WebAssembly を理解する
～ VM の作成を通して～

nsfisis (いまむら)

PHPerKaigi 2024

自己紹介

nsfisis (いまむら) 

@ デジタルサーカス株式会社

今年のコードゴルフ企画のシステム開発・運用

WebAssembly の概要

WebAssembly とは

WebAssembly (Wasm)

ブラウザなどで実行できる
ポータブルな仮想命令セット

WebAssembly の特徴

ポータブル
速い
安全

WebAssembly の出自

元々のモチベーション : ブラウザ上での高速な処理

動的な JavaScript だと限界がある

間にいくつかの技術が生まれたり消えたりし、
最終的に WebAssembly が策定された

Emscripten

Emscripten

C/C++ のソースコードを Wasm に変換

C/C++ で書かれた膨大な資産を
ブラウザの上で動かせる

Wasm の活用例

PHP の処理系は C で書かれている

Emscripten を使って Wasm に変換できる

Wasm に変換するとブラウザ上で動かせる

例 : PHP Playground、 コードゴルフ企画

WebAssembly の特徴

ポータブル

速い

安全

これらの特徴はどのように実現されているのか

処理系を自作して理解しよう

作成したものの紹介

完成したもの

```
>>> php -d memory_limit=4G -d opcache.enable_cli=on -d
opcache.jit=on -d opcache.jit_buffer_size=1G examples/p
hp-on-wasm/php-wasm.php
Decoding...
Instantiating...
Executing...
Hello, World!


Exit code: 0
```

完成したもの

今回作った Wasm 処理系

の上に、 Wasm に変換された PHP 処理系

の上に、 `echo "Hello, World!\n";`

完成したもの

普通の PHP 処理系

の上に、 今回作った Wasm 処理系

の上に、 Wasm に変換された PHP 処理系

の上に、 `echo "Hello, World!\n";`

完成したもの

普通の PHP 処理系

の上に、 今回作った Wasm 処理系

の上に、 Wasm に変換された PHP 処理系

の上に、 `echo "Hello, World!\n";`

多段になりすぎて実行に 30 秒かかる

Wasm の処理系を作る

仕様書

成長途中の規格
バージョンがいろいろある

バージョン 1
バージョン 2
WasmGC

仕様書の構成

Structure

Validation

Execution

Binary Format

Text Format

Binary Format と Text Format

バイナリ形式のプログラム表現と、
テキスト形式のプログラム表現の 2 種類がある

同じ意味のプログラムを別々の書き方で表せる
両形式は、　相互に変換可能

どちらか一方だけ実装してやればよい

どの順に読むべきか？

Structure
Validation
Execution
Binary Format
Text Format

どの順に読むべきか？

Structure (1)

Validation (4)

Execution (3)

Binary Format (2)

Text Format (2)

Structure

Wasm のプログラムを表すデータ構造を定義

命令セットの定義、 各種プリミティブ型の定義など

これを読みながら、 データ型を定義していく

Binary Format / Text Format

Wasm のプログラムがどのような
バイナリ / ソースコードで表されるかを定義

Structure で定義されたデータ構造へと変換していく

Binary Format のほうが楽そう？

Execution

Wasm をどう実行するか (VM 本体)

仕様書を忠実に翻訳するだけでは実装できない

Structured Control Flow/Label 周りが厄介

Structured Control Flow の扱い

仕様書の記述をそのまま実装するのは困難

(A) block/loop/if を直列に展開

(B) 今回の方法

Structured Control Flow の扱い : br 命令

br 命令

block 命令中では break 相当

loop 命令中では continue 相当

continuation の話は無視していい

Structured Control Flow の扱い : br 命令

br 命令

仕様書にある br 命令の処理の内容は

br 命令を実行する箇所ではなく

呼び出し元となる block と loop に書く

Validation

Wasm のプログラムが正しいかどうか
実行前に検証する

Wasm の安全性を担保している処理の一つ

Validation

Wasm のプログラムが正しいかどうか
実行前に検証する

Wasm の安全性を担保している処理の一つ

やらなくていい

世にある Wasm のプログラムは
Validation が通るものばかり

外界とのやりとり

Wasm 自体の仕様には
外の世界とやりとりする手段がない

例 : 入出力、 ファイル操作、 ネットワークアクセス等

必要に応じて明示的にインポートする

これによって安全性が高まっている

まとめ

1. データ構造を定義
2. Binary format のデコーダを実装
3. VM を実装
4. 外界とのインターフェースを実装
5. 頑張る

注目が集まるわけ

WebAssembly の特徴

ポータブル
速い
安全

WebAssembly の特徴 : ポータブル

Wasm 自体の仕様には
外の世界とやりとりする手段がない

特定の環境に依存しない

環境依存の処理は Wasm の外でおこなう

WebAssembly の特徴 : ポータブル

Web に依存しない

JavaScript や DOM 等の技術から独立している

ブラウザの外でも動く

WebAssembly の特徴 : 速い

バイナリ形式 : パースが速い

十分に低レベルな命令セット

WebAssembly の特徴 : 安全

検証しやすい (validation)

外界とのやりとりが明示的

サンドボックス化が容易

活用事例

ブラウザ上での高速な処理

C/C++ のソフトウェアをブラウザへ移植

ソフトウェアのプラグイン

コンテナ

活用事例

ブラウザ上での高速な処理

C/C++ のソフトウェアをブラウザへ移植

ソフトウェアのプラグイン

コンテナ

Web 以外の領域でも注目されている

言語処理系を作る

なぜ作るのか？

車輪の再発明

世に優れた Wasm の処理系はごまんとある

なぜ作るのか？

なぜ？

なぜ作るのか？

なぜ？

楽しい

言語処理系の面白さ

あるとき突然すべてが上手く動くようになる

自分の理解を遥かに超えたものがなぜか動く