RV32I Array Fundamentals Lab

> **Goal:** Build confidence manipulating small arrays in RV32I assembly. **Exercises:** 10 bite-sized prompts (8 easy, 2 medium). Estimated time ≈ 75–90 minutes.

Getting Started Checklist

- Use the RV32I simulator at https://riscv-simulator-five.vercel.app/.
- Every program should end with `li a7, 10` followed by `ecall` to exit.
- All data arrays live in the `.data` segment; code goes in `.text`.

Question 1 – Store a Constant

- **Task:** Allocate an integer array of four words. Store the constant value `7` into index `2`.
- **Tip:** Use `la` to fetch the base address, then `sw` with an offset of `8`.
- **C guide:**

```c
int arr[4] = {0};
arr[2] = 7;
```

Question 2 – Copy First Element

- **Task:** Given `.word 10, 20, 30, 40`, copy element `0` into element `3` (so index `3` also becomes `10`).
- **Hint:** Load with `lw` into a temp register, then store back with `sw`.
- **C guide:**

```c
int arr[4] = {10, 20, 30, 40};
arr[3] = arr[0];
```

Question 3 – Sum of Three Elements

- **Task:** Load the first three elements of an array and compute their sum in `a0`.
- **Requirement:** Use only one temporary register (`t0`) in addition to `a0`.
- **C guide:**

```c
int sum3(int *arr) {
    return arr[0] + arr[1] + arr[2];
}
```

Question 4 – Zero Out an Array

- **Task:** Initialize a four-element array to zero inside a loop.
- **Loop idea:** Start with an offset register initialized to `0`; increment by `4` each iteration.
- **C guide:**

```c
void clear(int *arr) {
    for (int i = 0; i < 4; i++) {
        arr[i] = 0;
    }
}
```

Question 5 – Count Non-Zero Entries

- **Task:** Count how many of five array entries are non-zero and leave the count in `a0`.
- **Branch reminder:** Use `beq`/`bne` against `x0` to check zero.
- **C guide:**

```c
int count_nonzero(int *arr) {
    int count = 0;
    for (int i = 0; i < 5; i++) {
        if (arr[i] != 0) count++;
    }
    return count;
}
```

Question 6 – Find Minimum (Five Elements)

- **Task:** Scan five integers and return the minimum in `a0`.
- **Suggested flow:** Assume element `0` is the current min, then compare each subsequent element with `blt`.
- **C guide:**

```c
int min5(int *arr) {
    int min = arr[0];
    for (int i = 1; i < 5; i++) {
        if (arr[i] < min) min = arr[i];
    }
    return min;
}
```

Question 7 – Reverse Copy (Four Elements)

- **Task:** Copy `src[4]` into `dst[4]` in reverse order (`dst[0] = src[3]`, …).
- **Hint:** Maintain two offsets: one that starts at `0` for `dst`, another that starts at `12` for `src`.
- **C guide:**

```c
void reverse_copy(int *dst, int *src) {
    for (int i = 0; i < 4; i++) {
        dst[i] = src[3 - i];
    }
}
```

Question 8 – Fill with Index Values

- **Task:** Write a loop that stores the element index into each slot of a six-element array (`arr[i] = i`).
- **Immediate reminder:** Use `addi` with `1` to increment the loop counter.
- **C guide:**

```c
void fill_indices(int *arr) {
    for (int i = 0; i < 6; i++) {
        arr[i] = i;
    }
}
```

Question 9 – Sliding Window Sum

- **Task:** Given five input values in `.data`, compute `arr[0] + arr[1] + arr[2]` and `arr[1] + arr[2] + arr[3]`, storing both results into an output array of length two.
- **Approach:** Reuse loads—after summing the first window, you already have two of the numbers needed for the second.
- **C guide:**

```c
void window_sum(int *in, int *out) {
    out[0] = in[0] + in[1] + in[2];
    out[1] = in[1] + in[2] + in[3];
}
```

Question 10 – Count Values Greater Than Threshold

- **Task:** With a threshold stored in `.word`, count how many of six array elements are strictly greater than that threshold. Return the count in `a0`.
- **Instruction hint:** `bgt` is a pseudo instruction; the assembler emits `blt` with operands swapped.

- **C guide:**

```c
int count_gt(int *arr, int threshold) {
    int count = 0;
    for (int i = 0; i < 6; i++) {
        if (arr[i] > threshold) count++;
    }
    return count;
}
```