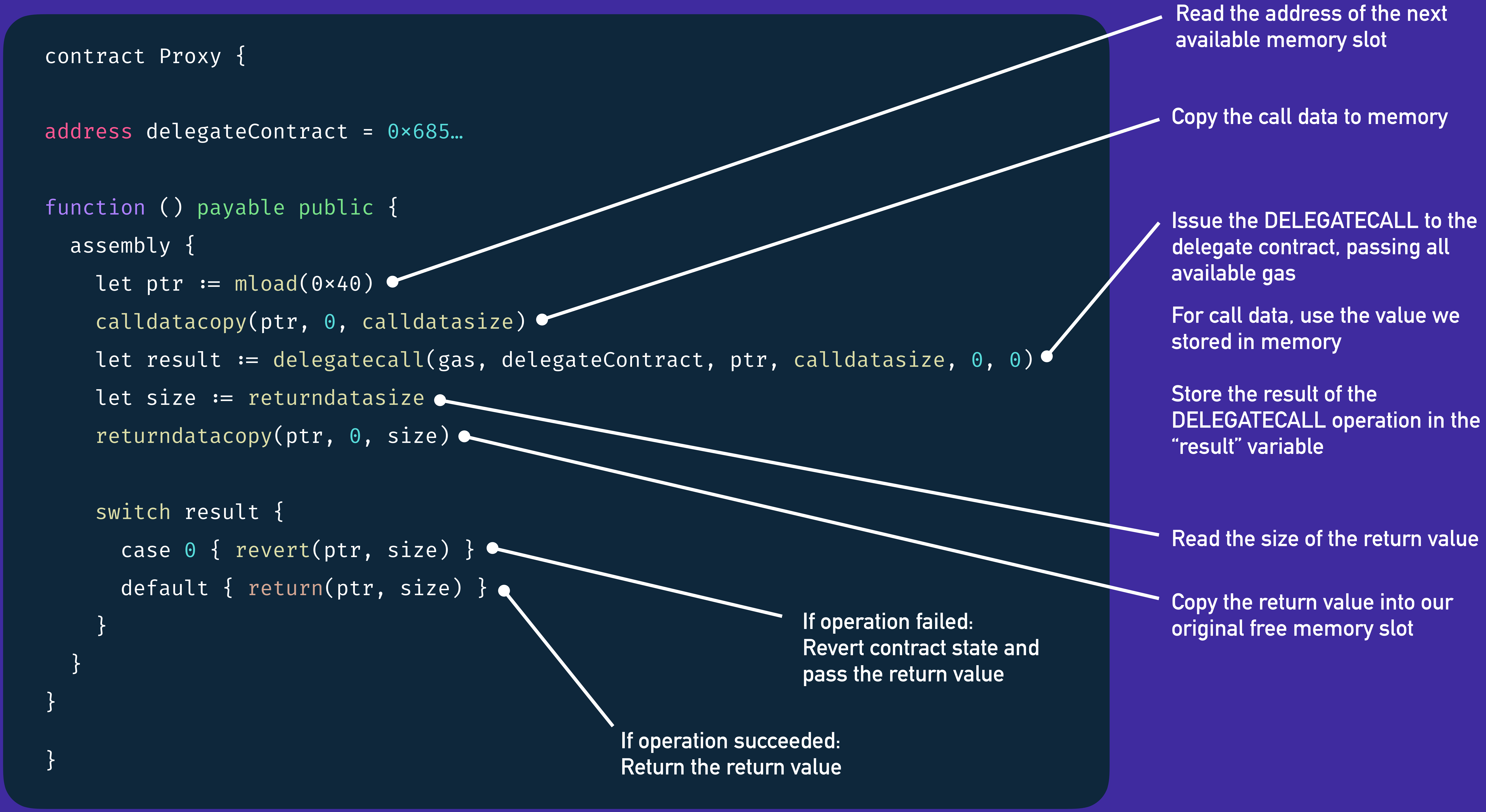
```
address                    0×685…

function    payable public

               mload
    calldatacopy      0  calldatasize
                  delegatecall                             calldatasize  0  0
                returndatasize
    returndatacopy      0

    switch
           0   revert
                return
```



```
contract Proxy {

address delegateContract = 0x685…

function () payable public {
  assembly {
    let ptr := mload(0x40)
    calldatacopy(ptr, 0, calldatasize)
    let result := delegatecall(gas, delegateContract, ptr, calldatasize, 0, 0)
    let size := returndatasize
    returndatacopy(ptr, 0, size)

    switch result {
      case 0 { revert(ptr, size) }
      default { return(ptr, size) }
    }
  }
}

}
```

Read the address of the next available memory slot

Copy the call data to memory

Issue the DELEGATECALL to the delegate contract, passing all available gas

For call data, use the value we stored in memory

Store the result of the DELEGATECALL operation in the "result" variable

Read the size of the return value

Copy the return value into our original free memory slot

If operation failed:
Revert contract state and pass the return value

If operation succeeded:
Return the return value

ForGIFs.com