Question 5. TLB [8 MARKS]

A 32-bit processor uses a two-level page table, with a page size of 4KB. It has two TLBs, an I-TLB (instruction TLB) that stores mappings for code addresses, and a D-TLB (data TLB) that stores mappings for data addresses. Both TLBs are fully associative (all TLB entries are looked up in parallel), and both have 64 entries. Consider the following code snippet:

```c
char array[4096 * 64];

void simple() {
    int i = 0;
    int or = 0;
    int and = 1;

    for (i = 0; i < 4096 * 64; i++) {
        or = or | array[i];
    }

    for (i = 0; i < 4096 * 64; i++) {
        and = and & array[i];
    }
}
```

Part (a) [1 MARK] How many I-TLB misses will take place when the `simple()` function is run? Provide a justification for each TLB miss. State any assumptions that you make.

One I-TLB miss, assuming all the code above fits in 4KB.

Part (b) [3 MARKS] What is the minimum number of D-TLB misses that will take place when the `simple()` function is run? Provide a justification for each TLB miss. State any assumptions that you make.

Minimum number of D-TLB misses = 1 + 63 + 1 + 1 (see reasons below) = 66

Notice that array is stored in 64 pages

- 1 D-TLB miss for a stack page (which stores the local variables of `simple()`).

Loop 1:
- 63 D-TLB misses when the first 63 pages of the array are accessed. Now the D-TLB is full.
- 1 D-TLB miss when the last element of array is accessed. This access will require invalidating a TLB entry. Let's invalidate the last TLB entry (for the 63th page), and replace it with the TLB entry for the 64th page.

Loop 2:
- 0 D-TLB misses for the first 62 page accesses (these entries are already present).
- 1 D-TLB miss for page 63. Let's invalidate TLB entry for page 62, and replace it with TLB entry for page 63.
- 0 D-TLB miss for page 64.
Part (c) [2 MARKS] Let us estimate the performance overhead of using the paging system. Assume that we have disabled the processor cache, so that every load and store instruction accesses memory (RAM). Assume also that there is no cost to accessing the TLB, and the compiler has performed no optimizations. Also, we will only consider accesses to array in the simple() function. How many memory accesses are performed to read array? How many memory accesses are performed to read the page table? Now estimate the overhead of the paging system. State your answer as a single number, possibly in terms of a power of 2.

Total number of memory access for reading array: 4096 * 64 * 2
Number of memory accesses to read the page table = 2 (nr. of levels of page table) * nr. of TLB misses = 2 * 66 (from previous answer)

Overhead = 2 * 66/(2*66 + 2 * 64 * 4096)
        ~ (2 * 64)/(2 * 64 * 4096)
        ~ 1/4096
        = 2^(-12)

If you said 2^(-11), we accepted the answer. This is reasonable if you simply assume that there is one TLB miss for each page. So you require two additional memory accesses for the page table for accessing each page (4096 memory accesses).

Part (d) [2 MARKS] Suggest a way of changing the code in the simple() function to reduce the number of D-TLB misses. What is the number of misses with your code change? Provide a justification.

Use one loop to do both calculations.

This reduces the one additional D-TLB miss that occurred in Part(b) when loop 2 executed.

Total number of D-TLB misses = 65