

<https://github.com/lazyparser/v8-internals>

V8 Internals: Porting to RISC-V

Build & Run, Ignition Bytecodes

Wei Wu | @lazyparser | lazyparser@gmail.com

2020-01-16

V8 是什么，现在发展到什么程度

- 介绍略，自己查
- 注意，V8 的架构变更过多次，看演讲视频学习的时候要注意演讲时间
- 目前是 Ignition + TurboFan 架构
 - 还有一堆的模块没有提到，跟我们做 porting 主要相关的是解释器和JIT
 - GC、Runtime 中的函数中是否存在 Porting 的工作量目前未知，TBD

项目的目标，以及我们这门课程的目标

- 项目目标：给V8添加一个RISC-V后端，可以在RV64Linux上跑

项目的目标，以及我们这门课程的目标

- 项目目标：给V8添加一个RISC-V后端，可以在RV64Linux上跑
- 课程目标：完成项目成员的新人培训、做成慕课用于以后开放培训

项目的目标，以及我们这门课程的目标

- 项目目标：给V8添加一个RISC-V后端，可以在RV64Linux上跑
- 课程目标：完成项目成员的新人培训、做成慕课用于以后开放培训
- 配套内容：教材《V8 Internals》、视频课程集、课程设计代码库和书

本次课程需要自学的视频内容

- Ignition - an interpreter for V8 [av83456458](#)
- TurboFan: A new code generation architecture for V8  [v=M1FBosB5tjM](#)
- Benedikt Meurer: A Tale of TurboFan: Four years that changed V8 forever [av83595492](#)
- Orinoco: The new V8 Garbage Collector Peter Marshall [av83595696](#)
- Sigurd Schneider- Inside V8- The choreography of Ignition and TurboFan - [av83277184](#)
- Mathias Bynens - V8 internals for JavaScript developers [av83563627](#)
- Lars Bak 最早介绍V8的视频 [av83456867](#)

本次课程需要完成的课后阅读

- <https://v8.dev/docs/build> 第一步：构建
- <https://v8.dev/docs/d8> 以后每天打交道的shell
- <https://v8.dev/blog/ignition-interpreter> 必读
- [docs.google](https://docs.google.com/presentation/d/1317d46c94775) DLS Keynote: Ignition: Jump-starting an Interpreter for V8
- 317d46c94775 Ignition的部分解读（看英文）
- [docs.google](https://docs.google.com/presentation/d/1317d46c94775) Ignition Design Doc
- <https://doar-e.github.io/blog/2019/01/28/introduction-to-turbofan/> 可以作为入门

本次课程有空完成的扩展课后阅读

- <https://v8.dev/docs/gdb> 没用过gdb的，学
- <https://v8.dev/blog/ignition-interpreter> 必读
- <https://docs.google> Ignition: Register Equivalence Optimization 本次不要求
- <https://v8.dev/docs/turbofan> 可以预习，后续主要开发都是在 turbofan 上
- <https://v8.dev/docs/trace> 有兴趣看看，本次不要求。
- <https://darksi.de/d.sea-of-nodes/> turbofan相关

接下来开始看解释器

前提条件：构建完成V8，可以进入 d8；看过架构介绍；看过字节码文档

解释器 (Ignition) 的字节码设计

- 不是基于栈的，是基于「寄存器」的设计的，但是有特殊设计

解释器 (Ignition) 的字节码设计

- 不是基于栈的，是基于「寄存器」的设计的，但是有特殊设计
 - 设定了一个累加器寄存器 (accumulator)
 - 寄存器对应寄存器文件 (register file) 是每个指令 (stmt) 对应一个

解释器 (Ignition) 的字节码设计

- 不是基于栈的，是基于「寄存器」的设计的，但是有特殊设计
 - 设定了一个累加器寄存器 (accumulator)
 - 寄存器对应寄存器文件 (register file) 是每个指令 (stmt) 对应一个
- 每个字节码 BARABARA 对应一个 DoBARABARA 函数 (handler)

解释器 (Ignition) 的字节码设计

- 不是基于栈的，是基于「寄存器」的设计的，但是有特殊设计
 - 设定了一个累加器寄存器 (accumulator)
 - 寄存器对应寄存器文件 (register file) 是每个指令 (stmt) 对应一个
- 每个字节码 BARABARA 对应一个 DoBARABARA 函数 (handler)
 - 注意不一定能直接搜索到函数定义，有不少是通过宏生成的

看代码之前要知道大概的架构逻辑

- Bytecodes 需要有定义和字典

看代码之前要知道大概的架构逻辑

- Bytecodes 需要有定义和字典
 - 定义文档 Ignition Design Doc 的 rel32Appendix A
 - v8/src/interpreter/bytecodes.h /* good in-comment-doc */

看代码之前要知道大概的架构逻辑

- Bytecodes 需要有定义和字典
 - 定义文档 Ignition Design Doc 的 rel32Appendix A
 - v8/src/interpreter/bytecodes.h /* good in-comment-doc */
- 在整体执行的流水线上的位置

看代码之前要知道大概的架构逻辑

- Bytecodes 需要有定义和字典
 - 定义文档 Ignition Design Doc 的 rel32Appendix A
 - v8/src/interpreter/bytecodes.h /* good in-comment-doc */
- 在整体执行的流水线上的位置
 - 谁创建的字节码；什么被转成了字节码；字节码是谁在执行
 - 跟TurboFan如何交互；跟GC如何交互；跟Runtime函数如何交互

rel32Appendix A: Table of Bytecodes

Prefix Bytecodes

Wide

ExtraWide

Accumulator Load

LdaZero

LdaSmi

LdaUndefined

LdaNull

LdaTheHole

LdaTrue

LdaFalse

LdaConstant

Load/Store Globals

LdaGlobal

LdaGlobalInsideTypeof

Load-Store lookup slots

LdaLookupSlot

LdaLookupSlotInsideTypeof

StaLookupSlotSloppy

StaLookupSlotStrict

Register Transfers

Ldar

Mov

Star

LoadIC operations

LoadIC

KeyedLoadIC

StoreIC operations

StoreICSloppy

StoreICStrict

Calls

Call

TailCall

CallRuntime

CallRuntimeForPair

CallJSRuntime

Intrinsics

InvokeIntrinsic

New operator

New

Test Operators

TestEqual

TestNotEqual

TestEqualStrict

TestNotEqualStrict

```

3 // found in the LICENSE file.
4
5 #ifndef V8_INTERPRETER_BYTECODES_H_
6 #define V8_INTERPRETER_BYTECODES_H_
7
8 #include <cstdint>
9 #include <iostream>
10 #include <string>
11 #include <vector>
12
13 #include "src/common/globals.h"
14 #include "src/interpreter/bytecode-operands.h"
15
16 // This interface and it's implementation are independent of the
17 // libv8_base library as they are used by the interpreter and the
18 // standalone mkpeephole table generator program.
19
20 namespace v8 {
21 namespace internal {
22 namespace interpreter {
23
24 // The list of bytecodes which are interpreted by the interpreter.
25 // Format is V(<bytecode>, <accumulator_use>, <operands>).
26 #define BYTECODE_LIST(V)
27 /* Extended width operands */
28 V(CWide, AccumulatorUse::kNone)
29 V(ExtraWide, AccumulatorUse::kNone)
30
31 /* Debug Breakpoints - one for each possible size of unscaled bytecodes */
32 /* and one for each operand widening prefix bytecode */
33 V(DebugBreakWide, AccumulatorUse::kReadWrite)
34 V(DebugBreakExtraWide, AccumulatorUse::kReadWrite)
35 V(DebugBreak0, AccumulatorUse::kReadWrite)
36 V(DebugBreak1, AccumulatorUse::kReadWrite, OperandType::kReg)
37 V(DebugBreak2, AccumulatorUse::kReadWrite, OperandType::kReg,
38 OperandType::kReg)
39 V(DebugBreak3, AccumulatorUse::kReadWrite, OperandType::kReg,
40 OperandType::kReg, OperandType::kReg)
41 V(DebugBreak4, AccumulatorUse::kReadWrite, OperandType::kReg,
42 OperandType::kReg, OperandType::kReg, OperandType::kReg)
43 V(DebugBreak5, AccumulatorUse::kReadWrite, OperandType::kRuntimeId,
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86 V(LdaLookupGlobalSlotInsideTypeof, AccumulatorUse::kWrite,
87 OperandType::kIdx, OperandType::kIdx, OperandType::kUIImm)
88 V(StaLookupSlot, AccumulatorUse::kReadWrite, OperandType::kIdx,
89 OperandType::kFlag8)
90
91 /* Register-accumulator transfers */
92 V(Ldar, AccumulatorUse::kWrite, OperandType::kReg)
93 V(Star, AccumulatorUse::kRead, OperandType::kReg0ut)
94
95 /* Register-register transfers */
96 V(Mov, AccumulatorUse::kNone, OperandType::kReg, OperandType::kReg0ut)
97
98 /* Property loads (LoadIC) operations */
99 V(LdaNamedProperty, AccumulatorUse::kWrite, OperandType::kReg,
100 OperandType::kIdx, OperandType::kIdx)
101 V(LdaNamedPropertyNoFeedback, AccumulatorUse::kWrite, OperandType::kReg,
102 OperandType::kIdx)
103 V(LdaKeyedProperty, AccumulatorUse::kReadWrite, OperandType::kReg,
104 OperandType::kIdx)
105
106 /* Operations on module variables */
107 V(LdaModuleVariable, AccumulatorUse::kWrite, OperandType::kImm,
108 OperandType::kUIImm)
109 V(StaModuleVariable, AccumulatorUse::kRead, OperandType::kImm,
110 OperandType::kUIImm)
111
112 /* Property stores (StoreIC) operations */
113 V(StaNamedProperty, AccumulatorUse::kReadWrite, OperandType::kReg,
114 OperandType::kIdx, OperandType::kIdx)
115 V(StaNamedPropertyNoFeedback, AccumulatorUse::kReadWrite, OperandType::kReg,
116 OperandType::kIdx, OperandType::kFlag8)
117 V(StaNamedOwnProperty, AccumulatorUse::kReadWrite, OperandType::kReg,
118 OperandType::kIdx, OperandType::kIdx)
119 V(StaKeyedProperty, AccumulatorUse::kReadWrite, OperandType::kReg,
120 OperandType::kReg, OperandType::kIdx)
121 V(StaInArrayLiteral, AccumulatorUse::kReadWrite, OperandType::kReg,
122 OperandType::kReg, OperandType::kIdx)
123 V(StaDataPropertyInLiteral, AccumulatorUse::kRead, OperandType::kReg,
124 OperandType::kReg, OperandType::kFlag8, OperandType::kIdx)
125 V(CollectTypeProfile, AccumulatorUse::kRead, OperandType::kImm)
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
288
289
289
290
291
292
293
294
295
296
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
318
319
319
320
321
322
323
324
325
326
327
328
329
329
330
331
332
333
334
335
336
337
338
339
339
340
341
342
343
344
345
346
347
348
349
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
368
369
369
370
371
372
373
374
375
376
377
378
379
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
748
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
788
789
789
790
791
792
793
794
795
796
797
797
798
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
888
889
889
890
891
892
893
894
895
896
897
897
898
899
900

```

```
23
24 // The list of bytecodes which are interpreted by the interpreter.
25 // Format is V(<bytecode>, <accumulator_use>, <operands>).
26 #define BYTECODE_LIST(V) \
27     /* Extended width operands */ \
28     V(Wide, AccumulatorUse::kNone) \
29     V(ExtraWide, AccumulatorUse::kNone) \
30 \
31     /* Debug Breakpoints - one for each possible size of unscaled bytecodes */ \
32     /* and one for each operand widening prefix bytecode */ \
33     V(DebugBreakWide, AccumulatorUse::kReadWrite) \
34     V(DebugBreakExtraWide, AccumulatorUse::kReadWrite) \
35     V(DebugBreak0, AccumulatorUse::kReadWrite)
```

```
|314 // static
|315 bool Bytecodes::IsUnsignedOperandType(OperandType operand_type) {
|316     switch (operand_type) {
|317 #define CASE(Name, _)          \
|318     case OperandType::k##Name: \
|319         return OperandTraits<OperandType::k##Name>::TypeInfoTraits::kIsUnsigned;
|320     OPERAND_TYPE_LIST(CASE)
|321 #undef CASE
|322 }
|323 UNREACHABLE();
|324 }
|325
```

然后开始进行动手实验

```
wuwei@hellogcc: ~> ~/repo/v8/v8
$ find . -name d8
./test/test262/data/implementation-contributed/v8/mjsunit/d8
./test/mjsunit/d8
./out/x64.release/obj/d8
./out/x64.release/d8
./src/d8
```

套路：从 help 中找 dump

```
wuwei@hellogcc: ~> ~/repo/v8/v8
$ ./out/x64.release/d8 --help | grep -i bytecode
--budget-for-feedback-vector-allocation (The budget in amount of bytecode executed by a function before
--ignition-elide-noneffectful-bytecodes (elide bytecodes which won't have any external effect)
--ignition-filter-expression-positions (filter expression positions before the bytecode pipeline)
--print-bytecode (print bytecode generated by ignition interpreter)
--print-bytecode-filter (filter for selecting which functions to print bytecode)
--trace-ignition-codegen (trace the codegen of ignition interpreter bytecode handlers)
--trace-ignition-dispatches (traces the dispatches to bytecode handlers by the ignition interpreter)
--trace-ignition-dispatches-output-file (the file to which the bytecode handler dispatch table is written)
--max-inlined-bytecode-size (maximum size of bytecode for a single inlining)
--max-inlined-bytecode-size-cumulative (maximum cumulative size of bytecode considered for inlining)
--max-inlined-bytecode-size-absolute (maximum cumulative size of bytecode considered for inlining)
--reserve-inline-budget-scale-factor (maximum cumulative size of bytecode considered for inlining)
--max-inlined-bytecode-size-small (maximum size of bytecode considered for small function inlining)
--max-optimized-bytecode-size (maximum bytecode size to be considered for optimization; too high value
--flush-bytecode (flush of bytecode when it has not been executed recently)
--stress-flush-bytecode (stress bytecode flushing)
--regexp-peephole-optimization (enable peephole optimization for regexp bytecode)
--trace-regexp-peephole-optimization (trace regexp bytecode peephole optimization)
--trace-regexp-bytecodes (trace regexp bytecode execution)
--print-regexp-bytecode (print generated regexp bytecode)
```



```
$ ./out/x64.release/d8 --print-bytecode
```

```
47: 648
48: 0x25a60821053d <String[#26]: [crazy non-standard value]>
Handler Table (size = 16)
  from to      hdlr (prediction,  data)
  ( 317, 340) -> 342 (prediction=1, data=11)
Source Position Table (size = 0)
[generated bytecode for function: isProxy (0x25a608210169 <SharedFunctionInfo isProxy>)]
Parameter count 2
Register count 0
Frame size 0
  0x25a608210a62 @ 0 : a7           StackCheck
  0x25a608210a63 @ 1 : 11           LdaFalse
  0x25a608210a64 @ 2 : ab           Return
Constant pool (size = 0)
Handler Table (size = 0)
Source Position Table (size = 0)
undefined
d8> █
```

```
d8> 1 + 2
```

```
d8> 1 + 2
[generated bytecode for function: (0x25a608210acd <SharedFunctionInfo>)]
Parameter count 1
Register count 1
Frame size 8
    0x25a608210b1e @ 0 : a7           StackCheck
    0x25a608210b1f @ 1 : 0c 03         LdaSmi [3]
    0x25a608210b21 @ 3 : 26 fb         Star r0
    0x25a608210b23 @ 5 : ab           Return
Constant pool (size = 0)
Handler Table (size = 0)
Source Position Table (size = 0)
```

```
d8> var x = "HelloGCC"
```

```
d8> var x = "HelloGCC"
```

```
[generated bytecode for function: (0x25a608210bd9 <SharedFunctionInfo>)]
```

```
Parameter count 1
```

```
Register count 4
```

```
Frame size 32
```

0x25a608210c56 @	0 : a7	StackCheck
0x25a608210c57 @	1 : 12 00	LdaConstant [0]
0x25a608210c59 @	3 : 26 fa	Star r1
0x25a608210c5b @	5 : 0b	LdaZero
0x25a608210c5c @	6 : 26 f9	Star r2
0x25a608210c5e @	8 : 27 fe f8	Mov <closure>, r3
0x25a608210c61 @	11 : 61 32 01 fa 03	CallRuntime [DeclareGlobals], r1-r3
0x25a608210c66 @	16 : 12 01	LdaConstant [1]
0x25a608210c68 @	18 : 15 02 02	StaGlobal [2], [2]
0x25a608210c6b @	21 : 0d	LdaUndefined
0x25a608210c6c @	22 : ab	Return

```
Constant pool (size = 3)
```

```
0x25a608210c21: [FixedArray] in OldSpace
```

- map: 0x25a6080404b1 <Map>
- length: 3

- 0: 0x25a608210c01 <FixedArray[4]>
 - 1: 0x25a608210ba1 <String[#8]: HelloGCC>
 - 2: 0x25a60820ff45 <String[#1]: x>

```
Handler Table (size = 0)
```

```
Source Position Table (size = 0)
```

```
undefined
```

```
d8> function foo(x) { return x + 1; }
```

unoptimized

```
d8> function foo(x) { return x + 1; }
[generated bytecode for function: 0x25a608210d35 <SharedFunctionInfo>]
Parameter count 1
Register count 3
Frame size 24
 0x25a608210de2 @ 0 : a7           StackCheck
 0x25a608210de3 @ 1 : 12 00         LdaConstant [0]
 0x25a608210de5 @ 3 : 26 fb         Star r0
 0x25a608210de7 @ 5 : 0b           LdaZero
 0x25a608210de8 @ 6 : 26 fa         Star r1
 0x25a608210dea @ 8 : 27 fe f9         Mov <closure>, r2
 0x25a608210ded @ 11 : 61 32 01 fb 03 CallRuntime [DeclareGlobals], r0-r2
 0x25a608210df2 @ 16 : 0d           LdaUndefined
 0x25a608210df3 @ 17 : ab           Return
```

d8> var y = foo(3)

```
d8> var y = foo(3)
[generated bytecode for function: (0x25a608210ee5 <SharedFunctionInfo>)]
Parameter count 1
Register count 4
Frame size 32
  0x25a608210f62 @ 0 : a7           StackCheck
  0x25a608210f63 @ 1 : 12 00         LdaConstant [0]
  0x25a608210f65 @ 3 : 26 fa         Star r1
  0x25a608210f67 @ 5 : 0b           LdaZero
  0x25a608210f68 @ 6 : 26 f9         Star r2
  0x25a608210f6a @ 8 : 27 fe f8       Mov <closure>, r3
  0x25a608210f6d @ 11 : 61 32 01 fa 03 CallRuntime [DeclareGlobals], r1-r3
  0x25a608210f72 @ 16 : 0d           LdaUndefined
  0x25a608210f73 @ 17 : 26 f9         Star r2
  0x25a608210f75 @ 19 : 13 01 02       LdaGlobal [1], [2]
  0x25a608210f78 @ 22 : 26 fa         Star r1
  0x25a608210f7a @ 24 : 0c 03           LdaSmi [3]
  0x25a608210f7c @ 26 : 26 f8         Star r3
  0x25a608210f7e @ 28 : 5f fa f9 02     CallNoFeedback r1, r2-r3
  0x25a608210f82 @ 32 : 15 02 04       StaGlobal [2], [4]
  0x25a608210f85 @ 35 : 0d           LdaUndefined
  0x25a608210f86 @ 36 : ab           Return
```

Constant pool (size = 3)

0x25a608210f2d: [FixedArray] in OldSpace

```
$ ./out/x64.release/d8 check01.js
```

```
$ cat -n check01.js
```

```
1 var foo = function (x) { return x + 1; }
2 var i = 3;
3 var s = "hellogcc";
4 console.log(foo(i));
5 console.log(foo(s));
```

```
$ ./out/x64.release/d8 check01.js
```

```
4
```

```
hellogcc1
```

接下来要做的工作内容

- **项目目标**：建立完整的字节码的文档说明，映射到具体实现
 - 总体架构我（ww）来写
 - 每个字节码对应的实现分析，每个人认领一些
 - 语言翻译到字节码部分按照语言要素进行分工和认领
- **培训目标**：熟练使用dump功能查看输出，会分析v8的代码

自由讨论环节

到目前为止做到哪一步了？