PARALLELIZATION AND VECTORIZATION IN GCC

1. What sort of dependence exists in the code given in file 1.c? Analyze the data dependence dump file generated by gcc (*.ckdd) to understand the computed dependence. Also figure out the distance and direction vector.

2. In the code given in file 2.c, categorize each subscript as SIV, ZIV, or MIV. Perform data dependence analysis on the code and analyze the gcc dump generated. Try to change the positions of each subscript and observe the effect in the dependence analysis.

3. What sort of dependence exists in the code given in file 3.c? Can it be vectorized? Co-relate your observation to the dependence distance given in gcc dump file.

4. In the code given in file 4.c, the statement in the loop exhibits what type of dependence? Can this loop be vectorized? Why? Without changing the statement, can you do something to change the vectorization result?

5. What is the result of vectorization on the code in file 5.c before and after the application of loop interchange? In the graphite dump file, analyze the CLAST to identify the loop structure after loop interchange.

6. For the code given in file 6.c, try to increase the number of threads starting with 2. Till what load (number of iterations per thread) does parallelization happen?

7. Try to parallelize the code given in file 7.c with and without Graphite's data dependence analysis. Why is the result different? Replace N by a number. Now observe the result. Why is the result different when N is a parameter?

8. In the code given in file 8.c, consider statements S1 and S2. Can they be parallelized? Try parallelizing the code with and without GRAPHITE. Why is the result different? How is the loop parallelized with GRAPHITE enabled?

COMMANDS TO GENERATE DUMP

- For Data Dependence :
  `gcc -fdump-tree-all -fcheck-data-deps -fdump-tree-ckdd-all -O3 filename.c`
- For Vectorization :
  `gcc -fdump-tree-all -fdump-tree-vect-all -msse4 -O3 filename.c`
- For Parallelization :
  `gcc -fdump-tree-all -ftree-parallelize-loops=x -fdump-tree-parloops-all -O3 filename.c`
- Graphite Parallelization :
  `gcc -fdump-tree-all -ftree-parallelize-loops=x -fdump-tree-parloops-all -floop-parallelize-all -O2 filename.c`
- For Loop Interchange :
  `gcc -fdump-tree-all -floop-interchange -fdump-tree-graphite-all -O3 filename.c`

Note : In parallelization dumps, in place of 'x', pass the number of threads you wish to create.

Note : If you wish to just look at the transformed code with minimum verbosity, remove '-all' from the tree dump option.