Course: Intro to RISC-V  
Task RISC-V Instruction Format

**Repo Link**

<https://github.com/YazanHussnain-10x/su21-lab-starter>

**Source Code**

<https://github.com/YazanHussnain-10x/su21-lab-starter/tree/main/lab03>

**Exercise 3: Translating from C to RISC-V**

1. The register representing the variable **k** is **“t0”**.
2. The register representing the variable **sum** is **“s0”**.
3. The registers acting as pointers to the source and dest arrays are **“s1”**, **“s2”**.
4. Assembly for the loop found in the C code is:

| loop:  slli s3, t0, 2  add t1, s1, s3  lw t2, 0(t1)  beq t2, x0, exit  add a0, x0, t2  addi sp, sp, -8  sw t0, 0(sp)  sw t2, 4(sp)  jal fun  lw t0, 0(sp)  lw t2, 4(sp)  addi sp, sp, 8  add t2, x0, a0  add t3, s2, s3  sw t2, 0(t3)  add s0, s0, t2  addi t0, t0, 1  jal x0, loop |
| --- |

1. As each int is 4 bytes, **"s3"** is used to calculate the offset of the current index in the array (**s3 = t0 \* 4**).

The location of the current index in the source array is determined by "**t1**" (**t1 = s1 + s3**).

The value at the current index of the source array is loaded using "**t2**" (**lw t2, 0(t1)**).

The address of the current index in the dest array is determined by "**t3**" (**t3 = s2 + s3**).

The output of **fun(source[k])** is stored in "**t2**" and is located at the current index of the dest array (**sw t2, 0(t3)**).

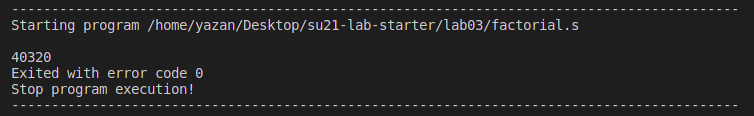
**Exercise 4: Factorial**

When a function call to function **“factorial”** is made, the first thing we need to do is to store the return address to register **“ra ''**, because we have to come back from the function to the next instruction from which function call is made. One more thing that needs to be stored on stack is the function arguments, because the recursive function of **“factorial”** is written, therefore, we need to store the variable used in each function to the stack.

Next we check the base case, if it's true then return 1 and jump to the return address set by the current function call. If the condition for base case is not fulfilled then apply the recursive condition on arguments. After returning from each function call. We have to load the value of the return address from the stack to register **“ra”** to go back to the address set by the previous function call. Arguments of each function call also need to be restored from the stack. One thing that needs to be taken care of is to save the result of multiplication on each return from the function call.

| .globl factorial  .data  n: .word 9  .text  main:  la t0, n  lw a0, 0(t0)  jal ra, factorial  addi a1, a0, 0  addi a0, x0, 1  ecall # Print Result  addi a1, x0, '\n'  addi a0, x0, 11  ecall # Print newline  addi a0, x0, 10  ecall # Exit  factorial:  # YOUR CODE HERE  addi sp, sp, -8  sw ra, 0(sp)  sw a0, 4(sp)  li t0, 1  beq a0, t0, base\_case  addi a0, a0, -1  jal factorial  add t0, a0, x0  lw a0, 4(sp)  lw ra, 0(sp)  addi sp, sp, 8  mul a0, a0, t0  jr ra  base\_case:  li a0, 1  jr ra |
| --- |

**Output**



**Exercise 5: RISC-V Function calling with map**

**Output**

