RV32I Array Pre-Lab Warmups

These five mini exercises prepare you for the full **Array Fundamentals** lab. Work through them using the RV32I simulator at https://riscv-simulator-five.vercel.app/. Each program should end with `li a7, 10` followed by `ecall`.

Pre-Lab 1 – Load and Print a Single Element

- **Goal:** Practise loading a word from `.data` into `a0`.
- **Task:** Store `.word 42` in memory, load it into `a0`, and call the integer-print ecall (`a7 = 1`).
- **C guide:**

```c
int main(void) {
    int value = 42;
    print_int(value);   // simulator-specific service
    return 0;
}
```

- **Answer (RV32I):**

```
.data
value:  .word 42

.text
.globl main
main:
    la      a0, value
    lw      a0, 0(a0)
    li      a7, 1           # print integer
    ecall

    li      a7, 10
    ecall
```

Pre-Lab 2 – Swap Two Elements

- **Goal:** Use two loads and two stores to exchange array elements.
- **Task:** Given `.word 5, 9`, swap the values so memory holds `9, 5`.
- **C guide:**

```c
void swap(int *arr) {
    int temp = arr[0];
    arr[0] = arr[1];
    arr[1] = temp;
}
```

- **Answer (RV32I):**

```
.data
arr:    .word 5, 9

.text
.globl main
main:
    la      t0, arr
    lw      t1, 0(t0)       # arr[0]
    lw      t2, 4(t0)       # arr[1]
    sw      t1, 4(t0)
    sw      t2, 0(t0)

    li      a7, 10
    ecall
```

Pre-Lab 3 – Initialize with a Loop Counter

- **Goal:** Build comfort with a simple counted loop.
- **Task:** Set `arr[0..2] = 0, 1, 2` using a loop that increments both the index and address pointer.
- **C guide:**

```c
void init(int *arr) {
    for (int i = 0; i < 3; i++) {
        arr[i] = i;
    }
}
```

- **Answer (RV32I):**

```
.data
arr:    .space 12

.text
.globl main
main:
```

```
    la      t0, arr
    li      t1, 0           # i
loop:
    sw      t1, 0(t0)
    addi    t0, t0, 4
    addi    t1, t1, 1
    blt     t1, 3, loop

    li      a7, 10
    ecall
```

Pre-Lab 4 – Sum Until Zero Sentinel

- **Goal:** Practise looping until a sentinel value appears.
- **Task:** Given `.word 3, 4, 5, 0`, sum the values until you see `0` and leave the sum in `a0`.
- **C guide:**

```c
int sum_until_zero(int *arr) {
    int total = 0;
    for (int i = 0; arr[i] != 0; i++) {
        total += arr[i];
    }
    return total;
}
```

- **Answer (RV32I):**

```
.data
arr:    .word 3, 4, 5, 0

.text
.globl main
main:
    la      t0, arr
    li      a0, 0           # total
loop:
    lw      t1, 0(t0)
    beq     t1, x0, done
    add     a0, a0, t1
    addi    t0, t0, 4
    j       loop
done:
    li      a7, 10
    ecall
```

Pre-Lab 5 – Compare Adjacent Elements

- **Goal:** Combine loads and branches for basic comparisons.
- **Task:** With `.word 8, 12`, compare the two values and set `a0 = 1` if `arr[1] > arr[0]`, else `a0 = 0`.
- **C guide:**

```c
int is_increasing(int *arr) {
    return arr[1] > arr[0];
}
```

- **Answer (RV32I):**

```
.data
arr:    .word 8, 12

.text
.globl main
main:
    la      t0, arr
    lw      t1, 0(t0)       # arr[0]
    lw      t2, 4(t0)       # arr[1]
    li      a0, 0
    blt     t1, t2, greater
    j       done
greater:
    li      a0, 1
done:
    li      a7, 10
    ecall
```

Work through these until the instruction patterns feel routine. They mirror the mechanics you'll rely on in the main lab (offset arithmetic, loops, and simple conditionals).