PASCAL USERS GROUP

Pascal News

NUMBER 21

COMMUNICATIONS ABOUT THE PROGRAMMING LANGUAGE PASCAL BY PASCALERS

APRIL, 1981

If this isn't APRIL...

does that mean we're late ?

EX LIBRIS: David T. Craig
736 Edgewater
[#_____] Wichita, Kansas 67230 (USA)

POLICY: PASCAL NEWS (15-Sep-80)

* Pascal News is the official but informal publication of the User's Group.

* Pascal News contains all we (the editors) know about Pascal; we use it as the vehicle to answer all inquiries because our physical energy and resources for answering individual requests are finite. As PUG grows, we unfortunately succumb to the reality of:

  1. Having to insist that people who need to know "about Pascal" join PUG and read Pascal News - that is why we spend time to produce it!

  2. Refusing to return phone calls or answer letters full of questions - we will pass the questions on to the readership of Pascal News. Please understand what the collective effect of individual inquiries has at the "concentrators" (our phones and mailboxes). We are trying honestly to say: "We cannot promise more that we can do."

* Pascal News is produced 3 or 4 times during a year; usually in March, June, September, and December.

* ALL THE NEWS THAT'S FIT, WE PRINT. Please send material (brevity is a virtue) for Pascal News single-spaced and camera-ready (use dark ribbon and 18.5 cm lines!)

* Remember: ALL LETTERS TO US WILL BE PRINTED UNLESS THEY CONTAIN A REQUEST TO THE CONTRARY.

* Pascal News is divided into flexible sections:

  POLICY - explains the way we do things (ALL-PURPOSE COUPON, etc.)

  EDITOR'S CONTRIBUTION - passes along the opinion and point of view of the editor together with changes in the mechanics of PUG operation, etc.

  HERE AND THERE WITH PASCAL - presents news from people, conference announcements and reports, new books and articles (including reviews), notices of Pascal in the news, history, membership rosters, etc.

  APPLICATIONS - presents and documents source programs written in Pascal for various algorithms, and software tools for a Pascal environment; news of significant applications programs. Also critiques regarding program/algorithm certification, performance, standards conformance, style, output convenience, and general design.

  ARTICLES - contains formal, submitted contributions (such as Pascal philosophy, use of Pascal as a teaching tool, use of Pascal at different computer installations, how to promote Pascal, etc.).

  OPEN FORUM FOR MEMBERS - contains short, informal correspondence among members which is of interest to the readership of Pascal News.

  IMPLEMENTATION NOTES - reports news of Pascal implementations: contacts for maintainers, implementors, distributors, and documentors of various implementations as well as where to send bug reports. Qualitative and quantitative descriptions and comparisons of various implementations are publicized. Sections contain information about Portable Pascals, Pascal Variants, Feature-Implementation Notes, and Machine-Dependent Implementations.

Policy

- - - - - - ALL-PURPOSE COUPON - - - - - - (1-Apr-81)

Pascal Users Group
P.O. Box 4406
Allentown, Pa. 18170-4406 USA

Note

- We will not accept purchace orders.
- Make checks payable to: "Pascal Users Group", drawn on a U.S. bank in U.S. dollars.
- See the Policy section on the reverse side alternate address if you are located in the Australasian Region.
- Note the discounts below, for multi-year subscription and renewal.
- The U. S. Postal Service does not forward Pascal News.

- -

		USA	Europe	Aust.
	[ ] 1 year	$10.	$14.	A$ 8.
[ ] Enter me as a new member for:				
	[ ] 2 years	$18.	$25.	A$ 15.
[ ] Renew my subscription for:				
	[ ] 3 years	$25.	#35.	A$ 20.

[ ] Send Back Issue(s) _____

[ ] My new address/phone is listed below

[ ] Enclosed please find a contribution, idea, article or opinion which is submitted for publication in the Pascal News.

[ ] Comments: _____

_____

_____

ENCLOSED PLEASE FIND: A$
$ _____.____

CHECK no. _____

NAME _____

ADDRESS _____

_____

_____

PHONE _____

COMPUTER _____

DATE _____

JOINING PASCAL USERS GROUP?

- Membership is open to anyone: Particularly the Pascal user, teacher, maintainer, implementor, distributor, or just plain fan.
- Please enclose the proper prepayment (check payable to "Pascal User's Group"); we will not bill you.
- Please do not send us purchase orders; we cannot endure the paper work!
- When you join PUG any time within a year: January 1 to December 31, you will receive all issues of Pascal News for that year.
- We produce Pascal News as a means toward the end of promoting Pascal and communicating news of events surrounding Pascal to persons interested in Pascal. We are simply interested in the news ourselves and prefer to share it through Pascal News. We desire to minimize paperwork, because we have other work to do.

- American Region (North and South America), and European Region (Europe, North Africa, Western and Central Asia): Join through PUGUSA

- Australasian Region (Australia, East Asia - incl. Japan): PUG(AUS). Send $A10.00 per year to: Pascal Users Group, c/o Arthur Sale, Department of Information Science, University of Tasmania, Box 252C GPO, Hobart, Tasmania 7001, Australia. International telephone: 61-02-23 0561 x435

PUG(USA) produces Pascal News and keeps all mailing addresses on a common list. Regional representatives collect memberships from their regions as a service, and they reprint and distribute Pascal News using a proof copy and mailing labels sent from PUG(USA). Persons in the Australasian Region must join through their regional representative. People in other places please join through PUG(USA).

RENEWING?

- Please renew early (before November and please write us a line or two to tell us what you are doing with Pascal, and tell us what you think of PUG and Pascal News. Renewing for more than one year saves us time.

ORDERING BACK ISSUES OR EXTRA ISSUES?

- Our unusual policy of automatically sending all issues of Pascal News to anyone who joins within a year means that we eliminate many requests for backissues ahead of time, and we don't have to reprint important information in every issue--especially about Pascal implementations!
- Issues 1 .. 8 (January, 1974 - May 1977) are out of print.
- Issues 9 .. 12 (September, 1977 - June, 1978) are available from PUG(USA) all for $15.00 and from PUG(AUS) all for $A15.00
- Issues 13 .. 16 are available from PUG(AUS) all for $A15.00; and from PUG(USA) all for $15.00.
- Extra single copies of new issues (current academic year) are: $5.00 each - PUG(USA); and $A5.00 each - PUG(AUS).

SENDING MATERIAL FOR PUBLICATION?

- Your experiences with Pascal (teaching and otherwise), ideas, letters, opinions, notices, news, articles, conference announcements, reports, implementation information, applications, etc. are welcome. Please send material single-spaced and in camera-ready (use a dark ribbon and lines 1[illegible] cm. wide) form.
- All letters will be printed unless they contain a request to the contrary.

APPLICATION FOR LICENSE TO USE VALIDATION SUITE FOR PASCAL

Name and address of requestor:
(Company name if requestor is a company) _____

_____

_____

Phone Number: _____

Name and address to which information should be addressed (Write "as above" if the same) _____

_____

_____

Signature of requestor: _____

Date: _____

In making this application, which should be signed by a responsible person in the case of a company, the requestor agrees that:

a) The Validation Suite is recognized as being the copyrighted, proprietary property of R. A. Freak and A.H.J. Sale, and

b) The requestor will not distribute or otherwise make available machine-readable copies of the Validation Suite, modified or unmodified, to any third party without written permission of the copyright holders.

In return, the copyright holders grant full permission to use the programs and documentation contained in the Validation Suite for the purpose of compiler validation, acceptance tests, benchmarking, preparation of comparative reports, and similar purposes, and to make available the listings of the results of compilation and execution of the programs to third parties in the course of the above activities. In such documents, reference shall be made to the original copyright notice and its source.

✗ Distribution charge: $50.00

✗ Make checks payable to ANPA/RI in US dollars drawn on a US bank. Remittance must accompany application.

Mail request to:

ANPA/RI . .
P.O. Box 598
Easton, Pa. 18042
USA
Attn: R.J. Cichelli

Source Code Delivery Medium Specification:
9-track, 800 bpi, NRZI, Odd Parity, 600' Magnetic Tape

( ) ANSI-Standard

a) Select character code set:
( ) ASCII ( ) EBCDIC

b) Each logical record is an 80 character card image.
Select block size in logical records per block.
( ) 40 ( ) 20 ( ) 10

( ) Special DEC System Alternates:
( ) RSX-IAS PIP Format
( ) DOS-RSTS FLX Format

Office use only

Signed _____
Date _____

Richard J. Cichelli
On behalf of A.H.J. Sale & R.A. Freak

Index

Contributors to this issue (#21) were:

EDITOR	Rick Shaw
Here & There	John Eisenberg
Books & Articles	Rich Stevens
Applications	Rich Cichelli, Andy Mickel
Standards	Jim Miner, Tony Addyman
Implementation Notes	Bob Dietrich, Greg Marshall
Administration	Moe Ford, Jennie Sinclair

Editor's Contribution

NEW ADDRESS

Yes, in my continued effort to bring you better service, (read this as: I can not do all the work effectively!) I have found someone else (read: sucker) to take over the PUG mailing list. I am sure that this will increase the satisfaction level for this task 100%. this will take a great load off of my back and allow me to devote all of my time to editing and publishing Pascal News.

LATE

I thought April first (April Fools Day) was an appropriate target date for this issue of Pascal News! I apologize for the tardiness, but my work (I have a real job that pays the bills) and the many pressing problems and issues of PN got in the way. I had to solve the PUG Europe problem, and try to gather as much as I could concerning the final vote on the ISO standard.

FUTURE OF PUG IN EUROPE

It took me more than a few months to correct the festering problems in Europe surrounding Pascal News. The previous coordinator was sinking under the mire of ever increasing job responsibilities as well as the editorship of clearly the best journal dealing with practical software implementation. (SP&E) As a result, the european region suffered from lack of attention. This is over! PUG cares. Please send your "job well done's" to David in Southampton, and send your complaints to PUGUSA. We will be handling all but the Australasia Region from the US. Please read the new APC carefully for policy and price changes. We will be mailing by surface mail to the UK and Europe, but I have been assured by the USPS that it should take no more than a month. I have been asked if I would mail by air for an extra surcharge. The answer has to be no, at this time. PUG can just not afford the special processing and handling that this would be required for two different types of mail. Sorry!

STANDARDS

Another delay was the standards effort. There is so much going on in the standards arena that we just could not afford to miss it. I think it was worth it. Over half of this issue is devoted to the vote on the ISO standard for Pascal (7185). Jim Miner has done another fine job.

THIS ISSUE

Now the good news! We have another jam packed issue. I think you will recognize out book reviewer this issue. He is an "occasional" contributor to PN. And I hope you will get a chuckle from our "sister" publication PUG PRESS. Andy Mickel brings this little gem to us. The other HERE and THERE article is a real puzzle. It came to me just as you see it!?

The application for this issue was so good I could not miss publishing it. It is a Pascal to EM1 pseudo code compiler by Andrew Tanenbaum. Its a real beauty. But it was sooooo big I could not publish it all ... yet. This issue contains the definition of the assembler language that is output and also an interpreter which serves as the EM1 machine definition. Issue 22 will contain the program text for the EM1 Pascal compiler. I hope everyone reviews the documentation and the code, even if they do not need the compiler. It is a fine example of elegant design and implementation using the language Pascal. Also included in the APPLICATIONS section is an article by Jeff Pepper om the implementation of extended precision integer arithmetic. A fine job.

The ARTICLES section contains a thought provoking extension to the read/write subroutines by David Rowland. Lets hear a response from the members. And finally Maragret Kulos has contributed a very comprehensive article comparing OMSI-1 Pascal and The Swedish Pascal compiler. There is a great deal of interest in these two compilers for the PDP-11. I hope this provides some answers.

All in all, a great issue. More to come on EM1 in issue #22.

Hope you like it!

Rick

****

Here and There With Pascal

BOOK REVIEW

The Pascal Handbook by Jacques Tiberghien
500pp, 270 Illustrations, SYBEX, Berkeley
(1980) $US14.95 (paper edition only),
ISBN 0-89588-053-9.

Overview

This is not a Pascal textbook; it is something very different. Perhaps the most succint description is that it is a Pascal *lexicon*: a sort of all-purpose reference manual. It is organized around entries keyed by an appropriate Pascal word (eg if, scope, writeln) arranged in alphabetical order. Each entry takes up one or more whole pages, and the standard sub-headings are SYNTAX, DESCRIPTION, IMPLEMENTATION-DEPENDENT FEATURES and EXAMPLE. The relevance of the entry to Standard Pascal and a number of particular implementations (HP1000, CDC, OMSI-1, Pascal/Z and UCSD Pascal) is encoded into the entry.

Thus the book is meant to be used as a dictionary to look up difficult points or to find out what some usage in a program you have received really means. As such, it follows a lot of reference manuals which are similarly structured (eg the B6700/7700 Pascal Reference Manual).

However, since Pascal is a small language with not very many things needing to be remembered, it needs to be asked why a lexicon of 500 pages is needed? Examination of the book indicates that its main purpose seems to be to document extensions and differences between implementations. Thus, since its topic is the union of all the quirks of 5 implementations, it has grown to this rather large size.

Target and reality

So much for the target; how does the book match up to it in reality? The answer seems to be that it does a reasonably good job of documenting what exists, but that it does not measure up to the very exacting standards that such an ambitious project warrants. The standard of accuracy against which a dictionary is judged is much higher than that appropriate for textbooks, in which a few lapses can be tolerated or justified on the grounds that pedantic accuracy would impede learning.

The slips in the book are far too numerous to detail (a list is being sent to the author), and a few examples will have to suffice. Dipping into the entry for the reserved word for is probably the richest source of examples. Faults which should be mentioned are:

(1) An "equivalent flow-chart" is given. The sense of defining a high-level construct such as while in flow-chart terms is questionable at the best of times, but for the complex for-statement it is extremely unfortunate in that it might make people think the flow-chart is right. It isn't.
(2) The prohibition on changing the value of the count variable is not mentioned.
(3) The limitations on what a count variable can be (only a local simple variable) are not mentioned.
(4) The correct restriction of the HP-1000 implementation is considered to be an implementation-dependent feature, whereas the corresponding flaw in the J&W/CDC implementation is not mentioned.
(5) The failure of many of these implementations to enforce the requirements of the for-statement is not mentioned; indeed for four implementations the entry is *None known* for implementation-dependent features.
(6) The possibility of the statement failing to terminate (incorrectly) for some limit values in the OMSI and UCSD implementations is not documented.
(7) The statement is made that *The A and B parameters may not be modified by the statement in the loop*. This is simply incorrect, though it is true to say that the loop limits are determined on entry to the loop.

Perhaps this is the worst case to show, but a few more examples will suffice to show that the problem is not isolated. The syntax for MARK shows that this non-standard procedure takes a parameter which is an integer expression. MAXINT is incorrectly described as determining the positive limit of representable integers (which it may be only coincidentally). The syntax for CASE statements is incorrect. And so on.

General issues

There are two major deficiencies in this book which deserve comment. First is the lack of formal definitions, and indeed the appearance of only a few English descriptions that resemble the actual requirements of Pascal. The author claims to be talking about Pascal (presumably the standard variety) as well as the others, but there is simply no basis for comparison if the reader cannot find out what sets, for example, are really supposed to be.

The second is the mystifying omission of any reference to the Pascal Validation effort. If one of the purposes of the book is to aid programmers who wish to write portable Pascal programs, then it is difficult to understand why the author did not carry out validation tests on the five compilers he regards as important, and print the results in a second section of the book. It would have added significantly to the value of the book as a reference.

Minor issues

Regrettably, once again it is necessary to point out that capitals were designed for carving into stone, not for ease of reading. This book perpetuates the habit of printing programs in capitals, with consequent loss of legibility.

It is difficult to deduce the author's criteria for choosing which topics to omit or include. To illustrate this, note that the UNIT feature of USCD Pascal, together with the corresponding USES, INTERFACE and IMPLEMENTATION reserved words, is not treated in the book, apart from a mention, despite their undoubted importance in use. On the other hand, such trivia as a pre-defined function EXP10 in OMSI Pascal takes up 2 pages.

Directing another comment to the publisher rather than the author, one wonders why the tremendous amount of white space in the book was tolerated. A little care in layout (perhaps two entries per page; perhaps denser printing) would have halved the number of pages, and perhaps reduced the price.

Summary view

Despite the criticisms made above, I believe the book would be useful to programmers who have to cope with Pascal programs which were developed on different systems or in different dialects. The level of detail and accuracy of information is not as high as it could be, but nevertheless the book has no competitors.

I doubt that it will be of much use to programmers learning Pascal, still less beginners at programming, because it is too difficult to see what is really Pascal and what is "extension". And of course, dictionaries are simply not meant to be read through.

A.H.J.Sale

BOOK REVIEW

INTRODUCTION TO PASCAL

- including UCSD Pascal

by Rodnay Zaks
320pp, 100 illustrations
Sybex, Berkeley (1980) US$12.95 (Paper Edition only)
ISBN 0-89588-050-4

Reviewed by A.H.J.Sale, Sandy Bay, Tasmania.

Overview

On receiving a book which proclaims that it will teach you a programming language, I conceive that most reviewers will groan and wonder what new there is to say. The more so if the language is a popular one, such as BASIC, Fortran, COBOL, or Pascal. For many educational book writers are plagiarists, and after the fifth to tenth version of the same ideas, my eyes get weary and the text fuzzy...

To start with, then, it is a pleasure to be able to write that Rodnay Zaks book is somewhat different from the run-of-the-mill Pascal books. Firstly it has a definite target readership, and is addressed to them. Dr Zaks' book is well-suited to microcomputer enthusiasts and programmers who want to learn a bit about Pascal but have no immediate intention of using it professionally. The exposition is gentle, fairly easy to read, and liberally interlaced with reading examples.

To enhance its value to such readers, Dr Zaks has decided to include material on one popular variant of Pascal in the microcomputer field: UCSD Pascal. This is interspersed throughout the book in clearly labelled sub-sections.

Secondly, the book has a good collection of examples, and they are not exactly the same examples you find in other textbooks! Learning a language is always easier if you can read it (and read a lot of it), since then you discover samplers (or templates) that you can modify to your own purposes, and thus gradually discover typical programming paradigms of that language.

The presentation is traditional, and there are no surprises. The chapter headings are: Basic Concepts, Programming in Pascal, Scalar Types and Operators, Expressions and Statements, Input and Output, Control Structures, Procedures and Functions, Data Types, Arrays, Records and Variants, Files, Sets, Pointers and Lists, UCSD and Other Pascals, Program Development (15 in all) followed by 12 Appendices including answers to selected exercises.

Shortcomings

In my opinion, the book is not likely to be widely used as a text in tertiary courses, for several reasons. Most importantly, it is very light on the concepts of Pascal and Dr Zaks treats of the language simply as another Fortran or BASIC. Instructors trying to get across the important advances in knowledge about computing will not forgive the lack, whereas readers using it as a self-tutorial almost certainly wouldn't notice the deficiency. Less important, but still relevant, is the typical American verbosity in this kind of book.

To illustrate the conceptual treatment, observe that 6 pages (pp135-140) deal with enumerated types and subranges, and 11 pages (pp247-257) for sets. Other data structuring methods seem to fare better, but this appearance disappears on close examination. For example the array chapter contains 39 pages, but 4 pages are devoted to a matrix addition program, 16 pages to a sorting program, and 8 pages to UCSD features (including UCSD strings which are not arrays at all!), leaving 11 pages of discussion of the syntax and semantics of arrays. The low-level obsession with flow of control is very obvious in this book.

A reviewer cannot pretend to check every program and statement in a book such as this, but I was pleased to note few errors or half-truths in "Introduction to Pascal". Notable amongst the omissions, however, are references to the axiomatic definition of Pascal (surely one of the most important sources!) and to the draft ISO Standard for Pascal. These omissions seem to be related to the book's orientation towards small computers and relatively naive programmers.

In spite of the great care put into this book (its technical presentation is excellent except for the blunder of printing program text in capitals), I had to come to the conclusion that the inclusion of UCSD Pascal in it is a mistake. The book is predominantly about "Standard Pascal", and purchasers who hope to learn something about UCSD Pascal that is not in the UCSD and SofTech manuals will be disappointed. It seems that the UCSD material acts as textual clutter, even if its inclusion on the cover sells more copies.

Summary

"Introduction to Pascal" by Dr Rodnay Zaks is a useful soft-cover book that will probably be useful to people trying to learn Pascal by themselves, due to the many examples. However, it will lead them up to the point of programming using Pascal, but thinking in traditional ways. Many of the insights and productivity improvements will require extensive further experience, but perhaps that is inevitable.

* Pug Press *

Volume One Issue Three March 1980

Publisher: Maryanne Johnson (612)-474-7167
510 Wheeler Drive Excelsior, Minnesota 55331
Editor: Patti Sue Selseth

Even with all the snow on the ground, SPRING IS IN THE AIR!!!! This is a good time to remember to bring your dog's shots up to date and don't forget about heartworm.

One of Marianne's Pug Family has passed away in early February. Helen Landon had only had her PUGS for 2½ years, but she truly loved them. Her love for all animals was a driving force in her life, and she will be missed. The family has requested memorials to Pet Haven or American Cancer Society.

* Congradulations - On Your Newly Aquired PUGS

Tracy Cunningham has a new little girl PUG named Miss Josie Posie Penelope. The day before Christmas she was brought home at the tender age of 2½ weeks. (This should be a reminder that not all breeders are as concerned for the dog's welfare as they should be. There is no excuse for selling a dog at this age for monatary gain. Remind people who are looking for puppies that they should be eating from a dish, and should be able to get along without their Mama and litter mates before they are taken home.)

Mr. and Mrs. Don Coen of South St. Paul are soon to be getting a new baby boy PUG. They recently lost a 13 year old PUG.

Mr. and Mrs. Don Donaldson of River Falls, Wisc. became owners of a male PUG at Christmas time. They bought him from Rachel Fishcher; he was at the Pug Party last fall as a puppy.

Mr. and Mrs. Joe Jenareo of Minot brought home a new female PUG in December. They have an eight year old male and are also looking for another male.

The John Kerschner Family recently bought an eight month old PUG puppy from Dorothy Justad.

* A PUG Name Contest

The John Healy Family would like to know some of the names that have been given to the pugs. So we thought it would be fun to have a "PUG NAME CONTEST." The contest will be based on the registered and/or call names our PUG people have named their PUGS (past and present). Some of the catagories will be: most unusual, most beautiful, most interesting, most common, and most humerous. To enter the contest, please write or call Maryanne before June 1, 1980. All entrants will be mentioned in the next newsletter.

* Have You Heard the Latest???

We have it on good authority that Pandy Wenz has visited Chipper Justad at his home. Early May will tell the tail!!!!

* Birth Announcements

Dorothy Justad is proud to announce the arrival of:

Woodcrofts Foster Fordyce arrived February 12th (the one and only)

Sire: AKC & CKC ptd in Bermuda Ch. Sheffields Shortening Bread
(better known as Chipper)
Dam: Sugar Plum Jen I

* Want Ads

WANTED - Small PUG Stud to breed with the Classiest Bitch in Town. Stud must be experienced yet gentle, loving, and discreet. Contact Ron or Marlys Hampe (612)-890-4141

John G. Waltz; 184 Amherst, St. Paul 55105; is the manager at Sherwoc Pet in St. Paul. He would like a male pet PUG at a reasonable price.

Eunice Thorson; 536 1st St., Proctor, Mn. 55810; recently lost her fourteen year old PUG. She would like another girl puppy or older PUG.

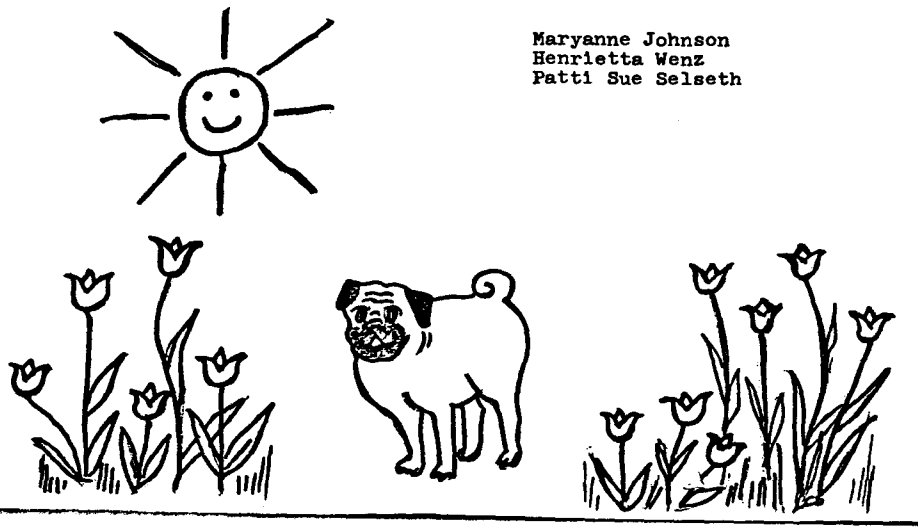
* Thank You - Dorothy

Our thanks to Dorothy Justad for the wonderful article on getting started in show biz. We know it will be useful to those of you interested in showing PUGS.

If you have any PUG news that you would like to share with fellow "PUG PEOPLE" please let us know. Deadline for the next newsletter is June 1, 1980. Just call or write Maryanne, and we'll get your news in the next issue of PUG PRESS.

HAVE A HAPPY SPRING !!!!!!!

Maryanne Johnson
Henrietta Wenz
Patti Sue Selseth

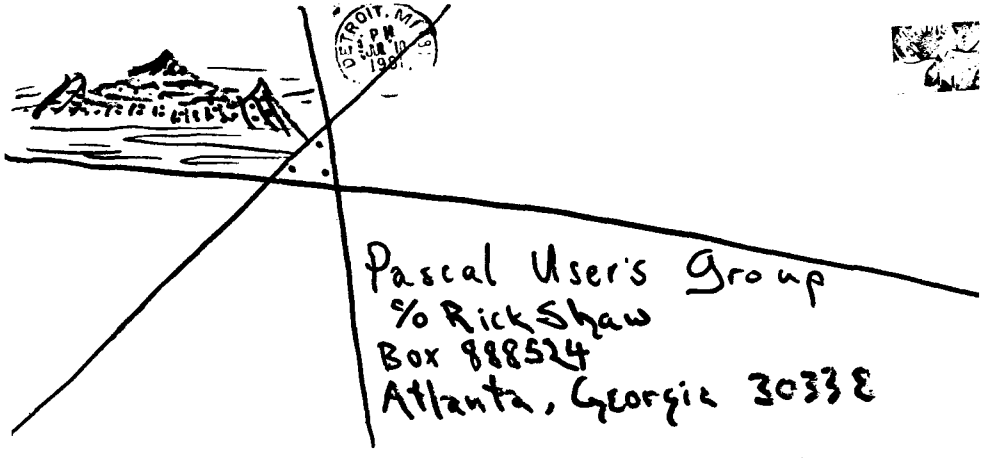


Dear Newsletter enthusiast, the following is a list of subjects that are likely to be examined in the upcoming newsletters. If you feel like it, please respond to any of the subject matter, adding suggestions, visions, or other comments. Bits of any incoming communication are likely to be recycled into the newsletter at some date. This being the first newsletters, the form may change from issue to issue, but my idea initially is to have each letter be a theme examining some proponent of the hypothetical floating sea city, of which we can all be a part.

a. the spiral method of accretion
b acquiring the necessary elements off the land: going into the recycling aspects of the project, recycling of cars, refrigerators, machines for the conducive materials, and also papers and (liquified) plant matter, for the papier mache structures.
c. A deeper tripping out on paper machait: how it can be used to invoke peoples' minds as to the process of accretion, selecting varieties of forms which scintillate. Drawings can be included of terrestrial motifs, walls, time capsules, zoomorphic borders of gardens, rises, walkways, spontaneous expressions of color and form.
d. aspects of energy acquisition and usage: Solar, wind models, underwater exploits; sharing-concepts, valuation.
e. Plantlife likely to evolve, and the natures of emergent ecosystems including overlappings, and new symbioses.
f. Food to be grown, produced, specialty items for shipping away, into the land: Pickles, sweets, noted cheeses, pastries, modes of eating; availability of different substances.
g. Separation of thirds of spaces: industrial/mercantile/co-operative; common/state-owned; and home spaces, privately ruled and operated.
h. Varieties of social forms, explorations of likely traditions to be fused for propulsion into pyremusical ambidextrously mobile batteries. Cultures to be examined including refugees, aliens, star-struck, dropped out, mutating, change-oriented.
i. Blasting of the closed-ended systems, reiterating the expansive potentials inherent in futuristic thinking: an invitation to recent explosions.
j. The inner workings and displayed aspects of the water system in the structure. Designs for waterfalls, ponds, pools, streams, bathing, plant feeding, recirculation, distillation.
k. Art- and Extrapolitical-aspects of lifestyles emerging on the sea. Options for peoples' expressions in career, craft, vocation, activities;
l. An examination of the effort to create groups of three melting, softing tetras, to meet and merge on the high seas, producing the interior lagoons and flatlands. Also known as triangulation, the tendencies of groups of threes to balance and stability.
m. Diagrammatic explanations of the various levels, including shipping ports, flotation devices, fluxuating shores, and sky-high properties. Proportions of spaces allotted to playgrounds, bycicle loops, orchards, cottages, mist gardens, arboretum/terminal stands, geodescic elevating modules, and sky light sculptures of varying densities will be suggested, examined, detailed.
n. Something to attract transient visitors: vacation playgrounds. There are fantasy worlds open for exploration, and technological and entertainment forms. Also perhaps, casino- and pub-like grottoes, looking out to under the waves; and varieties of sports presentations and activities. Contests, fairs, festivals, holidays, erections, revampings, scribblings.
o. Communication with other life forms, and inviting them along for the journey into spaces high and blue. The idea of having a dolphin embassy, a whale tavern in the sea (growing types of algae for them), platforms and niches to support many sea travellers, and those from the sky.
p. A continuously building mural made by contributions from each visitor in all the media. It will start from some initial point(s) and spread as more and more visitors come, make, and go. (Thanx, Yoko)
q. The idea of "not letting an enemy rise on any level", as Maharishi so aptly puts it. The foreign relations applicable as: Ideologies can be shared as love. Using the platform as a museum, a carousel of multiple nationalities and displays of bifurcate merging, develop events which can be generally supported by nations, groups, and factions. In them independent rovers can sniff around.
r. Examinations of the acoustics, the silent cave-likes, the public, open, airy ampetheaters. Electronic and other forms of communication running along its circuits, and extending from its structure.
s. Visions, ideas for schools, markets, subjects to be taught: seems likely there's to be a concentration of the space studies on board, so examining some of the fields briefly: exo-ecology, low gravity motion, non-terrestrial physics, neurogenetic engineering.
t. Health, wholeness, holiness: attaining it and keeping it, some of the newer medicinal statements have been waiting for somewhere like this to display themselves, and from which to fly.
u. The idea as the project not just an end, a new place, but as another link on the roadway. What then is to come next? What first? What has been encouraging this?
v. The extra-realist art movement, its principles and principals.
w. Tributes to those livers of the past who've sent good vibrations into our present sphere. Catacombs and hillsides.
x. The exposition and superimposition of the ideas of nakedness, nudity, nets of reality, and masturbation. Techniques.
y. Proposal for direct access networks to stretch across the land.
z. An animal's or plant's eye view of what we humans have been discussing, sometimes grave, sometimes humorous.

In closing, I would like to add that all flowing waters lead to the sea. Thanks for the initial interest. Direct correspondance to me at: Kevin Switzer, 1534 Ford, Lincoln Park, Michigan 48146.

* **** ** ** * * * ** * * **



Applications

ER-1 ASSEMBLY LANGUAGE

11.1. Introduction

An assembly language program consists of a series of lines, each containing 0 or 1 statements. A machine instruction may not be labeled. In other words, the label field on a machine instruction must be left blank. There are two kinds of labels, instruction and data labels. Labels start in column 1. Instruction labels are unsigned positive integers, and each must appear alone on a line by itself. The scope of an instruction label is its procedure.

The pseudoinstructions CON, ROM, and BSS may be labeled with a 1-8 character data label, the first character of which is a letter, period or underscore, followed by letters, digits, periods and underscores. Only 1 label per line is allowed. The use of the character "." followed by a number (e.g. .40) is recommended for compiler generated programs, since these are considered as a special case and handled more efficiently in compact assembly language (see below).

Each statement may contain an instruction mnemonic or pseudoinstruction. These must begin in column 2 or later (not column 1) and must be followed by a space, tab, semicolon or LF. Everything on the line following a semicolon is taken as a comment.

All constants are decimal unless started with a zero e.g. 0177, in which case they are octal. In CON and ROM pseudoinstructions, floating point numbers are distinguished by the presence of a decimal point or an exponent (indicated by E or e), or both. Double precision (long) integers are followed directly by an L or l.

Also allowed as initializers in CON and ROM are strings. Strings are surrounded by double quotes and may include \xxx, where xxx is a 3-digit octal constant, e.g. CON "hello\012\000". Each string element initializes a single byte. Strings are padded at the end up to a multiple of the word size.

Local labels are referred to as *1, *2, etc. in CON and ROM pseudoinstructions (to distinguish them from constants), but without the asterisk in branch instructions, e.g. BRF 3, not BRF *3.

The notation $procname is used to mean the descriptor number for the procedure with the specified name.

An input file may contain many procedures. A procedure consists of zero or more pseudoinstructions, a PRO statement, a (possibly empty) collection of instructions and pseudoinstructions and finally an END statement. The very last statement on the input file must be EOF. The END directly preceding the EOF may be omitted.

Input to the assembler is in lower case, if available. Upper case is used in this document merely to distinguish key words from the surrounding prose.

11.2. Pseudo instructions

First the notation used for the operands of the pseudo instructions.

<num>	=	an integer constant
<sym>	=	an identifier
<arg>	=	<num> or <sym>
<val>	=	<arg>, long constant (ending with L or l), real constant, string constant (surrounded by double quotes), procedure number (starting with $) or instruction label (starting with *).
<...>*	=	zero or more of <...>
<...>+	=	one or more of <...>

Four pseudo instructions request global data:

BSS <num>
Reserve <num> bytes, not explicitly initialized. <num> must be a multiple of the word size.

MOL <num>
Idem, but all following absolute global data references will refer to this block.

CON <val>+
Assemble global data words initialized with the <val> constants.

ROM <val>+
Idem, but the initialized data will never be changed.

Three pseudo instructions partition the input into procedures:

PRO <sym>,<num1>,<num2>
Start of procedure. <sym> is the procedure name. <num1> is the number of bytes for arguments. <num2> is 1 for procedure names to be exported out of the current module, 0 otherwise.

END
End of Procedure.

EOF
End of module.

Besides the export flag in PRO, six other pseudo instructions are involved with separate compilation and linking:

EXD <sym>
Export data. <sym> is exported out of this module.

IMA <sym>
Import address. IMA allows global symbol <sym> to be used before it is

defined. Note that <sym> may be defined in the same module.

IMC <sym>
Similar to IMA, but used for imported single word constants. These two different forms are necessary, because the assembler must know how much storage must be allocated if <sym> is used in CON or ROM.

FWA <sym>
Forward address. Notify the assembler that <sym> will be defined later on in this module, so that it may be used before being defined.

FWC <sym>
Similar to FWA, but for constants.

FWP <sym>
Forward procedure reference. FWP allows <sym> to be used before it is defined. <sym> must be defined in the same module and must not be exported. Normally, unknown procedure names are entered in the undefined global reference table, so that their names will be known outside this module. Procedure names introduced by FWP are treated differently, however, to prevent their being exported.

Three other pseudo instructions provide miscellaneous features:

LET <sym>,<arg>
Assembly time assignment of the second operand to the first one.

EXC <num1>,<num2>
Two blocks of instructions preceding this one are interchanged before being assembled. <num1> gives the number of lines of the first block. <num2> gives the number of lines of the second one. Blank and pure comment lines do not count.

MES <num>,<val>*
A special type of comment. Used by compilers to communicate with the optimizer, assembler, etc. as follows:

MES 0 -
An error has occurred, stop assembly.

MES 1 -
Suppress optimization

MES 2 -
Use virtual memory (EM-2)

MES 3,<num1>,<num2> -
Indicates that a local variable is never referenced indirectly. <num1> is offset in bytes from LB. <num2> indicates the class of the variable.

MES 4 -
Number of source lines (for profiler).

MES 5 -
Floating point used.

MES 6,<val>* -
Comment. Used to provide comments in compact assembly language (see below).

12. ASSEMBLY LANGUAGE INSTRUCTION LIST

For each instruction in the list the range of operand values in the assembly language is given. These ranges are all subranges of -32768..32767 and are indicated by letters:

m: full range, i.e. -32768..32767
n: 0..32767
x: 0..32766 and even
y: 1 or (2..32766 and even)
z: -32768..32766 and even
p: 2..32766 and even
r: 0, 1 or 2

The letters should not be confused with the letters used in the EM-1 instruction table in appendix 2. Instructions that check for undefined operands and underflow or overflow are indicated by (*).

GROUP 1: LOAD

LOC m - Load constant (i.e. push it onto the stack)
LNC m - Load negative constant
LOL x - Load local word x
LOE x - Load external word x
LOP x - Load word pointed to by x-th local
LAI y - Load auto increment y bytes (address of pointer on stack)
LOF m - Load offsetted. (top of stack + m yield address)
LAL x - Load address of local
LAE x - Load address of external
LEX n - Load lexical. (address of LB n static levels back)
LOI y - Load indirect y bytes (address is popped from the stack)
LOS - Load indirect (pop byte count, address; count is 1 or even)
LDL x - Load double local (two consecutive locals are stacked)
LDE x - Load double external (two consecutive externals are stacked)
LDF m - Load double offsetted (top of stack + m yield address)

GROUP 2: STORE

STL x - Store local
STE x - Store external
STP x - Store into word pointed to by x-th local
SAI y - Store auto increment y bytes (address of pointer on stack)
STF m - Store offsetted
STI y - Store indirect y bytes (pop address, then data)
STS - Store indirect (pop byte count, then address, then data)
SDL x - Store double local
SDE x - Store double external
SDF m - Store double offsetted

GROUP 3: SINGLE PRECISION INTEGER ARITHMETIC

ADD - Addition (*)
SUB - Subtraction (*)
MUL - Multiplication (*)

```
  DIV   - Division (*)
  MOD   - Modulo i.e.remainder (*)
  NEG   - Negate (two's complement) (*)
  SHL   - Shift left (*)
  SHR   - Shift right (*)

GROUP 4: DOUBLE PRECISION ARITHMETIC   (Format not defined)

  DAD   - Double add (*)
  DSB   - Double Subtract (*)
  DMU   - Double Multiply (*)
  DDV   - Double Divide (*)
  DMD   - Double Modulo (*)

GROUP 5: FLOATING POINT ARITHMETIC   (Format not defined)

  FAD   - Floating add (*)
  FSB   - Floating subtract (*)
  FMU   - Floating multiply (*)
  FDV   - Floating divide (*)
  FIF   - Floating multiply and split integer and fraction part (*)
  FEF   - Split floating number in exponent and fraction part (*)

GROUP 6: POINTER ARITHMETIC

  ADI m - Add the constant m to pointer on top of stack
  PAD   - Pointer add; pop integer, then pointer, push sum as pointer
  PSB   - Subtract two pointers (in same fragment) and push diff as integer

GROUP 7: INCREMENT/DECREMENT/ZERO

  INC   - Increment top of stack by 1 (*)
  INL x - Increment local (*)
  INE x - Increment external (*)
  DEC   - Decrement top of stack by 1 (*)
  DEL x - Decrement local (*)
  DEE x - Decrement external (*)
  ZRL x - Zero local
  ZRE x - Zero external

GROUP 8: CONVERT

  CID   - Convert integer to double (*)
  CDI   - Convert double to integer (*)
  CIF   - Convert integer to floating (*)
  CFI   - Convert floating to integer (*)
  CDF   - Convert double to floating (*)
  CFD   - Convert floating to double (*)

GROUP 9: LOGICAL

  AND p - Boolean and on two groups of p bytes
  ANS   - Boolean and; number of bytes is first popped from stack
  IOR p - Boolean inclusive or on two groups of p bytes
  IOS   - Boolean inclusive or; nr of bytes is first popped from stack
  XOR p - Boolean exclusive or on two groups of p bytes
  XOS   - Boolean exclusive or; nr of bytes is first popped from stack
  COM p - Complement (one's complement of top p bytes)
  COS   - Complement; first pop number of bytes from stack
  ROL   - Rotate left
  ROR   - Rotate right

GROUP 10: SETS

  INN p - Bit test on p byte set (bit number on top of stack)
  INS   - Bit test; first pop set size, then bit number
  SET p - Create singleton p byte set with bit n on (n is top of stack)
  SES   - Create singleton set; first pop set size, then bit number

GROUP 11: ARRAY

  LAR x - Load array element
  LAS   - Load array element;  first pop ptr to descriptor from stack
  SAR x - Store array element
  SAS   - Store array element;  first pop ptr to descriptor from stack
  AAR x - Load address of array element
  AAS   - Load address;  first pop pointer to descriptor from stack

GROUP 12: COMPARE

  CMI   - Compare 2 integers. Push negative, zero, positive for <, = or >
  CMD   - Compare 2 double integers
  CMF   - Compare 2 reals
  CMU p - Compare 2 blocks of p bytes each
  CMS   - Compare 2 blocks of bytes;  pop byte count
  CMP   - Compare 2 pointers

  TLT   - True if less, i.e. iff top of stack < 0
  TLE   - True if less or equal, i.e. iff top of stack <= 0
  TEQ   - True if equal, i.e. iff top of stack = 0
  TNE   - True if not equal, i.e. iff top of stack non zero
  TGE   - True if greater or equal, i.e. iff top of stack >= 0
  TGT   - True if greater, i.e. iff top of stack > 0

GROUP 13: BRANCH

  BRF n - Branch forward unconditionally n bytes
  BRB n - Branch backward unconditionally n bytes

  BLT n - Forward branch less (pop 2 words, branch if top > second)
  BLE n - Forward branch less or equal
  BEQ n - Forward branch equal
  BNE n - Forward branch not equal
  BGE n - Forward branch greater or equal
  BGT n - Forward branch greater

  ZLT n - Forward branch less than zero (pop 1 word, branch negative)
  ZLE n - Forward branch less or equal to zero
  ZEQ n - Forward branch equal zero
  ZNE n - Forward branch not zero
```

```
  ZGE n - Forward branch greater or equal zero
  ZGT n - Forward branch greater than zero

GROUP 14: PROCEDURE CALL

  MRK n - Mark stack (n = change in static depth of nesting - 1)
  MRS   - Mark stack; first pop the static link from the stack
  CAL n - Call procedure (with descriptor n)
  CAS   - Call indirect;  first pop procedure number from stack
  RET x - Return (function result consists of top x bytes)
  RES   - Like RET, but size of result on top of stack

GROUP 15: MISCELLANEOUS

  BEG z - Begin procedure (reserve z bytes for locals)
  BES   - Like BEG, except first pop z from stack
  BLM x - Block move x bytes; first pop destination addr, then source addr
  BLS   - Block move; like BLM, except first pop x, then addresses
  CSA   - Case jump; address of jump table at top of stack
  CSB   - Table lookup jump; address of jump table at top of stack
  DUP p - Duplicate top p bytes
  DUS   - Like DUP, except first pop p
  EXG   - Exchange top 2 words
  HLT   - Halt the machine ( Exit status on the stack )
  LIN n - Line number (external 0 := n)
  LNI   - Line number increment
  LOR r - Load register (0=LB, 1=SP, 2=HP)
  MON   - Monitor call
  NOP   - No operation
  RCK x - Range check; descriptor at (external) x; trap on error
  RCS   - Like RCK, except first pop x from stack
  RTT   - Return from trap
  SIG   - Trap errors to proc nr on top of stack (-2 resets default).  Static
          link of procedure is below procedure number.  Old values returned
  STR r - Store register (0=LB, 1=SP, 2=HP)
  TRP   - Cause trap to occur (Error number on stack)
```

13. KERNEL INSTRUCTION SET

Many of the instructions presented in the previous chapter are replacements for a small sequence of basic instructions. The basic instructions form less than half of the complete instruction set. Only a few basic instructions have operands. Most of them fetch their arguments from the stack. Very few basic instructions are provided to load and store objects.

For each of the groups of instructions.the basic ones are given:

```
GROUP  1: LOC, LAE, LEX, LOS
GROUP  2: STS
GROUP  3: ADD, SUB, MUL, DIV, SHL, SHR
GROUP  4: DAD, DSB, DMU, DDV
GROUP  5: FAD, FSB, FMU, FDV, FIF, FEF
GROUP  6: PAD, PSB
GROUP  7: -
GROUP  8: CID, CDI, CDF, CFD
GROUP  9: ANS, IOS, XOS, COS, ROL, ROR
GROUP 10: INS, SES
GROUP 11: AAS
GROUP 12: CMI, CMD, CMF, CMS, CMP, TGT, TLT, TEQ
GROUP 13: DRB, ZNE
GROUP 14: MRS, CAS, RES
GROUP 15: BES, BLS, CSA, CSB, DUS, EXG, HLT, LOR, MON, NOP, RCS,
          RTT, SIG, STR, TRP
```

Almost all the other instructions can be replaced in the assembly language by a short equivalent sequence of simpler instructions. By applying these replacements recursively a sequence of basic instructions can be found.

```
GROUP 1:
  LNC m  =  LOC -m
  LOL x  =  LAL x  +  LOI 2
  LOE x  =  LAE x  +  LOI 2
  LOP x  =  LOL x  +  LOI 2
  LAI y  =  DUP 2  +  DUP 2  +  LOI 2  +  ADI y  +  EXG    +  STI 2  +  LOI y
  LOF m  =  ADI m  +  LOI 2
  LAL x  =  LEX 0  +  ADI x
  LOI y  =  LOC y  +  LOS
  LDL x  =  LAL x  +  LOI 4
  LDE x  =  LAE x  +  LOI 4
  LDF m  =  ADI m  +  LOI 4

GROUP 2:
  STL x  =  LAL x  +  STI 2
  STE x  =  LAE x  +  STI 2
  STP x  =  LOL x  +  STI 2
  SAI y  =  DUP 2  +  DUP 2  +  LOI 2  +  ADI y  +  EXG    +  STI 2  +  STI y
  STF m  =  ADI m  +  STI 2
  STI y  =  LOC y  +  STS
  SDL x  =  LAL x  +  STI 4
  SDE x  =  LAE x  +  STI 4
  SDF m  =  ADI m  +  STI 4
```

```
GROUP 3:
  MOD    =  DUP 4  +  DIV    +  MUL    +  SUB
  NEG    =  LOC 0  +  EXG    +  SUB

GROUP 4:
  DMD    =  DUP 8  +  DDV    +  DMU    +  DSB

GROUP 6:
  ADI m  =  LOC m  +  PAD

GROUP 7:
  INC    =  LOC 1  +  ADD
  INL x  =  LOL x  +  INC    +  STL x
  INE x  =  LOE x  +  INC    +  STE x
  DEC    =  LOC 1  +  SUB
  DEL x  =  LOL x  +  DEC    +  STL x
  DEE x  =  LOE x  +  DEC    +  STE x
  ZRL x  =  LOC 0  +  STL x
  ZRE x  =  LOC 0  +  STE x

GROUP 8:
  CIF    =  CID    +  CDF
  CFI    =  CFD    +  CDI

GROUP 9:
  AND p  =  LOC p  +  ANS
  IOR p  =  LOC p  +  IOS
  XOR p  =  LOC p  +  XOS
  COM p  =  LOC p  +  COS

GROUP 10:
  INN p  =  LOC p  +  INS
  SET p  =  LOC p  +  SES

GROUP 11:
  LAR x  =  LAE x  +  LAS
  SAR x  =  LAE x  +  SAS
  AAR x  =  LAE x  +  AAS

GROUP 12:
  CMU p  =  LOC p  +  CMS
  TLE    =  TGT    +  TEQ
  TGE    =  TLT    +  TEQ
  TNE    =  TEQ    +  TEQ

GROUP 13:
  BRF n  =  LOC 0  +  ZEQ n
  BLT n  =  CMI    +  ZLT n
  BLE n  =  CMI    +  ZLE n
  BEQ n  =  CMI    +  ZEQ n
  BNE n  =  CMI    +  ZNE n
  BGE n  =  CMI    +  ZGE n
  BGT n  =  CMI    +  ZGT n
  ZLT n  =  TLT    +  ZNE n
  ZLE n  =  TLE    +  ZNE n
  ZEQ n  =  TEQ    +  ZNE n
  ZGE n  =  TGE    +  ZNE n
  ZGT n  =  TGT    +  ZNE n

GROUP 14:
  MRK n  =  LOC n  +  MRS
  CAL n  =  LOC n  +  CAS
  RET p  =  LOC p  +  RES

GROUP 15:
  BEG z  =  LOC z  +  BES
  BLM p  =  LOC p  +  BLS
  DUP p  =  LOC p  +  DUS
  LIN n  =  LOC n  +  STE 0
  LNI    =  INE 0
  RCK x  =  LAE x  +  RCS
```

The replacements for LIN and LNI are only equivalent if they precede the first HOL in that assembly module. The replacements for LAI and SAI are rather artificial. These instructions are most likely preceded by a LAL or LAE instruction. Then they replace the sequence:

```
LAL x  +  LAI y  =  LOL x  +  DUP 2  +  ADI y  +  STL x  +  LOI y
LAE x  +  LAI y  =  LOE x  +  DUP 2  +  ADI y  +  STE x  +  LOI y
LAL x  +  SAI y  =  LOL x  +  DUP 2  +  ADI y  +  STL x  +  STI y
LAE x  +  SAI y  =  LOE x  +  DUP 2  +  ADI y  +  STE x  +  STI y
```

The replacements for LAS and SAS would even be longer, because the size of the object to be loaded or stored must be fetched from the descriptor. If the size y is known, then LAS and SAS can be replaced by:

```
LAS    =  AAS    +  LOI y
SAS    =  AAS    +  STI y
```

APPENDIX 1. OFFICIAL EM-1 MACHINE DEFINITION

```
{ This  is an interpreter for EM-1.  It serves as  the official machine
  definition.  This interpreter must run on a machine which supports 32
  bit arithmetic.

  Certain aspects of the definition are over specified.  In particular:

    1. The representation of  an  address on the stack  need not be the
       numerical value of the memory location.

    2. The state of  the stack is not defined  after a trap has aborted
       an instruction in the middle.  For example, it is officially un-
       defined  whether the second  operand of an  ADD  instruction has
       been popped or not if the first one is undefined (-32768).

    3. The memory layout is implementation dependent. Only the most
       basic checks are performed whenever memory is accessed.

    4. The format of the mark block is implementation dependent.

    5. The format of the procedure descriptors is implementation
       dependent.

    6. The result of the compare operators  CMI etc.  are -1, 0  and  1
       here, but other negative  and  positive values will do  and they
       need not be the same each time.

    7. The shift count for SHL, SHR, ROL and ROR must be in the range 0
       to 15.  The effect of a count greater than 15 or less than 0  is
       undefined.
                                                                      }

program em1(tables,prog,output);

label 9999;

const
  t13   =  8192;        { 2**13    }
  t14   = 16384;        { 2**14    }
  t15   = 32768;        { 2**15    }
  t15m1 = 32767;        { 2**15 -1 }
  t16   = 65536;        { 2**16    }
  t16m1 = 65535;        { 2**16 -1 }
  t31m1 = 2147483647;   { 2**31 -1 }

  maxcode = 8191;       { highest byte in code address space }
  maxdata = 8191;       { highest byte in data address space }

  { mark block format }
  statd   = 6;          { how far is static link from lb }
  dynd    = 4;          { how far is dynamic link from lb }
  reta    = 2;          { how far is the return address from lb }
  mrksize = 6;          { size of mark block in bytes }

  { procedure descriptor format }
  pdargs  = 0;          { offset for the number of argument bytes }
  pdbase  = 2;          { offset for the procedure base }
  pdsize  = 4;          { size of procedure descriptor in bytes }

  dsize   = 4;          { size of double precision integers }
  rsize   = 4;          { size of reals }
  { header words }
  NTEXT   = 1;
  NDATA   = 2;
  NPROC   = 3;
  ENTRY   = 4;
  NLINE   = 5;

  escape  = 0;          { escape to secondary opcodes }
  undef   = -32768;     { the range of integers is -32767 to +32767 }

  { error codes }
  ESTACK  =  0; EHEAP   =  1; EILLINS =  2; EODDZ   =  3;
  ECASE   =  4; ESET    =  5; EARRAY  =  6; ERANGE  =  7;
  EIOVFL  =  8; EDOVFL  =  9; EFOVFL  = 10; EFUNFL  = 11;
  EIDIVZ  = 12; EFDIVZ  = 13; EIUND   = 14; EDUND   = 15;
  EFUND   = 16; ECFI    = 17; ECFD    = 18; ECDI    = 19;
  EFPP    = 20; ELIN    = 21; EMON    = 22; ECAL    = 23;
  ELAE    = 24; EMEMFLT = 25; EPTR    = 26; EPROC   = 27;
  EPC     = 28;
```