THE AMERICAN UNIVERSITY IN CAIRO

School of Sciences and Engineering

CSCE 2303– Computer Organization and Assembly Language Programming

Spring 2022

Project I: RISC-V RV32I Simulator

By:

Anas Ibrahim - 900204611

Abdelaziz Zakareya - 900203361

Shahd Elmahallawy - 900194441

Description:

Workflow:

1. First, the program takes the starting address from the user and the instruction file name.
2. Then it calls the function take_input which reads the instruction file line by line.
3. During the "take_input" function, while it is taking every line as an instruction, it calls the function decompose_instruction which understands every instruction by dividing it into labels (if exist), operations (or commands like AND,SUB etc), registers, and immediates (if exist). If the line is empty, it is converted to a dumb instruction (for example x = x + 0)
4. After that, we call the function compile_all which calls respective functions that compile an instruction based on the type (s-type, j-type, r-type, …), these types of functions compile each instruction and move the program counter accordingly. After each step, we print the current state of memory and registers.

Implementation:

We have a class called a compiler that holds everything we deal with, which are:
Programme counter - instructions - memory- registers - labels. It also holds all the functions that we deal with to handle everything.

1. Instructions: is a vector of "instruction" which is a struct that holds the command (add -sub - and … etc). It also has a vector of the operands and the instruction type (r-type , i-type, s-type, b-type, u-type, j-type, or exiting)
2. We have six functions one for each command. (Except exiting)
3. The memory is represented as a map called "memory" that has the address as a key and the values
4. Registers are represented as a vector of 32 integers called "registers", and we add the "x" at the end of the index when we show it.
5. "Instr_type" is a map that maps the instruction to its type as a string. For example (add, r_type)
6. "labels" is a map that maps the label's name to the instruction address (line of instruction, or its index in the vector of instructions)

Important Functions:

1. decompose_instruction:
   It takes the instruction as a string from the text file and then starts to translate it to the struct of type "instruction". For example, it converts a string like "   ADD   x1,   x3, x5" to an instruction (command = "ADD", type = r_type, operands = [1, 3, 5])
2. take _input:
   Read the instruction text file and calls decompose_instruction to construct the vector of instructions.
3. Compile_all:
   It checks the type of each operation and translates accordingly using the functions in the file called compiling_by_type.
4. compiling_by_type file: (compile_i_type , compile_r_type, compile_s_type, compile_b_type, compile_j_type, compile_u_type)
   There are 6 functions, each one of them has the operations that belong to that type and translate what the operation does accordingly.

Input:

From the user:

Starting address - filename of the instruction

From the text files:

Read the instruction text file

Output:

Register name - program counter in (decimal - binary- hexadecimal) - value in the register in (decimal - binary- hexadecimal)

Then, it prints whether "The memory is empty" or the address and the value if we store or load.

Bonus Features:

1. We added a function that converts a decimal to binary. We also implemented a function that converts a decimal to hexadecimal. Thus, the output is a map containing the registered name, decimal value, hexadecimal value, and binary value.

2. Second, we included 6 test programs and their equivalent C programs.

Design decisions and/or Assumptions:

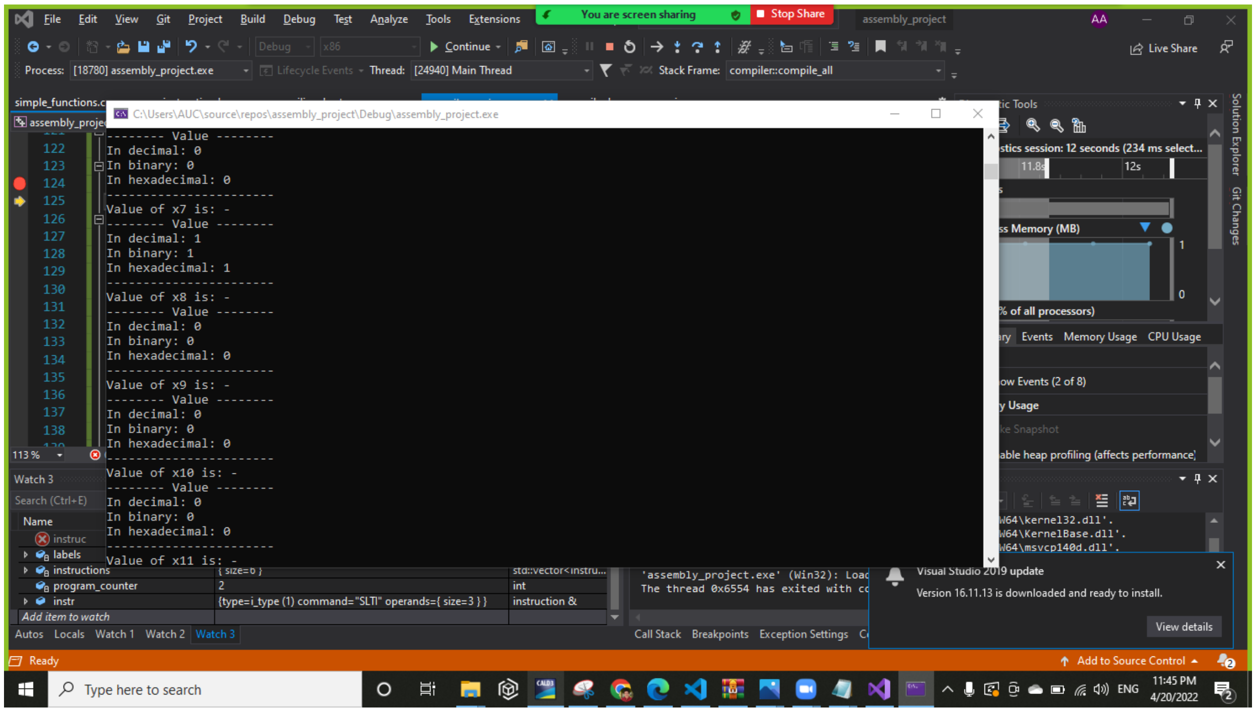
1. The user should input a working code with valid RISC-V syntax. The code does not have tolerance over compilation errors (except for modifying x0)
2. If a runtime error or an infinite loop exists, the C++ code itself would also give an exception by the same runtime error or give the same infinite loops.
3. Register names are x0, x1, ..., x31; alternative names are not handled.

Bugs or Issues:

Because of the bountifulness of text in the output console, the code sometimes seems to take so much time to output everything. Sometimes, it took so much time that we thought it loops infinitely, but we paused in the middle and felt like everything worked fine.

Screenshots:

(After setting variable x7 to 1 in file test.txt)



C:\Users\AUC\source\repos\assembly_project\Debug\assembly_project.exe

```
-------- Value --------
In decimal: 0
In binary: 0
In hexadecimal: 0
----------------------
Value of x7 is: -
-------- Value --------
In decimal: 1
In binary: 1
In hexadecimal: 1
----------------------
Value of x8 is: -
-------- Value --------
In decimal: 0
In binary: 0
In hexadecimal: 0
----------------------
Value of x9 is: -
-------- Value --------
In decimal: 0
In binary: 0
In hexadecimal: 0
----------------------
Value of x10 is: -
-------- Value --------
In decimal: 0
In binary: 0
In hexadecimal: 0
----------------------
Value of x11 is: -
```

Name		
instructions	{ size=6 }	std::vector<instru...
program_counter	2	int
instr	{type=i_type (1) command="SLTI" operands={ size=3 } }	instruction &

```
'assembly_project.exe' (Win32): Loa
The thread 0x6554 has exited with c
```

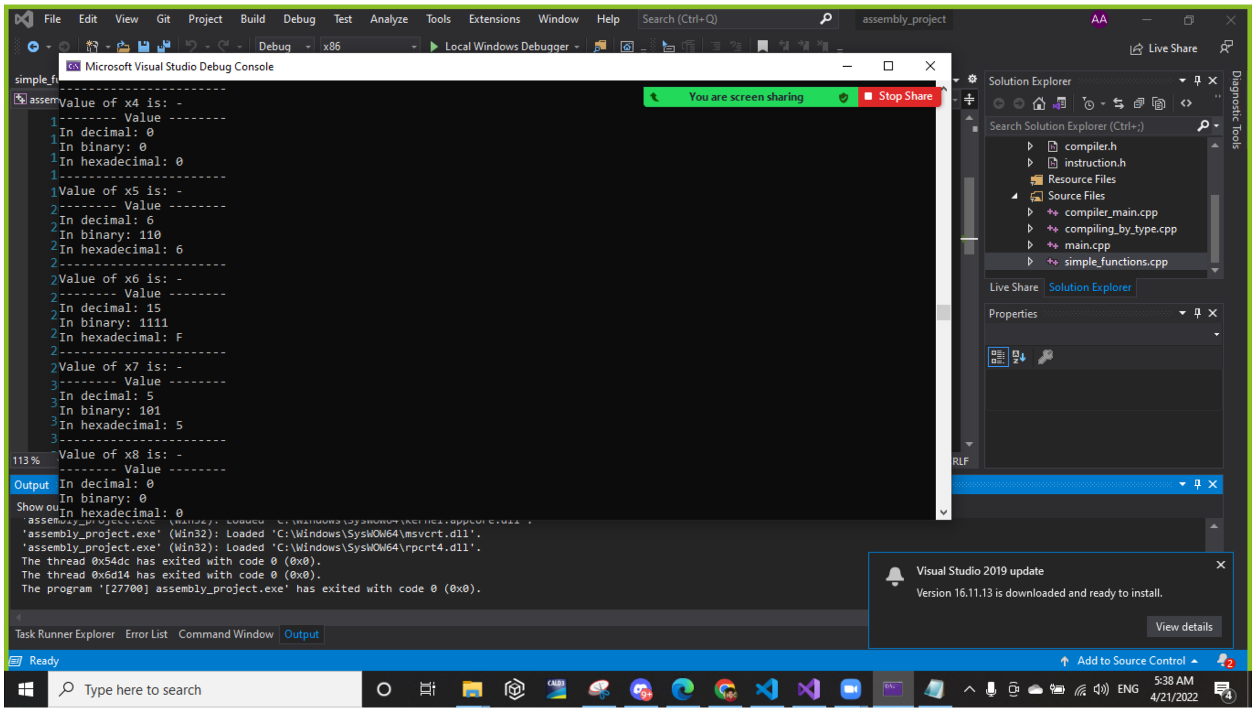
Visual Studio 2019 update
Version 16.11.13 is downloaded and ready to install.

At the end of program0, the registers have the expected values:

(compiles the codes: # int temp = 0;

# for (int i = 1; i <= 5; i++) temp += i;

Where temp is x6, i is x5, x7 is a constant)



```
----------------------
Value of x4 is: -
-------- Value --------
In decimal: 0
In binary: 0
In hexadecimal: 0
----------------------
Value of x5 is: -
-------- Value --------
In decimal: 6
In binary: 110
In hexadecimal: 6
----------------------
Value of x6 is: -
-------- Value --------
In decimal: 15
In binary: 1111
In hexadecimal: F
----------------------
Value of x7 is: -
-------- Value --------
In decimal: 5
In binary: 101
In hexadecimal: 5
----------------------
Value of x8 is: -
-------- Value --------
In decimal: 0
In binary: 0
In hexadecimal: 0
```

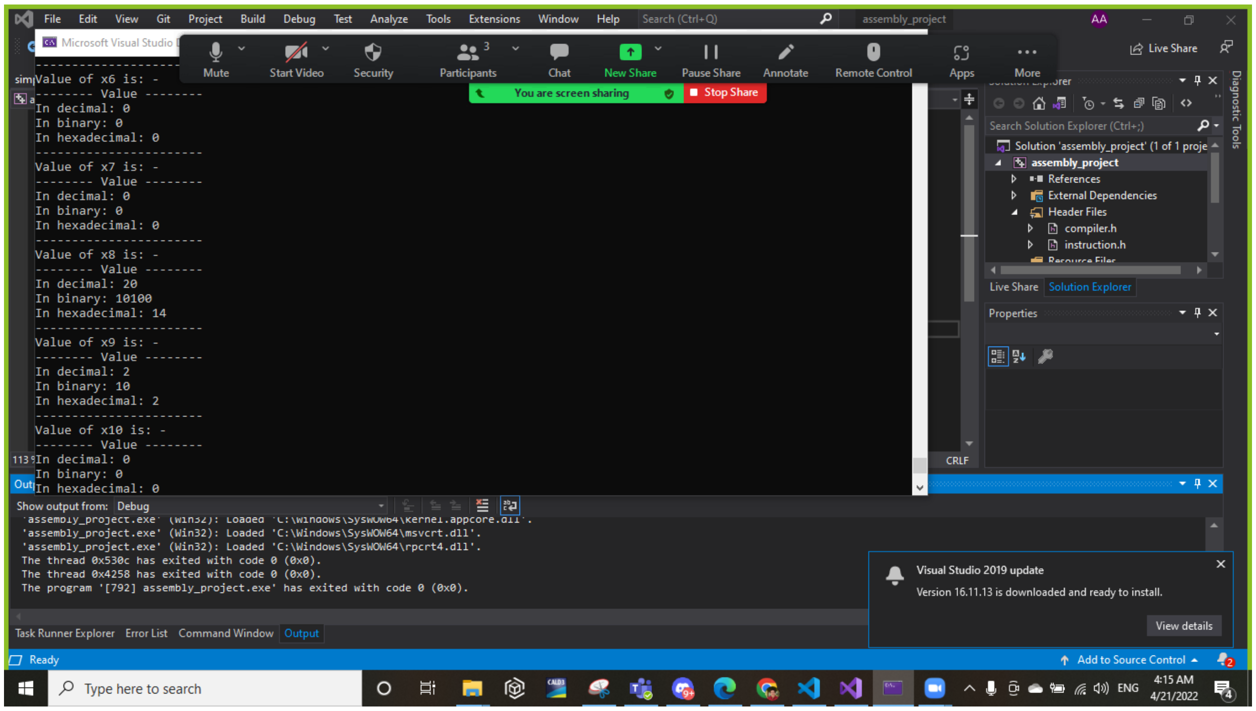
```
'assembly_project.exe' (Win32): Loaded 'C:\Windows\SysWOW64\msvcrt.dll'.
'assembly_project.exe' (Win32): Loaded 'C:\Windows\SysWOW64\rpcrt4.dll'.
The thread 0x54dc has exited with code 0 (0x0).
The thread 0x6d14 has exited with code 0 (0x0).
The program '[27700] assembly_project.exe' has exited with code 0 (0x0).
```

Visual Studio 2019 update

Version 16.11.13 is downloaded and ready to install.

After the end of program1

(for (int i = 0; i < 10; i++) a += b, a (x8) is initialized to 0, b (x9) is initialized to 2, and the final result is 20



```
Value of x6 is: -
-------- Value --------
In decimal: 0
In binary: 0
In hexadecimal: 0
-----------------------
Value of x7 is: -
-------- Value --------
In decimal: 0
In binary: 0
In hexadecimal: 0
-----------------------
Value of x8 is: -
-------- Value --------
In decimal: 20
In binary: 10100
In hexadecimal: 14
-----------------------
Value of x9 is: -
-------- Value --------
In decimal: 2
In binary: 10
In hexadecimal: 2
-----------------------
Value of x10 is: -
-------- Value --------
In decimal: 0
In binary: 0
In hexadecimal: 0
```

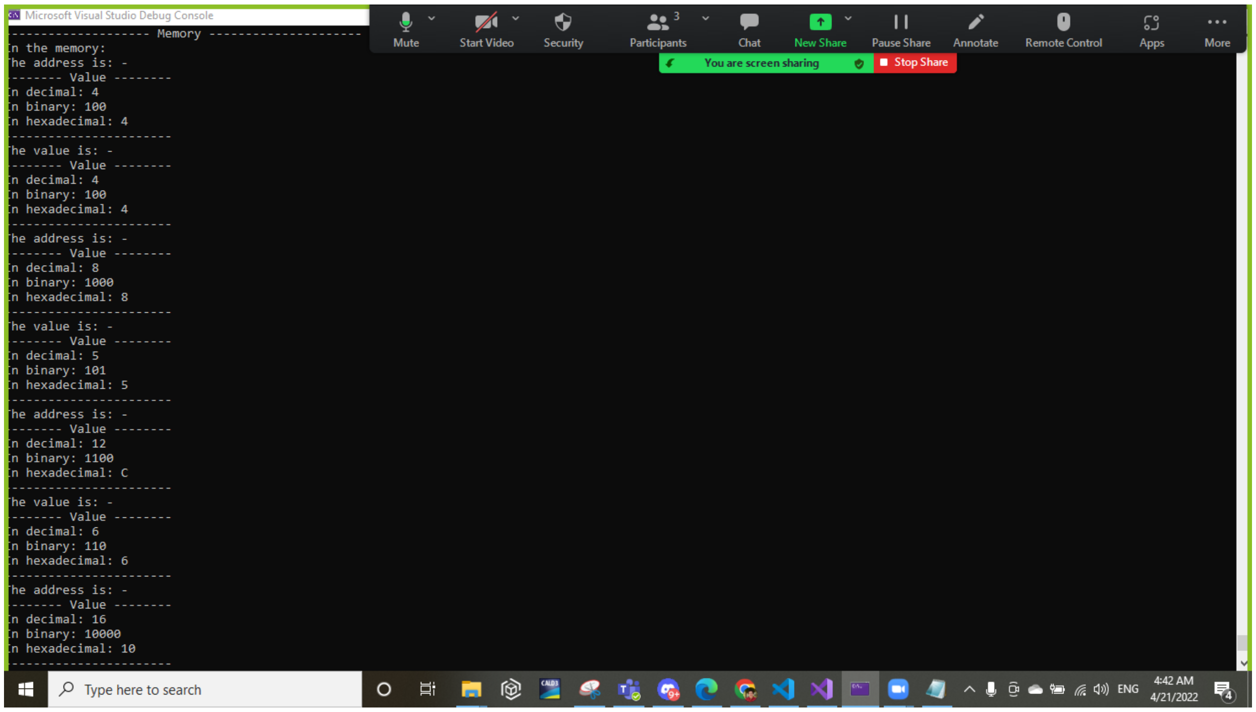
```
'assembly_project.exe' (Win32): Loaded 'C:\Windows\SysWOW64\kernel.appcore.dll'.
'assembly_project.exe' (Win32): Loaded 'C:\Windows\SysWOW64\msvcrt.dll'.
'assembly_project.exe' (Win32): Loaded 'C:\Windows\SysWOW64\rpcrt4.dll'.
The thread 0x530c has exited with code 0 (0x0).
The thread 0x4258 has exited with code 0 (0x0).
The program '[792] assembly_project.exe' has exited with code 0 (0x0).
```

Visual Studio 2019 update

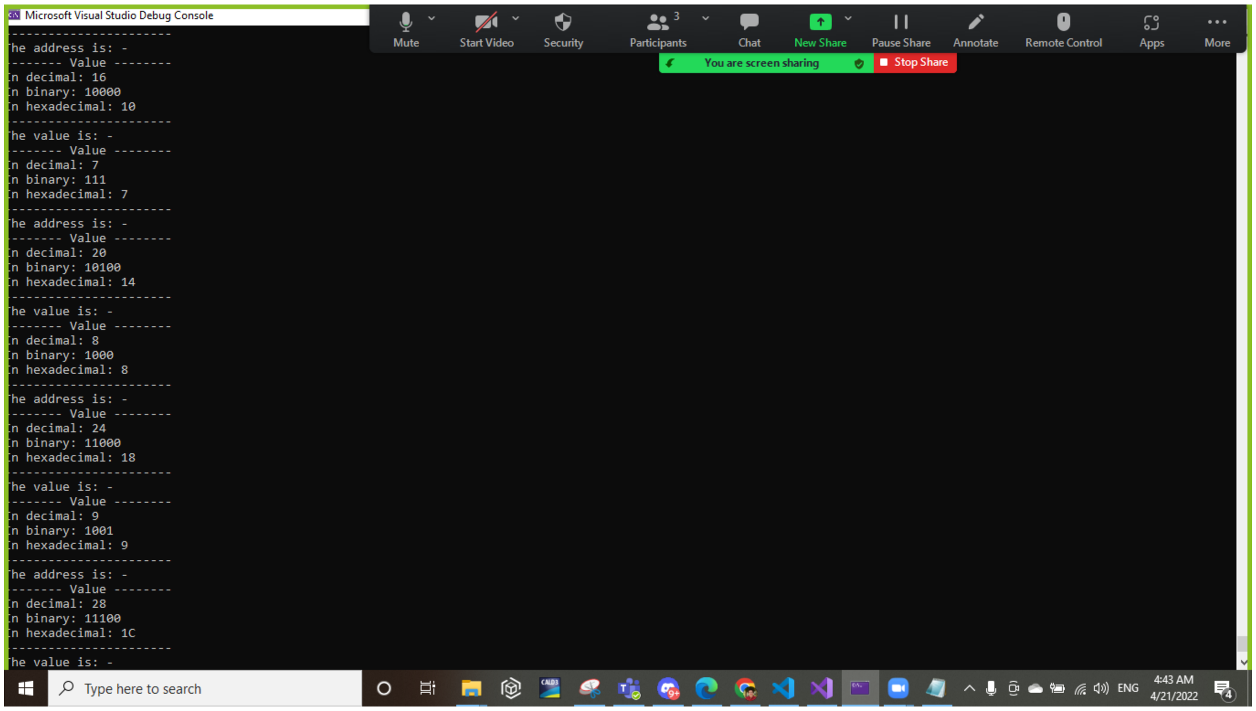
Version 16.11.13 is downloaded and ready to install.

After the end of program2 (int a = 1, b = 3; while (a<10) {D[a] = b+a; a+=1;})

Values in the memory

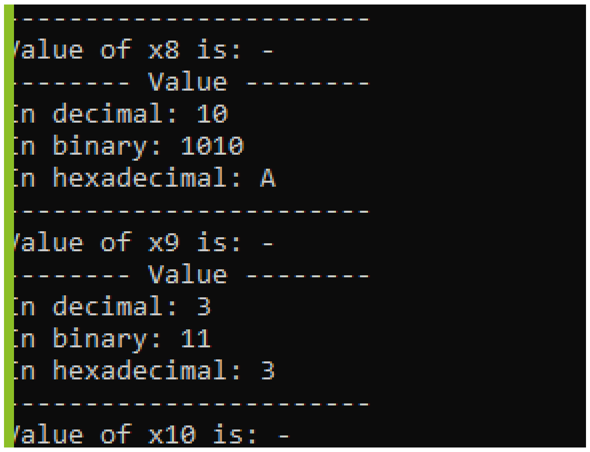


```
-------------------- Memory --------------------
n the memory:
he address is: -
------- Value --------
n decimal: 4
n binary: 100
n hexadecimal: 4
---------------------
he value is: -
------- Value --------
n decimal: 4
n binary: 100
n hexadecimal: 4
---------------------
he address is: -
------- Value --------
n decimal: 8
n binary: 1000
n hexadecimal: 8
---------------------
he value is: -
------- Value --------
n decimal: 5
n binary: 101
n hexadecimal: 5
---------------------
he address is: -
------- Value --------
n decimal: 12
n binary: 1100
n hexadecimal: C
---------------------
he value is: -
------- Value --------
n decimal: 6
n binary: 110
n hexadecimal: 6
---------------------
he address is: -
------- Value --------
n decimal: 16
n binary: 10000
n hexadecimal: 10
---------------------
```



```
----------------------
he address is: -
------- Value --------
In decimal: 16
In binary: 10000
In hexadecimal: 10
----------------------
he value is: -
------- Value --------
In decimal: 7
In binary: 111
In hexadecimal: 7
----------------------
he address is: -
------- Value --------
In decimal: 20
In binary: 10100
In hexadecimal: 14
----------------------
he value is: -
------- Value --------
In decimal: 8
In binary: 1000
In hexadecimal: 8
----------------------
he address is: -
------- Value --------
In decimal: 24
In binary: 11000
In hexadecimal: 18
----------------------
he value is: -
------- Value --------
In decimal: 9
In binary: 1001
In hexadecimal: 9
----------------------
he address is: -
------- Value --------
In decimal: 28
In binary: 11100
In hexadecimal: 1C
----------------------
he value is: -
```

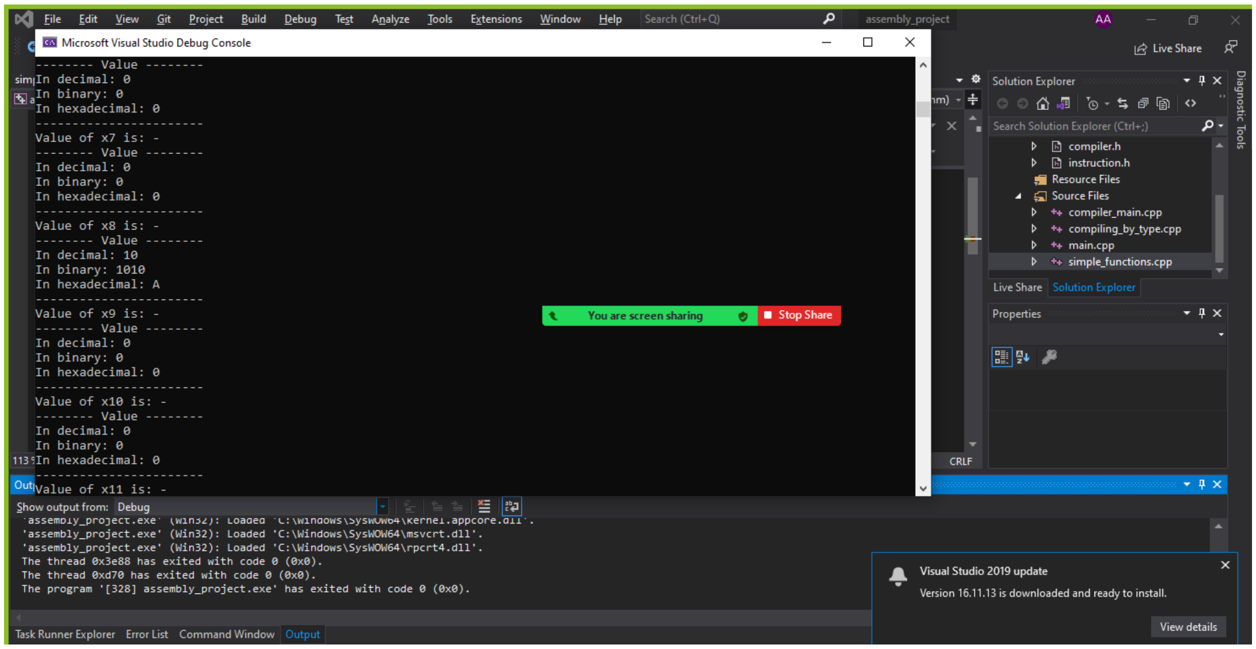
Values of a and b:



```
----------------------
/alue of x8 is: -
-------- Value --------
In decimal: 10
In binary: 1010
In hexadecimal: A
----------------------
/alue of x9 is: -
-------- Value --------
In decimal: 3
In binary: 11
In hexadecimal: 3
----------------------
/alue of x10 is: -
```

After the end of program3

(int k = 1; if (k == 0) f = 5; else f = 10;) (f is x8)



```
-------- Value --------
In decimal: 0
In binary: 0
In hexadecimal: 0
----------------------
Value of x7 is: -
-------- Value --------
In decimal: 0
In binary: 0
In hexadecimal: 0
----------------------
Value of x8 is: -
-------- Value --------
In decimal: 10
In binary: 1010
In hexadecimal: A
----------------------
Value of x9 is: -
-------- Value --------
In decimal: 0
In binary: 0
In hexadecimal: 0
----------------------
Value of x10 is: -
-------- Value --------
In decimal: 0
In binary: 0
In hexadecimal: 0
----------------------
Value of x11 is: -
```

```
'assembly_project.exe' (Win32): Loaded 'C:\Windows\SysWOW64\kernel.appcore.dll'.
'assembly_project.exe' (Win32): Loaded 'C:\Windows\SysWOW64\msvcrt.dll'.
'assembly_project.exe' (Win32): Loaded 'C:\Windows\SysWOW64\rpcrt4.dll'.
The thread 0x3e88 has exited with code 0 (0x0).
The thread 0xd70 has exited with code 0 (0x0).
The program '[328] assembly_project.exe' has exited with code 0 (0x0).
```

User Guide:

If it works, you may try to open the “a.exe” file. Run the code on Visual Studio. Or run on vs code and write in the terminal “g++ “ and add all the CPP file names together. Make sure that the file of instructions is in the same folder/directory as the CPP file.

Input the line index (1-based, first → 1, second → 2) from which you will start compiling

Input the name of the file in the directory