CHAPTER 1

INTRODUCTION

Evaluation of arithmetic expressions is an important and widely studied problem. With the widespread use of multiprocessors, parallel evaluation of arithmetic expressions has drawn the attention of several researchers. For parallel evaluation, the possible parallel operations in sequential expression(s) have to be detected. Data dependence analysis is a key factor in the detection of parallelism, among a block of expressions.

The existing hardware are sophisticated and support multiprocessing. But software support has always lagged behind the hardware growth. Before the hardware capability of a given computer is exploited in full, an improved hardware design comes into the market. This legacy seems to continue for ever. To effectively utilise the hardware, the programmer will have to identify the chunks that can be performed in parallel and write the program so as to coordinate all the processors and compute the result.

Identifying parallel processable components is a burden on the programmer. It is both tedious and error prone. Therefore to relieve the programmer from this burden, implicit parallelism is the solution.
1.1 MOTIVATION

The contribution by Y.N.Srikant (1990) and David J.Kuck (1972) served as the motivation behind this thesis. Y.N.Srikant introduced a method which parses the arithmetic expression and constructs the task graph. The task graph is an indication of the number of operations that could be done in parallel. The author uses nesting level of parenthesis to identify parallel operations. If nesting levels are not present his approach fails to identify parallel operations. Exploitation of parallelism is targeted by him at a single instruction level.

David J.Kuck has tackled exploiting parallel operations among a block of statements. The author has developed a method called back substitution method, for exploiting parallelism among a block of statements. In his back substitution method, to eliminate data dependence, he substitutes the entire expression representing the dependent variable to eliminate the dependency and all the statements are evaluated independently. Therefore more than one processor will perform the same operation.

The issues discussed above with the possibility of further avenues for parallel processing served as the motivation and basis for this work.

1.2 OBJECTIVE OF THE THESIS

Based on the issues discussed above, the objectives for the work are formulated as follows:

(i) trying to increase the number of parallel operations both at intrainstruction and interinstruction levels by rearranging
operators and operands, when an operation is not possible simultaneously.

(ii) to possibly increase the number of parallel operations limited by dependency at interinstruction level by rearranging operators and operands.

(iii) to devise algorithms that can work both under unlimited and limited number of processors, for the above situations.

(iv) to suggest an alternative to the back substitution method for interinstruction parallelism.

1.3 ORGANIZATION OF THE THESIS

Based on the objectives, stated above, the work has been carried out and presented as given below:

Chapter two covers the various aspects of parallel computing. Various forms of dependence that limit parallel operations are presented. The various steps involved in problem solving, mechanisms to support parallel capabilities and the architectural support are dealt with.

Chapter three covers exploitation of parallelism at intrainstruction and interinstruction levels. To exploit maximum parallelism at intrainstruction level and interinstruction level, rearrangement of operators and operands are carried out. An alternate to the existing solution for parallel operations at interinstruction level, when data dependency is encountered, is also discussed. A heuristic algorithm to guide whether to adopt the back substitution method or the rearrangement method at interinstruction level in case of data dependence, is presented.
Chapter four explains the developed algorithm for detecting and executing the parallel operations. A summary of the shortcomings of the existing work is presented. The data structures and the routines used in the implementation of the parallel algorithm are described. An explanation of the working of the algorithm is also presented. The availability of processors is assumed to be as many as the number of elements in the arithmetic expression.

The algorithm presented in chapter four is modified to work even when the number of processors are limited, and is discussed in chapter five. A heuristic mapping function to map the elements of the arithmetic expression onto a limited number of processors is presented. The modifications to the basic data structures and routines described in chapter four, are presented. The functioning of the algorithm is also explained with a suitable example.

Chapter six covers exploitation of parallelism at interinstruction level. The existing method of exploiting parallelism is discussed. The rearrangement method and an algorithm for achieving the maximum parallelism through this is explained. The data structures and routines used in the implementation of the algorithm are described. The working of the algorithm is also explained, with an example.

Chapter seven presents the simulation results. The algorithm to detect and execute parallel operations at interinstruction level was simulated and the results are discussed.

Chapter eight concludes by summarising the work done and the contributions made. It also suggests few topics for future work that could be carried out in continuation of this research.