LAB # 3
Introduction to Assembly Language & Venus Simulator

Objective:

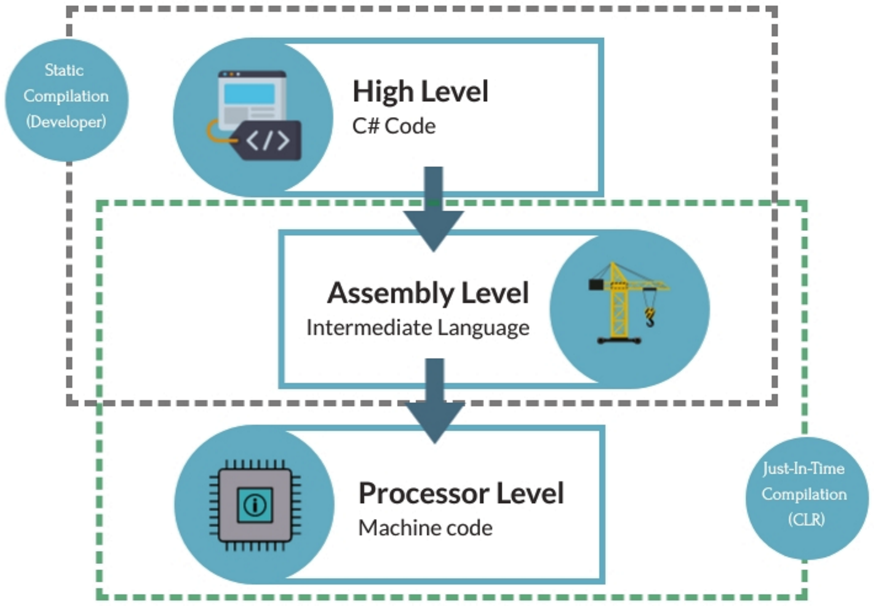
- Get Familiar with Assembly Language & Venus Simulator

System Module (Hardware/Software)

- Venus Simulator

- **Theoretical Background:**

Introduction:



Assembly Language:
Assembly language is the human-readable representation of the computer's native language. Each assembly language instruction specifies both the operation to perform and the operands on which to operate. We introduce simple arithmetic instructions and show how these operations are written in assembly language. We then define the RISC-V instruction operands: registers, memory, and constants.

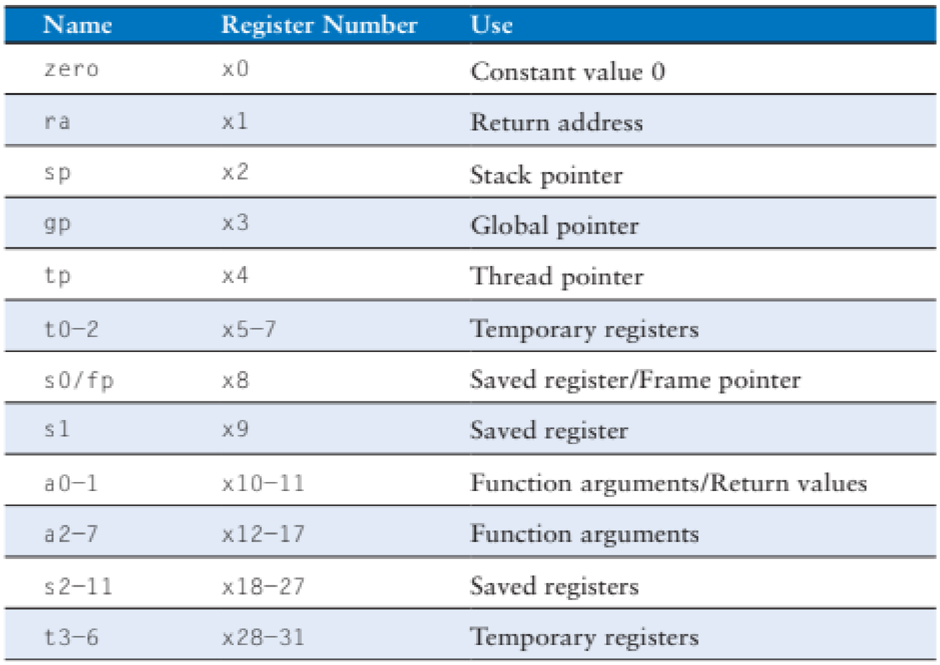
High - Level Code	Assembly Code
a = b + c	add a, b, c
a = b - c	sub a, b, c
a = b + c - d	add t, b, c sub a, t, d

Venus:

Venus is an emulator for RISC-V computers. It allows users to run programs written in RISC-V assembly language without having access to a computer running a RISC-V chip.

Registers:

Instructions need to access operands quickly so that they can run fast, but operands stored in memory take a long time to retrieve. Therefore, most architectures specify a small number of registers that hold commonly used operands. The RISC-V architecture has 32 registers, called the register set, stored in a small multi ported memory called a register file. The fewer the registers, the faster they can be accessed.



Name	Register Number	Use
zero	x0	Constant value 0
ra	x1	Return address
sp	x2	Stack pointer
gp	x3	Global pointer
tp	x4	Thread pointer
t0–2	x5–7	Temporary registers
s0/fp	x8	Saved register/Frame pointer
s1	x9	Saved register
a0–1	x10–11	Function arguments/Return values
a2–7	x12–17	Function arguments
s2–11	x18–27	Saved registers
t3–6	x28–31	Temporary registers

High - Level Code	RISC-V Assembly Code
a = b + c	add s0, s1, s2
a = b - c	sub s0, s1, s2
a = b + c - d	add t0, s1, s2 sub s0, t0, s3

Lab Exercise 1:

Translate the Given High Level Code to Assembly Language

High - Level Code	RISC-V Assembly Code

a = b − c;
f = (g + h) − (i + j);	

Machine Code:

Assembly language is convenient for humans to read. However, digital circuits understand only 1's and 0's. Therefore, a program written in assembly language is translated from mnemonics to a representation using only 1's and 0's, called machine language. RISC-V makes the compromise of defining four main instruction formats: R-type, I-type, S/B-type, and U/J-type.

31 27 26 25	24 20	19 15	14 12	11 7	6 0	
funct7	rs2	rs1	funct3	rd	opcode	R-type
imm[11:0]		rs1	funct3	rd	opcode	I-type
imm[11:5]	rs2	rs1	funct3	imm[4:0]	opcode	S-type
imm[12|10:5]	rs2	rs1	funct3	imm[4:1|11]	opcode	B-type
imm[31:12]				rd	opcode	U-type
imm[20|10:1|11|19:12]				rd	opcode	J-type

Lab Exercise 2:

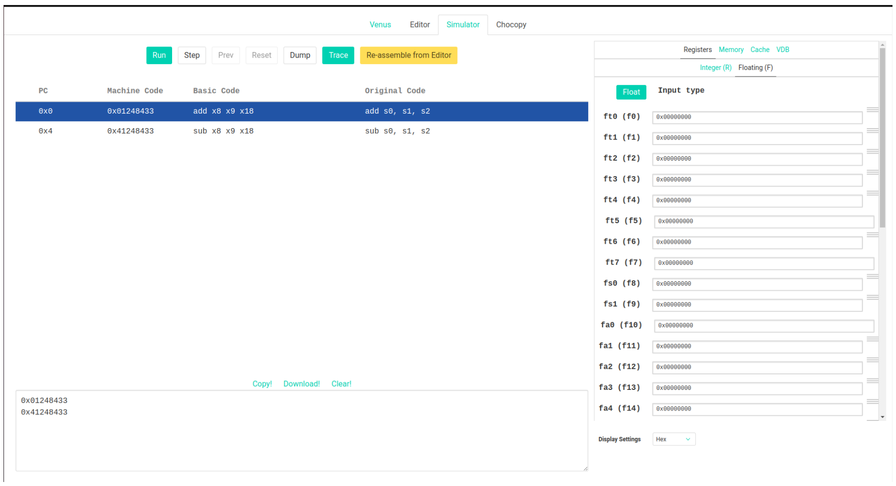
Convert the following RISC-V Assembly code into Machine Code

```
add x9, x5, x6
andi x10, x8, 0x6
or x5, x6, x7
slli x10, x6, 0x8
```

Procedure:

Venus User Interface:

1) Open the given link in Google Chrome. https://venus.cs61c.org/
2) Go to the Editor Tab to write the assembly code.
3) Go to the Simulator Tab to simulate your assembly code.
4) The Simulator screen is divided into three parts:
   a) **Register Display:** This shows the contents (bit patterns in hex) of all 32 general purpose registers, the floating point registers, and also the contents of memory.
   b) **Text Display:** This shows the assembly language program source, the machine instructions (bit patterns in hex) they correspond to, and the addresses of their memory locations.
   c) **Console:** This shows the machines code dump from the simulator



```
PC      Machine Code   Basic Code        Original Code
0x0     0x01248433     add x8 x9 x18     add s0, s1, s2
0x4     0x41248433     sub x8 x9 x18     sub s0, s1, s2
```

```
0x01248433
0x41248433
```

```
Input type
ft0 (f0)   0x00000000
ft1 (f1)   0x00000000
ft2 (f2)   0x00000000
ft3 (f3)   0x00000000
ft4 (f4)   0x00000000
ft5 (f5)   0x00000000
ft6 (f6)   0x00000000
ft7 (f7)   0x00000000
fs0 (f8)   0x00000000
fs1 (f9)   0x00000000
fa0 (f10)  0x00000000
fa1 (f11)  0x00000000
fa2 (f12)  0x00000000
fa3 (f13)  0x00000000
fa4 (f14)  0x00000000
```

Display Settings: Hex

Running a program:

1) After writing your assembly code in Editor Tab go to Simulator Tab.
2) Click on **Assemble & Simulate from Editor** button.
3) To run the whole code at once click on **Run** button.
4) To run the code step by step click on the **Step** button.
5) To dump the machine code click on the **Dump** button.
6) To Reset the value to default click on the **Reset** button.
7) To go back to the previous step click on the **Prev** button.
8) To reassemble the code after changes click on the **Re-assemble from Editor** button.

In-Lab Tasks:

1. **Convert the Given High Level Code on RISC-V Assembly and run it on Venus Simulator. Also write down its Machine Code**
   a) a = a + 4;
      b = a − 12;

   b) if (g < h)
      g = g + 1;
      else
      h = h − 1;

Post-Lab Tasks:

1. **Write down a simple C program to add, multiply and divide two integer numbers. Convert the C code into assembly and machine code and stimulate it on Venus.**
2. **Write down a simple C program to find out the prime numbers between 1-10 and give the final count of prime numbers. Convert the C code into assembly and machine code and stimulate it on Venus.**

Conclusion:

What have you learnt from this lab?

____

____

____

____

Learning Outcomes:

Upon successful completion of the lab, students will be able to:

LO1: Run the basic assembly code on Venus Simulator.