Master File
24 pp.

INFOPLEX -- Hierarchical Decomposition of a Large Information Management System Using a Microprocessor Complex

Stuart E. Madnick

REPORT CISR-7
SLOAN WP-770-75
March 3, 1975

2/25/75

INFOPLEX -- Hierarchical Decomposition of a Large Information Management System Using a Microprocessor Complex

Professor Stuart E. Madnick
Center for Information Systems Research
Alfred P. Sloan School of Management
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

ABSTRACT

By using the concept of hierarchical decomposition, both of the logical functions and physical storage management, it is possible to develop a highly parallel information management system architecture. Such a system design, based upon a complex of microprocessors, is under study at M.I.T. and has been named INFOPLEX. Besides providing very high performance, the INFOPLEX structure is shown to provide a basis for exceptionally high reliability and availability.

INFOPLEX -- Hierarchical Decomposition of a Large Information Management System Using a Microprocessor Complex

Professor Stuart E. Madnick
Center for Information Systems Research
Alfred P. Sloan School of Management
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

INTRODUCTION

The effective use of computer systems for large-scale information management rather than numerical computation is still a largely unsolved problem. In this paper important concepts and theories regarding computer architectures for information management are introduced. The underlying concept, hierarchical decomposition, is presented in the next section.

Several ongoing developments in computer technology have made a radical change in system architecture both necessary and feasible[21]. New memory technologies, some recently announced and others under development for the near future, make very large capacity memories possible. But, the physical organization of such memories and their logical information handling functionalities are yet to be determined. As an example, consider the objective of designing an information system with total storage capacity in excess of 10^{15} bits processing up to 10^{6} logical interactions (e.g., queries, updates) per second and capable of physical input/output rates of at least 10^{9} to 10^{10} bits per second. No present computer architecture or theoretical structure has explicitly addressed the design of such a system.

In order to attain very high performance and functionality, as indicated above, it is necessary to take advantage of extensive parellelism in the system. With the advent of microprocessor technology such a strategy is quite feasible. Whereas highly parallel computer systems of the past, such as ILLIAC IV and CDC STAR-100, were designed to solve numerical problems, totally new approaches are needed for information management problems.

By using hierarchical decomposition, both functional and physical, a highly parallel information management system architecture can be implemented by means of a processor complex. Such a system, called the INFOPLEX, is presently under study at the Center for Information Systems Research in the MIT Sloan School of Management.

HIERARCHICAL DECOMPOSITION

There are two major types of parallelism that can be exploited in an information system: functional and physical.

Function Decomposition

In almost all cases, the interactions with an information system are in terms of very high level concepts whether originating from a human at a terminal or another computer system. These requests must be converted into the more basic operations appropriate to the particulars of the physical hardware and information structures. There are many ways that this conversion can be accomplished but in our research[7] we have found the technique of hierarchical function decomposition to be very effective for advanced information management systems (similar techniques have been used successfully in operating systems[13] and basic file systems[11]).

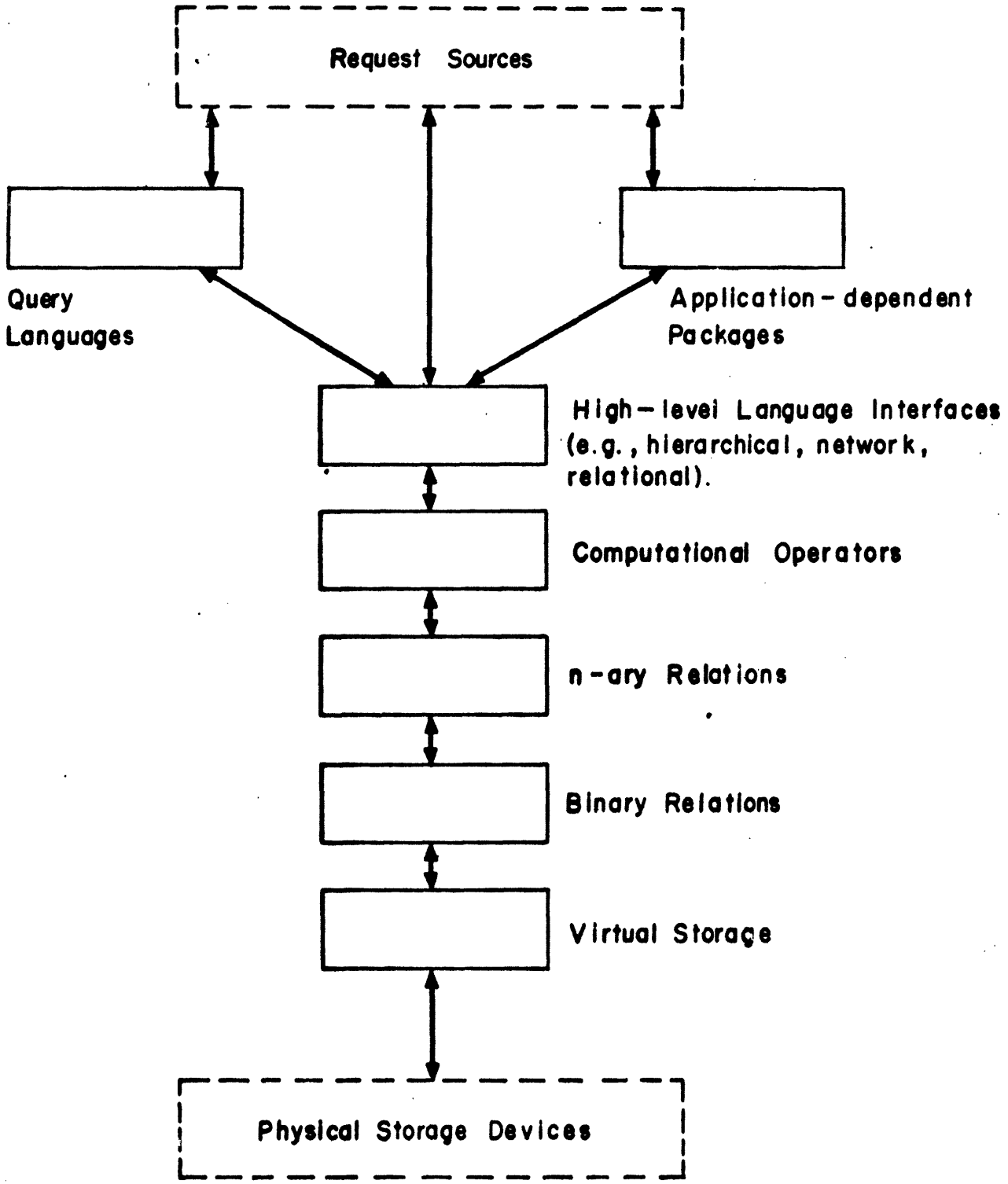


Figure 1 Hierarchical Function Decomposition

Figure 1 outlines the hierarchical function decomposition used in our Generalized Management Information System (GMIS) effort. This hierarchical concept has been used in the implementation of the New England Energy Management Information System (NEEMIS)[7]. Note that each functional level of the hierarchy is implemented in terms of the functions provided by the next lower level. This strictly hierarchical approach has resulted in an extremely powerful, flexible, and modular information management system -- a primary requirement for the NEEMIS application. In addition, the conciseness of the structuring greatly reduces the complexity of the system making optimization and debugging much more effective.

Physical Decomposition

To date, the technologies that lend themselves to low cost per byte storage devices (and, thereby, economical large capacity storage) result in relatively slow access times. If it were possible to produce ultra-fast limitless-capacity storage devices for miniscule cost, there would be little need for a physical decomposition of the storage. Lacking such a wonderous device, the requirements of high-performance yet low-cost are best satisfied by a mixture of technologies combining expensive high-performance devices with inexpensive lower-performance devices.

Figure 2 indicates the range of performance and cost for typical current-day storage technologies divided into 6 cost-performance levels. New storage technologies will undoubtedly make improvements at all levels, possibly

Storage Level	Random Access Time	Transfer Rate (bytes/second)	Cost per Byte	Technology
1. Cache	50 ns	100M	100¢	Semiconductor RAM
2. Main	1 μs	16M	10¢	Semiconductor RAM, Ferrite core
3. Block	50 μs	8M	2¢	Semiconductor shift registers, Bulk ferrite core, Charged-coupled devices, magnetic bubbles
4. Backing	1 ms	2M	.5¢	Fixed-head disks and drums, charge-coupled devices, magnetic bubbles
5. Secondary	50 ms	1M	.01¢	Moving-head disks
6. Mass	1 sec	1M	.0005¢	Automated tape-handlers, Laser devices

Figure 2

Spectrum of Storage Device Technologies

even collapsing some. In any case, it does appear that multiple levels of cost-performance storage devices will continue to exist for many years to come[15,22]. Note in particular that the current spectrum of devices represented in Figure 2 span over 6 orders of magnitude in both cost and performance.

If all references to information in the system were random and unpredictable, there would be little utility for the intermediate levels of storage technologies. Most practical applications result in clustered references such that during any interval of time only a subset of the information is actually used, especially when you consider the use of indexes and other control information. This phenomenon is known as locality of reference[6]. Under such circumstances, the intermediate levels of storage technologies can be used as paging devices or staging devices that hold these information clusters. This approach has been used in contemporary systems on the microscopic level[5] (e.g., IBM System/370 Model 158 and 168 cache systems), intermediate level[2,3,8,9,16,17,18] (e.g., Honeywell 68/80 Multics paging system), and macroscopic level[4,10] (e.g., IBM 3850 Mass Store System).

There are many ways that such an ensemble of storage devices may be structured but in our research[12,14] we have found the technique of hierarchical physical decomposition to be very effective. A detailed explanation of this approach is presented in reference 14. Briefly, information is moved between storage levels automatically depending upon actual or anticipated usage such that the information most likely to be referenced in the future is kept at the highest (fastest access) levels. Figure 3 depicts the general structure of a hierarchical storage system.

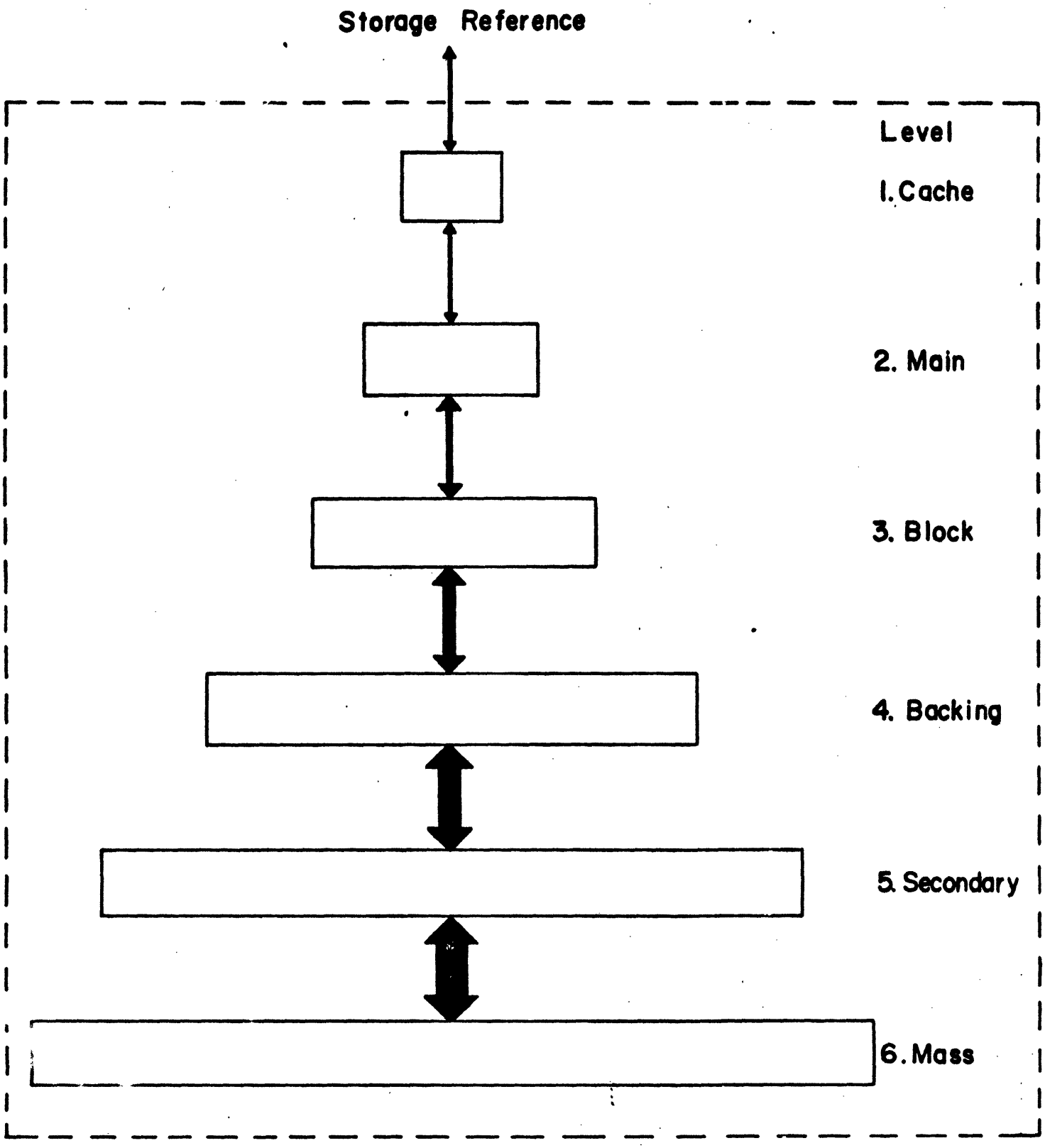


Figure 3. Hierarchical Physical Decomposition

PARALLELISM IN HIERARCHICAL STRUCTURE

The original research in hierarchical function decomposition was motivated by the desire for more structured implementation; the research in hierarchical physical decomposition was motivated by the desire for improved system performance. In addition to these original benefits, the hierarchical structure also lends itself to considerable parallelism.

Asynchronous Function Decomposition

As noted earlier, each level of function decomposition is implemented in terms of the primatives of the next lower level (refer to Figure 1). Furthermore, usually several lower level primatives must be used to implement each higher level primative. By using separate sets of processors for each functional level, it is possible to take advantage of parallel execution of lower level primatives, specialize processor functionality, simplify implementation, and enhance modularity. Thus, a request for a lower level function is accomplished by an inter-processor signal to one of the processors that implement that level. The hierarchical structure of the function decomposition makes such inter-processor communication relatively simple and efficient.

By incorporating queueing facilities and internal multiprogramming within each of the processors, a high performance "pipeline" can be attained. This makes it possible to maintain high rates of throughput. Futhermore, the simplicity of the structure makes relatively unlimited modularity possible thereby making it possible to assemble systems of enormous performance capacity.

The multiple processor implementation of the hierarchical function decomposition is depicted in Figure 4. Although such extensive use of processors has been quite expensive in the past, the advent of low-cost microprocessors makes such a system economically feasible. Furthermore, since each level implements only a limited amount of the total system's functionality, very simple processors can be used.

By taking advantage of the particular structure of the system and the somewhat specialized functions that are performed, highly reliable operation can be attained using techniques such as those used in the PLURIBUS system[19]. One of the key properties of the hierarchical function decomposition implementation is that all processors are anonymous and act as interchangeable resources (within a function level). Thus, if a processor malfunctions or must be removed from service, the system can continue to function without interruption. After a reasonable amount of time has elapsed, the higher level processors that had generated requests that were being performed by the defective processor merely need to reissue the same requests. Alternatively, the reissuing of requests could be accomplished automatically by the inter-level request queue mechanism.

Although the details are not elaborated in this paper, it should be clear that extensive parallelism, throughput, and reliability can be attained by means of a multiple processor implementation of the hierarchical function decomposition.

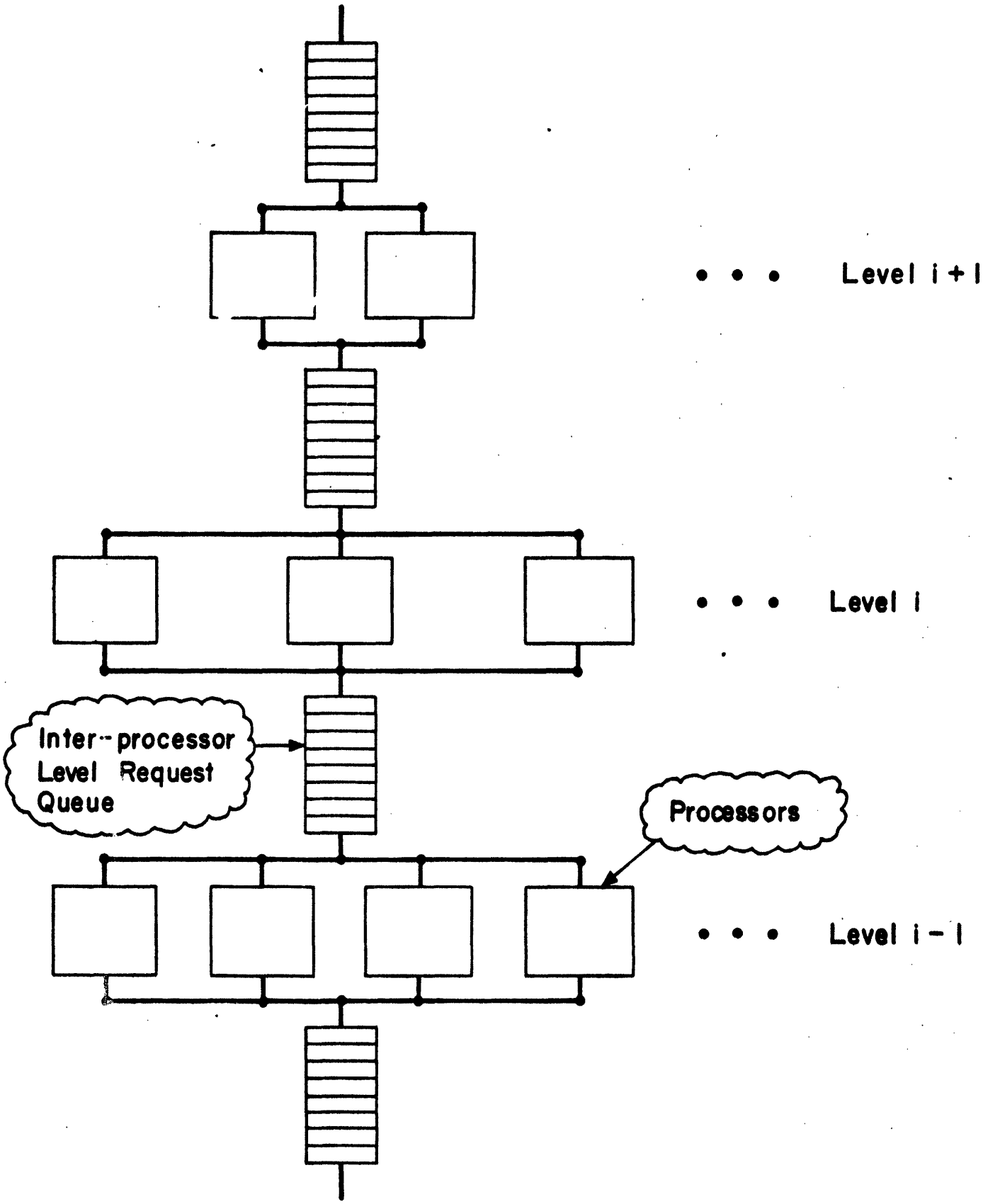


Figure 4. Hierarchical Function Decomposition Using a Microprocessor Complex

Asynchronous Physical Decomposition

As this author has noted in reference 14, it is possible to generalize the current-day specialized hierarchical storage systems (e.g., cache systems, paging systems, file archiving systems). In such a generalized system, the storage system is physically decomposed into a hierarchy of storage levels operating under local controls (i.e., decentralized control) with limited coordination necessary between levels.

Various physical storage management and movement techniques, such as page splitting, read through, and store behind, can be distributed into the hierarchy of levels. This facilitates parallel and asychronous operation in the hierarchy. Furthermore, these approaches can lead to greatly increased reliability of operation. For example, under the read through strategy, when data currently stored at level i (and all lower performance levels k > i) is referenced, it is automatically and simulataneously copied and stored into all storage levels j < i (i.e., all higher performance levels). The data itself is moved between levels in standard transfer units, also called pages, whose size N(i-1,i), depends upon the storage level from which it is being moved. (See Figure 5 for an illustration of this process.) Since all upper storage levels receive this information simultaneously, if a storage level must be removed from the system, there are no changes needed. In this case, the information is "read through" the level as if it didn't exist.

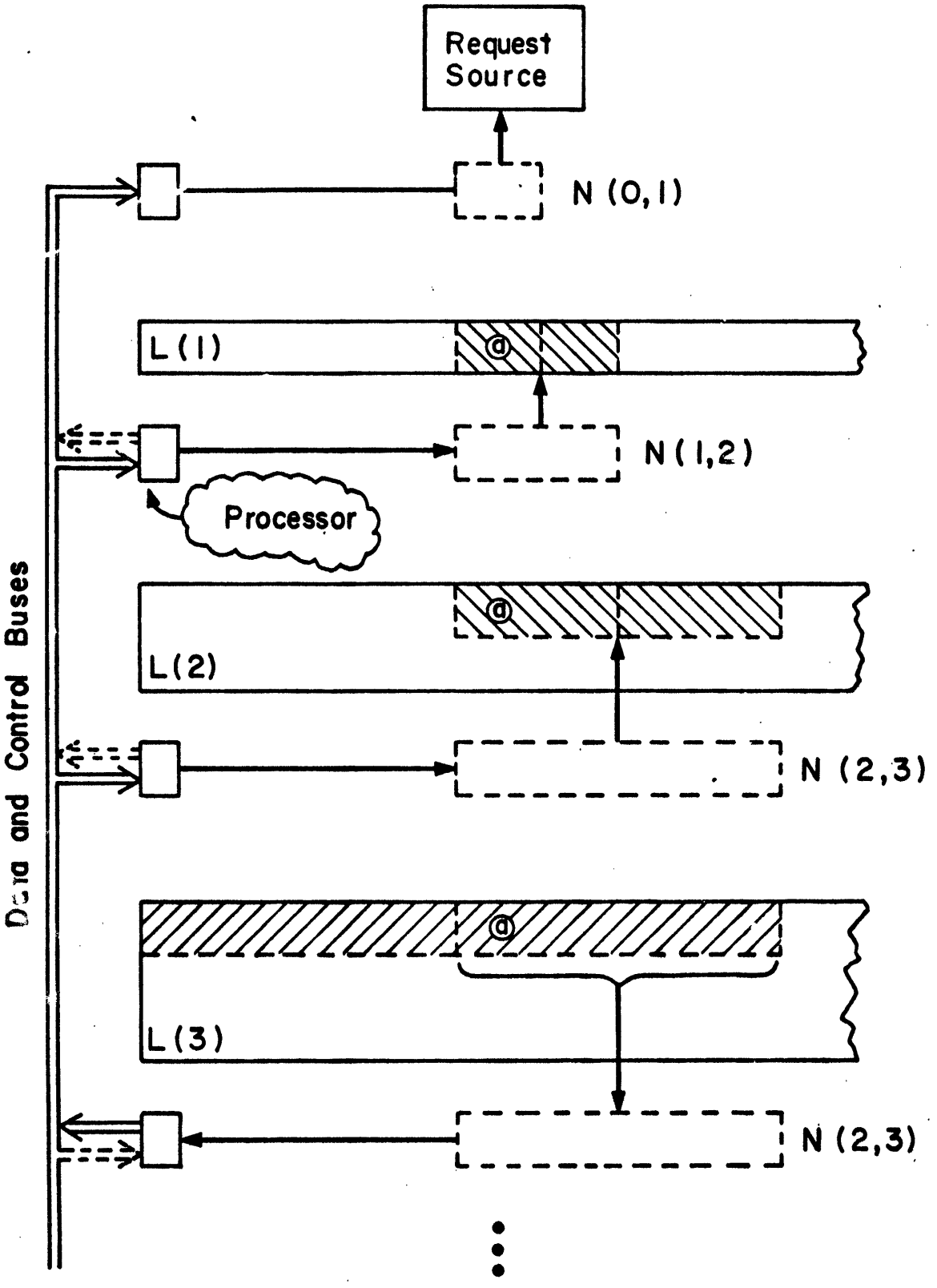


Figure 5 Example of Read Through with Page Splitting

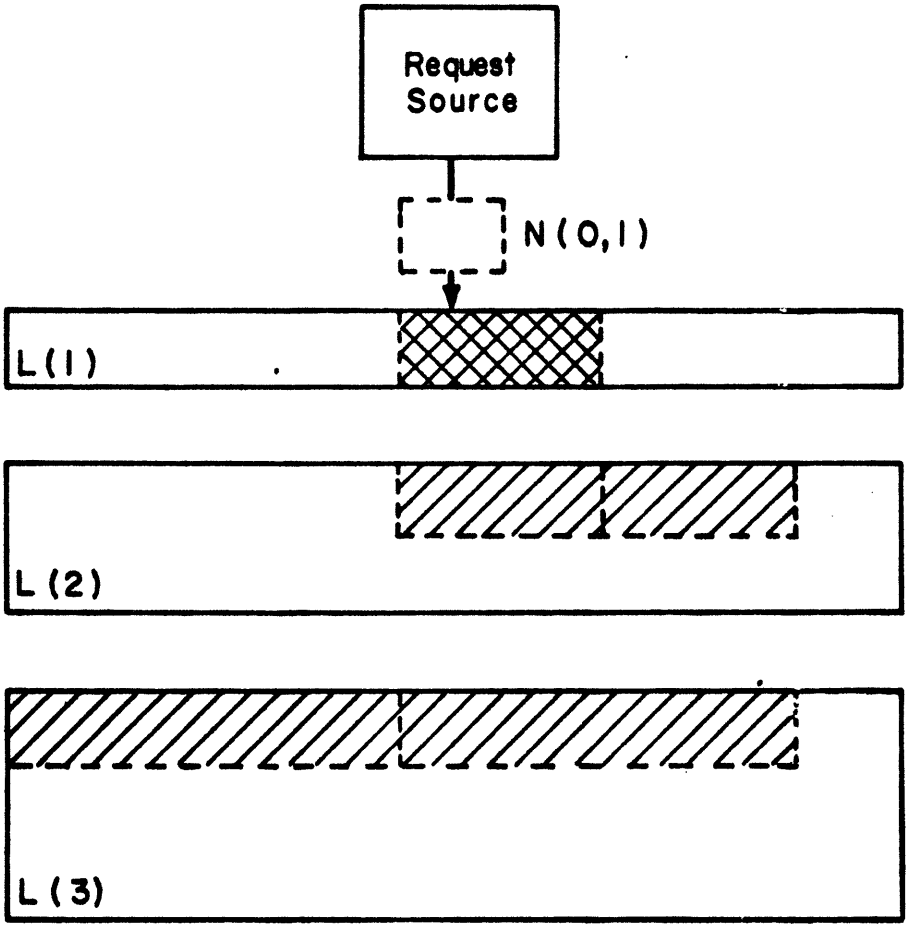
Since all data available at level i is also available at level i+1 (and all levels j > i), there is no information lost. Thus, no changes are needed to any of the other storage levels or the storage management algorithms although we would expect the performance to decrease as a result of the missing storage level. A limited form of this reliability strategy is employed in most current-day cache memory systems[5].

In a store behind strategy, all modified information is initially stored in L(1), the highest performance storage level. This information is marked "changed" and is copied into L(2) as soon as possible, usually during a time when there is no activity between L(1) and L(2). At a later time, the information is copied from L(2) to L(3), etc. A variation on this strategy is used in the Multics Multilevel Paging Hierarchy[8].

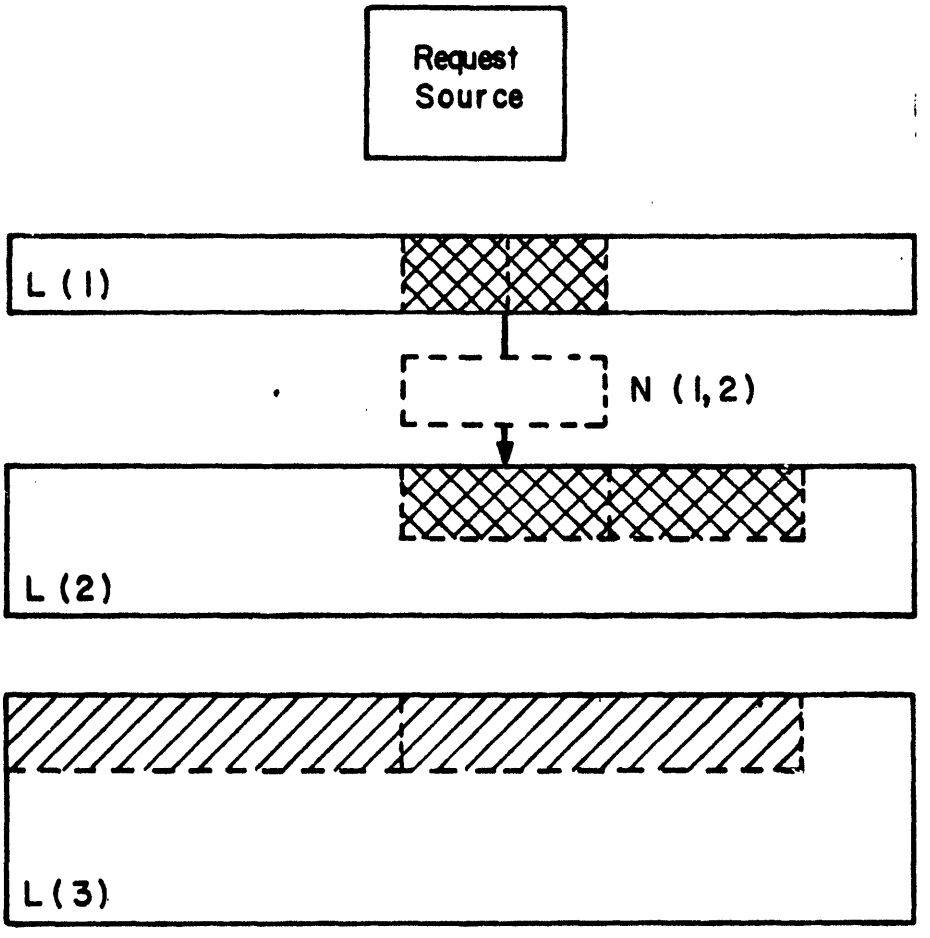
The store behind strategy can be used to provide high reliability in the storage system. Ordinarily, a changed page is not allowed to be purged from a storage level until the next lower level has been updated. This can be extended to require two levels of acknowledgement. For example, a changed page can not be removed from L(1) until the corresponding pages in both L(2) and L(3) have been updated. In this way, there will be at least two copies of each changed piece of information at levels L(i) and L(i+1) in the hierarchy. Other than delaying the time at which a page may be purged from a level, this approach should not adversely affect system

performance. As a result of this technique, if any level malfunctions or must be serviced, it can be removed from the hierarchy without causing any information to be lost. There are two exceptions to this process, L(1) and L(n). To completely safeguard the reliability of the system, it may be necessary to store duplicate copies of information at these levels only. Figure 6 illustrates the two-level store behind algorithm.

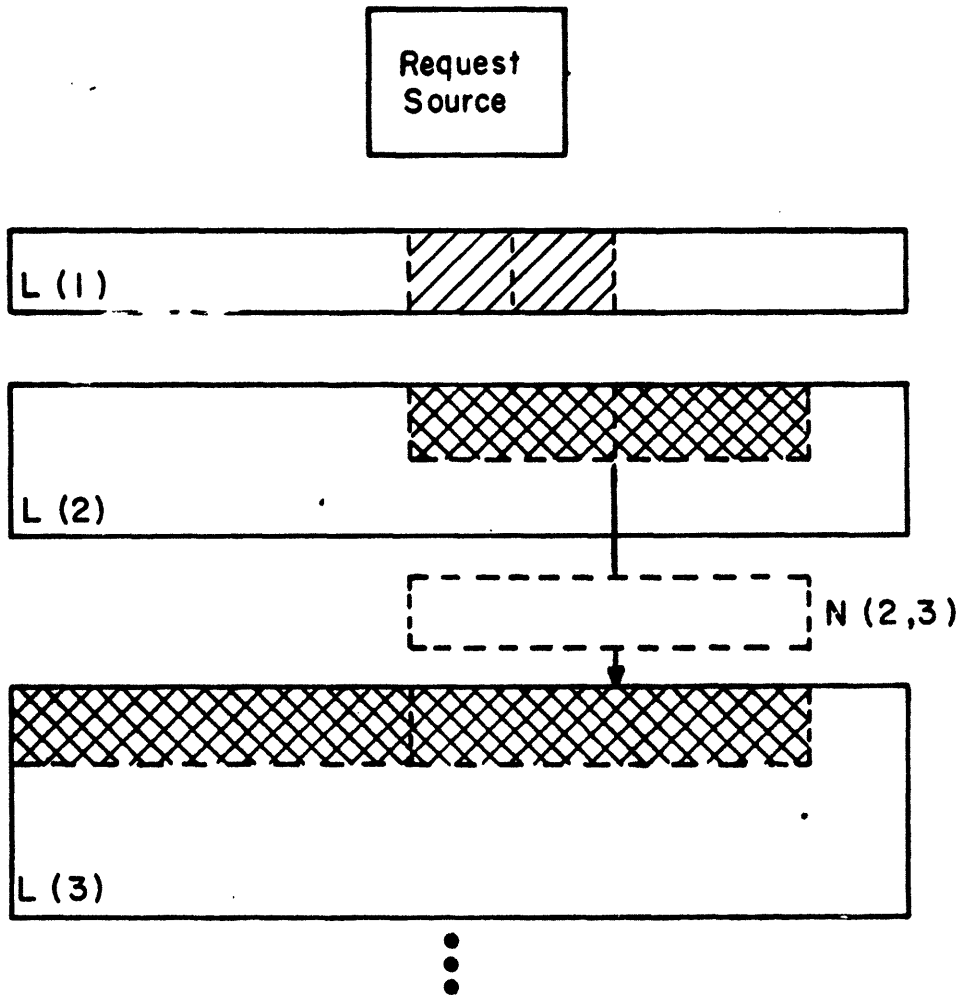
A considerable amount of work is required to manage the storage hierarchy. This work can be distributed by means of multiple processors servicing separate physical storage levels. In this manner, these processors may operate asynchronously and in parallel. A simplified example of this technique can be found in the IBM 3850 Mass Storage System[1,10].



(a) A processor stores into L(1), corresponding page is marked "changed".



(b) At a later time, the corresponding page in L(2) is updated and marked "changed".



(c) At a later time, the corresponding page in L(3) is updated and marked "changed". Since copies of the changed information exists in both L(2) and L(3), the "changed" indicator can be reset in L(1) and that page may be replaced if necessary.

Figure 6. Example of the Two-level Store-Behind Process

Decentralized Control

In order to develop information systems capable of managing extremely large memories and processing a tremendous number of requests, a design based upon decentralized control is essential. It is unlikely that a single processor would be capable of maintaining a centralized control over the large number of high-speed asynchronous operations needed. Furthermore, a centralized control could be seriously impared if there were a reliability failure. The hierarchical decomposition theory represents a straight-forward basis for a decentralized control design.

FURTHER RESEARCH

There are many areas of further research under investigation, such as:

Optimal function and physical decomposition. It is necessary to define a measure of performance and be able to prove that a particular decomposition is optimal.

Equivalence between functions. All information system functionalities must be mapped onto a standard set of functions. It is necessary to prove that the decomposed functionality is equivalent to the desired functionality.

Performance of physical implementation. There are various possible physical implementations of the optimal function and physical decomposition. It is necessary to provide measures for the performance of each such implementation and determine optimality.

Reliability. Although examples of reliability techniques have been described in this paper, a detailed reliability plan that encompasses all eventualities must be found.

Interlocks. Various interlock mechanisms must be used in an information system to coordinate various independent update operations. It is necessary to develop interlock techniques that lend themselves to a highly decentralized implementation without adversely effecting performance or reliability.

CONCLUDING COMMENTS

By using hierarchical physical decomposition (as depicted in Figure 3) to provide the virtual storage needed for the hierarchical function decomposition (as shown in Figure 1), a complete hierarchical decomposition of a large information management system can be attained. Furthermore, using the techniques illustrated in Figures 4 and 5, the hierarchical decomposition can be implemented by means of a microprocessor complex (i.e., the INFOPLEX).

Although such an INFOPLEX has not yet been completely implemented, portions are under development and investigation[20]. The overall theory of hierarchical decomposition remains an important research area.

REFERENCES

1. Ahearn, G. R., Y. Dishon, and R. N. Snively, "Design Innovations of the IBM 3830 and 2835 Storage Control Units," IBM Journal of Research and Development, Vol. 16, No. 1, pp. 11-18, January 1972.

2. Bensoussan, A., C. T. Clingen, and R. C. Daley, "The Multics Virtual Memory," Second ACM Symposium on Operating Systems Principles, Princeton Unviersity, pp. 30-42, October 1969.

3. Chu, W. W. and H. Opderbeck, "Performance of Replacement Algorithms with Different Page Sizes," Computer, Vol. 7, No. 11, pp. 14-21, November 1974.

4. Considine, J. P. and A. H. Weis, "Establishment and Maintenance of a Storage Hierarchy for an On-line Data Base Under TSS/360," FJCC, Vol. 35, pp. 433-440, 1969.

5. Conti, C. J., "Concepts for Buffer Storage," IEEE Computer Group News, pp. 6-13, March 1969.

6. Denning, P. J., "Virtual Memory," ACM Computing Surveys, Vol. 2, No. 3, pp. 153-190, September 1970.

7. Donovan, J. J. and H. Jacoby, "A Hierarchical Approach to Information System Design," MIT Sloan School of Management Report CISR-5 and WP 762-75, January 1975.

8. Greenberg, Bernard S. and Steven H. Webber, "Multics Multilevel Paging Hierarchy," IEEE INTERCON, 1975.

9. Hatfield, D. J., "Experiements on Page Size, Program Access Patterns, and Virtual Memory Performance," IBM Journal of Research and Development, Vol. 16, No. 1, pp. 58-66, January 1972.

10. Johnson, Clay, "IBM 3850 -- Mass Storage System," IEEE INTERCON, 1975.

11. Madnick, S. E., "Design Strategies for File Systems," MIT Project MAC Report TR-78, October 1970.

12. Madnick, S. E., "Storage Hierarchy Systems," MIT Project MAC Report TR-105, Cambridge, Massachusetts, 1973.

13. Madnick, S. E., and J. J. Donovan, Operating Systems, McGraw-Hill, New York, 1974.

14. Madnick, S. E., "Design of a General Hierarchical Storage System," IEEE INTERCON, 1975.

15. Martin, R. R., and H. D. Frankel, "Electronic Disks in the 1980's," Computer, Vol. 8, No. 2, pp. 24-30, February 1975.

16. Mattson, R., et al., "Evaluation Techniques for Storage Hierarchies," IBM Systems Journal, Vol. 9, No. 2, pp. 78-117, 1970.

17. Meade, R. M., "On Memory System Design," FJCC, Vol. 37, pp. 33-44, 1970.

18. Ohnigian, Suran, "Random File Processing in a Storage Hierarchy," IEEE INTERCON, 1975.

19. Ornstein, S. M., et al., "PLURIBUS -- A Reliable Multiprocessor," NCC, 1975.

20. Toong, H. D., "Microprocessor-based Multiprocessor Ring Structured Network," NCC, 1975.

21. Wensley, J. H., "The Impact of Electronic Disks on System Architecture," Computer, Vol. 8, No. 2, pp. 24-30, February 1975.

22. Withington, F. G., "Beyond 1984: A Technology Forecast," Datamation Vol. 21, No. 1, pp 54-73, January 1975.