**Part 1**

*1. Why can the Frobenious norm be nonzero when your code is correct?*

The Frobenious norm can be non-zero even if the code executed correctly because the calculations were all performed with single precision floating point numbers. There must have been digits that were lost to rounding.

*2. Compute a formula for how many memory accesses your MADD1 function makes to multiply matrices size . Does this agree with the memory access count in the data cache simulation tool? Don't forget to disable the call to check your answer as this will make memory accesses.*

Based on my algorithm’s pseudocode…

|  |  |
| --- | --- |
| float tmp  for i=0 to n-1  for j=0 to n-1  tmp=0  for k=0 to n-1  tmp=tmp+A[i][k]\*B[k][j]  c[i][j]=c[i][j]+tmp | N/A  N \* (  N \* (  N/A  N \* (  2 )  + 2 ) ) |

I should expect memory accesses. Using the MIPS data cache simulation tool, I got exactly the expected number for and , which yielded 8704 and 1152 memory accesses respectively.

**Part 2**