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.

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.
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.

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.