April 24, 1957

FILE MEMO: STRETCH

SUBJECT: Grouping-distribution feature

Abstract - Subject feature is an excellent one if implimented properly. The price paid for this feature, however, is increased circuitry and components, memory space in the Exchange, and reduced Exchange capacities. This memo proposes that the feature is not worth the price unless the following facilities are provided:

1. Random location of the control-word (specifically, inclusion of a control-word address in the control word)

2. Positioning of the dataword address within the control-word so that it can be used to index other instructions and can be addressed indirectly.

3. Special instructions to simplify and speed modification of both control-word and dataword addresses.

4. Even interval* spacing in memory of words within a record.

Indexing control-words and use of indirect addressing in them could substitute for some of these facilities.

The impliment of subject feature is the control-word. This word is sent to Exchange memory to monitor the execution of a READ or WRITE command. The control-word specifies the quantity of data involved, the location in main memory for the data and the location of the next control-word, if any.

There are a number of interesting and valuable facilities that subject feature can provide if designed for them. These facilities are enumerated below.

1. An input record can be split and the pieces sent to different memory destinations.

   a. A different control-word can be used for each piece.

   b. Each piece can be one or more full words in length.

*Interval size programmable.

c. The size of one piece is not dependent upon the size of any other.

d. In fixed-word-length operations the words in each piece can be spaced at even intervals in memory--a valuable facility in matrix work. (Observe that in effecting "1b" the Exchange carries a current dataword address which it increments by one for every word sent to memory. The "1d" facility permits this increment to be varied by the programmer).

e. The control-words can remember for the programmer where the pieces went in memory. It recalls by being an index register and also by being addressed indirectly.

Thus the programmer writes his computing instructions as if the records involved were always in the same location. In the case of multiple input areas, he indexes these instructions by appropriate control-words. The index address in his instructions are held constant, but when a shift to a new area is made, new control-words are moved to these index addresses. In the case of processing records that were entered into memory on a scatter-read instruction, it may be convenient to use this indexing technique, but, because references to each record may be fewer here, it may be more desirable to use indirect addressing.

2. An output record can be composed of separate pieces from divers memory locations without prior assembly in a memory output area.

a, b, c, d (Same as 1a, b, c, d).

e. In sorting, a new sequence of records in a block of output can be obtained by modifying control-word addresses of input control words. These control-words occupy fixed memory locations and are the records of the dataword counts and dataword addresses.

f. In "deleting" a record from an output block, it is merely necessary to modify the control-word address in the control-word of the preceding record.

g. In "inserting", it is merely necessary to modify the control words of the preceding record and the inserted record.

3. Successive control words, in a chain of them called by a single I/O instruction, can be in successive memory location or in random locations.

a. This is the cornerstone of "2e, f, g". In particular it permits sorting without relocation of the sorted records in memory. A short excursion into the sorting routine using this technique may be revealing.

This technique seeks to sort a single group of records contained entirely in memory. Assuming the routine is underway and a partial sequence of control words has been established, it is necessary to compare the sorting field of the next record with that of the lowest record among those previously examined. If the new record is lower, then the address in the WRITE instruction is transferred to the control-word of the new record as a modified control-word address and the address of the control word of the new record is inserted in the WRITE instruction. If the new record is equal or higher, then it is necessary to compare to the second lowest record among those previously examined. To find this record the dataword address in its control-word is used and to find its control-word it is necessary to use the control-word of its predecessor; then it is necessary to index by an appropriate constant to find the sorting field within the record*. This is repeated, progressing up the sequence until a "low" comparison is made or the end of the sequence is reached. In the "low" case an "insertion" is made and in the "end" case the control-word address in the control-word of the "end" record is modified to specify the new control word. The number of comparisons is between N-1 and N(N-1) A faster search of the sequence for the place to make an insertion could be made in binary fashion, comparing first with the middle record rather than the lowest. The location of this middle record could not be computed, however, and would have to be stored in a fixed working area.

*One might hastily conclude that the use of indirect-addressing with the COMPARE instruction would solve the problem of finding the next record in the sequence, but this would require multi-level indirect-addressing; in fact, another level would be needed with each succeeding record in the sequence. One address modification procedure coupled with single level indirect addressing will substitute quite neatly; e.g.---

I LOAD L (Preceeding control-word)

II STORE Control-word address part in L (I)

III COMPARE Indirect Address I, index (by constant)

or III COMPARE Constant (immediate), index by I

Merging two sequences is a faster routine and easier to program than sorting because it is possible to index from record to record within each sequence and also to index references to their control words for they can be stored in consecutive locations. The merge can be accomplished by "inserting" records of one sequence into appropriate spots in the other.

The 750 technique for sorting is so good that the Exchange should provide facilities to impliment it. This technique is a version of the 702-705 M-way merge wherein almost no penalty is paid by the programmer for increasing M and wherein a small penalty in extra computing time results in a large saving in number of passes. Basically the technique merges blocks of pre-sequenced records. (Pre-sequencing is done in a preliminary pass). Each block of N records upon entering the computer encounter a block of (M-1) N records already sequenced. The new block is merged with the other and a block of N records written as output. The merging is accomplished by "inserting" the records from the new block into the other. A clever scheme of relocating certain control-words is used to avoid the more time-consuming modification of control-word addresses.[1] The whole sorting technique is completely dependent upon the principle of random location of control-words.

Another sorting technique calls for fixing the sequence of control-words (i.e., fixing their control-word addresses), preferably storing them in consecutive memory locations so that the program can index from one to the next, and for modifying the dataword addresses.[2] Searching the partial sequence for the "insert" point is somewhat simpler because an indexed COMPARE instruction, that can address the dataword address in a control-word indirectly, is all that is needed. Making the "insertion", however, is so awkward that this sorting technique is impractical.

1 The necessity for this scheme may be peculiar to the 750, however.

2 This becomes awkward if the records vary in length for their the word count must be modified, too.

The way of making this "insertion" is to add a control- word to the "top" of the sequence. Move the dataword address from its predecessor to it. Obtain a new dataword address for the predecessor from its predecessor, and continue until the "insert" spot is reached. "Insert" by putting the new dataword address into this spot

Another sorting technique is a variation of this one in which the control words themselves would be relocated into consecutive memory locations. The "insert" problem is similar.

A sorting technique in which the "insert" principle is not used would be that in which the lowest item is found and its address is used as the dataword address of the first output control word; then the second lowest is found, etc. Here control words can occupy consecutive memory locations. The number of comparisons is fixed at $\frac{N(N-1)}{2}$.

b. Random location of control-words is valuable in file maintenance applications where it is desired to make insertions and deletions in a blocked master file particularly if keeping output block length constant is a requirement.

Preliminary studies of memory requirements for this work demonstrate that if N is the number of records per block (both in the master file and in the transaction file) and M is the number of words per record, then no more than 4N[1] storage locations are needed for record control-words and 4MN locations for record storage. These are not excessive requirements particularly when one observes that memory-to-memory transfer of the records or control-words is not done.

The technique considered here is to place a block from the master file into memory and also one from the transaction file. Insertions and deletions are accomplished by manipulating control-word addresses thereby forming an output chain. Control-words for deleted records would be formed into another chain.[2] When the output chain has N elements a WRITE instruction would be issued using it. This chain is then available for the next READ instruction, when more master records or more transactions are required. A long group of deletions

1N words for master records plus N for transactions plus N for output plus N for deleted records equals 4 N.

2Possibly the "deleted" control words could be absorbed into one of the other chains and memory requirements reduced to 3 N.

could call for input before an output chain is completed, but in this case the deleted-record chain can be used for input. (A long group of insertions completes the output chain as fast as it consumes transactions so that output chain will be ready for input in this case). No initialization of control-words need be done.

Another technique would provide two sets of control words for the master record - one for input and one for output. The data-word addresses of the output set are then modified (holding the control-word addresses constant) by the dataword addresses of the input. When insertions and deletions have been made including the last input item, the output block would be written and the next master block read. Thus block length will vary. To put some limit to memory requirements it is necessary to adopt some convention such as forming a new block if the number of consecutive insertions exceeds N. Holding block length constant would increase memory requirements significantly. For example; if N-1 deletions were made in each of N consecutive blocks, then all N blocks would have to be stored before an output set of control-words would contain N elements. In making an insertion one hopes to alter only two control words- the predecessor and the inserted word; and in deleting one desires to alter only the predecessor. Instead have one must set up a whole new set of control words.

4. Control words can be indexed

An alternative to the ld facility is to provide for the indexing of control words. This solves the memory problem in the case where one control-word for each word of input or output is used; a single control word is used instead and its dataword address part is indexed with each use.

Another application for indexing the dataword address part of the control-word lies in the use of multiple input areas for single files; multiple output areas, too. Here the index registers used for the control-words are also used for the computing or processing instructions.

5. The address in the control-word may be indirect.

a. This enables the programmer to achieve the effect of modifying addresses in the control word without actually doing so.

b. This may also be of use to the supervisory program for it would not need to be aware of the formats peculiar to each problem in order to assign memory.

6. A maskable "interrupt" can be provided on the occasion of the word count in chosen control-words going to zero. This facility makes it possible to solve the classic tape search problem by examining the identification field before the full record has entered memory from tape. The "interrupt" tells the program when to examine. If the record is not desired the remainder of it does not enter memory (occupy memory bus time). Another version of this facility makes possible an input area, one record length in size, to handle blocked input, many records long.

7. Use of the control-word to index instructions is of value in overlapping the input, output and computing of a single program. Here the control-word currently being used for input will next be the index register used by the computing instructions and still later will become the control-word for the output. Other uses of the control-word for indexing occur in a broad miscellany of problems.

## SUMMARY

One should observe that the results obtained by subject feature can also be obtained by relocating data in memory. The relocation technique, a good one for the 705, is a poor one for Sigma, which is memory limited*. The penalty, then for omitting subject feature from a Sigma system is to substantially increase total problem time because of the extra memory transfers required. A better balance between memory and the Harvest computer exists, however. Thus, subject feature is not a prime necessity for the Harvest. Nevertheless the same reasons for providing indexing (as an alternative to relocating data in memory) for the Harvest are pertinent here; the programming, speed, and memory advantages should not be taken lightly.

One of the principal advantages of subject feature is a time saving. A single WRITE instruction can ouput hundreds of records from an equal number of different locations, saving thereby hundreds of instruction-interpretation cycles in the computer.

Sorting and file maintenance can be speeded and simplified by use of the feature. Block lengths can be held constant.

Unless the facility of even interval spacing of words within a record is provided, subject feature loses its value when the record length is one word. This is a frequent occurance particularly in matrix algebra.

Critical to the advantages of subject feature are the facilities enumerated in the abstract.

Jack C. Gibson

JCG/jv

*Sigma can calculate upon data faster than it can get it from memory.