What is different

이 문서에서는 Titan과 이더리움 사이의 많은 차이점에 대해 설명합니다. Titan에서 dapp을 개발할 때 이러한 차이점들에 대해 아는 것이 좋습니다.

Opcode 차이점

변경된 Opcode

Opcode	Solidity equivalent	Behavior
`COINBASE`	`block.coinbase`	sequencer에 의해 지정됩니다. 현재 OVM_SequencerFeeVault(0x420...011)의 주소를 반환합니다.
`DIFFICULTY`	`block.difficulty`	항상 0을 반환합니다.
`BASEFEE`	`block.basefee`	현재 지원하지 않습니다.
`ORIGIN`	`tx.origin`	L1 -> L2 트랜잭션의 경우 tx.origin은 해당 트랜잭션을 발생시킨 L1 -> L2 트랜잭션의 aliased address 주소입니다. 다른 경우는, L1과 같이 동작합니다.

추가된 Opcode

Opcode	Behavior
`L1BLOCKNUMBER`	L2 상에서 알고 있는 가장 최신의 L1 블록 번호를 반환합니다. 일반적으로 이 블록 번호는 실제 최신 L1 블록 번호보다 최대 15분 뒤쳐집니다.

블록 번호와 Timestamp

블록 생성 시간 간격

- 이더리움에서 `NUMBER` opcode(solidity에서는 `block.number`)는 이더리움의 블록 번호와 같습니다. Titan에서는 이와 유사하게 `block.number`가 L2의 블록 번호와 같습니다. 그러나 L2의 각 트랜잭션은 별도의 블록에 배치되며, 블록은 일정한 시간 간격을 두고 생성되지 않습니다.
- 이것은 `block.number`가 신뢰할 수 있는 시간적 정보의 소스가 아님을 의미하기 때문에 중요합니다. 현재 시간을 가져오고 싶다면 `block.timestamp`를 사용해야 합니다.

Timestamp

- `TIMESTAMP` opcode(solidity에서는 `block.timestamp`)는 해당 트랜잭션이 처리되는 시점의 timestamp입니다.

컨트랙트에서 ETH 사용

- L2에서 ETH를 사용하는 과정은 이더리움에서 ETH를 사용하는 과정과 동일합니다. 단, 필요한 경우 다음 주소에서 ETH를 ERC20 형태로 접근할 수 있다는 점을 알아두세요.
  `0xDeadDeAddeAddEAddeadDEaDDEAdDeaDDeAD0000`
- 결과적으로, state trie 상의 사용자 잔액은 항상 0이며, 실제 사용자 잔액은 위 토큰 컨트랙트의 스토리지에 저장됩니다. 단, 이 컨트랙트를 직접 접근하는 것은 불가능합니다.

Address Aliasing

- CREATE opcode의 특성 때문에 컨트랙트의 bytecode는 다르지만 L1과 L2에서 동일한 주소를 갖는 컨트랙트를 만드는 것이 가능합니다. 이것은 신뢰를 깰 수 있습니다. 이것을 방지하기 위해 `ORIGIN`과 `CALLER`

opcode(solidity에서는 `tx.origin`, `msg.sender`)들이 L2에서는 다르게 동작하게 합니다. `tx.origin`은 이

Call source	tx.origin
L2 user (Externally Owned Account)	The user's address (same as in Ethereum)
L1 user (Externally Owned Account)	The user's address (same as in Ethereum)
L1 contract (using CanonicalTransactionChain.enqueue)	L1_contract_address + `0x1111000000000000000000000000000000001111`

- call-stack의 첫 단계에서 호출되는 `msg.sender`는 `tx.origin`과 동일합니다. 따라서, `tx.origin`가 위에서 정의된 것처럼 영향을 받게 되면 `msg.sender`도 동일하게 영향을 받습니다. 단, `tx.origin`은 사용되지 말아야 한다는 것을 알아두세요.