V8 移植简介

软件所智能软件中心PLCT实验室 邹小芳

2020/05/27

目录

01 V8 compilation overview

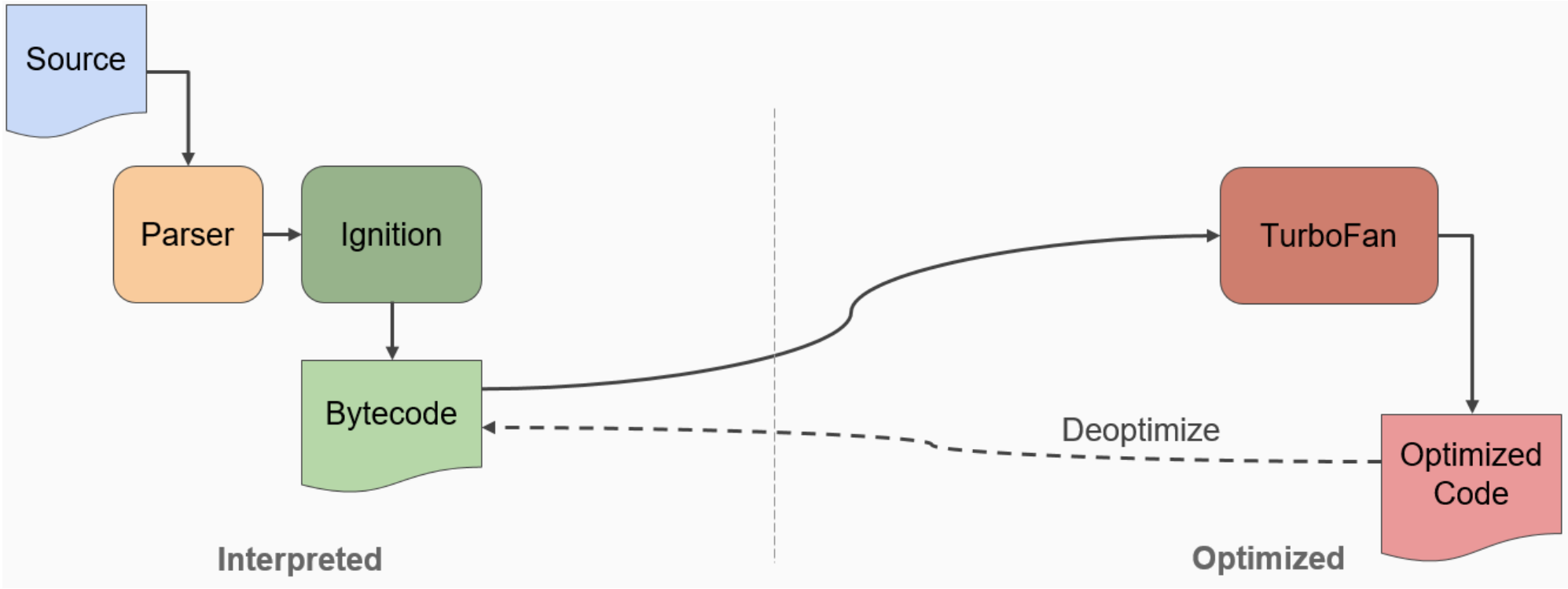
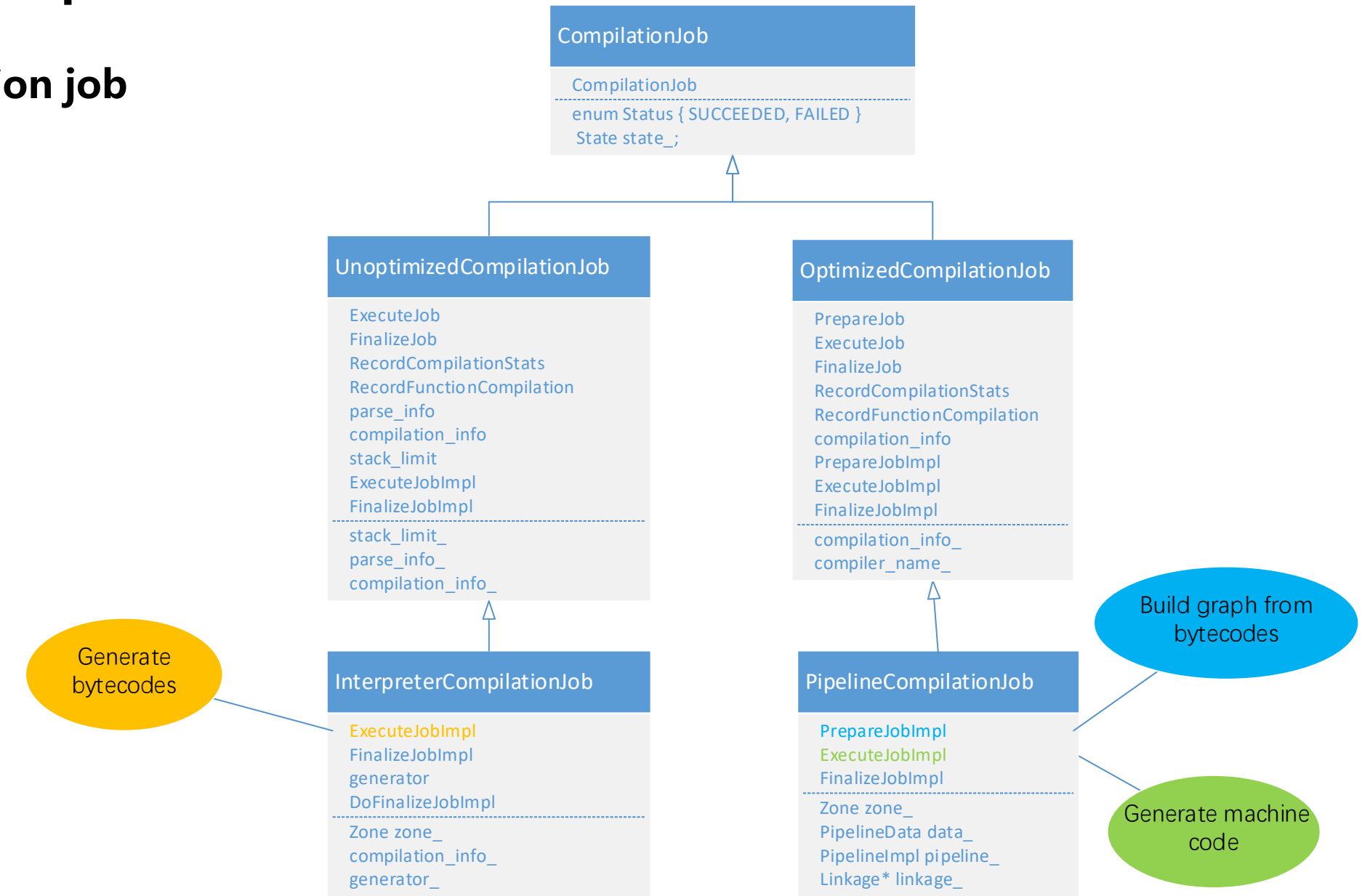


Image from:
https://docs.google.com/presentation/d/1OqjVqRhtwlKeKfvMdX6HaCIu9wpZsrzqpIVIwQSuiXQ/edit#slide=id.g1453eb7f19_0_391

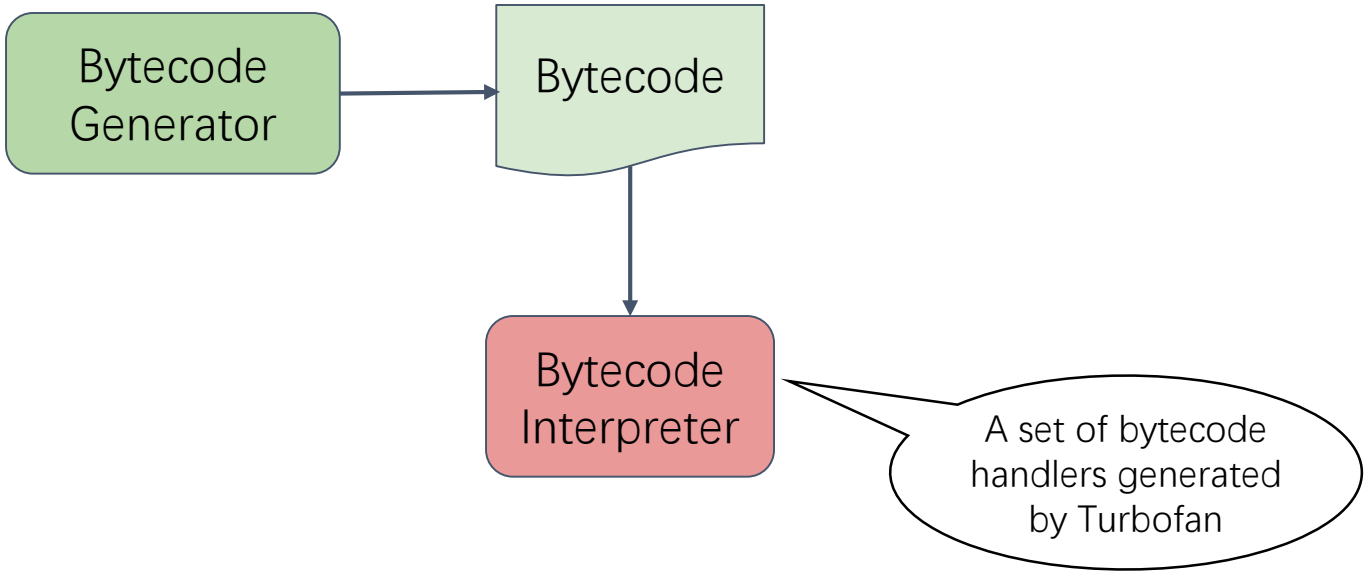
01 V8 compilation overview

Compilation job



02 Ignition

The name "**Ignition**" refers to both the **Bytecode Generator** and the **Bytecode Interpreter**.



Bytecode handlers: are written in a high level, machine architecture independent form of assembly code, as implemented by the CodeStubAssembler class and compiled by Turbofan.

03 Turbofan

Turbofan is the V8 optimizing compiler, it accepts Ignition’ s intermediate bytecode and generates architecture-specific machine code from it. In other words, V8 compiles JavaScript directly to native machine code using just-in-time compilation before executing it.

03 Turbofan

Turbof IR

- JavaScript:
  - JSAdd
  - JSSubtract
  - JSMultiply
  - JSDivide
  - JSModulus
  - JSDivide
  - JSBitwiseOr
  - JSBitwiseAnd
  - JSBitwiseXor
  - JSShiftLeft
  - JSShiftRight
  - JSEqual
  - JSStrictEqual
  - JSToBoolean
  - JSToNumber
  - JSCall
- Intermediate:
  - NumberAdd
  - NumberSub
  - NumberMul
  - NumberDiv
  - NumberMod
  - NumberEqual
  - NumberLessThan
  - LoadField
  - StoreField
  - StringEqual
  - StringAdd
  - ChangeTaggedToInt32
- Machine:
  - Int32Add
  - Int32Sub
  - Int32Mul
  - Float64Add
  - Float64Sub
  - Float64Mul
  - Load
  - Store
  - Call
  - ConvertFloat64ToInt32

Image from: https://docs.google.com/presentation/d/1sOEF4MIF7LeO7uq-uThJSulJlTh--wgLeaVibsbb3tc/edit#slide=id.g5499b9c42_01170

04 V8 porting

mksnapshot & d8

- **mksnapshot**

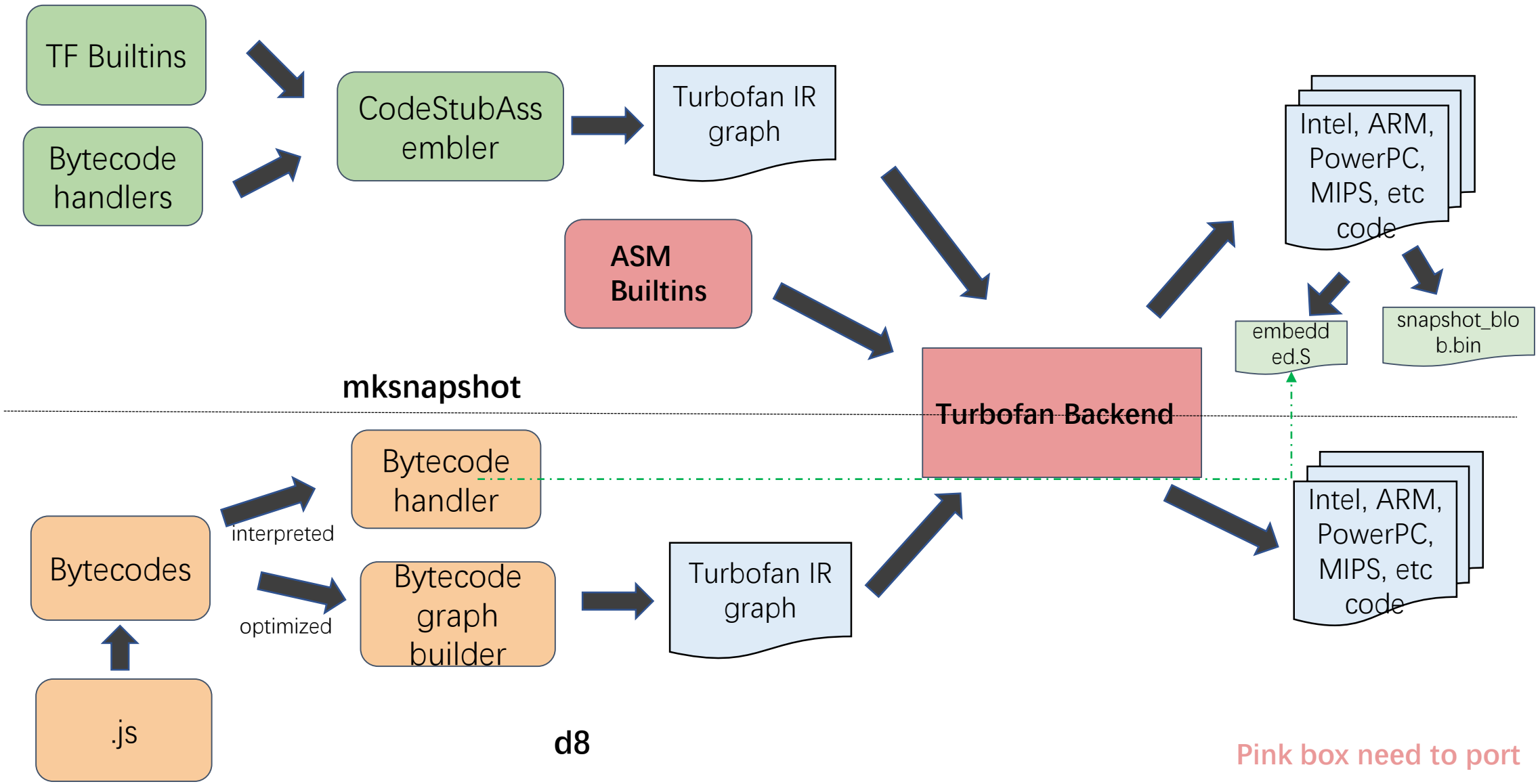
$./mksnapshot --turbo_instruction_scheduling --target_os=linux --target_arch=x64 --embedded_src gen/embedded.S --embedded_variant Default --random-seed 314159265 --startup_blob snapshot_blob.bin --native-code-counters --no-turbo-rewrite-far-jumps --no-turbo-verify-allocation --no-enable-slow-asserts --verify-heap

- **d8**

embedded.S->embedded.o ->./libv8.so

$../../third_party/llvm-build/Release+Asserts/bin/clang++ -pie ..... -o "./d8" --start-group @" ./d8.rsp" ./libv8.so ./libv8_libbase.so ./libv8_libplatform.so ./libicui18n.so ./libicuuc.so ./libc++.so –Wl,--end-group -ldl –lpthread –lrt -latomic

04 V8 porting



Pink box need to port

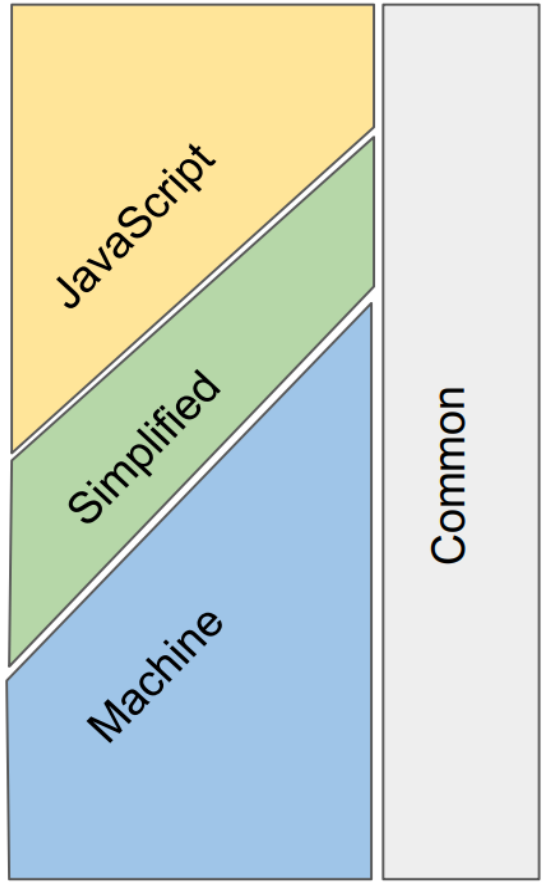
04 V8 porting

Main porting work

- **Turbofan Backend**
  - Instruction Selector
  - Code Generator
  - MacroAssembler/Assembler
  - Instruction scheduler
  - Register Allocation
- **ASM Builtins**
  - Implement ASM builtins

04 V8 porting

Instruction Selection



- Complex JavaScript at high-level（109个）
- Simplified in the middle（228个）
- Simple machine at low-level（399个）
- Common are shared. (63个)

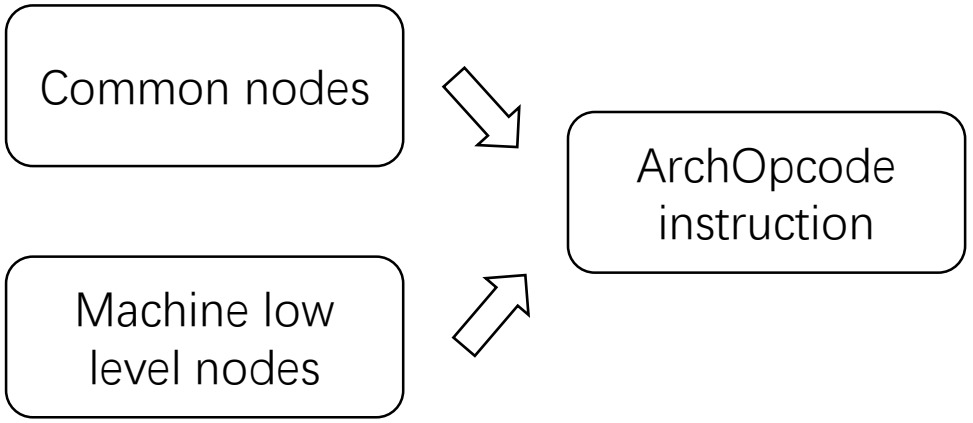
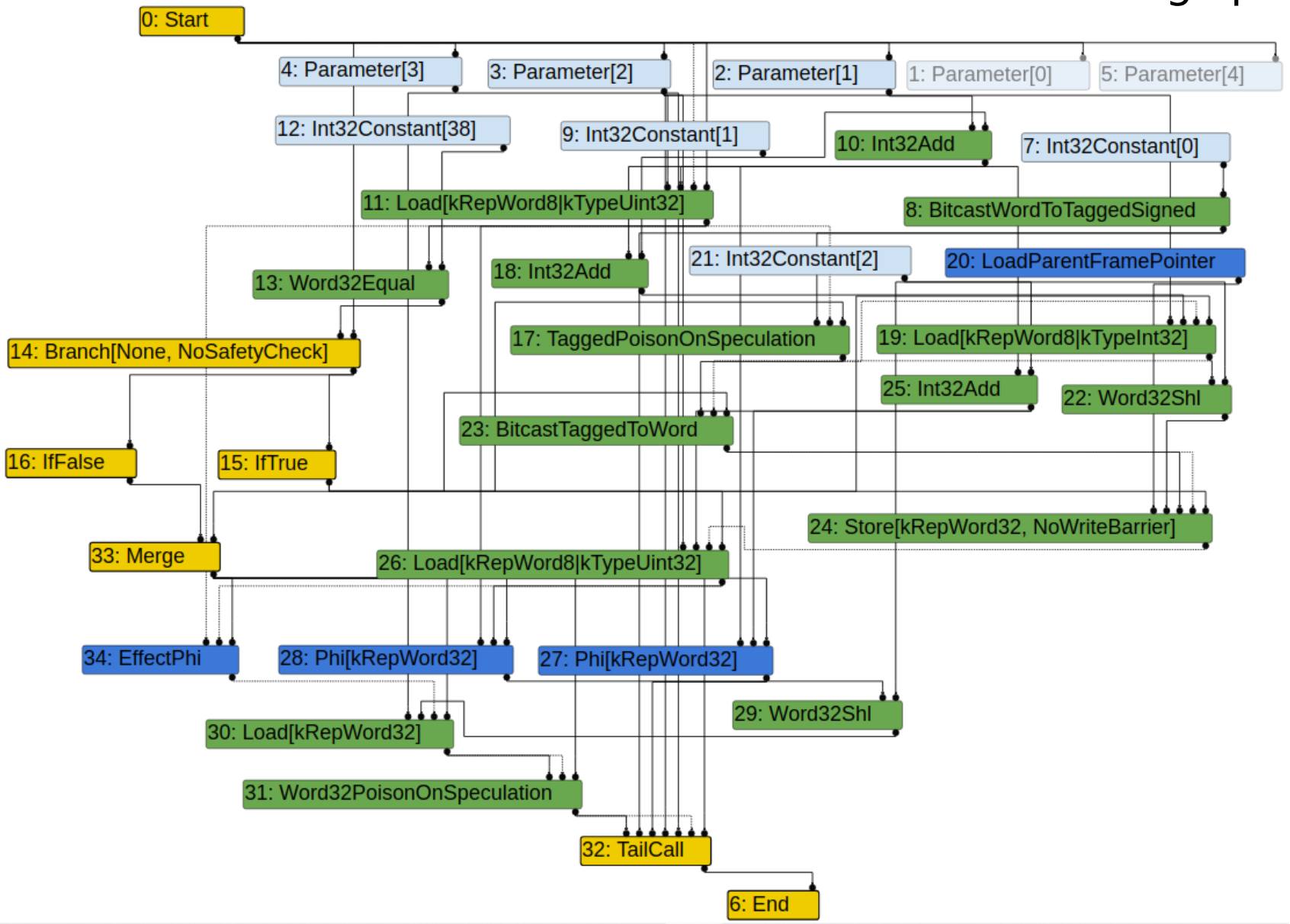


Image from: https://docs.google.com/presentation/d/1Z9iIHojKDrXvZ27gRX51UxHD-bKf1QcPzSijntpMJBM/edit#slide=id.g19134d40cb_0_400

04 V8 porting

Instruction Selection

LdaZero handler graph



04 V8 porting

Instruction Selection

```
Block B0
   0: Start
   4: Parameter[3](0)
   3: Parameter[2](0)
   2: Parameter[1](0)
   9: Int32Constant[1]
  10: Int32Add(2, 9)
  12: Int32Constant[38]
  11: Load[kRepWord8|kTypeUint32](3, 10, 0, 0)
  13: Word32Equal(11, 12)
   7: Int32Constant[0]
   8: BitcastWordToTaggedSigned(7)
  21: Int32Constant[2]
  14: Branch[None, NoSafetyCheck](13, 0)
  ➡ B2, B1
Block B1 - B0
  16: IfFalse(14)
  ➡ B3
Block B2 - B0
  15: IfTrue(14)
  17: TaggedPoisonOnSpeculation(8, 11, 15)
  18: Int32Add(10, 9)
  19: Load[kRepWord8|kTypeInt32](3, 18, 17, 15)
  23: BitcastTaggedToWord(17, 19, 15)
  22: Word32Shl(19, 21)
  20: LoadParentFramePointer
  24: Store[kRepWord32, NoWriteBarrier](20, 22, 23, 23, 15)
  25: Int32Add(10, 21)
  26: Load[kRepWord8|kTypeUint32](3, 25, 24, 15)
  ➡ B3
Block B3 - B1, B2
  33: Merge(16, 15)
  28: Phi[kRepWord32](11, 26, 33)
  34: EffectPhi(11, 26, 33)
  27: Phi[kRepWord32](10, 25, 33)
  29: Word32Shl(28, 21)
  30: Load[kRepWord32](4, 29, 34, 33)
  31: Word32PoisonOnSpeculation(30, 30, 33)
  32: TailCall[Addr:InterpreterDispatch Descriptor:r1s0i5f0](31, 8, 27, 3, 4, 31, 33)
  ➡ B4
Block B4 - B3
   6: End(32)
```

LdaZero blocks

04 V8 porting

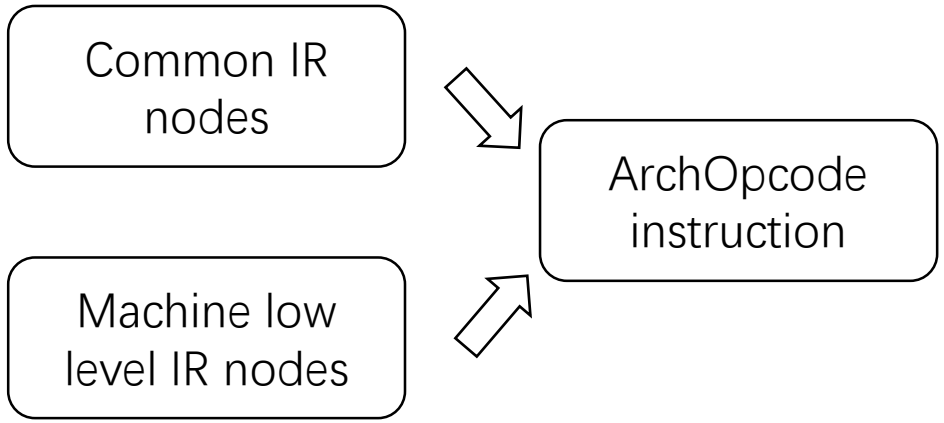
Instruction Selection

InstructionSelector::SelectInstructions

1. 通过BasicBlockVector* blocks = schedule()->rpo_order(); 得到所有的blocks，遍历每一个block，对于所有LoopHeader block, Mark the inputs of all phis as used。
2. Visit each basic block in post order，调用VisitBlock
3. Schedule the selected instructions

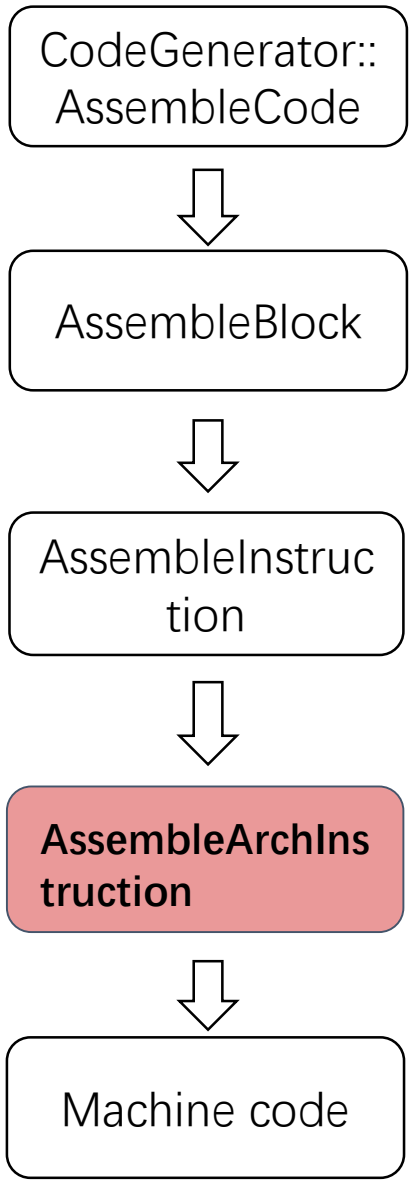
InstructionSelector::VisitBlock

1. 设置 current_block_ 及 current_block_end。
2. 为当前block里的每个node，设置effect level
3. 调用VisitControl，对control node做指令选择
4. 对每一个node调用VisitNode，做指令选择
5. 设置instruction block的code start/end.



04 V8 porting

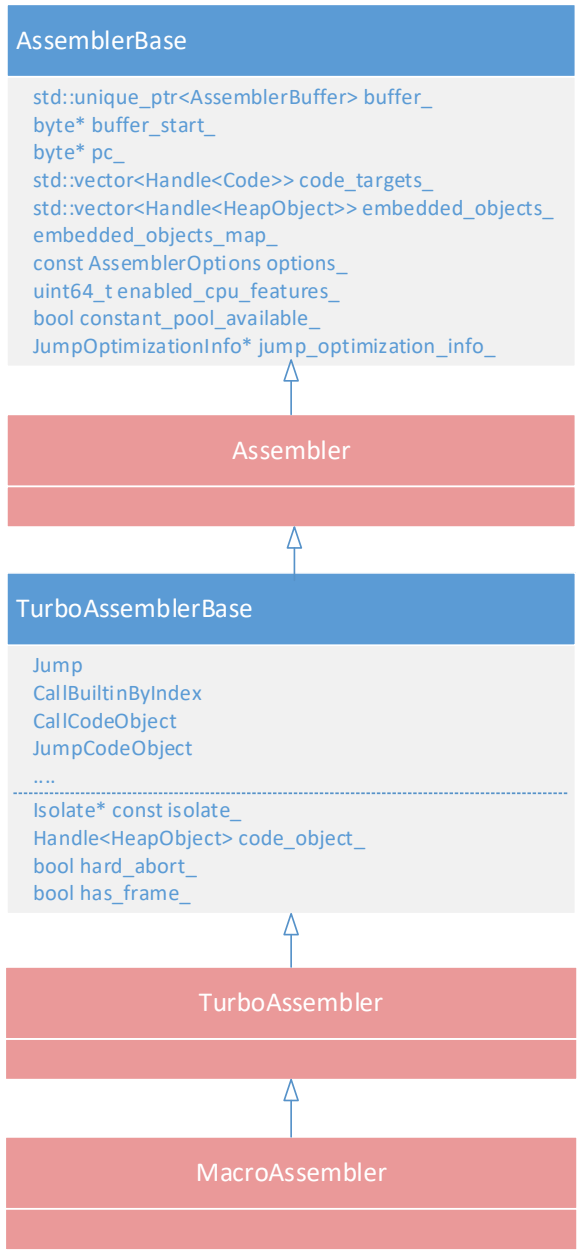
Code Generator



Member functions to be implemented in codeGenerator(26)
AssembleDeconstructFrame
AssemblePrepareTailCall
AssemblePopArgumentsAdaptorFrame
AssembleTailCallBeforeGap
AssembleTailCallAfterGap
AssembleCodeStartRegisterCheck
BailoutIfDeoptimized
GenerateSpeculationPoisonFromCodeStartRegister
AssembleRegisterArgumentPoisoning
AssembleArchInstruction
AssembleArchBranch
AssembleBranchPoisoning
AssembleArchDeoptBranch
AssembleArchJump
AssembleArchTrap
AssembleArchBoolean
AssembleArchBinarySearchSwitch
AssembleArchTableSwitch
FinishFrame
AssembleConstructFrame
AssembleReturn
FinishCode
PrepareForDeoptimizationExits
AssembleMove
AssembleSwap
AssembleJumpTable

04 V8 porting

MacroAssembler/Assembler



Implemented class
Assembler/TurboAssembler/MacroAssembler

04 V8 porting

ASM builtins (68个)

ASM: Builtin in platform-dependent assembly

Examples (from src/builtins/builtins-definitions.h):

name Interface descriptor

ASM(JSEntry, Dummy)

ASM(JSEntryTrampoline, JSTrampoline)

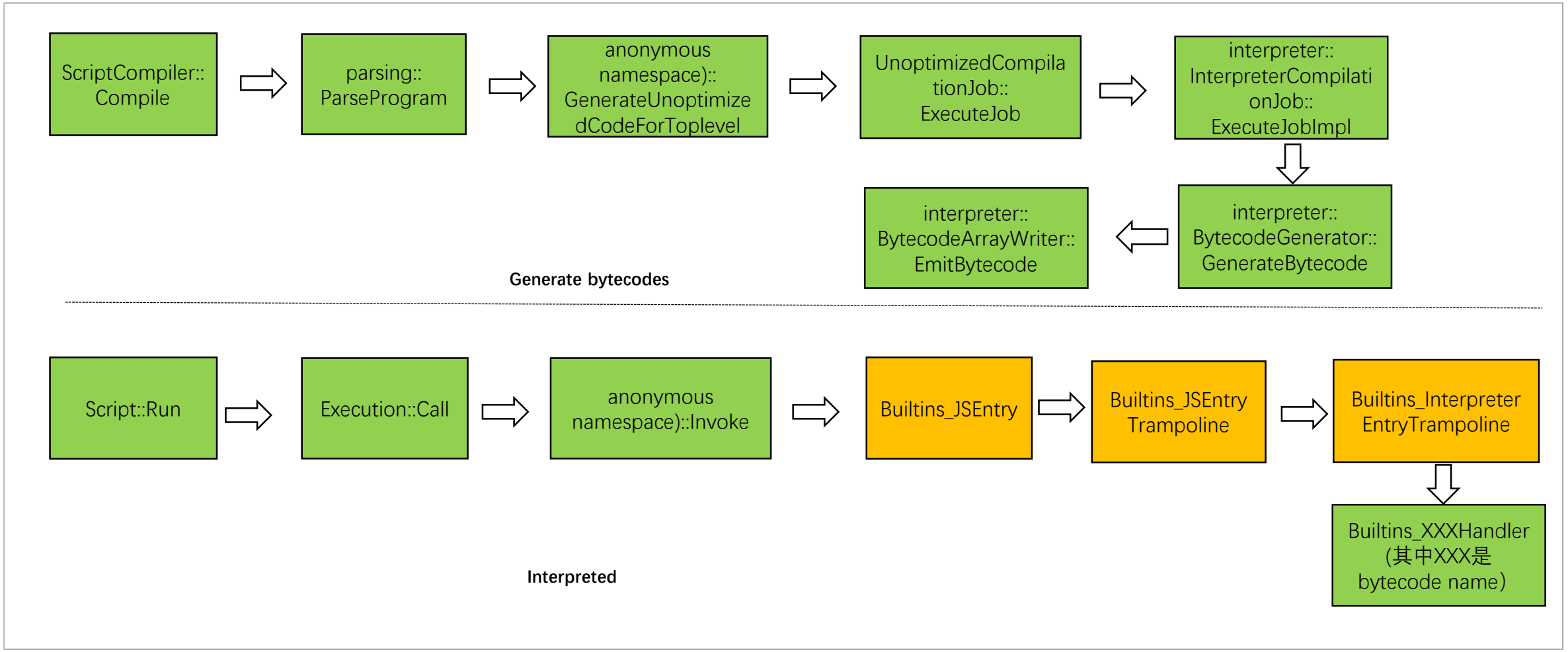
ASM(InterpreterEntryTrampoline, JSTrampoline)

```
code = BuildWithMacroAssembler(isolate, index, Builtins::Generate_JSEntry, "JSEntry");
AddBuiltin(builtins, index++, code);
code = BuildWithMacroAssembler(isolate, index, Builtins::Generate_JSEntryTrampoline, "JSEntryTrampoline");
AddBuiltin(builtins, index++, code);
code = BuildWithMacroAssembler(isolate, index, Builtins::Generate_InterpreterEntryTrampoline,
"InterpreterEntryTrampoline");
AddBuiltin(builtins, index++, code);
```

Implement in src/builtins/xxx /builtins-xxx.cc

04 V8 porting

D8 work flow



谢 谢

欢迎交流合作

2020/5/27