TO : STRETCH Committee January 19, 1957
FROM : William J. Worlton
SUBJECT : Automatic Programming Considerations for STRETCH
SYMBOL : T - 1

MANY INDEX REGISTERS

One of the major problems in FORTRAN is to determine the proper permutations of the index registers. The magnitude of this problem can be appreciated if one realizes that the compiling time increases factorially with the number of transfers. This problem alone would apparently invalidate any thought of limiting the number of index registers; thus, allowing only geometrical addressing, even with a twelve to eighteen bit tag field would certainly hamstring the automatic programming scheme for STRETCH. This consideration does not, of course, preclude using both direct and geometric addressing for index registers.

COMPARISONS

A good deal of the time spent in compiling a code is taken up by comparison work: does this BCD unit indicate fixed point? is it allowed? does it indicate the end of a formula? etc. Bob Hughs, who has worked on both FORTRAN and K-3, said that the most valuable instruction he could think of for automatic programming would be a BCD unit compare: is a selected unit in the accumulator equal to a similarly placed unit in the memory? One can expand this idea to include "BCD unit" addressing, so that the address specifies not only the memory location, but the BCD unit to be compared. The way this is done now is rather clumsy, and unless some sort of unit compare is put on STRETCH we will be no better off.

THE BASIC BCD UNIT

The basic BCD unit should probably consist of eight bits on STRETCH instead of six, as on the 704 and associated equipment. The reason for this is that it will be desirable not only to expand the number and type of symbols from the 704 variety to what we want on STRETCH but also to have combinations in the BCD unit left over which have no meaning to the programmer, the keypunch or the machine itself as such, but which the automatic coding scheme could use as special indicators. K - 3, for example, uses nonsense BCD symbols to indicate the following:

Name	Meaning
.)	absolute value
σ	scripts
c	constant
v	variable
f	function
∈	power
ω	end of field
fxd	fixed

A GENERALIZED "EXTRACT" ORDER

In automatic programming one is forced to do a great deal of work with *parts* of words, and it is now obvious that we shall want means to get at, store or compare any part of a word. A first approximation to this might be to consider the "Extract" order as it worked on the SEAC. The SEAC was a four address machine, and comparison is a little difficult, but STRETCH might use the idea as follows, assuming seven types of extracts: (1) partial replacement, (2) partial load, (3) partial compare, (4) partial *or*, (5) partial *and*,(6) partial add, and(7) partial subtract.

Partial replacement would consist of:

a) let the T_1 field specify the location of an extractor.

b) let the T_2 field specify the location of a word, part of which is to be replaced.

c) let the T_3 field specify the location of a word, part of which is to replace a portion of the word specified in T_2.

d) in the positions where $c(T_1)$ has ones, $c(T_3)$ would replace $c(T_2)$.

T_1, T_2 and T_3 could, of course, specify the A, B, A', B', etc. registers. The other Extract type orders mentioned above would work similarly, with exceptions obvious from their names. One can think of this type of extracting as a "sifting" operation, using the ones in the extractor as "holes" through which bits may pass. The above scheme is tremendously versatile, and one can readily see ways in which it would be used in almost any program.

SHIFTS

Two shifts have been suggested: (1) Logical Right Shift, and (2) a double register Circular Shift. These would presumably be useful in manipulating part words obtained in breaking down the source code into meaningful sections.

TABLE LOOK-UP

Table look-up work (Hughs says that's all a compiler is) consumes a good portion of the compiling time, and any orders which would facilitate this process would be welcomed. IBM appears to have made some progress in this direction, since the 709 has a table look-up feature.

INPUT FORMATS

This is just a note of interest on the K-3 three card system of allowing subscripts and superscripts. It turns out that it is necessary after the cards have been read in to reduce the three card content to essentially a single line of information, with the superscript being added as just another subscript. Since K - 3 allows both a subscript and a superscript in the same column, the whole thing turns out to be an added complexity to the input (this is in addition to the bother of the handling of three cards). K - 3 is, of course, quite readable. FORTRAN gets around the above problem by having all of its statements on a single line to begin with, but pays for this in that FORTRAN arithmetic formulas do not look much like the true mathematical formulas (subscripts in parentheses, for example). The ideal situation of having (1) readable mathematical formulas (2) a single card, and (3) a consecutive sequence of symbols, is not met in either of the systems. CONCLUSION: if STRETCH is to be a really handy machine for automatic coding, it should include something of the nature of the "Voorhees Typewriter" system.