JOURNAL OF THE FACULTY OF ENGINEERING, THE UNIVERSITY OF TOKYO(B).
Vol. XXXIV, No. 1, (1977)

PASCAL——Implementation and Experience

By
Masato TAKEICHI*

(Received Aug. 7, 1976)

The demands for a good programming language suitable for systematic programming and its efficient translator are increasing. The language PASCAL has been well appreciated in the fields of teaching programming and of software research. Its translator is, therefore, desired to be implemented on various computer systems. The author has tried to implement PASCAL compilers on different machines with different methods.

In this paper, the developments of four PASCAL systems, and the experiences with them are described from various aspects. Difficulties arisen in transferring a portable compiler and in developing an efficient one are also discussed.

1. Intoruction

The programming language PASCAL[1)–8)] has been well appreciated mainly in teaching programming and in developing large programs. PASCAL is suitable for describing basic programming concepts, and is useful in teaching programming. One of successful large PASCAL programs is the PASCAL compiler itself. PASCAL compilers so far developed are written in PASCAL and they show important features of PASCAL.

In July 1974, the author began to implement PASCAL for two computer systems, one for education and the other for software research. These machines are: a MELCOM 7700 which has 256 Kbytes of storage, adds two 32 bit integers in 1.8 microseconds, and a FACOM 230–38 which has 224 Kbytes of storage, adds two 16 bit integers in 2.4 microseconds.

Interpretive PASCAL, or PASCAL-P, was first brought up on both MELCOM 7700 and FACOM 230-38, which we will call PASCAL-P (MELCOM 7700) and PASCAL-P (FACOM 230-38) respectively. In July 1975, we obtained a compiler for the CII IRIS 80 which has the same instruction set as that of the MELCOM 7700. Modification was needed to get a compiler for the MELCOM system, which we call PASCAL (MELCOM 7700). A PASCAL compiler for the FACOM 230-38, which we call PASCAL (FACOM 230-38), was then implemented using PASCAL (MELCOM 7700).

In the following sections, the developments of these PASCAL systems, the experiences with them, and som quantitative results are described.

* Department of Mathematical Engineering and Instrumentation Physics.

2. Implementation of PASCAL-P systems

The PASCAL-P systems were brought up on our MELCOM 7700 and FACOM 230-38. The language processed by them is the "Revised PASCAL."[2]

PASCAL-P gives a simple method to implement PASCAL on various machines. A hypothetical stack computer (SC), on which compiled programs run, was designed by U. Ammann.[4] The SC code of the compiler depicted in the T-diagram[5] of Fig. 1 is easily transferred by copying from other PASCAL-P systems. This compiler translates PASCAL programs into the SC code. Therefore, if there is an interpreter for SC running on our machine M, both the compiler and the compiled program can be executed interpretively on M. The whole diagram representing compilation and execution of a PASCAL program, which solves a problem X to give a result Y, is shown in Fig. 2.

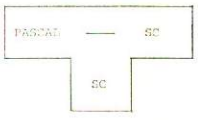

Fig. 1. T-diagram for the SC code of the PASCAL compiler.

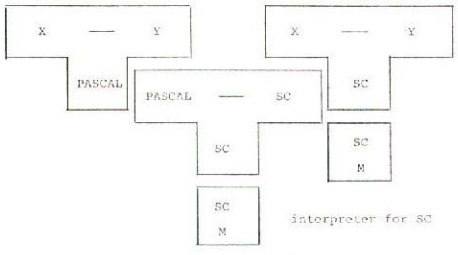

Fig. 2. Compilation and Execution of a PASCAL program by a PASCAL-P system.

We first implemented the system on a MELCOM 7700 installed at the Educational Computer Center of the University of Tokyo. Its main purpose was for students to work with PASCAL for their programming discipline. After five weeks to investigate SC and to write up the interpreter for it, the implementation was completed in September 1974. We studied the interpreter coded in the assembly language for the IBM 370 and the implementation notes,[6] which were only available documents at that time. At the start of this implementation, an experimental interpreter was written in FORTRAN to give support to understanding SC. The interpreter was then written in the assembly language with reference to the FORTRAN version. The size of this interpreter is about 10 Kbytes, and that of the SC code of the compiler is 59 Kbytes. These implementation details are reported in [7].

We next transferred PASCAL-P onto a FACOM 230-38 of the Graduate School of Information Engineering of the University of Tokyo. Experiences with PASCAL-P (MELCOM 7700) made this transportation easy. In March 1975, the first version of PASCAL-P (FACOM 230-38) was completed in only 10 days, which did not process input-output conversions for floating-point numbers.

We like to note that the full description of the PASCAL-P system with explanation on SC is now available.[8]

3. Experiences with PASCAL-P systems

PASCAL-P (MELCOM 7700) has been in practical use to process many student programs of up to 100 lines. PASCAL-P (FACOM 230-38) has been little used in software research except to process simple sample programs.

These systems work fairly well for small programs. They are, however, unsatisfactory for those who want to process practical programs which are of some hundreds lines and require much computation.

It is reported in [6] that compilation of the PASCAL-P compiler of 3400 lines by the interpretive PASCAL-P system requires 108 Kbytes of storage. In the PASCAL-P system, data of basic types such as char, Boolean, integer, and real occupy 64 bits, and no packed data are treated. Moreover, data of powerset types are also treated as those of basic types. It seems to be common in PASCAL implementations that the data of the powerset type require the same amount of some 64 bits as the number of different characters in PASCAL char type, so that each datum of type *set of* char is represented in a single data word. Data of basic types and of powerset types are pushed down into the stack by the same SC instruction. Though a unique size of elements on the stack simplifies the stack operations, the data word is necessarily large enough to contain their values. Large storage requirement is a serious disadvantage of our PASCAL-P systems. More effeorts are needed to improve the storage utilization.

Next deficiency of our PASCAL-P systems to be noted is the running time of the interpreter. Interpretation of each instruction of SC requires about 10 machine instruction steps in the interpreter. Roughly speaking, it takes 10 times more than machine coded programs. We would like to note that the actual running time is more serious than relative estimation in processing practical programs.

Though the PASCAL-P system is a simple method for implementing PASCAL, it is difficult to develop efficient programing systems on PASCAL-P on small computers. However, bootstrapping of a PASCAL compiler with the aid of PASCAL-P was successfully taken on a PDP-10,[9] and recently on a HITAC 8800.[10]

4. Implementation of PASCAL (MELCOM 7700)

In August 1975, a PASCAL compiler which processes the "Standard PASCAL"[3] was transferred onto our MELCOM 7700 from the CII IRIS 80,[11],[12] both of which share the same machine instructions as those of the XDS SIGMA 7.

The compiler for the CII works under the monitor SIRIS 7–8 which is different from the BPM monitor of the MELCOM. The PASCAL system consists of the

PASCAL-monitor with input-output facilities, the compiler, and several library procedures. The PASCAL-monitor and the library procedures written in the assembly language had to be rewritten in order to incorporate with the BPM monitor. It required two weeks in rewriting and debugging.

The trouble was how to transfer the compiler which is written in PASCAL and is also available in the form of the standard relocatable object module (ROM) of the SIRIS 7-8 monitor which is incompatible with that of our BPM monitor. Compiling the compiler program by our PASCAL-P systems seemed to be difficult because of their large storage requirement and of the differences in the languages. The method taken in this transportation is shown schematically in Fig. 3. We first transformed the binary modules ROM into the mnemonic instruction code of the assembly language METASYMBOL (1), which were then assembled under the BPM monitor (2). Undesirable usage of registers was changed in the text of the mnemonic code. After two weeks the first version of the compiler began to work. The compiler generating the same object code as itself was obtained by modifying the code generation part of the PASCAL program (3) and then compiling it by the first one.

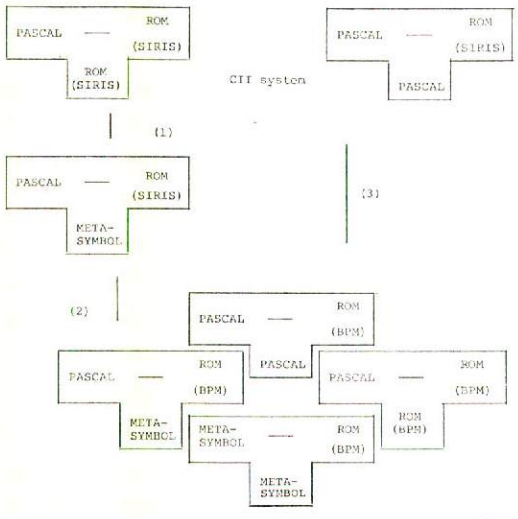

Fig. 3. Transportation of CII PASCAL onto the MELCOM.

5. Experiences with PASCAL (MELCOM 7700)

PASCAL (MELCOM 7700) works very well for practical programs. In order to avoid confusion arising from the differences in the language processed by PASCAL-P, it was not made open to the public until April 1976, while it had been continually used by the author. It is now in general use to process student jobs at the Educational Computer Center. We have found several errors in the compiler,

which are corrected in the PASCAL program.

6. Implementation of PASCAL (FACOM 230–38)

An efficient PASCAL compiler for the FACOM 230-38 was desired because PASCAL-P (FACOM 230-38) could not work well for large programs on this small computer. In the beginning of August 1975, just after the implementation of PASCAL (MELCOM 7700), a PASCAL compiler for the FACOM 230-38 was designed.[13] Among many programs of the compiler, the "Trunk PASCAL" by H. H. Nageli[14] was carefully chosen and rewritten to generate machine instructions of the FACOM 230-38. The trunk compiler is the most concise one in which machine dependent features of the code generation part are intensionally omitted, and is designed to find out errors in source programs by printing the same error messages as those of the CDC version listed in the user manual.[8] One-third of the original text of the trunk PASCAL does not require rewriting. The whole program amounts to indented 5800 lines, where each inserted line contains a single statement for possible changes. Implementation was taken in a way similar to one described in [15], which is shown in Fig. 4.

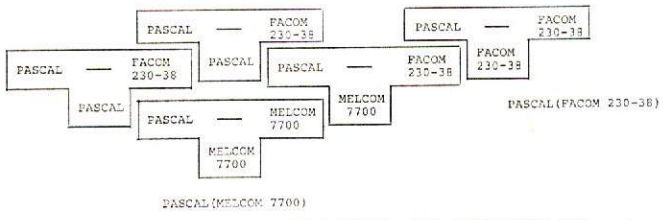

Fig. 4. Development of PASCAL (FACOM 230–38) using PASCAL (MELCOM 7700).

The compiler program was first compiled by PASCAL (MELCOM 7700) three times before it had no syntactic errors. Additional 9 compile/execute runs were necessary to eliminate semantic errors. Most of the errors difficult to find out in the course of debugging runs were related to the variant part of the record structure. PASCAL (MELCOM 7700) tells nothing about whether the selected field is consistent with the value of the tag, while it examines the values of the subscripts of arrays. The binary code was then brought to the FACOM 230-38 after two weeks. The PASCAL-monitor which supports the compiler was written in the assembly language FASP of this machine. We have found two logical errors in the compiler, which were corrected by patching the binary code. On contrast to PASCAL (MELCOM 7700), this compiler generates the absolute binary code which cannot be loaded by the loader of the standard operating system OS 2/VS. The PASCAL-monitor has, therefore, a part of loading the binary code generated by the compiler.

There are some differences between the standard PASCAL[8] and the language

processed by PASCAL (FACOM 230-38). Restrictions to the standard PASCAL are: no packed data type implemented except type alfa, no local file variables, and no parametric procedures. Extension is: a standard shrtint (short integer) type whose value is represented by 16 bits.

7. Experiences with PASCAL (FACOM 230–38)

In October 1975, PASCAL (FACOM 230-38) began to work. It had been used for three months by restricted users who could contact with the implementor if any trouble occurred to the system. Many programs from different research fields were processed and some of them revealed minor errors not found in compiling the compiler itself. Revision was taken occasionally so that the users who caught bugs could work well on the corrected system in a day. It is very important in developing software systems that the implementor is also a user of the system, and has colleagues who would say something about the current system.

PASCAL (FACOM 230-38) is now open to the public and is working very well. Several utility PASCAL programs have been developed.

8. Quantitative Results

8.1. *Storage Requirements (Size in Kbytes)*

	PASCAL-P (MELCOM 7700)	PASCAL-P (FACOM 230-38)	PASCAL (MELCOM 7700)	PASCAL (FACOM 230-38)
(1)	59	59		
(2)			108	117
(3)	10	9		
(4)			10	8
(5)	108	108	32	43

(1) Size of the SC code of the compiler.
(2) Size of the binary code obtained by compiling the compiler itself.
(3) Size of the interpreter for SC excluding library procedures.
(4) Size of the PASCAL-monitor with input-output procedures.
(5) Size of the requried working storage to compile the compiler itself.

8.2. *CPU Time to Compile the Compiler Itself*

PASCAL (MELCOM 7700)	PASCAL (FACOM 230-38)
150 sec.	309 sec.

8.3. *Quality of PASCAL Object Programs Compared with FORTRAN*
Examples are from [16]:

(1) Matrix multiplication $n=100$
(2) Sorting an array of 2000 integers
(3) Finding all possible additive partition of integers 1–30
(4) Counting the charaters in the file of source text of the compiler

	PASCAL (MELCOM 7700)	EXTENDED FORTRAN IV	PASCAL (FACOM 230-38)	OS2/VS FORTRAN S
(1)	1.90	1	1.35	1
(2)	1.75	1	1.24	1
(3)	0.48	1	0.96	1
(4)	0.34	1	0.63	1

8.4 *Comparison of the Time to Execute PASCAL Program on PASCAL-P Systems and on PASCAL Systems*

Execution time for the "Eight Queens Problem"[17]:

PASCAL-P		PASCAL
(MELCOM 7700)	12.4	1
(FACOM 230-38)	13.4	1

9. Comments on Implementation of PASCAL

Though a portable compiler PASCAL-P serves a simple tool to implement a PASCAL on various machines, more efforts would be required than expected to get an efficient compiler on the basis of PASCAL-P. As far as we process only small programs for teaching purposes, the PASCAL-P system works fairly well. Our experiences show that the transferring PASCAL-P onto an alien machine requires only one man-month. It is very attractive for those who want a PASCAL compiler for education. We would like to note that the transportation and the modification should not be taken in parallel. It is not too late after we have transferred PASCAL-P onto our available machine to decide whether its improvement is worth being taken with much work. Experiences will give some hints on refinement and on developing an efficient compiler. If we want an efficient compiler, transportation of PASCAL-P is a first part of the implementation. The next step is to design a machine oriented compiler. We must take much care in this step. Fortunately, we have several good compilers written in PASCAL. When we rewrite a PASCAL compiler to get an efficient one which generates machine instructions of our machine, it is recommended that we first extract the significant part of the existing compiler and then put new statements into its structure. We must carefully observe the whole construction of the original compiler. Though modifying an existing compiler is not always the best way, it reduces efforts considerably. We have much

time to review and refine many times after implementing the first version in this way. With such a simple method, we can obtain a compiler which generates object programs running as fast as those generated by the standard FORTRAN compilers.

Acknowledgment

The author would like to thank Prof. S. Moriguti and Prof. E. Wada of the University of Tokyo for their encouragement during the course of this work. He also wishes to thank Mr. H. Takahashi of Fujitsu Ltd. for writing the PASCAL-monitor of PASCAL (FACOM 230-38) when he was a research fellow in the University of Tokyo.

References

1) Wirth, N.: The programming language PASCAL. Acta Informatica, **1**, 35–63 (1971).
2) Wirth, N.: The programming language PASCAL (Revised Report). Berichte der Fachgruppe Computer-Wissenschaften der ETH Zurich, No. 5, 1973.
3) Jensen, K., Wirth, N.: Pascal——User Manual and Report. Lecture Notes in Computer Science 18, Springer-Verlag, 1974.
4) Ammann, U.: The method of structured programming applied to the development of a compiler. Proc. ACM Int. Comp. Sym. 1973, (Ed. by A. Gunter, B. Levrat and H. Lipps), North-Holland, 93–98, 1974.
5) Earley, J., Sturgis, H.: A formalism for translator interaction. Comm. ACM, **13**, 607 (1970).
6) Deverill, R. S., Hartman, A. C.: Interpretive PASCAL for the IBM 370. Information Science Technical Report No. 6, California Institute of Technology, 1973.
7) Takeichi, M.: On the portability of a PASCAL compiler. Proc. 16-th Programming Symposium of IPSJ, 90–96, 1975.
8) Nori, K. V., Ammann, U., Jensen, K., Nageli, H. H.: The PASCAL 'P' Compiler: Implementation Notes. Berichte der Fachgruppe Computer-Wissenschaften der ETH Zurich, No. 10, 1974.
9) Grosse-Lindemann, C.-O., Lorenz, P.-W., Nageli, H. H., Stirl, P. J.: A PASCAL compiler bootstrapped on a DEC-system 10. Fachtagung uber Programmiersprachen, Lecture Notes in Computer Science 3, Springer-Verlag, 101–113, 1974.
10) Ishihata, K., Hikita, T.: Bootstrapping PASCAL Using a Trunk. Technical Report 76–04, Dept. of Information Science, University of Tokyo, 1976.
11) Mancel, P., Thibault, D.: Transport d'un compilateur PASCAL: Ecrit en PASCAL d'un CDC 6400 sur un CII IRIS 80. These de Docteur Ingenieur, Universite Paris VI, 1974.
12) SFER PASCAL, Le Langage de programmation PASCAL——compilateur pour les ordinateurs CII 10070, IRIS 80. IRIA, 1975.
13) Takeichi, M.: PASCAL Compiler for the FACOM 230–38, Technical Report, 1975.
14) Nageli, H. H., private communication.
15) Welsh, J., Quinn, C.: A PASCAL Compiler for the ICL 1900 Series Computer. Software——Practice and Experience, **2**, 73–77 (1972).
16) Wirth, N.: The Design of a PASCAL Compiler, Software——Practice and Experience, 1, 309–333 (1971).
17) Dijkstra, E. W., Hoare, C. A. R., Dahl, O.-J.: Structured Programming. Academic Press, 1972.