APL2— a RISC Business

Dr Manuel Alfonseca,
IBM Spain SA,
Madrid Scientific Centre,
Paseo de la Castellana 4,
Apartado 179,
Madrid 1,
Spain.

David Selby,
IBM United Kingdom Ltd.,
Winchester Scientific Centre,
St Clement Street,
Winchester,
Hampshire.
SO23 9DR,
England.

UKSC Technical Report 160

Abstract

APL2 is one of the most exciting programming languages to evolve from IBM's original APL discovery. APL2 has been restricted to IBM 370 mainframe computers, due to the complexity of the interpreter required to implement all facets of the language.

This paper describes a joint research / feasibility study carried out by IBM's Madrid and Winchester Scientific Centres on generating a "Portable" Workstation APL2 interpreter.

Introduction

APL2 is one of the most exciting developments witnessed in the engineering scientific computing community in the last few years. It is marketed by IBM as program product 5668-899 for the 370 family of mainframe computers, operating under either VM or MVS.

This paper discusses a joint research / feasibility study carried out by Madrid and Winchester Scientific Centres to build a portable interpreter for a very large subset of APL2, for use on the IBM Personal Computer range, including the RT/PC and the IBM Personal System/2™[1]. family of computers.

The study, which is now complete, has resulted in the development of a machine independent APL2 interpreter, which can be translated into different target machines. In fact, the following target systems have been chosen:

- The 370 family, as a test case.
- The INTEL family of micro-processors, including the whole IBM Personal Computer family, as well as the IBM Personal System family.
- The ROMP microprocessor, which is being used in the IBM RT/PC.

The machine independent APL2 interpreter was designed with the following objectives:

- Machine independence
- Cross system consistency with 370
- Compactness
- Performance

The first objective ensures that the same language will work in the same manner in all our target machines. The second affects the design of the language, in the sense that the selected subset of APL2 must be compatible with the current IBM product in mainframes. The third condition makes it possible to use APL2 in small machines (up to 640 Kbytes). Finally, the fourth objective tends to assure that the final language will be effective and useful.

Selection of the systems programming language

A system will be portable if it has been programmed in an appropriate systems programming language.

We define portability as the ability to take the definition of our APL2 interpreter and fit it on to differing machine architectures and operating systems.

The typical language to perform tasks of this nature is "C". We decided against using this language for a variety of reasons:

[1] IBM Personal System/2 is a trademark of International Business Machines Corporation.

- Difficulties in the "C" language to manage the very complex data structures needed by an APL2 interpreter.
- Possible inefficiency of "C" compilers for some machines.
- Existence of a previous machine independent APL interpreter written in IL, which could be used as a basis for the APL2 interpreter (in the case of several algorithms, with no change at all).

Therefore, we decided to use IL (Madrid Scientific Systems Programming Language), as in previous developments of machine independent APL interpreters. (See Alfonseca and Tavera[1]).

The IL Compiler

Once a target machine is selected, the machine independent APL2 interpreter should be compiled into the corresponding machine language. This is done by appropriate IL compilers.

IL is a high level assembly language, and its translation to any target machine language is simple and straightforward. IL compilers, therefore, need only a single pass and are automatically optimising, without additional efforts, as they usually have to perform a symbol to instruction translation. However, further optimisations may be added to the compilers, to obtain a better performance of the resulting program, which is extremely compact.

A compiler translating IL into the PC machine language was already in existence, as a sub-product of the APL/PC project performed by Madrid Scientific Centre.

For the RT/PC, a new compiler had to be built. It was written in APL2, running under VM in the 370 family of machines, and compiles to ROMP assembly code, (ROMP is the processor used in the RT/PC range of machines), so that it would be very easy to perform hand optimisations, although, in general, this was not found to be necessary. This compiler represents about 200k of APL2 source code.

APL2 is an unusual choice for the development of a compiler, since it is interpreted, which will make the compiler operation slower. However, in this particular case, speed of compilation is not a consideration, for the interpreter should be compiled only once (barring debugging changes). Compilation of the whole interpreter (about 20000 IL executable statements) takes about 3 hours on a 4381 machine.

The compiler performs typical operations like:

Common sub-expression elimination
Register renumbering
Invariant code motion
Constant arithmetic
Constant folding

In addition, the compiler performs optimisation on multiplication and takes full advantage of the ROMP's overlapped branch instructions. This can yield about 33% improvement in performance.

The complexity of IL may be judged by its expansion factor which is about 7.5:1, that is each line of IL will on average generate 7.5 lines of ROMP assembler code.

Besides the two IL compilers mentioned in the above other IL compilers exist for the 370 and the Series/1 machine languages. The 370 version was used to generate a test case for our machine independent APL2 interpreter, and is used for debugging purposes and for comparison with the host product.

The APL2 Interpreter

As it has been mentioned in the introduction, the objective of this project was to generate a compact machine independent APL2 interpreter that could fit in a small machine with an appropriate performance and a sufficient workspace size.

The machine independent interpreter was built and translated to both the PC environment and the RT/PC environment with the following results:

1. The APL2/PC interpreter (running under PC DOS) has a size of less than 140k bytes. This leaves a maximum workspace size, in a 640k address space, of a little over 440k. This is further reduced by the size occupied by operating systems extensions and/or resident programs loaded by the user of the machine. This seems an appropriate size in this environment.
2. The APL2/RT interpreter has access to a much larger addressing space, and in principle can reach up to a gigabyte workspace size, this being the addressing limit of the ROMP processor. The total size of the interpreter is just under 200k bytes.

Progress in Personal System/2™ operating systems will permit us to "port" our system with little effort to IBM Operating System/2™[2]. which will allow megabyte sized workspaces in the PS/2™ environment.

The language

APL2/PC and APL2/RT implement a very large subset of the APL2 language. In fact, the following features of APL2 have been included in our system, with complete compatibility with the host system:

- Support of mixed arrays.
- Support of general arrays.
- Vector notation.
- Monadic grades extended to matrices.
- New dyadic primitives: Find, Match, Pick, Without.
- New monadic primitives: Depth, Disclose, Enclose, Enlist, First.
- Operators can be defined and executed.
- Mixed, defined and derived functions can be operands of operators.
- A new APL2 style error management.
- $S\Delta$ and $T\Delta$ reference.

[2] IBM Operating System/2 is a trademark of International Business Machines Corporation.

- New or extended system functions: *□AF*, *□AT*, *□FX*, *□TF*.
- New system variable: *□ET*.
- "Each" operator, applicable also to defined functions.
- Axis operator extended to take, drop, ravel, enclose, disclose.
- *)COPY*, *)PCOPY*, *)PIN*, *)NMS* and *)OPS* systems commands.

The following items have not yet been included with respect to host APL2:

- Complex arithmetic and associated data type.
- N-wise reduction.
- Axis operator on pervasive functions.
- Selective specification.
- System variables: *□L*, *□R*, *□TZ*.

  □NLT is not included directly but an APL2 function has been included to emulate its function.
- System function: *□NA*.
- System commands: *)EDITOR*, *)MCOPY*, *)MORE*, *)MSG*, *)OPR*, *)PBS*, *)QUOTA*, *)TIME*.

APL2 workstation embodies a syntax analyser built on the same lines as the host APL2 syntax analysis. This is a fundamentally important part of the system, and for the purposes of cross system consistency it is critical that this component works with perfect compatibility.

Also for compatibility reasons, and to allow the end user to be able to run his APL2 functions on any IBM machine, our system is able to import and export host APL2 workspaces via the ")IN and)OUT" system commands. For migration purposes we also transparently import workspaces in "AIO" format from APL/PC Version 1 and 2. This facility is a major contribution to ease of use.

Workspace management

The APL/PC systems (IBM[1-2]) were constructed around a workspace organisation where the space available was divided into two large pieces. One of them (the first 64 Kbytes) were easily accessible, while the second section (the elastic workspace) resided also in main memory, but was less easily accessible (see Tavera, Alfonseca and Rojas[1] for more details).

In the case of APL2, we made use of a different structure. The workspace is divided into two very different parts. One is a small "cache" workspace, where most of the actual operations are performed, but where no APL objects are stored permanently. The other is the workspace itself, the place where all APL objects are stored and moved around, including the symbol table, the reference table, and the execution stack.

System Services

System services is the all encompassing term for the part of the APL2 system which connects it to its environment through different input/output interfaces. Since this component of the system is dependent of the operating system to a great extent, it has not been programmed in the IL language (with the exception of AP998), but directly in the machine language of the target machine.

The machine independent APL2 interpreter includes a shared variable processor, which makes it possible for APL programs to communicate with auxiliary processors via shared variables (IBM[1]). This processor is compatible with the one defined in the APL/PC Version 2 product (IBM[2]). In this way, some of the processors incorporated with the system are compatible with APL2/PC.

The following auxiliary processors are currently programmed and working successfully:

- AP2: Dynamic program loader and executor (an operating system subsystem working under APL). This Auxiliary Processor will also allow the establishment of an interface with FORTRAN subroutines.
- AP80: Graphics Printer support.
- AP100: DOS Commands.
- AP101: Stack processor and profile manager.
- AP103: DOS/BIOS interrupt interface.
- AP120: Session manager APL2 compatible.
- AP124: Full screen processor, VSAPL compatible.
- AP172: PC Network transaction processor
- AP190: 3278/9 communications processor.
- AP206: Graphics processor.
- AP210: File processor.
- AP232: Asynchronous Communications processor.
- AP440: Music processor.
- AP488: GPIB interface processor.
- AP998: Logic programming processor.

For the RT/PC, we had the challenge to produce the first IBM APL system running under the UNIX™[4]. style of operating system (in this case, the AIX™[3]. operating system). We wished to embody a high degree of cross system consistency with the PC system and the mainframe program product.

For APL2/RT we built a global shared variable processor that allows us to perform task to task or user to user transactions, based on the formalisation of protocol defined by John Gerth (Gerth[1]). This permits APL2/RT to provide a high degree of communication between the sessions under its control. It also makes it possible to run some auxiliary processors as global servers on the system.

The following auxiliary processors are currently programmed and working successfully:

[3] AIX is a trademark of International Business Machines Corporation.

[4] UNIX is a trademark of AT&T Bell Laboratories.

- AP80 — Printer support
- AP100 — AIX Commands and subtasks
- AP101 — Stack processor
- AP120 — Session manager
- AP124 — Alpha numeric fullscreen
- AP127 — SQL Database processor
- AP172 — PC Network support
- AP190 — 3278/9 Interface
- AP207 — X-Windows graphics processor.
- AP210 — AIX file system.
- AP998 — Inference engine support

Conclusion

Our goal was to implement a desktop version of mainframe APL2. Most people, inside and outside IBM, felt this would be impossible, because the system would be too large to be useful. However, we have achieved the goal, and the amount of available workspace (up to 440 Kbytes) is sufficient to make the system usable, while the system has an acceptable performance.

However, what we have implemented is useful, not only for a machine as small as the IBM Personal Computer Convertible, but also for much larger machines, such as the RT/PC. The system is operating system independent and highly portable to different machine architectures and operating systems.

Acknowledgements

In completing this project, major thanks must go to Dr James Brown, for providing unstinting technical input from the project's inception to the current time.

Thanks must also go to the talented team of individuals working with Dr. Brown, who have provided a lot of input: Nancy Wheeler, Richard Dunbar, John Gerth, Mike Wheatley, Doug Aiton, Alan Graham and Ray Trimble, Jim Henry, and a great deal of thanks must also go to Ron Wilks, from the IBM Hursley Laboratory.

References

- IBM[1] APL/PC Version 1, program number 6024077
- IBM[2] APL/PC Version 2, program number 6322911
- IBM[3] APL2 Reference Manual SH20-9227
- IBM[4] IBM Corporate standard CB-3-9045-001
- IBM[5] IBM Corporate standard CB-3-9045-002
- Alfonseca, Tavera and Casajuana[1] "An APL interpreter and system for a small computer", IBM Systems J. 16:1, 1977, p.18-40.
- Alfonseca and Tavera[1] "A machine independent APL interpreter" ,IBM J. of Res. and Development 22:4, 1978, p.413-421.
- Brown[1] "Inside the APL2 Workspace" Dr James Brown, APL85 proceedings.
- Brown[2] "The principles of APL2" Dr James Brown, IBM Santa Teresa technical report TR 03.247
- Gerth[1] "Toward shared variable events implications of □*SVE* in APL2" John A. Gerth, APL83 proceedings.
- Tavera, Alfonseca and Rojas[1] "An APL System for the IBM Personal Computer", IBM Systems J. 24:1, 1985, p. 61-70.