Diss. ETH Nr. 10277

A Programming Language for Vector Computers

A dissertation submitted to the
SWISS FEDERAL INSTITUTE OF TECHNOLOGY ZURICH
(ETH Zürich)

for the degree of
Doctor of Technical Sciences

presented by
Robert Griesemer, Dipl. Informatik-Ing. ETH
born June 9, 1964
citizen of Güttingen, Thurgau

accepted on the recommendation of
Prof. Dr. H. Mössenböck, examiner
Prof. Dr. N. Wirth, co-examiner

1993

Abstract

This thesis introduces and elaborates on specific language constructs that allow a simple programming of vector computers and help to gain a better understanding for these programs. Thereby the emphasis lies on the support of explicitly vectorizable statements as well as on a concept for parameter passing adapted to the needs of numerical applications.

Vector computers provide powerful instructions for the processing of whole vectors. The speed of programs is often increasing by orders of magnitude if these programs allow the use of such instructions, i.e. if they are vectorizable. In order to make a program run faster, a compiler usually tries to vectorize its innermost loops. Unfortunately, the dependence analysis required therefore is quite complicated and often cannot be performed completely. The thesis therefore proposes a simple language construct allowing the explicit specification of independence and thus the parallel execution of statements. Hence, this language construct is much easier to vectorize than loops. It improves the readability and security of programs without reducing the quality of the generated code.

The main application area of vector computers are numerical applications of linear algebra. A problem arising with those programs is that parts of matrices such as rows, columns or diagonals must be passed as arguments to a subroutine. Yet, most programming languages do not support such a flexible way of parameter passing. Array constructors offer a simple and safe way to solve this problem.

The second part of the thesis focuses on the description of an experimental programming language called Oberon-V and of an appropriate cross-compiler for the Cray Y-MP. Oberon-V includes a subset of the language Oberon, which has been extended by the language constructs mentioned above. Compared to traditional compilers for vector computers, the Oberon-V compiler excels by its compactness and efficiency. Detail problems of implementation were solved in a new and more simple way: some of the achievements were a new way of generating symbol files to support separate compilation, the optimization of the generated code by eliminating redundant computations (common subexpression elimination) and the reorganization of instructions to increase the execution rate (instruction scheduling). The thesis finally investigates and judges the code quality of Oberon-V programs in comparison with corresponding Fortran programs.

Kurzfassung

In dieser Arbeit werden spezielle Sprachkonstrukte eingeführt, die das einfache Programmieren von Vektorrechnern erlauben und zu einem besseren Verständnis dieser Programme beitragen sollen. Im Vordergrund steht dabei die Unterstützung von explizit vektorisierbaren Anweisungen sowie ein an die Bedürfnisse numerischer Applikationen angepasstes Parameterübergabe-Konzept.

Vektorrechner stellen leistungsfähige Instruktionen für das Bearbeiten ganzer Vektoren zur Verfügung. Programme werden oft um Grössenordnungen schneller, wenn sie von solchen Instruktionen Gebrauch machen können; d.h. wenn sie vektorisierbar sind. Zu diesem Zweck versucht ein Compiler gewöhnlich die innersten Schleifen eines Programms zu vektorisieren. Die dazu notwendigen Abhängigkeits-Analysen sind kompliziert und können häufig auch nur unvollständig ausgeführt werden. Es wird ein einfaches Sprachkonstrukt vorgeschlagen, welches die Unabhängigkeit und somit die parallele Ausführbarkeit von Anweisungen explizit auszudrücken erlaubt, und deshalb wesentlich einfacher vektorisierbar ist als Schleifen. Es wird gezeigt, dass damit die Qualität der Programme bezüglich Lesbarkeit und Sicherheit verbessert werden kann ohne dabei die Qualität des erzeugten Codes zu vermindern.

Hauptanwendungsgebiet von Vektorrechnern sind numerische Applikationen aus dem Bereich der linearen Algebra. In solchen Programmen stellt sich oft das Problem, dass Teile von Matrizen, z.B. Zeilen, Spalten oder Diagonalen als Argumente einem Unterprogramm übergeben werden müssen. Die meisten Programmiersprachen unterstützen eine solch flexible Art der Parameterübergabe nicht. Mit Hilfe von Array-Konstruktoren lässt sich dieses Problem einfach und vor allem sicher lösen.

In einem zweiten Teil der Arbeit wird eine experimentelle Programmiersprache (Oberon-V) sowie ein dazugehörender Cross-Compiler für die Cray Y-MP vorgestellt. Oberon-V umfasst eine Teilmenge der Sprache Oberon, welche um die erwähnten Sprachkonstrukte erweitert worden ist. Der Oberon-V Compiler besticht durch seine Kompaktheit und Effizienz im Vergleich zu herkömmlichen Übersetzern für Vektorrechner. In der Implementierung wurden diverse Detailprobleme auf zum Teil neue und einfachere Art und Weise realisiert. Dazu gehören eine neue Art der Erzeugung von Symbol-Files für die Unterstützung getrennter Übersetzung, die Optimierung des erzeugten Codes durch Entfernen redundanter Berechnungen (Common Subexpression Elimination) sowie das Umordnen von Instruktionen zur Steigerung der Ausführungsgeschwindigkeit (Instruction Scheduling). Schliesslich werden Oberon-V Programme sowie die erreichte Codequalität im Vergleich zu entsprechenden Fortran-Programmen untersucht und kritisch beurteilt.