ISCAS 中国科学院软件研究所 Institute of Software Chinese Academy of Sciences | 智能软件研究中心 Intelligent Software Research Center

V8中HelloWorld的解释执行过程-part2

智能软件研究中心 邱吉

qiuji@iscas.ac.cn

2021/09/10

上集回顾

在上次课程中，讲述了：

- hello.js：print(“HelloWorld!”)的字节码和含义
- 如何从--trace-sim的log文件中，梳理hello.js的解释执行过程

hello.js的字节码

./d8 --print-bytecode hello.js

```
print( ”  hello”  )
```

```
Bytecode length: 13
Parameter count 1
Register count 3
Frame size 24
OSR nesting level: 0
Bytecode Age: 0
        0xdf23ca206e @    0 : 21 00 00        LdaGlobal [0], [0]
        0xdf23ca2071 @    3 : c2              Star1
        0xdf23ca2072 @    4 : 13 01           LdaConstant [1]
        0xdf23ca2074 @    6 : c1              Star2
        0xdf23ca2075 @    7 : 61 f9 f8 02       CallUndefinedReceiver1 r1, r2, [2]
        0xdf23ca2079 @   11 : c3              Star0
        0xdf23ca207a @   12 : a8              Return
Constant pool (size = 2)
0xdf23ca2019: [FixedArray] in OldSpace
 - map: 0x001bf11012c1 <Map>
 - length: 2
           0: 0x00df23c813a9 <String[5]: #print>
           1: 0x00df23ca1f71 <String[5]: #hello>
Handler Table (size = 0)
Source Position Table (size = 0)
```

概览./d8 --trace-sim hello.js 2>&1 |tee logtracesim.txt

- grep搜索所有的“Call to”和“Return to”的行，就可以得到执行流如何在各个builtins中传递
- 蓝色的部分就是解释器Ignition执行过程

CallImpl JSEntry
Call Builtin JSEntryTrampoline
Call Builtin Call_ReceiverIsAny
Call Builtin CallFunction_ReceiverIsAny

第一部分：Prologue

Call Builtin InterpreterEntryTrampoline
Call Builtin LdaGlobalHandler
Call Builtin LoadGlobalIC_NoFeedback
Call Builtin LoadIC_NoFeedback
Call Builtin
CEntry_Return1_DontSaveFPRegs_ArgvOnStack_NoBuiltinExit
Call host Runtime::LoadNoFeedbackIC_Miss
Return Builtin LdaGlobalHandler
Call Builtin LdaConstantHandler
Call Builtin CallUndefinedReceiver1Handler
Call Builtin Call_ReceiverIsAny
Call Builtin CallFunction_ReceiverIsAny
Call Builtin HandleApiCall
Call Builtin AdaptorWithBuiltinExitFrame
Call Builtin CEntry_Return1_DontSaveFPRegs_ArgvOnStack_BuiltinExit
Call host Builtin_HandleApiCall
Return Builtin InterpreterEntryTrampoline
Call Builtin ShortStarHandler
Call Builtin ReturnHandler
Return Builtin InterpreterEntryTrampoline

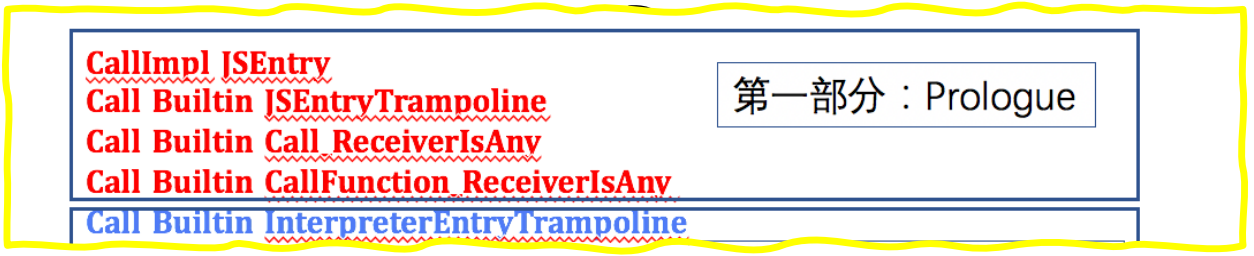
第二部分：解释器主体

Return Builtin JSEntryTrampoline
Return Builtin JSEntry

第三部分：Epilogue

本次内容

- hello.js是如何进入解释器Ignition的



- 调试的代码和log：https://github.com/qjivy/v8/tree/v8ignition-learn

整体过程 @ d8.cc

main -> v8::Shell::Main -> v8::Shell::RunMain ->
v8::SourceGroup::Execute -> v8::Shell::ExecuteString

i::parsing::ParseProgram(&parse_info, script,
i_isolate, i::parsing::ReportStatisticsMode::kYes))

生成AST

```
Local<Script> script;
if ( !CompileString<Script>(isolate, context, source, origin)
.ToLocal(&script))
{ return false; }
```

生成Bytecode

maybe_result = script->Run(realm);

解释执行

CompileString的结果：script


```
(gdb) jlh script
0xa3f29e22f1: [Function] in OldSpace
 - map: 0x0079a86813a1 <Map(HOLEY_ELEMENTS)> [FastProperties]
 - prototype: 0x00a3f29c42d9 <JSFunction (sfi = 0x846444df61)>
 - elements: 0x004a4f201309 <FixedArray[0]> [HOLEY_ELEMENTS]
 - function prototype:
 - initial_map:
 - shared_info: 0x00a3f29e21f1 <SharedFunctionInfo>
 - name: 0x004a4f2017b1 <String[0]: #>
 - builtin: InterpreterEntryTrampoline
 - formal_parameter_count: 0
 - kind: NormalFunction
 - context: 0x00a3f29c38c9 <NativeContext[256]>
 - code: 0x7fe6a2444381 <Code BUILTIN InterpreterEntryTrampoline>
 - interpreted
 - bytecode: 0x00a3f29e2291 <BytecodeArray[13]>
 - source code: print("hello")

 - properties: 0x004a4f201309 <FixedArray[0]>
 - All own properties (excluding elements): {
    0x4a4f204d51: [String] in ReadOnlySpace: #length: 0x008464441499 <AccessorInfo> (const accessor descriptor), location: descriptor
    0x4a4f204ee1: [String] in ReadOnlySpace: #name: 0x008464441429 <AccessorInfo> (const accessor descriptor), location: descriptor
    0x4a4f204221: [String] in ReadOnlySpace: #arguments: 0x008464441349 <AccessorInfo> (const accessor descriptor), location: descriptor
    0x4a4f2044c1: [String] in ReadOnlySpace: #caller: 0x0084644413b9 <AccessorInfo> (const accessor descriptor), location: descriptor
    0x4a4f2051c9: [String] in ReadOnlySpace: #prototype: 0x008464441509 <AccessorInfo> (const accessor descriptor), location: descriptor
 }
 - feedback vector: No feedback vector, but we have a closure feedback cell array
0x4a4f202fb9: [ClosureFeedbackCellArray] in ReadOnlySpace
 - map: 0x004a4f201f19 <Map>
 - length: 0
```

1. (gdb) source deps/v8/tools/gdbinit，using V8's gdb macros for low level debugging
2. 设置断点到CompileString函数之后，
3. 使用 jlh来查看local handler script
可以看到Script的SharedFunctionInfo的打印信息，data区域是Bytecode，code入口是InterpreterEntryTrampoline

Hello.js step by step- *HOWTO JSEntry*

CallImpl JSEntry
Call Builtin JSEntryTrampoline
Call Builtin Call_ReceiverIsAny
Call Builtin CallFunction_ReceiverIsAny

第一部分：Prologue

Call Builtin InterpreterEntryTrampoline
Call Builtin LdaGlobalHandler
Call Builtin LoadGlobalIC_NoFeedback
Call Builtin LoadIC_NoFeedback
Call Builtin
CEntry_Return1_DontSaveFPRegs_ArgvOnStack_NoBuiltinExit
Call host Runtime::LoadNoFeedbackIC_Miss
Return Builtin LdaGlobalHandler
Call Builtin LdaConstantHandler
Call Builtin CallUndefinedReceiver1Handler
Call Builtin Call_ReceiverIsAny
Call Builtin CallFunction_ReceiverIsAny
Call Builtin HandleApiCall
Call Builtin AdaptorWithBuiltinExitFrame
Call Builtin CEntry_Return1_DontSaveFPRegs_ArgvOnStack_BuiltinExit
Call host Builtin_HandleApiCall
Return Builtin InterpreterEntryTrampoline
Call Builtin ShortStarHandler
Call Builtin ReturnHandler
Return Builtin InterpreterEntryTrampoline

第二部分：解释器主体

Return Builtin JSEntryTrampoline
Return Builtin JSEntry

第三部分：Epilogue

第一部分：Prologue的进入

- 为什么是JSEntry？

```
@  v8::internal::(anonymous namespace)::Invoke (isolate=0x55b8725c59a0, params=...)
at ../../src/execution/execution.cc:358

{
...

  Handle<Code> code = JSEntry(isolate, params.execution_target, params.is_construct);

  using JSEntryFunction = GeneratedCode<Address(
        Address root_register_value, Address new_target, Address target,
        Address receiver, intptr_t argc, Address** argv)>;

  JSEntryFunction stub_entry =
        JSEntryFunction::FromAddress(isolate, code->InstructionStart());
...
}
```

第一部分：Prologue的进入

- 为什么是JSEntry？

```
@  v8::internal::(anonymous namespace)::Invoke (isolate=0x55b8725c59a0, params=...)
at ../../src/execution/execution.cc:350

{
...

  Handle<Code> code = JSEntry(isolate, params.execution_target, params.is_construct);

  using JSEntryFunction = GeneratedCode<Address(
        Address root_register_value, A
        Address receiver, intptr_t argc,

  JSEntryFunction stub_entry =
        JSEntryFunction::FromAddress(
...
}
```

```
Handle<Code> JSEntry(Isolate* isolate, Execution::Target execution_target,
                     bool is_construct) {
  if (is_construct) {
    DCHECK_EQ(Execution::Target::kCallable, execution_target);
    return BUILTIN_CODE(isolate, JSConstructEntry);
  } else if (execution_target == Execution::Target::kCallable) {
    DCHECK(!is_construct);
    return BUILTIN_CODE(isolate, JSEntry);
  } else if (execution_target == Execution::Target::kRunMicrotasks) {
    DCHECK(!is_construct);
    return BUILTIN_CODE(isolate, JSRunMicrotasksEntry);
  }
  UNREACHABLE();
}
```

如何进入JSEntry-1


```
@  v8::internal::(anonymous namespace)::Invoke (isolate=0x55b8725c59a0, params=...)
at ../../src/execution/execution.cc:376

value = Object(stub_entry.Call(isolate->isolate_data()->isolate_root(),
                               orig_func, func, recv, params.argc, argv));
```

```
@  v8::internal::GeneratedCode<unsigned long, unsigned long, unsigned long, unsigned long,
unsigned long, long, unsigned long**>::Call (this=0x7ffc32118f18, args=0x0, args=0x0,
args=0x0, args=0x0, args=0x0, args=0x0) at ../../src/execution/simulator.h:126
{
  Return Call(Args... args) {
    return Simulator::current(isolate_)->template Call<Return>( reinterpret_cast<Address>(fn_ptr_), args...);
  }

}
```

如何进入JSEntry-2


```
@ v8::internal::(anonymous namespace)::Invoke (isolate=0x55b8725c59a0, params=...)
@ v8::internal::GeneratedCode<unsigned long, unsigned long, unsigned long, unsigned long, unsigned
long, long, unsigned long**>::Call (this=0x7ffc32118f18, args=0x0, args=0x0, args=0x0, args=0x0, args=0x0,
args=0x0) at ../../src/execution/simulator.h:126
{
#ifdef USE_SIMULATOR
  Return Call(Args... args) {
    return Simulator::current(isolate_)->template Call<Return>( reinterpret_cast<Address>(fn_ptr_), args...);
  }
#else
return fn_ptr_(args...);
#endif }
```

```
@ v8::internal::Simulator::Call<unsigned long, unsigned long, unsigned long, unsigned long,
unsigned long, long, unsigned long**> (this=0x55f6c89be450, entry=140326543809824,
    args=0x0, args=0x0, args=0x0, args=0x0, args=0x0, args=0x0)
at ../../src/execution/riscv64/simulator-riscv64.h:369

template <typename Return, typename... Args>
  Return Call(Address entry, Args... args) {
    return VariadicCall<Return>(this, &Simulator::CallImpl, entry, args...);
  }
```

如何进入JSEntry-3

@ v8::internal::(anonymous namespace)::Invoke (isolate=0x55b8725c59a0, params=...)

@ v8::internal::Simulator::Call<unsigned long, unsigned long, unsigned long, unsigned long, unsigned long, long, unsigned long**> (this=0x55f6c89be450, entry=140326543809824, args=0x0, args=0x0, args=0x0, args=0x0, args=0x0, args=0x0)
at ../../src/execution/riscv64/simulator-riscv64.h:369

```
template <typename Return, typename... Args>
  Return Call(Address entry, Args... args) {
    return VariadicCall<Return>(this, &Simulator::CallImpl, entry, args...);
  }
```

@ v8::internal::SimulatorBase::VariadicCall at ../../src/execution/simulator-base.h:46

@ v8::internal::Simulator::CallImpl (this=0x55f6c89be450, entry=140326543809824, argument_count=6, arguments=0x7ffc32118c30) at ../../src/execution/riscv64/simulator-riscv64.cc:3575

@ src/execution/riscv64/simulator-riscv64.cc
CallInternal(entry); -> Execute() // Start the simulation.

如何进入JSEntry-4

@ v8::internal::(anonymous namespace)::Invoke (isolate=0x55b8725c59a0, params=...)

@ v8::internal::Simulator::Call<unsigned long, unsigned long, unsigned long, unsigned long, unsigned long, long, unsigned long**> (this=0x55f6c89be450, entry=140326543809824,
args=0x0, args=0x0, args=0x0, args=0x0, args=0x0, args=0x0)
at ../../src/execution/riscv64/simulator-riscv64.h:369

```
CallImpl: reg_arg_count = 6 entry-pc (JSEntry) = 0x56392d7cf580 a0 (Isolate) = 0x56392edd3fe0 a1 (orig_func/new_target) = 0xba28901599 a2 (func/target) = 0xb7df2a20a9 a3 (receive
r) = 0xb7df2838a9 a4 (argc) = 0x0 a5 (argv) = 0x0
  0x56392d7cf580  1    f9810113       addi      sp, sp, -104          00007f9b67ef2f58    (1)    int64:140305440386904 uint64:140305440386904
  0x5639                                                                                  (2)    int64:-2 uint64:18446744073709551614 --> [addr: 7f9b67ef2fb8]
  0x5639                                                                                  (3)    int64:0 uint64:0 --> [addr: 7f9b67ef2fb0]
  0x56392d7cf58c  4    05b13823       sd        s11, 80(sp)                              (4)    int64:0 uint64:0 --> [addr: 7f9b67ef2fa8]
  0x56392d7cf590  5    05a13423       sd        s10, 72(sp)                              (5)    int64:0 uint64:0 --> [addr: 7f9b67ef2fa0]
  0x56392d7cf594  6    05913023       sd        s9, 64(sp)                               (6)    int64:0 uint64:0 --> [addr: 7f9b67ef2f98]
  0x56392d7cf598  7    03813c23       sd        s8, 56(sp)                               (7)    int64:0 uint64:0 --> [addr: 7f9b67ef2f90]
  0x56392d7cf59c  8    03713823       sd        s7, 48(sp)                               (8)    int64:0 uint64:0 --> [addr: 7f9b67ef2f88]
  0x56392d7cf5a0  9    03613423       sd        s6, 40(sp)                               (9)    int64:0 uint64:0 --> [addr: 7f9b67ef2f80]
  0x56392d7cf5a4  10    03513023       sd        s5, 32(sp)                              (10)    int64:0 uint64:0 --> [addr: 7f9b67ef2f78]
  0x56392d7cf5a8  11    01413c23       sd        s4, 24(sp)                              (11)    int64:0 uint64:0 --> [addr: 7f9b67ef2f70]
  0x56392d7cf5ac  12    01313823       sd        s3, 16(sp)                              (12)    int64:0 uint64:0 --> [addr: 7f9b67ef2f68]
  0x56392d7cf5b0  13    01213423       sd        s2, 8(sp)                               (13)    int64:0 uint64:0 --> [addr: 7f9b67ef2f60]
```

trace log的开头

@ src/execution/riscv64/simulator-riscv64.cc
CallInternal(entry); -> Execute() // Start the simulation.

Hello.js step by step- *HOWTO JSEntryTrampoline*

CallImpl JSEntry
Call Builtin JSEntryTrampoline
Call Builtin Call_ReceiverIsAny
Call Builtin CallFunction_ReceiverIsAny

第一部分：Prologue

Call Builtin InterpreterEntryTrampoline
Call Builtin LdaGlobalHandler
Call Builtin LoadGlobalIC_NoFeedback
Call Builtin LoadIC_NoFeedback
Call Builtin
CEntry_Return1_DontSaveFPRegs_ArgvOnStack_NoBuiltinExit
Call host Runtime::LoadNoFeedbackIC_Miss
Return Builtin LdaGlobalHandler
Call Builtin LdaConstantHandler
Call Builtin CallUndefinedReceiver1Handler
Call Builtin Call_ReceiverIsAny
Call Builtin CallFunction_ReceiverIsAny
Call Builtin HandleApiCall
Call Builtin AdaptorWithBuiltinExitFrame
Call Builtin CEntry_Return1_DontSaveFPRegs_ArgvOnStack_BuiltinExit
Call host Builtin_HandleApiCall
Return Builtin InterpreterEntryTrampoline
Call Builtin ShortStarHandler
Call Builtin ReturnHandler
Return Builtin InterpreterEntryTrampoline

第二部分：解释器主体

Return Builtin JSEntryTrampoline
Return Builtin JSEntry

第三部分：Epilogue

如何进入JSEntryTrampoline

从ASM builtin的生成函数中去探究：builtins/riscv64/builtins-riscv64.cc

```
void Builtins::Generate_JSEntry(MacroAssembler* masm) {
  Generate_JSEntryVariant(masm, StackFrame::ENTRY, Builtin::kJSEntryTrampoline);
}
```

```
void Generate_JSEntryVariant(MacroAssembler* masm, StackFrame::Type type,
                             Builtin entry_trampoline) {
...
  Handle<Code> trampoline_code =
      masm->isolate()->builtins()->code_handle(entry_trampoline);
  __ Call(trampoline_code, RelocInfo::CODE_TARGET);
 ...
}
```

Hello.js step by step - *HOWTO Call_ReceiverIsAny*

CallImpl JSEntry
Call Builtin JSEntryTrampoline
Call Builtin Call_ReceiverIsAny
Call Builtin CallFunction_ReceiverIsAny

第一部分：Prologue

Call Builtin InterpreterEntryTrampoline
Call Builtin LdaGlobalHandler
Call Builtin LoadGlobalIC_NoFeedback
Call Builtin LoadIC_NoFeedback
Call Builtin
CEntry_Return1_DontSaveFPRegs_ArgvOnStack_NoBuiltinExit
Call host Runtime::LoadNoFeedbackIC_Miss
Return Builtin LdaGlobalHandler
Call Builtin LdaConstantHandler
Call Builtin CallUndefinedReceiver1Handler
Call Builtin Call_ReceiverIsAny
Call Builtin CallFunction_ReceiverIsAny
Call Builtin HandleApiCall
Call Builtin AdaptorWithBuiltinExitFrame
Call Builtin CEntry_Return1_DontSaveFPRegs_ArgvOnStack_BuiltinExit
Call host Builtin_HandleApiCall
Return Builtin InterpreterEntryTrampoline
Call Builtin ShortStarHandler
Call Builtin ReturnHandler
Return Builtin InterpreterEntryTrampoline

第二部分：解释器主体

Return Builtin JSEntryTrampoline
Return Builtin JSEntry

第三部分：Epilogue

如何进入Call_ReceiverIsAny-1

从ASM builtin的生成函数中去探究：builtins/riscv64/builtins-riscv64.cc

```
void Builtins::Generate_JSEntryTrampoline(MacroAssembler* masm) {
  Generate_JSEntryTrampolineHelper(masm, false);
}
```

```
static void Generate_JSEntryTrampolineHelper(MacroAssembler* masm,
                                             bool is_construct) {
...
    // Invoke the code.
    Handle<Code> builtin = is_construct
                           ? BUILTIN_CODE(masm->isolate(), Construct)
                           : masm->isolate()->builtins()->Call();
    __ Call(builtin, RelocInfo::CODE_TARGET);
  ...
}
```

```
@ src/buitlin.h(.cc)
Handle<Code> Call(ConvertReceiverMode = ConvertReceiverMode::kAny);
```

如何进入Call_ReceiverIsAny-2


```
static void Generate_JSEntryTrampolineHelper(MacroAssembler* masm,
                                             bool is_construct) {
...`
    // Invoke the code.
    Handle<Code> builtin = is_construct
                           ? BUILTIN_CODE(masm->isolate(), Construct)
                           : masm->isolate()->builtins()->Call();
      Call(builtin, RelocInfo::CODE_TARGET);
```

@ src/buitlin.h(.cc)

```
} Handle<Code> Call(ConvertReceiverMode = ConvertReceiverMode::kAny);

Handle<Code> Builtins::Call(ConvertReceiverMode mode) {
  switch (mode) {
    case ConvertReceiverMode::kNullOrUndefined:
      return code_handle(Builtin::kCall_ReceiverIsNullOrUndefined);
    case ConvertReceiverMode::kNotNullOrUndefined:
      return code_handle(Builtin::kCall_ReceiverIsNotNullOrUndefined);
    case ConvertReceiverMode::kAny:
      return code_handle(Builtin::kCall_ReceiverIsAny);
  }
  UNREACHABLE();
}
```

Hello.js step by step - *HOWTO CallFunction_ReceiverIsAny*

CallImpl JSEntry
Call Builtin JSEntryTrampoline
Call Builtin Call_ReceiverIsAny
Call Builtin CallFunction_ReceiverIsAny

第一部分：Prologue

Call Builtin InterpreterEntryTrampoline
Call Builtin LdaGlobalHandler
Call Builtin LoadGlobalIC_NoFeedback
Call Builtin LoadIC_NoFeedback
Call Builtin
CEntry_Return1_DontSaveFPRegs_ArgvOnStack_NoBuiltinExit
Call host Runtime::LoadNoFeedbackIC_Miss
Return Builtin LdaGlobalHandler
Call Builtin LdaConstantHandler
Call Builtin CallUndefinedReceiver1Handler
Call Builtin Call_ReceiverIsAny
Call Builtin CallFunction_ReceiverIsAny
Call Builtin HandleApiCall
Call Builtin AdaptorWithBuiltinExitFrame
Call Builtin CEntry_Return1_DontSaveFPRegs_ArgvOnStack_BuiltinExit
Call host Builtin_HandleApiCall
Return Builtin InterpreterEntryTrampoline
Call Builtin ShortStarHandler
Call Builtin ReturnHandler
Return Builtin InterpreterEntryTrampoline

第二部分：解释器主体

Return Builtin JSEntryTrampoline
Return Builtin JSEntry

第三部分：Epilogue

如何进入CallFunction_ReceiverIsAny-1

从ASM builtin的生成函数中去探究： builtins/builtins-call-gen.cc

```
void Builtins::Generate_Call_ReceiverIsAny(MacroAssembler* masm) {
  Generate_Call(masm, ConvertReceiverMode::kAny);
}
```

```
@ src/builtins/riscv64/builtins-riscv64.cc
void Builtins::Generate_Call(MacroAssembler* masm, ConvertReceiverMode mode) {
…
  __ Jump(masm->isolate()->builtins()->CallFunction(mode),
          RelocInfo::CODE_TARGET, Uless_equal, range,
          Operand(LAST_JS_FUNCTION_TYPE - FIRST_JS_FUNCTION_TYPE));
…}
```

如何进入CallFunction_ReceiverIsAny-2

从ASM builtin的生成函数中去探究： builtins/builtins-call-gen.cc

```
void Builtins::Generate_Call_ReceiverIsAny(MacroAssembler* masm) {
}
```

```
@ src/builtins/riscv64/builtins-riscv64.cc
void Builtins::Generate_Call(MacroAssembler* masm, ConvertReceiverMode mode) {
...
  __ Jump(masm->isolate()->builtins()->CallFunction(mode),
          RelocInfo::CODE_TARGET, Uless_equal, range,
          Operand(LAST_JS_FUNCTION_TYPE - FIRST_JS_FUNCTION_TYPE));
...}
```

```
@ src/builtins/builtins.cc
Handle<Code> Builtins::CallFunction(ConvertReceiverMode mode) {
  switch (mode) {
    case ConvertReceiverMode::kNullOrUndefined:
      return code_handle(Builtin::kCallFunction_ReceiverIsNullOrUndefined);
    case ConvertReceiverMode::kNotNullOrUndefined:
      return code_handle(Builtin::kCallFunction_ReceiverIsNotNullOrUndefined);
    case ConvertReceiverMode::kAny:
      return code_handle(Builtin::kCallFunction_ReceiverIsAny);
  }
...
```

Hello.js step by step - *HOWTO InterpreterEntryTrampoline*

CallImpl JSEntry
Call Builtin JSEntryTrampoline
Call Builtin Call_ReceiverIsAny
Call Builtin CallFunction_ReceiverIsAny

第一部分：Prologue

Call Builtin InterpreterEntryTrampoline
Call Builtin LdaGlobalHandler
Call Builtin LoadGlobalIC_NoFeedback
Call Builtin LoadIC_NoFeedback
Call Builtin
CEntry_Return1_DontSaveFPRegs_ArgvOnStack_NoBuiltinExit
Call host Runtime::LoadNoFeedbackIC_Miss
Return Builtin LdaGlobalHandler
Call Builtin LdaConstantHandler
Call Builtin CallUndefinedReceiver1Handler
Call Builtin Call_ReceiverIsAny
Call Builtin CallFunction_ReceiverIsAny
Call Builtin HandleApiCall
Call Builtin AdaptorWithBuiltinExitFrame
Call Builtin CEntry_Return1_DontSaveFPRegs_ArgvOnStack_BuiltinExit
Call host Builtin_HandleApiCall
Return Builtin InterpreterEntryTrampoline
Call Builtin ShortStarHandler
Call Builtin ReturnHandler
Return Builtin InterpreterEntryTrampoline

第二部分：解释器主体

Return Builtin JSEntryTrampoline
Return Builtin JSEntry

第三部分：Epilogue

如何进入InterpreterEntryTrampoline-1

从ASM builtin的生成函数中去探究： builtins/builtins-call-gen.cc

```
void Builtins::Generate_CallFunction_ReceiverIsAny(MacroAssembler* masm) {
  Generate_CallFunction(masm, ConvertReceiverMode::kAny);
}
```

```
@ src/builtins/riscv64/builtins-riscv64.cc
void Builtins::Generate_CallFunction(MacroAssembler* masm,
                                     ConvertReceiverMode mode) {
  // ----------- S t a t e -------------
  //  -- a0 : the number of arguments (not including the receiver)
  //  -- a1 : the function to call (checked to be a JSFunction)
  // -----------------------------------
...
  __ InvokeFunctionCode(a1, no_reg, a2, a0, InvokeType::kJump);\
...
}
```

如何进入InterpreterEntryTrampoline-2


```
@ src/builtins/riscv64/builtins-riscv64.cc
void Builtins::Generate_CallFunction(MacroAssembler* masm,
                                     ConvertReceiverMode mode) {
  // ----------- S t a t e -------------
  //  -- a0 : the number of arguments (not including the receiver)
  //  -- a1 : the function to call (checked to be a JSFunction)
  // -----------------------------------
...
  __ InvokeFunctionCode(a1, no_reg, a2, a0, InvokeType::kJump);\
...
}
```

```
@ src/codegen/riscv64/macro-assembler-riscv64.cc
void MacroAssembler::InvokeFunctionCode(Register function, Register new_target,
                                        Register expected_parameter_count,
                                        Register actual_parameter_count,
                                        InvokeType type) {
...
  LoadTaggedPointerField(code,
                         FieldMemOperand(function, JSFunction::kCodeOffset));
  JumpCodeObject(code)

...}
```

如何进入InterpreterEntryTrampoline-3


```
@ src/builtins/riscv64/builtins-riscv64.cc
void Builtins::Generate_CallFunction(MacroAssembler* masm,
                                     ConvertReceiverMode mode) {
  // ----------- S t a t e -------------
  //  -- a0 : the number of arguments (not including the receiver)
  //  -- a1 : the function to call (checked to be a JSFunction)
  // -----------------------------------
...
  __ InvokeFunctionCode(a1, no_reg, a2, a0, InvokeType::kJump);\
...
}
```

```
@ src/codegen/riscv64/macro-assembler-riscv64.cc
void MacroAssembler::InvokeFunctionCode(Register function,
                                        Register expected_parameter_count,
                                        Register actual_parameter_count,
                                        InvokeType type) {
...
  LoadTaggedPointerField(code,
                         FieldMemOperand(function, JSFunction::kCodeOffset));
  JumpCodeObject(code)

...}
```

CompileString的结果：script

```
(gdb) jlh script
0xa3f29e22f1: [Function] in OldSpace
 - map: 0x0079a86813a1 <Map(HOLEY_ELEMENTS)> [FastProperties]
 - prototype: 0x00a3f29c42d9 <JSFunction (sfi = 0x846444df61)>
 - elements: 0x004a4f201309 <FixedArray[0]> [HOLEY_ELEMENTS]
 - function prototype:
 - initial_map:
 - shared_info: 0x00a3f29e21f1 <SharedFunctionInfo>
 - name: 0x004a4f2017b1 <String[0]: #>
 - builtin: InterpreterEntryTrampoline
 - formal_parameter_count: 0
 - kind: NormalFunction
 - context: 0x00a3f29c38c9 <NativeContext[256]>
 - code: 0x7fe6a2444381 <Code BUILTIN InterpreterEntryTrampoline>
 interpreted
 - bytecode: 0x00a3f29e2291 <BytecodeArray[13]>
 - source code: print("hello")

 - properties: 0x004a4f201309 <FixedArray[0]>
 - All own properties (excluding elements): {
    0x4a4f204d51: [String] in ReadOnlySpace: #length: 0x008464441499 <AccessorInfo> (const accessor descriptor), location: descriptor
    0x4a4f204ee1: [String] in ReadOnlySpace: #name: 0x008464441429 <AccessorInfo> (const accessor descriptor), location: descriptor
    0x4a4f204221: [String] in ReadOnlySpace: #arguments: 0x008464441349 <AccessorInfo> (const accessor descriptor), location: descriptor
    0x4a4f2044c1: [String] in ReadOnlySpace: #caller: 0x0084644413b9 <AccessorInfo> (const accessor descriptor), location: descriptor
    0x4a4f2051c9: [String] in ReadOnlySpace: #prototype: 0x008464441509 <AccessorInfo> (const accessor descriptor), location: descriptor
 }
 - feedback vector: No feedback vector, but we have a closure feedback cell array
0x4a4f202fb9: [ClosureFeedbackCellArray] in ReadOnlySpace
 - map: 0x004a4f201f19 <Map>
 - length: 0
```

1. (gdb) source deps/v8/tools/gdbinit，using V8's gdb macros for low level debugging
2. 设置断点到CompileString函数之后，
3. 使用 jlh来查看local handler script 可以看到Script的SharedFunctionInfo的打印信息，data区域是Bytecode，code入口是InterpreterEntryTrampoline

总结

- 在上次课程中，讲述了：
  - hello.js：print(“HelloWorld!”)的字节码和含义
  - 如何从--trace-sim的log文件中，梳理hello.js的解释执行过程
- 本次课程：
  - d8上hello.js的整体执行流程
  - 如何进入第一部分Prologue部分开始执行
    - Builtin by Builtin

  - 掌握技能：
    - using v8 gdb macro commands
    - 如何通过阅读Builtin的生成代码，了解Builtin的执行流程

谢 谢

欢迎交流合作

2020/09/10