Memorandum M-529

EVALUATION OF THE CYBER 205 VECTOR PROGRAMMING COURSE

23 - 26 October 1984

by

J.H. van Dijk and J.W. de Heer

Delft - The Netherlands

December 1984
Summary

The Cyber 205 vector programming course was organized by the Amsterdam Universities Computing Centre (SARA) and ACCU, the Academic Computing Centre of the Utrecht University.

The course was held at the Free University of Amsterdam (VU) during 23 - 26 October 1984.

This memorandum gives a short description of the contents of the course and will also present some general information concerning the use of the Cyber 205 supercomputer.

J.H. van Dijk attended the course within the framework of the project: "Investigation of the flow and combustion in a Solid Fuel Combustion Chamber". This project is financially supported by The Netherlands Foundation for Technical Research (STW) under project nr. DRL 14.0120, and the Project Office for Energy Research (PBE) under the project nr. 90753.140 and is carried out by the Department of Aerospace Engineering of the Delft University of Technology together with the Prins Maurits Laboratory TNO.

J.W. de Heer participated the course in connection with the project: "Evaluation of the capabilities of supercomputers for shell collapse load calculations". In the future extensional research will be incorporated in other projects.
<table>
<thead>
<tr>
<th>Contents</th>
<th>page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>1</td>
</tr>
<tr>
<td>Contents</td>
<td>2</td>
</tr>
<tr>
<td>1. Introduction</td>
<td>3</td>
</tr>
<tr>
<td>2. Hardware and Operating Systems</td>
<td>5</td>
</tr>
<tr>
<td>3. Vectorization</td>
<td>7</td>
</tr>
<tr>
<td>3.1. Vector processor</td>
<td>7</td>
</tr>
<tr>
<td>3.2. Automatic vectorization</td>
<td>7</td>
</tr>
<tr>
<td>3.3. Explicit vectorization</td>
<td>10</td>
</tr>
<tr>
<td>3.3.1. General</td>
<td>10</td>
</tr>
<tr>
<td>3.3.2. Explicit vector syntax</td>
<td>10</td>
</tr>
<tr>
<td>3.3.3. Descriptors</td>
<td>11</td>
</tr>
<tr>
<td>3.3.4. Vector functions</td>
<td>11</td>
</tr>
<tr>
<td>3.3.5. Control vectors</td>
<td>11</td>
</tr>
<tr>
<td>3.3.6. Q8-functions</td>
<td>12</td>
</tr>
<tr>
<td>4. Memory use of the Cyber 205</td>
<td>12</td>
</tr>
<tr>
<td>5. Available software</td>
<td>13</td>
</tr>
<tr>
<td>5.1. Basic software</td>
<td>13</td>
</tr>
<tr>
<td>5.2. Numerical software</td>
<td>13</td>
</tr>
<tr>
<td>6. Computing costs of the Cyber 205</td>
<td>14</td>
</tr>
<tr>
<td>7. Conclusions</td>
<td>15</td>
</tr>
<tr>
<td>8. References</td>
<td>16</td>
</tr>
</tbody>
</table>
1. Introduction

The course "Cyber 205 vector programming" was held at the Free University of Amsterdam (VU) during 23-26 October 1984 and was organized by SARA and ACCU.

The course was meant as an introduction to vector programming techniques for the Cyber 205 supercomputer.

The course was split up in three parts:
- hardware and operating systems;
- explicit- and automatic vectorization;
- experiments with the FTN200 compiler.

In the near future several universities in Holland will be connected to the Cyber 205 under which is the Delft University of Technology. This course is the second one (the first was held in spring 1984) and was attended by 37 participants from several universities and research-institutes.

The lectures were given by:
- J. Koot (SARA) : hardware and operating systems;
- J.v. Kats (ACCU) : vectorization technics;
- P. Vons (SARA) : special options of the FTN200 compiler;
- A. Rozendaal (SARA) : practical computer assistance.
Fig. 2.1. Cyber 205 configuration.

Fig. 2.2. Characteristics of the Cyber 205.
2. Hardware and operating systems

In fig.2.1 the Cyber 205 configuration has been sketched. You cannot access the Cyber 205 supercomputer directly. One has to connect to a front-end computer to prepare batch jobs for the supercomputer.

Two types of front end computers are currently in use at SARA. You can choose between the Cyber 750 and the Cyber 825. During the course we used the Cyber 750 to communicate with the Cyber 205. The Cyber 750 runs under the operating system NOS/BE (Network Operating System for Batch Environment). The Network Operating System (NOS) runs on the Cyber 825. NOS and NOS/BE are operating systems designed for mainframe computers. For details we refer to ref.(6) and ref.(7).

The Cyber 205 supercomputer runs on the operating system VSOS (Virtual Storage Operating System, see ref.(5)). The Cyber 205 is a machine with virtual memory and a large physical memory on disk. If a program is too large to fit in central memory it is put on a virtual file by VSOS. The Cyber 205 has two different central processors:
- scalar processor;
- vector processor.

See fig.2.2.

The Cyber central memory holds one million 64 bit words. Because of its virtual memory concept, space can be much larger. The clock cycle (internal unit of time) is 20 nanoseconds.
The scalar processor uses mainly the register file, while the vector processor mostly deals with main memory. The Cyber 205 at SARA has one vector-pipeline, the theoretical maximum speed (in 32 bit mode) is 200 Mflops. 1 Mflop = 1 million floating point operations per second. Cyber 205 versions with 2 or even 4 pipelines are also available from Control Data.
Fig. 3.1.1. The Cyber 205 vector processor consists of 1, 2 or 4 vector-pipelines. At SARA there is only 1 vector-pipeline.

Fig. 3.1.2. The addition unit in a vector-pipeline can be thought of having 7 segments.
3. Vectorization

3.1. The vector processor

The vector processor of the Cyber 205 at SARA consists of 1 vector-pipeline (see fig.3.1.1.). The pipeline consists of segmented units (see fig.3.1.2.). Each unit performs a small part of an arithmetic operation, so that each pair of operands has to be processed in several steps. The vector-pipeline works with 64 or 32 bit vectors. A vector is defined as a set of contiguous storage locations in memory. Contiguous storage is essential to achieve a continuous data-flow through the pipeline. As an example we show the addition of 2 vectors:

DO 10 J=1,N
    Z(J)=X(J)+Y(J)
10 CONTINUE

The addition has been sketched in fig.3.1.3. for cycles 0 up to 9 (results will follow at a rate of 1 per cycle).

In general the time needed for a vector instruction is:

\[ \text{time} = \text{startup} + N \times \text{stream rate}, \]

where:
- \( \text{startup} \) : time needed to initiate the vector pipe;
- \( N \) : vector length;
- \( \text{stream rate} \) : time which depends on the kind of floating point operation (+, -, *, or /).

The timing for the four basic arithmetic floating point operations on the 1 pipe Cyber 205 (in 64 bit mode):

- add/subtract: 51 + N cycles
- multiply: 52 + N cycles
- divide: 80 + 25N/4 cycles

3.2. Automatic vectorization

The Cyber 205 has two ways of vector processing:
- automatic vectorization;
- explicit vectorization.

The automatic vectorizer is invoked by including the letter V in the string of OPTIMIZE options in the FTP200 statement and can be regarded as a form of compiler optimization. Explicit vectorization is treated in the next chapter.

The only type of code structure that can qualify for automatic vectorization are DO-loops. The complete set of criteria for vectorizable DO-loops appears in the Fortran 200 manual (see ref.(3)). The basic problem is that many scalar Fortran codes are incompatible with vector processing.

We recall that a vector is a set of contiguous storage locations in memory. This implies that a DO-loop that accesses arrays in a nonsequential manner lacks vector structure, and therefore is not directly vectorizable. Another point of consideration is the maximum iteration count. The maximum allowed vector length is 65535.
<table>
<thead>
<tr>
<th>( T = 0 )</th>
<th>( T = 1 )</th>
<th>( T = 2 )</th>
<th>( T = 3 )</th>
<th>( T = 4 )</th>
<th>( T = 5 )</th>
<th>( T = 6 )</th>
<th>( T = 7 )</th>
<th>( T = 8 )</th>
<th>( T = 9 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( (x_7, y_7) )</td>
<td>( (x_6, y_6) )</td>
<td>( (x_7, y_7) )</td>
<td>( (x_5, y_5) )</td>
<td>( (x_6, y_6) )</td>
<td>( (x_7, y_7) )</td>
<td>( (x_4, y_4) )</td>
<td>( (x_5, y_5) )</td>
<td>( (x_6, y_6) )</td>
<td>( (x_7, y_7) )</td>
</tr>
</tbody>
</table>

**Fig. 3.1.3.** The addition of 2 vectors at cycles \( T = 0 \) up to \( T = 9 \). In this example the vector-pipe is assumed to be 5 cycle segmented.
The presence of a startup time in the general formula for a vector instruction (time = startup + N*stream rate) indicates that loops with small N-values may be more efficiently executed in scalar mode.

Maximum performance of the Cyber 205 supercomputer can be obtained by using the vector extensions of the FTN200 compiler, this subject will be treated in the next chapter.
3.3. Explicit vectorization

3.3.1. General

To make an optimal use of the Cyber 205 FTN200 compiler one can use the special vector syntax and vector functions. For instance in case of many DO-loops in a program have very small iteration counts, automatic vectorization is not efficient due to the long startup times involved. In such cases the FTN200 compiler offers the following possibilities:
- a vector syntax is provided as an extension of the scalar syntax, allowing the programmer to vectorize when it makes sense. The vector syntax itself is in turn extended by a large number of highly optimized system library functions (trigonometric and others) that accept vector arguments and deliver vector results;
- the complete set of machine instructions are available in the form of a special call-syntax.

3.3.2. The explicit vector syntax

Defining a vector in FTN200 three things must be specified:
- the data type;
- starting address;
- length.

Since a vector is defined as contiguous storage locations in memory the starting address can be represented by an element of the array in question. The data type of each array is already defined in the code declared either implicitly or by use of type declaration statements. So the array element used as a pointer will therefore also define the data type of the vector element.
The length is specified as an additional subscript preceded by a semi-colon (;). For example consider array A:

```
DIMENSION A(100)
```

The explicit vector syntax code:

```
A(1;60)
```

is a real vector consisting of:

```
( A(J),J=1,60 )
```

Or:

```
A(5;90)
```

is a real vector consisting of:

```
( A(J),J=5,94 )
```

Legal data types are real, integer, complex, half precision, double precision and bit.

Contrary to automatic vectorization loop dependent subscripts are allowed to be more complicated.
The notation A(1;60) is an example of the explicit vector syntax. The most important other type is the implicit or descriptor notation.
3.3.3. Descriptors

On the machine level a pointer to a vector is represented as a 64-bit word containing the integer length in the leftmost 16 bits, and the address of the starting location in the rightmost 48 bits. Such a word is called a descriptor and may be considered as a scalar variable. Descriptors exist in FTN200 as a special data type. The descriptor must appear in a non-executable DESCRIPTOR statement:

```
DESCRIPTOR AD
ASSIGN AD,A(1:N)
```

Such a descriptor represents a vector: it is linked to that vector by the ASSIGN statement.
The following example shows how explicit vector syntax and descriptors can be used in case of a DO-loop:

```
DIMENSION A(100),B(100),C(100)
DESCRIPTOR AD,BD,CD
N=100

Standard Fortran 77 code:

```
DO 10 I=1,N
   A(I)=B(I)+C(I)
10 CONTINUE
```

Explicit vector syntax:

```
A(1:N)=B(1:N)+C(1:N)
```

Implicit vector syntax:

```
ASSIGN AD,A(1:N)
ASSIGN BD,B(1:N)
ASSIGN CD,C(1:N)
AD=BD+CD
```

The use of descriptors will result in efficient machine code and may save overhead.

3.3.4. Vector functions

Almost all of the standard scalar system library functions are available as vector functions in the FTN200 compiler. The name of such a vector function is simply the scalar name prefixed by a V.

In addition the programmer may also write his own vector functions.

3.3.5. Control vectors

A logical constant or variable occupies one word of storage. In FTN200 it is possible to pack 64 logical values (1 or 0) in one word by using the data type BIT. Arrays of the type BIT are typically used to control vector operations, reason why they are often called control vectors.

Among other cases they may be used to replace IF structures which otherwise could not have been vectorized.
3.3.6. Q8-functions

The prefixing of a name by Q8 indicates that the named function corresponds to a direct machine instruction.
For instance: non contiguous stored data can be made contiguous by using a Q8-function. For other applications we refer to ref.(3).

4. Memory use of the Cyber 205

The Cyber 205 has 2 types of storage devices:
- central memory of 1 million words;
- several disk units, each having a capacity of 600 Mbytes.
Some restrictions on the use of the disks by the current version of VSOS are:
- the maximum file size is 256 Mbytes, but in practice it is not possible to allocate a file longer than 80 Mbytes;
- each file must reside on one disk.

VSOS, the Cyber 205 Virtual Storage Operating System knows the following types of files:
- local files : exist for the duration of the job;
- permanent files : (1) : private files (belong to 1 user);
- (2) : pool files (files that can be shared among users).

Permanent files remain on disk until the owner destroys them or until they have not been used for eight days. In that case SARA destroys the files; SARA doesn't account permanent file usage at the Cyber 205.
SARA recommends to keep copies of important files on the front-end computer, preferably on magnetic tape. There are no tape units connected to the Cyber 205.

Some final remarks on memory use will be made here concerning the subject "paging". Data transfer between disk and central memory takes place in units of either small or large pages:
- 1 small page = 2048 words;
- 1 large page = 65536 words.

This implies that as soon it is established that the page needed is not in memory, a "page fault" will occur. Processing stops and a system request is issued for that page. This may be a very time- and SBU consuming fact (see table 6.1.). Much attention should be paid to minimize the number of "page faults" (see ref.(5)).
5. Available software

5.1. Basic software

The most important basic software on the Cyber 205 is the FTN200 compiler. FTN200 is a vectorizing Fortran 77 compiler. The CDC FTN200 compiler translates Fortran 77 programs and has an option for automatic vectorization of a number of DO-loop constructs. With special vector syntax extensions, you can vectorize by hand in a compact manner without writing assembler code.

Important tools that facilitates optimization are the utilities VAST and SPY:
- VAST is a precompiler that analyses DO-loops and converts these loops, if possible into explicit Fortran syntax;
- SPY determines which part of the program is using most of the CPU time. For more detailed information see ref.(1) and ref.(8).

5.2. Numerical software

NAG : general numerical and statistical libraries. The Fortran versions installed at the Cyber 205 contains a number of routines that are vectorized. Subroutine calls and documentation are in standard NAG form.
LINPACK : a collection of linear algebra routines.
EISPACK : a collection of routines designed to compute the eigenvalues and eigenvectors of matrices.
MINPACK : a collection of routines designed for the numerical solution of non-linear equations and non-linear least square problems.
QOLIB : the CDC library QOLIB is a collection of routines that are highly optimized for use on the Cyber 205. The library contains a variety of mathematical algorithms including the solving of matrices, finding eigenvalues, inverting matrices.
ABAQUUS : package based on the finite element method, performs advanced structural and heat transfer calculus.
6. Computer costs of the Cyber 205

There are three user categories:

a
Publicly maintained educational institutes. Research institutes financed by the "Ministerie van Onderwijs en Wetenschappen" for at least 50 %.

b
Governmental agencies that do not belong to category a. Non-profit semi-governmental institutes which are working in the same area and the manner as Governmental Agencies do.

c
Third parties not mentioned under a or b.

The tariffs differ for the different types of accounting units.
On the Cyber 205, the use of each system component is expressed in System Billing Units (SBU's). For the different user categories the tariff accounted is:
fl 0.24 for category a;
fl 0.96 for category b;
fl 2.50 for category c.

See table 6.1. for the System Billing Units charged for each System Component.

<table>
<thead>
<tr>
<th>System Component</th>
<th>SBU</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 CPU second</td>
<td>1</td>
</tr>
<tr>
<td>1 large page fault</td>
<td>0.1</td>
</tr>
<tr>
<td>1 small page fault</td>
<td>0.01</td>
</tr>
<tr>
<td>1 disk access</td>
<td>0.005</td>
</tr>
<tr>
<td>1 sector disk I/O</td>
<td>0.0003</td>
</tr>
<tr>
<td>1 virtual system call</td>
<td>0.0003</td>
</tr>
</tbody>
</table>

Table 6.1. The use of the Cyber 205 System Components expressed in System Billing Units.
7. Conclusions

The Cyber 205 vector programming course gave a good overview of the possibilities of vector programming techniques in connection with the FTN200 compiler.

The theoretical maximum speed of the Cyber 205 with 1 vector pipeline is 200 Mflops in 32 bit mode. Most programs run slower, a speed of 45 - 49 Mflops is often a good result (in 64 bit mode). To achieve this result explicit vectorization is necessary.

Careful consideration should be taken by using the VAST option as a vectorizing preprocessor since it does not always imply the most effective vectorization. Explicit vectorization of existing Fortran 77 programs is time consuming and leads to programs which are not compatible with other (scalar) computers.

Finally the authors have some critical remarks:
- participants with only IBM experience should have some more training with the Cyber editors;
- an earlier delivery of the course material would enable the participants to prepare themselves better;
- a copy of the set of overhead sheets was missing.
8. References


(3) Fortran 200, version 1, Reference Manual; CDC 60480200.


(5) VSOS Versions 2 Reference Manual Volume 1; CDC 60459410.

(6) - NOS 2 Reference Set, Volume 1, Introduction to Interactive Usage; CDC 60459660.
   - NOS 2 Reference Set, Volume 2, Guide to System Usage; CDC 60459670.
   - NOS 2 Reference Set, Volume 3, System Commands; CDC 60459680.
   - NOS Version 2, Applications Programmers Instant; CDC 60459360.

(7) - NOS/BE Version 1 Reference Manual; CDC 60493800.
   - Intercom Version 4 Reference Manuel; CDC 60494600.
