# Problems, Directions and Issues in Memory Hierarchies*†§

Alan Jay Smith
Computer Science Division
EECS Department
University of California
Berkeley, California 94720
USA

**Abstract**

The effective and efficient use of the memory hierarchy of the computer system is one of the, if not **the** single most important aspect of computer system design and use. Cache memory performance is often the limiting factor in CPU performance and cache memories also serve to cut the memory traffic in multiprocessor systems. Multiprocessor systems are also requiring advances in cache architecture with respect to cache consistency. Similarly, the study of the best means to share main memory is an important research topic. Disk cache is becoming important for performance in high end computer systems and is now widely available commercially; there are many related research problems. The development of mass storage, especially optical disk, will promote research in effective algorithms for file management and migration. In this paper, we look at each component of the memory hierarchy and address two issues: what are likely directions for development, and what are the interesting research problems.

## 1. Introduction

The memory hierarchy of a computer system includes almost all aspects of a computer system's storage, including cache memory, main memory, disk, tape, and mass storage. The technology of the memory hierarchy has advanced as or more quickly than any other aspect of the computer system hardware, and the Williams Tubes, delay lines and rotating random access drums used in the 1940's and 1950's, with total storage of hundreds or at most a few thousands of words, have given way to cache memories with access times under 50 ns and main memories of over 32 megabytes. Despite this advance, effective and efficient use of the memory hierarchy is still one of the, if not **the** most important aspect of computer system design and use. This point is illustrated by comments in two recent papers (see also [Mati84]):

From [Hopk83]:

"The performance of the storage hierarchy may be more important than any details of the computational instruction set. This suggests an approach

*The material presented here is based on research supported in part by the National Science Foundation under grant MCS82-02591 and by the Defense Advanced Research Projects Agency under contract N00039-82-C-0235.

†This paper has been prepared by invitation for the 18'th Annual Hawaii International Conference on System Sciences, January 2-4, 1985, Honolulu, Hawaii.

§The opinions expressed in this paper are solely those of the author and do not represent his institution or sponsoring agencies.

for those who want to exploit VLSI design. Expend your effort on the memory hierarchy, not on exotic instructions."

From [Mate84]:

"The high performance 16-bit microprocessors introduced over the last five years have broken ground in a new market for microprocessors: high performance systems such as engineering and CAD workstations, and even general-purpose mainframe-level computers. The 16-bit microprocessors generally have plenty of computing power, but suffer in these applications from an inefficient use of memory. The principal purpose of the 32 bit microprocessors now reaching the market is to overcome this difficulty and to provide efficient engines for high performance systems.

"Designing high-performance microprocessor-based systems requires viewing the memory and its buses as the critical elements. DMA, graphics, and multiple CPUs must all contend for this resource, and the key design criterion for CPUs intended for this environment is that they provide high levels of computing power without hogging the bus."

Each component of the memory hierarchy is important for a different reason. The access time to the cache is often the critical path in the CPU. The size and ease of access to main memory affects both performance, through memory access time, paging, swapping and frequency of other I/O, and the ease of programming, in the ability to trade space for running time and programming time. The efficient use of disk affects performance, the size of the disk address space affects the ease of programming, the cost of the disk system is a major component of the cost of the overall computer system, and finally, the physical size of the disks is often a problem in the computer installation. Efficient use of tapes and mass storage impacts system performance, and overall cost by its effect on the number of disks required, operations cost and operations errors.

There are some changes common to all components of the memory hierarchy. First, all memories are becoming logically bigger. This is due to two related trends: memories are becoming denser and are also becoming cheaper; thus it is both physically and economically possible to increase the memory size. Second, the increasing performance of the processor is also necessitating additional memory. The performance of cache and main memories are increasing steadily, as are the capacities of disks.

Among the most important trends in general computer system design is one toward multiple CPU's and distributed systems. This is having and will continue to have an important effect on memory hierarchy design, and as explained below, accounts for some of the most pressing problems in memory hierarchies.

In the remainder of this paper, we present an up to date view of problems, directions and issues in memory hierachies; this paper can be considered to be an update of a previous survey of this topic by the author [Smit78e]. There will be a section each on cache memories, main memory, gap filler memory, disk, and tapes and mass storage. Some discussion will also be provided on the topic of the "logical" view of memory, as opposed to its performance and physical configuration.

## 2. Cache Memories

Cache memories are used in modern medium and high speed CPUs and new high end microprocessors to temporarily hold those portions of the contents of main memory which are currently (believed to be) in use. A thorough survey of cache memories appears in [Smit82] and has been updated in [Smit84a]; we assume that the reader is familiar with cache memories.

### 2.1. Basic Issues

The *two basic performance issues* in cache memories are access time and hit ratio. **Access time** is the time to read or write cache memory, when the desired information is cache resident, and **hit ratio** is the probability of finding the target of the reference in the cache. Access time is crucial because in many, most or almost all computers, *the cache access time is the critical path in the machine* and is the factor most tightly limiting the cycle time and overall machine performance. The hit ratio is important not only for the traditional reason, that it affects the average access time, but also for a relatively new reason: *memory or bus bandwidth is a critical and limiting resource in multiprocessor systems, and the hit ratio directly affects memory traffic.*

### 2.2. Multiple CPU Systems

Recent trends in computer system technology are encouraging the development of multiple CPU systems. There are two reasons for this tendency: (a) The performance of high-end CPUs is not keeping pace with the demands for CPU power in certain applications, such as modeling, simulation and numerical computation. (b) The cost-performance ratio is significantly better for small machines these days than large ones, which means that it is cheaper to get a given "amount" of computation by combining many small machines than having one (or few) large ones. (See [Saty80] for a bibliography on multiprocessor systems.)

*The basic and critical architecture and hardware issue in multiple CPU systems is that of resource sharing.* Such systems, in particular, may share many of the parts of the memory hierarchy, such as main memory, disk and mass storage. There are ***two resulting problems***: (a) ***maintaining consistency of shared and modifyable information***, and (b) ***avoiding or minimizing queueing and arbitratation delays in accessing the shared resources.*** Cache memories are directly concerned in both problems.

### 2.2.1. Cache Consistency

In the case that processors have cache memories and also share main memory, the problem is to ensure that the many processors see consistent values of the shared data. There are a number of ways to do that, none of them entirely satisfactory; a detailed discussion of these appears in [Smit82] and [Smit84a]; we summarize here. (1) The cache can be shared; this solution is usually poor, since the cache doesn't have sufficient bandwidth, and the shared design increases the access time. (2) All writes by each CPU can be broadcast to all other CPUs, and the relevant lines either updated or purged; this solution fails for more than 2 or 4 processors, as write traffic begins to interfere with access to each cache. (3) Directory methods (see, e.g. [Arch84] for a recent study) maintain distributed (in each CPU) or centralized (in main memory) directories, that ensure that for each line there is only one CPU able to access a line that has been or is about to be modified. (I.e. many readers or one writer.) Directory methods are expensive to implement and can slow down the caches and memory system due to the need to synchronize use of writable data. (4) In the event that an architecture is new, requirements may be placed on the software, causing it to issue certain hardware commands (e.g. cache purge) that will maintain consistency. This solution is only feasible if the architecture is new, so that old software need not be supported, and if the appropriate synchronizing commands have been implemented. (5) A special type of directory method, most suitable for multiple microprocessors sharing the same bus, has recently been developed and is the preferred method for this type of system. (See [Good83].) In it, all microprocessors have logic to watch the bus and ensure that the 1 writer /many readers condition holds at all times. The principal limitation of this method is that the bus must be shared; the method does not extend to an arbitrary number of processors.

Each of the above methods has been or is being implemented in one or more systems, but as noted, none effectively and efficiently solves the general problem of maintaining consistency among N processors sharing memory, where N is reasonably large (e.g. >4). ***Finding such a solution to the cache consistency problem is a difficult, significant and important research problem.***

### 2.2.2. Cache and Memory Bandwidth

The second major problem with shared memory is that of memory bandwidth, and this problem is most apparent in the case of multiple microprocessors sharing a single memory bus. (See [Bask76] for an analysis of interleaved memory.) One would like to connect several microprocessors to one memory bus. Using a high performance microprocessor, and a moderate performance bus, it is possible to saturate the bus with from 1 to 5 microprocessors; the addition of more microprocessors brings no increase in overall performance, as the processors spend their time waiting for memory access.

*Cache memories are one of the primary mechanisms to solve the memory bandwidth problem.* A cache memory can be associated with each processor, either on the microprocessor chip or off-chip. If such a cache has a 16 byte line, a 5% miss ratio per instruction, reads 6 bytes per instruction and writes 2 bytes per instruction, and uses write through, then the memory traffic has dropped from 8 bytes per instruction to 2.8 bytes per instruction. With copy back, and half of the replaced lines being dirty, the memory traffic drops to 1.2 bytes per instruction, an 85% decrease. Thus, ***cache memories will be necessary for high performance multi-microprocessor computer systems.*** The trend, then, will be for future high performance microprocessors to have on-chip or outboard cache memories. (However, it is possible to design a cache memory in such a way that memory traffic actually increases; we are presuming good design.)

### 2.3. On Chip Cache

With increases in circuit density and chip area, it is becoming possible to place useful cache memories on the microprocessor chip; the new Motorola 68020 [Moto84] has a 256 byte instruction cache on chip, and the Zilog Z80,000 [Alpe83] has a general purpose 256 byte on-chip cache. These caches serve two purposes: they significantly decrease memory access times on hits, and they also reduce the bus traffic.

There are a number of design issues regarding on-chip caches; these issues have been addressed primarily in the context of large, mainframe sized caches and need to be reconsidered for small caches. Further, the tradeoffs are somewhat different for on-chip caches, since transfer times dominate latency times for misses, in contrast to larger machines, and main memory traffic is also important. These differences affect optimal choices for parameters such as line size and cache organization; in [Hill84] some of these parameter choices are considered and the sub-block cache organization (sector cache) is analyzed and evaluated. Also, VLSI permits "cheap and easy" associativity, which affects the design (fully associative vs. set associative) of caches and TLBs.

The trend here is toward on-chip caches. The research issues are ones of evaluating and selecting cache design parameters in the context of small size and limited off chip bandwidth, and the availability of "easy" associativity.

## 2.4. Off Chip Cache

Of no less interest are off-chip caches. Microprocessors recently announced and/or under development are sufficiently powerful that they can benefit from, and in fact need, more cache than current technology permits to be on-chip. A single cache chip could be expected to hold 4Kbytes or more of cache, and a cache board could easily contain upwards of 32Kbytes of cache. We can expect to see within the next 1-3 years the availability of cache chips, and also of boards with a substantial amount of cache in addition to the microprocessor. Research issues here are the same as apply to cache memories in general, including virtual vs. real address caches, data/instruction caches, multilevel caches, line size, cache size, cache organization and associativity, main memory update algorithm, and multicache consistency. Some of these are further discussed below.

## 2.5. Multilevel Cache

There are trends in computer architecture suggesting the further development and use of multilevel cache. On-chip or on-board caches may be too small to be fully effective, but are much faster than more remote caches. Similarly, it may be cost effective to use different consistency methods for each cache level, with different cost and performance tradeoffs. Fujitsu, in its Facom 382 [Fuji82, Hatt83] uses a two level cache and reports that such a design has advantages. We believe that multilevel caches will become more common.

## 2.6. Virtual Address Caches

Most cache memories are addressed using real addresses; see [Smit82] for a discussion. There are performance advantages, however, for caches addressed using virtual addresses, as is done in the Amdahl 580 [Amda82]. We predict greater use of this design, especially in new designs and architectures where the synonym problem can be eliminated or avoided. (The *synonym problem* occurs when two different virtual addresses map into the same real address.)

## 2.7. Data and Instruction Caches

Most current cache designs use a single cache which serves for both instructions and data. The advantage to splitting the cache into instruction and data halves is that the bandwidth is doubled; the disadvantage is that if instructions can be modified, then new consistency and correctness problems arise. For newer machine designs, in which compatibility with old, self-modifying software is not a problem, we predict that split caches will become increasingly frequent.

## 2.8.

Microcode Caches In one special case, that of a cache for microcode, the workload is predictable and static. In that case, the cache can be optimized for the workload, and conversely, the workload (microcode words) can be specially modified to instruct the cache with respect to fetching, replacement and branching. The identification and parameterization of such optimizations is an open and useful research problem.

## 2.9. Vector Processors and Caches

Vector processors rely on a steady stream of data to drive the vector unit. Cache memories can be a problem in such a system, since cache misses cause the vector unit to wait until the main memory fetch completes. This can affect performance and can also cause increases in complexity in the vector and cache control logic. (Page faults are an even worse problem.) The proper design of a cache in the context of a vector processor has not, to the author's knowledge, been addressed in the research literature and is an interesting problem.

## 2.10. Workload and Performance Evaluation

The primary technique used for the performance evaluation of cache memory designs is trace driven simulation. There has been a tendency, however, to use small applications programs for these traces, and then to find that actual cache performance is significantly worse than predicted, since large programs and systems programs tend to dominate. The problem of selecting an appropriate workload is very important to any cache memory evaluation or research effort and is being addressed in [Smit84c].

## 2.11. And So What Else is New?

Cache memories have existed since the late 1960's and the IBM 360/85 [Lipt68], and yet are still an active area of research and are an important aspect of computer design; the extensive bibliographies of [Smit82] and [Smit84a] testify to this. The author also consults widely on the subject of cache memories and with every new design and system, there are special twists that raise new issues. We expect that the activity in cache memory studies will continue for several more years.

## 3. Main Memory

The use and management of main memory has been an active area of research since the early to mid 1960s; see [Smit78d] for a large bibliography of the relevant literature and [Denn80] for an overview of the research. Traditionally, the research issues have stemmed from the fact that up until the late 1970's, memory was expensive and thus was a critical resource; the efficient use of memory was very important. Problems relating to paging, such as replacement algorithms and control of the degree of multiprogramming were stressed. These problems were exacerbated by the many computer architectures with too few bits of addressing, e.g. 16, which also limited the amount of memory that could be usefully used.

*Within the last few years, main memory has become both plentiful and cheap.* At a price (in quantity) of $2 for a 64Kbit memory chip, the parts for a megabyte of memory cost about $250; it is possible to buy main memory on a board with all access logic for less than $5000/megabyte. Home computers such as the Apple MacIntosh are available with up to 512Kbytes and microprocessor based workstations such as the SUN come with 2-4megabytes. (Apple is offering to add 384K of memory to the MacIntosh for $995.) Thus, except for rare cases, *memory is no longer a scarce resource, and the traditional research problems are of much less interest.*

There are still some research and development problems relating to main memory. Probably the most important question is *how to design a cost and performance effective shared main memory.* For example, a crossbar, such as was used in C.mmp [Wulf72] is expensive, somewhat slow, has some queueing delays [Bask76], and poses reliability problems. For a project at the University of Texas, [Oppe83], a banyan network is planned. Many of the various multiprocessor research and development projects have as their central issue the processor and memory interconnection strategy, and its impact on access time, cost, reliabilitv and queueing delays. This is and is likely to continue to be an important and difficult research problem.

Another important question, which has thus far been left almost entirely to the memory manufacturers, *is what is the best functional design for a high density memory chip?* Should the data come out 1 bit wide, 4 bits wide, 8 bits, or more? If it is desired to read a word at a time, and a word is 32 bits, then 32 chips one bit wide are required; if these are 256Kbit chips, then 32 chips yield a megabyte, which is too much for some applications. It has been remarked to this author that current chip designs are very poor for computer designers; memory chips should be designed to latch the inputs and outputs [Koto84]. Now the external logic to perform this function is expensive and it has a negative impact on performance. Another question is whether "nibble mode" (by which additional sequential bits can be obtained in much less time than the first bit referenced) is useful and represents a good design choice.

The function of *replacement in extremely large main memories* has also been described to the author as a problem. The clock replacement algorithm [Smit78c] is easy to implement given only a reference bit, but the number of page frames that have to be examined becomes unreasonably large in very large memories. The use of set associative replacement for main memories has been previously suggested and analyzed [Smit78a] and was found to be quite effective; the use of set associative clock replacement should provide an effective replacement algorithm yielding adequate performance at low overhead and needing no additional hardware.

Another aspect of the replacement problem is to find a way to have a consistent reference bit, when there are multiple cache memories in the system, and the reference bit is maintained locally.

A number of factors suggest that *optimal page sizes will increase* over the next decade. Large pages load and transfer information with fewer page faults, if memory is plentiful. TLBs (of a constant number of entries) can address more memory as page size increases. The number of sets can be increased in real address caches as page size increases, since additional bits are available for addressing without translation. As transmission speeds to and from I/O devices increase, the delay due to large transfers will decrease relative to latency and will become less significant. Right now, 4K pages are a reasonable choice; 512 bytes, as is used on some smaller computers is clearly much too small. We believe that over the next decade, page sizes of 8K or 16K will become desirable.

## 4. Gap Filler and Disk Cache

The management of main memory has in the past been an interesting research problem because the large gap in access times between main memory (<1 microsecond) and disk storage (10-100ms) meant that transfers, expecially due to page faults, often caused substantial CPU idle time. This large access gap still exists, and is likely to get worse, since CPUs and main memory continue to get faster; disk performance has improved very little recently and is not likely to improve overall by a significant amount [Hoag79]. The impact of the access gap still exists, but its focus has shifted more towards explicit I/O. Essentially, the problem is that with the increasing density of disks, their nonincreasing performance and the slowly or nonincreasing number of data paths to disk, *the disk system will be unable to provide sufficient I/O bandwidth to serve high performance CPUs and multiprocessor systems.* This problem is not yet major, but is expected to become worse in the next few years.

The existence of a "gap filler technology", i.e. one intermediate in performance and cost between main memory and disk, would possibly provide a solution to the performance bottleneck projected above. There is, however, no such technology available. Although a few years ago CCDs, magnetic bubles and EBAM (electron beam accessed memory) were considered promising, none is viable for a gap filler role at this time nor is likely to be soon. (See e.g. [Spec84].)

It is possible to make the useful observation that there is significant locality in I/O reference patterns: data is both referenced sequentially and for some data sets, there is significant reuse; see [Smit84b, Smit75, Smit76, Smit78b] for data supporting this observation. Thus it would make sense to cache portions of the disk address space in main memory or outboard in an MOS RAM based disk cache. The effectiveness of this idea is shown in [Smit84b]. There are already several disk cache products, including the Sybercache by STC [Stor82] and the IBM 3880 model 11 and model 13 storage controllers [IBM81,IBM83]; for both companies' products, the cache is outboard at the storage controller; NEC has a disk cache which is associated with although outboard to the CPU [Toku80].

The *design of disk caches* has been neglected in the research literature (except for [Smit84b]) and is a fruitful area for study. Questions to be answered include: what is the best location in the system for a cache? How large should the cache be for good performance? What algorithms should be used for fetch and replacement? How large should the blocks be? Should the design be write-through or copy-back, and what are the performance implications of each? The answers to these questions are quite sensitive to the workload and additional workload data needs to be gathered.

The disk caching problem also extends to distributed systems. A recent trend in computer systems is toward a number of processors, including personal computers, workstations, and mainframes linked together with a local area network, and backed by a file server. In such a system, the file server and the network are limited resources, both subject to congestion, and the overheads of remote I/O are substantial. Thus, there are benefits to be gained in caching at the processors, either on local disk or in RAM, and also perhaps to caching in RAM as well at the file server itself, to avoid unnecessary physical disk I/Os. The *research problems of caching in a distributed system* include not only those mentioned above, but the following: should caching occur at the processor, at the file server or both? What policies and parameters are appropriate at each, and are the same or different choices appropriate? How can one maintain consistency in such a system when multiple copies of the same data can exist?

We believe that the use of disk cache will be both more important and more common in the future. The number of interesting research problems in disk cache will also stimulate related research activity.

## 5. Disks

Disk technology is evolving slowly and has been so for some time [Hoag79]. Disk density is increasing at about 20% per year; i.e. it doubles every 3-4 years. Conversely, *disk access times are improving very little if at all,* with rotation times remaining essentially flat and arm seek times dropping at a slow rate. These trends should continue at comparable rates over much of the next decade.

It can be expected, however, that *disk I/O transfer rates will increase significantly.* Currently, they reflect the linear density of bits on the disk surface and also the data transmission technology between disk and the CPU; the transmission rate on high end IBM systems is 3 megabytes/second. There are three reasons to expect this rate to increase: (a) Increasing physical bit density on the disk surface will increase the rate of reading and writing. (b) Technologies such as optical fibers are becoming available and cost effective as a way to achieve high I/O rates. (c) Performance considerations suggest that higher transfer rates will help alieviate the bottleneck discussed in the section above on gap fillers.

*There is a very rapid rate of change and improvement at the "low end" for hard disks.* It is possible to buy 8 inch disks with up to 200 megabytes and 14 inch disk drives that fit in a rack mount, take less than 4 cubic feet and hold close to 500 megabytes for under $10,000. These high density small disks are rapidly being incorporated into high end personal computers, workstations and small shared computers. Performance and density will continue to increase and cost will continue drop. Local disks are becoming an important part of the memory hierarchy in a distributed system and will become more so.

Another likely change in disk system design is to *associate more electronics with each disk.* These electronics will act to correct errors, buffer tracks, cache information and/or buffer I/O data streams so as to mask physical latency and decouple transmission from physical disk position and rotation speed. This change is long overdue and should begin to occur at any time.

There are a few areas of research in disk systems. *The most promising is the general area of I/O optimization,* which is surveyed in [Smit81b]; see also [Smit81d].

With changes in technology, optimal solutions to issues such as block size, angular placement of blocks, disk loading, etc. change. For a recent implementation of many disk optimizations, see [Mcku84]. Another research problem is to determine the most promising uses of electronics in the disk system.

## 6. Mass Storage and Tape

We define mass storage to be any storage system with significantly larger capacity than disk, and it is usually cheaper per byte stored and slower than disk. Included in this category are tape and tape subsystems (such as the IBM 3850, the Ampex Terabit memory and the Calcomp automated tape library) and optical disk. The technology currently undergoing the most rapid development is optical disk, and Storage Technology is promising a very high density and cost effective optical disk system in the near future. We believe that over the near term, optical storage technology will advance the most rapidly, and within 5 or 10 years, *very large optical archival stores will become common in large computer systems*. No other technology seems to be competitive.

It is worth noting that IBM has announced a *new high density tape technology*, using tape cartridges. These new tapes and drives have advantages in terms of density and convenience, but are likely to come into use only slowly. It could easily be a decade before the majority of drives in a typical high end installation are of the new type, especially since not only are the drives incompatible with the existing ones, but so are the tapes.

Mass storage can be and is used for explicit I/O, whereby one issues read and write commands to specific devices and data volumes. The most promising use for mass storage, however, is as an automatic backing store for disk. In that circumstance, the disk address space would be expanded, much like virtual memory is used to expand main memory addressing, and automatic migration algorithms would be used to move data between mass storage and disk as needed.

There are numerous research problems relating to the efficient and effective use of mass store as an automatic backing store for disk. Questions such as when to fetch data, how much to fetch, when to remove data on disk and migrate it back to mass store, when to compact mass storage volumes and which volume to transfer it to are all interesting and significant research problems. Some relevant work on file replacement algorithms and file reference patterns appears in [Smit81a] and [Smit81c]; see also [Lawr82] for more work in the same area.

## 7. Logical View of Memory Hierarchies

In addition to the problems discussed above, which are primarily concerned with the physical design and algorithmic use of memory hierarchies, there are some interesting issues having to do with the logical view of the memory system.

The most important issue is one of *how large an address space is needed*. It has only been recently that there has been a general realization that 16 bits of addressing are **not** enough, and the newest generation of microprocessors use 32 bits of address. Likewise, the IBM System 370 architecture has been extended in the newest machines (308x series machines [IBM82]) to use 31 bits of addressing. 31 or 32 bits (4 gigabytes) should be sufficient through 1990 or 1995 for a single processor system, but one can expect to need more addressable memory for a uniprocessor sometime in the 1990's, and earlier for a shared memory multiprocessor system. We expect, therefore, to see architectural changes and extensions to permit this over the next ten years. (Anyone designing a computer at this time would be wise to keep this in mind.)

The idea of object based architectures and capability based addressing was a popular one in the 1970s, and machines such as the Intel 432, the Cambridge CAP machine and the IBM System/38 all embody and use such concepts. Despite the advantages in programming productivity and data security achieved by that approach, the current belief is that inherant performance penalties of such architectures suggest instead the use of simple load/store (reduced instruction set -like) architectures with simple addressing. This issue is still an appropriate one for research and it may yet be possible to use a sophisticated logical addressing scheme and yet have good performance.

Protection in most computer systems is primarily associated with memory and the memory hierarchy. Protection bits are usually associated with pages, segments and/or files. Existing protection systems tend to be insufficient to stand up under sophisticated penetration efforts, and further research in this direction may be warranted.

It has long been a goal of computer designers and users to have a one level store, in which all stored information would be addressable within the same address space. Virtual memory is a step in that direction, but does not usually include the file system. We expect some additional research and possibly minor commercial moves in the direction of a one level store, but over the near term, results are not likely to be substantial.

## 8. Conclusions

In this paper, we have reviewed memory hierarchies, and have addressed two particular points: (a) what are the likely directions for the development of memory hierarchies, and (b) what are the interesting research problems. As was explained in the introduction, the memory hierarchy is one of the or perhaps the most important part of the computer system with respect to both performance and utility. We therefore believe that with respect to both research and development, memory hierarchies will be a central area of focus over the next decade.

## Bibliography

[Alpe83] Don Alpert, Dean Carberry, Mike Yamamura, Ying Chow and Phil Mak, "32-bit Processor Chip Integrates Major System Functions", Electronics, July 14, 1983, pp. 113-119.

[Amda82] Amdahl Corp., "580 Technical Introduction", 1982.

[Arch84] James Archibald and Jean-Loup Baer, "An Economical Solution to the Cache Coherence Problem", Proc. 11'th Annual Symposium on Computer Architecture, June 5-7, 1984, Ann Arbor, Mi., and in SIGARCH Newsletter, 12, 3, June, 1984, pp. 355-362.

[Bask76] Forest Baskett and Alan Jay Smith, "Interference in Multiprocessor Computer Systems with Interleaved Memory", CACM, 19, 6, June, 1976, pp. 327-334.

[Denn80] Peter Denning, "Working Sets Past and Present", IEEETSE, SE-6, 1, 1980, p. 64-84.

[Fuji82] Fujitsu Corp., "FACOM M-382", third edition, September, 1982, Tokyo, Japan.

[Good83] James Goodman, "Using Cache Memory to Reduce Processor-Memory Traffic", Proc. 10'th Ann. Symp. on Computer Arch., June, 1983, pp. 124-131.

[Hatt83] Akira Hattori, Minoru Koshino and Shigemi Kamimoto, "Three Level Hierarchical Storage System for Facom M-380/382",

[Hill84] Mark Hill and Alan Jay Smith, "Experimental Evaluation of On-Chip Microprocessor Cache Memories", Proc. 11'th Annual Symposium on Computer Architecture, June, 1984, Ann Arbor, Michigan, pp. 158-166.

[Hoag79] A. S. Hoagland, "Storage Technology: Capabilities and Limitations", Computer, 12, 5, May, 1979, pp. 12-18.

[Hopk83], M. E. Hopkins, "Compiling High Level Function on Low Level Machines", Proc. IEEE International Conference on Computer Design: VLSI In Computers, November, 1983, Port Chester, New York, pp. 617-619.

[IBM81] International Business Machines Corp., "Introduction to IBM 3880 Storage Control, Model 11", IBM Pub. No. GA32-0060, Sept., 1981, IBM Corp., Tucson, Ariz.

[IBM82] IBM Corp., "IBM 3081 Functional Characteristics", GA22-7076, Poughkeepsie, New York, 1982.

[IBM83] International Business Machines Corp., "Introduction to IBM 3880 Storage Control Model 13", IBM Pub. No. GA32-0062, January, 1983, IBM Corp., Tucson, Ariz.

[Koto84] Alan Kotok, private communication.

[Lawr82] D. H. Lawrie, J. M. Randal and R. R. Barton, "Experiments With Automatic File Migration", IEEE Computer, July, 1982, pp. 45-55.

[Lipt68] J. S. Liptay, "Structural Aspects of the System/360 Model 85, II The Cache", IBM Systems J., 7, 1, 1968, pp. 15-21.

[Mate84] Richard Mateosian, "System Considerations in the NS32032 Design", Proc. NCC, 1984, pp. 77-81.

[Mati84] R. E. Matick and D. T. Ling, "Architecture Implications in the Design of Microprocessors", IBM Systems J., 23, 3, 1984, pp. 264-280.

[Mcku84] Marshall K. Mckusick, William Joy, Samuel Leffler, and Robert Fabry, "A Fast File System for UNIX", ACM TOCS, 2, 3, August, 1984, pp. 181-197.

[Moto84] Motorola Corporation, "MC68020 Technical Summary", 1984.

[Oppe83] Eli Opper, Miroslaw Malek and C. Jack Lipovski, "Resource Allocation in Rectangular CC-Banyans", Proc. 10'th International Symposium on Computer Architecture, June, 1983, Stockholm, Sweden, (also Sigarch News, 11, 3), pp. 178-184.

[Saty80] Satyanarayanan, "Multiprocessing - An Annotated Bibliography", IEEE Computer, 13, 1980, pp. 101-116.

[Smit75] Alan Jay Smith, "A Locality Model for Disk Reference Patterns", Proc. IEEE Computer Society Conference, February, 1975, San Francisco, Ca., pp. 109-112.

[Smit76] Alan Jay Smith, "Analysis of a Locality Model for Disk Reference Patterns", Proc. Second Conference on Information Sciences and Systems, The John Hopkins University, Baltimore, Md., April, 1976, pp. 593-601.

[Smit78a] Alan Jay Smith, "A Comparative Study of Set Associative Memory Mapping Algorithms and Their Use for Cache and Main Memory", IEEETSE, SE-4, 2, March, 1978, pp. 121-130.

[Smit78b] Alan Jay Smith, "On the Effectiveness of Buffered and Multiple Arm Disks", Proc. Fifth Computer Architecture Symposium, April, 1978, Palo Alto, Ca., pp. 242-248.

[Smit78c] Alan Jay Smith, "Sequentiality and Prefetching in Data Base Systems", IBM Research Report RJ 1743, March 19, 1976, and ACM Transactions on Data Base Systems, 3, 3, September, 1978, pp. 223-247.

[Smit78d] Alan Jay Smith, "Bibliography on Paging and Related Topics", Operating Systems Review, 12, 4, October, 1978, pp. 39-56.

[Smit78e] Alan Jay Smith, "Directions for Memory Hierarchies and Their Components: Research and Development", Proc. COMPSAC Conference, Chicago, Ill., November, 1978, pp. 704-709.

[Smit81a] Alan Jay Smith, "Analysis of Long Term File Reference Patterns for Application to File Migration Algorithms", IEEETSE, SE-7, 4, July, 1981, pp. 403-417.

[Smit81b] Alan Jay Smith, "Input/Output Optimization and Disk Architecture: A Survey", Performance Evaluation, 1, 2, 1981, pp. 104-117.

[Smit81c] Alan Jay Smith, "Long Term File Migration: Development and Evaluation of Algorithms", CACM, 24, 8, August, 1981, pp. 521-532.

[Smit81d] Alan Jay Smith, "Bibliography on File System and Input/Output Optimization and Related Topics", Operating Systems Review, 15, 4, October, 1981, pp. 39-54.

[Smit82] Alan Jay Smith, "Cache Memories", Computing Surveys, 14, 3, September, 1982, pp. 473-530.

[Smit84a] Alan Jay Smith, "CPU Cache Memories", to appear in *Handbook for Computer Designers*, ed. Flynn and Rossman.

[Smit84b] Alan Jay Smith, "Disk Cache - Miss Ratio Analysis and Design Considerations" submitted for publication.

[Smit84c] Alan Jay Smith, "The Effect of Workload Choice on Cache Memory Evaluation", in preparation.

[Spec84] "Whatever Happened to Magnetic Bubble Memories", IEEE Spectrum, September, 1984, p. 22.

[Stor82] Storage Technology Corporation, "Sybercache 8890 Intelligent Disk Controller", Louisville, Colo. 1982. Proc. IFIP 1983, pp. 693-697.

[Toku80] T. Tokunaga, Y. Hirai and S. Yamamoto, "Integrated Disk Cache System With File Adaptive Control, Proc. IEEE Computer Society Conference, September, 1980, pp. 412-416.

[Wulf72] W. Wulff and C. G. Bell, "C.mmp, a Multi-Mini Processor", Proc. FJCC 1972, pp. 765-777.