a.)
We will want to use a smaller block size; this is due to the fact that we are using a nested loop so we will need more items in the blocks of the cache to identify the rest of the elements, this will allow for a less miss rate.
b.)
There are two options that we are facing with this code. It is either having a large array or a small array size. If the array size is large, then we have to use a smaller block size. It will be the same reason as the one stated above. Having more blocks in the cache will allow us to have a better hit rate for the rest of the elements, allowing for a small miss rate. If the array size is small then we will need a larger block size, this reduces the amount of total elements that are going into the blocks of cache, having a larger block size will avoid compulsory misses.
c.)
Translate sub_total += A[ j * N + i ] into mips code
mul $t0, $s1, $a1 // first we have to multiply because of pemdas
add $t0, $s1, $s0 // then we add to finish the brackets, and because of pemdas again
lw $a0, 0($t0) // we now change A into the new value that we created, no offset needed
add $s2, $s2, $a0 // adding the final values with is += in the equation
sw $s2, 0($a0) // saving the results to the new sub_total
d.)
Load instruction and data:

1.) Read Instructions: check TLB if instruction is in the memory by looking up its virtual page number
2.) If virtual page number is not present in TLB, read miss. Check L1 page table to see if virtual page number is present
3.) If it exist in the L1 page table then access the data, read hit.
4.) If does not exist then look into the L1 I-cache, L1-Dcache, L2 Cache.
5.) If it exist in step 4 then read hit and access the data from the cache.
6.) If it does not exist in step 4 then we read as a miss, add it to the cache and save the page number in the TBL, then it will be accessed through the memory.
7.) Else if it is present in the TLB then read hit and don't access the page table

Target instructions or data are not in cache or page tables:

1.) Check to see if the instruction is found
2.) If not in the cache, then its a cache miss and now check in the TBL
3.) Check to see if it is in the TBL, if it is not there the page table checks
4.) If the page table can not find the page then load the instructions from secondary memory
5.) Add the secondary memory to the physical memory

e.)

1.) first we have to multiply j * N because of pemdas and store it into a temporary slot '$t0'
2.) then we add this new temp value '$t0' to i to finish the brackets, and because of pemdas again, then store it back into this temp value '$t0'
3.) we now change A into the new value that we created, loading the new value to A where the new value is the temp value we calculated for '$t0' and storing it to the value of A

4.) adding the final values with is += in the equation so it will be sub_total value plus the new A value storing it in sub_total value.
5.) Last we are saving the result of sub_total plus the value of A into the value of sub_total

f.)

We will skip steps #4-7 on loading the instructions and data, since all the memory will fit in the L1-Cache so we don't have to check the rest of the cache, Also we will skip all of the Target instruction steps since everything will already be present in the L1-Cache.