# Optimizing Chrome V8 Just-In-Time Compilation Based on RISC-V J and Customized Instruction Extension

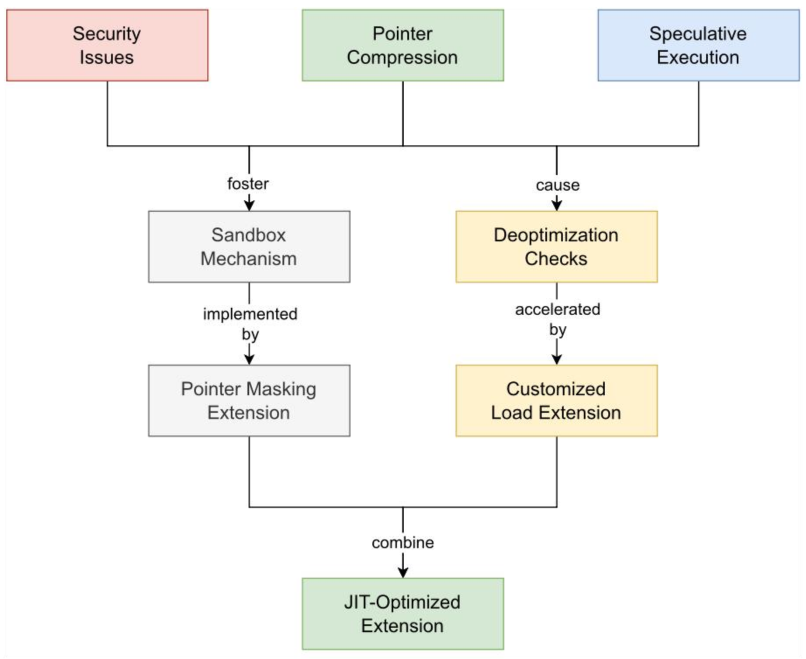
Qiaowen Yang and Zhangxi Tan*
RISC-V International Open-Source Laboratory,  TsingHua University

## OVERVIEW

This work proposes and implements an RISC-V extension to accelerate the Just-In-Time (JIT) compilation in the open-source Chrome V8 engine. The new extension is composed of the pointer masking specification from the RISC-V J extension and self-designed supplementary instructions tailored for V8's dynamic checks. Our results present the potential this extension shows in reducing instruction count and improving performance.

## BACKGROUND AND OBJECTIVES

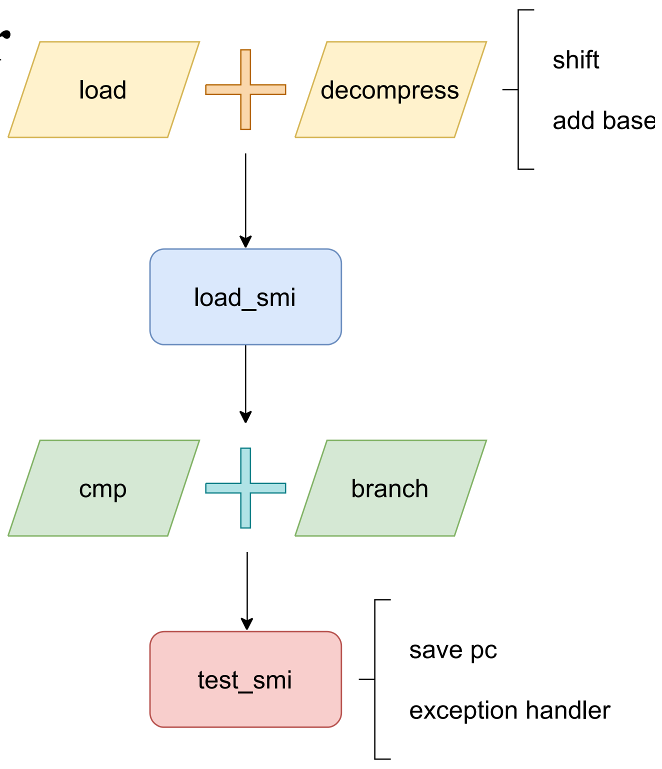
- **Dynamic checks in V8's JIT compilation can benefit from hardware-software co-design implementation**
  - Sandbox and pointer compression
  - Deoptimization checks from type speculation
- **Integrate part of RISC-V J extension to V8**
  - Our focus: pointer masking, which has been finalized
  - The J extension also includes other meaningful specifications
    - I/D consistency
    - Memory tagging: under active discussion



V8 JIT optimization roadmap

## DESIGN AND IMPLEMENTATION

- **Accelerate sandbox mechanism with pointer masking extension**
  - Hardware-assisted method: nearly agnostic to the software
  - We also explore the possibility of moving the whole sandbox mechanism to hardware

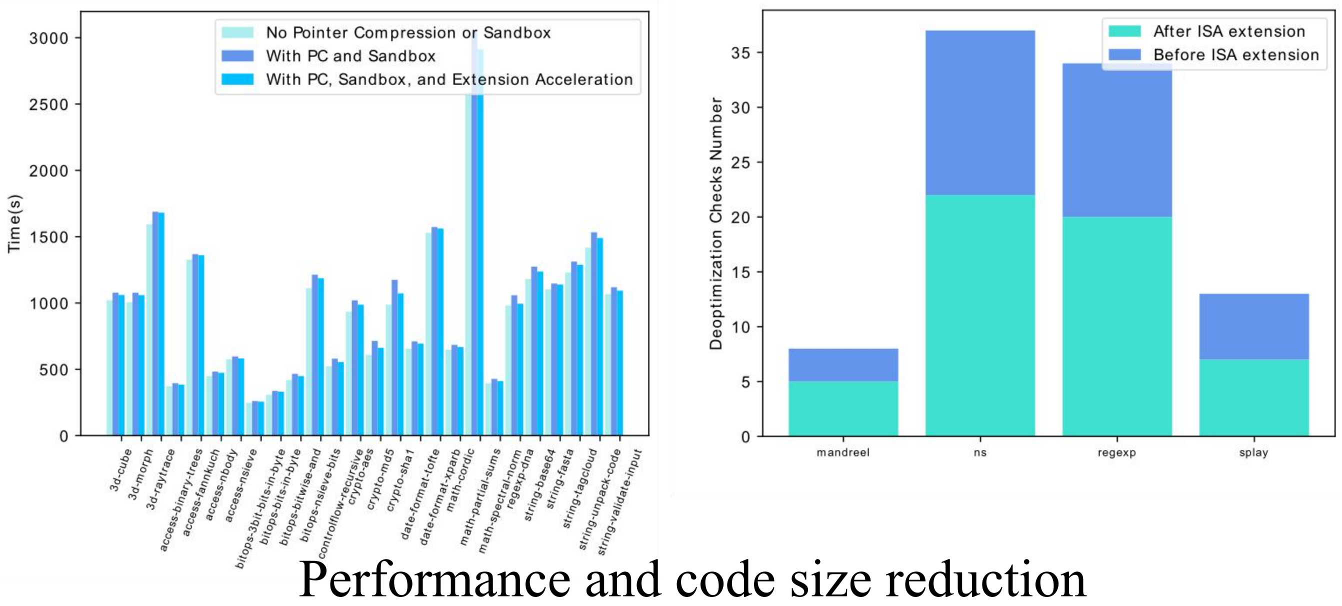


An example of new instructions

- **Optimize deoptimization checks using customized memory instructions**
  - Objectives: checks related to compressed values
- **Implementation**
  - Add nodes to V8's IR graph structure
    - For customized instructions, we need to merge nodes to generate new ones
  - Modify the code generation phase to make room for new instructions
  - Leverage pointer masking extension to control the access to untrusted pages

## RESULTS AND CONCLUSION

- **For pointer masking**
  - V8 by default disables the sandbox and pointer compression for RISC-V, which cause extra overhead
  - We enable them and focus on the performance improvement the pointer masking brings
- **For supplementary instructions**
  - We focus on the reduction of deopt checks generated by V8, and observe a 3% cut-down in the # of insts



Performance and code size reduction

- **Conclusion**
  - The pointer masking extension shows its potential in mitigating the overhead of V8's security mechanisms
  - The changes to V8 incorporating our customized instructions yield a satisfactory outcome in cutting down on the generated code size

* Corresponding author: xtan@rioslab.org
Contact me: yqw21@mails.tsinghua.edu.cn