pyevmasm Documentation

Trail of Bits

Jul 18, 2018

Contents:

CHAPTER 1

API Reference

1.1 evmasm

class pyevmasm.evmasm.**Instruction**(*opcode*, *name*, *operand_size*, *pops*, *pushes*, *fee*, *description*, *operand=None*, *pc=0*)

bytes
Encoded instruction

description
Colloquial description of the instruction

fee
The basic gas fee of the instruction

group
Instruction classification as per the yellow paper

has_operand
True if the instruction uses an immediate operand

is_arithmetic
True if the instruction is an arithmetic operation

is_branch
True if the instruction is a jump

is_endtx
True if the instruction is a transaction terminator

is_environmental
True if the instruction access enviromental data

is_starttx
True if the instruction is a transaction initiator

is_system
True if the instruction is a system operation

is_terminator
True if the instruction is a basic block terminator

mnemonic
Alias for name

name
The instruction name/mnemonic

opcode
The opcode as an integer

operand_size
The immediate operand size

parse_operand(*buf*)
Parses an operand from buf

Parameters **buf** (*iterator/generator/string*) – a buffer

pops
Number words popped from the stack

pushes
Number words pushed to the stack

reads_from_memory
True if the instruction reads from memory

reads_from_stack
True if the instruction reads from stack

reads_from_storage
True if the instruction reads from the storage

semantics
Canonical semantics

size
Size of the encoded instruction

uses_block_info
True if the instruction access block information

uses_stack
True if the instruction reads/writes from/to the stack

writes_to_memory
True if the instruction writes to memory

writes_to_stack
True if the instruction writes to the stack

writes_to_storage
True if the instruction writes to the storage

class pyevmasm.evmasm.**InstructionTable**(**args*, ***kwargs*)

EVM Instruction factory Implements an immutable, iterable instruction LUT that can be indexed by both mnemonic or opcode.

Example:

```
>>> from pyevmasm import instruction_table
>>> instruction_table[0]
Instruction(0x0, 'STOP', 0, 0, 0, 0, 'Halts execution.', None, 0)
>>> instruction_table['STOP']
Instruction(0x0, 'STOP', 0, 0, 0, 0, 'Halts execution.', None, 0)
>>> i = instruction_table.__iter__()
>>> i.__next__()
Instruction(0x0, 'STOP', 0, 0, 0, 0, 'Halts execution.', None, 0)
>>> i.__next__()
Instruction(0x1, 'ADD', 0, 2, 1, 3, 'Addition operation.', None, 0)
>>> i.__next__()
Instruction(0x2, 'MUL', 0, 2, 1, 5, 'Multiplication operation.', None, 0)
>>> i.__next__()
Instruction(0x3, 'SUB', 0, 2, 1, 3, 'Subtraction operation.', None, 0)
```

exception pyevmasm.evmasm.**UnknownMnemonicError**

exception pyevmasm.evmasm.**UnknownOpcodeError**

pyevmasm.evmasm.**assemble**(*asmcode*, *pc=0*)

Assemble an EVM program

Parameters

- **asmcode** (*str*) – an evm assembler program
- **pc** (*int*) – program counter of the first instruction(optional)

Returns the hex representation of the bytecode

Return type str

Example use:

```
>>> assemble('''PUSH1 0x60
                      BLOCKHASH
                      MSTORE
                      PUSH1 0x2
                      PUSH2 0x100
                    ''')
...
b"```@R`}a{\"
```

pyevmasm.evmasm.**assemble_all**(*asmcode*, *pc=0*)

Assemble a sequence of textual representation of EVM instructions

Parameters

- **asmcode** (*str*) – assembly code for any number of instructions
- **pc** (*int*) – program counter of the first instruction(optional)

Returns An generator of Instruction objects

Return type generator[Instructions]

Example use:

```
>>> assemble_one('''PUSH1 0x60
                 PUSH1 0x40
                 MSTORE
                 PUSH1 0x2
```

(continues on next page)

(continued from previous page)

```
                PUSH2 0x108
                PUSH1 0x0
                POP
                SSTORE
                PUSH1 0x40
                MLOAD
                ''')
```

pyevmasm.evmasm.**assemble_hex**(*asmcode*, *pc=0*)

Assemble an EVM program

Parameters

- **asmcode** (*str | iterator[Instruction]*) – an evm assembler program
- **pc** (*int*) – program counter of the first instruction(optional)

Returns the hex representation of the bytecode

Return type str

Example use:

```
>>> assemble_hex('''PUSH1 0x60
                            BLOCKHASH
                            MSTORE
                            PUSH1 0x2
                            PUSH2 0x100
                          ''')
...
"0x6060604052600261010"
```

pyevmasm.evmasm.**assemble_one**(*asmcode*, *pc=0*)

Assemble one EVM instruction from its textual representation.

Parameters

- **asmcode** (*str*) – assembly code for one instruction
- **pc** (*int*) – program counter of the instruction(optional)

Returns An Instruction object

Return type *Instruction*

Example use:

```
>>> print assemble_one('LT')
```

pyevmasm.evmasm.**disassemble**(*bytecode*, *pc=0*)

Disassemble an EVM bytecode

Parameters

- **bytecode** (*str | bytes | bytearray*) – binary representation of an evm bytecode
- **pc** (*int*) – program counter of the first instruction(optional)

Returns the text representation of the assembler code

Example use:

```
>>> disassemble("```@R`}a{\")
...
PUSH1 0x60
BLOCKHASH
MSTORE
PUSH1 0x2
PUSH2 0x100
```

pyevmasm.evmasm.**disassemble_all**(*bytecode*, *pc=0*)

Disassemble all instructions in bytecode

Parameters

- **bytecode** (*str | bytes | bytearray | iterator*) – an evm bytecode (binary)
- **pc** (*int*) – program counter of the first instruction(optional)

Returns An generator of Instruction objects

Return type list[*Instruction*]

Example use:

```
>>> for inst in disassemble_all(bytecode):
...     print(instr)

...
PUSH1 0x60
PUSH1 0x40
MSTORE
PUSH1 0x2
PUSH2 0x108
PUSH1 0x0
POP
SSTORE
PUSH1 0x40
MLOAD
```

pyevmasm.evmasm.**disassemble_hex**(*bytecode*, *pc=0*)

Disassemble an EVM bytecode

Parameters

- **bytecode** (*str*) – canonical representation of an evm bytecode (hexadecimal)
- **pc** (*int*) – program counter of the first instruction(optional)

Returns the text representation of the assembler code

Return type str

Example use:

```
>>> disassemble_hex("0x6060604052600261010")
...
PUSH1 0x60
BLOCKHASH
MSTORE
PUSH1 0x2
PUSH2 0x100
```

pyevmasm.evmasm.**disassemble_one**(*bytecode*, *pc=0*)

Disassemble a single instruction from a bytecode

Parameters

- **bytecode** (*str | bytes | bytearray | iterator*) – the bytecode stream
- **pc** (*int*) – program counter of the instruction(optional)

Returns an Instruction object

Return type *Instruction*

Example use:

```
>>> print disassemble_one('`')
```

pyevmasm.evmasm.**instruction**

Instance of InstructionTable for EVM. (see; InstructionTable)

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index