LA-3546-MS

LOS ALAMOS SCIENTIFIC LABORATORY
of the
University of California
LOS ALAMOS • NEW MEXICO

The Administration of Secondary Storage in Nuclear Codes

UNITED STATES
ATOMIC ENERGY COMMISSION
CONTRACT W-7405-ENG. 36

Printed in USA. Price $3.00. Available from the Clearinghouse for Federal Scientific and Technical Information, National Bureau of Standards, United States Department of Commerce, Springfield, Virginia

LA-3546-MS
UC-32, MATHEMATICS
AND COMPUTERS
TID-4500

LOS ALAMOS SCIENTIFIC LABORATORY
of the
University of California
LOS ALAMOS • NEW MEXICO

Report written: June 14, 1966

Report distributed: August 17, 1966

The Administration of Secondary Storage in Nuclear Codes

by

William J. Worlton

ABSTRACT

The class of problems to which a particular computer can be applied is closely limited by the specifications of the computer's storage hierarchy and the organization of the data flow. The storage requirements of nuclear codes frequently exceed the capacity of the main storage, and it is then crucial to be able to address further levels of storage without large discontinuities in access time and transmission rate. The most exacting requirements occur when very little computing occurs between accesses to secondary storage. Recent developments in computer storage technology and design offer systems which do not have large discontinuities in the storage hierarchy. These developments include Extended Core Storage (ECS), parallel disks and drums, wide tapes, and magnetic strip transports. Extended core storage is particularly important because it can be used to interface the main storage with very little gap in access time and transmission rate. In spite of these developments, the flow of information must still be programmed from one level to another because the addressing technique is different for each level of the hierarchy. This burden can be mediated for the user by the use of a "single-level" storage system, a concept pioneered in the Atlas computer. The storage

requirements of two- and three-dimensional nuclear calculations will continue to exceed the available systems, and the careful administration of the storage system will be necessary to avoid severe limitations in the development of these programs.

1. THE IMPORTANCE OF SECONDARY STORAGE

The selection of the storage system for a computer is a crucial part of matching performance to requirements in computing systems. It is a common error to focus attention only on the central memory, rather than to place this memory in its context of actual usage with other forms of storage. The flow of information throughout the system must be analysed in order to achieve a correct balance between problem requirements and storage capability. For example, the discontinuity between the speed of main memory and the speed of the tapes on the 704 imposed such a severe limiting factor on problems which exceeded the main storage that this actually caused a threshold in problem development beyond which users seldom ventured. Oddly enough, the relative nature of this gap has remained, even though both main memory and secondary storage speeds have been improved. The 7030 main memory can be read in 1.2 μsec, but access to the disk is 4 to 5 orders of magnitude more expensive in time; drums could provide a somewhat better interface with this memory, but even here the discontinuity would be 3 to 4 orders of magnitude.

An ideal storage system for a computer would have a single addressing method, a uniform access time, a cycle time commensurate with the cycle time of the Central Processing Unit (CPU), a capacity sufficient to hold all information required, and a reasonable cost. It is possible to build a storage system with all of these characteristics except the last: a billion-bit main memory could be fabricated, but it would cost on the order of $1 per bit.[1] Since it is thus an economic impossibility to satisfy large storage requirements by extending main memory, a hierarchy of storage devices of larger capacity but lower capability and cost must be resorted to, and this raises a number of problems.[2] First, problem preparation is more complex because of the need to address several types of storage and to move data to the fastest member of the hierarchy. Second, there is a greater likelihood of error in the calculation because secondary storage devices are less reliable than main memory. Third, the calculation time is increased, because of the slower access times and transmission rates of secondary storage. These problems imply the need for great care in both the selection of equipment and the administration of available storage space if artificial barriers are not to be erected for both running time and problem development.

The importance of secondary storage has been increasing, and will evidently continue to increase for some time. Consider the <u>frequency</u> of access to secondary storage: this is proportional to the speed of the computer, inversely proportional to the main memory capacity, and proportional to the volume of information required. All other things being equal, when computers differ only in speed, such as the 7094 and the 7094-II, the faster one will have a higher access frequency. Similarly, the smaller the main memory, the more frequent must be the use of secondary storage. If 100,000 words must be processed in blocks of 1,000 words due to limitations in buffer size, then 100 accesses are necessary; if the buffer size could be increased to 10,000 words, only 10 accesses would be necessary. Finally, if computer speed and memory size were held constant and the volume of information handled by a program were increased, the frequency of access to secondary storage would increase. This has in fact been happening due to the application of computers to a broader class of problems. These three parameters do not vary at the same rate, however. Computer speed has been increasing faster than main memory capacity, as can be seen in Figure 1. If speed and memory size had increased at the same rate, the points of this figure would all fall on the "line of equal development"; in fact, they fall considerably above this line. The balance between speed and memory capacity can be compared easier if the speed is divided by the memory capacity (in 704 units) as in Figure 2. Computers falling on the same horizontal line have the same balance between speed and memory capacity; on a vertical line, the speed is out of balance to the capacity in the higher points. Note that the highest speed computers are tending to greater imbalance, emphasizing the importance of secondary storage.

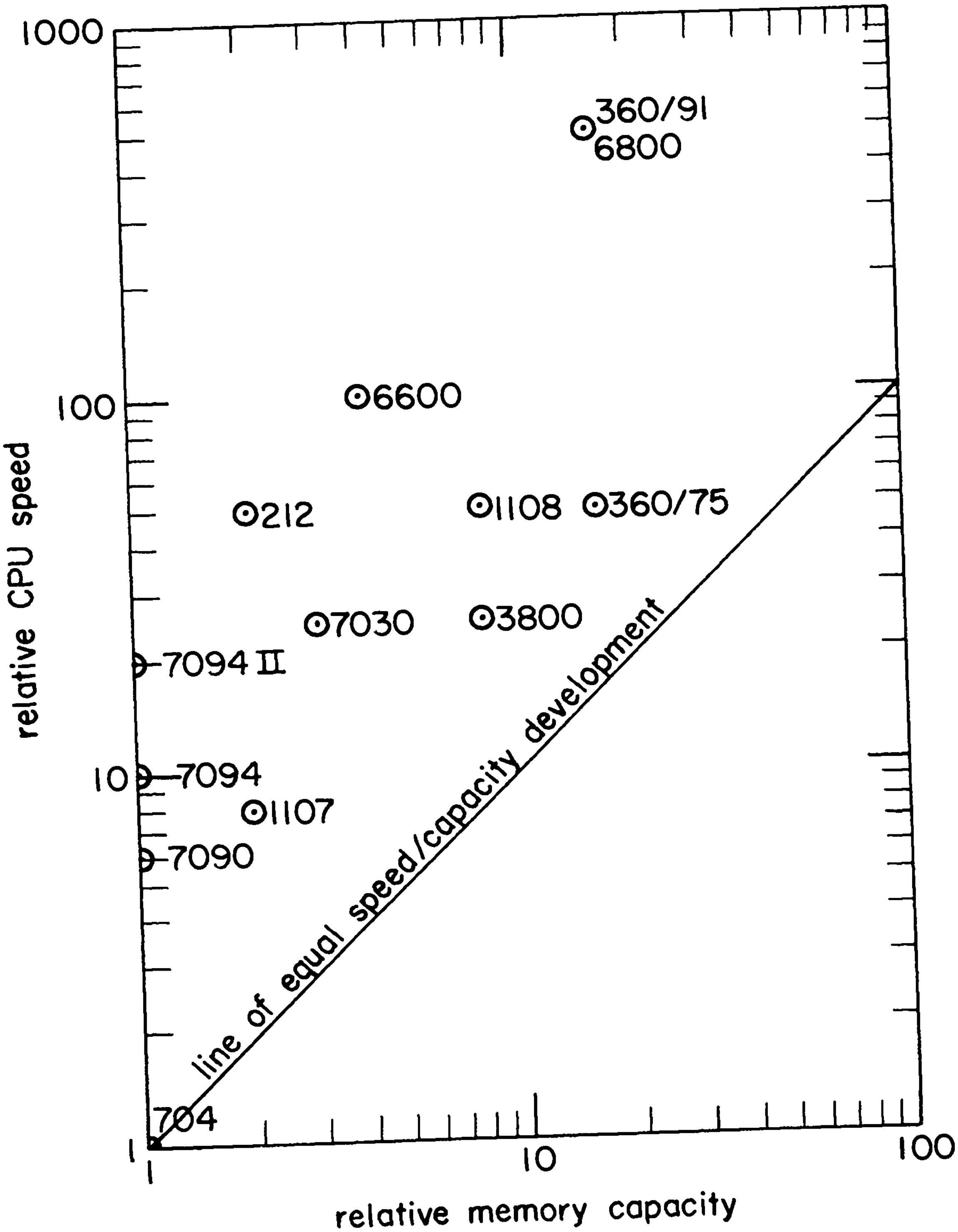

Figure 1. Relative CPU-Speed vs. Memory Capacity

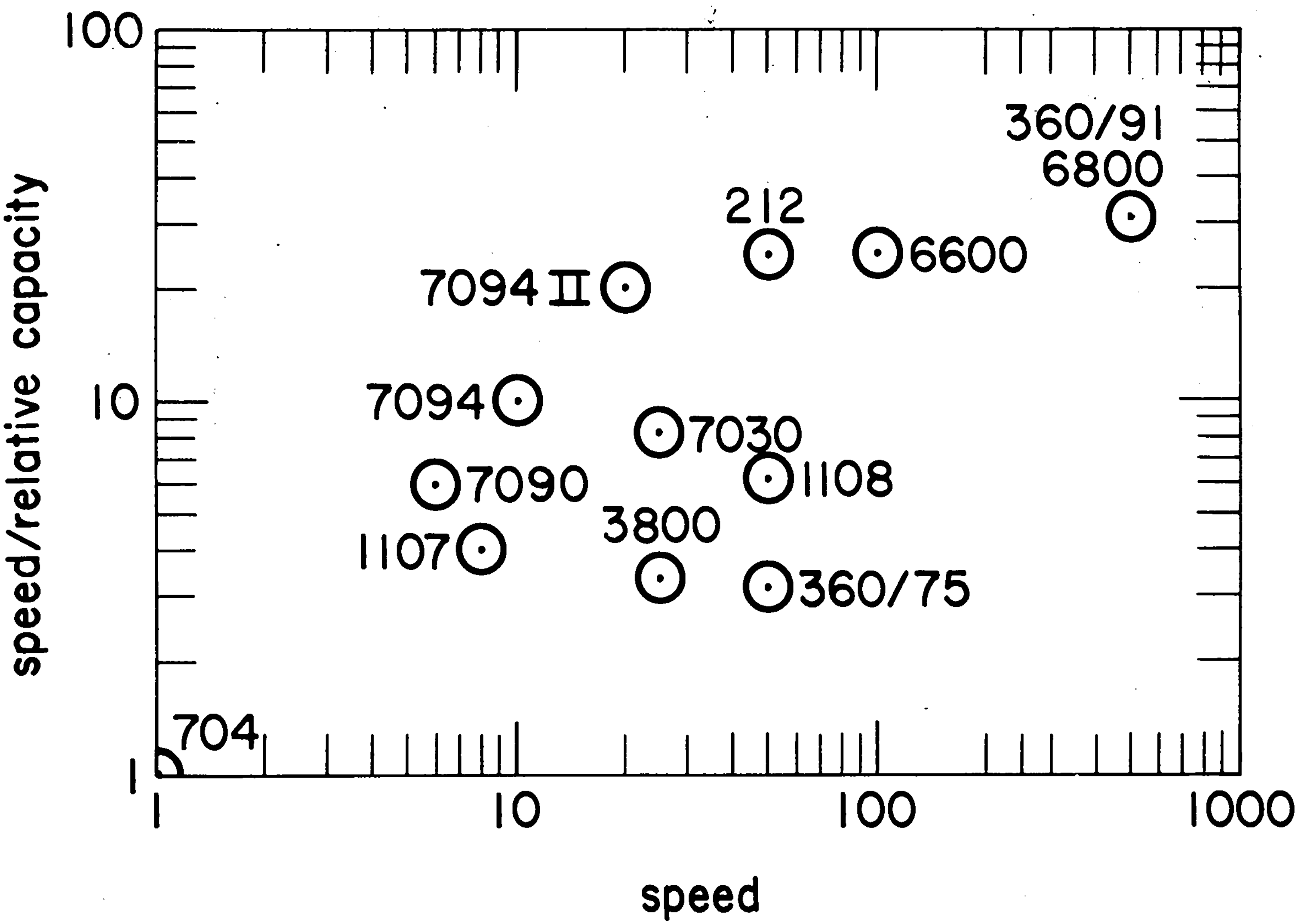

Figure 2. Normalized Speed/Capacity Balance

2. REVIEW OF MAIN MEMORY DEVELOPMENTS

Because main memory is so important in judging secondary storage requirements, it may be well to review recent developments in main memory design. Figure 3 shows the trend in available memory capacity relative to the year of first delivery of typical scientific computers. Computers with a 32K memory capacity dominated the field from the days of the 704 to about 1962, but since then there has been a marked increase in the main memory capacity offered with new designs. Currently available computers offer capacities from 32K to 512K, but actual installations are more often in the 32K to 128K range due to the relatively high cost of main memory. This diagram is somewhat deceptive in that a trend line would seem to extrapolate to higher and higher capacities in main memories, but in practice there is some hesitation in making this prediction.

These same computers show a trend toward lower main memory cycle time, as illustrated in Figure 4. In less than a decade, memory cycle times have decreased about an order of magnitude, and this trend is expected to continue; we should see 50 to 75 nsec memories offered with computers in the next decade.[3] The actual cycle times shown in the figure have been effectively decreased even further by interleaving several independent boxes of memory, which in the most favorable case (no access conflicts) provides access to memory in the read time, which is usually less than half of the full cycle time. Furthermore, when

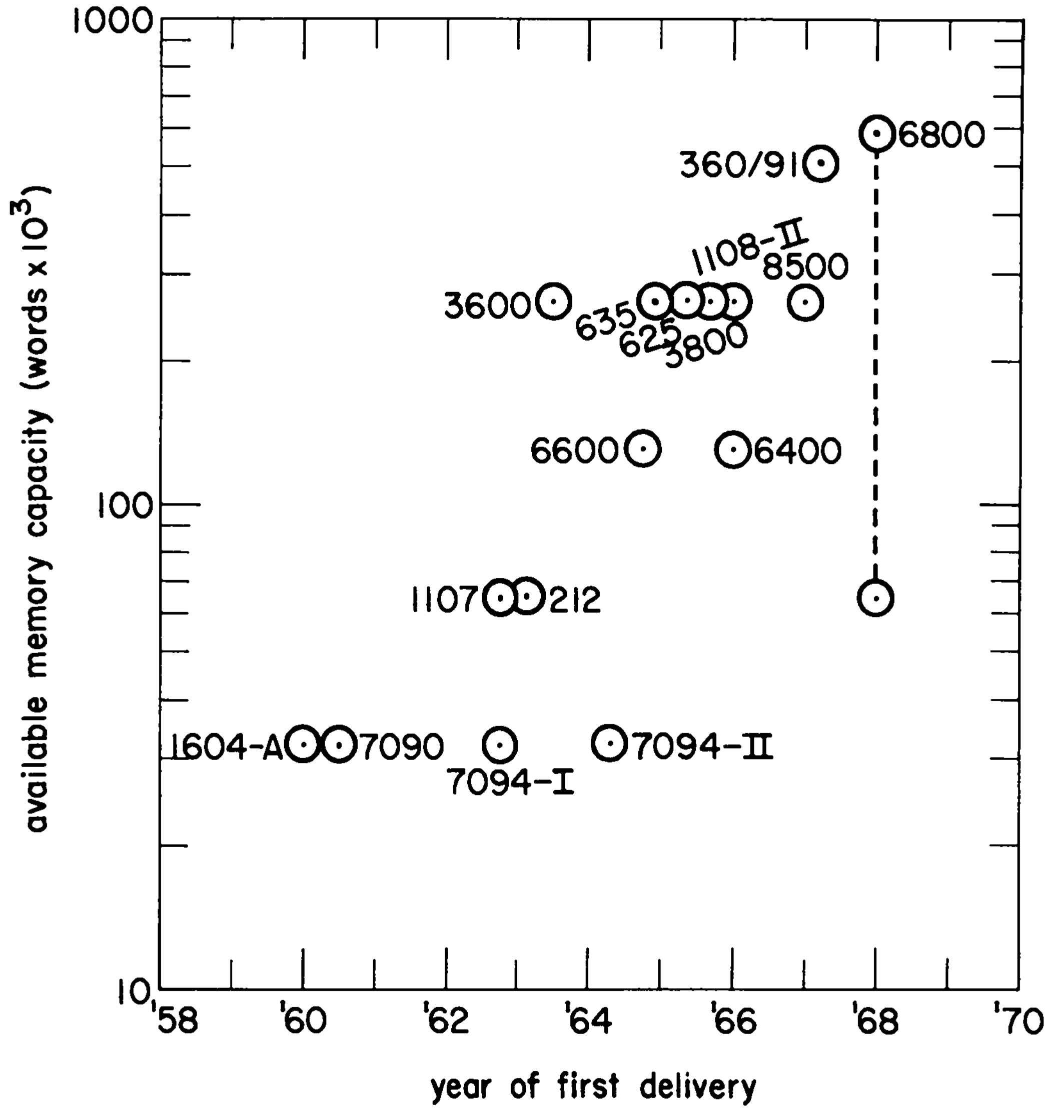

Figure 3. Trend in Available Memory Capacity

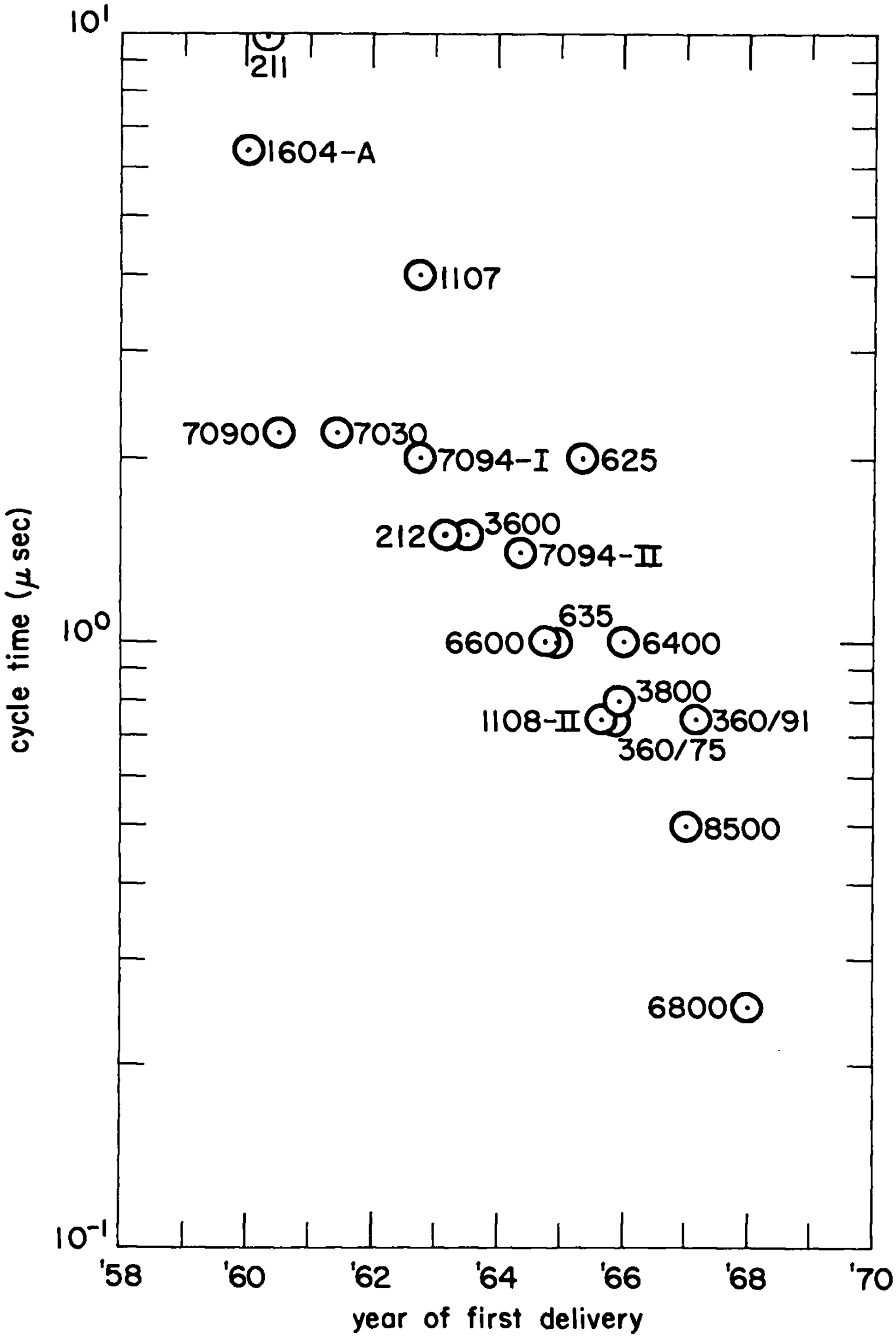

Figure 4. Trend in Memory Cycle Time

many accesses are made to independent boxes of memory, the transmission rate is frequently equal to one word per bus slot; this is about an order of magnitude higher than the reciprocal of the full cycle time. The maximum transmission rate from a memory is the number of independent boxes divided by the cycle time; this upper limit is restricted by the bus time, which is usually equal to one word per CPU cycle time, and instruction issuing restrictions.

The parallel development of higher capacities and lower cycle times is illustrated in Figure 5. There has been a long line of progressively faster 32K memories. There is a rather well established trend from this area to higher capacities at lower cycle times. The three bulk core storages in the figure (indicated by squares) equal, and even surpass, the main memory capabilities of just a few years ago. Speed and capacity may well be running head on into each other, however, and this is the message in the dashed line for the 6800. This system offers a 64K central memory, with a closely integrated bulk memory which is actually faster than the central memory in some applications. This may well be the trend for the future: a limited capacity central store used mainly for programs and a closely integrated bulk store used largely for data. It is a fact of memory design that a small memory can be made faster than a large memory with the same technology, and CPU speed has at the moment outdistanced memory speed. To obtain a main memory commensurate with CPU speed, an obvious solution is to build smaller main memories and extend this memory with a bulk core. Another

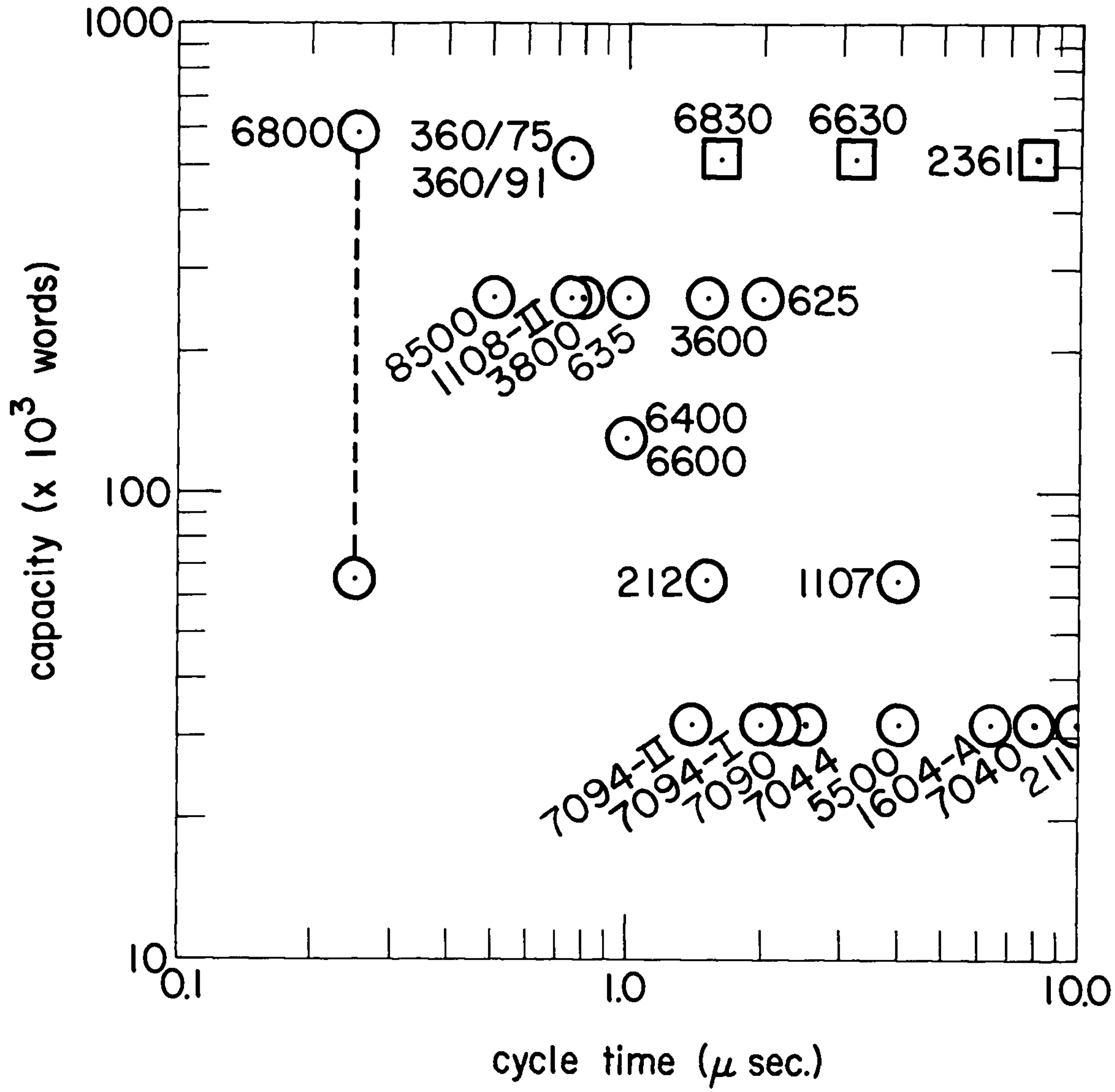

Figure 5. Memory Capacity and Cycle Time

indication of this incompatibility of speed and capacity is the fact that the 360/91 is designed to be about as fast as the 6800, but it uses a memory several times slower. We may well be faced with a choice between speed and capacity, and this will probably lead to a finer structure of main memory: a small, very fast local memory; a larger, but slower, main memory; and a very large bulk storage. This would imply more effort in the administration of main memory and is not very attractive compared to a large memory of uniform access.

A final trend to be aware of is the decreasing cost of main memory. Figure 6 shows the decrease of main memory cost per bit over the last six years. Present costs are clustered around 20¢ per bit, or about $12.50 for a 64-bit word. Even at these lowered costs, million-word memories are out of the question, and even 512K word memories are difficult to justify economically.

These three trends of larger memory capacities, faster memory speeds, and lower memory costs would almost justify Leibnitz's belief that this is the best of all possible worlds! It is easy to become blase and take these achievements for granted, but the computer manufacturers have done a remarkable job of advancing the state of the art and still retaining economy in their products.

3. STORAGE SYSTEM ORGANIZATION

<u>3.1 Central Memory Organization</u>. The conventional computer organization uses the main memory for all transfers of information to or from each

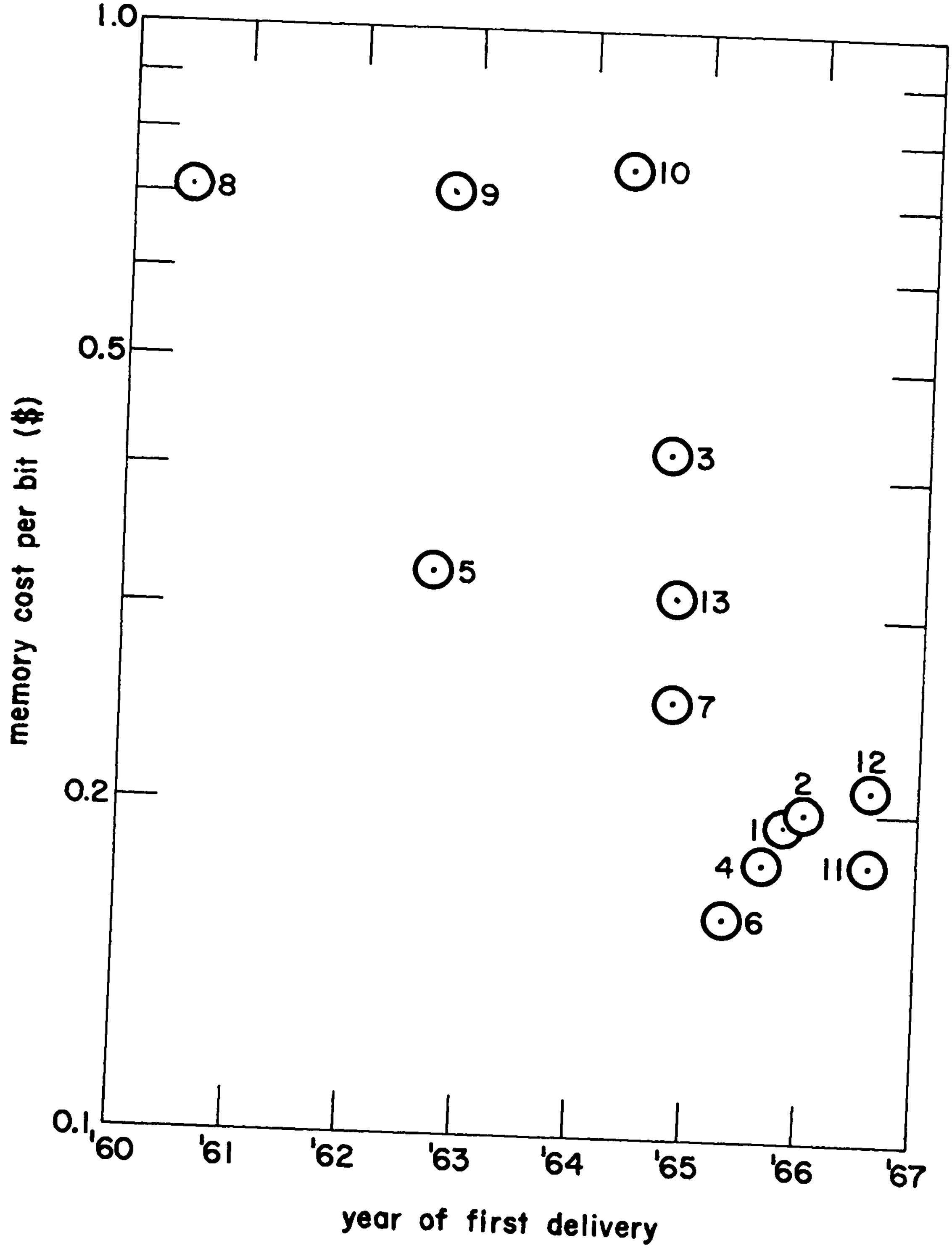

Figure 6. Trend in Main Memory Costs

member of the storage hierarchy, as illustrated in Figure 7. If it is desired to move information from tapes (say) to the disk, it is first necessary to read the information into main memory before writing it onto the disk. The main memory is thus the center of the traffic flow, and this causes several problems. First, the size of the main memory is effectively reduced because space must be allowed for buffers and software systems to control the traffic. Second, the speed of the memory is effectively slowed down because the extra traffic causes memory access conflicts. Third, the speed of the CPU is effectively slowed down, since the CPU must control the traffic, and the software system overhead detracts from problem execution time. Several techniques are available to improve on this organization.

3.2 Hierarchical Organization. Opler[4] has suggested an organization which he calls hierarchical. In this organization, the capability to transfer information to or from main memory is retained, but an additional capability for interdevice communication is added. Thus, strip devices would transmit directly to disk, disk could transmit directly to drum, drum could transmit directly to bulk--without going through main memory. The advantage of this organization is that it reduces the traffic through main memory and eliminates the need of main memory buffers. The disadvantage is that it requires special hardware to implement, e.g., interdevice data channels, and buffers and controls to mediate differences in transmission rates. No such design exists, and the idea is difficult to evaluate.

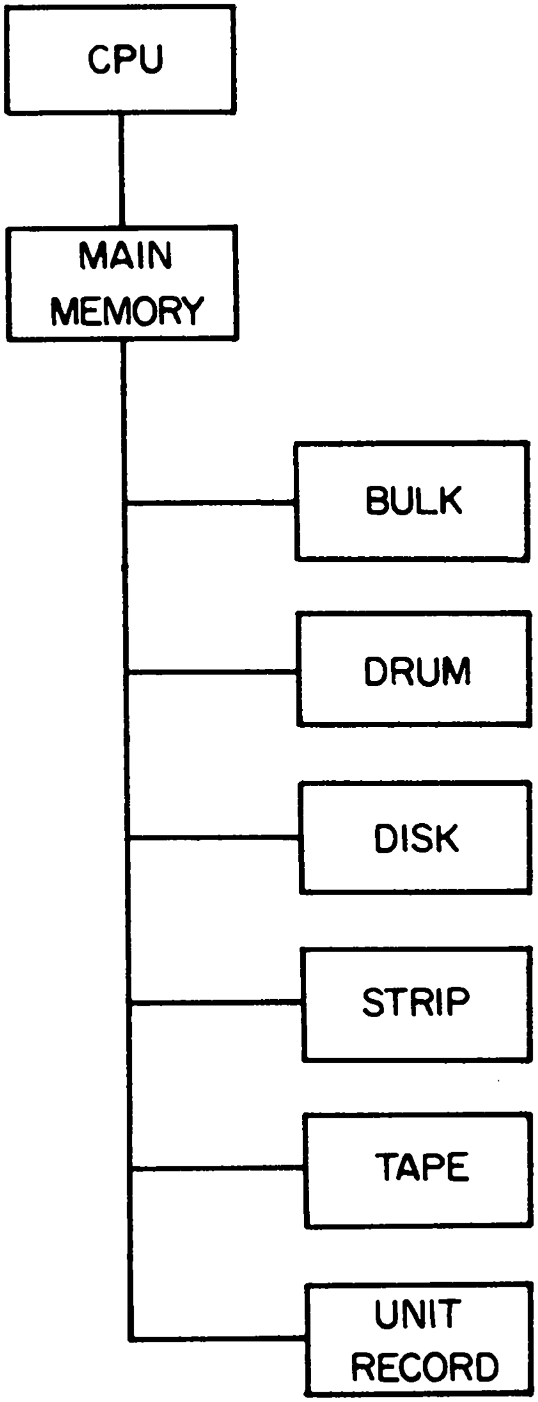

Figure 7. Central Memory Organization

3.3 Bulk-Centered Organization. Bulk storage can be used to mediate the flow of information throughout a system in a more efficient manner than the central memory organization, as illustrated in Figure 8. Here information can flow directly to or from the bulk storage in addition to main memory, and this eliminates much of the traffic from main memory and moves the buffers out of the main into bulk storage. By anticipating the flow of information within a calculation, it is sometimes possible to eliminate the delays inherent in the slower devices by moving the information to bulk ahead of time. This gives the effect of bulk speeds for all secondary storage requirements but takes considerable effort in optimum programming by the user of a very efficient software system. The IBM 360/91 has essentially this type of organization.

3.4 Coupled-Computer Organization. Coupled-computer organizations are not new, but their importance may well be increasing in the design of ultra-high-speed computers, for the simple reason that it takes a separate computer to handle I/O for a fast scientific computer. A representative design is illustrated in Figure 9. The interface between the computers may be as simple as a transmission line, as in the 7094/7044 system; it may be a disk, as in the 7090/1410 system; it may be bulk storage, as in a 6600/6411 system; or, it may be a combination of these. The basic advantage here is that the control of information flow is handled by a separate computer, thus eliminating the overhead time for the large scientific processor. The power of a large-scale

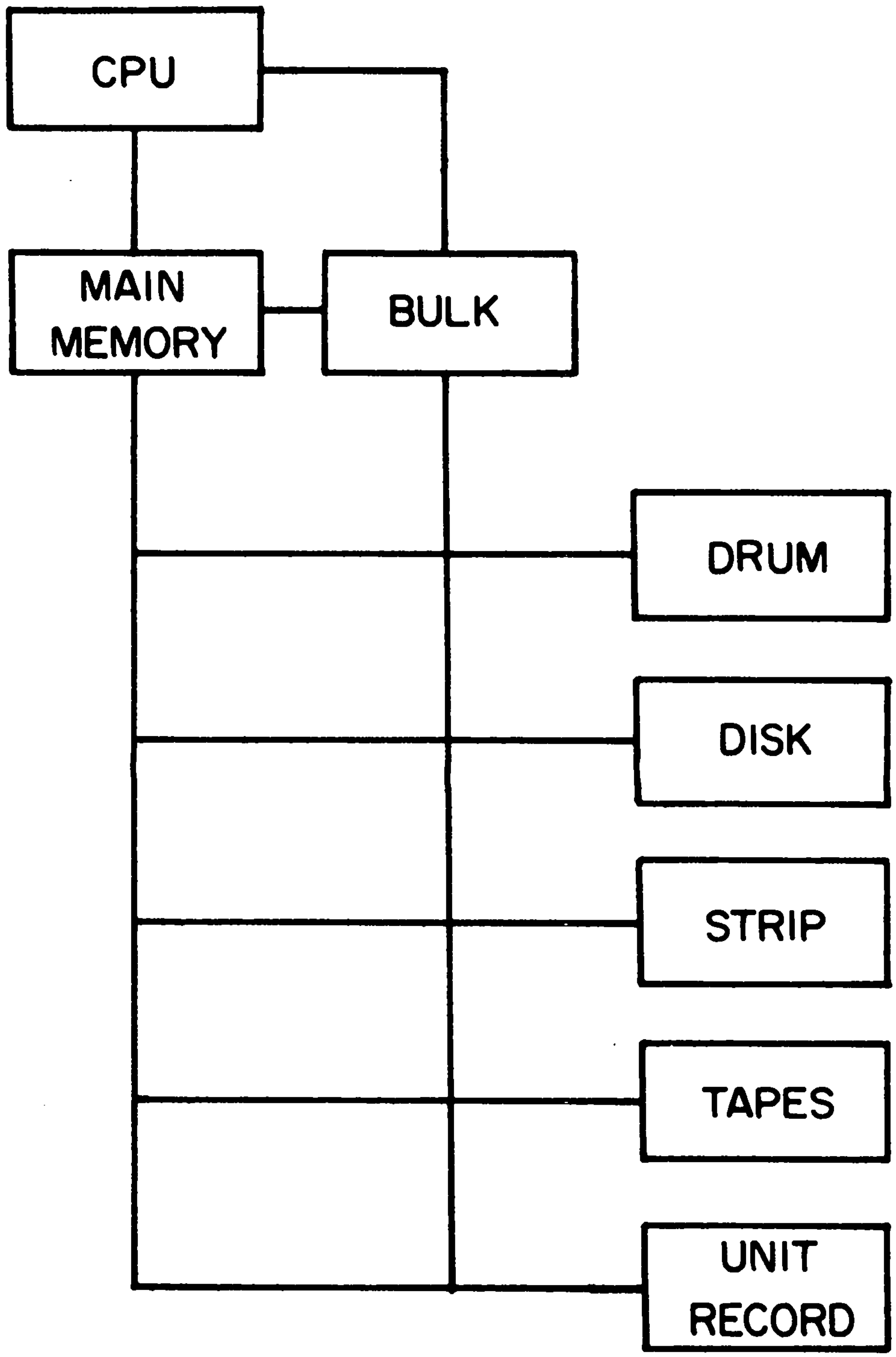

Figure 8. Central/ECS Organization

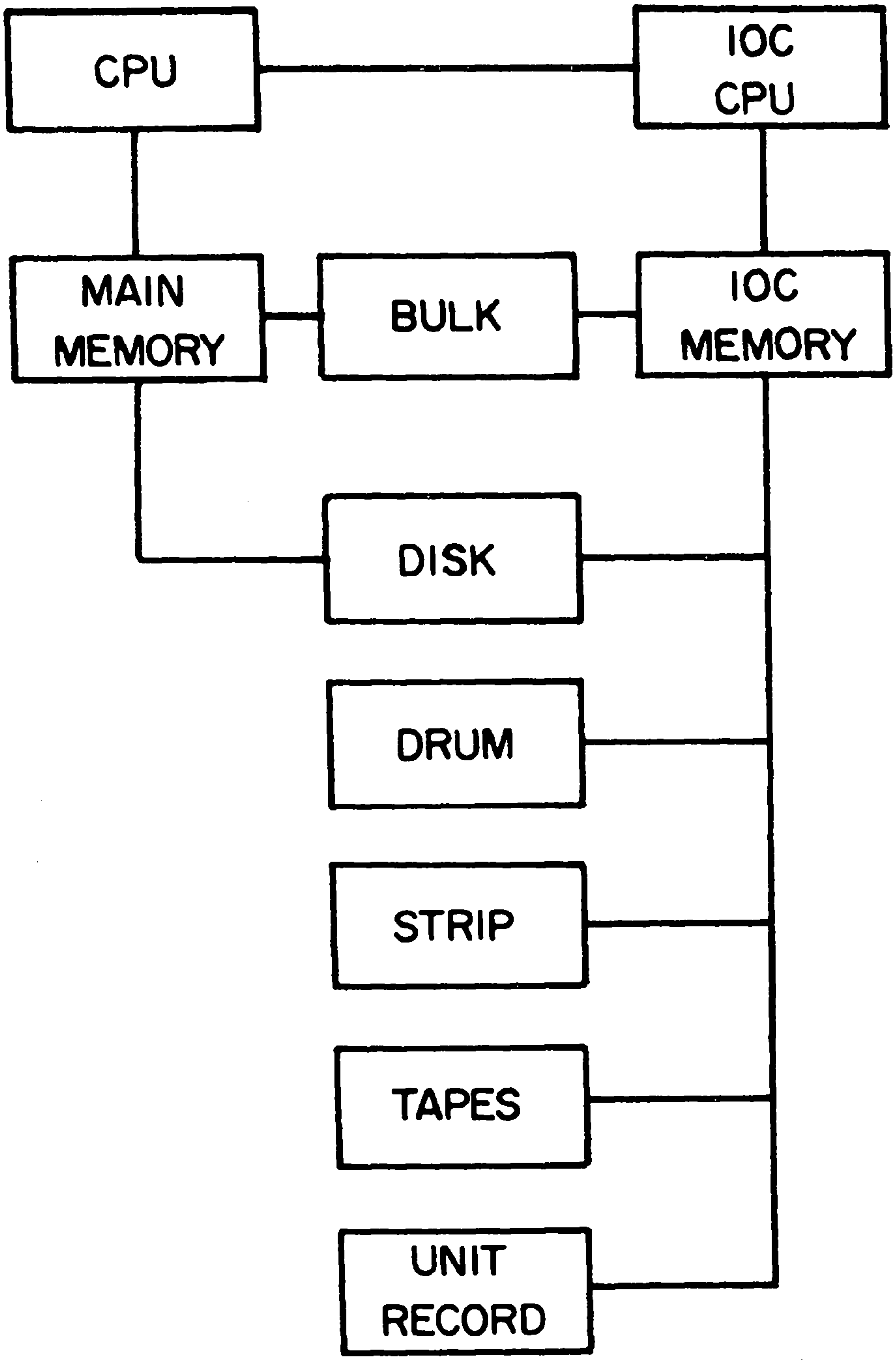

Figure 9. Coupled Computer System

processor is not needed to control secondary storage and I/O traffic, and it can devote its power to high-speed calculation. Another advantage is that it moves much of the software out of main memory. The smaller computers range in size from just a message switching center up to full-scale computers which can perform conversions and compilations.

3.5 One-Level Storage Organization. Kilburn[5] and his associates at the University of Manchester designed a storage organization for the Atlas which allows the user to address secondary storage as though it were all part of main memory. Main memory is arranged in 32 blocks of 512 words, called "pages," and associated with each page is a Page Address Register (PAR), which identifies the current contents of that page. The upper portion of a memory address is compared with all 32 PAR's; if a match is found, access to that page is allowed. If no match is found, an interrupt to a supervisor program occurs, and the supervisor compares the page identification with the table of contents of a high-speed drum. One page is always left empty in main memory; as soon as the drum page with the correct identifier is available under the reading heads, this page is read into memory. The PAR for this memory page is then updated, an empty memory page created by writing out another page onto the drum, and the table of contents of the drum updated. The selection of which page to remove from memory is very important for overall efficiency, and this decision is made on the basis of "use" digits associated with each page which are used to determine which page is the least active.

Although the Atlas design uses this "paging" method for only the main memory and the drum, the method can be extended to a full hierarchy of secondary storage devices, allowing the user to address the full capacity of all secondary storage as though it were part of main memory.

There is a potential penalty for this convenience, however, since the access time must still be paid for the information. If the user ignores the need to optimize the use of main memory and uses secondary storage for random access as though it were main memory, he will find this organization convenient but inefficient. The algorithms for determining which page to remove from memory can never replace the user's knowledge of which page is needed next, and this can lead to a page being removed for one access, and returned for the very next instruction--which effectively turns the computer into a drum computer. Several methods to improve the efficiency of the one-level storage concept are known: (1) to allow the user to dynamically specify the expected activity level for his assigned pages, (2) to use multiprogramming to cover the secondary storage access and transmission times, and (3) to use content addressable PAR's so that the comparison of an address with memory contents is essentially instantaneous.

In summary, the one-level storage organization offers automatic relocatability of storage and the convenience of addressing all storage devices as though their contents were in main memory. Great care must be taken to avoid inefficiency, however.

4. TECHNIQUES TO MINIMIZE THE USE OF SECONDARY STORAGE

4.1 General. The most efficient way to use secondary storage is: not at all. While this may sound facetious, it is intended quite seriously. The use of secondary storage is inherently more expensive in time than the use of main storage, hence its use is never justified until optimum use has been made of main storage.

4.2 Dynamic Storage Assignment. Probably the most consistent and flagrant waste of main memory occurs because of the use of fixed-dimension statements, mostly in Fortran. The importance of this type of waste increases as problem size and complexity increase, as shown in Figure 10. Consider a dimension statement in which 100 words are reserved, but only 70 are commonly used. In a one-dimensional array, the waste is 30%; in a two-dimensional array the waste is 51%; in a three-dimensional array the waste is 66%; and in a four-dimensional array the waste is 76%.

This type of waste is really inexcusable, because it could be avoided by systems programming, such as the following Fortran capability:

```
READ (5,4) I, J, K
DIMENSION A(I,J), B(J,K), C(I,K)
```

We need to convince systems programmers that a DIMENSION statement should be an executable statement, which assigns storage dynamically on the basis of input or computed parameters; that storage assignment is too important to be done only once at the compilation of a code; and

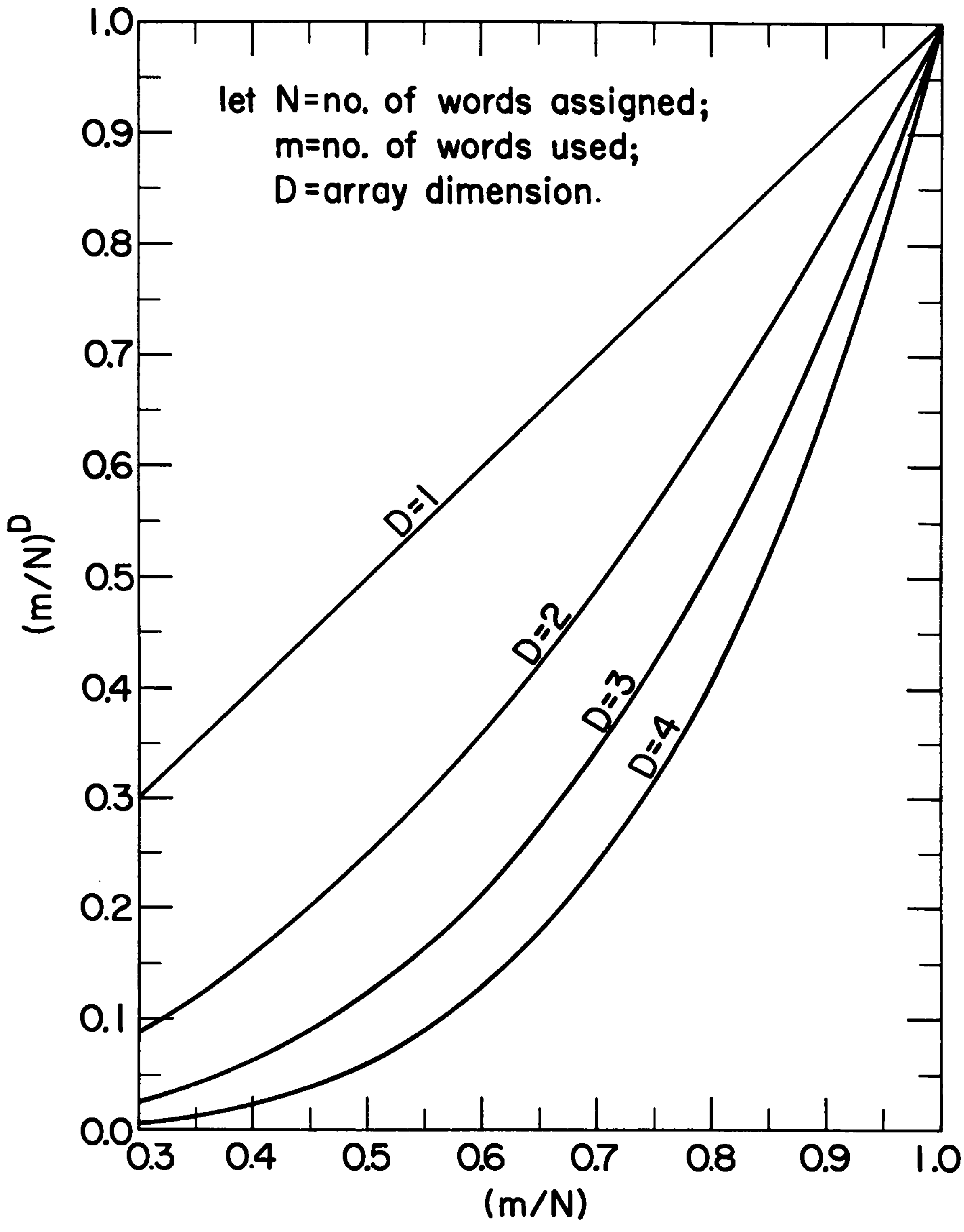

Figure 10. Waste of Storage with Fixed Dimensions

that storage assignment should be done for each problem run. The variable dimensioning allowed in Fortran-IV is only half of the solution, because the main program requires fixed dimensions. By using this capability, however, it is possible to partially program around this inadequacy by assigning a single data block for all data, calculating how much of it is actually needed for each array, and packing all arrays together without waste. This method of using a fixed dimension for total usage is superior to the method of using a fixed dimension for each array, but it still leaves unused storage.

4.3 Multi-Function Usage. Although some storage requirements remain constant throughout a code, others are often of a recurring nature, occurring and disappearing. This latter situation allows the use of EQUIVALENCE techniques, which are so commonly known that it is only necessary to mention them.

4.4 Telescopic Methods. A less obvious technique, but one which is equally valuable is the method of "telescoping" arrays. Any array which can be calculated with a first-order recurrence relation can be telescoped into a single memory cell. For example, assume that we wish to calculate $R = \sum_{i=1}^{I} a_i$, where $a_{i+1} = f(a_i)$; then it is unnecessary to allow more than a single cell for the a-array. The same method can be applied to higher order recurrence relations by assigning only the number of cells which equals the order of the recurrence relation rather than the length of the array.

The DDF two-dimensional transport code uses this method; its inner loop consists of four third-order recurrence relations. The storage required is telescoped from a single four-dimensional array to four three-dimensional arrays. Let the array dimensions be given by I, J, G, and M. The original requirement is I*J*G*M, and the telescoped requirement is 2*I*M*G + 2*J*M*G; this saves storage whenever $I*J/(I+J) > 2$, which is always true in a real problem environment. The magnitude of the saving is very impressive: For $I = J = 30$, $G = 25$, and $M = 16$, the telescoping technique would save 310,000 words of storage.

4.5 Mapping Functions. An auxiliary function which defines the region of applicability of another function is called a mapping function. In numerical formulations these occur as recursive subscripts; e.g., in the term

$$\sigma_{m,n_{i,j}},$$

the second subscript is defined by a mapping function, $n_{i,j}$. Mapping functions are valuable in computer programming because they can be used to conserve storage. The amount of storage required for the above function is (M*N + I*J), where the capitalized index indicates its maximum value. This function could be expressed without the use of the mapping function as

$$\sigma_{m,i,j},$$

but the storage required would then be three dimensional: M*I*J. In this example, the mapping function would conserve storage whenever $(M*N + I*J) < (M*I*J)$, that is, whenever

$$M > \frac{I*J}{I*J - N} .$$

This inequality points out two things about the use of mapping functions in programming: first, they should be used only when the number of elements in the mapping function exceeds the maximum value of the mapping function (i.e., where $I*J > N$); and second, they should be used only when the function to be mapped is multidimensional (i.e., where $M > 1$).

4.6 Sparsely Populated Arrays. An array which has a large population of zero elements can be more efficiently stored if only the nonzero elements are retained, and a binary control vector is used to control the use of the array. The control vector is a one-bit-per-element array which corresponds to each element of the initial array; zero bits correspond to zero words, and one bits correspond to nonzero words. If the initial array has W total words in it, of which only N are nonzero, and there are B bits per word, then storage can be saved whenever

$$N + \frac{W}{B} < W ,$$

or, put in terms of the usage density, N/W, whenever

$$\frac{N}{W} < \frac{B - 1}{B} .$$

For example, in a computer with 64 bits per word, the nonzero array plus the control vector will save storage whenever the usage density is less than 0.984.

In practice, the programming for these compacted arrays and their associated control vectors is considerably more complex than the programming for the initial array, and it is only when the usage density is very low that the extra effort is undertaken. This effort could be reduced if there were bit-manipulation facilities in Fortran, and these would be easier to implement if the hardware included bit-manipulation facilities, specifically including a left-zones count and an all-ones count.

The importance of this method of storage conservation increases as problem size increases: a usage density of 25% in a 100-word array offers a potential saving of less than 75 words, but in a 100,000-word array the potential saving is almost 74,000 words, depending on the word length of the computer.

5. A REVIEW OF SECONDARY STORAGE DEVICES

5.1 Scope. The secondary storage devices considered here will be limited to those which are used as extensions of main memory: bulk core, drums, disks, and magnetic strip devices. Tapes are excluded for two reasons: first, their use in nuclear codes has been covered in an excellent paper by Cadwell;[6] and second, the serial nature of the recording makes them inappropriate for random retrieval of information. Photographic storage devices, or "write-once" stores, are not considered because they are essentially limited to reading functions and cannot extend the read/write capability of main memory.

5.2 Continuity in Secondary Storage. Given that it is necessary to extend the storage capacity by using a hierarchy of storage devices rather than a homogeneous storage system, this hierarchy should offer extra capacity in uniform increments of access time. A storage system in which there are large gaps between the access times of the devices will impose an artificial barrier to program development. The gap between the access times of main memory and the fastest device in the secondary storage hierarchy can quite accurately be called the interface gap. The performance characteristics of the device which interfaces main memory are very important, because this device must buffer access to other members of the hierarchy. If the interface gap is small, then program development beyond main memory capacity is orderly; if the interface gap is large, then program development will be retarded. For example, if the interface device is magnetic tape, the interface gap is at least milliseconds and at most minutes; if the interface device is magnetic strips, the interface gap is several hundred milliseconds; if the interface device is magnetic disk, the interface gap is tens to hundreds of milliseconds; if the interface device is magnetic drum, the interface gap is a few to tens of milliseconds; and if the interface device is bulk storage, the interface gap is a few microseconds. It is useful to define the access ratio, i.e., the ratio of the access time of the secondary storage device to the access time of main memory. The access ratio for magnetic strip devices is on the order of 10^6; for disks it is on the order of 10^5; for drums it is on the order of 10^4;

and for bulk it is on the order of 10^1. These numbers vary, of course, from one computer to another, and are an important consideration in equipment selection.

Notice that there are no storage devices which have an access ratio of 10^2 or 10^3; that is, for a computer with a 1-μsec main memory, there is no secondary storage device available which has an access time of 100 μsec to 1 msec. There is a performance area which would have valuable applications if it could be made available at a cost between the cost of drums and the cost of bulk storage, say $0.50 per word.

5.3 Magnetic Strip Devices. The largest total capacity available in secondary storage devices is a category I have called magnetic strip. Although designs vary in their method of implementation, the recording medium is a strip of polyester with an iron oxide coating;[7] Figure 11 shows the organization of the IBM-2321. These strips are 2-1/4 by 13 by 0.005 in.; there are ten strips to a subcell, 20 subcells to a data cell, and 10 data cells to a data cell array. The total capacity is 400 million 8-bit bytes, or 3.2 billion bits. The strip transport mechanism is shown in Figure 12. The access times are comparatively long, several hundred milliseconds, and the transmission rates are relatively low. The main advantages of these devices are the large total capacity and the low cost per word of storage.

5.4 Magnetic Disks and Drums. The first disk unit was made by the Bureau of Standards in 1952;[8] it consisted of a stack of donut-shaped

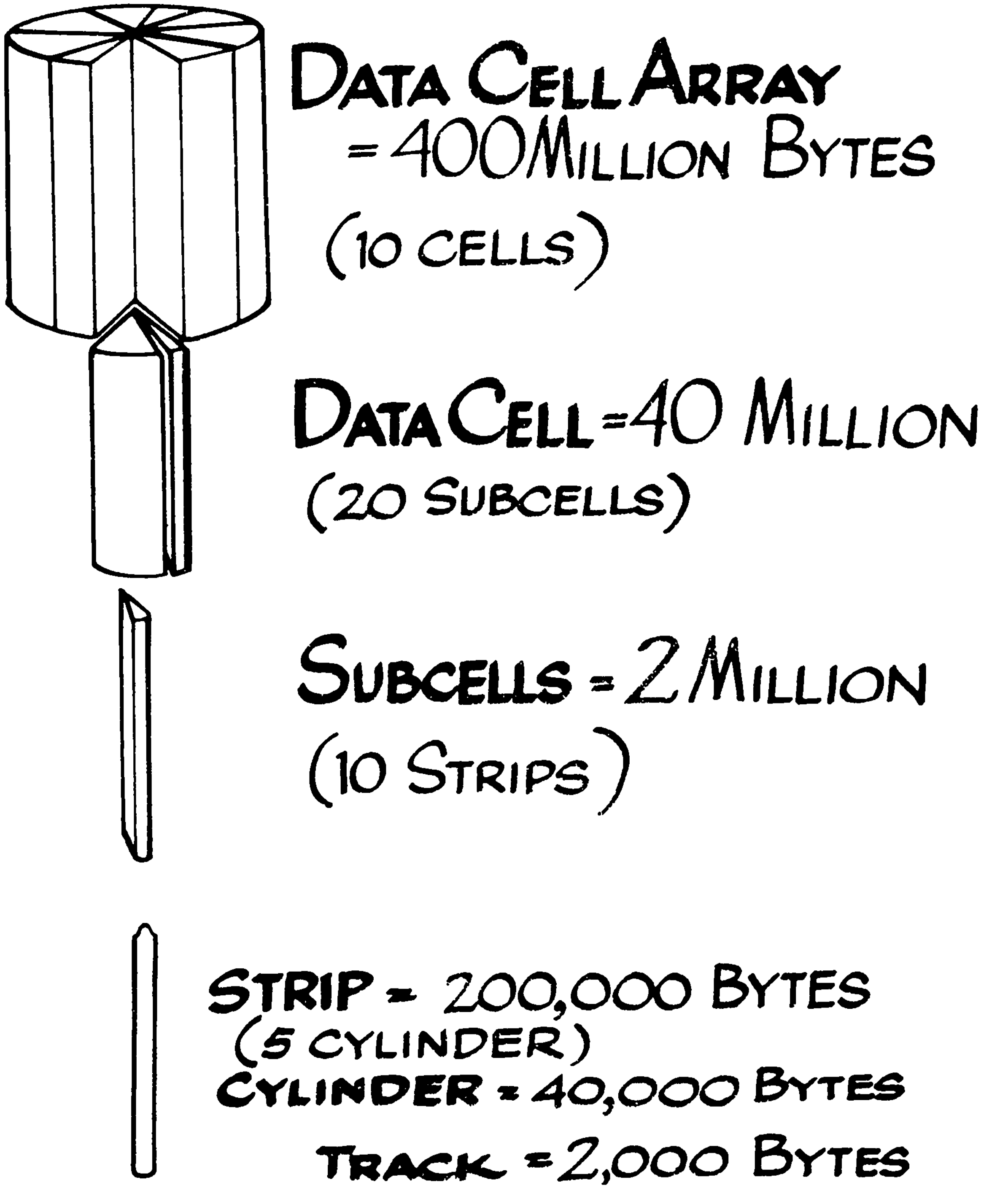

Figure 11. Organization of the IBM-2321

STRIP PICKUP CYCLE

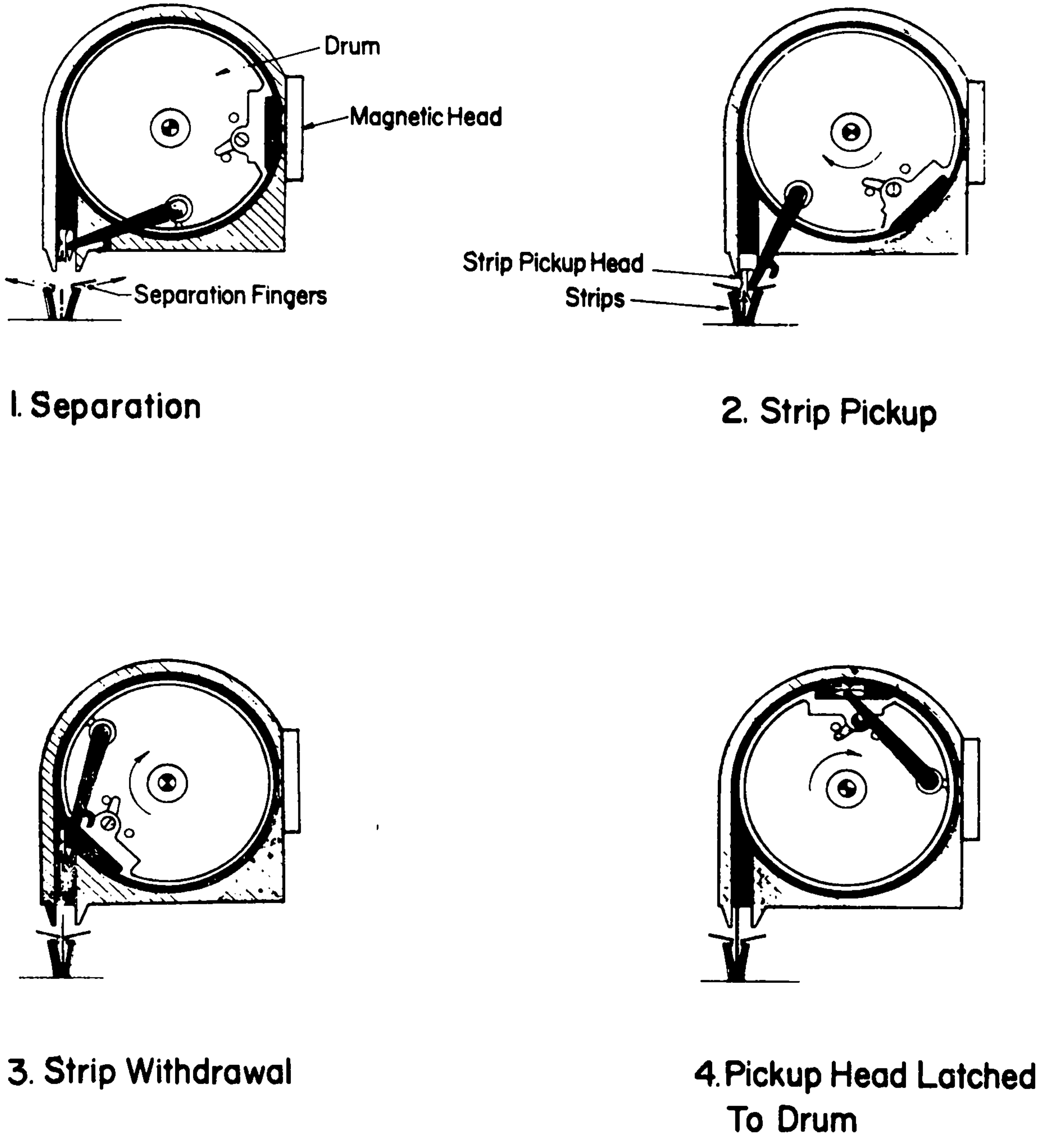

Figure 12. Access Mechanism of the IBM-2321

plates with an access mechanism in the center that moved axially to the correct disk, which was then accelerated and read. The first commercial disk was the RAMAC, introduced by IBM in 1956. It consisted of a stack of 50 plates accessed by a single access arm with two read-write heads, one for the upper and one for the lower surface. The access arm was positioned vertically to the correct plate, moved horizontally to the correct track, and the information read (written) when the rotation of the plate brought the correct position under the head as shown in Figure 13. Although this can be classified as a "three-motion" device, four motions were often necessary, since repositioning of the access arm required a horizontal withdrawal of the arm before the other motions could take place. This design was improved upon with the design of access mechanisms with an access arm per plate, which eliminated the withdrawal and the vertical motion. By placing more than one read-write head per arm, the amount of horizontal positioning motion can be reduced. The logical extension of this design is the "head-per-track" disk offered by Burroughs, in which there is no horizontal motion necessary. CDC has recently announced an "opposing-access" disk, which is much more stable than single access devices. Two stacks of plates are used rather than just one, and a central access arm positioner moves the opposing access arms at the same time.

Access techniques for drums have developed from the single read-write head which moved along the surface of the drum, to the multiple

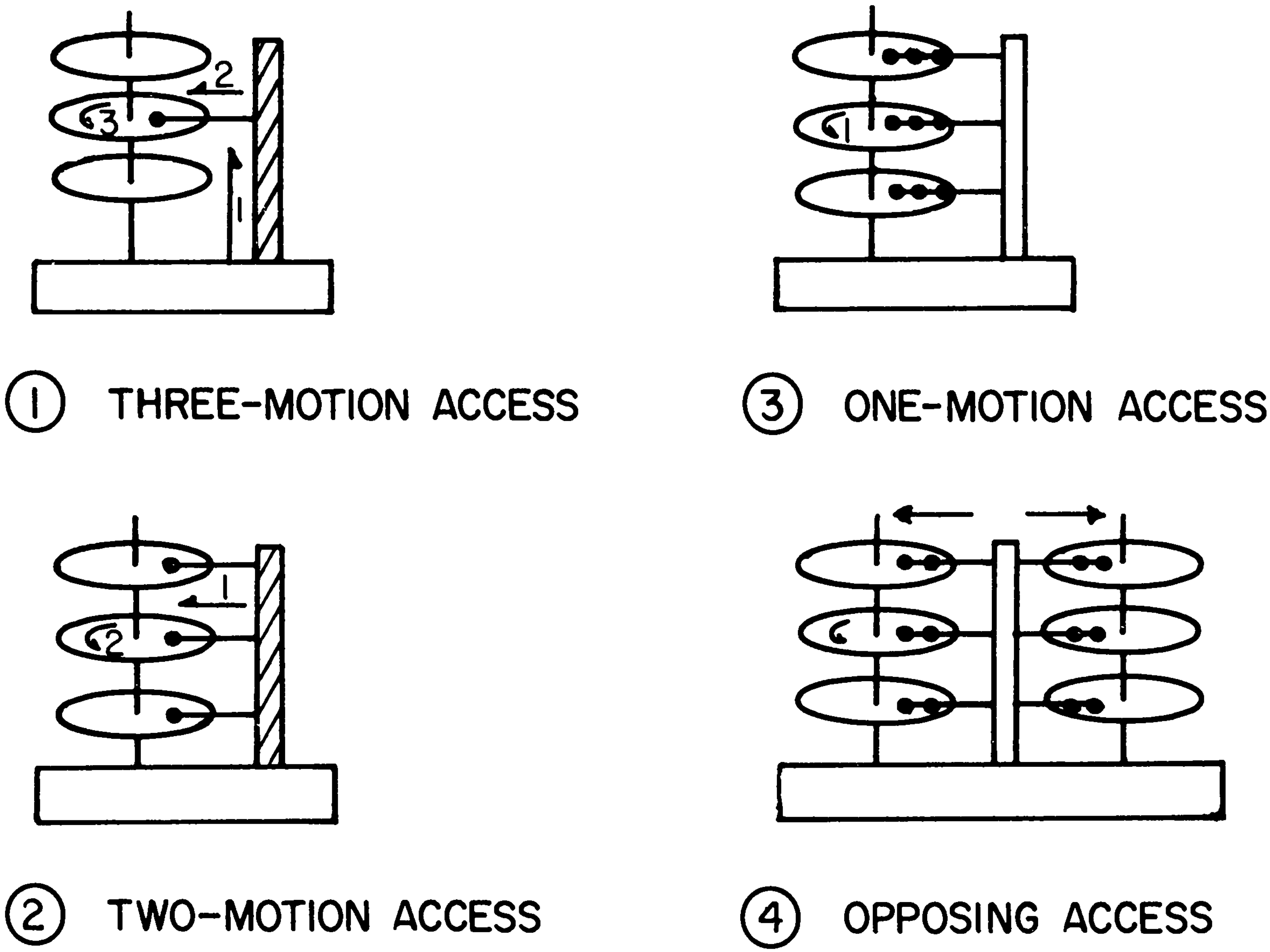

Figure 13. Disk Access Mechanisms

read-write head designs which use head-per-track reading and writing, but require a positioning motion for the bank of heads, to the complete head-per-track for all tracks. Most currently available drums are of the head-per-track design; Univac has a two-motion access drum, the Fastrand, which uses multiple read-write heads which are laterally positioned.

Improvements in transmission rate for disks and drums have come about through higher density recording and parallel transmission. The iron oxide coating of disks and drums was more efficient in terms of readback voltage up to about 500 bpi,[9] but has now been supplanted by the cobalt-nickel alloys at higher recording densities. The oxide coatings are typically much thicker than the metallic alloys, and bit resolution is difficult to maintain with increasing thickness due to flux-spreading and self-demagnitization. Parallel transmission from several heads simultaneously is used on most modern disks and drums to increase the transmission rate. The effect of parallel transmission is to multiply the transmission rate by the number of heads which are transmitting simultaneously; e.g., the Stretch disk was a parallel transmission version of the IBM-1301; it was 12 times as fast as the 1301 because it transmitted from all surfaces simultaneously.

5.5 Bulk Storage. The highest performance in secondary storage devices is obtained in bulk storage--as well as the highest cost! Bulk storage is essentially core memory with lower performance specifications and lower cost. For example, IBM has designed an 8-μsec bulk storage, the

2361, which sells for less than $2 per word, as compared to about $12.50 per word for their 0.75-μsec main memory. CDC has designed a 3.2-μsec bulk store for their 6600 and a 1.6-μsec bulk store for their 6800. These CDC designs have a very high transmission rate due to the fact that they fetch 8 and 16 words, respectively, in the memory cycle time. Because bulk storage is the closest in performance of any secondary storage device to main memory, it is very valuable in providing a buffer between the main memory and the slower, larger capacity, secondary storage devices.

5.6 Evaluation of Secondary Storage Devices. Figure 14 is a state-of-the-art diagram for disks, drums, and strip devices; it plots the transmission rate and the capacity of moving media storage devices attached to current computers. The symbols differentiate between the types of devices; the numbers are an identification key, the first digit identifying the manufacturer and the second the device in his line. In this diagram it is an advantage to achieve the upper-right-hand corner, indicating larger capacity and high transmission rate. If the highest points in the diagram were connected, they would indicate the frontier of transmission rate for moving-media storage devices; they cluster around 1 to 2 x 10^7 bits per second. Note that there is little to choose between disks and drums as far as transmission rate is concerned. On the far right of the diagram is the frontier of storage capacity, and the devices which are farthest to the right are the magnetic strip devices. Note, however, that higher capacities than about 10^9 bits can

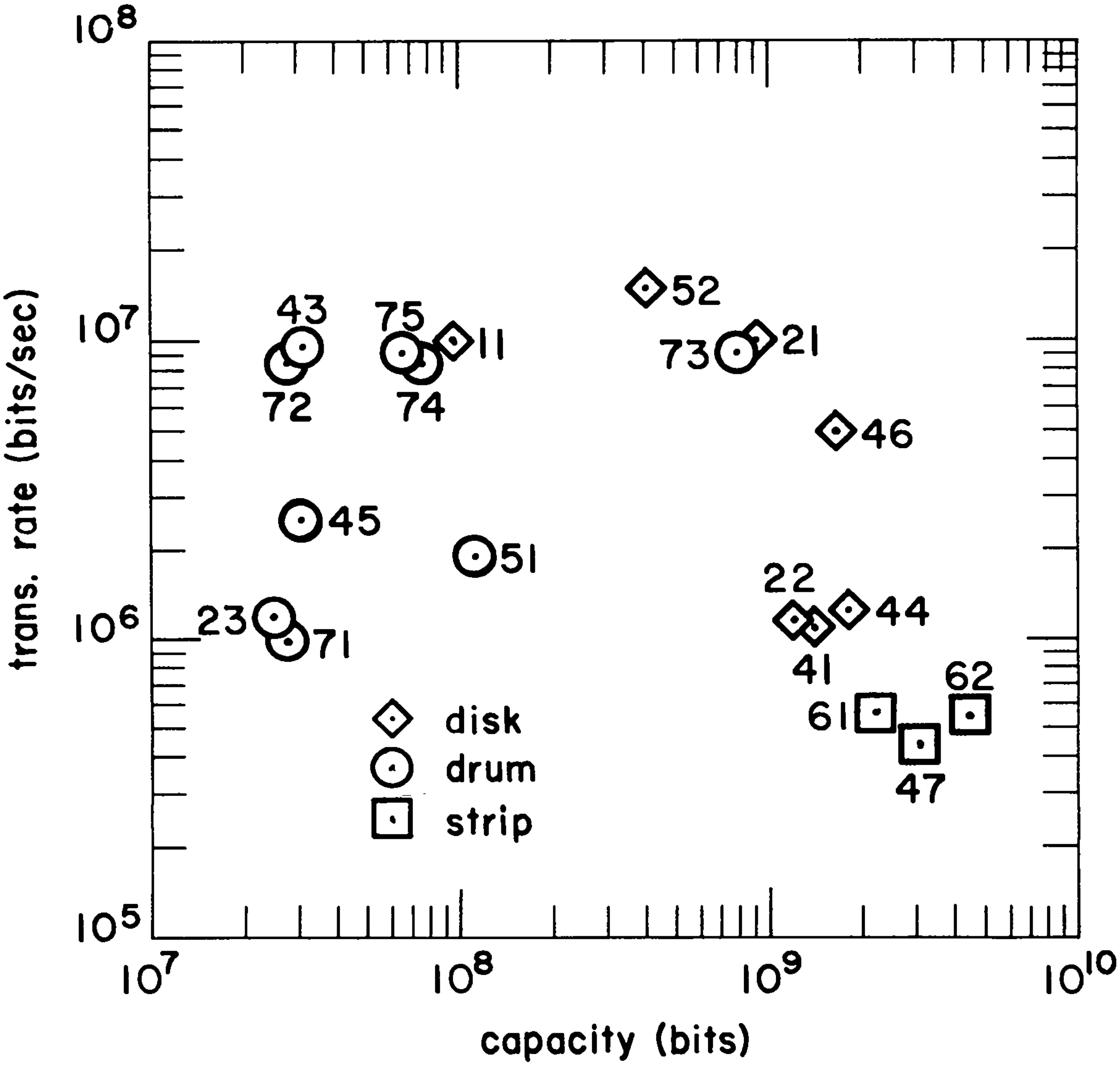

Figure 14. State-of-the-Art for Transmission Rate and Capacity

be achieved only by accepting lower transmission rates. Two of these points seem to be out of place: the disk marked "11" and the drum marked "73". What is common to the upper-left cluster is that they are all head-per-track devices, including the disk; the upper-right cluster is characterized by two-motion access, including the drum. The drums in the lower left are low-cost, low-performance devices.

For blocks of less than about 10^3 to 10^4 words, transmission rate is less important than access time, and Figure 15 is a state-of-the-art diagram for access time and capacity. Here the most advanced devices are in the lower-right-hand corner, indicating large capacity and low access time. This diagram separates the types of devices rather distinctly by their access motion, with the strip devices showing up with highest capacity, but longest access time, two-motion access disks and drums in an intermediate position, and one-motion access drums and one disk having lowest access and capacity. While advances in the state-of-the-art will move points in this diagram lower and to the right, the relationship between the classes will continue. Note also that it is difficult to say that any one of these classes is the "best" in any absolute sense, since they each have their advantages. The position of the two-motion devices can be lowered in the diagram by pre-positioning, and they then compete quite effectively with one-motion devices.

While the total capacity has its place in storage system design, another parameter which is often overlooked is equally important: the

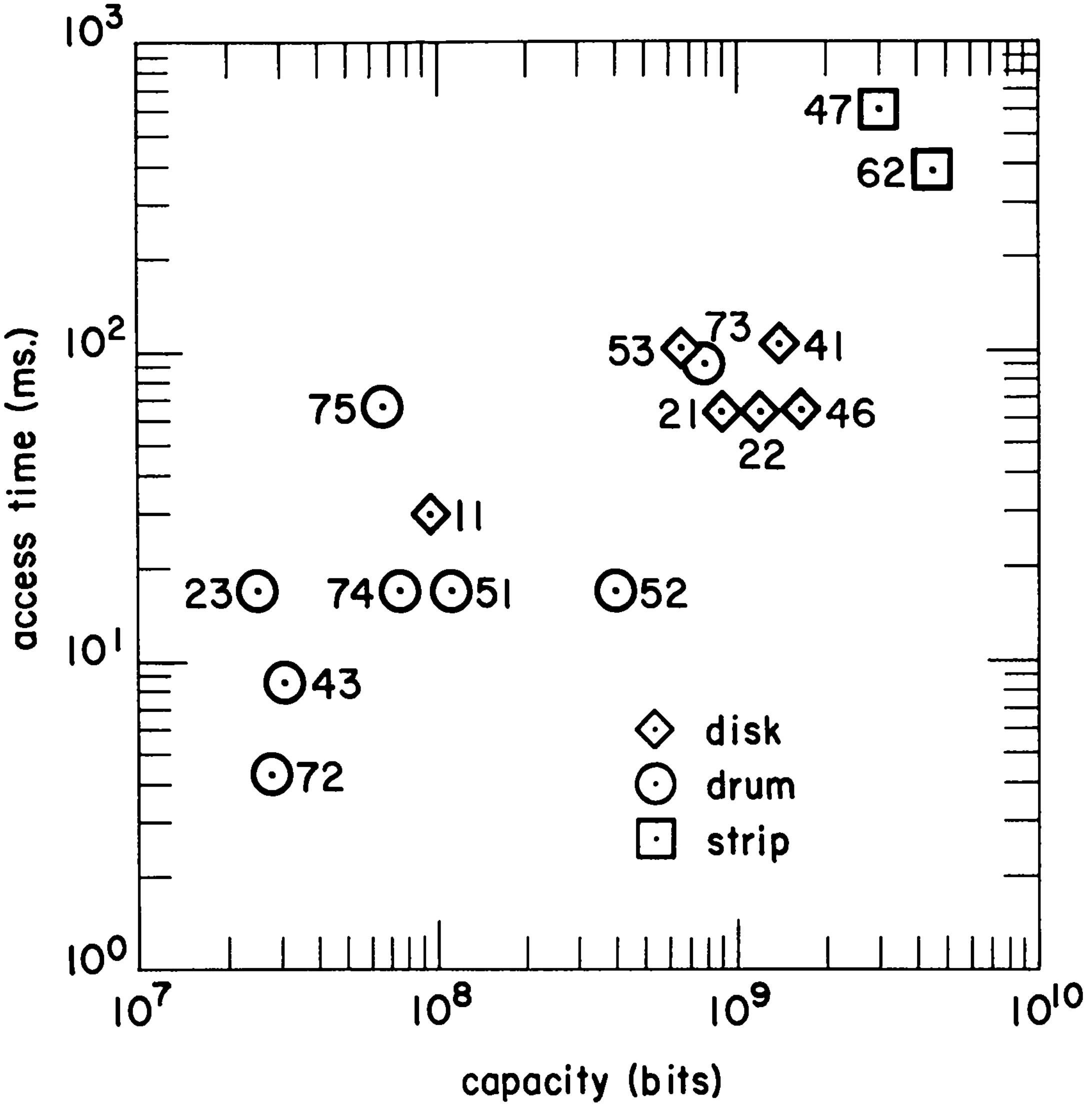

Figure 15. State-of-the-Art for Access Time and Capacity

addressable capacity without repositioning. Figure 16 shows this parameter plotted against access time, and here it is an advantage to achieve the lower-right part of the diagram. The identification key here is different than the previous diagrams. Again there are three clusters of points, and for this parameter there is a clear advantage to be seen: the head-per-track devices have a significant advantage over the two- and three-motion devices. Item "7" is the Burroughs head-per-track disk. The two disks which are clustered with the strip devices are removable-pack disks, and this cluster has a common feature, i.e., it is possible to physically remove the recording medium from the transport. This is a valuable feature, but it is paid for by a low capacity without repositioning. As a rule of thumb, there is approximately an order of magnitude between the removable-media devices, the two-motion disks and drums, the head-per-track drums, and the head-per-track disk. It would appear that the largest capacity without repositioning is achievable in a head-per-track disk.

Figure 17 shows the cost per bit of storage plotted versus the storage capacity, and here bulk storage is included. In this diagram, it is an advantage to achieve the upper-left corner, and this position is held by the magnetic strip devices. Two other strip devices are equal in cost to the two-motion access disks and drum; this is probably because they are low capacity devices. A rule of thumb for secondary storage costs could be drawn from this diagram: there is about an order of magnitude in cost between bulk, head-per-track drums, disks

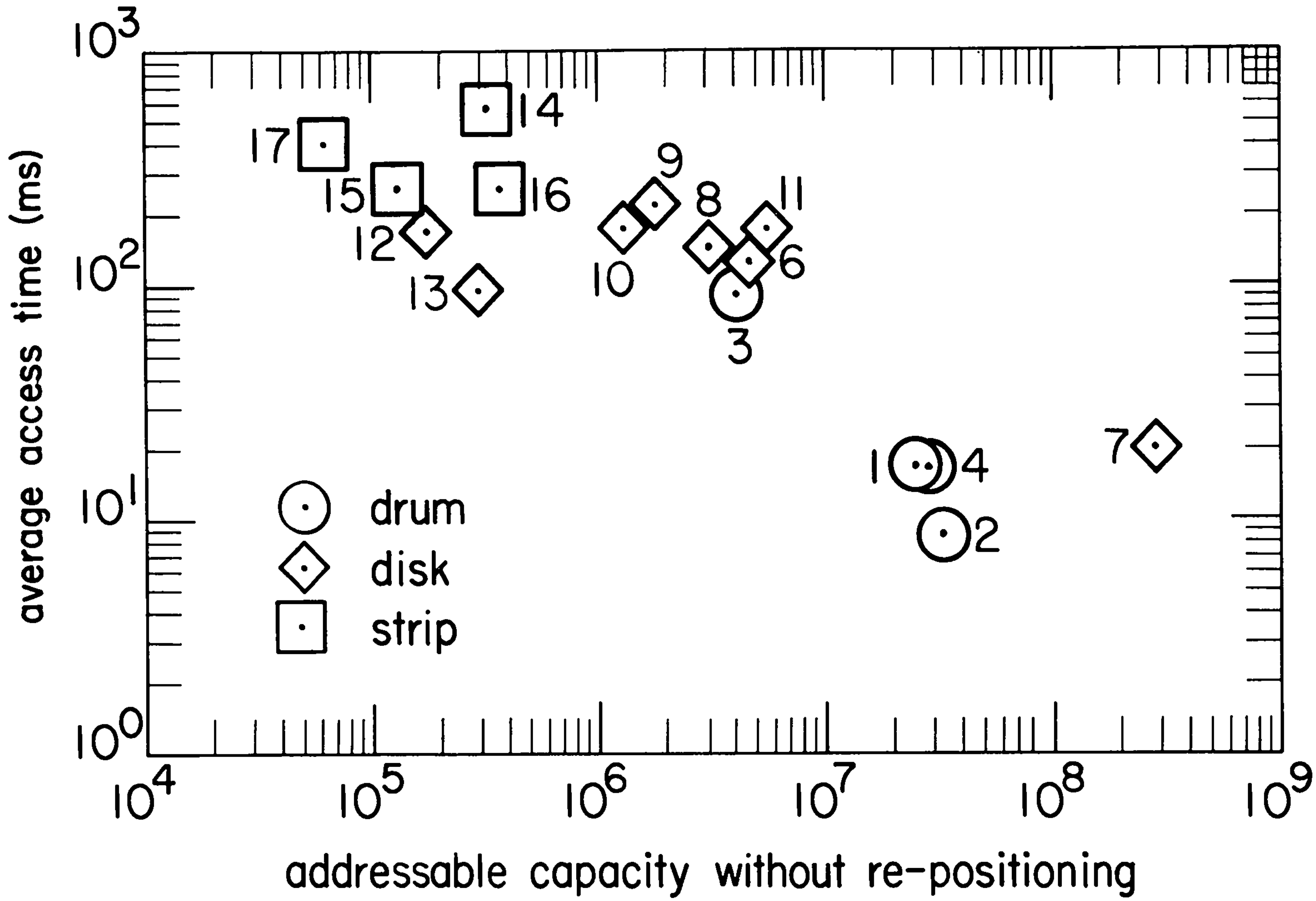

Figure 16. State-of-the-Art for Addressable Capacity without Repositioning

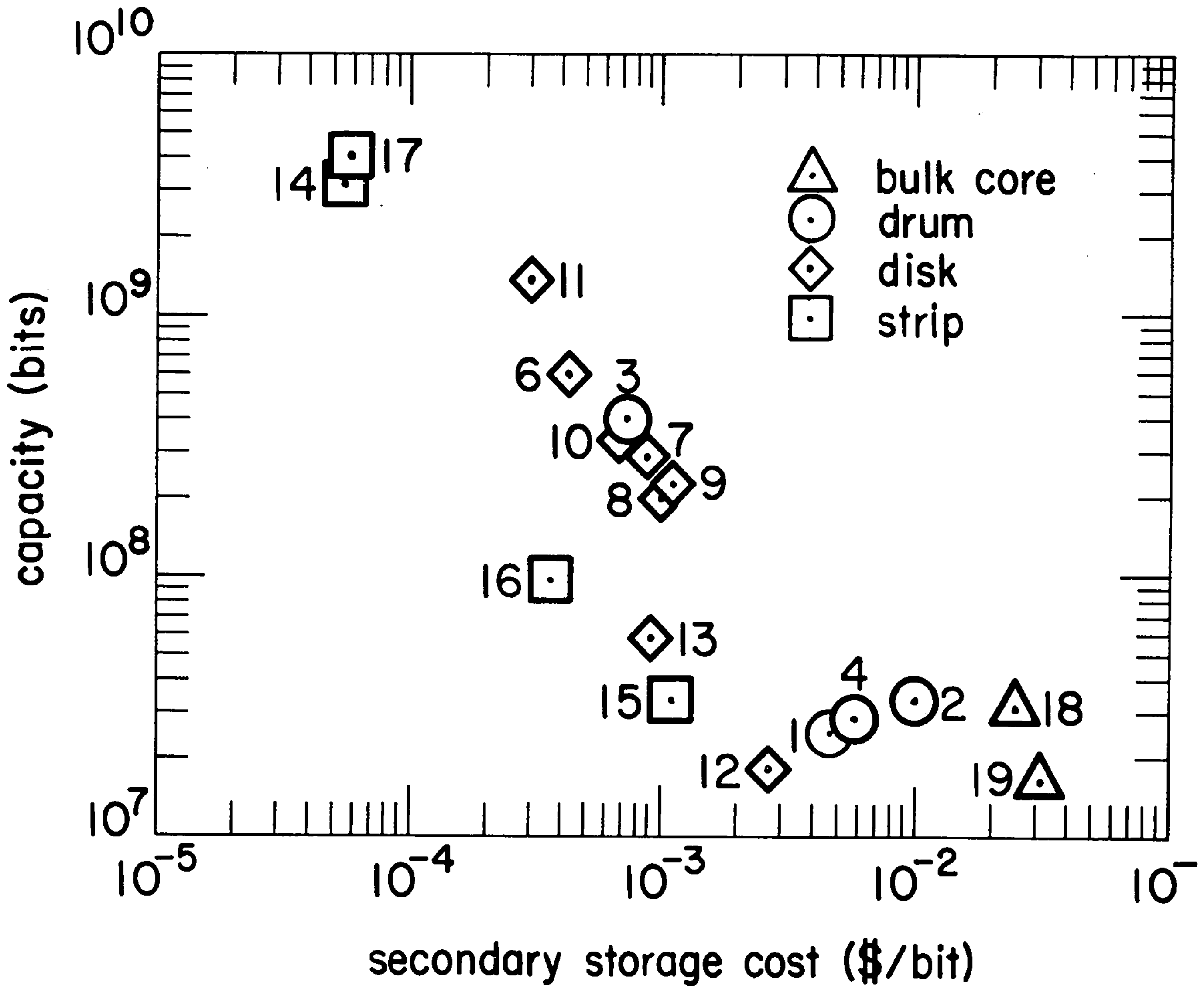

Figure 17. Secondary Storage Cost and Capacity

and the two-motion drum, and the magnetic strip devices. It is noteworthy that the head-per-track disk, item "7", is competitive in cost with other disks.

Figure 18 is a comparable diagram showing cost versus average access time. The first thing to note about this diagram is that item "1" is plotted wrong: it belongs right on top of item "4". In this diagram it is an advantage to achieve the lower-left part of the diagram, and here lower costs mean higher access time, so that there is an inverse relationship between cost and access time. Note again, however, that the head-per-track disk, item "7", has the low cost of a disk and the low access time of a drum.

5.7 Summary of Advantages and Disadvantages. A summary of the parameters which describe secondary storage devices shows advantages and disadvantages in each category. The highest capacity is found in magnetic strip devices, with disks in an intermediate position, and drums and bulk having the lowest capacity. These positions are exactly reversed, however, for addressable capacity without repositioning: drums and bulk have the highest capacity, disks are again intermediate, and strip devices are the lowest. Access time follows very closely with total capacity, with the highest-to-lowest order being strip, disk, drum, and bulk. Transmission rate has an inverse relationship to access time, with the highest-to-lowest order being bulk, drum, disk, and strip. Cost is also inversely related to access time, the highest-to-lowest order being the same as for transmission rate: bulk, drum,

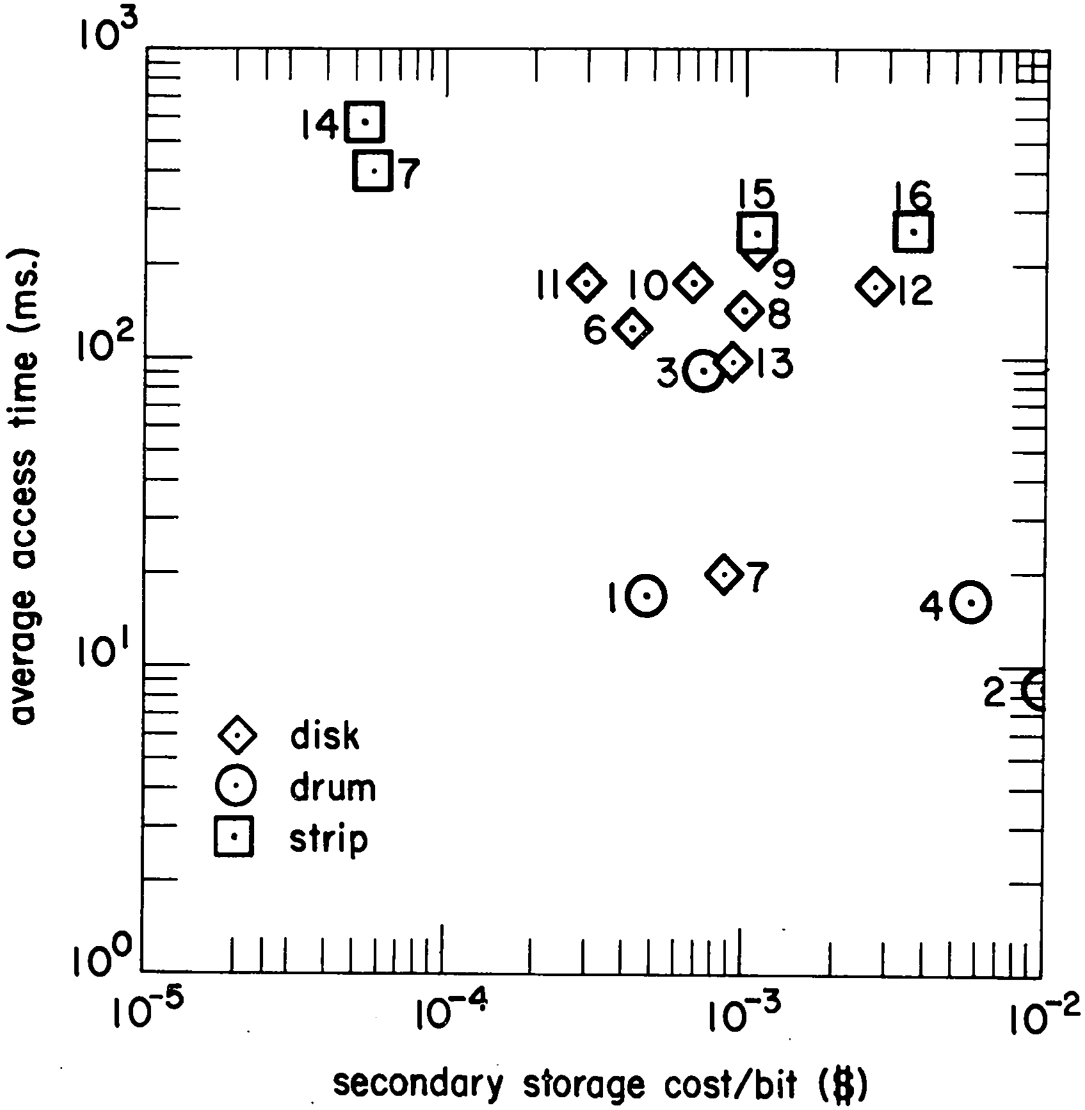

Figure 18. Secondary Storage Cost and Access Time

disk, and strip. Note that there is no single device which is better in all categories than other devices; this makes it impossible to select only one of these devices to satisfy all requirements, and a hierarchy is in general necessary. The single exception to this is the head-per-track disk, which has the technical capabilities of a drum, and the economic advantages of a disk: this may well lead to the replacement of drums with head-per-track disks.

6. SOFTWARE CONSIDERATIONS

6.1 Multiprogramming. A multiprogramming operating system is one which controls the concurrent operation of several user programs in order to overlap their input-output activities with computation. This has a severe impact on the storage system because storage must be provided for several programs rather than just one. Techniques vary for accomplishing this: some operating systems assume that there is sufficient space in main memory for several programs, and control is alternated only between main-memory contained programs. The flaw here is that a single program may occupy all of main memory, and multiprogramming then ceases. A more general approach is to provide for moving I/O-limited jobs out of main memory until their I/O activity is complete; this makes room in main memory for computation-limited programs which can be multiprogrammed. The problem with this method is that it creates an artificial flow in information into and out of main memory which may conflict with program requirements. Bulk storage solves many problems for multiprogramming, since the overhead for moving codes between main memory and bulk is very low compared to moving-media storage devices. A Fortran language addition which would help decrease the "roll-in" and "roll-out" traffic would be the option to specify a read-write operation and proceed with the calculation. The distinction between the use of secondary storage as an extension of main memory and its use for input-output needs to be made clear to systems programmers. It is often unnecessary to switch control to another program just because secondary storage is called for: the same calculation can often proceed in parallel with the information transfer, and this eliminates the need for much roll-in/roll-out traffic.

6.2 Time-Sharing. An even more serious impact on the storage system is made by time-sharing operating systems, since the number of active programs may be on the order of 20 or 30 rather than just 2 or 3 as in multiprogramming. Here movement of programs between main memory and secondary storage is essential, and the specifications of the secondary storage system determine the overall efficiency of the time-sharing system. For example, if it takes 100 msec to swap programs in main memory and the time slice allotted per program is 100 msec, then the system efficiency cannot exceed 50%. There are time-sharing systems in operation in which the efficiency is only about 30%, and a recent time-sharing demonstration attended by a Los Alamos staff member showed an efficiency of less than 20%! The use of bulk storage rather than a moving-media storage device to interface the main memory will probably turn out to be the salvation of time-sharing, since not many scientific computing installations can tolerate these low efficiency levels.

6.3 Fortran Improvements for Handling Secondary Storage. In the course of this report several suggestions have been made for improvements in the Fortran language to improve the ease and efficiency of storage administration, and it may be well to reiterate them.

6.3.1 Variable-dimensioning. Fortran should include true variable-dimensioning, in which storage is assigned on the basis of input or computed parameters.

6.3.2 Bit-manipulation. Fortran should include the ability to define arrays whose elements are defined by a single bit, to allow the efficient handling of sparsely populated arrays by control-vectors.

6.3.3 Parallel read-write. Fortran should include the ability to specify that the calculation can continue in parallel with an input-output operation, to minimize roll-in/roll-out in multiprogramming systems, and to increase efficiency in nonmultiprogramming systems.

6.3.4 One-level storage systems. Fortran should include the ability to cooperate with a one-level storage system by specifying which pages are active and which are inactive.

6.3.5 Array definition by control word. Although a true one-level storage system requires a hardware implementation, some of the advantages can be achieved by defining arrays by control words, such that the user need not give an array description for every access to secondary storage, and need not specify which level of secondary storage is to be used. The user should specify "recall" or "remove" for an array and let the operating system decide where in the storage hierarchy there existed space <u>at the fastest level</u>. By using a table of array descriptions, this decision would be trivial, and would make optimum use of the secondary storage hierarchy. Only the operating system knows the use-status of the secondary storage hierarchy, and it, not the user, must make the decision as to how to remove or recall a data block most efficiently. "Reading" and "writing" in specific devices should be eliminated: the function, not the device, should be specified.

7. CONCLUSIONS

Predicting future trends in computing systems is a hazardous business because of the rapidity with which changes in design and technology occur. Several practitioners of the art have given estimates of the trends in storage systems[1,3,10,11,12]. Although all-electronic storage systems will continue to increase in importance, moving-media storage devices offer economic and capacity advantages which will guarantee them an important role in future computers. The use of bulk storage to interface the main memory will become common; and with the close integration of bulk stores, it is possible that smaller, very fast, main memories will be used: this design can be thought of as a multilevel main memory. The use of many levels of capability and capacity, with smaller gaps in the hierarchy, is indicated for the forseeable future.

REFERENCES

1. Rajchman, J. A., "Magnetic Memories--Capabilities and Limitations," J. Appl. Phys. Vol. 34, No. 4 (part 2), pp. 1013-1018, April 1963.

2. Worlton, W. J., and Voorhees, E. A., "Recent Developments in Computers and Their Implications for Reactor Calculations," Proc. of the Conf. on the Application of Computer Method to Reactor Problems, ANL-7050, pp. 15-32, May 17-19, 1965.

3. Louis, H. P., and Shevel, W. L., Jr., "Storage Systems--Present Status and Anticipated Development," IEEE Trans. on Mag., Vol. 1, No. 3, Sept. 1965.

4. Opler, A., "Dynamic Flow of Programs and Data Through Hierarchical Storage," Proc. IFIP Congress 1965, pp. 273-276, Spartan Books, 1965.

5. Kilburn, T., et al., "One Level Storage System," IRE Trans. on Electr. Comp., Vol. EC-11, No. 2, pp. 223-235, April 1962.

6. Cadwell, W. R., "The Input-Output Problem in Two-Dimensional Neutron-Diffusion Programs with Large Meshes," ANS Trans., Vol. 7, No. 1, Philadelphia, June 1964.

7. Shugart, A. F., and Tong, Y., "IBM 2321 Data Cell Drive," Proc. Spring Joint Computer Conference 1966, pp. 335-345, Spartan Books, 1966.

8. Hobbs, L. C., "Review and Survey of Mass Memories," Proc. Fall Joint Computer Conference 1963, pp. 295-310, Spartan Books, 1963.

9. Jacoby, M., "A Critical Study of Mass Storage Devices and Techniques with Emphasis on Design Criteria," IRE-PG MIL, National Winter Convention on Military Electronics, 1962, pp. 165-179.

10. Rajchman, J. A., "Computer Memories--Possible Future Developments," RCA Review, pp. 137-151, June 1962.

11. Hoaglund, A. S., "Storing Computer Data," International Science and Engineering, pp. 52-58, January 1965.

12. Rajchman, J. A., "Memories in Present and Future Generations of Computers," IEEE Spectrum, pp. 90-95, November 1965.