Sorting on Electronic Computer Systems*

EDWARD HARRY FRIEND

New York Life Insurance Co., New York, N. Y.

INTRODUCTION

The efficient utilization of an electronic computer system for the sorting of large amounts of data is an important step toward helping electronic equipment reduce the man hours required to process scientific investigations as well as business transactions.

Sorting is one of the basic operations which is indispensable to most business and scientific data processing procedures. Firstly, it makes possible the collation, inside a limited capacity high speed memory, of two or more input (usually magnetic tape) files with common control fields, thereby permitting the operation of one input on another in smooth fashion. All non-random file maintenance type procedures utilize this approach. Secondly, it facilitates ready reference to any single item in a large file of information. If a large file is stored on many reels of magnetic tape, the proper tape reel may be selected with an indexing system, provided the file is ordered.

Most sorting techniques utilized for the ordering of large quantities of data fall into one of two general categories, "Sorting by Merging" and "Radix Sorting". Generally, both Sorting by Merging and Radix Sorting may be accomplished in one, two, or three stages, depending on the absence or presence of three types of storage media, commonly referred to as low speed (usually magnetic tape), intermediate speed (usually magnetic drum), and high speed (usually magnetic core) memory. Since it is of major concern to sort large quantities of data, it shall be assumed that the low speed medium, usually magnetic tape is always available. At least one of the other two media will normally be available for program storage and intermemory data manipulation and transmission. Unless otherwise stated a high speed random access memory will be assumed.

It is well to emphasize in advance that most business and scientific data are comprised of items of considerable length in characters or "words" (groups of characters handled as a unit). These items are sorted with respect to their "control field" (or "key") but it is essential that the usually lengthy "associated" (or "satellite") information in each item follow the item's control field throughout the sorting procedure. Sometimes, to expedite processing, it is advantageous to separate the control field from the associated information, temporarily, inside the higher speed memory but programmed contact must always be maintained to permit subsequent processing of the item as a single unit on the lower speed memory medium. Particular reference to the special handling of an item's associated information is made in the body of the paper in the section on "Internal Sorting" only, but the problems surrounding the proper handling of the associated information should be borne in mind at all times.

* Presented in excerpt form at the meeting of the Association, September 14–16, 1955.

(All parts of this paper except the one dealing with "Multi-Reel Sorting" are concerned with the sorting of an amount of data which may be contained on one reel of tape and thus completely sorted without manual intervention.)

ONE REEL SORTING

A. *Sorting by Merging*

The Fundamentals of Sorting by Merging

According to the ACM Glossary, merging is the "producing of a single sequence of items, ordered according to some rule, (i.e. arranged in some orderly sequence), from two or more sequences previously ordered according to the same rule, without changing the items in size, structure, or total number". Clearly, if a single ascending sequence of numbers 2, 13, and 19 in one stream is merged with a single ascending sequence of numbers 5, 14 and 17 in another stream, a stream with one sequence is produced. This is a case of Sorting by Merging in one pass over the data. If, however, each stream contains two ascending sequences of numbers such as 8, 26, 37, 6, 22, and 2, 5, 30, 43, 48, 19, 35, two passes over the data will be required to sort by merging. The first pass will produce two new streams each containing one ascending sequence, i. e. 2, 5, 8, 26, 30, 37, 43, 48 and 6, 19, 22, 35. A second pass will produce the completely sorted stream. A merging pass may form ascending or descending sequences with the same results (providing consistency is maintained). Further discussion of sorting techniques will apply equally well to either ascending or descending order.

It should be apparent that any merging pass over two streams of data having an equal number of sequences will reduce the total number of sequences by $\frac{1}{2}$. Thus, if a number of data sequences, n, are distributed evenly on two magnetic tape reels, the number of passes, p, required to order the data under a two stream merging sort is the smallest integer equal to or greater than $\log_2 n$. (Expressed algebraically, $p = [\log_2 n]$, where the brackets imply "the smallest integer equal to or greater than".)

Generally, if n sequences are distributed evenly on k magnetic tape reels, the number of passes required to order the data under a k-stream merging sort is the smallest integer equal to or greater than $\log_k n$. (If the number of sequences are not evenly distributed, the required number of passes is $1 + [\log_k n']$ where n' is the number of sequences on the stream with the highest number of sequences.) Clearly, the number of passes required to sort by merging is related to the number of sequences, not the number of items, per stream. The number of items per stream has a direct effect, however, on the number of operations required to complete a pass.

The Role of Internal Sorting in Sorting by Merging

On an electronic computer system equipped with magnetic tapes which serve as input and output streams to a higher speed memory, the most advantageous approach toward Sorting by Merging is one in which the first pass of the data through the higher speed memory is utilized for the production of long sequences internally, thereby reducing the subsequent number of merging passes required to complete the sort. Techniques for Internal Sorting are discussed separately.

If, on a $2k$ tape reel system, one input reel containing N randomly distributed items is read into a higher speed memory in sections of F items, sorted internally, and each sorted section (now a sequence) written out alternately on k output reels, the number of k-reel merging passes required to complete the sort is $[\log_k N/F]$. If the higher speed memory is not utilized for Internal Sorting, the expected number of merging passes required to effect a sort of N items is $[\log_k (N + 1)/2]$ since under random assumptions N items form $(N + 1)/2$ sequences.

The advantage gained from sorting sections of F items internally during an initial pass prior to a k-tape merge may be measured by comparing the time required to complete the internal sorting pass with the time required to complete $\left[\log_k \left(\frac{F}{2} \cdot \frac{N+1}{N}\right)\right]$ or, for large N, $[\log_k F/2]$ merging passes. (Of course, if the data are not random and sequences approaching length F already exist in the data, nothing is to be gained from Internal Sorting.)

In order to evaluate the time required to complete any pass of tape stored data through a higher speed memory, the time required to read, write, and rewind tape reels as well as the time required for internal processing must be considered. If a system is designed to read and write simultaneously, writing time may be ignored. If a system is designed to permit reading in one direction and writing in the other direction, rewinding is unnecessary. (The items on the initial input tape reels may be read in the backward direction, merged internally, and written in the forward direction. Each subsequent pass would involve the reading backward of the previous output reels and the writing forward of current output reels). If the system is not designed to permit reading in one direction and writing in the other direction, output reels must be rewound after each pass. Rewind time, however, is much less costly than read or write time since all tape reels may usually be rewound simultaneously and sometimes at speeds many times in excess of read and write speeds.

Advantages Gained from Item Grouping

On a machine designed to read and write variable lengthed data from or to tape, one of the most valuable techniques used for the reduction of read and write time during a pass is the grouping of items on tape. It should be recognized that the time required to read (or write) a single item on tape includes the time required to bring the tape reel to the speed necessary for reading or recording. This start/stop time becomes substantial in relation to total reading or writing time even if items are sizeable. By grouping items, not only are the number of start/stops reduced, thereby reducing significantly the read/write time, but more items may be recorded on a tape reel due to the reduction in the number of space consuming inter-item gaps. Consequently more items may be sorted in any one sorting operation. The maximum number of items which may be sorted during one sorting operation without human intervention is that number of items which may be stored in grouped fashion on one reel of tape. Several tape reels of ungrouped items may usually be contained in a grouped fashion on one reel.

It should be noted that manual intervention for reel changing is unnecessary even on the first and last pass as long as k is larger than the number of "ungrouped data" reels which may be contained on "grouped data" reels. By way of explanation, for the initial (internal sorting) pass, the "ungrouped data" reels are mounted on the k input tape stations and the internal sorting program then reads data from one reel at a time until each in its turn is exhausted. Output from this pass is grouped and distributed evenly to the k output tape reels. During the final merging pass (which is predictable through establishing that no output tape received more than one data sequence on the preceding pass) the items may be separated and written individually (without grouping) back to the same number of reels as contained the items at the outset.

Optimizing the Parameters of the Merging Pass

The number of items which can be combined into a group is sometimes dictated by the data processing system employed since on some systems data must be recorded in blocks of fixed length. Where the groups may be of variable length, the optimum number of items, G, which may be combined into a group is interdependent on the optimum number of tape reels, k, to be merged and the optimum value of F, the length (in number of items) of the internal sequences produced during the internal sorting pass. The basis for determination of these optimum values is that arrangement which will effect the complete sort in the least amount of time.

Although it is true that the larger the value of k, the smaller is $\log_k N/F$, three facts must be recognized. First, extra internal processing is required during a larger k stream merging pass since the merging logic is more complex. Second, the integral exponents of a larger k are more widely separated so that as k increases the savings increment in number of passes decreases. Third, the size of the optimum G is usually reduced and the read/write time consequently increased during a larger k stream merging pass due to the necessity for more input-output areas and more instruction space.

The optimum values for k, F, and G are dependent on each other, as well as on the memory space required for the program instructions, on whether or not the data processing system permits simultaneous reading and writing, on the number of items to be sorted, and on the effect of buffer storage (if any). Algorithms for developing the optimum values of k, F, and G based on the relative effect of the aforementioned factors (except for buffer storage which effect is considered subsequently) may be found in Appendix 1.

It should be recognized that time, not tape efficiency, is the criterion by which the optimum parameters are chosen in Appendix 1. By way of illustration, if a six tape merge will permit the sorting of a given set of data in two hours and an eight tape merge will produce the same results in 100 minutes, the former is more efficient tape-wise (consuming only 720 reel-minutes as compared with 800 reel-minutes) but the latter is faster and should be preferred on an electronic computer system capable of controlling eight tape drives.

Buffers and Sorting by Merging

The role played by buffers in Sorting by Merging can be extremely significant. Buffers are designed to permit simultaneous tape reading, tape writing, and internal processing, thus minimizing the total time required to effect a complete sort. (A computer system without buffers may be designed to effect tape reading and writing simultaneously but not during internal processing.) If a computer system employs at least one input and one output external buffer, then while the high speed memory is processing data from internal input areas to internal output areas, the input buffer reads data from an input tape and the output buffer writes data to an output tape.

On a computer system which is "balanced" for a given operation, not only are tape reading and writing effected simultaneously while overlapping internal processing, but all three functions are performed continuously, no one function requiring any longer time than any other, so that an evaluation of the time required to complete the operation may be based entirely on either tape reading, tape writing, or computer time, that is, except for the housekeeping procedures. However, the concept of a balanced system, for a given operation, is actually only a theoretical one unless the amount of internal manipulation for each item is exactly equivalent and items from two or more inputs are always processed in such a fashion that no input (or output) tape is read or written more frequently or for a longer period than any other input (or output) tape. These conditions do not prevail for a merging pass. Nevertheless, a good approximation to a balanced computer system for a specific merging pass is a practical concept which can be realized and one from which sorting efficiency can be obtained. This practical concept is discussed in a later section.

Most computer systems do not approach a balanced condition for merging passes. They are "tape limited". In the final analysis they are even unsuccessful in causing the complete overlap of internal processing by simultaneous tape reading and writing because of too few buffers or operative restrictions. (The complete overlap of internal processing by simultaneous tape reading and writing requires the continuous operation of both input and output tapes, i.e. continuous and simultaneous reading and writing.)

Buffer Deficiencies

It is well to consider the several causes for failure of a single external input and a single external output buffer to cause continuous and simultaneous reading and writing. Although it is possible for a computer system to initiate both the reading of tape to buffer and the reading of the buffer into high speed memory at the same time, (the buffer to memory transmission having a head start because of the need to accelerate the tape before reading may commence), most computers are not so designed. Generally the buffer to memory operation must be well underway, if not completed, before tape reading begins. On output, since memory to buffer transmission is a faster operation than writing from buffer to tape, it is impossible to initiate these two operations simultaneously. Tape writing must be completed before the output buffer may be refilled and then tape operations must wait while the memory to buffer operation is completed. Therefore, at the outset, no matter how completely tape to buffer and buffer to tape operations are successful in overlapping internal processing time, the input and output "buffer time" described above occurs independently of tape reading and writing time and consequently does not overlap internal processing time. If input to buffer reading may be initiated simultaneously with buffer to memory transmission, then input buffer time will not interrupt internal processing.

For purposes of further investigation into the effect of an input and an output buffer on a merging pass, the concept of "unbalance" must be introduced. Let us define "x to one unbalance" as a condition wherein the average time required to read an item from tape to buffer is x times as great as the average internal processing time per item (including buffer time, if any). It should be noted that, among other factors, both the number of tapes being merged and the number of characters per item affect the degree of unbalance. (As the number of input streams increases, the number of comparisons required to establish the low item increases in number and the degree of unbalance decreases. As the number of characters per item decreases, more input items are handled, more processing is required per read/write operation, and again the degree of unbalance decreases.) Clearly then, it is not possible to contend that for a merging pass, a given computer system operates with a specific degree of unbalance. The number of input tapes and the length of the items being sorted must also be considered.

If a computer system is unbalanced in a ratio of k to 1 for sorting items of a given length during a k tape merging pass, and if one input area is assigned to each tape, reading and writing will be simultaneous and continuous and internal processing completely overlapped except for buffer time and "forecasting" time. Before justifying this statement it is well to discuss briefly the need for forecasting. Forecasting is necessary on input when only one input area is assigned to each tape in order that the buffer will be filled (while internal processing continues) with data from that one of the k tapes whose input area is first exhausted. In view of the unpredictable order in which the items are processed, it is impossible to program in advance which tape is to be called upon for the next input, hence the need for forecasting. Forecasting is unnecessary on output because the order in which each tape is selected for writing is logically defined. See Appendix 2 for a detailed discussion of forecasting.[1]

Under the conditions as stated above, reading and writing will be continuous except for buffer time and forecasting time because the longest internal processing interval that may occur between read and write operations is that amount of time

[1] Forecasting may be eliminated by the provision of extra input areas as discussed subsequently.

which is required to process virtually all of the items in each of the k input areas, an amount of time which, by definition for k to 1 unbalance, is less than the time required to fill the single buffer from one of the k tape reels. Thus, at no time (except during forecasting and buffer to memory transmission) does reading or writing cease.

Clearly, with less than k to 1 unbalance during a k-tape merge, the longest possible internal processing interval exceeds the time required to read a group of items from one tape reel to the buffer and consequently reading and writing are not always continuous even if forecasting and buffer time are absent. (It should be noted that reading and not writing is interrupted by a long internal processing interval because each successive input item, no matter which input area it comes from, is transmitted to one output area until said output area is filled. Nevertheless, writing is interrupted but only because reading is interrupted. Data cannot move out of a computer system any faster than it moves into the system.) This interruption of tape operation during internal processing occurs despite the fact that the internal processing will often be "hung-up" waiting for the reading of a tape.

Nevertheless, the amount of reading hang-up time under less than k to 1 unbalance will be less than the amount of reading hang-up time under k to 1 unbalance and consequently under the former, less total time is required to effect a merging pass. This may be explained as follows. The required time for a merging pass under k to 1 unbalance is the total of tape reading time plus buffer and forecasting time. Under less than k to 1 unbalance, say m to 1 unbalance, the required time is tape reading time plus "excess computer time" (i.e. computer time not overlapped by tape reading) plus buffer and forecasting time. The reading time under m to 1 unbalance, however, is m/k of the reading time under k to 1 unbalance and the excess computer time is far less than $(k - m)/k$ of that same reading time, as will be seen momentarily. Therefore, the sum of reading time plus "excess computer time" for m to 1 unbalance is less than the reading time alone under k to 1 unbalance.

Further analysis of the quantity "excess computer time" is worthy of consideration. Assume for the m to 1 unbalance condition that there are G items in each of the k input areas and that it takes 1 unit of time to process one complete area, i.e. $1/G$ units per item. Assume further that it takes m $(m < k)$ units of time to read a group of items earmarked for one area from tape to buffer. If after the first mG items are processed, at least one input area has become exhausted, then only mG/G $(= m)$ units of time have passed and there will be no excess computer time. If more than m units of time pass before one input area is exhausted, then the excess computer time becomes a factor. Only if the maximum interval of $(k(G - 1) + 1)/G$ units of time are necessary to exhaust at least one area will excess computer time approach $(k - m)/m$ of the reading time under k to 1 unbalance and then only for the kG items currently being processed. Clearly the probability of such an occurrence for every set of kG items is infinitesimal.

Although it would seem that a merging pass executed under conditions of "1 to 1 unbalance" is a merging pass performed on a computer system that is balanced for the merging pass, this is not the case. It will be recalled that on a mputer system which is balanced for a given operation, tape reading and riting occur simultaneously with internal processing. On the other hand, the quirements for 1 to 1 unbalance are merely that the average time required to ad an item from tape to buffer be equal to the average internal processing me per item. Under 1 to 1 unbalance, as under all degrees of unbalance less ian k to 1, not only will the computer be hung up whenever two or more input eas are exhausted almost simultaneously, but also excess computer time will e present whenever the internal processing time exceeds the time required to fill an empty buffer. Both of these conditions are contrary to the concept of alance.

By way of review, if a computer system having one external input buffer and ne external output buffer is utilized for a k tape merging pass, and if $k + 1$ reas are set aside in memory, one for each of the k inputs and a single area for the utputs, then reading and writing should be simultaneous and continuous except or the following interruptions:

(a) buffer time, i.e. the time required for the transmission of data from memory to output buffer and sometimes from input buffer to memory.

(b) forecasting time, i.e. the time required for determination of which input tape will be the next called upon for data, so that said tape may be read into buffer while data already in memory is being processed.

(c) excess computer time, i.e. uninterrupted internal processing intervals exceeding the time required to read a group of items from one tape to the buffer (Excess computer time is absent under k to 1 or greater than k to 1 unbalance; however read/write time is that much longer than under less than k to 1 unbalance.)

With good computer system design and clever programming, not only may all of the aforementioned interruptions of continuous and simultaneous reading and writing be eliminated, but internal processing will no longer be delayed by reading "hang-up" time, and an approximation to complete balance will be achieved.

Elimination of Buffer Deficiencies

Of the four deterrents to the balanced system—forecasting time, buffer time, excess computer time, and reading hang-up time—the last two are most important. Computer systems and programs may be designed to eliminate any combination of or all four of these deterrents (as will be seen momentarily) but at considerable expense. A computer system and program sufficiently versatile to eliminate excess computer time and reading hang-up time must in a practical sense be considered successful in balancing the "sorting by merging" operation. Discussion of how such a system and program may be designed follows the more idealistic discussion of the methods applicable to eliminating all of the deterrents.

The elimination of forecasting time may be accomplished by providing k alternate input areas so that the decision to read from a specified tape is made after rather than before an input area is exhausted but without interrupting the processing. By way of explanation, assume that input areas i_1, i_2, $\cdots$ i_k and alternate input areas i_1', i_2', $\cdots$ i_k' are set aside for a k tape merge. Assume further that during housekeeping all areas except i_k' are filled from their respective tapes (with the alternate input area being filled second). The input data allocated to area i_k' remains in the input buffer. When the first input area, say i_j, is exhausted, the contents of the input buffer are transmitted to i_k' and a new group of items is read from the tape associated with the exhausted area (tape j). Internal processing continues either on the alternate area, i_j', or, to avoid address modification, on the original input area i_j after the contents of i_j' are transmitted accordingly. (Actually, it is sufficient to provide just $k - 1$ roving alternate input areas and in effect utilize the buffer as the kth alternate. For the above example, if such were the case, there would be no area i_k'. When i_j became exhausted, i_j' would be transmitted to i_j, the contents of buffer transmitted to i_j', and tape j read to buffer. Area i_j' would then be the alternate for i_k, etc.) The elimination of forecasting has no effect on the number of output areas. As before, only one output area is required.

The elimination of buffer time for a merging pass requires considerable design modification. As was the case in the elimination of forecasting, two areas are set aside for each input. Only two areas need be set aside for output. Each of these input/output areas serves as an internal buffer, i.e. groups of items are read directly therein from tape to memory and vice versa. On input a given tape is read directly into one of its two associated input buffer areas while the other input buffer area is being processed item by item, to an output buffer area. On output, one of the two output buffer areas receives merged items one at a time during internal processing while the other output buffer area is being written directly to tape. Since an output buffer area may be written to tape as soon as it is full (independent of the status of input buffer areas), two output buffer areas are sufficient.

The following describes what may transpire on a computer system designed to utilize internal buffers in their most powerful version. Each pair of input buffer areas has a single address which, when referenced by a "transmit" instruction at the outset, offers up the first item in the first of the two buffer areas related to said address. A subsequent "transmit" instruction with the same address will cause the second item to be offered in like fashion, the address counter within the buffer area having been modified automatically. Similarly, an item may be referenced by a "compare" instruction, this time, however, without affecting the automatic address. One of the outstanding features of this system is the fact that the "transmit" instruction permits transmission either to that part of high speed memory set aside to function as a work area or to that part of high speed memory functioning as an internal output buffer area. In the latter case, the automatic address counter advances for both the input and output buffer areas. When the first of two buffer areas related to a single input (or output) address is exhausted (or filled), the address counter automatically advances to the first position in the second buffer area and the first buffer area is automatically refilled (or emptied) from (or to) tape. In a less powerful version of this same computer system, each location in the buffer area is addressable and address modification is performed by the program.

On an internal buffering system there need be no distinction between the confines of internal buffer storage and high speed memory except as the buffer areas are allocated for a particular sorting application. If a system is designed accordingly, buffer capacity is flexible.

Excess computer time is the third factor which causes the interruption of continuous and simultaneous reading and writing. Its elimination may be effected "hand in glove" with the elimination of reading "hang-up" time. As will be seen momentarily, this may be accomplished with or without the use of internal buffers and consequently with or without the elimination of buffer time.

For purposes of exposition, assume that a given computer system with one input and one output external buffer is operating under a condition of 1 to 1 unbalance for a specific k tape merging pass. Assume further that the data stored on the input tapes are arranged in such an unusual order that all of the items in any given input group will be processed directly, one after another, to an output area. In effect, no merging occurs within groups because the highest item in the first input group is lower than the lowest item in the second input group, etc. for all k input groups and this condition is repeated in cyclic fashion throughout the merging pass. Disregarding forecasting and buffer time the resultant operation is completely balanced because as the G items in any input area are processed to an output area the input buffer is being filled from tape and the output buffer is being written on tape. All three processing modes are functioning simultaneously. This simultaneous processing cannot be achieved on such a system during a realistic merging pass for one of two reasons, namely reading "hang-up" time or excess computing time. In other words, either more than one input area is exhausted and internal processing is delayed because a new "read" instruction cannot be initiated while a previous one is tying up the input buffer, or the input buffer has already been filled from tape and is awaiting further action.

One solution to the problem of eliminating reading "hang-up" time and excess computing time under conditions of 1 to 1 unbalance lies in the utilization of a "reservation unit" and $k + 1$ input areas for each of the k input tapes, i.e. $k(k + 1)$ input areas. The reservation unit is a special memory storage unit capable of accepting and reserving up to $k - 1$ "read" instructions until such time as the input buffer is free and ready to act upon each "read" instruction in its turn. The number of "read" instructions in the reservation unit will fluctuate during the merging pass because in accordance with the usual merging logic each time a single input area becomes exhausted a read instruction is transmitted to the reservation unit. The $k + 1$ input areas must be present to permit the internal processing to proceed despite the fact that a read instruction is sent to the reservation unit and not executed immediately. It is impossible for internal processing to exhaust more than k of the $k + 1$ input areas associated with one input tape before the input buffer will be free to reload from this tape. Thus, under the worst conditions, while the $k + 1$st area is being internally processed, a group of items is read to the input buffer from tape in time to allow internal processing to continue uninterrupted except for buffer time. Of course, with internal buffers, the buffer time is not present. The usual number of output areas are necessary, i.e. two buffer areas if the system utilizes internal buffers, and one area if the system utilizes external buffers.

Recognition should be given to the fact that a reservation unit makes possible simultaneous and continuous reading and writing (except for buffer time) on a computer system unbalanced to any degree greater than 1 to 1 where without the reservation unit this is possible for degrees of unbalance equal to or greater than k to 1 only.

The reservation unit does not necessarily function automatically and, in fact, when external input buffers are utilized "buffer interrogation" is required. After each item is processed to the output area during the merging sort, reference is made (by way of a special instruction) to the status of the input buffer. If it is being filled from tape, the internal processing continues uninterrupted. If, however, the buffer is already filled with a group of items, an appropriate buffer to memory transmission is actuated and the next instruction in the reservation unit is initiated. With the use of internal buffer areas the reservation unit may be designed to function either with or without buffer interrogation.

The elimination of reading "hang-up" time under conditions of m to 1 unbalance (m = integer greater than 1) requires the utilization of "multiprogramming". Under multiprogramming on an external buffering system m programs are stored in high speed memory at one time, each program utilizing one input and one output buffer, all buffers capable of operating independently, and each program utilizing one reservation unit. When the reservation unit for any one program reaches capacity, control is transferred to the next program and remains with said program until its reservation unit reaches capacity, etc. Transfer of control is effected by buffer interrogation in much the same way that buffer interrogation is utilized for operating the reservation unit.

It should be noted that multiprogramming alone (i.e. without a reservation unit associated with each program) is not sufficient to eliminate reading "hang-up" time and excess computer time if transfer of control is effected whenever an input area has become exhausted and the associated input buffer is busy. It is certainly possible for all programs to be "hung-up" at the same time. Also, excess computer time will inevitably appear under less than k to 1 unbalance if each program is a k-tape merge.

A Practical Concept of the Ideal System

An electronic computer system equipped with buffer interrogation, a single input buffer, and a single output buffer and which operates under 1 to 1 unbalance is capable of eliminating excess computer time and reading hang-up time for a k-tape merge. This may be accomplished with a cleverly designed program utilizing $3k$ input areas and a single output area. (If the degree of unbalance is greater than 1 to 1, say s to 1 where s is an integer, then a similar result may be obtained with s programs, s input buffers, and s output buffers, all capable of functioning independently.)

Contrary to the usual merging logic expounded earlier, wherein each time an input area becomes exhausted an instruction is initiated to refill this area with a new group of items from tape (via buffer), this program is designed to call for the filling of the input buffer (for subsequent refilling of any one of perhaps several exhausted input areas) when the input buffer is ready. A group of items is read to the input buffer from that tape whose corresponding input areas in memory are most depleted. Buffer interrogation is utilized to determine the readiness of the input buffer and forecasting is utilized to determine which tape should be read to the input buffer. No reservation unit is necessary because read instructions are deferred until the input buffer is free.

The resultant k-tape merging sort is effected in the following manner. Assume an internal sorting pass has been completed. During the housekeeping preceding each merging pass two groups of G items are read from each of the k input reels via buffer to memory. The first group of G items from reel 1 is assigned to input area 1, the second to input area $k + 1$. The first group of G items from reel 2 is assigned to input area 2, the second to input area $k + 2$. Finally, the first group of G items from reel k is assigned to input area k, the second to input area $2k$. The main routine commences with the interrogation of the input buffer to determine its status. The buffer is found ready and the forecasting routine determines (arbitrarily at the outset) that reel 1 should be read. Merging commences internally while the third group from reel 1 is read to buffer. After each low item is transmitted to the output buffer the buffer is interrogated once again to determine its status. Assuming 1 to 1 unbalance, G items will be transmitted to the output area before the input buffer is again found ready. (Output is handled as usual.)

The G items may evolve evenly from each of the first k input areas or in lopsided fashion from one of the first k input areas. In any event, when the input buffer is found ready after the third group of G items from reel 1 is read thereto (and after the G items in memory are processed to the output area), the forecasting routine is entered and that reel whose corresponding input area is most depleted is chosen to be the next reel read to buffer. Reading commences after (or simultaneously with, if the computer system permits) the transmission of the contents of the buffer to input area $2k + 1$. Following this the merging continues from the point at which it was interrupted for the buffer activity. (It should be noted that, in general, items from reel $j(\leq k)$ are read in to memory and processed from input areas j, $k + j$, and $2k + j$ respectively in rotating fashion).

The key to the successful elimination of reading hang-up time and excess computer time lies in the postponing of "read" instructions until the input buffer is available for their execution and in the fact that at no time will a given input reel be represented by less than one unprocessed group of items in memory at the time that forecasting demands the reading of a new group of items from the given reel.

B. *Internal Sorting*

General Principles of Internal Sorting

As was previously indicated, for Sorting by Merging the first pass of data through the high speed memory is most advantageously utilized for sorting sections of the data internally, thereby reducing the total number of sequences and permitting a reduction in the requisite number of merging passes.

At least six general techniques have been devised for high speed internal sorting. They may be applied independently or in combination with one another for greater advantage. A limited evaluation of the relative merits of those techniques is presented. Nevertheless, accurate comparisons for a specific computer may only be made after each method is programmed. It should be noted that one technique for internal sorting, titled "Nth Degree Selecting", is probably superior to all others when the data are random and the items are large in comparison with their control fields.

At the outset, regardless of the internal sorting technique to be applied, a section of items is read into memory from tape, either directly or indirectly by way of buffers. The maximum number of items which may be included in such a section depends on the number of characters in an item and the internal sorting technique to be utilized, i.e. the memory space required for program instructions. The optimum number of items included in such a section depends on k and G. (See Appendix 1.)

Four of the six general internal sorting techniques involve the internal manipulation of items until they form an ordered sequence. Before internal sorting commences under any one of these four techniques it is usually desirable to have the program extract the control field from each item and store it along with the item's input address in a working area. This procedure facilitates manipulation of abbreviated items made up of only a control field and an associated address rather than items made up of a control field and all of the other associated information in the input area. Sometimes it is desirable to store just the control field addresses in a working area and manipulate them. When two items are to be compared for relative size the control field addresses are inserted into a comparison instruction and the comparison instruction is duly executed. When the address of the item itself is desired, an appropriate address increment is added to the address of the control field.

After the abbreviated items are completely ordered, the appropriate original items are selected one at a time from the input area by means of the associated address, and an ordered output group is created. This group may include all of the input items or a fraction of them. The size of the output group will depend primarily on the size of the input group acceptable to subsequent merging passes (see Appendix 1) but may be dictated by the capacity of a buffer. It is essential, however, that each sequence contain an integral number of groups (except for the last sequence).

If the "control field plus address" approach is used and if the program has been cleverly designed, the control field and address of each input item will have been transmitted directly to the work area as soon as it is read in so that each subsequent item may be read into memory overlapping the previous item's control field, thereby saving valuable memory space. When the output areas are formed, the control fields will be contributed by the abbreviated items and the associated information contributed by the original items in the input area. This scheme will only be possible of course, if the control field is positioned properly in the item by prearrangement or by programming.

Internal Merging

The first and most obvious of the four techniques involving manipulations of items until they form an ordered sequence is "Internal Merging". Areas are set aside for shuttling strings of control fields back and forth until a completely sorted sequence is obtained. Optimum results are realized (both with respect to time and memory space) under two string merging since the extra program steps required for greater than two string merging are more costly than the savings in passes over the data.

There are two fundamental approaches which might be followed in designing a two way internal merging sort. The first, or von Neumann approach (named after its originator, John von Neumann) takes advantage of existing sequences in the data in an attempt to reduce the number of internal passes required to complete the sort. As was illustrated in the section on the Fundamentals of Sorting by Merging, the number of passes required to complete such a sort is $\log_2 n$ where n is the number of original sequences in the data. The following is a description of the logic involved.

The items (i.e. control fields and/or addresses) in the two streams are merged into a "receiving area" until an item in one of the streams is found to be smaller than the item last merged from its stream, a situation identified as a "single stepdown" condition. At this point merging, per se, ceases, and items are sent from the non-stepdown stream only, until it, too, yields an item smaller than the last item sent to the receiving area. This is the "double stepdown" condition. At this stage merging commences once again, only this time to a second receiving area. When the double stepdown is reached for the second time, the receiving area is switched back to the first and the cycle is repeated. When all items are transmitted to the receiving areas, those areas operate as the "initiating" areas and the two new streams are merged back again. This procedure continues until the sort is complete.

Two facts should be noted about the von Neumann logic. Firstly, each of the two initiating streams must always be monitored so that the last item in each stream may be identified. Secondly, each of the receiving areas must be large enough to accommodate all of the items in the initiating areas. Because of these two facts it is usually found that, in taking advantage of existing sequences in the data, both the number of operations required to process each item and the amount of required storage are increased over similar requirements for the alternative approach toward designing a two way internal merging sort.

In support of these observations it is well to review the alternative approach identified as the Straight Internal Merging Sort. At the outset two areas of equal size, say A and B, are assumed to contain altogether a number of items, F, such that $F = 2^e$. Two additional areas C and D, equal in size to A and B, are assumed to be available for use. On the first pass over the items (i.e. control fields and/or addresses) the following transpires: items (A_1) and (B_1) are compared, the smaller placed in C_1 and the larger in C_2; items (A_2) and (B_2) are compared, the smaller placed in D_1 and the larger in D_2; items (A_3) and (B_3) are compared, the smaller placed in C_3 and the larger in C_4, etc., with C and D alternating as "receiving areas." On the second pass over the items strings of length four are produced a follows: items (C_1) and (D_1) are compared and the smaller placed in A_1; th larger is then compared with the next item in the area from which the smalle item came, i.e. if (D_1) were lower than (C_1), then (C_1) is compared with (D_2), etc When two items have been contributed from either C or D, the remaining singl item or pair of items in the opposite area are transmitted forthwith to A t complete the sequence of four. The next sequence of four is formed accordingl in area B and the two areas A and B now alternate as receiving areas. The thir pass produces strings of length eight. The first four items in A and the first fou items in B are merged to form the first string of eight items in C, etc.

It should be observed that as each item is placed in a new area a test is made t determine whether or not this is the last item to be transmitted from the sendir area for this particular string. (For example, when strings of eight are beir produced, as each item is processed out of a sending area a test is initiated t determine whether or not it is the fourth item to be so processed from th sending area. When the fourth item is processed the item or items remaining fro the corresponding four in the other sending area are duly transmitted.) This analogous to the test in the von Neumann Merging Sort which determin whether or not a single stepdown has occurred. The completion of an entire stri is analogous to the von Neumann double stepdown.

After each new string is completed another test must be made to determi whether or not the merging pass has been completed. Clearly, this test is ma half as often for each successive merging pass since the number of complet strings is reduced by one half during each merging pass. On the average the t is made $(F - 1)/c$ times per pass. The analogous test in the von Neuma Merging Sort is the interrogation to determine whether or not a particular it is the last item to be processed from its area. This test must be made with ev item processed and hence the number of operations required to process each it is less under the straight internal merging pass than under the von Neuma internal merging pass. By way of summary, the normal operations per it processed include one comparison of control fields, one comparison to determ the possible attainment of a single stepdown condition, $c/(2^c - 1)$ of a c parison to determine the possible attainment of the end of the pass, one it transmission, and at least four address modifications. Since c passes are requi under the logic outlined, the number of operations required to effect the inte merging sort exceeds $\left(7 + \frac{c}{2^c - 1}\right) Fc$.

Under the Straight Internal Merging Sort, each of the four areas A, B, C, D need only be of sufficient size to accommodate $F/2$ of the items to be ord (since on the last pass, receiving area A (or C) overflows into area B (or without harm because there is no alternation). Under the von Neumann Mer Sort, each of four areas must be of sufficient size to accommodate F items. He more storage is required for the von Neumann logic.

Offsetting the two disadvantages to the von Neumann logic are the prob savings in passes over the data and increased flexibility. Under random assu tions the expectation is for the saving of one pass. Of course, if, as is often the case, strings of varying length are known to be present in the data, then more than one pass may be expected to be saved. With respect to flexibility it should be noted that, unless $F = 2^c$ under the Straight Merging Sort, additional logic must be included. This handicap, together with the iron-clad necessity for c passes regardless of the data arrangement, may in some instances cause the von Neumann Sort to be superior.

Inserting

The second of the four techniques involving manipulation of items until they form an ordered sequence is "Inserting". It resembles the technique used by many bridge players in sorting a bridge hand. Under Inserting, the first abbreviated item is transmitted to the first position in the work area. The control field of the second item is then compared with the control field of the first item and, if the second is higher, it is transmitted in abbreviated form to the second position of the work area. If, however, the second item is lower, the first abbreviated item is shifted to the second position in the work area and the second item is transmitted in abbreviated form to the first position of the work area. Likewise, the third item is compared with the control fields of the items in first and second position of the work area and after these items are appropriately shifted (if necessary), the third item is transmitted in abbreviated form to its proper relative position. Successive items are handled accordingly. If F items are in the ultimate sequence and if the items are in random order, the average number of non-housekeeping comparisons per inserted abbreviated item is F/4 and the average number of abbreviated items to be shifted per inserted abbreviated item is also F/4.

Inserting may most often be used to best advantage in combination with Internal Merging, depending on the size of F and the speed with which the necessary instructions are performed. If F is divided into two parts and Inserting is performed on each part separately, the average number of non-housekeeping comparisons per inserted item is $F/8$ and the average number of abbreviated items to be shifted is also $F/8$. If the two parts are then merged, there are F additional non-housekeeping comparisons and F additional shifts. If both comparisons and shifts are performed in the same time and further, if the housekeeping instructions, i.e. the instructions other than those required for non-housekeeping comparisons and shifts, require as much time under Inserting as they do under Inserting with Internal Merging, it is possible to determine the more advantageous approach by equating $F\left(\frac{F}{4} + \frac{F}{4}\right)$ with $2 \cdot \frac{F}{2}\left(\frac{F}{8} + \frac{F}{8}\right) + 2F$ and solving for F. Under these particular assumptions, the balancing value of F is 8. Thus, if internal sorted sequences are to be greater than 8, Inserting with Internal Merging is more advantageous. Otherwise Inserting alone should be the choice.

The role of Internal Merging in combination with Inserting may be increased for even greater advantage as F increases in size. If F is broken into m parts

($m = 2^i$, i an integer), there will be $\log_2 m$ merges after Inserting is performed on each part. The required number of comparisons and shifts is $m \cdot \frac{F}{m}\left(\frac{F}{4m} + \frac{F}{4m}\right) + mF$. Under the above assumptions, for F between 8 and 16, the optimum value of m is 2; for F between 16 and 64, the optimum value of m is 4; for F between 64 and 256, the optimum value of m is 8, etc.

Exchanging

The third technique involving manipulation of items until they form an ordered sequence is "Exchanging". Under Exchanging, an indeterminate (less than F) number of passes must be made over the abbreviated items in the work area. At the beginning of each pass the first abbreviated item is compared with the second and the smaller assumes first position. The larger is then compared with the third abbreviated item and the smaller of these two assumes second position. This procedure continues so that in each case the larger abbreviated item from the last comparison is compared with the next item and the smaller placed in the position next to the smaller of the previous comparison. The number of passes required to complete the sort is equal to the distance (measured in number of positions) which separates the item furthest out of place in a higher order position from its final place in a lower order position. For example, the numbers 9, 3, 4, 6, 8 will be sorted by Exchanging in one pass because no number is more than one place removed from its final position in the lower order direction. The 9, although four places out, is moving to a higher order position, not a lower order position. The numbers 3, 4, 6, 8, 1, on the other hand, will require four passes since the number 1 is four places removed from its final position in the low order direction.

Evaluation of the time required to complete an Exchanging Sort depends on the expected number of passes and the expected number of exchanges per pass. It may be shown that the expected number of passes for random data is

$$\frac{1 + F \cdot F! - \sum_{1}^{F} t!\, t^{F-t}}{F!}$$

Since the expected number of passes exceeds $F/2$ for $F \geq 5$, Exchanging requires at least twice as many non-housekeeping comparisons as Inserting and for most computers will be inferior. Of course, Exchanging may also be used in combination with Internal Merging but with the same results relative to Inserting in combination with Internal Merging.

Internal Digital Sorting

Internal Digital Sorting, the fourth technique involving manipulation of items until they form an ordered sequence, is a technique quite similar to that utilized in effecting punched card sorting. On a given pass over the data each item is dispersed to one of ten (assuming the control fields are decimal) storage areas depending on the value of the digit in the particular control field position with which the pass is associated. On a punched card sorting machine this is accomplished by providing ten bins, each one representing a single digit. If the pass in question concerns, say, the 19th column of the punched card, then, as each card is fed into the sorter from the hopper, the 19th column of the card passes over the selecting brush whose electrical contact through the hole (if any) in each card's 19th column causes the card to be carried along the proper track to the proper bin.

The important difference between Internal Digital Sorting on a high speed computer system and punched card sorting is that for the latter operation the storage bins receiving the punched cards are of "infinite" size, i.e. the operator may remove cards from any bin that fills up and set them aside. The allocation of storage space causes a problem, however, in Internal Digital Sorting.

To effect Internal Digital Sorting two main storage area sections must be set aside so that the items (in this case control fields and/or addresses) may be shuttled back and forth between the two sections during each successive pass. The problem arises in determining assignments of storage space within each section. If there is an overabundance of, say, terminal digit 5's in the control fields, then for the first pass the recipient section must be split up so that a larger part of the section is assigned to digit 5 and a corresponding smaller part to those digits having a frequency below the average.

When a plethora of storage space is available it is possible to assign to each digit an area large enough to accommodate all of the items. However, since storage space is usually at a premium, a count of the number of items which correspond to each digit is necessary before each pass may commence. For the second and subsequent passes this count is made in conjunction with the disbursement of items during the pass immediately preceeding but, of course, this is not possible for the first pass.

By way of further explanation, assume that the control field items 315, 712, 319, 413, 224, 113, 219, 319, 225, 532, 122, and 201 are to be sorted according to the foregoing principles. An initial frequency count pass reveals a single item ending in 1, three items ending in 2, two ending in 3, one ending in 4, two ending in 5, and three ending in 9. The recipient storage space is allocated accordingly so that when the first item (315) appears with digit ending 5, it is transmitted to the eighth position in the twelve position recipient area. During the first disbursement pass a frequency count is taken of the digits in second position. This new count is then used to assign sections of the alternate storage area (initially the sending area) to the digits as they will appear for the second disbursement.

It is important to recognize that the positions in the control field must be considered in order from "least significant" to "most significant" with each one requring a separate pass. Clearly, the number of passes required to effect the complete Internal Digital Sort is one more than the number of digits in the control field.

The two remaining general Internal Sorting techniques do not involve internal manipulation of items. Consequently the control fields need not be extracted to form abbreviated items in a work area. "Counting" is the first of these two.

Counting

Under Counting, after each item is read into memory its control field is compared in turn with the control fields of each of the other items already in memory. For each comparison in which the new item is equal to or higher than another item, a counter to be associated with the new item is increased by one. For each comparison in which the new item is less than another item, the counter associated with the other item is increased by one. After the last item is processed accordingly, each counter will contain a total of the number of items lower than the item associated with the counter.

Utilizing the counters for address modification an appropriate sized output group is created. If the input section of F items is to be broken into output groups of G items (F a multiple of G), only the items whose counters have totals of 0 through $G - 1$ are transmitted to the output area as the first output group. Those items whose counters have totals of G through $2G - 1$ are transmitted to the output area as the second output group, etc.

The average number of comparisons per input item is $F/2$ and consequently the average number of additions to a counter is $F/2$. On a serial machine where comparisons of control fields consume more time than single digit additions, Counting compares favorably with Inserting. However, since Counting may not be used in combination with Internal Merging, it will normally be inferior for large F.

Selecting

Just as its name implies, Selecting is a technique wherein the lowest item is selected and set aside first, the next lowest item selected and set aside second, etc. Under the most rudimentary version of selecting titled "Linear Selecting", the input section is read into memory, the first item compared with the second and the control field and address of the lower transmitted to a "linear" storage area (a section of memory set aside for storing a control field and an address). The linear storage area is then compared with the control field of the third item and if the third item is lower, its control field and address are transmitted to the linear storage area. The linear storage area is compared in turn with each of the remaining items. If the control field of any item is low, the linear storage area is replaced accordingly. In effect, the contents of the linear storage area are replaced only when the control field of a new item is lower than the control field of all previous items. At the end of the first pass over all of the items, the control field and address of the lowest item remains in the linear storage area. The item associated with this address is then transmitted to the output area and its control field in the input area is replaced by a series of 9's (or some other characters whose values are necessarily higher than the highest possible control field) so that said item will never again be selected.

This procedure continues until an output group of size G is formed. The group is then written on tape and more output groups of size G are formed until the original section of F is exhausted. For F input items, the total number of comparisons is $F\ (F - 1)$, i.e. $F - 1$ comparisons per item.

Assuming random conditions, it should be apparent in selecting the first item that the probability of replacing the contents of the linear storage area with, say, the xth item is equivalent to the probability that the xth item's control field is ower than all previous control fields, or, $1/x$. Hence, with F items the expected number of times in which the linear storage area will be replaced is $\sum_{x=2}^{F} 1/x$, a series sum which, though divergent, is quite small in relation of F.

An extension of the general Selecting technique is titled "Quadratic Selecting". Under Quadratic Selecting the F input items are split into $\sqrt{F}$ sections of $\sqrt{F}$ items, or, if F is not a perfect square, the nearest integer to $\sqrt{F}$ sections are formed and the sections made as close to the same size as possible. Further discussion assumes that F is a perfect square.

At the outset Linear Selecting is applied to each of the $\sqrt{F}$ sections. $\sqrt{F}$ linear storage areas are created, each containing the control field and address of that item which is lowest in its associated section. Next, Linear Selecting is applied to the $\sqrt{F}$ linear storage areas, with a newly created quadratic storage area receiving the lowest control field and accompanying address. At the end of this procedure the item whose address is in the quadratic storage area is transmitted to the output area and its control field in the input area is replaced by a series of 9's. The real saving over Linear Selecting is realized in the selection of the second and subsequent items since for these items an initial linear selecting pass need be made only over that one of the $\sqrt{F}$ sections which contributed the last low item to the output area.

Under Quadratic Selecting on F input items the total number of comparisons is:

$$[\sqrt{F}\,(\sqrt{F} - 1) + (\sqrt{F} - 1)] + (F - 1)[(\sqrt{F} - 1)\cdot 2]$$

i.e. approximately $2\sqrt{F} - 2$ comparisons per item.

The next step from Linear Selecting and Quadratic Selecting is Cubic Selecting. Under Cubic Selecting the F input items are divided into $\sqrt[3]{F}$ parts of $\sqrt[3]{F}$ sections of $\sqrt[3]{F}$ items. $F^{\frac{2}{3}}$ linear storage areas, $F^{\frac{1}{3}}$ quadratic storage areas and 1 cubic storage area are required to effect the sort. Under Cubic Selecting approximately $3\sqrt[3]{F} - 3$ comparisons are required per item.

Of course, the technique of Selecting may be extended to Quartic, Quintic, or in general to Nth Degree Selecting. For Nth Degree Selecting, approximately $N\sqrt[N]{F} - N$ comparisons are required per item. Theoretically, the optimum value of N with respect to the number of comparisons is infinity, but, practically, the optimum value of N will be no less than that number which will produce sections of two items. (For example, if F is 128, the optimum value of N will be no less than 7.) It will often be found that the value for N which will produce sections of two items is not optimum because the number of storage areas which must be created would be excessive, thus cutting down on space available for the input items themselves. The number of storage areas required is the largest integer less than $\dfrac{F}{\sqrt[N]{F} - 1}$.

It should be recognized that programming for Nth Degree Selecting will be all but impractical if address modification is attempted. Consequently, it is necessary and advantageous to introduce no iterative routines into an Nth Degree Selecting program, but to include individual instructions for each item stored in memory. On a one-address machine no more than three or four instructions will be required per item. On a two or three address machine the number of instructions per item will be even fewer. Clearly, then, Nth Degree Selecting is particularly applicable when the items are large (a) in comparison with their control fields and (b) with the space required for several associated instructions.

Replacement Selecting

As previously explained, all of the aforementioned techniques for internal sorting are designed to reduce the total number of sequences and thereby reduce the requisite number of merging passes. All are designed to sort the F items which may be accepted by the high speed memory and dispense the sorted items in F/G output groups of G items. However, a sometimes more powerful version of Selecting, titled "Replacement Selecting", permits the production of sequences significantly in excess of F despite the fact that the high speed memory may accommodate only F items. Replacement Selecting may be Linear, Quadratic, Cubic, or Nth Degree.

Under Replacement Selecting a new item is read into the input area replacing the low item after each selecting pass. This permits the newly read item to be eligible for selection (and eliminates the need for storing a series of 9's in the control field of the low item). Replacement Selecting will necessarily require more comparisons per item than a comparable degree of non-replacement selection because an item must satisfy two conditions before its control field and address are sent to a linear storage area. Such an item's control field must not only be lower than the control field in the associated linear storage area but it must also be equal to or higher than the control field of the previous high item transmitted to the output area. (A variation of this procedure accepts an item which satisfies the first condition and is also higher than the last item in the group just written on tape. Accordingly, if necessary, an item is inserted into its proper relative position in the output group being formed.) Advantageously, however, the average number of comparisons per item is not doubled under Replacement Selecting since if an item does not satisfy the first condition it is passed over.

The expected length of the sequences produced eludes formulation but experiment suggests that $2F$ is a reasonable expectation (especially for the second and subsequent sequences since the residue from previous sequences includes items whose control fields are on the average relatively lower than under random conditions).

One of the major disadvantages to this technique is the expectation that the usual resultant sequence will not be divisible by G. Extra programming and storage is required to provide for this eventuality. The other disadvantages lie in the unequal sequence lengths and unpredictable number of sequences.

The Magnetic Drum and Internal Sorting

It is well to note that Linear Selecting is probably the optimum internal sorting technique for a computer system which incorporates only a magnetic drum as a higher speed memory. If F items are interlaced on one circumference around the drum, then one revolution is required to select each item for the output sequence.

On a system which incorporates both a high speed random access memory and an intermediate speed drum type memory the optimum approach involves utilization of the drum's capacity during an internal sorting pass to form internal sequences of maximum length, thereby significantly reducing the total number of sequences stored on tape below the number which might have been produced by a smaller capacity high speed memory alone. To form these maximum length sequences, the high speed random access memory is called upon to produce internal sequences in the usual manner, one after another. As each high speed memory sequence is formed it is transmitted to the larger capacity drum. The many sequences on the drum are then merged into one large sequence utilizing the internal memory for temporary storage and program instructions. When the single sequence is finally produced it is written on tape and the whole procedure repeated.

C. *Radix Sorting*

The Fundamentals of Radix Sorting

"Radix Sorting" is a generalized expression for what is sometimes titled "Digital Sorting". Since "digit" implies "decimal" or "base 10" Digital Sorting would seem applicable to a data handling system only capable of distinguishing one decimal digit from another. Computer systems are sufficiently versatile to handle any radix, hence the term "Radix Sorting".

(Item grouping and buffering operations are applicable to Radix Sorting in much the same way that they are applicable to Sorting by Merging. Consequently, no specific reference is made to these devices in the following discussion.)

Summarily, if a computer system may actuate up to $2k$ tape stations, then said computer system may effect Radix Sorting to the base k without manual intervention by shuttling the items back and forth between two sets of k tape reels. By way of illustration, assume that a computer system incorporates six tape stations. Assume further that the control fields of the items to be sorted range from 0 to 1000. During the first pass all items whose control fields are a multiple of three are shipped to the first reel, all items whose control fields are one greater than a multiple of three are sent to the second reel, and all items whose control fields are two greater than a multiple of three are shipped to the third reel. Under the additional assumption that the system may not read tape backwards all reels must be rewound following this and subsequent passes. On the second pass, starting with the reading of the first reel and ending with the reading of the third reel, all items whose control fields are from 0 to 2 greater than a multiple of 3^2 or 9 are shipped to the fourth reel, all items whose control fields are 3 to 5 greater than a multiple of 9 are shipped to the fifth reel and all items whose control fields are 6 to 8 greater than a multiple of 9 are shipped to the sixth reel. The third pass is oriented on 3^3 or 27, etc. Since the range of the control fields is greater than 3^6 but less than 3^7, seven passes will effect a complete sort but an eighth pass will be required to collect the items back on to one reel of tape.

Thus, on a $2k$ tape system utilized for a Radix Sort to the base k, the pth pass processing occurs as follows. Those items whose control fields exceed a multiple of k^p by 0 to $k^{p-1} - 1$ are shipped to one reel; those items whose control fields exceed a multiple of k^p by k^{p-1} to $2k^{p-1} - 1$ are shipped to a second reel, and in general, those items whose control fields exceed a multiple of k^p by $(b - 1)k^{p-1}$ to $b \cdot k^{p-1} - 1$ are shipped to a bth reel ($b \leqq k$). The number of passes required to effect the Radix Sort is $[\log_k R] + 1$, i.e. the smallest integer greater than $\log_k R$ (where R is the range of the control fields) plus one extra pass for collecting the data on one tape reel after the sort is completed.

It should be noted that if the computer system permits reading in both directions, then rewinding is unnecessary but care must be taken with respect to the order in which the tape reels are selected to effect the "next" pass. It is clear that on a system which does not permit backward reading the $p + 1$st pass is effected by first distributing on to k new reels the contents of that tape reel which contains the items whose control fields exceed a multiple of k^p by numbers between 0 and $k^{p-1} - 1$, second distributing the contents of that reel which contains the items whose control fields exceed a multiple of k^p by numbers between k^{p-1} and $2k^{p-1} - 1$, and finally distributing the contents of that reel which contains the items whose control fields exceed a multiple of k^p by numbers between $(k - 1)\, k^{p-1}$ and $k \cdot k^{p-1} - 1$. When a system permits backward reading, however, this order must be reversed on every other pass, i.e. distribution of items must commence (assuming the $p + 1$st pass is one for which procedure is reversed) with the tape reel containing items whose control fields are between $(k - 1)k^{p-1}$ and $k \cdot k^{p-1} - 1$ in excess of k^p. The first pass, of course, is always effected by distributing the data from the single input reel according to remainders produced from division of control fields by k. The second pass is the crucial one. If an odd number of passes (including the final collection pass) are required to effect the complete Radix Sort, then the second pass must be performed in "reverse" order; if an even number of passes are so required, then the second pass must be performed in the "usual" order. All passes subsequent to the second must be performed in an order opposite to that of the preceding pass.

A simple modification of the approach outlined above may be advantageous in view of the excessive time required for some computers to perform a division. The modification involves converting all control fields to numbers with base k (assuming $k < 10$) during the first pass over the data. This facilitates internal processing, since, on each subsequent pass, reference may be made to nothing more than the value of a single digit for purposes of transmitting each item to the proper tape reel. On the last pass reconversion to base 10 may follow the aforementioned internal processing. See Appendix 3 for an arithmetic example of

Radix Sorting with conversion of control fields to the base 4 on a system capable of reading tape in both directions.

An Internal Sorting Pass and Radix Sorting vs. Sorting by Merging

Internal Sorting may be utilized to reduce the required number of radix sorting passes in much the same fashion as was described for Sorting by Merging. However, as an aid to Radix Sorting the internal sorting pass is executed after (rather than before) all of the necessary radix sorting passes have been completed, and, in so doing, replaces the collection pass previously mentioned. (For the sake of clarity it is well to note that the Radix Sort is here composed of a given number of radix sorting passes and one internal sorting pass, just as the Merging Sort may be composed of one internal sorting pass and a given number of merging passes.)

In general, when an internal sorting pass is to conclude a Radix Sort, the first and subsequent radix sorting passes consider each control field for allocating its associated item to the proper reel after the control field is divided by F_R. F_R is the largest integral span which at any point in the range of control fields, R, will include no more than F items (where F is the maximum number of items which may be sorted internally.) If $R > N$ and each control field is unique, then F_R must be made equal to F to accommodate the possibility that a span of control fields running from, say, control field "a" to control field "b" includes exactly $b - a$ items. The corresponding number of passes required to effect the radix sort will be $[\log_k R/F_R\ (= R/F)] + 1$. To sort the same items by merging requires $[\log_k N/F] + 1$ passes and Sorting by Merging is clearly superior. If $R = N$ and each control field is unique, then $F_R = F$ and both Radix Sorting and Sorting by Merging will require the same number of passes. If $R < N$ then duplicate control fields are present and $F_R < F$, that is a span of control fields running from control field "a" to control field "b" includes more than $b - a$ items. If, at worst, there may be n_r ($n_r < F$) items with a single control field, then, in order to accommodate this worst possibility, F_R must be made equal to F/n_r and the number of passes required to effect the complete Radix Sort equals $[\log_k R/F_r\ (= R\cdot n_r/F)] + 1$. Unless each control field is represented n_r times, $R\cdot n_r > N$ and consequently $[\log_k N/F] + 1$, the number of passes required to effect a Merging Sort with the same data, is smaller, and Sorting by Merging is again clearly superior.

Radix Sorting, then, is superior to Sorting by Merging only when Internal Sorting cannot be of assistance to the Radix Sort, i.e. when $R < N/F$.

The Cascading Pseudo-Radix Sort

The Cascading Pseudo-Radix Sort is a specialized type of sorting operation which is only practicable if (1) the available computer system permits backward tape reading and (2) the distribution of the item control fields is known almost exactly. Assuming these two criteria are satisfied a complete sort may be effected in $[\log_{2k-1}N/F] + 1$ passes.

The Cascading Pseudo-Radix sort is performed as follows. The single input tape is read into the computer and each item is distributed to one of the other $2k - 1$ tape reels depending on the integral value of its control field. Each of the aforementioned $2k - 1$ tape reels receives all items whose control fields fall between certain predetermined lower and upper bounds, these bounds predetermined so that the N items will be distributed evenly over the $2k - 1$ reels. The upper and lower bounds determined for any given reel do not necessarily cover the same span of control fields as the upper and lower bounds determined for another given reel unless the distribution of control fields is perfectly flat, hence the word "pseudo" in the title of the sorting method. After the first pass is completed, processing is continued on that one of the $2k - 1$ reels which received the $N/(2k - 1)$ items whose control fields are at the beginning of the range. For convenience let this be the α reel. The α reel is read backwards into the computer and each item on this reel is transmitted to one of the other $2k - 1$ tape reels (including the initial input reel) depending on the integral value of its control field. On $2k - 2$ of these reels items will be written on tape directly behind the items still recorded from the previous pass. Items written on the initial input reel are, of course, recorded at the beginning of the reel. For this second pass over the items on the α reel each of the remaining $2k - 1$ reels is assigned a new upper and lower bound predetermined so that the items on the α reel will be distributed evenly.

After the second pass over the α reel items is completed, processing is continued on that one of the $2k - 1$ reels which received the $N/(2k - 1)^2$ items whose control fields are at the beginning of the range of the original α reel items. For convenience let this be the α_1 reel. (The α_1 reel may also be a reel which contains $N/(2k - 1)$ of the original input items, say the β reel, or it may be the original input reel. If the α_1 reel is in fact also the β reel, reference to the α_1 reel implies reference to those $N/(2k - 1)^2$ items on the β reel written behind the section of $N/(2k - 1)$ items initially written on the β reel.) The α_1 reel is read backwards into the computer and each item on this reel again distributed as before. At this stage, after the third pass over the items on the α_1 reel, sections of $N/(2k - 1)^3$ items are recorded on $2k - 1$ reels, in most cases behind sections of $N/(2k - 1)^2$ and $N/(2k - 1)$ items respectively. Let us assume that $F \geq N/(2k - 1)^3$ so that we may now sort internally.

During the pass over the α_1 items, the $\alpha_{1\alpha}$ reel is conveniently chosen to be the α reel so that no other items appear on this reel except the $N/(2k - 1)^3$ items representing reel $\alpha_{1\alpha}$. Reel $\alpha_{1\alpha}$ $(= \alpha)$ is read backwards into the computer, the $N/(2k - 1)^3$ items sorted internally and written back in ordered fashion to the $\alpha(= \alpha_{1\alpha})$ reel. Reel $\alpha_{1\beta}$ is then read backwards into the computer, the $N/(2k - 1)^3$ items sorted internally and written back in ordered fashion to the α reel behind the previously sorted $N/(2k - 1)^3$ items. The next $2k - 3$ reels are processed accordingly and subsequently the α reel contains an ordered string of the items designated earlier as α_1 items. The same procedure is followed for reels α_2,

α_8, etc. and eventually the α reel will contain an ordered string made up of the $N/(2k - 1)$ items originally distributed to the α reel. The whole process is repeated for all of the items originally distributed to reels β, γ, δ, and finally the α reel will contain the original N items in one sorted sequence.

MULTI-REEL SORTING

The preceding comments regarding the sorting of "large" amounts of data have been limited to that quantity of data which may be stored on one reel of tape. If during a single sorting operation an attempt is made to sort more data than may be contained on one reel of tape, an overflow will not only occur by definition on the last pass (since all the data eventually gravitate to one tape) but may also occur on the next to last pass. An explanation of this possibility lies in the fact that the total number of items, N, may barely exceed some power of k, say k^t, wherein k^t items will be contained on one reel prior to the final merge. Though k^t may be less than N it may nevertheless exceed the capacity of one reel if N items may not be contained on one reel. Consequently, even if it were desirable to program around the overflow on the last pass (by planning a two reel output) the results would not always be satisfactory because of the danger of overflow on the preceeding pass.

One procedure for sorting data contained on more than one reel of tape is "Multi-Reel Merging". Each tape reel is sorted independently and these sorted reels are then merged into longer and longer multi-reel streams and ultimately into one completely ordered multi-reel stream.

In effecting the Multi-Reel Merge it is desirable to utilize the available tape stations in an optimum fashion, i.e. in such a manner that a minimum number of single tape reel passes are required. For example, assume that a given computer system has four tape stations making possible the merging of data from three input tape stations to one output tape station. Assume further that 17 reels of pre-sorted data are to be merged to form one 17 reel ordered sequence. One method of attack might be as follows. Merge three reels to form one three reel sequence. Repeat this procedure four more times. Merge the last two reels to form one two reel sequence. Now merge three of the three reel sequences producing one nine reel sequence. Then merge the two remaining three reel sequences and the single two reel sequence producing one eight reel sequence. Complete the

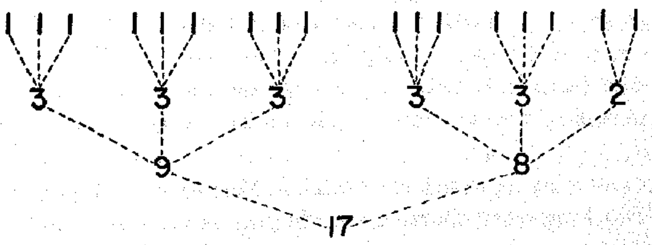

FIGURE 1

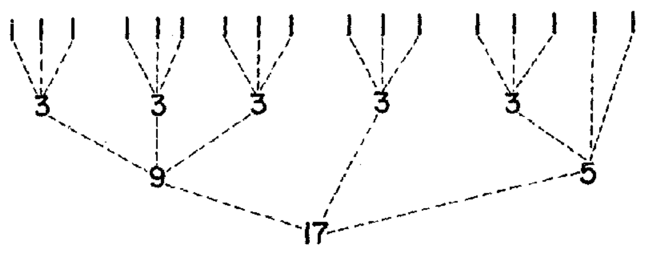

FIGURE 2

Multi-Reel Merge by merging the nine reel sequence with the eight reel sequence. The total number of single tape reel passes is 51. (See Figure 1).

The optimum method of attack requires only 46 single tape reel passes. Figure 2 illustrates this optimum method of attack. Algorithms which yield the optimum method of attack for all possible parameters may be found in Appendix 4.

Another procedure for sorting data contained on more than one reel of tape is "Multi-Reel Blocking". Under this procedure "Multi-Reel Merging" is in effect reversed. The contents of the original reels are broken down into smaller and smaller multi-reel sections until finally the original data are redistributed onto single reels of tape in such a manner that after each reel is sorted with respect to itself a complete multi-reel ordered sequence emerges. For example, again assume four tape stations and 17 reels of tape (this time 17 unsorted reels) and assume further that the control fields are evenly distributed between 1 and 1,700,000. The following describes the optimum redistributional breakdown. From the 17 reels mounted one at a time on tape station 1, distribute all items with control fields ranging from 1 to 900,000 to tape station 2, from 900,001 to 1,200,000 to tape station 3 and from 1,200,001 to 1,700,000 to tape station 4. Continue as illustrated in Figure 3. The congruence of this diagram to that illustrated for Multi-Reel Merging should be noted.

Obviously optimum "Multi-Reel Blocking" is only feasible if the exact distribution of the control fields is known. Where only an approximation to the exact distribution is available it is advisable to assume the presence of more unsorted reels at the outset than actually exist. (In the above example assume, say, 20 reels exist instead of 17.) The optimum multi-reel distribution for the assumed number of reels leaves a margin for error, thus helping to prevent the overflow of data beyond the expected number of reels under any distributional pass. If the distribution of control fields is non-random, an approach similar to that described in the Cascading Pseudo-Radix Sort may be used for distributing the items evenly with respect to optimum requirements. The upper and lower bounds for the cascading distribution may be determined for any given run during the preceding run over the same items by statistical analysis and inverse interpolation.

In general, barring systems restrictions, Multi-Reel merging is superior to Multi-Reel Blocking where Sorting by Merging is used for the individual reels. However, when individual reels are to be sorted by Radix Sorting, Multi-reel

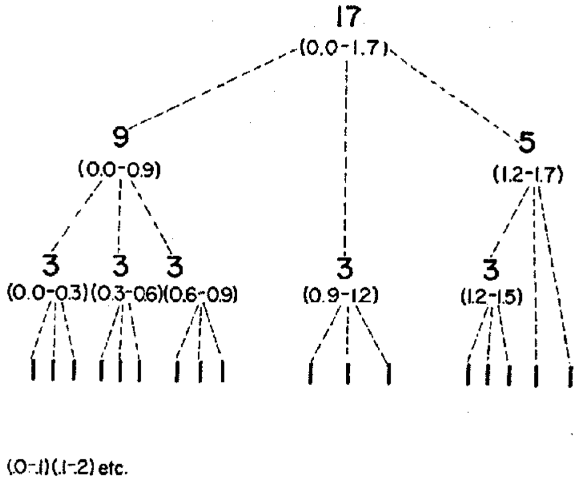

FIG. 3. (The numbers in the parentheses represent 10^{-6} times the upper and lower bounds of the control fields.)

Blocking is superior because the range, R, will then be minimal for each reel, thereby reducing the number of passes required to sort the individual reel.

CONCLUSION

A. *Tape Sorting vs. Punched Card Sorting*

Considerable savings over traditional punched card handling methods may be realized through the utilization of electronic tape-handling equipment for sorting large quantities of data. The savings arise both from the increased speed with which sorting operations may be completed and the increased accuracy resultant from automatic rather than manual handling of the data. Not only are the savings within reach for "one time" jobs (through the use of generalized sorting programs) but they are compounded for situations requiring repeated sorting operations on data having the same or similar parameters. In some cases where approximately the same data are being rearranged time and again the savings are even more marked. (See Pseudo-Radix Sorting).

The economics of sorting on electronic tape handling equipment is, however, another matter entirely. In many quarters it has been contended that the cost of sorting tape stored data is so much more expensive than sorting punched card data that all of the apparent savings are negated. Before remarking on this contention it should be pointed out that the cost of sorting punched card data is a function of the number of characters in the control fields, the number of punched card items, the percentage efficiency of the punched card sorting operation, and the cost per hour of operating a punched card sorter; whereas the cost of sorting tape stored data is basically a function of the log to the base k of the number of tape stored items (where k is the largest integer equal to or less than half the number of tape units under the control of the system) and the cost per hour of operating the electronic equipment. As a consequence two facts are apparent: (1) tape sorting is effected in approximately the same time regardless of the size of the control field and (2) an increase by a factor, f, in the number of items to be sorted increases the ratio of card sorting cost to tape sorting cost by a factor of $(fn/\log_k fn) \div (n/\log_k n)$ or, expressed in simplified form, by a factor of

$$\frac{f}{\dfrac{\log_k f}{\log_k n} + 1}$$

Another less apparent but very important fact (and also one which is difficult to measure relatively) is that as the number of punched cards increases the efficiency of punched card sorter operations decreases. Clearly, then, as the size of the control field and the number of items increase, the cost of sorting tape stored data relative to the cost of sorting punched card data decreases substantially.

Even if under a given set of circumstances, it may be decided that tape sorting is more "expensive" than punched card sorting despite appropriate consideration of the elements of time and accuracy, another strong argument in favor of tape sorting may be advanced. This argument pertains to the advantageous utilization of scheduled idle time on the electronic equipment. It is generally agreed among users of electronic equipment that scheduling this equipment to its capacity may prove quite disastrous in view of the many contingencies that may arise. Consequently, if a sorting operation which may be accomplished satisfactorily on mechanical equipment were conditionally scheduled for the otherwise idle computer system time, considerable advantage could be realized when the system "unexpectedly" functions at capacity. When adverse conditions arise which wipe out the idle time the sorting could be effected by drafting punched card equipment for emergency use or after-hour use, or, alternatively, the punched card sorting could be accomplished on a service bureau basis. Other possibilities would be the pooling of idle computer system time by several users with the same equipment or the service bureau use of computer systems when necessary.

B. *Choosing the Optimum Tape Sorting Approach*

If the computer system permits backward as well as forward reading of tape stored information, and if the data being ordered are sufficiently fixed so that the distribution of control fields is known within the requirements of Pseudo-Radix Sorting, that method will generally prove most advantageous.

Barring the application of this specialized approach, it is necessary to choose between Sorting by Merging and the usual methods of Radix Sorting. Normally, control fields will cover such a range (highest control field minus lowest control field) that Radix Sorting must be eliminated almost immediately, but for data with a relatively small range of control fields it is well to investigate the possible superiority of Radix Sorting in the following manner. Divide the range, R, by the estimated number of tape reels required to contain the data in grouped fashion. This will yield the approximate range of control fields per reel. Divide this result by N/F where N is the number of items which may be contained in grouped fashion on one reel and F is an estimate of the maximum number of items which may be sorted internally without access to tape. If the result of this division is less than unity, then Internal Sorting will be of no advantage and Radix Sorting will be superior.

The optimum number of tape units which may be used in ordering the data stored on a single tape reel will normally be the maximum number of tape units which may be put under control of the computer system at one time. There are exceptions to this for Sorting by Merging as may be observed in the section on Optimizing the Parameters of the Merging Pass.

Determination of the optimum technique for Internal Sorting as it applies to Sorting by Merging will depend on the form and arrangement of the data prior to the sorting operation. For large items with large control fields, Nth Degree Selecting will normally be superior but detailed analysis of the other techniques as they might apply to the particular problem at hand is usually desirable.

APPENDIX 1

Derivation of the Optimum Parameters for a Tape Sorting by Merging Operation Which is Initiated by an Internal Sorting Pass and Completed With a Series of Merging Passes

The parameters to be optimized in a sorting by merging operation include: (1) "k", the number of input (as well as output) tapes involved in the merging passes, (2) "F", the length (in number of items) of the internal sequences produced during the internal sorting pass, and (3) "G", the length (in number of items) of the input and output areas utilized in the merging passes. Values of these parameters are to be considered optimum if they bring about the completion of the sort in the minimum amount of time.

Before introducing the algorithms which will yield the necessary optimum values, certain pertinent observations should prove enlightening. At first appearances it would seem that the optimum number of tape inputs and outputs (totalling $2k$) involved in the merging passes would be the maximum number of tape inputs and outputs capable of being placed under control of the computer system at one time. That this is not necessarily true may be made obvious by considering the unlikely situation wherein the number of input tapes, k, is so numerous that the amount of input space allotted to each tape is less than the amount required for one item. Not only does this fail to optimize the conditions of the merging pass, but it makes the merging pass an impossibility. To a lesser degree, a number of input tapes which permit only one item from each tape to be contained in memory will often be less desirable than a reduced number of input tapes which permit two or more items from each tape to be contained in memory in grouped fashion.

This situation occurs when the saving in stop/start time and the reduced number of operations per item processed more than offsets the loss in extra passes over the data, if any. (Sometimes the number of merging passes required to complete a merging sort will not increase despite the use of fewer input and output tapes, i.e. $k^{p-1} < n \leqq k^p$ and $(k-1)^{p-1} < n \leqq (k-1)^p$.)

In line with the comments of the preceding paragraph the approach used in deriving the optimum values of F, k, and G is as follows:

(1) Determine the maximum value of $F(=\bar{F})$. (Essentially $\bar{F}$ is the maximum number of items which may be contained in memory along with an internal sorting program and the necessary working space. If the items are relatively large, $\bar{F}$ will vary very little from one internal sorting technique to the next.)

(2) Determine the maximum value of $k(=\bar{k})$. (This will be the greatest integer equal to or less than one-half the number of tape units capable of being placed under control of the computer system at one time.)

(3) Find $p_{\bar{k}}$

$p_{\bar{k}} = [\log_{\bar{k}} N/\bar{F}]$ where N is the total number of items to be sorted.

(4) Determine the minimum value of $F(=\underline{F})$ which will not disturb $p_{\bar{k}}$.

$$\underline{F} = \left[\frac{N}{\bar{k}^{p_{\bar{k}}}}\right]$$

(5) Determine the maximum value of G for a $\bar{k}$-tape merging pass $(=\bar{G}_{\bar{k}})$

Let M = memory size in units of storage

I_k = units of storage required for instructions and working space under a series of k-tape merging passes.

(a) Assume that the computer system is capable of sequential reading and writing only. Then $\bar{G}_{\bar{k}} = (M - I_{\bar{k}})/(\bar{k} + 1)$

(b) Assume that the computer system is capable of simultaneous reading and writing. Then $\bar{G}_{\bar{k}} = (M - I_{\bar{k}})/2\bar{k}$

(6) Determine the smallest value of F, equal to or greater than $\underline{F}$ but less than or equal to $\bar{F}$, which is a multiple of $\bar{G}_{\bar{k}}(=\underline{F}_{\bar{k}})$.

If no such value exists, reduce $\bar{G}_{\bar{k}}$ by one and determine $\underline{F}_{\bar{k}}$ accordingly.

If there is still no such value, continue reducing $\bar{G}_{\bar{k}}$ by one until a suitable value of $\underline{F}_{\bar{k}}$ and $\bar{G}_{\bar{k}}$ is found.[2]

(7) Evaluate the time required to effect the complete sort. Let

$T_{\bar{k}}$	=	the average time required to process an item internally during a $\bar{k}$-tape merging pass. This may be estimated by evaluating the time required to perform the logic of the main loop.
S	=	tape stop/start time.
$T_{\underline{F}_{\bar{k}}}$	=	the average time required to place one item in its final position during an internal sorting pass in which sequences of length $\underline{F}_{\bar{k}}$ are formed. (Where the internal sort evolves only after continuous manipulation of all items such as under "exchanging" or "internal merging", take the total time divided by $\underline{F}_{\bar{k}}$.)
V	=	time required to read or write one item from or to tape excluding stop/start time.
R	=	rewind time for one reel. (Assume that all tape reels may be rewound simultaneously after a merging pass on a system which does not permit backward reading.)

Assuming that the items are ungrouped at the outset, grouped during the internal sorting pass and separated during the final merging pass and assuming further that rewinding is necessary, the time required to effect the complete sort is:

$$N\left(V + S + T_{\underline{F}_{\bar{k}}} + \frac{S}{\bar{G}_{\bar{k}}} + V\right) + R$$

$$+ N(p_{\bar{k}} - 1)\left(V + \frac{S}{\bar{G}_{\bar{k}}} + T_{\bar{k}} + \frac{S}{\bar{G}_{\bar{k}}} + V\right) + \frac{R}{\bar{k}}(p_{\bar{k}} - 1)$$

$$+ N\left(V + \frac{S}{\bar{G}_{\bar{k}}} + T_{\bar{k}} + S + V\right) + R$$

[2] In isolated cases $\bar{G}_{\bar{k}}$ may be reduced so low that it will be beneficial, alternately, to increase $p_{\bar{k}}$ by one, thereby reducing both $\bar{F}$ and $\underline{F}$. If, however, $p_{\bar{k}} + 1 \geqq p_{\bar{k}} - 1$ then nothing is to be gained since (8) will indicate that in any event $\bar{k} - 1$ will be superior to $\bar{k}$.

for sequential reading and writing with obvious adjustments for simultaneous reading and writing.

(8) Reduce $\bar{k}$ by one and reevaluate the time required to effect the complete sort. If the total time increases when $\bar{k} - 1$ is used, then the parameters established using $\bar{k}$ are optimum. If the total time decreases when $\bar{k} - 1$ is used, then $\bar{k} - 1$ may be optimum but a reevaluation using $\bar{k} - 2$ is necessary, etc.

APPENDIX 2

Detailed Discussion of the Role of Forecasting in Sorting by Merging

Forecasting is a technique which permits a single input buffer to be loaded from that one of k input tape reels whose corresponding input area in memory will be exhausted first during a merging sort of k pre-sorted input groups stored in the k input areas.

Forecasting is accomplished in the following manner for an ascending sort. At the outset, after a pre-sorted group from each of the k reels is read into its corresponding area in memory, that reel which contributed the group with the current lowest terminal item is caused to refill the buffer because said group will be exhausted first.

Subsequently, as soon as the input buffer is again read into memory, a comparison is made between the first item in the newly filled area and the last item written to the output area. If the first item in the newly filled area is low, a stepdown has occurred and that tape which contributed the group with the lowest terminal item among the remaining non-stepdown groups refills the buffer. If the first item in the newly filled area is high, that same area participates in the search for the lowest terminal item among the other non-stepdown groups.

If all areas contain stepdown groups, the procedure reverts to that utilized at the outset.

APPENDIX 3

Illustration of the Application of Radix Sorting to a Set of Control Field Items

Assume that the following fourteen control fields are part of fourteen items contained on a single input reel of a computer system capable of controlling eight tape reels at one time: 141, 98, 94, 183, 96, 216, 155, 178, 99, 139, 121, 196, 213, and 137. To facilitate radix sorting, as each item is read into memory its control field is converted to the base 4 and the item dispensed to that one of four output tapes associated with the least significant number in the converted control field. The converted control fields are: 2031, 1202, 1132, 2313, 1200, 3120, 2123, 2302, 1203, 2023, 1321, 3010, 3111, and 2013.

The results of the first dispersion pass are as follows:

Tape 0	Tape 1	Tape 2	Tape 3
1200	2031	1202	2313
3120	1321	1132	2123
3010	3111	2302	1203
			2023
			2013

Assuming that tapes may be read backward and recognizing that an odd number of passes are required to complete the radix sort (four dispersion passes and one collection pass), the second dispersion pass is initiated in "reverse" order, i.e. with the reading backwards of the tape assigned to the highest number in the preceding pass (Tape 3). Each succeeding pass is performed in an order opposite to that of the pass immediately preceding. The results of dispersion pass 2, 3, and 4 are illustrated below:

Dispersion Pass 2

Tape 4	Tape 5	Tape 6	Tape 7
1203	2013	2023	1132
2302	2313	2123	2031
1202	3111	1321	
1200	3010	3120	

Dispersion Pass 3

Tape 0	Tape 1	Tape 2	Tape 3
3010	3111	1200	2302
2013	3120	1202	2313
2023	2123	1203	1321
2031	1132		

Dispersion Pass 4

Tape 4	Tape 5	Tape 6	Tape 7
	1321	2313	3120
	1203	2302	3111
	1202	2123	3010
	1200	2031	
	1132	2023	
		2013	

The collection pass completes the radix sort by reading Tapes 5, 6, and 7 backwards in that order.

It should be noted that if an even number of passes had been required to complete the radix sort, then the second dispersion pass would have been initiated in the usual order with each succeeding pass performed in an order opposite to that of the pass immediately preceding.

APPENDIX 4

Algorithms Leading to Optimum Methods of Multi-Reel Merging and Multi-Reel Blocking

The optimum approach to Multi-Reel Merging may be determined by reversing the optimum approach to Multi-Reel Blocking. Hence, it is sufficient to demonstrate algorithms yielding the optimal approach to accomplishing the latter.

The parameters of the Multi-Reel Blocking problem are "n", the number of reels to be blocked, and "c", the maximum number of reels which may be blocked at one time. (If a computer system is capable of controlling $2k$ tape stations as either input or output media, then $c = 2k - 1$.)

Step 1: If $n > c$, find the smallest value of s such that

$$n \leqq (2c - 1)c^s \qquad \text{where } s > 0$$

If $1 < n \leqq c$, continue at Step 3
If $n = 1$, continue at Step 5

Step 2: Find the largest value of r $(<c)$ such that

$$n = rc^s + (c - r - 1)c^{s-1} + n' \qquad \text{where } n' > 0$$

(Note that $c^{s-1} \leqq n' \leqq c^{s+1}$)

Step 3: If $n > c$, divide n into c blocks: r with c^s reels, $c - r - 1$ with c^{s-1} reels and 1 with n' reels.
If $n \leqq c$, divide n into n blocks of 1 reel and continue at Step 5.

Step 4: Subdivide each of the r blocks with c^s reels and the $c - r - 1$ blocks with c^{s-1} reels into c equal parts. Repeat this operation on each of the new blocks again and again until each block is composed of a single reel.

Step 5: Determine the number of single reel passes required to effect the blocking in Steps 3 and 4 from one of the following expressions:

If $n > c$,

$$p_n = r(s + 1)c^s + (c - r - 1)sc^{s-1} + n' + p_{n'}$$

If $1 < n \leqq c$, $\quad p_n = n \quad p_1 = 0$

Step 6: If $n > c$, substitute n' for n and repeat Steps 1 through 6. When $n \leqq c$, Step 5 will have completed the series of algorithms.

Example:

Assume $n = 17$ and $c = 3$
Step 1: $s = 2$
Step 2: $r = 1, n' = 5$
Step 3: blocks of 9, 3, and 5
Step 4: block of 9 is subdivided into blocks of 3, 3, and 3 and then into blocks of 1, 1, 1, 1, 1, 1, 1, 1, and 1; block of 3 is subdivided into blocks of 1, 1, and 1.
Step 5: $p_{17} = 38 + p_5$
Now $n = 5$ and $c = 3$
Step 1: $s = 1$
Step 2: $r = 1, n' = 1$
Step 3: blocks of 3, 1, and 1
Step 4: block of 3 is subdivided into blocks of 1, 1, and 1.
Step 5: $p_5 = 8 + p_1$
Now $n = 1$ and $c = 3$
Step 5: $p_1 = 0$
$p_{17} = 46$

See part of paper entitled "Multi-Reel Sorting" for diagrams of this result as applied to Multi-Reel Merging as well as Multi-Reel Blocking.

ACKNOWLEDGEMENTS

Special acknowledgements to Charles R. Blair of U. S. Department of Defense; Richard L. Cline of International Business Machines Corp.; Frances E. Holberton of U. S. David Taylor Model Basin; Joseph A. Meyer of U. S. Department of Defense; Richard M. Roehm of International Business Machines Corp.; Edwin J. Steinberg of New York Life Insurance Co.

Additional acknowledgements to Daniel Barry of New York Life Insurance Co.; Patrick Billingsley of U. S. Department of Defense; John Busby of Radio Corporation of America; James J. Connors of New York Life Insurance Co.; Miguel de la Fe of New York Life Insurance Co.; John A. Graham of Sperry Rand Corp.; R. Jane Hostetler of International Business Machines Corp.; John C. McGrath of New York Life Insurance Co.; Michael P. Parker of International Business Machines Corp.; Rita D. Pittman of New York Life Insurance Co.; John A. Rixse of U. S. Department of Defense; Walter Shur of New York Life Insurance Co.

BRIEF GLOSSARY OF SYMBOLS

N:	the total number of individual items to be sorted; also, the "degree" of an internal sorting technique titled "Selecting"
n:	the total number of item sequences (ascending or descending) at any given phase of a sorting operation

p: the number of passes required to sort under a given set of circumstances
k: the number of input (or output) tape reels utilized in a merging or radix sorting pass; also usually one-half of the number of tape reels which may be controlled by an electronic computer system at one time
F: the length (in number of items) of the internal sequences produced during a given internal sorting pass
G: the length (in number of items) of the input and output areas utilized in a given merging pass
R: the range, or span, of item control fields in reference to a given Radix Sorting operation

REFERENCES

1. Brief Description of Sorting Methods for the UNIVAC System. Electronic Computer Department Training Section, Remington Rand, January 19, 1954.
2. EDPM Program Brief #3: 705 Sorting Table. I.B.M. Corporation.
3. EDPM Program Brief #8: 702 Sorting Techniques. I.B.M. Corporation.