UNIVAC

DATA PROCESSING DIVISION

1004/1005

SYSTEMS

INTRODUCTION TO RANDOM PROCESSING

UP-4093

CONTENTS

CONTENTS i

1. INTRODUCTION 1

2. APPLICATION 2

3. STORAGE DESIGN 6

4. FILE LAYOUT 7

5. METHODS OF ADDRESSING 8

6. RANDOM VS. SEQUENTIAL FILES 15

7. TIMING RANDOM ACCESS OPERATIONS 16

1. INTRODUCTION

A. WHAT IS RANDOM ACCESS?

In a pure sense, random access is the ability of a storage medium to access information without time penalty, regardless of its location in storage. Core storage, for example, is a true random access device, because one accessing of data does not determine the time required for the following accesses. In punched cards or magnetic tape files on the other hand, more time is required to go from the first record of the file to the last, than is needed to go from a record located at the midpoint to the last. The time penalty is even greater when going from the last record to the first. In actual practice, random access is a relative term, meaning that access time is considerably less dependent upon record location than it is in a sequential filing system.

B. FEATURES OF RANDOM ACCESS

The most common random access devices are magnetic drums and discs. Data are recorded on the surfaces of these devices in the form of magnetic spots. One significant design feature of the random access device is the clustering of data around the surface, eliminating the great distances over which records may be spread. This type of design, when combined with the ability of the reading and writing units to move directly to an area on the recording surface, provides the random access capability. These design features are translated into operational characteristics which spell the difference between random and sequential processing.

The random access unit can:

1. Select a record from a file by taking a direct path to it. This method is opposed to sequential searching used by serial filing systems.

2. Search a file in either direction with equal facility. Card systems are restricted to either ascending or descending operation, while most tape systems change the direction of a search with reduced efficiency.

3. Select a record, deliver it to the central processor for alteration, and then return it to its original location. Changes to punched card or tape records require the creation of new records.

2. APPLICATION

A. GENERAL

The seemingly few features of the random access unit combine to provide many unique advantages when they are applied to a data processing situation. Just how the random access approach differs from the sequential can best be viewed by comparing systems. Figures 1 and 2 outline two methods of invoicing. The following is a discussion of the significant differences in results.

B. PRE-SORTING

The Billing Detail Cards in figure 1 are sorted into Customer Number sequence, because the proper name and address is selected from the master file by a collation technique, which demands that both files be in the same order.

With a random access system pre-sorting is not required, because the selection of the name and address is not accomplished by collating. Instead, the system analyzes the Customer Number in the Billing Detail Card, and then instructs the disc to go directly to the place where the corresponding name and address is stored. The disc responds by retrieving the desired record and delivers it to the main processing system, which, in the example given in figure 2, is the UNIVAC® 1004/1005.

While the magnetic tape or punched card system demands pre-sorting the random access system makes it optional.

C. FILE SEARCHING

In the sequential system, illustrated by figure 1, the selection of the name and address from the master file is accomplished by reading and comparing each name and address card to the current detail card. On each non-match condition, another name and address card is read. This process continues until a match condition occurs; then the printing of the invoice occurs. Thus, the entire 5,000 names and addresses are read when only 150 were needed; the result is a lower throughput rate.

The random access approach, using the ability to read only the desired record, eliminates the reading of unwanted records. Only the names and addresses required are read into the central processor, because the disc retrieves records, not by collation, but by directly accessing the ones desired.

® *Registered Trademark of the Sperry Rand Corporation.*

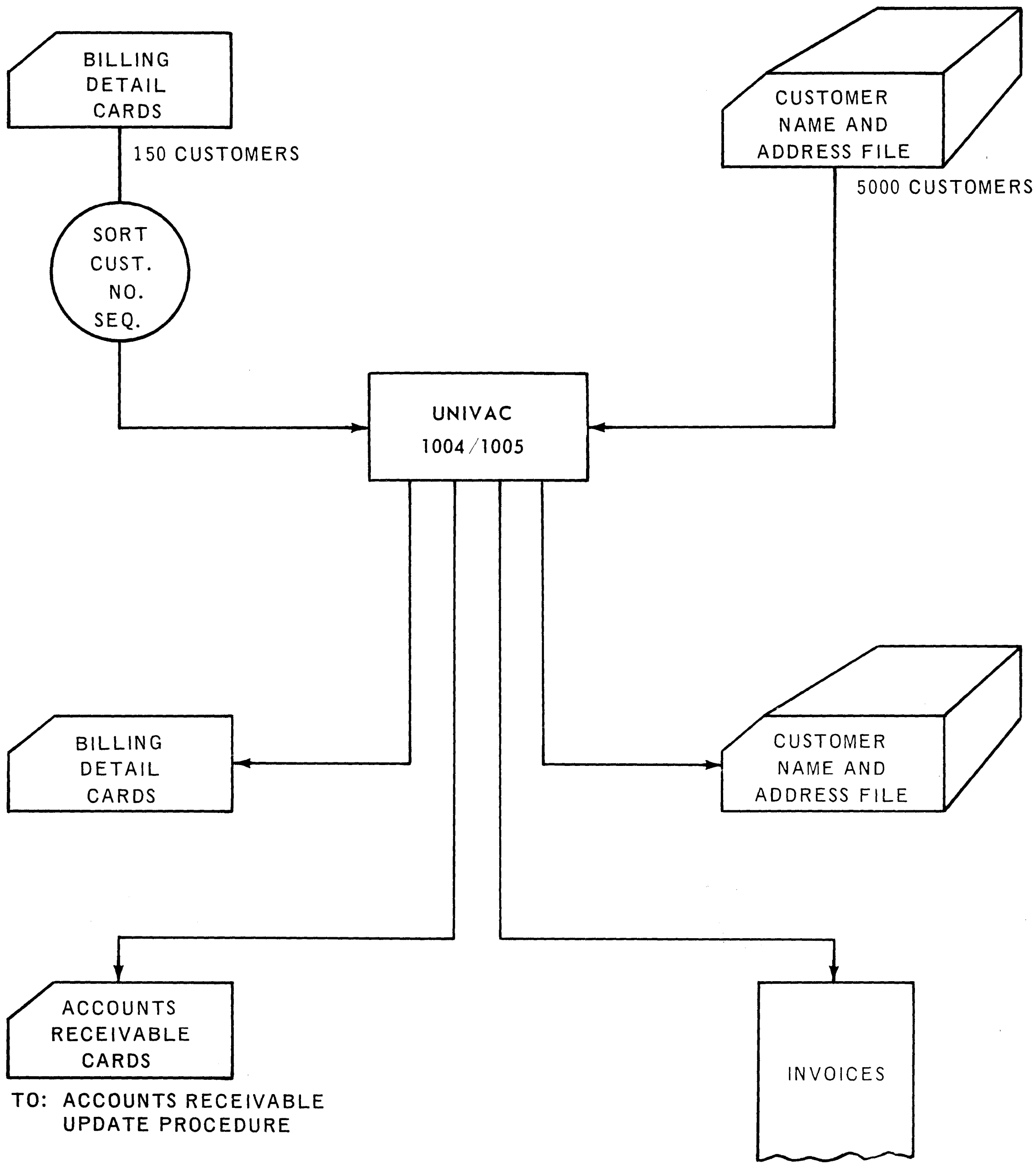
BILLING DETAIL CARDS
150 CUSTOMERS
SORT CUST. NO. SEQ.
CUSTOMER NAME AND ADDRESS FILE
5000 CUSTOMERS
UNIVAC 1004/1005
BILLING DETAIL CARDS
CUSTOMER NAME AND ADDRESS FILE
ACCOUNTS RECEIVABLE CARDS
TO: ACCOUNTS RECEIVABLE UPDATE PROCEDURE
INVOICES

Figure 1.

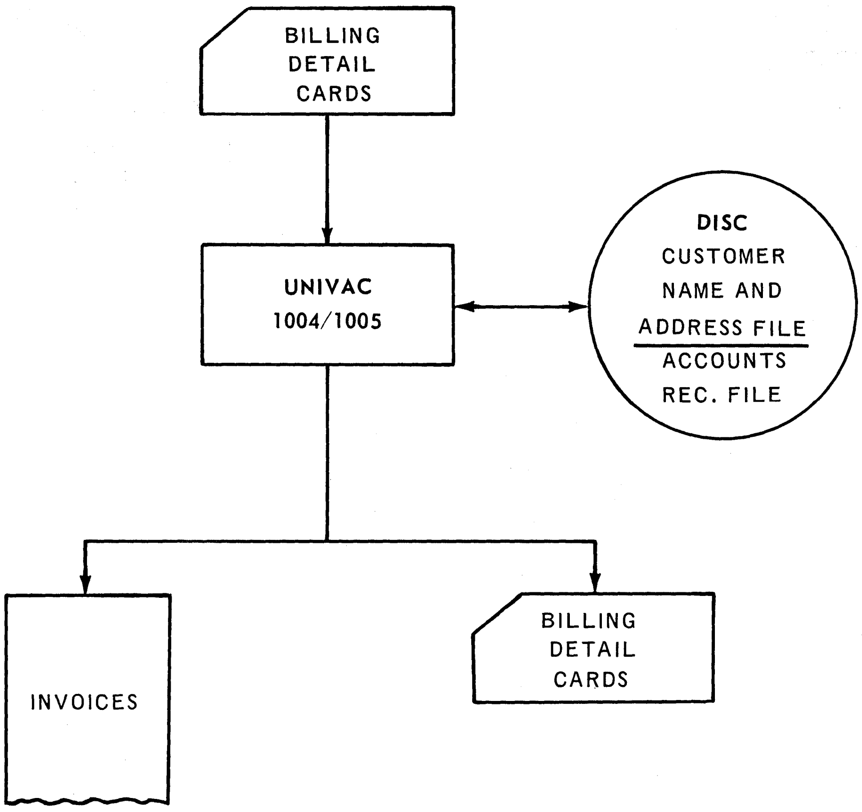

Figure 2.

D. BATCHING

If the number of invoices to be written in the example is 150 per day, the weekly time loss is relatively high, because 4850 unused (5000–150) name and address records are passed through the system 5 times each week. This problem is usually reduced by "batching" the transactions and processing against the master file less frequently. For instance, it is possible to accumulate transactions for 5 days and perform the major operations on a weekly basis. Thus, in the billing illustration, the number of unused cards processed is reduced from 24,500 (4850 x 5) to 4250. Batching is a compromise between efficient use of the data processing system and satisfying the dynamic needs of the application.

With a random access system, the invoices can be prepared daily, because the system running time is directly proportional to the number of transactions, not the number of other master file records screened. Therefore, the system can be used efficiently and still respond dynamically.

E. FILE SEGREGATION BY ACTIVITY

Another method of making the sequential system more efficient, is to segregate master file records by frequency of use. In the invoicing example, the customer name and address file can be separated into two or more files depending upon the number of transactions with each particular customer. When invoices are run, only those name and address cards for active customers are passed through the main computing system.

The low activity addresses are inserted prior to the invoicing run, or invoices are prepared separately. This method requires additional peripheral operations and introduces a certain loss of control to the overall system.

Occasionally, random access systems use this very same technique to achieve higher throughput rates, but without the inherent problems of additional peripheral operations or loss of control. In the application being discussed here, the random access system would not have to employ the technique at all.

F. UPDATING MULTIPLE FILES

The Accounts Receivable summary cards are outputs of the invoicing run. In a separate procedure, these cards are used to update the accounts receivable totals.

With a random access system it is possible to store both the name and address and accounts receivable files on the same media, thereby having both files available at the same time. This feature enables the system to update the accounts receivable at the same time, eliminating the need to process the transaction cards twice and providing an up-to-the-minute accounts receivable.

In addition, the unique ability of the random access system to read records, have the processor update them, then return the adjusted records to their original places in the file, has interesting systems application. In the case of the accounts receivable, returns or partial payments can be applied against the open items without the need to create new cards or tape records as balances are adjusted. Nor is there a need to either entirely reproduce the file or interfile updated records.

G. STATUS INQUIRY

One of the prime advantages of a punched card system is the ease of reference. If a credit check is required on a particular account, a visual reference to the accounts receivable file is usually sufficient, except at those times when the file is out for processing or when the most recent transactions have not as yet been applied. Magnetic tape systems on the other hand make reference to files relatively difficult and time consuming.

Random access systems seek to balance both the need for rapid updating of files and the ability for ready reference. As more files are added to the random access storage medium to provide greater multiple file updating capability, more data is available for external inquiry. Thus, no matter what operation is in progress, credit checks can be made as long as the accounts receivable file is on-line to the processor.

H. SELECTING RANDOM ACCESS

There are no categorical answers to the question of deciding between punched cards, magnetic tape, or random access as the solution to the data processing problem. The selection of a particular system depends upon such factors as cost, speed, file activity, the need for rapid updating and inquiry. Each of the three basic systems has merit, depending upon the specific requirements of the applications. In fact, many installations use a combination of all three methods to produce the best results.

3. STORAGE DESIGN

The recording surface of a random access device is divided into sectors, each of which can contain one or more records. For purposes of explanation, a sector should be thought of as having the ability to store one record only. On the Unidisc System, for example, these sectors are arranged in a circular fashion on the disc. See figure 3 below.

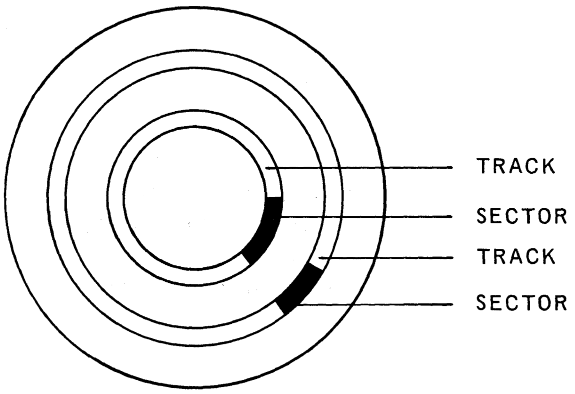

Figure 3

The average storage capacity of a Sector is 100 alpha-numeric characters. One complete circle on the disc contains 100 Sectors and is called a track. There are 100 tracks on the disc, each of which contains 100 sectors. Thus, there are 10,000 sectors to each disc surface (100 x 100).

Each sector on a track is identified by a two digit number, from 00 to 99, which is known as the sector address. Further, each track is identified by a number from 00 to 99, which when placed in front of the sector produces a four digit number ranging from 0000 to 9999. Thus, each sector is separately identified by a number which is called the *address*. When more than one disc unit is used, a single digit number is added to the left of the original address, producing a 5 digit address.

As is the case with storage in the central processor, each address refers to a specific location in the storage medium. Whenever the central processor requires a particular record, it sends the address of the record to the Random Access device, which proceeds directly to the location specified and retrieves its contents. The random access device is thus freed of the task of reading and comparing record identifiers such as customer numbers, employee number, *etc.* Random access systems perform at higher rates of speed when operating in the address mode, although they possess and often use the ability to search for records by comparing record identifiers.

4. FILE LAYOUT

The most common method of organizing a file is to store the records in sequence, as in a card or magnetic tape file. The first record is stored at address 00000, the second at 00001, *etc.* Instead of merging new records into the file as they occur, an *overflow* area in random access storage, probably at the end, is reserved to receive records added after the file is sorted and stored. Periodically, these newly added records are incorporated into the sequenced portion of the file by a sort program. The sort program clears the *overflow* area so that it is again ready to receive new records.

The frequency at which the sort operation is performed depends entirely upon the needs of the particular application. The file may be resequenced prior to the printing of a sequential report, or when the *overflow* area becomes filled. The size of the *overflow* area is determined by the frequency of transactions. It must be large enough to store all records added between sequencing operations.

Opposed to the sequential file is the random file, which as the name implies, operates without regard to the order of the stored files. As records are added to the random file, the central processor converts the record identifier to an address, by means of a randomizing *routine*. Once the record is stored, further references to it are made by re-calculating the address, using the same randomizing routine.

5. METHODS OF ADDRESSING

A. DIRECT ADDRESSING

In some cases, a technique known as *direct addressing* can be used to reference records stored in a Random Access system. The term *direct addressing* means that the address of a record to be referenced enters the system as part of the input. There are several ways in which direct addressing can be accomplished.

In certain of these cases, the record numbering scheme is similar to the addressing structure employed by the random access system. Assume, for example, a single disc system with an address range of 0000 to 9999, being used for a payroll system. Employee numbers could be assigned as five digit numbers within the range of 10000 to 19999, enabling the system to handle up to 10,000 employees. Thus, whenever a particular employee record is referenced, the central processor simply uses the last four digits of employee number as an address when the access is made.

This technique, while easy to use, will require reassignment of record identification numbers in most cases. In addition, it does not provide for alphabetic record identifiers.

In other cases the address of a record is made a suffix of the record identifier and appears on all external documents as part of the record number. Thus, each record brings its own address into the central processor, simplifying the address calculation within the system. However, this method is rather inflexible and is burdensome on external operations, particularly those requiring manual effort.

Occasionally, external *indices* or *directories* are used to convert record identifiers to random access addresses before entries are made to the system. The index is a cross reference file, sequenced by record number, in which each record number is paired with the address of its location in storage. This *index* could be in the form of card file in which each card would contain a record number and its respective address. Transactions to be applied to the master file are sorted into record number order and matched against the *index* file. The addresses are then reproduced into the transaction cards.

Index files allow optimum use of the data processing system at the cost of external processing time and adding the need for batching. Furthermore, the index file will require maintenance as records are added and deleted.

B. REFERENCE ADDRESSING

It is a common practice in UNIVAC random access installations, to use the data processing system to convert record numbers to addresses. This conversion is made automatically by sub-routines in the main programs, eliminating all external activity.

Reference Addressing is a very important technique for creating addresses from record numbers. It is an extension of the external index method with the principle difference being the location of the index. When *Reference Addressing* is employed, the index is stored in the data processing system, either in the storage of the central processor or in the random access unit.

Special techniques, working in concert with the sequential file layout, are used to reduce the amount of storage required to store the index. An index to 5,000 master records would be stored in 5,000 cards if an external index were used. Reference Addressing, which is explained by example below, can reduce the index to only 50 entries.

Figure 4 represents a storage layout chart for a master file of 5,000 records stored on a disc in locations 0000 through 4999. The number of the first record is 598; the last is 5597. In this example, the records in the file are numbered consecutively, although consecutive numbering is not a requirement. The number of the record located in the last sector of each track is stored, along with the respective track number, in a table called the *track index*.

Whenever a particular record is to be referenced, its number is compared to the first record number in the table. If the desired record number is equal to or less than the first number in the table, the entire record is stored in track 00. If the record number is greater than the first number in the table, the comparison moves to the next entry in the table and continues until an equal or less than result occurs.

In the example shown in figure 4, the comparison will start with the table entry to the extreme left of the *track index*. The desired number (1004) will test high for the first four comparisons and low in the fifth. Since the track numbers are stored in the table, the program is able to determine the track number as soon as the "less than" condition occurs. This technique does not require that the transactions be applied in any particular sequence.

The track number of a desired record is thus determined by searching the track index. The specific record is obtained by searching the specified track in the data mode. That is, by comparing record numbers.

If the record is not found in the main portion of the file, a search of the *overflow* track(s) is made, because the record was added after file was last sorted and reassembled.

MAIN FILE

	SECTORS												
	00	01	02	03	04	05	06	07	08	09			99
00	0598	0599	0600	0601	0602	0603	0604	0605	0606	0607			0697
01	0698	0699	0700	0701	0702	0703	0704	0705	0706	0707			0797
02	0798	0799	0800	0801	0802	0803	0804	0805	0806	0807			0897
03	0898	0899	0900	0901	0902	0903	0904	0905	0906	0907			0997
04	0998	0999	1000	1001	1002	1003	1004	1005	1006	1007			1097
05	1098	1099	1100	1101	1102	1103	1104	1105	1106	1107			1197
06	1198	1199	1200	1201	1202	1203	1204	1205	1206	1207			1297
07	1298	1299	1300	1301	1302	1303	1304	1305	1306	1307			1398
08	1398	1399	1400	1401	1402	1403	1404	1405	1406	1407			1497
09	1498	1499	1500	1501	1502	1503	1504	1505	1506	1507			1597
49	5498	5499	5500	5501	5502	5503	5504	5505	5506	5507			5597

TRACK INDEX

697	00	0797	01	0897	02	0997	03	1097	04	1197	05	1297	06	1397	07	1497	08	1597	09		5597	49

Figure 4.

C. RANDOM ADDRESSING

1. General

Airline reservation applications served by UNIVAC Real-Time systems, reserve seats, cancel reservations, and answer inquiries concerning the availability of passenger space while the customer is standing at the reservation desk. Transactions and inquiries in this application occur at very high frequency without interruption.

Applications such as this require that the records in random access storage be available without interruption. Sorting, required to incorporate new records from overflow storage, causes an interruption in the availability of the master file. The frequency of sorting is determined by the transaction volume because the overflow area must be cleared as it becomes filled. Thus, in a dynamic situation, an alternative to sequential filing must be employed.

Applications with a requirement for dynamic response such as airline reservation systems, employ random file organization which allows all data to be available at all times, because sorting to enable the addressing scheme to be efficient is not necessary. Instead of locating records by reference to indices, random filing systems use mathematical techniques to calculate addresses from record numbers.

The object of the address calculation is to convert each record number into a unique address with as few duplications as possible, and to distribute the records evenly among the available sectors to make efficient use of storage. These objectives are achieved by converting record numbers on a random basis, hence the term *randomizing routine*.

There are many techniques for calculating random addresses. By way of the following example, one method (Prime Number Division) will be explained:

2. Application

A file of 20,000 master records with an 8 digit key is to be stored on a cleared Unidisc system. The lowest numbered record in the file is 00100000; the highest number is 99999999. The number of sectors assigned as storage is 40,000. These sectors are numbered consecutively from 00000 to 39999.

3. Planning the Conversion

Prime number division will be used to reduce the eight digit record number to a 5 digit address within the range of 00000 to 39999. A prime number is a number, other than zero, that can be divided evenly only by 1 and itself.

Step 1 – Using the highest allotted address – in this case 39999 – select the corresponding prime number from the table (figure 5). This prime number is 39989.

Step 2 – Enter this number as a constant to all programs that reference this file.

HIGHEST ADDRESS ALLOTTED	PRIME NUMBER	HIGHEST ADDRESS ALLOTTED	PRIME NUMBER
09999	09973	35999	35999
11999	11987	37999	37997
13999	13999	39999	39989
15999	15991	41999	41999
17999	17989	43999	43997
19999	19997	45999	45989
21999	21997	47999	47981
23999	23993	49999	49999
25999	25999	51999	51997
27999	27997	53999	53993
29999	29989	55999	55997
31999	31991	57999	57991
33999	33997	59999	59999

Figure 5. Table of Prime Numbers

4. The Program

The randomizing routine is a sub-routine of all programs that enter, delete, or reference records in the file.

This sub-routine is entered prior to any reference to random access storage.

Step 1 – Divide the record number by the prime number – a constant – seeking a quotient in whole numbers. Assuming the record number is 00246859, the results will be as follows:

$$\frac{00246859}{39989} = 6 \text{ and } 06925 \text{ remainder}$$

Step 2 – Using the remainder (06925) from step 1 as an address, read a record from random access storage.

NOTE: The highest possible remainder is 39988. Therefore, this calculated address will always be within the limits of the storage area provided.

Step 3 – Check this sector for a prior entry. If none, go to the next step. If a record has already been written into the sector, go to step 5.

The purpose of the test is to prevent the entry of a record into a sector that already contains a valid entry.

The reason for the prior entry is that it is possible for more than one record number to generate the same address, no matter which randomizing method is used. In this case, the duplicate address is caused by the fact that different dividends can leave the same remainder. For example, seven divided into either 22 or 29 will leave a remainder of 1.

One way of ascertaining whether or not a sector is available is to clear all sectors to spaces before the disc is put into service and to clear sectors to spaces as they become available. Another way is store a unique code in all unused sectors.

Step 4 – Using the remainder from step 1 as an address, write the record into the sector.

Step 5 – There are two ways of filing a duplicate record within the limits of the storage area allocated. First, attempt to file the record at the next higher address – 06926 in the example – repreating the test for a prior record. Each time a prior record is found, try the next higher sector until an opening appears. If the search runs beyond the highest address allowed, return to the lowest address and continue the search. The second method is to turn control over to the system, allowing it to find the first open sector on the track, either higher or lower than the original sector.

The number of duplicate addresses generated is related to the size of the random access storage area provided to store the file. As the number of sectors is increased, the divisor becomes larger, providing a greater number of possible remainders.

D. ALTERNATE FILING TECHNIQUES

1. Overflow

In the method described, records with duplicate addresses are filed within the confines of the storage allocated. This method conserves storage, at the cost of additional accesses when duplicates occur. These results can be reversed by adding an *overflow* area. In this case the original storage area is made smaller and is called the *primary* area. An even smaller area, called *overflow* storage, is provided to store duplicate records. However, the total number of sectors allocated is larger than that provided in the single area.

In the example above, sectors 00000 to 39999 were allocated. With the *overflow* method the *primary* area would be 00000 to 29999. The *overflow* area could be 30000 to 43999. All references, both reading and writing, would first be made to the primary area. On a duplicate condition, the following references would be made to the *overflow* area. Thus, if a record is to be written into sector 05625 and that sector already contains a record, the base address of the overflow area is added to the original address (05625 + 30000 = 35625). This sum is compared to the highest allocated address in *overflow* storage (43999). If the sum is the lesser value, it becomes the address at which the next filing attempt is made. If the sum is greater than the highest overflow address allowed, the highest address is subtracted from it, and the result is substituted for the original address. The program re-enters the add and compare steps, repeating the process until a satisfactory address is obtained.

The next reference would be to the new address. If a duplicate address is encountered here, either the technique described in step 5 of the sample method or reference to a *second overflow* area would follow.

The advantage of the *overflow* method is that the primary area never contains duplicates. When the *overflow* technique is not used, a duplicate can cause an artificial duplicate, which in turn raises the average number of accesses required to reference a particular record. If for instance, an attempt is made to file a record at 06925 when it already contains a record, the record might be filed at 06926. Sometime later a calculated address can be 06926, but this record cannot be filed at its natural location, because a duplicate is in its place. The *overflow* method results in higher throughput rates by reducing the number of accesses required to file or locate a record. However, more storage is required.

The number of *overflow* areas provided, if any, and how they are to be handled is determined by the amount of storage available, the throughput rates required and the characteristics of the record numbering system.

6. RANDOM vs. SEQUENTIAL FILES

A. GENERAL

Early in the systems design phase of a random access system, the type of file layout should be selected. This selection can be made by reviewing the requirements of each application against the virtues of each method. A discussion of the values of these two fundamental methods is presented to assist in making the choice.

B. SEQUENTIAL FILES

Sequential filing provides for more efficient use of random access storage because records are filed in consecutive sectors with no gaps and smaller overflow areas. Only the requirement for storing the index prevents 100% utilization of storage. Still, the utilization of storage in sequential filing is 99%, which is rarely attained in a random file.

Sequential files enable the user to take advantage of the economy of a removable disc. With a random file all records must be on line during a run. If a file requires several discs because of its size, the system must include enough disc handlers to permit all data on the discs to be available at the same time. With sequential files, however, each disc can be loaded onto a handler as the run demands, just as in a card operation only one tray of cards need be handled at one time.

On the average, fewer accesses are required to locate records in random access storage with sequential filing. Throughput rates increase as the number of accesses is reduced.

When the choice between filing methods is applied to Unidisc specifically, one other consideration exists. Unidisc is unique among disc storage systems in that additional circuitry and random access storage are provided for the specific purpose of streamlining the sequential file system. These additional features reduce the average time required to retrieve a record, minimize the programming effort, and eliminate the need to use the central processor for searching the index.

C. RANDOM FILES

Random Files offer the advantage of never requiring sorting to incorporate new records into the main body of the file, although sorting may be required before reports are run. They do not require the system to store index tables. In short, random files are capable of responding dynamically to demands for data.

7. TIMING RANDOM ACCESS OPERATIONS

A. GENERAL

The time required to retrieve a record from a random access device is based upon two fundamental motions which occur outside the central processor (see figure 6). First is the time required to move the access arm to the desired track. On Unidisc, this time varies from 0 milliseconds, when the arm is already in position, to 196 milliseconds, when the maximum distance of 49 tracks must be traversed. The design of Unidisc provides two sets of read/write heads on the access arm. These heads, working in tandem, service 50 tracks each ($\frac{100 \text{ Tracks}}{2} = 50$ Tracks). The longest travel is from Track 0 to Track 49 or 50 to 99 which is 49 tracks.

The time required to start and stop the motion of the access arm is always 45 ms. The time required to cross each track is 4 milliseconds. Thus, the time required to move 1 track is 49 ms, while the time needed to move 49 tracks is 241 ms.

The other factor is the rotation of the disc on the handler. The selection of a particular sector on a track is accomplished by means of the rotation. Once the read/write head is in position, as a result of arm movement, it becomes stationary on the desired track. From this position on the track, the read/write heads wait for the desired sector to pass by.

If the sector desired is approaching the read/write head as it assumes its rest position, very little time delay will be required to read the record.

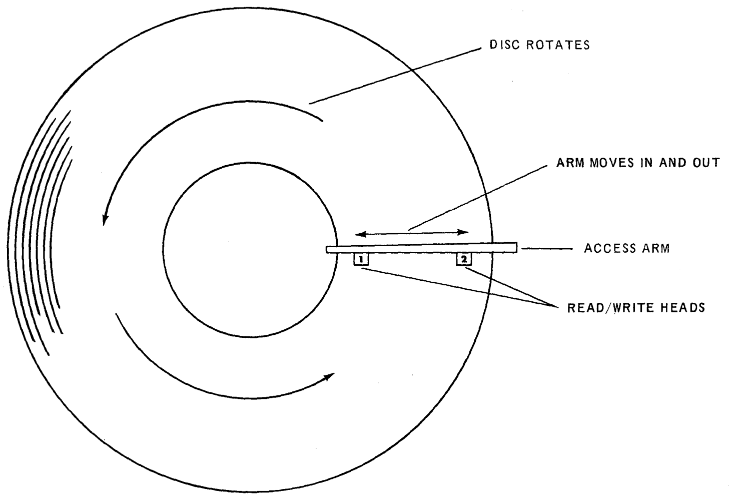

Figure 6.

If, on the other hand, the sector has just passed the read/write head when it settles into position, the maximum read time will occur. Maximum read time is 50 milliseconds, which is the time required for one complete revolution of the disc. These timings are applied differently for sequential and random operations.

B. RANDOM FILES

Random operation timings are based upon averages, because the distances involved in any one access of data are not predictable. Average arm positioning time is derived as follows:

$$\frac{45 + 4\,(N-1)}{3} = 110 \text{ ms; where } N = \text{the maximum number of tracks traversed (50)}$$

Average Rotational Time is 25 ms.

The average time required to read a record from storage is 110 ms + 25 ms or 135 ms.

Write operations are followed by automatic read/backs of data to verify the recording operation. This check requires an additional revolution of the disc, adding 50 milliseconds to the rotational time. Average write time is; 110 ms + 75 ms = 185 ms. The total access time for a run is estimated by multiplying the average time per reference by the number of accesses expected. The total access time for obtaining customer names and addresses--assuming the random file technique is used--in the invoicing operation (Figure 2) is: 150 x 135 = 20250 ms or an average of 20 seconds. Because of the possibility of duplicate addresses, some transactions will require more than one access to select the desired record. The overall effect of these extra accesses varies with the situation.

C. SEQUENTIAL FILES

Since sequential files may be referenced by transaction inputs that are either pre-sequenced or purely random, two different timing methods are needed. If as in the invoicing application (figure 2), the transactions are not pre-sorted, average access times are used to estimate run times. Since this operation is essentially random, the access time is 135 milliseconds per record. Because a prior reference to the *index* is required, an additional access is normally required to search the index. However, the special feature for reference addressing provided with Unidisc reduces the average time for this first access to 25 milliseconds. The average access time per record is 160 milliseconds (135 + 25 ms = 160 ms). The total access time for the invoicing application is: 160 ms x 150 = 24,000 ms or 24 seconds.

If references to the file are made sequentially, the average access time per reference will be reduced. In the invoicing example, the name and address file would be stored in first fifty tracks on the disc. Assume for purposes of explanation that there is one detail card for each invoice and that the references will be distributed evenly over the fifty tracks. Then, there will be three references to each track. Further, the tracks will be referenced sequentially, beginning with the first track.

Once the run has started, the access time for any three references to a track will be as follows.

First Record	
Search Index (average time)	25 ms
Move access arm from preceding track (1 track max.)	49
Read Record (average time)	25
	99 ms
Second Record	
Search Index (average time)	25 ms
Move access arm from preceding track	00
Read Record (average time)	25
	50 ms
Third Record	
Search Index (average time)	25 ms
Move Access Arm from preceding track	00
Read Record (average time)	25 ms
	50 ms
Total for 3 records	199 ms

Access arm positioning time is quite low, because the movement is always from an adjacent track. This time is incurred only once for every three records, because the second and third records are on the same track as the first record. Since there are 50 groups of records (3 per group), the total access time for this run is: 199 ms x 50 = 9,950 milliseconds or 10 seconds to be compared to 24 seconds for non-sequential input.

Also worthy of note is the fact that no other distribution of desired records in the tracks can increase the total access time, although it may be reduced. However, record layout can affect read time. This effect is described in the reference manual.

While the presequencing of inputs reduces the access time, it is not an absolute requirement. In fact, the scheduling of runs and the resulting demands on peripheral equipment and operators may make it more desirable to operate with unsorted inputs. Furthermore, the programming requirements for both sequenced and unsequenced inputs are precisely the same, making it possible for the Operations Manager to decide whether or not to sort on a day to day basis.

One other feature of Unidisc that affects throughput rates is its ability to operate independently of the 1004/1005. Unidisc is known as a *buffered* device, because it contains control circuitry and core storage, which relieve the central processor of the task of monitoring random access operations. Once commanded, Unidisc operates independently. Thus, computing and any other input/output function can occur at the same time as a random access function in a manner similar to the way in which the Card Punch can overlap the 1004/1005. In the invoicing application, for example, the access time is not added to the card read time. Only the longer of the two times affects the throughput rate.

UNIVAC

DIVISION OF SPERRY RAND CORPORATION

UP-4093

PRINTED IN U.S.A.