**COAL Lab-06**

**Load & Store Instruction Assembly Code**

**------------------------------------------------------**

**Name:** Saad Nisar Butt

**Reg. no:** cs211246

**Class:** BSCS-3C-1

**------------------------------------------------------**

**Literature Review:**

**Load Instructions:**

Load instructions are used to move data from memory to registers (before operation). Loads are encoded in the I-type format. The effective byte address is obtained by adding register rs1 to the sign-extended 12-bit offset. Loads copy a value from memory to register rd. The assembly representation for load instructions are:

**lw (destination\_register), (offset)(source\_register)**

or

**lw (rd), offset(rs1)**

The LW instruction loads a 32-bit value from memory into rd. LH loads a 16-bit value from memory, then sign-extends to 32-bits before storing in rd. LHU loads a 16-bit value from memory but then zero extends to 32-bits before storing in rd. LB and LBU are defined analogously for 8-bit values.

**Store Instructions:**

Store instructions are used to move data from registers to memory (after operation). Stores are encoded in the S-type format. The effective byte address is obtained by adding register rs1 to the sign-extended 12-bit offset. Stores copy the value in register rs2 to memory.. The assembly representation for store instructions are:

**sw (source\_register\_2), (offset)(source\_register\_1)**

or

**sw (rs2), offset(rs1)**

The SW instruction stores a 32-bit value from the low bits of register rs2 to memory. SH stores a 16-bit value from the low bits of register rs2 to memory. SB stores a 8-bit value from the low bits of register rs2 to memory.

**Lab Exercise 1:**

**Task:**

Run the below assembly code on Venus Simulator

li s0, 0x12345678 # Data to be store

li s1, 0x00000020 # memory address

sb s0, 0x0(s1)

sh s0, 0x4(s1)

sw s0, 0x8(s1)

**sb s0, 0x0(s1)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0000000 | 01000 | 01001 | 000 | 00000 | 0100011 |

Machine Code: 0000 0000 1000 0100 1000 0000 0010 0011

Hexadecimal Code: 00848023

**sh s0, 0x4(s1)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0000000 | 01000 | 01001 | 001 | 00100 | 0100011 |

Machine Code: 0000 0000 1000 0100 1001 0010 0010 0011

Hexadecimal Code: 00849223

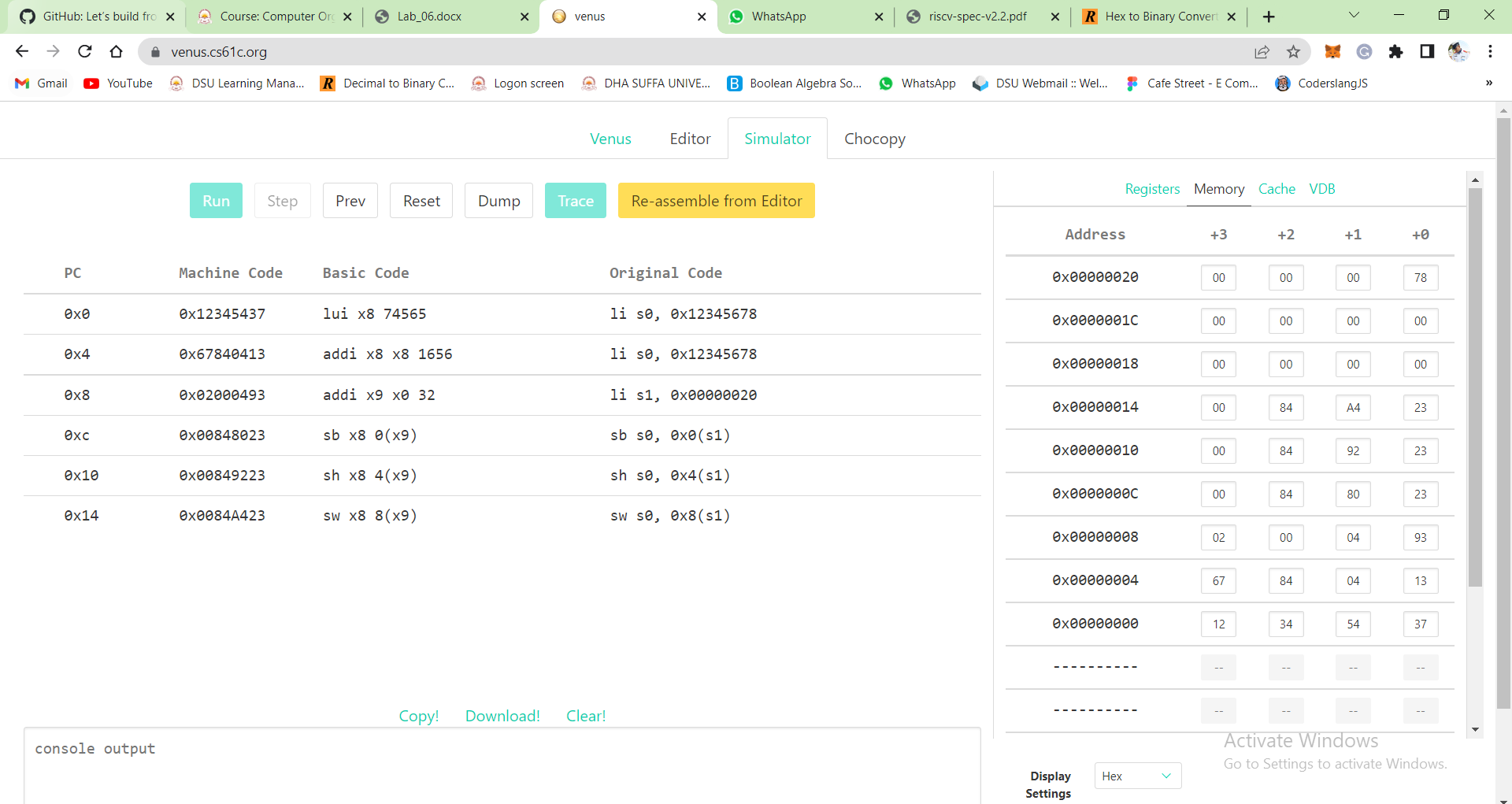
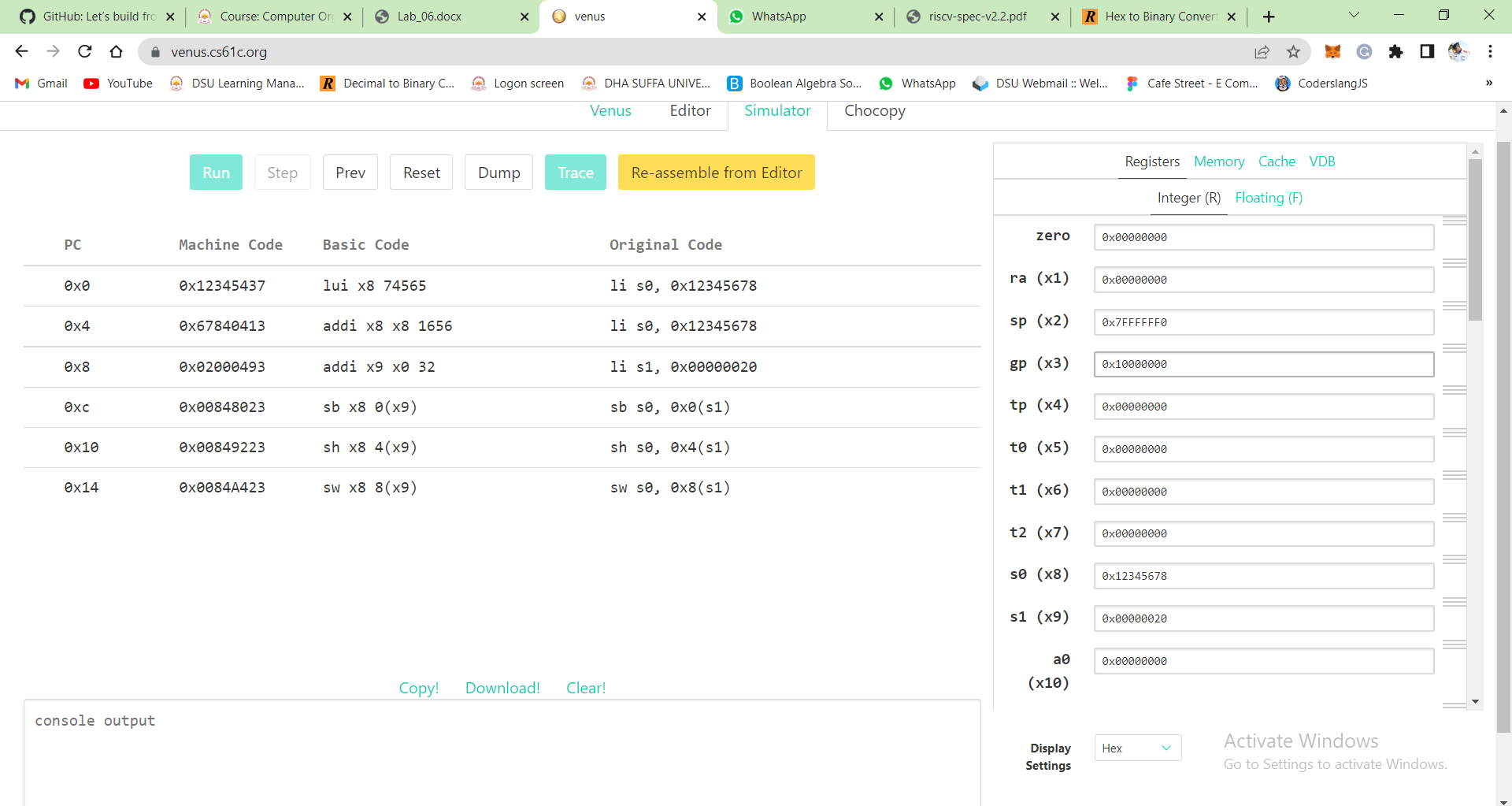
**sw s0, 0x8(s1)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0000000 | 01000 | 01001 | 010 | 01000 | 0100011 |

Machine Code: 0000 0000 1000 0100 1010 0100 0010 0011

Hexadecimal Code: 0084A423

**Venus Simulation:**

****

**Lab Exercise 2:**

**Task:**

Run the below assembly code on Venus Simulator

lb t0, 0x0(x0)

lbu t1, 0x4(x0)

lh t2, 0x8(x0)

lhu s0, 0xC(x0)

lw s1, 0x10(x0)

**lb t0, 0x0(x0)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 000000000000 | 00000 | 000 | 00101 | 0000011 |

Machine Code: 0000 0000 0000 0000 0000 0010 1000 0011

Hexadecimal Code: 00000283

**lbu t1, 0x4(x0)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 000000000100 | 00000 | 100 | 00110 | 0000011 |

Machine Code: 0000 0000 0100 0000 0100 0011 0000 0011

Hexadecimal Code: 00404303

**lh t2, 0x8(x0)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 000000001000 | 00000 | 001 | 00111 | 0000011 |

Machine Code: 0000 0000 1000 0000 0001 0011 1000 0011

Hexadecimal Code: 00801383

**lhu s0, 0xC(x0)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 000000001100 | 00000 | 101 | 01000 | 0000011 |

Machine Code: 0000 0000 1100 0000 0101 0100 0000 0011

Hexadecimal Code: 00C05403

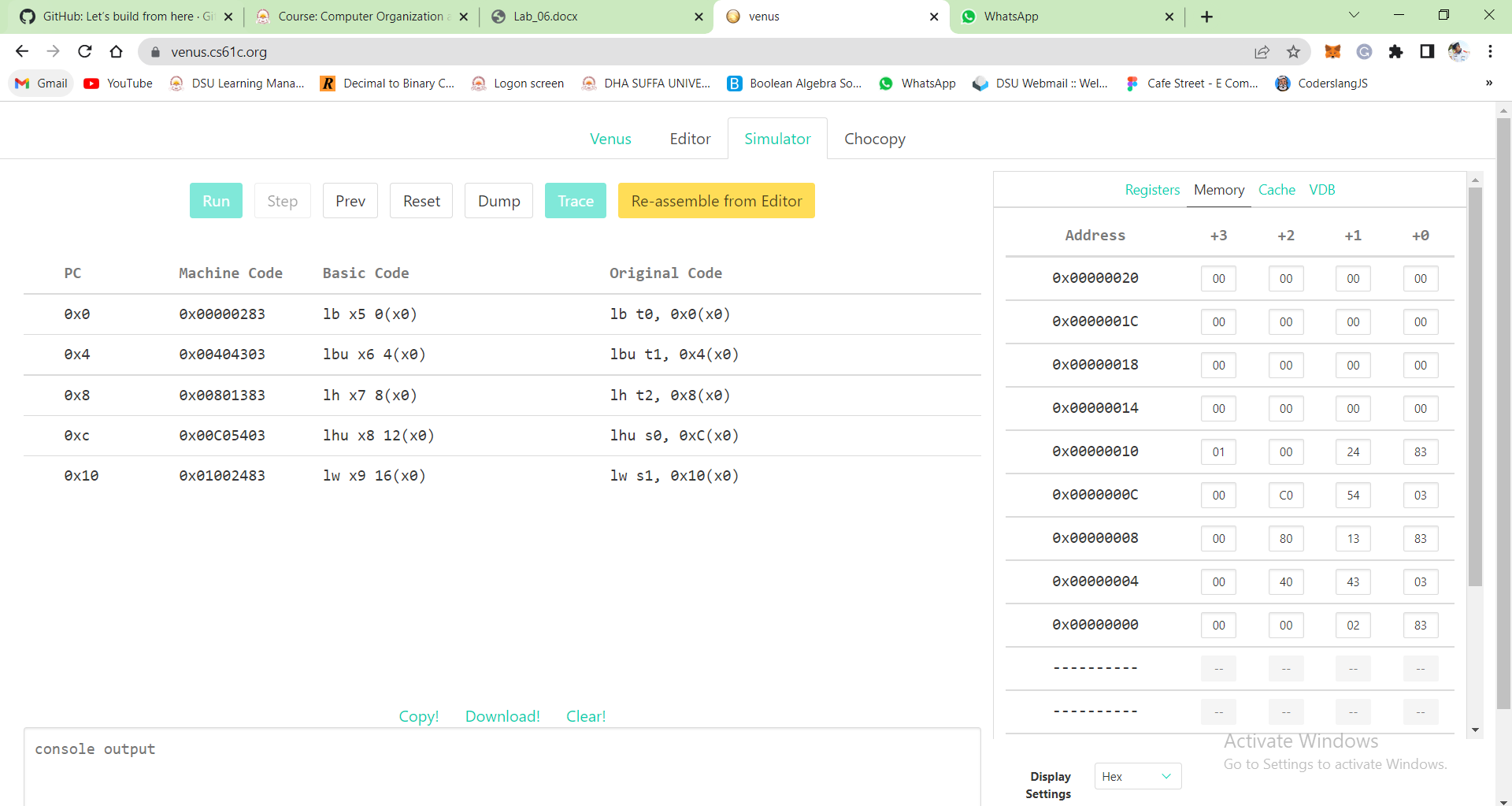
**lw s1, 0x10(x0)**

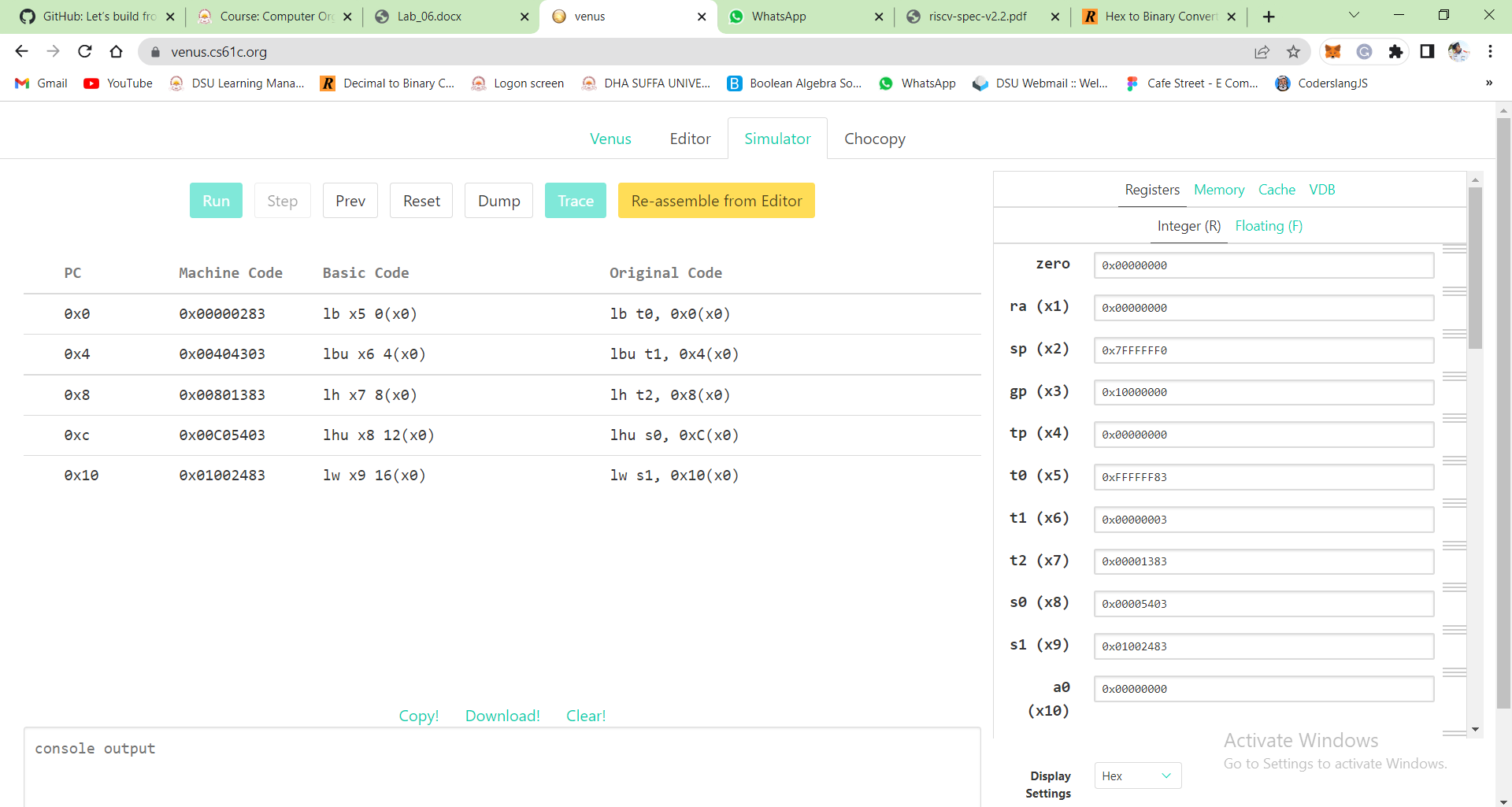
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 000000010000 | 00000 | 010 | 01001 | 0000011 |

Machine Code: 0000 0001 0000 0000 0010 0100 1000 0011

Hexadecimal Code: 01002483

**Venus Simulation**

****



**InLab Tasks:**

**Task 1:**

Write down a simple assembly program to add, and subtract two integer numbers and store their result into different memory locations. Stimulate the code on Venus.

**RISC-V Assembly Code:**

addi x5, x0, 4 # storing 4 in register x5

addi x6, x0, 6 # storing 6 in register x6

add s0, x5, x6 # addition

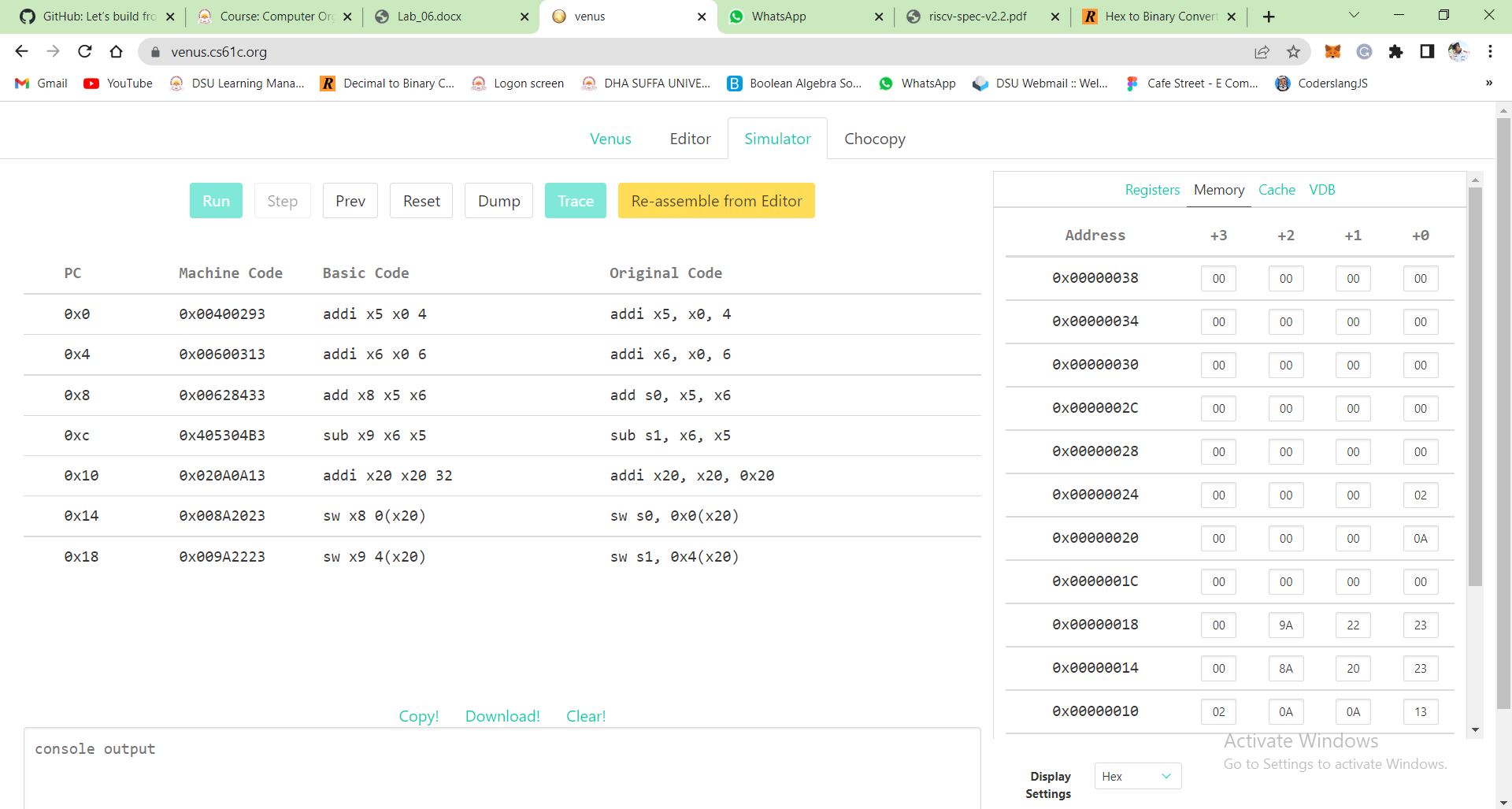
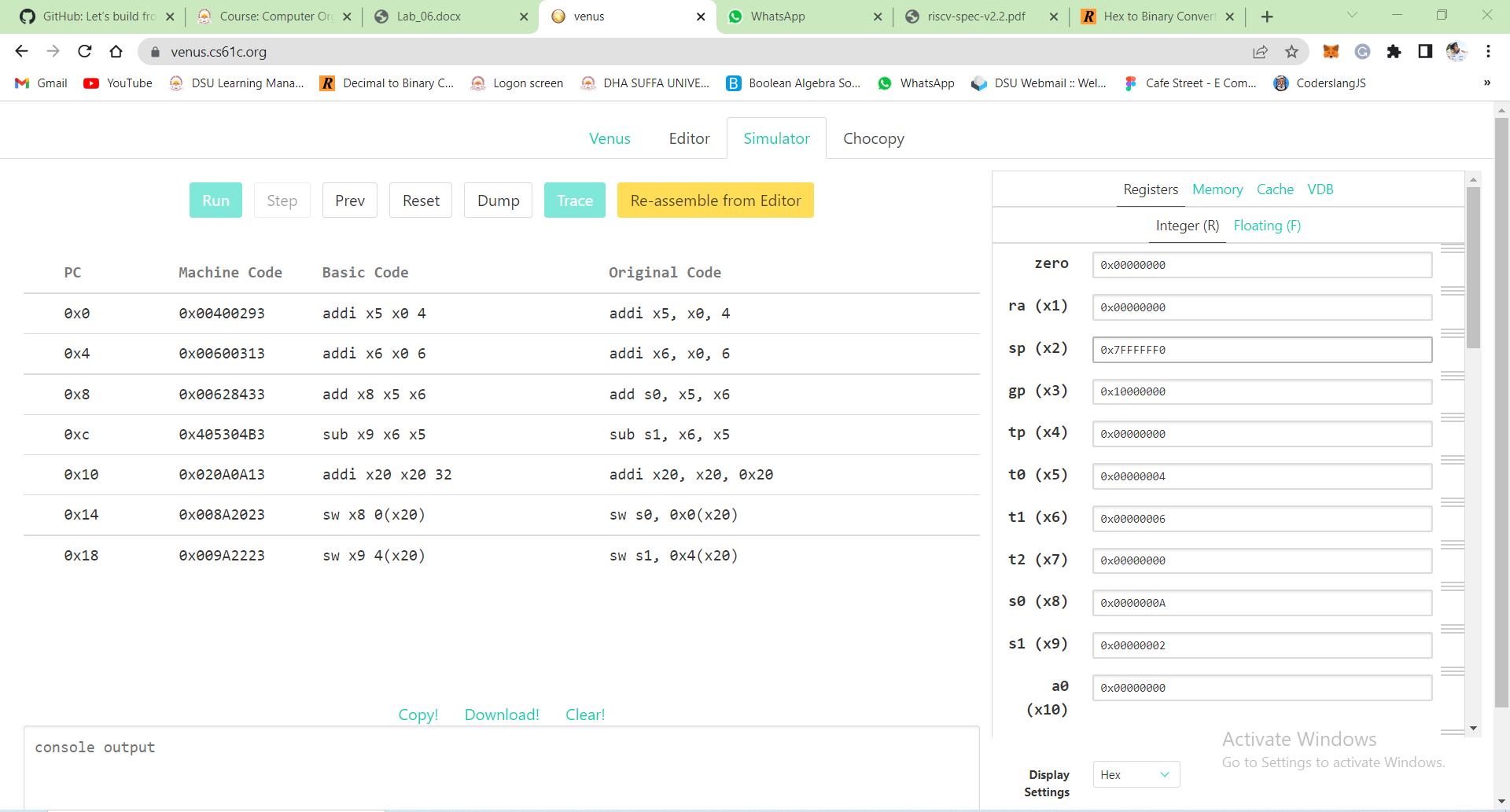
sub s1, x6, x5 # subtraction

addi x20, x20, 0x20 # setting memory’s base address

sw s0, 0x0(x20) # storing s0’s added value in memory

sw s1, 0x4(x20) # storing s1’s subtracted value in memory

**Venus Simulation:**



**Task 2:**

Write down a simple assembly program to load the contents from memory into registers and perform the logical operations on them. Stimulate the code on Venus.

**RISC-V Assembly Code:**

addi x20, x20, 0x20 # setting memory’s base address

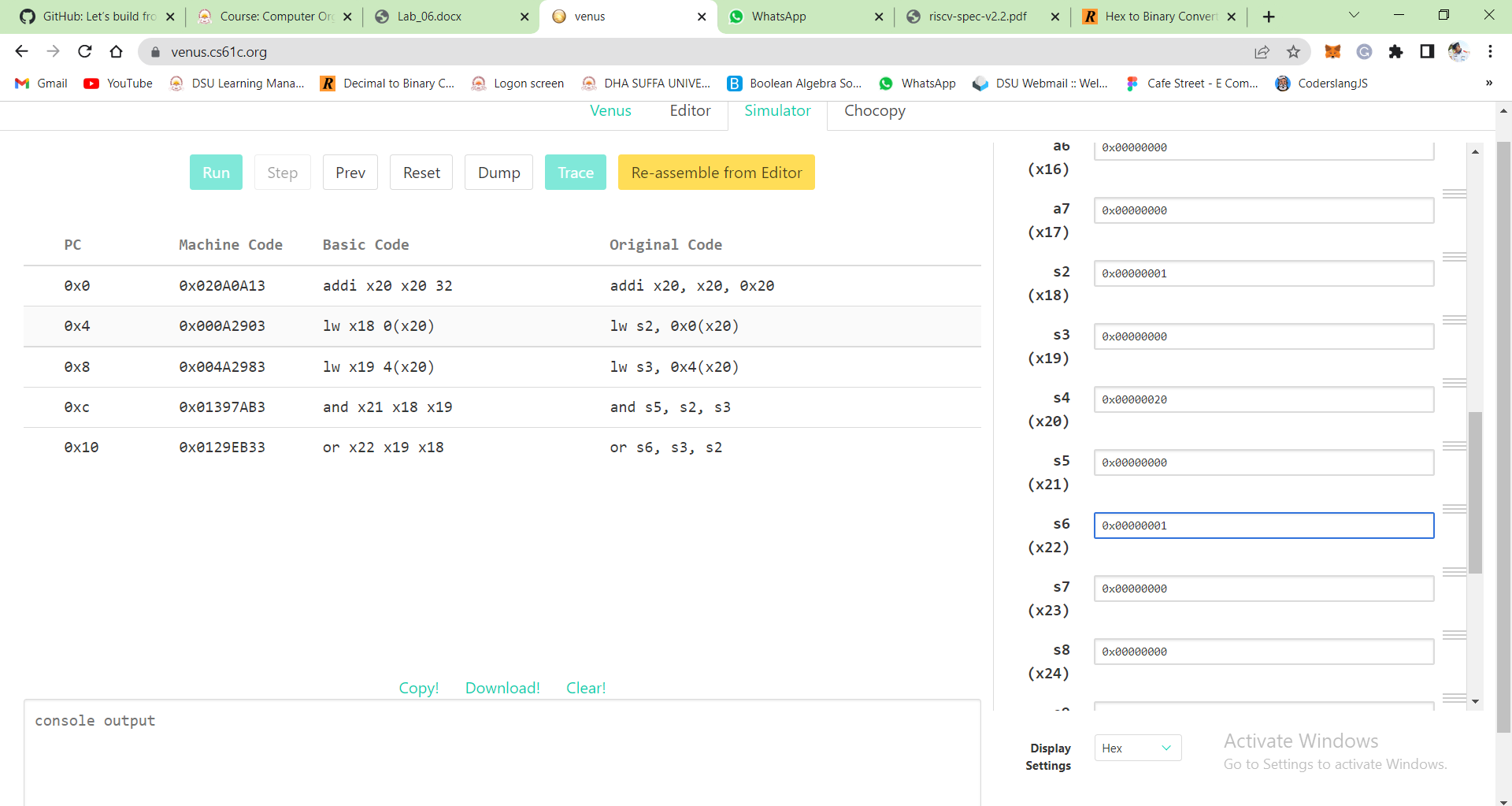
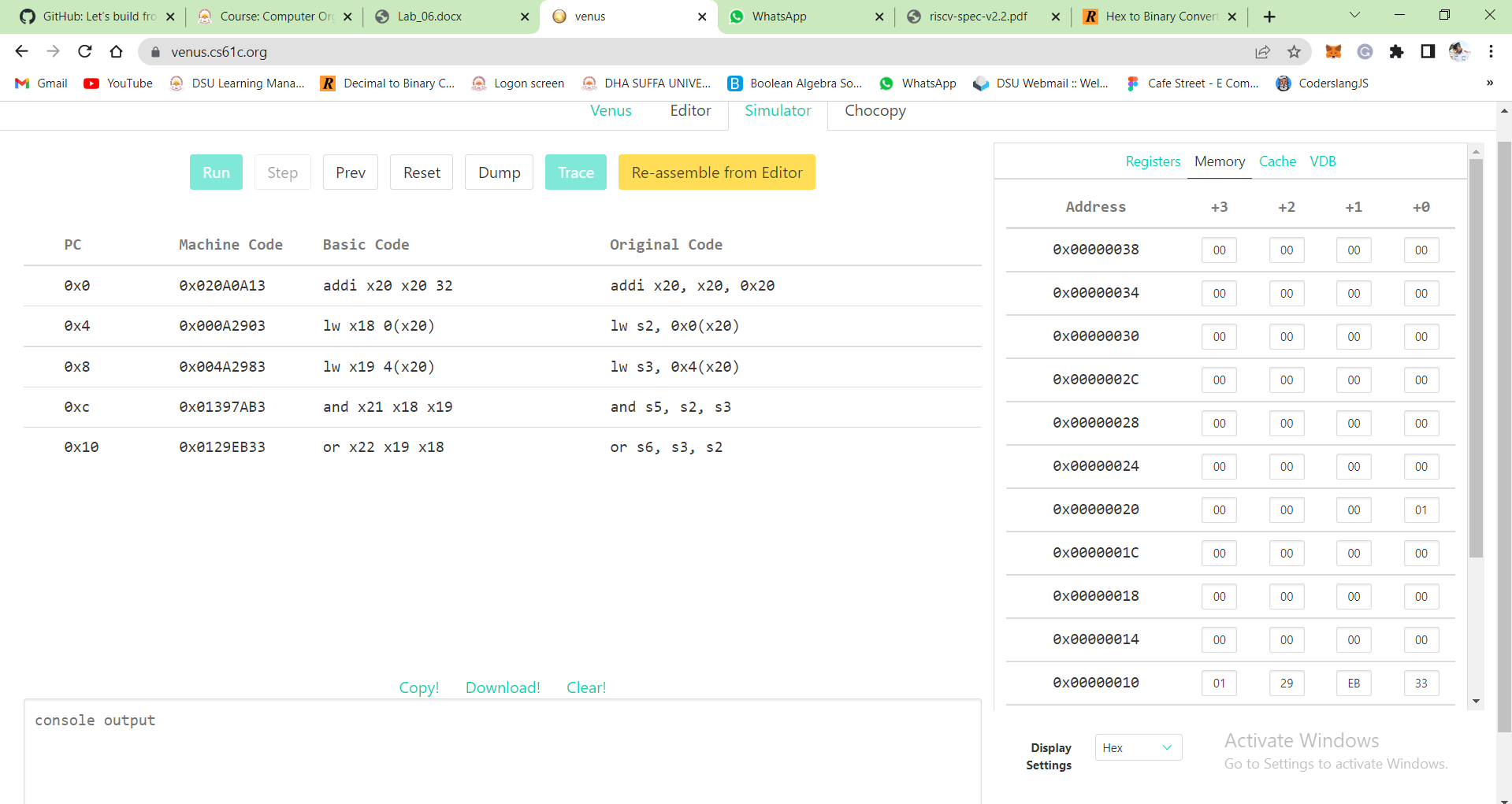
lw s2, 0x0(x20) # loading data from memory into register

lw s3, 0x4(x20) # loading data from memory into register

and s5, s2, s3 # ANDing data loaded from memory

or s6, s3, s2 # Oring data loaded from memory

**Venus Simulation:**

****

**\* -------------------------------------- \***