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

PLCT Group, part of ISRC, ISCAS & HelloGCC Community & HelloLLVM Community

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 Scheider- 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 DLS Keynote: Ignition: Jump-starting an Interpreter for V8
- 317d46c94775 Ignition的部分解读（看英文）
- docs.google 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
```

Calls

```
Call
TailCall
CallRuntime
CallRuntimeForPair
CallJSRuntime
```

Intrinsics

```
InvokeIntrinsic
```

New operator

```
New
```

Test Operators

```
TestEqual
TestNotEqual
TestEqualStrict
```

```
// found in the LICENSE file.

#ifndef V8_INTERPRETER_BYTECODES_H_
#define V8_INTERPRETER_BYTECODES_H_

#include <cstdint>
#include <iosfwd>
#include <string>
#include <vector>

#include "src/common/globals.h"
#include "src/interpreter/bytecode-operands.h"

// This interface and it's implementation are independent of the
// libv8_base library as they are used by the interpreter and the
// standalone mkpeephole table generator program.

namespace v8 {
namespace internal {
namespace interpreter {

// The list of bytecodes which are interpreted by the interpreter.
// Format is V(<bytecode>, <accumulator_use>, <operands>).
#define BYTECODE_LIST(V)                                                       \
  /* Extended width operands */                                                \
  V(Wide, AccumulatorUse::kNone)                                               \
  V(ExtraWide, AccumulatorUse::kNone)                                          \
                                                                               \
  /* Debug Breakpoints - one for each possible size of unscaled bytecodes */   \
  /* and one for each operand widening prefix bytecode                    */   \
  V(DebugBreakWide, AccumulatorUse::kReadWrite)                                \
  V(DebugBreakExtraWide, AccumulatorUse::kReadWrite)                           \
  V(DebugBreak0, AccumulatorUse::kReadWrite)                                   \
  V(DebugBreak1, AccumulatorUse::kReadWrite, OperandType::kReg)                \
  V(DebugBreak2, AccumulatorUse::kReadWrite, OperandType::kReg,                \
    OperandType::kReg)                                                         \
  V(DebugBreak3, AccumulatorUse::kReadWrite, OperandType::kReg,                \
    OperandType::kReg, OperandType::kReg)                                      \
  V(DebugBreak4, AccumulatorUse::kReadWrite, OperandType::kReg,                \
    OperandType::kReg, OperandType::kReg, OperandType::kReg)                   \
  V(DebugBreak5, AccumulatorUse::kReadWrite, OperandType::kRuntimeId,          \
  V(LdaLookupGlobalSlotInsideTypeof, AccumulatorUse::kWrite,                   \
    OperandType::kIdx, OperandType::kIdx, OperandType::kUImm)                  \
  V(StaLookupSlot, AccumulatorUse::kReadWrite, OperandType::kIdx,              \
    OperandType::kFlag8)                                                       \
                                                                               \
  /* Register-accumulator transfers */                                         \
  V(Ldar, AccumulatorUse::kWrite, OperandType::kReg)                           \
  V(Star, AccumulatorUse::kRead, OperandType::kRegOut)                         \
                                                                               \
  /* Register-register transfers */                                            \
  V(Mov, AccumulatorUse::kNone, OperandType::kReg, OperandType::kRegOut)       \
                                                                               \
  /* Property loads (LoadIC) operations */                                     \
  V(LdaNamedProperty, AccumulatorUse::kWrite, OperandType::kReg,               \
    OperandType::kIdx, OperandType::kIdx)                                      \
  V(LdaNamedPropertyNoFeedback, AccumulatorUse::kWrite, OperandType::kReg,     \
    OperandType::kIdx)                                                         \
  V(LdaKeyedProperty, AccumulatorUse::kReadWrite, OperandType::kReg,           \
    OperandType::kIdx)                                                         \
                                                                               \
  /* Operations on module variables */                                         \
  V(LdaModuleVariable, AccumulatorUse::kWrite, OperandType::kImm,              \
    OperandType::kUImm)                                                        \
  V(StaModuleVariable, AccumulatorUse::kRead, OperandType::kImm,               \
    OperandType::kUImm)                                                        \
                                                                               \
  /* Propery stores (StoreIC) operations */                                    \
  V(StaNamedProperty, AccumulatorUse::kReadWrite, OperandType::kReg,           \
    OperandType::kIdx, OperandType::kIdx)                                      \
  V(StaNamedPropertyNoFeedback, AccumulatorUse::kReadWrite, OperandType::kReg, \
    OperandType::kIdx, OperandType::kFlag8)                                    \
  V(StaNamedOwnProperty, AccumulatorUse::kReadWrite, OperandType::kReg,        \
    OperandType::kIdx, OperandType::kIdx)                                      \
  V(StaKeyedProperty, AccumulatorUse::kReadWrite, OperandType::kReg,           \
    OperandType::kReg, OperandType::kIdx)                                      \
  V(StaInArrayLiteral, AccumulatorUse::kReadWrite, OperandType::kReg,          \
    OperandType::kReg, OperandType::kIdx)                                      \
  V(StaDataPropertyInLiteral, AccumulatorUse::kRead, OperandType::kReg,        \
    OperandType::kReg, OperandType::kFlag8, OperandType::kIdx)                 \
  V(CollectTypeProfile, AccumulatorUse::kRead, OperandType::kImm)              \
                                                                               \
```

```
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 befor
  --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 writ
  --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; }

```
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？