FROM THE EDITOR

The fourth newsletter is long overdue, the third being published in February 1975. There have been many significant events that need announcing. The highlights are:

- Release 2 of PASCAL 6000-3.4 has been made available by Dr. Urs Ammann at Eidgenossische Technische Hochschule in Zurich, Switzerland. It is also available from Mr. George Richmond at the University of Colorado Computing Center in Boulder, Colorado and Mr. Carroll Morgan at the Basser Department of Computer Science, University of Sydney, Australia. Many improvements have been made over Release 1 and future developments are promised. See page 73.

- Mr. Carroll Morgan of the Basser Department of Computer Science at the University of Sydney, Australia has kindly agreed to distribute ETH Pascal and portable Pascal for Australia and neighboring regions. Interested parties should contact Mr. Morgan for more information.

- Mr. Andy Mickel of the University of Minnesota kindly agreed to take over editorial control and publication of the Pascal Newsletter commencing with issue Number 5 in September 1976. He is also organizing a User's Group. See pages 88 and 89.

- An improved portable Pascal has been released from ETH, Zurich. See page 81.

- News of Pascal compilers for numerous machines has been received. See pages 96 and following.

- An expanded bibliography of Pascal literature has been compiled. See pages 100 and following.

I have enjoyed producing the first four newsletters but I have found it to be very time consuming. I am grateful to Andy for taking over these duties with enthusiasm. I will continue to distribute Pascal in North America thru the University of Colorado.

~ George Richmond

Technische Hogeschool Twente
Enschede, Nederland
postbus 217

Comment on an extension of PASCAL

In a recent PASCAL-newsletter (#3) we find a proposal by N. Wirth regarding "A generalization of the READ and WRITE procedures".
The proposal, which has been implemented in the PASCAL 6000-3.4 compiler, considers the procedure

read(f,x) to be synonymous to the sequence:
x := f↑ ; get(f) , and
write (f,x) to be synonymous to the sequence
f↑ := x ; put(f)

It is my opinion that standard procedures should only be introduced when

a) their body cannot be described in the language PASCAL, or when
b) there exists an essentially faster mapping of the body directly onto the machine-code (e.g. one instruction that does the job), than the compiled code of the PASCAL-body admits.

Standard procedures of type a) should be a standard for the languages as such.
Standard procedures of type b) should be standard for a particular configuration only, in order to give the programmer a convenient form of access to particular aspects of the machine at hand.
When transferring a program based on type b) standard procedures to another machine, the portability is ensured by supplying the appropriate procedure-declarations.
Procedures that do not satisfy any of the above criteria may nevertheless be placed for convenience in a library of (basically) source-code procedures.
In line with the above philosophy we have decided not to implement the arithmetic functions like sin, atan, ln etc., in our PASCAL-version for the PDP/11 series. Apart from the relief for the builder of the system who has to implement these routines in machine code, one may well suppose that an abundance of standard procedures can be disadvantageous to the compactness of both compiler and runtime system.

Now let us take a look at the file-proposal with the above in mind. Then the proposal only satisfies criterion (a). This criterion is satisfied

since if the procedures were to be described in PASCAL the type of the parameters f and x had to be fixed.
It would, however, be a very minor job to declare the procedures "read" and "write" to be specific for a particular type of parameter. The bodies would all be equal, only the parameter specification having to be adapted. This seems hardly a problem since the number of different file types that are dealt with in one program will be quite limited.
My suggestion is that - if the proposed standard procedures are not available - one simply declares:

```
readplop (f : file of plop ; x : plop);
begin x := f↑ ; get(f) end
```

and

```
writeplop (f : file of plop ; x : plop);
begin f↑ := x ; put(f) end
```

There is, however, a much more important aspect of the current procedures "read" and "write" that asks for a generalization.
Whereas the files they implicitly operate on (input and output) are "file of char" , the actual parameters, may be of arithmetical type and a conversion from or to character sequence is specified by the somewhat extraordinary remaining parameters (in the case of write).
It seems to me that there is a greater need to generalize the procedures "read" and "write" in this respect, viz. making the conversion available for all files of char. One may now visualise how a PASCAL program may build two output streams in parallel, a feature even the compiler could use.
Alternatively, the conversion routines themselves could be made available, but because of rigid data-sizes in PASCAL there is no way of properly dealing with the format specifications. This appears one of the major arguments for considering "read" and "write" as standard procedures.

One might counter my arguments by stating that it is possible to describe "arithmetic quantity ↔ character sequence" conversion wholly in PASCAL but apart from the above arguments concerning the need for flexible data-formats, it is impossible to describe the conversion from a real

quantity to a character sequence in terms of real operations because of the implicit inexactness of real arithmetic. In that case at least a standard procedure has to be supplied for the conversion of a real quantity to (a number of) integer quantities. In other words: if one has a real quantity x, for which . $1 \leq x < 1$, it is not guaranteed that $1 \leq x * 10 < 10$, and therefore trunc (10 * x) may not deliver the correct digit! This example shows, by the way, that "trunc" is a very illdefined function, which should be abolished from programming languages!

C. Bron.

Enschede 25.5.1975.

c.c. Wirth.
Richmond.

LUCIEN FEIEREISEN
INSTITUT F. BIOKYBERNETIK U. BIOMED. TECHNIK
UNIVERSITAET KARLSRUHE
D-75ØØKARLSRUHE 1
KAISERSTR.12

Germany

KARLSRUHE, 30-JUN-75

MR. GEORGE H. RICHMOND
UNIVERSITY OF COLORADO
COMPUTING CENTER
PSRB #3
BOULDER, COLORADO 80302

DEAR MR. RICHMOND,

I THANK FOR YOUR LETTER FROM THE 16-MAY-75. THE PASCAL COMPILER BASED ON JANUS HAS BEEN IMPLEMENTED ON THE PDP-11/45 RUNNING UNDER THE DOS/BATCH OPERATING SYSTEM AND A PRELIMINARY VERSION HAS BEEN RELEASED TO A LIMITED NUMBER OF SITES.

THE AVAILABLE PASCAL-COMPILER (PAS74) IS WRITTEN IN THE LANGUAGE IT TRANSLATES. ITS JOB IS TO ANALYSE A PASCAL PROGRAM FOR SYNTACTIC ERRORS AND TO GENERATE CODE FOR A STANDARD ABSTRACT MACHINE CALLED JANUS [COL73].

THE MACROGENERATOR STAGE2 [WAI73, MEI74] MAPS THIS SYMBOLIC JANUS CODE INTO THE PDP-11 ASSEMBLER CODE, MACRO-11. THE JANUS/MACRO-11 TRANSLATOR IS DEFINED BY SET OF STAGE2-MACROS. AS THE PRODUCED CODE IS ORGANIZED AROUND AN IDEALIZED ABSTRACT MACHINE THERE IS LEFT CONSIDERABLE ROOM FOR OPTIMIZATION. CODE AND DATA ARE SPLIT INTO 2 SEPERATE FILES.

THE FINAL TRANSLATION TO ABSOLUTE CODE IS PROVIDED BY THE NORMAL PDP-11 ASSEMBLER AND LINKER. THE PASCAL LOADER LOADS THE DATAFILE INTO THE USER-DATA-SPACE (32 K MAXIMUM) AND THE CODEFILE INTO THE USER-INSTRUCTION-SPACE (32 K MAXIMUM).

THE PASCAL-COMPILER REQUIRES TO COMPILE ITSELF (PASCAL/JANUS) 64 K WORDS OF MEMORY AND 600 SEC. THE TRANSLATION AND EXECUTION TIME OF PASCAL AND FORTRAN WERE COMPARED: THE WHOLE TRANSLATION PROCESS IS ABOUT THE FACTOR 1,29 THAT OF EQUIVALENT FORTRAN PROGRAMS ON THE PDP-11, WHEREAS THE EXECUTION SPEED IS ABOUT THE FACTOR 0,65.

ALL FEATURES OF THE USED PASCAL LANGUAGE (CLASS AND ALFA VARIABLES, VALUE AND FILE DECLARATIONS, GLOBAL EXITS, ...) ARE IMPLEMENTED EXCEPT FOR PARAMETRIC PROCEDURES. THE FLOATING POINT PROCESSOR IS USED FOR REAL ARITHMETIC AND FOR TEXT-HANDLING (ALFA TYPE). THE I/O CONCEPT INCLUDES CONCURRENCY AND EXPLIZIT OUTPUT CONTROL (GRAPHIC OUTPUT POSSIBLE TOO).

THE PASCAL-11 USER'S GUIDE PROVIDES INFORMATION NECESSARY TO INSTALL THE PASCAL-11 SYSTEM AND TO TRANSLATE AND EXECUTE PASCAL PROGRAMS ON THE PDP-11/45.

THE PASCAL-COMPILER CAN BE MOVED TO ANOTHER COMPUTER WITH A LIMITED AMOUNT OF EFFORT: THE JANUS TRANSLATOR, WHICH TRANSLATES THE MACHINE INDEPENDANT JANUS CODE INTO THE ASSEMBLY CODE FOR THE TARGET MACHINE, HAS TO BE REWRITTEN, I.E. ANOTHER SET OF MACROS FOR THE NEW MACHINE HAS TO BE DEFINED. THE NECESSARY DOCUMENTATION IS AVAILABLE FROM [WEB73].

THE PASCAL-COMPILER (WRITTEN IN PASCAL AND JANUS), THE STAGE2 MACROGENERATOR (DOS AND RSX-11 VERSION) [FEI74] AND THE PASCAL-11 USER'S GUIDE [FEI75] (20 PAGES) AS WELL AS THE WHOLE PASCAL-11 SYSTEM ARE AVAILABLE.

REFERENCES:

COL73 COLLIAN,S.S., POOLE,P.C., , WAITE,W.M.
THE MOBILE PROGRAMMING SYSTEM: JANUS
UNIVERSITY OF COLORADO, BOULDER, 1973
SOFTWARE PRACTICE AND EXPERIENCE

FEI75 FEIEREISEN,L.
PASCAL-11 USER'S GUIDE
UNIVERSITAET KARLSRUHE MAI-75

HEI74 HEINRICH,F.H.
STAGE2 FOR THE PDP-11 DECUS 1974

PAS74 PASCAL COMPILER
AMMAN,U., SCHILD,R. DATE:23/11/72
FACHGRUPPE COMPUTERWISSENSCHAFTEN
EIDG. TECHNISCHE HOCHSCHULE
CH-8006 ZUERICH

REVISION TO PRODUCE JANUS
LARRY B. WEBER
UNIVERSITY OF COLORADO SPRING 1973

REVISED BY LUCIEN FEIEREISEN
UNIVERSITY OF KARLSRUHE FALL 1974

WAI73 WAITE,W.M.
IMPLEMENTING SOFTWARE FOR NON-NUMERIC APPLICATIONS
PRENTICE-HALL IN., ENGLEWOOD CLIFFS, N.J. (1973)

WEB73 WEBER,L.B.
A MASCHINE INDEPENDANT PASCAL COMPILER
MS-THESIS, UNIVERSITY OF COLORADO, BOULDER, 1973

I AM PLEASED TO BE OF ASSISTANCE, BOTH NOW AND IN THE FUTURE.

YOURS SINCERELY

LUCIEN FEIEREISEN

UNIVERSITÄT HAMBURG

INSTITUT FÜR INFORMATIK
Prof. Dr. H.-H. Nagel

Institut für Informatik
2 Hamburg 13, Schlüterstraße 66-72

Datum July 1st, 1975

Dear Mr. Richmond,

The enclosed summary informs you about our PASCAL-compilers available for the DECSystem-10. In case you enquired recently about our compiler and did not yet receive an answer, please excuse me. I have been busy (amongst other tasks) to prepare this version for distribution.

☐ You are on a distribution chain for three DECtapes and scheduled to receive them from

☐ Please check here if you are interested to obtain our PASCAL compiler.

Shipment requires (please check)

☐ 1 Dectape for the PASCAL-compiler generating directly executable, sharable object code

☐ 2 Dectapes for the PASREL-compiler generating LINK-10 compatible relocatable object code, the PASCAL source level debugging system (PASDDT), the crossreferencing program CROSS and the PASCAL-Help file (the latter one in German since I did not want to delay shipment any further by the time required to translate this file into English).

☐ 1 small MAGtape if you don't have Dectape drives at your installation. Since MAGtapes require more trouble at our site, DECtapes are preferred.
Please do not send tapes if you are located in continental US or Canada since I intend to refer your name to someone in your vicinity who has received these versions.

☐ Would you see a possibility to provide a copy of these files to someone else if asked to do so?

If you are interested, please return this questionaire to
H.-H. Nagel, Institut für Informatik
Schlüterstraße 70, D-2000 Hamburg 13

Yours sincerely

H.-H. Nagel

The PASCAL implementation for the DECSystem-10 has been considerably improved and enlarged. It now supports all phases in the generation, debugging and maintenance of PASCAL programs.

1. The editing phase by the formatting features of CROSS.
2. The compilation phase by offering:
   2.1 a compiler named PASREL generating relocatable object code compatible with LINK-10. Output from this compiler will automatically direct the loader to search the FORTRAN-library for standard functions SIN, COS etc. if necessary. External procedures (e.g. written in MACRO-10 or separately compiled PASCAL procedures) can be linked on. A source level debug option is available (see p. IV).
   2.2 a more compact, faster compiler named PASCAL generating directly executable, sharable object code. Use of this compiler is recommended, if

       no external procedures
       no debug option
       no standard functions from the FORTRAN library

   are required.
3. The debugging phase: breakpoints can be set at runtime based on source program line numbers. After a program stops at such a breakpoint, variable locations can be inspected and modified using the source program identifiers (see p. IV).
4. The maintenance phase: the program CROSS generates
   - an incentated source program listing with extension .CRL
   - markers in the left margin for each start and termination of nested statements
   - a crossreference list of all source program identifiers
   - a survey of the static procedure nesting
   - for each procedure a list of which procedures it activates and by which procedures it is activated
   - an indentated source program file with extension .NEW

The following section summarizes new features available in both compilers (see the PASCAL.HLP file distributed with PASREL for further information):

5.1 READLN {(<file identifier>)} skips over the rest of the current line until the next end-of-line is detected. It accepts further arguments.

5.2 PAGE {(<file identifier>)} appends a <carriage return><form feed> to the FILE OF CHAR denoted by <file identifier>. If none is given OUTPUT is assumed. <formfeed> advances to the beginning of the next page.

5.3 The procedures PACK and UNPACK have been implemented with an optional fourth argument. These procedures are much more effective in packing or unpacking larger arrays than a FOR-loop using indexed access to components of such an array.

5.4 The sequences (* and *) are recognized as opening and closing comment brackets in addition to % and \.

5.5 CROSS and both compilers accept the general file specification allowed by the TOPS-10 monitor.

5.6 A constant subrange may be given in sets, e.g. ['A' .. 'C'] instead of ['A', 'B', 'C'].

5.7 An OTHERS branch may be specified in CASE-statements.

5.8 Compiler options (see also 5.14):

5.8.1 %$C+\ generates instructions for runtime checks at array indices and assignment to scalar and subrange variables.
%$C-\ suppresses code generation for runtime checks
Default: C+

5.8.2 %$L+\ appends the symbolic version of the object code generated to the source program listing for each procedure and adds the starting address of the object code for each source program line
%$L-\ no symbolic object code listing
Default: L-

Since runtime errors still give only the object code address besides the message identifying the error, compile the program with the compiler option L+ in addition to C+ in order to learn from this listing, to which source program line the error address belongs.

5.9 The following standard functions are available to both compilers

function	of result type	yielding
TIME	INTEGER	time in milliseconds
RUNTIME	INTEGER	CPU-time in milliseconds

5.10 For initialisation of global variables at compile time use INITPROCEDURE.

5.11 The LOOP statement is available.

5.12 The standard procedures RESET/REWRITE can be used with up to four optional arguments allowing full file specifications at runtime

5.13 Pascal programs to be compiled by PASREL may use the following additional standard functions (all functions and arguments are of type REAL) from the FORLIB on the logical device SYS:

SIN	COS	ARCTAN	EXP	SQRT	RANDOM
SIND	COSD	TANH	LN		
ARCSIN	ARCCOS		LOG		
SINH	COSH				

function	of result type	yielding
DATE	PACKED ARRAY[1..9] OF CHAR	'DD-MMM-YY' with D=day, M=month, Y=year

5.14 Following the head of a procedure/function declaration by

EXTERN <language symbol>;

will direct the compiler to provide for linkage to an external procedure/function.

<language symbol>:::=empty|FORTRAN|ALGOL|COBOL.

The language symbol determines the conventions for parameter passing to an external procedure/function. If none is provided, PASCAL is assumed. If any of the three nonempty language symbols is indicated, the loader is directed to search the corresponding library on logical device SYS.

If a group of PASCAL procedures without a main program has to be compiled separately, use %SM-\ at the beginning of the corresponding PASCAL source file. In this case the outermost procedure/function names will automatically be declared as ENTRY by the compiler. Include their .REL files when loading!

5.15 BREAK {(<file identifier>)} forces the current buffer contents to be output to the file specified by <file identifier>. If none is specified, TTY is assumed. This feature is useful, too, for intercomputer-communication at the PASCAL level.

To use the crossreference listing program

```
.RUN CROSS
FILE: <filename>          after FILE: : is typed by CROSS give
                          source filename in the format
                          DEVICE: FILNAM.EXT [project# , progr.#]
                          everything except FILNAM may be omitted
```

To use PASCAL:

```
.RUN PASCAL
*<filename>               source filename specification (see CROSS)
{NO} ERROR DETECTED       If an error has been detected, object
                          code is not available for execution!
EXIT
.RUN <filename> ##        The core requirement indicated by ## must
                          be estimated from the length of LOW and
                          SHR file plus space for stack and heap.
                          Since error messages will appear if core
                          space is insufficient, trial is often
                          the quickest approach.
```

To use PASREL:

```
.RUN PASREL

*<filename>               source file specification (see CROSS)
{NO} ERROR DETECTED       (see PASCAL)
HIGHSEG  : M K            M indicates size of high segment (code)
                          in Kwords
LOWSEG   : N K            N indicates size of low segment (data)
                          in Kwords
RUNTIME  : T              T indicates CPU-time used for compilation
EXIT
.LOAD <filename>          loading the program
LINK: LOADING
EXIT
.SAVE <filename> W        The total core requirement W must be
                          given. W ≥ M + N + 4 Kwords
                          If no debug option has been specified in
                          the source program, the save-file can be
                          made sharable by using the monitor com-
                          mand SSAVE <filename> W.
JOB SAVED
.RUN <filename>           execute the program
```

Instructions to generate a new compiler version can be found at the beginning of each PASCAL-compiler source version.

Note:

Not yet implemented:

- formal procedure/function arguments
- branch out of a procedure/function (label declaration is not yet required)

6. DEBUG option

6.1 Indicate debug option in (part of the) source program text: %$D+\. If no debugging is required in later parts, follow the section to be debugged by %$D-\ since this will save core and runtime in those sections not to be debugged.

6.2 Use PASREL etc. (see above) to generate an executable SAV file, giving W ≥ M + N + 8 to allow working space for the debug code.

6.3 Get the listing of the compiled program in order to know exactly where to set breakpoints by

```
PRINT <filename>.LST
```

6.4
```
.RUN <filename>           begin execution of this program
*                         answer with <carriage return>
$ STOP AT MAIN BEGIN
$                         enter breakpoints using the follow-
                          ing commands
```

6.5 `$ STOP <LINE>` set a stop at the begin of the source line indicated by line number <LINE>

<LINE> ::= <LINENUMBER> | <LINENUMBER>/<PAGENUMBER>

<LINENUMBER>::= <UNSIGNED INTEGER>

<PAGENUMBER>::= <UNSIGNED INTEGER>

If no pagenumber is given, 1 is assumed.

6.6 `$ STOP NOT <LINE>` delete breakpoint at line <LINE>

6.7 `$ STOP LIST` list all current breakpoints on the terminal

6.8 `$ <VARIABLE> =` give the current contents of the location indicated by the source identifier (possibly expanded by qualifiers); the scope rules applying to the source statement corresponding to the current breakpoint uniquely determine the variable location from the identifier given. All qualifiers legal in Pascal may be used (i.e. pointers, record fields, array components)

6.9 `$ <VARIABLE>:=<VARIABLE OR CONSTANT>` The variable or constant value on the right hand is assigned as current value to the variable indicated in the left side

6.10 `$ TRACE` Backtrace of procedure nesting from Breakpoint to main; exposes activating procedures with linenumber/pagenumber of activation points

6.11 `$ END` leave debug mode and continue execution. If any breakpoint is reached, the message

6.12 `$ STOP AT <LINE>` appears on the terminal

6.13 asynchronous stop If the program has been compiled with the debug option, it's execution can be interrupted by two successive control C:

```
↑C↑C
.DDT
```

typing DDT will then transfer control to the debugging mode

`$ STOP BETWEEN <LINE1> AND <LINE2>` will appear on the terminal to indicate where the program has been interrupted. Use of commands described in 6.5 through 6.11 is now possible.

INSTITUT NATIONAL POLYTECHNIQUE
DE LORRAINE
DÉPARTEMENT INFORMATIQUE

Nancy, le 4 Juillet 1975

Monsieur Alain TISSERANT
Ecole des Mines
Département Informatique
Parc de Saurupt
54042 NANCY CEDEX
France

Mr George H. RICHMOND
Pascal Newsletter Editor
University of Colorado
Computing Center
3645 Marine Street
BOULDER, Colorado 80302
U.S.A.

Dear Sir,

A Pascal compiler for Télémécanique T1600 and Solar minicomputers is under development; a first version will be available in September 1976. These computers have a 16 bits words size, and no virtual memory facility. Our compiler will run with 24K words.

We are implementing a segmentation mechanism, reflecting both Pascal programs structure and the Solar computer architecture. At each procedure call, a new "segment" will be created for code and local data. A specialised monitor manages core memory, and swap operations.

The Pascal-P compiler is being modified (without change in the language accepted), in order to get code adapted to our data structures representation and our particular procedure linkage method.

We are using an existing Pascal compiler (on the CII Iris 80) for first binary code generation of the Solar compiler.

All these mechanisms are fully transparent to the user. By careful use of the particularities of special instructions and the architecture of the computer, we hope to get a high speed, easy to use Pascal system.

Sincerely,

A. TISSERANT

(Enc.)

11

THE UNIVERSITY OF MANITOBA
DEPARTMENT OF COMPUTER SCIENCE
WINNIPEG, CANADA R3T 2N2

WBF/CMcL

15th July, 1975.

George H. Richmond,
Computing Center,
University of Colorado,
3645 Marine Street,
BOULDER,
Colorado 80302,
U.S.A.

Dear Mr. Richmond,

The enclosed is being sent to all the people who have written to us requesting information about our PASCAL implementation.

I realize that up until now, very little information about the project has been released. I think that this description gives a fair representation of our compiler as it currently exists.

The compiler was written as my Ph.D. project under the supervision of Professor James M. Wells. Professor Wells is currently on sabbatical leave in Ottawa. For this reason, I would appreciate it if you would include my name on your PASCAL Newsletter distribution list.

I am very interested in descriptions of other PASCAL Compilers and interpreters for IBM machines. If you have any information on core requirements, compile speeds, and whether or not the full language is supported, for the Grenoble, Stanford or Cambridge projects, I would appreciate hearing from you.

Yours sincerely,

W. Bruce Foulkes

12

THE UNIVERSITY OF MANITOBA

DEPARTMENT OF COMPUTER SCIENCE

WINNIPEG, CANADA R3T 2N2

WBF/CMcL

July, 1975.

Dear Sir or Madam,

We are announcing the availability of a PASCAL compiler for IBM 360/370 computers, developed by the Department of Computer Science at the University of Manitoba. The compiler was written by Mr. W. Bruce Foulkes under the supervision of Professor James M. Wells.

The compiler is one-pass and uses a top-down parsing strategy. A generated assembler parser is produced by the translator writing system SYNTICS. All semantic routines are written in PL360, while system interfaces are written in assembler.

The compiler is not a re-write, modification, or bootstrap of any previous PASCAL compiler. The compiler uses some routines provided by the SYNTICS system and borrows some ideas and code from the ALGOLW compiler for code generation, built-in functions, and I/O.

This version of the compiler requires approximately 170K bytes. This size is variable, but the minimum size for compiling a meaningful program is approximately 150K.

Compile speed for test programs has been in the range 125-200 lines per second on an IBM 370/158. This excludes the set-up time of approximately 0.4 sec.

A great deal of compile-time checking is done and approximately 130 different error and warning messages are provided.

The production of run-time checking code for array subscripts, subrange assignments, values returned by PRED and SUCC, etc., can be turned on or off at will. Run-time interrupts are trapped with a SPIE macro. There are about 40 run-time error diagnostics in total. Each error diagnostic consists of an error message, location in the current segment, the invalid value if

.../2

appropriate, and a traceback of all segments invoked.

Linkage, although not completely standard between PASCAL segments, appears to be standard IBM to any external segments, allowing linkage to routines written in other languages.

The compiler supports a subset of the language described in the Revised Report. The main omissions are the following:

- only the standard input and output files SYSIN and SYSPRINT are supported. All I/O is done through the use of READ, READLN, WRITE, WRITELN, EOLN, and EOF. The I/O is not exactly standard; in particular, formatting is also allowed on input.
- the program header is not used. SYSIN and SYSPRINT must always be provided.
- packed arrays and records are not supported.
- only the simple forms of procedures NEW and DISPOSE are allowed. Tagfield values may not be specified. No garbage collection is done.
- global labels are not implemented.
- subranges of characters are not allowed.

With the above exceptions, the language supported is very close to that described in the Revised Report.

Seven standard scalar types are provided: SHORT INTEGER, INTEGER, REAL, LONG REAL, BOOLEAN, CHAR and STRING.

Built-in functions include: ABS, SQRT, EXP, LN, LOG, SIN, COS, ARCTAN, SQR, SUCC, PRED, ODD, ROUND, TRUNC, ORD, CHR, CARD and CPUTIME.

The compiler checks for overflows on all tables and produces terminal error messages. The main table sizes may be modified using parameters on the EXEC card. The source for an initialization routine will be provided which sets the size limits for all compile-time tables, and also sets defaults for compiler flags (such as whether run-time checking code should be produced). This should allow the compiler to be tailored to suit the needs of any installation. The remainder of the source will not be released at this time.

There are two main limitations imposed by the compiler. The maximum nest allowed for procedure and function declarations is 5, and all program segments are restricted to 4K bytes of code.

The compiler has not undergone large-scale production testing; for this reason, no guarantees are made as to its reliability. Considering the interest which has been shown in the compiler, we feel that we cannot justify delaying its release any longer.

.../3

The PASCAL compiler may be acquired by sending the PASCAL Order Form and the signed DISTRIBUTION AGREEMENT together with $50 (payable to the Department of Computer Science, University of Manitoba), to the PASCAL Distribution Manager.

The tape will contain the object modules necessary to generate the compiler along with sample JCL, test programs and a user's guide.

After a suitable test period an updated version of the compiler may be offered, but no promises to this effect are made.

We hope that few problems will be encountered.

Please see the enclosed material if you are interested in ordering the compiler.

W. Bruce Forsythe

PASCAL Distribution Manager

UNIVERSITY OF MINNESOTA
TWIN CITIES

University Computer Center
227 Experimental Engineering Building
Minneapolis, Minnesota 55455

July 16, 1975

To the Editor, Pascal Newsletter:

We at the University of Minnesota would like to participate in a Pascal User's Group for North America to help distribute and support PASCAL. In communications with various other sites: Alfred Towell at Indiana University, Dave Tarabar at University of Massachusetts, and George Richmond at Colorado University, there seems to be a desire to form such an association. Perhaps a kind of conference would be appropriate for getting started.

The usage of PASCAL at our site has been heavy lately. PASCAL is being used in general applications on not only the MERITSS CDC6400 state-wide timesharing network but also at the University of Minnesota's batch computing facility, a CDC CYBER 74. We have locally modified the PASCAL system totally in a cosmetic way: fixing bugs, making interface changes for the KRONOS 2.1 operating system and making PASCAL available under the TELEX timesharing supervisor available with KRONOS. A dozen or so sites have our modifications for interactive usage, however we don't know as yet how useful they are for INTERCOM under SCOPE 3.4, although Urs Ammann of Zurich seems to think we went about our changes in a decent way, which may make them a good model to follow.

We would like to caution others about what "improvements" they make to their implementation of standard PASCAL.* For one thing the old compile to core PASCAL compiler for CDC machines was changed by many people in ways that violated the underlying principles of the language. For example: simplicity in design (which implies simplicity in description- in other words, few exceptions to the rules); a specific infraction being the addition of a step specification in for loops by one installation.

In reply to a letter to the editor of 6 August, 1974 by George Poonen in Pascal Newsletter No. 3, we would like to reply that we also deplore "dialect_s" of PASCAL. However, BLAISE and SUE are not dialects but other PASCAL-like languages. Further, the Axiomatic Definition and the Revised Report define the standard semantics and syntax of the language.

* See the very interesting article: "An Assessment of the Programming Language PASCAL" by Niklaus Wirth in the June, 1975 issue of SIGPLAN Notices, Proceedings: International Conference on Reliable Software.

If what is meant is trying to resolve a standard for extensions to the language (such as a value-initialization facility) then that is another question. Perhaps this should be investigated.

To have PASCAL succeed at a given installation (with the goal of being used as much or more than FORTRAN) may require the local maintainer's recognition of his or her resonsibility to and power over the lives of all the PASCAL users affected. Also consider the ideas put forth by Prof. William Waite, Colorado University, in a guest editorial to the Vol. 3, No. 3 1973 issue of Software Practice and Experience. He uses the analogy of organisms (language processors) in an ecosystem (computer center). PASCAL is a very good product which sells itself-but by giving the compiler inadequate support, it can fail with certainty. Support includes not only simple availability, but also publicity and all the other amenities of programming life which now make FORTRAN easy to use. Examples are utility routines, libraries, program preparation equipment with the proper character sets, etc.

This summer we are engaged in additional enhancements to PASCAL's support at the University of Minnesota. The Computer Science Department here has now adopted PASCAL throughout its curriculum. By October we will be willing to share with other sites several of the documents we will have produced.

Andrew Mickel John Strait

Andrew Mickel and John Strait
University Computer Center
227 Experimental Engineering
University of Minnesota
Minneapolis, MN 55455

17

AM/JS/ks

DIKU DATALOGISK INSTITUT KØBENHAVNS UNIVERSITET
SIGURDSGADE 41, DK-2200 KØBENHAVN, DANMARK, TLF. (01) TA 94 66

Mr. George H. Richmond
University of Colorado
3645 Marine Street
Boulder, Colorado 80302
U.S.A.

July 25, 1975
JSM/HG

Dear Mr. Richmond.

Having received your list of PASCAL implementations I will ask you to correct the information of our compiler thus:

Implementation Route : PASCAL-P1 Bootstrap
Implementation Status: Complete, Available for distribution.

Sincerely

J. Steensgaard-Madsen

18

encl.

PRELIMINARY DESCRIPTION OF A PASCAL COMPILER FOR UNIVAC 1100.

J. Steensgaard-Madsen
Datalogisk Institut
Sigurdsgade 41
DK-2200 Copenhagen
DENMARK

Introduction.

The following text describes in short a PASCAL compiler for UNIVAC 1100 machines operating with EXEC 8. The system is developed from a PASCAL P compiler obtained from professor Niklaus Wirth. The work has been done at Datalogisk Institut, University of Copenhagen by three students

Arne Kjær
Jan Højlund Nielsen
Henrik Snog

and a teacher

Jørgen Steensgaard-Madsen

The present (preliminary) description is not complete in every detail, but tries to convey all relevant information to the vast majority of users. It should be used together with the book

PASCAL User Manual and Report
by Kathleen Jensen and Niklaus Wirth
Springer Verlag 1974
(Lecture Notes in Computer Science no. 18).

Representation of PASCAL programs.

The representation of PASCAL programs for UNIVAC 1100 is based on the standard representation using ASCII character set. This is converted to FIELDATA using the rules fixed by UNIVAC, except for opening and closing brace, { and } , which are not used. This means that ↑ is converted to Δ and that comments are enclosed in (* and *).

Restrictions. (July 75)

Variables of type TEXT, except INPUT and OUTPUT, cannot be used. Page procedure is not implemented.

DISPOSE is not implemented.
File components containing files cannot be used.
Standard procedures cannot be passed as parameters.
Fields of packed structures cannot be substituted for var parameters.
Sets must be over base types containing at most 72 values (in case of INTEGER it must be a subrange contained in 0 .. 71).

Additional standard identifiers.

```
const     ALFALENG = 12;
type      ALFA = PACKED ARRAY [ 1 .. ALFALENG ] OF CHAR;
         HALFA = PACKED ARRAY [ 1 .. 6 ] OF CHAR;

procedure HALT;
   (* terminate execution *)

procedure MARK ( var I : INTEGER );
   (* returns with I information to be used in recollecting
      storage allocated by subsequent calls of NEW *)

procedure RELEASE ( I: INTEGER );
   (* releases storage allocated by calls of NEW since
      the call of MARK that must have set I *)

procedure WRITEPAGE;
   (* advances the printer so that next line is printed
      as fist line on a new page *)

procedure CLOSE ( var F : any file );
   (* this is a file operation, which must be executed
      as the final operation on external files *)
```

Input / Output.

The procedures read and readln take as parameters variables of type CHAR, INTEGER, REAL, HALFA and ALFA. Except in case of CHAR, where just one character is read, leading blanks are skipped and the following characters are analysed. In case of ALFA (HALFA) at most 12 (6) nonblank characters are read and stored left justified and blankfilled. With multiple parameters an error exit caused by an end of file condition will only occur if EOF is TRUE prior to the call.

Parameter specifications.

A name may be associated with a specification of formal parameters. This is done in the parameter definition part placed after the variable declaration part. The syntax is

```
<parameter definition part> ::=
  param  <parameter declaration> ;
        {<parameter declaration> ;}

<parameter declaration> ::= <parameter identifier> =
  ( <formal section> {; <formal section>} )
```

The parameter names may then be used in the declaration of procedures and functions

```
 <procedure heading> ::=
   procedure <identifier> ; |
   procedure <identifier> ( <specification> );

 <function heading> ::=
   function <identifier> : <result type> ; |
   function <identifier> ( <specification> ): <result type>;

 <specification> ::=
   param <parameter identifier> |
   <formal section> {; <formal section>}

 <formal section> ::=  <formal parameter section> |
   procedure <identifier> ( <parameter identifier> ) |
   function  <identifier> ( <parameter identifier> ) :
   <result type>
```

This syntax allowes the complete specification of formal procedures and function which is required in PASCAL for UNIVAC 1100.

Loop statement.

For historical reasons the loop statement is included in PASCAL for UNIVAC 1100.

```
<loop statement> ::=
  loop {<statement> ;}
  exit if <expression> {; <statement>}
  end
```

The program scheme

```
  loop P1; exit if B1; P2 end
```

is equivalent to

```
  P1; while not B1 do begin P2; P1 end;
```

Case statement.

In the case statement case labels may be specified by subranges in usual notation. The final end in a case statement may be replaced by otherwise <statement> meaning that the statement following otherwise will be executed if none of the labelled statements is selected for execution.

TECHNISCHE HOGESCHOOL TWENTE

ONDERAFDELING DER TOEGEPASTE WISKUNDE

Mr. G.H. Richmond
University of Colorado
Boulder Colorado 80302
Computer Center

ONDERWERP: KENMERK: TW75/INF/302 ENSCHEDE, 11 augustus 1975

Dear Mr. Richmond,

Regarding the status of the PASCAL-implementation for the PDP11 series.

Date: 8 august 1975

Implementation Route: PASCAL-P1, Cross Compiler described in PASCAL to be run on PASCAL system for DEC-10.

Target Machine : PDP11 series, no O.S. requirements (all models).

Implementation Status: Testphase nearing completion. Available for Distribution by Dec. 1975.

Restrictions : except for standardfiles INPUT & OUPUT, files are not implemented. Jump out of procedure not implemented.

Extensions : formal/procedure/function specification required. Array-parameters with unspecified bounds are allowed. Functions may deliver results of any type.

Yours sincerely,
C. Bron
Drs. C. Bron.

ENSCHEDE - DRIENERLO - POSTBUS 217 - TELEFOON: 05420 - 99111 - TELEX 44200

BASSER DEPARTMENT OF COMPUTER SCIENCE
School of Physics (Building A28),
University of Sydney, N.S.W. 2006

15th August, 1975

Mr. G.H. Richmond,
University of Colorado,
Computing Centre,
3645 Marine St.,
Boulder, Colorado,
U.S.A. 80302

Dear Mr. Richmond,

In response to your circular of June 5, I am providing the following information.

As Dr. Sedgwick recently left the Department in order to take up a position in Toronto, the contact for the local Pascal-P2 implementation is myself. The status of the implementation is "progressing" with completion anticipated around the end of this year. The main hold-up has been the lack of documentation for the B1726's operating system.

Since the B1726 is user-microprogrammable and bit-addressable, the implementation strategy is basically that of microcoding the Pascal-P interpreter. (In fact, all languages on the B1726 are implemented in this manner.) As well, the compiler has been modified so that it supports the EBCDIC character set and generates "machine" instructions which support bit-addressable data items of arbitrary length. Our configuration parameters have been chosen as follows: 16777215, 25, 34, 8, 1, 72, 24, 16, 24. Please note that the unit of storage is obviously the bit, and that the setsize of 72 is arbitrary and will be extended to 256 eventually, to allow sets of chars. Another consequence, of bit-addressability is that the packed keyword becomes superfluous, thus eliminating one "restriction" in the present compiler.

It is also encouraging to observe that initial estimates of the compiler's size place it well below that of the Burroughs-supplied compilers (including Fortran, Cobol, RPG), with the exception of the Basic compiler.

Yours sincerely,

Antony J. Gerber

Antony J. Gerber

University of Florida
Gainesville, 32611

Intercollege Department of
COMPUTER AND INFORMATION SCIENCES
512 Weil Hall
904-392-2371

Degree Programs in the Colleges of
ARTS AND SCIENCES,
BUSINESS ADMINISTRATION
and ENGINEERING

August 18, 1975

Mr. George H. Richmond
University of Colorado
Computing Center
3645 Marine Street
Boulder, Colorado 80302

Dear Mr. Richmond:

As I mentioned in our telephone conversation last week, I have successfully transported the PASCAL 'P' compiler to the TI 980A, a 16-bit word minicomputer.

Machine requirements are:

1. Minimum of 36K of main memory (4K for operating system)
2. disk
3. Silent 700 console with dual cassettes .

The P-code for the compiler occupies 27,936 words (2 words/instruction) and the loader-interpreter occupies 3,744 words. Storage in main memory is dynamically allocated so the system can be run on any machine having at least 36K of main memory.

Since the P-code for the compiler is so large (approximately 17,000 records) it was physically split into three parts on a large computer and transmitted across telephone lines to cassettes. Then it was merged back into one file on the TI disk.

Pertinent configuration parameters are:

INTSIZE = 1	SETSIZE = 4
REALSIZE = 2	PTRSIZE = 1
CHARSIZE = 1	STRGLGTH = 6
BOOLSIZE = 1	INTBITS = 15

Sincerely yours,

Gilbert J. Hansen

Gilbert J. Hansen
Assistant Professor

The University of Tasmania

Postal Address: Box 252C, G.P.O., Hobart, Tasmania, Australia 7001

Telephone: 23 0561. Cables 'Tasuni' Telex: 58150 UNTAS

Information Science Department.

IN REPLY PLEASE QUOTE:

FILE NO. .....

IF TELEPHONING OR CALLING

ASK FOR .. ..

18th August, 1975.

Mr. G.H. Richmond,
Computing Center,
University of Colorado,
BOULDER, COLORADO.

Dear Mr. Richmond,

In response to your letter relating to communication between Pascal implementors, I can only heartily agree. I therefore give you the following points relating to our work on Pascal-P2.

Status: incomplete (1975 August)

Purpose of implementation: investigation of portability; use of Pascal in teaching.

Route: Piggybacking via PASCAL-1 on Cyber 72 to produce Burroughs B6700 'assembly code', thence 'assembly code' to be processed for route: This two-step process separates the easy part (generating B6700 code) from the hard part (getting the B6700 to accept it).

Impression of package: Far too little thought given to portability and to documentation. Pascal-P2 still betrays its heritage as a CDC-biassed product in subtle but annoying ways, though vastly better than earlier Pascals; it remains a test-bed product designed for a restricted purpose, and despite the claims made for it in the documentation, could be made vastly easier to port (whether bootstrapped or piggybacked).

Main implementation difficulties: Doing sensible things on a highly structured computer (without a linear von-Neumann memory) with multiple-word objects, particularly allocated in the heap. It seems a great pity to have to forego the many advantages of the B6700 architecture because of some of Pascal's features, and yet trying to use all the good features may well lead to excessive memory fragmentation and segmentation; also possibly complex code generation for different handling of constructs. For example should all *records* be individual *segments?* or should a simulated linear store (= a large declared vector) be used to pretend to be a more conventional machine?

Useful information for other implementors: A few sheets showing the (static) frequency of occurrence of each of the SC-machine instructions is available on request, also doublet frequencies. This shows which instructions may be ignored or simplified in interpreting or macro-expanding the code, and where most of the space goes (and very likely the time too). Since this route was discarded as unnecessarily difficult, no dynamic execution frequencies are available.

Also note that sets of 48 bits are sufficient to bootstrap up the Pascal-P compiler itself (the largest set has 48 elements). The statement in the documentation relating to 59 bit sets is simply a CDC hangover which has not been checked. This has significance to 48-bit machines (as B6700).

Likely completion date: November/December 1975.

Yours sincerely,

A.H.J. Sale

A.H.J. SALE,
Professor of Information Science.

UNIVERSITY OF CALIFORNIA, SAN DIEGO

DEPARTMENT OF APPLIED PHYSICS AND INFORMATION SCIENCE
COMPUTER SCIENCE DIVISION, C-014

LA JOLLA, CALIFORNIA 92093

August 22, 1975

Professor A.H.J. Sale
University of Tasmania
Box 252C, G.P.O.
Hobart,
Tasmania 7001

Dear Professor Sale:

We are indeed working with PASCAL on the B6700. Whether the work is of immediate interest to you is another question. Making PASCAL into a stable B6700 product for users is a secondary objective of our project. Our primary aim is to create an interactive student debugging environment on the PDP-11, with virtually all of the software written in PASCAL.

The overall objectives of the project are described in the enclosed project prospectus. Students will interact with PASCAL on the small machines in a manner very similar to the debugging environment of APL on IBM 360/370 systems. PASCAL is interpreted using a modified version of the Zurich P-Machine recently released. The main purpose of the modifications is to reduce the size of the compiled code so that the PASCAL compiler can fit within the limited core of a small machine. Yes we have done the same kinds of statistical studies represented in the reports you kindly sent, though our data is not in as elegant a form. We are confident that the compiler can be run on a PDP11 with at least 20K words of memory. We are hoping to reduce that amount further to perhaps 16K, when time permits. Currently the interpreter is operating, but has yet to be tried with the whole compiler on the PDP-11.

We are using the modified PASCAL compiler on the B6700 as a tool for developing the new PASCAL system, and generating pseudo-code for the PDP-11. Having started with an interpreter for the Zurich P-Machine, we have progressed through various stages of bootstrapping to get a system compatible with the PDP-11 objective, and the interactive system objective. Concurrent with the work using the interpreter, we also have an advanced student programmer writing an assembler which converts the compiler P-code output into directly executable B6700 code. The B6700 code has been executed with small programs, and should be running the whole expanded compiler within a week or so. This compile-assemble system manages its memory in one large array in a fashion similar to that used on conventional machines. We are using the B6700 SWAPPER for much of our batch work, and hence have been able to use DIRECT (non-overlayable) array space for this purpose to enhance the speed of the processing.

The short-term objective for the B6700 compile-assemble system is to provide a back-up means for students to use for PASCAL homework problems starting in late September. Our PDP-11 equipment is not all here yet, and we clearly will not be ready to use the small machines with students during the first few weeks of the Fall Quarter. Over the time period of the academic year about to start, we will almost certainly have someone complete the job of making a PASCAL compiler that can generate B6700 code directly. Yet to be resolved is the question of whether we can map the PASCAL data structures into the array-row structures of the B6700 without doing violence to the basic approach of the P-compiler.

The interpretive system is slow on the B6700, as might be expected. The major consumer of time is the low level character processing in the INSYMBOL and NEXTCH procedures. We have changed these procedures completely, so as to depend upon installation intrinsic functions (Standard Procedures) that make use of the B6700 string processing hardware. The GETSYM intrinsic returns information on each successive token in an area of stack that serves as a scanner information block. This provides a clean interface between compiler and interpreter, but it runs about half as fast as an earlier less-clean version (part way through the bootstrapping) in which virtually all of the work of INSYMBOL was done in an ALGOL intrinsic. The current B6700 interpretive version takes about 10 minutes of processor time to compile the source file from Zurich. We expect the compile-assemble version, and also the PDP-11 version, to run roughly five times faster than that.

During the next two months we will be up to our ears in alligators getting this system completed well enough to use for teaching. At a later stage, I would be happy to share more details with you.

Sincerely,

Ken Bowles

Kenneth L. Bowles
Professor (Computer Science)

The University of Tasmania

Postal Address: Box 252C, G.P.O., Hobart, Tasmania 7001

Telephone: 23 0561. Cables 'Tasuni' Telex: 58150

IN REPLY PLEASE QUOTE:

FILE NO.

IF TELEPHONING OR CALLING

ASK FOR

Information Science Dept.

9th September, 1975.

Mr. Richmond,
Computing Centre,
University of Colorado,
3645 Marine St.,
BOULDER, COLO.

Dear Mr. Richmond,

Pascal-P Documentation

I have received, in response to some of my correspondence, a copy of some error notes on Pascal-P detected by the University of Karlsruhe. The documentation is in German, and I have attempted to translate the sense of the notes with the results attached. I hope that this may be of use to other Pascal-P implementors. I have asked the originators to see if my translation accords with what they thought they said in case I have missed some idiomatic nuance (technical German terms are quite as mystifying as English ones until decoded; witness "bugs") and I shall let you know if there are any alterations or additions.

Yours sincerely,

A.H.J. SALE,
Professor of Information Science.

Encl.

LIST OF PASCAL DEFECTS

The following errors and defects were found during the implementation of the PASCAL system.

1. Defects in the compiler which adversely affect the bootstrapping to different computers.

   (a) Assumptions are made about the collating sequence of the character set which are neither warranted nor defined in the axioms concerning character type. In particular:

   - The translator assumes that the characters '+' and ';' enclose all operators in the declarations of

     ```
     SSY, SOP:  array ['+;..';']
     ```

   - The test which determines whether a character is alphanumeric is formulated as

     ```
     ORD(CH)  >= ORD('A') AND ORD(CH)  <= ORD('9')
     ```

   (b) The compiler accumulates errors only during the translation of a line of source text. At the end of the translation the compiler cannot determine automatically whether the generated code is correct, or whether the PASCAL program was in error or not.

   (c) The constant CHARSIZE, which is used to parameterize the compiler for various computers, is employed with the meaning "Storage units per character" and not "characters per storage unit". While this is clearly set out in the documentation it is unexpected.

2. Deliberate restrictions imposed by the compiler.

   (a) Only the first 8 characters of identifiers are significant in distinguishing them; remaining characters are ignored.

   (b) String constants are limited to a maximum of 16 characters.

3. Errors in the compiler.

   (a) When translating a PASCAL program which does not have the terminating symbol *END.* the compiler hangs in an infinite loop. The error is in procedure NEXTCH. It has the structure:

```
PROCEDURE NEXTCH;
BEGIN
   IF EOL THEN BEGIN ... END;
   IF NOT EOF(INPUT) THEN
   BEGIN ..."read next symbol and assign value to global"
   END ELSE WRITE(OUTPUT,'EOF ENCOUNTERED')
END;
```

In this case NEXTCH is called by SKIP (via INSYMBOL) until the terminating symbol is read. If NEXTCH in the compiler is not altered, the compiler loops endlessly. An error exit in NEXTCH would solve the problem but this is not permitted in PASCAL-P. (The error was probably not noticed in the CDC-implementation as multiple calls to EOF with the value TRUE cause a program-dump in that system. This meaning of EOF is not endorsed in the PASCAL definition and is not obvious.)

(b) When in the procedure INSYMBOL a symbol is read which does not belong to the recognised symbols, an error is signalled but the next symbol is not read in. When SKIP initiates the consequent search to recover from the error, the compiler finds itself in a loop, as subsequent calls continue to find the undefined symbol.

(c) The variable EOL should be initialised to the value FALSE (not TRUE), since otherwise the first action of the compiler is to issue a read command, and the counting of source lines starts at two.

4. Aesthetic defects in the compiler.

(a) To comply with PASCAL-P, the LOOP-EXIT construction has had to be deformed into a REPEAT-UNTIL construction. The change was made with too narrow a view, with the result that the constructions are much more difficult to understand. The loop construction, which is to be found in most procedures, has for example the following form (taken from INSYMBOL):

```
REPEAT
    WHILE CH = ' ' AND NOT EOL DO NEXTCH;
    TEST:=EOL;
    IF TEST THEN NEXTCH
UNTIL NOT TEST;
```

(b) There is an error message which is emitted not only by error number 117, but also with the message *'TYPE-ID' <identifier>*. This message does not match the source line and the following error-messages (and their lines) in the procedure ENDOFLINE, but seems to be issued immediately when the error is recognised in the syntactic and semantic analysis. The cause has not been found despite an intensive search. Thereafter the layout of the program listing, error numbers, and error text becomes unrecognisable.

5. Properties of the compiler which affect interpretation.

(a) The compiler produces the same commands as for all other objects for packed arrays of characters (strings). Also at the level of the interpreter strings must be treated as arrays of single characters.

(b) Parameterising the compiler for the storesize of sets (SETSIZE=2) does not result in efficient store utilisation in the interpreter. The formal parameters of procedures are set up with the maximum size of objects of type INTEGER, REAL, CHAR and SET. Similarly all load and store commands ignore the type of the data-types and are not parameterised. As a consequence If the stack is not to be full of objects all of the largest size, the load and store instructions have to inspect type-tages in the interpretive store.

Keeping type-tags (as implied by a direct interpreter implementation) is incredibly wasteful of store, and isufficient thought has been given to the problems of wordsize in claiming portability.

(c) The convention of the interpreter is not followed when code is generated for formatted output of strings. For example:

WRITE(OUTPUT,'ABC':10) is compiled to:

```
LCA 'ABC'
LDCI  10
LDCI   3
LAO    5
CSP  WRC
```

The interpreter expects the code:

```
LCA 'ABC'
LDCI   3
LDCI  10
LAO    5
CSP  WRC
```

NOTE: Translated from the original GERMAN supplied by the University of Karlsruhe (1975 August 21), with free translation.

1975 September 1
A.H.J. Sale,
Department of Information Science,
University of Tasmania,
G.P.O. Box 252C,
HOBART, TASMANIA. 7001

Department of Electrical Engineering
University of Colorado.

1 September 1975

Today is our projected release date for PASCAL J, a compiler which translates PASCAL [Jensen 1975] into the intermediate language Janus [Waite 1975]. The compiler has successfully translated a number of programs, including itself. All translations have been carried out on the Control Data 6400; the programs (with the exception of the compiler) have then been executed correctly on the General Precision L3055 and/or the XDS Sigma 3. As of today, we have been unable to complete a bootstrap of the compiler onto either target machine because of size constraints and hardware failures.

At the moment, we feel that our implementation aids (test programs, step-by-step procedures) for Janus are inadequate. Therefore our estimate of implementation effort required by those unfamiliar with Janus is 3-6 man-months. We hope to correct this deficiency in the near future.

Our estimate of the quality of the current object code resulting from a translation of PASCAL to Janus to assembly language shows it to be larger than "best hand code" by a factor of 3. We are working on improvements in the compiler which we expect will reduce this factor considerably.

A good deal of interest has been shown in this project, and we therefore feel that we should adhere to our announced release date. On the other hand, in view of the problems stated above, we must warn prospective users that the present version may not be all that they desire. We are willing to distribute copies of the compiler and currently-available test programs with the understanding that they are not to be released by the recipients without a copy of this letter. Any comments or suggestions which could improve further releases should be sent directly to us. Please be specific in your suggestions.

The attached price list specifies the material available in this release. Please make all checks payable to the Department of Electrical Engineering. Our next release is projected for 1 February 1976. Anyone who orders material now will automatically be informed of the project's status at that time. If you would prefer to wait until then to find out what is available, please let us know.

- Software Engineering Group

References

Jensen, K., Wirth, N. PASCAL User Manual and Report. Lecture Notes in Computer Science, 18 (1974).

Waite W.M., Haddon, B.K. A Preliminary Definition of Janus. Tech. Rept. SEG-75-1, Dept. of Electrical Engineering, University of Colorado, 1975.

A. IDENTIFICATION

Program name: PASCALJ

Authors: B.W. Ravenel, C.B. Mason

Date: 1 September 1975

B. GENERAL DESCRIPTION

PASCALJ is a compiler which translates the high-level language PASCAL to the intermediate language Janus. This compiler was originally written in PASCAL, and used to translate itself to Janus. The original PASCAL program is included in the Janus text as comments, and may be extracted by selecting only those lines which have a period in the first character position. All other lines should be ignored during extraction, and the periods in position 1 should be replaced by spaces.

PASCAL is defined in the following book, which is available from its publisher (Springer-Verlag):

Jensen, K., Wirth, N. PASCAL User Manual and Report. Lecture Notes in Computer Science, 18 (1974).

It is necessary to indicate the character set of the host computer on which PASCALJ is to be implemented, as the compiler must provide a mapping from this character set to its own internal representations. We are providing several standard character sets from which one should be chosen; should some other character set be required please send a detailed collating sequence for it.

Character sets available:

ASCII (full set, 96- or 64-character subset)
EBCDIC
CDC display code

As an option, the compiler can produce Janus code in which the Janus stack of anonomous operands is restricted to a depth of one through the use of named temporaries. This may be useful for implementations on single accumulator machines. Indicate with your order whether the compiled version of the compiler should be so restricted or not.

C. DOCUMENTATION

SEG-75-1	A Preliminary Definition of Janus	5	oz	$ 3.50
SEG-75-3	PASCALJ User's Guide and Implementation Notes	2	oz	$ 2.00

D. TEXT

7-track magnetic tape (1200-foot reel)	1.75	lb	$25.50
9-track magnetic tape (1200-foot reel)	1.75	lb	$35.00

(Deduct $8.00 from tape cost if you supply a 1200-foot reel. We will accept longer reels, but we must charge more for postage.)

PASCAL - USER MANUAL AND REPORT

Corrections to 2nd Edition

p.51, l.16:	"setop(output)" → "setop(output);"
p.56, l.-6:	"fi" → "f(i)" "g(i+1)" → "g(j+1)" "gi" → g(j)
p.63, Fig.10a:	Number sequence should be reversed.
p.69, l.23:	"stricly" → "strictly"
p.77, l.18:	move line 3 places to the left
p.98, l.10:	append " ';"
p.102, l.7:	last word should be "or"
l.20:	"bufffer" → "buffer"
p.103, l.-6:	"scaler" → "scalar"
l.-7:	"char, and alfa" → "and char are listed"
p.124, l.14 and l.15:	"14" → "15"
p.127, l.27:	"18.A" → "4.A"
p.133, l.3:	"two" → "to"
p.135, l.5:	"althought" → "although"
l.30:	"subtstitute" → "substitute"
p.140, l.11:	"structure type" → "structured type"
p.158,	delete lines -12... -8.

N. Wirth
10.May 1975

Additional Suggested Corrections

September 26, 1975 University Computer Center
University of Minnesota

Page/Line	
13/-3	"if" → "If"
69/-8	"the readability" → "readability"
72/-3	"element in the array" → "component in the structure"
81/2	"extent" → "extend"
117/	in the syntax chart for expression, change: ≠ ≤ ≥ to <> <= >=
119/13	move the message right by 1 position
126/-19 to -14	move the "i" index entries to the next page
153/16	"(and at least once)" → "(at least once)"
162/3	"end of line" → "end of line"

THE COLLEGE OF WOOSTER
COMPUTER CENTER
WOOSTER, OHIO 44691

October 3, 1975

Mr. George H. Richmond
Computing Center
University of Colorado
3645 Marine Street
Boulder, Colorado 80302

Dear George:

The reply to complaints about interactive I/O in PASCAL (Newsletter, Number 3, November 1975, p. 13) is not satisfactory. READ, READLN, WRITE, and WRITELN should all function as easily (although possibly not exactly the same) for an interactive user as they do in the batch environment. The solution suggested is not appropriate to beginning students. Moreover, "segmented" files are peculiar to CDC systems. I'd like to see some other solutions to the problem.

Sincerely,

Carl

E. C. Zimmerman
Director of Computer Services

CALIFORNIA INSTITUTE OF TECHNOLOGY

PASADENA, CALIFORNIA 91125

INFORMATION SCIENCE 286-80

October 22, 1975

Mr. George H. Richmond
Computing Center
University of Colorado
Boulder, Colorado 80102

Dear Mr. Richmond,

I enclose an announcement of the distribution of two compilers for Sequential and Concurrent Pascal. They have been running on a PDP 11/45 computer at Caltech since January, 1975. This announcement may be of interest to readers of your excellent Pascal Newsletter.

Yours sincerely,

Per Brinch Hansen

PBH:mh

Enclosure

SEQUENTIAL AND CONCURRENT PASCAL FOR THE PDP 11/45

Multi-pass compilers for Sequential and Concurrent Pascal (written in Pascal) have been running on a PDP 11/45 computer at Caltech since January 1975. Concurrent Pascal is a programming language that extends Sequential Pascal with classes, monitors, and processes (see: P. Brinch Hansen, The programming language Concurrent Pascal. IEEE Transactions on Software Engineering 1, 2, June 1975). The compilers generate virtual machine instructions simulated by threaded code.

A single-user operating system written in Concurrent Pascal has been in use at Caltech since May 1975. The Solo system supports editing, compilation, execution, and storage of Sequential and Concurrent Pascal programs. These programs can call one another recursively with arbitrary parameters. Input, processing, and output of files on console, cards, printer, magnetic tape, and disk are handled by concurrent processes.

Solo is the first working operating system written exclusively in a high-level programming language that includes concurrent processes and abstract data types. System protection is achieved largely by means of compile-time checking of access rights. It is not supported by hardware mechanisms at run-time.

Manuals and distribution tapes for the Solo system and the Pascal compilers are now available from Caltech. For more information on how to obtain copies, please write to

Per Brinch Hansen

Information Science 286-80
California Institute of Technology
Pasadena, California 91125

CERN

DD-HJ-pp 28-10-1975

Memorandum

To : Anybody interested

From : Hans Jøraandstad, DD

Subject : PASCAL

An initial version of PASCAL is available under SCOPE 2.1.2 on the CDC 7600 computer system. The compiler should be 100% compatible with the PASCAL compiler currently available under SCOPE 3.4 (CDC6000), as seen from a users point of view. However, the I/O system under SCOPE 2.1.2 is somehow different from SCOPE 3.4; no CIO exists, hence Record Manager is used.

The project was initialised because it was thought to be a relatively small amount of work. This was based on the following compatibilities between the two operating systems:

1. Instruction repertoire and word length the same
2. Loader relocatable binary files created on CDC 6000 can be loaded on the 7600.

In fact, the design (of new I/O system), coding and testing (not yet completely finished) was accomplished in ∿.5 weeks full work (1 man).

Briefly, the following changes were necessary:

1) Compiler:
   a) Redefinition of EFETSZ and CHEFETSZ
   b) Generation of RJ P.TERM instead of "end request".
   c) Insertion of a small loop at the end of the compiler which will ensure reading of the input file until EOF condition true (for compatibility with all other SCOPE 2.1.2 utilities).
2) I/O System:

   Rewriting, changes and additions amounting to ∿ 1000 cards of assembler code.

A subset of the record manager record types under SCOPE 2.1.2 are supported according to the following table:

	W (default)	S	Z	U (unknown)
Text files				
EOLN	Record exhausted	12 bits zero byte encountered	As for W	U files not allowed for text files
READLN	Get next record	Skip until 12 bits zero byte	As for W	
WRITELN	Put record, blank filled 12 zero bits at end	Put partial, blank filled 12 zero bits at end	As for W	
Other files				
EOS	EOS/EOF	End of S-record	EOS/EOF	Tapemark or EOF
EOF	EOS/EOF	End of S-regord or EOF	EOS/EOF	Tapemark or EOF
GETSEG	Skip EOS marks	Skip records	As for W	Skip tapemarks
PUTSEG	Write EOS	Full put = end of S-record	As for W	Write tapemark

UNIVERSITY OF CAMBRIDGE COMPUTING SERVICE

Pascal Interpretive System (Cambridge)

The enclosed document gives details of this system, which is now available for distribution.

If you would like a copy of the system, you should write to:-

Dr. J. Larmouth (Pascal)
University of Cambridge Computer Laboratory
Corn Exchange Street
CAMBRIDGE CB2 3QG
England Tel: Cambridge 52435

stating whether you want the simple system (first two files described on the attached sheets) or the full system. You should also state any preferences for magnetic tape density, tracks, or format, and may send us a tape if you wish.

You should be prepared to pay a handling charge of £5.00, plus any cost of tapes, postage, etc. We will send you an invoice, and you should state any Order Number you wish us to quote, and the details of who to bill. The total charge is unlikely to exceed £10.00.

You should also complete the attached Distribution forms, which allow for software to be sent either to a named person for use anywhere, or to an installation. Section 4 on the forms should be deleted as appropriate. The two lines following Section 5 should be deleted in a corresponding way, and either an individuals name inserted, or the title of the installation. The forms should be signed at the end by an appropriate authority, or by the individual. We will complete the forms and return a copy with the software.

Also enclosed is an incomplete copy of a letter which we send to "Commercial" installations. This may well not be relevant to you.

If you have any further queries, please do not hesitate to write and ask.

J. Larmouth

Encs: Distribution Forms
Honorarium Form
Listing

SUMMARY OF THE SYSTEM.

THE COMPILER PRODUCING 'STACK COMPUTER' CODE WAS WRITTEN BY KESAV NORI AT ETH ZURICH. THE ASSEMBLER INTERPRETER WAS WRITTEN BY DR. J. LARMOUTH AT THE UNIVERSITY OF CAMBRIDGE COMPUTER LABORATORY, ENGLAND. THE I/O PACKAGE WAS WRITTEN BY DR. P. HAZEL, L.B.MATTHIESON, AND B.LANDY AT CAMBRIDGE. THE TRIG ROUTINES AND THE 'WRITE REAL' ROUTINE WERE WRITTEN BY J. GLUZA AT CAMBRIDGE.

THE COMPILER IS WRITTEN IN PASCAL, AND HAS BEEN WIDELY DESCRIBED. THE REST OF THE SYSTEM IS WRITTEN IN 360/370 ASSEMBLER. ALL IBM FILE FORMATS ARE HANDLED FOR BOTH INPUT AND OUTPUT.

THERE ARE A NUMBER OF DEVELOPMENTS NEEDED ON THE SYSTEM, (E.G. TRAPPING BY THE USER AND THE SYSTEM, MULTIPLE INPUTS AND OUTPUTS) AND THESE MAY HAPPEN IN LATER VERSIONS. HOWEVER, BEFORE HOW IS GOING INTO A COMPILER SYSTEM WHICH WILL GENERATE 370 CODE DIRECTLY, BASED ON A PORTABLE FRONT-END AND THE 'HANDSHAKE' CONCEPT, AND FURTHER DEVELOPMENT OF THE INTERPRETER IS UNLIKELY IN THE NEAR FUTURE.

THE COMPILER MODULE COMPILES A NULL PROGRAM IN ABOUT 90K, A LARGE ONE IN ABOUT 120K, AT ABOUT 30 LINES A SECOND ON A 370/165. THE RUN MODULE WILL RUN A SMALL PROGRAM IN ABOUT 40K.

A CROSS-COMPILER IS AVAILABLE WHICH ALLOWS CODE TO BE GENERATED FOR MACHINES WITH A CHARACTER SET CODING DIFFERENT FROM THAT USED AT CAMBRIDGE. THE FOLLOWING NOTES DESCRIBE THE MAIN PARAMETERS AND FEATURES OF THE SYSTEM YOU HAVE BEEN SENT. A 'SIMPLE SYSTEM' TAILORED TO YOUR CHARACTER SET IS THE FIRST TWO FILES ON THE TAPE, AND SOME USERS MAY REQUIRE NO MORE THAN THIS. THIS IS TWO COMPLETE ASSEMBLER PROGRAMS. ALTERNATIVELY THE FULL SYSTEM CAN BE SUPPLIED.

THE SYSTEM MAY BE ACQUIRED BY SENDING DETAILS OF THE CHARACTER CODE OF YOUR INSTALLATION FOR THE PASCAL CHARACTER SET BELOW. (THE ONLY CHARACTERS WHICH USUALLY GIVE TROUBLE ON 370S ARE [ ] AND ↑).

YOU SHOULD ALSO SEND DETAILS OF THE TAPE DENSITY, LABEL, ETC THAT YOU WANT. AT PRESENT THE I/O PACKAGE CONTAINS 370 DEPENDENCIES, BUT THIS WILL BE CHANGED.

YOU WILL BE ASKED TO SIGN AN AGREEMENT NOT TO MISUSE OR DISPOSE OF THE SYSTEM, AND TO AGREE TO BEING INVOICED FOR THE COST OF HANDLING AND A TAPE (UNLESS YOU SEND US ONE). THE INVOICE SHOULD NOT EXCEED ABOUT TEN POUNDS.

UNIVERSITY OF WISCONSIN - EAU CLAIRE / EAU CLAIRE WISCONSIN 54701

DEPARTMENT OF COMPUTER SCIENCE

November 5, 1975

Mr. George H. Richmond
University of Colorado
Computing Center
3645 Marine Street
Boulder, Colorado 80302

Dear George,

I believe the following information would be of interest to readers of the PASCAL newsletter, especially those contemplating the implementation of a PASCAL system.

A PASCAL P-code interpreter has been designed and implemented in compatible ALGOL on a Burroughs B5700 at the University of Wisconsin Eau Claire. Proof that the major design goal of the PASCAL-P project (easy portability) was indeed achieved is provided by the fact that an interpretive PASCAL-P system was up and running on the B5700 in only 4 student-weeks. The undergraduate student writing the P-code interpreter in ALGOL had seen only one PASCAL program prior to this project (the POSTFIX program on page 75 of the PASCAL User Manual) and had taken one 3 semester-hour course in Burroughs compatible ALGOL. The implementation of the P-code interpreter in compatible ALGOL was rather straightforward except for three areas- (1) character set representations, (2) the basic PASCAL I/O operations on text files, and (3) the basic PASCAL set operations. I suspect that other implementers of the PASCAL-P system have encountered difficulties in these same areas.

The (ALGOL) P-code interpreter and the (P-code) PASCAL compiler were tested by compiling each of the sample PASCAL programs appearing in the PASCAL User Manual. The results of these 29 runs are summarized below.

20 programs compiled and executed without error.

3 programs compiled with expected errors.

7 programs compiled with unexpected errors.

(One program, "FCOUNT", had both an expected and an unexpected error

The documentation supplied with the PASCAL-P implementation package warned that every standard procedure or function involving files must specify the file concerned: the usual default interpretations do not hold. This difference between "standard" PASCAL (defined, presumably, by the PASCAL User Manual and Report) and PASCAL-P was verified in the compilation of "FCOUNT" (p. 60), "INSERT" (p. 61) and "MINMAX3" (p. 70). Each of these programs omits the file specifications required by PASCAL-P. These were the three runs which terminated with expected errors - although the missing file specification in "MINMAX3" caused the compiler to go into an infinite loop.

However, there are additional differences between standard PASCAL and PASCAL-P. These differences, unexpected since the documentation supplied with the PASCAL-P implementation package does not mention them, were revealed in the compilation of the following seven programs: "GRAPH1" (p. 30), "GRAPH2" (p. 38), "CONVERT" (p. 17), "SETOP" (p. 51), "BISECT" (p. 79), "PRIMES" (p. 54) and "FCOUNT" (p. 60).

The compilations of "GRAPH1", "GRAPH2" and "CONVERT" revealed that the built-in function "ROUND" is not a part of the PASCAL-P system. After adding a user-defined function called "ROUND" to each of these three programs, they compiled and executed correctly.

The compilations of "SETOP", "PRIMES", and "FCOUNT" revealed a more serious problem with the PASCAL-P system. PASCAL-P does not recognize as a legal <factor>the <set> alternative

[<expression> .. <expression>]

See Appendix D of the PASCAL User Manual. Program "SETOP" was modified to compile and execute correctly by changing the assignment statement WK:=[M..SU] to WK:=[M,T,W,TH,FR,SA,SU]. Similar changes to "PRIMES" and "FCOUNT" would no doubt make them acceptable to the PASCAL-P system.

The compilation of "PRIMES" also revealed that the PASCAL-P system does not support the intrinsic functions "SUCC" and "PRED". This is a serious omission and an effort is under way here at University of Wisconsin-Eau Claire to add them to the PASCAL-P system.

The compilation of "BISECT" revealed that the PASCAL-P system does not support formal parameters of type function. (In this same vein, an attempt to compile the "READINTEGER" procedure on page 160 of the PASCAL User Manual, revealed that files are not acceptable as formal parameters either.)

Additional tests are currently in progress to determine if there are still other differences between standard PASCAL and PASCAL-P.

The experience gained in compiling the sample programs from the PASCAL User Manual also revealed the following facts about the PASCAL-P system.

(1) Errors in PASCAL source programs are flagged by number with no message attached. Presumably the error numbers generated by the PASCAL-P compiler correspond to the error number summary appearing as Appendix E of the PASCAL User Manual. However, it is impossible for me to say with any degree of certainty that such is the case.

(2) Keypunch errors in PASCAL source programs which generate references to undeclared identifiers will cause the compile phase of a PASCAL-P job to terminate with an "invalid index" or "subscript out of bounds" message if you are lucky enough to be running the P-code interpreter on a system which checks for such errors (such as the Burroughs B5700). I hesitate to predict the results if the interpreter is running on a system without such checks.

(3) There is no compile-time or execution-time range checking done by the PASCAL-P system.

These last three items force me to the conclusion that the PASCAL-P system as it now exists is not suitable for use in a student environment. My hope is that the necessary modifications and improvements can be made to the PASCAL-P system so that in the near future the benefits of programming in PASCAL will be available to students on a wide scale.

Sincerely,

Bruce A. Pumplin

Bruce A. Pumplin
Assistant Professor of Computer Science

Electro Scientific Industries, Inc.
13900 N.W. SCIENCE PARK DRIVE
PORTLAND, OREGON 97229
(area code 503) 646-4141
TELEX No. 360273

George H. Richmond
Univ. of Colorado

Nov. 17, 1975

Dear Mr. Richmond:

Enclosed is a copy of a paper to be given at the coming DECUS (Digital Equipment Corp. User's Society) meeting in Los Angeles, Dec. 5. It outlines what we have been doing for the past year.

Pascal really has been a boon to us, and we are especially pleased that our adaptation to the PDP-11 for hardware control has worked out so well.

David Rowland

David Rowland
manager, programming

PASCAL FOR SYSTEMS

David Rowland
Electro Scientific Industries, Inc.
Portland, Oregon

ABSTRACT

Electro Scientific Industries has implemented a Pascal compiler for the PDP-11. An extension to the language allows the direct control of external hardware at the Pascal level. Experience indicates that Pascal can be a very good real-time process control language and that the use of a single, high-level language is a great aid to a programmer's efficiency.

I Electro Scientific Industries (ESI) builds computer-controlled laser trimming systems for the productions of hybrid thick and thin film circuits. The computer (a PDP-11/04) controls all details of the trimming process: selection of measurement probes, position and motion of the laser beam, power of the laser, and action of the digital voltmeter which makes the measurement. It handles a variety of interrupts and may have to do much data manipulation.

The software for the system must cover all these activities, be flexible enough to satisfy unpredictable future uses and be easy to understand and modify.

II Pascal is a general purpose programming language proposed in 1971 by Niklaus Wirth, now at the Federal Institute of Technology in Zurich.[1,2] It was named to honor Blaise Pascal (1623-1662), physicist, mathematician and philosopher. He was also the first person to build and offer for sale a calculating machine.[3]

The language Pascal was proposed because Prof. Wirth wanted a simple, modern language with which he could teach programming concepts without becoming tangled in details of the language. Existing languages were inadequate because they were too complex (PL/I), not convenient for teaching general concepts (Cobol), outdated and full of idiosyncracies (Fortran), or too simple and lacking in expressive power (Basic).

Pascal has these virtues:

1. It has a clean, modern syntax, allowing easy parsing and one-pass compilation. Variable names may be any length. Statements are not bound to one line or limited one-to-a-line.

2. The statements encourage the application of structured programming ideas, leading to programs that are easy to read and modify.

3. The basic data types (integer, real, boolean, character) and data structures (array, record, set, file) may be combined and nested almost indefinitely to yield data structures that accurately model the problem.

4. All procedures and functions are recursive.

III As we looked at existing languages, Pascal seemed ideal, but it had not been designed to handle interrupts or control hardware. On the other hand, no language of like power had either. We decided to try to adapt Pascal for our own needs.

To control hardware, we added two things to the language; octal constants and "ORIGIN".

An octal constant is simply an integer written in base 8 and ending with "B", e.g. 16B = 14. It is convenient in dealing with bit information and core addresses.

"ORIGIN" is a keyword used in variable declarations. It allows the programmer to fix a variable at a chosen core location rather than letting the compiler choose the location.

```
VAR PSW ORIGIN 177776B:INTEGER;
```

allows one, without descending to assembler code, to read and alter the program status word. For example:

```
PSW:=240B;
```

Less dangerously,

```
VAR TABSTATUS ORIGIN 164016B:INTEGER;
```

refers to part of our hardware and allows us to sample the status of the laser beam positioning system, again without using any assembler code.

```
IF (TABSTATUS AND 100B=0) THEN
                /* etc.*/
```

"ORIGIN" is a machine-dependent idea, and it violates the spirit of Pascal. However, it seems like the shortest and most natural bridge between the language and our hardware.

IV Our compiler accepts in-line assembler code, and we use it to set up the interrupt vectors. The interrupt handling routines, however, are written in Pascal, which holds the assembler code down to about a dozen lines.

⋮

V Our implementation of Pascal is written in Macro-11 assembler language. It is based on a limited compiler written at the University of Illinois. We changed the support from DOS to RT-11 and expanded the compiler to handle full Pascal. We were also able to shrink it drastically. More than 90% of the present compiler is new code.

⋮

Execution times appear very good. Informal benchmarks indicate that Pascal is about 4 times as fast as DEC RT-11 Fortran IV in a compute-bound sorting job.

VI Our customers receive the Pascal compiler plus a file of Pascal procedures which we have written and tested. They handle system interrupts, select the measurement probes, move the laser beam, control the measurement subsystem, etc.

⋮

VII It took about 14 man-months to bring the compiler to its present state. Another 4 were spent to create the support package and 4-6 on the basic control routines for the trimming hardware.

⋮

X Our Pascal does have weaknesses, some curable, some not. I have already mentioned the inconvenience of using Macro-11 as part of the compilation. We shall modify the compiler to produce binary directly, but it is tempting to write a new compiler in Pascal and use the existing one to bootstrap it.

There is also the penalty of less-than-optimum code. This is partly made up for by the ability to produce well-structured programs, thus gaining in overall efficiency.

The support package is a burden every Pascal program must carry. It could be reduced for special cases by eliminating the unused routines. Better would be to build it as a library and link only the routines needed.

Teaching Pascal is occasionally troublesome. Most of our customers are engineers, not programmers, and they might be more comfortable with something that looked like Basic. Pascal's recursively defined syntax and its data structures appear to be the most difficult concepts. However, for any program longer than about 25 lines, Basic's superficial attraction fades. Engineers are trained to solve problems and document their solutions clearly. This is quite in harmony with the top-down design encouraged by Pascal. Pascal's virtues start to become evident after about 3 days of training.

⋮

XI There does seem to be an ecology of programming languages. They grow and decline, adapt and produce descendants. What is to be Pascal's niche? Since it was designed as a teaching tool its obvious place is in schools and its obvious competitor is Basic. Pascal deserves to replace Basic for countless reasons, but inertia will probably prevent that. Pascal is catching on at many universities and Basic may find itself pushed into the fen country of grade schools and engineering offices.

Our real-time Pascal lives in a world where standards scarcely exist. Each OEM expects to have to create his own language to match his own hardware. As OEMs learn the virtues of high-level programming, Pascal may become a model, but we are probably fated to see as many varieties as there are manufacturers.

XII The work outlined here was done by a talented and spirited staff: John Ankcorn, Donald Baccus, Bruce Johnson and James O'Brien.

The partial compiler that was the basis of our work came from Prof. Donald Gillies at the University of Illinois. We were startled to hear of his untimely death this summer and we are sorry it is now not possible to thank him for the great assistance he gave us.

THE HEBREW UNIVERSITY OF JERUSALEM
Computation Center

27th November, 1975

Mr. G.H. Richmond
University of Colorado
Computing Center
3645 Marine Street
Boulder, Colorado 80302
U.S.A.

Dear Mr. Richmond,

In PASCAL Newsletter No. 3 (February 1975) a sample program illustrating interactive operation with PASCAL 2 appeared. We tried to run it here but found that this program fails to run unless the RESET(TTYIN) and REWRITE(TTYOUT) are replaced by GETSEG(TTYIN) and REWRITE(TTYOUT,1) respectively, to avoid the rewind associated with the former.

The corrected method was used successfully in interactive programs written here, with no need to change any supporting routine.

Sincerely yours,

Shmuel Peleg

4 December 1976

Mr. Alan Kortesoja said in a phone conversation that he had worked on the Comshare, Inc. Pascal system. It is a threaded interpreter and runs on a proprietary XEROX Sigma (?) operating system. Anyone interested in further details can contact him at:

Manufacturing Data Systems
320 N. Maine
Ann Arbor, Michigan 48104
(313) 761-7750

Budapest 12.12.1975.

Dear Mr. Richmond

Last year I had a study-tour in Switzerland, at EPF Lausanne in order to develop a PASCAL Compiler for CDC 3300 computer. This work was based on a PASCAL-P2 Compiler.

In Hungary a team /Cs. Lehel, L. Almási and J. Lehel/ finished the first step of this work. Now we have got a PASCAL Compiler - the first one in Hungary - for processing programs written in a subset of standard PASCAL. This subset does not process the type real and it contains some restrictions of PASCAL-P2.

The second step of this work is to develop a compiler of standard PASCAL.

Yours faithfully

László Almási and
Jenő Lehel
Computing and Automation
Institute of Hungarian
Academy of Sciences
H-1014 Budapest, Uri u. 49.
HUNGARY

Csaba Lehel

Technical University of
Budapest, Departement
of Process Control
H-1111 Budapest, Müegyetem
rkp.9.
HUNGARY

StonyBrook

State University of New York
at Stony Brook
Stony Brook, New York 11794

Department of Computer Science
telephone: (516) 246-7146

December 19, 1975

Mr. George Richmond
Department of Computer Science
University of Colorado
Boulder, Colorado 80302

Dear Mr. Richmond:

Enclosed is a notice for readers of the PASCAL Newsletter, announcing the availability of a new PASCAL compiler for IBM 360 and 370 computers.

Sincerely,

Richard B. Kieburtz
Professor

Announcing a New PASCAL Compiler for IBM 360 and 370 Computers

A new compiler fully implementing the programming language PASCAL, as defined by Niklaus Wirth has been designed and built at SUNY-Stony Brook. The new compiler features extensive compile-time and execution-time diagnostics, but also generates efficient, reloadable code. It should be suitable both for student use and for compilation of applications programs.

Distribution tapes will be available about March 15, 1976. The designers plan to provide maintenance updates for a period of one year following the initial release. A nominal charge will be made to cover costs of distribution, documentation, and updates. For additional information, please address inquiries to:

PASCAL Compiler Project
Department of Computer Science
SUNY at Stony Brook
Stony Brook, New York 11794

30 December 1975

A student of Dr. Charles Fischer at the University of Wisconsin in Madison related during a visit that Dr. Fischer is working on a Pascal compiler for the Univac 1110. The initial release should be in the Fall of 1976.

COMPUTATION CENTRE
POLISH ACADEMY OF SCIENCES
P.O.Box 22
00-901 WARSAW PKIN

Mr.George H.Richmond
University of Colorado
Computing Centre,3645 Marine St.

Dear Sir,

Our Computation Centre has an access to the CDC 6200 computer.Since the beginning of 1974 we have used PASCAL compilers.At present we are using the 6000 - 3.4 version and we must stress that the work with both compilers is a real satisfaction!

In November 1974 we received the PASCAL - P system. On the basis of this system we have realised the interpretative compiler of the PASCAL - P language working on the IBM 370/145 computer.The access to the PASCAL 6000 - 3.4 compiler gave us the possibility to modify the stack computer and to generate a new compiler written in P-code.We have made several modifications,e.g.,we have canceled PMD information and have added new instructions for sets processing.After these changes self-compilation executed on 114 k bytes in time of 7 min.30 sec. We have detected several errors in source compiler.There are as follow

1. If the program structure is wrong and eof (input) becomes true,then the compiler will be looping and printing "EOF ENCOUNTERED";

2. In references of p↑.a type the condition p<>nil is not always checked, e.g., procedure parameterlist
...if (lkind = actual) and (lsp↑.size<ptrsize) then...

3. Decreasing counter lcmax in procedure body should be parameterised similarly to initialisation of field vaddr for variables input, output, prr, prd in procedure entstdnames should be also parameterised.

4. In procedure insymbol the call of nextch is lacking for the last alternative in case instruction.

5. If the procedure is omitted, the variable forw in procedure procdeclaration is not initialized.

6. The construction write(ch) is correct, but write (input) is not. It does not seem to be consistent.

In the received source compiler there are often more than 72 or even 80 characters in a line, what makes additional difficulties in correcting.

At present, we are working on the compiler of the whole PASCAL language for the IBM 370 computer.

Yours sincerely

M. Iglewski M. Missala

M Iglewski M Missala

EIDGENÖSSISCHE
TECHNISCHE HOCHSCHULE ZÜRICH

Institut für Informatik

Clausiusstrasse 55
CH-8006 Zürich
☎ 01/32 62 11

Mr. George H. Richmond
University of Colorado
Computing Center
3645 Marine Street

Boulder, Colorado 80302

January 12, 1976

Dear George,

I have just read the letter by Prof. Pumplin of November 5 addressed to you concerning our Pascal-P system. It brings up problems of which we had mostly been aware. In short, the Pascal-P system is not in a state where it can be used in a student environment.

As you know, Pascal-P was produced on a side-line, and released because of urgent request for support of other implementation endeavours. Due to lack of manpower and in the hope that these efforts would soon produce other compilers of high quality, I refrained from pursuing Pascal-P any further.

With some surprise I now recognise the growing influx of requests for Pascal-P, and have decided to invest some additional effort into this system. We shall concentrate on making the compiler foolproof, on incorporating a number of still missing features, and on providing a satisfactory documentation. I suggest that no more deliveries are made until this is done, and enclose a list of points that are going to be covered. A basic premise is to minimize the changes to the P-code, such that existing interpreters will still be usable with minimal additions.

Also, an effort is now being launched at preparing a Release 2 of our Pascal 6000-3.4 system. I also enclose a list* of points to be covered, emphasis being on removing some deficiencies and restrictions, and on a single system, from which those for other character sets can automatically be derived.

Sincerely yours,

N. Wirth

Prof. Niklaus Wirth

14.1.76

Intended improvements on the PASCAL-P system

1. Improve robustness of compiler. Compiler must not hang when erroneous programs are compiled.
   - compile-time checks of constants as array bounds and set elements,
   - complete runtime checks,
   - compiler must continue when encountering undeclared identifiers,
   - test of system through "bug farm",
   - systematic search for uninitialised pointers within compiler.

2. Relax restrictions
   - accept read and write procedure calls without explicit file specification (allow defaults),
   - include missing standard function (round, trunc, succ, pred ...),
   - allow set expressions of the form [a..b] ,
   - allow empty record declaration (correction of syntax),
   - formal procedures and functions (perhaps).

3. Remove the tag field in the elements of the run-time stack. (Reduce stack size by half.) This requires a slight extension of the instruction repertoire of the interpreter.

4. Include the possibility to satisfy alignment conditions in address generation (storage allocation).

5. Provide a complete documentation on
   - ramaining restrictions
   - how to bootstrap
   - the interpreter (hypothetical stack machine)
   - tape contents.

RECAU
DET REGIONALE EDB-CENTER VED AARHUS UNIVERSITET
NY MUNKEGADE . 8000 AARHUS C . DANMARK . GIRO 180639 . TLF. 06-12 83 55

January 12, 1976

PASCAL Newsletter
Mr. George H. Richmond
University of Colorado
Computing Center
3645 Marine Street
Boulder, Colorado 80302
USA

At the regional EDP Center, University of Aarhus, we are working on a PASCAL manual. We are interested in getting in contact with others working on similar projects. The reasons for starting this work were the following,

- PASCAL is our most used compiler and therefore we have a responsibility to offer a precise and complete description of the language.
- Our existing documentation is covered by a number of publications.
- Not all local modifications are documented.
- The existing documentation (known by RECAU) contains a number of loose and undefined constructs.

Yours,

Jørgen Staunstrup

Ewald Skov Jensen

CALIFORNIA INSTITUTE OF TECHNOLOGY

PASADENA, CALIFORNIA 91125

INFORMATION SCIENCE 286-80

January 19, 1976

Mr. George H. Richmond
Computing Center
University of Colorado
Boulder, Colorado 80302

Dear Mr. Richmond,

I would appreciate it if you would supplement my note of October 1975 with the more recent announcement enclosed for publication in the Pascal Newsletters.

Yours sincerely,

Per Brinch Hansen

NEW CONCURRENT PASCAL REPORTS

Concurrent Pascal is a language for structured programming of operating systems. A portable compiler developed on the PDP 11/45 computer is now being distributed.

The available reports describe Concurrent Pascal and several Model Operating Systems written in the language. They also explain how to move the compiler and the operating systems to other computers.

Reports

Concurrent Pascal Introduction
Concurrent Pascal Report

The Solo Operating System
The Job Stream System
A Real-time Scheduler

Concurrent Pascal Machine
Implementation Notes

Tapes

PDP 11/45 Tape
Other Computer Tape

To obtain copies, please write to

Per Brinch Hansen

Information Science 286-80
California Institute of Technology
Pasadena, California 91125

UNIVERSITY OF MINNESOTA
TWIN CITIES

University Computer Center
227 Experimental Engineering Building
Minneapolis, Minnesota 55455

January 21, 1976

Andy Mickel

NOTICE OF THE FORMATION OF A PASCAL USER'S GROUP AND A CHANGE IN THE PASCAL NEWSLETTER MACHINERY

Since the time John and I wrote the letter to the editor about forming a Pascal User's Group, one has formed. At the national ACM '75 conference held in Minneapolis last October, Richard Cichelli of Lehigh University and Bob Johnson of St. Cloud State University urged me to convene an ad hoc User's Group meeting. Thirty-five persons attended, and besides talking about all aspects of Pascal, we decided to work toward a more permanent organization and use Pascal Newsletter as the vehicle for communication among members.

I have subsequently talked several times to George, the current editor, about changes in Pascal Newsletter. After this issue I will become the new editor. Several goals I have as editor are:

1) publish the newsletter more regularly (4 times per year).
2) expand its scope; up until now subjects in the newsletter have included: notices of important events (new Pascal books, questionaires, etc.), history of Pascal, notices of other implementations and contacts, letters to the editor, etc. I would like to add to these: articles on Pascal philosophy, a forum for portable (applications) programs exchange, hints on promoting Pascal at your computer center, announcement of new program writing tools for a Pascal environment, etc.
3) the newsletter as the official publication of the User's Group, should provide the perfect forum for sounding out proposed changes to various implementations among the users before they are undertaken.
4) accept articles from others on Pascal topics.
5) expand the readership to users of Pascal as well as maintainers and implementors.

Recently I have talked with several Pascalers about how to have a successful User's Group. Alfred Towell of Indiana University, Richard Cichelli, and Wilhelm Burger of the University of Texas, and I think that we should have membership dues. Members will automatically receive Pascal Newsletter and the dues will cover its production costs and mailing. In this regard, remember to urge your fellow Pascal friends whether in person or by announcement in your local computer center newsletter, to join in order to keep costs down. In the past the Newsletter has been funded by the charges George made for distributing Pascal compilers ($15.00 - 30.00). I propose a membership fee of four ($4) per year which includes 4 issues of the newsletter. Make checks payable to:

University Computer Center

64

and send dues to:

Pascal User's Group
c/o Andy Mickel
UCC: 227 Exp Engr
University of Minnesota
Minneapolis, MN 55455

SEL DEVELOPMENT CORPORATION
1900 QUAIL ST., NEWPORT BEACH, CALIFORNIA 92660
(714)833-9752

February 12, 1976

Mr. Nicklaus Wirth
Berichte des Instituts für Informatik
Eidgenössische Technische Hochschule
Zürich, Switzerland

Dear Mr. Wirth:

We have just completed installation of PASCAL-P on the SEL 8600 computer and would like to share with you some of the details of our experience.

We obtained the configured compiler from Mr. George Richmond in Colorado. Our effort required two man-months. The implementation consists of an assembler/interpreter (including extensive P-level debugging tools) written in 8600 FORTRAN. The P-compiler self-compiles in about thirty minutes, which is acceptable for our purposes.

Our overall impression of the PASCAL-P kit is positive. The compiler is well-written and readable. The architecture of the P-machine was not difficult to emulate. However, our experience revealed several areas where improvement could save future implementers at least twenty-five per cent of the effort we required. In the suggestions that follow, we have omitted those which are specific to the 8600.

1. Character set difficulties

Our original compiler used the ETH character set, with translation to the external ASCII set occurring on input and output. (There is no choice for the implementer here, see configuration difficulties section.) We wished to convert the package to the ASCII character set for two reasons. First, we planned to convert PASCAL-P to produce "real" object code and thought that the best time to switch character sets was before other error-inducing modifications were made. Second, we felt that removing translation functions would increase the speed of the

65

DEPARTMENT OF ELECTRICAL ENGINEERING
UNIVERSITY OF COLORADO
BOULDER, COLORADO 80309 USA

2 February 1976

PASCALJ

The 1 February release of PASCALJ is now available for distribution. There have been no substantive changes in either the compiler or the definition of Janus since 1 September.

We have completed the bootstrap of PASCALJ to the Xerox Sigma 3, and in the process uncovered and corrected a number of errors in the compiler. Storage mapping proved to be a serious stumbling block for the implementor, and hence we have developed a package of macros which accepts some target machine parameters (sizes and alignments of data objects) and maps the Janus specifications onto a linear memory. These macros now form a part of the standard distribution tape, and are described in report SEG-76-1.

Our next release is planned for 1 September 1976. At that time we hope to have made further progress in simplifying the task of Janus implementation. We are also studying the results of the Sigma 3 implementation and trying to complete at least one other bootstrap.

Next September we shall automatically report our progress to anyone who orders the current release. Others who would like further information at that time should let us know.

-Software Engineering Group

23 January 1976

Dear Mr. Richmond,

We have just finished implementation of our PASCAL-P compiler (configuration parameters: 65535, 1, 1, 1, 1, 3, 1, 8, 23). Please note that according to your instructions we need a 3 word setsize, but a scan of the compiler-generated-sets indicated that two 24 bit words would suffice (for our computer which has double word load and store hardware instructions, the efficiencies are significant).

Our implementation approach was to convert the Interpreter for P-code into SFOR (a structured FORTRAN) on the FOX1 24 bit computer (target machine). We have just compiled the first test program using the P-compiler. Implementation required about 6 programmer-months (July 1975 to January 1976) to complete so far.

Our most time-consuming problems involved the character codes (both at the beginning and at the end). Warn your user's that when the compiler assembly code wants to compare a character with the top of the stack, it uses the integer which represents the ordinal position of the character in the character set of your CDC machine. This surprise was not found till a mouth ago late in the implementation.

Overall we were very pleased with the service and materials that were provided.

Mr. Robert J. Matherne
Senior Research Engineer
The Foxboro Company
Foxboro, Massachusetts 02035
(617) 543-8750 x2032

interpreter significantly (we were wrong; the speed increase was only four per cent).

The following items made changing character sets difficult (finding the items was our biggest problem; by enumerating them here, we hope to spare future implementers this search time):

a. The compiler presumes a 64-character set, there is little reason to not make the limit a configuration parameter (thus allowing for EBCDIC, as well as ASCII). Even more disturbing is the fact that the limit is bound in an executable statement at line 1232 (line width = 80 characters), instead of being a constant definition.

b. The scanner uses the arrays SSY and SOP declared to be indexed by '+' .. ';' for mapping special characters to symbols. This declaration not only assumes ORD (';') ≥ ORD ('+'), but also assumes that all characters between '+' and ';' in the ETH set are between '+' and ';' in all other sets. We circumvented this difficulty by changing the declarations to index the arrays by the entire CHAR type (promptly causing an error due to point (a)). We feel that this is the most portable solution, costing little storage increase, and requiring only minor change to initialization code in procedure RATORS.

   The scanner presumes that the characters '0' to '9' and 'A' to 'Z' are contiguous and ordered, such that ORD ('Z') < ORD ('0'). That is, of course, not true in all character sets. We feel that the most portable alternative would be to use the excellent facilities of PASCAL to define sets named LETTERS and DIGITS, using set membership operations instead of relational comparisons in the scanner. This, of course, requires a PASCAL implementation where the maximum number of set elements is greater than or equal to the number of characters in the implementation's character set. CDC PASCAL notwithstanding, we think this is a good idea.

2. Configuration difficulties

   The SEL 8600 is a 32-bit word machine with byte, half-word and double-word addressing. Since this is not unlike an IBM 360, we followed the example in the kit order form, specifying INTSIZE = 4, BOOLSIZE = 1, CHARSIZE = 1, and SETSIZE = 8. The P-compiler, as well as the P-architecture, unfortunately, seems to be based on the assumption that characters occupy the same amount of space as integers. By far, the worst manifestation of this assumption (and the worst feature of the entire P-kit) is the use of LDCI <ORD (char)> by the compiler to indicate the load of a single character. We can see no difficulty that would prevent LDC 'char' from being emitted instead. The lack of this small change required us to order a second configuration tape (kindly provided by G. Richmond at no charge). This configuration was based on the use of half-words for integers, characters, and booleans.

In conclusion, we wish to reiterate our overall satisfaction with PASCAL-P. The above mentioned difficulties point to considerations for future portable compiler efforts, and should not obscure the fact that we have a working compiler after a remarkably short amount of time.

If an ongoing effort exists to support portable PASCAL, please let us know. We have noted a number of additional minor problems not mentioned here that we will detail at your request.

Sincerely,

James Gilbert Michael Richmond

JG/MR:dp

cc: N. H. Nageli

George Richmond
University of Colorado

INFORMATION SCIENCE LABORATORIES

FACULTY OF SCIENCE, UNIVERSITY OF TOKYO
2-11-16 YAYOI, BUNKYO-KU TOKYO, 113 JAPAN

T. Hikita, K. Ishihata
Department of Information Science
Faculty of Science
University of Tokyo
Tokyo 113 Japan

February 21, 1976

Mr. George Richmond
Computing Center
University of Colorado
3645 Marine Street
Boulder, Co. 80302
U.S.A.

Dear Mr. Richmond;

This is a short notice on our recent implementation of PASCAL based on a "trunk" compiler. Dr. H. H. Naegeli told us to send it to you for the PASCAL Newsletter. (Please see an enclosed copy of his letter.)

An extended version of Standard PASCAL named PASCAL 8000 was designed, and its compiler was implemented on a Japanese computer HITAC 8800/8700 at the Computer Center of the University of Tokyo. This computer is a multi-processor system of 4 CPU with 4 megabytes main memory, and it has a quite similar machine instruction set to that of the IBM 360 and 370 series computers.

Our language extensions are concerned with constant definitions for structured types, variable initializations, new control structures named "forall" and "loop" statements, and "procedure skeletons" for procedure and function parameters proposed by Lecarme-Desjardins.

The implementation was done by bootstrapping using the PASCAL-P interpretive compiler developed by U. Ammann. Our new compiler is based on H. H. Naegeli's "trunk" compiler. This is a source program of a PASCAL compiler written itself in PASCAL, in which machine dependent parts (code generation, addressing, etc.) are marked and strictly separated from other machine independent parts, and detailed comments are provided which indicate the data or algorithms to be described to the final form by an actual implementor. The version we used was still at developmental stage, though almost completed.

INFORMATION SCIENCE LABORATORIES

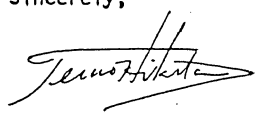
FACULTY OF SCIENCE, UNIVERSITY OF TOKYO
2-11-16 YAYOI, BUNKYO-KU TOKYO, 113 JAPAN

About 1200 lines of the source program have been rewritten to the final form for our HITAC 8800/8700 out of total 5260 lines, and about 400 lines have been newly added for our own language features. This amount of rewriting is considered to be fairly small compared with other works of bootstrapping appeared in the literatures, and one of the values of the trunk compiler of course lies in this point. We felt difficulty during the rewriting process mainly in the expression evaluation scheme. For the other parts, the coding was rather simple and straightforward, though not trivial.

For the details of the language definition of PASCAL 8000 and its implementation based on the trunk, the following two technical reports will be available from our department in March.

T. Hikita, K. Ishihata, PASCAL 8000 Reference Manual.
K. Ishihata, T. Hikita, Bootstrapping PASCAL Using a Trunk.

Sincerely,



Teruo Hikita

Kiyoshi Ishihata

CENTRO DE CALCULO DE LA
UNIVERSIDAD POLITECNICA
DE BARCELONA.
Avda. Doctor Gregorio Marañón, s/n.
BARCELONA-14 Tel. 333 29 49

76/03/02

Mr. George H. Richmond
University of Colorado
Computing Center
3645 Marine Street
Boulder, Colorado 80302
USA

Dear Mr. Richmond,

Our Computing Center is currently involved in a project leading to the implementation of Concurrent Pascal Machine (Prof. Per Brinch Hansen - Caltech) on other real and virtual Machines. Our main interest is in the field of concurrent processes and Operating Systems but since we use Pascal, we have problems the Pascal community has.

We are using professor Steensgaard-Madsen version of Pascal on our Univac 1108. Pascal is used for educational purposes and also to write and test sequential algorithms for the Operating Systems written in Concurrent Pascal.

Would you be so kind as to include us in your mailing list?.

Thanking you in advance, I remain

Sincerely yours.

M. Vergés
DIRECTOR

72

EIDGENÖSSISCHE
TECHNISCHE HOCHSCHULE ZÜRICH

Institut für Informatik
Clausiusstrasse 55
CH-8006 Zürich
Ø 01 / 32 62 11

March 5, 1976

To those who have received Release 1 of our PASCAL 6000-3.4 system

This is to inform you about the availability of Release 2. Its main characteristics are:

- full compatibility with Release 1,
- all known bugs of Release 1 (update level 10) are corrected,
- slightly improved compile speed,
- reduced compile field length (small programs now compile with CM46000),
- execution in REDUCE mode is now possible,
- no restriction on the number of externals, external references, and length of code per procedure,
- print out of an error summary (i.e. a verbal explanation of the error numbers) in case of nonsuccessful compilation,
- runtime test on invalid pointer values,
- implementation of the standard procedure DISPOSE.

Based on Release 2, two extensions will in the near future be considered for implementation. This concerns:

- dynamic array parameters
- a variable initialization facility.

The handling charge for Release 2 is Sfr. 100.-- and includes the costs for the tape.

If you are interested to receive Release 2, you are invited to fill out the enclosed order form. Please don't pay in advance. You will be charged at delivery.

We regret to be unable to continue the support of Release 1.

Sincerely yours,

U. Ammann

Dr. U. Ammann

encl.

73

ORGANISATION EUROPÉENNE POUR LA RECHERCHE NUCLÉAIRE
CERN EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH

SIÈGE: GENÈVE/SUISSE

CERN LABORATOIRE I

Adresse postale / Postal address:
1211 GENÈVE 23
SUISSE / SWITZERLAND

Votre référence
Your reference

Notre référence
Our reference PS/CCI/RC/afcs

Mr. George H. Richmond
University of Colorado
Computing Center
3645 Marine Street
Boulder, Colorado 80302
U. S. A.

Geneva, 12th April 1976

Dear Mr. Richmond,

We are very much interested in a Pascal compiler to be used for production programming on our control system.

If you have a list of implementations, could you please send me the sublist for PDP-11 computers. Requirements for the compiler are the following :

- produce code for a PDP-11/45 or PDP-11/10 machine, (i.e. for both types of floating point!),
- run under DEC's RSX-11/D real time operating system,
- compile itself on the PDP,
- the object code produced must be relocatable and fully compatible with the TKB (linking editor) of RSX,
- the sources of both compiler and run time library must be available (on DECtape or Magtape).

For several months now I have been looking for such a compiler, and I would be glad even to find one that does not satisfy all the requirements. I looked at the following implementations :

- Mr. Feiereisen (Karlsruhe, West Germany) : it is one of the first P-code compilers and implements Pascal-1, it runs under DOS-11, compiles itself, but is not useful since it is not standard Pascal.
- Prof. P.B. Hansen's Solo System : ordered months ago, has not arrived yet, but uses the hardware of the PDP 11/45 and its I & D space which is not supported by RSX. Since it is a stand-alone system, it will probably be difficult to lift it and put it in RSX.
- Mr. H. Nägeli (ETH Zürich) has written a trunk (skeleton) compiler and fills up the code-generation parts. It looks like an efficient compiler, but at present has no real arithmetic and is designed as a cross-compiler : it is far too big to put on an 11. Implementation

under RSX would probably take 6 months to a year.

- A letter has been sent to Bron & De Vries (TH Twente, Holland), and I am awaiting their reply.

In fact, it would be a nice thing once a year to spend an entire Pascal Newsletter on implementations. Such a list would be welcome, and you are probably in the best position to prepare it. However, some details must be given with each implementation, such as :

- minimum hardware required (model & peripheral),
- operating system under which it runs,
- core taken (in bits or bytes),
- whether it implements full standard Pascal or not,
- whether it compiles itself on its minimum configuration,
- where it can be obtained, at what cost,
- what the distribution medium is.

From the above, you will have noticed that a simple mentioning of "a PDP-11 implementation" is far too little!

Looking forward to your reply and thanking you in advance,

Yours sincerely,

R. Cailliau

R. Cailliau
PS Division

VÝSKUMNÉ VÝPOČTOVÉ STREDISKO
PROGRAM OSN

VVS OSN

COMPUTING RESEARCH CENTRE
UNITED NATIONS D. P.

Dúbravská cesta, BRATISLAVA, Czechoslovakia

Bratislava March 19, 1976

Mr. George H. Richmond
University of Colorado
Computing Center
3645 Marine Street
Boulder, Colorado 80302
U.S.A.

Dear Editor,

Unfortunately I failed to respond to your call for existing Pascal /like/ compilers. Please accept my somewhat belated information:

In December 1971 it was decided to implement a Pascal compiler at the Computer Research Centre in Bratislava Czechoslovakia. We have the CDC 3300 Computer. This compiler was to be used as systems programming tool and therefore we felt justified in making some minor changes to Pascal. These modifications were conceived and implemented on the basis of what you call Pascal-0 version of language and compiler. These modifications are the following:

- providing for separate compilation of procedures,
- sharing of variables via ext-entry mechanism,
- allocating static and dynamic memory for non-recursive or recursive procedures
- omitting the file concept
- slight modifications-unifications in the syntax of declarations
- adding new statements, such as SELECT statement to cater for long IF-THEN-ELSE chains
- introducing multi-level exists instead of gotos /this was higly fashionable then/.

Theso changes, and the fact that the portability of the compiler was one of the main design goals, have led us to implement a completely new compiler, without using the ETH code. This compiler is a three-pass compiler with syntax and semantic part written in totally machine independent way, using intermediate language.

Now some basic facts about the implementation and the history of the use of the compiler:

During the months of January and February 1972 a Pascal-to--assembler /CDC 3300/ bootstrap compiler was written in SNOBOL by one person. In the following six months two people

Telephone: 46531-6
Telex: 09587
P. O. Box: 1101

/later three/ coded the compiler in the bootstrap Pascal. After September 1972 bootstrap, compiler was in extensive use by some fifty odd programmers in our centre.

Until now they coded some 60 thousand lines of Pascal code for the following main applications:

- a data base system
- a compiler for input editing and correction of statistical data
- a compiler for table and report generation /also for statistics/
- various editing and library systems used as a support for systems programming
- a Translator Writing System.

The above mentioned compilers generate the intermediate language directly fed to the Pascal code generator and use TWS for syntax analysis.

The Pascal compiler was since then twice expanded, improved, "beautified" and subsequently bootstrapped.

During the months of December 1973 and January 1974 I wrote a code generator in to 360-assembler. The 360 compiler was then successfully installed on the SIEMENS 150 computer /the same hardware-different OS/ in a period of one week; and in two days time it was installed on 370/145 computer. /This was in February and March 1974/. One year later /December 1974 - January 1975/ I modified the 370 compiler to produce directly the relocatable code by writing in Pascal my own and nearly complete 370 assembler, with the commands specified via the procedure calls, instead of the symbolic format. This I consider to be a rather elegant solution and my assembler was already used in an independent 360 compiler project. This modified 360/370/ /SIEMENS Pascal compiler was then installed in more computer installations at home and abroad. In the USA the compiler is installed since February 1975 at the Bureau of Labor Statistics, Washington DC.

Needless to say that all 360/370/SIEMENS Pascal compilers run also as cross-compilers on our machine.

Our SIEMENS implementation is used for teaching purposes at the University of Bratislava. There is also an almost complete implementation for czechoslovak TESLA-200 computer /BULL-License/ at the Technical University of Prague.

The compiler is also available on Commecon RJAD Computer Series.

During the summer of 1975 some 20.000 lines of code which form part of the above mentioned systems were successfully transported to IBM-370. Finally, in January 1976, I extended our compiler so that it might incorporate processes and Hoare-Hansen's monitors as built-in language constructs. The entire change

was made in a period of a week; as a result we have got a concurrent version of the Pascal compiler /this is not the Hansen s Concurrent Pascal, because it was not possible to integrate totally its constructs without making extensive modifications in the existing compiler/. Nevertheless, we have obtained a consistent and proper extension of our compiler, such that the normal sequential programs are subset of the language accepted by our compiler. /Note that Hansen s Concurrent Pascal is used only for parallel programs and that sequential programs and processes must be coded in the normal Pascal./

Since then I with another person have implemented a better part of an interactive terminal system /named Bratislava Terminal System - BTS/. This editor-compiler system consists of some 3000 lines of Pascal code and contains 9 different monitors and 5 process classes.

By this time the major part of the system is running and is being gradually used by our programmers.

We did not experience any difficulties with implementation of BTS and are not aware of any serious programming errors either. This I not mean as a boast but rather I want to stress the advantages of the combination of two very powerful tools: excellent language on one side and elegant monitors on the other side. This combination made the programming of BTS an exciting and rewarding experience in the use of the latest programming methodologies.

Yours sincerely,

P. J. Voda

Paul Voda

Dr. P. J. Voda
Computing Research Centre
Dubravska 3
885 31 Bratislava
Czechoslovakia

A PASCAL-Cross-Compiler for the PDP11-series (all models) written in PASCAL is now operational on the DEC System 10. With slight modifications it can be adapted to other PASCAL systems.
The system is available on 9-track magnetic tape or on DEC-tape (two reels are necessary) and can be obtained -- free of charge - by anyone who sends a tape (please do not send tape as parcel, but as letter) to

C. Bron or J. Entrop
Dept. of Electrical Engineering,
TW-building,
Twente University of Technology,
P.O. Box 217,
Enschede, Netherlands.

5 files will be loaded on tape
- The Compiler's Source Code
- A Documentation File
- The text of the runtime support in PAL-11.
- The code of the runtime support in compiler input format
- An auxiliary program to transform output from the MACRO-11 assembler into compiler input format.

The latter two files are only relevant for those who wish to alter the existing runtime-package.

The compiler generates absolute load-modules in "position independent code". The code makes no assumptions with regard to the Operating System in use and can - if wanted - be run on a bare PDP11.
PASCAL has been fully implemented with the exception of files. The only files available are "input" and "output" for which the teletype can be used. Provisions for the programming of other peripherals in PASCAL are present.

The following extensions to PASCAL have been implemented:

- user defined procedures for interrupt handling
- functions may deliver results of non-scalar type
- procedures/functions as formal parameters
- arrays can be passed as var-parameters without requirements on their size.
- strings (of any length) can be passed as input parameters to procedures
- access to elements of packed boolean arrays is allowed.

EIDGENÖSSISCHE
TECHNISCHE HOCHSCHULE ZÜRICH

Institut für Informatik

Clausiusstrasse 55
CH - 8006 Zürich
ø 01 / 32 62 11

July 76

A new Release of the PASCAL-P system.

This is to inform you about the availability of a new Release of our PASCAL-P system.

Terminology used in the sequel:

Pascal P1: either of the early Pascal P systems (released in March and July 1973 respectively).
Pascal P2: the Pascal P system released in May 74.
Pascal P3: the new Pascal P system with the same hypothetical machine as the one underlying the Pascal P2 system.
Pascal P4: the new Pascal P system with a slightly modified hypothetical machine (allowing a more efficent implementation).

Pascal P3

The compiler is improved in many details. It does, however, still generate code for the old P2 assembler interpreter. The characteristics of the P3 system are:

- 'Full' compatibility with the P2 system.
- The two records of the assembly code produced by the compiler are terminated by the symbol 'Q' instead of two 'end of line'.
- All known bugs are corrected.
- Character set independence.
- Runtime tests are included (indexing, assignement to subrange variables, and case selection are checked for legality.)
- The standard functions 'succ' and 'pred' are implemented.
- The usual default conventions 'readln = readln(input)' etc. hold.

Pascal P4

The compiler generates code for a modified assembler-interpreter. The characteristics of the P4 system are:

- It contains all the improvements of the Pascal P3 system.
- An enlarged set of instructions is used. All instructions now handle exactly one type of expression (or have a type indicator). This allows to eliminate book-keeping of type information at runtime and of tag fields in the stack. No implicit type conversion takes place any more. Instead, explicit type conversion instructions are generated by the compiler.
- The compiler respects possible alignment conditions for the allocation of data.
- A runtime test on pointer values is provided.
- A test on runtime stack overflow is generated by the compiler at procedure entry.

Explanations of the installation parameters

intsize,realsize,charsize,boolsize,setsize,ptrsize:
Number of addressable storage units to be reserved for variables of type integer, real, character, boolean, set, pointer. As to 'setsize', remember that a set must be able to hold at least 48 elements if you intend to use the system to bootstrap the compiler.

intal,realal,charal,boolal,setal,ptral:
Variables of the corresponding types will be given an address which is a multiple of these alignment constants.

stackelsize: Minimum size for a value on the expression stack. The expression stack is that portion of the stack which is used for the evaluation of expressions. 'Stackelsize' has to be equal to or a multiple of 'stackal'.

stackal: Alignment constant for a valu on the expression stack. 'Stackal' must be a multiple of all other alignment constants and must be less or equal to 'stackelsize'.

strglgth: Maximum length of a string. (in fact all strings will be of length 'strglgth'). A string must be able to hold the character representation of a number (real or integer) with its sign. The minimum length for a bootstrap is 12.

intbits: Number of bits used for representing an integer without the sign. So the largest integer is

$$2^{intbits} - 1$$

sethigh,setlow: Maximum and minimum ordinal values for the element of a set.

ordmaxchar,ordminchar: Maximum and minimum ordinal values of the character set.

Depending on the alignment conditions there may be two possibilities for the assignment of store on top of the expression stack.
- Each stack element requires the same amount of store: In this case 'stackelsize' has to be greater than or equal to the maximum of the other size constants.(Remember: 'stackelsize is a multiple of 'stackal')
- No waste of store: A new element on the expression stack has to be placed at the next position allowed by the alignment constant 'stackal'. In this case 'stackelsize' has to be less than or equal to the maximum of the other size constants.

The handling charges include the costs for generating the binary versions of the compiler, a minitape, and postage.

with kind regards

 Ch. Jacobi

Order form for the revised Pascal P system:

Please provide us with your revised Pascal P system according to the specifications on next page.

Address for delivery of the system

The characteristics of our installation are

Machine type	
Operating system	

Installation parameters (to be filled for case 'A' and 'B' below)

intsize		intal	
realsize		realal	
charsize		charal	
boolsize		boolal	
ptrsize		ptral	
setsize		setal	
stackelsize		stackal	
strglgth			
intbits			
sethigh		setlow	
ordmaxchar		ordminchar	

We order

I ☐
- Pascal P4 compiler(in Pascal).
- Pascal P4 compiler (in P4 code).
- An assembler interpreter of P4 code (in Pascal, for documentation purposes, all alignement and size constants are set to 1).
- Pascal P compiler implementation notes with update list..
- Pascal P3 compiler (in Pascal).

  With line numbers, to indicate where it differs from PASCAL-P2. (All installation parameters set to a standard value.)

Charge SFr 160.-

II ☐ (For users who have access to a CDC 6000 Computer and want to experiment with the compiler)
- Pascal P4 compiler with some changes, so that it is accepted by the Pascal 6000 compiler (in Pascal). (All installation parameters set to a standard value.)
- An assembler interpreter (in Pascal, as in package 'A').
- Pascal P compiler implementation notes with update list.

Charge SFr 80.-

III ☐ - Update list to 'PASCAL-P compiler implementation notes'.
Charge SFr 5.-

Date : Signature :

NB Order forms should be sent to the closest PASCAL-P distributor.

PASCAL Distributors

Mr. Christian Jacobi
Eidgenössische Technische Hochschule Zürich
Institut für Informatik
Clausiusstrasse 55
CH-8006 Zürich
Switzerland
(prices at left)

Mr. George Richmond
University of Colorado
Computing Center
3645 Marine Street
Boulder, Colorado 80309
USA
(write for prices)

Mr. Carroll Morgan
Basser Department of Computer Science
University of Sydney
Sydney, New South Wales 2006
Australia
(write for prices)

PASCAL - USER MANUAL AND REPORT

Corrections to 2nd Edition

p = page, l = line, c = code (i.e. r = replace, i = insert)

p	l	c	
13	-3	r	"if" by "If"
45	-18	r	"<unsigned constant>" by "<constant>"
51	16	r	"(output)" by "(output);"
56	-6	r	"fi" by "f(i)", "g(i+1)" by "g(j+1)", "gi" by "g(j)"
63	2	r	"1" by "n", "2" by "n-1", "3" by "n-2", "(n-1)" by "2", "n" by "1"
69	23	r	"stricly" by "strictly"
70	7	r	"i,j" by "i"
77	18	r	" begin inorder(p↑.llink);" by "begin inorder(p↑.llink);"
78	-14	r	", <formal" by "; <formal"
81	2	r	"extent" by "extend"
84	-15	r	"as ne" by "as one"
86	8	i	The procedure read can also be used to read from a file f which is not a textfile. read(f,x) in this case stands for begin x := f↑; get(f) end
87	8	i	The procedure write can also be used to write onto a file f which is not a textfile. write(f,x) in this case stands for begin f↑ := x; put(f) end
98	10	r	"debby" by "debby ';'"
102	7	r	" r" by "or"
102	20	r	"bufffer" by "buffer"
103	-6	r	"scaler" by "scalar"
103	-7	r	" char, and .alfa" by "and char are listed"

105	0	r	" " by "105"
105	12	r	" nly" by "only"
105	-1	i	dispose(p,t1,...,tn) can be used to indicate that storage occupied by the variable p↑ (with tag field values t1...tn) is no longer needed.
117		r	(diagram expression) "≤" by "<=", "≥" by ">=", "≠" by "<>"
120	-18	r	"neither be formal nor non local" by "not be declared on intermediate level"
121	4	i	177: assignment to function identifier not allowed here 178: multidefined record variant 179: X-opt of actual proc/func does not match formal declaration 180: control variable must not be formal 181: constant part of address out of range
121	8	i	205: zero string not allowed 206: integer part of real constant exceeds range
121	-8	i	260: too many exit labels
124	-15	r	"14" by "15"
124	-14	r	"14" by "15"
127	27	r	"18.A" by "4.A"
133	3	r	"two" by "to"
135	5	r	"althought" by "although"
135	30	r	"subtstitute" by "substitute"
140	11	r	"structure type" by "structured type"
161	-17	r	whole line by "addition to the procedures get and put. The textfiles these"
161	-16	r	whole line by "standard procedures apply to must not necessarily represent"
162	3	r	"end of line" by "end of line"
162	-15	i	The procedure read can also be used to read from a file f which is not a textfile. read(f,x) is in this case equivalent to x := f↑; get(f).
162	-6	i	The procedure write can also be used to write onto a file f which is not a textfile. write(f,x) is in this case equivalent to f↑ := x; put(f).

UNIVERSITY OF MINNESOTA
TWIN CITIES

University Computer Center
227 Experimental Engineering Building
Minneapolis, Minnesota 55455
(612) 376-7290

March 18, 1976

Dear Pascal User,

As you are probably aware there is an enormous growth of interest in the new programming language Pascal as both a vehicle for teaching programming concepts and for developing reliable software. In response to this interest a new Pascal User's Group is currently being formed. Members of the group will receive quarterly issues of the new expanded *Pascal Newsletter* which will contain a wide range of information both submitted by and of interest to the Pascal user, teacher, maintainer, implementor, or just plain fan. (This Pascal User's Group will be supplanting the informal Pascal Newsletter published previously by the University of Colorado and which some of you may have been receiving.)

We would like to formally invite you and any member of your staff to join. Membership is $4.00 / Academic year and entitles you to 4 issues of the Newsletter. (The first issue will appear September, 1976.) Please fill out the enclosed coupon along with your check or money order made payable to "Pascal User's Group" and mail to the indicated address. Extra copies of the coupon are included to facilitate any future correspondence.

We would also appreciate it if you would please forward this letter and a coupon to any other persons who you feel may be interested in joining the User's Group. In particular, institutional memberships (e.g. libraries) are encouraged.

Yours truly,

Andy Mickel, Editor

G. Michael Schneider, Associate Editor

John P. Strait, Associate Editor

Pascal Newsletter

PASCAL USER'S GROUP

USER'S

GROUP

ALL PURPOSE COUPON

****

(Clip, photocopy, or reproduce and mail to:

Pascal User's Group / c/o Andy Mickel

University Computer Center: 227 Exp Engr

University of Minnesota

Minneapolis, MN 55455)

/ / Please enter me as a member of the *Pascal User's Group* for the current Academic Year ending June 30, 1977. I understand that I shall receive all 4 issues of *Pascal Newsletter* for the year. Enclosed please find $4.00.

/ / My new address is printed below. Please use it from now on. I'll enclose an old mailing label if I can find one.

/ / Enclosed are some bugs I would like to report to the distributor of the _____ version of Pascal. Please forward it to the appropriate person so that something can be done about it.

/ / Enclosed please find a contribution (such as what we are doing with Pascal at our computer installation), idea, article, or opinion which I wish to submit for publication in the next issue of the *Newsletter*.

/ / None of the above. _____

_____

_____

_____

Other comments:

From:

name _____

address _____

_____

_____

phone _____

PASCAL IMPLEMENTORS

The following lists contain the names and addresses of people who are working on Pascal compilers for various machines. The first list is of recipients of the PASCAL-P2 system written in Zurich and distributed from the University of Colorado. The date given is that of the most recent contact with the person(s) given. In some cases, it is the shipment date. After the date is the configuration parameters, the target machine, and the implementation status when known.

Mr. S. Kamal Abdali
Rensselaer Polytechnic Institute
Department of Mathematical Sciences
Troy, New York 12181

7 November 1975

Mr. Eric W. Anderson
12049 Greenwalk Drive
St. Louis, Missouri 63141

31 May 1976
131071, 4, 4, 1, 1, 8, 3, 11, 31

Mr. Mike Ball
Code 2522
Naval Undersea Center
San Diego, California 92132

21 May 1975
65535, 1, 1, 1, 1, 2, 1, 12, 35

Mr. Henry Bauer
P. O. Box 3682
Computer Science Department
University of Wyoming
Laramie, Wyoming 82071

1 May 1975
131071, 1, 1, 1, 1, 2, 1, 11, 31
Xerox Sigma 7

Mr. Steven Bellovin
University of North Carolina
Department of Computer Science
New West Hall
Chapel Hill, North Carolina 27514

7 January 1976
16777215, 4, 8, 1, 1, 8, 4, 16, 31

Mr. David A. Bennett
Pattern Analysis+Recognition Corp.
228 W. Dominick Street, On the Mall
Rome, New York 13440

15 December 1975
65535, 2, 4, 1, 1, 8, 2, 8, 15

Mr. Lou Beverino
Computer Center
California State University
Northridge, California 91324

8 May 1975
CDC 3300

Mr. Stephen W. Borden
Biology Data Center
Inst. of Animal Resource Ecology
University of British Columbia
2075 Westbrook Place
Vancouver, British Columbia
Canada V6T 1W5

7 May 1975
16777215, 4, 8, 1, 1, 8, 3, 11, 31
IBM 370/168 under MTS

Mr. Kenneth L. Bowles
UCSD Computer Center
P. O. Box 109
La Jolla, California 92037

22 August 1975
131071, 1, 1, 1, 1, 2, 1, 16, 47
Burroughs B6700, PDP-11
Proceeding

Mr. C. E. Bridge
E. I. du Pont de Nemours & Company
Engineering Physics Laboratory
101 Beach Street
Wilmington, Delaware 19898

10 November 1975
32767, 2, 4, 1, 1, 8, 2, 11, 15

Mr. Robert Bruce
Harris Corporation
Computer Systems Division
1200 Gateway Drive
Fort Lauderdale, Florida 33309

5 November 1975
32767, 1, 2, 1, 1, 3, 1, 8, 23

Dr. John Earl Crider
7201 Brompton Road, Apt. 114
Houston, Texas 77025

27 May 1975

Mr. Edouard J. Desautels
University of Wisconsin-Madison
Computer Science Department
1210 West Dayton Street
Madison, Wisconsin 53706

17 October 1975
32768, 1, 2, 1, 1, 3, 1, 16, 23

Mr. Edward E. Ferguson
Texas Instruments, Inc.
P. O. Box 2237
Huntsville, Alabama 35804

1 September 1975

Mr. Charles V. Gaylord
Mr. Micheal Richmond
Mr. James Gilbert
Manager, Software Design
Systems Engineering Laboratories
1900 Quail Street
Newport Beach, California 92660

24 October 1975
262143, 4, 4, 1, 1, 8, 4, 12, 31
65535, 1, 2, 1, 1, 4, 1, 20, 15
SEL 8600
Complete

Dr. James E. George
Mr. Robert T. Johnson
Los Alamos Scientific Laboratory
C10 - Mail Stop 268
P. O. Box 1663
Los Alamos, New Mexico 87544

2 May 1975

Mr. A. J. Gerber
University of Sydney
Basser Dept. of Computer Science
Sydney, New South Wales, 2006
Australia

15 August 1975
16777215, 25, 34, 8, 1, 72, 24, 16, 24
Burroughs B1726
Proceeding, Expected December 1975

Mr. K. W. Giese
Virginia Polytechnic Institute
Computer Science Department
560 McBryde Hall
Blacksburg, Virginia 24061
or
Dr. Johannes J. Martin
1001 Highland Circle SE
Blacksburg, Virginia 24060

28 April 1975
16777215, 4, 8, 1, 1, 8, 3, 11, 31
IBM 370/158 under ASP

Mr. Jonathan R. Gross
University of Minnesota
Soc. Sci. Res. Facilities Center
25 Blegen Hall
Minneapolis, Minnesota 55455

6 March 1975
DEC PDP 8/E

Dr. Gilbert Hansen
Dept. of Computer+Info. Sciences
512 Weil Hall
University of Florida
Gainesville, Florida 32611

22 April 1975
3145728, 4, 8, 1, 1, 8, 3, 11, 31
IBM 370/165 under OS/MVT
22 April 1975
65536, 1, 2, 1, 1, 4, 1, 8, 15
TI 980A
Complete, Available for distribution

Mr. Robert Hartmann
2551 Jacaranda Street
Santa Ana, California 92701

3 March 1976
16777215, 4, 8, 1, 1, 8, 4, 16, 32

Mr. Andrews W. Hastings
Information Processing Systems 9297
Raytheon
Electromagnetic Systems Division
93 Castilian
Goleta, California 93017

31 May 1976
327679, 4, 8, 1, 1, 8, 3, 11, 33

Prof. T. S. Heines
Computer and Information Science
Cleveland State University
Cleveland, Ohio 44115

21 May 1975
65535, 1, 2, 1, 1, 4, 1, 8, 15
TI 980A

Mr. Earnest E. Hughes
Systems Programmer
CHI Corporation
11000 Cedar Avenue
Cleveland, Ohio 44106

31 March 1976
65535, 1, 1, 1, 1, 2, 1, 12, 35

Dr. Fred M. Ives
Dept. of Math. and Computer Sci.
Western Washington State College
Bellingham, Washington 98225

22 July 1975
16777215, 4, 8, 1, 1, 8, 3, 11, 31
IBM 370, Interdata 70

Mr. George D. Jelatis
University of Minnesota
Dept. of Lab. Medicine+Pathology
Div. of Health Computer Sciences
Medical School
P. O. Box 511, Mayo Memorial Bldg.
Minneapolis, Minnesota 55455

15 May 1975
CDC 3300

Mr. Larry J. Jensen
Datapoint Corporation
9725 Datapoint Drive
San Antonio, Texas 78284

5 November 1975
65535, 2, 4, 1, 1, 8, 2, 8, 15

Mr. Peter Kornerup
Computer Science Department
Univ. of Southwestern Louisiana
USL Station Box 4-4330
Lafayette, Louisiana 70504

3 March 1976
16777215, 1, 1, 1, 1, 2, 1, 12, 35

Dr. Robert M. Lansford
Burroughs Corporation
3620 Greenhill Road
Pasadena, California 91107
or
Mr. William C. Price
Burroughs Corporation
460 Sierra Madre Villa Ave.
Pasadena, California 91109

10 February 1975
Burroughs B4700, B6700

Prof. Kyu Y. Lee
University of Montana
Department of Computer Science
Missoula, Montana 59801

31 March 1975
58367, 1, 1, 1, 1, 2, 1, 12, 35

Prof. Daniel W. Lewis
Dept. of EE and Comp. Sci.
The University of Santa Clara
Santa Clara, California 95053

31 May 1976
65535, 1, 2, 1, 1, 4, 1, 6, 16

Mr. Hugh MacKenzie
CSIRO
P. O. Box 1800
Canberra City, A. C. T.
Australia 2601

18 December 1974

Mr. Robert Matherne
Senior Research Engineer
Research Department
Foxboro Corporation
Foxboro, Massachusetts 02035

23 January 1976
65535, 1, 1, 1, 1, 3, 1, 8, 23
FOX 1

Mr. Ronald L. McDaniels
Varian Data Machines
2722 Michelson Drive
Irvine, California 92664

21 February 1975
32767, 1, 2, 1, 1, 4, 1, 6, 15
32767, 1, 1, 1, 1, 4, 1, 16, 15

Mr. Carlton Mills, Soft. Eng.
Mills International
203 North Gregory
Urbana, Illinois 61801

10 February 1975
1048575, 1, 1, 1, 1, 2, 1, 17, 39
Burroughs B6700, Modcomp

Mr. Richard O'Brien
Texas Instruments, Inc.
13510 North Central Expressway
North Building
Dallas, Texas 75232

9 October 1975
16777215, 4, 8, 1, 1, 8, 3, 16, 31

Mr. Stephen A. Pitts
305 East Jarman Drive
Midwest City, Oklahoma 73110

15 May 1975

Mr. Fred Powell, Director
Computer Center
Mary Baldwin College
Staunton, Virginia 24401

21 May 1975
16383, 1, 2, 1, 1, 4, 1, 8, 15
IBM 1130

Prof. Bruce A. Pumplin
University of Wisconsion
Department of Computer Science
Garfield and Park Avenue
Eau Claire, Wisconsin 54701

5 November 1975
32767, 1, 1, 1, 1, 2, 1, 13, 39
Burroughs B5700
Complete

Mr. James D. Rogan
Comshare, Inc.
Wolverine Tower
P. O. Box 1588
Ann Arbor, Michigan 48106

13 May 1975
131071, 1, 1, 1, 1, 2, 1, 11, 31
Xerox Sigma 9

Mr. Richard L. Roth
1052 Clark Avenue
Mountain View, California 94040

21 April 1976
65535, 2, 4, 1, 2, 8, 2, 11, 15
Microdata 800

Mr. David Rowland, Manager
Electro Scientific Industries, Inc.
13900 N. W. Science Park Drive
Portland, Oregon 97229

17 November 1976
65536, 2, 4, 1, 1, 8, 2, 16, 15
DEC PDP 11/05
Proceeding, Expected Summer 1976

Mr. Mark D. Rustad
Moorhead State University
Computer Center
1104 7th Avenue South
Moorhead, Minnesota 56560

31 May 1976
UNIVAC 90/30, Motorola 6800

Prof. A. H. J. Sale
Department of Information Science
University of Tasmania
G. P. O. Box 252C
Hobart 7001, Tasmania

18 August 1975
196607, 1, 1, 1, 1, 2, 1, 11, 39
Burroughs B6700
Proceeding, Expected December 1975

Mr. James P. Shores
344 Glenwood Avenue
New London, Connecticut 06320

21 May 1975
262143, 1, 1, 1, 1, 2, 1, 12, 35
Univac 1108

Mr. Thomas C. Socolofsky
221 Agriculture Hall
Michigan State University
East Lansing, Michigan 48823

22 July 1975

Mr. Henry Spencer
P. O. Box 302
Sub Post Office 6
Saskatoon, Saskatchewan
Canada S7N 0W0

13 February 1975
65535, 2, 8, 1, 1, 8, 2, 8, 15
DEC PDP 11

Mr. Gordon Stuart
Camosun College
Technical and Vocational Institute
1950 Lansdowne Road
Victoria, B. C., Canada V8P 5J2

3 March 1976
65535, 2, 4, 1, 1, 8, 2, 8, 15
PDP 11/40

Mr. Robert A. Stryk
Honeywell, Inc.
Mathematical Sciences Department
Corporate Research Center
10701 Lyndale Avenue South
Bloomington, Minnesota 55420

18 December 1975
71000, 1, 1, 1, 1, 2, 1, 12, 20

Dr. Andrew S. Tanenbaum
Wiskundig Seminarium
Der Vrije Universiteit
Amsterdam - 1001 Postbus 7161
De Boelelaan 1081
Netherlands

7 February 1975
65535, 2, 4, 1, 1, 8, 2, 6, 15
DEC PDP 11/45

Mr. Michael Teener
Technology Service Corporation
2811 Wilshire Blvd.
Santa Monica, California 90403

31 May 1976

Mr. Greg B. Thagard
60 Linda Isle
Newport Beach, California 92660

31 March 1976

Mr. Alfred I. Towell
Wrubel Computer Center/Hper
Indiana University
Bloomington, Indiana 47401

17 October 1975

Mr. Robert D. Vavra
Sperry Univac Computer Systems
2276 Highcrest Drive
Roseville, Minnsota 55165

7 February 1975

Prof. Jean G. Vaucher
Departement d'informatique
Universite de Montreal
Case postale 6128
Montreal 101, Canada

15 December 1975

Mr. Scott K. Warren
warren, rowe & associates
2715 Bissonnet, suite 212
Houston, Texas 77005

25 July 1975
549755813887, 1, 1, 1, 1, 2, 1, 16, 39

Mr. Masaru Watanabe
Institute of Industrial Science
University of Tokyo
22-1, Roppongi 7 Chrome, Minato-Ku
Tokyo 106, Japan

13 February 1975
16777215, 4, 8, 1, 1, 8, 3, 11, 31
FACOM 230/55

Mr. Ben Watson
Texas Instruments, Austin
Mail Station 2091
P. O. Box 2909
Austin, Texas 78767

18 March 1976
65535, 2, 4, 1, 1, 8, 2, 8, 15

Mr. Waldo M. Wedel
University of Texas
Computation Center
Austin, Texas 78712

18 December 1974

Prof. David B. Wortman
Computer Systems Research Group
University of Toronto
Toronto, Ontario
Canada M5S 1A4

14 March 1975
16777215, 4, 8, 1, 1, 32, 4, 32, 31
16777215, 8, 8, 8, 8, 8, 8, 8, 31
IBM 370

Mr. Walter Wuensch
Mobydata, Inc.
P. O. Box 462
Ontario, New York 14519

27 January 1975
65535, 2, 4, 1, 1, 4, 1, 11, 31
Data General 840

Mr. John S. Yates
Computer Identics Corp.
31 Dartmouth Street
Westwood, Massachusetts 02090

23 June 1975
65535, 2, 4, 1, 1, 8, 2, 8, 15

Mr. Kenneth Young 3311 West 3rd Street, Apt. 1-319 Los Angeles, California 90020	19 December 1975 524287, 4, 8, 1, 1, 8, 3, 11, 31 IBM 370/145 under VM/CMS
California State Univ. of Chico Computer Center Chico, California 95926	22 December 1975 32768, 1, 2, 1, 1, 3, 1, 8, 23
Stanford University Stanford Linear Accelerator Center 2575 Sand Hill Road Menlo Park, California 94025	21 May 1975

The second list is of other Pascal implementation efforts. Following the date of last contact with the implementor(s) is the implementation route, the target machine, and the current status when known.

Mr. Laszlo Almasi & Mr. Jeno Lehel Computing and Automation Institute Hungarian Academy of Sciences H-1014 Budapest, Uri u. 49 Hungary	12 December 1975 PASCAL-P2 CDC 3300 Proceeding
Prof. Tore Amble Division of Informatics University of Trondheim N-7034 Trondheim - NTH Norway	24 April 1974 PASCAL-P1 Univac 1108 Complete, Available for distribution
Dr. C. Bron Dept. of Electrical Engineering Technical University of Twente P. O. Box 217, Enschede Netherlands	8 August 1975 PASCAL-P1 PDP 11 Proceeding, Expected December 1975
Mr. Pierre Desjardins Departement D'Informatique Universite de Montreal Case Postale 6128 Montreal 101, Quebec Canada	1 November 1974 Xerox Sigma 6
Monsieur Marcel Dupras Institut de Programmation Tour 55-65 11 Quai Saint Bernard F-75 Paris France	1 December 1974 PASCAL-P1 CDC 3600, CII 10070 Complete
Mr. Lucien Feiereisen Inst. f. Biokyber. u. Biomed. Techk. Universitaet Karlsruhe D-7500 Karlsruhe 1 Kaiserstrasse 12 Germany	30 June 1975 JANUS PDP-11/45 under DOS/BATCH Complete

Dr. Charles N. Fischer University of Wisconsin MACC: 1210 W. Dayton Street Madison, Wisconsin 53706	30 December 1975 Univac 1110 Proceeding, Expected Fall 1976
Dr. W. Bruce Foulkes PASCAL Distribution Manager Department of Computer Science University of Manitoba Winnipeg, Manitoba Canada R3T 2N2	16 April 1974 New compiler written in PL360 IBM 370 Proceeding, Available for distribution
Prof. Dr. G. Goos Institut Fuer Informatik II 75 Karlsruhe 1 Zirkel 2 Germany	1 November 1974 PASCAL-P1 Burroughs B6700 Complete
Prof. Per Brinch Hansen Information Science 286-80 California Institute of Technology Pasadena, California 91125	19 January 1976 DEC PDP 11/45 Complete, Available for distribution
Mr. Al Hartmann Mail Code 286-80 California Institute of Technology Pasadena, California 91109	29 April 1974 PASCAL-P1 IBM 370 Complete, Distribution terminated
Monsieur Gerard Henneron IREP Department Informatique Boite Postale 47 38040 Grenoble CEDEX France	1 December 1974 PASCAL1 Bootstrap IBM 360 under OS/MVT Complete
Mr. Teruo Hikita Mr. Kiyoshi Ishihata Mr. Michiaki Yasumura Information Sciences Laboratories University of Tokyo Faculty of Science Bunkyo-ku Tokyo 113 Japan	21 February 1975 PASCAL-P2 Hitac 8800/8700 Complete
Mr. M. Iglewski Computation Centre Polish Academy of Sciences P. O. Box 22 00-901 Warsaw PKiN Poland	30 December 1975 PASCAL-P2 IBM 370/145 Proceeding
Prof. Richard B. Kieburtz PASCAL Compiler Project Department of Computer Science SUNY at Stony Brook Stony Brook, New York 11794	15 March 1976 New compiler IBM 360 and 370 Complete, Available for distribution

Dr. John Larmouth (Pascal) Univ. of Cambridge Comp. Lab. Corn Exchange Street Cambridge CB2 3QG England	1 November 1974 Interpreter IBM 370 Complete, Available for distribution
Mr. R. Moll Leibniz-Rechenzentrum der BAW Barerstrasse 21 D-8000 Muenchen 2 Germany	4 December 1974 Telefunken TR440 Complete
Prof. Dr. H.-H. Nagel Institut Fuer Informatik Universitaet Hamburg Schlueterstrasse 70 D-2000 Hamburg 13 Germany or Mr. Waldo M. Wedel University of Texas Computation Center Austin, Texas 78712	1 July 1975 PASCAL-P1 DEC System 10 Complete, Available for distribution
Pascal Group 267 Digital Computing Laboratory University of Illinois Urbana, Illinois 61801	29 August 1974 PASCAL1 Bootstrap DEC PDP 11/20 Complete
Dr. S. V. Rangaswamy School of Automation Indian Institute of Science Bangalore 560 012 India	24 October 1974 PASCAL-P1 IBM 360/44
Mr. David L. Russell Digital Systems Laboratory Stanford University Stanford, California 94305	3 September 1974 PASCAL0 Bootstrap IBM 370 Complete, Distribution terminated
Software Engineering Group University of Colorado Department of EE Boulder, Colorado 80302	1 September 1975 Intermediate language - Janus Portable Preliminary version available Next release - 1 February 1976
Mr. Rod Steele Tektronix, Inc. P. O. Box 500 Beaverton, Oregon 97002	11 December 1975 Burroughs B3700 Complete
Dr. Jorgen Steensgaard-Madsen Datalogisk Institut University of Copenhagen Sigurdsgade 41 DK-2200 Copenhagen N Denmark	25 July 1974 PASCAL-P1 Bootstrap Univac 1100 under EXEC 8 Complete, Available for distribution
Mr. H. C. de Ruyter van Steveninck Philips Research Laboratories Eindhoven Netherlands	1 November 1974 PASCAL-P1 Philips P1400 Complete

Mr. Masato Takeichi Dept. of Mathematical Engr. Faculty of Engineering University of Tokyo Bunkyo-ku Tokyo 113 Japan	1 September 1974 PASCAL-P1 Interpreter MELCOM 7700, Xerox Sigma 7 (monitor BPM) Complete 1 March 1975 PASCAL-P1 Interpreter FACOM 230-38 (OS2/VS) Complete 1 August 1975 Rewrite of CII IRIS 80 PASCAL1 Compiler MELCOM 7700, Xerox Sigma 7 (monitor BPM) Complete
Monsieur Didier Thibault 17 Rue Gay-Lussac 75005 Paris France	2 May 1974 PASCAL1 Bootstrap CII IRIS 80, CII 10070, Xerox Sigma 7 Complete, Available for distribution
Monsieur Alain Tisserant Ecole des Mines Departement Informatique Parc de Saurupt 54042 NANCY CEDEX France	4 July 1975 PASCAL-P2 Telemecanique T1600, Solar minicomputer Proceeding
Dr. P. J. Voda Computing Research Centre Dubravska 3 885 31 Bratislava Czechoslovakia	19 March 1976 PASCAL0 Bootstrap CDC 3300, IBM 360, SIEMENS 150, Commecon RJ Complete
Dr. Jim Welsh Department of Computer Science Queen's University Belfast BT7 1NN North Ireland	29 July 1974 PASCAL0 Bootstrap ICL 1900 Complete, Available for distribution

Literature about the Programming Language Pascal

Ammann, U., "The Method of Structured Programming Applied to the Development of a Compiler", "International Computing Symposium 1973", Guntner, et al., Eds., pp. 93-99 North Holland (1974)

Ammann, U., "Die Entwicklung eines Pascal-Compilers nach der Methode des strukturierten Programmierens, ETH-Diss. 5456 (1975)

Ammann., U., "On Code Generation in a PASCAL Compiler", Berichte des Instituts fur Informatik, Nr. 13, ETH Zurich (April 1976)

Bachmann, K. H., "Die Programmiersprachen Pascal und Algol 68", Akademie-Verlag, Berlin (1976)

Burger, W. F., "Pascal Manual", Department of Computer Sciences, TR-22, The University of Texas at Austin (July 1973)

Burger, W. F., "BOBSW - A Parser Generator", Department of Computer Sciences, SESLTR-7, The University of Texas at Austin (December 1974)

Bron, C., de Vries, W., "A Pascal Compiler for PDP11 Minicomputers", Department of Electrical Engineering, Twente University of Technology, Enschede, Netherlands (1974); SOFTWARE-PRACTICE AND EXPERIENCE 6, 1, pp. 109-116 (January 1976)

Conway, R., Gries, D., Zimmerman, E., "A Primer on Structured Programming Using PASCAL", Winthrop Publishers, Inc., Cambridge, Massachusetts, approx. 480 pages (due June 1976)

Desjardins, P., "A Pascal Compiler for the Xerox Sigma 6", SIGPLAN NOTICES 8, 6, pp. 34-36 (1973)

Deverill, R. S., Hartmann, A. C., "Interpretive PASCAL for the IBM 370", Information Science Technical Report No. 6, California Institute of Tecnnology (1973)

Feiereisen, L., "Implementation of PASCAL on the PDP 11/45", DECUS Conference, Zurich, pp. 259 (September 1974)

Findlay, W., "The Performance of Pascal Programs on the MULTUM", Report No. 6, Computing Department, University of Glasgow, Scotland (July 1974)

Friesland, G., et al., "A Pascal Compiler Bootstrapped on a DEC-System 10", Lecture Notes in Computer Science 7, pp. 101-113, Springer-Verlag (1974)

Friesland, G., Sengler, H.-E., "Zur Uebertragung von Compilern durch Selbstcompilation am Beispiel des PASCAL-Compilers", Institut fuer Informatik des Universitaet Hamburg, report 1F1-HH-B-13/74 (December 11974)

Grosse-Lindemann, C.-O., Lorenz, P.-W., Nagel, H.-H., Stirl, P. J., "A PASCAL Compiler Bootstrapped on a DEC-System10", Fachtagung uber Programmiersprachen, pp. 101-113, Lecture Notes in Computer Science 3, Springer-Verlag (1974)

Grosse-Lindemann, C.-O., Nagel, H.-H., "Postlude to a Pascal-Compiler Bootstrap on a DEC System-10", Bericht Nr. 11, Institut fur Informatik, Universitat Hamburg, Germany (1974); SOFTWARE-PRACTICE AND EXPERIENCE 6, 1, pp. 29-42 (January 1976)

Habermann, A. N., "Critical Comments on the Programming Language Pascal", ACTA INFORMATICA 3, 1, pp. 47-57 (1973)

Hansen, P. B., "Operating System Principles", Prentice-Hall, Englewood Cliffs, New Jersey (1973)

Hansen, P. B., "The Purpose of Concurrent Pascal", SIGPLAN NOTICES 10, 6, pp. 305-309 (1975)

Heistad, E., "Pascal - Cyber Version", Teknisk Notat S-305 Forsvarets Forskningsinstitutt, Norwegian Defense Research Estaplishment, Kjeller, Norway (June 1973)

Hikita, T., Ishihata, K., "PASCAL 8000 REFERENCE MANUAL, Version 1.0", Technical Report 76-02, Department of Information Science, Faculty of Science, University of Tokyo (March 1976)

Hoare, C. A. R., Wirth, N., "An Axiomatic Definition of the Programming Language Pascal", ACTA INFORMATICA 2, 4, pp. 335-355 (1973)

Illum, K., "En introduktion til programmeringssporget Pascal", Danmarks Ingeniorakademi, Aalborg (1973)

Ishihata, K., Hikita, T., "Bootstrapping PASCAL Using a Trunk", Technical Report 76-04, Department of Information Science, Faculty of Science, University of Tokyo (March 1976)

Jensen, K., Wirth, N., "Pascal User Manual and Report", Lecture Notes in Computer Science, 18, Springer-Verlag, New York (1974); Springer Study Addition (1975)

Knobe, B., Yuval, G., "Making a Compiler Indent", Computer Science Department, The Hebrew University of Jerusalem, Israel (November 1974)

Kristensen, B. B., Madsen, O. L., Jensen, B. B., Eriksen, S. H., "A Short Description of a Translator Writing System (BOBS-System)", Daimi PB-11, University of Aarhus, Denmark (February 1973)

Kristensen, B. B., Madsen, O. L., Jensen, B. B., "A Pascal Environment Machine (P-code)", Daimi PB-28, University of Aarhus, Denmark (April 1974)

Kristensen, B. B., Madsen, O. L., Jensen, B. B., Eriksen, S. H., "User Manual for the BOBS-System", Unpublished English Version, University of Aarhus, Denmark (April 1974)

Lecarme, O., "Le langage de programmation Pascal", Universite de Montreal (1972)

Lecarme, O., "Structured Programming, Programming Teaching, and the Language Pascal", SIGPLAN NOTICES 9, 7, pp. 15-21 (July 1974)

Lecarme, O., Desjardins, P., "Reply to a Paper by A. N. Habermann on the Programming Language Pascal", SIGPLAN NOTICES 9, 10, pp. 21-27 (October 1974)

Lecarme, O., Desjardins, P., "More Comments on the Programming Language Pascal", ACTA INFORMATICA 4, pp. 231-243 (1975)

MacLennan, B. J., "A Note on Dynamic Arrays in Pascal", SIGPLAN NOTICES 10, 9, pp. 39-40 (September 1975)

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)

Marmier, E., "A Program Verifier for Pascal", Information Processing 74 (IFIP Congress 1974), North-Holland (1974)

Molster, T., Sundvor, V., "Unit Pascal System for the Univac 1108 Computer", Teknisk Notat 1/74, Institutt for Databehandling, Universitetet I Tronheim, Norway (February 1974)

Nagel, H.-H., "Pascal for the DEC-System 10, Experiences and Further Plans", Mitteilung Nr. 21, Institut fur Informatik, Universitat Hamburg (November 1975)

Nori, K. V., Ammann, U., Jensen, K., Nageli, H. H., "The Pascal(P) Compiler: Implementation Nrtes", No. 10, Berichte des Instituts fur Informatik, Eidgenossische Technische Hochschule, Zurich (December 1974)

Richmond, G., "Pascal Newsletter", No. 1, University of Colorado Computing Center, Boulder (January 1974); SIGPLAN NOTICES 9, ?, pp. ?-? (? 1974)

Richmond, G., "Pascal Newsletter", No. 2, University of Colorado Computing Center, Boulder (May 1974); SIGPLAN NOTICES 9, 11, pp. 11-17 (November 1974)

Richmond, G., "Pascal Newsletter", No. 3, University of Colorado Computing Center, Boulder (February 1975); SIGPLAN NOTICES 11, 2, pp. 33-48 (February 1976)

Rowland, D., "Pascal for Systems", paper presented at DECUS (Digital Equipment Corporation User´s Society) (December 1975)

Saxena, A. R., Bredt, T. H., "A Structured Specification of a Hierarchical Operating System", SIGPLAN NOTICES 10, 6, pp. 310-318 (June 1975)

Schauer, H., "PASCAL fuer Angaenger", Oldenbourg-Verlag, Wien, Muenchen (1976)

Schild, R., "Implementation of the Programming Language Pascal", Lecture Notes in Economics and Mathematical Systems, 75 (1972)

"SFER PASCAL, Le Langage de programmation PASCAL - compilateur pour les ordinateurs CII 10070, IRIS 80", IRIA (1975)

Solntseff, N., "McMaster Modifications to the Pascal 6000 3.4 System", Computer Science Technical Note 74-CS-2, McMaster University, Ontario, Canada (November 1974)

Takeichi, M., "On the Portability of a PASCAL Compiler", Proceedings of the 16th Programming Symposium, pp. 90-96 (1975) in Japanese

Takeichi, M., "PASCAL Compiler for the FACOM 230-38: Implementation Notes", Internal Report, University of Tokyo, Department of Mathematic Engineering and Instrumentation Physics (1975)

Hochschule, Zurich (June 1975)

Wirth, N., "Comment on A Note on Dynamic Arrays in Pascal", SIGPLAN NOTICES 11, 1, pp. 37-38 (January 1976)

Takeichi, M., "PASCAL -- Implementation and Experience", University of Tokyo, Department of Mathematic Engineering and Instrumentation Physics (December 1975)

Thibault, D., Mancel, P., "Implementation of a Pascal Compiler for the CII Iris 80 Computer", SIGPLAN NOTICES 8, 6, pp. 89-90 (1973)

de Vries, W., "An Implementation of the language Pascal for the PDP 11 series, based on a portable Pascal compiler", Technische Hogeschool Twente, Enschede (March 1975)

Welsh, J., Quinn, C., "A Pascal Compiler for the ICL 1900 Series Computer", SOFTWARE-PRACTICE AND EXPERIENCE 2, 1, pp. 73-77 (1972)

Wirth, N., Hoare, C. A. R., "A Contribution to the Development of Algol", COMMUNICATIONS OF THE ACM 9, 6, pp. 413-432 (1966)

Wirth, N., "The Programming Language Pascal", ACTA INFORMATICA 1, 1, pp. 35-63 (1971)

Wirth, N., "The Design of a Pascal Compiler", SOFTWARE-PRACTICE AND EXPERIENCE 1, 4, pp. 309-333 (1971)

Wirth, N., "Program Development by Step-Wise Refinement", COMMUNICATIONS OF THE ACM 14, 4, pp. 221-227 (April 1971)

Wirth, N., "The Programming Language Pascal and Its Design Criteria", presented at the Conference on Software Engineering Techniques (NATO Science Committee), Rome (October 1969); published in "High Level Languages", Infotech State of the Art Report 7 (1972)

Wirth, N., "Systematisches Programmieren" (Taschenbuch), Teubner-Verlag, Stuttgart (1972)

Wirth, N., "The Programming Language Pascal (Revised Report)", Nr. 5, Berichte des Instituts fur Informatik, Eidgenossische Technische Hochschule, Zurich (November 1972)

Wirth, N., "On Pascal, Code Generation, and the CDC 6400 Computer", Computer Science Department, STAN-CS-72-257, Stanford University (1972) (out of print, Clearinghouse stock no. PB208519)

Wirth, N., "Systematic Programming: An Introduction", Prentice-Hall, Englewood Cliffs, New Jersey (1973)

Wirth, N., "On the Composition of Well-Structured Programs", COMPUTING SURVEYS 6, 4, pp. 247-260 (December 1974)

Wirth, N., "Algorithmen und Datenstrukturen", Teubner-Verlag, Stuttgart (1975)

Wirth, N., "Algorithms + Datastructures = Programs", Prentice-Hall, Englewood Cliffs, New Jersey (1975)

Wirth, N., "An Assessment of the Programming Language Pascal", IEEE TRANSACTIONS ON SOFTWARE ENGINEERING 1, 2, pp. 192-198 (1975); SIGPLAN NOTICES 10, 6, pp. 23-30 (June 1975)

Wirth, N., "PASCAL-S: A Subset and its Implementation", Nr. 12, Berichte des Instituts fur Informatik, Eidgenossische Technische