

Question 1: Problem 1: Load and Store Instructions Assume the following initial state:

- **Register values:** $x1 = 0x100$, $x2 = 200$, $x3 = 5$
- **Memory contents:** [20, 40, 60] starting at address $0x00000100$

Consider the following RISC-V instructions:

```
lw    x4, 0(x1)
addi  x4, x4, 10
sw    x4, 4(x1)
lw    x5, 4(x1)
```

1.1. What value is loaded into $x4$ after the first instruction?

1.2. What is the value of $x4$ after the second instruction?

1.3. What value is written to memory, and at what address?

1.4. What is the value of $x5$ at the end of execution?

1.5. What are the final values stored in memory at addresses $0x100$, $0x104$, and $0x108$?

Question 2: Problem 2: RISC-V Memory Model & Addressing Consider a RISC-V processor executing the following code snippet. Assume the registers initially contain $x10 = 0x2000$ and $x11 = 0x40$. The memory at address $0x2040$ contains the 32-bit word $0xDEADBEEF$.

```
0x00400000: addi x12, x10, 0x40
```

```

0x00400004: lw    x13, 0(x12)
0x00400008: slli x13, x13, 4
0x0040000C: sw    x13, 4(x12)

```

- 2.1. After the execution of the `lw` instruction, what is the value stored in the Program Counter (PC)?

- 2.2. Determine the effective memory address accessed by the `sw` instruction and the exact 32-bit value written to that address in hexadecimal.

- 2.3. If the memory system is Little-Endian, what is the value of the byte stored at memory address `0x2044` after the code completes?

Question 3: Problem 3: Bubble Sort (Inner Loop Translation) The code below sorts an integer array `A` of length `size` in increasing order.

Register mapping: `s0` = base address of `A`, `s1` = `size`, `t0` = `i`, `t1` = `j`, `t3` = `size - 1 - i`. Complete the inner loop and swap by filling in each blank **(1) through (9)**.

```

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

    # (4) Load A[j]
    lw t4, -----(t5)

    # (5) Load A[j+1]
    lw t6, -----(t5)

```

```

# (6) If A[j] <= A[j+1], skip the swap
----- t4, t6, inner_next

# Swap A[j] and A[j+1]
# (7) Write A[j+1]'s value into A[j]
sw t6, -----(t5)

# (8) Write A[j]'s value into A[j+1]
sw t4, -----(t5)

inner_next:
# (9) Increment j
----- t1, t1, 1
j inner_loop

```

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!>