PCA Lab3

Adur Saizar
Josep Oriol Vilarrubí

March 22, 2014

Contents

1 Primers

In this application we have removed all the memory accesses to the struct that contained the sizes of the values in the array, how many bytes an integer has, how many bits does a byte has... And we have added some defines to do that function.

But when you run both versions of the applications with the level of optimization 'O3' you do not observe any speedUp at all, that is because the variables are propagated all over the code when you apply the optimization level 'O3', so the compiler now sees the value of the variables so it make the same optimization we have done.

2 Trigon

In this application we have performed both memoization and buffering optimizations.

To perform memoization optimization, we have precomputed the sinus and cosinus for all possible input values as they are a priori known and bounded.

For the buffering optimization, we have replaced the write system calls in the inner loop for a buffer storage, so all the inner loop generated output is printed once.

3 Pi

In thes application we have performed three kind of optimizations: Specialization, Memoization and Buffering.

Basically, we have specialized the most computationaly expensive function, that is DIVIDE(...). To do so we have created different functions derived of DIVIDE for every divisor used in the app: DIVIDE5, DIVIDE25 and DIVIDE239. In each of them we have used memoization in order to avoid long latency instructions like division and multiplication by precomputing the results for all possible input cases.

Furthermore, we have also changed the way the remainder is computed in the non-specialized functions. We have replaced the original implementation for the modulus operand. The reason of doing this is that because of the processor's architecture, the remainder is computed during the division operation, so by using the modulus operand the compiler can profit of this fact and avoid extra computation to calculate it.

Finally, we have also done some buffering in the epilogue(...) function to reduce the number of write system calls.