File -
Multics,
redesign
proposals.

A PROPOSAL FOR IMPLEMENTING SNOBOL4 ON MULTICS

by Richard H. Gumpertz

Messrs. Griswold, Poage, and Polonsky at BTL have developed a macro implementation of SNOBOL4 which is fairly machine independent. I have contacted Mr. Griswold and discussed the possibility of using this program to implement SNOBOL4 on the Multics system. He was quite helpful and has sent me listings and documentation of the program. I have reviewed this carefully and believe that the program would fit nicely into the Multics environment.

The program is formed from 131 macros and is written in a format acceptable to most macro-assemblers. The definition of these macros is left to the implementor for a particular machine. In addition, a few assembly parameters and tables are left for definition. Most of these macros can expand into a few lines of GE-645 machine code. The others can expand into calls to subroutines.

There are several avenues open for compiling/assembling the program on Multics:

1) Define the macros in GE-635 GMAP (this has actually been done already). Assemble under GMAP and load as a GECOS-SEG activity.
2) Extend the Multics assembler (mass, mal, ma_645, ma, alm, or whatever it ends up being called) to handle macros in a pass 0. Define the macros to run under this assembler.
3) Write a prepocessor for PL/I which could implement the compile time facilities defined for the language.
4) Write a specialized program to read the macro deck and produce assembly language source. It would implement <u>only</u> the 131 macros being used.
5) Same as 4 except produce PL/I source code.

Each method has its own distinct advantages and disadvantages. Some arguments are listed below.

1) Use of GMAP, as witnessed for eplbsa, is a pain in the neck to maintain. Also it is very difficult to produce pure code using GMAP.
2) This has the advantage that we would obtain a useful by-product from the project - a macro assembler. There is some debate, however, whether a macro assembler would be worthwhile.
3) This would probably be the most difficult route but might have the highest overall value in that we would have greatly extended the capabilities of PL/I.

most efficient code

4) This implementation would probably be the fastest to realize (other then number 1).

5) This would also be fairly good to implement, but would have the distinct disadvantage of inefficiency. The SNOBOL4 program is semi-interpretative. Statements are translated into tree structures but are executed interpretively. After much consideration, I think that the overhead involved in maintaining the lists and accessory data would greatly slow down execution. I would estimate a possible order of magnitude decrease in speed between assembly language and PL/I.

so do I

I feel that method 4 is probably the best one to choose. It would allow good execution speed, easy maintenance (at the macro level) and would not be too hard to implement.

The method I propose for implementing the system is as follows:

1) All data will be kept in a data segment, pointed to by the ap base register pair ~~point~~. This would be initiated by calling hcs_$initiate on the segment with the copy switch on. This data would use up to 64K words.

2) The tables which are used for syntax analysis would be assembled into the linkage section of the parsing routine (pointed to by lb|table). These syntax tables would take up about 3K. The reason for putting the tables in the linkage section is that one of them is writable. It would conceivably be possible to special-case this table and put the rest in the text portion of the program.

b ?

?

3) The program would run only as a bound segment. This would mean that tsxo can be used for subroutine calls and no entry sequence would be needed at internal entry points ~~ports~~ (the linkage pointer ~~portion~~ would not need resetting).

why not tsp ?

4) Index register 7 would be reserved for use as an internal stack pointer.

whose stack

5) Most data would be manipulated as double word entries known as descriptors. ?

6) Standard Multics ASCII will be used internally.

7) Initial implementation would simulate the batch system which the program originally handled. That is 3 files would be used to simulate the card reading, card punch, and line printer. As soon as the system runs and is debugged, the I/O structure will be modified to allow use from an interactive console.

how short

In conclusion, I believe that it would be possible to install SNOBOL4 on Multics in a reasonably short period. I am tempted to make a time estimate of one man-month but I know that Murphy's law would invalidate it. I would be very much interested in working on this project, and would be willing to continue into the school year if it cannot be completed before the fall.

good