PseudoCode:

Main.cpp:

```
measure_latency()
{
        Initialize array 'data' of size (array_size / size_of_data_element)

        // Setup linked-list-like access pattern with steps
        For i from 0 to size of data:
        data[i] = (i + STEP) % size_of_data  // Circular jump setup
        Start timer
        Initialize idx to 0  // Starting index
        For iteration from 0 to LARGE_NUMBER:
                // Access repeatedly for time measurement
        idx = data[idx]  // Follow the linked-list-like access path
        Stop timer
        Calculate and print elapsed time and array size

}

measure_bandwidth()
{
        Initialize array 'data' of size s

        // Warm-up: Populate the array to ensure it's in memory
        For i from 0 to size of data:
        data[i] = i % 256
        Start timer

        // Repeat the memory write operations many times for accurate measurement
        For repeat from 0 to 100:
        For i from 0 to size of data:
        data[i] = i % 256  // Write operation
        Stop timer
        Calculate bandwidth as (Total data size written) / (Elapsed time)
        Print the elapsed time and calculated bandwidth
}
```

TLB_test.cpp:

Define constants:
    PAGE_SIZE = 248 KB
    NUM_PAGES = 16,384 (16 * 1024)

main()
{
    Allocate an array of size (NUM_PAGES * PAGE_SIZE) bytes

    // Step 1: Warm-up phase
    For each page in NUM_PAGES:
    Access the first element of each page to ensure pages are allocated

    // Step 2: Measure access time with TLB misses
    Start timer

    For each page in NUM_PAGES:
    Access the first element of the current page
    // This forces a TLB lookup for each page accessed

    Stop timer

    Calculate and print the elapsed time

    Free the allocated memory
}