ISSN 0254-2757

QI QUAESTIONES INFORMATICAE

Volume 4 Number 3 October 1986

An official publication of the Computer Society of South Africa and of the South African Institute of Computer Scientists

'n Amptelike tydskrif van die Rekenaarvereniging van Suid-Africa en van die Suid-Afrikaanse Instituut van Rekenaarwetenskaplikes

QUÆSTIONES INFORMATICÆ

An official publication of the Computer Society of South Africa and of the South African Institute of Computer Scientists

'n Amptelike tydskrif van die Rekenaarvereniging van Suid-Africa en van die Suid-Afrikaanse Instituut van Rekenaarwetenskaplikes

Subscriptions

Annual subscriptions are as follows:

	SA	US	UK
Individuals	R10	$ 7	£ 5
Institutions	R15	$14	£10

Circulation and Production

Mr C.S.M. Mueller
Department of Computer Science
University of the Witwatersrand
1 Jan Smuts Avenue
Johannesburg, 2001

Quæstiones Informaticæ is prepared by the Computer Science Department of the University of the Witwatersrand and printed by Printed Matter, for the Computer Society of South Africa and the South African Institute of Computer Scientists.

IN PRAISE OF SOLID STATE DISCS

P.F.Ridler,
Dept. of Computing Science,
University of Zimbabwe.

Various ways of using random access memory to emulate a magnetic disc storage system are described and the advantages and disadvantages are enumerated. An algorithm for making use of a solid state disc is given which avoids the user having to take any action when the wanted file is on a magnetic disc or when the solid state disc becomes full.

1. WHAT IS A SOLID STATE DISC?

A solid state disc (s.s.d) is simply a section of random access memory which is treated as though it were a disc. It is assumed to have tracks and sectors, to hold directories and data, and have the same layout as any other disc permitted by the operating system in use. It does not, of course, have gaps between its sectors as with a magnetic disc and it is read by a simple data transfer operation which requires no commands to a disc controller chip. Its tracks are purely logical sub-divisions of what might otherwise be thought of as a linear array of data elements. The number of elements in a track may be related to the length of the physical section of memory which can be made available at any particular time.

2. WHY USE A SOLID STATE DISC?

Why should a computer system use a solid state disc? Well, in the olden days, the larger mainframes used to and it was referred to as 'fast backing store' or something similar. It was usually a relatively slow magnetic core store and was used because it was much faster than a disc or drum but not as expensive as the fastest cores which were available.

The price of high capacity semi-conductor chips is falling rapidly. The price per kilobyte has fallen spectacularly over the last few years as may be observed in the following table which is given for small quantities :

month	year	US$	chip size
DEC	1977	20.00	4k
DEC	1980	2.50	16k
DEC	1982	1.00	64k
DEC	1983	.50	64k
DEC	1984	.25	64k
DEC	1984	.95	256k
JUNE	1985	.20	256k
JUNE	1986	.10	256k

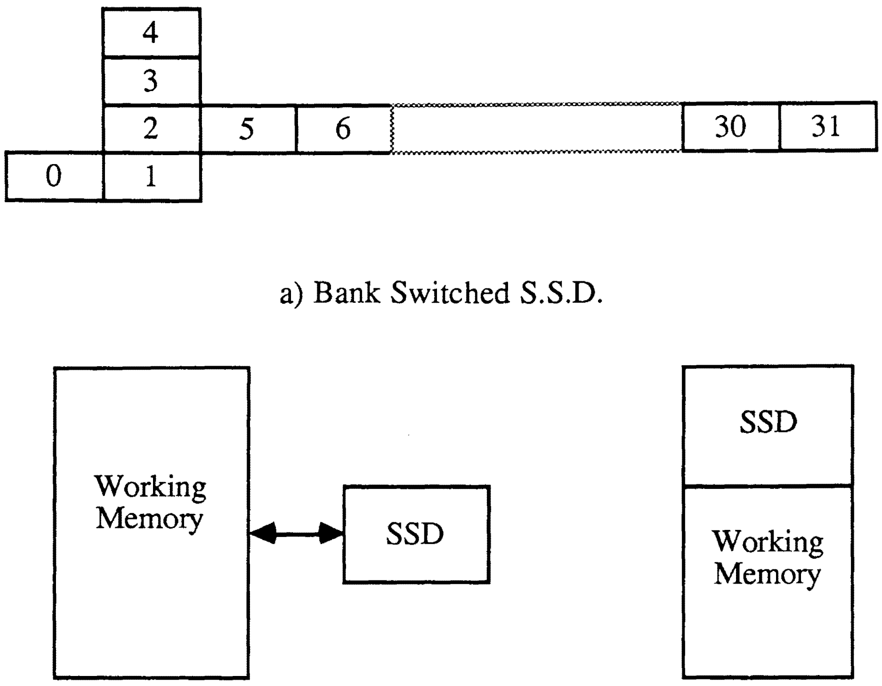
The overhead cost for a complete board holding thirty two of the chips referred to has remained roughly constant over the whole period.

This dramatic fall in the cost of memory has made it possible to use relaively high speed memory as backing store: a 1M-byte solid state disc now costs only about US$300.

3. HOW IS A SOLID STATE DISC IMPLEMENTED?

There are four different methods of implementing a solid state disc:

(i) banked switched memory
(ii) port accessed memory
(iii) segmented memory
(iv) directly addressed memory.



a) Bank Switched S.S.D.

b) Port Accessed S.S.D.

c) Memory Mapped S.S.D.

figure 1

3.1 Page switched memory

One way of implementing a s.s.d is to use a memory board which is divided into logical sections (pages), one or more of which can be made available at a time. One of the boards which the author has used is that referred to in [3]. This board is designed to work on the S100 bus, and consists of 256k of 4164 64k-byte chips. The memory controller chip is a 74S409 (equivalent to a DP8409) which contains the necessary circuitry to provide refreshing for the dynamic memory chips. This controller also has 18 address lines which enable it to access the whole 256k bytes on the board. However, as the Z80 processor with which it is used has only 16 address lines, the whole memory cannot be addressed directly.

To enable the Z80 to address the entire 256k, the two most significant bits of the Z80 address lines are not used to address the memory directly but are translated to a four bit address via registers which are written through I/O ports.

The result of all this is that any four of sixteen 16k memory pages are available at any one time, the particular pages enabled being detemined by the address translation registers.

The use of the board as a s.s.d is illustrated in fig. 1a. Memory pages 1,2,3 and 4 are used as a 64k main memory. Page 0 is used to hold the major part of the operating system so as to free space in the main memory and pages 5 through 16 are used as s.s.d.

The s.s.d is logically composed of tracks each having 128 128-byte data sectors. To access a particular data sector, the memory page containing the track and sector is enabled in place of one of pages 2 or 3 of the main memory bank taking care that the destination is still available. The 128-byte sector is then transferred to its destination. The transfer process takes about 5 µs per byte instead of 32 µs per byte from a flexible disc, and is about the same speed as for a Winchester drive. However if the actual disc drive head is not at the wanted track then it must be moved to the wanted track and this movement takes a time which is of the order of milli-seconds and is approximately the same for flexible or hard discs. Of course, the s.s.d is truly random access and no extra time is taken to move between logical tracks.

3.2 Port Accessed Memory

The second method of implementing a s.s.d which uses a separate memory array is that which accesses the memory through I/O ports [2]. Separate ports are used to set the track and sector and the data is read or written through the data port (fig 1b). This method handles data at the same rate as the previous one but is marginally easier to implement as far as software is concerned. The main disadvantage of the method is that a specialised memory board is required while in the paged system ordinary memory boards may be used.

3.3 Segmented Memory

A number of the available 16-bit microprocessor chips [4,5] have memory management circuitry on board. Addressing is done by adding an offset to a segment address. A segment register can thus be used to indicate the logical track of the s.s.d while a simple mapping of the sector number will give the required offset (fig 1c).

3.4 Directory Addressed Memory

The 68000 processor has a linear addressing system. All memory access is done by simply emitting a 32-bit address on the bus. It is relatively simple to devise a method of implementing a s.s.d using memory in an address range which will not be used by other programs. The s.s.d layout is relatively unimportant and any simple track and sector mapping can be used (fig 1c).

4. PRO'S AND CON'S OF SOLID STATE DISCS

There is little differences between the four methods of implementing a s.s.d which have been outlined above, and the decision as to which method to use will depend on other factors in the design of the machine.

4.1 Speed

The major advantage of a s.s.d is in its speed. It is many times faster than a flexible disc and considerably faster than a hard disc.

4.2 Volatility

All these methods of implementing a s.s.d are potentially volatile. If the power fails, or the operator turns the machine off before backing up data to magnetic discs, information will be lost. It is possible to use a battery which comes into play when the power supply feeding the memory falls below a certain voltage or an uninterruptable supply may be used for the whole machine. In a typical urban situation power supply failures are rare and the extra expense may be hard to justify.

A more subtle situation arises from the use by the electricity supply authorities of "ripple relay" switching. A voltage in the range 500-1500 Hz is superposed on the mains voltage to turn on or off water heating and other equipment at peak load times and this may interfere with a computer. The author has had several cases where data in a s.s.d has become corrupted at about 5 a.m. and surmises that this is due to the ripple tone creeping through the power supply to the bus; better power suppy design would probably eliminate the trouble.

4.3 Cost

The cost of a s.s.d may be as low as $300 per megabyte at the present time. The cost of a 5" flexible disc drive is around $500 including its controller. It is not possible to evaluate the cost/benefit ratio for a general case because the job mix affects it intimately, and estimates of the relative values of time and cash vary enormously.

The author has been using two different s.s.d's for a period of a year and would certainly not be without one in the future. Programming is now limited by "thinking time", as it should be, rather than by machine time.

4.4 Back-up

Periodically it is necessary to write files held in the s.s.d to some more permanent medium. Files such as editors and other utilities need never be saved, for they contain no new information, but program and data files may well have been modified and must be preserved. These files may be saved as they are created but this slows down the process to a speed which is limited by the magnetic disc drive. At the very slight risk of losing data through power failure, files which are to be saved should only be archived at intervals and on an incremental basis. At the command of the programmer, or at close down, those files which have been modified since the last archiving process are all written back to the discs from which they came [1]. This saves only the new data without slowing down the process excessively by unnecessary writing to slow backing store.

5. AUTOMATIC ROLL-IN OF FILES

When the system is switched on initially, the solid state disc will be empty. It must then have its directories initialised and those files which it is anticipated will be used brought into the memory space of the solid state disc. This process is necessarily slow, the time depending on the transfer rate and the track-to-track stepping time of the disc drive. Typically, to bring in a menu system, an editor and assembler and a few utility programs totalling 50k bytes will take 100 seconds.

As other files are required they must be transferred from the magnetic disc to the s.s.d; normally, this is done by means of a file transfer program.

Again, as the s.s.d becomes full it is necessary to transfer files from it back to a magnetic disc. Those files which are the least recently used will probably be those least needed and can either be erased if they have not changed or transferred to backing store if they hold new information.

These procedures may be carried out automatically, files being brought into the s.s.d on demand and transferred back to magnetic discs as space in needed in the s.s.d for new files. A flow diagram of the process is shown in fig.2.

6. OPERATING SYSTEM REQUIREMENTS

The extra requirements placed on the operating system by the use of a solid state disc are fairly small. The disc read and write routines are quite small and the backup function would normally be carried out by a utility program. Automatic file roll-in and roll-out is a major addition, but can be dispensed with if so desired.

6.1 Solid State Disc Handling Routines

The read and write routines for accessing data on s.s.d's are almost trivial; much simpler than those for mechanical disc drives. Obviously they must be made part of the operating system and must be made compatible with the disc formats which the operating system can handle.

Otherwise, s.s.d's pose no problems in this area.

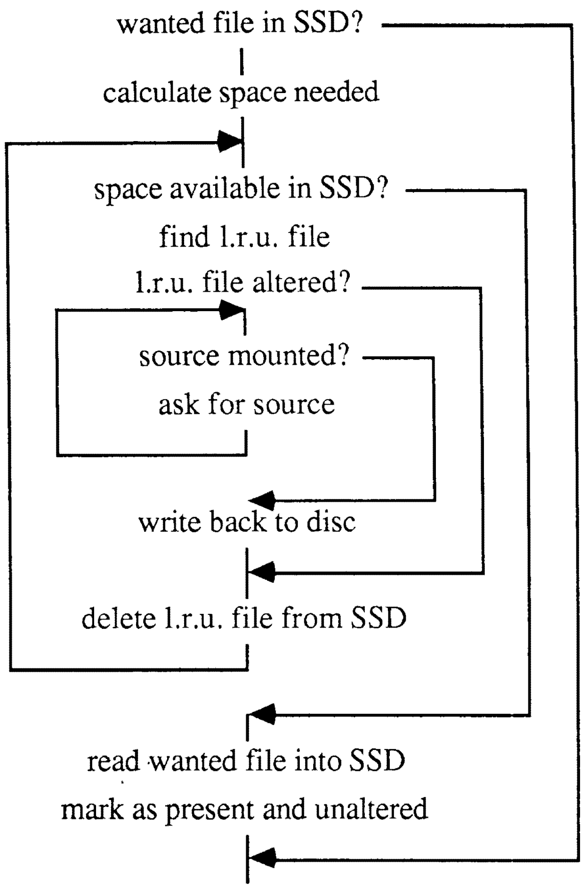


figure 2

6.2 File Roll-In/Roll-Out Routine

Automatic roll-in of wanted files and roll-out of redundant files in the s.s.d seems to be a highly desirable feature of an operating system. However, with a fair-sized (256k-1Mbyte) s.s.d such a system can be dispensed with in a typical system programming environment. The author presently uses a 512k byte s.s.d and is working on a compiler the source program of which occupies 120k and the source of the runtime package another 50k bytes.

6.3 Archiving Routines

As mentioned in 4.4, back-up of files must be done at intervals. Programs to do this are available, but it seems desirable that the operating system, in its directory area, keep track of those files which have been written to so that the archiving utility can keep track of random access files.

It also seems desirable that the operating system have the ability to write the file currently in use back to magnetic discs without using the archiving utility. This would enable more frequent back-up of a workfile without distracting attention from the job in hand, the use of the archive utility then being only necessary at power down.

REFERENCES

1 Fiedler.D., "QBAX": an incremental backup utility. (review), *MicroSystems*, 4 No 10, Oct 83, Amanuensis

Inc., RD1 Box 236 Grindstone, PA 15442, USA.
2 4931 512k S-100 Semidisk handbook., Semidisk Systems, Box GG, Beaverton, OR 97075, USA.
3 California Digital CT256-I handbook., California Digital, 4738 156th St., Lawndale, CA 90260, USA.
4 8086 family user's manual. Intel Corporation.
5 Z8000 manual., Zilog Corporation.
6 68000 manual., Motorola Corporation.

Second Announcement and Call for Papers

Interdisciplinary Conference on

MATHEMATICAL LOGIC AND RELATED SUBJECTS

Durban, 6-10 July 1987.

This conference is intended to bring together logicians and those who use logic in other disciplines. The organizers interpret "related subjects" in a wide sense in order for the conference to be of interest to mathematicians, computer scientists, linguists, philosophers and in general to all those doing research in areas overlapping logic.

The conference is organized by the Departments of Mathematics of the University of Cape Town and the Rand Afrikaans University; it is sponsored by the Department of Mathematics of the University of Natal (Durban), and will be presented as the annual Hanno Rund Colloquium of the Deaprtment. Accommodation will be available in a University residence as well as in a moderately-priced hotel. The registration fee for the conference is R25,00, but this will be waived for students. In special cases limited financial assistance may be available to intending participants.

The format envisaged for the conference is that survey lectures will be presented by invited speakers, with other participants contributing shorter research papers. The survey lectures will give different perspectives on logic and its applications, and should be accessible to all participants. Several speakers from abroad have already accepted invitations to present survey lectures. A subcommittee has been formed to compile a program for the subject are *Logic and Informatics*, of interest especially to computer scientists and sponsored by the South African Institute of Computer Scientists. Enquiries regarding this part of the conference should be directed to one of its organizers: Proff. S.W. Postma and N.C.K. Phillips, Department of Computer Science, University of Natal, Pietermaritzburg. The *Proceeding* of the conference will appear as a special edition of the *South African Journal of Philosophy*. All invited and contributed papers may be submitted for publication at the time of the conference. All submitted papers will be subject to the normal refereeing procedures.

Research papers of approximately 30 minutes duration on any aspect of logic are hereby solicited.

There is as yet no deadline for final commitment to take part in the conference, but those who are interested are asked to inform the organizers of their decision as soon as it is taken. In particular, early commitment to contribute a paper will be much appreciated, even if full details cannot be given at present.

The organizers are:

- Prof. C. Brink, Department of Mathematics, University of Cape Town, Rondebosch 7700.
  Telephone (021) 69-8531; Telex 57-2208 SA. (from 1 January 1987.)
- Prof. J. Heidema, Department of Mathematics, Rand Afrikaans University, PO Box 524, Johannesburg 2000, Telephone (011) 726-5000; Telex 42-4526 SA.

NOTES FOR CONTRIBUTORS

The purpose of the journal will be to publish original papers in any field of computing. Papers submitted may be research articles, review artilces and exploratory articles of general interest to readers of the journal. The preferred languages of the journal will be the congress languages of IFIP although papers in other languages will not be precluded.

Manuscripts should be submitted in triplicate to:

Prof. G. Wiechers
INFOPLAN
Private Bag 3002
Monument Park 0106
South Africa

Form of manuscript

Manuscripts should be in double-space typing on one side only of sheets of A4 size with wide margins. Manuscripts produced using the Apple Macintosh will be welcomed. Authors should write concisely.

The first page should include the article title (which should be brief), the author's name and affiliation and address. Each paper must be accompanied by an abstract less than 200 words which will be printed at the beginning of the paper, together with an appropriate key word list and a list of relevant Computing Review categories.

Tables and figures

Tables and figures should not be included in the text, although tables and figures should be referred to in the printed text. Tables should be typed on separate sheets and should be numbered consecutively and titled.

Figures should also be supplied on separate sheets, and each should be clearly identified on the back in pencil and the authors name and figure number. Original line drawings (not photocopies) should be submitted and should include all the relevant details. Drawings etc., should be submitted and should include all relevant details. Photographs as illustrations should be avoided if possible. If this cannot be avoided, glossy bromide prints are required.

Symbols

Mathematical and other symbols may be either handwritten or typewritten. Greek letters and unusual symbols should be identified in the margin. Distinction should be made between capital and lower case letters; between the letter O and zero; between the letter I, the number one and prime; between K and kappa.

References

References should be listed at the end of the manuscript in alphabetic order of the author's name, and cited in the text in square brackets. Journal references should be arranged thus:

1. Ashcroft E. and Manna Z., The Translation of 'GOTO' Programs to 'WHILE' programs., *Proceedings of IFIP Congress 71*, North-Holland, Amsterdam, 250-255, 1972.
2. Bohm C. and Jacopini G., Flow Diagrams, Turing Machines and Languages with only Two Formation Rules., *Comm. ACM,* **9**, 366-371, 1966.
3. Ginsburg S., Mathematical Theory of Context-free Languages, McGraw Hill, NewYork, 1966.

Proofs and reprints

Proofs will be sent to the author to ensure that the papers have been correctly typeset and *not* for the addition of new material or major amendment to the texts. Excessive alterations may be disallowed. Corrected proofs must be returned to the production manager within three days to minimize the risk of the author's contribution having to be held over to a later issue.

Only orginal papers will be accepted, and copyright in published papers will be vested in the publisher.

Letters

A section of "Letters to the Editor" (each limited to about 500 words) will provide a forum for discussion of recent problems.