School of Sciences and Engineering

Spring 2022

CSCE2303-02 – Computer Organization and Assembly Language Programming

Assembly Simulator

Submitted to:

Dr. Nourhan Zayed

Dr. Noha Aboelfadl

Submitted by:

Ahmed Fathy - 900192723

Mohamed Ali - 900192994

Mona Kamel - 900191833

20/04/2022

We have implemented the required simulator using C++ code. We searched for the 37 different RISC-V instructions and their format and implementation. A modular programming approach was used to implement the project as we implemented a function for every command that gets called whenever this command is read while execution. Moreover, we used the suitable data structures, maps, to implement different functionalities in the simulator. For example, only used registers were initialized in a map that maps the name of a register to its content. A similar approach was used in the memory initialization where only the used data addresses where initialized and stored their corresponding values in them. Finally, the final map was a map for the commands of a program where each command was mapped to a specific order in the program to facilitate the process of branching and jumping. These three maps were declared as global variables to be able to use them freely instead of passing them each time they are called to decrease the coding complexity. Also a global Boolean variable was declared to identify whether a program is halted or not as well as the program counter to be able to monitor and edit it freely.

Program Execution Steps:

The simulator runs through a series of steps. First, the program reads the program counter and memory initialization file. It initializes both variables accordingly. Second, the simulator reads the commands file and store them sequentially in a map in which they are mapped to their ascending order according to the program counter. Third, the simulator iterates over the commands map. Note: the iteration over the commands map is not done sequentially, however, the iteration is done according to the specific commands that is mapped to the program counter that has the turn to be executed. Commands are not executed sequentially as the branching and jumping needs to skip some commands. Each command was written an if-else statement in which the its function is called and its parameters are parsed accordingly. We have divided the

parsing functions into two formats where one is responsible for parsing of commands that take two source registers, like ADD and SLL, or commands that take one source register and one immediate value, like ADDI or SRAI. The other parsing function is responsible for parsing commands that take offset values, such as LW or SW. Upon the calling of each command, the program counter before and after the calling is displayed as well as the content of the destination register. Finally, the content of each memory address is edited and the memory file is edited to hold the final values after the execution of the whole program.

Bonuses:

We have opted to outputting the content of the registers in three forms, decimal, binary, and hexadecimal.

Design Decision and Assumptions:

We decided to design the jump and branching commands to deal with the program counter instead of labels, which is accepted in RISC-V commands. Moreover, to further stick to the RISC 32 ISA, we decided to increment the program counter and the memory addresses by 4 each times as this means 4 bytes or 32 bits which the size specified by the ISA for the registers and memory locations.

For the registers, their names are x0,x1,…,x31 and they are case sensitive. As for the initialization files, they are called “Test 1.txt”, ”Test 2.txt”, “Test 3.txt” and they correspond to programs P1, P2, and P3 respectively.

The format of the initialization file is to write the starting address on the first line. The next line are written in the following format: <memory_address> <value>

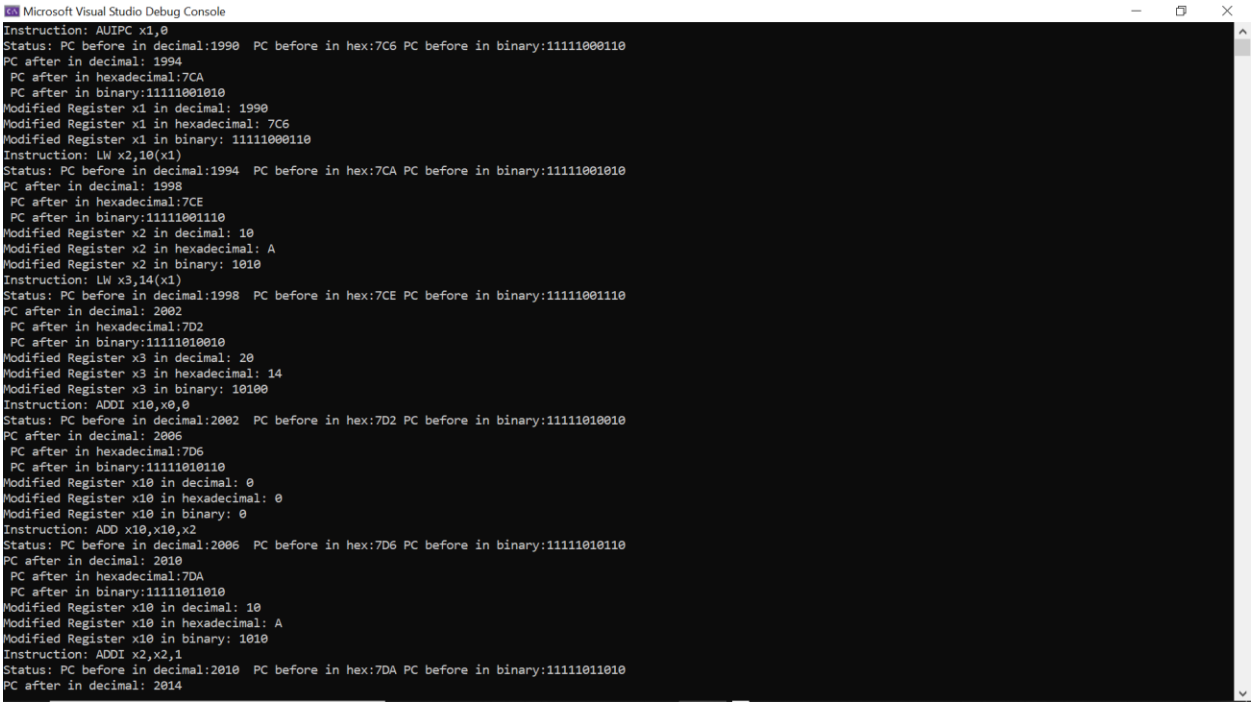
Bugs:

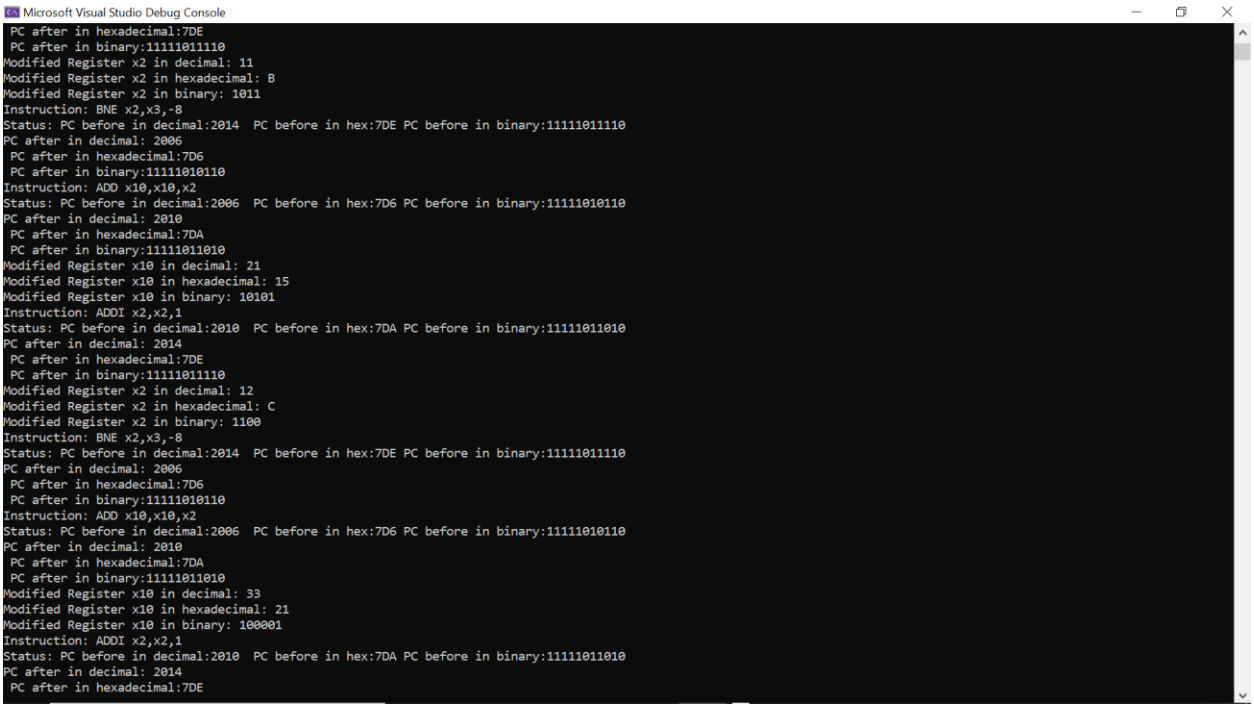
The simulator was tested multiple times and was fixed to solve all the errors found and there are no remaining bugs.

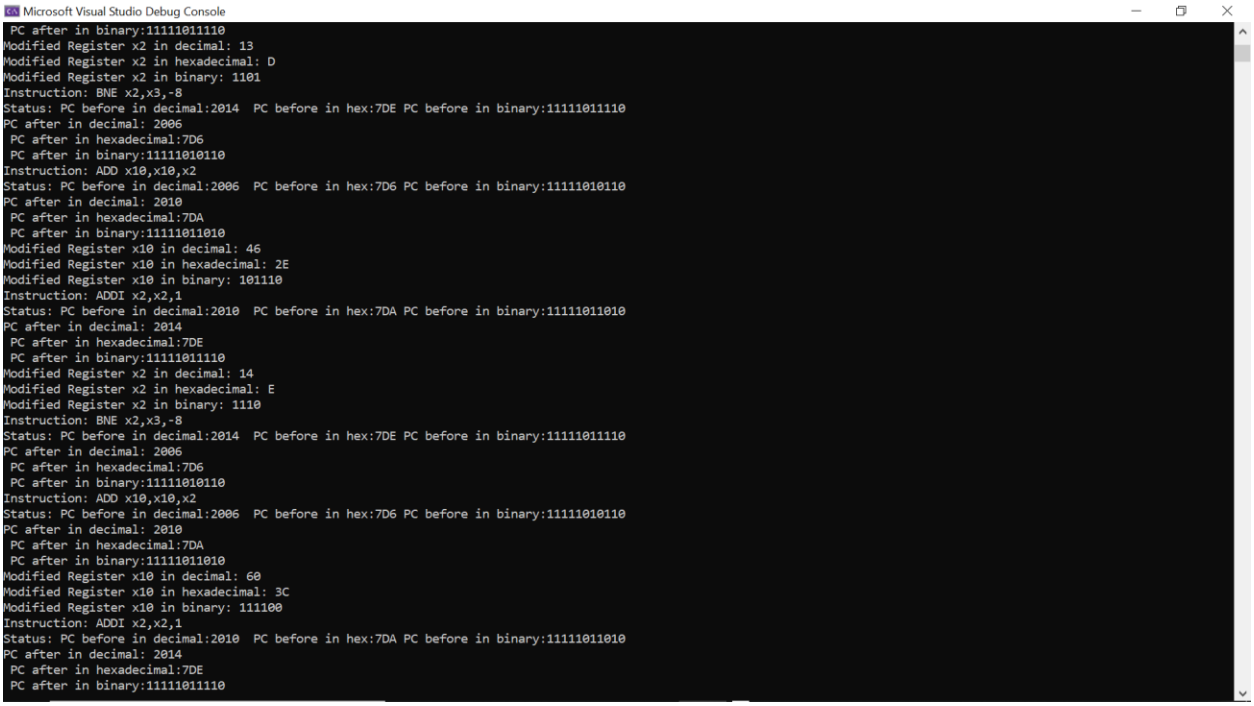
Guide to Use the simulator:

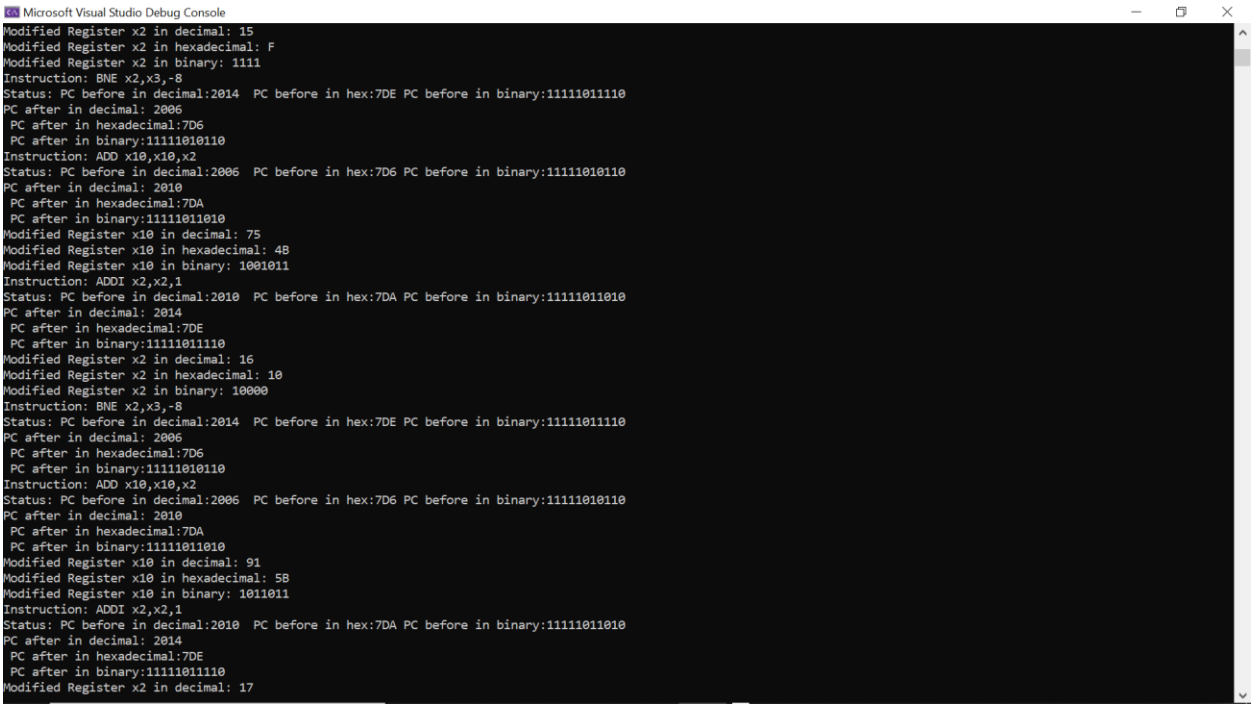
In order to run the program on a specific test case the user must edit the name of the two text files entered to the simulator which are shown in the screenshots below:

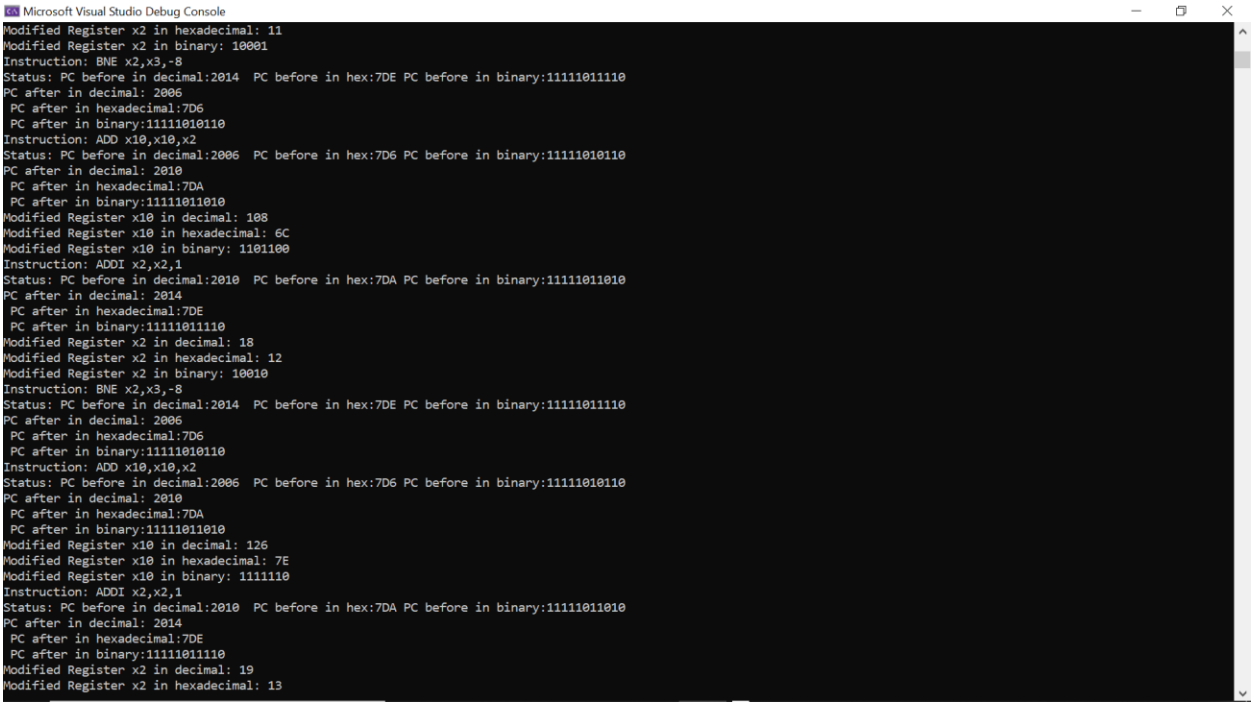
```
823     int start;
824     // ...
829     data.open("Test 3.txt");
830     if (!data.is_open())
831         cout << "Error in opening data file";
832     else
833     {
834         int address, value;
835         getline(data, comm);
836         start = stoi(comm);
837         pc = start;
838         while (!data.eof())
839         {
840             getline(data, comm);
841             address = stoi(comm.substr(0, comm.find(' ')));
842             value = stoi(comm.substr(comm.find(' ') + 1, comm.find('\n')));
843             mem[address] = value;
844         }
845     }
846     data.close();
847     file.open("P3.txt");
848     if (!file.is_open())
849         cout << "Error";
850     else
851     {
852         while (!file.eof())
853         {
854             getline(file, comm);
855             program_mem[pc] = comm;
856             pc += 4;
857         }
858     }
859     pc = start;
860     while (cont)
861     {
862         inst_exec(program_mem[pc]);
863     }
864     file.close();
865     remove("Test 3.txt");
866     data.open("Test 3.txt", ios::app);
```

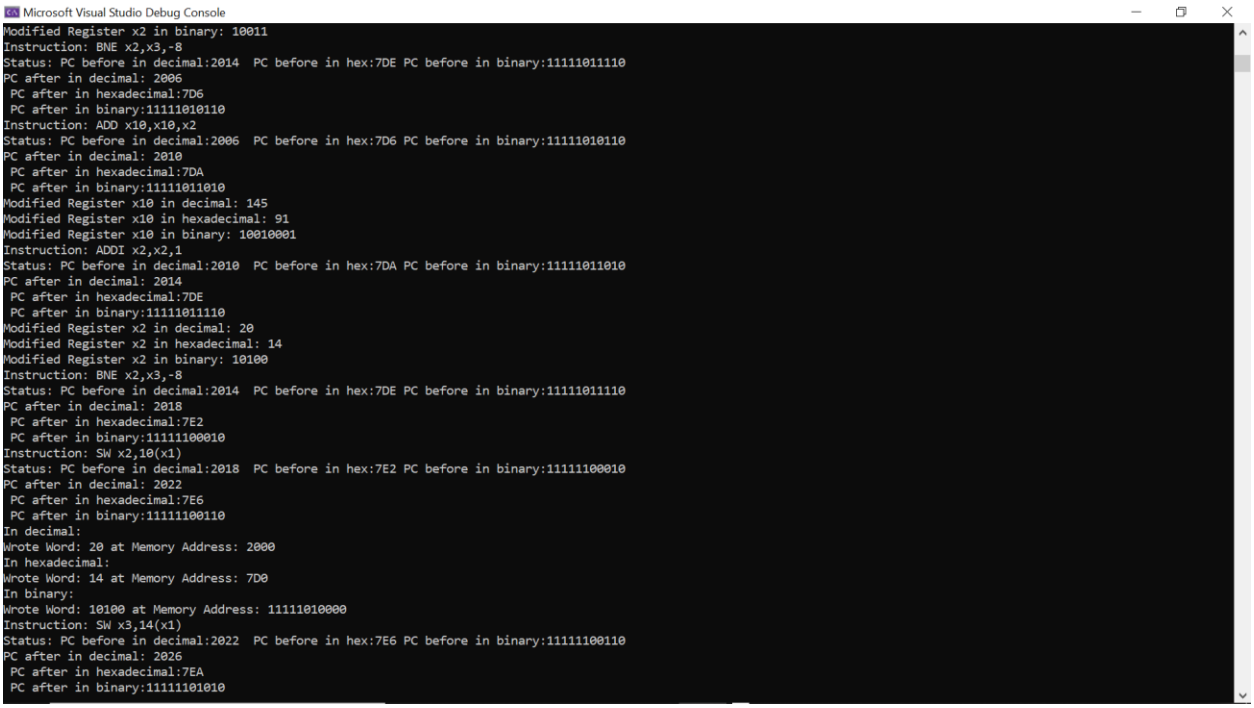

Microsoft Visual Studio Debug Console
Instruction: AUIPC x1,0
Status: PC before in decimal:1990 PC before in hex:7C6 PC before in binary:11111000110
PC after in decimal: 1994
 PC after in hexadecimal:7CA
 PC after in binary:11111001010
Modified Register x1 in decimal: 1990
Modified Register x1 in hexadecimal: 7C6
Modified Register x1 in binary: 11111000110
Instruction: LW x2,10(x1)
Status: PC before in decimal:1994 PC before in hex:7CA PC before in binary:11111001010
PC after in decimal: 1998
 PC after in hexadecimal:7CE
 PC after in binary:11111001110
Modified Register x2 in decimal: 10
Modified Register x2 in hexadecimal: A
Modified Register x2 in binary: 1010
Instruction: LW x3,14(x1)
Status: PC before in decimal:1998 PC before in hex:7CE PC before in binary:11111001110
PC after in decimal: 2002
 PC after in hexadecimal:7D2
 PC after in binary:11111010010
Modified Register x3 in decimal: 20
Modified Register x3 in hexadecimal: 14
Modified Register x3 in binary: 10100
Instruction: ADDI x10,x0,0
Status: PC before in decimal:2002 PC before in hex:7D2 PC before in binary:11111010010
PC after in decimal: 2006
 PC after in hexadecimal:7D6
 PC after in binary:11111010110
Modified Register x10 in decimal: 0
Modified Register x10 in hexadecimal: 0
Modified Register x10 in binary: 0
Instruction: ADD x10,x10,x2
Status: PC before in decimal:2006 PC before in hex:7D6 PC before in binary:11111010110
PC after in decimal: 2010
 PC after in hexadecimal:7DA
 PC after in binary:11111011010
Modified Register x10 in decimal: 10
Modified Register x10 in hexadecimal: A
Modified Register x10 in binary: 1010
Instruction: ADDI x2,x2,1
Status: PC before in decimal:2010 PC before in hex:7DA PC before in binary:11111011010
PC after in decimal: 2014

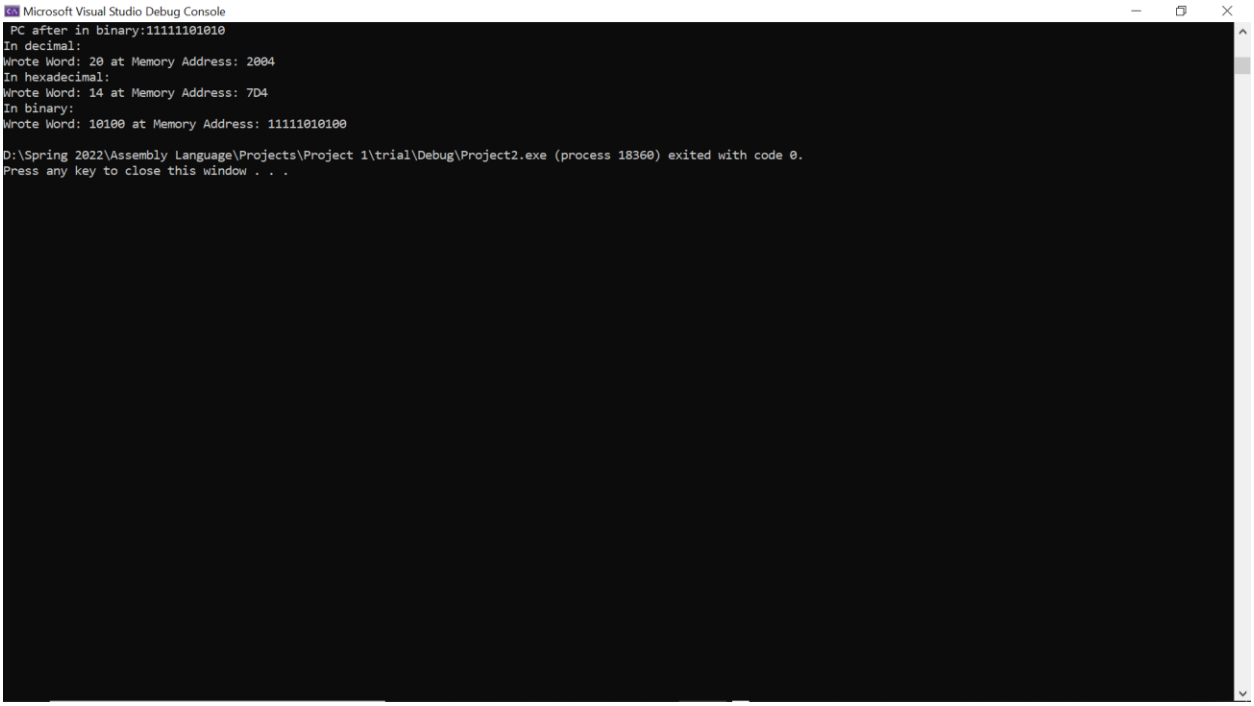

Microsoft Visual Studio Debug Console
 PC after in hexadecimal:7DE
 PC after in binary:11111011110
Modified Register x2 in decimal: 11
Modified Register x2 in hexadecimal: B
Modified Register x2 in binary: 1011
Instruction: BNE x2,x3,-8
Status: PC before in decimal:2014 PC before in hex:7DE PC before in binary:11111011110
PC after in decimal: 2006
 PC after in hexadecimal:7D6
 PC after in binary:11111010110
Instruction: ADD x10,x10,x2
Status: PC before in decimal:2006 PC before in hex:7D6 PC before in binary:11111010110
PC after in decimal: 2010
 PC after in hexadecimal:7DA
 PC after in binary:11111011010
Modified Register x10 in decimal: 21
Modified Register x10 in hexadecimal: 15
Modified Register x10 in binary: 10101
Instruction: ADDI x2,x2,1
Status: PC before in decimal:2010 PC before in hex:7DA PC before in binary:11111011010
PC after in decimal: 2014
 PC after in hexadecimal:7DE
 PC after in binary:11111011110
Modified Register x2 in decimal: 12
Modified Register x2 in hexadecimal: C
Modified Register x2 in binary: 1100
Instruction: BNE x2,x3,-8
Status: PC before in decimal:2014 PC before in hex:7DE PC before in binary:11111011110
PC after in decimal: 2006
 PC after in hexadecimal:7D6
 PC after in binary:11111010110
Instruction: ADD x10,x10,x2
Status: PC before in decimal:2006 PC before in hex:7D6 PC before in binary:11111010110
PC after in decimal: 2010
 PC after in hexadecimal:7DA
 PC after in binary:11111011010
Modified Register x10 in decimal: 33
Modified Register x10 in hexadecimal: 21
Modified Register x10 in binary: 100001
Instruction: ADDI x2,x2,1
Status: PC before in decimal:2010 PC before in hex:7DA PC before in binary:11111011010
PC after in decimal: 2014
 PC after in hexadecimal:7DE


Microsoft Visual Studio Debug Console
PC after in binary:11111011110
Modified Register x2 in decimal: 13
Modified Register x2 in hexadecimal: D
Modified Register x2 in binary: 1101
Instruction: BNE x2,x3,-8
Status: PC before in decimal:2014 PC before in hex:7DE PC before in binary:11111011110
PC after in decimal: 2006
PC after in hexadecimal:7D6
PC after in binary:11111010110
Instruction: ADD x10,x10,x2
Status: PC before in decimal:2006 PC before in hex:7D6 PC before in binary:11111010110
PC after in decimal: 2010
PC after in hexadecimal:7DA
PC after in binary:11111011010
Modified Register x10 in decimal: 46
Modified Register x10 in hexadecimal: 2E
Modified Register x10 in binary: 101110
Instruction: ADDI x2,x2,1
Status: PC before in decimal:2010 PC before in hex:7DA PC before in binary:11111011010
PC after in decimal: 2014
PC after in hexadecimal:7DE
PC after in binary:11111011110
Modified Register x2 in decimal: 14
Modified Register x2 in hexadecimal: E
Modified Register x2 in binary: 1110
Instruction: BNE x2,x3,-8
Status: PC before in decimal:2014 PC before in hex:7DE PC before in binary:11111011110
PC after in decimal: 2006
PC after in hexadecimal:7D6
PC after in binary:11111010110
Instruction: ADD x10,x10,x2
Status: PC before in decimal:2006 PC before in hex:7D6 PC before in binary:11111010110
PC after in decimal: 2010
PC after in hexadecimal:7DA
PC after in binary:11111011010
Modified Register x10 in decimal: 60
Modified Register x10 in hexadecimal: 3C
Modified Register x10 in binary: 111100
Instruction: ADDI x2,x2,1
Status: PC before in decimal:2010 PC before in hex:7DA PC before in binary:11111011010
PC after in decimal: 2014
PC after in hexadecimal:7DE
PC after in binary:11111011110


Microsoft Visual Studio Debug Console
Modified Register x2 in decimal: 15
Modified Register x2 in hexadecimal: F
Modified Register x2 in binary: 1111
Instruction: BNE x2,x3,-8
Status: PC before in decimal:2014 PC before in hex:7DE PC before in binary:11111011110
PC after in decimal: 2006
PC after in hexadecimal:7D6
PC after in binary:11111010110
Instruction: ADD x10,x10,x2
Status: PC before in decimal:2006 PC before in hex:7D6 PC before in binary:11111010110
PC after in decimal: 2010
PC after in hexadecimal:7DA
PC after in binary:11111011010
Modified Register x10 in decimal: 75
Modified Register x10 in hexadecimal: 4B
Modified Register x10 in binary: 1001011
Instruction: ADDI x2,x2,1
Status: PC before in decimal:2010 PC before in hex:7DA PC before in binary:11111011010
PC after in decimal: 2014
PC after in hexadecimal:7DE
PC after in binary:11111011110
Modified Register x2 in decimal: 16
Modified Register x2 in hexadecimal: 10
Modified Register x2 in binary: 10000
Instruction: BNE x2,x3,-8
Status: PC before in decimal:2014 PC before in hex:7DE PC before in binary:11111011110
PC after in decimal: 2006
PC after in hexadecimal:7D6
PC after in binary:11111010110
Instruction: ADD x10,x10,x2
Status: PC before in decimal:2006 PC before in hex:7D6 PC before in binary:11111010110
PC after in decimal: 2010
PC after in hexadecimal:7DA
PC after in binary:11111011010
Modified Register x10 in decimal: 91
Modified Register x10 in hexadecimal: 5B
Modified Register x10 in binary: 1011011
Instruction: ADDI x2,x2,1
Status: PC before in decimal:2010 PC before in hex:7DA PC before in binary:11111011010
PC after in decimal: 2014
PC after in hexadecimal:7DE
PC after in binary:11111011110
Modified Register x2 in decimal: 17


Microsoft Visual Studio Debug Console
Modified Register x2 in hexadecimal: 11
Modified Register x2 in binary: 10001
Instruction: BNE x2,x3,-8
Status: PC before in decimal:2014 PC before in hex:7DE PC before in binary:11111011110
PC after in decimal: 2006
PC after in hexadecimal:7D6
PC after in binary:11111010110
Instruction: ADD x10,x10,x2
Status: PC before in decimal:2006 PC before in hex:7D6 PC before in binary:11111010110
PC after in decimal: 2010
PC after in hexadecimal:7DA
PC after in binary:11111011010
Modified Register x10 in decimal: 108
Modified Register x10 in hexadecimal: 6C
Modified Register x10 in binary: 1101100
Instruction: ADDI x2,x2,1
Status: PC before in decimal:2010 PC before in hex:7DA PC before in binary:11111011010
PC after in decimal: 2014
PC after in hexadecimal:7DE
PC after in binary:11111011110
Modified Register x2 in decimal: 18
Modified Register x2 in hexadecimal: 12
Modified Register x2 in binary: 10010
Instruction: BNE x2,x3,-8
Status: PC before in decimal:2014 PC before in hex:7DE PC before in binary:11111011110
PC after in decimal: 2006
PC after in hexadecimal:7D6
PC after in binary:11111010110
Instruction: ADD x10,x10,x2
Status: PC before in decimal:2006 PC before in hex:7D6 PC before in binary:11111010110
PC after in decimal: 2010
PC after in hexadecimal:7DA
PC after in binary:11111011010
Modified Register x10 in decimal: 126
Modified Register x10 in hexadecimal: 7E
Modified Register x10 in binary: 1111110
Instruction: ADDI x2,x2,1
Status: PC before in decimal:2010 PC before in hex:7DA PC before in binary:11111011010
PC after in decimal: 2014
PC after in hexadecimal:7DE
PC after in binary:11111011110
Modified Register x2 in decimal: 19
Modified Register x2 in hexadecimal: 13


Microsoft Visual Studio Debug Console
Modified Register x2 in binary: 10011
Instruction: BNE x2,x3,-8
Status: PC before in decimal:2014 PC before in hex:7DE PC before in binary:11111011110
PC after in decimal: 2006
PC after in hexadecimal:7D6
PC after in binary:11111010110
Instruction: ADD x10,x10,x2
Status: PC before in decimal:2006 PC before in hex:7D6 PC before in binary:11111010110
PC after in decimal: 2010
PC after in hexadecimal:7DA
PC after in binary:11111011010
Modified Register x10 in decimal: 145
Modified Register x10 in hexadecimal: 91
Modified Register x10 in binary: 10010001
Instruction: ADDI x2,x2,1
Status: PC before in decimal:2010 PC before in hex:7DA PC before in binary:11111011010
PC after in decimal: 2014
PC after in hexadecimal:7DE
PC after in binary:11111011110
Modified Register x2 in decimal: 20
Modified Register x2 in hexadecimal: 14
Modified Register x2 in binary: 10100
Instruction: BNE x2,x3,-8
Status: PC before in decimal:2014 PC before in hex:7DE PC before in binary:11111011110
PC after in decimal: 2018
PC after in hexadecimal:7E2
PC after in binary:11111100010
Instruction: SW x2,10(x1)
Status: PC before in decimal:2018 PC before in hex:7E2 PC before in binary:11111100010
PC after in decimal: 2022
PC after in hexadecimal:7E6
PC after in binary:11111100110
In decimal:
Wrote Word: 20 at Memory Address: 2000
In hexadecimal:
Wrote Word: 14 at Memory Address: 7D0
In binary:
Wrote Word: 10100 at Memory Address: 11111010000
Instruction: SW x3,14(x1)
Status: PC before in decimal:2022 PC before in hex:7E6 PC before in binary:11111100110
PC after in decimal: 2026
PC after in hexadecimal:7EA
PC after in binary:11111101010



The program then runs and shows the contents of registers are displayed on the console.

Testing Programs:

1- P1 and Test 1 are the first test case and they are used to mimic the behavior of a C-code. Given two integers in the memory num1 and num2 where num2>num1, it calculates the summation of integers between them including num1 and excluding num2. It is used to test loading, storing, branching, and adding commands.

2- P2 and Test 2 are the second test case and they are used to identify whether two numbers are divisible by each other or not. First it identifies which one of them is bigger and decrements it with the value of the smaller value and if the remainder is zero then they are divisible and a flag is set to 1. Otherwise, they are not divisible and the flag is set to 0. It is used to test loading, storing, branching, setting, and subtracting.

3- P3 and Test 3 are the third test case and they are used to test logical and Boolean functions like loading, storing, anding, oring, xoring, and shifting left and right.