L. C. Hobbs

GENERAL PRECISION | LIBRASCOPE GROUP

808 WESTERN AVENUE, GLENDALE 1, CALIFORNIA

HIGH-SPEED, CONTENT SEARCH IN A LARGE, ROTATING, MASS MEMORY

LIBRASCOPE GROUP
GENERAL PRECISION
RESEARCH AND SYSTEMS CENTER

Prepared by

Don W. Warren

HIGH-SPEED, CONTENT SEARCH IN A LARGE, ROTATING, MASS MEMORY

The Librascope L-1500 Series magnetic disc has 120 one-million-bit data bands with individual, fixed, read/write heads. Transfer and various content-comparison operations can be carried out at speeds of over two million bits per second, independently of the central computer.

Applications of this type of memory to large data base operations for command and control systems will be discussed; and some comparison with parallel search facilities will be made to indicate some of its advantages and limitations.

THE LIBRASCOPE L-1500 SERIES DISC MEMORY

Librascope's L-1530 System is a "Search Memory" in the sense of the Symposium Nomenclature. It is "Content-Addressable", it is primarily serial, and it is fairly fast for a rotating memory — at least for one of multi-million-word capacity.

Physically, (See Fig. 1A and 1B) the disc is 48 inches in diameter, 1/2 inch thick, rotates at 900 RPM, and has a fixed-position read/write head for each data track. Six discs can be mounted in a single module sharing the same drive shaft and much of the logic circuitry. Further, as many as seven "slave" modules can be added to one "master" module without duplicating the search logic — for a total capacity of more than a billion characters.

The vital statistics are given in Figure 2: those just mentioned toward the top, and several more in the table. The 420 data Tracks (one-dimensional, circular strings of bits) are grouped into Track Sets of three; i.e.: three radial bits are always read in parallel.* Two such triplets represent a character, eight characters a word, etc. The semantic organization stems from the definition of BLOCK AND DATA BAND.

Ideally, the Data Band would correspond to a small File or Subfile, and the Block to an Entry or Record, though no great inefficiency is entailed so long as the correspondence can be maintained in multiples: e.g., η blocks per entry or η entries per block. The 21,000-word Data Band might be termed the "unit of search-area". Once the search criteria are specified and the program directed to the desired Band, the search can be completed and all "response" Blocks delivered to core without further attention from the Central Processor — a parallel, asynchronous operation which merely notifies the Central Control must step in to initiate action for each new Band, but this requires only a very few command executions.

*This number can vary from one to six.

The Block is the atomic unit of information within a Data Band. The search can only select or reject a complete Block; any finer, interior discriminations must be programmed in the Central Processor. However, the search operation itself can (for the purpose of selecting or rejecting the whole Block) refer to any specified bit-locations without restriction. This is accomplished by placing the appropriate Argument and Mask in special purpose registers in the Disc's high-speed core buffer.

Two search modes are available. The fastest searches only a special set of "Tags", 18 Tag-characters to a block, which permits searching an entire million-bit Data Band in one 70 Ms revolution. The tags for all Blocks in a Band are stored in a single Track Set as shown in Figure 2. The corresponding 110-character "data sections" of the Blocks are interleaved on the remaining six Track Sets of the Band. This interleaving technique staggers the data in such a way that when the Tag search yields a response, its data section can be read out immediately without requiring another revolution. There is a qualification, however. If two overlapping blocks "respond", only one can be read per revolution; the second is "flagged" * and read out subsequently. The worst possible case is where six consecutive Blocks respond. One could be read on the first pass and the other flagged; then five more revolutions are required to read the remaining bands. In any ordinary search, where the responses are less than 1% or so of the Blocks searched, this case is unlikely; but the system will handle it automatically, scheduling the necessary member of revolutions, whenever it might occur. This worst case defines the maximum search time per Band: 7 revolutions at 15 rps, or about 1/2 second — which is also the time required to load or unload a million-bit Band.

If this search mode, relying solely upon Tags, proves inadequate, it is possible to apply the same Argument-and-Mask technique to the Data Sections of the Blocks. Due to the interleaved storage, this mode would usually require a full seven revolutions, but it permits search on any combination of the 768 bits in a Block or in any selected subsets of these bits. Conventional random-access is provided also. Given the Data Band

*There are two independent Tracks for flags (and one for addresses which were not previously mentioned. (See Fig. 2)

and Block number, the entire Block is retrieved in an average of one-half revolution (35 Ms) and a maximum of one revolution (70 Ms).

Search Types provided include most of the usual comparisons: Equal, Not Equal, Greater than, Less than, and Between upper and lower bounds. These comparisons can be applied to any selected field in the Block.

CORE PROGRAMMED CONTENT SEARCH

To search the content, strictly speaking, of a million-bit file in core storage is not impossible, but is rather inefficient. Consider a file of 1,000 entries of 20 words each, with each entry having fixed length and identical format so that the desired attribute can be addressed directly. Now to search the file, even under these ideal conditions, and using a single key (attribute), requires 1,000 comparisons. A comparison involving at least a couple of additions and a couple of memory accesses - say 20 microseconds (μs) — would require 20 milliseconds (Ms). To search on the conjunction of 10 keys would thus take 200 Ms — about the same as the average for a comparable search on the Librascope Disc.

Of course, the core operation has a flexibility that allows search programs to be tailored to special problems with much greater efficiency. (For example, the set of entries satisfying the first key can be retrieved and then that set searched on subsequent keys; but this procedure is efficient only if there is a known upper bound on the set resulting from any key that is small compared to the original file — otherwise, the storage requirements would double.)

But a somewhat more sophisticated approach is available, based on the construction of index tables (matrices or trees) which may be entered with a key value and will yield the set of addresses of file entries satisfying that value. This requires considerable data preparation and programming, but is quite efficient timewise and imposes moderate storage requirements. The major storage demand is for storing the sets of file-entry addresses associated with each key value. For one key, the sum of these sets, for all values of the key, may be estimated at about 1,000 (the number of entries in the file) if there is little redundancy —

i.e., if, on the average an entry is associated with a single key value. But if, e.g., an entry is associated, on the average, with two key values, double the storage would be required. For ten keys (with 100 values each) and no redundancy, the storage required would be, for 1,000 10-bit addresses, 10^5 bits (with perfect packing of bits into words) or at least 10% added to the file store. With a redundancy factor of 2, the extra storage would double to 20%. (In addition, program storage would add a percent or two to these requirements.) See Figure 4.

PARALLEL SEARCH MEMORY

These memories are not yet on the market, but laboratory results are promising. It appears possible in these memories to simultaneously test the entire content of, perhaps, 2,000 words for all "matches" (=, <, etc.) with a given key word. There are many variants of this basic technique in various stages of development and no evaluation will be attempted here beyond the following assumption: that such memories are feasible and will be available in the near future with approximately the parameters shown in Figure 5.

The major limitation here, in addition to initial high cost, is the small capacity of about 10^5 bits. While there are many applications for such a powerful tool (in programming, indexing, etc.) its use for storing large files does not seem to be on the immediate horizon. Until its size can be increased by orders of magnitude, its use, other than on small files that must be searched very frequently, would have to be by reading in sections from a store of large capacity. But if an economical mass memory is used, two problems must be faced: first, the time lost in loading the search memory, and second, the problem of locating the required section of a large file which must be searched — with a file of only 10^6 bits, this requires selecting one out of ten sections.

Thus, for the near future, any straightforward use of the parallel search memory for large file storage seems doubtful. Nevertheless, it is reasonable to conjecture that its use for storing index tables, similar to those discussed for core memory searching, might prove very promising, particularly in "heavy duty" situations where search time is a critical factor.

MAGNETIC TAPE STORAGE

Magnetic Tape cannot, conventionally, be searched by content or any other means, except for one or two levels of "punctuation" marks that can be inserted to divide the stored data into files and subfiles. And these, of course, can be found only by counting serially along the tape. Tape can be used, however, in a content search procedure, where very large files are needed and extreme speed is not necessary. A fairly simple procedure has been devised, for example, for combining low-cost tape storage with the medium-cost Librascope Disc for moderate speed searching.

Only one disc data-band (10^6 bits) is required. One tape might hold a file of 10^8 bits — 100 band-sized sections. Now if such a file can be conveniently organized into 100 meaningful subfiles, in such a way that a majority of searches can be carried out within a single one of these subfiles, a simple index table can be stored in core to select the desired subfile. This subfile can then be located (by counting markers) and read onto the disc at continuous tape speeds. Searching the subfile can then proceed as a standard disc content-search (requiring 1/10 second or so). The total search, therefore, requires almost exactly the time to move the tape to the desired section.

CATEGORIZATION OF FILE PROBLEMS

Resisting the time-honored approach of touting ones own product as the solution to any and all problems, I will try to define some problems for which the L1530 Disc is not ideal, as well as some for which we believe it is. In practice, we usually are given a specific file problem and must show that our memory device provides an efficient solution. This is not always easy (at least where honesty is one of the ground rules), but it is usually easier than the present problem. What I have attempted is to show that memories are best suited to certain types of problems. This attempt has not succeeded, but perhaps I can offer a few clues to an approach to the problem. The difficulty is that we don't seem to have any standard means of defining problem types. In fact, I have not been able

to discover even a procedure for categorizing files. (We usually mention their size and then give a sample format.) Let me, therefore, suggest (Figure 6) some factors that might be useful in such categorization.

The list of "File Parameters" (Fig. 6) gives three "static" parameters exemplified in the table and three "dynamic" parameters concerning how the file is to be used. The table introduces some sample file structures of various sizes. The first three rows (E1, E2, E3) define alternative 'entry' sizes — an entry being the basic meaningful word set (element, unit, or record) defined for a given file. The last four rows (F1-F4) are some possible subfiles — a subfile being a set of entries or other subfiles that are related by a significant semantic concept (e.g., a Military Equipment file might contain subfiles of Aircraft, Trucks, Ordnance, etc.). The ramifications of subfile structure deserve further investigation. For present purposes, however, the important aspect of structure is to allow most searches to be carried out within one subfile. Admittedly, this is a very "strong" and restrictive definition of 'subfile' but for a first rough cut at the problem, it adds great simplification.

The sizes chosen have no special significance other than to offer concrete examples within a reasonable range. The "F" columns define five file structures ranging from small to large and with varied depth of structure — cf: F4A and F4B for two structures in the same size file. Again, there is no special significance in these particular files other than to offer a few varied samples. (They may, however, suggest some of the difficulties: Whether or not to call a file "deeply structured" without qualification as to size of file and size of entry, for instance).

The details of this little analysis are not worth belaboring beyond the point of emphasizing that there is a problem of File Categorization here worthy of further study and some such approach as this might serve as a start.

The last chart (Figure 7) uses a condensed version of this File Problem Categorization to indicate six types of File Problems that might be associated efficiently with certain memory types. The levels "Lo, Medium, Hi", must be assumed intuitively obvious (even if they are not). A blank box

indicates that that parameter is not restrictive for the associated Memory Type. The two parenthetical entries '(Hi)' indicate a restriction in the special sense that only Hi Usage would justify the high cost of these memories. The table is obviously incomplete. Even with these condensed categories and level, 243 Problem Categories could have been defined. Only a few samples that appear intuitively to be of special interest are presented. The largest size file, if not highly structured, for example, is assumed to be a problem for which there is presently no efficient solution.

Finally, to indicate briefly the rationale behind the assignments in Figure 7, The Parallel Search Memory can handle only small files and its cost can be justified only if usage is high.

Requirements for the Core Programmed Search are somewhat similar. File size can be moderately larger, but efficiency decreases drastically if many combinations of keys are used since this requires preparation and storage of many voluminous indexing tables. Lack of subfile structure has the same effect to a lesser degree, and retrieving many entries per search, of course, raises the access time linearly, thereby lessening the speed advantage over Disc Search.

The Librascope Disc is most appropriate for a medium size file with enough subfile structure (Medium) to permit direct selection of data bands. With a multiple disc system, large files can be handled, similarly, with moderately high speed.

Lastly, Magnetic Tape or Drum storage might be combined with the Librascope Disc (or with a Core Program if problem characteristics warrant) to permit reading sections (subfiles) into a searchable memory as described previously. This technique trades speed for economy.

FIGURE IA. LIBRASCOPE L-1500 DISC

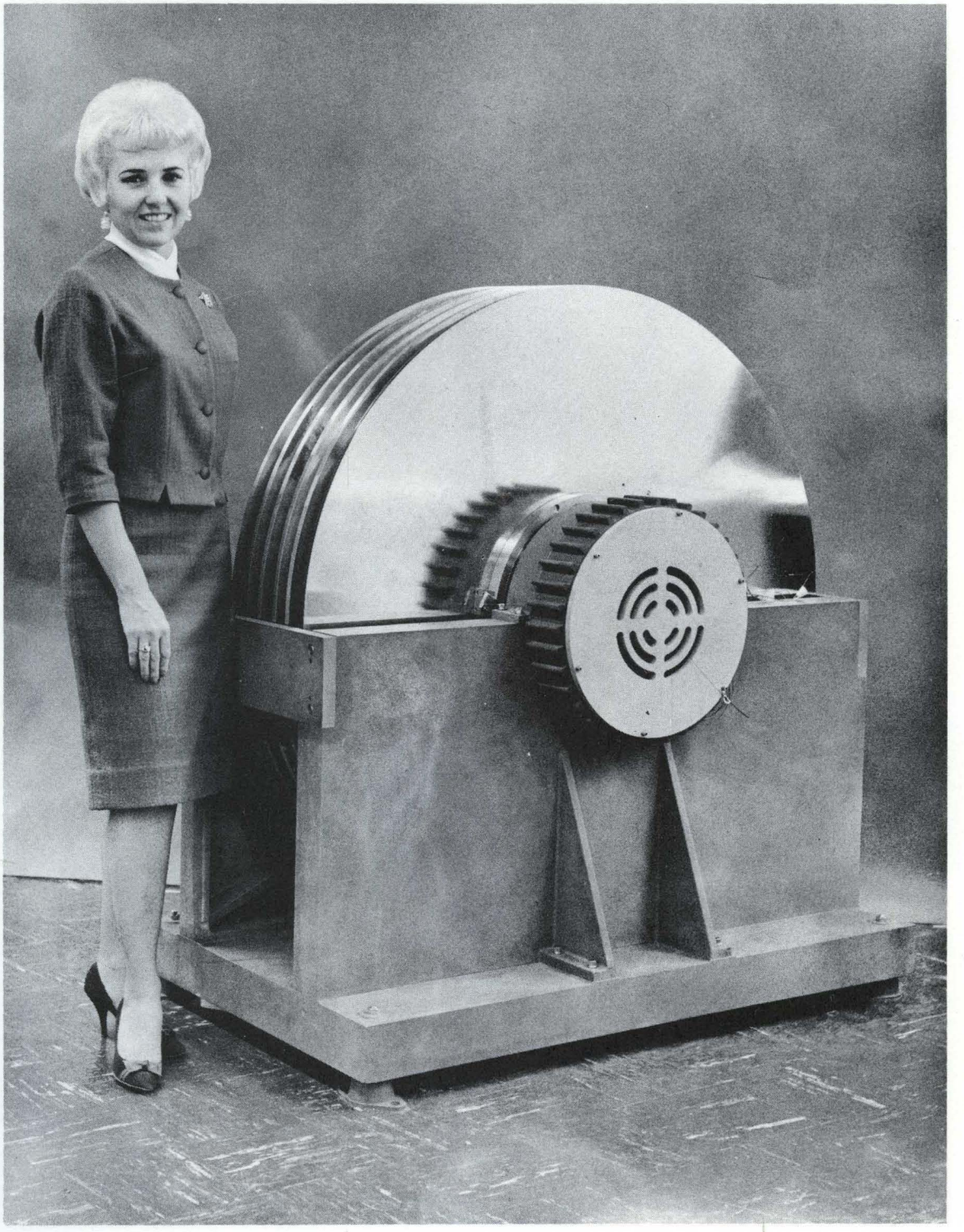

FIGURE IB. LIBRASCOPE L-1500 DISC

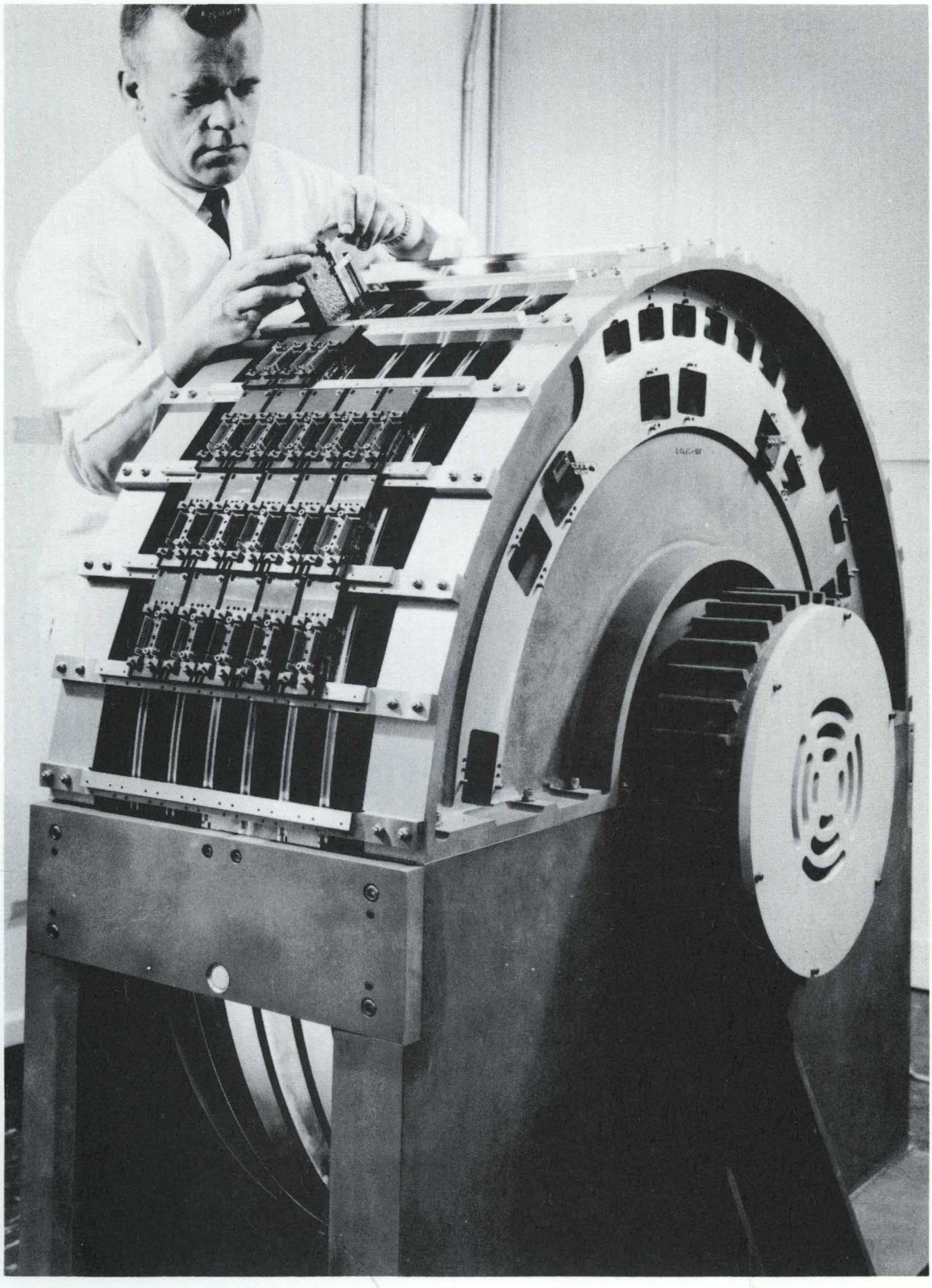

FIGURE 2. LIBRASCOPE L-1500-SERIES DISC-PARAMETERS

PHYSICAL CHARACTERISTICS

48 IN DIAM. - ~ 500 FIXED-POSITION READ/WRITE HEADS

900 RPM (15RPS) - 6 DISCS PER MODULE (1 SHAFT) - 7 'SLAVE' MODULES PER MASTER

DATA STRUCTURE

	BITS	CHARACTERS	WORDS	TRACKS	T-SETS	BLOCKS	DATA BANDS	DISCS
CHARACTER	6	1						
WORD	48	8	1					
TRACK	50×10^3	8.3×10^3	10^3	1				
TRACK-SET	150×10^3	25×10^3	3×10^3	3	1			
BLOCK	768	128	16	*	*	1		
DATA BAND	10^6	167×10^3	21×10^3	21	7	1350	1	
DISC	20×10^6	~~28×10^3~~	0.4×10^6	420	140	27×10^3	20	1
DISC MODULE	120×10^6	20×10^6	2.5×10^6			162×10^3	120	6

*A Block subdivides a Data Band cutting across all 21 Tracks.

FIGURE 3. LIBRASCOPE DISC - ORGANIZATION OF ONE DATA BAND

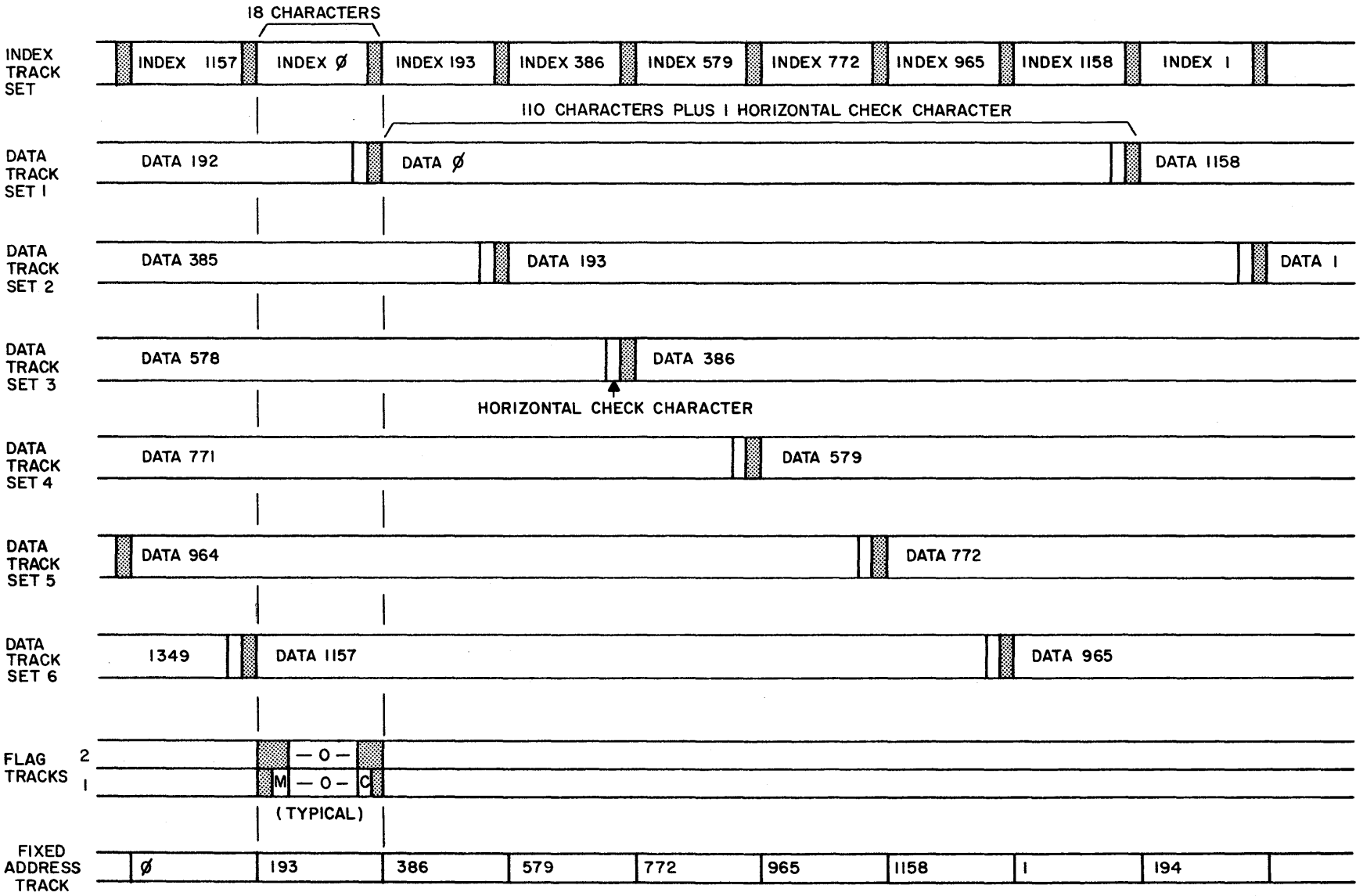

FIGURE 4. A SIMPLE SCHEME FOR CORE SEARCH BY CONTENT

VALUES OF K_1	0	1	2	3	••	••	••	••	••	••	i	••	••	••	••	••	99	100
ADDRESSES OF	A_{10}	A_{11}	••	••	•	••	••	••	••	••	A_{1i}	••	••	••	••	••	••	A_{199}
FILE ENTRIES	A_{20}	A_{21}	••	••	•	••	••	••	••	••	A_{2i}	••	••	••	••	••	••	A_{299}
CORRESPONDING	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
TO K_1 VALUES.	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
AVG $n_i = 10$	A_{no}	A_{n1}	•	•	•	•	•	•	•	•	A_{ni}	•	•	•	•	•	•	An_{99}

SAMPLE KEY VALUE-FILE ENTRY ADDRESS TABLE

FILE HAS 1,000 ENTRIES; ENTRY ADDRESS HAS 10 BITS.

THERE ARE 10 SEARCH KEYS, $K_1 \ldots K_{10}$, EACH WITH 100 VALUES.
EACH KEY VALUE YIELDS (av.) SET OF 10 FILE ENTRY ADDRESSES.

$\therefore$ 10 BITS × 10 ADDRESSES × 100 SETS PER KEY × 10 KEYS = 10^5 BITS

THIS ADDS 10% TO STORAGE OF A 10^6-BIT FILE.

(PROGRAM WOULD ADD ANOTHER 1 OR 2%.)

FIGURE 5. MEMORY TYPES

(VALUES ARE APPROX. ORDER OF MAGNITUDE ESTIMATES)

PARAMETERS	LIBRASCOPE L-1530 DISC	CORE	PARALLEL SEARCH	MAGNETIC TAPE
SIZE MILLIONS OF BITS	120 ~~(5,000)~~	1 (3)	0.1	120 (5,000)
COST ¢ PER BIT	0.2	10	?>100	0.05
ACCESS SPEED IN FILE OF 10^6 BITS: SERIAL READ	1 sec	0.1 sec	0.2 sec.	3 sec
SINGLE WORD	35 m sec	5μs/word*	10μs/word	3+ sec*
n WORDS	70 ms to 1/2 sec	5μs/word*	10μs/word	?>3 sec

*Not including search time which is widely variable (~100μs) increasing, probably, as log n.

FIGURE 6. FACTORS IN CATEGORIZING FILE TYPES

FILE PARAMETERS

STATIC
- Total Size
- Entry Size
- Subfile Structure

DYNAMIC
- NO. OF ENTRIES FOUND PER SEARCH
- NO. OF SEARCH KEYS AND COMBINATIONS USED
- USAGE: FREQUENCY OF SEARCH AND UPDATE

SAMPLE FILE STRUCTURES

		BITS	WORDS	F_1	F_2	F_3	F_{4A}	F_{4B}
ENTRY	E_1	10^2	2	1,000				
	E_2	10^3	20		1,000	1,000		1,000
	E_3	10^4	200				100	
SUBFILE	F_1	10^5	2K	1				
	F_2	10^6	20K		1	100	100	10,000
	F_3	10^8	2M			1	100	
	F_4	10^{10}	200 M				1	1

FIGURE 7. ASSIGNMENT OF FILE PROBLEMS TO MEMORY TYPES

FILE SIZE	SUBFILE STRUCTURE	ENTRIES PER SEARCH	KEY COMBINATIONS	USAGE	MEMORY TYPES
LO				(HI)	PARALLEL SEARCH
MED	MED	MED	LO	(HI)	CORE PROGRAM
MED	MED			MED+	L-1530 DISC
HI	HI			MED+	L-1530 DISC - MULTIPLE
				LO	L-1530 DISC AND TAPES
		MED	LO	MED	CORE PROGRAM AND TAPES OR DRUMS