i16 "[R184|I]"
1649 4 a15 "[R183|L]"
1650 5 i29 "[R196|I]"
1651     end_locals
1652     end_states
1653     begin_HIR
1654 55 1 i34 1 <|@
1655 .55 1 "[R205|I]" i35 i29 + i34 <|@
1656 .58 0 36 goto B7 (safepoint) <|@
1657     end_HIR
1658     begin_LIR
1659 158 label [label:0xf80210d8] <|@
1660 160 move [R196|I] [R205|I] <|@
1661 162 add [R205|I] [int:1|I] [R205|I] <|@
1662 164 move [metadata:0x3301c100|M] [R206|M] <|@
1663 166 move [Base:[R206|M] Disp: 112|I] [R207|I] <|@
1664 168 add [R207|I] [int:8|I] [R207|I] <|@
1665 170 move [R207|I] [Base:[R206|M] Disp: 112|I] <|@
1666 172 move [metadata:0x32dd2238|M] [R208|M] <|@
1667 174 logic_and [R207|I] [int:65528|I] [R207|I] <|@
1668 176 cmp [R207|I] [int:0|I] <|@
1669 178 branch [EQ] [CounterOverflowStub: 0xf802a488] <|@
1670 180 label [label:0xf802a4b0] <|@
1671 182 safepoint [bci:58] <|@
1672 184 move [metadata:0x3301c100|M] [R209|M] <|@
1673 186 add [Base:[R209|M] Disp: 336|J] [int:1|I] [Base:[R209|M] Disp: 336|J] <|@
1674 188 move [R205|I] [R196|I] <|@
1675 190 branch [AL] [B7] <|@
1676     end_LIR
1677     end_block
1678     begin_block
1679     name "B10"
1680     from_bci 52
1681     to_bci 54
1682     predecessors "B8"
1683     successors
1684     xhandlers
1685     flags
1686     dominator "B8"
1687     first_lir_id 192
1688     last_lir_id 196
1689     begin_states
1690     begin_locals
1691     size 6
1692     method "virtual jint java.lang.String.indexOf(jint, jint)"
1693 5 i29 "[R196|I]"
1694     end_locals
1695     end_states
1696     begin_HIR
1697 .54 0 i33 ireturn i29 <|@
1698     end_HIR
1699     begin_LIR
1700 192 label [label:0xf8020db8] <|@
1701 194 move [R196|I] [rax|I] <|@
1702 196 return [rax|I] <|@
1703     end_LIR
1704     end_block
1705     begin_block
1706     name "B9"
1707     from_bci 61
1708     to_bci 62
1709     predecessors "B7"
1710     successors
1711     xhandlers
1712     flags
1713     dominator "B7"
1714     first_lir_id 198
1715     last_lir_id 202
1716     begin_states
1717     begin_locals
1718     size 6
1719     method "virtual jint java.lang.String.indexOf(jint, jint)"
1720     end_locals

```

```

1721     end_states
1722     begin_HIR
1723 61 1 i37 -1 <|@
1724 .62 0 i38 ireturn i37 <|@
1725     end_HIR
1726     begin_LIR
1727 198 label [label:0xf8020a98] <|@
1728 200 move [int:-1|I] [rax|I] <|@
1729 202 return [rax|I] <|@
1730     end_LIR
1731     end_block
1732     begin_block
1733     name "B6"
1734     from_bci 63
1735     to_bci 69
1736     predecessors "B3"
1737     successors
1738     xhandlers
1739     flags
1740     dominator "B3"
1741     first_lir_id 204
1742     last_lir_id 222
1743     begin_states
1744     begin_locals
1745     size 6
1746     method "virtual jint java.lang.String.indexOf(jint, jint)"
1747 0 a12 "[R177|L]"
1748 1 i13 "[R178|I]"
1749 2 i22 "[R189|I]"
1750     end_locals
1751     end_states
1752     begin_HIR
1753 .66 0 a39 null_check(a12) (eliminated) <|@
1754 .66 0 v41 profile a12 java/lang/String.indexOf() <|@
1755 .66 1 "[R213|I]" i42 a12.invokespecial(i13, i22)
1756         java/lang/String.indexOfSupplementary(II)I <|@
1757 .69 0 i43 ireturn i42 <|@
1758     end_HIR
1759     begin_LIR
1760 204 label [label:0xf8020138] <|@
1761 206 move [R177|L] [R212|L] <|@
1762 208 profile_call indexOf.java/lang/String @ 66 [R210|L] [R212|L] [R211|J] <|@
1763 210 move [R178|I] [rdx|I] <|@
1764 212 move [R189|I] [rcx|I] <|@
1765 214 move [R177|L] [rsi|L] <|@
1766 216 optvirtual call: [addr: 0x0] [recv: [rsi|L]] [result: [rax|I]] [bci:66] <|@
1767 218 move [rax|I] [R213|I] <|@
1768 220 move [R213|I] [rax|I] <|@
1769 222 return [rax|I] <|@
1770     end_LIR
1771     end_block
1772     begin_block
1773     name "B4"
1774     from_bci 20
1775     to_bci 21
1776     predecessors "B2"
1777     successors
1778     xhandlers
1779     flags
1780     dominator "B2"
1781     first_lir_id 224
1782     last_lir_id 228
1783     begin_states
1784     begin_locals
1785     size 6
1786     method "virtual jint java.lang.String.indexOf(jint, jint)"
1787     end_locals
1788     end_states
1789     begin_HIR
1790 20 1 i23 -1 <|@
1791 .21 0 i24 ireturn i23 <|@
1792     end_HIR
1793     begin_LIR
1794 224 label [label:0xf801faf8] <|@
1795 226 move [int:-1|I] [rax|I] <|@
1796 228 return [rax|I] <|@
1797     end_LIR
1798     end_block
1799 end_cfg
1800 分配物理寄存器
1801 LIR生成后,因为LIR目前使用的是虚拟寄存器,所以需要通过线性扫描算法分配物理寄存器.
1802 Call stack to create register allocation:
1803
1804 int Compilation::compile_java_method()
1805 void Compilation::emit_lir()
1806     LinearScan::LinearScan(IR* ir, LIRGenerator* gen, FrameMap* frame_map)

```

```

1807         void LinearScan::do_linear_scan()
1808
1809         //代码在c1_LinearScan.cpp中.
1810         //方法先调用build_intervals()生成所有寄存器的活动区间,然后调用allocate_registers()为所以的虚拟寄存器分配物理寄存器.
1811         void LinearScan::do_linear_scan() {
1812             NOT_PRODUCT(_total_timer.begin_method());
1813
1814             number_instructions();
1815
1816             .....
1817             CHECK_BAILOUT();
1818
1819             build_intervals();
1820             CHECK_BAILOUT();
1821             sort_intervals_before_allocation();
1822
1823             NOT_PRODUCT(print_intervals("Before Register Allocation"));
1824             NOT_PRODUCT(LinearScanStatistic::compute(this, _stat_before_alloc));
1825
1826             allocate_registers();
1827             CHECK_BAILOUT();
1828
1829             resolve_data_flow();
1830             .....
1831         }
1832
1833         //循环每个block的每个instruction,生成每个寄存器的活动区间(interval)
1834         void LinearScan::build_intervals() {
1835             TIME_LINEAR_SCAN(timer_build_intervals);
1836
1837             .....
1838             LIR_OpVisitState visitor;
1839
1840             // iterate all blocks in reverse order
1841             for (i = block_count() - 1; i >= 0; i--) {
1842                 BlockBegin* block = block_at(i);
1843                 LIR_OpList* instructions = block->lir()->instructions_list();
1844                 int block_from = block->first_lir_instruction_id();
1845                 int block_to = block->last_lir_instruction_id();
1846
1847                 .....
1848
1849                 // iterate all instructions of the block in reverse order.
1850                 // skip the first instruction because it is always a label
1851                 // definitions of intervals are processed before uses
1852                 assert(visitor.no_operands(instructions->at(0)), "first operation must always be a label");
1853                 for (int j = instructions->length() - 1; j >= 1; j--) {
1854                     LIR_Op* op = instructions->at(j);
1855                     int op_id = op->id();
1856
1857                     .....
1858
1859                     // visit definitions (output and temp operands)
1860                     //如果此instruction有output,则生成寄存器使用定义,默认为当前instruction位置到+1.
1861                     int k, n;
1862                     n = visitor.opr_count(LIR_OpVisitState::outputMode);
1863                     for (k = 0; k < n; k++) {
1864                         LIR_Opr opr = visitor.opr_at(LIR_OpVisitState::outputMode, k);
1865                         assert(opr->is_register(), "visitor should only return register operands");
1866                         add_def(opr, op_id, use_kind_of_output_operand(op, opr));
1867                     }
1868
1869                     n = visitor.opr_count(LIR_OpVisitState::tempMode);
1870                     for (k = 0; k < n; k++) {
1871                         LIR_Opr opr = visitor.opr_at(LIR_OpVisitState::tempMode, k);
1872                         assert(opr->is_register(), "visitor should only return register operands");
1873                         add_temp(opr, op_id, mustHaveRegister);
1874                     }
1875
1876                     // visit uses (input operands)
1877                     //如果此instruction有input,则添加寄存器的使用区间,默认从block开始到当前instruction位置.
1878                     n = visitor.opr_count(LIR_OpVisitState::inputMode);
1879                     for (k = 0; k < n; k++) {
1880                         LIR_Opr opr = visitor.opr_at(LIR_OpVisitState::inputMode, k);
1881                         assert(opr->is_register(), "visitor should only return register operands");
1882                         add_use(opr, block_from, op_id, use_kind_of_input_operand(op, opr));
1883                     }
1884
1885                     // Add uses of live locals from interpreter's point of view for proper
1886                     // debug information generation
1887                     // Treat these operands as temp values (if the life range is extended
1888                     // to a call site, the value would be in a register at the call otherwise)
1889                     n = visitor.info_count();
1890                     for (k = 0; k < n; k++) {
1891                         CodeEmitInfo* info = visitor.info_at(k);
1892                         ValueStack* stack = info->stack();

```

```
1893     for_each_state_value(stack, value,
1894         add_use(value, block_from, op_id + 1, noUse);
1895     );
1896 }
1897
1898 // special steps for some instructions (especially moves)
1899 handle_method_arguments(op);
1900 handle_doubleword_moves(op);
1901 add_register_hints(op);
1902
1903 } // end of instruction iteration
1904 } // end of block iteration
1905
1906 .....
1907 }
1908 最后,列出寄存器分配前后的对比.
1909
1910 begin_intervals
1911     name "Before Register Allocation"
1912 //格式为:
1913 regNum type opr parentRegNum regHintRegNum (rngStart_rngEnd)+ (usePosAndKind)* spillState
1914 0 fixed "[rsi|I]" 0 177 [0, 4[ [214, 217[ "no spill store"
1915 1 fixed "[rdi|I]" 1 -1 [0, 1[ [216, 217[ "no definition"
1916 2 fixed "[rbx|I]" 2 -1 [0, 1[ [216, 217[ "no definition"
1917 3 fixed "[rax|I]" 3 196 [0, 1[ [194, 196[ [200, 202[ [216, 218[ [220, 222[ [226, 228[ "no optimization"
1918 4 fixed "[rdx|I]" 4 178 [0, 6[ [210, 217[ "no spill store"
1919 5 fixed "[rcx|I]" 5 189 [0, 8[ [212, 217[ "no spill store"
1920 6 fixed "[r8|I]" 6 -1 [0, 1[ [216, 217[ "no definition"
1921 7 fixed "[r9|I]" 7 -1 [0, 1[ [216, 217[ "no definition"
1922 8 fixed "[r11|I]" 8 -1 [0, 1[ [216, 217[ "no definition"
1923 9 fixed "[r13|I]" 9 -1 [0, 1[ [216, 217[ "no definition"
1924 10 fixed "[r14|I]" 10 -1 [0, 1[ [216, 217[ "no definition"
1925 177 object 177 0 [4, 108[ [204, 214[ 4 M 36 M 206 S 214 S "no spill store"
1926 178 int 178 4 [6, 192[ [204, 210[ 6 M 92 M 142 S 191 L 210 S "no spill store"
1927 179 int 179 5 [8, 56[ [66, 86[ 8 M 40 M 68 M 86 S "no spill store"
1928 180 *metadata* 180 -1 [10, 16[ 10 M 12 M 16 M "no spill store"
1929 181 int 181 -1 [12, 22[ 12 M 14 M 16 M 20 M 22 M "no optimization"
1930 182 *metadata* 182 -1 [18, 24[ 18 M 24 M "no spill store"
1931 183 object 183 -1 [36, 192[ 36 M 38 M 136 M 140 M 191 L "no spill store"
1932 184 int 184 -1 [38, 192[ 38 M 68 S 116 S 191 L "no spill store"
1933 185 *metadata* 185 -1 [42, 50[ 42 M 46 M 50 M "no spill store"
1934 186 long 186 -1 [44, 50[ 44 M 46 M 50 M "no spill store"
1935 187 long 187 -1 [46, 50[ 46 M 48 M 50 M "no spill store"
1936 188 *metadata* 188 -1 [58, 60[ 58 M 60 M "no spill store"
1937 189 int 189 179 [62, 66[ [86, 110[ [204, 212[ 62 M 86 M 110 S 212 S "no optimization"
1938 190 *metadata* 190 -1 [70, 78[ 70 M 74 M 78 M "no spill store"
1939 191 long 191 -1 [72, 78[ 72 M 74 M 78 M "no spill store"
1940 192 long 192 -1 [74, 78[ 74 M 76 M 78 M "no spill store"
1941 193 *metadata* 193 -1 [94, 102[ 94 M 98 M 102 M "no spill store"
1942 194 long 194 -1 [96, 102[ 96 M 98 M 102 M "no spill store"
1943 195 long 195 -1 [98, 102[ 98 M 100 M 102 M "no spill store"
1944 196 int 196 189 [110, 160[ [188, 194[ 110 M 116 M 134 M 136 M 138 M 160 S 188 M 191 L 194 S "no optimization"
1945 197 *metadata* 197 -1 [118, 126[ 118 M 122 M 126 M "no spill store"
1946 198 long 198 -1 [120, 126[ 120 M 122 M 126 M "no spill store"
1947 199 long 199 -1 [122, 126[ 122 M 124 M 126 M "no spill store"
1948 200 long 200 196 [134, 140[ 134 M 140 M "no spill store"
1949 201 int 201 -1 [140, 142[ 140 M 142 M "no spill store"
1950 202 *metadata* 202 -1 [144, 152[ 144 M 148 M 152 M "no spill store"
1951 203 long 203 -1 [146, 152[ 146 M 148 M 152 M "no spill store"
1952 204 long 204 -1 [148, 152[ 148 M 150 M 152 M "no spill store"
1953 205 int 205 196 [160, 188[ 160 M 162 M 188 S "no spill store"
1954 206 *metadata* 206 -1 [164, 170[ 164 M 166 M 170 M "no spill store"
1955 207 int 207 -1 [166, 176[ 166 M 168 M 170 M 174 M 176 M "no optimization"
1956 208 *metadata* 208 -1 [172, 178[ 172 M 178 M "no spill store"
1957 209 *metadata* 209 -1 [184, 186[ 184 M 186 M "no spill store"
1958 210 object 210 -1 [208, 209[ 208 M "no definition"
1959 211 long 211 -1 [208, 209[ 208 M "no definition"
1960 212 object 212 177 [206, 207[ [208, 209[ 206 M 208 M "no spill store"
1961 213 int 213 3 [218, 220[ 218 M 220 S "no spill store"
1962 end_intervals
1963 begin_intervals
1964     name "After Register Allocation"
1965 0 fixed "[rsi|I]" 0 177 [0, 4[ [214, 217[ "no spill store"
1966 1 fixed "[rdi|I]" 1 -1 [0, 1[ [216, 217[ "no definition"
1967 2 fixed "[rbx|I]" 2 -1 [0, 1[ [216, 217[ "no definition"
1968 3 fixed "[rax|I]" 3 196 [0, 1[ [194, 196[ [200, 202[ [216, 218[ [220, 222[ [226, 228[ "no optimization"
1969 4 fixed "[rdx|I]" 4 178 [0, 6[ [210, 217[ "no spill store"
1970 5 fixed "[rcx|I]" 5 189 [0, 8[ [212, 217[ "no spill store"
1971 6 fixed "[r8|I]" 6 -1 [0, 1[ [216, 217[ "no definition"
1972 7 fixed "[r9|I]" 7 -1 [0, 1[ [216, 217[ "no definition"
1973 8 fixed "[r11|I]" 8 -1 [0, 1[ [216, 217[ "no definition"
1974 9 fixed "[r13|I]" 9 -1 [0, 1[ [216, 217[ "no definition"
1975 10 fixed "[r14|I]" 10 -1 [0, 1[ [216, 217[ "no definition"
1976 //寄存器177分配后,使用物理寄存器rsi.
1977 177 object "[rsi|L]" 177 0 [4, 108[ [204, 214[ 4 M 36 M 206 S 214 S "no spill store"
1978 178 int "[rdx|I]" 178 4 [6, 192[ [204, 210[ 6 M 92 M 142 S 191 L 210 S "no spill store"
```

```

1799 179 int "[rcx|I]" 179 5 [8, 56[ [66, 86[ 8 M 40 M 68 M 86 S "no spill store"
1800 180 *metadata* "[rax|M]" 180 -1 [10, 16[ 10 M 12 M 16 M "no spill store"
1801 181 int "[rdi|I]" 181 -1 [12, 22[ 12 M 14 M 16 M 20 M 22 M "no optimization"
1802 182 *metadata* "[rax|M]" 182 -1 [18, 24[ 18 M 24 M "no spill store"
1803 183 object "[rax|L]" 183 -1 [36, 192[ 36 M 38 M 136 M 140 M 191 L "no spill store"
1804 184 int "[rdi|I]" 184 -1 [38, 192[ 38 M 68 S 116 S 191 L "no spill store"
1805 185 *metadata* "[rbx|M]" 185 -1 [42, 50[ 42 M 46 M 50 M "no spill store"
1806 186 long "[r8r8|J]" 186 -1 [44, 50[ 44 M 46 M 50 M "no spill store"
1807 187 long "[r9r9|J]" 187 -1 [46, 50[ 46 M 48 M 50 M "no spill store"
1808 188 *metadata* "[rcx|M]" 188 -1 [58, 60[ 58 M 60 M "no spill store"
1809 189 int "[rcx|I]" 189 179 [62, 66[ [86, 110[ [204, 212[ 62 M 86 M 110 S 212 S "no optimization"
1810 190 *metadata* "[rbx|M]" 190 -1 [70, 78[ 70 M 74 M 78 M "no spill store"
1811 191 long "[r8r8|J]" 191 -1 [72, 78[ 72 M 74 M 78 M "no spill store"
1812 192 long "[r9r9|J]" 192 -1 [74, 78[ 74 M 76 M 78 M "no spill store"
1813 193 *metadata* "[rbx|M]" 193 -1 [94, 102[ 94 M 98 M 102 M "no spill store"
1814 194 long "[r8r8|J]" 194 -1 [96, 102[ 96 M 98 M 102 M "no spill store"
1815 195 long "[r9r9|J]" 195 -1 [98, 102[ 98 M 100 M 102 M "no spill store"
1816 196 int "[rcx|I]" 196 189 [110, 160[ [188, 194[ 110 M 116 M 134 M 136 M 138 M 160 S 188 M 191 L 194 S "no optimization"
1817 197 *metadata* "[rsi|M]" 197 -1 [118, 126[ 118 M 122 M 126 M "no spill store"
1818 198 long "[rbxrbx|J]" 198 -1 [120, 126[ 120 M 122 M 126 M "no spill store"
1819 199 long "[r8r8|J]" 199 -1 [122, 126[ 122 M 124 M 126 M "no spill store"
2000 200 long "[rsinsi|J]" 200 196 [134, 140[ 134 M 140 M "no spill store"
2001 201 int "[rsi|I]" 201 -1 [140, 142[ 140 M 142 M "no spill store"
2002 202 *metadata* "[rsi|M]" 202 -1 [144, 152[ 144 M 148 M 152 M "no spill store"
2003 203 long "[rbxrbx|J]" 203 -1 [146, 152[ 146 M 148 M 152 M "no spill store"
2004 204 long "[r8r8|J]" 204 -1 [148, 152[ 148 M 150 M 152 M "no spill store"
2005 205 int "[rcx|I]" 205 196 [160, 188[ 160 M 162 M 188 S "no spill store"
2006 206 *metadata* "[rsi|M]" 206 -1 [164, 170[ 164 M 166 M 170 M "no spill store"
2007 207 int "[rbx|I]" 207 -1 [166, 176[ 166 M 168 M 170 M 174 M 176 M "no optimization"
2008 208 *metadata* "[rsi|M]" 208 -1 [172, 178[ 172 M 178 M "no spill store"
2009 209 *metadata* "[rsi|M]" 209 -1 [184, 186[ 184 M 186 M "no spill store"
2010 210 object "[rbx|L]" 210 -1 [208, 209[ 208 M "no definition"
2011 211 long "[raxrax|J]" 211 -1 [208, 209[ 208 M "no definition"
2012 212 object "[rdi|L]" 212 177 [206, 207[ [208, 209[ 206 M 208 M "no spill store"
2013 213 int "[rax|I]" 213 3 [218, 220[ 218 M 220 S "no spill store"
2014 end_intervals
2015 这个例子的物理寄存器刚好都够用,不需要split interval.下面给一个需要分割区间的例子.
2016 虚拟寄存器194的120-250区间就无法找到一个完整的物理寄存器.
2017 寄存器分配完后,区间分配为120-190, 190-201, 201-230, 230-250.
2018 其中,190-201, 201-230使用了栈栈空间,230以后又回到了rdi寄存器的使用.
2019 总之,算法是尽可能使用物理寄存器来加速程序执行.
2020
2021 begin_intervals
2022   name "Before Register Allocation"
2023   0 fixed "[rsi|I]" 0 179 [0, 4[ [200, 203[ [356, 359[ [466, 469[ "no optimization"
2024   1 fixed "[rdi|I]" 1 -1 [0, 1[ [202, 203[ [358, 359[ [468, 469[ "no definition"
2025   2 fixed "[rbx|I]" 2 -1 [0, 1[ [202, 203[ [358, 359[ [468, 469[ "no definition"
2026   3 fixed "[rax|I]" 3 227 [0, 1[ [202, 204[ [358, 360[ [442, 444[ [448, 450[ [468, 470[ [472, 474[ [478, 480[ "no optimization"
2027   4 fixed "[rdx|I]" 4 202 [0, 6[ [198, 203[ [354, 359[ [462, 469[ "no optimization"
2028   5 fixed "[rcx|I]" 5 179 [0, 8[ [202, 203[ [358, 359[ [464, 469[ "no spill store"
2029   6 fixed "[r8|I]" 6 -1 [0, 1[ [202, 203[ [358, 359[ [468, 469[ "no definition"
2030   7 fixed "[r9|I]" 7 -1 [0, 1[ [202, 203[ [358, 359[ [468, 469[ "no definition"
2031   8 fixed "[r11|I]" 8 -1 [0, 1[ [202, 203[ [358, 359[ [468, 469[ "no definition"
2032   9 fixed "[r13|I]" 9 -1 [0, 1[ [202, 203[ [358, 359[ [468, 469[ "no definition"
2033   10 fixed "[r14|I]" 10 -1 [0, 1[ [202, 203[ [358, 359[ [468, 469[ "no definition"
2034   .....
2035   194 object 194 -1 [98, 116[ [120, 250[ [254, 274[ [452, 454[ [476, 478[ 98 M 100 M 122 M 146 M 232 M 256 M 454 M 478 S "no spill store"
2036   .....
2037 end_intervals
2038 begin_intervals
2039   name "After Register Allocation"
2040   0 fixed "[rsi|I]" 0 179 [0, 4[ [200, 203[ [356, 359[ [466, 469[ "no optimization"
2041   1 fixed "[rdi|I]" 1 -1 [0, 1[ [202, 203[ [358, 359[ [468, 469[ "no definition"
2042   2 fixed "[rbx|I]" 2 -1 [0, 1[ [202, 203[ [358, 359[ [468, 469[ "no definition"
2043   3 fixed "[rax|I]" 3 227 [0, 1[ [202, 204[ [358, 360[ [442, 444[ [448, 450[ [468, 470[ [472, 474[ [478, 480[ "no optimization"
2044   4 fixed "[rdx|I]" 4 202 [0, 6[ [198, 203[ [354, 359[ [462, 469[ "no optimization"
2045   5 fixed "[rcx|I]" 5 179 [0, 8[ [202, 203[ [358, 359[ [464, 469[ "no spill store"
2046   6 fixed "[r8|I]" 6 -1 [0, 1[ [202, 203[ [358, 359[ [468, 469[ "no definition"
2047   7 fixed "[r9|I]" 7 -1 [0, 1[ [202, 203[ [358, 359[ [468, 469[ "no definition"
2048   8 fixed "[r11|I]" 8 -1 [0, 1[ [202, 203[ [358, 359[ [468, 469[ "no definition"
2049   9 fixed "[r13|I]" 9 -1 [0, 1[ [202, 203[ [358, 359[ [468, 469[ "no definition"
2050   10 fixed "[r14|I]" 10 -1 [0, 1[ [202, 203[ [358, 359[ [468, 469[ "no definition"
2051   .....
2052   194 object "[rdi|L]" 194 -1 [98, 116[ [120, 190[ 98 M 100 M 122 M 146 M "one spill store"
2053   .....
2054   263 object "[stack:3|L]" 194 194 [190, 201[ "one spill store"
2055   264 object "[stack:3|L]" 194 194 [201, 230[ "one spill store"
2056   .....
2057   267 object "[rdi|L]" 194 194 [230, 250[ [254, 274[ [452, 454[ [476, 478[ 232 M 256 M 454 M 478 S "one spill store"
2058   .....
2059 end_intervals
2060 生成代码
2061 最后,LIR分配物理寄存器后如下:
2062
2063 begin_cfg
2064   name "Before Code Generation"

```



```
2065 begin_block
2066 name "B12"
2067 from_bci 0
2068 to_bci 0
2069 predecessors
2070 successors "B13"
2071 xhandlers
2072 flags
2073 first_lir_id 0
2074 last_lir_id 28
2075 begin_LIR
2076 0 label [label:0xfc0043b8] <|@
2077 2 std_entry <|@
2078 10 move [metadata:0x3301c100|M] [rax|M] <|@
2079 12 move [Base:[rax|M] Disp: 108|I] [rdi|I] <|@
2080 14 add [rdi|I] [int:8|I] [rdi|I] <|@
2081 16 move [rdi|I] [Base:[rax|M] Disp: 108|I] <|@
2082 18 move [metadata:0x32dd2238|M] [rax|M] <|@
2083 20 logic_and [rdi|I] [int:8184|I] [rdi|I] <|@
2084 22 cmp [rdi|I] [int:0|I] <|@
2085 24 branch [EQ] [CounterOverflowStub: 0xf8025908] <|@
2086 26 label [label:0xf8025930] <|@
2087 end_LIR
2088 end_block
2089 begin_block
2090 name "B13"
2091 from_bci 0
2092 to_bci 0
2093 predecessors "B12"
2094 successors "B0"
2095 xhandlers
2096 flags "std"
2097 dominator "B12"
2098 first_lir_id 30
2099 last_lir_id 32
2100 begin_LIR
2101 30 label [label:0xfc0046c8] <|@
2102 end_LIR
2103 end_block
2104 begin_block
2105 name "B0"
2106 from_bci 0
2107 to_bci 7
2108 predecessors "B13"
2109 successors "B2" "B1"
2110 xhandlers
2111 flags "std"
2112 dominator "B13"
2113 first_lir_id 34
2114 last_lir_id 54
2115 begin_LIR
2116 34 label [label:0xf801ee78] <|@
2117 36 move [Base:[rsi|L] Disp: 12|L] [rax|L] <|@
2118 38 move [Base:[rax|L] Disp: 12|I] [rdi|I] [bci:4] <|@
2119 40 cmp [rcx|I] [int:0|I] <|@
2120 42 move [metadata:0x3301c100|M] [rbx|M] <|@
2121 44 cmove [GE] [lng:152|J] [lng:168|J] [r8r8|J] <|@
2122 46 move [Base:[rbx|M] Index:[r8r8|J] Disp: 0|J] [r9r9|J] <|@
2123 48 leal [Base:[r9r9|J] Disp: 1|I] [r9r9|J] <|@
2124 50 move [r9r9|J] [Base:[rbx|M] Index:[r8r8|J] Disp: 0|J] <|@
2125 52 branch [GE] [B2] <|@
2126 end_LIR
2127 end_block
2128 begin_block
2129 name "B1"
2130 from_bci 10
2131 to_bci 12
2132 predecessors "B0"
2133 successors "B3"
2134 xhandlers
2135 flags
2136 dominator "B0"
2137 first_lir_id 56
2138 last_lir_id 64
2139 begin_LIR
2140 56 label [label:0xf801f198] <|@
2141 58 move [metadata:0x3301c100|M] [rcx|M] <|@
2142 60 add [Base:[rcx|M] Disp: 184|J] [int:1|I] [Base:[rcx|M] Disp: 184|J] <|@
2143 62 move [int:0|I] [rcx|I] <|@
2144 64 branch [AL] [B3] <|@
2145 end_LIR
2146 end_block
2147 begin_block
2148 name "B2"
2149 from_bci 15
2150 to_bci 17
```

```

2151     predecessors "B0"
2152     successors "B3" "B4"
2153     xhandlers
2154     flags
2155     dominator "B0"
2156     first_lir_id 66
2157     last_lir_id 82
2158     begin_LIR
2159     66 label [label:0xf801f4b8] <|@
2160     68 cmp [rcx|I] [rdi|I] <|@
2161     70 move [metadata:0x3301c100|M] [rbx|M] <|@
2162     72 cmove [LT] [lng:208|J] [lng:224|J] [r8r8|J] <|@
2163     74 move [Base:[rbx|M] Index:[r8r8|J] Disp: 0|J] [r9r9|J] <|@
2164     76 leal [Base:[r9r9|J] Disp: 1|I] [r9r9|J] <|@
2165     78 move [r9r9|J] [Base:[rbx|M] Index:[r8r8|J] Disp: 0|J] <|@
2166     80 branch [GE] [B4] <|@
2167     end_LIR
2168     end_block
2169     begin_block
2170     name "B3"
2171     from_bci 22
2172     to_bci 25
2173     predecessors "B1" "B2"
2174     successors "B6" "B7"
2175     xhandlers
2176     flags
2177     dominator "B0"
2178     first_lir_id 90
2179     last_lir_id 106
2180     begin_LIR
2181     90 label [label:0xf801f7d8] <|@
2182     92 cmp [rdx|I] [int:65536|I] <|@
2183     94 move [metadata:0x3301c100|M] [rbx|M] <|@
2184     96 cmove [GE] [lng:240|J] [lng:256|J] [r8r8|J] <|@
2185     98 move [Base:[rbx|M] Index:[r8r8|J] Disp: 0|J] [r9r9|J] <|@
2186     100 leal [Base:[r9r9|J] Disp: 1|I] [r9r9|J] <|@
2187     102 move [r9r9|J] [Base:[rbx|M] Index:[r8r8|J] Disp: 0|J] <|@
2188     104 branch [GE] [B6] <|@
2189     106 branch [AL] [B7] <|@
2190     end_LIR
2191     end_block
2192     begin_block
2193     name "B8"
2194     from_bci 43
2195     to_bci 49
2196     predecessors "B7"
2197     successors "B11" "B10"
2198     xhandlers
2199     flags "bb"
2200     dominator "B7"
2201     loop_index 0
2202     loop_depth 1
2203     first_lir_id 132
2204     last_lir_id 156
2205     begin_LIR
2206     132 label [label:0xf8020778] <|@
2207     134 convert [i2l] [rcx|I] [rsirsi|J] <|@
2208     136 cmp [rcx|I] [Base:[rax|L] Disp: 12|I] <|@
2209     138 branch [AE] [RangeCheckStub: 0xf80296d8] [bci:47] <|@
2210     140 move [Base:[rax|L] Index:[rsirsi|J] * 2 Disp: 16|C] [rsi|I] <|@
2211     142 cmp [rsi|I] [rdx|I] <|@
2212     144 move [metadata:0x3301c100|M] [rsi|M] <|@
2213     146 cmove [NE] [lng:304|J] [lng:320|J] [rbxrbx|J] <|@
2214     148 move [Base:[rsi|M] Index:[rbxrbx|J] Disp: 0|J] [r8r8|J] <|@
2215     150 leal [Base:[r8r8|J] Disp: 1|I] [r8r8|J] <|@
2216     152 move [r8r8|J] [Base:[rsi|M] Index:[rbxrbx|J] Disp: 0|J] <|@
2217     154 branch [EQ] [B10] <|@
2218     end_LIR
2219     end_block
2220     begin_block
2221     name "B11"
2222     from_bci 55
2223     to_bci 58
2224     predecessors "B8"
2225     successors "B7"
2226     xhandlers
2227     flags "lle"
2228     dominator "B8"
2229     loop_index 0
2230     loop_depth 1
2231     first_lir_id 158
2232     last_lir_id 190
2233     begin_LIR
2234     158 label [label:0xf80210d8] <|@
2235     162 add [rcx|I] [int:1|I] [rcx|I] <|@
2236     164 move [metadata:0x3301c100|M] [rsi|M] <|@

```

```

2237 166 move [Base:[rsi|M] Disp: 112|I] [rbx|I] <|@
2238 168 add [rbx|I] [int:8|I] [rbx|I] <|@
2239 170 move [rbx|I] [Base:[rsi|M] Disp: 112|I] <|@
2240 172 move [metadata:0x32dd2238|M] [rsi|M] <|@
2241 174 logic_and [rbx|I] [int:65528|I] [rbx|I] <|@
2242 176 cmp [rbx|I] [int:0|I] <|@
2243 178 branch [EQ] [CounterOverflowStub: 0xf802a488] <|@
2244 180 label [label:0xf802a4b0] <|@
2245 182 safepoint [bci:58] <|@
2246 184 move [metadata:0x3301c100|M] [rsi|M] <|@
2247 186 add [Base:[rsi|M] Disp: 336|J] [int:1|I] [Base:[rsi|M] Disp: 336|J] <|@
2248     end_LIR
2249     end_block
2250     begin_block
2251         name "B7"
2252         from_bci 37
2253         to_bci 40
2254         predecessors "B11" "B3"
2255         successors "B9" "B8"
2256         xhandlers
2257         flags "plh" "llh"
2258         dominator "B5"
2259         loop_index 0
2260         loop_depth 1
2261         first_lir_id 114
2262         last_lir_id 130
2263         begin_LIR
2264         114 label [label:0xf8020458] <|@
2265         116 cmp [rcx|I] [rdi|I] <|@
2266         118 move [metadata:0x3301c100|M] [rsi|M] <|@
2267         120 cmove [GE] [lng:272|J] [lng:288|J] [rbxrbx|J] <|@
2268         122 move [Base:[rsi|M] Index:[rbxrbx|J] Disp: 0|J] [r8r8|J] <|@
2269         124 leal [Base:[r8r8|J] Disp: 1|I] [r8r8|J] <|@
2270         126 move [r8r8|J] [Base:[rsi|M] Index:[rbxrbx|J] Disp: 0|J] <|@
2271         128 branch [GE] [B9] <|@
2272         130 branch [AL] [B8] <|@
2273         end_LIR
2274         end_block
2275         begin_block
2276             name "B10"
2277             from_bci 52
2278             to_bci 54
2279             predecessors "B8"
2280             successors
2281             xhandlers
2282             flags
2283             dominator "B8"
2284             first_lir_id 192
2285             last_lir_id 196
2286             begin_LIR
2287             192 label [label:0xf8020db8] <|@
2288             194 move [rcx|I] [rax|I] <|@
2289             196 return [rax|I] <|@
2290             end_LIR
2291             end_block
2292             begin_block
2293                 name "B9"
2294                 from_bci 61
2295                 to_bci 62
2296                 predecessors "B7"
2297                 successors
2298                 xhandlers
2299                 flags
2300                 dominator "B7"
2301                 first_lir_id 198
2302                 last_lir_id 202
2303                 begin_LIR
2304                 198 label [label:0xf8020a98] <|@
2305                 200 move [int:-1|I] [rax|I] <|@
2306                 202 return [rax|I] <|@
2307                 end_LIR
2308                 end_block
2309                 begin_block
2310                     name "B6"
2311                     from_bci 63
2312                     to_bci 69
2313                     predecessors "B3"
2314                     successors
2315                     xhandlers
2316                     flags
2317                     dominator "B3"
2318                     first_lir_id 204
2319                     last_lir_id 222
2320                     begin_LIR
2321                     204 label [label:0xf8020138] <|@
2322                     206 move [rsi|L] [rdi|L] <|@

```

```

2323 208 profile_call indexOf.java/lang/String @ 66 [rbx|L] [rdi|L] [raxrax|J] <|@
2324 216 optvirtual call: [addr: 0x0] [recv: [rsi|L]] [result: [rax|I]] [bci:66] <|@
2325 222 return [rax|I] <|@
2326     end_LIR
2327 end_block
2328 begin_block
2329     name "B4"
2330     from_bci 20
2331     to_bci 21
2332     predecessors "B2"
2333     successors
2334     xhandlers
2335     flags
2336     dominator "B2"
2337     first_lir_id 224
2338     last_lir_id 228
2339     begin_LIR
2340 224 label [label:0xf801faf8] <|@
2341 226 move [int:-1|I] [rax|I] <|@
2342 228 return [rax|I] <|@
2343     end_LIR
2344 end_block
2345 end_cfg
2346 发射汇编代码的过程如:
2347
2348 int Compilation::compile_java_method()
2349     int Compilation::emit_code_body()
2350         void LIR_Assembler::emit_code(BlockList* hir)
2351             void LIR_Assembler::emit_block(BlockBegin* block)
2352                 void LIR_Assembler::emit_lir_list(LIR_List* list)
2353                     virtual void emit_code(LIR_Assembler* masm)
2354                         void LIR_Op1::emit_code(LIR_Assembler* masm) //比如前面的lir_move
2355
2356 void LIR_Assembler::emit_op1(LIR_Op1* op) {
2357     switch (op->code()) {
2358     case lir_move:
2359         if (op->move_kind() == lir_move_volatile) {
2360             assert(op->patch_code() == lir_patch_none, "can't patch volatiles");
2361             volatile_move_op(op->in_opr(), op->result_opr(), op->type(), op->info());
2362         } else {
2363             move_op(op->in_opr(), op->result_opr(), op->type(),
2364                 op->patch_code(), op->info(), op->pop_fpu_stack(),
2365                 op->move_kind() == lir_move_unaligned,
2366                 op->move_kind() == lir_move_wide);
2367         }
2368     break;
2369     .....
2370 }
2371
2372 void LIR_Assembler::move_op(LIR_Opr src, LIR_Opr dest, BasicType type, LIR_PatchCode patch_code, CodeEmitInfo* info, bool pop_fpu_stack, bool
2373     if (src->is_register()) {
2374         if (dest->is_register()) {
2375             assert(patch_code == lir_patch_none && info == NULL, "no patching and info allowed here");
2376             reg2reg(src, dest);
2377         } else if (dest->is_stack()) {
2378             .....
2379         }
2380
2381 void LIR_Assembler::reg2reg(LIR_Opr src, LIR_Opr dest) {
2382     .....
2383     // move between cpu-registers
2384     if (dest->is_single_cpu()) {
2385         .....
2386         move_regs(src->as_register(), dest->as_register());
2387     } else if (dest->is_double_cpu()) {
2388         .....
2389     }
2390
2391 void LIR_Assembler::move_regs(Register from_reg, Register to_reg) {
2392     if (from_reg != to_reg) __ mov(to_reg, from_reg);
2393 }
2394
2395 //Code in assembler_x86.cpp
2396 void Assembler::mov(Register dst, Register src) {
2397     LP64_ONLY(movq(dst, src)) NOT_LP64(movl(dst, src));
2398 }
2399
2400

```