0862003-003

# VULCAN

# CONCEPTS AND FEATURES

Original Issue
March, 1977

Revision A
December, 1977

Revision B
May, 1978


COMMUNICATIONS AND
INFORMATION HANDLING

**HARRIS CORPORATION** Computer Systems Division
1200 Gateway Drive, Fort Lauderdale, Florida 33309 305/974-1700

# LIST OF EFFECTIVE PAGES

TOTAL NUMBER OF PAGES IN THIS PUBLICATION IS: 72
CONSISTING OF THE FOLLOWING:

| Page No. | Change No. | Page No. | Change No. | Page No. | Change No. |
|---|---|---|---|---|---|
| Title | Rev B | | | | |
| A | Rev B | | | | |
| B | Original | | | | |
| i | Rev B | | | | |
| ii, iii | Original | | | | |
| iv, v | Rev B | | | | |
| 1-1 | Rev B | | | | |
| 1-2, 1-3 | Original | | | | |
| 1-4, 1-5 | Rev B | | | | |
| 2-1 thru 2-5 | Rev B | | | | |
| 3-1 | Original | | | | |
| 3-2, 3-3 | Rev B | | | | |
| 4-1 thru 4-6 | Rev B | | | | |
| 4-7, 4-8 | Original | | | | |
| 4-9 | Rev B | | | | |
| 5-1 thru 5-5 | Rev B | | | | |
| 6-1 | Original | | | | |
| 6-2 | Rev B | | | | |
| 6-3 | Original | | | | |
| 6-4, 6-5 | Rev B | | | | |
| 6-6, 6-7 | Original | | | | |
| 6-8, 6-9 | Rev B | | | | |
| 6-10 | Original | | | | |
| 6-11 | Rev B | | | | |
| 7-1, 7-2 | Rev B | | | | |
| 7-3 | Original | | | | |
| 7-4 thru 7-6 | Rev B | | | | |
| 8-1, 8-2 | Rev B | | | | |
| 8-3 | Original | | | | |
| A-1 thru A-4 | Rev B | | | | |
| A-5 | Original | | | | |
| A-6 thru A-9 | Rev B | | | | |
| B-1, B-2 | Rev B | | | | |
| B-3 | Original | | | | |
| I-1 thru I-5 | Rev B | | | | |

Insert Latest Revision Pages. Destroy Superceded Pages.

## PROPRIETARY DATA

# PREFACE

VULCAN is the operating system for Harris Virtual Memory Computer Systems. This manual describes what VULCAN can do and how well it does it and should be read by those who are evaluating a Harris Virtual Memory System for purchase or who are preparing to be VULCAN users.

If you are evaluating computer systems, you are probably more interested in the "what's" of VULCAN rather than the "how's". And you may be more interested in the way VULCAN manages users rather than the way users manage VULCAN. If indeed these are your interests, read Section I (VULCAN — An Overview), Section III (User Management) and Section VII (System Operation) in their entirety. Depending upon the types of processing you need, the following paragraphs of Section II (Processing Modes) are recommended: Interactive Processing, Batch Processing, Job and Program Sequencing, and What Is a Real Time Program.

If you still want more information, the subsection entitled Job Control Example, Table 4-2 (Job Stream File Example), and the paragraph on Job Control Macros, all in Section IV, will give you an idea of the capabilities and the ease of use of VULCAN's interactive command language. The first paragraph of Section V has some important information on VULCAN's language processors. Figures 6-2, 6-3, and 6-5 in Section VI illustrate the efficiency of VULCAN's disc storage techniques. If you are concerned about the security of your disc files, read Disc Area Security in Section VI. If you are interested in data base management, read the first paragraph of Section VII on the TOTAL Data Base Management System and the paragraphs on Security and Backup and Recovery. If you have a requirement for Remote Job Entry or VULCAN's Remote Job Hosting capabilities, read Remote Job Entry (RJE) Subsystems in Section VII.

For programmers and analysts, this manual describes what tools VULCAN offers for the solution of your tasks. Read Section I through IV and Section VII in their entirety. In Section V, Programming Facilities, the first three paragraphs are of general interest to all programmers and analysts. Look over the rest of the chapters to see what other programming facilities you can use. Section VII, Optional Software, describes the TOTAL Data Base Management System and Remote Job Entry Subsystems. Read these paragraphs if they are applicable to your installation.

For more detailed discussions of the VULCAN operating system and its features, consult the following manuals: VULCAN Job Control and System Processors (Publication No. 0860002), VULCAN Operator Communications — OPCOM (Publication No. 0860005), VULCAN System Services (Publication No. 0860003), VULCAN I/O Service (Publication No. 0860004), and VULCAN GENASYS (Publication No. 0860006). If you need details on using a language processor, Harris has a reference manual for each language it supports. Assembly language programmers should also have the appropriate CPU reference manual for their hardware configuration. Finally, there are manuals on Harris TOTAL (Publication No. 0862002), Harris TOTAL Interactive Questioner IQ (Publication No. 0862005), and RJE under VULCAN (Publication No. 0863001).

# CONTENTS

## CONTENTS (CONT'D.)

## CONTENTS (CONT'D.)

## CONTENTS (CONT'D.)



## ILLUSTRATIONS



## TABLES

# I. VULCAN
# AN OVERVIEW

## 1-1 WHAT IS VULCAN?

VULCAN is a multi-use, multi-programming, multi-lingual virtual memory operating system designed to make the best use of the hardware features of Harris Virtual Memory Computer Systems.

VULCAN is a *multi-use* operating system. Because of its wide variety of capabilities and flexibility, it is an excellent operating system for general scientific, commercial and time-sharing applications.

VULCAN is a *multi-programming* operating system. It supports concurrent operation of interactive time-sharing, multiple batch and real-time processing as well as the TOTAL Data Base Management System, remote job entry and remote job hosting.

VULCAN is a *multi-lingual* operating system. It supports seven language processors, all of which are concurrently available to interactive, batch and real-time processing. Programs written in any set of languages can all execute concurrently in any combination of processing modes.

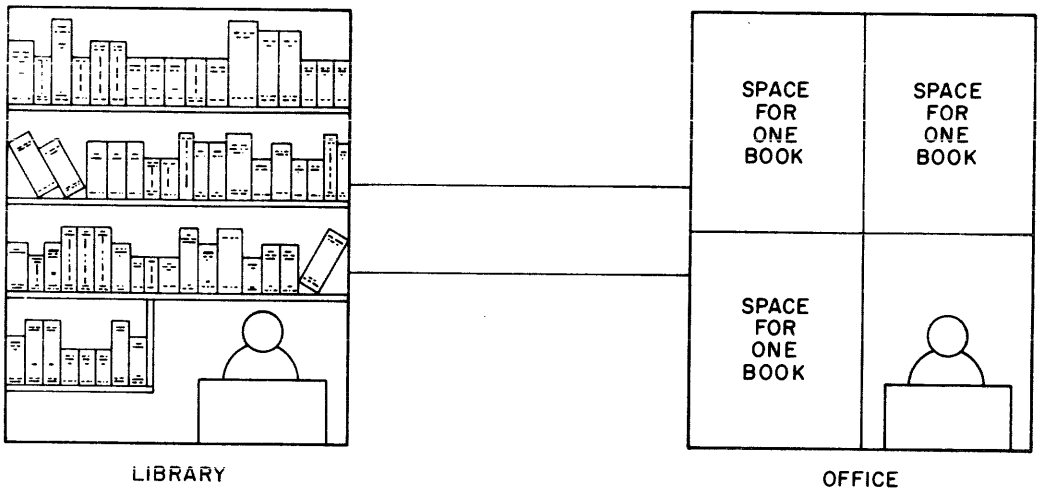
VULCAN is a *virtual memory* operating system. In addition to its other operating system functions, VULCAN works in conjunction with the virtual memory hardware in keeping track of and directing memory allocation.

## 1-2 WHAT IS VIRTUAL MEMORY?

Virtual memory is a method by which one or more programs can execute in less real memory than the combined sizes of the programs. How this works can be explained by means of an analogy involving a research project.

A researcher has a very small office and is using a 26-volume encyclopedia for reference. However, the office is so small that it can only accommodate a maximum of three volumes at one time. Whenever a book is required, the researcher goes to a nearby library and gets the specific volume from the library. See the illustration on the next page.

At the beginning of a new project, the office is empty. The first book that is needed is volume five which is requested from the librarian and placed in a corner of the office. After a few minutes of reading, a reference to volume fifteen is encountered, so that book is obtained and is placed in another available corner. A little later on, volume eleven is sought and placed in the only remaining corner of the office. After another hour passes, the researcher finds that once again another volume must be requested, only this time there is no room for it. The solution is to "swap" one of the books already in the office for a new one. Since volume five has not been used for quite a while, it is selected to be "swapped out".

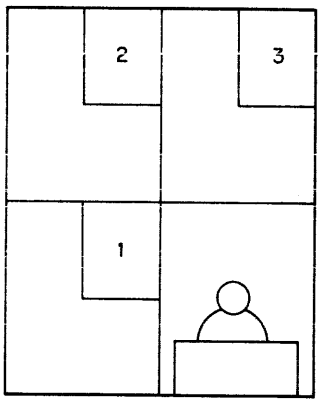



**The Researcher's Office**

In a virtual memory computer, instead of encyclopedias there are programs. These are not broken up into volumes, but rather into units called "pages". (Unlike the encyclopedia, however, all "pages" are the same size – 1,024 computer words in a VULCAN system.) The office is the computer's memory and requesting a page that is not in memory is known as a "demand page". The library is analogous to a disc pack. The researcher is the Central Processing Unit (CPU) and both the CPU and the swapping of memory pages are directed by VULCAN.

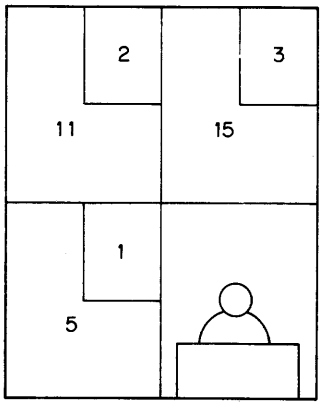
## 1-3 ADDITIONAL VIRTUAL MEMORY CONCEPTS

To go back to the analogy, each corner in the office which can accommodate a book has a number by which the corner can be identified. These numbers, illustrated below, are called real, or physical, addresses.




**Real/Physical Addresses**

Each volume of the encyclopedia also has an identifying number called a logical, or virtual, address. Logical addresses are independent of physical addresses since it doesn't matter in which corner a book is placed. In other words, any (logical) page of a program may reside in any (physical) page of memory that is available. The illustration shows this relationship; volume eleven (virtual/logical address = 11) resides in corner two (real/physical address = 2).




**Logical Addresses vs. Physical Addresses**

The analogy demonstrates how a program can be executed when the physical size of memory is less than the logical size of the program. And had the researcher been working on two projects concurrently, books from another encyclopedia could have been swapped in just as easily — just as VULCAN can swap pages in and out that belong to different programs.

For a more technical description of VULCAN's addressing and paging techniques, see Appendix B.

## 1-4 FEATURES OF THE HARRIS VIRTUAL MEMORY IMPLEMENTATION

Not all virtual memory systems are alike. Because of an extremely efficient and flexible virtual memory implementation in both hardware and software, the Harris Virtual Memory System has several particularly advantageous features.

*High-speed program loading:* Program loading is little more than a simple matter of copying program pages from disc directly into memory with no modification. The Harris virtual memory hardware associates each logical address in the program page with its physical location.

*Hardware/software memory protection:* Every program page in memory is protected against access or inadvertent destruction by another concurrently executing program. In addition, pages containing instructions and constants, as opposed to variable data, are hardware write-protected, even within the same program.

*Extensive logical memory:* Over 3.1 megabytes of logical address space are available for running application programs regardless of the physical memory available or the size of the operating system.

*Efficient use of memory:* A program need not occupy contiguous memory. Any physical page not occupied by the operating system can hold any logical page. Memory, therefore, never needs reorganizing or compacting.

## 1-5 APPLICATIONS

All of VULCAN's facilities provide a wide range of problem solving capabilities for all applications.

For applications that are scientifically oriented, VULCAN supports ANSI-standard FORTRAN IV and BASIC. Both have numerous extensions. The powerful Harris Macro Assembler supports over 600 instruction mnemonics including bit and byte instructions and extensive pseudo-operations as well. VULCAN contains a number of real-time processing services which are callable from all three of these languages. For those installations that do extensive mathematical computation (number crunching), VULCAN supports the Scientific Arithmetic Unit (SAU), a high-speed floating point processor which has forty-seven of its own instructions. For those instances where an interface to other computer installations is required, VULCAN supports several remote job entry (RJE) protocols.

For commercially oriented applications, VULCAN supports COBOL, RPG II, and the TOTAL Data Base Management System. Since VULCAN hosts several RJE protocols, it can be used to offload other computers in a large company or can itself be offloaded by submitting jobs to other computers via RJE.

In the educational world, VULCAN can concurrently process administrative functions, such as class scheduling, in batch while students are developing programs interactively, and periodic samples of experimental data are being taken in real-time. The wide variety of programming languages includes SNOBOL IV and FORGO. The latter is a diagnostic load-and-go ANSI-standard FORTRAN compiler which is particularly useful for students who are learning the language.

In a time-sharing environment, be it scientifically, commercially, or educationally oriented, VULCAN can support up to 128 local and remote interactive terminals, depending on the hardware/software configuration. For unsharable devices, such as magnetic tape drives, peripheral resourcing is provided. Spooling for devices such as line printers and card readers is automatic, and the spool queues may be manipulated and controlled by the console operator and the interactive user.

In many applications, user management is an important consideration. It involves controlling what each user shall be permitted to do and keeping track of what each user has done. VULCAN provides the capability to place restrictions on users in such categories as execution priorities, program size, access to monitor services, access to peripheral devices, and use of the operator's console. VULCAN automatically keeps records on each user's accumulated CPU and connect time and peripheral usage and does not let the user exceed his limits on computer and disc space utilization. An accounting utility is also provided which may be used to generate system utilization reports and for calculating charges to users, departments, etc.

## 1-6 USABILITY

In some computer systems, the value of good features is diminished because the features are difficult to use. Such is not the case with VULCAN. It has a powerful, interactive command language which uses common English words and is easy to learn. The command language, which is available to both interactive and batch users, includes a wide variety of text editing commands which help reduce the time required for program development and the maintenance of data. All commands are entirely device independent making it unnecessary to have different forms of commands to interface with different peripherals.

Commonly executed Job Control procedures may be stored on and executed off of disc files. Some such procedures, called Job Control Macros, become, in effect, new Job Control commands. Job Control Macros, which are extremely easy to implement, may be used to tailor the Job Control language to meet the specific requirements of an installation. Diagnostic commands are helpful both for program debugging and for learning to use the system. (A separate processor, VBUG, provides an interactive symbolic capability for high-level and Assembler languages – see "VBUG (VULCAN Symbolic Debugger".)

## 1-7 REENTRANCY – THE SHARING OF CODE

It is extremely easy for VULCAN to produce reentrant code, i.e., code that can be shared concurrently by different people. It's so easy, in fact, that all language processors and support software under VULCAN are reentrant. In addition, FORTRAN, COBOL, BASIC and RPG II automatically generate reentrant code. Through the use of simple assembly language pseudo-operations, assembly language programs can also be made reentrant. Physical memory pages that contain reentrant code are hardware write-protected.

To explain the concept of reentrancy in more detail, let us rejoin the researcher.

The researcher is now working on two jobs, Project A and Project B. The same encyclopedia is being used for both projects. While working on Project A, the researcher had occasion to refer to volume three which is still in the office. Now, work is being done on Project B and the same volume, volume three, is needed. The researcher knows that he can use the same book; he doesn't have to bring in another copy. After all, the text hasn't changed. It is said to be reentrant because any researcher on any project can "enter" it when required and get the same information. Of course, this would not be true if any of the projects involved modifying the book. In that case, the book would be non-reentrant.

Program pages, then, that are not modified can be used concurrently by different users calling the same program. This sharing of program pages has two advantages. First of all, it saves on memory requirements by eliminating the need for multiple copies of the same machine instructions or fixed data to be loaded. This leads to a double saving in time – unnecessary program loading is eliminated and fewer demand pages are generated resulting in less swapping.

# II. PROCESSING MODES

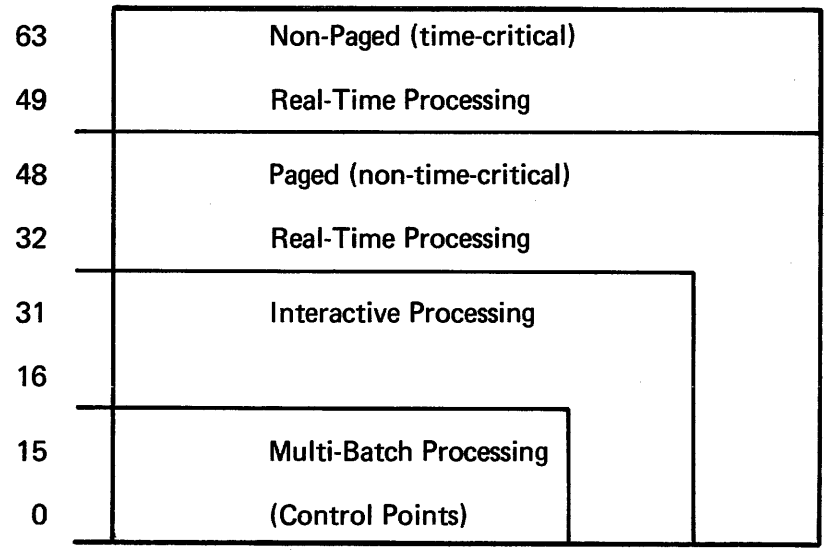
## 2-1 INTRODUCTION

VULCAN supports three processing modes, all of which are available concurrently. These processing modes are interactive processing, batch processing and real-time processing. The primary way that VULCAN distinguishes between the modes is by software priority. It is software priorities that ensure that high priority jobs such as those submitted by interactive users can be serviced before batch jobs and that real-time requirements are met before those of the interactive and batch modes.

## 2-2 THE SOFTWARE PRIORITY STRUCTURE

VULCAN supports up to 128 software priorities. They are used by the operating system to determine which program or interactive terminal will have control of the CPU at any one time. Programs with a high priority are allocated to the various processing modes in a way that best reflects the requirements of any particular installation. Table 2-1 is an example of a typical software priority layout.

**Table 2-1. Sample Software Priority Layout**




Whenever programs of equal priority are executing concurrently, the technique of time-slicing is used to allocate CPU time. A time slice is a small unit of time which is allocated sequentially to each of the equal-priority programs. Whenever one of these programs either uses up its time slice or is in a state of suspension (as when waiting for an I/O function to be completed), control passes to another program of equal priority that is requesting CPU time. The duration of a time slice is selected when a VULCAN system is generated. A typical value is 1/60th of a second.

## 2-3 INTERACTIVE PROCESSING

Interactive processing permits the user to communicate with VULCAN on-line through an interactive terminal. Interactive users can easily invoke either of the other processing modes when they are needed. It is the interactive mode which gives VULCAN its time-sharing capability.

An interactive terminal user has access to all language processors, support software and peripherals; can initiate batch jobs, interactive programs (those running at an interactive processing priority), and real-time programs; can submit and control RJE jobs; and can query the status of the computer system and exert some control over it. In effect, then, terminal users feel as if they each have their own computer.

The primary interface between the interactive terminal user and VULCAN is the VULCAN interactive command language, VULCAN Job Control. It is through the use of this language that the interactive terminal user can invoke a compiler, initiate a program, print a file, edit data, etc. If certain sequences of Job Control commands are used frequently, the commands can be stored in disc files and become job stream files. The command sequence can then be invoked by a single command. (See "The Command Language — VULCAN Job Control" for a more comprehensive discussion of the command language.)

An interactive terminal has a software priority associated with it, that is, with the terminal itself. It is at this priority that normal interactive commands are processed. From an interactive terminal, however, a program of any priority can be initiated (provided that the user initiating the program has the authority to use the selected priority). In a time-sharing environment, all interactive terminals may run either at the same priority or at different priorities depending on the requirements of the installation.

## 2-4 BATCH PROCESSING

Batch jobs are initiated from an interactive terminal or a card reader and generally run unattended. Batch jobs communicate with the operating system through the same command language that is used interactively. As a matter of fact, in most cases, to submit an interactive job stream file as a batch job, all that is required is a JOB "card" at the beginning of the job stream and an EOJ (End Of Job) "card" at the end. Hence, a batch job can involve more than the execution of a single program. It can be composed of multiple program initiation requests and Job Control commands.

Once an interactive user submits a job stream file to batch, that user can continue performing interactive tasks. The batch job is executed concurrently and the user can query its status, find out when it's done, and retrieve its output at the interactive terminal.

### 2-4.1 Multiple Batch Execution

Although a system operator can declare how many batch jobs should be permitted to run at any one time, under the normal mode of operation it is VULCAN that makes that decision. VULCAN constantly monitors the load placed on the system by the other processing modes and dynamically adjusts the optimal batch load accordingly. The maximum number of concurrent batch jobs to be permitted at an installation is determined in advance by the computer center staff. A typical maximum is 16.

### 2-4.2 Control Points

Not all batch jobs are equally important. That's why several priorities are usually set aside for batch (as in Table 2-1). Batch jobs can be run at any batch priority.

The priorities allocated to batch are distributed among several groups called control points. A control point is a software entity within VULCAN that monitors and controls the execution of a batch job. Just as an interactive program is associated with a terminal, a batch job is always associated with a control point. If four control points are defined in the system, then up to four batch jobs can run concurrently.

The number of control points and the range of priorities associated with each control point varies with the number of concurrent batch jobs permitted. If, for example, VULCAN determines that only four batch jobs can be run, then there are four control points available. Control points are identified by a letter of the alphabet.

The priority ranges associated with the control points overlap. If there are four control points in a system with sixteen batch priorities, for example, the priorities would be allocated like this:

| Control Point | Priorities |
|---|---|
| A | 0-15 |
| B | 4-15 |
| C | 8-15 |
| D | 12-15 |

This method of distributing priorities helps ensure that the more important batch jobs will be processed first since they can run at any control point that is available, whereas the very lowest priority batch jobs must wait for control point A to be available.

If a batch job is initiated and there is no appropriate control point free, the job is placed at the end of a single batch job queue. As soon as any control point becomes free or is created by VULCAN or the system operator, the queue is scanned for the highest priority job that can be run at that control point and the job is executed. In case of a tie, the job that was initiated first is processed first.

The operator can alter the priority of a batch job in the queue. This facility is useful when a system is running a large number of batch jobs and those with lower priorities have difficulty getting initiated or getting resources in time for them to complete when needed.

### 2-4.3 Resourcing

Batch processing differs from interactive processing in the area of peripheral device resourcing, i.e., getting control of an unsharable device, such as a magnetic tape drive. An interactive user can request that a tape drive be allocated and that a particular tape be mounted and can then continue with other processing until the tape is ready. However, batch processing requires that all such resource requests for a particular job be satisfied before that job is executed. If all resource requests cannot be satisfied immediately, VULCAN tries again periodically to satisfy all the requests. This process prevents a control point from being tied up waiting for a particular device for what could be a considerable amount of time. It also prevents those resources that are immediately allocatable from being tied up waiting for those that aren't.

### 2-4.4 Job and Program Sequencing

Frequently it is necessary for one batch job to be completed before another one can be started. Rather than having an operator constantly check to see whether the first job has completed, VULCAN's job sequencing facility can be used. With this facility, a job can be initiated but is not executed until one or more specified jobs are completed.

In addition, certain contingencies can be accounted for within the job stream. For example, let's say that one job stream usually invokes three programs in sequence – first PROG-A, then PROG-B and finally PROG-C. However, if a certain condition is detected during the execution of PROG-B, it's necessary to skip PROG-C and execute PROG-D instead. Program sequencing, then, can be built into the job stream.

The job and program sequencing facilities lessen the requirement for human intervention.

## 2-5 REAL-TIME PROCESSING

### 2-5.1 What Is A Real-Time Program?

Real-time programs monitor or interact with ongoing events outside of the computer system itself. Such ongoing events can be experiments or industrial processes which require that samples be taken periodically and that valves be adjusted or which involve stages in a production line environment that need to be monitored and controlled.

Response time in such situations is often more critical than the response time at an interactive terminal. VULCAN has many features besides just a high software priority which make it possible for real-time programs, whether they are written in FORTRAN, BASIC or assembly language, or a combination of these, to react quickly to external situations.

In particular, VULCAN supports three aspects of real-time operation: program initiation, program execution and inter-program communication. (Much of this support is available through the use of VULCAN System Services. See "System Services" for more information on System Services.)

### 2-5.2 Real-Time Program Initiation

In cases where initiation speed is not critical, real-time programs can be initiated via command from the operator's console, an interactive terminal, a job stream file, or even a batch job stream. They can also be initiated by another program without external intervention.

An even faster method of initiating real-time programs is in response to an external hardware priority interrupt. Any number of external interrupts in the system can be set aside for this purpose. They can be "connected" to and "disconnected" from programs either by another program or by command from the operator's console or interactive terminal. The interrupts can similarly be controlled (i.e., enabled and inhibited).

The initiation of a real-time program can also be a time-related event. A real-time program can be scheduled to initiate at a certain time of day and/or on a certain date, or after a certain specified time period has elapsed. It can also be scheduled to automatically re-initiate at fixed time intervals such as every hour on the half-hour or every three seconds. In between periodic executions, the program can remain in memory for faster re-initiation. Timer scheduling can be specified when the program is initiated, either by command or by another program.

Finally, real-time programs that must run whenever the computer is in operation can be automatically loaded and initiated when the operating system is initiated.

### 2-5.3 Program Execution

In Table 2-1, real-time programs are broken down into two categories. The highest priority, most time-critical programs fall into the category of non-paged real-time. These programs are loaded in their entirety when they are initiated and stay in physical memory until they are complete. Thus, their ability to respond quickly is not degraded by swapping. In cases where response time during execution is less critical or during the development and testing of programs, real-time programs fall into the second category of paged real-time. Such programs are paged just like any interactive or batch program. Because paged real-time programs do not tie up a large block of memory, less swapping is required for other concurrent processing. (Think what happens in the library analogy if two books for Project A are chained to their locations. Every time another book is needed for any other project, the researcher has to run back and forth from the library to swap the one free book.)

Some programs fall into both categories. Such a program could be monitoring non-time-critical functions. In this case a program can modify its own execution priority or have it modified by another concurrently executing program.

During execution of any type of program, internal system accounting information is taken periodically. If a real-time program cannot afford the overhead required for this function, the generation of accounting information for the execution of that program can be inhibited by authorized users.

### 2-5.4 Inter-Program Communication

Several real-time services are available which permit one program to query and/or control another real-time program. Some of these have already been mentioned, such as dynamic program initiation, dynamic timer-scheduling, and dynamic modification of a program's execution priority. Additional services include the ability of one program to suspend and restart another program, to query its status, and to abort it.

As for passing data or parameters from one program to another, there are again several methods. First of all, when a program is initiated, either dynamically or by a command, it can be passed a 24-bit initiation parameter. During the execution, real-time programs can use the four VULCAN message communications services which are generally available to all types of programs. These four services are: send a binary or ASCII message of arbitrary length; receive a message of arbitrary length; check to see if there are any messages to be received; and count how many messages are waiting to be received.

For the passing or sharing of large amounts of data, a common area, known as Monitor Common, is available. See "Monitor Common".

# III. USER MANAGEMENT

## 3-1 THE NEED FOR USER MANAGEMENT

Any operating system that supports a time-sharing environment must provide facilities for user management.

It must be able to prevent unauthorized access to data.

It must prevent any one user from occupying an inordinate amount of on-line storage.

It must keep track of how much each person uses the system and prevent anyone from using it too much.

It must offer each user as much power and flexibility as possible without endangering its own operation.

It must ensure that each user gets a fair share of the available processing power.

This section will show how VULCAN meets these requirements.

## 3-2 USER IDENTIFICATION

Before discussing how users are managed, it is necessary to define how users are identified, that is, how VULCAN determines which user is doing what.

### 3-2.1 The User I.D.

The user i.d. uniquely identifies each person who is authorized to use the computer system. VULCAN does not permit anyone to use the system unless a valid user i.d. is presented first.

User i.d.'s can have one of two forms. If a strictly numeric i.d. is desired, such as a social security number, any string of from one to twelve decimal digits can be used. If an alphanumeric i.d. is desired, such as a set of initials, a string of from one to six alphanumeric and special characters can be used, the first of which must be alphabetic. In either case, imbedded blanks and commas are not allowed. Each installation should choose either alphanumeric or numeric i.d.'s and not use a mixture of both.

### 3-2.2 The Qualifier

A valid user i.d. is necessary but not sufficient to access the system. A valid qualifier must be presented too. (There is one exception to this rule — the operator's console. See "User Access To OPCOM".)

The qualifier has two parts. The first part contains from one to four digits signifying an account number or a department number. The second part of the qualifier is a one to four character alphanumeric identifier which can be used to indicate an account name or a subgrouping within a department.

Each user i.d. can be associated with more than one qualifier and each qualifier can be associated with numerous user i.d.'s. For example, let us say that the account number for the engineering department of a company is 1234. The department has two subgroupings — development and applications — and so there are two qualifiers associated with the engineering department:

1234DEV
1234APPL

A user in the engineering department has the user i.d. "FRED??". Since this user sometimes works on development and sometimes maintains current applications, his user i.d. is associated with both qualifiers. Hence, both of the following combinations are defined to VULCAN as valid sign-on identification at this installation:

1234DEV FRED??
1234APPL FRED??

### 3-2.3 Signing On

Signing on is the process by which users identify themselves to VULCAN. To work in the interactive mode, signing on is accomplished by keying in a valid qualifier/user-id combination as illustrated above. When submitting a batch job, this same information must appear on the first record of the job stream or card deck. When initiating a real-time program, it must appear on the program initiation request. Since most users are not authorized to use the operator's terminal at all or in part, the operator's terminal does not accept any commands unless a valid user i.d. is presented first; that is, even operators must sign on.

In addition to their function of identification, the qualifier and user i.d. are also used by the internal accounting system and the disc file security system (see "Internal System Accounting" and "Disc Area Security").

## 3-3 HOW USERS ARE MANAGED

### 3-3.1 User Control Parameters

Once a user i.d. and qualifier are assigned to a new user, they must be given to VULCAN (through the operator's console) along with several additional parameters. Eight of these additional parameters define the privileges that a user is to have — including the privilege of giving VULCAN new user parameters. These parameters are:

1. Access level — The access level is a number between zero and fifteen, a higher number indicating a more privileged status. Disc areas have an access level too, and a user cannot access a disc area that has an access level greater than that user's access level.

2. Program size limit — The maximum allowable value is 256 virtual pages (768K bytes).

3. Control point time limit — Batch jobs which exceed the number of seconds of CPU time indicated are aborted. This prevents programs that "run away" from tying up computer resources indefinitely. Users can impose lesser time on their jobs, if they desire, when they initiate them.

4. Output spool size limit — One function of this limit is to prevent users from printing or punching an inordinate amount of data. It is also useful in the case of run-away programs.

5. Execution priority limit — This limit prevents low priority users from dominating the system.

6. CPU time limit — Users who have accumulated the indicated amount of CPU time are not permitted to sign on.

7. Disc space limit — This limit indicates the maximum amount of disc space a user can occupy at one time.

8. Access bit mask — This 24-bit mask is associated with several privileges. It indicates which operator console commands a user can issue, if any; it can be used to prevent a user from accessing particular peripheral devices; and it can be used to prevent access to some of VULCAN's System Services (see "System Services").

### 3-3.2 Internal System Accounting

VULCAN automatically records CPU and peripheral device utilization for each user. The latter includes such data as number of lines printed, number of tape records processed, number of disc accesses, number of cards processed, connect time, and amount of disc space occupied.

The Accounting Utility program (ACUTIL) is used to generate utilization reports from this information. ACUTIL optionally accepts charge rates and produces billing reports for individuals, qualifiers, and accounts. Different rates for different time periods can be specified too, giving the capability of charging different rates for prime time, evenings, weekends, etc.

# IV. THE COMMAND LANGUAGE
# VULCAN JOB CONTROL

## 4-1 INTRODUCTION

VULCAN Job Control is the interface between VULCAN and interactive and batch users. This section gives an overview of the capabilities of VULCAN Job Control. The VULCAN Terminal User's Guide is recommended for those who wish to learn the command language. Detailed information on the commands can be found in the VULCAN Job Control and System Processors manual.

## 4-2 COMMAND STRUCTURE

Job Control commands are composed of one to three parts: the command word, command options, and command parameters.

Command words are one to eight alphanumeric characters long. Except for four commands which must start with a dollar sign ($), all command words of three or more characters be abbreviated to two characters to allow faster input for users who are familiar with the command language. With only a few exceptions, the abbreviations are the first two characters of the command word.

Example: COPY is equivalent to CO

Command options modify commands. Options are indicated with a letter of the alphabet and are separated from the command word by a period. Multiple options are specified by a string of characters in any order.

Examples:

Compile a COBOL program to check syntax only – COBOL.S

Get size and date information for a group of disc files belonging to the signed-on user – MAP.SDGU or MA.DUSG etc.

Command parameters supply further information. They are separated from one another and from the command word or options by either a comma or a space. Parameters can be specified in any order.

Example:

Create a disc file named MYFILE on disc pack 3
with public read access – GENERATE MYFILE P=3 PR or GE MYFILE PR P=3 etc.

Job Control commands can be keyed in one at a time directly at an interactive terminal or they can be pre-stored on and executed off of disc. (A disc file containing Job Control commands is called a job stream file. See "Job Control Programming" for more information on job stream files.) Job Control commands can also be submitted via the card reader or a disc file for batch processing.

## 4-3 TEXT EDITING

Text editing commands are a subset of VULCAN Job Control rather than being part of a separate processor. This means that while editing a file the user can initiate and query batch jobs, create new disc files, change the mode of operation of the terminal, query the status of the system, etc.

Most text editing commands fall into two categories – record editing commands and character editing commands.

The record editing commands operate on records, or lines, within a disc file. The records are identified by line numbers which are provided by Job Control when a file is listed. These commands have the following functions:

insert a record (e.g., a line of text)

delete a record

change (delete and add) a record

locate a record that contains a particular character string

locate a record that does not contain a character string

change all occurrences of one character string to another

The character editing commands operate on character strings within a record. Character editing commands are exactly one character in length. These commands have the following functions:

insert a character string

delete a character string

change a character string

locate a character string

replace a line (only for interactive CRT terminals)

When editing commands are entered, they do not immediately cause a change to the file being edited. Instead, they are pre-stored until an UPDATE command is issued. Postponing the updating has two advantages. First of all, if one is referring to a printed listing of the data or program being edited, the line numbers on the listing can be used for record identification regardless of how many records have been added or deleted from the middle of the file. Secondly, records can be edited in any order since Job Control sorts them by record number and command hierarchy before actually performing the update. If desired, however, a user can request continuous updating.

The user has the option of preserving or not preserving the original version of the file being edited. A facility also exists to recover editing commands that were entered but not updated prior to an abnormal termination of processing, such as a disconnect from a remote terminal.

## 4-4 PERIPHERAL AND DISC FILE CONTROL

There are no special utility programs under VULCAN for file manipulation or media-to-media conversion. Instead, all such functions are available in the form of simple Job Control commands. The basic file manipulation commands include rewind, wind (the opposite of rewind, meaning position the file at the end), write end-of-file, advance file(s), advance record(s), backspace file(s), and backspace record(s).

The COPY command is the basic command for transferring data from one medium to another. Copying of one or more files or one or more records can be specified. This command can be used for any two media.

The MAP command is used to supply information about one disc area or a group of disc areas. (For an explanation of the relationship between disc areas and disc files, see "Disc Areas and Disc Files".) Standard output includes areaname, disc pack number, and the current size of the area. Additional information can be requested by specifying options, e.g., option D for date information (date generated, expiration date, last date accessed, and last date written on); option S for additional size information; option A for access information (i.e., what security restrictions apply to the area); or option E for everything (equivalent to specifying the D, S, and A options).

See "Disc Area Backup and Retrieval" for more information on the MAP command.

## 4-5 JOB CONTROL EXAMPLE

Table 4-1 is an example of the use of Job Control commands. The purposes of the terminal session illustrated are: (1) create a FORTRAN source program on disc (this program will be keyed in directly but it could have been punched on cards); (2) compile and execute the program; and (3) edit the source file.

**Table 4-1. Job Control Example**

| Command | Description |
|---|---|
| GENERATE MYFILE | Reserve space on disc and name the area MYFILE. |
| COPY * MYFILE | Copy the statements to follow from my terminal to MYFILE. |
| . . . | (Key in the statements to be added.) |
| $EOF | Write an end-of-file on MYFILE and terminate the COPY command. The next statement is a command, not data. |
| FORTRAN MYFILE | Compile the source program on MYFILE. |
| VX | Vulcanize (link-edit; catalog) and execute the program (this could be done in two steps, too). |
| EDIT MYFILE | Identify an area to be edited. |

Table 4-1. Job Control Example (Cont'd.)

| Command | Description |
|---|---|
| CHANGE 41-43 | Indicate some records to be changed. |
| . . . | (Key in the statements to replace 41-43.) |
| E 57 C L O | Identify a record for character editing and change the first L on the line to an O. |
| INSERT 22 | The statements following this command will be inserted after line 22 in MYFILE. |
| . . | (Key in the statements to be added.) |
| $$$ | Stop inserting. The next statement is a command. |
| INSERT 49 FILEX 9-13 | Insert lines 9-13 of FILEX into MYFILE following line 49. |
| DELETE 5-10 | Delete lines 5 through 10 of MYFILE. |
| UPDATE | Incorporate the above changes to MYFILE. |

In this example, the Vulcanized version of the program, the executable code itself, is stored on a temporary disc file. The following command stores the program on a permanent file called MYPROG:

```
VX MYPROG
```

To execute the program at a later time, the following command can be issued:

```
MYPROG
```

## 4-6 MESSAGE COMMUNICATIONS

There are three commands that fall into this category. They are SEND, RCEIVE (ReCEIVE), and OM (Operator Message).

With the SEND command, a one-line message can be sent to another interactive terminal, another user, or a program. When sending a message to another terminal, the terminal is identified by a number. If no one is signed on to that terminal when the message is sent, the message is held until someone does sign on. When sending a message to another user, the user is identified by a username. (The username is not the same as the user i.d., which is private.) If the specified user is not signed on when the message is sent, it is delivered when that user does sign on (to any terminal). Messages can be sent to any program. It is the

destination program's responsibility to check for and receive messages. (Message communication services for programs are discussed under "Inter-Program Communication".)

If a message is sent to a teletype device or to a user who signs on to a teletype device, the message is output after the next command is issued from that terminal. If the terminal in question is a CRT, a "message" light illuminates. The message itself is then received by issuing a RCEIVE command.

## 4-7 COMMANDS TO ASSIST THE TERMINAL USER

Error messages generated by VULCAN in response to an invalid interactive command or an error in an executing program, or error messages generated by system processors such as compilers, all have three levels of diagnostic aids. These are an error number, a one-line extended message, and up to four lines of additional information which frequently includes suggestions for getting around the problem. All of these aids are available at the terminal itself — the user does not have to locate an appropriate manual and look up the error number.

Error numbers are always provided. The associated extended message is automatically written to CRT devices, although this output can be inhibited. On teletype devices, the automatic extended message facility is optional: new users can "turn on" the extended message at their terminal for an entire interactive session; those already familiar with many error numbers can request that the extended message be written only when they ask for it — they need not be burdened with slow and often noisy typing when they don't need the output.

If, after reading the extended message, the user still does not understand the problem or doesn't know what to do about it, the HELP command can be invoked to get the additional diagnostic information described above.

If a person working at a teletype device gets an error message, the teletype hardcopy can be examined to help find the source of the error. For CRTs, however, there is no hardcopy to examine (unless the CRT has a hardcopy option). Therefore, a temporary log is kept of all commands executed from that terminal — either directly or from a job stream file — which the user can examine and/or save and/or print. A CRT user can also request that these same commands be echoed back to the CRT screen after they have been transmitted. This function is particularly useful when entering source statements and data and to monitor the progress of a job stream file.

To assist in program debugging, the PMD (Post-Mortem Dump) command is available. With this command it is possible to get a dump of user-specified sections of a program that has aborted. It can dump all words of the specified area; dump only words that have been changed during execution (e.g., data areas); and dump changed words along with their original values. The output may be requested in octal, ASCII, truncated ASCII, decimal, hexadecimal, or floating-point notation. The Post-Mortem Dump can be processed in batch or interactively; portions of the aborted program can be selectively examined interactively.

The Interactive Debugger (VBUG) is also available to assist in program debugging. See "VBUG (VULCAN Symbolic Debugger)".

## 4-8 JOB CONTROL PROGRAMMING

Job Control commands can be stored on a disc file called a job stream file and executed off of that disc file. In response to a JSTREAM (Job STREAM) command, which names the job stream file to be executed, the stored commands are executed sequentially just as if they had been keyed in directly from a terminal. After the last command in the job stream file has executed, control automatically returns to the user's terminal (i.e., the terminal is then ready to accept a keyed-in command) or to a batch job stream if the job stream file is initiated from batch.

A more sophisticated form of job stream file is a Job Control program. A program does not necessarily execute commands in a linear sequence — it can make decisions and, depending upon the outcome, it can "branch", or "jump".

There is a set of commands available only to job stream files. These commands include several types of jumps: jump unconditionally, jump if an error has or has not occurred, jump if any Job Control error has or has not occurred, and jump if a specific option is or is not set.

Whenever there are jumps, there must be labels to identify their destinations. Under VULCAN, any Job Control statement in a job stream file can have a label of from one to six alphanumeric characters.

Programs, in general, need some place to store and manipulate data, and Job Control programs are no exception. For this purpose, there are Job Control registers. Although the use of these registers is not restricted to job stream files (they can even be accessed by programs), it is there that they are most useful. The registers can have mathematical, logical, and string manipulations performed on and among them. Registers have user-supplied names of one to three alphanumeric characters. Wherever a register name appears in a Job Control command, its contents are substituted. (See the Job Control programming example, Table 4-2.)

Job Control has I/O commands too. In their simplest form, these commands make it easy to program a dialogue between an interactive terminal user and VULCAN. They can be used to prompt and query. In addition, these commands can perform I/O to any device.

Finally, there is the IF command which evaluates an expression and, based upon the result, causes another command to be executed or not executed. The expression to be evaluated can be quite complex. Mathematical (addition, division, etc.), logical (and, or, exclusive or), and relational (less than, greater than or equal to, not equal to, etc.) and other operations are available. Multiple levels of parentheses can be used to modify the normal hierarchy of expression evaluation.

The applications of Job Control programs are too numerous to list. The example illustrated in Table 4-2 involves controlling the execution sequence of a series of programs.

Here is the problem. Three programs are normally executed in sequence. The first program is either PROG-1A or PROG-1B depending on an option set by the user when the Job Control program is initiated. Then PROG-2 executes. If all goes well, PROG-3 is executed next and the job is done. However, if a certain condition is detected by PROG-2, it immediately terminates. In that case, PROG-3 must be skipped, PROG-4 executed instead, and a message must be sent to the operator so that an entry will be made in the system log.

Table 4-2. Job Stream File Example

| Label | Command | Comment |
|---|---|---|
| !START | JOPTION.A !DO-1A | $ JUMP TO DO-1A IF OPTION A HAS BEEN SELECTED. |
| | JOPTION.B !DO-1B | $ JUMP TO DO-1B IF OPTION B HAS BEEN SELECTED. |
| !INVOPT | PRINT INVALID OPTION – DO YOU WANT A OR B? | $ THIS MESSAGE WILL APPEAR AT THE INTERACTIVE TERMINAL<br>$ FROM WHICH THE JOB WAS INITIATED. |
| | SREG.I #OPT | $ SET A REGISTER FOR INPUT. WHATEVER IS KEYED IN IS<br>$ STORED IN JOB CONTROL REGISTER #OPT. |
| | IF (#OPT = A) SOPTION A | $ SINCE THIS OPTION WAS EVIDENTLY NOT SET CORRECTLY ON<br>$ THE JSTREAM COMMAND IT MUST BE SET NOW WITH A SET<br>$ OPTION COMMAND. |
| | IF (#OPT = B) SOPTION B | $ (SAME COMMENT AS ABOVE.) |
| | JUMP !START | $ GO BACK TO THE TOP AND TRY AGAIN. |
| !DO-1A | PROG-1A | $ EXECUTE PROG-1A. |
| | JUMP !DO-2 | $ JUMP TO WHERE PROG-2 IS INITIATED. |
| !DO-1B | PROG-1B | $ EXECUTE PROG-1B |
| !DO-2 | PROG-2 | $ EXECUTE PROG-2. |
| | JERROR.P !DO-4 | $ JUMP ON AN ERROR IN THE LAST PROGRAM TO DO-4. |
| | PROG-3 | $ OTHERWISE EXECUTE PROG-3. |
| | JSTREAM | $ A JSTREAM COMMAND WITHOUT A PARAMETER WILL RETURN<br>$ CONTROL TO THE TERMINAL. |
| !DO-4 | /OM PROG-4 EXECUTED INSTEAD OF PROG-3 | $ THIS OPERATOR MESSAGE WILL BE WRITTEN ON THE OPERATOR'S<br>$ CONSOLE AND WILL BE ENTERED IN THE SYSTEM LOG. |
| | PROG-4 | $ EXECUTE PROG-4. |
| (End-of-File) | | $ AN EOF MARK WILL RETURN CONTROL TO THE TERMINAL. |

All the commands necessary to execute this procedure are stored in a disc file named EXAMPLE. To indicate whether PROG-1A or PROG-1B is to be executed first, a Job Control option will be set when the job stream file EXAMPLE is initiated. To execute PROG-1A first, option A is set; to execute PROG-1B first, option B is set. Here then are two ways that job stream EXAMPLE can be initiated:

```
JSTREAM.A EXAMPLE
JSTREAM.B EXAMPLE
```

Some special characters used in the example are used by Job Control to identify special fields. An exclamation point (!) indicates a label. A pound sign (#) indicates a Job Control register. A dollar sign ($) followed by a space indicates that the rest of the line is to be treated as a comment. The commands are free-format and are organized for easy reading.

## 4-9 JOB CONTROL MACROS

Far more powerful than a Job Control program is a Job Control macro, another special type of job stream file. Several characteristics distinguish macros from other job stream files.

First of all, Job Control commands in macros can have dummy arguments which are filled in when the macro is called. Thus it is possible to write generalized Job Control macros and fill in the particulars when the macros are initiated.

Secondly, a macro becomes, in effect, a new Job Control command. Keying in the name of the macro is all that is needed to set the macro into execution.

Thirdly, there are three levels of macros. The lowest is a user macro. Anyone can write one, but their use is normally restricted to one user or all users having a particular qualifier. Next is a system macro. These are available to all users. Only authorized users, however, can create system macros. Finally, there are Harris macros which are actually an integral part of the command language.

There is a hierarchy associated with macros. A user macro can override a system macro, a Harris macro, or a regular Job Control command. A system macro can override the latter two.

Here are some applications of macros.

1. A member of the accounting department is making an on-line inquiry to the data base and needs to make a few arithmetic calculations. A macro is called up that evaluates arithmetic operands, much like a hand calculator, and the answer is displayed.

2. A manufacturer frequently has to calculate the average cost of a series of parts. A macro called AVERAGE is invoked followed by the prices to be averaged and the answer is displayed.

3. A former keypuncher has been trained to key in data at an interactive terminal. Each of several possible data formats requires a different set of tab stops. For example, invoice line items require tab stops at columns 3, 7, 15, 35, 49 and 78. By keying in INVTAB, the name of a macro, the keypuncher gets all these tabs set. Such a a macro would look like this:

```
MSTART                  (Macro Start)
TAB 3,7,15,35,49,78
MEND                    (Macro End)
```

4. A new VULCAN installation is installed at a site where it will be used by seventeen application programmers. The programming staff is accomplished in the use of the editing commands of a large time-sharing service and the management doesn't want to retrain everybody. So the time-sharing service editing commands are emulated through the use of macros.

# V. PROGRAMMING FACILITIES

## 5-1 INTRODUCTION

The features and facilities described in this section are useful for those who will be programming under VULCAN. The Format Scanner and System Services are used primarily by assembly language programmers; however, they can be incorporated into assembly language subroutines which are then called by programs written in COBOL, FORTRAN, BASIC or RPG II. Monitor Common can be used only by assembly language and FORTRAN programmers. Monitor programs are useful to systems-level programmers and are not often used by application programmers. Libraries and overlays are of interest to all programmers.

## 5-2 THE LANGUAGE PROCESSORS

VULCAN supports seven language processors. They are: FORTRAN IV, COBOL, BASIC, RPG II, Harris Macro Assembler, FORGO, and SNOBOL 4. All of these processors are reentrant. In addition, FORTRAN IV, COBOL, BASIC and RPG II generate reentrant code automatically. With the help of simple pseudo-operations, assembly language programs can also be reentrant.

With the exception of FORGO and SNOBOL 4, subroutines written in any languages can be linked together to form a single executable program. For example, an RPG program can call a FORTRAN subroutine to do complex mathematical processing.

A description of each of these languages and of the extensions and enhancements that Harris offers with each can be found in Appendix A.

## 5-3 VBUG (VULCAN Symbolic Debugger)

VBUG is an interactive, user-oriented debugging tool that provides extensive capabilities for supervising and modifying the execution of a program.

To use VBUG, a program is compiled and linked with a VBUG option set. When the program is subsequently initiated, VBUG writes a greeting message on the user's terminal and awaits command input. Some of the command functions are: modify a variable; display a variable (in decimal or octal); restart program execution at a different point; display and modify the contents of an internal register; and set and reset breakpoints.

A breakpoint is a user-specified location in a program at which the user wishes to suspend execution for the purpose of entering additional VBUG commands. These additional commands can be entered interactively. The user can also pre-define a series of commands to be executed automatically whenever a particular breakpoint is reached (e.g., display a variable and continue execution). A breakpoint need not occur every time the specified location is reached – the user can specify breakpoint frequency (e.g., every fifth time) and a limit on the number of times the breakpoint is executed (e.g., until seven breakpoints are taken at this location).

Through the use of the terminal interrupt facility, users can break into executing programs at an arbitrary time by hitting a special key on the terminal. At that time, the user can enter new VBUG commands just as if a breakpoint had been preset. This feature is particularly useful when a program is caught in a loop.

The user can specify the location of breakpoints in one of two ways: (1) by label or statement number (e.g., for FORTRAN programs) or (2) by line number (e.g., for assembly language programs). A combination of both can be used (e.g., label + 2 lines) and arithmetic is permitted among labels, line numbers, and user-specified constants.

Variables are referenced by variable name. Arithmetic can be performed with variable names and values as well.

## 5-4 LIBRARIES

### 5-4.1 Subroutine Libraries

A subroutine library is a collection of subroutines that are used by more than one program or concurrently by separate initiations of the same program. It consists of "link ready" modules — subroutines that have been compiled or assembled but not linked by the linkage editor. The *particular library subroutines* required for a program are linked along with the main program and other user-written subroutines to form a single integrated program ready for execution.

VULCAN itself has a system subroutine library that contains standard subroutines used by the FORTRAN, COBOL and BASIC compilers. Users can add new modules to the system library or change or delete existing ones through the use of the Library Editing Job Control commands. Users can easily create and maintain their own subroutine libraries as well. Any number of subroutine libraries can be specified to the linkage editor.

The subroutines in the system library are reentrant, as can be those in a user subroutine library. This reentrancy is beneficial when several users are executing *the same* program concurrently.

### 5-4.2 Reentrant Libraries

A reentrant library is a collection of subroutines that are used by many different programs concurrently. It consists of Vulcanized subroutine modules that are linked to a main program when the main program is Vulcanized. In this case, the *entire library* is loaded as a separate entity and shared by *different* concurrently executing programs, each calling different subroutines or the same subroutines from the reentrant library.

Only one reentrant library per program is permitted, and reentrant libraries cannot exceed 45K bytes. If the size of a reentrant library changes, all programs using that library must be re-linked.

### 5-4.3 The COBOL COPY Library

COBOL has a built-in facility for source statement libraries. See "The COBOL COPY Library" in Appendix A.

## 5-5 MONITOR COMMON

Monitor Common Blocks are common blocks containing data that is shared by several programs. The data is stored on disc and is paged into memory as needed by executing programs, or the data can be declared resident. Monitor Common Blocks are updated periodically and, in the case of paged Monitor Common, they are always swapped out to their original disc location. Values set into Monitor Common remain there until modified or eliminated, even if VULCAN is reloaded from disc.

Any number of Monitor Common Blocks can be generated and any number can be used by an individual program. Any number of programs can share a single Monitor Common Block and all can have concurrent update access. Monitor Common Blocks are defined in the user's program (either FORTRAN- or assembler-coded) as regular common blocks with one additional statement indicating which common blocks in the program are Monitor Common Blocks.

## 5-6 SYSTEM SERVICES

System Services, sometimes known as monitor calls, are VULCAN routines, most of them non-resident, which are available to users. All System Services are directly available to assembly language programmers, although some are subject to security restrictions (see "User Control Parameters"). The real-time services that are discussed under "Real-Time Processing" are also directly available to FORTRAN and BASIC programmers and are indirectly available to all languages through calls to user-written assembly language subroutines.

### 5-6.1 General Capabilities

Some of the System Services invoke selected Job Control functions from within a program; for example, the generation and elimination of disc areas, the initiation of batch jobs, the retrieval of the system date and time, the retrieval of the accumulated CPU time for a batch job or a terminal session or a program. Other services provide batch program chaining, push and pop stacks and control, dynamic allocation of logical memory, and binary-to-ASCII conversion. A contingency return service is also available to allow a user program to regain control after it has aborted.

The following services are discussed under "Real-Time Processing": dynamic program initiation, connecting and disconnecting external interrupts to and from programs, controlling such external interrupts, timer-scheduling of programs, changing the priority of an executing program, suspending and restarting programs dynamically, querying the status of executing programs, and aborting programs. VULCAN's message communication services are also discussed in this section.

The I/O Services and the Format Scanner each represent a group of services. They are discussed below.

### 5-6.2 I/O System Services

The I/O System Services perform logical I/O (as opposed to physical I/O). That means that it is not necessary to decide what physical devices a program is to use until just before the execution of the program, via Job Control commands or the linkage editor, or during

execution of the program through the use of the ASSIGN System Service. Once VULCAN is informed what physical devices are to be used, it can set up and execute the appropriate physical I/O instructions. Physical I/O commands are not generally available to applications programmers.

### 5-6.3 The Format Scanner

A number of VULCAN system programs and routines accept command input — Job Control, the linkage editor, the Operator Communications processor, etc. Input to these different processors conforms to a common format, or syntax. The Format Scanner consists of a number of System Services which are used to analyze the various commands that use this common syntax. When writing application programs which process input that has some of the syntax characteristics common to those of VULCAN processor command input, the programmer may find it convenient to use the Format Scanner instead of writing new analysis routines.

## 5-7 MONITOR PROGRAMS

User-written programs normally have the following characteristics: their addresses are mapped by the hardware rather than biased by the software; they are paged (except for high-priority real-time programs); they cannot address real memory, that is, specify an absolute memory address; they cannot access logical memory outside of their own logical limits; and they cannot execute virtual memory instructions. These characteristics describe what is known as the "User Mode".

When necessary, programmers can be given the authority to write programs to run in the "Monitor Mode". (The Monitor Mode is meant to be used by system programmers only.) Such programs have software-biased addresses; they are not paged; they may address absolute memory locations; they may access any real memory in the system; they can execute restricted virtual memory instructions; and they run in the same mode as VULCAN which is itself a monitor program. A special type of monitor program, called a Non-Resident Handler, is designed to be called as a subroutine rather than being a complete program by itself.

Examples of monitor programs are: I/O handlers, hardware interrupt service routines, and System Services. The monitor program facility is also used to create absolute, stand-alone load modules.

## 5-8 UTILITY PACKAGES

### 5-8.1 Sort/Merge Package

VULCAN's reentrant Sort/Merge Package sorts and merges files stored on any medium and of arbitrary size. It can handle up to ten ascending and descending fields at a time.

The Sort/Merge Package includes a Sort/Merge Utility program which accepts simple parameters and which can be used by interactive and batch users. The rest of the package

consists of individual routines which can be called from high level languages. Thus it is possible to sort or merge an input or output file dynamically, i.e., during the execution of a program. (Dynamic sort/merge capability is built into COBOL which has its own SORT and MERGE verbs.)

### 5-8.2 VULCAN Indexed Sequential Package (VISP)

VISP provides the ability to access records on a disc file sequentially, or directly by key. VISP keeps records arranged logically by the collating sequence of one or more key fields contained within each record. A tree structure of pointers is maintained to allow the user to add, change, delete or query selected records of the file. This structure also allows sequential processing (i.e., in increasing order of one of the keys) of the file. The search technique that is used to determine the location of a record once its key is known is both fast and efficient – it guarantees that the time to get any record in a file is the same as any other.

No initialization of a new indexed sequential file is required other than generating it and writing an EOF on it. The user need never worry about reordering it either since the structure of the file is such that maximum disc utilization is always attained. If the need for reordering ever arises, VISP does it dynamically. Any number of VISP files can be used by a program.

VISP includes an Indexed Sequential Utility program for interactive and batch users. It also contains individual subroutines which may be called from high-level languages. (Support for VISP is built into COBOL.) The functions performed by the Utility program and subroutines are:

Random read (with key)
Random write (with key)
File Positioning
Record deletion
Sequential read (yields an implicit file sort)
Random rewrite
Sequential rewrite

## 5-9 OVERLAYS

Under VULCAN, programs can have up to 192K bytes of instructions and data. Programs written in FORTRAN or Assembler can have up to 576K additional bytes of data for a total program size of 768K bytes. Single sub-programs (e.g., a main program or any one subroutine) can contain up to 96K bytes of instructions and data.

Larger programs can take advantage of the linkage editor's ability to generate overlay segments. (COBOL users may also use COBOL Segmentation.) An overlayed program is one in which two or more program partitions, or overlay segments, share logical memory space. These segments are relatively independent from one another since they cannot reference one another. An example of two segments is two independent subroutines which are called from a main program. The main program segment in an overlay structure is always logically resident.

Any number of overlay segments and levels of overlaying can be specified. Hence, the structure of an overlay program can be as simple or as complex as required.

# VI. DISC FILE MANAGEMENT

## 6-1 DISC SPACE ALLOCATION

### 6-1.1 Sectors

A *sector* is a subdivision of a circular track on a disc pack. It is 336 bytes long and is the smallest element of disc space that can be addressed or physically retrieved. The number of sectors per track varies with different disc drives. A disc pack that has eight sectors per track is pie-sliced as illustrated in Figure 6-1.

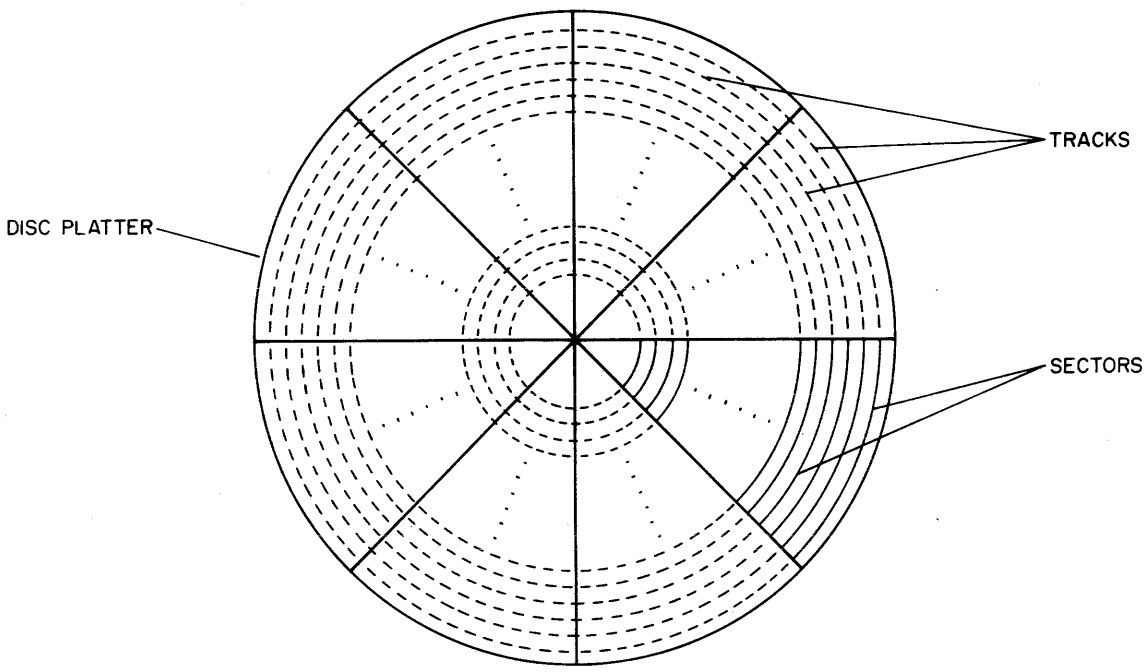



**Figure 6-1. Disc Sector**

### 6-1.2 Space Allocation Bits

When a number of sectors of disc space is requested, VULCAN locates available sectors by scanning an area on the designated disc pack which contains the Space Allocation Bits, or SABs. Each SAB is associated with one or more sectors. An SAB is turned on to indicate that its associated sectors are currently allocated.

The number of sectors to be associated with each SAB is determined when the operating system is generated and can vary from one disc drive to another.

Disc packs with a small SAB ratio (e.g., one sector per SAB) incur the greatest amount of overhead when a search for disc space is underway or when disc space is being deallocated because many SABs may have to be scanned and/or altered. On the other hand, disc packs with a large SAB ratio incur some disc storage overhead. For example, if one SAB represents 25 sectors and a one-sector file is created, the remaining 24 sectors are wasted. The best ratio for any one disc drive depends on the pack size that the drive accommodates and whether most of the files to be stored on a pack will be large or small.

### 6-1.3 Disc Areas and Disc Files

When disc space is requested, certain information must be supplied such as the number of sectors required (a default value is used when omitted) and a name (up to eight ASCII characters) for that space. The name provides a convenient way of referring to the allocated disc space without the necessity of knowing where it is. The sectors associated with the name are collectively referred to as a disc area. A disc area may be thought of as a reel of magnetic tape in that it can contain one or more files. Within the tape, the files are known only as the first file, the second file, etc.

Because of the conceptual similarities between tape reels and disc areas, certain terms that are commonly used to describe magnetic tape are used for disc areas as well: BOT (Beginning Of Tape) refers to the first sector of a disc area; EOT (End Of Tape) refers to the end of the area; and EOF refers to an End Of a File in the area. There may be multiple EOFs between the BOT and the EOT.

### 6-1.4 Granules

Many operating systems traditionally require that disc areas occupy contiguous disc space. This requirement often results in a significant waste of disc space. For example, if the exact amount of disc space required for a particular file varies from week to week, sufficient space has to be allocated in advance to accommodate the maximum anticipated size for the file – and usually a little extra, just in case. If the file during a particular week requires only one-fourth of the area, the remaining three-fourths are wasted since the entire area is reserved.

Under VULCAN, however, disc areas dynamically and transparently expand. If part of a disc area is no longer needed, the area can be "squeezed", that is, unused sectors can be returned to VULCAN and are immediately available for reallocation. The unit by which disc areas are initially allocated and by which they expand and contract is called a *granule*. A granule is an element of contiguous disc space, but different granules need not be contiguous. If an area needs more room to expand, another granule is automatically allocated from any available space on the disc until a user-specified maximum area size is reached. (See Figure 6-2.)

The size of a granule is specified when a new area is generated. Thus, the size of a granule varies from disc area to disc area. This variation significantly reduces waste caused by disc fragmentation – small granules can fill in the "holes" that may develop between larger granules. (See Figure 6-3.)

The choice of the granule size for a particular disc area depends on several factors. One is the SAB ratio. The granule size should always be a multiple of that ratio. To do otherwise would be wasteful. For example, if one SAB is associated with two sectors, then a granule size of five sectors would waste one sector for every granule.

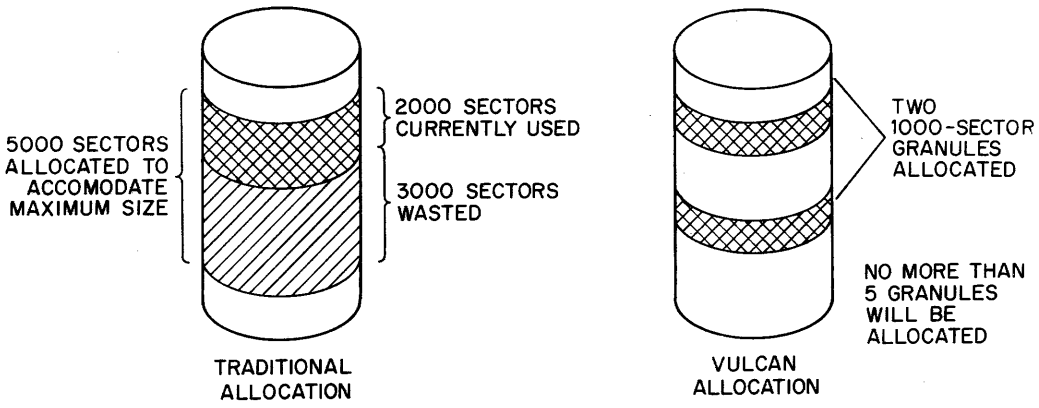



**Figure 6-2. Disc Space Allocation**

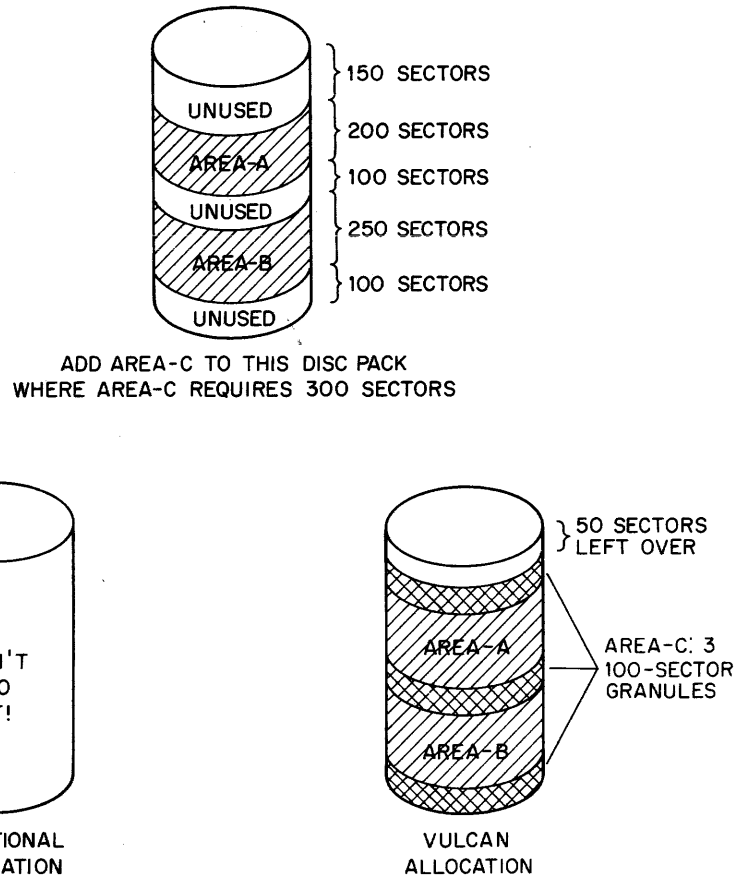



**Figure 6-3. Disc Space Fragmentation**

Of course, choosing a large granule size for an area which contains a small file is wasteful too, and it's harder to allocate space for large granules than for small ones. However, some overhead is associated with locating granules, although that overhead is generally minimal. Additional overhead is incurred for granules beyond the 87th. This additional overhead is explained in the next section ("The MAI and the EAI").

Granules for blocked files should also contain an integer multiple of the blocking factor of the area. As a matter of fact, VULCAN ensures that this is always true. Disc file blocking is discussed in the section entitled "Blocked Disc Areas".

### 6-1.5 The MAI and the EAI

The Master Area Index, or MAI, is the mechanism with which VULCAN keeps track of granules. It resides on an "unseen" sector that is tacked onto the beginning of the first granule of every disc area.

The MAI can only accommodate pointers to 86 granules. If a disc area requires more than that number, an Extended Area Index, or EAI, is tacked onto the front of the 87th granule. A significant amount of overhead is incurred when accessing an area randomly across different EAIs. Therefore, whenever possible, the chosen granule size for a disc area should be sufficiently large to prevent the creation of an EAI.

### 6-1.6 Disc Directories — The MDD and the QDD

Disc areas are referenced by user-specified names rather than by their locations (e.g., DATA1, PROGRAM5, XYZ123, etc.). The burden is on VULCAN to keep track of the first sector of each disc area. This information, as well as other data, is stored in disc directories.

VULCAN uses two types of directories. These are called a Master Disc Directory, or MDD, and a Qualifier Disc Directory, or QDD. The "master pack", which is defined as the pack that contains VULCAN, contains an MDD and a QDD which define all the disc areas known to the system. In addition, there is an MDD and a QDD on each satellite (i.e., non-master) pack which define the areas that reside on that pack. The directories on the master pack are the ones that are generally used. The satellite directories are used for special functions, such as transporting a disc pack from one VULCAN system to another or for backing up an entire disc pack.

Disc areas are associated with qualifiers (see "Disc Area Security"). The QDD maintains the following information, in qualifier order, for each disc area in its realm:

areaname
disc pack number
starting sector number
user i.d.
security data
granule size

There is one MDD entry for each qualifier in the system for which disc areas exist. Each entry simply defines the disc location of the QDD for its qualifier. (See Figure 6-4.)

When disc areas are being used from an interactive terminal or from a batch job, probably most if not all of the disc areas referenced are associated with the sign-on qualifier of the terminal or the job. Therefore, when any such area is first accessed, the MDD pointers for the sign-on qualifier QDD are loaded into main memory. This action reduces from two to one the number of disc accesses required to locate additional disc areas associated with that qualifier. (When extremely fast access to a disc area is required, as in some time-critical real-time situations, the QDD entry for that area can optionally be stored permanently in main memory. The memory directory is always checked before any disc directories.)

Both the MDD and the QDD are "hashed", that is, areanames are passed through a randomizing formula to produce a random sector within the directory. Each sector can handle a number of entries (28 MDD entries or 9 QDD entries). When a particular sector of either directory becomes full, VULCAN automatically expands the directory and rehashes the entries.

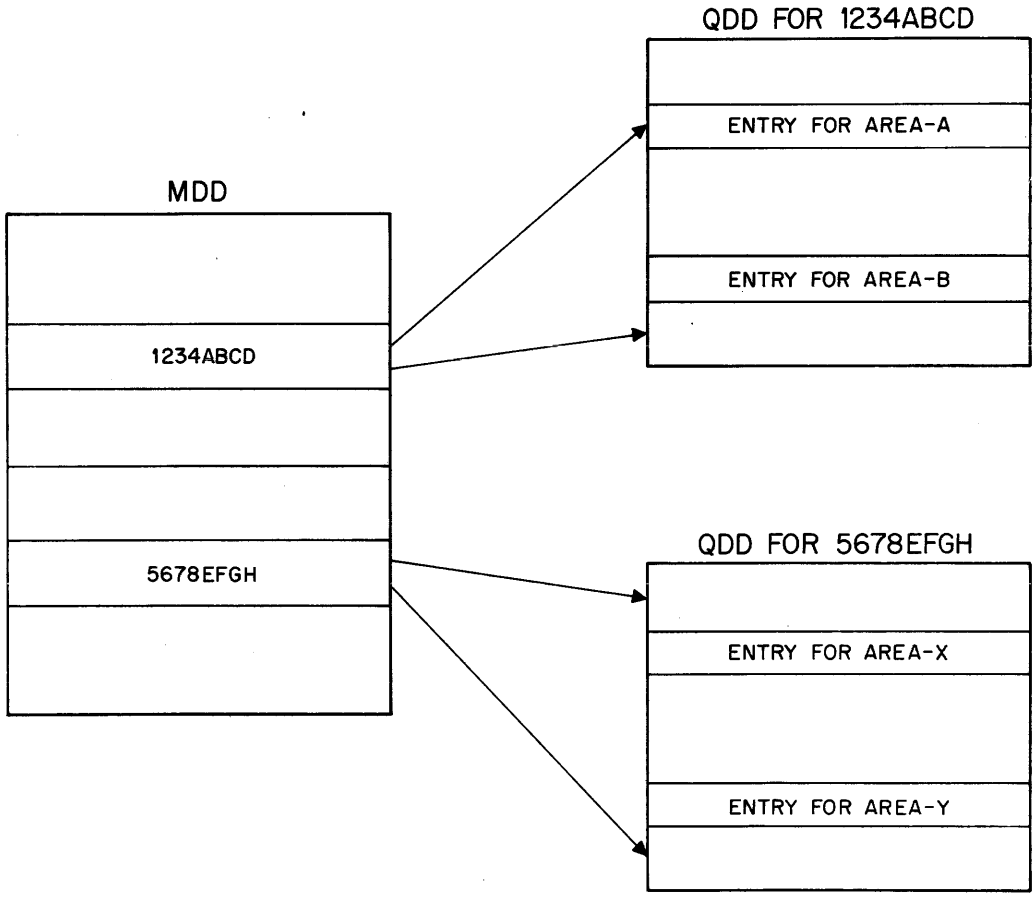



**Figure 6-4. Disc Directory Structure**

## 6-2 DISC AREA TYPES

### 6-2.1 Blocked Disc Areas

The term "blocking" usually describes a scheme whereby a fixed number of (logical) records are handled as a single (physical) record for purposes of I/O. That is, although a program requests that only one record be read, an entire block may be physically retrieved, stored in a blocking buffer in memory, and only the requested logical record passed onto the program. Most of the time, though, a read request does not require any disc access at all but only a memory-to-memory transfer of the logical record from the blocking buffer to the program's logical record buffer. Similarly, when writing, records are stored in a blocking buffer and are written to disc only when the block is full. The number of records in a block is referred to as the blocking factor.

The advantage of blocking is that it significantly reduces the overhead associated with I/O. In a disc-oriented multi-programming environment, where several programs are sharing a disc, this savings is extremely important.

Under VULCAN, the term "blocking" has a slightly different meaning when applied to disc areas. Disc areas are blocked, not by logical records, but rather by sectors. The blocking factor determines how many sectors are read or written at a time. The number of records in a block can vary from block to block.

Data is stored in blocks in an efficient manner that uses every available bit of disc space. If a block can only hold 3-1/2 records, then the rest of the fourth record is stored at the beginning of the next block. Logical records can cross sector boundaries in a similar fashion, but a block always starts on a new sector.

When a blocked disc file contains ASCII, or symbolic, data as opposed to binary data, blank compression takes place. Any string of two or more blanks is converted into a one-byte blank count. Blank compression can produce a rather dramatic saving of disc space. (See Figure 6-5.)

Both blocking and blank compression are completely transparent to the user. All records coming from or going to a blocked disc area automatically pass through the VULCAN Blocked File Handler which does all the required housekeeping behind the scenes.

Blocked records are addressed by their relative record number within an area.

### 6-2.2 Unblocked Disc Areas

When records are stored in an unblocked disc area, each record begins on a new sector. Unblocked disc areas are used for direct access files such as indexed sequential files and COBOL SORT work files. Executable program load modules, which have a special format, are also stored on unblocked files.

Unblocked records are addressed by relative sector number.

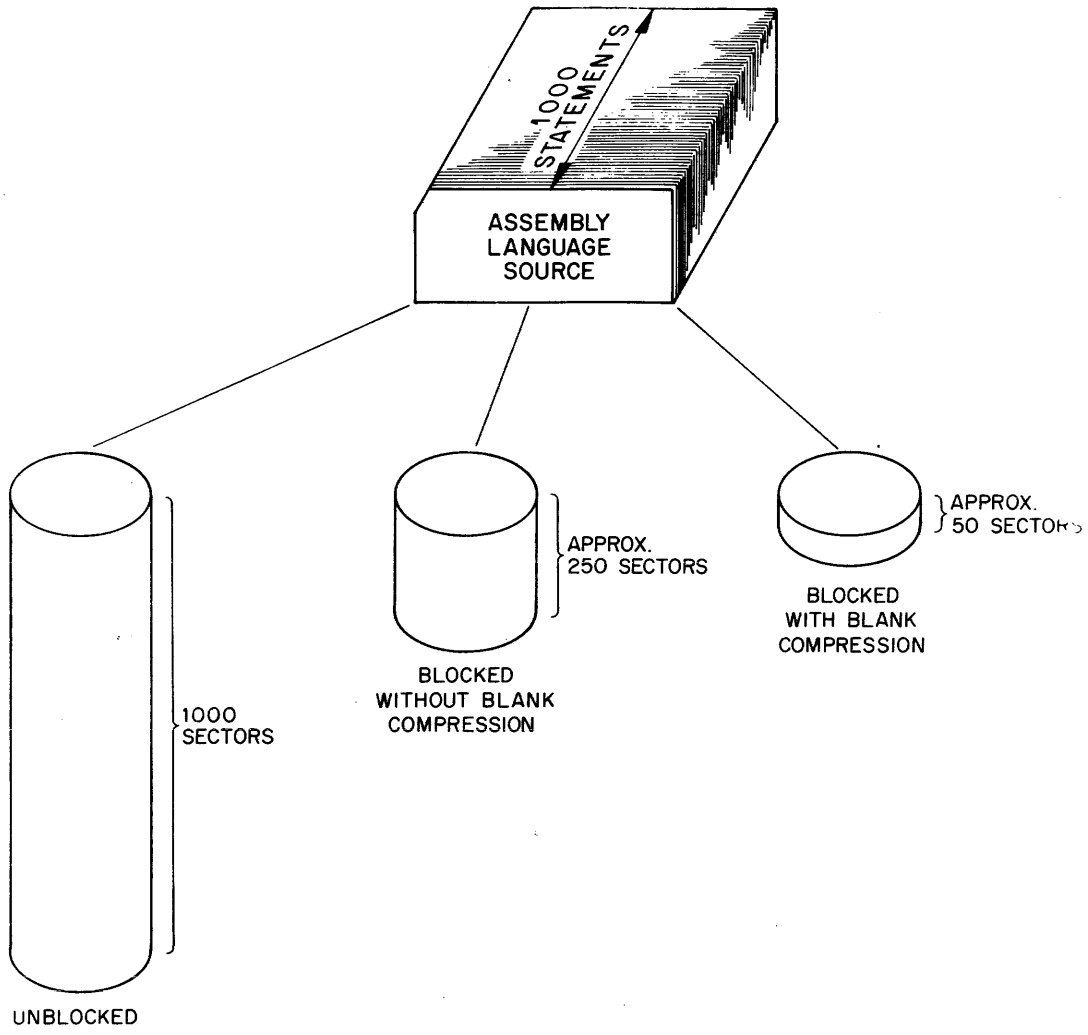



**Figure 6-5. Comparison of Data Density among Disc Area Types**

### 6-2.3 Concurrent Access Files (Random Disc Areas)

Both blocked and unblocked disc areas are provided with concurrent read/write protection. That is, although any number of programs can concurrently read a disc area, no writing can be done as long as any program is reading the area. If a program is writing to a disc area, no other program can read it or write on it until the program doing the writing is finished using the area.

There is one exception to this rule. There is a special type of unblocked area called a random area. It is identical to a regular unblocked area in every respect except that more than one program can read and write to it concurrently.

### 6-2.4 VULCAN Temporary Work Areas

Certain work areas are automatically supplied to interactive terminals and control point jobs when they are needed. They are temporary because they are automatically eliminated when a terminal is signed off or when a control point job terminates. Some of these areas are blocked and some are unblocked.

Temporary work areas are identified by two-character names such as W1 (Work area 1) and LO (List Output). However, the number of the terminal or the letter of the control point is appended transparently to each work area name to make the name unique. Therefore, there is a set of VULCAN work areas available to each terminal.

## 6-3 DISC AREA SECURITY

VULCAN supports three disc area security schemes. Any combination of these security schemes may be used.

### 6-3.1 The Qualifier

A qualifier, part of the identification of each interactive or control point user, is also an integral part of the name of every disc area. Whenever an interactive terminal user or a control point job references a disc area by areaname but does not explicitly state a qualifier, the sign-on qualifier is automatically appended to the areaname. To access any disc area having another qualifier, the full qualifier must be supplied.

The full format of an areaname with an explicit qualifier is

qualifier*areaname

An asterisk without a qualifier preceding it implies a special system qualifier 0000SYST. For example,

*VULCFILE is equivalent to 0000SYST*VULCFILE

### 6-3.2 Access Bits

When a user generates a disc area, the user i.d. that was supplied at sign-on time is used to identify the "owner" of the disc area. The owner of a disc area always has read access to the area (that is, the owner can read it) and, for programs, the owner always has execute access to the program (the owner can always execute it). In addition, when an area is first generated, the owner has write access and delete access, either or both of which the owner can take away and later restore.

The owner of a disc area can also grant any combination of read, write, execute and delete access to other users who have the same account number (the first four digits of the qualifier) as that associated with the area. Or, the owner of a disc area can grant any access combination to all users of the system. Regardless of the accesses granted to other users, only the owner of the disc area can alter the access states, or access bits as they are called.

The states of the access bits are represented by two-character codes as illustrated in Table 6-1.

**Table 6-1. Access Bits**

| | OWNER | ACCOUNT | PUBLIC |
|---|---|---|---|
| READ | * | AR | PR |
| WRITE | OW | AW | PW |
| EXECUTE | * | AX | PX |
| DELETE | OD | AD | PD |

* The owner can always read or execute his own disc areas.

### 6-3.3 Access Level

An access level is a number which, like qualifiers, is associated both with users and disc areas. An access level can have a value from zero to fifteen, with fifteen being associated with the highest access privilege. The access level for each user is determined by the computer center staff at each installation.

When a disc area is generated, it can be given an access level equal to or less than the owner's. To access a disc area, a user must have an access level equal to or greater than that of the disc area – regardless of the settings of the access bits.

By using the access level, access can be restricted to a subset of the "public" or a subset of the "account". For example, assume that all managers of a company have an access level of seven or more. A disc area with an access level of seven with public read access, then, can be read by all the managers in the company, not by the entire computer "public". If the disc area has account read access, then only managers with the appropriate account number can read the area.

## 6-4 DISC AREA BACKUP AND RETRIEVAL

### 6-4.1 KEEP (Backup)

Disc areas can be backed up to an auxiliary medium such as magnetic tape through the use of the KEEP command. The format of the KEEP output contains periodic checksums to ensure the integrity of the data. The data can be checked with the VERIFY command; data being retrieved is automatically verified.

Areas to be "kept" can be specified in a myriad of ways. Most of the following specifications can be used in conjunction with one another to indicate the areas that fall within the intersection of the specifications. The use of some of the following specifications is restricted to authorized users only.

*User* – includes all areas belonging to the signed-on user.

*Individual names* – includes only the areas listed, e.g., KEEP FILE1, FILE2,FILE3. . .

*Pack number* — includes all areas on a specified pack or on several packs within a specified range.

*Last written date or time* — includes areas that were modified during the specified period.

*Generation date or time* — includes areas that were created during the specified period.

*Elimination date* — includes areas that have expired or will expire during the specified period.

*Areaname mask* — provides a "template" describing characteristics to which areanames must conform (e.g., a particular character or numeric character in a particular position) for inclusion.

*Exception mask* — causes the exclusion of areas that conform to the areaname "template".

*Access level* — includes areas having a particular access level or one that falls within a particular range.

*Name list* — includes areas with a particular name regardless of their qualifier.

*Name bounds* — includes areas with names that fall between the designated alphanumeric limits, regardless of their qualifier.

*Name exception* — excludes areas with the specified name regardless of their qualifier.

*Qualifier list* — includes only those areas which have the specified qualifiers.

*Qualifier bounds* — includes areas with qualifiers that fall between the designated alphanumeric limits.

*Qualifier exception* — excludes areas with the specified qualifier.

*Account list* — includes areas with a specified account number (first part of the qualifier).

*Account bounds* — includes areas with an account number that falls within the specified numeric limits.

*Account exception* — excludes areas with the specified account number.

*Type* — includes data or program areas that have the designated type. For data areas, the type is a user-supplied field which may have a number from zero to seven and an optional specification for blocked, unblocked and random areas. For program areas, the type indicates the type of program, such as interactive/control point program, real-time program, and monitor program.

The KEEP command provides information on every area that is "kept". Standard output includes areaname, type, pack and current size. Less or more information can optionally be requested about the disc areas. The output can be in order by qualifier, areaname, type, user i.d. or pack number. When more than one order is specified, the first order is the primary order, the second is the secondary order, etc.

### 6-4.2 FETCH (Retrieval)

The opposite of KEEP is FETCH. The FETCH command has exactly the same format and capabilities as the KEEP command except that it retrieves previously "kept" areas and restores them on disc. The checksums previously added to the data are also checked.

### 6-4.3 Incremental Backup

Each VULCAN installation can determine its own method for backing up data on its disc packs. One suggested method is the "incremental save" technique.

Using the "last-written date" parameter, only those areas that have been modified since the date and time of the last backup are "kept" periodically (e.g., daily or weekly). If disc areas are destroyed and restoration is necessary, all areas on the most recent tape are "fetched" first, then those on the next most recent tape, etc. Since VULCAN does not "fetch" areas that already exist, the net result is restoration with the most up-to-date copy of each disc area.

### 6-4.4 MAP

The MAP command provides information about disc areas. It has the same format as the KEEP and FETCH commands and the same capabilities for selecting disc areas.

# VII. OPTIONAL SOFTWARE

## 7-1 THE TOTAL DATA BASE MANAGEMENT SYSTEM

The TOTAL Data Base Management System is the most widely used Data Base Management system, having over 1200 installations world-wide. It is known for its low memory, disc, and CPU time overhead; its ease of use; and because it is extremely fast to implement. Because TOTAL provides data and program independence, changes to a data base do not affect previously written user programs and vice versa. TOTAL is also open-ended meaning that new applications can be added to the existing data base as these applications arise. TOTAL provides complete logging, backup, and retrieval facilities as well. With TOTAL, data bases can be accessed by programs written in COBOL, FORTRAN, BASIC, RPG II and assembly language.

### 7-1.1 Data Base Files

A TOTAL data base is composed of a group of files. Each file is one of two types — a master file, or master data set, and a variable file, or variable data set.

Each record within a master data set contains a unique control key such as a part number, a student i.d. code, a social security number, or a customer code. Each record also contains information that is unique to that control key, such as name and address or part description. The control key is passed through a randomizing algorithm to determine the location of the record on disc. Master data sets are independent in that, although they may have relationships with other data sets, master records can be accessed without regard to those other data sets.

In a variable data set, control keys are not unique to each record; there can be several records with the same control key. For example, a customer transaction data set can contain several records for any customer reflecting several transactions over a period of time. Or a student course data set can contain one record for each course in which a particular student is enrolled. The associated master data set keeps track of the first and last variable record for each control key. Variable data sets, therefore, are dependent upon a master data set.

### 7-1.2 Data Base Networks

Master data sets can be associated with more than one variable data set. For example, a customer master record can have associated order number records in an order number variable data set and associated item number records in an item number variable data set. Another example is a university course master data set which is associated with an enrollment variable data set and a schedule variable data set.

In addition, each variable data set can be associated with more than one master data set. For example, a transaction variable data set can be associated with a customer master data set and with a master data set containing commodity information. Or, a university transcript variable data set containing several records for each student can be associated with a student master data set and with a course master data set.

A data base that is composed of master data sets with multiple associated variable data sets and variable data sets associated with multiple master data sets forms a network data base structure.

### 7-1.3 Data Base Definition and Formatting

The first step in creating a data base is to define, or describe, the data base to the TOTAL Data Base Management System. It is not necessary to define the entire data base at one time – one or more data sets can be defined and added to the data base system in a piecemeal fashion, thus making it possible to get the most critical applications running while others are still being analyzed.

The data base is defined through the use of a Data Base Definition Language (DBDL). English-like DBDL statements are analyzed by the DBDL Processor which converts them into assembly language source statements. These are then, in turn, assembled by the Harris Macro Assembler creating a Data Base Descriptor Module. The Data Base Descriptor Module is first linked to the Data Base Format program (DBFMT) which prepares the physical disc space on which the actual data base is to reside. Then, the Data Base Descriptor Module and TOTAL are linked to user programs. (See Figure 7-1.)

### 7-1.4 Accessing the Data Base

User programs use the Data Management Language (DML) to access the data base. DML commands are implemented via a CALL statement, or its equivalent, from the host language. The DML uses the facilities of TOTAL and VULCAN to validate the command and to do the physical access to the data base.

If an invalid command is encountered, the data base is left in its original state, and a diagnostic message is returned to the user program where it may be analyzed and an appropriate action taken. DML functions include several methods for reading, writing, adding, and deleting master and variable data set records.

User programs are independent from the data base. Therefore, changes to user programs do not require modification of the data base; and changes to the data base, such as the addition of a data set or a field, do not affect application programs.

### 7-1.5 Security

Data is password-protected to the field level. A user need only know the passwords to the fields required by a program. Although TOTAL may read an entire record as a result of a read command (it actually may read several records), only those fields specified in a program are passed to that program. The reverse is true with a write command.

A data set can be "locked" by a user such that no other user programs may access it while that program has it locked. Data sets are automatically locked in the case of an abnormal termination of a user program that was updating rather than just reading the data set, since an abnormal termination of such a program may leave the data set in an unknown state. The data set remains locked until it is restored to a stable state through recovery procedures.

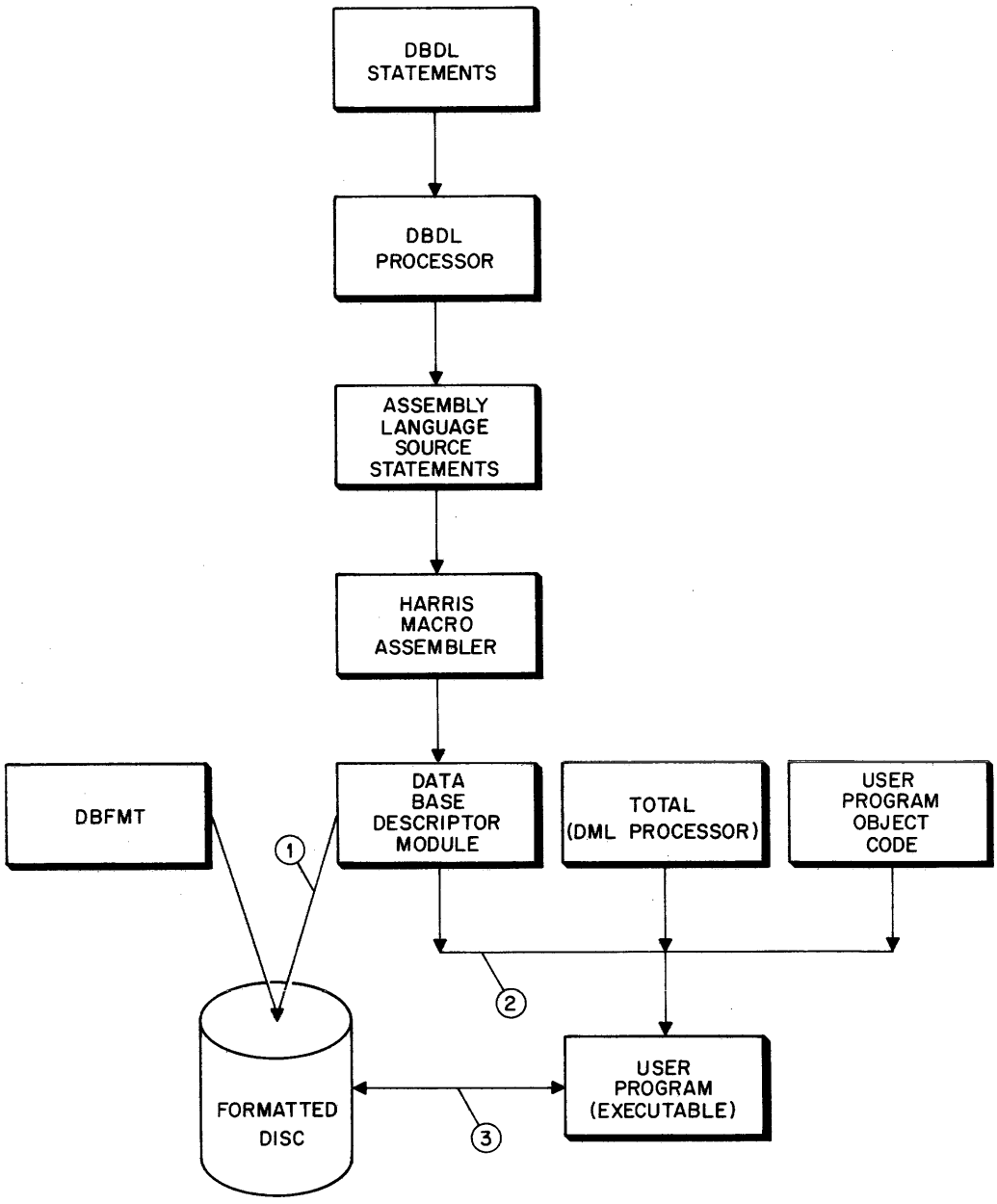



**Figure 7-1. Setting Up a Data Base**

### 7-1.6 Backup and Recovery

TOTAL incorporates three logging facilities which facilitate the restoration of damaged data sets.

The *before-image logging* option automatically records updated records as they were before the updating took place. This log file can then be used to "backout" the data base to a previous, known state.

The *mark log* DML function permits the user to record any information on the log data set, such as transactions and comments. If transaction data is recorded, it can be used to update the data after it has been backed out without resubmitting the transactions.

The *log quiet* DML function forces all pending updates to the data base to be executed; puts an indication on the log data set that a "quiet point" has been reached; and can optionally be used like the mark log function to record user information on the log data set. If it becomes necessary to backout a master or variable data set, the data set need be backed out only to the last quiet point.

## 7-2 REMOTE JOB ENTRY (RJE) SUBSYSTEMS

VULCAN supports six optional RJE subsystems. Two of these provide VULCAN with a host capability and four permit sending jobs to a foreign host. (See Table 7-1 for a list of the protocols that are available.) It should be noted that VULCAN supports the concurrent operation of multiple RJE subsystems which can be communicating with more than one foreign site.

**Table 7-1. VULCAN RJE Subsystems**

| Host | Protocol |
|---|---|
| IBM 360/370 | 2780 and HASP |
| CDC 6000/7000 | UT-200 |
| UNIVAC 1100 | 1004 |
| HARRIS S100 | 2780 and HASP |
| HARRIS S200 | 2780 and HASP |

Sending jobs from VULCAN to a foreign host is considered to be just another batch or interactive function to VULCAN. Remote sites can submit standard control point jobs to VULCAN. The remote terminal can submit a variety of operator commands to VULCAN for processing just as VULCAN can send messages to the operators of foreign hosts.

No special terminals are required to submit an RJE job to a foreign host. Any interactive terminal will do. Jobs submitted through a card reader can initiate an RJE job as well.

When jobs are being submitted from VULCAN, it is not necessary to actually have communications established when the RJE job is initiated. RJE jobs will be queued until communications are established. At that time, they are sent either concurrent with or following (depending on the protocol) the transmission of output (from previous RJE jobs)

from the host site to VULCAN. (If long distance phone lines are the means of communications, this maximizing of line utilization can result in a significant savings of phone charges.) Output from a job submitted to a foreign host can be directed to a local line printer, card punch, or disc area. The user can also specify a program to be initiated after the output from a foreign host has been received.

## 7-3 BASIC-V

Harris BASIC-V (V as in Virtual) is a powerful, high-level programming language processor for educational, scientific, and business applications.

BASIC-V is an interactive, conversational language processor that uses simple English words, abbreviations, and familiar mathematical symbols to define and control programming operations. Harris BASIC-V is a superset of ANSI BASIC standard X3.60.

It is particularly well suited for time-sharing applications because the processor is reentrant and it generates reentrant code.

### 7-3.1 Modes of Operation

BASIC-V is a language processor system consisting of two, separate language processors. These language processors provide two modes of operation — incremental and compiler.

Incremental Mode

In the incremental mode, each line is analyzed for syntactic errors as it is entered. If a statement is entered without a line number, BASIC-V executes the statement immediately. The incremental mode is useful as a "super calculator" and also to query or modify variables. The incremental mode is useful for the interactive development of programs since BASIC-V has extensive debugging facilities. The incremental compiler technique used produces code that executes 10 to 15 times faster than most commercially available BASIC interpreters. However, as with an interpreter, changes to large programs can be performed rapidly.

Compiler Mode

In the compiler mode, BASIC-V generates link-ready code like that of the other standard Harris processors. This permits the BASIC-V programmer to call routines written in FORTRAN, COBOL, and Assembler. The VULCAN Indexed Sequential Package (VISP), Sort/Merge, and TOTAL Data Base Management System can also be accessed by the BASIC-V programs. The compiler mode executes three to four times faster than the incremental mode. Once the user has developed a BASIC-V program in the incremental mode, more efficient use of memory space and maximum execution speed can be obtained by processing the program in the compiler mode.

### 7-3.2 Naming Conventions

Names for variables and functions may be combinations of letters and digits — up to 31 characters in length — beginning with a letter. The usual restriction of using names consisting of a letter followed by an optional digit is not imposed.

### 7-3.3 Multiple Statements per Line and Lines per Statement

Multiple statements per line are permitted by separating each statement from the next with a colon. Multiple lines per statement are permitted by breaking the statement with a symbol at the end of the line. Multiple statements per line and multiple lines per statement may also be combined. Therefore, loops can be written in a single line and complex statements may be broken across several lines to reflect the structure of the statement.

### 7-3.4 I/O

Input and Output under BASIC-V can be transacted with the terminal, other physical devices, or disc areas.

Input may take place within the program, between the program and the user's terminal, or between the program and logical files.

Output in BASIC-V is performed with variations of the PRINT command. The output may be directed to the user's terminal or to a file or physical device. Expressions may be output in a specified format with the PRINT USING statement. The format specified is an exact image (picture) of the output line.

Record I/O deals with the handling of records composed of fixed-length fields, as COBOL I/O does. With certain statements the user can manipulate portions of the I/O buffer directly to provide very powerful I/O capabilities.

### 7-3.5 Mathematical and String Operations

BASIC-V supports both numeric and string data types. Numeric data types can be stored as floating point, integer, or complex numbers. Standard arithmetic and relational operators are supported for all numeric data types. Strings are a series of characters usually representing alphabetic character codes. Strings may be concatenated by the symbols + or &. Relational operators are supported for string data.

There are 29 intrinsic mathematical functions and 17 intrinsic string functions.

### 7-3.6 Arrays and Matrices

Array is a generic name for any ordered set of data: vector, matrix, and higher-order arrays. Arrays may be of any single data type. Array and matrix handling is greatly extended in BASIC-V. The size of the array is limited only by the maximum virtual memory storage. An array may have up to 255 dimensions.

Numeric arrays may be added, subtracted, multiplied, and divided — on an element-by-element basis — with the standard BASIC-V mathematical operators. The cross product of two conformable matrices can also be taken. Two BASIC-V functions are available to transpose arrays and invert matrices. Square matrices may be inverted. Both transposing and inverting of matrices may be done in place.

# VIII. SYSTEM OPERATION

## 8-1 THE OPERATOR'S FUNCTION

### 8-1.1 OPCOM – The Operator's Terminal

All VULCAN systems require a specific CRT, Harris model number 2310, to be dedicated to operator communications. This CRT is commonly called OPCOM. The operator's terminal accepts input commands from the operator which are different from those available at interactive user (i.e., non-OPCOM) terminals. Besides using its screen to display output, OPCOM also uses a light panel next to the screen to inform the operator of significant events requiring some action.

### 8-1.2 System Logs

The operating system keeps a series of log files which supply a history of computer usage and problems. These logs can be displayed on OPCOM and can be output to an alternate device, such as the line printer, as required. The logs are:

*Input log* – includes all commands that have been entered at the operator's console including all sign-on and sign-off messages (an alias is recorded rather than the user i.d. of the operator for security reasons).

*Message log* – includes all messages that have been sent to the operator's console either by the system or by system users.

*Terminal log* – includes information regarding the use of interactive terminals including when each terminal has been in use and by whom.

*Control point log* – includes information regarding the use of control points including a history of what jobs have been run, when run, by whom, and for how long.

### 8-1.3 Ease of Operating the System

The operator's function is very uncomplicated. Most of the operator's efforts are required to manage peripheral devices, e.g., replacing line printer paper, mounting a magnetic tape, etc. System initialization usually only requires the use of two CPU control panel switches and the keying in of the date and time; the procedure takes only a few seconds.

### 8-1.4 General Command Capabilities

Commands can be executed either one at a time as they are keyed in or, for frequently used operator command sequences, they can be executed off of a disc file. Below are the major classifications of operator commands and some of the capabilities that fall within each category.

*Program control* — abort, query status, hold, release, initiate (real-time only). Programs may be referenced by name, terminal number (for interactive programs), and control point.

*System and peripheral status* — system status, peripheral status, disc status (how much space is available on a pack), active program list, disc area map, peripheral down, peripheral up, terminate terminal (i.e., sign it off), bad sector, modify system, tape ready, disc pack ready. (The last two commands are not used frequently since VULCAN usually detects the mounting of a tape or disc that has been requested automatically.)

*Spool control commands* — hold, release, terminate, advance (e.g., skip some lines), backup (e.g., repeat some lines), keep, multiple copies, change priority, route output to another device, eliminate, postpone, spool queue display.

*Control point control* — control point list; job queue display; query, hold, release, change job priority; change number of control points.

*User number control* — add user, add qualifier, change user parameter, remove user, remove qualifier, remove user files.

*Diagnostic commands* — get address, patch address, sector display, modify disc, disc dump, modify program.

### 8-1.5 User Access to OPCOM

VULCAN does not accept any OPCOM commands until a user signs on to OPCOM with a user i.d. However, not all users can execute all operator commands. As a matter of fact, some users are not permitted to use OPCOM at all. It is up to the administrator of the computer center to decide who has which operator privileges, if any.

Access to operator commands is controlled through the use of access bit masks (see "User Control Parameters"). Each of several groups of OPCOM commands has a 24-bit word associated with it called an access bit mask. Each user also has an access bit mask. In order for the user to be able to use OPCOM and to use a command, at least one bit must be on in the user's access bit mask that is also on in the appropriate OPCOM access bit mask.

## 8-2 SYSTEM GENERATION (GENASYS)

A full system generation is required only to implement a new release of VULCAN or to accommodate major changes in the system configuration. System generation is accomplished through the stand-alone GENASYS program. This program, along with all the modules necessary to configure any VULCAN system, are available on magnetic tape.

System generation is an interactive procedure. After the GENASYS tape is loaded (through the use of one switch when the hardware bootstrap option is available), questions are output to any terminal in the system as designated by CPU control panel switches. The entire procedure can be executed in an interactive manner.

However, frequently few parameters need to be changed, if any, as in the case of a new VULCAN release. In that case, most system generation parameters can be stored on another medium, such as punched cards. During the procedure, GENASYS can be directed to get more input from the alternative medium.

At the end of the procedure, which usually takes ten to fifteen minutes, GENASYS optionally outputs a summary describing the hardware configuration of the system.

# APPENDIX A
# THE LANGUAGE PROCESSORS

## A-1 COBOL

Harris COBOL is a full high-level implementation of the American National Standard Institute COBOL X3.23–1974. Included is full support for the following features:

*Indexed-sequential I/O* – for the processing of indexed-sequential files

*COBOL source (COPY) library* – (see "The COBOL COPY Library" following)

*Segmentation* – for the establishment of overlay segments

*In-line sort/merge* – with the SORT and MERGE verbs

*Inter-program communciation* – with the CALL statement and argument passing

*Report-Writer* – for the generation of reports without the need for coding detailed procedures

### A-1.1 The COBOL COPY Library

The COBOL COPY Library consists of source statement modules which can be inserted in a COBOL source program via the COBOL COPY verb. The source statements in the library module specified by a COPY verb are then compiled along with the rest of the source program as if they had been an original part of it.

One type of module that is frequently placed in a COBOL COPY Library is a lengthy data description that is used in more than one program. Instead of re-coding the description for each program that needs it, one COPY statement is used. Besides the obvious convenience and time savings to the programmer, each description is accurate, up-to-date, and compatible with the data. Another module frequently kept in a COPY library is part of the Environment Division.

### A-1.2 Compiler Options

At compilation time, the user can specify a number of options to the COBOL compiler. Some of these options are used to optimize debugged programs or programs that do not use certain COBOL facilities. Other options control the content of the source listing (list errors only, generate object code listing, etc.). An option is also available which causes the compiler to check for syntax only without producing any object code.

As with the FORTRAN compiler, a programmer can insert statements, identified by a D in column 7, throughout a program which are not needed for proper execution of the program but are an aid to program debugging. The Debug option specifies that all statements with a D in column 7 are to be compiled. Without this option, all such statements are treated as comments.

## A-2 FORTRAN IV

The Harris Optimizing FORTRAN IV Compiler is an extended version of the 1966 ANSI-standard FORTRAN, level X3.9. The compiler's numerous extensions include structured FORTRAN, Encode/Decode, Buffer In/Buffer Out, and Free-Format I/O. Real-time capabilities are provided by the compiler's ability to access VULCAN's real-time System Services directly via a CALL statement.

### A-2.1 Structured FORTRAN

The Harris FORTRAN Compiler provides statements for constructing blocks of a structured program with minimal or no use of the GO TO statement and statement numbers. The compiler can produce an indented source listing reflecting the structure of the program thus making the source code easy to read and understand.

### A-2.2 Datapool

The Harris Datapool Definition Processor is an aid in the development of large scale software systems. It centralizes the definition of global or shared data items, thus eliminating a common source of error. Development time is often saved since programs do not need to be recompiled when changes are made to the Datapool Definition; at most they need to be re-linked by the linkage editor.

The Datapool Definition Processor accepts Harris FORTRAN declarative statements such as DIMENSION, COMMON, and EQUIVALENCE, and constructs a Datapool Directory area. Variables in the datapool may then be referenced by their standard names without regard to their locations.

### A-2.3 Other Extensions

*Access to VULCAN real-time services* — (see "System Services")

*Support for Workshop on Standardization of Industrial Computer Languages* (Purdue Extensions) — facilitates data acquisition and analog/digital conversions

*Random access I/O*

*Encode/Decode* — does memory-to-memory data conversion with I/O-like statements

*Asynchronous I/O* (Buffer In/Buffer Out) — does double-buffering and overlapped I/O and processes records of arbitrary length and format (e.g., a magnetic tape of unknown format)

*Free-Format I/O* – permits I/O without FORMAT statements

*END and ERR control functions* – specify an action to be taken when an end-of-file condition is detected or when an I/O or format error occurs

*Tab specifications* – provides the ability to position text or data on an output record, or to read input data starting at a specified position in the input record

*DATA statement extension* – permits array names to be used to refer to all elements in the array

*Octal constants* – interprets numbers preceded by an apostrophe as octal values (e.g., '100 = 64)

*Logical operator extensions* – defines logical bits and logical integers which may be shifted, rotated, and exclusive OR'ed

*Multiple subroutine returns* – allows different subroutine returns with different destinations

*RECUR statement* – facilitates writing recursive subprograms

*In-line assembly code* – maintains compatibility between variables, constants and statement numbers; requires only one control statement to enter assembly language and one to exit it

*Literal constants* – permits the use of literal constants in place of real or integer constants throughout a source program, not just in DATA or FORMAT statements

*Implied DO-loops in DATA statements*

*CPU timing routines* – provide accumulated time for program execution

*PAUSE/STOP message output*

*Pseudo-random number generators*

*Integer byte* – reduces by a factor of three to one the size of an array whose values range between 0 and 255

*Double Integer* – extends integer capability to the range -70,368,744,177,664 to +70,368,744,177,663

### A-2.4 Debugging Aids

A storage map and object listing are always available as is a Post-Mortem Dump and the symbolic debugger, VBUG. In addition, FORTRAN has some built-in debugging aids.

At compile time, the Walkback option can be set. If the program subsequently aborts during execution, the statement in which the error occurred is identified. Additionally, if the error occurred within a subprogram, the source location of the corresponding call is identified for up to ten levels of nested subprograms.

The programmer can also include in a program statements identified by a D in column 1 which are not needed for proper program execution but which are helpful for debugging. By setting a compile time option, these statements are either compiled or treated as comments. If more selective control over which statements are compiled is desired, the programmer can turn the debug and walkback options on and off through the use of in-line control statements. Or the programmer can use the conditional compilation facility. With this mechanism, blocks of statements are compiled or skipped depending on the settings of flags which are set at compilation time.

Once a program is debugged, an executable module can be easily created that does not contain the instructions associated with the debugging statements without actually editing out these statements. If, at a later time, changes are made to the source program, the debugging statements can easily be reinstituted.

### A-2.5 Control Features

The compiler is I/O independent and logical file-oriented so that all input and output functions (both compile time and execution time) can be assigned to the required physical devices by means of Job Control commands.

Many other control features are available to control the format and content of the program listing and the generated code (e.g., indented listing, object code listing, and IBM 026 to IBM 029 code conversion). These options can be set at compile time and most can be selectively controlled with in-line control statements within the body of the source program.

## A-3 BASIC

The Harris BASIC Compiler is an extended version of Dartmouth BASIC. It supports several features not often found in other implementations of the language. These include string manipulation, unlabeled statements, structured programming control statements, multiple statements on a line (including comments), real-time capabilities and formatted I/O.

### A-3.1 String Manipulation

Functions are available to concatenate, compare and determine the length of strings, and to manipulate substrings. Character string arrays can be created and strings within an array can be identified by up to three levels of subscripting. The EDIT and DE EDIT statements provide a method to transfer between strings and numbers — the EDIT statement converts a number to a string; the DE EDIT statement converts a string to a number.

String data can be read into a program just as any other variables are. Strings can also be formatted with format strings (see "Formatted I/O").

### A-3.2 Unlabeled Statements

Usually, all BASIC statements must have a sequence, or statement, number which determines the statement execution sequence of the program. Under VULCAN, however, the user has the option of not using such labels on all lines. The user can also have VULCAN supply consecutive statement numbers when the program is compiled. The use of unlabeled statements is particularly convenient when using structured programming control statements.

### A-3.3 Structured Programming Control Statements

Harris BASIC supports extensive control statements for the design of a structured program. The ability to have more than one statement on a line facilitates the use of these control statements.

The IF statement provides conditional branching that depends on the evaluation of one or more relational expressions.

```
Examples:  IF A < 3 GO TO . . .
           IF A = 3 DO . . .
                    .
                    .
                    .
           ELSE . . .
           IF A > 3.2 & B = 5 DO . . .  ELSE . . .
```

The FOR. . .NEXT statement provide an iterative capability (similar to that of FORTRAN DO-loops).

```
Example:  FOR I = 1 TO 25
          FOR J = 1 TO 5
          P(I,J) = I * J
          NEXT J
          NEXT I
```

The WHILE. . .ELIHW and REPEAT. . .UNTIL statements are similar to the FOR. . .NEXT statements except that the number of iterations is determined by a general condition rather than a loop count.

### A-3.4 Real-Time Capabilities

There are five statements which are used for inter-program communications. There is one statement (CALL) for passing control to an external subroutine; two statements for receiving a value from an external routine including values from the Scientific Arithmetic (hardware floating-point) Unit when available; and one statement for receiving an initiation parameter supplied when the BASIC program is initiated. Another statement (TRAP) allows the user to keep control of a program after it has been aborted by the system.

The CALL statement can be used to link to FORTRAN and Assembler subroutines, the FORTRAN library and the real-time services.

### A-3.5 Formatted I/O

Formatting gives the programmer finer control over the appearance of input and output data than does the standard free-format I/O. Format strings, which are used to describe the data, can be literals or can be separately defined in an IMAGE statement.

Any INPUT or PRINT statement can specify a format string with a USING clause. The format string specification can either be a literal or a statement number of an IMAGE statement.

### A-3.6 Compile-Time Options

Several compile-time options are available. These include options to control the content of the source listing (e.g., list errors only); to direct the compiler to check for syntax only and not produce any object code; to generate an object code listing; to permit unlabeled statements; to supply statement labels; and several debugging options. Some options can be selectively controlled within the body of the program.

## A-4 HARRIS MACRO ASSEMBLER

### A-4.1 Instruction Set

The Harris Macro Assembler supports over 600 mnemonics, most of which fall into the following categories:

Arithmetic
Branch
Compare
I/O
Logical
Priority Interrupt
Shift
Transfer
Bit Processor (contains seventeen bit manipulation instructions)
SAU* (Scientific Arithmetic Unit)

Operations can be performed on words, doublewords, bits, and bytes.

Assembly language source statements can be fixed-format or free-format. Free-format input is formatted on the assembly listing to aid legibility.

### A-4.2 Addressing Capabilities

Most instructions permit direct addressing of up to 96K bytes of logical memory. Several "long" branch instructions provide direct addressing to 192K bytes as do indexed and indirect addressing. Addressing of up to 768K bytes can be accomplished with indirect addressing.

### A-4.3 Data Formats

The Harris Macro Assembler supports the following internal data formats:

*Integer* (single and double precision)
*Floating-point* (single and double precision)
*Fixed-point* (scaled integers) (single and double precision)
*Alphanumeric text* (three ASCII characters per word)
*Truncated alphanumeric text* (four truncated ASCII characters per word)
*Address constants* (for words and bytes)

* The SAU, optional on some systems, provides hardware floating point capability.

### A-4.4 Conditional Assembly

Twenty-seven pseudo-operations are available for skipping blocks of assembly language source statements or single statements. These pseudo-operations test various conditions of assembler macro operands (see below) as well as external flags set at assembly time.

### A-4.5 Assembler Macros

An extensive macro capability provides a means for writing sophisticated generalized source statement routines from which the Harris Macro Assembler generates tailored source code. The generated source code is tailored according to operands supplied by the user at assembly time and becomes an integral part of any source program that invokes it. Assembler macros can be stored in an assembler macro library.

Assembler macros can have numeric and text pseudo-registers for operand manipulation. Extensive branching, looping, and conditional branching capabilities are also provided.

## A-5 RPG II

RPG stands for Report Program Generator and, as the name implies, the language is oriented towards the generation of reports. Harris offers a standard RPG II compiler.

RPG II is user-oriented and easy to learn. An extensive programming background is not required; the emphasis in RPG programming is on problem description rather than computer manipulation. A user can become proficient in the use of RPG II after writing only a few programs. Input and output formats are defined on special pre-printed specification forms as are all files to be used, calculations to be performed, etc.

### A-5.1 Debugging Aids

RPG has a DEBUG operation code which, when encountered, causes one or more debug records to be written to a file. The information in these records depends upon user specifications. They may contain such information as the settings of indicators (internal "switches" used in RPG programs) and the contents of a user-specified field.

Debug statements are treated as comments unless the Debug option is set at compilation time (see the next section). Therefore, such statements can be left in a debugged program without penalty in anticipation of changes to the program or in case any new problems are discovered while the program is being used in a production mode.

### A-5.2 Compile-Time Options

Four compile-time options are available with RPG. The "debug" option enables DEBUG statements in a program. If this option is not set, such statements are treated as comments. The Errors Only option suppresses the list output of all statements unless they contain errors. The Object option outputs an assembly language object code listing. The Suppress-Sequence-Check option suppresses the listing of warning messages that occur if the sequence numbers on source statements are out of order.

## A-6 FORGO

FORGO (Load and Go FORTRAN) is a diagnostic FORTRAN compiler which provides extensive compile-time checking as well as execution-time error checking to detect subtle run-time errors in a FORTRAN program. Such checking includes detection of illegal DO-loop parameters and subscripts that exceed array bounds as defined by their DIMENSION statements. These execution-time errors are precisely located within the source program for easier correction.

FORGO also contains two debugging aids. It can trace the execution of a program or any portion of a program, indicating branching and changes in user-specified variables. It can also provide a dump of variables at the termination of a program.

Throughput for small jobs is greatly increased under FORGO since it compiles directly into memory and eliminates subroutine link time. Large programs can be run under FORGO for debugging purposes and subsequently under FORTRAN to greatly decrease execution time.

FORGO statements are in standard FORTRAN syntax and accept all standard data types including integer, real, double precision, complex, logical, and literal. Unextended 1974 ANSI standard FORTRAN and some facilities of the 1975 standard are supported.

## A-7 SNOBOL 4

The SNOBOL language is designed to process and manipulate character strings. It is used in such areas as compilation techniques, machine simulation, symbolic mathematics, text preparation, natural language translation, linguistics and music analysis. Harris offers an extended version of this language interpreter. SNOBOL has operations for joining and separating strings, for testing their contents, and for making replacements in them. It can break a sentence string into phrases or words, or break up a formula into components and reassemble them into another format.

Different data types are available for use in string manipulation including: integer and real numbers for arithmetic functions; arrays and tables; names for indirect references; patterns for matching complicated string structures; and strings which themselves may be literals or variables.

Pattern matching is the examination of the subject string contents for a desired structure of characters. This operation can match: any one of a list of characters; an arbitrary number of repetitions of any pattern; a string consisting of all characters up to any of a list of specified characters; a string of specified length; any one character not in a specified list; a character in a specified position in a string; a string of characters up to a specified position; a string of characters consisting of the characters in any order from a specified string; an arbitrary string delimited by two other patterns; a parenthetically balanced string; a string consisting of the remaining characters of the subject string after other matches; a string consisting of any one of a set of alternative patterns or arbitrarily complex combinations of the above.

### A-7.1 Extensions

Extended features of Harris' implementation of SNOBOL include:

automatic recursion
user-defined datatypes
up to 15 user-redefinable binary operators
up to 14 user-redefinable unary operators
REWIND, BACKSPACE, ADVANCE, and ENDFILE functions for magnetic tape or disc I/O
user-defined functions
extensive functions to perform data value tests

### A-7.2 Diagnostic Aids

SNOBOL features two important diagnostic aids: DUMP and TRACE. DUMP can occur during program execution or either by error or normal termination, at the user's option. TRACE permits tracing of user-selected variables, function calls and returns, statement label references, keyword values, and any combination of these.

Selective tracing can be further facilitated by setting limits on the number of lines of trace desired and optionally executing user-defined tracing procedures.

# APPENDIX B
# VIRTUAL HARDWARE OPERATION

## B-1 THE VIRTUAL MEMORY REGISTERS

The hardware that supports VULCAN's virtual memory capabilities includes a variety of registers. These registers decrease the burden on the software of many of VULCAN's housekeeping requirements.

### B-1.1 The Addressing Registers

A number of registers are used to implement the virtual address mapping scheme. They are the Virtual Address Registers, or VARs; the Virtual Base Register, or VBR; and the Virtual Limit Register, or VLR.

The VARs are used to relate logical addresses with physical addresses. Each logical page of an active program is associated with one VAR, and VARs belonging to one program are contiguous. The single VBR indicates the first VAR belonging to the currently executing program, and the single VLR indicates how manyVARs are associated with that program.

The two high-order bits of each VAR contain status information about its associated page indicating whether or not the page is loaded and, if it is loaded, whether or not the page may be written on (i.e., whether or not it is reentrant). For a page that is loaded, the lower bits of its associated VAR contain the physical location of that page. The physical location is expressed as a memory page number. In a system with 192K bytes, or 64K words, of real memory, the memory page number can have a value between 0 and 63. (In actuality, the low end of the range is the first page above the operating system, which is loaded in lower memory.)

### B-1.2 The Page Access Registers

The Virtual Usage Registers, or VURs, and the Virtual Not-modified Registers, or VNRs, are one-bit registers. There is a VUR and a VNR associated with each page of real memory. Every time a memory page is accessed, whether it is accessed for reading, writing, or executing, its associated VUR is turned on (i.e., set). If the page is written on, i.e., modified, the associated VNR is turned on also. The VURs contain information that is needed to determine what page will get swapped out when a page swap is required. The VNRs help to determine whether a page to be swapped out needs to be copied to disc. More on the use of these registers can be found in "In Search of a Page".

## B-2 VIRTUAL MEMORY INSTRUCTIONS

The virtual memory instructions load and query virtual memory registers and implement the User Mode in the CPU. They can be executed only by monitor programs, Non-Resident Handlers, and System Services, i.e., they can only be executed when the CPU is in the Monitor Mode. They are not privileged instructions since having a privileged program is neither necessary nor sufficient to execute them.

## B-3 ADDRESS MAPPING

Following are the events that take place when mapping an address, i.e., converting a logical address to a physical address.

1. Any indirect or indexed addressing specifications are evaluated.

2. The high-order bits of the resulting address define the logical page address. This address is compared to the VAR count in the VLR (Virtual Limit Register). If the logical page address exceeds the VLR count, a memory protect violation has occurred, the instruction that is pending is not executed, a memory protect interrupt is generated, and the program is aborted.

3. The logical page address is added to the contents of the VBR (Virtual Base Register). The sum is the address of one of the VARs belonging to the program.

4. The low-order ten bits of the address contain the address of a word within the desired page. The contents of the indicated VAR, the actual physical address of the logical page in question, are then combined with the word address to form an effective memory address.

## B-4 IN SEARCH OF A PAGE

Following is a description of the logic involved in locating a page in memory in response to a demand page interrupt.

The Virtual Usage Registers (VURs) are read. For every VUR that is found set, a value is stored in an internal table in a page-corresponding slot. This value is the number of demand pages that have occurred since the last time the system was initialized; it is kept up-to-date by VULCAN. All VURs are reset.

Scanning of internal tables then begins starting with those entries that are associated with the top of physical memory. These tables keep track of which pages are empty, which are eligible for swapping, and which are ineligible. (Pages are ineligible if they are either part of a permanently resident real-time program or are currently involved in an I/O transfer, i.e., if they contain I/O buffers for a current I/O operation.) If an empty page is found, the search is terminated and the demand page is honored. If no empty page is found, a formula is evaluated for each eligible page which yields the relative eligibility of these pages for swapping. The "most eligible" page is swapped out.

There are two factors that are taken into consideration by the eligibility formula. One is the difference between the current demand page count and the one that was last stored for each page. This indicates how long ago each page was used (in a relative sense). The second factor is the execution priority of the program to which the page belongs. Reentrant pages have one-half of a priority point added to them, so they are less likely to be swapped out than is a non-reentrant page with the same priority and usage.

Once a page to be swapped out is selected, further information about the page is evaluated. If the page has not been swapped out since it was first loaded, it is copied to a "swap area" on disc. If the page has been swapped out before and is reentrant (and hence not modifiable) or if it is non-reentrant and has not been modified since it was loaded, no further action is required besides loading the new page in place of the old and updating the appropriate virtual memory registers. If the page was modified, it is again copied to the swap area before the new page is loaded.

# INDEX