evm storage

Created	@October 8, 2021 10:12 AM
Created By	
Last Edited By	
Last Edited Time	@October 8, 2021 10:24 AM
Stakeholders	
Status	In Review
Type	Translate

The Ethereum Virtual Machine has three areas where it can store items.

The first is “storage”, where all the contract state variables reside. Every contract has its own storage and it is persistent between function calls and quite expensive to use.

The second is “memory”, this is used to hold temporary values. It is erased between (external) function calls and is cheaper to use.

The third one is the stack, which is used to hold small local variables. It is almost free to use, but can only hold a limited amount of values.

For almost all types, you cannot specify where they should be stored, because they are copied everytime they are used.

The types where the so-called storage location is important are structs and arrays. If you e.g. pass such variables in function calls, their data is not copied if it can stay in memory or stay in storage. This means that you can modify their content in the called function and these modifications will still be visible in the caller.

There are defaults for the storage location depending on which type of variable it concerns:
- state variables are always in storage
- function arguments are always in memory

- local variables of struct, array or mapping type reference storage by default
- local variables of value type (i.e. neither array, nor struct nor mapping) are stored in the stack

---

La Ethereum Virtual Machine tiene tres áreas dónde puede almacenar elementos

La primera es "almacenaje" dónde todas las variables de estado "contrato" residen. Cada "contracto" tiene su propio "almacenaje" y persiste entre llamada de funciones. Es cara de usar.

La segunda es "memoria", la cual se usa para mantener valores temporales. Esta es es borrada entre (externas) llamadas de funciones y es mas barata de usar

La tercera es "apilar", la cual es usada para mantener pequeñas variables locales. Es casi gratis de usar pero solo puede mantener una limitada cantidad de valores.

Para casi todos los tipos, no puedes especificar dónde deberían ser almacenados porque son copiados cada vez que son usados.

Los tipos dónde la, así llamada, "lugar de almacenamiento" es importante son las estructuras y arrays. Si, por ejemplo, pasas dichas variables en llamadas de funciones, sus datos no son copiados si es que pueden quedarse en "memoria" o "almacenaje". Esto significa que, puedes modificar su contenido en el llamado de la función y esas modificaciones seguirán estando visible en el "llamador"