DATA BASE PROCESSOR MAGE

G.BERGER SABBATEL - Ph.NAVAUX*
Computer Architecture Group
IMAG
BP 53X - 38041 GRENOBLE cédex - France

ABSTRACT

In this paper, we present the design of data base processor MAGE. This DBP is based on a hierarchical DB access method. It is composed of two microprocessors, a disk processor and a moving head disk. The processor requirements, design decisions, and architecture are discussed.

We start with an overview of DBP framework. The MAGE DB access method is then summarized. Finally, we present the design and implementation of the DBP, with a particular emphasis on the disk processor.

I - INTRODUCTION

During the last few years,one of the most active research areas in computer architecture has been the data base machines(DBMS). These machines consist of specialized computer hardware supporting basic data base management functions found in most contemporary software management systems [HSI79].

Two factors have contributed significantly to the appearence of the DBM: the distributed systems ans the new technologies. The interconnection of computers and processors has resulted in different systems namely"distributed systems", "distributed functions computers" or " computer networks"[AND75]. By distributed processing we mean a logical set of processing functions implemented on a number of physical devices, so that each device performs part of the required processing job. In conventional centralized computer systems, processing may be seen as a multilayer structure in which the nucleus represents the user tasks, the first layer represents the system resources and the external layer represents the peripheral devices.From this point of view,a distributed system (fig.I.1) results from the splitting of these layers, which creates specialized modules.Some of these modules, called functional processors,consist of peripheral devices,their management functions and the system functions they need.The other modules,called computing processors,are created by splitting the user tasks [ANC78]. The functional processors are dedicated to particular functions such as data base management,peripheral device control, library management, etc...

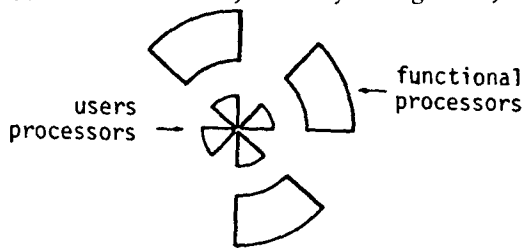

Fig.I.1. Distributed system

Thus a DBP could be used either in centralized system as an intelligent peripheral device,or as a functional processor in a functionally distributed system.The DBP relieves the overloaded centralized computer systems from their DB management tasks.This is an important point,since it allows the DBP to progress from present centralized computer systems towards the future distributed systems.

The other area that contribute to the DBMS are the new technologies.The increasing performance/cost ratio of microprocessors is making their inclusion in large systems.Now it is possible to put "intelligence" closely where it is needed.

The MAGE project("Matériel Adapté à la Gestion d'Entités")[BER78][NAV79] was oriented in two directions firstly,to organize the machine as a distributed system, and secondly to realize the machine using microprocessors. Present research data base processors is carried out in areas surch as:associative processors for relational data base, processors with data base primitives,special purpose d.b. processors to be use

---

* *On leave from UFRGS, Pos Graduçao em Ciencia de Computaçao, 90 000 Porto Alegre, BRAZIL*

. *This work was supported by IRIA under contract n°74136 and was carried out in cooperation with SAGEM C°.*

in heterogeneous networks and DBP managed by a set of microprocessors [MAR78].

The MAGE project is mainly concerned in the investigation of the fourth domain namely "DBP managed by a set of processors" and with the following principal motivations:

a) to show a DB access method implemented with microprocessors,
b) to show the feasibility of a DB hardware processor adapted to small systems,
c) to free central units from the DBMS processing,
d) to oriente the DBP suitable in the future distributed systems,
e) to minimize the maintenance and design costs of the system,
f) to be adaptable to updating with the evolution of new technologies.

The main objective of the design was to obtain a data base access method with microprocessors.Unlike the projects CASSM [COP73], RAP [OZK75] or RARES [LIN76] which use hardware associative techniques to access data with one cellular logic devices per head of disks units, MAGE is simpler, economic and implements the DB access method using microprocessors so that to retrieve the data, a computed addressing is used instead of associative addressing.

The main problem in most of the DBP based on associative access, is their use of some specific mass memory technologies like fixed head disks with parallel access to each tracks, or new technologies like bubble memories.

As MAGE is oriented towards industrial production, we had to proceed in a very pragmatic way, in order to be able to adapt MAGE to any future new mass memory technology.

In effect, it is difficult to predict the probable future mass memory technology :

- new technologies like bubbles,
- a compromise between disks and new technologies,
- some new architecture of disk memories,
- ...

Therefore if MAGE is designed to use disks memories, the design of the access method must be independent of the mass memory technology, and the best solution that appeared to us was to implement a computed addressing. Such an addressing would be easily implemented with any mass memory technology.

The centralization of some access descriptors will allow some compromise between disk memories and new technologies, with an important upgrading of performances, since such information requiring a high access rate and a low volume can be stored in faster memories.

Some associative operations will be accelerated through very simple pattern research techniques, implemented on the mass memory controller (disk controller).

II - MAGE : A DATA BASE ACCESS METHOD

The DBP architectural aim is to move the data base management task from central processors to a processor closely associated with the mass memory device [AND76].

The previous section gave the framework on which the MAGE project is based. We have seen that a DBP can be a functional processor in a distributed system. We will now discuss its main characteristics.

MAGE is an access method and it has the following general specifications :

a) to manage more than 100 M bytes of secondary storage,
b) to allow other processors to share its secondary memory,
c) to be designed independently from the user processor,
d) to support simultaneous dialog with up to 32 user processors,
e) to provide good information security.

The access method was defined from a benchmark study [BER1.78][BER78]. This study showed that hierarchical D.B. possibilities are sufficient for solving most storage management problems for small real-time business systems. Consequently, an access method can be based upon an existing D.B. access method for conversational systems such as SOCRATE[ABR70], a data base system developed at the University of Grenoble.

The D.B. access method MAGE was defined from SOCRATE

by taking into account its easy adaptation to small systems, modifications in the data access method being made necessary due to the fact that MAGE will be implemented with microprocessors.

The resulting access method to data is made through a structure description and a virtual space of pointers called internal names.

When defining the data structure, the user must specify the maximum number of instances that every record can have. This permits statical assignment of an internal name to each instance which can exist. The structure file contains the characteristics of the data block and the necessary information for calculating the internal names. The link between an internal name and the corresponding data block in the disk, if it exists, is insured by the dictionary. This dictionary, which is accessed by hash coding on the internal names, contains the disk address of the data block. The dictionary will occupy about 1/6 of the disk capacity used, and the size of the structure file is generally less than 4 Kbytes (Fig.II.1.).

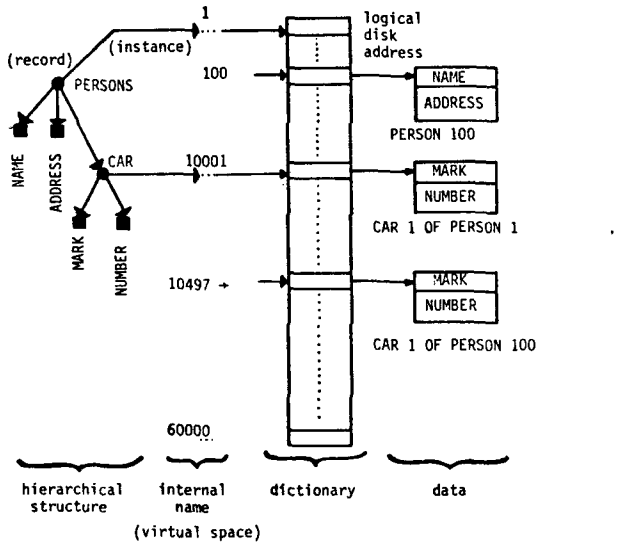

Fig.II.1. Addressing mode

III - SOFTWARE ARCHITECTURE

The processing activity of the access method can be decomposed into 7 modules (see Fig. III.1) :

EMP = echange management process, which interfaces users and MAGE,

RP = request process: it calculates the internal name and commands the access to the information instance through INP,

SAP = structure access process: it provides a structure record from its number and optimizes the accesses to the structure file,

INP = internal name process: it performs the access to an instance, knowing its internal name, through DAP and RAP,

DAP = dictionary access process: its performs the access to an entry of the dictionary, knowing its internal name,

RAP = realization access process: it performs the access to an item (entity) of information (instance) whose length, internal name and address on the disk are known,

DMP = disk management process: it converts the given logical address into a physical address. It manages the requests and optimizes the moves of the disk heads. This process must be modular enough to be adapted to different disk configurations.

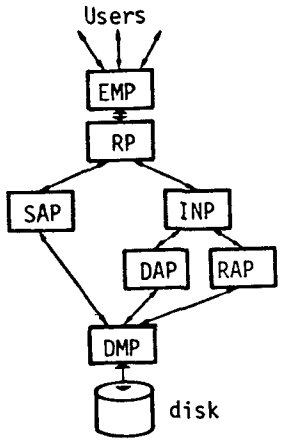

Fig. III.1. Software architecture

IV - ARCHITECTURE DEFINITION

The main problem in the architecture definition was to determine if one or several microprocessors are needed to execute efficiently the MAGE access method.

The objective is to get the best utilization of the disk which is the most expensive resource in the system. The processor must be fast enough to provide an optimal disk utilization and a maximal speed, so that the disk unit should practically not wait for requests. This also allows a queue of requests to the disk which is large enough to permit an optimization of disk head moves. For a user request, the average processing time must therefore be less than the average disk access time.

The average processing time and the average number of disk accesses, have been evaluated for a MC 6800 [MOT76] microprocessor, from an analysis of the average relative frequency of each kind of requests, and an analysis of the processing of MAGE.

The average number of disk accesses for a request is of about 0.83 distributed as follows :
- structure : 0.1 access
- dictionary : 0.37 access
- data : 0.36 access.

The average processing time of a request is of about 13 milliseconds. It is interesting to notice that, of the 13ms, 4ms are spent in multiplications and division , and 4ms are spent in data transfers in memory.

The average time between two consecutive disk accesses for the CDC 9760 [CDC77] disk family is 38ms. If we use an optimization algorithm for the disk arm moving (SCAN algorithm [THE72]), it can be reduced to about 29ms for a queue of 5 requests.

For a single request, the average time of disk access is: 0.83 x 29 = 24 ms.

This value, compared to 13ms for the evaluation of the processing time shows that it is possible to use just one microprocessor such as MC 6800.

If we consider the decreasing cost of microprocessor, and the need of achieving an optimal utilization of disk and communication systems, we are led to distribute the processing between several microprocessors, one of them being connected to the communication system, and the others dedicated to the disk.

This would also improve the safety, since each processor can monitor the others.

In the next paragraph, we shall expose further details on the architecture of MAGE processor.

V - DESCRIPTION OF THE ARCHITECTURE

The processing will be distributed between 3 microprocessors (refer to section 3 for process definition):
- a processor P1 will execute EMP and RP,
- a processor P2 will execute INP, SAP, DAP, RAP,
- a processor DP (disk processor) will execute DMP.

We shall see later in section IV that DP will be closely a "file processor".

The main problem in using several cooperating microprocessors, is how to communicate. This is not just the problem of the microprocessors, it also exists for the communication to and from the disk and to and from the user. The solution is either to use a bus for exchanging messages, or to use a shared memory for common data. The first solution increases the software cost, and decreases the power gained by the existence of the second microprocessor. So, we prefered the second solution for the communication between the microprocessors, even though it increases the hardware complexity. For the communication between the disk and the microprocessors, we also used a shared memory, because of the high traffic density. Since we use a medium level language for the communication with the user, the traffic is lower and the solution of exchanging messages is the most economical.

The resulting architecture is presented in figure V.1. P1 is connected to the communication system through an interface and communicates with P2 asynchronously through a common memory (user context memory).

P2 also communicates with the disk processor through a FIFO memory and a direct memory access system(DMA).

Each processor P1 and P2 has its own program memory (ROM) and working memory (RAM).
Each processor P1, P2 and DP can be a single MC 6800 microprocessor, but it can also be a more recent microprocessor (like MC 6809 or 68000) in order to allow simpler addressing and sharing of memories, to be able to program in high level languages such as PASCAL, and to adapt with the evolutive new technologies of mass memories.

V.1. Communication system interface

The communication interface will be designed in a modular way, in order to allow its adaptation to the different communication characteristics, with specifics communication protocols, hardware configurations, and communication speed.
A specific hardware and software communication interface may be used, depending upon the communication system.

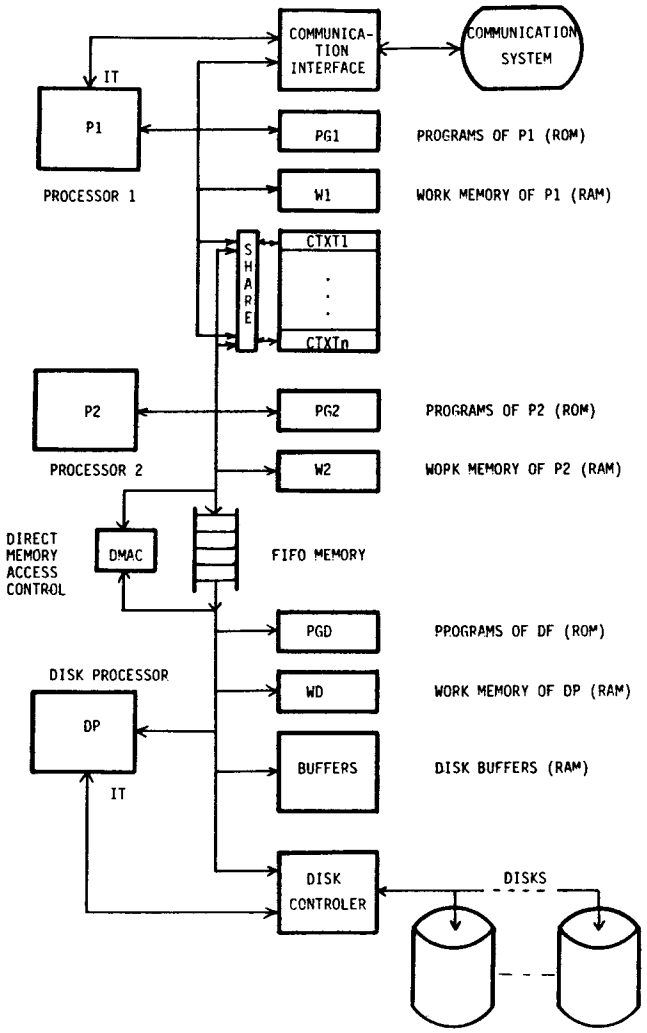

Fig.V.1. Hardware architecture of MAGE

V.2.Communication between P1 & P2:the context memory

Every user working on the data base needs one or several contexts. These contexts contain all the information needed for the processing of the user's requests. The contexts are in a memory which is shared by P1 and P2 and constitute the communication and synchronization means. This memory is called "context memory". There are 64 contexts (fig. V.2) in the memory, each having 512 bytes (32 Kbytes for the entire context memory, in the maximum version). Each context contains a byte which indicates its status which can be either :

. not affected,
. idle (no request to be processed in this context),
. waiting for processing by P1,
. waiting for processing by P2,
. waiting for a disk access,
. communicating with the user.

When a context by a processor requires the intervention of the other processor, it is sufficient to put

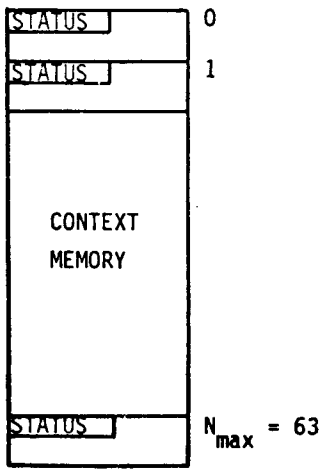

Fig. V.2. Context memory

the context in 'wait state', it will then be processed by the second processor. When a processor finishes the processing of a context, it scans the context memory until it finds a context waiting to be processed. Then, it calls the program corresponding to the code contained in the context. At any given time, a context may be processed by only one processor, but both processors may access "simultaneously" a context in order to test its status.

The hardware solution to the problem of sharing the context memory, will essentially depend on the average access rate to the context memory. The evaluation of the processing needs, gives a probable conflict rate of 1.4% that both processors will try to access simultaneously the context memory during a cycle. With this rate, the problem of the simultaneous access by both processors is solved by suspending one processor during the access of the other one. This is performed by stretching out one of the clock phases of the delayed microprocessor.

VI - DISK PROCESSOR

The functions of the disk processor will be divided in several parts :

- the control of the disk itself,
- the management of the waiting queue of the requests, allowing an optimization of the head positioning,
- the conversion of the disk addresses. This allows the disk processor to use the mass memory irrespective of its configuration, and to use logic addresses of the data. From this point of view,

the disk processor is seen to be like a "file processor".

VI.1. Functional definition of the DP

The DP alloww the management of a number of files, whatever may be their structure, dictionary or data file of a data base, or any file external to the data base.
It performs the operation such as :
- disk space management,
- address conversion,
- requests management,
- privacy and security of the data.

a/ file_operations : The DP allows creation or deletion of a file, giving to it a name that can be up to 8 bytes.

An existing file must be opened before any access. This operation allows access to a file having an "open number", and not the whole name of the file. This allows a simpler processing of data access operations. The access rights of the user processor (DBP, or any processor in the distributed system) are controlled at the OPEN operation.

b/ data_access_operations : Data having the following parameters can be accessed :
- the "open number" of the file,
- the number of the sector inside of the file,
- the offset of the data in the sector,
- the length of the data requested.

The DP converts this information to physical disk addresses, and puts the requests into a disk waiting queue, ordered by cylinder number, to provide an optimization of the disk head positioning (SCAN algorithm). The operations provided are :
- to read data (a field in a given sector),
- to write a whole sector,
- to modify data. In this operation, the DP read the necessary sectors, modify them with the data provided by the user (the DBP), and rewrite the sector on the next rotation.

In addition, an operation is provided to allow an associative access to the data. The data can be accessed providing :
- the "open number" of the file,
- the number of the first sector where the search must be performed,
- the number of sectors where the search must be performed,
- the associative key to search,
- the offset of the first key to compare, in the first sector,
- the length of the data requested, which is also the repetition interval for the search.

The search is performed until there is a match between the provided key and the key on the disk. Then the information following that particular key is returned to the user processor. If no match is found at the end of specified number of sectors, the operation is stopped and a "no match" condition is returned to the user.

The key have a fixed length of 4 bytes, which is enough to significantly upgrade the performances of the MAGE access method.

VI.2. Communication with DBP

The hardware communication can be seen in fig.V.1. It is composed of a FIFO memory which allows P2 to send the addresses of requests to DP. The requests and the data are exchanged through DMAC circuits which allows a fast transfer between the memory of P2 and the memory of DP. The access to the memories by the DMAC (direct memory access controller) circuits can be allowed by halting the two processors during the transfer.

To send a request to DP, P2 must put it into its own memory, and write the address in the FIFO. DP read the FIFO and get the request from the memory of P2 through the DMAC. The exchanges of data are also executed through the DMA , and the end of the operation is signaled to P2 by sending to it a report at the address of the request.

VI.3. Disk control

The disk used (CDC 9760) have a high transfer rate, up to 10 Mbits/s, thus it is not yet possible to use a MOS microprocessor as a direct disk controller.The data transfer task is left to TTL and ECL circuits ; a bit-slice component is used for controlling the transfer mechanism and a microprocessor (MC 6800) is used to execute functions such as optimization of head position, address conversion, file request management and other tasks (requiring processing).

So the disk processor has a hierarchical structure of three levels in the hardware (fig.VI.1). The highest level is represented by the microprocessor, the intermediate level being the bit slices, and the lowest level is hardwired. The characteristics of these three levels are :

- highest level: slow speed, powerful language,
- intermediate level : medium speed, microprogrammed language,
- lowest level : high speed, hardware commands.

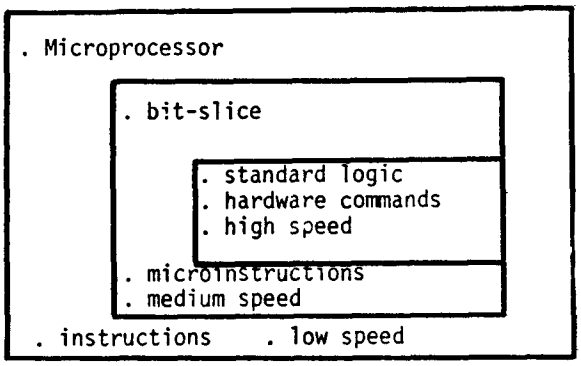

Fig.VI.1. Different levels in the disk processor architecture

The bit-slices (AMD 2900 family) may support, as an upper limit, rates in the range of 10MHz. This allows the bit slice to be part of the controller functions, but it is not possible to execute all the functions because the disk rate is also 10MHz and it is necessary to execute more than one bit slice instruction between two events. The solution is to have the data transfer mechanism realized in both standard logic and bit slices.

The main functions executed by the standard logic are serialization/deserialization on 8 bits and the CRC(Cyclic Redundancy Check). These circuits will bring the frequency of the events (transfer of a byte) to 1.2MHz. At this rate, the bit slices may work more easily and they will control the serial to parallel or parallel to serial conversion during the transfer of the data to and from memory, research of patterns, checking of the CRC, synchronization detection, etc... All these commands and procedures of the bit slices will perform four disk functions : data read, write, and verify, and pattern research.

The bit slice microprograms have the key role in the data transfer control. One of the interesting point of its design was that they are synchronized with the active disk unit clock. This allows the bits of the microinstructions to activate directly a major part of the logic circuits. Some more sophisticated functions, such as comparisons, test and counting, are executed by the ALU Bit slice unit. The whole bit slice control is initialized by the microprocessor through the transfer of load parameters and by signal activations.

The microprocessor (fig. VI.2) will control, at a higher level, all these mechenisms. It activates the bit slices, executes slow processing such as address conversion, head positioning, request management, transfer initialization. In association with some hardware circuits, the microprocessor will also execute the fourth function of the disk : the seek. This research is executed by the bit slices directly at the rate of the disk.

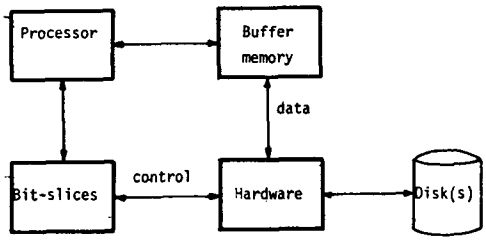

Fig.VI.2. Disk processor

The sharing of the memory buffers between the microprocessor and the disks, present some problems. In fact, the disk may transfer approximately one byte every microsecond. So, the buffer cannot be accessed by the microprocessor during the transfer. To solve this problem we use the halt signal of the microprocessor. Simply suspending the processor activity during the transfer (about 2% of its time), seems to be the most economical way to perform the sharing of the buffers between the microprocessors and the disk unity. This approach will not introduce any inconvenience, since the origin of the microprocessor interrupts is the disk itself. So, if it is suspended by a disk transfer, it may not be interrupted by any other event other the end of this transfer. Another advantage of this technique, is that the halt and restart of the microprocessor done fastly as they do not require to save internal registers.

VII - CONCLUSIONS AND PERSPECTIVES

Hardware and software implementation of the project MAGE is being carried out. A first prototype of the Disk Processor is ready an is used in test and performance measurements.

We think that the kind of architecture used in project MAGE is well adapted, having sufficient performance, for small "turn key" systems and for the future technological evolution. In the future industrial versions we intend to use more powerfull microprocessors. The disk processor will be designed as a file management processor.

Advances in device technology of processors, semiconductor random-access memories and all-electronic bulk memories will change the software and hardware trade-offs. The aims will no longer be to share processing power, but rather to share information.

In the MAGE project, the advances in technology, especially in semi-conductor memories, may lead us to put the "structural" information (from the data access method) physically closer to the first processor (P1). This clustering technique will eliminate a number of costly disk accesses and increase system performances by about 12%.

In a second step, improvements in all-electronic bulk memories will allow the "dictionary" information to be stored close to the processor P1, and so the mapping information will be fastened. With these improvements, the needs for communication with the disk will decrease by about 50% and the performance will increase. With this design process the data base becomes more and more clustered while disk accesses are reduced.

In the near future, the information used to access the data will probably be close to the processors and only the users data bulk will be continuing to stay on the moving head disks.

REFERENCES

[HSI79] - DAVID K.HSIAO, "Data base machines are coming" IEEE Computer, march 79, pp.7/9

[AND75] - G.A.ANDERSON & E.D.JENSEN, "Computer interconnection structures: taxonomy characteristics and examples", Computer Surveys, voL7,n°4, december 1975, pp. 197/213.

[ANC78] - F.ANCEAU, "Systèmes fonctionnellement distribués", AFCET Congress, Paris, november 1978, pp. 85/94.

[NAV79] - Ph.NAVAUX, G.BERGER SABBATEL, "A Data Base Processor: hardware design and implementation based on MAGE", EUROMICRO Symposium 79, august 79, pp. 91/98.

[BER78] - G.BERGER SABBATEL, "Etude fonctionnelle d'un processeur base de données hiérarchiques", Thesis, University of Grenoble, France, june 1978.

[MAR78] - F.J.MARYANSKI, "A survey of developments in distributed data base management systems", Computer 11, 2, février 1978, pp.28/38.

[COP73] - G.P.COPELAND, G.J.LIPOVSKI, S.Y.SU "The architecture of CASSM: a cellular system for non-numeric processing", Proc. of the 1st An.Symp. on Comp. Arch., december 1973, pp. 121/128.

[OZK75] - E.A.OZKARAHAN, S.A.SCHUSTER; K.C.SMITH "RAP, an associative processor for data base management", AFIPS NCC 75, vol.44, pp. 379/387.

[LIN76] - C.S.LIN, DCP SMITH, J.M.SMITH, "The design of a rotating associative memory for relational data base applications", ACM Trans. on DB syst., vol.1, n°1, march 76, pp. 53/65.

[NAV79] - Ph.NAVAUX "Processeur base de données MAGE: aspect matériel", Thesis, University of Grenoble, France, november 1979.

[AND76] - D.R.ANDERSON, "Data base processor technology", Proc. AFIPS National computer conference 1976.

[ABR70] - J.R.ABRIAL & al., "Projet SOCRATE: spécifications générales", University of Grenoble, august 1970.

[MOT76] - MOTOROLA, M 6800 Microcomputer system design data, 1976.

[CDC77] - Control Data Corp., Product specification for the 9760 drive family, 1977.

[THE72] - T.S.THEOREY, T.B.PINKERTON, "A comparative analysis of disk scheduling policies", CACM, vol.15, n°3, 1972.