WEBASSEMBLY IN AVIONICS

Decoupling Software from Hardware

Current state

Aircraft have long lifecycles

- Established: **Retrofitting** modern avionics
- Possible: **Porting** existing software to new hardware
  - Common API like ARINC 653 helps!

But: Certification bases on EOC

- Change of hardware/hypervisor requires recompilation
  - Generates new EOC
  - Invalidates existing certification evidence

Goals

Enabling

- Cheaper re-certification of existing software
  - Reduces effort on hardware upgrade
- Re-use of certified EOC as far as possible
  - Leaner certification process

Executable **O**bject **C**ode:

Binary machine code that is directly executable on target system processor.
For example: statically linked, position independent object code.

Dimensions of EOC Deliverables

- Version: v.0.1.0, v1.2.3,
- Target architecture: PowerPC, Aarch64,

Lets get rid of the second dimension!

Software Lifecycle

1. Compile software **once** to EOC
2. Assure and certify **functional behavior** on EOC
3. Use EOC on **any** OS and hardware
4. Remaining certification effort focused on integration
5. After hardware upgrade, functional behavior remains certified

How to get there?

Recipe

Decoupling EOC from hardware requires:

- Common instruction set
  - Achieves: independence from CPU architecture
  - Means: Emulation/Interpretation
- Agreement on a common ABI
  - Allows: two pieces of EOC work together
  - Means: industry-wide agreement

Application **P**rogramming **I**nterface

Software interface on source code level

- Header files with type & function declaration
- Virtual base class in CPP
- Traits in Rust

Application **B**inary **I**nterface

Software interfaces on binary level

- Semantics of syscalls
- In-memory representation of data-types
- Convention on providing arguments and return address to function call

Overview

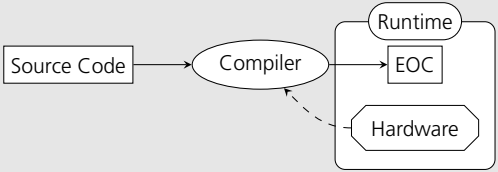
- Virtual machine, stack based
- Interpreter runs on any machine
  - From microcontroller to server
  - From bare-metal to full desktop OS
- First release in 2017
- Widely adopted
- Made for sandboxing
  - Memory isolation without MMU/MPU
  - Temporal isolation via instruction counting
- Open Standard, many non-proprietary implementation
- Formally specified

Example

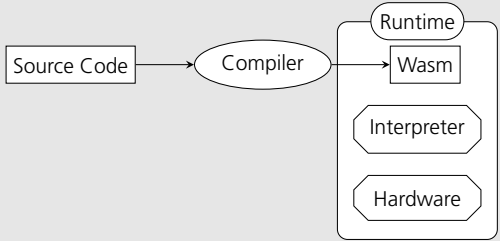
```
(func (result i32)
  i32.const 6
  i32.const 7
  i32.add
)
```

Compilation/Execution of EOC and WebAssembly

Deployment with EOC



Deployment with WebAssembly



Currently: EOC in Partition

- EOC is executed directly in a partition
- No notable runtime overhead

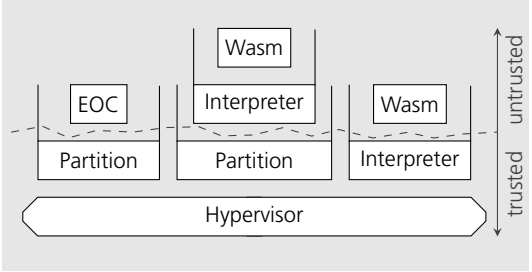
Scenario 1: Interpreter in Partition

- WebAssembly interpreter inside a partition
- Interpreter creates compatibility
- Partition enforces isolation

Scenario 2: Interpreter as Partition

- WebAssembly interpreter instead of partition
- Interpreter creates compatibility
- Isolation by WebAssembly sandbox

Integration Scenarios



Impact on performance

- Runtime slow-down
  - Interpretation $\approx 22\times$
  - AOT & JIT $\approx$ 10 to 50%
- Memory
  - WebAssembly modules are $\approx 2.2\times$ the size of EOC
  - In addition, the EOC of the interpreter comes on top **once**
  - Only 4 GiB addressable

Takeaway Points

- Pure interpretation is very slow compared to EOC
- With AOT or JIT, the overhead is negligible
- Memory consumption is increased moderately

How to certify WebAssembly?

- DO-178C: WebAssembly is
  - **Code**: computer program in symbolic form
  - **not EOC**: not directly executable on CPU
  - *Parameter Data Item*: changes EOC behavior?
- DO-297 clarifies:
  - Interpreter input shall be treated like EOC

Noteworthy Points

- DO-178C:
  - **allows emulation** for **certification credit**
- DO-297: interpretation is emulation
  - Thus, interpreter can generate evidence
  - on the **development** machine

Open Questions

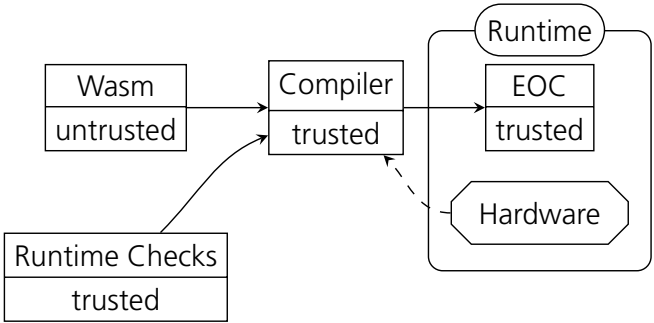
- How to certify AOT/JIT compilation?
  - Standards are unclear
  - **All** EOC is subject to **DO-178C objectives**
  - **Modifying EOC** on the target platform is **critical**
- Can instruction-counting enhance/provide temporal isolation?

Conclusion

- WebAssembly enable cross-platform ABI compatibility
- Integration to existing ARINC 653 hypervisors is feasible
- WebAssembly comes with temporal and spatial isolation
  - Partitioning without hypervisor may be possible
- DO-178C + DO-297 define certification
- Interpretation is slow
  - AOT/JIT are desirable

Bonus: WebAssembly to EOC

- WebAssembly can be transpiled to EOC
  - Formal proven correctness
  - Formal proven sandbox properties
  - Near to native performance
  - No need for Interpreter



Imprint

Topic:	**WebAssembly in Avionics** Decoupling Software from Hardware
Date:	2023-10-3 Copyright l' 2023 by Deutsches Zentrum für Luft- und Raumfahrt e. V.
Author:	Wanja Zaeske
Institute:	Institute of Flight Systems
Credits:	All images DLR (CC BY-NC-ND 3.0)