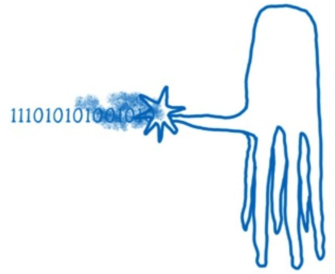
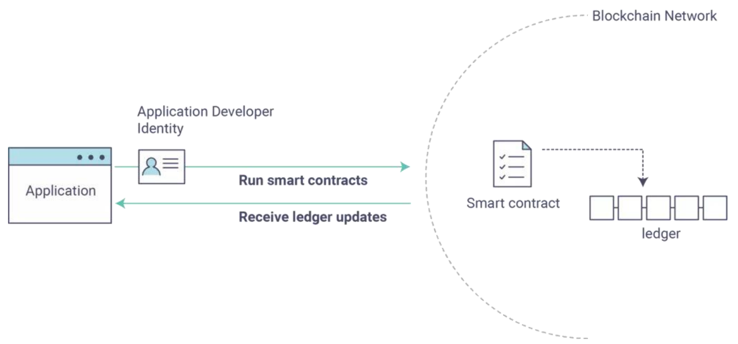
Introduction to Wasm Virtual Mechine && Block chain Rust Smart Contract

fospring





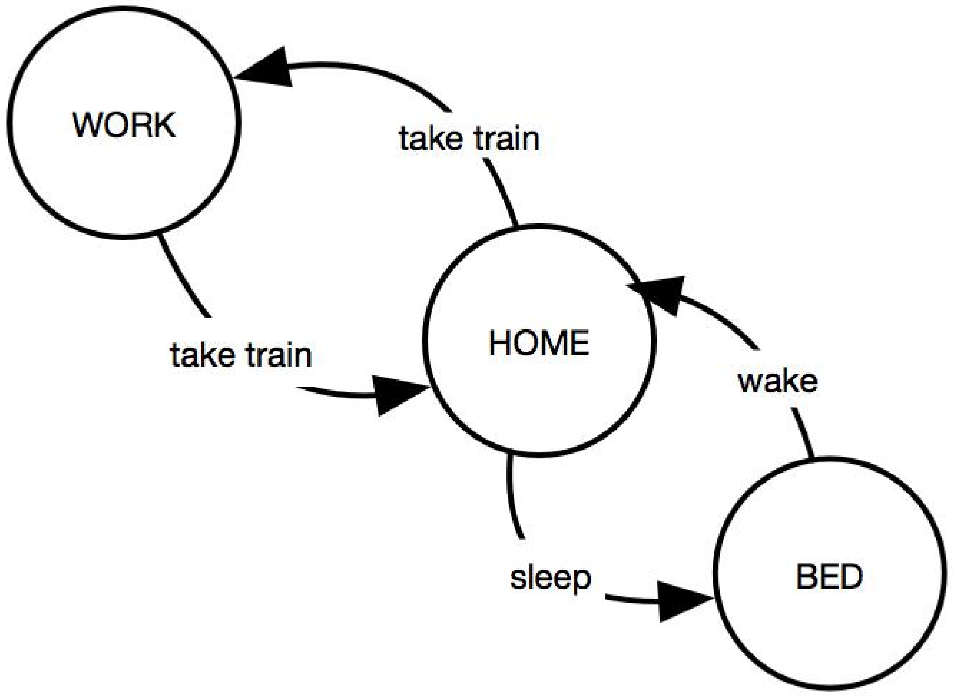
Outline

- Motivation(动机)
- WebAssembly(汇编)
- Why Wasm VM(选择)
- Communicate with machine(计算)
- Wasm Core specification(标准)
- Deploy && System check(部署)
- Block Chain Context && System call(调用)
- Toolchain (适配)
- Compile Options(编译)
- Binaryen Tool(工具)
- Utils && Resource

Motivation(动机)

- * Consensus
- Business Code For Users
  - execute buisiness code
  - store data incontract
- Consistency For Block State Set
  - State Machine

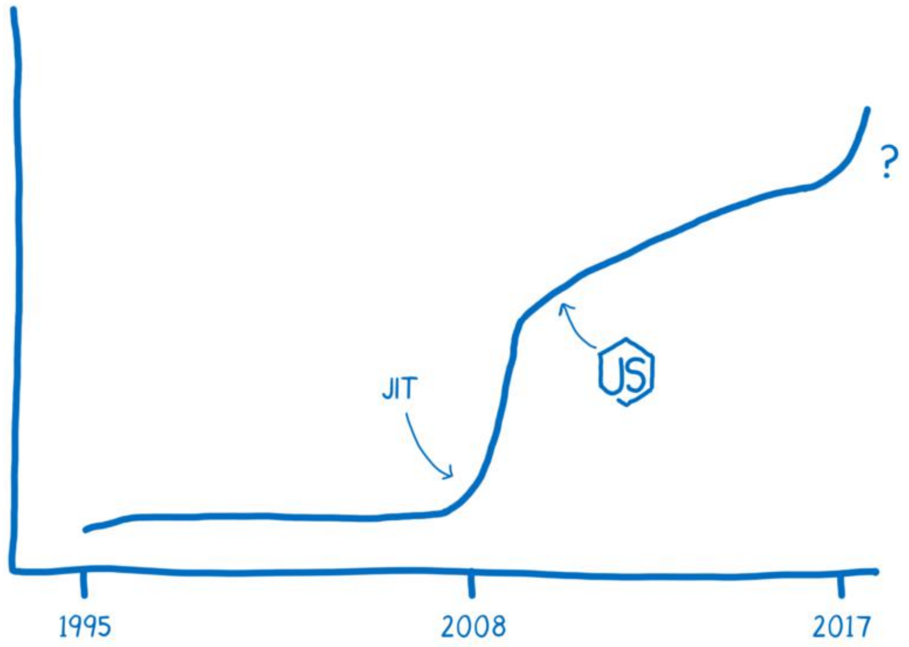
    deterministic — whenever an output is triggered from the same initial state and the same input, that output will always be the same.
  - *floating point number calculate
  - *random factor
- Safety Sandbox For Node



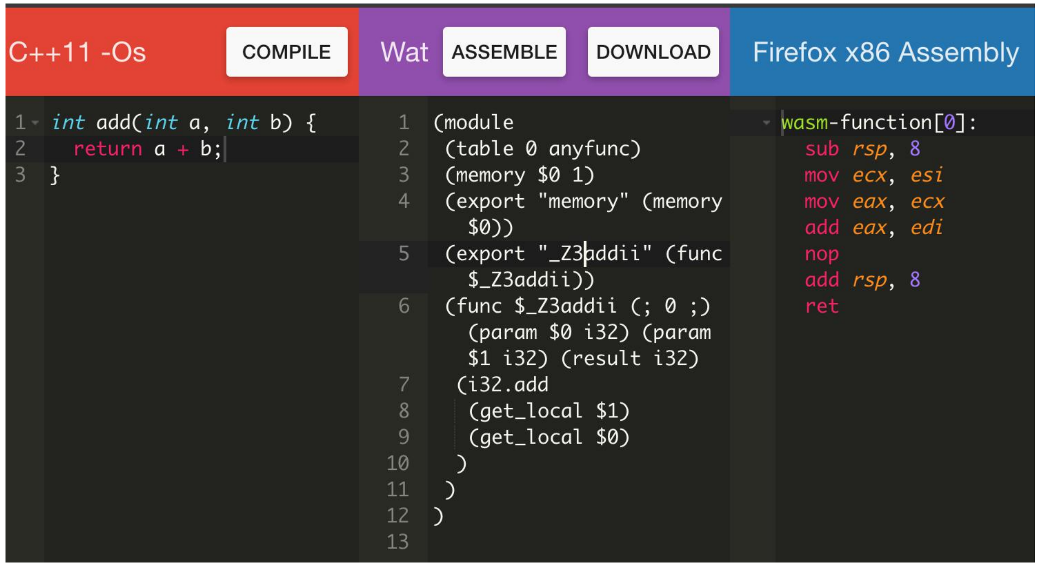
state machine

WebAssembly(汇编)

A little performance history



WebAssembly is a way of taking code written in programming languages other than JavaScript and running that code in the browser.



```
int add(int a, int b) {
  return a + b;
}
```

```
(module
 (table 0 anyfunc)
 (memory $0 1)
 (export "memory" (memory
   $0))
 (export "_Z3addii" (func
   $_Z3addii))
 (func $_Z3addii (; 0 ;)
   (param $0 i32) (param
   $1 i32) (result i32)
  (i32.add
   (get_local $1)
   (get_local $0)
  )
 )
)
```

```
wasm-function[0]:
  sub rsp, 8
  mov ecx, esi
  mov eax, ecx
  add eax, edi
  nop
  add rsp, 8
  ret
```

Why Wasm VM （选择）

- Hyperledger Fabric 1.0 VM
  - chaincode
    - a docker containerScenes for alliance blockchian
  - Undeterministic factor：
    - float point calculation
    - local time and other random refector
- EVM
  - stack machine
  - operation base on 256bit integer
    - not efficiency， because most cpu and 64-bit and 32-bit,friendly for:
      - 8bit/16bit/32bit/64bit operand
  - lack of standar library
  - difficult to test and debug

- Eos Virtual Machine
  - wasm virtual machine
  - support float calculate
    - bacuase all super nodes use same type handware

Why wasm VM （选择）

- Turing completeness
  - All computational problem can be solved
  - Computational Problem：
    - representing a collection of questions that computers might be able to solve.

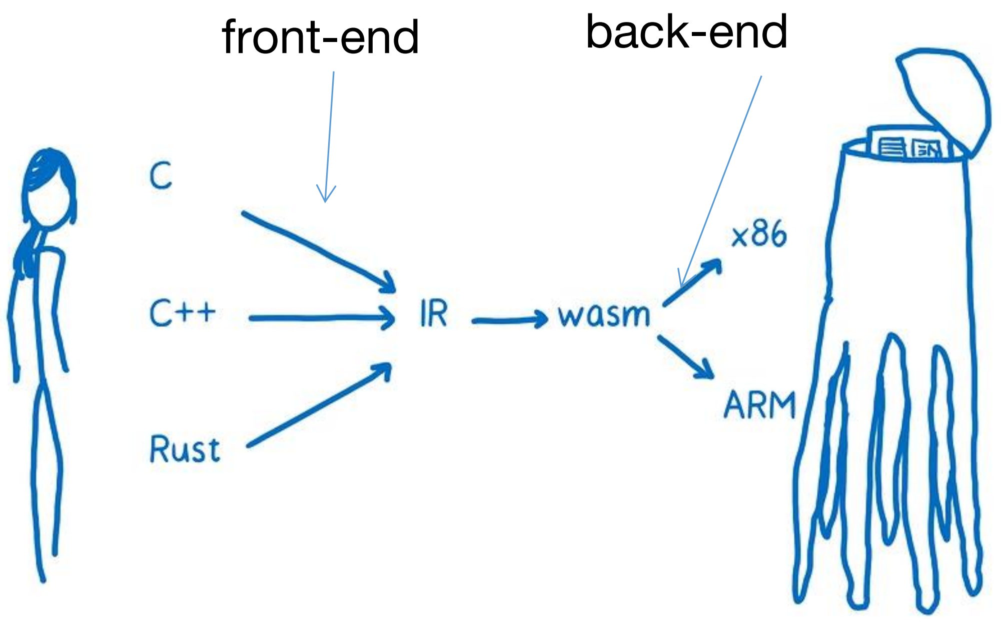
  a instruct sets： include jump(conditional and unconditional)

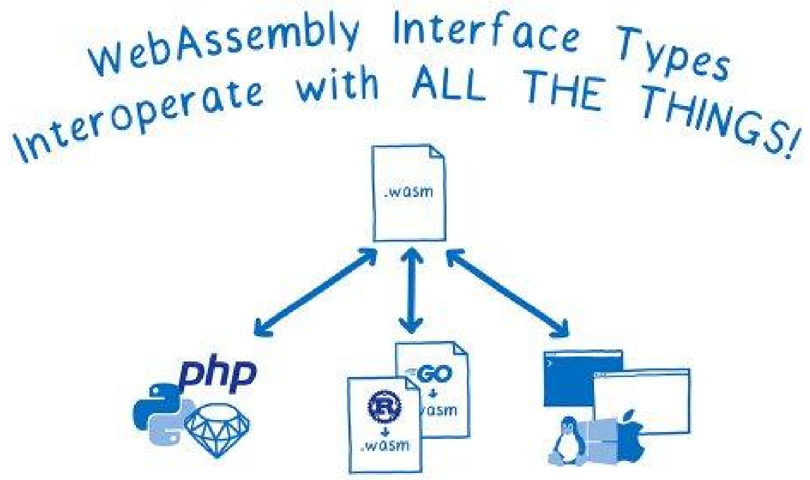
✗ bitcoin script(no exist instuct jump,loop)

● chaincode、solidity、eos wasm code

- Wasm Design Goals：
  - Fast
  - Well-defined
  - Safe: sandboxed environment
  - Hardware-independent
  - Language-independent
  - Open:programs can interoperate with their environment in a simple and universal manner.

Communicate with machine（交互）



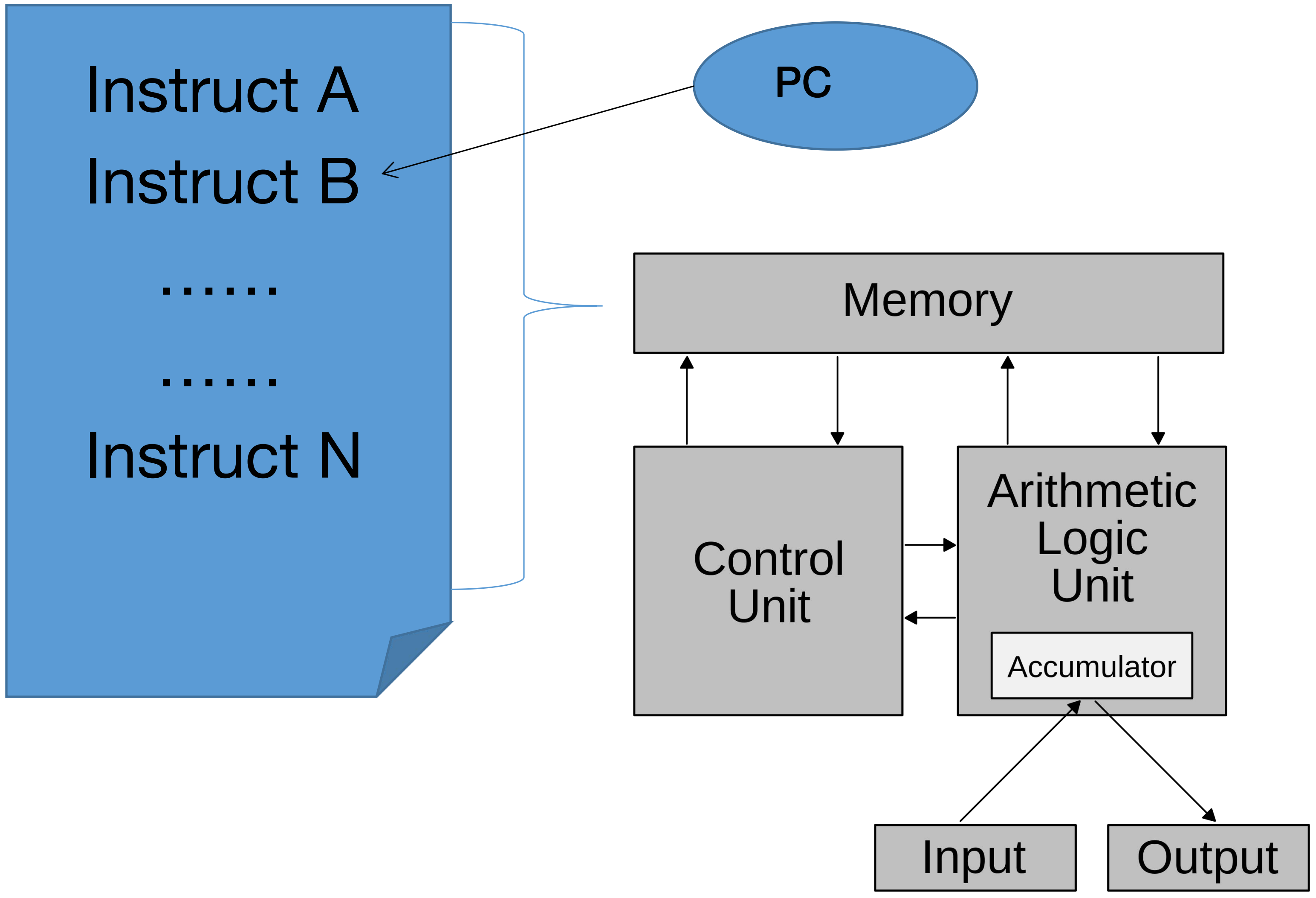

WebAssembly Interface Types
Interoperate with ALL THE THINGS!
.wasm
php
GO
.wasm
.wasm

Hight level programming language
(friendly to humans) → Byte code
(friendly to machine)

virtual machine implement by soft ware which different from:

x86、i386、ARM cpu

Communicate with machine （交互）



Von Neumann architecture

Wasm Core specification（核心标准）

- Runtime Structure:
    - memory instance
    - globals
    - function Instances
    - stack
    - External Values
        - *function instance
        - globals
    - result
    - ……

- compiled function instance and execute context
- compiled function ::=

```
{
        code
        ……
        args
        returns
        is imported
        name
}
```

- execute context ::=

```
{
        stack
        locals
        code
        pc
        current function
}
```

Wasm Core specification （核心标准）

Virtual machine content and
Execute Enviroment

```
• VM ::=
    {
        context
        globals
        memory
        ……
        ExecuteEngine
    }
• ExecuteEngine ::=
    {
        account of contract;
        serialized args;
        contract reference;
        wasm import functions
        resource policy
        result slot
    }
```

Wasm Core specification （核心标准）

- Execution
  - Stack Based Virtual Machines:
    - Compiler convert high level language to native code
    - different from Register Based Machines
  - Operand and operators：
    - example:
      - i32.add: [i32 i32] -> [i32]
    - Instructions:
      - Numeric、Parametric、Variable、Memory、Control、Blocks、Function Calls

- basic data types:
  - 32-bit ;64-bit interger
  - 32-bit ;64-bit float point(forbid)
- Linear memory module
  - mutable array of raw bytes
  - can be grown dynamically
  - load and store values from/to a linear memory at any byte address

Deploy && System check（部署及调用检查）

- Deploy
  - validate check
- Invoke：
  - like EOS：
    - count net usage:
    - cpu:
    - ram:
  - like Ethereum：
    - gas

Block Chain Context && System call (区块链上下文&&虚拟机系统调用)

- input params&&return value
- handle storage
- get contract information
- get blockchain information
- timestamp
- ……

- utils：
  - convert function
  - hash function
  - ……
- Sand box with fix api

Toolchain (合约开发工具链)

- Chain Specification
  - Serialize && deserialize params
  - Serialize && deserialize core data types
    - block、tx、action……
  - Auto generate Eventlog
  - Generate abi
  - API to interate with virtual machine
- Virtual Machine system API
  - fetch input,return output
  - assertion && abort
  - basic Cryptographic function:
    - base58
    - sha256
    - ……
  - *debug

- API for Interact with Block chain context:
  - event log
  - call contract
  - handle state set
  - authority
  - chain data
  - contract message
  - get pseudo random number
  - get current time

Compile Options(编译)

- Install Rust nightly toolchain:
  - $rustup install nightly-2018-11-12
- Install `wasm32-unknown-unknown` target:
  - $rustup target add wasm32-unknown-unknown
- Use nightly toolchain
  - $rustup default nightly
- crate type(Compile as a dynamic link):
  - crate-type = ["cdylib"]
- add `#[no_mangle]` for entry fucntion

Utils && Resource

- Core Specification
  - https://webassembly.github.io/spec/core/bikeshed/index.html
- Online compiler
  - http://mbebenita.github.io/WasmExplorer/
- compiler and toolchain infrastructure library for WebAssembly
  - https://github.com/WebAssembly/binaryen
    - （checkout tag 。。。）convert wasm2wat； wat2wasm； wasm2c； wasm2js……
- Rust-Wasm Example:
  - https://github.com/paritytech/pwasm-tutorial
- Virtual Machine：
  - https://github.com/go-interpreter/wagon
  - Browser
  - https://github.com/perlin-network/life
  - https://github.com/wasmerio/wasmer

Thanks