COMP311 Extra Practice 6 Name: _____

Question 1: Advanced Procedure Calls and State Persistence Consider a procedure `process_data(int n)` that computes a result using the formula:

$$\text{result} = n + \text{transform\_value}(n - 1)$$

This procedure must preserve the value of `n` across the call to `transform_value`. Assume `transform_value` expects its argument in `a0` and returns its result in `a0`.

1.1. **Conceptual Fill-in-the-Blank:** To implement this procedure, `process_data` must store `n` (which is initially in `a0`) into a saved register (e.g., `s1`) before the function call. List the registers that must be saved on the stack in the prologue.

1.2. **Assembly Implementation:** Write the complete assembly for `process_data`. Include the prologue, the preservation of `n`, the call to `transform_value`, the addition, and the epilogue. (Hint: Use `addi` with the zero register `x0` to perform register-to-register moves).

Question 2: Bubble Sort — Fill in the Inner Loop The C code below sorts an integer array `A` of length `size` in increasing order.

```
for (int i = 0; i < size - 1; i++) {
  for (int j = 0; j < size - 1 - i; j++) {
    if (A[j] > A[j+1]) {
      int t = A[j+1];
      A[j+1] = A[j];
      A[j]  = t;
    }
  }
}
```

Register mapping:

Register	Contents
s0	Base address of array `A`
s1	`size`
t0	`i`
t1	`j`
t2	`size - 1`
t3	`size - 1 - i`
t4	Value of `A[j]`
t5	Byte address of `A[j]`
t6	Value of `A[j+1]`

The outer loop is provided below. Complete the inner loop and swap by filling in each blank.

```
bubble_sort:
  li  t0, 0      # i = 0
  addi t2, s1, -1    # t2 = size - 1

outer_loop:
  bge t0, t2, end_sort # if i >= size-1, exit
  li  t1, 0      # j = 0
  sub t3, t2, t0    # t3 = size-1-i
  j  inner_loop

outer_next:
  addi t0, t0, 1    # i++
  j  outer_loop

end_sort:

inner_loop:
  # (1) Exit inner loop if j >= size-1-i
  ________ t1, t3, outer_next

  # (2) Compute byte offset of A[j] (each int is 4 bytes)
  slli t5, t1, ________

  # (3) Compute address of A[j]
  ________ t5, s0, t5
```

```
26
27    # (4) Load A[j]
28    lw  t4, ________(t5)
29
30    # (5) Load A[j+1]
31    lw  t6, ________(t5)
32
33    # (6) If A[j] <= A[j+1], skip the swap
34    ________ t4, t6, inner_next
35
36    # --- Swap A[j] and A[j+1] ---
37    # (7) Write A[j+1]'s value into A[j]
38    sw  t6, ________(t5)
39
40    # (8) Write A[j]'s value into A[j+1]
41    sw  t4, ________(t5)
42
43  inner_next:
44    # (9) Increment j
45    ________ t1, t1, ________
46    j  inner_loop
```

Question 3: Short Answer 3.1. In blank (2) you wrote `slli t5, t1, 2` to compute the byte offset. A classmate suggests using `add t5, t1, t1` instead. Explain why this is incorrect.

3.2. What is the value of `t0` when `end_sort` is reached? What does the state of array `A` guarantee at that point?

To get more practice with assembly writing, check out the **RISC-V Assembly Practice** on the course website here: `https://github.com/kakiryan/comp-311-assembly-practice-problems`!