Risc-V Simulator

Mostafa Abdelkhalik 900203356
Ahmed Eltokhy 900202729
Ahmed Badr 900202868

Implementation Description:

The program handles taking input from the user as two files, one for the memory and the other for the assembly code, then processes the inputs to print to the console the simulation.

Taking Input:

First, the user can choose whether to add a memory file or not and has to include an assembly program file. The memory file is checked to make sure that all the memory locations included are valid and can be used. The assembly code is used to process the data and generate the simulation. The user also can choose the way he wants to view the output either in Decimal, Hexadecimal, or binary.

Processing assembly code:

The program counter is initially set to the value that the user wants and then we start executing the program through checking the instructions map to execute each line and show the effect on the simulation to the user through the console. Then the program counter gets updated each execution to move to the next instruction except when it faces a function that edits it.

Handling errors:

The program checks the user input to make sure that the memory locations are valid and if not it will terminate. Moreover, when the program encounters a function that is out of the scope it will also terminate. Then we also check for the registers to make sure that it is valid and update it. In addition to the common errors like loading from an empty memory location, syntax errors and so on.

Output:

The program generates a simulation to the user that includes every step that includes the changes that happen in the registers and the used memory locations if any.

Bonus:

We implemented two bonus features, one of them is the way the output is produced in different types: Decimal, Hexa, Binary. And we have created 9 test files: 3 of them are generic to test all the functions and 6 of them are real programs:

- Fibonacci
- array Comparison
- LCM (least common Multiple) {Out of scope to check termination}
- Insertion Sort

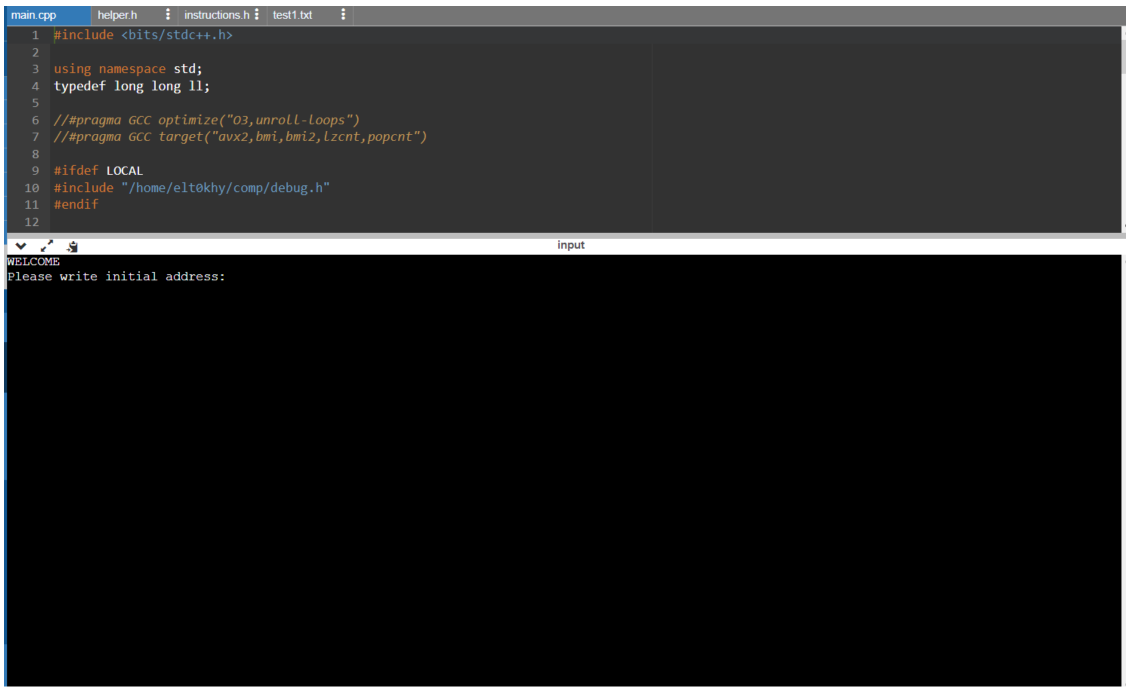
- Maximum element in an array
- Summation of array elements

Design Choices and Assumption:

- The main data structure in our program is map as we use it to map the registers to their values, the memory locations to their values.
- We decided to make the whole program case sensitive in order to simulate the rars simulation as much as we can.
- Each instruction is implemented as a function to make sure all of them are accessible and lead to a definite set of instructions.
- We handled different types of errors to make sure that the program won't suddenly give a runtime error to the user. It will terminate smoothly in some cases and in others it will point out the error for the user to fix.
- As per the tests that we made we did not find any bugs that hinders the usage of our program.
- We assumed that the files given by the user will be free of any unnecessary empty lines. And when the program encounters one it will terminate.
- The address specified by the user must be divisible by 4 or else the program will give an error to the user, and the memory location should be aligned correctly.

User Guide:

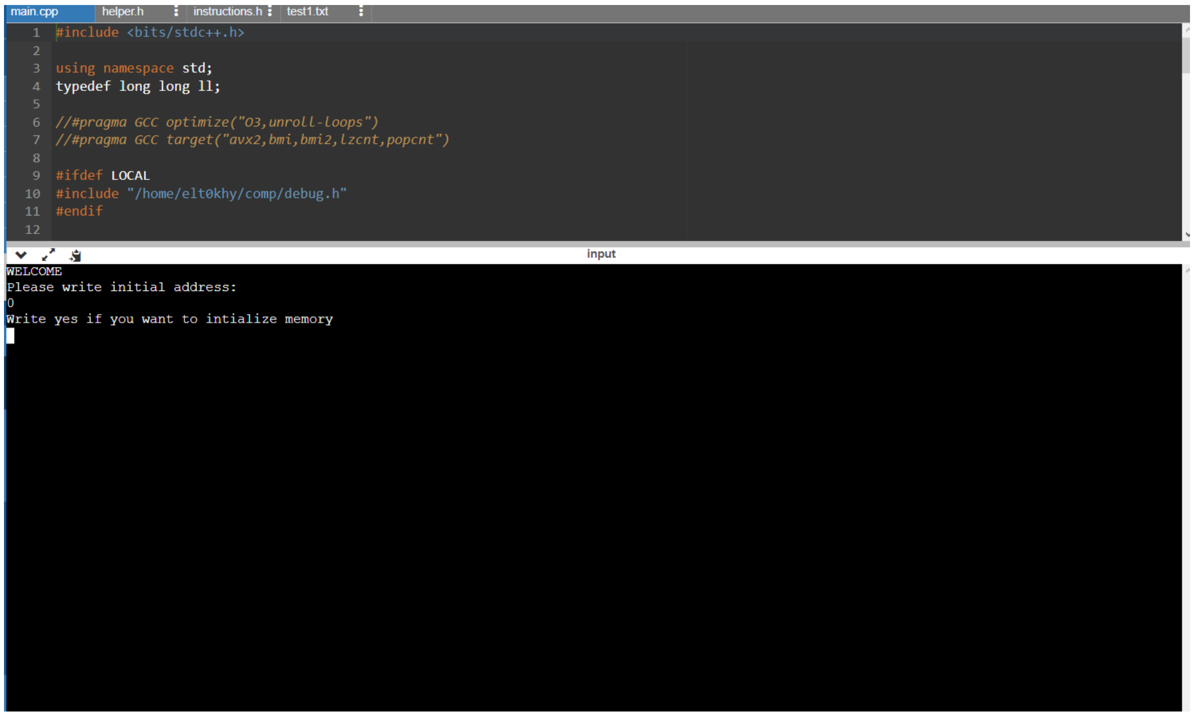
1.The user starts with writing the base of the program counter:



```cpp
#include <bits/stdc++.h>

using namespace std;
typedef long long ll;

//#pragma GCC optimize("O3,unroll-loops")
//#pragma GCC target("avx2,bmi,bmi2,lzcnt,popcnt")

#ifdef LOCAL
#include "/home/elt0khy/comp/debug.h"
#endif

```

```
WELCOME
Please write initial address:
```

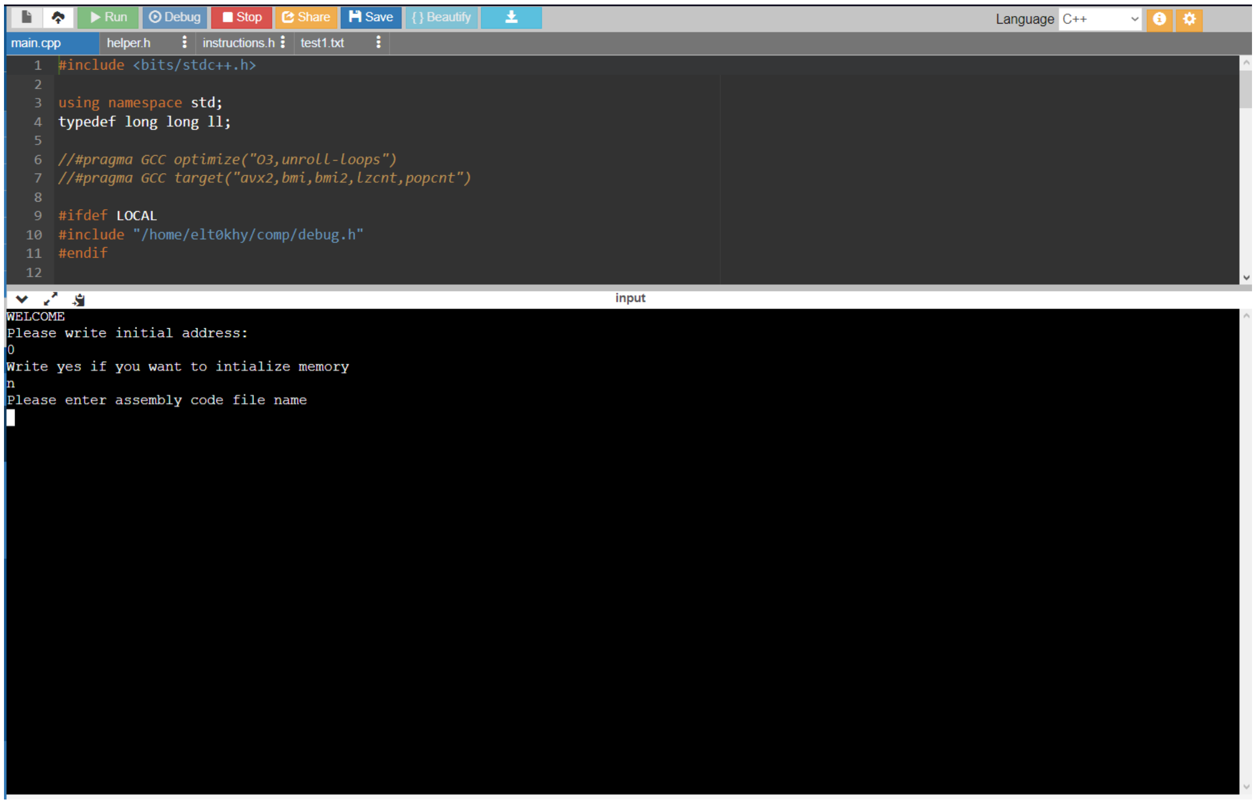
2.Then he is asked whether he wants to add a memory file or not



```cpp
#include <bits/stdc++.h>

using namespace std;
typedef long long ll;

//#pragma GCC optimize("O3,unroll-loops")
//#pragma GCC target("avx2,bmi,bmi2,lzcnt,popcnt")

#ifdef LOCAL
#include "/home/elt0khy/comp/debug.h"
#endif

```

```
WELCOME
Please write initial address:
0
Write yes if you want to intialize memory
```

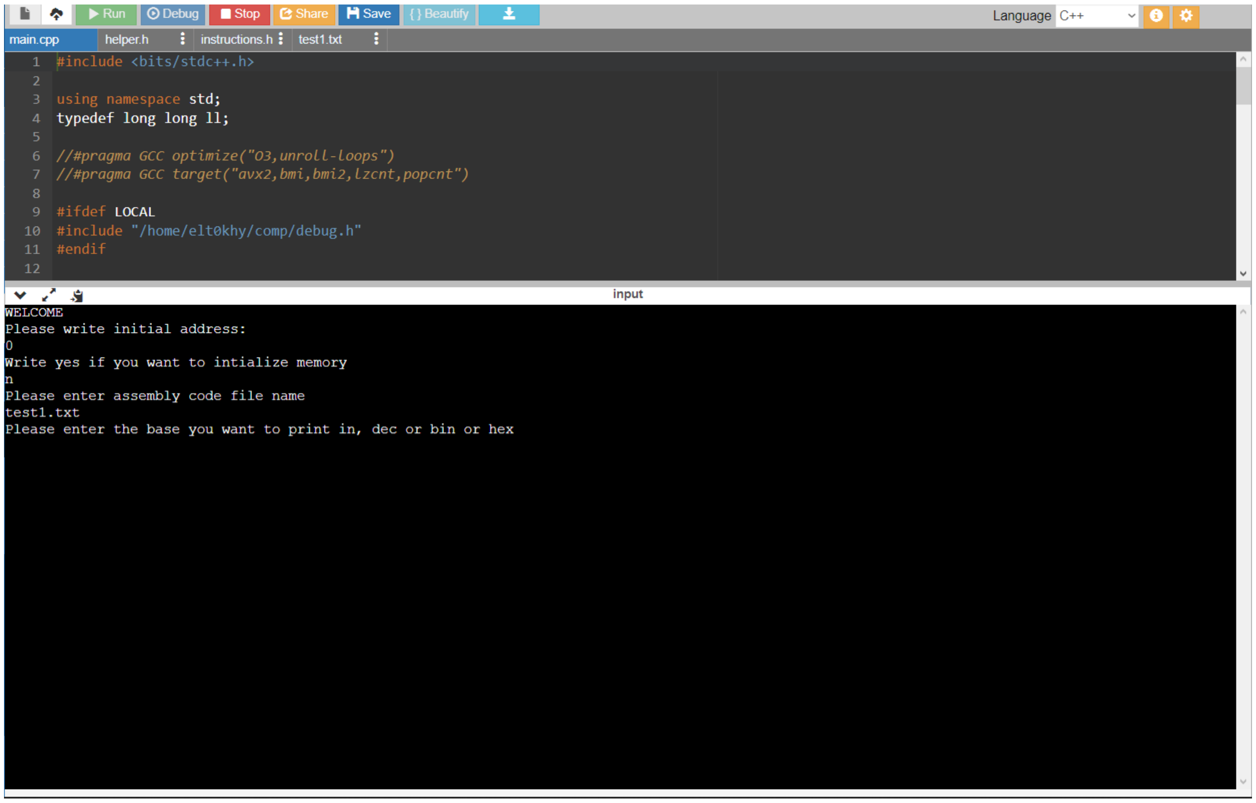
3.If he chooses no then he will be asked to enter the name of the file containing assembly



```cpp
#include <bits/stdc++.h>

using namespace std;
typedef long long ll;

//#pragma GCC optimize("O3,unroll-loops")
//#pragma GCC target("avx2,bmi,bmi2,lzcnt,popcnt")

#ifdef LOCAL
#include "/home/elt0khy/comp/debug.h"
#endif

```

```
WELCOME
Please write initial address:
0
Write yes if you want to intialize memory
n
Please enter assembly code file name
```

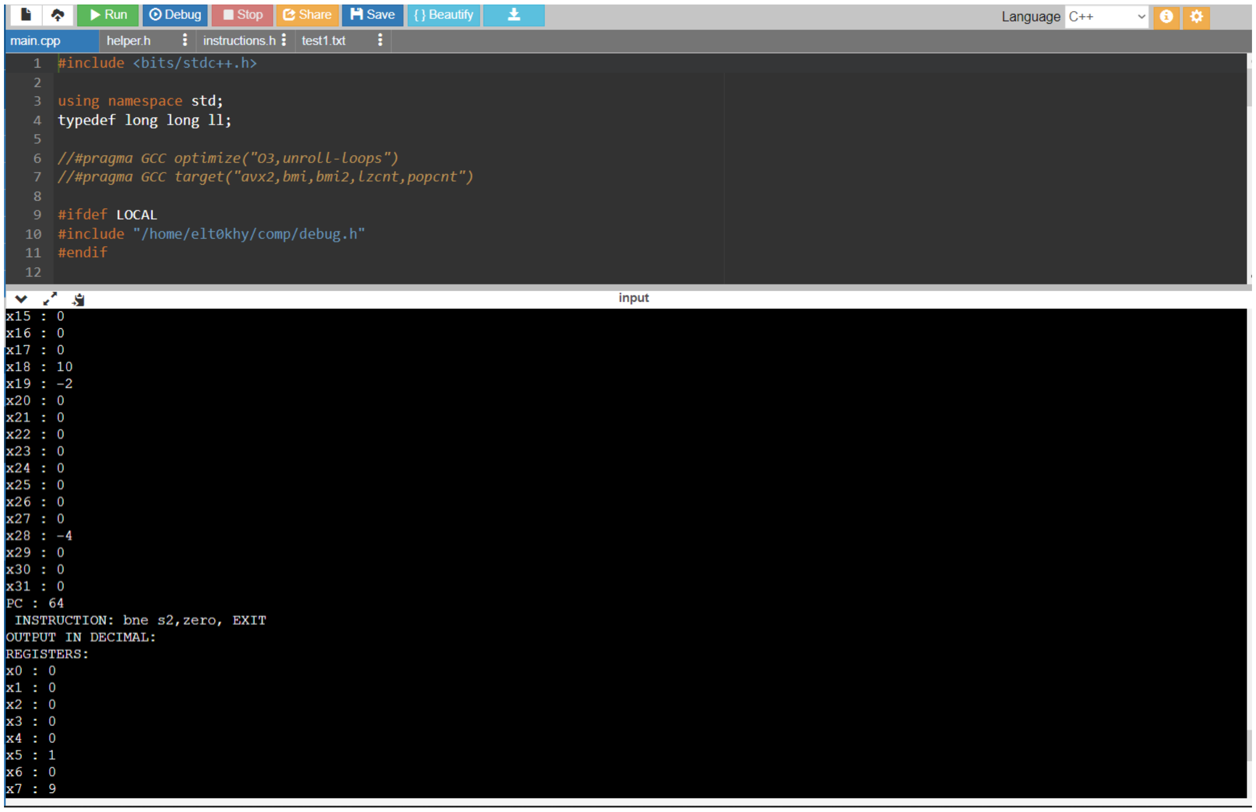
4.After Adding the name he will be asked to choose the type of data:



```cpp
#include <bits/stdc++.h>

using namespace std;
typedef long long ll;

//#pragma GCC optimize("O3,unroll-loops")
//#pragma GCC target("avx2,bmi,bmi2,lzcnt,popcnt")

#ifdef LOCAL
#include "/home/elt0khy/comp/debug.h"
#endif

```

```
WELCOME
Please write initial address:
0
Write yes if you want to intialize memory
n
Please enter assembly code file name
test1.txt
Please enter the base you want to print in, dec or bin or hex
```

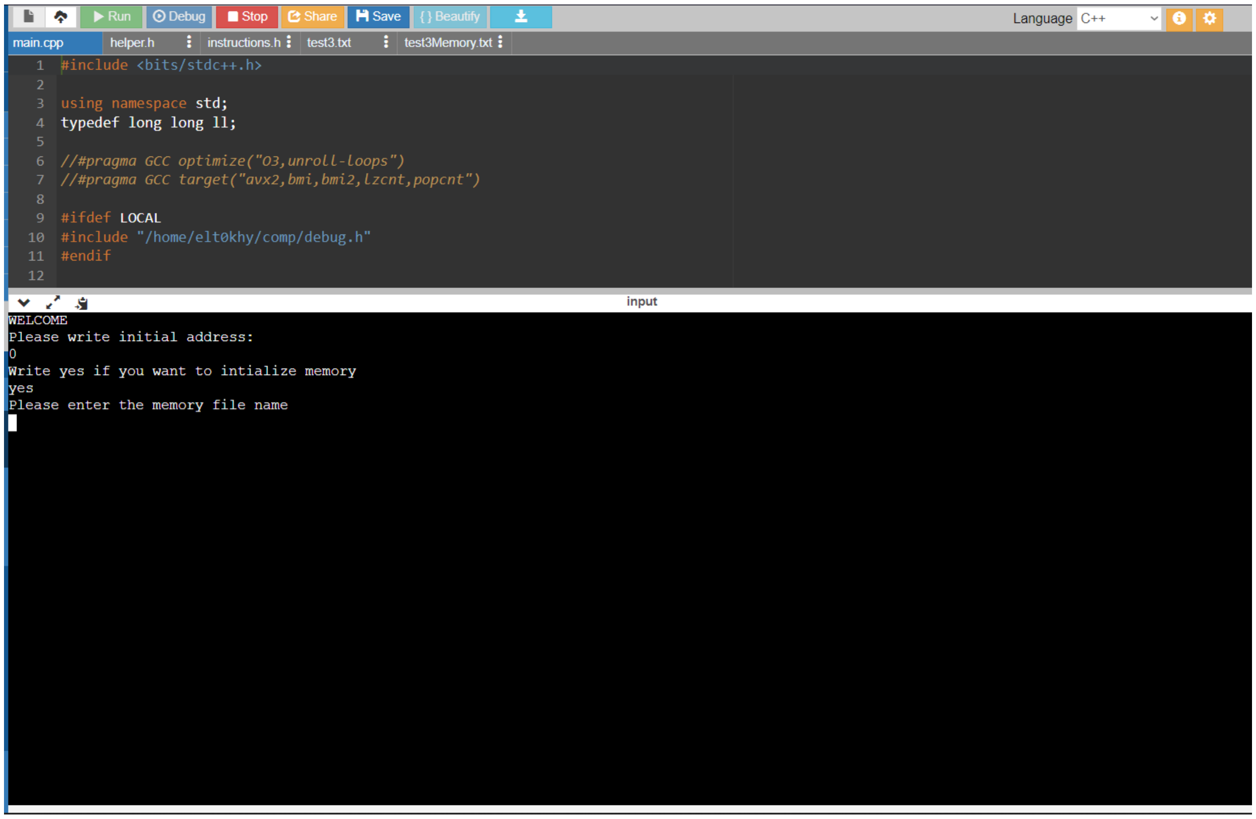
5.Then the simulation will start



```cpp
#include <bits/stdc++.h>

using namespace std;
typedef long long ll;

//#pragma GCC optimize("O3,unroll-loops")
//#pragma GCC target("avx2,bmi,bmi2,lzcnt,popcnt")

#ifdef LOCAL
#include "/home/elt0khy/comp/debug.h"
#endif

```

```
x15 : 0
x16 : 0
x17 : 0
x18 : 10
x19 : -2
x20 : 0
x21 : 0
x22 : 0
x23 : 0
x24 : 0
x25 : 0
x26 : 0
x27 : 0
x28 : -4
x29 : 0
x30 : 0
x31 : 0
PC : 64
 INSTRUCTION: bne s2,zero, EXIT
OUTPUT IN DECIMAL:
REGISTERS:
x0 : 0
x1 : 0
x2 : 0
x3 : 0
x4 : 0
x5 : 1
x6 : 0
x7 : 9
```

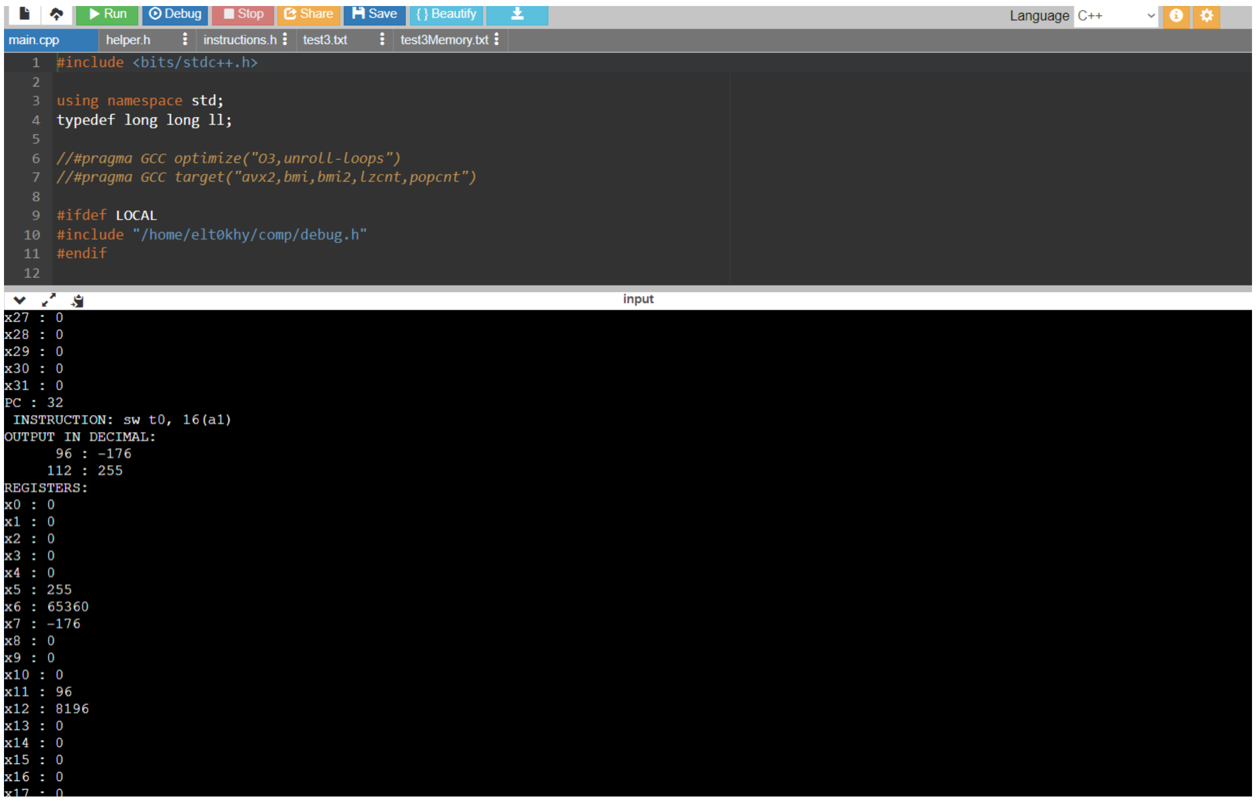
6. If in step 2 he chose yes, the program will ask for the file:



```
#include <bits/stdc++.h>

using namespace std;
typedef long long ll;

//#pragma GCC optimize("O3,unroll-loops")
//#pragma GCC target("avx2,bmi,bmi2,lzcnt,popcnt")

#ifdef LOCAL
#include "/home/elt0khy/comp/debug.h"
#endif

```

```
WELCOME
Please write initial address:
0
Write yes if you want to intialize memory
yes
Please enter the memory file name
```

7. Then he will repeat from step 3



```
#include <bits/stdc++.h>

using namespace std;
typedef long long ll;

//#pragma GCC optimize("O3,unroll-loops")
//#pragma GCC target("avx2,bmi,bmi2,lzcnt,popcnt")

#ifdef LOCAL
#include "/home/elt0khy/comp/debug.h"
#endif

```

```
x27 : 0
x28 : 0
x29 : 0
x30 : 0
x31 : 0
PC : 32
 INSTRUCTION: sw t0, 16(a1)
OUTPUT IN DECIMAL:
      96 : -176
     112 : 255
REGISTERS:
x0 : 0
x1 : 0
x2 : 0
x3 : 0
x4 : 0
x5 : 255
x6 : 65360
x7 : -176
x8 : 0
x9 : 0
x10 : 0
x11 : 96
x12 : 8196
x13 : 0
x14 : 0
x15 : 0
x16 : 0
x17 : 0
```

The affected memory location will be shown as well.