

ON THE DESIGN OF A WEBASSEMBLY INTERPRETER FOR AVIATION

Wanja Zaeske, A. Cem Önem, Florian Hartung, Umut Durak



What is interpretation?



- Compile code to bytecode in virtual Instruction Set Architecture (ISA)
- Execute on virtual Central Processing Unit (CPU)
- Interpretation variants:
 - **Ad hoc**: translate bytecode per instruction
 - **Ahead-of-Time (AOT)**: compile bytecode to native machine code **before** execution
 - **Just-in-Time (JIT)**: compile bytecode to native machine code **during** execution

Execution of code not written in the CPU's native language, via an interpreter software

Why interpret?



- **Portability:**

- Compile once, run everywhere
- Decouple compilation from CPU ISA

- **Isolation:**

- Isolate host-environment resources per default
- Allow access only where needed

- **Observability:**

- Tap into any aspect of execution
- Without instrumentation

Why interpret in avionics?



■ Portability:

- Compile once, **certify once**, run everywhere?
- Decouple compilation from CPU ISA **which becomes obsolete over aircraft type life-cycle**
- **Generate certification evidence without the bottleneck of target hardware**

■ Observability:

- Tap into any aspect of execution **enabling advanced health-monitoring**
- **Measure statement-, decision-coverage or MC/DC**, without instrumentation
- **Measure (and mitigate?) the effect of multi-core interference**

What is interpreted bytecode in the avionics context?



1. Guess: It's a Parameter Data Item (PDI)

- Massive simplification
- Assurance becomes significantly cheaper
- But: "Programming instructions processed [by an interpreter][...] are **not** to be considered **data, but executable code**" [DO-332, OO.D.1.7.3]

What is interpreted bytecode in the avionics context?



1. Guess: It's a Parameter Data Item (PDI)

- Massive simplification
- Assurance becomes significantly cheaper
- But: "Programming instructions processed [by an interpreter][...] are **not** to be considered **data, but executable code**" [DO-332, OO.D.1.7.3]

2. Guess: It's object code

- All objectives for code assurance apply
- Limited benefit over compiling to native ISA

What is interpreted bytecode in the avionics context?



1. Guess: It's a Parameter Data Item (PDI)

- Massive simplification
- Assurance becomes significantly cheaper
- But: "Programming instructions processed [by an interpreter][...] are **not** to be considered **data, but executable code**" [DO-332, OO.D.1.7.3]

2. Guess: It's object code

- All objectives for code assurance apply
- Limited benefit over compiling to native ISA

3. Guess: It's Executable Object Code (EOC)

- All objectives for code assurance apply
- But: EOC related evidence remains intact upon ISA change

What is interpreted bytecode in the avionics context?



1. Guess: It's a Parameter Data Item (PDI)

- Massive simplification
- Assurance becomes significantly cheaper
- But: "Programming instructions processed [by an interpreter][...] are **not** to be considered **data, but executable code**" [DO-332, OO.D.1.7.3]

2. Guess: It's object code

- All objectives for code assurance apply
- Limited benefit over compiling to native ISA

3. **Assumption: It's Executable Object Code (EOC)**

- All objectives for code assurance apply
- But: EOC related evidence remains intact upon ISA change

What virtual ISA to choose?



- Available options:
 - Java Virtual Machine (JVM): Java et al.
 - Common Language Runtime (CLR): C# et al.
 - ...
 - **WebAssembly (Wasm)**

What virtual ISA to choose?



- Available options:
 - Java Virtual Machine (JVM): Java et al.
 - Common Language Runtime (CLR): C# et al.
 - ...
 - **WebAssembly (Wasm)**
- Why Wasm?
 - Language and OS agnostic
 - No Garbage Collector (GC)
 - Compact implementation possible
 - Formal specification available
 - Harvard architecture, close resemblance of real hardware

Requirements for interpretation in avionics — 30 000 feet view



- **REQ-1:** Certification of the Wasm interpreter shall be feasible
- **REQ-1.1:** Resource usage at runtime shall be deterministic
- **REQ-1.2:** The total code-closure of the interpreter shall be less than 100 kLOC

- **REQ-1:** Certification of the Wasm interpreter shall be feasible
- **REQ-1.1:** Resource usage at runtime shall be deterministic
- **REQ-1.2:** The total code-closure of the interpreter shall be less than 100 kLOC
- **REQ-2:** Certification evidence reuse for interpreted applications shall be maximized
- **REQ-2.1:** Wasm bytecode shall directly be interpreted as EOC

- **REQ-1:** Certification of the Wasm interpreter shall be feasible
- **REQ-1.1:** Resource usage at runtime shall be deterministic
- **REQ-1.2:** The total code-closure of the interpreter shall be less than 100 kLOC
- **REQ-2:** Certification evidence reuse for interpreted applications shall be maximized
- **REQ-2.1:** Wasm bytecode shall directly be interpreted as EOC
- **REQ-3:** The Wasm interpreter shall be implemented independent of additional core software

No-Go checklist for interpreter in avionics



- **Huge code-base** [REQ-1.2]
- **Require AOT compilation** [REQ-2.1]
- **Require JIT compilation** [REQ-2.1, REQ-1.1]
- **Interpretation of custom representation/IR** [REQ-2.1]
- **Implementation in esoteric languages** [REQ-1]
- **Implementation in a garbage-collected language** [REQ-1.1]
- **Reliance on syscalls** [REQ-3]
- **Use of recursion** [REQ-1]

Interpreter comparison



- Many Wasm interpreters exist
- Checked the nine most comprehensive interpreters¹
- Each fails at least two checks
- **None tick all boxes**

Wasm Interpreter	Huge Code-base	Require AOT	Require JIT	Interpret IR	Esoteric Language	GC Language	Relies on Syscalls	Relies on Recursion	Has Dependencies
wasmtime (v31.0.0)	310 kLOC	!	!	!	✓	✓	!	✓	!
wasmtime (v31.0.0 - minimal build)	47 kLOC	!	✓	✓	✓	✓	✓	✓	!
wasmi (v.0.44.0)	45 kLOC	✓	✓	!	✓	✓	✓	✓	✓
wizard (commit #3cced75 - JVM)	26 kLOC	✓	✓	✓	!	!	✓	✓	!
wizard (commit #3cced75 - x86_64-linux)	34 kLOC	✓	✓	✓	!	!	!	✓	✓
Reference Interpreter (commit #05949f5)	11 kLOC	✓	✓	✓	✓	!	✓	!	✓
wasmer (v5.0.4)	193 kLOC	!	!	✓	✓	✓	!	✓	!
WAMR (v2.2.0 - classic interpreter mode)	92 kLOC	✓	!	✓	✓	✓	!	!	✓
wasm3 (v0.5.0)	10 kLOC	✓	✓	!	✓	✓	✓	!	✓
WasmEdge (v0.14.1)	36 kLOC	!	!	✓	✓	✓	!	✓	!
WAVM (nightly/2022-05-14)	32 kLOC	!	!	✓	✓	✓	!	✓	!

¹ignoring those shipped as component of a web browser

...develop our own Wasm interpreter!

- Tailored to avionics use-case
- Written in Rust
- Licensed under Apache 2.0 / MIT
- 9 kLOC implementation + 23 kLOC of own tests
- Passes official Wasm test-suite² (51737 asserts over 146 test sets)
- $\approx$ 92 % Line coverage
- $\approx$ 89 % Branch coverage

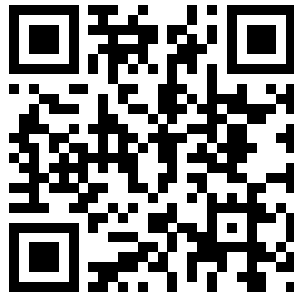
²Except for SIMD instructions and unfinished proposals

- Certification evidence reuse possible . . .
... if interpreted bytecode equals EOC
- Status quo from standards:
 1. DO-332 says: (byte-)code interpreted by a virtual machine **is code**, not data [OO.D.1.7]
 2. “EOC consists of a form of code that is **directly usable** by the **processing unit** of the **target computer**” [DO-178C, 11.12]
 3. “The term **target computer** can be interpreted to **mean target environment**. The target environment is either a target computer or a **combination** of **virtualization software** and a **target computer**” [DO-332, OO.4.0]

- Certification evidence reuse possible . . .
... if interpreted bytecode equals EOC
- Status quo from standards:
 1. DO-332 says: (byte-)code interpreted by a virtual machine **is code**, not data [OO.D.1.7]
 2. “EOC consists of a form of code that is **directly usable** by the **processing unit** of the **target computer**” [DO-178C, 11.12]
 3. “The term **target computer** can be interpreted to **mean target environment**. The target environment is either a target computer or a **combination** of **virtualization software** and a **target computer**” [DO-332, OO.4.0]
- Problem: target computer $\neq$ target computer's processing unit

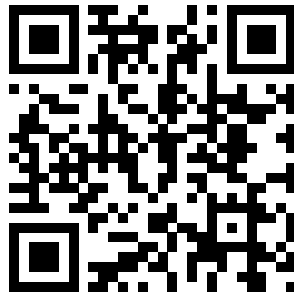
- Certification evidence reuse possible . . .
... if interpreted bytecode equals EOC
- Status quo from standards:
 1. DO-332 says: (byte-)code interpreted by a virtual machine **is code**, not data [OO.D.1.7]
 2. “EOC consists of a form of code that is **directly usable** by the **processing unit** of the **target computer**” [DO-178C, 11.12]
 3. “The term **target computer** can be interpreted to **mean target environment**. The target environment is either a target computer or a **combination** of **virtualization software** and a **target computer**” [DO-332, OO.4.0]
- Problem: target computer $\neq$ target computer's processing unit
- Question: virtualization software $\subseteq$ target computer's processing unit?

- Avionics can benefit from interpretation
 - Especially: certification evidence reuse
But: clarification of certification-authorities needed
- Common interpreter techniques conflict with certification
 - Existing Wasm interpreters not perfect for avionics
 - Least compromise: WAMR & wasmi
 - Our interpreter (QR-Code) aims to fit the niche
- Future work:
 - Wasm and ARINC 653?



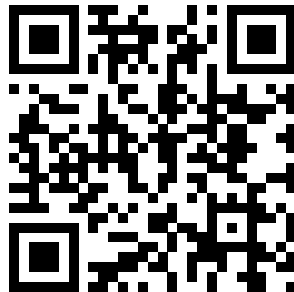
<https://github.com/DLR-FT/wasm-interpreter>

- Avionics can benefit from interpretation
 - Especially: certification evidence reuse
But: clarification of certification-authorities needed
- Common interpreter techniques conflict with certification
 - Existing Wasm interpreters not perfect for avionics
 - Least compromise: WAMR & wasmi
 - Our interpreter (QR-Code) aims to fit the niche
- Future work:
 - Wasm and ARINC 653?
 - Multi-core interference mitigation via Wasm fuel?



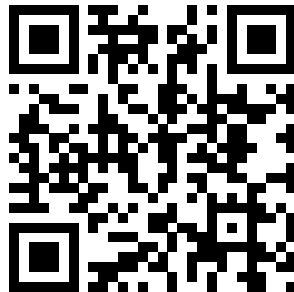
<https://github.com/DLR-FT/wasm-interpreter>

- Avionics can benefit from interpretation
 - Especially: certification evidence reuse
But: clarification of certification-authorities needed
- Common interpreter techniques conflict with certification
 - Existing Wasm interpreters not perfect for avionics
 - Least compromise: WAMR & wasmi
 - Our interpreter (QR-Code) aims to fit the niche
- Future work:
 - Wasm and ARINC 653?
 - Multi-core interference mitigation via Wasm fuel?
 - Coverage measurement via Wasm Interpreter?



<https://github.com/DLR-FT/wasm-interpreter>

- Avionics can benefit from interpretation
 - Especially: certification evidence reuse
But: clarification of certification-authorities needed
- Common interpreter techniques conflict with certification
 - Existing Wasm interpreters not perfect for avionics
 - Least compromise: WAMR & wasmi
 - Our interpreter (QR-Code) aims to fit the niche
- Future work:
 - Wasm and ARINC 653?
 - Multi-core interference mitigation via Wasm fuel?
 - Coverage measurement via Wasm Interpreter?
 - Static analysis of Wasm bytecode?



<https://github.com/DLR-FT/wasm-interpreter>

Topic: **On the Design of a WebAssembly Interpreter for Aviation**

Date: 17th of September, 2025

Author: Wanja Zaeske, A. Cem Önem, Florian Hartung, Umut Durak

Institute: Institute of Flight Systems

Credits: All images „DLR (CC BY-NC-ND 3.0)“