DISCLAIMER

= F O R - W O R D = >

DRAFT Report Describes Department of Energy Fortran Extensions

The Language Working Group of the Advanced Computing Committee, U.S. Department of Energy, has produced a DRAFT report entitled Fortran Language Requirements. This is the Fourth Report of the Language Working Group.

The Advanced Computing Committee was established by the Office of ADP Management, to provide a forum for the exchange of information regarding computing needs among Laboratories and Program Offices of the Department of Energy. The Advanced Computing Committee chartered a Language Working Group to study and make recommendations concerning language needs. This report describes Fortran language requirements identified by the Language Working Group.

The intent of this publication is to solicit comments on the Language Model and on the specific Language Extension Features described in this Report.

DISCLAIMER This publication in no way reflects a policy of the Department of Energy for a preferred or desired Fortran compiler. The Language Model and the specific Language Extension Features described therein are intended for review, discussion, and comment only; their implementation will not necessarily provide any Vendor with a competitive advantage in Department of Energy procurement at this time.

Copies of this report may be obtained from Loren P. Meissner (address in Mailing Area below). Ask for "LWG Report". A Comment Form is included with each copy distributed. Readers are invited to submit comments concerning their portability requirements and their opinion of the overall LWG Fortran extension project, as well as detailed comments concerning the individual features proposed as Fortran extensions.

An important motive for the work of the Language Working Group has been the need for a common Array Processing syntax to be used by the major DOE Laboratories. Other features are also included, however. These include Data Structures, Dynamic Storage, Macro Processing, Control Structures, Precision Specification, and Environmental Inquiry.

The proposed extensions are incorporated into a multi-level language model. The nucleus of this model is Fortran 77. The next level consists of features that are generally available, but for which a common syntax is needed: these include word-oriented bit data, timing functions, asynchronous input and output, and Namelist. The third level defines advanced features, including Array Processing and the other facilities described above.

Between now and the middle of 1980, a number of meetings will be held at DOE computing sites to present this proposal. The comments received will be evaluated, and recommendations will be made to the Department of Energy. One possible further action would be a "common front end" compiler implementation, which would accept extended Fortran 77 syntax and would generate code for any of several of the hardware configurations in use at DOE laboratories.

Loren P. Meissner
CSAM - 50B
Lawrence Berkeley Laboratory
Berkeley CA 94720

FIRST CLASS

PUB-301 [5 : 4 12/79] (2100)

FORTRAN STANDARDS COMMITTEE ACTIVITIES

The ANSI Technical Committee on Fortran (X3J3) met in Boston during the week of October 15 to 19. The six "action items" considered by the Committee are described below. Remember that X3J3 actions may be rescinded or extensively revised before final standardization.

1. Free Form for Source Input Defined

An alternative "Free Form" for Fortran source statement input was adopted. Record length is processor dependent, but must be at least 72 positions. A statement delimiter is an end-of-record or a semicolon. A statement label is permitted on a new statement that begins within a record. A comment is a blank character, or the rightmost characters of a record following "!" . Continuation is indicated by "&" as the rightmost non-blank character of a record, ignoring any trailing comment in the record. An incomplete character constant resumes following "&" as the leftmost character of the next (non-comment) record; a character constant cannot be continued over a trailing comment.

2. Modifications to Bit Data Type.

New operators, .BNOT., .BAND., .BOR. and .BXOR. were adopted. These operators are needed to avoid syntactic ambiguity with logical operators.

3. Data Structures.

A basic data structures proposal was adopted. A "form declaration" defines and names a form and declares the arrangement of its constituent fields. A "structure declaration" creates an instance of the form, with a specified identifier as the structure name. A structure name, or a name of a component of a structure, may be referenced. A whole structure may be referenced for input and output, assignment, or comparison; as an actual argument; or in a SAVE statement. An expression or function may produce a structure as its value.

4. Procedures for Defining the Content of Core Fortran.

A procedure was adopted, whereby all features go into the Language Extension Module by default, and can be moved into the Core only by formal action.

5. Automatic Creation of Local Arrays

A subprogram may contain an adjustable array declaration, where the array name is not a dummy argument. Thus storage may be allocated upon entry to the subprogram. Such an array must not appear in a SAVE statement; allocated storage is released upon exit from the subprogram. Such an array must not be initialized nor Equivalenced, nor in Common.

6. Significance of Blanks

A proposal for a source program form with significant blanks was discussed, but was tabled to the following meeting.

Other Proposals Considered

A global naming facility for inter-procedure communication was discussed. This facility could replace storage association, Common, Equivalence, multiple entry points, extended range of DO loops, and statement functions. It also provides an internal procedure capability. Data names as well as procedure names and form names (of data structures) are permitted. Names may be explicitly "exported" and "imported". The feature is compatible with Fortran 77.

A generalized specification for numeric precision was discussed.

Variable (within maximum) length character data was proposed. String length would be determined by assignment. Length could be inquired by the LEN function, but would not be explicitly available as a datum. Maximum length is declared, and is the same for all elements of a given array. Null strings are permitted.

Also discussed was the question of whether certain intrinsic functions should be permitted in constant expressions (e.g., at compile time).

Report of ISO Fortran Meeting Available

Selected papers from the ISO Fortran Experts' Group meeting, held at Turin, Italy during November, are compiled in a report available from Loren P. Meissner. Ask for X3J3/123.

Future Meetings of X3J3

(Further information is available from the X3J3 Vice Chairman, Martin Greenfield, MS 844a, Honeywell Information Systems, 300 Concord Rd, Billerica MA 01821.)

7 to 11 Jan 1980, San Diego CA
10 to 14 Mar 1980, San Francisco CA
12 to 16 May 1980, Aberdeen MD (Anyone planning to attend this meeting, who is not a US citizen, should notify Martin Greenfield as soon as possible.)
11 to 15 Aug 1980, Aspen CO
(Oct 1980, possibly Florida)
Nov 1980, Netherlands (ISO Fortran)

CORRESPONDENCE

Walt Brainerd, Los Alamos NM:

X3J3 has been considering a change in the status of blanks in Fortran. In the current standard, blanks may appear anywhere and can never be used as meaningful separators between names and keywords. It seems inevitable that a change must eventually occur.

Groups of computer users are attempting to develop packages of software tools to provide a stable, uniform environment across different languages, different machines, and different operating systems. Such packages generally include text editors, report generators, compilers, macro processors, symbolic debuggers, command language interpreters, file system managers, directory handlers, spelling checkers, cross reference listers, and flow charters. Within such an environment, "symbols" should always be recognized in the same way since the existence of variants destroys uniformity. Readable text is one of the basic elements of such an environment; therefore, the rules that govern "symbols" within text would have to prevail. That is, blanks would necessarily be significant.

Standardization is a great boon to the development of a uniform environment. There are five programming languages currently undergoing standardization both nationally and internationally. Fortran is one of these. Others that are candidates for incorporation within a uniform environment are Cobol, PL/1, and Pascal. All of these recognize blanks. In addition there are application areas being standardized: data base, graphics, real-time. These application areas must interface with the principal programming languages. Certainly such a task will be easier if greater uniformity is provided among the principal languages.

For the next Fortran standard (early to mid 80's), X3J3 is proposing to change the card-oriented, fixed-field program form currently standardized. A free-field source form is much more in keeping with the communication media in use today. This will necessitate a period of adjustment when both forms must be handled, perhaps by a compiler switch or even by directives that are interspersed in text, such as the commonly implemented LIST, NO LIST directives. Or it could be handled by a software tool that translated from the old form to the new. The same mechanisms to effect an orderly transition could be applied to a change in the significance of blanks, if there is a general consensus that such a change should be undertaken.

The opinions of the Fortran community would be very helpful. Please check one of the following and return to X3J3 via Loren P. Meissner (address on first page of this newsletter).

Check one:

_____ Strongly opposed to significant blanks, ever.
_____ Strongly opposed to significant blanks in the next standard.
_____ Uncertain as to whether a change should be made.
_____ Favor a change eventually, but not in the next standard.
_____ Favor a change in the next standard.

Any additional comments you may have regarding this issue will be carefully considered.

Haim I. Kilov, Latvian SSR:

This comment is due to the letter by Doug Pearson (For-Word, September 1979). He mentions a certain, probably convenient form of a DO loop as allegedly not existing in any language. But look at Algol 68! The simplest example is:

```
WHILE
  read (x); x ≠ 0
DO
  process (x)
OD;
```

(The while-part in Algol-68 is an expression in the sense of Algol 68!)

I can quote with pleasure from the extraordinarily well-written paper by C.H. Lindsey: "Never mind the quality, feel the style." (From Proceedings of the Conference on Applications of Algol 68, Norwich, 1976): "... if there are two reasons for exiting a loop, and afterwards two different actions are to be taken ... then the correct way to do it in Algol 68 is to put the actions inside the while-part of the loop, where they belong ...". This conclusion was made there after discussing at length the problem of creating and maintaining a tree dictionary.

Richard A. Leeds, Santa Clara CA:

Pearson's remarks on loop constructs are a subset of the features available in "Data Basic" by Microdata. (This language is database oriented, rather than novice oriented.) Constructs of the following form are allowed:

```
LOOP
<statement block>
WHILE <expression> DO
<statement block>
REPEAT
```

Adam Boyarski, Stanford CA:

This note describes a feature that I would very much like to have included in the next ANSI Fortran standard. It does not need the addition of a new concept, but is confined to existing elements of the Fortran language. The feature can be summarized a follows:
"The name of a statement function can be used as an argument in an argument list."

With the current Fortran standard, a function whose name is used as an argument would have to be a separately compiled function. I find this additional complexity to be both distasteful and a source of error, especially in large applications. We were fortunate enough to have had a compiler with this feature, and its usefulness was very evident. The feature is especially useful for fitting routines, integration routines, and graphic display routines. I urge the ANSI committee to include this simple extension.

David L. Wilson, Madison WI:

Concerning the new binary type: please ban equivalences between binary entries with differing bit lengths. This will give implementers the freedom to start each element of a binary array at a word boundary, resulting in much faster execution.

TAKE YOUR CHOICE

LtCol William Whitaker, Report to ISO/TC 97/SC 5 on Ada: "A rigorous definition will allow control of the language to make possible wide portability. It is our intent that there be no subset or superset compilers and that a validation facility be used to assure compliance.

"Our economic analyses show that even more benefit may be attributed to the commonality resulting from exactly compatible systems than what would be attributed to the technical improvements postulated from introduction of Ada."

- -

Computer Weekly, 11 October 1979, page 7: "The proposed European Systems Language, ESL, will probably be a subset of Ada. According to sources close to the joint CII Honeywell-Bull/Siemens team working on the project, the ESL requirement can be met by an Ada subset and this is the solution likely to be recommended to the EEC."

ANNOUNCEMENTS

Software Tools User Group Formed

A collection of software tools was described by Brian W. Kernighan of Bell Labs and P.J. Plauger of Yourdon inc. ("Software Tools", Addison-Wesley 1976). These tools have been implemented on UNIX as well as a number of other systems.

The first software tools users meeting was held in June 1979. The second meeting will be held at Boulder CO on January 29, 1980. Special interest groups in four areas have been formed: the Ratfor preprocessor, Network applications, Text processing, and Text formatting. The Ratfor group is studying the feasibility of converting the tools to Fortran 77.

A Software Tools Newsletter is available from Debbie Scherrer
CSAM - 50B
Lawrence Berkeley Laboratory
Berkeley CA 94720

Fortran 77 Programming by Jerrold L. Wagener (Wiley, 1980). This book is designed around Fortran 77 programming techniques. Emphasis is on the elements of good program construction and documentation. Part I contains the fundamentals of the language; Parts II and III explore the more advanced topics.

Fortran 77 by Loren P. Meissner and Elliott I. Organick (Addison-Wesley, 1980). A thorough revision of "Fortran IV" by the same authors. Features structured programming control constructs of Fortran 77. Includes new chapters on Character Data Type, and Advanced Input and Output. A new Appendix summarizes ANSI Standard X3.9-1978 (Fortran 77).

CONCERNING FOR-WORD

This Newsletter is prepared for the U.S. Dept. of Energy under Contract W-7405-ENG-48.

Reference to a company or product name does not imply approval or recommendation of the product by the University of California or the U.S. Dept. of Energy, to the exclusion of others that may be suitable.

The editor's name and address appear in the Mailing area on the first page of each issue. Requests for additions or corrections to the mailing list should be directed to the editor.

Correspondence on all Fortran-related topics is welcomed. Especially solicited are reviews of recent Fortran textbooks, software products, literature, etc.

= F O R - W O R D =>
Fortran Newsletter
Volume 5, Number 2 -- June 1979 (Pages 5 - 8) PUB-301 (2100)

X3J3 Votes to Adopt Six New Features

At its May 1979 meeting the ANSI Technical Committee on Fortran, X3J3, tentatively adopted the following specific features for inclusion in the next revision of the Fortran Standard:

(1) A BIT data type, patterned after the CHARACTER data type of Fortran 77. (See next page.)

(2) An expanded character set, with the addition of the nine characters ! " % ; < > ? _ &
Each of these characters is in both the ASCII and EBCDIC codes and absent from the ISO reserved list. With the exception of the underscore, no specific use of these additional characters is identified at this time.

(3) Longer symbolic names -- up to 31 characters as opposed to the limit of 6 in Fortran 77. (There have been many requests for longer names in Fortran, including that from the X3J3 data base task group X3J3.1) Also names may contain the underscore character, but a name must not begin with an underscore.

(4) An IMPLICIT NONE statement. This extends the IMPLICIT statement of Fortran 77 to allow the user to "turn off" the default implicit typing.

(5) Implied DO-list variables are allowed in substring expressions in DATA statements.

(6) No distinction between upper and lower case letters (if both are supported as non-Fortran characters), except in character constants and apostrophe or H editing in a format specification.

In the language architecture area, X3J3 at this meeting further refined the definition of the core plus modules architecture for the Fortran language. In particular, *criteria for the core* were discussed, which included the following aspects:

(1) Complete and consistent
(2) Concise
(3) Portable
(4) Efficient compilation and execution
(5) Minimal duplication of functionality
(6) Features needed for use of application modules
(7) Elegant
(8) Desired and implementable by all vendors

The core would contain all features necessary for the effective use of "external facility" modules. Work is currently progressing on the possibility of allowing designers of such modules considerable freedom in devising syntax appropriate to their application areas.

(Based on X3J3 News Release, May 1979)

A Word of Caution (Editorial)

Past experience with previous published lists of X3J3 actions suggests that some implementors tend to take such lists more seriously than is warranted. Features "adopted" by X3J3 are often deleted or extensively revised before the next version of the Fortran Standard is adopted. Compiler implementations of these features may prove useful as test beds, but implementors should not make serious investment decisions on the assumption that such actions by X3J3 are final.

Loren P. Meissner
50B 3239
Lawrence Berkeley Laboratory
Berkeley CA 94720

FIRST CLASS

Summary of BIT data type features

The following features are included in the BIT data type proposal adopted by X3J3 at its May, 1979 meeting.

1. Type declaration for BIT data entities. The syntax is identical to that for CHARACTER data in Fortran 77. In accordance with the "minimal dynamic storage" policy adopted earlier, restrictions on actual arguments whose length is not known at compile time have been deleted.

2. Bit substring notation is the same as for character data. Thus bits are numbered from left to right, and the first position is 1.

3. Three classes of operators are defined.
--Concatenation of bit strings;
--.NOT., .AND., .OR., and .XOR. operations on bit strings, which produce a new string. It was noted that there is an implied definition of a "1" bit as "true" and "0" as "false", in the way the resulting string is defined.
--Relational operators on strings, based on an implied "collating sequence" with "1" greater than "0".

4. Bit assignment, with truncation and zero-padding, analogous to CHARACTER assignment, except that truncation and padding is from the left.

5. Binary, octal, and hexadecimal constants, e.g. B'0010111', O'7653107', Z'98ABC71F'.

6. Bit entities in DATA statement.

7. B, O, and Z edit descriptors

8. List-directed input and output. Input forms may include B, O, and Z; output forms are B, O, or Z at processor option, and are enclosed in apostrophes. It has been suggested that an additional "unformatted stream" input and output mode is needed for transmitting bits to or from devices or processes.

9. Bit intrinsic functions, for converting to or from integers (with user indication of the processor sign convention), for converting to or from a logical datum, shifting, pattern matching (INDEX), bit count, and string length.

Other X3J3 Actions

The proposal to permit implied DO-lists for substrings in a DATA statement was viewed primarily as an oversight in the Fortran 77 specification.

The "dual case" proposal is effective only if upper and lower case characters are both recognized, and requires that no distinction be made between the two cases in identifiers, keywords, etc. (However, in character values the distinction is maintained.)

X3J3 also tabled a proposal to make blanks significant in a Fortran program. There was discussion of the means for maintaining compatibility with older programs if such a proposal were adopted.

X3J3 also decided not to prepare an Interim Technical Report defining DOUBLE PRECISION COMPLEX data type.

Discussion (but no formal action) took place concerning arrays, general precision specification, criteria for core and modules, macro processing, source form, data structures, CASE control structure, and Global (inter-procedure) data and control information processing. It is expected that proposals in these areas will be adopted by mid 1980.

During the discussion of core and modules criteria, it was suggested that certain "Applications modules" may require new syntax and require special compiler support, and yet may be developed outside X3J3. A "Pseudo-preprocessor" approach was suggested, and will be studied further during forthcoming meetings. A set of rules would be established, describing the way of transforming Pseudo-preprocessor source into Fortran; however, there would be no requirement that an actual processor implement the transformation as a separate preprocessor step (hence "pseudo"). The transformation could involve input other than pure text (e.g., environment or system description), and could produce output other than Fortran text (e.g., predigested symbol table or run-time assertions). The advantage of the preprocessor form for describing the transformation is that arithmetic etc. need not be described for each application module, and that problems of overlap among different applications could be identified and resolved.

Also under discussion in the language architecture area is the idea of a dynamically changing "input" and "output" area for experimental and obsolete features. Core Fortran would remain stable on an approximate 10 year cycle, while a shorter period would be provided for changes in the content of the dynamic areas.

International Liaison

International processing of Fortran 77 has progressed to the point where X3J3 has been asked to recommend a US position on the final international vote to adopt Fortran 77 as an ISO standard. At the May meeting, X3J3 voted to recommend that the US vote affirmatively.

A meeting of ISO Technical Committee 97, Subcommittee 5, will be held in Italy during November 1979. It is planned that one and a half days will be available during the meeting for working group sessions on Fortran. Individuals interested in attending should contact their ISO representative.

Retain WHILE

(K W Loach, Plattsburgh NY): I wish to make a brief comment on an item appearing recently in For-Word (March 1979, page 2). The X3J3 committee took a "straw-vote" and decided *against* a "WHILE" control clause, proposing instead a "DO forever with EXIT" structure. I disagree with this tentative decision.

The proposed structure is indeed flexible and can indeed be embedded within an outer DO loop to ensure termination. However, if these were sufficient criteria, we might just as well abolish all control entirely, except for the "IF condition GO TO..." construct. This meets the same criteria of flexibility and embeddability. The point is that the "IF..." and "DO forever..." constructs are both indirect and lack the clarity of the simpler "WHILE..." construct. In fact, a "DO forever with EXIT" requires an exit condition which is the *negative* of the WHILE condition. Thus a theoretical "WHILE (C) DO..." would have to be rendered as "DO forever; IF (not C) EXIT". I find these negated control conditions loose, confusing, and irritating.

By all means introduce a "DO...REPEAT" loop, but please allow the "WHILE (C)... REPEAT" construct as well. It is a very slight extension and would allow a much greater clarity of program structure.

```
                /*  Threnody in C  */

{
for (the first time; until the last time; time after time)
     while (wondering in front of my terminal)
          if (programming is an endless task);
          else {
               why end every statement with a semicolon;
          }
}
                                              --j. knight
```

Errata list available

The article "Fortran 77" by W.S. Brainerd et al, was published last fall in Communications of the ACM. A short list of errata is now available from For-Word on request. (The list will probably be published soon in CACM.)

References

A description of Intel Fortran-80, an extension of Subset Fortran 77, appeared in SIGPLAN Notices, April 1979 (pages 64-76). The article discusses the reasons for extending the subset in various ways, and for not implementing the full Fortran 77 language.

Request for Fortran Tutorial Program

Many BASIC systems have implemented a self-paced tutorial, written in BASIC, and intended for introducing students to the language who have no previous experience with BASIC.

For-Word recently received a request for assistance in locating a similar introductory tutorial on Fortran, that would run in an on-line mode and would presumably be written in Fortran.

If anyone knows of such an implementation, or is developing one, or would like to discuss the development of one, please get in touch with

Ed Sowell (714) 773-3876
Engineering Dept., Cal State Fullerton
Fullerton CA 92634

Announcements

The Federal COBOL Compiler Testing Service, Dept. of the Navy (Washington DC 20376) announces "Fortran 78 Compiler Validation System", version 1.0, which is designed to test the Subset language. For further information, write to the address given above. The package of tests is available from National Technical Information Service, at a cost of just under $500.00

Virtual Systems (1500 Newell Ave., Walnut Creek CA 94596 -- 415 935-4944) announces a Fortran 77 Subset compiler for INTEL 8086 Microprocessor. Extensions include ROM/RAM allocation, Boolean functions, and debug assertions. Cross compilers operate on PDP-11 and LSI-11, and runtime support for 8086 is included. Other related products for Intel, Zilog, and Motorola computers are available. Contact Ralph Swearingen at Virtual Systems.

(From DATAMATION, May 1979, page 43): The Sperry Univac V77-800 Miniframe is ... designed for both commercial and scientific data processing. There is an optional new high speed 64-bit floating point processor that works in conjunction with a new globally optimized ANS '77 Fortran.... For more information, write to us at Sperry Univac Mini-Computer Operations, 2722 Michelson Drive, Irvine CA 92713, or call (714) 833-2400, ext 536.

New products from Pelorus Software (Suite 114, 1000 East Apache Trail, Tempe AZ 85381) include Execution Tracer, Structured Fortran Pre-compiler Rev2, Dump Interpreter Rev3, and Interactive Debugger Rev2. The Fortran transporter has been shelved. A brochure is available.

A Fortran preprocessor developed in Sweden by Volvo Flygmotor is being marketed by Software Consulting Services, 901 Whittier Dr., Allentown PA 18103 (215) 797-9690

Softool Corp (340 S Kellogg Ave, Goleta CA 805-964-0560) has released three new products: Interface Documenter, Memory Management Package, and Module Orderer. Tools currently available from Softool include

ANSI Fortran Checker and Error Detector
Fortran Instrumenter I and II
Documenter A
Interface Documenter
Memory Management Package
Module Orderer
IBM/DG/Fortran Transportation Package

Off-the-shelf versions of most of these products are available for IBM 360/370 and Data General computers, and some of the products are available in SEL versions as well. All products apply to Fortran programs, and some can be applied to programs written in other languages.

Concerning For-Word

This Newsletter is prepared for the U.S. Department of Energy under Contract W-7405-ENG-48.

Reference to a company or product name does not imply approval or recommendation of the product by the University of California or the U.S. Department of Energy to the exclusion of others that may be suitable.

The Editor's name and address appear in the Mailing area on the first page of this Newsletter.

Correspondence on all Fortran-related topics is welcomed. Especially solicited are reviews of up-to-date Fortran books or software products, especially those based on Fortran 77.

The mailing list currently includes about 2000 addressees. These include about 60 from Canada, 150 from the United Kingdom, and 150 from about 30 other countries. The newsletter has been published approximately quarterly since 1975.

The first issue was published in February 1975, and was titled "Newsletter of the West Coast Working Group on Fortran Langauge Development." The initial mailing consisted of 64 copies. The first mailing list was compiled by Guy de Balbine, after the Workshop on Fortran Preprocessors at Pasadena CA in the fall of 1974. The "West Coast Working Group" was initially formed by Reifer, Meissner, and others to survey structured Fortran preprocessors. From the beginning, however, a further interest was the dissemination of information concerning developments that were taking place in the Fortran language, both in the X3J3 committee and elsewhere. Reports of activities of X3J3 have continued to occupy considerable space in the columns of this Newsletter. The issue of April 1976 reported the tentative adoption by X3J3 of IF-THEN-ELSE.

Forthcoming Meetings of X3J3

July 30 - Aug 3: Santa Fe NM
Oct 15 - 19: Boston MA
Nov 12 - 16 (ISO meeting): Italy
Jan 1980: San Diego CA
Mar 1980: Northern CA
May 1980: Aberdeen MD
Aug 1980: Colorado
Oct 1980: possibly Florida

Further information is available from the X3J3 Vice Chairman, Martin Greenfield, MS 844a, Honeywell Information Systems, 300 Concord Rd, Billerica MA 01821.

Feedback from Users Needed (Editorial)

Reaction from readers to reports of proposals adopted by X3J3 are of considerable value. Particularly useful are reports of existing implementations of a feature with the same or somewhat different syntax. For example, have you used a version of Fortran that permits recursion? How has it helped with non-toy applications examples? Does your input system accept a significant blank convention, and if so, how do you maintain compatibility between significant and nonsignificant blank assumptions in different input files?

Another area where feedback could be extremely useful is surveys of applications involving proposed features such as array manipulation or bits. Hardware vendors and compiler implementors often have a perception of the usefulness of such a feature, but greater breadth of experience would be helpful in designing a standard language version.

X3J3 has recently discussed the uses of bit data. The question of whether to left justify or right justify a bit string during assignment, input or output, or conversion to other data types appears to depend upon the application. One use of bits is for unformatted stream input and output to devices such as real-time process control, telemetry, and the like; in this case, a field of consecutive bits is to be treated as an integer. In most other uses, the bits either form what is in effect a logical array (e.g., a sparse matrix map, or a graph connectivity matrix), or else they represent a pattern of some sort (image processing, for example). Are there other categories not covered by these descriptions? Are there any "unusual" bit applications you know of? Let us know how bits are used at your installation. The bit data proposal just passed by X3J3 may need to be extended or revised on the basis of user feedback. It is already obvious that additional input and output modes are probably needed for bit data.

Keep those cards and letters coming, folks.

= F O R - W O R D = >
Fortran Newsletter

Volume 5, Number 3 -- September 1979 (Pages 9 - 12) PUB-301 (2100)

November ISO Meeting to Include Fortran Sessions

During the week of November 12, 1979, Subcommitee 5 (Programming Languages) of Technical Committee 97 (Computers and Information Processing) of ISO (International Organization for Standardization) will meet in Turin, Italy. Delegates are appointed by the National Member Bodies (for the U.S., this is ANSI and its subsidiary organizations such as X3). Plenary sessions of Subcommittee 5 are scheduled for Wednesday, Thursday, and Friday (Nov. 14 - 16), and meetings of subgroups working under the auspices of Subcommittee 5 will meet on Monday and Tuesday (Nov. 12 - 13).

A meeting of the ad hoc Fortran group of experts will be held on November 12 and 13, in connection with this Subcommittee 5 meeting. This Fortran meeting will take place at the same location as the Subcommittee 5 meeting: Jolly Hotel Ambasciatori, Torino, Italy. For further information, contact your ISO National Member Body.

Canada Adopts Fortran 77 Standard

Canadian Standards Association has adopted Fortran 77 as its standard Fortran language. The document reference is CSA-Z243.18-1979. The technical content of the Canadian standard is identical to that of ANSI X3.9-1978.

Other International Activities

ANSI Committee X3 has adopted a favorable recommendation as the US position on ISO standardization of Fortran 77.

In the Real-Time Fortran area, a standard on Tasking is in preparation. It is expected that the final proposed standard will include Instrument Society of America S61.3 as a subset.

Federal Information Processing Standard for Fortran proposed

A proposed Federal Information Processing Standard for Fortran has been submitted for 45-day review. The announcement appeared in the Federal Register for the week of July 20. It incorporates the Fortran 77 standard by reference, and includes a further requirement that a facility for detecting non-standard syntax must be available in some form.

WHET is WhatSTONE?

A number of inquiries have been received concerning the meaning of the term WHETSTONE. Some investigation (thanks especially to Betty Holberton and Werner Schenk) has turned up the fact that it refers to a "benchmark" or set of programs for measuring the speed and other performance parameters for various hardware and software combinations. The "Whetstone benchmark" has come to be so called because of the method of analysis used by the original developers, B. A. Wichmann and H. J. Curnow (see "A Synthetic Benchmark", in Computer Journal for February 1976), that involved translation of the higher-level language into an intermediate form called Whetstone code.

The particular feature of interest is that this benchmark is carefully designed in an attempt to accurately reflect the performance of the computer system while executing "typical" scientific programs. There are four different performance profiles, one of which is supposed to represent the performance of scientific programs running under Fortran.

According to a recent article in a Digital Equipment Corp. newsletter, "The Whetstone benchmark has now become the industry-wide standard for measuring performance capabilities...."

Loren P. Meissner
50B 3211a
Lawrence Berkeley Laboratory
Berkeley CA 94720

FIRST CLASS

Fortran Standards Committee (X3J3) Activities

A meeting of ANSI Technical Committee X3J3 was held at Santa Fe NM, July 30 - August 3. Six "action items" were considered, the first of which related to Committee positions on proposals on the ISO-SC5 meeting agenda for November. The other five are summarized here:

Array Processing. A basic proposal for array processing was adopted. "All of the current Fortran 77 operators (arithmetic, relational, logical, character, and replacement) are extended to accept arrays as operands." Operands must be conformable arrays or scalars. "The operation is performed element by element on corresponding elements of the [operand] arrays." For assignment, it is assumed that the entire right-hand side is evaluated before any new values are stored. Also, all Fortran 77 intrinsic functions are extended to operate elemetwise on array arguments.

Special operations are also added for multiplication of matrices (and, to some extent, vectors) and for solving A * X = B ("matrix division"). Several new intrinsic functions are added.

A WHERE statement, analogous to IF...THEN, is controlled by a logical array expression. Statements in the "where block" must be conformable array assignment statements.

CASE Statement. A CASE construct proposal was adopted. The opening statement includes a control expression, which must be of one of the enumerable types (integer, logical, character, or bit). The following case blocks are introduced by a list of expressions and ranges. If the control expression matches an expression in the list, or falls within a range, the corresponding block is executed and an exit from the construct occurs. A "residual" case block is executed if none of the others applies. (The expressions and ranges in the lists for all the blocks must be disjoint, and the residual block may appear anywhere among the case blocks.)

Form of Character Constants. A character constant may be enclosed either in single quotes or in double quotes; an apostrophe edit descriptor is extended in the same manner. Either form may include the opposite character within the delimited string. This proposal was adopted.

Data Structures. A proposal for data structures was presented and discussed, but action was deferred. A form or template is defined, consisting of a collection of fields. A structure is then declared as an instance of the form. Fields within a structure are referenced by a qualified name. An entire structure may be referenced for unformatted input or output, for assignment (of all elements of an equivalent structure), and for equality tests between two equivalent structures. A structure may also be used as an actual argument, when the dummy argument is an equivalent structure. Equivalence of structures is "strict" -- i.e., the configurations and field attributes (including numeric type) must match exactly.

Significant Blanks. A proposal for significant blanks was discussed at length, and finally deferred to a following meeting. A "compiler switch" might be necessary to distinguish old and new forms; this might be the same as the source form switch (controlling continuation convention and source record length). Keywords such as GO TO, END IF, ELSE IF, END FILE would, in effect, exist in two alternative forms. There was some difficulty with the wording of the proposal; the most straightforward statement appeared to require blanks on either side of an exponentiation operator and to prohibit blanks in a character constant -- thus a few "exceptional cases" need to be added.

Other Items Discussed

Items relating to "core and modules" definition, relationships, and criteria included:
Procedures for defining core content;
Inter-module dependences; and
Criteria for syntax extensions.

A "name management" proposal was presented, relating to the idea of groups of program units. (Conceptually, all the program units of a group are compiled together, although this would not be explicitly required.) Declarations specify the scope of names (of data and of procedures) to be a program unit, a group, or the entire "executable program", and permit "import" and "export" of names from one program area to another. Most need for internal procedures, and some requirements for GLOBAL data (including some uses of COMMON) are covered by this proposal.

Dynamic storage was discussed. The particular topic was permitting an array declaration within a subprogram without a corresponding actual array parameter, where the amount of storage allocated is determined at run time. Two mechanisms were suggested, and both were viewed favorably (according to a straw vote): allocating an array on entry with deallocation on exit; and allocating in response to a special (ALLOCATE ?) statement within the subprogram. This discussion excluded "based" (pointer-controlled or multiple-copy) allocation -- a given storage area, as discussed, is either present or absent.

Future Meetings of X3J3

Oct 15 - 19, 1979; Braintree (Boston) MA
Jan 7 - 11, 1980; San Diego CA
(Call M. Greenfield at 617 667-3111, ext 2912.)

Caveat *(a Latin word meaning beware)*

As we have pointed out many times in These Pages, "X3J3 actions" should never be considered final, and implementors should not make serious investment decisions based on reports of votes taken at X3J3 meetings. X3J3 actions are often reversed, deleted, or extensively revised before final standardization.

CORRESPONDENCE (excerpted)

Kenneth A. Redish, Hamilton, Ontario, Canada: In the minutes of X3J3, I have recently noticed discussion of on-line syntax checking. We have provided an interactive Editor which allowed free-format typing of Fortran statements each of which was syntax checked at the end of its entry. This system ... allowed only about 30 active terminals. Later, we changed the system so that syntax checking is applied only when the user requests it - usually after entering a complete program. We now find we can provide adequate response time to 60 active terminals. (The users are almost all beginners, so the runs do not demand much execution time or space.) To allow the free-form input we require (1) comments to begin with an asterisk (not a 'C'); (2) continuation lines begin with a plus sign - moved to column 6 by the editor.

Doug Pearson, Mountain View CA: A few years ago I attended a workshop on structured programming. The instructors used a certain loop form they said was not available in any language. However it was very general and very simple, and I find it appealing:

```
DO
[zero or more executable statements]
WHILE condition
[zero or more executable statements]
OD
```

No doubt some people would prefer UNTIL; no doubt some people need FOR; and we're back to the real problem: too many different ways to do one thing. ... Is that really a problem?

[Ed note: The loop construct adopted by X3J3 in March 1979 (see For-Word Vol 5 No 1) includes a slight variation of this construct. Instead of WHILE condition, you may write IF (complementary condition) EXIT.]

Robert J Horn III, Concord MA: We work with a fairly wide sample of scientific programming problems, including environmental simulations, weather forecasting, scientific database, program development, and general computing support. We work way beyond the limits of the 66 Fortran standard or the 77 standard. We use RATFOR, which provides additional control structures, macros, variable length character data, upper and lower case characters, and in-line comments. We use Data General Fortran 5, with unlimited recursion, and bit variables. We would not give up these without a reasonable replacement.

Our heaviest uses of bit data are in the conversion of foreign data tapes, and in machine dependent plotter driver code. We also use bits for some telecommunications problems, and in a few instances to implement very large boolean arrays.

We use recursion for much more than just toy programs. In particular, we have three heavily used production programs where recursion is vital. One is a least cost navigation through an N by M matrix subject to moderately complex constraints. In this program, a recursive dynamic programming solution substantially improved the time and quality of the solution. (This program is used about 10,000 times annually.) The second program is a tree analysis program where recursion is used heavily. We use it for decision tree analysis and program structure analysis. (This one gets about 1000 runs annually.) The third is our key index subroutines for database activity. In another instance we use indirect recursion where two routines use each other to solve sub-problems.

A low cost recursion implementation is feasible and very much worth the cost. The only penalty we have seen is that the DG implementaion discards local variables on exit (standard-compatible) and prohibits ENTRY's. (These hinder the use of outside software.) Any new standard should permit both direct and indirect recursion.

Our use of RATFOR also provides some feedback on loop structures, array operations, and character variables. About half of the DO loops are either simple array statemens or conditional array statements. [Ed note: The remainder of this paragraph describes conditional array statements, which closely resemble the WHERE structure adopted in the X3J3 array proposal.] These usually arise because our simulations involve equations with two or more different regimes (e.g., near-field and far-field equations). When considering array operators, include an "array if" operation.

Finally, variable length strings are worth some effort. We use them extensively, and we feel that fixed length defeats the flexibility and ease of use that inspired the use of strings. We never overlap operands or result, and we set maximum size limits.

I would like to see standard Fortran reach the usability of our version of RATFOR-Fortran 5. Based on the problems and complaints I get when we use 66 Fortran, I would not switch to a new language unless it is as convenient as RATFOR.

¤ ¤

ANNOUNCEMENTS

Programming in Standard Fortran, by Alex Balfour and David H. Marwick, was published during July by Elsevier in North America and by Heinemann in the UK. This book provides comprehensive coverage of the new standard Fortran 77 suitable both for experienced Fortran users and for those with some experience of elementary programming techniques. It aims to encourage devotees of the language to write good Fortran programs and to adopt the new Standard. Bearing in mind the limitations of Fortran 77, great emphasis has been placed on structured programming and design techniques. The value of adhering to the Standard in these times of ever-increasing demands for program portability is also stressed. Many worked examples and exercises are provided. [From the book cover.]

LINPACK is a collection of Fortran subroutines which analyze and solve various systems of simultaneous linear algebraic equations. Documentation is in LINPACK User's Guide, SIAM, 1979, by J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W. Stewart (available from Soc. for Ind. and Appl. Math.). The software is available from IMSL or NESC for a nominal charge.

The LINPACK routines employ a relatively new technique for condition number estimation that gives efficient estimates which are more realistic than those obtained by most older methods.

IMSL (International Mathematical and Statistical Libraries, Inc.) develops and distributes mathematical software, as well as serving as a distribution center for software developed elsewhere (including LINPACK). IMSL publishes a newsletter, which is available from IMSL, Sixth Floor GNB Bldg, 7500 Bellaire Blvd, Houston TX, 77036.

Statistical software is also available from the BMPD project. A BMPD Newsletter is also available, from Health Sciences Computing Facility, Center for Health Sciences, University of California, Los Angeles CA 90024.

¤ ¤ ¤

We are still getting anguished requests like the following: "We are using Univac's 'ASCII Fortran' which is based on the Fortran 77 standard. While we are anxious to use all the features of this compiler, we are hesitant to do so due to portability problems. I would like to know what other vendors have a Fortran 77 based compiler. Also, does IBM ever intend to make such a compiler available? Any help you can supply will be appreciated."

Readers of For-Word are aware that a few Fortran 77 software products have been announced. According to rumors we are hearing, a number of others are just about to spring forth. (Several of these will be coming from vendors of the smaller computers; one of the larger computer manufacturers is also on the point of making an announcement. We have heard nothing definite from IBM.) Most developers of Fortran 77 products are following the (basically commendable) practice of witholding product announcements until there is an actual product available. Most users of particular systems seem to be aware of the latest rumors regarding systems, but this is of little help to those, like our recent correspondent, who want to know how widely available Fortran 77 is on other systems.

For-Word can only sit tight, along with the rest of you, and continue to extend an offer to announce any Fortran 77 compilers (and related products) as soon as we hear of them.

¤ ¤ ¤

In SIGPLAN Notices, July 1979 (page 2), there is a letter concerning "extended range" of DO and "inactive" loops -- concepts that were present in the 1966 Fortran standard and in the draft published in 1976 but not in Fortran 77. The writer speculates that "the problem of determining when a DO loop becomes inactive during execution is undecidable" [by the compiler]. However, at some small expense, it can presumably be detected during execution, and the writer suggests that this be done. Compilers can much more readily enforce the Fortran 77 rules, since the range of a DO loop is a contiguous block of statements.

¤ ¤ ¤

Concerning For-Word

This Newsletter is prepared for the U.S. Department of Energy under Contract W-7405-ENG-48.

Reference to a company or product name does not imply approval or recommendation of the product by the University of California or the U.S. Department of Energy, to the exclusion of others that may be suitable.

The Editor's name and adress appear in the Mailing area on the first page of each edition.

Correspondence on all Fortran-related topics is welcomed. Especially solicited are reviews of recent Fortran textbooks, software products, literature, etc.

KEEP THOSE CARDS AND LETTERS COMING, FOLKS!

¤ ¤ ¤

It is often better to, rather than to, in order to pedantically avoid an allegedly improper form, engage in unnecessary circumlocution, split an infinitive.

=F O R - W O R D =>
Fortran Newsletter

PUB-301
Volume 5, Number 1 *(Pages 1 - 4)* 2100/March 1979

X3J3 Refining "Core-plus-Modules" Language Architecture

(X3J3 News Release, March 1979)

Subsequent to the adoption about a year ago of "Fortran 77" as the official American National Standard Fortran, the ANSI Fortran Standards technical committee (X3J3) has been hard at work preparing for the next version of the Fortran standard. A major activity is the exploration of a "core-plus-modules" organization for Fortran for the purposes of providing an orderly environment

(1) conducive to introducing state-of-the-art language features,

(2) for identifying obsolescent features, and

(3) in which to accommodate standard facilities for major specialized application areas.

Item (3) may include, for instance, the ISA-ANSI standard facilities for process control applications, and the CODASYL Fortran data base facility. (A task force of X3J3 has been formed to further develop this CODASYL facility.) X3J3 is in the process of devising guidelines governing the form that such a proposed applications module may take, and the standard Fortran mechanisms by which Fortran programs may utilize the facilities of such a module.

In addition to applications modules the core-plus-modules architecture includes a core Fortran langauge, a module for archaic, obsolescent, or redundant Fortran features, and one or more language extension (i.e., specialized or experimental features) modules. The Fortran core is currently envisioned as being a small yet rather complete general-purpose language, and exhibiting state-of-the-art capabilities, efficient execution, and ease of implementation.

In addition to the core-plus-modules organization, X3J3 has tentatively adopted (1) a policy permitting the inclusion in Fortran of language features requiring automatic dynamic allocation of temporary storage, and (2) a new "block-DO" control structure (analogous to the "block-IF" of Fortran 77, but for repetition control rather than selection control) which includes loop-exit facilities and is extendable.

Specific language features under active consideration for inclusion in Fortran include:

(1) "free-form" program source,
(2) array operations,
(3) internal procedures,
(4) bit-string data type,
(5) record data structures, and
(6) enhanced subroutine calling mechanisms.

SIGNUM Conference Proceedings Available

A conference on the Programming Environment for Development of Numerical Software was held in Pasadena CA during October 1978. Members of X3J3 participated in the program, and interaction with IFIP Working Group 2.5 was also featured. (See also For-Word, Volume 4, No. 3, page 11.) Proceedings have been published in SIGNUM Newsletter for March 1979, copies of which may be ordered for $8.00 from ACM, PO Box 12105 Church Street Station, New York 10249.

Loren P. Meissner
50B 3239
Lawrence Berkeley Laboratory
Berkeley CA 94720

FIRST CLASS

Activities of the Languages Working Group,

Advanced Computing Committee, Dept. of Energy

It has been noted that groups of users with special interests may be of increasing importance for language development, in contrast to an earlier period when much of the development was done by hardware vendors. The Languages Working Group was formed as a joint effort of several "supercomputer" sites supported by the Department of Energy: Lawrence Livermore Laboratory, Magnetic Fusion Energy at Lawrence Livermore Laboratory, Sandia at Livermore, Los Alamos Scientific Laboratory, Sandia at Albuquerque, Lawrence Berkeley Laboratory, Argonne National Laboratory, Oak Ridge National Laboratory, and associate participation by Stanford Linear Accelerator Center and National Center for Atmospheric Research.

This group designed a 4-level extended Fortran structure:
--Level 0 is the current ANSI Fortran standard;
--Level 1 is "common extensions already in use": Asynchronous input-output, Namelist, Word-oriented bit manipulation, Time inquiries, and On-off listing controls.
--Level 2 is "required extensions": Array processing and dynamic storage allocation, Record (data) structures, Macro facility (intimately involved with compiler), Control structure extensions, Free source form, Complex double precision, Environmental inquiry, and advanced Namelist.
--Level 3 is hardware dependent and experimental features.

A draft functional specification has been prepared, but has not yet been released.

An effort has been funded by the Department of Energy, with the goal of translating this functional description into a language design. Walt Brainerd (Los Alamos Scientific Laboratory), Jean Martin (Lawrence Livermore Laboratory), and Larry Rudsinski (Argonne National Laboratory) are involved in this effort. It is hoped that the Department of Energy can fund the conversion of the language design into an actual implementation of the extended language.

X3J3 Adopts "Minimal" Dynamic Storage

Several working groups within X3J3, in their consideration of features for inclusion in a possible future Fortran standard, have encountered an apparent need for "implicit" (system-managed) dynamic storage. For example, a "matrix divide" operation appears to require an amount of temporary storage that cannot necessarily be predicted at compile time.

At its March, 1979, meeting, X3J3 adopted a proposal whose intent was to instruct working groups not to discard such features simply on the basis of an apparent dynamic storage requirement.

X3J3 Adopts General Loop Control Construct

At its meeting in March, 1979, X3J3 adopted a proposal to include a "Block-DO" looping construct.

This construct is bracketed by the keywords DO - REPEAT. An optional control clause is set off by parentheses. Thus the opening statement may have one of the following three forms, according to the current proposal.

```
DO                                  [meaning "DO forever"]
DO ( <indexing clause> )
DO ( <expression>  TIMES )
```

An indexing clause is identical to a Fortran 77 DO-loop indexing specification. It is contemplated that other control clause forms may be adopted in the future.

An EXIT statement is also provided, to permit termination of execution of a loop.

Possible extensions that were discussed, but not adopted at this time, include names for DO and IF blocks, which could permit multi-level EXIT operations; a CYCLE feature (also possibly with multi-level capability); and special exit processing. The latter feature, which would permit exit processing statements following the REPEAT statement but within the static range of the block, would be terminated by an END DO statement.

Other possible extensions include a comprehensive control clause such as FROM ... BY ... UNTIL ..., which would permit traversing a linked structure or other more general loop conditions.

In a "straw vote", the committee expressed a definite bias against a WHILE control clause. It is felt that "DO forever with EXIT" can handle those programming situations that could be served by WHILE, but more flexibly (since the test is not constrained to appear at the beginning or end of the loop), and more safely (since the test can be imbedded in a DO n TIMES loop to provide assurance of loop termination).

X3J3 Favors Internal Procedures

At the August, 1978, meeting, X3J3 adopted a proposal to include internal procedures in the future Fortran language. Reference to an internal procedure could occur within an expression (as a function reference) or in a separate statement (as a subroutine reference). Internal procedures would be permitted to have arguments, i.e., a facility beyond simple "remote code blocks" is contemplated.

The proposal was further refined at the meeting in March, 1979. However, the crucial issues of name (and label) scoping have not yet been resolved. There is some sentiment for deferring a final decision on scoping for internal procedures pending a comprehensive review of scoping in general. For example, data items with "global" scope might be introduced.

On Being "Fortran-like"

(K. Hirchert, Urbana IL -- reprinted from X3J3 Minutes): In the past, one of the guiding principles of X3J3 in considering changes to Fortran has been whether the change was "Fortran-like". With the passing of many of the more concrete criteria for judging whether something was "Fortran-like" (e.g. fixed column requirements), the feeling has developed that the criterion of being "Fortran-like" is no longer meaningful. I suggest that if we cannot identify some more basic criteria for deciding what is "Fortran-like", then we will have effectively lost our historical tie to Fortran, and the results of our efforts might as well be called the successor to PL/I or Algol 68 or an entirely new language as be called the new Fortran standard. With that idea in mind, I will offer one possible criterion for a language being "Fortran-like".

In general, there appears to be a tradeoff between the intellectual elegance of a language and the optimum performance that can be achieved in it, even with optimizing compilers. It seems that every language feature has some pathological case which is more difficult to implement than the feature is in general. "Elegant" languages typically require the handling of those cases as well as the simple ones. "Performance" languages typically disallow the funny cases. Optimizing compilers will admittedly allow the simple code to be generated for the simple cases much of the time, but factors such as the use of variables and separate compilation make compile time analysis impossible some of the time, and that means that the complicated code must be generated when the simple code might have sufficed, hurting performance. If the compiler is not an optimizing compiler, the loss of performance is more direct.

Fortran has traditionally been a "performance" language. A couple of examples will illustrate this:

1. The pathological case in the association of arguments with formal parameters comes when one argument is associated with two or more formal parameters. Rather than force the semantics of one particular type of parameter-argument association method on the Fortran implementer (e.g. call by address), Fortran has disallowed this case, thus allowing implementers to choose other forms of parameter-argument association which might be more efficient on their particular machine (e.g. call by value-result).

2. The pathological case in moving character substrings comes when the source and destination fields overlap. Again, rather than force all moves through temporaries or other "protective" approaches, Fortran disallows overlap.

I do not mean to imply that all decisions in Fortran have been strictly performance oriented. For example, incorporating backwards DO loops in the same structure as forwards DO loops was more an "elegance" decision. Still, on the whole, most decisions in the design of Fortran have weighed towards performance. My concern is that we may be forgetting this basic characteristic in our desire to "do simple things simply." May I suggest that a better goal would be to "do reasonable things reasonably." We must balance the human engineering benefits of any of our decisions against their possible performance costs. In many cases, we may be able to have our cake and eat it too, but when a choice is necessary, let us not forget what it is that we are selling. After all, if the current body of Fortran users wanted to choose elegance over performance, they might already be programming in PL/I or Algol or Pascal.

Toward a Standard for Floating-Binary Hardware

It is clear that the cause of Language Standardization would be greatly aided by the appearance of a hardware standard for floating-point arithmetic. A giant step in this direction is being taken within the IEEE Microprocessor user community.

An article in SIGNUM Newsletter, March 1979, sets forth "Principles and Preferences for Computer Arithmetic". [This is the same Newsletter that is referenced on page 1 of this For-Word.] There is apparently a real prospect that microprocessors will adopt a common form of floating binary arithmetic. (This article should be read in conjunction with two other references:
1. Coonen, J.T., "Specifications for a Proposed Standard for Floating Point Arithmetic", Oct 78, Memo UCB/ERL M78/72, Electronics Res. Lab., Univ. of Calif., Berkeley;
2. A Proposed Standard for Floating Point Arithmetic, Feb 79; available from Richard H. Delp, Signetics Corp., PO Box 9052, Sunnyvale CA 94086)

For example, on "proper rounding", the article states:

"Proper rounding is simple to define and describe, not difficult to implement and offers many desirable properties including ... the minimum average error. It would be interesting to know the arguments of the machine designers who have decided against it. It is our opinion that the rounding strategy inherent to a floating-point microprogram of a computer (or calculator) is the worst place for a machine designer to demonstrate originality, in particular if his own experience with numerical calculation is rather limited. The chaos of careless and exotic rounding strategies in our present computers seriously impedes the production of clean numerical software. Dodges and tricks, costly to develop, which might be necessary to overcome the difficulties with rounding effects on one computer may be unnecessary or even damaging on another, thus reducing the portability of programs. Therefore,... *A universally standard scheme should be adopted for the rounding in the floating-point operations on all computers.*"

(Current efforts are concentrating on binary.)

"Core-plus-modules" and Subroutine Libraries

(J. Rice, West Lafayette IN)

One important consequence of the "Core-plus-modules" concept in a future Fortran standard that needs consideration is its effect on libraries. It is possible that the Core will be large enough that libraries can be written in it; even now the best libraries are written in "intersection Fortran". However, the time will eventually come when it is unreasonable to write certain library routines in the Core language. The following questions then arise: Is a different library needed for each Module? or for every combination of Core plus Module?

It is obvious to note that library routines are normally used already compiled so one might conclude that it is sufficient to ensure that a program compiled in one Fortran environment (Core plus Module X plus Module Y) will work with a program compiled in another Fortran environment (Core plus Module Z). This would be the case if the principle of "separate compilation" is retained and I assume that it will be. Thus one might conclude that libraries would be adequately served merely by adopting standards on program linkages.

However, libraries are also used as an important source of working software in source form for constructing large programs. Thus one takes the library's Sorter or Linear Equation Solver and makes minor modifications on it to fit it into a large program package. Good coding and documentation makes this practical. No problems should arise if the package is envisaged to be in Core plus Module X and the library routine is in Core plus Module Y. That is to say, the programmer should not have to do anything special nor suffer any substantial penalty if he incorporates a library routine into his program.

The point made here is actually more general in nature. Fortran should support and facilitate a *software parts technology*. The program library is one aspect of such a technology but, as programming matures, we can expect to see other modes of operation for this technology (e.g., a Sears catalog of programs). Such a technology is already widely anticipated in scientific computing circles where it is viewed as a way to significantly reduce software costs. It is important that a future Fortran standard facilitate this technology and remove some of the barriers that now exist in Fortran 77.

SIGNUM Appoints X3J3 Representative

ACM Special Interest Group on Numerical Mathematics has agreed to provide travel support for an individual, to permit him to participate as a Voting Member of X3J3. Brian Smith of Argonne National Laboratory will serve in this role, and will be assisted by six other SIGNUM members who will attend some X3J3 meetings, will keep up to date on X3J3 activities, and will help analyze Fortran proposals.

BOOK REVIEW

Fortran: A Structured, Disciplined Style, by G. B. Davis and T. R. Hoffmann; McGraw Hill, 1978

This beginning Fortran text is based on the new Fortran 77 language. The authors must be given the highest praise for presenting only features of Fortran 77 and discouraging their readers from using nonstandard features provided by a local system. The authors provide a very accurate and complete description of the Fortran 77 language.

However, since not everyone is expected to have a Fortran 77 compiler available, the reader is always shown how to do things "the old way." This approach allows the instructor to teach Fortran 77 and still use an older system, but the presentation of multiple ways to do almost everything will create a great deal of confusion.

In most respects, the book admirably lives up to the claim in its title. Many excellent suggestions are given to improve programming style. However, there are many examples that use GO TO statements, where the Fortran 77 IF-THEN-ELSE would have been more appropriate.

[Excerpted from a review by W.S. Brainerd, to appear in Computing Reviews.]

Product Announcements

Prime Computer, Inc. (40 Walnut St., Wellesley Hills MA 02181) has announced a complete Fortran 77 compiler, featuring global optimization, fast compilation, extensive and comprehensible error diagnostics, and re-entrant coding. Extensions to Fortran 77 include Namelist, Double complex, Encode-decode, data initialization in specification statements, and continued support for Hollerith and extended DO range. Minimum trip count for DO loops is programmer selectable. For further information write W. A. Burke at Prime.

HPAC Ltd. (Cherwell House, 1-5 London Place, Gt. Clements, Oxford OX4 14T, England) announces the introduction of Fortran 77. For further information, phone Oxford (0865) 724851/2 or TELEX 887106.

Softool Corp. (Leon Presser, 340 S. Kellogg Av Goleta CA 93017) announces Fortran Instrumenter II for providing execution profiles, and a compiler which guarantees conformance to both IBM and Data General Fortran dialects.

Concerning For-Word

This Newsletter is prepared for the U.S. Department of Energy under Contract W-7405-ENG-48.

Reference to a company or product name does not imply approval or recommendation of the product by the University of California or the U.S Department of Energy to the exclusion of others that may be suitable.

The Editor's name and address appear in the Mailing area on page 1 of this Newsletter.