

#Team Member

#Qingyun Zhang 2095754

#Sicong Liu 2089129

#Ying Lu 2093558

The program specifically handles two integers, prints their values, addresses, and results of arithmetic operations, and concludes by safely exiting.

The program is structured into two main sections: the `.data` section for declaring variables and the `.text` section containing the executable instructions.

- Data Section

Variables Declared:

- ``input1``: Initially set to 5.
- ``input2``: Initially set to 3.

- Text Section

Global Label:

- ``_start``: The entry point of the program.

- Operations and System Calls:

1. Print the Value of ``input1``:

- Load the address and value of ``input1`` into registers.
- Use a system call to print the integer value.

2. Print a Newline Character:

- Output a newline to separate outputs clearly.

3. Print the Address of ``input1``:

- Reload the address of ``input1`` and print it.

4. Print the Value of ``input2``:

- Similar to ``input1``, load and print the value and address of ``input2``.

5. Increment and Print New Values for ``input1`` and ``input2``:

- Add 1 to the values of ``input1`` and ``input2``.
- Store the new values back into their respective addresses.
- Print these new incremented values.

6. Calculate and Print Sum of ``input1 + 1`` and ``input2 + 1``:

- Load the incremented values from memory, compute their sum, and print the result.

7. Store the Sum in ``input1``'s Address and Print the Address:

- Store the computed sum back at the address of `input1`.
- Print the address where the sum has been stored.

## 8. Exit the Program:

- Perform a system call to exit the program cleanly.

This RISC-V assembly program demonstrates fundamental assembly operations including loading and storing data, arithmetic calculations, and performing system calls for printing and exiting. The program efficiently uses system calls to interact with the system for printing integers, characters, and exiting. It also manipulates memory directly to store and retrieve values, showcasing typical low-level programming tasks in an assembly language.

## ● Screenshot

The screenshot displays a RISC-V assembly simulator interface. At the top, there are control buttons: Run, Step, Prev, Reset, Dump, Trace, and Re-assemble from Editor. Below these is a table showing the execution progress with columns for PC, Machine Code, Basic Code, and Original Code.

PC	Machine Code	Basic Code	Original Code
0x0	0x10000597	auipc x11 65536	la a1, input1 # Load address of input1 into a1
0x4	0x00058593	addi x11 x11 0	la a1, input1 # Load address of input1 into a1
0x8	0x0005A603	lw x12 0(x11)	lw a2, 0(a1) # Load value of input1 into a2
0xc	0x00100513	addi x10 x0 1	li a0, 1 # System call number for print integer
0x10	0x00060593	addi x11 x12 0	mv a1, a2 # Move value to a1
0x14	0x00000073	ecall	ecall # System call to print input1 value

Below the table, there are links: Copy!, Download!, and Clear!. A scrollable output window shows the following values:

```

5
268435456
3
268435460
6
4
10

```

On the right side, there is a register window showing the values of registers a0 through a7 and s2. The values are:

- a0 (x9): 10
- a1 (x10): 268435456
- a2 (x11): 10
- a3 (x12): 4
- a4 (x13): 0
- a5 (x14): 0
- a6 (x15): 0
- a7 (x16): 0
- s2 (x17): 0

At the bottom right, there is a 'Display Settings' section with a dropdown menu set to 'Decimal'.