APL 82, W. H. Janko, W. Stucky (ed.),
APL Quote Quad, Vol. 13 No. 1,
© 1982 by ACM, Inc.

DESIGN CONSIDERATIONS OF A MICRO-BASED APL

J.C. Wilson
T.A. Wilkinson

Computer Systems Group
University of Waterloo
Waterloo, Ontario, Canada, N2L3G1
(519) 885-1211 ext. 2253

Introduction

This paper discusses some of the decisions involved in the creation of Waterloo microAPL. It describes the goals of the project and some of the concerns with respect to a sucessful implementation. An explanation of the portable software philosophy is included along with some details of the versions of Waterloo microAPL presently in existence.

Background

Waterloo microAPL was created by members of the Computer Systems Group at the University of Waterloo, Ontario, Canada in a period of 15 months in 1980-81. This group is a "practical-research" oriented department of the University and has for years been involved in the production of computer software for use in the field of education. Examples of the group's work include WATFOR, WATFIV, WATBOL and other popular language processors. More recently, much energy has been directed towards educational applications in the field of microcomputers.

It was evident from the beginning of this activity that the microcomputer field was changing rapidly and that new products were evolving every day. To keep up with this rapid change, software would have to be developed very quickly and accurately. As a result we became involved in the science of software portability, a technique whereby software could be moved from one machine to another with less cost and greater speed than that required to completely rewrite the component programs. This technology also results in a high degree of compatibility of software across various machines and can be applied to computers ranging from micro's to maxi's.

Using a language called WSL (the Waterloo Systems Language), the group created a family of processors. This family includes interactive interpreters for the languages APL, BASIC, COBOL, FORTRAN and Pascal and is known as Waterloo microSystems. The remainder of this paper is about the APL project.

Goals

The Waterloo microAPL project was designed to create a standard and complete implementation of APL that would run on a number of different computers ranging from microcomputers to large mainframe machines. All the standard primitive functions were to be included as well as the common set of systems variables and functions. A useful complement of file-handling facilities were designed in order to store data external to the workspace and to make it available to the other language processors in the Waterloo microSystems family. It was also considered important to support the hardware features of the various machines in as "complete" a way as possible (e.g., colour graphics in some computers). A common goal of APL and the other languages in the system was to provide a useful interface to machine language routines, including parameter passing. This would make it easy to call machine language routines from the high level languages. Performance goals were also established. It was expected that APL would perform "similar" to BASIC on the same hardware.

© 1982 ACM 0-89791-078-8/82/007/0380 $00.75

The plan was to use the WSL portable implementation language to create an APL for a 6809-based microcomputer, prototypes of which were being developed in parallel with the software project. (The prototypes were based on the Commodore 8032 and eventually led to the SuperPET.) After the team gained some experience with the microcomputer and restricted memory space found there, the processor would be "ported" to the IBM VM/CMS mainframe system. We would first produce a complete APL processor and optimize the algorithms for performance later.

Concerns

There were a number of concerns related to the APL project. Even though the Computer Systems Group had a lot of experience in the area of language processors and portability, there had been no previous experience with the internals of APL and very little experience with microcomputers. Problems were anticipated dealing with the memory constraints of the small machines. Some concern existed with respect to the WSL high-level language. By its nature, the machine code it generates is not as efficient as hand-written code. This, coupled with the fact that APL's powerful primitives sometimes require a lot of processor time, caused some apprehension. Most of these concerns, however, were eventually dispelled.

Staffing

The project was to be attempted by a team of two senior designer/supervisors and five undergraduate student employees. At most, two students were active on the project at any given time since they alternated this effort with their academic programmes. One of the students was assigned as the "language feature" specialist while the other was given the responsibility of handling systems functions and operating system interfaces. None of the students were familiar with the APL language before the beginning of the project.

Method

Waterloo microAPL was written in WSL in small individually compiled modules which were combined using a linker facility. This technique allows re-packaging to take advantage of specific segmentation and overlay facilities provided by the various computers, particularly those with small address spaces. Compilations and linking were done using a WSL cross-compiler running on a PDP-11 under the RSTS/E operating system. The resulting 6809 processor module was down-line loaded to the Commodore SuperPET via a 9600 baud RS232C asynchronous line where it was tested and debugged.

Memory was viewed as being divided into four basic areas:

1 The WSL function library and associated system routines including the file interface and a floating point emulator. This was used by all the language processors.

2 The APL language processor.

3 The APL user workspace.

4 "Free Space" to contain user's machine language routines callable from APL. When required, this is taken from the workspace.

Cleverness was not a goal in the internal design of the processor. The intention was to keep it simple, get it done quickly with few errors, and optimize it later. This optimization was viewed as an ongoing process involving improvements to the WSL code-generator routines as well as changes to the actual algorithms used in the APL processor.

The 100 modules in the APL processor comprise about 600 routines, averaging about 50 lines of WSL source code each. The resulting 30,000 lines were debugged and tested in approximately 3 man-years.

The main reference document used in the project was the Falkoff/Orth ACM APL Standard [1]. It is certain that the project could not have been completed under the circumstances described here without this reference. Points which were considered ambiguous or inconsistent were usually settled by referring to Sharp APL or to IBM VS APL.

Two other references were followed closely: Jenkins' paper [2] describing the IBM implementation of quad-divide, and L. J. Woodrum's paper [3] on the grade (i.e., sorting) functions.

Final testing was accomplished in parallel with the latter stages of the development process. A number of very bright high-school students were employed and given specific projects to be completed using APL. They proved very adept at finding (and sometimes even fixing) bugs.

Results:

Two versions of Waterloo microAPL have been created and a third is almost complete at the time of writing (Jan 1982). They are described below:

Commodore SuperPET:

Waterloo microAPL was implemented first on this computer. It is a 6809-based machine which has read-only memory (ROM) and two kinds of random-access memory (RAM). The WSL function library and system routines reside permanently in the 22KB (kilobytes) of ROM. When the machine is turned on, a menu is presented allowing the user to select from a number of language processors and applications. If APL is chosen, the processor is loaded into the machine's 64KB of "bank-switched" RAM. This is a specially configured set of sixteen 4KB "pages" which implement a hardware segmentation and overlay facility. This allows the entire APL processor to exist using only 4KB of the 6809's address space. In addition, about 4KB of the machine's "normal" RAM is used for the data areas of the APL processor, leaving room for a 28KB APL workspace.

The Commodore SuperPET supports the IEEE-488 bus which accesses disk drives, printers and other devices. As well, there is an RS232C asynchronous line which can be used to control serial devices or to connect the SuperPET to other computers. Waterloo microAPL supports these devices through its file system which includes a powerful host-communication facility used in conjunction with a special HOSTCM program running on a host computer. The SuperPET screen and keyboard support the entire APL character set including overstrikes (but not the underscored alphabet), and all the keys on the keyboard repeat if held down.

IBM VM/CMS

The second implementation of Waterloo microAPL was on an IBM 4341 running VM/CMS. MicroAPL is simply executed as a program under CMS and the entire processor, workspace and WSL library reside in the memory of the virtual machine. The standard system file interface is used, providing access to all CMS files.

The VM/CMS microAPL processor is 164KB of 4300 native machine code. This includes the data area required by the processor. An additional 138KB is required by CMS. The workspace size is governed by the size of the virtual machine. For example, a workspace of 210KB is obtained by setting the virtual machine size to 164+138+210=512KB.

Access to VM/CMS microAPL is through standard VM/CMS terminals such as 3270's, 3278's, etc. and "dumb" APL/ASCII terminals. No comprehensive performance tests or comparisons have been done.

IBM Personal Computer

The most recent effort is to "port" the Waterloo microAPL processor to the new IBM Personal Computer. This machine is based on the Intel 8088 microprocessor. It was decided to leave the standard IBM ROM set intact and put the entire APL processor, workspace and WSL library in RAM. A workspace as large as 59KB can be obtained. The systems function routines in the existing ROM set will be used as much as possible.

The APL character set is obtained by using the software-controllable character set associated with the colour monitor. Obtaining APL characters with the monochrome monitor requires replacing the character generator ROM. This has been done successfully as an experiment but there is as yet no commercially available APL ROM for the monochrome monitor.

Summary

The group is pleased with the results obtained from this project. Most of the concerns were successfully addressed and solved and an APL processor was produced which could be "ported" with relative ease from one machine to another using the WSL technology. A manual was written which includes a short tutorial on the system as well as a comprehensive reference for the APL language.

Future plans include refining and extending the set of system functions, optimizing the performance and "porting" the processor to other computers.

References:

[1] A.D. Falkoff and D.L. Orth, "Development of an APL Standard", _APL79 Conference Proceedings_, ACM, New York, 1979.

[2] M.A. Jenkins, "The Solution of Linear Systems of Equations and Linear Least Squares Problems in APL", _IBM New York Scientific Center Technical Report No., 320-2989_, June, 1970.

[3] L.J. Woodrum, "Internal Sorting with Minimal Comparing", _IBM Systems Journal_, No 3, 1969.