UNIVERSITY OF ADELAIDE
11/8/82

"VIRTUAL STORAGE IMPLEMENTATION ON A MICROCOMPUTER"

CHRISTOPHER SAU-SHAN YAM, B.E.

BEING A THESIS SUBMITTED

FOR THE

DEGREE OF MASTER OF ENGINEERING SCIENCE

IN THE

DEPARTMENT OF ELECTRICAL ENGINEERING

THE UNIVERSITY OF ADELAIDE

JULY 1980

This thesis embodies the results of supervised project work making up two-thirds of the work for the degree.

DECLARATION

This thesis contains no material which has been accepted for the award of any other degree or diploma in any University, and to the best of my knowledge and belief it contains no material previously published or written by another person, except where due reference is made in the text of the thesis.

C. S. Yam

ACKNOWLEDGEMENT

The author wishes to acknowledge the contributions of those who have, directly or indirectly, assisted in this research and preparation of this thesis. The author is pleased to thank Dr. D.A. Pucknell, the project supervisor, for his invaluable guidance and encouragement.

The author likes to thank the Radio Research Board of Australia for providing financial assistance for this project.

The author is grateful to his colleagues at the Control Systems Section, The Electricity Trust of South Australia for their helpful comments, in particular to Mr. A.P. Williamson for proofreading the manuscript.

Finally the author wishes to thank his wife for her encouragement throughout the project.

SUMMARY

Virtual storage is a hardware memory management technique which is gaining popularity amongst modern mainframe computers and top of the range minicomputers. It offers huge memory resources at lower cost in comparison to solid state memories of compatible size. The associated benefits are reduced component counts and lower power requirement. The trade off for the above mentioned advantages is an increase in the memory average access time. However this increase in the average access time does not appear to be a major handicap for many computer applications.

With the widespread acceptance of microcomputer systems these days, more sophisticated and increasingly more complex programmes are being run on microcomputers. Demands are placed on the microcomputers for bigger memory recourses. Also the new generation microprocessors can now cater for an addressing space of many mega bytes. It is obvious that virtual storage technique is ideally suited for these new generation microprocessors since a large real memory implementation is impracticable and is out of question economically.

This project investigated the feasibility of adapting an existing IMP-16C microcomputer system into a virtual storage system, assuming that a microcomputer system is single user orientated. Some of the virtual storage techniques were reviewed, in particular those that have been studied with a computer simulation of a virtual storage system. A working virtual storage computer system was implemented on the National Semiconductor IMP-16C microcomputer. The design was based on the results of the simulation study. A virtual memory of 256 K

words of 16 bits was achieved. The cost to equip a system with an equivalent size real memory is about ten times the cost to manufacture the virtual storage controller. The average access time of the virtual storage computer system as implemented is an order higher than the conventional real memory system. Supplementary techniques and faster auxiliary storage can be employed to improve the average access time.

This project demonstrated that a virtual storage controller can be coupled to an existing microprocessor to provide virtual memory storage at about one tenth the cost to provide the equivalent real memory storage. The virtual storage controller can be divided quite readily into functional blocks. Each block is suitable for chip level integration with LSI or VLSI technology. Implementing a virtual storage system around an existing microprocessor would be a much simpler task when these functional chips are available.

Glossary And Notation

Address - A number used by the operating system and user software to identify a storage location.

Address Space - The set of all possible addresses available to a process. Virtual address space refers to the set of all possible virtual addresses. Physical address space refers to the set of all possible physical addresses sent out on the physical address bus.

Base Page - Locations within the address boundary of X'ϕϕϕϕ and X'ϕϕFF.

Demand Paging - A programme page is not brought into physical memory until it is actually needed. If there is insufficient physical memory, the least recently used page in the system is moved out of memory to make room for the needed page.

Hit Rate - The percentage of times the CPU requests data and that data are in main memory, therefore not requiring a transfer from auxiliary storage.

Page - A set of 512 contiguous byte locations used as the unit of memory mapping.

Page Fault - A condition when the CPU requests data and that data are not in the main memory yet. A page pull after the page fault will satisfy the CPU requirement.

Pulling a Page - Transferring a page of data from the auxiliary storage into main memory.

Pushing a Page - Transferring a page of data from the main memory back to the auxiliary storage.

Virtual Memory - The set of storage locations in physical memory and on disk that are referred to by virtual addresses. From the programmer's viewpoint, the secondary storage locations appear to be locations in physical memory.

Working Page Registers - Storage locations not part of the main memory containing the virtual page addresses that have been mapped into physical memory.

X'ϕϕEF - Hexadecimal number notation, equivalent to decimal 127.

CONTENTS

1. Introduction 1

1.1 Introductory Notes 1

2. Review of Virtual Storage Techniques 5

2.1 Concept of Virtual Storage 5

2.2 Mapping 9

2.3 Replacement Policies 12

2.4 Other Techniques 15

2.4.1 Pre-Paging 15

2.4.2 Prevention of a Push 16

2.4.3 Buffered Technique 17

2.4.4 Restructuring of Programmes 18

3. Computer Simulation Study on Virtual Storage Techniques 20

3.1 Introduction 20

3.2 Replacement Algorithms 20

3.3 Pre-Paging 22

3.4 Prevention of Pushes 23

3.5 Average Access Time of Virtual Storage Systems 24

4. VSC System For National Semiconductor IMP-16C Microcomputer 26

4.1 Introduction 26

4.2 Design Criteria 27

4.2.1 Mapping 27

4.2.2 Pre-Paging 28

4.2.3 Prevention of Pushes 29

4.3 The VSC Controller 32

4.3.1 The Working Page Registers 38

4.3.2 Page Fault And Physical Page Generators 40

4.3.3 The LRU Stack 43

4.3.4 VSC Controller Initialisation Circuit 45

4.4 Floppy Disk Interface Circuit 50

4.5 Memory Segmentation 55

5. Operating The IMP-16C VSC System 59

5.1 System Evaluation and Results 59

5.2 System Limitation 61

6. Conclusion 63

6.1 Discussion 63

6.2 Conclusion 65

References 68

Supplement I - IMP-16C Virtual Storage Computer User's Guide S-1

Appendix A - IMP-16C VSC Special Instruction Set A-1

Appendix B - IMP-16C VSC Controlware Flow Diagram And Listing B-1

Appendix C - Wiring Schedule And Part List C-1

Back Cover - Circuit Diagrams

UNIVERSITY OF ADELAIDE LIBRARY

1. Introduction

1.1 Introductory Notes

A digital computer is a complex programme-controlled logic device. As such, it requires storage area for both the controlling programmes and the data being processed. The requirements on the storage size for a particular job varies according to the amount of data being processed and the complexity of the programme. Nevertheless, an upper limit is always imposed on the storage size either by the addressing space of the computer, or by the availability of physical memory cells. This storage limitation is an acute problem; particularly with general purpose computers in a multi-user environment, although in general it does restrict the scope of a project envisaged by the intended single user.

There are several options currently available to solve this storage problem. The obvious one is to provide enough physical memory cells in the form of magnetic cores or solid state memories for the problem programme, or up to the addressing space limit of the computer. This solution is easy to implement but is also the most expensive one. It is estimated that the cost for 1 K words of 16 bit MOS memory including power supplies and assembly would cost $60 (Australian currency in 1st quarter of 1980). It other words it would cost approximately A$4000 to equip a 16 bit computer to its addressing limit of 128 K bytes (1 byte = 8 bits). The MOS memories being a volatile device would require some form of back up permanent storage such as magnetic disks or tapes. These back up storage devices would add further

cost to the overall system. This is why the majority of computers do not have main memories up to their address limits.

The generally accepted solution both in terms of economy and flexibility is to provide a reasonable main memory size, for example a quarter of the computer address space, and an auxiliary storage in the form of magnetic disks and/or tapes. The programme and the data are normally held in the auxiliary storage which usually has a capacity exceeding the address space of the computer. Overlay technique is employed traditionally to exchange memory words with external storage words whenever needed. Overlay enables efficient usage of the memory cells since more than one programme word can be assigned to a memory cell at different times. The overall cost of the system is much reduced on the basis of cost per word storage. However, this technique requires additional planning in programme lay out and places a coding burden on the programmer. This is a burden the programmers would be glad to relinquish. The ideal solution would be an automatic overlaying scheme providing economy and flexibility yet free of manual intervention.

Virtual storage is the technique developed to meet these requirements. It can be considered as a memory management scheme that permits the user to treat the main memory and auxiliary storage such as the disk drive as a homogenous unit. Information is transferred between the auxiliary storage and the main memory as needed for the processing of a programme. These transfers are transparent

to the user who refers to the information with an unvarying address - the virtual address - with no need to know its exact location in the main memory. The computer itself must keep track of the changing physical locations of information, and must also convert the user's longer virtual address to a shorter physical address for the smaller main memory.

Virtual storage technique has been applied to some of the modern mainframe and mini computers such as the IBM System/38 and the DEC VAX 11/780. Virtual address space of 2 gigabytes or over is available to the users in these machines. With the removal of the storage constraint in these computers, complex engineering investigations such as power system network studies or structural design problèms can be programmed in a more direct and simpler approach. The investigator or designer can devote all his effort to the engineering aspect rather than having to resort to programming tricks to conserve storage space; and this would produce a more efficient and economical programme to tackle the same problems. Complex programmes of this nature which are traditionally run on mainframe computers can now be brought down to the mini or microcomputer level if virtual addressing technique is extended to these small computers. As was pointed out above some form of permanent storage such as magnetic disks are required to back-up the volatile MOS memories generally used nowadays. This auxiliary permanent storage can be considered a necessary part of a small computer system. Instead of linearly increasing the main memory by the addition of more physical memories, there are

great economical and operational incentives to develop the virtual addressing technique in small computers so the auxiliary storage becomes a homogenous extension of the main memory. It has been estimated that a virtual storage controller for a 16 bit mini computer could be constructed for approximately A$900 using some of the techniques discussed in Chapter 2.

The average access time of a virtual storage computer system is largely dependent on the average access time of the auxiliary storage and the frequency of a page fault - hence the chosen replacement algorithm. A design based on the Least Recently Used (LRU) algorithm and page associative mapping techniques was implemented on a National Semiconductor IMP-16C microcomputer. The auxiliary storage chosen was the SMS FT ϕ4ϕϕ Floppy Disk System. The average access time of the complete Virtual Storage system is 43 u sec., which compares very favourably with the 50 - 100 m sec. access time of the disk unit. A technique is proposed and partly implemented to extend the virtual address space beyond the 128 K bytes without physical additions on the 16 bits address structure.

Chapter 3 analyses the merits offered by the various algorithms and techniques in virtual addressing. The rest of the thesis described the design and operation of the IMP-16C Virtual Storage Computer system. Proposals to improve the average access time and methods of extending the virtual address space are also discussed.

2. Review of Virtual Storage Techniques

2.1 Concept of Virtual Storage

The technique of organising the memory into pages facilitates the transfer of data in block transfer storage devices such as magnetic disks. It also provides a more efficient memory management scheme when dealing with a smaller virtual address space. The virtual address as seen by the programmer can be considered to be composed of two parts as shown in Fig. 2.1. These two parts are the virtual page address and the word address within the page. The virtual page portion of the virtual address is the portion that requires mapping and translation into the physical page. Once the correspondence between the virtual page and the physical page is established, the word address identifies the individual memory locations within a certain page. It is obvious from Fig. 2.1 that the programme memory resource can be much larger than the physical memory.

The correspondence between the virtual page address and the physical page address is provided by an appropriate mapping algorithm. In general a mapping table is required to facilitate the translation from the virtual page to the physical page. The virtual pages that are currently residing in the main memory are filed in the mapping table or Working Page Registers. Whenever a reference is made to the part of a programme that is not in the main memory, it is detected by the absence of its virtual page address in the Working Page Registers. The user programme is interrupted temporarily until the referenced page is brought into the main memory

from the auxiliary storage. When this interruption takes place, a Page Fault is said to have occurred. The detection of a page fault and the carrying out of programme or data exchange between the main memory and the auxiliary storage is performed automatically under the supervision of an intelligent controller. Of course before a newly referenced page is brought or pulled into the main memory, one of the pages already in the main memory has to be returned or pushed back into the external storage to make room for the incoming page. This replacement guideline greatly determines the overall efficiency of the Virtual Storage Computer (VSC) system. Ideally the page being removed should be a page no longer required in the future. However, there is an inherent uncertainty about the occurrence of page usages. In practice, a page is so chosen to minimise page faults and thus reduce the overall response time.

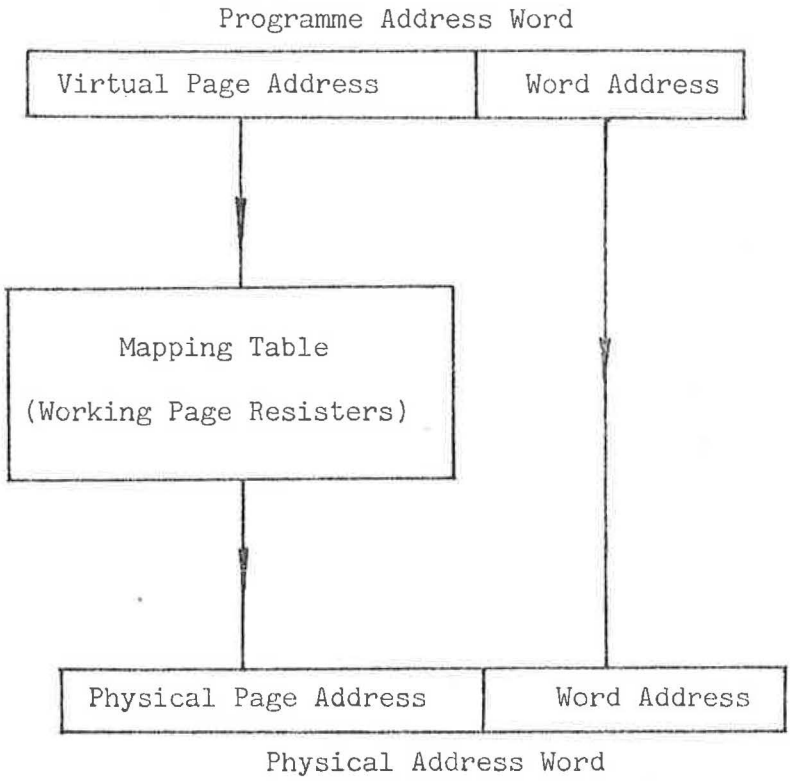


Fig. 2.1. Virtual Address Organisation

One of the parameters involved in the design of a virtual storage computer is the page size N. Conceptually the smallest page size would be limited to a single word. However, the access time on any external storage device consists mainly of the time required to locate the data; once located the time involved for a single word transfer is very close to that of a block transfer in comparison to the location time. Also it is common experience that once a word is used, the probability is high that adjacent words in the same block will soon be needed. The size of the Working Page Registers can also be reduced considerably by dealing with large blocks. On the other hand, the larger the block, the higher the proportion of its words that will not be required soon. Another parameter that requires consideration is the main memory size. A small main memory is inexpensive, but entails a high frequency of page faults. A more thorough discussion on the choice of optimum design parameters will be presented in Chapter 3. A simplified virtual storage computer system showing only the memory address structure is shown in Fig. 2.2.

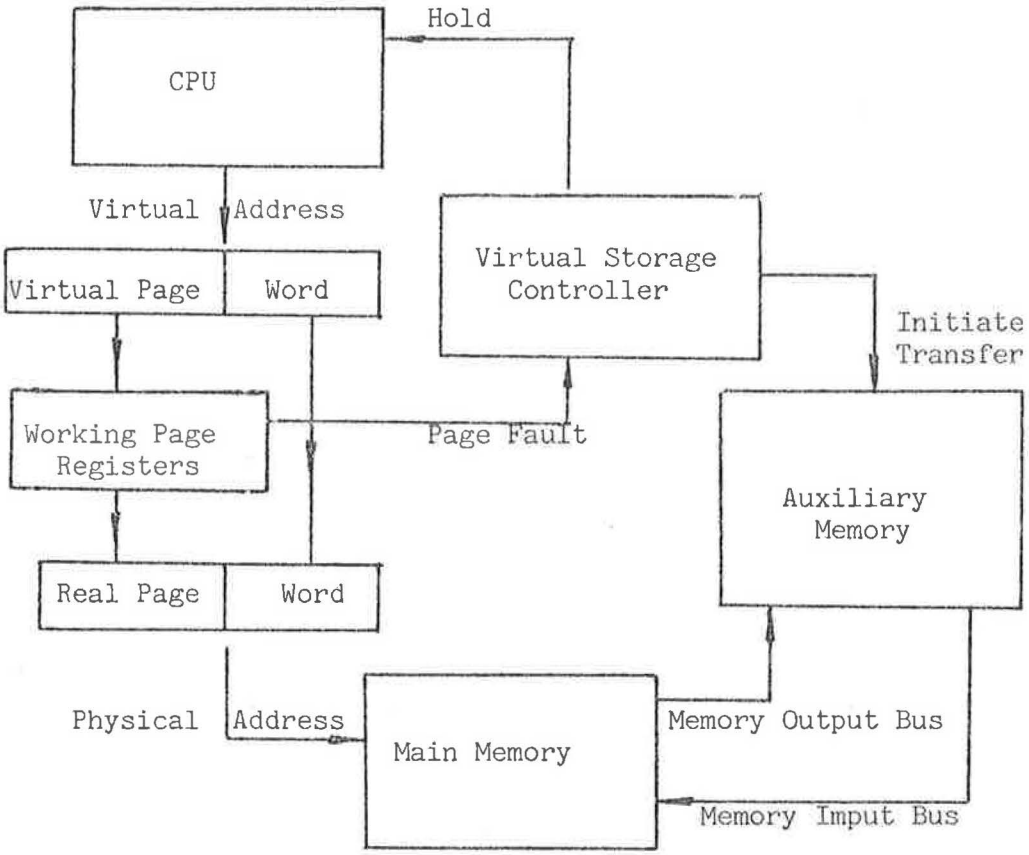


Fig. 2.2. Virtual Storage Computer System

2.2 Mapping

It has been observed that most programmes will run with more frequent references to certain portions of the programme than other portions.[8] Also many routines and sub-programmes are functionally independent, there is no need for them to be in the main memory at the same time. Traditional overlay technique makes good use of these observations and enables a more cost effective usage of the main memory by storing functionally independent subprogrammes in the same memory locations. These subprogrammes are brought in from the auxiliary storage as they are required. The drawback of the overlay technique is that it requires the programmer to identify these independent subprogrammes and define them as overlay segments. Virtual storage technique aims to provide an algorithm that can identify the working portions of the programme and retain them in the memory for as long as they are required. The working portions will be limited by the main memory size, and different working portions will share the same memory locations at different points of time. In general the virtual page address space of a virtual storage computer is much larger than the physical page address space. Thus a memory map is necessary to identify which portions of the virtual addresses are in the main memory and where they are located. Also some sort of mapping scheme is required to enable orderly allocation of the main memory. The two most common approaches in mapping are direct mapping and page associative mapping.

Direct Mapping

In this approach the auxiliary storage is divided into sectors of M pages. Each sector is equal in size to the

main memory. A direct correspondence is set up between each sector of the auxiliary memory and the main memory. Thus the second page of each sector can only be allocated to the second page of the main memory and so on. A pictorial representation of direct page mapping is shown in Fig. 2.3.

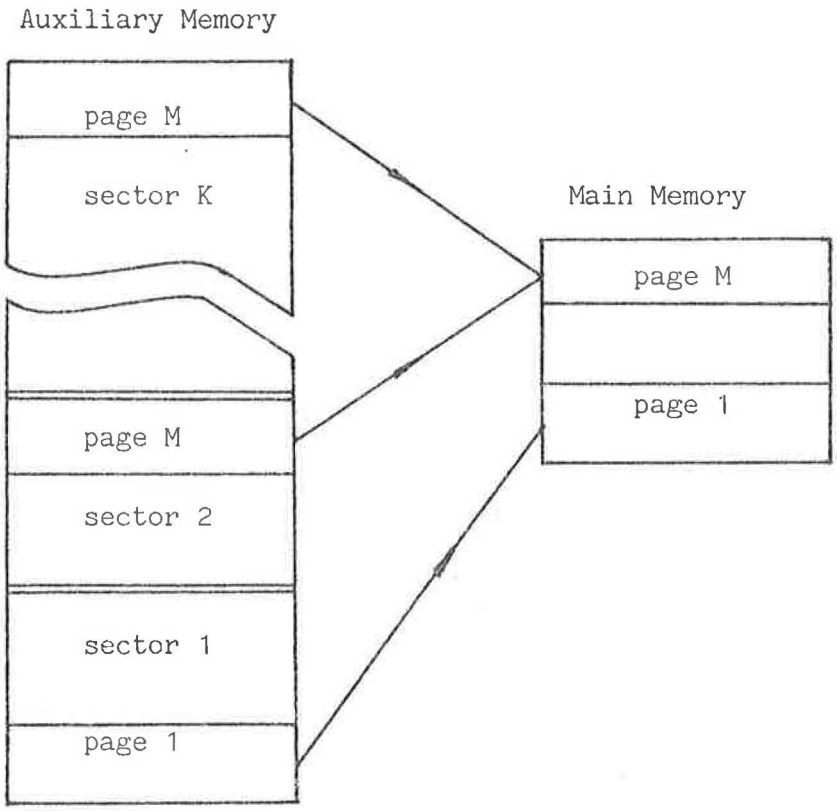


Fig. 2.3. Direct Page Mapping

The major drawback of this scheme is that corresponding pages in any sector of the auxiliary storage can only be allocated to the same page in the main memory. When corresponding pages from different sectors are referenced in succession, frequent page faults are encountered. A more flexible mapping scheme is the page associative mapping approach.

Page Associative Mapping

In this approach both the main memory and the auxiliary storage are divided into pages of N words. Any page in the auxiliary storage can be allotted to any page in the main memory. Since a simple one to one correspondence as in the case of direct mapping no longer exists, a replacement policy is required in associative mapping to select an outgoing page to make room for the incoming one. Also a mapping table is necessary in this case to relate the virtual page to the physical page. It is thus a more complex approach, however the flexibility of associative mapping is considered to be worthwhile. A diagram of page associative mapping is shown in Fig. 2.4.

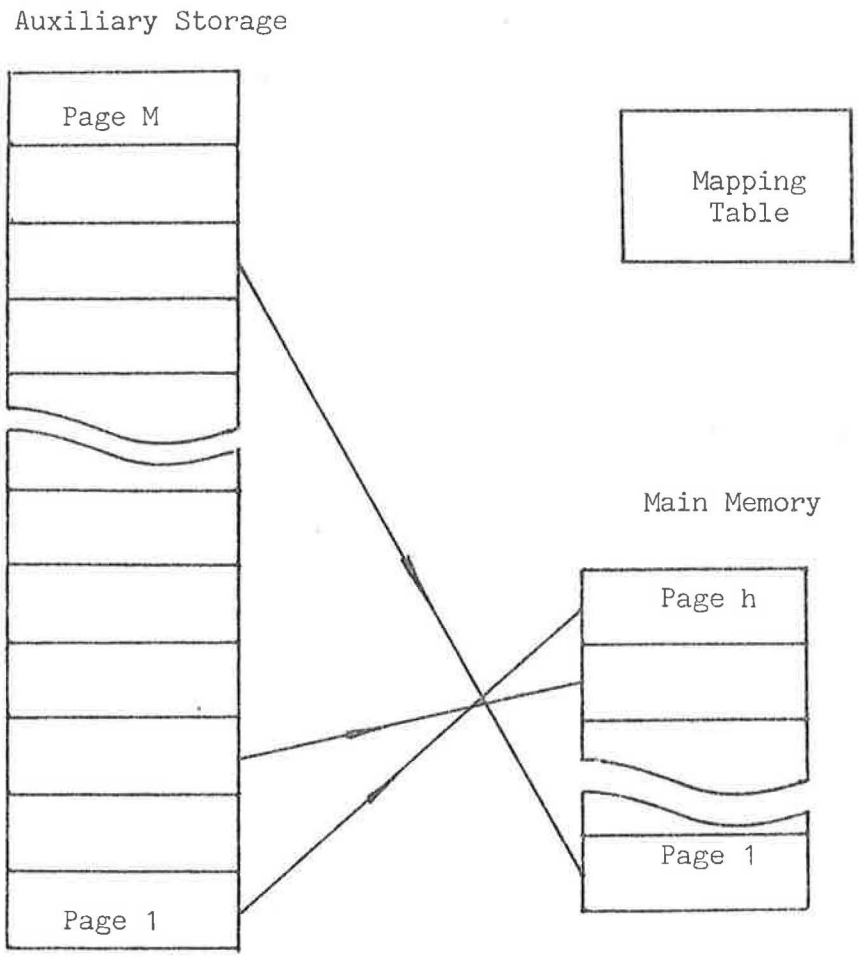


Fig. 2.4. Page Associative Mapping

2.3 Replacement Policies

A replacement policy is necessary only when the mapping between the auxiliary storage and the main memory is associative. Basically there are two main classes of replacement criterion,[3] static and dynamic, both having different assumptions about the behaviour of the running programmes. A brief description of each class of algorithm and some typical examples follow.

Static Replacement Algorithms

In this class of replacement algorithm, it is assumed that all pages are equally likely to be referenced at any time. The replacement criterion is not based on information about main memory usage. Under this class of algorithm comes the Random Algorithm (RAND) and the First In First Out Algorithm (FIFO).

The Random Algorithm (RAND) selects the outgoing page randomly, thus both the page fault frequency and the response time vary from one run to the next of the same programme. In the First In First Out Algorithm (FIFO), the longest serving page in the main memory is always the one to retire first when extra memory is required. This scheme has practical advantages over RAND in that a cyclic counter is easier to implement than a random number generator.

The assumption that all references are uniformally distributed over the range of all programme pages is certainly not true. This class of algorithm offers no practical application, thus sophisticated replacement decisions based on previous references are proposed to dynamically allocate memory according to previous usage.

Dynamic Replacement Algorithms

This class of algorithm dynamically reorders all pages in the memory according to the sequence of references to them. Pages are classified by the history of their most recent use in memory. When a replacement is necessary, the page to which reference has not been made for the longest time is replaced. It is reasoned that the fact that this page has not been required during the recent past indicates that it will not be referenced in the near future. This is a significant refinement on the static FIFO algorithm since now frequency of usage rather than the duration of stay in memory is the criterion. Typifying algorithms of this class are the Least Recently Used (LRU) algorithm and the Working Set (WS) algorithm.

The Least Recently Used (LRU) algorithm assigns preference of removal to the working pages on the basis of how frequently that page is referenced. The order of preference can be organised by a stack process. Whenever a page is being referenced, it is brought to the top of the stack while the rest of the pages are pushed down the stack one place. The bottom of the LRU stack is the preferred page to be removed when the occasion arises. The stack can be implemented easily with hardware shift registers or by assigning different priority values to the stack elements with an intelligent controller. However, with LRU in a multi-programming situation, a page in one process can compete with another page in the other process, thus causing inefficiency.

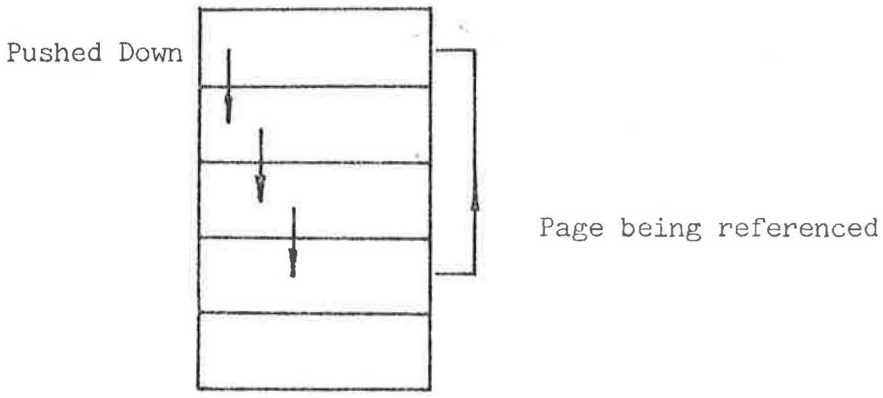


Fig. 2.5. LRU Stack

The Working Set (WS) algorithm is explained by P.J. Denning in his paper [1]. It is essentially a LRU algorithm with an added parameter of process time. In a WS algorithm, the page frames are freed whenever they are not referenced during the last τ m sec of the process time. The process time τ is an important parameter which determines the working set. The WS algorithm was shown to perform better than the LRU algorithm in a multi-programming environment because it can be applied individually to each process, thus avoiding the competing situation as in the case of the LRU algorithm. However it is a more complicated algorithm to implement. In a single user environment both the WS and the LRU algorithm are compatible.

Algorithms of this class provide reasonably good performance and are relatively easy to implement. Most of the virtual storage computers in production these days employ this class of algorithm. The Working Set algorithm is employed in the DEC VAX 11/780.

2.4 Other Techniques

Although proper choice of a replacement policy has the predominant influence on the efficiency of a VSC system, other techniques have been developed to enhance the overall performance of a virtual storage computer. These techniques aim to reduce the page transfer time by predictively bringing in more pages when a page fault occurs and by suppressing a push on a page whose content has not been modified.[6]

2.4.1 Pre-Paging

The argument for prepaging is that, statistically, when a page is being referenced, the next adjacent page is likely to be requested soon. If the prediction is correct, many page faults can be avoided, resulting in less programme interruptions. Since the time required to transfer two or three pages consecutively is about the same as to transfer one page, there is a considerable saving in total transfer time. A faulty prediction on the other hand will increase the number of page faults because each prediction failure would decrease the active memory size, causing further page faults.

The simplest form of prediction method is that when a page fault is generated by a reference to a page S, both the pages S and S + 1 are brought into the memory, if page S + 1 is not already in the main memory. This form of prediction technique is easy to implement and should be very effective if the flow of the programme is straight through without much branching. In reality this kind of consecutive reference rarely happens in programmes of significant size where memory space is of concern. When a branch occurs and

the prediction fails the effective memory size is reduced by one page thus increasing the probability of further page faults.

To prevent the continued reduction of effective memory size by successive prediction failures, the predicted page can be brought into a buffer instead of the main memory. If the prediction is successful the content of the buffer is then transferred to the main memory, which involves no external storage access time. If, however, the prediction fails it is no worse off then a normal VSC system without prediction and the buffer is overwritten by the next prediction. Two or more prediction page buffers can be incorporated to cater for programmes whose address space is growing in more than one direction as in the case of a matrix multiplication. This kind of prediction technique is termed One Block Lookahead (OBL). The average access time improves with each successful prediction, but OBL offers the benefit of not degrading the performance with faulty predictions. Naturally a more complex controller is required.

2.4.2 Prevention of a Push

Prepaging technique brings in the next adjacent pages of the page currently required at page fault time. The task of addressing these adjacent pages in the auxiliary storage is a simple matter and requires no additional access time. Depending on the technique chosen, at least one page in the main memory will be overwritten by the incoming pages. In the event that the content of the page to be overwritten has been modified during the course of programme execution,

this outgoing page has to be written back to the external storage for future reference. This means that two separate accesses on the external storage are required; one for the outgoing page and the other for the incoming page. In the case of pages containing programme instructions which are not altered, the pushing is unnecessary. As much as 50% reduction of the access time can be achieved at each page fault if these unnecessary pushes are prevented. One way to achieve this is to provide an identification for each page in the main memory. The identification bit or modification bit of a particular page is set whenever one or more words in that page are modified. When a page fault occurs, the VSC controller can check if the outgoing page has been modified. It if has, then a push is needed. If it has not, the new page can be brought in to overwrite it immediately. It would seem that control of the modification bit has to be designed into the central processing unit (CPU). However this technique is applied by DEC on the VAX 11/780.

2.4.3 Buffered Technique

Nowadays most of the external storage devices are under the control of a peripheral processor instead of the CPU of the host computer. If a buffer storage of the size of a page is available in the peripheral processor, at the time of a page fault the incoming page can be sought while the outgoing page is being transferred to the buffer concurrently. The transfer from main memory to the buffer takes much less time than a seek on the external storage. Thus the incoming page can be brought into main memory as soon as the seek is completed. At the completion of a transfer from the external

storage, the VSC controller can furnish the peripheral processor with the appropriate information so that the outgoing page can be returned to the auxiliary storage from the buffer under the control of the external processor while the main programme execution resumes. As far as the host computer is concerned each page fault consists only of the access time of the incoming page. No complex circuit or algorithm is required to detect if any modification has taken place as in the case of prevention of a push, yet the same result can be achieved by the buffered technique with all outgoing pages returned to the external storage.

2.4.4 Restructuring of Programmes

Unlike the other techniques which modify the computer system to suit the running programme, this technique attempts to optimise the execution by rearranging the programme.[9] With most programmes, there are always parts that can be placed anywhere within the programme. Example of these parts are subroutines and function subprogrammes. If these relocatable parts can be placed on the same page as the calling instructions, instead of located on some remote pages, a potential page fault can be avoided. It might even be beneficial to duplicate the relocatable parts on more than one page to achieve a minimum page fault rate.

This technique required a pattern of all references of the programme before a restructured version can be produced. It appears that a restructuring algorithm incorporated into the compiler or assembler would be ideal. To make this scheme really effective a variable page size is necessary so that

a whole relocatable part can be accommodated without crossing a page boundary. This of course imposes a practical restriction on implementing such a scheme. However restructuring of programmes promises further reduction in average response time when applied together with other techniques.

3. Computer Simulation Study On Virtual Storage Techniques

3.1 Introduction

Extensive computer simulation studies on virtual storage performance on a small computer were carried out by J.N.T. Thuy [2] at the University of Adelaide. The studies were confined to the single user environment in the following three main areas:

a) Replacement Algorithm

b) Pre-paging

c) Prevention of Pushes.

This chapter describes briefly his simulation studies and summarises the results of his investigation.

3.2 Replacement Algorithms

The efficiency of a VSC system can be measured by its Fault Rate. Fault Rate is defined as the ratio of page faults to total memory references expressed in percentage. The complement of Fault Rate is Hit Rate which is the percentage of successful references with respect to total memory references.

Direct Mapping was compared with Page Associative Mapping employing the Least Recently Used algorithm (LRU) in Thuy's study. The comparison was done along the three variables:

i) Programme Language

ii) Main memory size

iii) Page size.

The programmes chosen for his studies are shown in Table 3.1. A main memory size from 512 words to 4 K words and a page size from 64 words to 512 words were investigated. The virtual storage system was modelled on a NOVA 2 minicomputer. The following points were observed in his study.

- Hit rate is higher with Assembly Language than with high level language with a given mapping and replacement scheme.
- In Direct Mapping, Hit Rate decreases with increase in page size. Also an increase in memory size does not always result in an increase in Hit Rate if page size is also increased.
- Page Associative Mapping with LRU algorithm always gives a higher Hit Rate than Direct Mapping.
- A main memory size of 2 K or 4 K words with a page size of 128 or 256 words were the optimum combinations in the LRU algorithm.

Programme Languages	Functions	Size (Words)	Number of Memory References
Assembly	FFT	5 K	1.5×10^6
Algol	Matrix Inversion	8 K	2.1×10^6
Fortran	Octal-Decimal Conversion	5 K	4×10^5

Table 3.1. Programmes Used In VSC Study

Table 3.2 lists the fault rates for the three test programmes under direct mapping and page associative mapping with a LRU stack. The system was configured for a main memory of 4 K words and a page size of 256 words.

Language / Mapping	Assembly	Algol	Fortran
Page Associative with LRU	0.01 %	0.15 %	0.1 %
Direct	0.04 %	5.7 %	6.0 %

Table 3.2. Percentage Fault Rates on 4 K Words Main Memory And 256 Words Page Size.

3.3 Pre-Paging

When a page is brought into main memory only after a page fault, it is called demand-paging. If, on the other hand, a page is brought in before it is referenced, it is called pre-paging. It is obvious that any successful prediction will reduce the number of page faults and hence the average access time. Three prediction methods were investigated with page associative mapping employing the LRU algorithm.

Basically all the three predictive methods studied brought in the next adjacent page N + 1 when the page N is referenced. It was based on the assumption that when a page is needed, the next adjacent page will be needed soon. The difference between the three predictive methods lies in the different priorities assigned to the predicted page on the LRU stack. The first method assumed the next adjacent page would be required immediately after the currently referenced page. Thus the predicted page was assigned the second highest priority on the LRU stack. However when a prediction failed the main memory would be quickly filled up with unrequested pages and thus causing further conflicts.

The second method aimed to maintain control over the effective main memory size at times of a faulty prediction. The predicted page was assigned the lowest priority on the LRU stack and it was replaced immediately on a faulty prediction. At the worst the effective main memory was reduced by one page only.

The study showed both the first and the second pre-paging methods generated more page faults than demand paging in all cases. In fact the second method generated more page faults than the first method. It was believed that the second method did not allow the prediction to stay in memory long enough to reap the benefit. The third method was introduced as a compromise by assigning a priority to the predicted page somewhere in the middle of the LRU stack. This still did not produce less page faults than demand paging.

All the pre-paging techniques studied brought the prediction into the main memory and reduced its effective size by at least one page. Buffered Pre-paging technique as described in Section 2.4.3 Chapter 2 was not investigated. As it was pointed out Buffered Pre-paging will reduce average response time even though the number of page faults remains the same as demand paging.

3.4 Prevention of Pushes

One method of prevention of page pushes, Modified Page Checking, was also investigated. In this system a flag is provided for each page in the main memory. The flag associated with a certain page is set whenever that page has been modified in any way. At times of a page fault, the

VSC controller checks if the outgoing page has been modified during its stay in the main memory by checking the appropriate flag. If the flag is not set, then the push is suppressed. The effectiveness of this technique was measured by the ratio R of pushes over pulls. The average values of R are given in Table 3.3. The reason for the difference in R's between high level languages and assembly languages was that high level languages make use of more subroutine libraries whose contents are never changed.

Programme Languages	$R = \frac{\text{Pushes}}{\text{Pulls}}$
Assembly	0.7
Algol	0.25
Fortran	0.22

Table 3.3. Values Of R For Different Languages

3.5 Average Access Time of Virtual Storage System

The average access time of a virtual storage computer system is a vital parameter which represents its performance. It is a measure of the efficiency of the chosen techniques. A mathematical model of this function was derived for a VSC system employing a magnetic disk or drum as the auxiliary storage.

For any programme executed under a VSC system let:

n = number of total memory references

t_1 = average access time of main memory

t_s = average locating and settling times of disc/drum head

t_t = data transfer time of external storage per word

p = number of words per page

F = total number of page faults

α = modify coefficient = $\frac{\text{Pushes + Pulls}}{\text{Pulls}}$

= 1 + R

The total memory reference time would be the sum of all main memory reference time plus the sum of all push-pull time of the external storage.

i.e. $\Sigma T = nt_1 + F(1 + R)(t_s + pt_t)$

$= nt_1 + \alpha F (t_s + pt_t)$

Average access time T for a reference is:

$$T = t_1 + \alpha f (t_s + pt_t) \qquad (3.5.1)$$

where $f = \frac{F}{n}$ is the Fault Rate.

4. VSC System For National Semiconductor IMP-16C Microcomputer

4.1 Introduction

Although virtual storage techniques have been applied to mainframe computers such as the IBM system/38 and top of the range minicomputers such as the DEC VAX 11/780, little work has been done towards hardware memory management in the area of microcomputers and the basic range of minicomputers. Most system manufacturers produce operating software to provide some sort of memory management, but they are costly to develop and incur high overhead. Virtual storage techniques not only reduce the requirement on physical memory size but also can simplify the operating system. The additional hardware involved can easily be justified by a less complex operating system.

Other benefits associated with a virtual storage computer system as compared to all solid state memories are:

1/ Permanent storage is produced automatically.

2/ Reduced chip count and power requirement.

3/ Upward expansion beyond addressing space is possible.

The only penalty inherent in the VSC system is the increase in average access time. This penalty might be a big drawback in real time processing. Nevertheless a low cost in-house small computer with unlimited memory resources can be a valuable tool for research purposes and can open new avenues for investigations that were not possible to undertake before. It was thus decided to implement a working virtual storage

computer on one of the microcomputers available at the Electrical Engineering Department of the University of Adelaide.

The National Semiconductor IMP-16C microcomputer was chosen because it was the only 16 bit microcomputer available to the department at the time. A 16 bit machine was favoured because of the bigger addressing space and the similarity to most minicomputers. A floppy disk was chosen as the auxiliary storage because of the speed and the medium cost. A main memory size of 4 K words and a page size of 256 words were shown to be a good combination by Thuy [2], also the chosen page size fits in with the structure of the floppy disk system. Design criteria and operation of the system will be discussed in the rest of this chapter and chapter 5.

4.2 Design Criteria

4.2.1 Mapping

As Thuy demonstrated in his investigation that associative mapping always returns less page faults than direct mapping with a given programme, it was decided that a page associative mapping scheme would be adopted. A page size of 512 bytes (256 words) was chosen because it fits in the sector size of the floppy disk structure, also an 8 K bytes (4 K words) main memory gave a reasonable fault rate with this page size.

Replacement policies involving processor time such as the Working Set (WS) algorithm required complex hardware or operating software to monitor the processor time. The

added complexity is not justified in the single user environment of the IMP-16C since the performance of the WS is compatible with the much simpler Least Recently Used (LRU) algorithm. A LRU stack can be easily implemented with shift registers and some additional control logic. The only consideration required with the design of the LRU stack is that the speed in maintaining the stack matches the worst case of two successive references of pages that are already in the main memory.

4.2.2 Pre-Paging

Sequential pre-paging was shown by Thuy and confirmed by other researchers [6], [7] to be ineffective in a virtual memory system. Non-sequential pre-paging algorithm experimented by Baer and Sager [7] produced decreases in page fault rate of from 2 percent to 14 percent for a page size of 256 words. The slight improvement did not justify the additional complexity and hardware. Thus pre-paging into main memory was discarded; however sequential pre-paging into a buffer memory was considered.

Sequential pre-paging into a buffer memory is simple to implement. Although the fault rate is not reduced by this technique, yet on successful prediction the transfer time from buffer to main memory is much smaller than the transfer time involving an external storage. With the limitation on funding it was decided to leave this as a second phase of the implementation with the first phase concentrating on the basic controller circuit. Hence no pre-paging was applied to the IMP-16C VSC system.

4.2.3 Prevention of Pushes

The floppy disk system chosen as the auxiliary storage is the SMS FT ϕ4ϕϕ floppy disk system. The SMS FT ϕ4ϕϕ features a Host Interface Processor (HIP) and a buffer of 512 bytes. The floppy disk system responds to the macro commands issued by the host computer and data transfer can be between the disk and the host directly or through the buffer. Data transfer between the host and the buffer can be carried out while the track seek operation is in progress and data transfer between the buffer and the disk is under the control of the HIP.

It was proposed that prevention of pushes would be unnecessary if the outgoing page was pushed into the buffer while the incoming page was being sought from the disk. The time estimated to transfer 512 bytes from the host to the buffer is 2.7 m sec. This is the minimum time calculated from the specification for a push operation in the proposed system. This push time is totally absorbed by the head step time of 8 m sec. On completion of the seek operation the incoming page would be transferred directly from disk to host. The outgoing page would then be returned to the disk under the control of the HIP when the pull operation is completed. The flow diagram of the process is shown in Fig. 4.1.

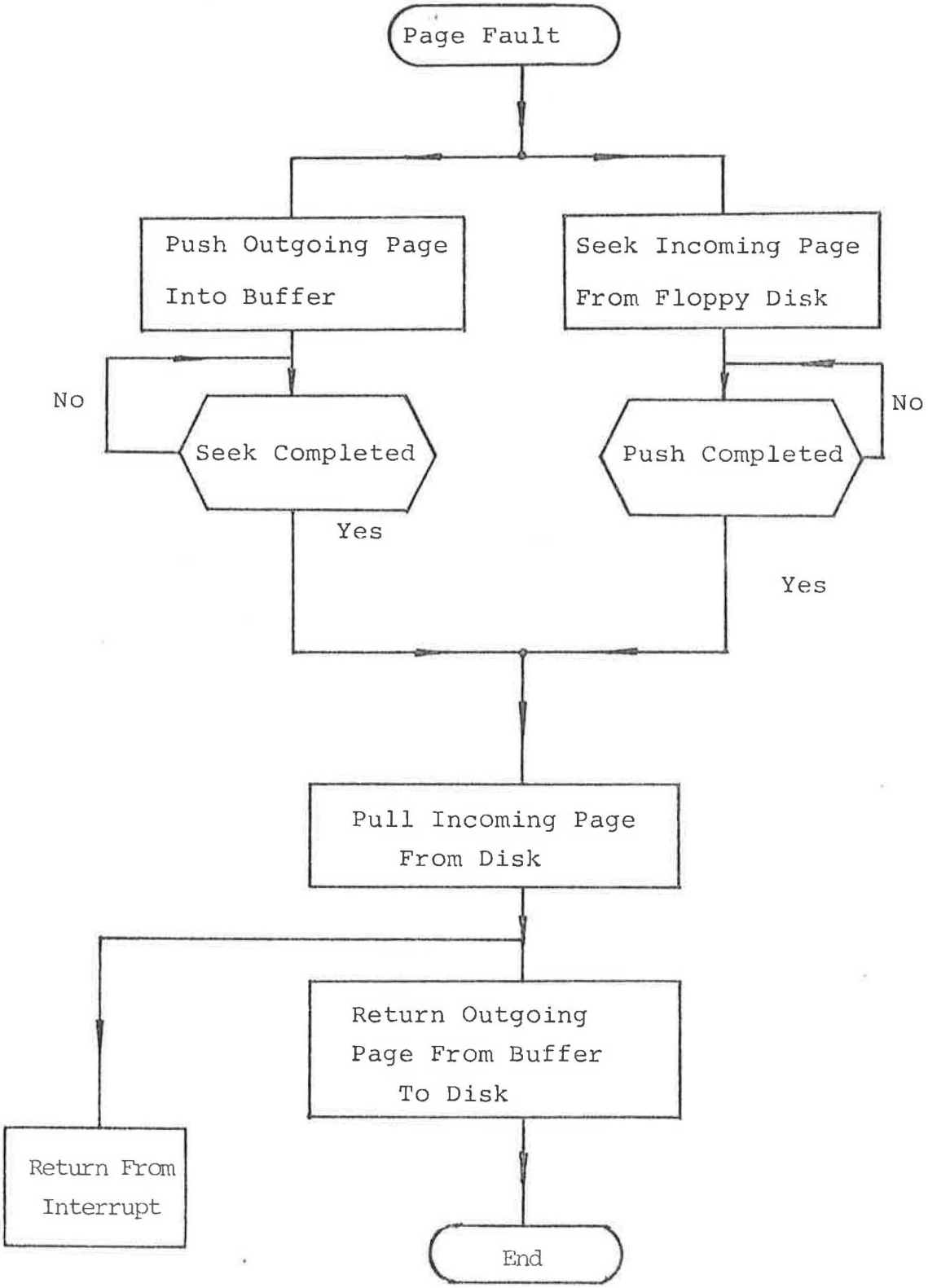


Fig. 4.1. Buffered Push-Direct Pull VSC System

Equation (3.5.1) as derived in Section 3.5 Chapter 3 would have to be modified into the form shown in (4.2.1) where the modify coefficient α equals to unity.

$$T = t_1 + f\ (t_s + pt_t)$$
$$= f\ t_s \qquad (4.2.1)$$
$$\text{since } t_s >> pt_t \text{ and } ft_s >> t_1$$

Here the average access time of the VSC system is the function of fault rate f and the access time t_s of the external storage only.

To be able to employ the system described above the host computer must be capable of receiving data directly from the floppy disk at the rate of 16 microseconds per byte. This is beyond the capability of the IMP-16C with programmed data transfer. The only way data can be received at the rate of 16 microseconds per byte for the IMP-16C is to provide a Direct Memory Access (DMA) interface for the floppy disk system. Again funding was a consideration, also it was felt that a DMA interface for the floppy disk system was outside the scope of this project. With the basic VSC controller as the first phase of the implementation, neither prevention of pushes nor buffered push-direct pull technique was incorporated into the final design. A slower but much simpler buffered push-buffered pull data transfer technique was adopted. The average access time of the final design is given in (4.2.2) where t_b is the transfer time of data between buffer and host, and t_t is that between buffer and disk per byte.

$$T = t_1 + f[\underbrace{(t_s+pt_t)}_{\text{Push}} + \underbrace{(t_s+p(t_b+t_t))}_{\text{Pull}}] \qquad (4.2.2)$$

While the transfer from host to buffer can be carried out during the track seek operation at PUSH times, transfer from disk to buffer cannot commence until the required track is found at PULL times. The buffer transfer time t_b being a programmed data transfer time is two to three times that of t_t and cannot be ignored. This results in a PULL time being slightly longer than a PUSH time.

4.3 The VSC Controller

A simplified block diagram of the IMP-16C is shown in Fig. 4.2. Separate input and output buses as well as the address bus are available at edge connectors. The 16 bit address bus provides a maximum memory space of 64 K (1 K = 1024) words. When the memory is organised into pages of 256 words each, then there will be a total of 256 addressable pages represented by the high byte of the address word. The low byte of the address word represents the word address within a page. The concept of organising the address word into page address and word address is shown in Fig. 4.3. Since a main memory size of 4 K words was found to be an optimum size for a virtual memory system, the 8 bits virtual page address is then required to be mapped into a 4 bits physical page address. In other words the 256 virtual pages are mapped into 16 physical pages of 256 words.

The VSC controller is the logic unit that performs the translation of virtual pages into physical pages; maintain the LRU stack and the working page registers and initiates the data exchange sequence when a page fault arises. It is situated between the address latches from the computer and

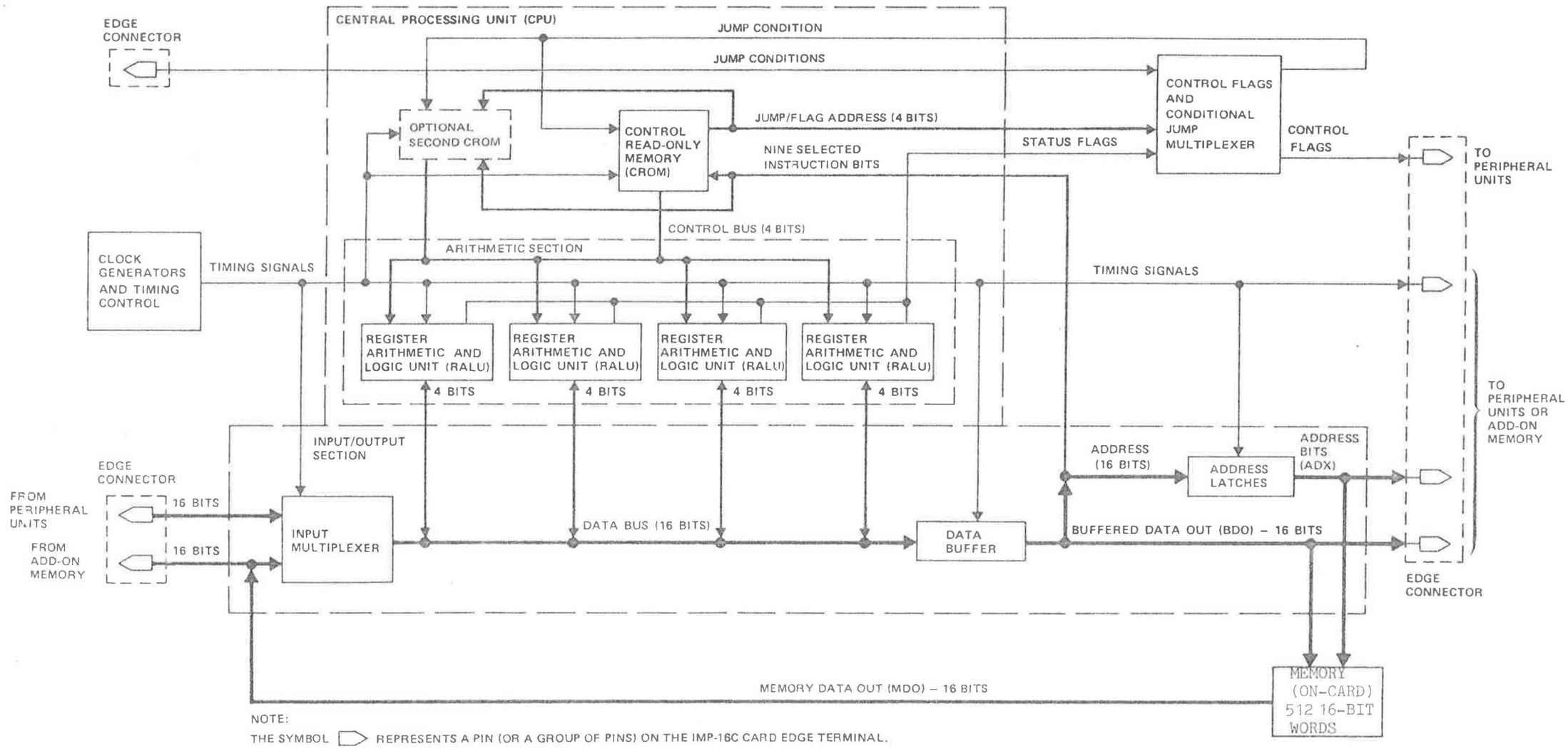


Fig. 4.2. IMP-16C Simplified Block Diagram

the add-on 4 K memories as shown in Fig. 4.3. The 512 words of on-card memories shown in Fig. 4.2 consist of 256 words of Read Only Memories (ROM) containing the firmware of control panel and TTY service routines, and 256 words of Random Access Memories (RAM) for the base page X'ϕϕϕϕ to X'ϕϕFF. The design of the IMP-16C dictates that the base page be accessable from all parts of a programme and since the address bus is hard wired to the base page, it is excluded from the virtual storage system. The firmware ROM occupies the top page memory address of X'FFϕϕ to X'FFFF, it is also exempted from the virtual storage activity. In reality 254 virtual pages are mapped into the 4 K add-on memory, while the base page and the top page occupy the on-card memory permanently.

The VSC controller is composed of three basic sections:

i) The Working Page Registers

ii) Page Fault and Physical Page Generators

iii) LRU Stack

The operations of each section will be described with reference to Fig. 4.4, the simplified block diagram of the VSC controller. The detailed circuit diagrams of the VSC controller and the floppy disk interface are enclosed in the back cover. Fig. 4.4a shows the timing on the VSC Controller functions. The page fault service routine is under the control of the IMP-16C processor. It is initiated by a hardware interrupt which is the page fault signal generated by the VSC controller.

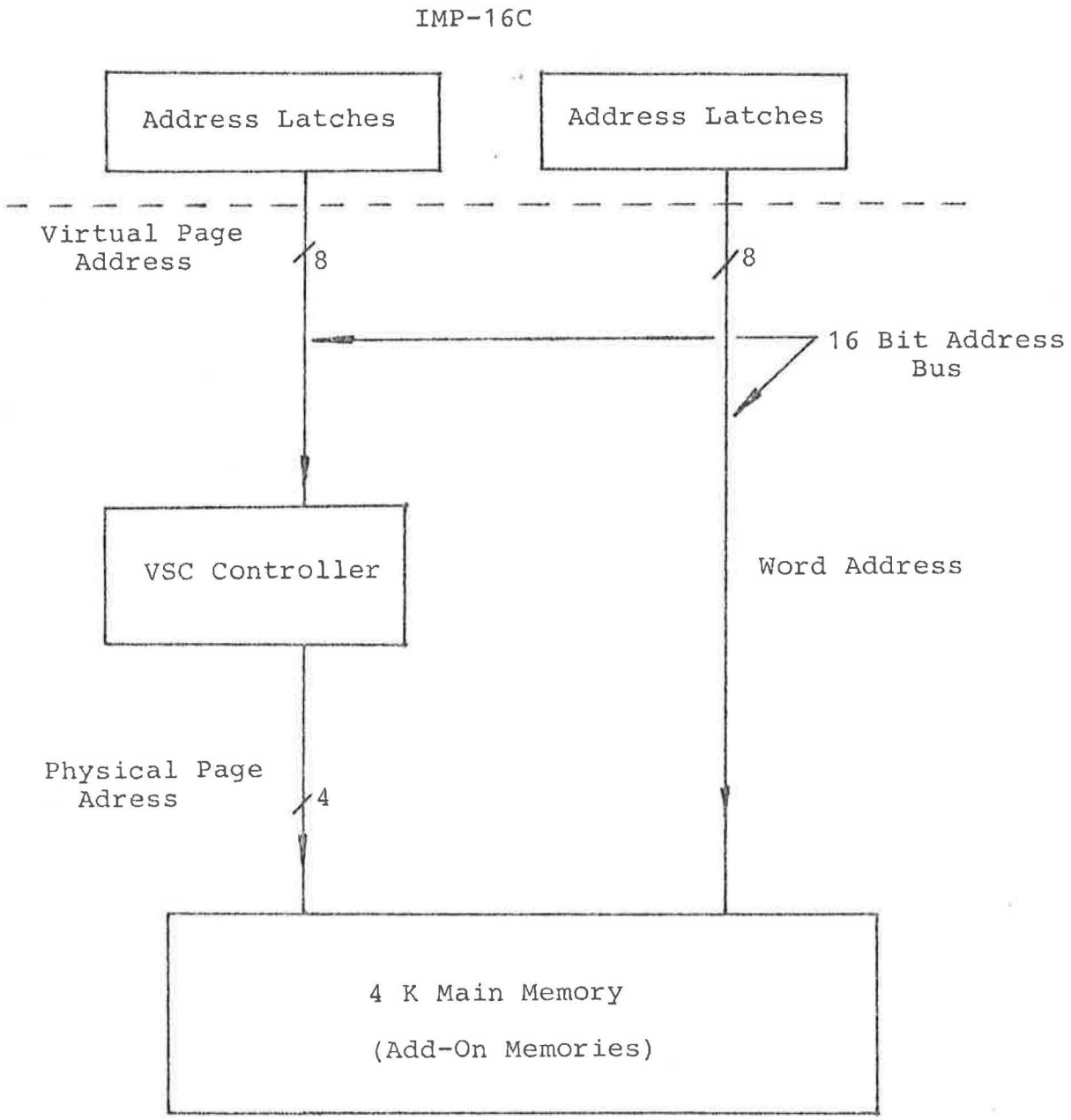


Fig. 4.3. Virtual Page Addressing

Memory Address MEMAD*

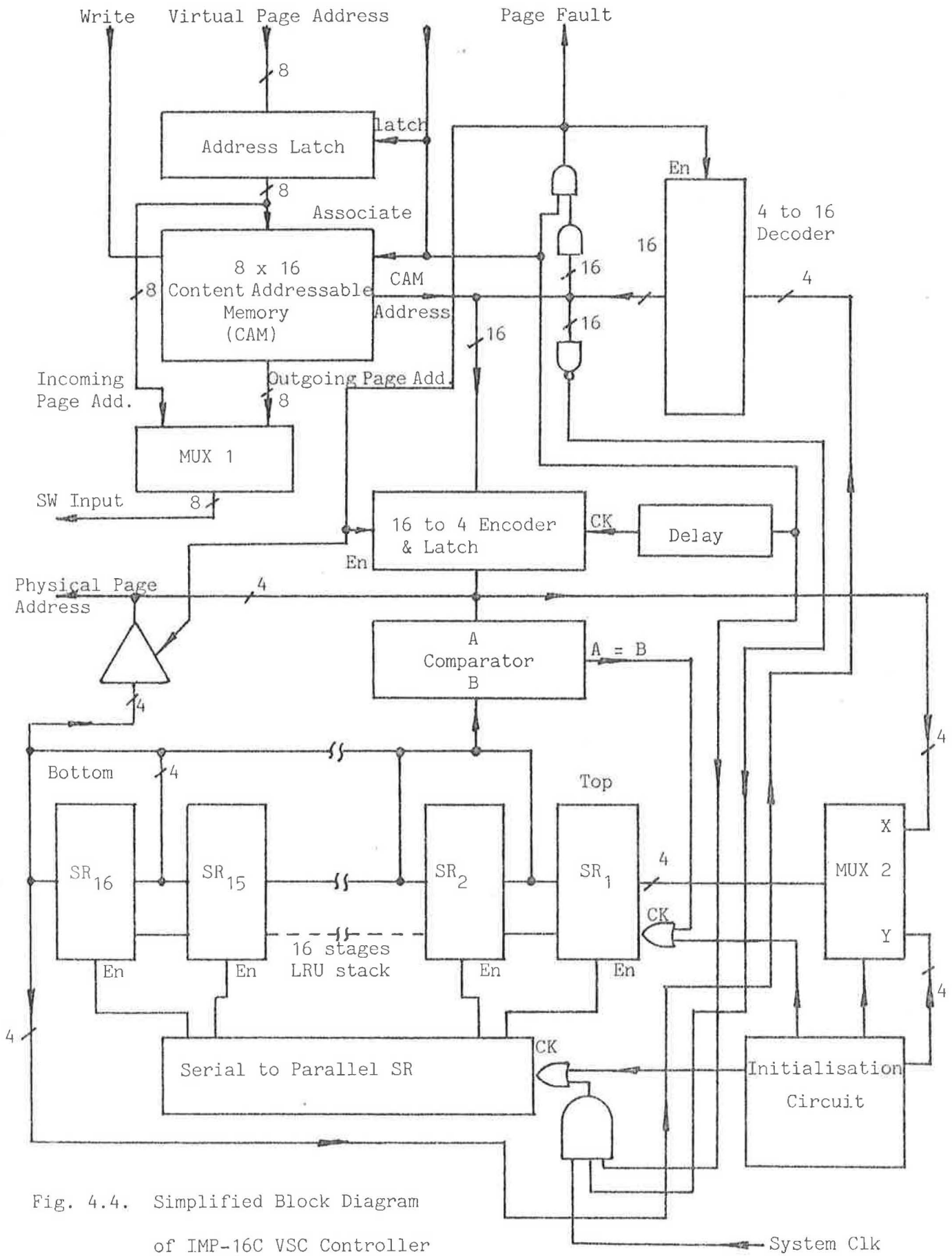


Fig. 4.4. Simplified Block Diagram of IMP-16C VSC Controller

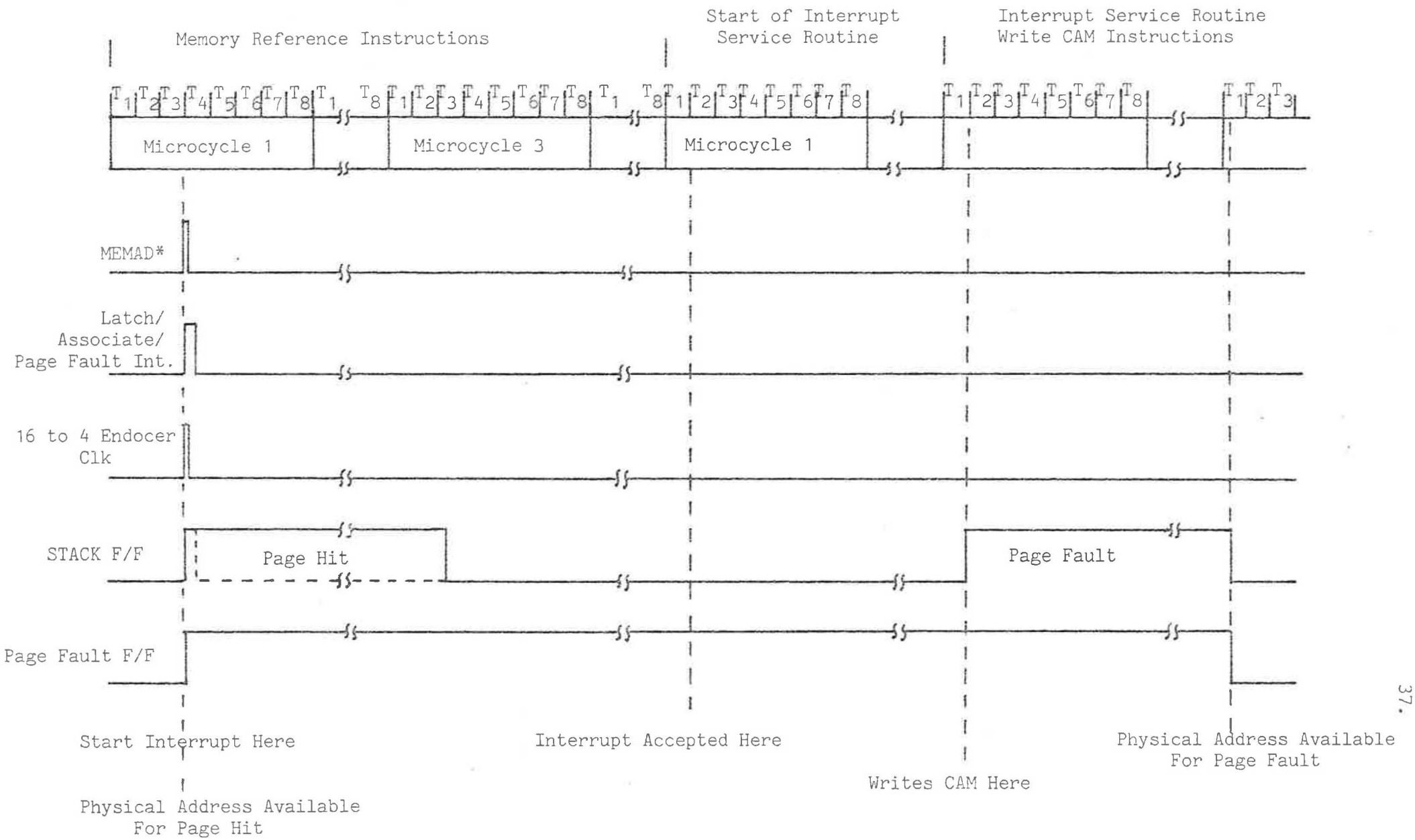


Fig. 4.4a. VSC Controller Timing Diagram

4.3.1 The Working Page Registers

The working page registers block consists of the address latch, the content addressable memories (CAM) and a multiplexer MUX 1 to channel the incoming and outgoing virtual page addresses to the host processor. This logic block is shown on the top left hand corner of Fig. 4.4.

The content addressable memories (CAM) chosen are the Signetics 10155 bipolar emitter coupled Logic (ECL) devices. The CAM's are ideally suited as the registers for the working pages because their whole content can be compared with the input with a single exclusive-OR action within a maximum of 12 n sec. The other alternative would be sequential comparison employing shift registers to store the working pages. Sequentially compared working page registers would slow down the response of address mapping to an unacceptable level.

The IMP-16C generates a memory address pulse MEMAD* eyerytime a memory reference instruction is being executed. This signal is used to latch the required virtual page into the address latch and initiate the associate process on the CAM. The CAM is arranged into an array of 16 memory cells of 8 bits each. Each virtual page that is currently residing in the main memory is stored in one of the 16 locations. An address line is associated with each location and this line goes to logic "ϕ" when its content matches that of the input. If the requested virtual page is not one of the working page, then none of the 16 address lines will go to logic "ϕ". Thus the AND output of the 16 address lines gives the mismatch condition.

The page fault signal is generated by ANDing this signal with the memory address pulse MEMAD*.

Any one of the 16 locations in the CAM can be selected for read or write operation by applying logic "ϕ" to the appropriate address line. At the time of a page fault one of the locations is selected by the LRU stack. The virtual page address from this location pointed to by the LRU stack can be channelled to the host processor through the multiplexer MUX 1 so the outgoing page content can be restored to the correct location in the external storage. Similarly the requested virtual page address is fed to the host processor for external storage reference from the address latch.

The base page and the top page are excluded from the activity described above by an address decoding logic not shown in Fig. 4.4. This address decoding logic inhibits the memory address pulse MEMAD* from the VSC controller on the detection of the base page or top page addresses.

MEMAD* is also inhibited from the controller as soon as a page fault occurs and remains inhibited until the requested virtual page address is written into the CAM. When a page fault occurs the LRU stack selects one of the sixteen locations in the CAM. The content of the physical page associated with the selected virtual page in the CAM is first returned to the external storage. The requested virtual page is then brought into the same physical page location in the main memory. All this time the address latch contains the requested virtual page address and the CAM location pointed at by the LRU stack remains engaged. When the required virtual page

is brought into the main memory, its address is then written into the CAM. The LRU stack is updated and the running programme resumes.

4.3.2 Page Fault and Physical Page Generator

This section contains the 16 to 4 encoder, a 4 to 16 decoder and the logic gates to determine a match or a mismatch condition when an associate function is performed by the CAM. If all 16 address lines from the CAM remain at logic "1" when an associate function is executed, a page fault condition is said to have occurred because the referenced page is not in the working page registers. The page fault pulse signal generated by ANDing the MEMAD* pulse with the AND output of the CAM addresses is used to set a page fault flip-flop not shown in Fig. 4.4. When this flip-flop is set the 16 to 4 encoder output is disabled and the pointer from the bottom of the LRU stack is enabled onto the physical page address bus. At the same time this pointer is decoded to engage one of the CAM locations whose content is the virtual page to be discarded from the main memory.

The VSC controller is inhibited for as long as the page fault flip-flop is set. The physical page address is the pointer on the bottom of the LRU stack for the duration when the page fault flip-flop is set. Virtual incoming and outgoing page addresses are accessable through the multiplexer MUX 1 by the host processor. The page fault service routine performs the required data transfer and then writes the incoming virtual page address into the CAM before returning to the running programme. The write signal to the CAM also

initiates the up-dating of the LRU stack. At the completion of the stack maintenance the page fault flip-flop is reset. The inhibit on the VSC controller is removed and subsequent memory references can be monitored again.

When the same virtual page is referenced again after the page fault interrupt, the address line from the CAM location holding the same virtual address will go to logic "ϕ". This signal is encoded into a 4 bits binary code to represent the 1 of 16 CAM locations. These 4 bits are latched into the physical page address bus by a delayed MEMAD* pulse. Thus the virtual address is mapped into a binary address of the CAM where it is stored. The binary address of the CAM becomes the physical page address. It is estimated that a delay of about 50 n sec is involved in such a translation. It is however insignificant compared to the 875 n sec memory access time allowed by the IMP-16C hardware. The address mapping is carried out in such a fashion for each successful reference and the LRU stack is up-dated accordingly. The performance of the IMP-16C computer is in no way affected by the VSC controller when the running programme can be accommodated by the 4 K words main memory.

The CAM is filled with all zeros (base page address) and the LRU stack is loaded with the 4 bits binary codes, from zero to fifteen in ascending order, during start up initialisation. That is the top of the LRU stack contains the binary code of zero and runs in ascending order to the bottom of the stack which contains the binary code of fifteen. The page fault service routine suppresses any pushes involving the base page. Virtual pages are brought in as required and allocated

a physical page address designated by the binary code on the bottom of the LRU stack. The push operation commences at page fault time when the CAM is filled with non zero virtual address.

In the actual implementation, the physical pages are allocated the addresses of X'3ϕ to X'3F. Fig. 4.5 shows the address allocation for the physical memories. The page fault service routine is stored in two EPROM's sharing the same address as the base page, but the use of each is mutually exclusive. As far as the user is concerned the EPROM's for the VSC controlware do not exist.

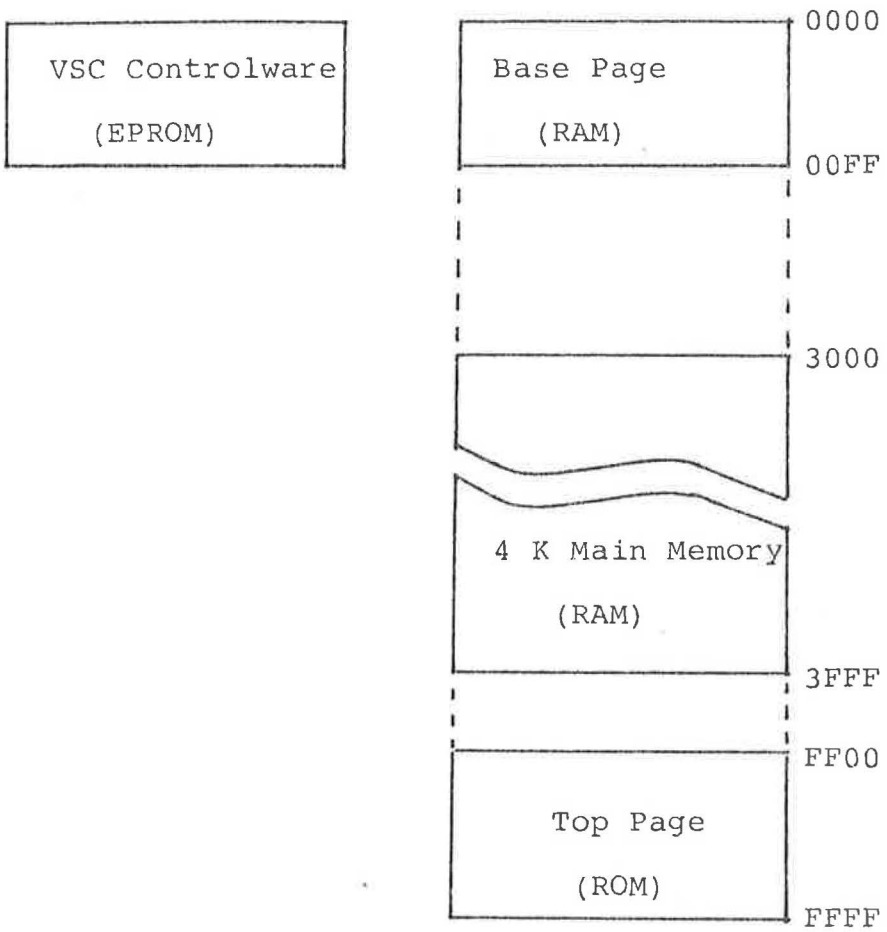


Fig. 4.5. Address Allocation for Physical Memories

4.3.3 The LRU Stack

This section is comprised of the 16 stages of 4 bits shift registers that form the LRU stack, a serial to parallel shift register, a 4 bits comparator, and the associated logic to decode a match condition from the CAM.

A match condition from the CAM is indicated by a logic "ϕ" on one of the 16 address lines when an associate function is executed. In fact the ANDed signal of the MEMAD* pulse and the NAND output of the 16 address lines sets a LRU stack flip-flop not shown in Fig. 4.4. At the time the stack flip-flop is set the binary code of the CAM address is latched onto the physical page address bus which is connected to one port of the 4 bits comparator and to the entry of the LRU stack. As soon as the stack flip-flop is set the LRU stack is stepped through with the system clock. Starting from the top of the stack the contents of each stage are gated to port B of the 4 bits comparator. Initially the clock inputs of all the LRU stages are inhibited. They are enabled a stage at a time after its content is gated to the comparator. The stage by stage enabling of the clock input is controlled by the serial to parallel shift register. The content of one of the LRU stages must be equal to the physical page address because each stage of the LRU stack was loaded with a unique 4 bits binary code during start up initialisation.

An equal condition is detected by the comparator, for example after the fourth stage was gated out. This implies that the virtual page just referenced is at the fourth place from the top of the LRU stack. At this moment only the clock inputs of the top four stages are enabled. The equal signal

from the comparator will only clock the top four stages through. A closed loop rotation of these four stages is achieved with the comparator determining the point of closure. The rest of the stack with their clock inputs still disabled remain intact. What happens to the LRU stack at this time is that the newly referenced page which was fourth from the top of the stack has been put to the top. The "hole" left by it is filled by the stage above which in turn created a "hole" filled by its preceeding stage and so on until the top of the stack is reached. Of course, all this propagation happens simultaneously in a closed loop rotation as indicated in Fig. 2.5.

The equal signal from the comparator also resets the LRU stack flip-flop thus stopping further gating out of the stack to the comparator. The stack maintenance is completed when the LRU stack flip-flop is reset.

All IMP-16C memory reference instructions require at lease three microcycles with each microcycle containing eight system clock periods. Two more clock periods are inserted by the IMP-16C hardware to the cycle when memory access actually takes place. A minimum of eighteen clock periods exist between each MEMAD* pulse. There is thus enough time to up-date the stack, even if the required page is at the bottom, before the next reference is issued.

The above description applies to a successful memory reference with the virtual page already in the working page registers. When a page fault occurs the page fault flip-flop is set, inhibiting the latch output from the 16 to 4 encoder. The binary code from the bottom stage of the LRU stack is

switched onto the physical page address bus which is connected to the entry of the stack and port A of the comparator. A similar condition to a successful reference is set up for the stack except the comparator will always generate the equal signal when the last stage has been gated out, because both ports of the comparator are fed with the contents of the bottom stage at that moment.

The binary code from the bottom stage is also fed to a 4 to 16 decoder whose outputs address the CAM and are also re-encoded by the 16 to 4 encoder for the latch. At the completion of data exchange between the main memory and the external storage, the controlware issues a write signal to the CAM. This write signal also sets the LRU stack flip-flop and clocks the 16 to 4 encoder output into the latch (whose output is still inhibited). The stack maintenance procedure described previously can now proceed. All sixteen stages of the LRU stack would have been gated out to the comparator in this case. The page fault interrupt is terminated when the equal signal from the comparator resets the page fault flip-flop. The latch output from the 16 to 4 encoder is reconnected to the physical page bus while the bottom stage of the stack is removed from this bus. Thus a LRU stack maintenance is carried out at every page hit as well as page fault condition.

4.3.4 VSC Controller Initialisation Circuit

It is most important for the proper functioning of the VSC controller that the LRU stack contains 16 unique binary codes. Any equal codes within the stack will result in more

than one working virtual page being mapped into the same physical page. Also the contents of the CAM have to be set to a known value (zero in this case) initially because none of the programme pages are in physical memory at that stage. The zeroth or base page is detected by the controlware so that a push involving the base page is not performed. The working virtual page addresses are written into the CAM as they are brought into main memory at each page fault.

After the computer system is powered up the VSC controller is initialised by momentarily engaging a "VSC Reset" switch. The output from this reset switch resets the address latch in Fig. 4.4 to zero and momentarily activates all the 16 address lines to the CAM. At the same time the system clock is gated as a write signal to the CAM for the duration that the CAM address lines are active. Sixteen system clock pulses are gated through but the first pulse would have written the base page address into all sixteen locations because all sixteen address lines are activated at that time.

The sixteen system clock pulses are generated primarily for the initialisation of the LRU stack. The rising edge of the VSC Reset pulse loads a binary count of fifteen (X'F) to the output of a 4 bits, parallel loaded, down counter 74193. The initialisation circuit comprising this counter and the associated control logic is shown in Fig. 4.6. The ZERO OUTPUT signal from pin 13 of the counter 74193 goes to logic "ϕ" only when the counter output equals binary zero (X'ϕ). The clock input to the counter 74193 is controlled by this ZERO OUTPUT signal. Hence the only stable state of the counter is at binary zero. The clock input to

the counter is enabled as soon as the output is parallel loaded with the count of fifteen, and count down commences. The output from the 74193 is fed to the Y input of the multiplexer MUX 2 shown in Fig. 4.4. MUX 2 is normally switched to the X input so the physical page bus is connected to the stack entry, but for the duration when the ZERO OUTPUT signal from 74193 is high, it is switched to the Y input. As the 74193 is being clocked through, its outputs are fed to the stack entry and loaded into the stack by the complement of the system clock. The gated system clock is also fed to the serial to parallel shift register in the LRU stack circuit so the shift register stages are clock enabled 90^{o} ahead of the clock signal. Count down of the 74193 ends when its output registers a binary zero. A delay on the system clock gating arrangement ensures the last count is clocked into the stack before MUX 2 is switched back to the X input. Initialisation of the LRU stack is completed with the bottom stage containing the 4 bits binary code of fifteen and the stage before a code of fourteen. The rest of the stack are loaded with the rest of the 4 bits binary codes in decending order. The IMP-16C VSC system is now properly initialised for correct functioning.

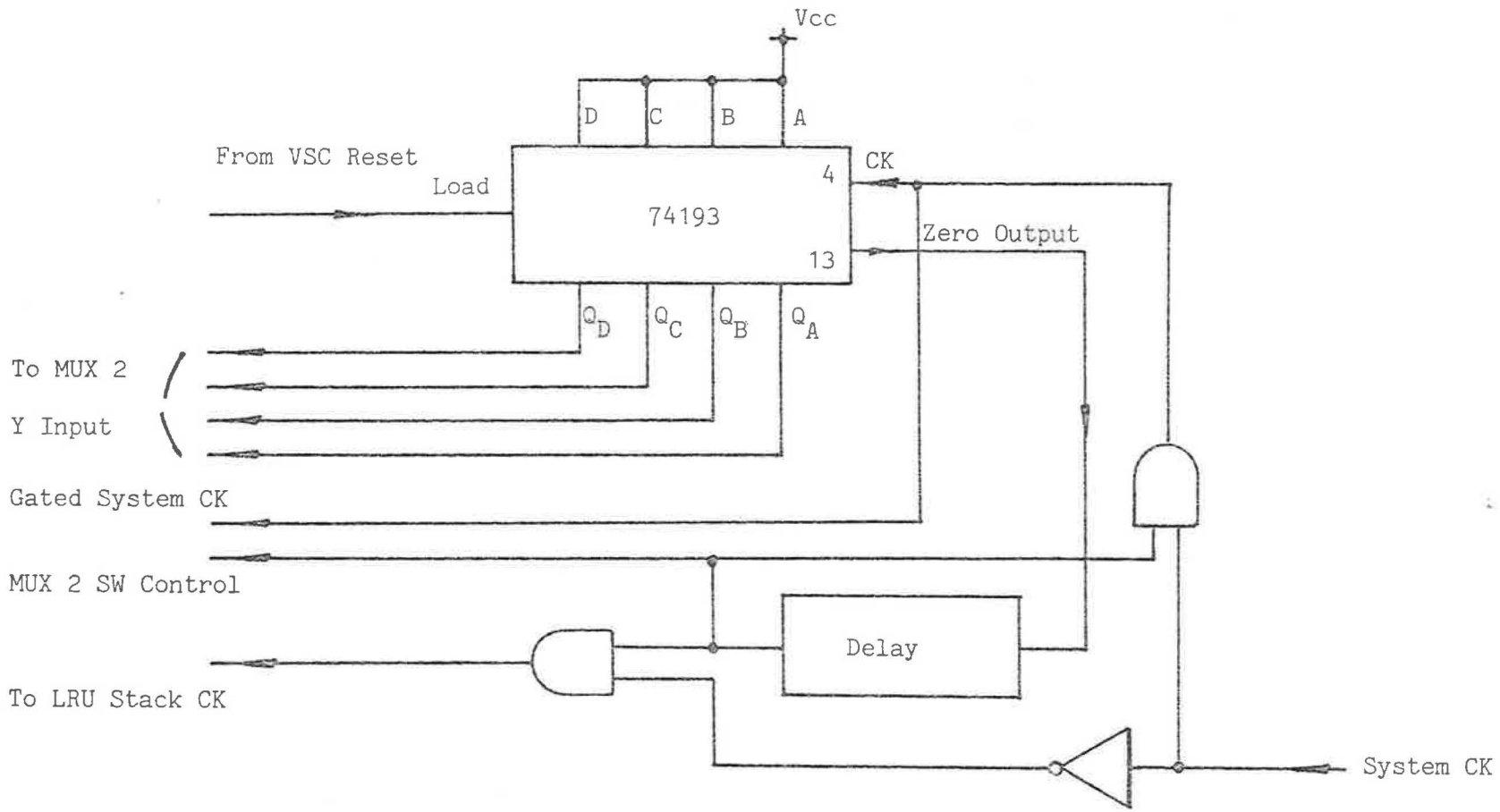


Fig. 4.6. IMP-16C VSC Controller Initialisation Circuit

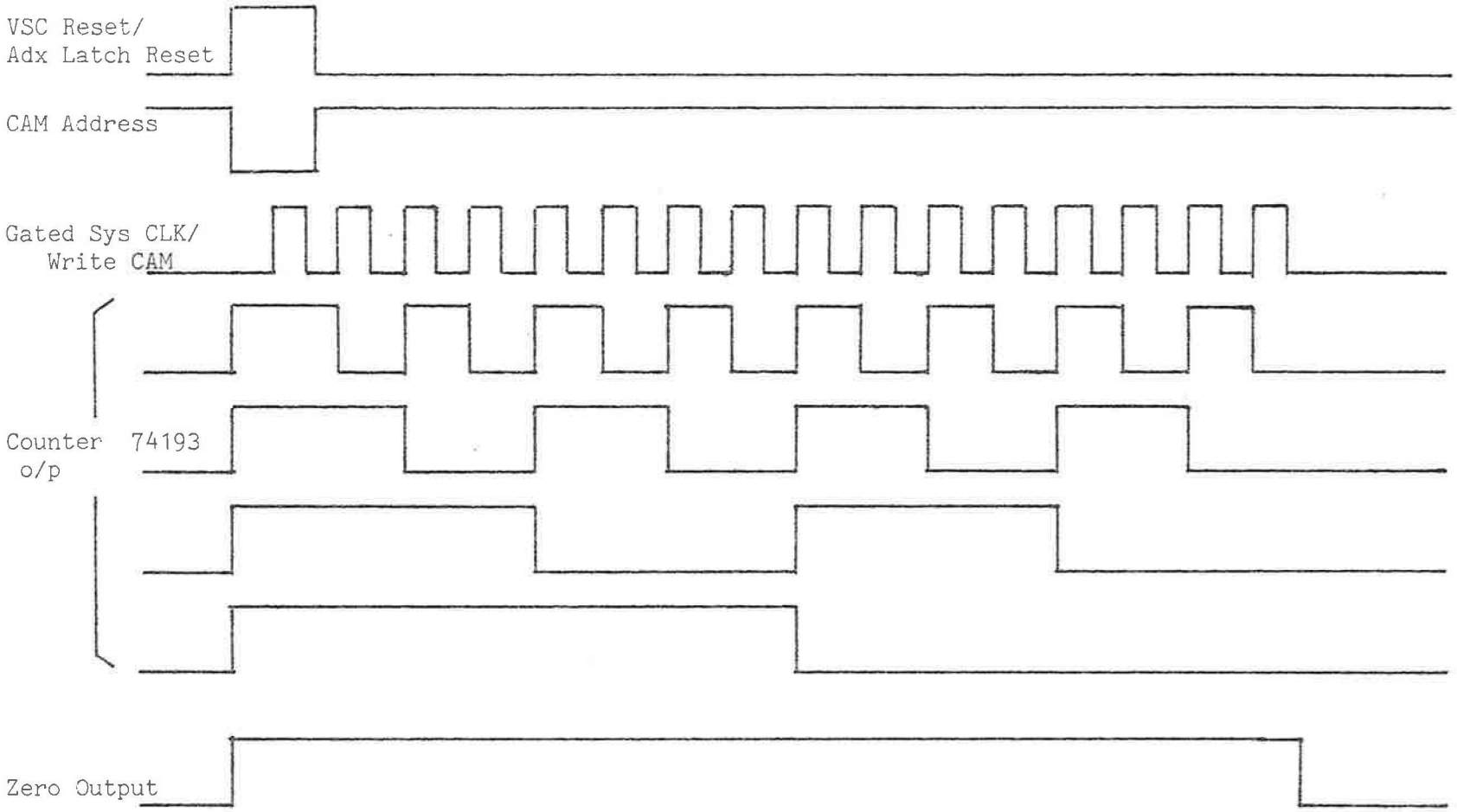


Fig. 4.6a. VSC Controller Initialisation Timing

4.4 Floppy Disk Interface Circuit

The SMS FTϕ4ϕϕ floppy disk system contains an intelligent floppy disk controller which responds to macro-commands from the host system. Data can be transferred either directly between the host system and the floppy disk at the rate of 16 u sec per byte, or through a sector buffer provided by the FTϕ4ϕϕ at a data rate determined by the host. Subsequent transfers between the sector buffer and the floppy disk are under the control of the floppy disk controller.

The interface circuit to the floppy disk system was designed with simplicity in mind and is only suitable. for progamme controlled data transfer between the sector buffer and the IMP-16C. The operation of this circuit will be described with reference to a simplified block diagram shown in Fig. 4.8. A detailed circuit diagram of the interface board is inside the pocket at the back cover.

The IMP-16C Input/Output commands RIN (Register In) and ROUT (Register Out) are employed in this mode of programme controlled data transfer. Two of the four general registers in the IMP-16C, ACϕ and AC3, serve as the data and address/ function registers respectively for the I/O commands. The high byte of the I/O command word constitutes the command code whilst the low byte of the command word is a user defined control function as shown in Fig. 4.7.

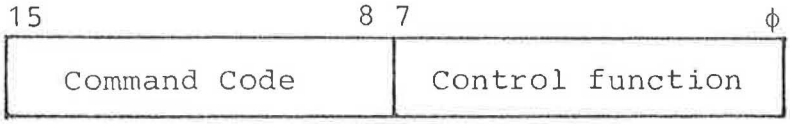


Fig. 4.7. IMP-16C Input/Ouput Command Structure

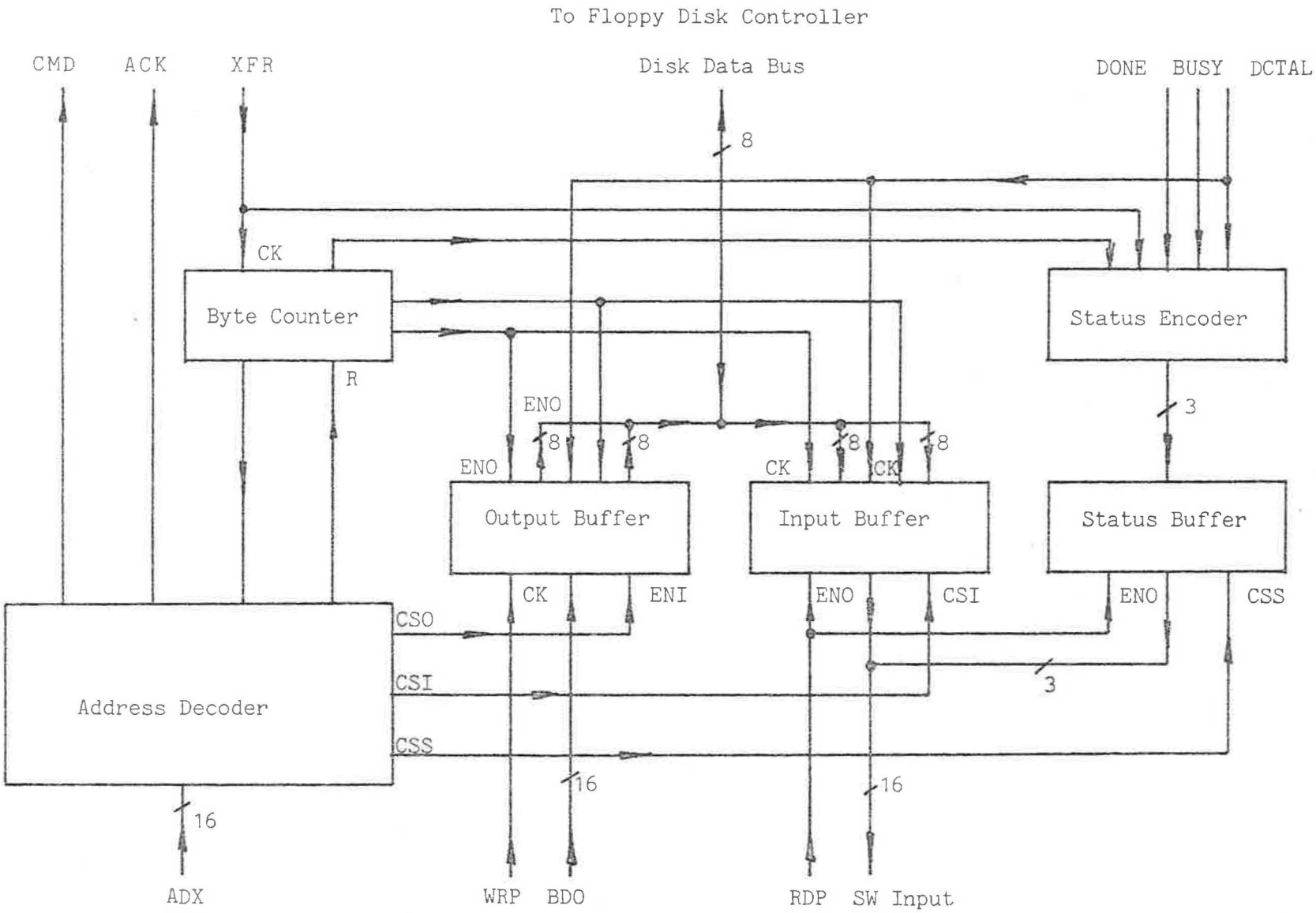


Fig. 4.8. Floppy Disk Interface Circuit

When either RIN or ROUT is executed, the control function byte of the command word is added to the content of AC3 and sent out to the address bus ADX. At the same time the content of ACϕ is sent to the buffered data output bus BDO for an output command ROUT, or the data on the switched input bus SW (data bus from peripheral units connected to the Input Multiplexer in Fig. 4.2) is loaded into ACϕ for an input command RIN. The interface board decodes the device address and the desired control function from the address bus. Appropriate handshake signals are generated according to the control function.

The interface board is allocated a device address code of X'FFXX where XX is the control function. Table 4.9 lists all the control functions recognised by the interface circuit. The controlware loads AC3 with X'FFϕϕ as device address. The macrocommand or data destined for the floppy disk is loaded into ACϕ. On the execution of ROUT with the appropriate control function XX, X'FFXX is sent to the address bus and decoded by the interface circuit to generate the required handshake signals. The WRP (Write Peripheral) signal generated by the IMP-16C for every ROUT command will latch the content of ACϕ, which is available on the BDO bus, into the output buffer. The byte counter is reset so the first byte of the output buffer is enabled onto the Disk Data Bus. The CMD (Command) signal is raised if the output is a command. The first command byte is received by the FDC (Floppy Disk Controller) and the XFR (Transfer) signal is returned to advance the byte counter so the second command byte is available to the FDC. (Most macrocommands are two bytes commands. The four byte commands for Block Mode data transfer

Control Code XX	Function
Xϕ	Data In Ack
X1	Status In
X2	Data In
X4	Data Out
X8	Command
1ϕ	Outgoing Page Address
2ϕ	Incoming Page Address

Table 4.9. Floppy Disk Interface Function Codes

is not supported by this interface.) A second XFR is returned to indicate the reception of the second byte. The byte counter is again advanced thus disabling both output ports of the output buffer, ending one transmission cycle.

All floppy disk controller activity is initiated by the IMP-16C via commands, which may be issued whenever the floppy disk controller is not busy. The IMP-16C keeps a check on the state of the floppy disk controller and the interface circuit by issuing a STATUS IN function before every transmission cycle. The primary status information bits from the floppy disk controller are fed to the status encoder circuit. The three status bits available to the IMP-16C are FDC STATUS, OUTPUT STATUS and INPUT STATUS. The FDC STATUS is checked before every macrocommand is issued. The OUTPUT STATUS is checked before every data output cycle and the INPUT STATUS is checked before every data input cycle.

The DCTAL (Directional) signal indicates the direction of data flow on the Disk Data Bus. The XFR signal is raised by the FDC to initiate a data transfer cycle. The acknowledge signal ACK is raised by the interface to indicate a data byte is available on the Disk Data Bus for a Data Out operation or the data byte is latched into the input buffer for a Data In

function. The sector buffer size is determined by hardware strapping and it is the responsibility of the IMP-16C to transfer the correct number of data bytes.

For a Data Out operation, the OUTPUT STATUS bit shows the output buffer is empty after the second byte is accepted by the FDC. The IMP-16C then latches the next two data bytes into the output buffer and checks the OUTPUT STATUS bit again. In the mean time the XFR and the ACK signals coordinate the transfer of the output buffer content to the FDC. The transmission cycle is completed when the second ACK is sent to the FDC and the OUTPUT STATUS bit is set again.

For Data In operation the data bytes are latched in the input buffer by the XFR signals from the FDC. The DCTAL signal enables the appropriate data buffers for the required operation. Again ACK is returned to the FDC after each byte is received by the interface. The INPUT STATUS bit is set when the input buffer is full. The RIN command from the IMP-16C then reads the data from the interface input buffer and resets the byte counter. Further data are latched in and the cycle starts again.

The primary status bits from the FDC are presented to the status buffer continuously. IMP-16C can access the FDC status and the data buffer status by issuing a RIN with a STATUS IN function.

The control function from the I/O command is decoded to generate the correct CS (Chip Select) signal for the various buffers. The incoming page address from the Address latch and the outgoing page address from the CAM in the VSC controller board are read in the same way with the respective control codes.

4.5 Memory Segmentation

The double density floppy disk employed in the SMS FTϕ4ϕϕ has a capacity exceeding half a mega bytes. The 16 bits virtual address from the IMP-16C produces an address space of 64 K words (128 K bytes). That is, only about one quarter of the floppy disk storage is being exploited. Obviously the cost per unit virtual storage would be reduced by a factor of four if the rest of the disk could be utilised. In other words an eighteen bits address bus would make use of floppy disk resources more efficiently. However, it would require major re-design effort on the IMP-16C to modify the address bus from 16 bits to 18 bits. Thus the only alternative was to simulate the extra two address bits by software.

The floppy disk is formatted into 77 tracks, each with 17 sectors of 512 bytes. It is more convenient to access only 16 sectors per track, thus the 16 bits virtual address can only access 16 tracks of the floppy disk. Fig. 4.10 shows how the virtual address bits are related to the floppy disk track and sector addresses.

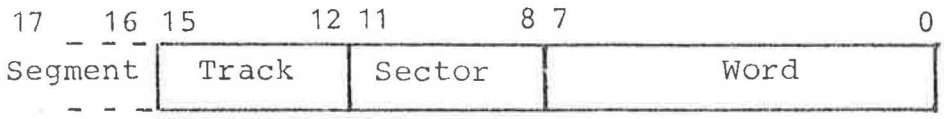


Fig. 4.10. Mapping Of Virtual Address Into Floppy Disk Track And Sector Addresses

Each sector of the floppy disk is equal to one page of the virtual address. The virtual page address, which is the high byte of the virtual address, designates the track and sector addresses by bits 12 to 15 and bits 8 to 11 respectively.

All data transfers between the host computer and the floppy disk are initiated by a TRACK SEEK command with the appropriate track number then followed by a READ SECTOR or WRITE SECTOR command with the desired sector number. The four bits sector address is masked off from the virtual address for the READ/WRITE SECTOR commands. Similarly the track address is masked off from the virtual page, but a segment address is added to the track address to produce an effective 6 bits track address for the SEEK command. The segment address can assume the values of $\phi\phi$ to 11 thus an effective virtual address space of 4 x 64 K words is achieved. The segment address is set by the user and once set subsequent references are referred to the chosen segment until a new segment is assigned. Multiple segment references are not supported by this scheme because the segment address is not part of the virtual address and is not being checked for by the VSC controller.

Cross segment reference is achieved via a service call issued by the programmer. A utility routine to return the main memory contents to the disk before the new segments are brought in has been incorporated in the controlware. Frequent references across the segment boundary are not recommended because of the delay involved. However up to 256 K words of memory are available for a single programme by employing this technique. The segment address is stored in the two reserved locations X'$\phi\phi\phi$F and X'$\phi\phi$1F. Both locations are set to the desired segment by the programmer through the control panel after system initialisation. Subsequent changes to the segment address are done on X'$\phi\phi$1F only as shown in the flow diagram in Fig. 4.11 for the cross

segment service call. The cross segment utility routine pushes the sixteen pages in the main memory back to the tracks specified by the old segment address in X'ϕϕϕF and at the same time pulls in new segment tracks according to the content of X'ϕϕ1F. When the sixteen pages from the old segment have been returned to the floppy disk, control is passed to the destination address specified by X'ϕϕ1E and programme execution can continue on the new segment.

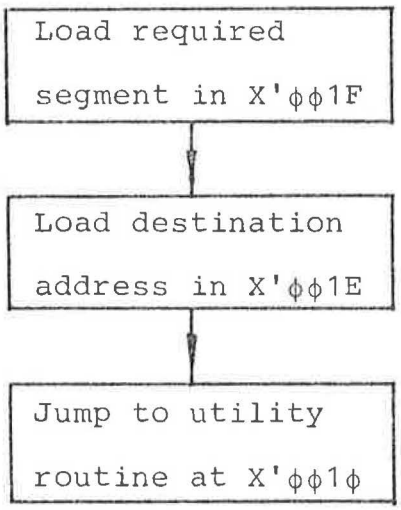


Fig. 4.11. Flow Diagram For Cross Segment Service Call

It is the programmer's responsibility to issue the cross segment service call before referencing a new segment. The programmer is also responsible for the continuity of programme flow across the segments. Under this arrangement the floppy disk can be regarded as being divided into four segments each containing 16 tracks of 16 sectors at 512 byte per sector. Fig. 4.12 shows diagrammatically the segmentation arrangement within the floppy disk.

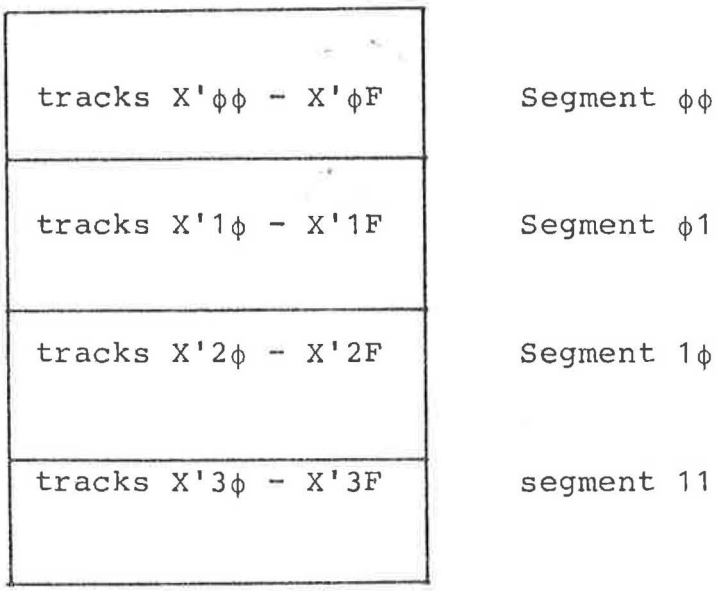


Fig. 4.12. Segmentation Of The Floppy Disk

The first sector and the last sector of each segment represent the base page and the top page of the virtual address respectively. These two sectors are not accessed. Fig. 2.1 in Supplement I shows the structure of the 256 K words of virtual memory from the user point of view.

5. Operating The IMP-16C VSC System

5.1 System Evaluation And Results

The IMP-16C VSC system was designed for easy operation with built-in features to assist in system maintenance. Supplement I contains a User's Guide produced separately, detailing the operating procedure and the functions of both hardware and software supports.

The utility routines provided are an EXIT routine to return the main memory to the floppy disk before power off; a CROSS SEGMENT service routine to return the old segment to disk before bringing in the new segment; a CONTROLWARE IMAGE routine to maintain an image of the controlware in the floppy disk; and a FORMAT routine to format disks both in single or double density at various bytes per sector configuration.

The hardware supports are the VSC Controller Reset switch to initialise the LRU stack and the CAM; a Page Fault LED to indicate a page fault condition; a Floppy Disk Reset switch; a Floppy Disk Status switch to display the floppy disk status on the front panel of the IMP-16C and a maintenance switch to display the incoming and outgoing virtual pages and the programme counter (PC) where a page fault occurs. The locations of these switches and LED are identified in Fig. 3.1 of Supplement I.

A correct power up procedure is important to avoid accidental corruption of data stored in the floppy disk. Once the VSC controller is initialised and the desired segment allocated, the user programme can be called in through the

control front panel of the IMP-16C. Test programmes to sequentially read and write through the entire segment have confirmed the correct operation of the Working Page Registers and the LRU stack. Cross segment reference is successful; giving an effective virtual memory space of 256 K words from a 16 bits virtual address. Although only one segment can be memory resident at a time, it is possible to cater for multi-segment coresidence with additional hardware as suggested in the next chapter.

The average access time T derived for this system was shown in (4.2.2) as

$$T = t_1 + f\,[2t_s + p\,(2t_t + t_b)]$$

where t_1 = average access time of main memory

t_s = average locating time of disk head

t_b = average transfer time per byte between buffer and main memory

t_t = average transfer time per byte of disk

f = fault rate

p = number of bytes per page

The term $[2t_s + p\,(2t_t + t_b)]$ is the average time required to service one page fault interrupt. With the present configuration, access is confined to a single segment of 16 tracks at a time. The average number of tracks to be located is $^{15}/_{2}$. The average page fault interrupt service time is 420 m sec. based on the following data derived from design specifications:

t_1 = 1 μ sec.

t_s = 186.3 m sec.

t_b = 53.8 μ sec.

t_t = 19.14 μ sec.

p = 512 bytes

This figure agrees closely with the measured average time of 425 m sec. between the set and reset of the Page Fault Flip Flop on the VSC controller. Using the fault rate value of 0.1×10^{-3} for assembly language from Table 3.2 of Chapter 3, the average access time of the system is 43 μ sec. A representative practical value for the fault rate is not available at this stage due to the lack of available test programmes of suitable size.

5.2 System Limitation

The programme counter (PC) of the IMP-16C has already been advanced when the page fault interrupt occurs. Servicing of the interrupt cannot commence until the current instruction is executed. Since the virtual address required by the current instruction is not in the main memory yet, the PC has to be decremented after a page fault interrupt so the instruction can be carried out with valid data. The main memory is disabled for one instruction cycle as soon as the page fault flip flop is set. The code word returned from an instruction fetch cycle is X'FFFF. This dummy instruction is executed whenever a page fault interrupt is caused by fetching instructions that are not in the main memory. Similarly X'FFFF is the data word for a page fault interrupt caused by referencing data whose addresses are not in the Working Page Registers.

Certain data reference instructions involving the accumulators such as ADD, SUB, AND, OR etc. would have operated on the dummy data and produced invalid results in the accumulator if they were allowed to generate a page fault.

Thus these instructions are not supported by the IMP-16C VSC system. Appendix A lists the memory reference instructions which fall into this category. This problem can be overcome by issuing a simple LOAD AC instruction before any of the listed instructions is executed. The suggested equivalent instructions are also listed in Table A.2 in Appendix A.

The present configuration of the VSC controller does not check the segment addresses. The value stored in location X'ϕϕ1F is taken as the segment address and added to the track address for the controlware PULL routine. Similarly location X'ϕϕϕF is used for the controlware PUSH routine. The programmer has to assign a new segment number to X'ϕϕ1F and issue a CROSS SEGMENT service call before a new segment can be referenced. Hence the programmer has to keep track of the routines or data he places in other segments. The time involved in clearing the main memory of the old segment everytime the segment boundary is crossed would also slow down the execution time, but the clearing of the main memory could be avoided with some additional hardware to check the segment address. Some proposals to improve the system average access time and relax the existing limitations will be presented in the next chapter.

AMENDMENT TO CHAPTER 5

To prove that the system does possess 64K words of virtual memory in each segment, a test programme was developed to sequentially write the address word of each memory location into the memory cell. Hence location X'ϕϕFF will contain the data word X'ϕϕFF and location X'FFEF will contain the data word X'FFEF and so on. On subsequent reading of any memory location after the execution of this test programme, the data returned matched the memory address specified every time. This test procedure demonstrated the existance of the 64K virtual memory and the correct functioning of the working page registers and the LRU stack.

The proper functioning of the working page registers was also checked by noting the page addresses that had been referenced. Referencing any pages that were already in the working page registers would not generate a page fault. A page fault condition always arises once a page that was not in the working page registers was accessed. The LRU stack was also checked with a logic probe after each read or write operation. The content of the LRU stack was tabulated and the correct operation of the stack was confirmed by tracing through the page movement.

6. Conclusion

6.1 Discussion

The average access time of a Virtual Storage Computer (VSC) system is a function of fault rate f and the time required to service a page fault interrput T_f. Fault rate depends on the chosen address translation technique and replacement algorithm. It is also greatly dependent on the memory reference pattern of the running programme. Much work has been done on VSC systems to reduce fault rate. Any further significant reduction on fault rate is unlikely even with sophisticated predictive paging techniques. However the idea of restructuring the programme before running it on a virtual storage computer promises realistic improvement on the hit rate. It is feasible to incorporate the restructuring algorithm as part of the assembler programme. Further research effort towards this direction may prove fruitful.

Improvement of the page fault interrupt service time T_f is primarily a system design problem. In the case of the IMP-16C VSC system a Direct Memory Access (DMA) interface to the floppy disk system would reduce data transfer time and eliminate one disk head locating overhead as explained in section 2.4.3 of Chapter 2. Faster external storage devices such as hard disk or even bubble memories when funds are available would reduce a big proportion of the access time. With the floppy disk system the track seek time and the head load time contribute the bulk of the locating time. This locating time can be reduced if a track pre-seeking technique is employed. At present during a page fault the disk head

is first moved to the track designated by the virtual address specified by the bottom of the LRU stack, then to the track specified by the incoming virtual address after the outgoing page has been transferred back to the disk. The disk head will remain at this track until the PUSH time of the next page fault. It is highly probable that the page being pushed out at the second page fault is the page which was at the second bottom stage of the LRU stack at the first page fault. Since the track seek operation is under the control of the Floppy Disk Controller (FDC), the track corresponding to the bottom page of the LRU stack can be sought while the running programme resumes from a page fault interrupt.

For some microprocessors with segmented addressing capability such as the Zilog Z 8001 (23 bits address bus) and Intel 8086 (20 bits address bus), software simulated segmentation is not necessary. A virtual page address size of 15 bits can provide a virtual memory space of 8 mega bytes. The CAM's forming the Working Page Registers would have to be increased in size accordingly. In the case of the IMP-16C VSC system, an addition of 2 bits to the width of the CAM's will enable it to map a 10 bits virtual page address into a 4 bits physical page address. If a 2 bit extension to the address latch in the controller is provided to accept the software generated segment address, different memory segments can reside in the main memory at the same time. Clearing the main memory of the old segment is no longer required before a new segment is referenced. However, a service call is still necessary to change the segment address.

The limitation of having some instructions not being supported by the IMP-16C VSC system and the programmers having to issue special equivalent instructions can be overcome by modifying the assembler. The offending instructions are converted to their equivalents at assembly time. Also the segment address allocation and issuing of the cross segment service call can be the responsibility of the assembler. All these refinements will make the IMP-16C VSC system a more versatile computer system with a true 256 K words virtual memory.

6.2 Conclusion

It has been demonstrated that a virtual storage computer offers many advantages over a conventional computer. These advantages are summarised as follows:

1/ Cost-effective.

2/ Transparent to programmers

3/ Reduced chip-count and power requirement.

4/ Speed improvement as compared to traditional usage of auxiliary storage.

5/ Upward expansion beyond the addressing space is possible.

When compared with the low cost solid state memories available in the market these days, virtual storage technique still offers a cost benefit of up to ten times over a conventional system with equivalent memory size. The controller in the IMP-16C VSC system was constructed for approximately A$900.00. This price included the power supply unit and the printed circuit board. When the controller was

coupled to an auxiliary storage system, (which is regarded as a necessary part of a computer system, hence not an additional item to provide virtual storage), a virtual memory size of 256 K words was achieved. The cost per 1 K word storage in the VSC system is A$3.50. The same A$900.00 can probably purchase 22 K words of RAM with the accompanying power supply unit. That is less than one tenth of the virtual storage size.

It has also been demonstrated that a VSC controller can be designed to work around an existing processor. The average access time achieved on the IMP-16C microcomputer is 43 μ sec. which is an order higher than the conventional real memory system. This figure can be improved by the application of other techniques. The track pre-seeking technique in particular when applied to an auxiliary disk storage involves no additional hardware. The translation of virtual address into physical address can take place within 50 n sec. by using ECL Content Addressable Memories as the mapping table. Performance of the central processor is not affected by the insertion of the VSC controller on the address bus. No special measure is required to run small size programmes that fit into the available real memories.

The results indicate that most microcomputer systems with existing auxiliary disk storage can acquire the virtual memory feature at present only available in expensive mainframe computers. The conversion from a conventional system into a VSC system involves slight or no additional cost because of the reduced demand on real memories. Of course it is also feasible to design a microprocessor with virtual storage in

mind hence eliminating some of the problems in designing a VSC controller around an existing processor. The three basic blocks of the VSC controller are the Working Page Registers, the Page Fault And Physical Page Generator and the LRU stack. These basic blocks all lend themselves towards chip level design with LSI or VLSI technology. Virtual storage probably would be a standard feature of most microcomputer systems with the availability of these supporting chips.

REFERENCES

1. P.J. Denning, "The Working Set Model for Programme Behaviour". Comm. ACM11, No. 5, 323-333, May, 1968.

2. J.N.T. Thuy, "Virtual Memory Techniques For Small Computers". M. Eng. Sc. Thesis, University of Adelaide, Nov. 1976.

3. L.A. Belady, "A Study of Replacement Algorithms for a Virtual Storage Computer". IBM Systems Journal 5, No. 2, 78-101, 1966.

4. National Semiconductor, "IMP-16C Application Manual".

5. T. Kilburn, *et al.* "One Level Storage System". IRE Trans. EC-11, 2, 223-235, April, 1962.

6. M. Joseph, "An Analysis of Paging and Program Behaviour". The Computer Journal, Vol. 13, No. 1, Feb. 1970.

7. J.L. Baer and G.R. Sager, "Dynamic Improvement of Locality in Virtual Memory System". IEEE Trans. Soft. Eng. Vol. SE-2, No. 1, Mar. 1976.

8. R.P. Parmalee, *et al.* Virtual Storage and Virtual Machine Concepts". IBM Systems Journal 11, No. 2, 99-130, 1972.

9. D.J. Hatfield and J. Gerald, "Program Restructuring for Virtual Memory". IBM Systems Journal, Vol. 10, No. 3, 168-192, 1971.

10. Scientific Micro Systems, "FDO400 OEM Manual" Sept. 1977.

11. Digital Equipment Corporation, "VAX11/780 Hardware Handbook" 1978-78.

SUPPLEMENT I

IMP16C VIRTUAL STORAGE COMPUTER

USER'S GUIDE

CONTENTS

	Page
- Introduction	S-1
- Memory Structure of the IMP16C VSC System	S-3
- Hardware Functions	S-5
- Power Up Procedure	S-7
- Exit Procedure	S-8
- Cross Segment Reference	S-9
- Utility Programmes	S-10

1. Introduction

The IMP16C VSC system is a Virtual Storage Computer employing the SMS FTØ4ØØ Floppy Disk system as the auxiliary storage. The 4k words of 16 bits main memory is mapped into the 256k words in the auxiliary memory by the Least Recently Used (LRU) alogithm. Each time the main memory is referenced, a mapping table is consulted to determine whether the required location is in main memory or not. If the required address is in main memory, no swapping of data from the auxiliary storage is necessary. If, however, the required address is not in the main memory (referred to as a page fault) it will be brought in automatically without programmer intervention, and the data pointed at by the bottom of the LRU stack will be replaced. Whenever an address is referenced it is brought to the top of the LRU stack while the rest of the stack is being pushed down by one. Thus the address on the bottom of the LRU stack holds data that has not been referenced for the longest time and hopefully will never be required any more.

Instead of checking for individual locations and swapping them in and out one at a time, the data are transferred in blocks of 256 words called a page. This way only the high byte of the 16 bits addresses need be monitored. The design of the IMP16C dictates that the base page (ØØXX) be accessable from any part of the programme, thus the base page is exempted from VSC activity, and it is in the main memory all the time. The top page (FFXX) contains the control panel and TTY service routine in ROM's. It is also excluded from VSC activity. At any time up to 4.5k words can be in the main memory including the top page and the base page, but only 4k words are part of the VSC system.

A 16 bits address word provides an addressing space of 64k words. The 256k words virtual memory is composed of 4 segments of 64k words each. Cross segment reference is achieved through a service call. The present configuration only supports single segments in the main memory at a time. Thus cross segment reference is very time consuming because all 16 pages of the old segment have to be returned to the auxiliary storage before the required new segment is brought into the main memory. Additional hardware is required to support multi-segment coresidence in the main memory. Most of the basic instruction set is supported by the VSC system. Refer to Appendix A for those instructions that require special attention.

The Power Up Sequence should be strictly adhered to in order that programmes or data in the auxiliary storage not be destroyed inadvertently. After the hardware reset of the mapping table, its content is set to zero. The software data exchange routine detects the page zero and skips the section that writes data from the main memory to the auxiliary storage. So writing back to the auxiliary memory does not occur until the LRU stack is filled with pages other than the base page. An exit routine is provided to restore all the data from the main memory to the auxiliary storage so that any fresh data or programme modifications will be retained after power is turned off to the computer. Other utility programmes such as the diskette format routine and a routine to maintain an image of the VSC data exchange programme in track ØØ sector Ø1 of segment ØØ are provided. These utility programmes are stored in the system disk in page FEØØ of segment 11.

A maintenance switch is provided so that the bottom page of the LRU stack, the incoming page address and the programme counter (PC) at which the new page is first referenced can be displayed on the control panel. In the event of a fault occurring within the floppy disk system, the red LED on the front of the drive unit will sustain. The type of fault can also be displayed by pushing the "Disk Status" button and the fault can be cleared by pushing the "Disk Reset" button after noting the type of disk fault.

The reader is assumed to have a thorough understanding of the IMP16C and its control panel operation as described in the IMP16C Application Manual. For more detailed description on the floppy disk system the reader should consult the SMS FDØ4ØØ OEM Manual.

2. Memory Structure of the IMP16C VSC System

The segment address is held in locations ØØØF and ØØ1F. These locations should be set to the desired segment through the control panel as described in the Power Up Procedure before any programme is executed. Locations ØØØØ through ØØ1F are reserved for the Control Panel and the VSC data exchange routines. Any data stored at these locations by the user programme will be over-written. Each time the VSC data exchange programme is executed, Location ØØØB is incremented by 1. Thus Location ØØØB can be used as the VSC activity counter if it is set to zero initially. The 64k words of each segment are partitioned as shown in Fig 2.1. Every segment is partitioned in the same manner, thus the total memory available to the user is (63.5k x 4 + 224) words. The 224 words are the remainder of the base page. The VSC controller ROM shares the base page address of ØØØØ to ØØFF, but the use of each is mutually exclusive. The users normally cannot access the VSC controller ROM and as far as they are concerned, it does not exist as part of the memory system.

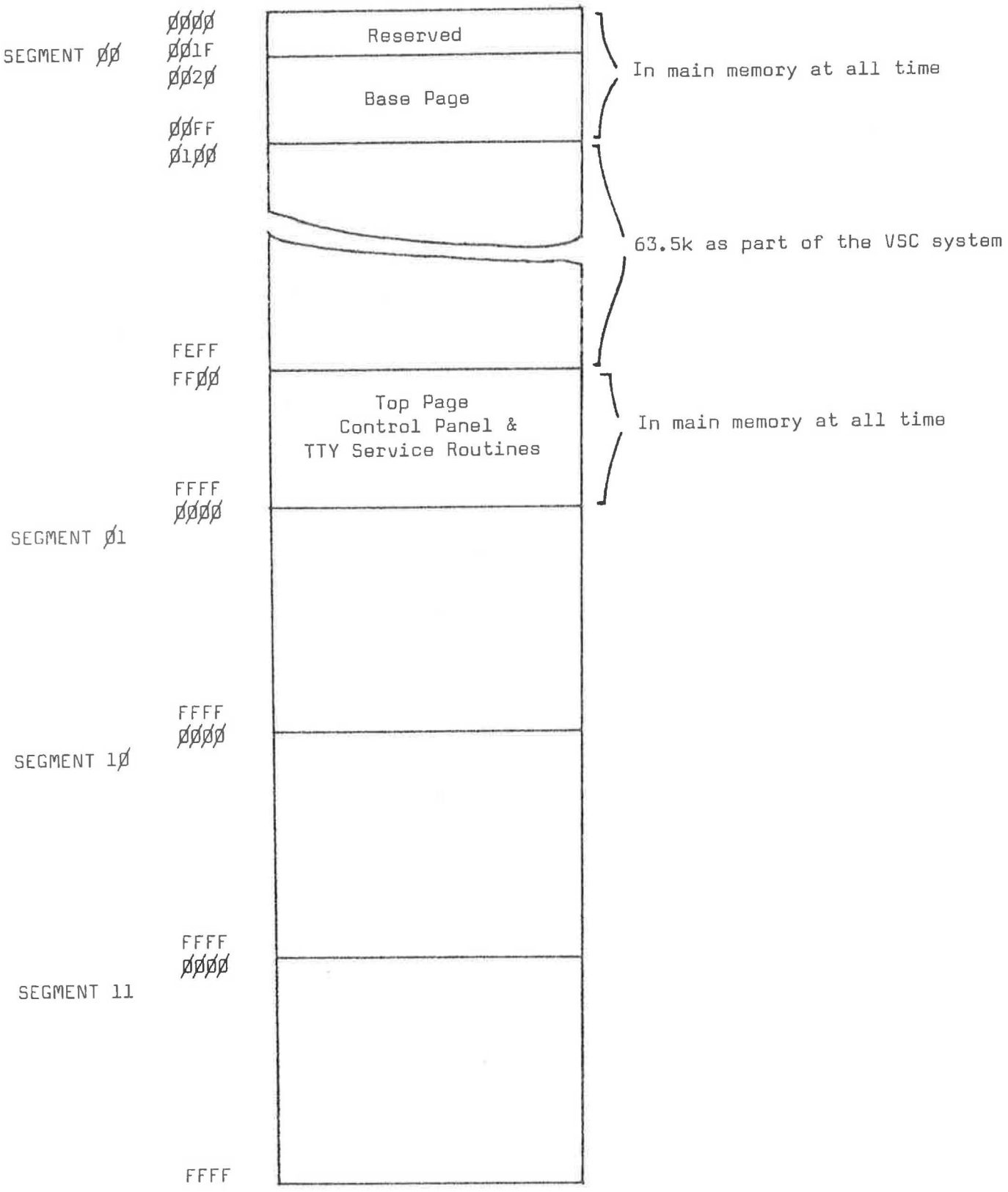


Fig 2.1 Partition of the Memory Segment

3. Hardware Functions

This section describes the functions of the three push buttons and the maintenance switch on boards 4 and 6. Board 4 contains the hardware mapping table in the form of Content Addressable Memories (CAM's) and the LRU stack plus the associated control Logic circuits. Board 6 contains the TTY and Floppy Disk interfaces and some of the VSC circuits.

Figure 3.1 shows the layout of the various switches and connectors at the back of the computer. The functions of each are labelled in the diagram.

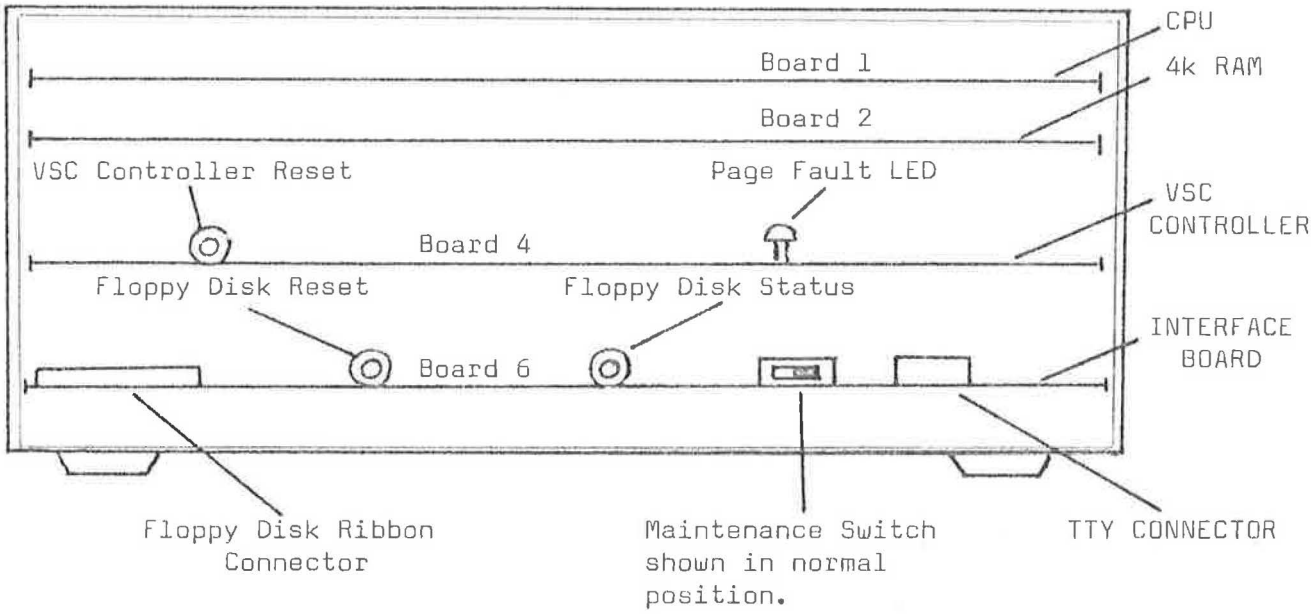


Fig 3.1 Rear View of the IMP16C

When the maintenance switch is not in the normal position, the bottom page on the LRU stack will be displayed on a page fault and the processor halts. By pressing the EXECUTE switch with the front panel switches set to non zero, the processor will continue and the incoming page is next displayed with the processor halts again. The low bytes of the above two quantities should be ignored because they are not being monitored. Next continue will display the PC at which the page fault occurs and a final continue will execute the data exchange programme. The page fault LED comes on whenever the processor is executing the VSC control software.

Table 3.2 shows the possible combination of the floppy disk unit status bits. These bits will be displayed on the high byte of the panel when the Floppy Disk Status button is pushed. The MSB is bit Ø and the LSB is bit 7 in table 3.2.

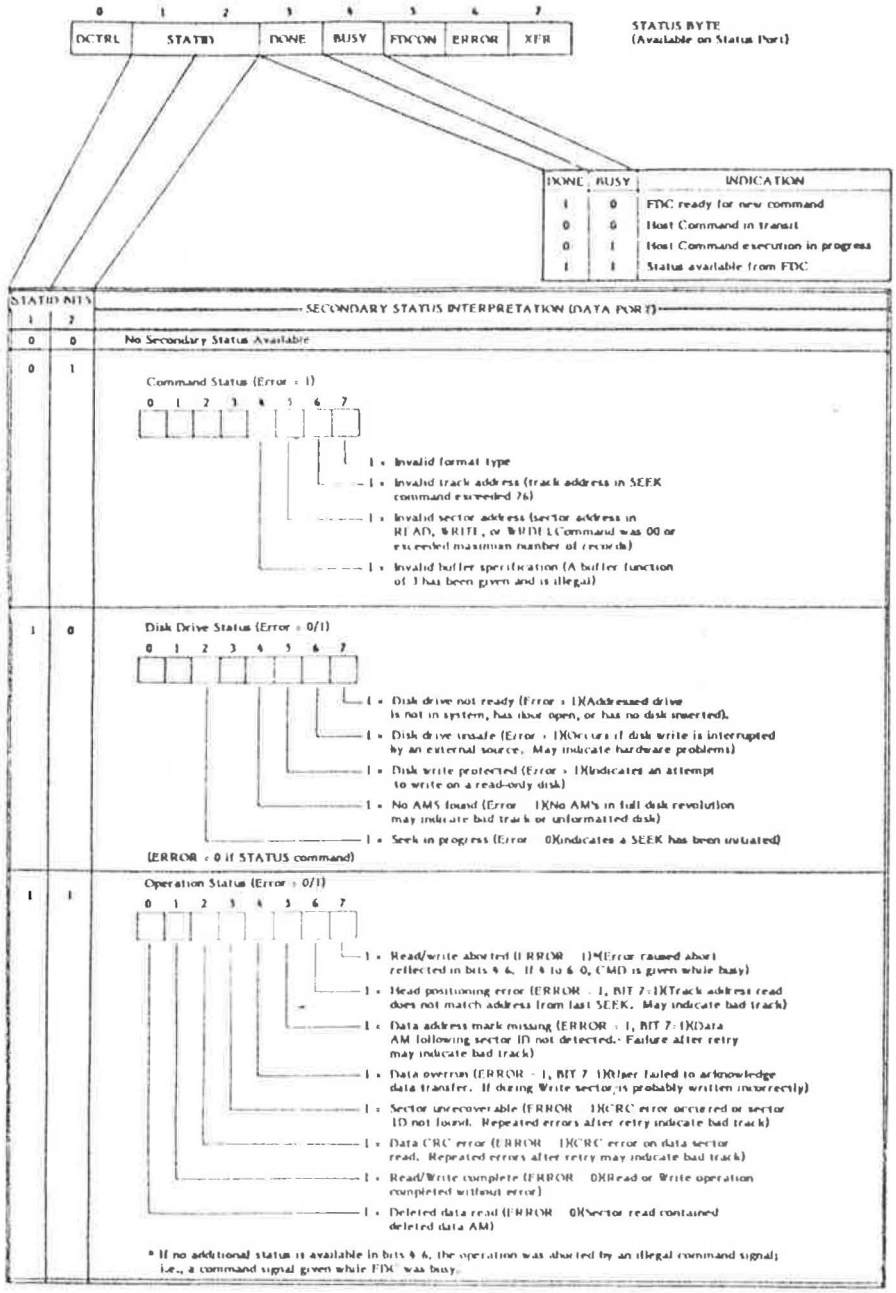


Table 3.2 SMS FTØ4ØØ STATUS

4. Power Up Procedure

The sequence should be followed strictly to avoid loss of data in the disk during power up.

a. Apply power to the Floppy Disk unit and the computer at the same time.

b. Reset the VSC controller by pushing the button on board 4 twice.

c. Initialise the IMP16C computer from the front panel.

d. Load the desired segment address in Locations ØØØF and ØØ1F. The segment address for the four segments are:

ØØØØ	Segment ØØ
ØØ1Ø	Segment Ø1
ØØ2Ø	Segment 1Ø
ØØ3Ø	Segment 11

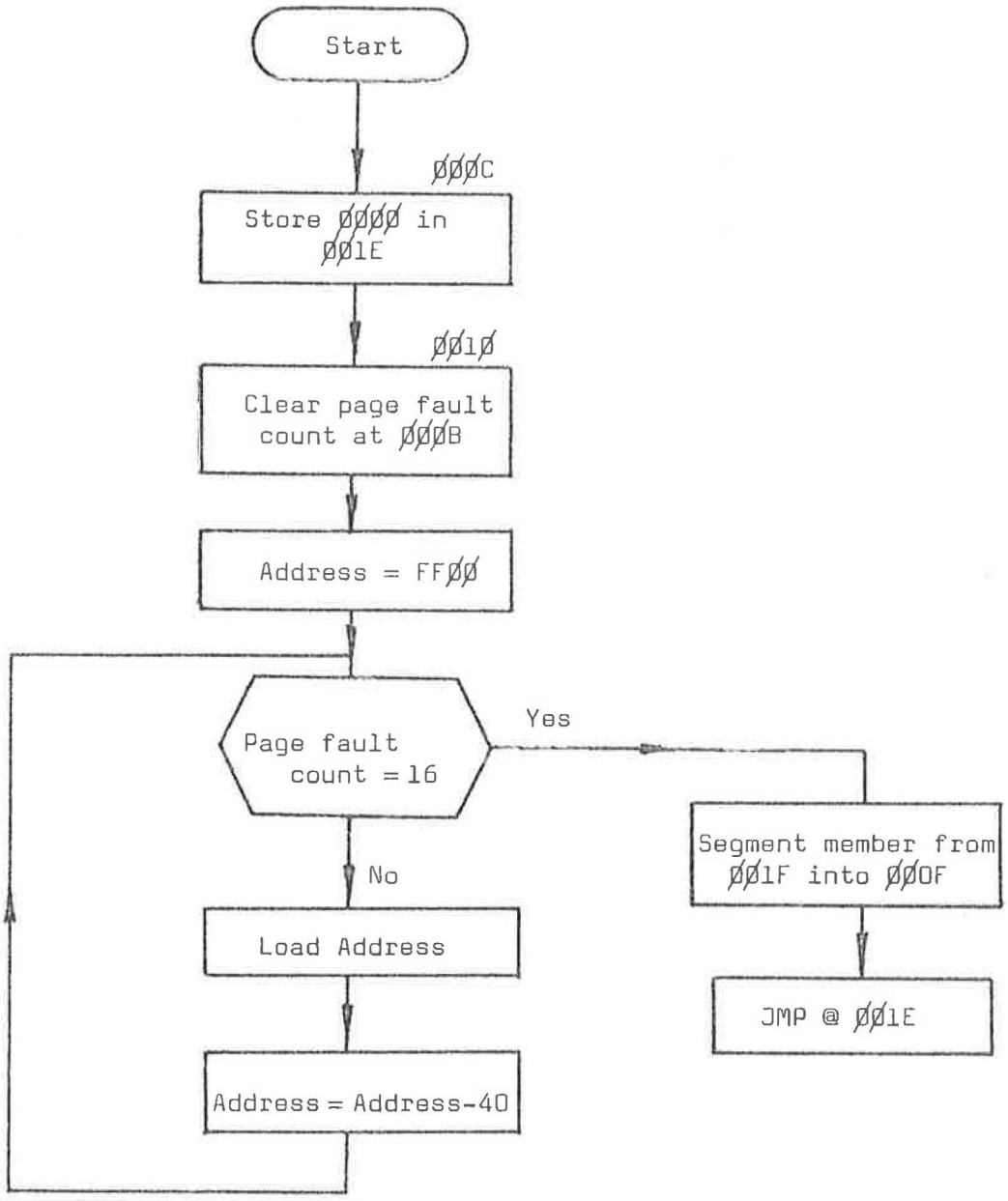
e. Reset the VSC controller once again by pushing the button on board 4.

The VSC system is now properly initialised with the LRU stack and the mapping table cleared. The user can request his programme by simply Loading the required address. The Floppy Disk system normally issues a software reset on power up by seeking track ØØ. If for any reason the user suspects the Floppy Disk unit is not reset or lack of floppy disk activity, the Disk Reset button on board 6 should be pushed.

Note: Do not run the VSC system without the fan.

5. Exit Procedure

On termination of a job, some of the user programmes and the associated data are still in the main memory. If the user wishes to restore them back to the diskette, he can execute the Exit routine at Location ØØØC. When the 16 pages in the main memory are returned to the disk, Location ØØØB will show a count of 16 (ØØ1Ø), and control is returned to the panel.



Exit Routine Flow Diagram

6. Cross Segment Reference

Whenever a new segment is required, the old segment has to be returned to the disk. The Exit routine is made use of in this instance to write back the 16 pages in the main memory belonging to the old segment. The following flow diagrams indicate the requirements of the service call and it is the programmer's responsibility to provide such a service call before new segments are referenced. The different segments are assembled separately, it is also the programmer's responsibility to ensure continuity of the programme flow across the segment.

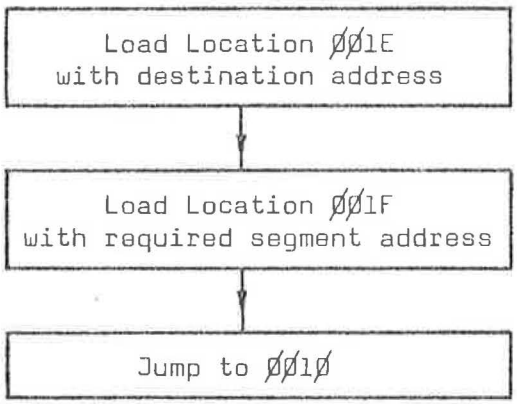


Fig 6.1 Flow Diagram of Cross Segment Reference Service Call

On the completion of the service call shown in Fig 6.1 control is passed to the destination address of the new segment. If it is desired to return control to the panel after the service call, the flow diagram shown in Fig 6.2 can be used.

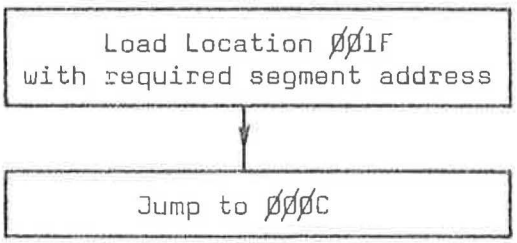


Fig 6.2 Flow Diagram of Cross Segment Reference Service Call With Return to Control Panel

7. Utility Programmes

An image of the VSC control software is stored at track ØØ, sector Ø1 of segment ØØ on the system diskette - the equivalent of the base page in segment ØØ. If for any reason the VSC control software needs modification and the EPROM's are revised, the control software image can be up-dated by calling the 'Read Image' and the 'Write Image' routines stored in page FEØØ of segment 11 from the system diskette. The 'Read Image' routine brings the control software from the system disk to the base page. After the necessary revision the base page can be returned to the system disk by calling the 'Write Image' routine.

Also on the same page of FEØØ there is a 'Format Diskette' routine that can format the floppy disk either in single density or double density at various bytes per sector configuration. Fig 7.1 is the flow diagram showing the procedure for calling the above routines from the control panel. The step that changes the segment address to ØØ3Ø is common to all routine calls. On return from the change segment routine, control is passed back to the panel and the required utility routine can be called by Loading the appropriate address. All utility routines terminate with a HALT, control can be returned to the panel by an EXECUTE with the panel switches set to non-zero. Table 7.2 shows the code word required in Location FE72 to select the desired format. 8ØØ6 is the default value in Location FE72 i.e. the default format is 512 bytes/sector at double density.

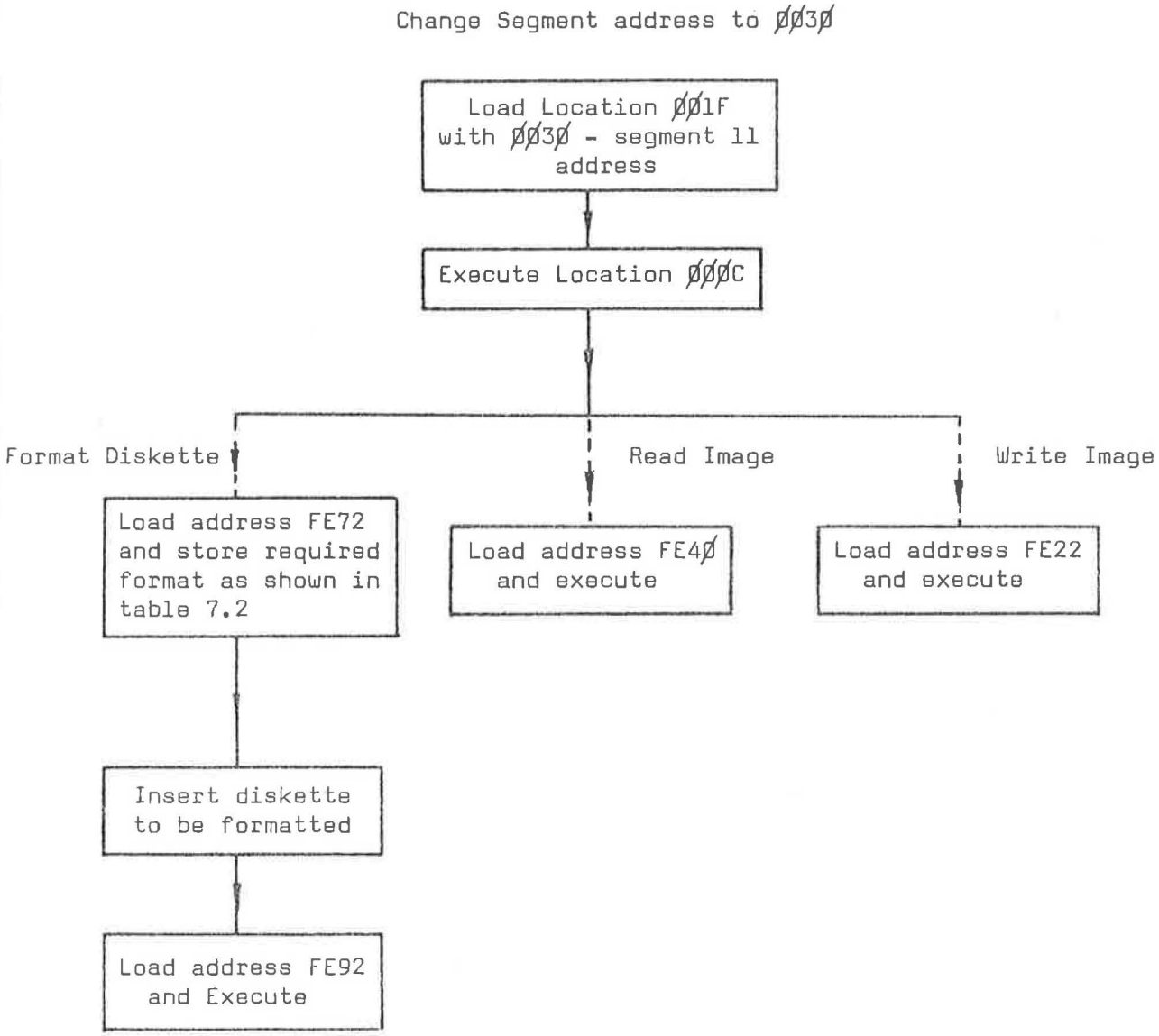
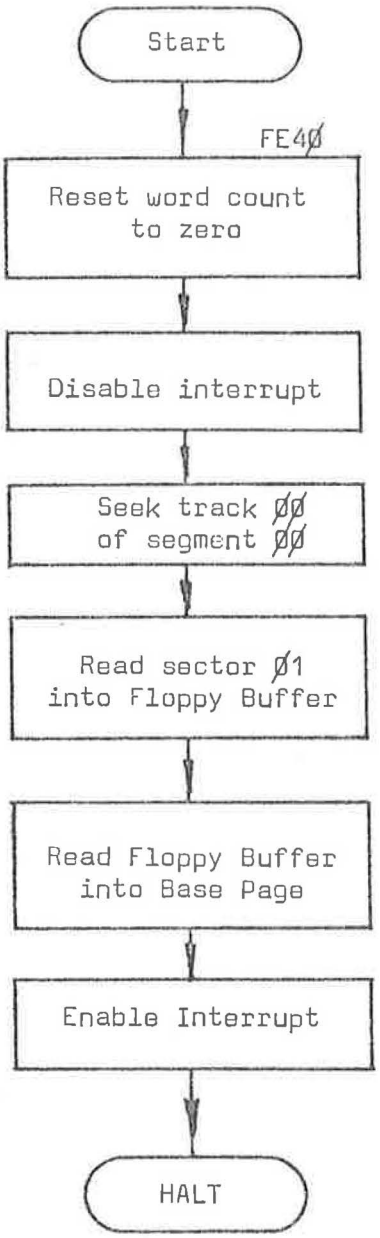


Fig 7.1 Flow Diagram for Calling Utility Routines

Code Word in FE72	Density	Bytes/Sector
8ØØØ	S	128
8ØØ1	S	256
8ØØ2	S	512
8ØØ3	S	1Ø24
8ØØ4	D	128
8ØØ5	D	256
* 8ØØ6	D	512
8ØØ7	D	1Ø24

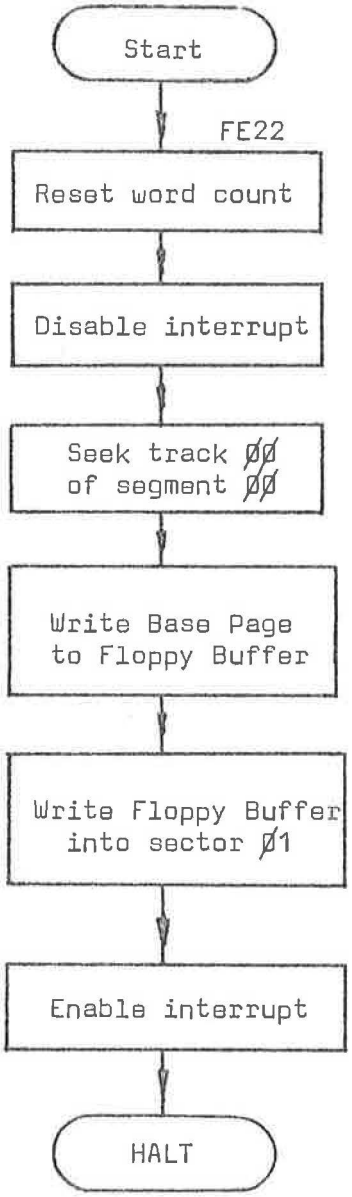
Table 7.2 Code Words for Different Formats

* Default Format



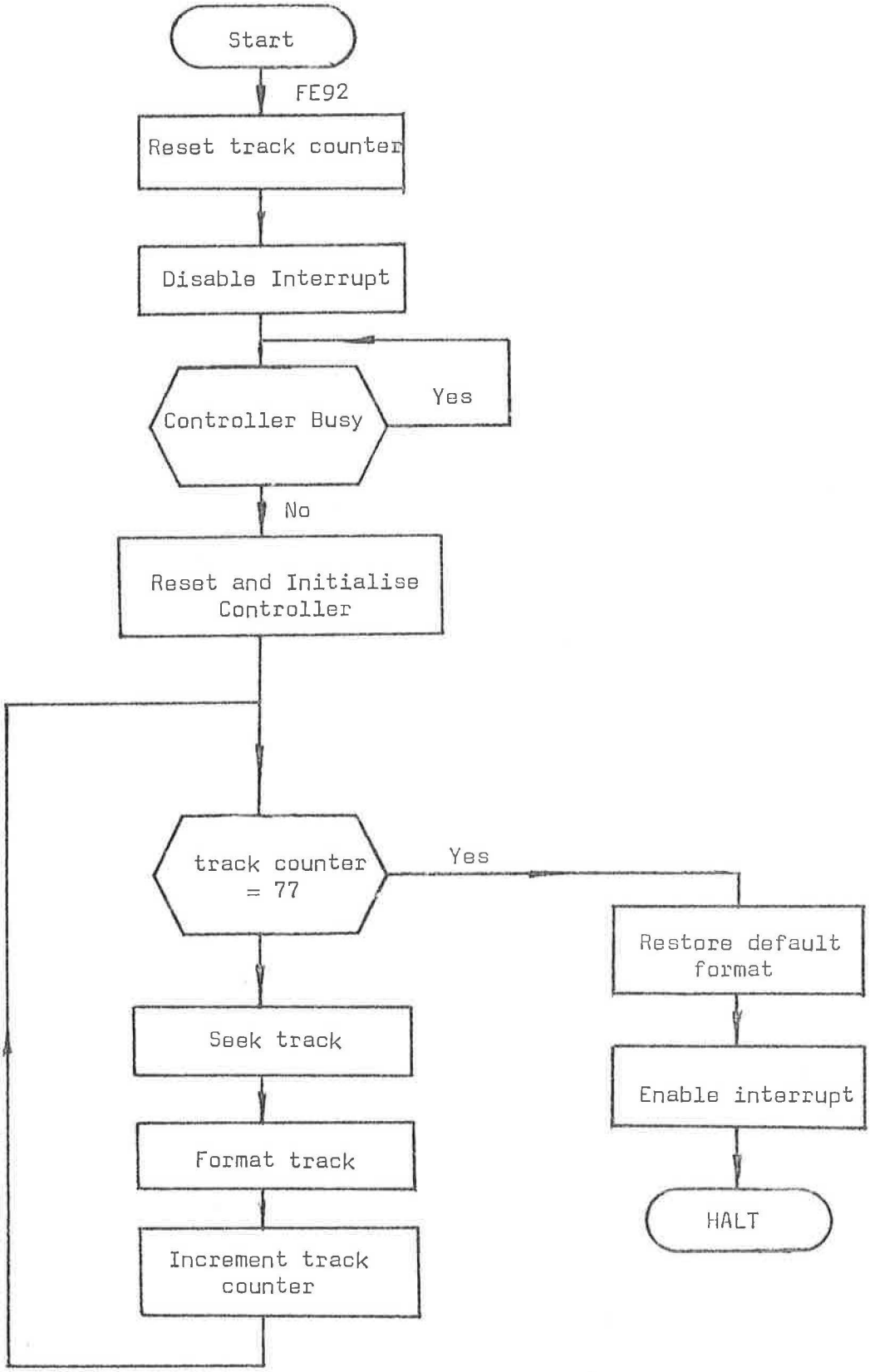
Read Image Flow Diagram

```
0000                    .TITLE  READ CONTROLWARE IMAGE
0000                    .ASECT
0000  0002              C2=2                    ;BIT 1 OF ACO IS 0
0000  0003              C3=3                    ;BIT  0 OF ACO IS 1
0000  0004              C4=4                    ;BIT  1 OF ACO = 1
0000  FF40              .=X'FE40
FF40  6525  ST:         LD 1,ZERO
FF41  A505              ST 1,PULL
FF42  2105              JMP COM
FF43  FF00              .WORD X'FF00            ;DISKETTE DEVICE ADDRESS
FF44  0100              .WORD X'0100            ;SEEK TRACK 00 COMMAND CODE
FF45  4201              .WORD X'4201            ;READ 1-8 AT SEC 1 COMMAND CODE
FF46  4201              .WORD X'4201            ;READ 9-M COMMAND CODE
FF47  0000  PULL:       .WORD X'0000            ;START ADDRESS OF READ BUFFER
FF48  0100  SEC:        .WORD X'0100            ;BUFFER WORD SIZE
FF49  0950  COM:        PFLG 1
FF4A  61F3              LD 3,ST+3               ;DISKETTE DEVICE ADDRESS
FF4B  0401              XIN 1                   ;CHECK FDC STATUS
FF4C  12FF              BOC C2,.-2
FF4D  61F5              LD 0,ST+5               ;SEEK TRACK 00 COMMAND TO BUFFER
FF4E  0805              XOUT 5                  ;SEEK TRACK 00
FF4F  0401              XIN 1                   ;CHECK FDC STATUS
FF50  12FE              BOC C2,.-2
FF51  61F3              LD 0,ST+3               ;SET UP READ COMMAND TO BUFFER
FF52  0805              XOUT 5                  ;READ
FF53  0401              XIN 1                   ;CHECK FDC STATUS
FF54  12FE              BOC C2,.-2
FF55  61F0              LD 0,ST+0               ;READ 1-8 COMMAND TO BUFFER
FF56  0808              XOUT 8                  ;READ
FF57  65F0              LD 1,SEC                ;SET UP END AND OF READ BUFFER
FF58  C5EE              ADD 1,PULL
FF59  49FF              ISZ 1,-1
FF5A  0400              XIN 0                   ;INITIATE INPUT BUFFER
FF5B  0401  ILOOP:      XIN 1                   ;CHECK INPUT STATUS
FF5C  1301              BOC C3,.+1
FF5D  21FD              JMP ILOOP
FF5E  0402  IN:         XIN 2
FF5F  61E7              ST 0,@PULL
FF60  79E5              ISZ PULL
FF61  FF25              SKNE 1,PULL
FF62  2101              JMP LSIN
FF63  21F7              JMP ILOOP
FF64  0402  LSIN:       XIN 2
FF65  61E1              ST 0,@PULL
FF66  0000  ZERO:       HALT
FF67  2000              JMP X'00
FF67                    .END
```



Write Image Flow Diagram

```
0000                    .TITLE  WRITE CONTROLWARE IMAGE
0000                    .ASECT
0000  0002              C2=2                ;BIT 1S OF AC0 IS 0
0000  0003              C3=3                ;BIT  0 OF AC0 IS 1
0000  0004              C4=4                ;BIT  1 OF AC0 = 1
0000  FE00              .=X'FE00
FE00  FF00  AD:         .WORD X'FF00        ;DISKETTE DEVICE ADDRESS
FE01  0100              .WORD X'0100        ;SEEK TRACK 00 COMMAND
FE02  8401              .WORD X'8401        ;WRITE 0-0 AT SECTOR 1
FE03  4401              .WORD X'4401        ;WRITE 0-1 AT SECTOR 1
FE04  4201              .WORD X'4201        ;READ  0-1 AT SECTOR 1
FE05  8201              .WORD X'8201        ;READ  1-0 AT SECTOR 1
FE06  0000  PUSH:       .WORD X'0000        ;START ADDRESS OF WRITE BUFFER
FE07  1200  PULL:       .WORD X'1200        ;START ADDRESS OF READ BUFFER
FE08  0100  SEC:        .WORD X'0100        ;SECTOR SIZE
FE09  0980  ST:         PFLG 1              ;DISABLE PANEL INTERRUPT
FE0A  8DF5              LD 3,AD
FE0B  0401              RIN 1               ;CONTROLLER STATUS READY
FE0C  12FE              BOC C2,.-2
FE0D  81F3              LD 0,AD+1           ;SEEK TRACK 00 COMMAND TO BUFFER
FE0E  0608              BOUT 8              ;SEEK
FE0F  85F6  RD:         LD 1,SEC            ;SET UP END ADDRESS
FE10  C5F5              ADD 1,PUSH
FE11  0401              RIN 1               ;CONTROLLER READY
FE12  12FE              BOC C2,.-2
FE13  81EE              LD 0,AD+2           ;WRITE R-W COMMAND TO BUFFER
FE14  0608              BOUT 8              ;WRITE
FE15  0401  XLOOP:      RIN 1               ;OUTPUT STATUS READY
FE16  14FE              BOC C4,.-2
FE17  91EE              LD 0,@PUSH
FE18  0604              BOUT 4              ;DATA OUT
FE19  74EC              ISZ PUSH
FE1A  F5E8              SKNE 1,PUSH
FE1B  2101              JMP XTD
FE1C  21F8              JMP XLOOP
FE1D  0401  XTD:        RIN 1               ;CONTROLLER READY
FE1E  12FE              BOC C2,.-2
FE1F  81E3              LD 0,AD+3           ;WRITE R-O COMMAND IN BUFFER
FE20  0608              BOUT 8              ;WRITE R-O
FE21  2103  DR:         JMP OUT
FE22  8502              LD 1,OUT
FE23  A5E2              ST 1,PUSH
FE24  21E4              JMP ST
FE25  0000  OUT:        HALT
FE26  2000              JMP X'00
FE26                    .END
```



Disk Format Flow Diagram

```
0000                    .TITLE FORMAT DISKETTE
0000                    .ASECT
0000   0002             C2=2                ;BIT 15 OF AC0 IS 0
0000   0003             C3=3                ;BIT  0 OF AC0 IS 1
0000   0004             C4=4                ;BIT  1 OF AC0 = 1
0000   FE70             .=X'FE70
FE70   FF00   AD:       .WORD X'FF00        ;DISK DEVICE ADDRESS
FE71   0000             .WORD X'0000        ;RESET COMMAND CODE
FE72   8006             .WORD X'8006        ;INITIATE COMMAND CODE 512 BYTE
FE73   0930   ST:       PFLG 1              ;DISABLE INTERRUPT
FE74   80FA             LD 3,AD             ;DISK ADD TO AC3
FE75   0401             RIN 1               ;READ DISK STATUS
FE76   12FE             BOC C2,.-2          ;CONTROLLER BUSY
FE77   81F9             LD 0,AD+1           ;RESET IN COMMAND BUFFER
FE78   0608             BOUT 8              ;RESET DISK
FE79   0401             RIN 1               ;READ STATUS
FE7A   12FE             BOC C2,.-2
FE7B   81F6             LD 0,AD+2           ;INITIATE COMMAND IN BUFFER
FE7C   0608             BOUT 8              ;INITIATE DISK
FE7D   0401             RIN 1
FE7E   12FE             BOC C2,.-2
FE7F   810C   SK:       LD 0,TK             ;SEEK COMMAND IN BUFFER
FE80   F10C             SKNE 0,SEVN
FE81   2100             JMP DFS
FE82   0608             BOUT 8
FE83   0401             RIN 1
FE84   12FE             BOC C2,.-2
FE85   8106             LD 0,TK
FE86   C107             ADD 0,FIVE
FE87   0508             BOUT 8              ;FORMAT TRACK
FE88   0401             RIN 1
FE89   12FE             BOC C2,.-2
FE8A   7901             ISZ TK
FE8B   21F3             JMP SK
FE8C   0100   TK:       .WORD X'0100
FE8D   0140   SEVN:     .WORD X'0140
FE8E   0500   FIVE:     .WORD X'0500
FE8F   8106   DFS:      LD 0, INIT
FE90   A1E1             ST 0, AD+2
FE91   2105             JMP OUT
FE92   8002             LD 3,ROUND
FE93   80FA             ST 3,TK
FE94   210E             JMP ST
FE95   0100   ROUND:    .WORD X'0100
FE96   8006   INIT:     .WORD X'8006,       ;DEFAULT FORMAT
FE97   0608   OUT:      BOUT 8              ;RESTORE DEFAULT FORMAT
FE98   0000             HALT
FE99   2000             JMP X'00
FE99                    .END
```

APPENDIX A

IMP-16C VSC SPECIAL INSTRUCTION SET

APPENDIX A

A microcycle of the IMP16C consists of eight system clock periods as shown in Fig. A.1. The basic instruction set requires from three to over ten execution cycles per instruction. The address of any memory reference instruction is sent out at T4 on the data bus and is latched in the Address Register. For memory read operations, data is expected back at T7 of the same microcycle. However, a memory write operation requires two microcycles; the first microcycle sends out the address to the Address Register at T4 and the data destined for the memory arrive at T4 of the following microcycle.

The high byte of the Address Register is compared with the mapping table (CAM) everytime a memory read or memory write instruction is issued. An interrupt signal is generated soon after T4 if the required page is not in the CAM. However this page fault interrupt signal will not be processed until T2 of the next instruction fetch cycle. To prevent the processor from referencing a location that is not in the main memory and causing unpredictable results, the main memory is disabled from the start of a page fault till the page fault interrupt is accepted. The data returned under this circumstance is FFFF. Some of the memory reference instructions involving the accumulators will be affected and unwanted results may arise. These instructions are listed in Table A.2 together with the equivalent instructions to overcome the problem.

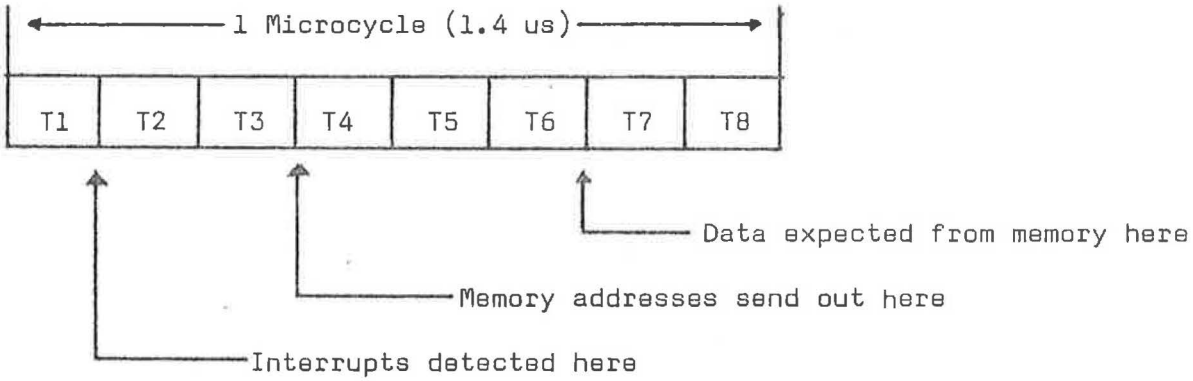


Fig. A.1 IMP16C Timing Control

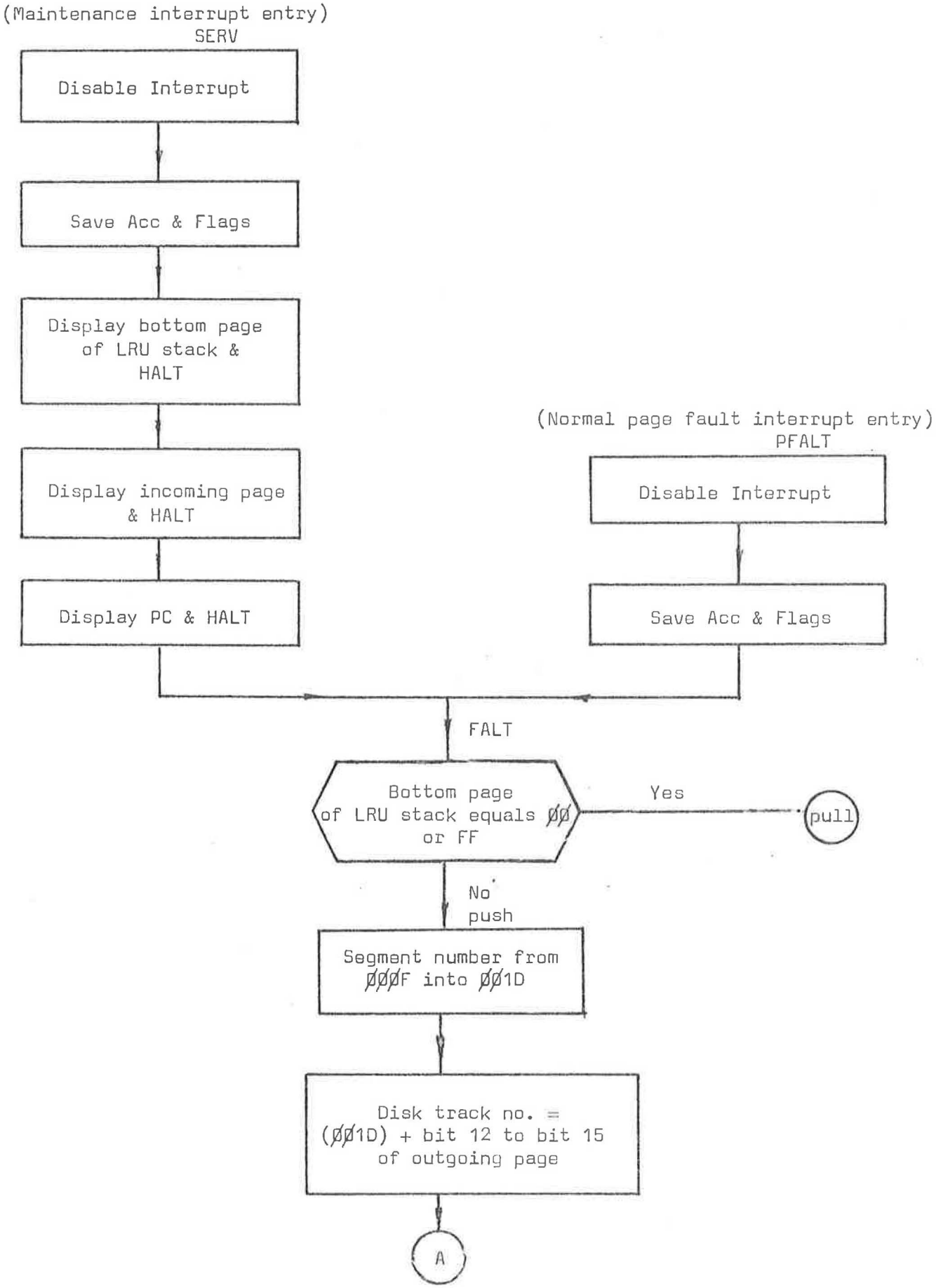
Operation	IMP16C Instructions	IMP16C VSC Instructions
ADD, SUB, OR, AND SKG, SKNE	Operation r, disp (Xr)	LD rs , disp (Xr) Operation r, disp (Xr)
ISZ	ISZ disp (Xr)	LD rs, disp (Xr) ISZ disp (Xr)
LD (Indirect)	LD r, @ disp (Xr)	LD rs, disp (Xr) LD r, 0 (rs)

rs is a spare accumulator

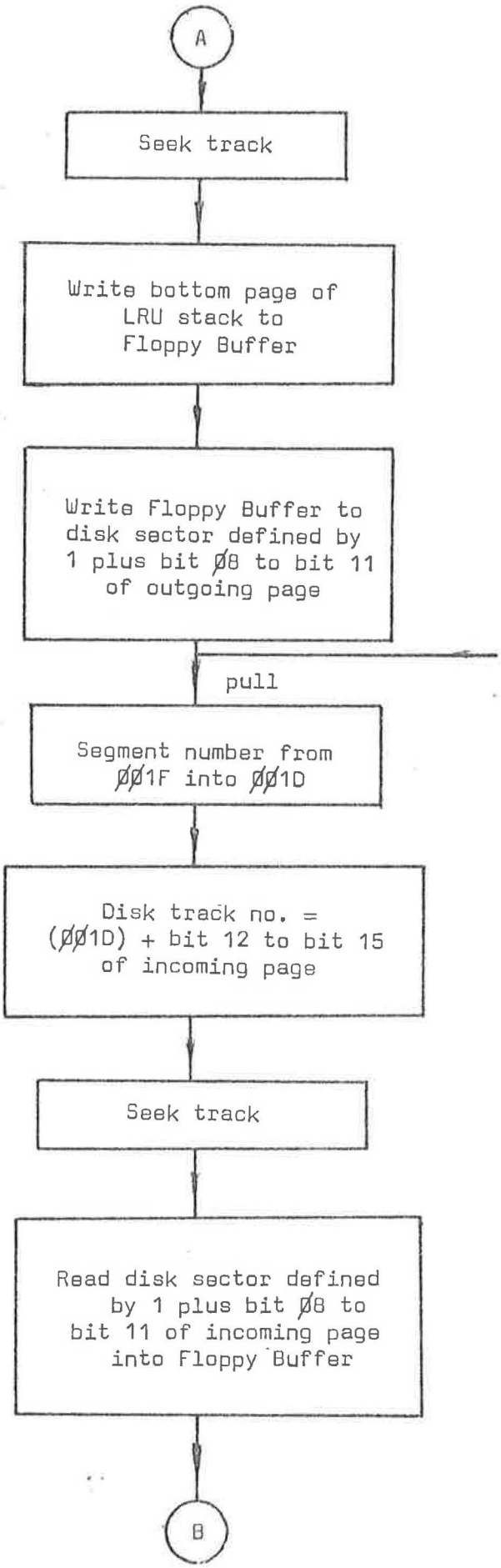
Table A.2 IMP16C VSC Special Instruction Set

APPENDIX B

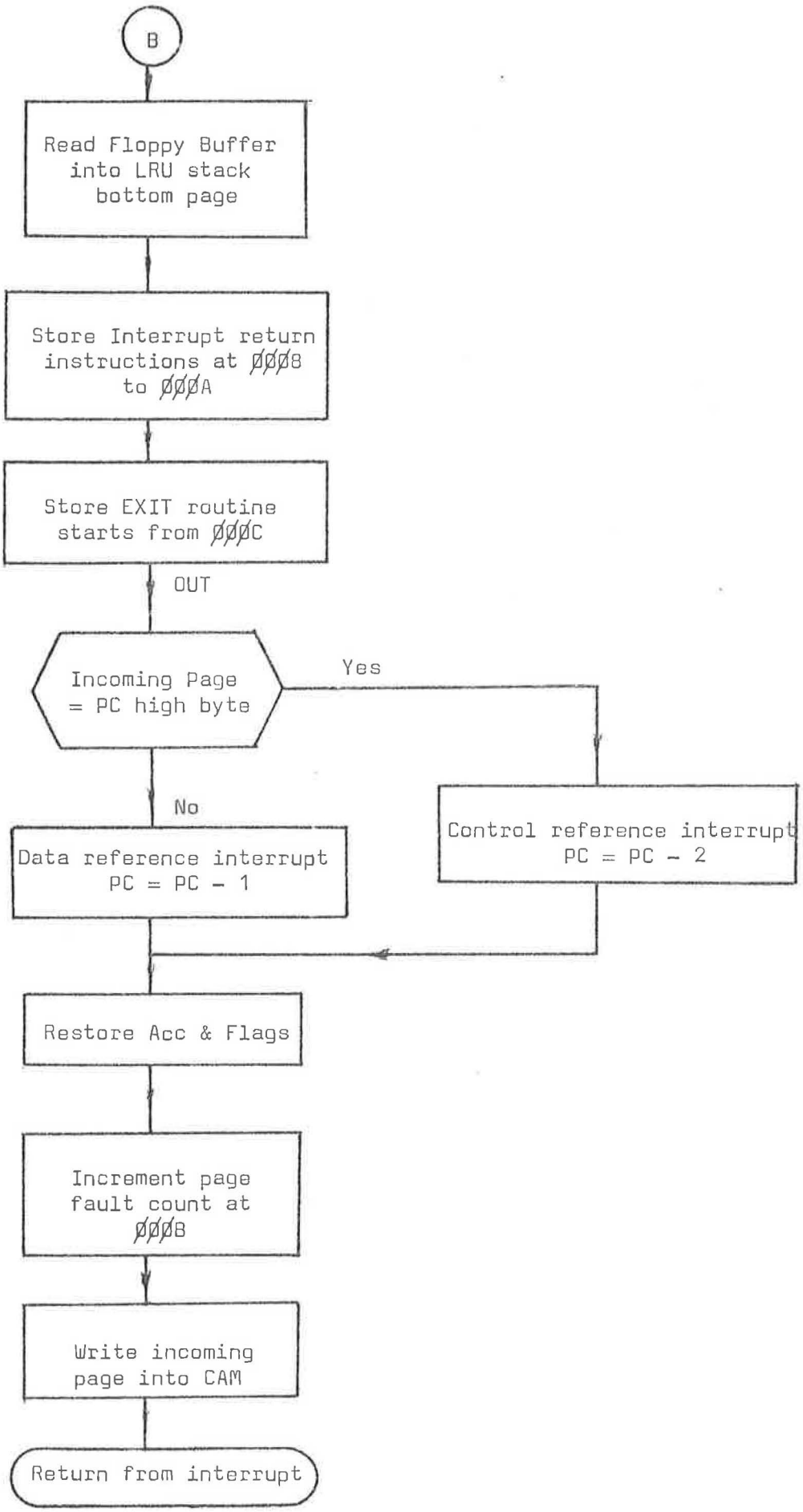
IMP-16C VSC CONTROLWARE FLOW DIAGRAM AND LISTING



IMP16C VSC Control Software Flow Diagram



IMP16C VSC Control Software Flow Diagram



IMP16C VSC Control Software Flow Diagram

IMP-16C VSC CONTROLWARE

```
0000                    .TITLE  IMP-16C VSC CONTROLWARE
0000                    .ASECT
0000  0002              C2=2                ;BIT 15 OF ACO IS 0
0000  0003              C3=3                ;BIT  0 OF ACO IS 1
0000  0004              C4=4                ;BIT  1 OF ACO = 1
0000  FFF7              ZERO=FFF7           ;ADDRESS FOR ZERO VALUE
0000  0020              .=X'0020
0020                    ;
0020                    ;VSC CONTROLWARE FOR NATIONAL SEMICONDUCTOR  IMP-16C
0020                    ; C.S.YAM   ELECTRICAL ENGINEERING DEPT.
0020                    ;             ADELAIDE UNIVERSITY            SEPT.1979
0020                    ;
0020                    ;MAINTANCE ROUTINE DISPLAYS CAM CONTENT. INCORRECT PAGE
0020                    ;ADDRESS AND PC WHERE PAGE FAULT OCCURS. TO CONTINUE
0020                    ;EXECUTE WITH PANEL SWITCHS SET TO NON ZERO
0020                    ;
0020  0980  SERV:   PFLG 1              ;DISABLE INTERRUPT
0021  2884              JSR SAVE            ;SAVE ACCUMULATORS
0022  8CCD              LD 3,AD             ;FLOPPY DISK DEVICE ADDRESS
0023  0410              RIN 10              ;READ AND DISPLAY OUTGOING PAGE
0024  0600              ROUT 0              ;ADDRESS FROM CAM
0025  0000              HALT
0026  0420              RIN 20              ;READ AND DISPLAY INCORRECT PAGE
0027  0600              ROUT 0              ;ADDRESS FROM ADDRESS LATCH
0028  0000              HALT
0029  4400              PULL 0              ;INTERRUPT PC FROM STACK
002A  4000              PUSH 0              ;RESTORE PC TO STACK
002B  0600              ROUT 0              ;DISPLAY PC
002C  0000              HALT
002D  2106              JMP FALT            ;JUMP TO DATA EXCHANGE
002D                    ;
002D                    ;FLOPPY DISC CONTROLLER STATUS CHECK ROUTINE
002D                    ;
002F  0401  DISC:   RIN 1               ;CHECK IF CONTROLLER IS BUSY
002F  12FE              BOC C2,.-2
0030  0200              RTS 0
0030                    ;
0030                    ;NORMAL PAGE FAULT INTERRUPT ENTRY
0030                    ;
0031  0980  PFALT:  PFLG 1              ;DISABLE INTERRUPT
0032  2884              JSR SAVE            ;SAVE ACC AND FLAGS
0033  8CCD              LD 3,AD             ;FLOPPY DISK DEVICE ADDRESS
0034  0410  FALT:   RIN 10              ;READ OUTGOING PAGE FROM CAM
0035  60CD              AND 0,AD            ;MASK OFF BITS 0-7
0036  70CD              SKAZ 0,AD           ;=00?
0037  F0CD              SKNE 0,AD           ;=FF?
0038  2113              JMP PULL            ;JUMP TO DATA IN SUBROUTINE
0038                    ;
0038                    ;SUBROUTINE PUSH RETURNS DATA FROM MEMORY TO DISK
0038                    ;
0039  880F  PUSH:   LD 1,X'0F           ;OLD SEGMENT NO. IN AC1
003A  A810              ST 1,X'10           ;OLD SEGMENT NO. IN  0010
003B  2967              JSR SEEK            ;SEEK TRACK
003C  88CD              LD 2,AD             ;VSC CONTROLLER DEVICE ADDRESS
003D  80C5              LD 0,WRM            ;LOAD MACRO COMMAND
003E  0608              ROUT 0              ;WRITE FROM HOST TO BUFFER
003F  0401  WLP:    RIN 1
0040  14FE              BOC C4,.-2          ;FDC OUTPUT STATUS READY?
0041  9007              LD 0,X'07           ;DATA WORD IN ACO
0042  0604              ROUT 4              ;DATA WORD TO INTERFACE
```

```
0043  7807          ISZ X'07        ;INCREMENT ADDRESS
0044  F807          SKNE 2,X'07
0045  2101          JMP RTD
0046  21F8          JMP WLP
0047  29E6  RTD:    JSR DISC        ;CHECK FDC STATUS
0048  3481          RCPY 1,0        ;SECTOR NO. IN AC0
0049  C17C          ADD 0,WBD       ;WRITE BUFFER TO DISK
004A  0608          ROUT 8
004B  29E2          JSR DISC        ;CHECK FDC STATUS
004B          ;
004B          ;SUBROUTINE PULL BRINGS DATA FROM DISK TO MEMORY
004B          ;
004C  881F  PULL:   LD 1,X'1F       ;NEW SEGMENT NO. IN AC1
004D  A81D          ST 1,X'1D       ;NEW SEGMENT NO. IN 001D
004F  0420          RIN 20          ;INCOMING PAGE IN AC0
004F  2953          JSR SEEK        ;SEEK TRACK
0050  3481          RCPY 1,0
0051  C175          ADD 0,RDB
0052  0608          ROUT 8          ;READ FROM DISK TO BUFFER
0053  290A          JSR DISC        ;CHECK FDC STATUS
0054  8173          LD 0,RBH
0055  0608          ROUT 8          ;READ FROM BUFFER TO HOST
0056  8572          LD 1,FEF
0057  0400          RIN 0           ;INITIALISE DATA INPUT BUFFER
0058  0401  ILOOP:  RIN 1           ;CHECK FDC INPUT STATUS
0059  1301          BOC C3,IN
005A  21FD          JMP ILOOP
005B  0402  IN:     RIN 2           ;DATA WORD FROM FLOPPY INTERFACE
005C  B007          ST 0,@X'07      ;STORE DATA WORD IN MEMORY
005D  7807          ISZ X'07        ;INCREMENT ADDRESS
005E  F407          SKNE 1,X'07
005F  2101          JMP LSIN
0060  21F7          JMP ILOOP
0061  0402  LSIN:   RIN 2           ;LAST DATA WORD
0062  B007          ST 0,@X'07
0063  812C          LD 0,WCM1       ;STORE RETURN FROM INTERRUPT
0064  A008          ST 0,X'08       ;ROUTINE IN 0008 - 000A
0065  812B          LD 0,WCM2
0066  A009          ST 0,X'09
0067  812A          LD 0,WCM3
0068  A00A          ST 0,X'0A
0069  8129          LD 0,WCM4       ;STORE EXIT ROUTINE ENTRY
006A  A00C          ST 0,X'0C       ;IN 000C
006B  8128          LD 0,WCM5
006C  A00D          ST 0,X'0D
006D  8127          LD 0,WCM6
006E  A00E          ST 0,X'0E
006F  8126          LD 0,WCM7
0070  A010          ST 0,X'10
0071  8125          LD 0,WCM8
0072  A011          ST 0,X'11
0073  8124          LD 0,WCM9
0074  A012          ST 0,X'12
0075  8123          LD 0,WCMA
0076  A013          ST 0,X'13
0077  8122          LD 0,WCMB
0078  A014          ST 0,X'14
0079  8121          LD 0,WCMC
007A  A015          ST 0,X'15
007B  8120          LD 0,WCMD
007C  A016          ST 0,X'16
007D  811F          LD 0,WCME
007E  A017          ST 0,X'17
```

```
007F  811E          LD 0,WCMF
0080  A018          ST 0,X'18
0081  811D          LD 0,WCM10
0082  A019          ST 0,X'19
0083  811C          LD 0,WCM11
0084  A01A          ST 0,X'1A
0085  811B          LD 0,WCM12
0086  A01B          ST 0,X'1B
0087  811A          LD 0,WCM13
0088  A01C          ST 0,X'1C
0089  2146          JMP OUT
0089  0090          .=X'0090
0090  0800  WCM1:   SFLG 8          ;WRITE INCOMING PAGE INTO CAM
0091  0880  WCM2:   PFLG 8
0092  0100  WCM3:   RTI 0           ;RETURN FROM PAGE FAULT INTERRUPT
0093  8DEA  WCM4:   LD 3,ZERO       ;EXIT ROUTINE ENTRY - RETURN
0094  AC1E  WCM5:   ST 3,X'1E       ;ADDRESS IS 0000
0095  2010  WCM6:   JMP X'10
0096  81E6  WCM7:   LD 0,ZERO
0097  A00B  WCM8:   ST 0,X'0B       ;RESET PAGE FAULT COUNTER
0098  8C07  WCM9:   LD 3,X'07       ;START CALLING IN PAGE FEFF
0099  800B  WCMA:   LD 0,X'0B       ;CHECK PAGE FAULT COUNTER
009A  48F0  WCMB:   ASZ 0,-16
009B  2104  WCMC:   JMP WCM11      ;COUNTER _ 16 MORE PAGING
009C  801F  WCMD:   LD 0,X'1F      ;COUNTER = 16 OLD SEGMENT EQUALS
009D  A00F  WCME:   ST 0,X'0F      ;NEW SEGMENT
009E  8C1E  WCMF:   LD 3,X'1E      ;DESTINATION ADDRESS IN AC3
009F  2300  WCM10:  JMP 0(3)       ;JUMP TO DESTINATION
00A0  8300  WCM11:  LD 0,0(3)
00A1  4BC0  WCM12:  ASZ 3,-40      ;PAGE = PAGE - 40
00A2  21F6  WCM13:  JMP WCMA
00A2        ;
00A2        ;SUBROUTINE SEEK POSITIONS HEAD TO REQUIRED TRACK
00A2        ;
00A3  0A80  SEEK:   PFLG 2         ;ENABLE SHIFT
00A4  5804          ROL 0,4        ;EXTRACT THE MOST SIGNIFICANT
00A5  3181          RCPY 0,1       ;4 BITS FOR TRACK NO.
00A6  2987          JSR DISC       ;CHECK FDC STATUS
00A7  3481          RCPY 1,0
00A8  6123          AND 0,F1       ;RETAIN TRACK NO. ONLY
00A9  C01D          ADD 0,X'1D     ;ADD SEGMENT NO. TO TRACK NO.
00AA  6123          AND 0,3F       ;MASK OFF UNWANTED SEGMENT BITS
00AB  C118          ADD 0,SK       ;ADD SEEK MACRO COMMAND
00AC  0608          ROUT 8         ;SEEK TRACK
00AD  811C          LD 0,FF        ;PHYSICAL MEMORY ADDRESS IN X'0007
00AE  A007          ST 0,X'07
00AF  5904          ROL 1,4        ;EXTRACT SECTOR NO. AND ADD ONE
00B0  651B          AND 1,F1
00B1  C519          ADD 1,ONE
00B2  282E          JSR DISC       ;CHECK FDC STATUS
00B3  0200          RTS 0          ;RETURN
00B3        ;
00B3        ;SUBROUTINE SAVE PRESERVES ACC AND FLAGS
00B3        ;
00B4  A002          ST 0,X'02      ;AC0 IN 0002
00B5  A403          ST 1,X'03      ;AC1 IN 0003
00B6  A804          ST 2,X'04      ;AC2 IN 0004
00B7  AC05          ST 3,X'05      ;AC3 IN 0005
00B8  0080          PUSHF
00B9  4500          PULL 1
00BA  A406          ST 1,X'06      ;FLAGS IN 0006
00BB  0200          RTS 0          ;RETURN
00BB        ;
```

```
00BB          ;SUBROUTINE RSTOR RESTORES ACC AND FLAGS
00BB          ;
00BC  8406    RSTOR:  LD 1,X'06       ;RESTORE FLAGS FROM 0006
00BD  4100            PUSH 1
00BE  0280            PULLF
00BF  8002            LD 0,X'02       ;RESTORE  AC0 FROM 0002
00C0  8403            LD 1,X'03       ;RESTORE  AC1 FROM 0003
00C1  8804            LD 2,X'04       ;RESTORE  AC2 FROM 0004
00C2  8C05            LD 3,X'05       ;RESTORE  AC3 FROM 0005
00C3  0200            RTS 0           ;RETURN
00C3          ;
00C3          ;MACRO COMMAND CODES AND CONSTANTS
00C3          ;
00C4  0100    SK:     .WORD X'0100    ;SEEK COMMAND WORD
00C5  8400    WHB:    .WORD X'8400    ;WRITE HOST TO BUFFER COMMAND
00C6  4400    WBD:    .WORD X'4400    ;WRITE BUFFER TO DISK COMMAND
00C7  4200    RDB:    .WORD X'4200    ;READ DISK TO BUFFER COMMAND
00C8  8200    RBH:    .WORD X'8200    ;READ BUFFER TO HOST COMMAND
00C9  FEFF    FEF:    .WORD X'FEFF
00CA  FE00    FE:     .WORD X'FE00
00CB  0001    ONE:    .WORD X'0001
00CC  000F    F1:     .WORD X'000F
00CD  FF00    AD:     .WORD X'FF00
00CE  003F    3F:     .WORD X'003F
00CF  0000    HALT:   .WORD X'0000
00CF          ;
00CF          ;PROGRAM SECTION TO DETERMINE INTERRUPT IS CAUSED BY
00CF          ;DATA REFERENCE OR BY CONTROL REFERENCE  ALSO MAINTAINS
00CF          ;PAGE FAULT COUNTER
00CF          ;
00D0  4400    OUT:    PULL 0          ;PC FROM STACK TO AC0
00D1  3181            RCPY 0,1        ;INTERRUPT ADDRESS IN AC1
00D2  0420            RIN 20          ;INCOMING PAGE IN AC0
00D3  61F9            AND 0,AD        ;MASK OFF BITS 0-7 OF INCOMING ADD
00D4  3281            RCPY 0,2        ;INCOMING PAGE IN AC2
00D5  3481            RCPY 1,0        ;INTERRUPT ADDRESS BACK IN AC0
00D6  65F6            AND 1,AD        ;MASK OFF BITS 0-7 OF INTERRUPT ADD
00D7  3982            RXOR  2,1       ;INCOMING PAGE EQUALS INTERRUPT PAGE
00D8  75F4            SKAZ  1,AD      ;YES - CONTROL REFERENCE INTERRUPT
00D9  2106            JMP DINT        ;NO  - DATA TRANSFER INTERRUPT
00DA  48FE            AISZ 0,-2       ;CONTROL REFERENCE INTERRUPT PC=PC-2
00DB  4000    INAD:   PUSH 0          ;RESTORE PC TO STACK
00DC  29DF            JSR RSTOR       ;RESTORE  ACC AND FLAGS
00DD  7808            ISZ FCNT        ;INCREMENT PAGE FAULT COUNTER
00DE  2008            JMP X'08        ;JUMP TO RETURN FROM INTERRUPT
00DF  2008            JMP X'08
00E0  48FF    DINT:   AISZ 0,-1       ;DATA REFERENCE INTERRUPT PC=PC-1
00E1  21F9            JMP INAD
00E1                  .END
```

APPENDIX C

WIRING SCHEDULE AND PART LIST

Wiring Schedule

Signal \ Board	4k RAM J2	CPU J1	Interface J6	Front Panel	VSC Controller J4	4k RAM J2
GND	1	1	1	E1	1	
"	2	2	2		2	
"	3	3	3		3	
"	4	4	4		4	
+5V	5	5	5	E2	5	
"	6	6	6		6	
"	7	7	7		7	
"	8	8	8		8	
		9	9		9	
ADXØØ	63	1Ø	1Ø			
-12V		11		E48		
-12V		12	16		12	
-9V		13	13			
-9V		14	14			
INTR		15	15			
INIT*	135	16	17	E3	4Ø	64
CQØ		17	18		39	62
ADXØ7	66	18	19	E4	44	65
C7		19	2Ø		45	66
ADXØ6	65	2Ø	3Ø	E1Ø		
CSØ	78	21	22		21	
MEMAD*		26			46	
ADXØ2	88	23	23			
ADXØ1	86	24	24			
ADXØ3	85	25	25			
		22	26	E7		
BDOØ8	37	27	27		42	
ADXØ5	62	28			28	
ADXØ4	64	29			29	
C3B*		3Ø	29			
-12V		31	31	E46		
-12V		32	32	E46		
		33	33	E11		
CS1	82	34	4Ø			

Signal \ Board	4k RAM J2	CPU J1	Interface J6	Front Panel	VSC Controller J4	4k RAM J2
CS2	81	35	41		41	
		36	36	E13		
		37	37	E2Ø		
		38	38	E21		
		39	39	E23		
MD0ØØ	35	4Ø	34	E12		
MD0Ø1	49	41	35	E14		
		42	42	E22		
MD0Ø2	17	43	43		43	
MD0Ø3	21	44	44		47	
		45	45	E28		
		46	46	E29		
MD0Ø4	42	47	47		1Ø5	
MD0Ø5	45	48	48		48	
		49	49	E31		
MD0Ø6	18	5Ø	5Ø		53	
		51	51	E3Ø		
MD0Ø7	22	52	52		52	
WRPB		53	53	E41		
		54	54	E45		
		55	55	E35		
BD0Ø5	54	56	56			
DISTR*	1Ø6	57	57	E34		
BD0Ø1	4Ø	58	58			
BD0Ø4	43	59	59			
BD0ØØ	36	6Ø	6Ø			
BD0Ø3	2Ø	61	61			
MD014	48	62	62	E15		
BD0Ø7	23	63	63			
BD0Ø2	58	64	64			
BD0Ø6	19	65	65			
WRPA		66	66		51	
BD0Ø9	41	67	67		38	
BD011	52	68	68		3Ø	
BD014	47	69	69		18	
BD015	59	7Ø	7Ø		14	
GND	71	71	71		71	
GND	72	72	72		72	

Signal \ Board	4k RAM J2	CPU J1	Interface J6	Front Panel	VSC Controller J4	4k RAM J2
BDO1Ø	51	73	73		34	
BDO13	73	74	74		22	
BDO12	74	75	75		26	
MDO1Ø	46	76	76	E33		
MDO13	39	77	77	E32		
MDO12	28	78	78	E27		
SWØØ		79	79			
		8Ø	8Ø	E26		
SWØ2		81	81			
MDOØ8	25	82	82	E24		
CI	1Ø4	83	83	E25		
SWØ1		84	84			
INTCTL		85			127	
SWØ3		86	86			
MDOØ9	38	87	87	E18		
MDO11	55	88	88	E17		
		89	89	E16		
SWØ8		9Ø	9Ø		1Ø8	
SW1Ø		91	91		11Ø	
SWØ9		92	92		1Ø9	
SW11		93	93		111	
CLK 81	1Ø5	94	94	E9		
SW12		95	95		112	
SW13		96	96		113	
SW14		97	97		114	
SWØ4		98	98			
C45		99	99	E44	121	
SWØ6		1ØØ	1ØØ			
ADX13		1Ø1	1Ø1			
EXHOLD	122	1Ø2	1Ø2	E8		
SWØ5		1Ø3	1Ø3			
JC14		1Ø4	1Ø4			
SWØ7		1Ø5	1Ø5			
SW15		1Ø6	1Ø6		115	
JC15		1Ø7	1Ø7			
MDO15	57	1Ø8	1Ø8	E6		
ADX11		1Ø9	1Ø9		37	1Ø3
ADX1Ø		11Ø	11Ø		35	77

Signal \ Board	4k RAM J2	CPU J1	Interface J6	Front Panel	VSC Controller J4	4k RAM J2
ADXØ8		111	111		54	6Ø
ADXØ9		112	112		49	5Ø
RDM	113	113	113	E5		
ADX12		114	114			
ADX14		115	115			
ADX15		116	116			
CPINP		117			117	
CPINT	124	118	118		118	
JC13		119	119		119	124
WRMP	93	12Ø	12Ø		12Ø	
RDP		121	121			
WRP		122	122			
EXEC		123	123,21	E37		
		124	124		124	
HALT		125		E4Ø		
SYSCLR		126		E43		
CLK*	121	127	127			
CLK		128	128		5Ø	
JC12		129	129			
F8		13Ø			135	
INTEN		131	131		128	
		132	132		132	
F11		133	133			
		134	134		134	
		135	135		123	
F12		136	136			
+5V	137	137	137		137	
"	138	138	138		138	
"	139	139	139		139	
"	14Ø	14Ø	14Ø		14Ø	
GND	141	141	141		141	
"	142	142	142		142	
"	143	143	143		143	
"	144	144	144		144	
			125		125	
			126		126	
			13Ø		13Ø	
			85	E19		
			117	E36		

Board / Signal	4k RAM J2	CPU J1	Interface J6	Front Panel	VSC Controller J4	4k RAM J2
	(+5V) 68,114 118, 119,120 123, 134. (GND) 11, 14 69, 83, 115, 117.		12 ——— E39 11 ——— E38		(-5.2V) 23, 25, 27, 31 (GND) 11, 15, 129, 131, 133.	

Note: The following modifications apply to the CPU board.

1. Jumper W6 is removed for EXHOLD.
2. Jumper pad W4 is connected to connector pin 83 for Memory Cycle Initiate (CI).
3. Jumpers W1, W2 and W3 are removed for external CSØ, CS1 and CS2 Memory Chip Selects.

VSC CONTROLLER PART LIST

ECL

Type No.	Function	Quantity
10124	TTL/ECL Converter	7
10102	Quad. 2 IP NOR	4
10165	8 to 3 Encoder	2
10158	Quad. 2 IP MUX	1
10125	ECL/TTL Converter	4
10155	CAM	8

TTL

Type No.	Function	Quantity
DM 8551	Quad. D F/F	19
DM 8200	4 Bit Comparator	1
SN 7404	Hex Inverter	5
SN 74150	16 to 1 MUX	4
SN 7493	Binary Counter	1
SN 74164	Shift Register	2
SN 74125	Tri-state Buffer	5
SN 74154	4 to 16 Decoder	1
SN 7473	Dual JK F/F	1
SN 7408	Quad. 2 IP AND	2
SN 7432	Quad. 2 IP OR	4
SN 7400	Quad. 2 IP NAND	1
SN 7402	Quad. 2 IP NOR	1
SN 7411	Triple 3 IP AND	1
SN 74193	Binary Down Counter	1

FLOPPY DISK INTERFACE PART LIST

TTL

Type No.	Function	Quantity
DM 7542	Tri-state Quad. I/O Registers	8
DM 74126	Tri-state Quad. Buffer	1
DM 74175	Quad. D F/F with clear	1
DM 7476	Dual J/K F/F	1
SN 7400	Quad. 2 I/P NAND	4
SN 7408	Quad. 2 I/P AND	6
SN 7486	Quad. XOR	1
SN 7432	Quad. 2 I/P OR	3
SN 7411	Triple 3 I/P AND	2
SN 74157	Quad. 2 I/P Multiplexer	4
SN 74367	Tri-state Hex Buffer	3
SN 74S30	8 I/P NAND	1