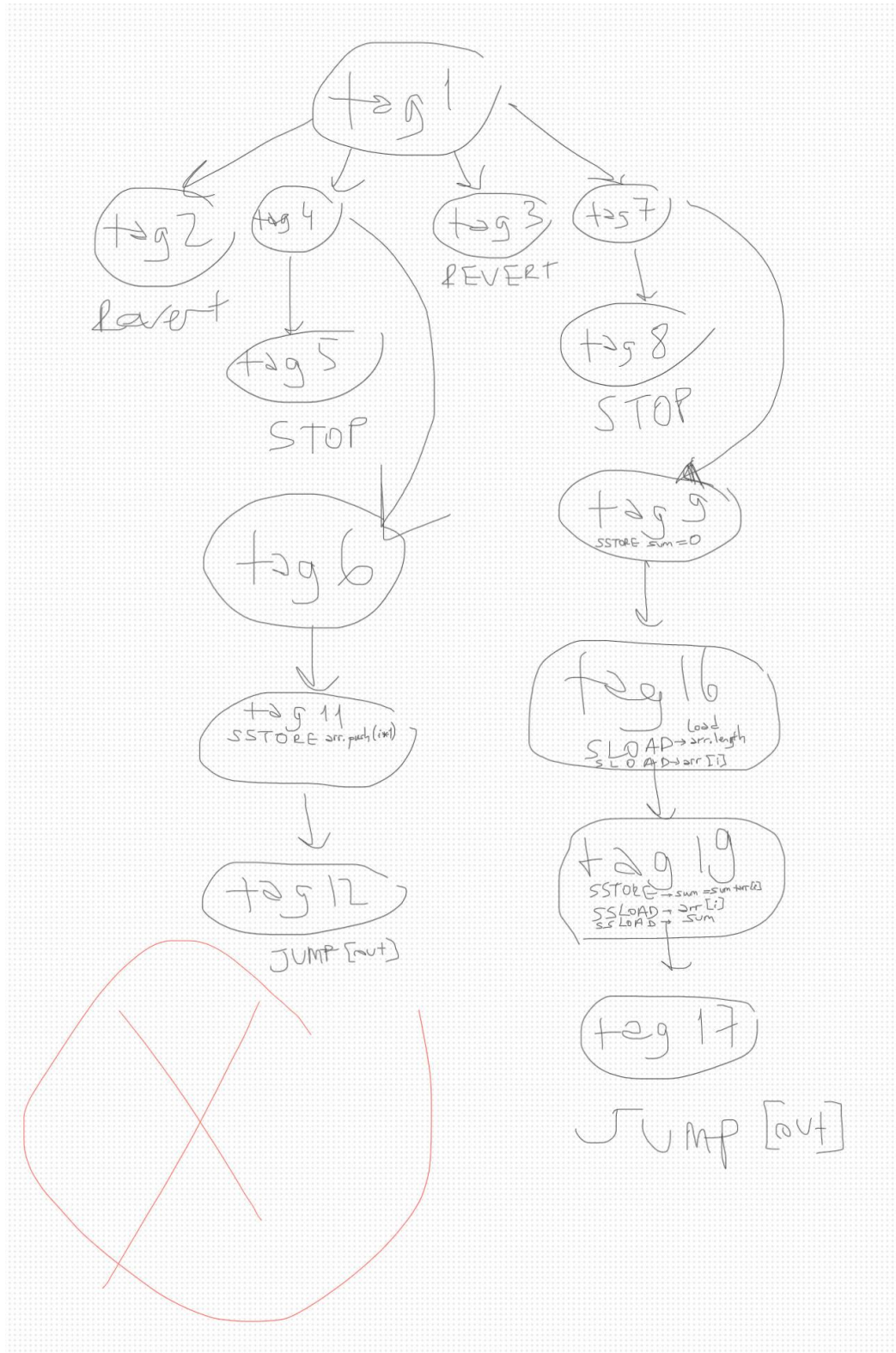


## INTRO TO BLOCKCHAIN – ASSIGNMENT 6

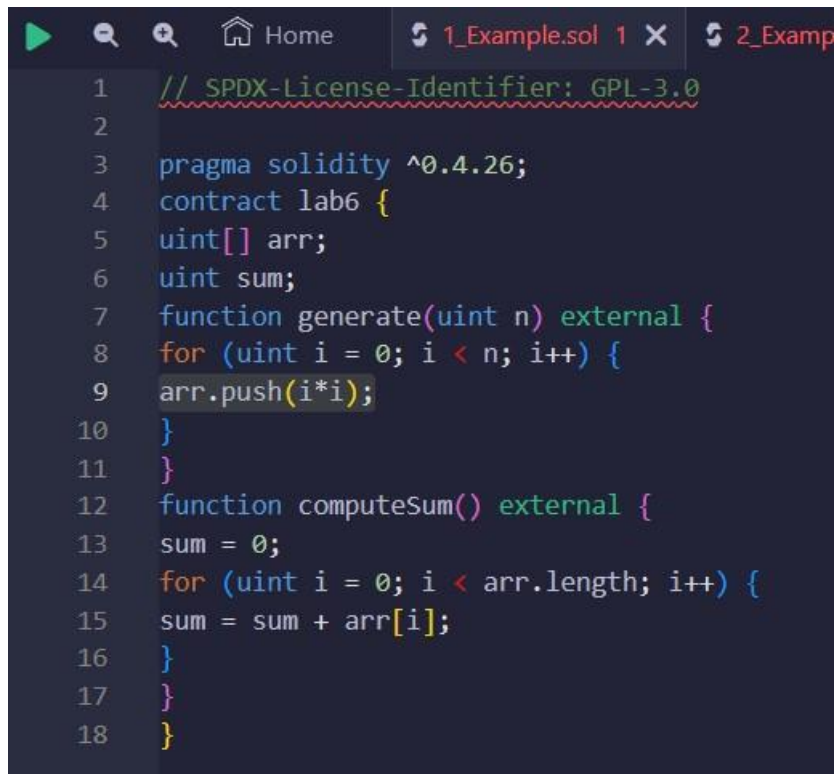
GROUP 22 – ARDA HARMAN

1. The assembly txt file can be found from the zip file
- 2 & 3. Answer for exercise 2 and 3 is below at this image



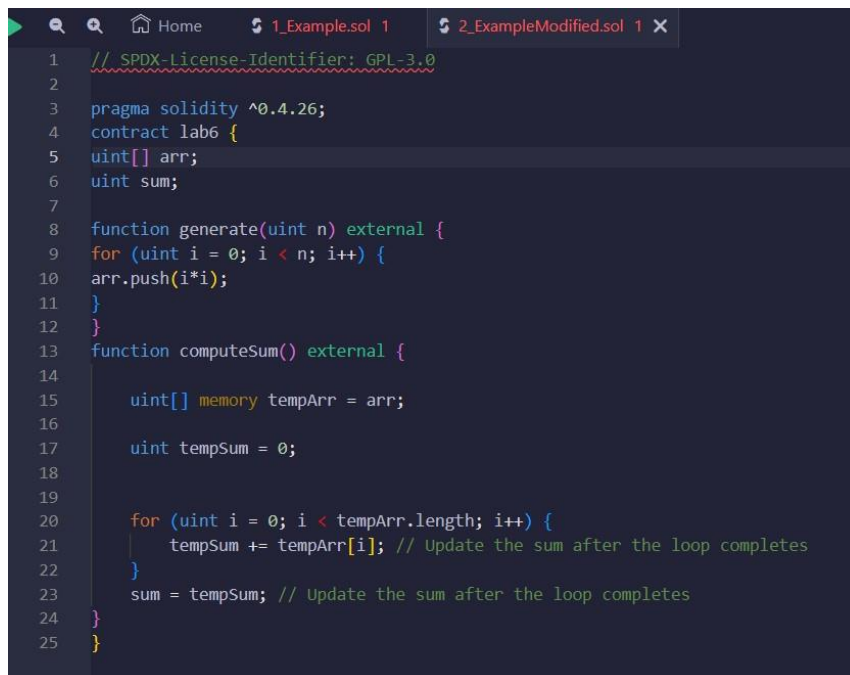
4. Here I implemented an improved version of the given code

Given code:

A screenshot of a code editor with a dark theme. The editor has tabs at the top: 'Home', '1\_Example.sol 1', and '2\_Example'. The code is Solidity, starting with a license identifier and a pragma statement for Solidity 0.4.26. It defines a contract 'lab6' with two variables: 'uint[] arr;' and 'uint sum;'. The 'generate' function is an external function that takes 'uint n' and loops from 0 to n-1, pushing 'i\*i' into the array. The 'computeSum' function is also external, loops through the array, and calculates the sum. The code is as follows:

```
1 // SPDX-License-Identifier: GPL-3.0
2
3 pragma solidity ^0.4.26;
4 contract lab6 {
5     uint[] arr;
6     uint sum;
7     function generate(uint n) external {
8         for (uint i = 0; i < n; i++) {
9             arr.push(i*i);
10        }
11    }
12    function computeSum() external {
13        sum = 0;
14        for (uint i = 0; i < arr.length; i++) {
15            sum = sum + arr[i];
16        }
17    }
18 }
```

Improved version:

A screenshot of a code editor with a dark theme. The editor has tabs at the top: 'Home', '1\_Example.sol 1', and '2\_ExampleModified.sol 1'. The code is Solidity, starting with a license identifier and a pragma statement for Solidity 0.4.26. It defines a contract 'lab6' with two variables: 'uint[] arr;' and 'uint sum;'. The 'generate' function is an external function that takes 'uint n' and loops from 0 to n-1, pushing 'i\*i' into the array. The 'computeSum' function is also external, but instead of accessing 'arr' and 'sum' directly, it uses local variables 'tempArr' and 'tempSum' to store the array and sum respectively. The code is as follows:

```
1 // SPDX-License-Identifier: GPL-3.0
2
3 pragma solidity ^0.4.26;
4 contract lab6 {
5     uint[] arr;
6     uint sum;
7
8     function generate(uint n) external {
9         for (uint i = 0; i < n; i++) {
10            arr.push(i*i);
11        }
12    }
13    function computeSum() external {
14
15        uint[] memory tempArr = arr;
16
17        uint tempSum = 0;
18
19        for (uint i = 0; i < tempArr.length; i++) {
20            tempSum += tempArr[i]; // Update the sum after the loop completes
21        }
22        sum = tempSum; // Update the sum after the loop completes
23    }
24 }
25 }
```

I improved by “using temporary variables” instead of accessing variables from storage

For this I used “tempArr” and “tempSum”

## Deploying contracts

ACCOUNT: 0xAb8...35cb2 (99,999,999,999 Wei)

GAS LIMIT: 3000000

VALUE: 0 Wei

CONTRACT: lab6 - contracts/2\_ExampleModified.sc

evm version: byzantium

Deploy

Publish to IPFS

At Address Load contract from Address

Transactions recorded: 32

Deployed/Unpinned Contracts

- LAB6 AT 0XA13...EAD95 (MEMORY)
- LAB6 AT 0X652...BA595 (MEMORY)

Transaction 1 (Block 31):

- status: 0x1 Transaction mined and execution succeed
- transaction hash: 0x83d5e41c65d91926d8ffabc893432c280bc019613b6e62d8cc1ae3ccc950b4
- block hash: 0x1d1516597de9896e4b827f95b651951acbb87158d1bf234428a1ca3b676e3a
- block number: 31
- contract address: 0xa131AD147055F02e2aAb8156A11bdc81b9eAD95
- from: 0xAb8483F64d9C6d1EcF9B849Ae677d03315835cb2
- to: lab6.(constructor)
- gas: 147468 gas
- transaction cost: 128257 gas
- execution cost: 60917 gas
- input: 0x608...00029
- decoded input: {}
- decoded output: -
- logs: []
- creation of lab6 pending...

Transaction 2 (Block 32):

- status: 0x1 Transaction mined and execution succeed
- transaction hash: 0x25213a26ca24fe53b8ce8848a8a077f0e19b528d0aa2e7b5ed8222359c7d03b5
- block hash: 0xd6ab310caa179b33620ad03a09e4625aca46ec8e806883cb996c48df1eac1a
- block number: 32
- contract address: 0x652c9ACcC53e765e1d96e2455e618dAa879bA595
- from: 0xAb8483F64d9C6d1EcF9B849Ae677d03315835cb2
- to: lab6.(constructor)
- gas: 167375 gas
- transaction cost: 145573 gas

Here is the different execution costs:

1st version: 291417

CONTRACT: lab6 - contracts/2\_ExampleModified.sc

evm version: byzantium

Deploy

Publish to IPFS

At Address Load contract from Address

Transactions recorded: 34

Deployed/Unpinned Contracts

- LAB6 AT 0XA13...EAD95 (MEMORY)

Balance: 0 ETH

computeSum

generate: 100

Transaction 1 (Block 34):

- status: 0x1 Transaction mined and execution succeed
- transaction hash: 0x3b585b13760e7d6317f7332532454fe7fcc55082324f629c4c291438fc76de47
- block hash: 0x459f1aa3ea332995036f2e73b981fbaddfd7114171c6e13fd98772cb55a106c
- block number: 34
- from: 0xAb8483F64d9C6d1EcF9B849Ae677d03315835cb2
- to: lab6.computeSum() 0xa131AD147055F02e2aAb8156A11bdc81b9eAD95
- gas: 359354 gas
- transaction cost: 312481 gas
- execution cost: 291417 gas
- input: 0xc2a...970d1
- decoded input: {}
- decoded output: {}

Transaction 2 (Block 35):

- status: 0x1 Transaction mined and execution succeed
- transaction hash: 0x3b585b13760e7d6317f7332532454fe7fcc55082324f629c4c291438fc76de47
- block hash: 0x459f1aa3ea332995036f2e73b981fbaddfd7114171c6e13fd98772cb55a106c
- block number: 35
- from: 0xAb8483F64d9C6d1EcF9B849Ae677d03315835cb2
- to: lab6.computeSum() 0xa131AD147055F02e2aAb8156A11bdc81b9eAD95
- gas: 359354 gas
- transaction cost: 312481 gas
- execution cost: 291417 gas
- input: 0xc2a...970d1
- decoded input: {}
- decoded output: {}

2nd (improved version) version: 253440

☐ Publish to IPFS

At Address Load contract from Address

Transactions recorded 36 ⓘ >

Deployed/Unpinned Contracts

> LAB6 AT 0xA13...EAD95 (MEMORY) ⓘ ✕

▼ LAB6 AT 0x652...BA595 (MEMORY) ⓘ ✕

Balance: 0 ETH

computeSum

generate 100

Transaction Log:

- ✓ [vm] from: 0xAb8...35cb2 to: lab6.generate(uint256) 0x652...bA595 value: 0 wei data: 0x4a7...00064 logs: 0 hash: 0x2df...763b8
- transact to lab6.computeSum pending ...
- ✓ [vm] from: 0xAb8...35cb2 to: lab6.computeSum() 0x652...bA595 value: 0 wei data: 0xc2a...970d1 logs: 0 hash: 0x879...71123

Transaction Details:

- status 0x1 Transaction mined and execution succeed
- transaction hash 0x87980904cf18528895f4be42d1be3ce8bc3ae8133ca74213d237ae66d371123 ⓘ
- block hash 0xa574544946a13dd16b9c0dc97dca9fa00dceb4bf0189881ba2a672c0cc580161 ⓘ
- block number 36 ⓘ
- from 0xab8483f64d9c6d1fcf9b849ae677d03315835cb2 ⓘ
- to lab6.computeSum() 0x652c9ACcC53e765e1d96e2455E618Aa879bA595 ⓘ
- gas 315680 gas ⓘ
- transaction cost 274504 gas ⓘ
- execution cost 253440 gas ⓘ