# Homework 5

**Assembly 1**

1. Look at the example of init code in today's notes
   See gist
   When we do the CODECOPY operation, what are we overwriting?

The codecopy overwrites the first memory slots including the free memory pointer

2. Could the answer to Q1 allow an optimisation?

We could remove the free memory pointer initialisation code since it is not used

3. Can you trigger a revert in the init code in Remix ?

One way to do this would be by sending wei as you deploy the contract.

4. Write some Yul to:
   1. Add 0×07 to 0×08
   2. store the result at the next free memory location.
   3. (optional) write this again in opcodes

```
function add() pure public returns (uint256) {
        assembly{
            let ptr := mload(0x40)
            let free_mem := add(ptr, 0x20)
            let result := add(0x07, 0x08)
            mstore(free_mem, result)
            return(free_mem, 0x20)
        }
    }
```

5. Can you think of a situation where the opcode EXTCODECOPY is used?

It is used in delegate calls and proxy contracts and upgradable contracts

6. Complete the assembly exercises in this repo
   Exercises

**Assembly_1.sol**

```
pragma solidity ^0.8.4;

contract Intro {
    function intro() public pure returns (uint16) {
        uint256 mol = 420;

        assembly {
            let v := mol
                mstore(0x00, v)
                return(0x00, 32)
        }

    }
}
```

**Assembly_2.sol**

```
pragma solidity ^0.8.4;

contract Add {
    function addAssembly(uint256 x, uint256 y) public pure returns (uint256)
{

        assembly {
            mstore(0x11, add(x, y))
        }
        assembly {
            return(0x11, 32)
        }

    }
}
```

### Assembly_3.sol

```
pragma solidity ^0.8.4;

contract SubOverflow {

    function subtract(uint256 x, uint256 y) public pure returns (uint256) {

        assembly {
            let res := 0

            if gt(x,y){
                res := sub(x,y)
            }

            mstore(0x00, res)
            return(0x00, 32)

        }
    }
}
```

### Assembly_4.sol

```
pragma solidity ^0.8.4;

contract Scope {
    uint256 public count = 10;

    function increment(uint256 num) public {

        assembly {
            sstore(count.slot, add(sload(count.slot), num))
        }

    }
}
```