I have received the following statement from R.M. Worthy, Manager, Special Products, regarding Bunker-Ramo’s position on SPITBOL (SNOBOL Bulletin #7):

"In the SNOBOL bulletin of SIGPLAN (Jan, 1970), you printed a statement which declared that The Bunker-Ramo Corporation would have exclusive rights to a SNOBOL4 compiler under development by Prof. Robert B.K. Dewar of Illinois Institute of Technology. This is not true—Bunker-Ramo does not have any propriety rights to Dewar’s work. The compiler (called SPITBOL by Dewar) is, to the best of my knowledge, still the exclusive property of Prof. Dewar.

The Bunker-Ramo Corporation’s role in the SPITBOL saga was that of a gad-fly, patron, prodder, and project conscience. Bunker-Ramo’s Data Sciences Department had been using the SNOBOL languages for years and had longed for a faster system. We (the Data Sciences Department) found out that Dewar thought a SNOBOL4 compiler feasible. We persuaded B-R management to provide funds for Dewar to begin work—without strings attached. We wanted a compiler as compatible with Bell Lab’s version as possible and that would run faster and would take less core. We contributed several things to the project: we initiated it, we provided funds for the first version, we kept Dewar on the straight and narrow (he wanted to change SNOBOL4 extensively) in terms of the official versions, we provided large test programs, and we kept after him until he got a working version. We asked only for an advanced copy of the compiler (we haven’t gotten it yet).

This has been an interesting reversal of the usual industrial versus academic conflict of ideals.

In any event, as a great believer in SNOBOL4 I hope Dewar finishes the project and makes it available to all of us at some reasonable charge.

Would you print a simple statement in the next SNOBOL4 Bulletin to the effect that Bunker-Ramo’s role in the Dewar compiler project was that of initial sponsor and God-father. Bunker-Ramo does not have any propriety rights to the compiler; all rights are currently owned by Professor Dewar."

Dr. Harry J. Saal, of the Stanford Linear Accelerator Center, is studying a hardware implementation of SNOBOL4:

"We are investigating the use of microprogramming techniques to improve the hardware-software interface."
Because of the macro-implementation technique used in SNOBOL4, and my affection for the language, we have worked several months on one possible approach. Thus far we are going the straightforward route by providing microroutines for each "macro" in the implementation language. The size of the assembled system is consequently much smaller than standard implementations (since it is "assembled" not expanded) and execution speeds are expected to be significantly improved because of fewer memory fetches and a more efficient use of the true hardware of the machine by the macros.

This work was begun with the use of a 360/25 in mind, but our arrangements have been slowed by IBM policy decisions! We are now hoping to continue this work on a more sophisticated processor such as the Standard MLP-900.

We consider the translated SNOBOL internal format and the pattern and/or tree as instances of intermediate languages. Rather than interpret the macros interpreting the internal code, we could move the SNOBOL interpreter (perhaps leaving the built-in functions in macro-code) into the micro-machine. Similarly we will investigate the introduction of parts of the scanner into microcode. We are definitely not proposing rewriting the SNOBOL system using micro-order instead of machine orders, but view the definitions of the macros, the internal prefix form, and the and/or pattern trees as three different kinds of machine language for which micro-interpreters should be written."

The Bell Laboratories group has released Version 3 of SNOBOL4. It "contains a number of new features and some generalizations in both syntax and semantics. The changes are largely in the upward direction and almost all programs that run properly under Version 2 will also run properly under Version 3. Some features of Version 3 will enable programmers to write simpler and more efficient programs, and to approach some problems in a better way.

Major new features are:
1. A new data type, TABLE
2. Operator definition
3. Error control

Tables are similar to arrays in appearance but are not restricted to integer references. Any data object can be referenced, permitting simple and direct construction of
symbol tables. Tables may be thought of as "associative arrays" and used to replace awkward programming techniques involving the application of indirect referencing to constructed strings.

The treatment of functions and operators has been generalized so that operators may be redefined as functions and vice versa. For example, an operator can invoke a programmer-defined function. Commonly used functions can be replaced by operators for compactness. The set of unary and binary operator symbols has been extended to provide a supply of symbols for operator definition.

The major area of generalization in language features is the treatment of real numbers. Included are:

1. Mixed-mode arithmetic
2. Exponentiation of real numbers
3. Comparison of real numbers
4. Extension of data type conversions involving real numbers

New functions compute the remainder of integer division, dump variables during execution, and duplicate a string any desired number of times.

A number of less significant changes have been made, including new keywords for the control of input and output, defaults for arguments to certain primitive functions, a new control card, and so forth."

The manual for CAL SNOBOL4 (see SNOBOL Bulletin #6) is now available. CAL SNOBOL4 is approximately equivalent to Version 1.3 of SNOBOL4, with the following important differences:

1. Keywords are replaced by standard procedures having the same purpose.
2. Truth Predicates have been eliminated.
3. The set of standard procedures differs slight:
   -- ARRAY has only one argument (the first); all array values are preset to null.
   -- I/O associations are established using the functions INPUT(vname,fname,recl) and OUTPUT(vname,fname,prefix) On output, there is no limit to the length of the output record length.
   -- COMPILe replaces CODE.
-- No standard tracing facilities are included.

-- CONVERT has only one argument and converts from integer (or integer string) to real or vice versa.

Limitations:

-- No detailed summary of compile time diagnostics is currently available.

-- Integer arithmetic is defined only for numbers less than 10^{10}. In addition, operands of an integer multiplication must both be less than 10^5.

-- Deferred definition may only be used with variables, not full expressions. Thus a pattern can include: BREAK('abc') $ VAR ARB *VAR, but cannot include LEN(*N).

-- Arrays and programmer-defined data structures are not garbage collected.

-- Code compiled with the standard procedure COMPILe may not jump to labels in the main program, other than by calling functions.

Copies of both the manual and code may be obtained from

Paul McJones
Computer Center
University of California
Berkeley, Calif. 94720

We have been evaluating CAL SNOBOL4 here at the University of Colorado, and it appears to live up to its author's claims of speed and compact size. (They say it is 5 times as fast as the IDA release and 1/7th as large.) Our major difficulty in this evaluation has been the fact that it would only run one of our existing programs! All of the others, written by diverse users, fell foul of the restriction on unevaluated expressions. This seems the most serious limitation of the system, and should be corrected if possible. We have found that unevaluated expressions can be avoided if the programmer conciously desires to do so, but this leads to awkward constructions in many cases.

The questions raised in SNOBOL Bulletin #6 concerning the efficiency of macro coding remain unanswered. We are currently
developing more sophisticated measurement tools for tracking down inefficiencies in the macro implementation of version 3, and perhaps some improvement over the IDA release can be obtained.

I would appreciate hearing from other implementors concerning the macro language and its efficiency (or lack thereof). Address correspondence to:

Prof. William M. Waite
Dept. of Electrical Engineering
University of Colorado
Boulder, Colorado 80302

SNOBOL Bulletin
No. 8, April 70

So on the third go-around we had a language our system programmers thought they could implement because they were getting this new meta-compiler ...  

And with their metalaucus they'd soon have our compiler running in time for our revised revised revised schedule ...  

Only it didn't work too well but they could fix that with a few hexadecimal patches ...  

At revised revised revised cost--That's what they said ...  

So, I'm celebrating that along with my thirty-third birthday.