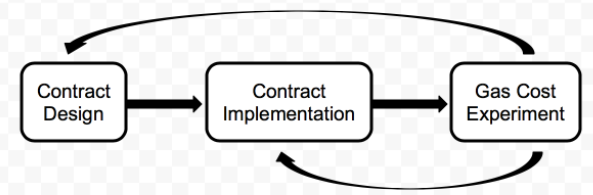
## 如何降低合约中gas费

1. 合理设计合约，必须链上存储的才存到链上，并且以合理的方式存。例如：  
   如果有多个数据要存到链上，但是不需要参与其他运算，只需要必要时读取验证，或者链下验证也可以，此时可以考虑将其生成merkle树，将树根存到链上即可。必要时可使用SPV证明提供原始数据证据。
2. 理解并利用EVM的存储结构  
   EVM存储区分为5个区，其中storage区的存储最贵，一笔SSTORE至少要20000的gas，因此如何减少在storage的存储最重要。  
   storage区是个巨大的map，有2^256个slot，每个插槽是32byte。因此存储时需尽可能复用slot，减少对不同slot的存储。例如：  
   struct Data {  
    uint64 a;  
    uint64 b;  
    uint128 c;  
    uint256 d;  
   }  
   Data public data;  
   constructor(uint64 \_a, uint64 \_b, uint128 \_c, uint256 \_d) public {  
    Data.a = \_a;  
     Data.b = \_b;  
     Data.c = \_c;  
     Data.d = \_d;  
   }  
   这样写可以用2个slot存储Data  
   需要理解EVM里各种类型的存储，合理使用。
3. 熟悉常用指令的gas消耗，合理使用指令。如SSTORE消耗gas最多，有2w，SLOAD也很多，而MLOAD，MSTORE只要3，差别巨大。但这个跟app的设计有关，需要合理规划。
4. 考虑使用Yul，更加优化合约开发。
5. 链上+链下结合，去中心化和中心化结合，优化链上部分。
6. 合理控制开发流程，反复测试和优化代码，降低gas消耗

## 一个降低MEV的想法

抢跑者经常是通过检测交易执行前后，账户的余额来判断是否有机可乘的。

因此可以考虑自己部署一个简单的，类似交易所的合约（或者捐赠合约），在每次交易完有获利后，将获利的部分在该交易所中执行一次额外交易，换成一种自己发行的币，或者让交易看起来不划算，而该交易所使用白名单，只有自己控制的账户才允许交易。

目标就是让抢跑者用机器人无法检测出该交易的获利，或者根本无法检测出有效的结果。从而避开MEV。  
这个想法非常初级，可能并不合适。