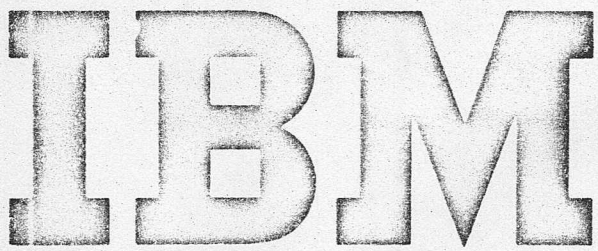
GH20-1029-0

Type III Class A Program



Control Program-67/Cambridge Monitor System (CP-67/CMS) Version 3.2

Program Number 360D-05.2.005

Operating Systems in a Virtual Machine

General guidelines for running S/360 operating systems under CP-67 are discussed in this manual. The main orientation is toward OS/360, although some other systems characteristics are mentioned. Specific information is given for running the CP-67 utility programs MINIDASD and SAVEOS.

This document is directed toward system programmers, and assumes knowledge of OS/360 as well as CP-67/CMS.

NOTICE

This Type III Program performs functions which may be fundamental to the operation and maintenance of a system. It has not been subjected to formal test by IBM.

Until program reclassification, IBM will provide:

- Central Programming Service, including design error correction and automatic distribution of corrections
- FE Programming Service, including:
  (1) Design error verification
  (2) APAR documentation and submission
  (3) Application of Program Temporary Fixes or development of an emergency bypass when required.

IBM does not guarantee service results or represent or warrant that all errors will be corrected.

The user is expected to make the final evaluation as to the usefulness of this program in his own environment.

THE FOREGOING IS IN LIEU OF ALL WARRANTIES EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

First Edition (September 1971)

This edition, together with Technical Newsletter GN20-2616, dated May 15,1973, applies to Version 3, Modification Level 2, of the Control Program-67/Cambridge Monitor System (360D-05.2.005) and to all subsequent versions and modifications until otherwise indicated in new editions or Technical Newsletters.

Information in this publication is subject to change. Therefore, be sure you have the latest edition and any pertinent Technical Newsletters.

Changes to text and illustrations are indicated by a vertical line to the left of the change.

Requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form has been provided at the back of this publication for readers' comments. If this form has been removed, address comments to: IBM Corporation, VM/370 Publications, 24 New England Executive Park, Burlington, Massachussetts, 01803.

TABLE OF CONTENTS

FIGURES

PREFACE

Information relative to this manual is contained in the following IBM publications:

IBM System/360 Operating System: System Generation, GC28-6554

IBM System/360 Operating System: System Programmer's Guide, GC28-6550

IBM System/360 Operating System: Storage Estimates, GC28-6551

IBM System/360 Operating System: Utilities, GC28-6586

IBM System/360 Operating System: Initial Program Load and Nucleus Initialization Program PLM, CY28-6661

IBM Control Program-67: Operator's Guide, GH20-0856

IBM Control Program-67/Cambridge Monitor System: Installation Guide, GH20-0857

IBM Control Program-67/Cambridge Monitor System: User's Guide, GH20-0859

IBM Control Program-67: Program Logic Manual, GY20-0590

IBM Control Program-67: Online COBOL Symbolic Debug: Application Description, GH20-0920

INTRODUCTION

A virtual machine as created by CP-67 is a functional simulation of a real IBM System/360 and its associated I/O devices; it is comprised of virtual memory, a virtual CPU, virtual I/O devices, and a (remote) terminal acting as an operator's console. Any System/360 operating system can control any given virtual machine, so long as none of the code includes timing dependencies or dynamically modified channel programs. Popularly used System/360 operating systems which have run under CP-67 include OS/360, DOS, and PS44.

The virtual machine environment created by CP-67 provides great flexibility to installations which use other System 360 operating systems, whether for equipment or application reasons. This document discusses those benefits gained from running these systems under CP-67 control. It defines techniques for preparing job streams for the virtual machine operating system, and discusses other operational and performance characteristics. It documents certain CP-67 utility programs which provide operational and set-up aids to installations which use OS/360 or DOS in a virtual machine.

The CP-67 environment of multiple virtual machines permits multiple versions of OS/360 to be executing concurrently. One virtual machine may be controlled by a Release 19 PCP system, another by the release which is normally operational at the installation, another by a new release still in the process of production test. The back-release system may be running an application which requires extensive system modifications and is difficult to upgrade; it is practical to assign that application to a virtual machine under CP-67, and to achieve concurrency of operation between it and other systems through CP-67 time sharing facilities. A current release system may be set up to run production work, while other virtual machines are executing in a terminal-oriented time-sharing mode; OS/360 and CMS might be run this way during the day, and OS run at night on the real S/360 Model 67 in 65 mode. A current release system with new PTFs may be running as a test system. Similarly, a new release or option may be completely generated and tested on a virtual machine. Two points should be noted: testing PTFs to, and new releases of, OS/360 in a virtual machine is effective and accurate; each virtual machine is completely independent of any other virtual machines that may exist concurrently.

The environment of multiple virtual machines permits different configurations of OS/360 to be run concurrently. An MFT system may be run on one, an MVT system on another. Thus application program development which requires MVT may

use it, while other virtual machines are using MFT or PCP as most applicable.

When there are multiple virtual machines set up to do OS/360 work, each can be loaded with the same copy of OS. This is the environment demanded by the CP-67 Online COBOL Symbolic Debug Program RPQ, program number 5799-AAE; each user interactively debugs his programs on his own virtual machine, and each machine runs under control of the same version of OS. A single copy of the OS system is set up on disk, to be shared by all users; each virtual machine which IPLs this shared system must include adequate virtual memory space for its core residence. This manual discusses shared systems residence volumes in the section "OS/360 as a Saved, Read-Only System".

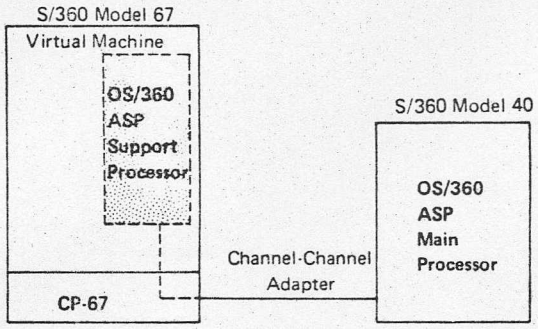


Figure 1. A Virtual Machine in an ASP Configuration

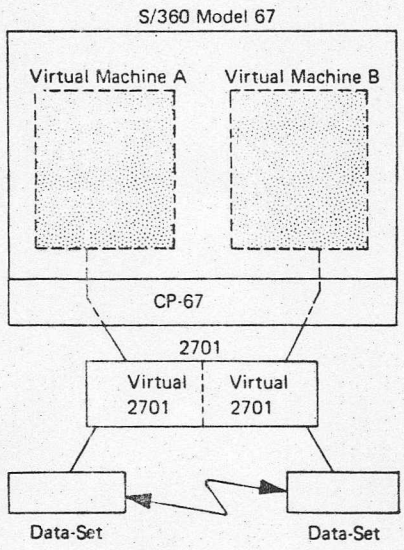


Figure 2. Virtual Machines for a Communications System Test

Virtual device definition permits great flexibility in system design and system test. An ASP system can be generated to run one processor on a virtual machine, the other on a real machine; see Figure 1. A communications system can be tested using multiple virtual machines in place of multiple real machines, as shown in Figure 2. The virtual 2701's could each be defined as a one-line 2701, while on the real machine there exists a single two-line 2701.

Execution of OS/360 in a virtual machine provides several facilities either difficult, impractical, or impossible to attain on a real machine. Some of these are listed below; the lists are grouped by system programming facilities, application programming facilities, and operational facilities.

OS/360 System Programming Facilities:

1. testing of new or modified SVC's on a virtual machine
2. applying and testing PTF's on a virtual machine
3. generating and testing new releases of OS/360
4. debugging on a virtual machine at the (pseudo) operator's console

OS/360 Application Programming Facilities:

1. debugging while under OS/360 control, from a terminal
2. designing of application programs without real memory constraints

OS/360 Operations Facilities:

1. training of operators on a virtual machine
2. defining a virtual machine and its devices as back up to a real machine
3. defining minidisks for improved DASD utilization

The above discussion has dealt with running various OS/360 configurations under CP-67. All of the points illustrated apply equally well to DOS. Further, the environment of multiple virtual machines permits DOS and OS/360 to run concurrently.

A fairly typical CP-67 system might normally operate with twenty virtual machines. Of these, two might be configured for multiprogramming and batch mode operation where one

virtual machine is controlled by DOS and the other by OS/360. Perhaps, on the average, three other virtual machines are used for system programmer development work; whether active for DOS or OS/360 work, or both, depends upon current installation concerns. In all likelihood the remaining virtual machine would be providing time sharing services under control of CMS, the Cambridge Monitor System.

The installation which uses CP-67 and CMS primarily as a multi-access time-sharing system and only secondarily for an OS/360 backup or test system will probably not require a copy of OS/360 as a shared system which can be IPL-ed by name; thus the section "OS/360 as a Saved, Read-Only System" could be bypassed. Performance considerations are discussed under "Operation of OS/360 in a Virtual Machine", and under "Paging Considerations". How to flip-flop between CMS and OS in a virtual machine, how to run OS from a virtual machine, and how to use minidisks as OS/360 DASD volumes are discussed in this manual. These topics should be of interest to any installation which runs OS under CP-67.

Certain points discussed in the remainder of this manual do not apply to the operation of DOS in a virtual machine. When this is true, and not apparent from the subject, it will be so stated. DOS systems prior to Release 25 cannot be set up as read-only, and no CP-67 utility programs exist to save a page-form copy; the entire section "OS/360 as a Saved, Read-Only System" consequently applies to OS/360 only.

In addition DOS users should be aware that the language processors available under CMS are OS/360 level compilers with interfaces rewritten for execution in the CMS environment. The application programmer facilities listed above are enhanced for the OS/360 user by the option of executing compilations and tests under CMS. The DOS user may use CMS to create and maintain source program files, and, via the flip-flop technique discussed in "Preparing OS/360 Job Streams", use DOS for program compilation and test. The flip-flop technique is enhanced for DOS users by specifying the ipl-commands through the card reader; this is not possible with OS/360, and is one of the justifications for setting up a named copy of that system for use in CP.

OPERATION OF OS/360 IN A VIRTUAL MACHINE

OPERATING GUIDELINES

OS/360 may be executed without modification on a virtual machine created and controlled by CP-67. PCP, MFT, or MVT will run provided that the host virtual machine has a configuration compatible with the one for which the operating system was generated.

The operator first logs into CP from his terminal. Either the OS/360 systems residence volume must have been defined as a virtual device in this operator's virtual machine directory, or the real device on which it is physically located must now be ATTACHed to the operator's virtual machine. He then issues the IPL command, giving as the ipl-device the virtual device address of the OS sysres volume. From that point, operation in the virtual machine is controlled by OS. The differences between operating on a real machine and operating under control of CP-67 are 1) the console functions of CP are made available whenever the attention key is depressed once, and 2) the request button of a real operator's console is simulated by two depressions of the attention key.

To reply to OS/360 messages, or to enter OS/360 commands, hit the attention key once to enter CP mode, and a second time to simulate the request key on a real operator's console. The second attention interrupt will be lost if it is entered before the status message "CP" appears. However, if the SET ATTN ON/OFF command has been issued this procedure will change; in that case, a single attention will simulate the console request key. If this command is in effect, CP functions are not available to the virtual machine except through use of the DIAGNOSE instruction; this implies that the facilities listed below will generally not be available to the user.

The OS/360 command processor does not accept line delete or character delete symbols; the real machine operator uses the cancel key on his console whenever he must correct input errors. The function of the cancel key is not simulated by CP-67. To enable line and character corrections to operating system commands, the CP-67 command SET LINEDIT ON/OFF should be issued. Under control of this command, CP-67 will edit all terminal input lines to the virtual machine; it will interpret the @ and ¢ characters as character delete and line delete symbols respectively.

When OS/360 requests that a device be mounted, a frequent

response on a real machine is to physically take the device out of and then back into ready status. This can also be done when running virtually if the device is a real device ATTACHed to the virtual machine. From a remote terminal, or when using minidisks, the equivalent procedure is to hit the attention key to enter CP mode, and to enter the command READY CUU followed by the command BEGIN. The virtual machine will resume execution, and CP will present to it a device end interrupt for the device specified in the CUU parameter.

During system execution on a real machine it is often reassuring to glance at the flashing lights and, from them, to determine that the machine is not in a wait state. When operating with a virtual machine the user can enter CP mode and issue the DISPLAY PSW console function. If the virtual machine is in a wait state, the wait state bit in the PSW will be on. If not, a BEGIN command will resume virtual machine execution. If the virtual machine is looping, a series of 1) enter CP, 2) issue DISPLAY PSW, and 3) issue BEGIN will soon indicate it.

Often tape devices are not defined in the directory as part of the virtual machine configuration. If that is so, and a tape device is required for a job running on the virtual machine, the operator must ATTACH an available real drive to the virtual machine; the device can be ATTACHed as any suitable virtual device address. When the job involving tape completes the user should issue a DETACH, naming the virtual device address. In this way, real tape drives are allocated to a virtual machine only during a period of use.

An external interrupt is simulated on the virtual machine by first entering CP mode and then entering the command EXTERNAL. This is useful only if the executing programs have set up external interrupt routines in the OS/360 system; e.g., the Online COBOL Symbolic Debug Program RPQ uses the external interrupt to allow debug mode to be entered at any time.

Refer to the section "Preparing OS/360 Job Streams" for information on handling the OS reader.

OPERATIONAL CONSIDERATIONS

CP-67 does not include any channel switching algorithm. If the real configuration includes 2816 tape switching units, these can be made to operate under control of a virtual machine's operating system; e.g. if 580 and 680 are the alternate device addresses for a particular drive, then:

1. Sysgen OS/360 for a 2816 tape switch on channels 5 and 6.

2. Sysgen CP-67 as though 580 and 680 are different devices.

3. Issue the ATTACH CP-67 operator command for both device addresses (580 and 680) whenever the real device is to be ATTACHed to the virtual machine.

If the virtual printer is not a real printer, CP-67 will spool the output directed to it onto disk and schedule it for later printing. When printer spooling is performed by CP-67, the CP console function CLOSE, specifying the virtual printer address, should be issued periodically. This will release currently stacked output to the spooling program and thus tend to avoid overflow of the CP spooling areas.

The virtual machine operating system can communicate with CP-67 during execution through use of the DIAGNOSE instruction. The console functions of CP-67 can be invoked from the virtual machine by issuing DIAGNOSE with a code of 8, and passing in R1 and R2 respectively the console function request text and its length. (Refer to the CP-67 PLM for further details.) The installation may wish to include in its operating system a request to CLOSE the printer after each job step, thereby making the current CP spooling area available for real printer output without operator intervention.

Use of the DIAGNOSE instruction can be conditioned by a simple test of the environment of the OS/360 machine. The address 0FF is assumed to be available for CP's real time clock. If a TESTIO to device 0FF indicates that the device is present, then the machine is a virtual one.

PERFORMANCE GUIDELINES

Introduction

The performance characteristics of OS/360 when it is run in a virtual machine are variable. Thruput comparisons between OS/360 running under CP-67 on a System 360 Model 67 and OS/360 running stand-alone on a Model 65 yield results that vary from 1.4 times the stand-alone time to over 4. This tremendous variability is a result of several factors: the total number of virtual machines in execution; the type of work being done by each virtual machine; the type and capacity of the primary paging devices; the amount of real core available. The topics discussed or introduced in this section address those software options that will affect the execution of OS/360 in the virtual machine environment.

As the control program for the real machine, CP-67 initially

processes all real interrupts and intercepts all privileged instructions. The amount of work to be done by CP for a virtual machine initiated interrupt or intercept depends upon its type. An SVC interrupt is simply reflected back to the virtual machine whereas an SIO privileged instruction initiates extensive CP-67 processing. Reducing the number of I/O operations in the virtual machine will improve its performance by decreasing CP overload.

The number of I/O operations done while executing under control of OS/360 can be reduced in several ways for both system and user code. The use of preallocated data sets will reduce initiator time. The use of blocking for user data sets will reduce problem program execution time. The use of chained scheduling will reduce the number of SIO instructions, although the chain of channel commands will still have to be translated.

The planned use of virtual memory will reduce execution time in both OS/360 code and problem program code. This latter is fully discussed in the section "Paging Considerations".

Multi-tasking Systems under CP-67

Virtual machines that are created by CP-67 are controlled in such a way as to provide concurrency of operations. When OS/360 is run on a virtual machine its resource algorithms interact with those of CP, as follows:

1. If during its execution an OS/360 created task must wait for a CP-provided service, the virtual machine is marked non-dispatchable. An example of such a wait condition is a page wait. Other tasks which might exist in the OS/360 controlled virtual machine cannot be dispatched by OS until the virtual machine resumes execution.

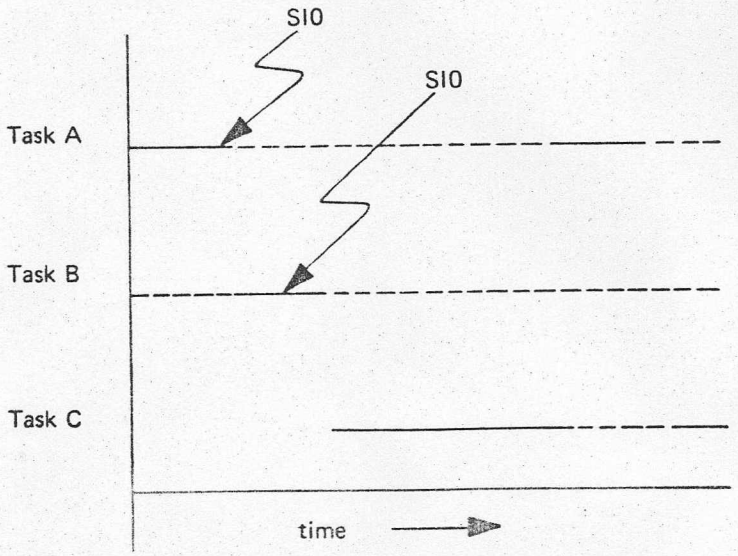


Figure 3. Concurrencies in Real Machine Operation

2. Figure 3 illustrates the relationships of executing tasks on a real machine; when Task A is waiting for an I/O operation to complete, the lower priority tasks are given use of the CPU. Under CP-67, the I/O operations of a particular virtual machine are overlapped with the CPU execution of that and other virtual machines. Consequently lower priority tasks created by OS/360 are given the CPU resource less frequently when executing in a virtual machine than when executing in a real machine. It is as though the I/O operations were "compressed" when done on the virtual machine.

Often an installation which requires concurrent task execution will discover a performance improvement when running single-task systems on multiple virtual machines. However, strong reasons for choosing a multi-tasking system (MFT or MVT) may exist. If the OS system is run on a virtual machine during the day and on a real machine at night, there are clear operational advantages to using a single configuration of OS. Also, a programming development project may require multi-tasking facilities; for proper program test a multi-tasking system must be run on the virtual machine assigned to that project.

Virtual Machine Options

CP-67 provides two optional services to virtual machines which directly affect the amount of CP overhead incurred by those virtual machines. These options are the ISAM option and the RTIMR (Real Timer) option; to be later provided to selected virtual machines, these options must be selected at CP-67 system generation. The ISAM option allows the virtual machine to execute certain types of self-modifying CCW command sequences, specifically those generated by the OS/360 ISAM modules. This option is not required for the proper functioning of ISAM in DOS. The Real Timer option provides updating of a virtual machine's timer when that virtual machine is in a self-imposed wait state.

These optional facilities should not be given to a userid unless they are required. For example, if a particular user requires the ISAM capability only occasionally, it may be well to assign to him two userids -- one with, and one without, the option; in this way, the additional overhead would be incurred only when it was essential.

CP-67 optionally provides an extensive interrupt trace facility. If this facility is to be used, it must be selected as an option in the CP-67 system generation process. (Refer to the CP-67 Installation Guide for details on the &TRACE(6) parameter.) This trace facility introduces no overhead in a virtual machine until such time as the SET TRACE command is issued. Since it is a selective facility,

specifying it is generally preferable to selecting the trace table option during an OS/360 system generation.

Printing in a Virtual Machine

CP-67 and OS/360 both provide unit record spooling. If the OS/360 is configured with HASP, additional spooling facilities are introduced. When OS/360 is run in a virtual machine, double spooling of printer-destined data files may occur. If a significant amount of printing is to be done, one of the spooling facilities should be eliminated. The following points should be noted:

1. If PCP is used, OS/360 performs no printer spooling.

2. If the JCL is of the form UNIT=00E instead of SYSOUT=A, no printer spooling will be performed by OS/360. The unit 00E refers to the virtual printer address.

3. If a real printer is ATTACHed to the virtual machine on which OS/360 is running, CP-67 will perform no printer spooling for that virtual machine. Printer interrupt processing can be initiated by the virtual machine whenever it receives control of the CPU from CP.

4. CP spooling has priority over any virtual machine execution.

If JCL is being set up for use specifically in a virtual machine, then it should be of the type UNIT=00E. In this way, overall performance will be enhanced both by the elimination of one spooling operation and by the shared use of a single printer device. Since CP spooling operations are given a high priority, the printer device is better utilized from CP than from a virtual machine.

It should be noted that OS/360 systems after Release 19.6 have the Direct SYSOUT Writer (DSO) facility. Use of this facility instead of the standard writer or HASP allows the user to use SYSOUT=A in his JCL and also eliminates one level of spooling. Refer to Operating System/360: System Programmer's Guide, Order No. GC28-6550, for a description of the DSO facility.

Virtual Machine System Considerations

CP-67 provides certain system facilities which affect the operational and performance characteristics of a virtual machine's operating system.

The operating system will IPL much faster if it is set up as a saved, or named, system for CP. A system is saved by writing a pageable copy of that system to disk from a

virtual machine; the system can be saved after its initialization work is completed. Thereafter, the system is IPL-ed by name, rather than by device. The user is freed from typing in that part of the IPL sequence which was done at the time the system was saved. OS/360 should be set up as a saved system if the virtual machine use is such that frequent IPL-ing will be done. Refer to the section "OS/360 as a Saved, Read-Only System". Note that the programs described there pertain to OS/360 only.

If multiple virtual machines will use OS/360 concurrently, the systems residence volume and library volumes should be set up as shared. The savings in DASD space requirements will be considerable. In the virtual machine environment, a shared systems volume implies that all the data sets of that volume be classed read-only to the general user population. Refer to the section "OS/360 as a Saved, Read-Only System".

If a given virtual machine will run OS/360 and good performance in that machine is especially required, certain pages should be LOCKed into real memory. If multiple virtual machines are running the same version of OS/360 and similar job characteristics exist, certain pages should be shared in real core. Both of these topics are discussed in "Paging Considerations for OS/360".

PREPARING OS/360 JOB STREAMS

If OS/360 is to be run as a closed-shop batch operation on a virtual machine, then it may be most convenient to dedicate a reader (and perhaps a printer) device to it on the real machine. In that case, jobs would be entered through the reader exactly as though the Operating System were running on a real machine. If however the OS is to be run from a remote user's terminal, then the following technique for preparing job streams will be useful.

The technique described below is a flip-flop approach. It alternates virtual machine operation between CMS and OS/360. It uses the CP-67 command IPL to accomplish the flip-flop.

CMS provides facilities for terminal-originated creation and maintenance of input files. These files may be transferred to a different operating system running on the same, or a different, virtual machine. For example, an OS/360 job stream may be created under CMS and then transferred to the card reader of an OS/360 virtual machine.

The CP-67 XFER command provides the means for transferring card decks from the card punch of one virtual machine to the card reader of that same or another virtual machine. Cards may be prepared under CMS and punched with XFER on; OS/360 should be IPL-ed; a START RDR command to the virtual card reader should be issued. During this process no real cards will be punched or read; CP-67 manages the transfer of data through disk operations only.

Many factors will determine the particular job stream transfer approach most frequently used by an installation. The following paragraphs illustrate three implementations of the flip-flop technique.

The first approach is more or less manual. It presumes the terminal user has created source program files under CMS. Perhaps he has used the CMS compilers with some of these files and has thus named his files according to the CMS compiler conventions, e.g. PL1MAIN PLI. To execute under OS/360 he must create files of JCL records which specify compilation, link edit, or execution, as appropriate. These would be created in CMS and named with an esoteric filename-filetype, e.g. PLICOMP JCL. Various source program files, job control record files, and data files can then be concatenated into the virtual card reader to form a single OS/360 job stream. A combination of CMS commands and CP-67 console functions accomplishes this job stream creation and transfer. When the OS/360 system is IPL-ed, a reader should be started to that virtual card reader address.

```
CMS
cp close 00d
cp close 00c
cp xfer 00d to *
closio punch off
offline punch jobcard jcl
offline punch pl1comp jcl
offline punch pl1main pli
offline punch asmcomp jcl
offline punch asmsub sysin
offline punch linkgo jcl
offline punch godata data
offline punch slshstar jcl
closio punch on
cp ipl 230
IEE007A READY
set date=71.200,Q=(233)
start rdr, 00c
start wtr, 00e
start
```

Figure 4. OS/360 Job Stream Transfer

Figure 4 is a specific example of this approach. The virtual machine is in CMS mode at the start of the example. CLOSIO commands have been used to concatenate the separately punched files into a single job stream file. Virtual device 230 is an OS/360 systems volume. All CMS responses (e.g. R;T=0.04/0.12 09.36.08) have been omitted from the example, however, the OS/360 READY message is included to more fully illustrate the IPL sequence. Of course, the attention key must be used prior to entering each OS/360 command, although this is not indicated in the figure.

After the OS messages READER CLOSED and READY appear, the user should enter CP mode by hitting the attention key and issue the CLOSE console function to the virtual card reader. This is required if there are multiple files in the card reader and is a good practice to follow in all cases.

If a flip-flop between CMS and OS/360 will be used frequently, a CMS EXEC procedure should be created and made available to users. The CP-67 Online COBOL Symbolic Debug Program RPQ includes such a procedure. This ALLPURP command, which is shown in Figure 5, requires that all files to be transferred to the virtual card reader have a filetype JOB. It presumes that the copy of OS/360 to be IPL-ed is a saved system to CP-67; in this way the IPL process is quickened, and the terminal user need not enter the OS/360 commands required by the IPL sequence.

ALLPURP is a simple EXEC procedure which enhances the basic flip-flop technique. It automatically transfers the OS/360 printer output to the user's virtual card reader so that,

```
&TYPEOUT OFF
&TIME
CP Q USERS
CP CLOSE 00E
CP CLOSE 00D
CP CLOSE 00C
CP PURGE RDR
CP PURGE PUN
CP PURGE PTR
CP XFER D TO *
CLOSIO PUNCH OFF
OFFLINE PUNCH &* JOB
CLOSIO PUNCH ON
CP READY 00C
CP READY 00E
CP XFER 00E TO *
CP IPL OS
```

Figure 5. The ALLPURP EXEC Procedure

under CMS, it can be scanned at the terminal or printed offline. Its parameters are the names of those CMS files which are the components of the OS/360 job stream, for example:

```
ALLPURP COBCOMP LKEDGO
```

An EXEC procedure such as ALLPURP, like an OS/360 cataloged procedure, provides a concise statement for invoking multiple system facilities. Figure 4 then is an example of terminal-entered statements which, when executed, will create and then begin execution of an OS/360 job stream. Figure 5 is an example of a simple EXEC procedure which when invoked performs a similar function, and additionally accounts for printer output.

The most flexible implementation of the flip-flop technique is an EXEC procedure which fully utilizes the power of the CMS EXEC processor. A procedure can be created which interacts with the terminal user, which accepts any filename-filetype combination, and which optionally transfers printer output. For example, the following statements could follow the ALLPURP statement

```
CP XFER 00E TO *
```

and thus allow the terminal user to dynamically decide the destination of his printer output:

```
&PRINT OUTPUT AT M67 PRINTER? (YES OR NO)
&READ ARGS
&IF &$ EQ YES CP XFER 00E OFF
```

No general purpose EXEC procedure for flip-flop is distributed with the CP-67 system. Each installation is encouraged to tailor one to its own characteristics and needs. Reference should be made to the CP-67/CMS User's Guide for a complete description of the EXEC processor.

If it is desirable to continually reread a job stream during a debugging run, the CP-67 command SET CARDSAVE ON may be used. This command will prevent any CLOSE command to the virtual card reader from deleting the current reader file. The READ issued after the CLOSE will begin its read with the card that is at the beginning of the current file; in this way, one may "rewind" the card reader. This file may be normally CLOSEd after a SET CARDSAVE OFF command has been issued.

If virtual printer output is transferred to the virtual card reader for later use in CMS, the files should generally be named with a filetype of PRINTER. This filetype implies 132-byte data records, without a leading carriage control character; CP spooling generates this type of record when OS writes to the virtual printer.

MINIDISKS FOR A VIRTUAL MACHINE

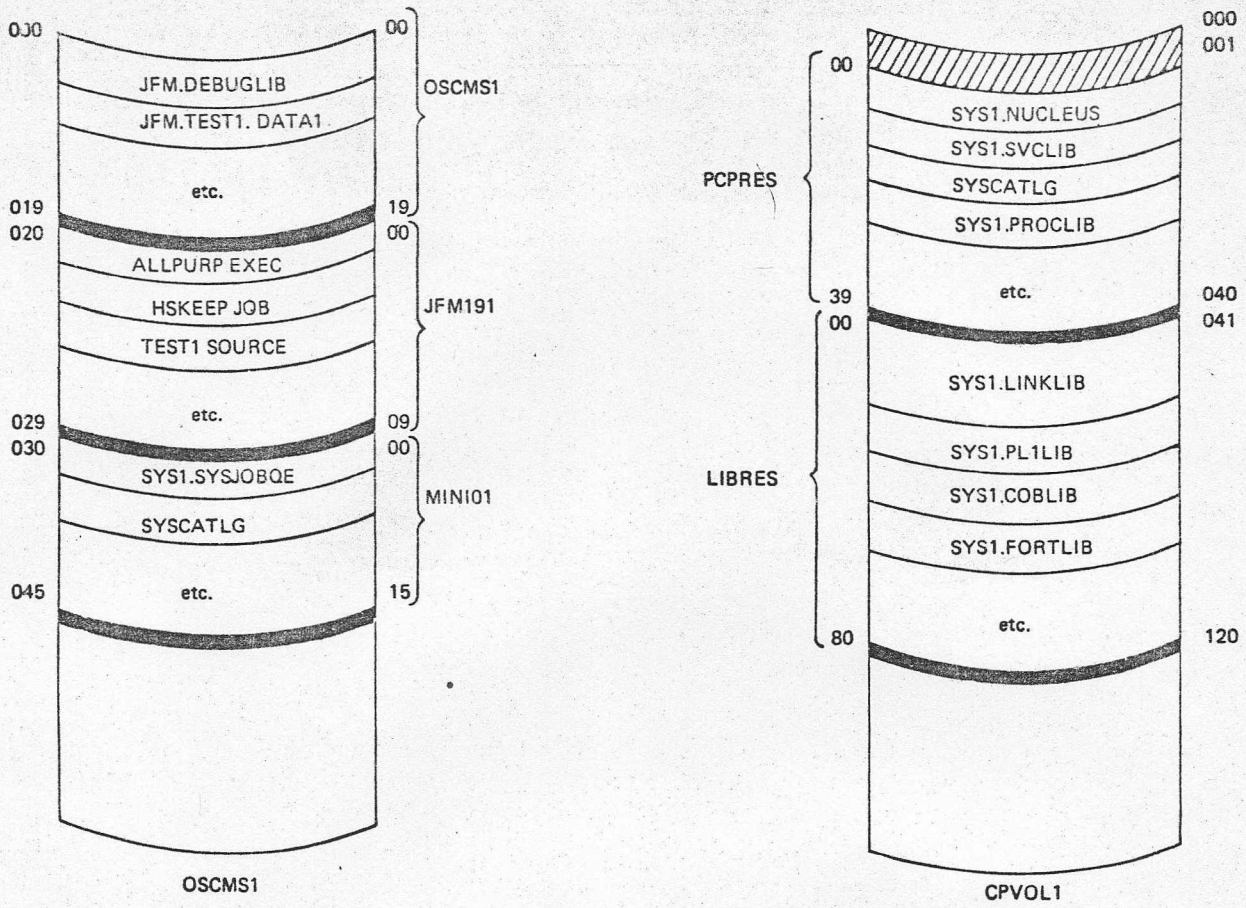
INTRODUCTION

CP-67 provides an environment in which multiple systems may be run concurrently. OS/360 may be running on some virtual machines, CMS on others, APL under DOS on yet another, etc. The external storage requirements would be highly uneconomical if direct access storage space for each of these systems were to be assigned in a conventional manner.

In CMS operations, each user is allocated the amount of disk space he is apt to require, and that space is considered to be a complete virtual device. Thus, as shown in Figure 6, a ten-cylinder area on real volume OSCMS1 is considered to be device 191 by the user; the area behaves like a full DASD volume when accessed by CMS and is referred to as a minidisk. The boundaries of the minidisk are controlled by CP-67. Should an attempt be made to reference a location outside these boundaries, CP-67 will present a command-reject I/O error to the virtual machine.

Users of OS/360 or DOS may also be assigned minidisks as virtual devices. OS/360 bases all of its space allocation parameters on the VTOC written on each disk; it determines the amount of space available on the volume from the Format 5 (Space Accounting) DSCB. Thus, what must be done in order for OS/360 to support minidisks is (1) to write a VTOC whose space accounting DSCB is equal to the desired size of the minidisk, and (2) to format the disk properly for OS use. The CP-67 utility program MINIDASD performs these functions; it is described below. If minidisks are to be used, OS/360 must be modified as described in Appendix A; it is the installation's responsibility to insure that the locations specified are accurate for the particular version and release of the Operating System.

Figure 6 illustrates the use and definition of minidisks. On the disk marked OSCMS1 are several minidisks, some formatted to OS/360 requirements and others to CMS requirements. The directory entries for the userid JFM describe the virtual device 191 as a ten-cylinder area on real volume OSCMS1, and the virtual device 233 as a twenty-cylinder area on real volume OSCMS1. CMS will normally access virtual device 191 as a P-disk; it must be assumed that the address 233 has been SYSGEN-ed into the OS/360 system available to userid JFM. The real disk marked CPVOL1 illustrates that minidisks may be used for system residence volumes as well as user-data volumes. In the illustration provided by Figure 6, userid JFM is given access to these minidisks on CPVOL1 on a read-only basis; in this way, multiple users can have access to the systems



DIRECTORY ENTRY FOR USER JFM:

```
JFM     USER    13915bbb,ACCNTING,D50,X'40'
        CORE    256K
        UNIT    009,1052
        UNIT    00E,1403
        UNIT    00D,2540P
        UNIT    00C,2540R
        UNIT    0FF,TIMR
        UNIT    190,2314,CMS190,000,202,RDONLY
        UNIT    191,2314,OSCMS1,020,029
        UNIT    230,2314,CPVOL1,001,040,RDONLY
        UNIT    231,2314,CPVOL1,041,120,RDONLY
        UNIT    232,2314,OSCMS1,030,045
        UNIT    233,2314,OSCMS1,000,019
```

Figure 6. Minidisks for OS/360 and CMS

residence volumes concurrently.
Each user's minidisk(s) should be backed-up on a scheduled basis. The CP-67 utility program CPDMPRST performs this function for all minidisks whether used by CMS, OS, or DOS. This program is interactive and runs under CMS; it can be used by an operations department to back up multiple minidisks on a scheduled basis, or by a user to back up a single minidisk whenever necessary. CPDMPRST is described in the CP-67 Operator's Guide since it is assumed that the installation's operations department will be most frequently responsible for minidisk maintenance.

LABELS

A physical disk volume may be processed under CP-67 as an entity, or as a combination of logical disks. These logical disks, or minidisks, may be distributed in any fashion across the physical disk.

The record at real cylinder 0, track 0, record 3 always has significance to CP-67. This record identifies the physical volume to CP; the following label formats are recognizable:

```
CMS=vvvvvv
VOLxvvvvvv
vvvvvv
```

The first label type is created by the CMS FORMAT command; vvvvvv represents the six-character volume label. The second label type is the one created by OS/360, DOS, etc., after running a standard DASD initialization program; it is the type created by MINIDASD. The third label type is created by the CP-67 FORMAT utility program.

A physical volume which holds only user minidisks may have the first of those minidisks starting at real cylinder 0. CP will be able to recognize the physical volume if this first minidisk has one of the label formats described above. In Figure 6, the volume indicated as OSCMS1 has its real cylinder 0 allocated to a minidisk which is formatted for use by OS/360. The volume serial number of that minidisk must be OSCMS1, the label that is associated with the real volume. Since the minidisk label identifies the physical volume, changing it affects the directory entries of all users who have minidisks assigned on that volume.

Since a user is free to execute MINIDASD or CMS FORMAT and rewrite the label of his minidisk, real cylinder 0 should never be assigned as a user data area.

TRACK CHARACTERISTICS

A minidisk is created by a suitable CP-67 directory entry. A minidisk is initialized for subsequent data recording by executing one of the following in a virtual machine: the CMS FORMAT command; the CP-67 FORMAT utility program; the CP-67 MINIDASD utility program.

The CMS FORMAT command formats all tracks in the specified disk to 829-byte blocks. For 2311 and 2314 devices, the CP-67 FORMAT utility program also formats each track to 829-byte blocks; when this FORMAT utility is run against a drum storage device, the tracks are formatted to 4096-byte blocks. The MINIDASD program, which executes under CMS, operates like the stand-alone DASDI program of OS/360; it writes R0 records as track descriptor records for each track, and clears the remaining portion of each track. Additionally, MINIDASD assigns alternate tracks to replace any defective tracks encountered on the disk.

The Home Address (HA) records that are written onto the real disk always specify real addresses. CP-67 translates the home address record contents to actual whenever the virtual machine issues a Read Home Address, Write Home Address, or Search Home Address channel command. The virtual machine always references home addresses which are relative to the beginning of the minidisk; the home address translation done by CP-67 is transparent.

CP-67 and CMS do not support the alternate track technique. The following alternatives exist if a track should become defective:

1. Run the CP-67 utility program DIRECT and allocate the cylinder in which the defective track exists as PERM. This will prevent CP from ever allocating that cylinder for paging, spooling, or temporary file space.

2. Set up a minidisk which includes the defective track, and format it with MINIDASD. An alternate track will be assigned. Thereafter use the minidisk only with systems which can recognize these alternate track assignments, i.e. OS/360 only.

3. Set up the full pack for OS/360 or DOS file residence. If the pack is for DOS, the DOS disk initialization program must be used since the OS/360 alternate track assignment is misinterpreted by DOS as an end-of-file flag.

INITIALIZATION WITH MINIDASD

MINIDASD is a CP-67 utility program. It formats real or virtual 2311 or 2314 disk volumes. It initializes disk surfaces as the OS/360 DASDI program does, except that: 1) it executes under CMS, and 2) it has the ability to format minidisks. It writes a Volume Table of Contents indicating the total number of cylinders available. It formats the volume as an IPL volume if so directed in the input control records.

Data to the program is supplied from a specified CMS file and from the terminal. The CMS file holds the control records for the program. These control records are exactly like those used by the OS/360 DASDI program; refer to Operating System 360: Utilities for a description of the parameters permissible for initialization of a direct access storage device. The example in Figure 7 below shows the contents of the CMS file that might be created to initialize the minidisk LIBRES (shown in Figure 6) for OS/360 data set residence. Note that the label of the real pack, CPVOL1, cannot be used as the VOLID parameter.

```
INITLR   JOB  INITIALIZE LIBRES AS A 80-CYL VOLUME
     MSG  TODEV=1052,TOADDR=009
     DADEF     TODEV=2314,TOADDR=231,              C
          VOLID=SCRATCH
     VLD  NEWVOLID=LIBRES,OWNERID=OPERATIONS
     VTOCD     STRTADR=10,EXTENT=5
     END
```

Figure 7. Sample MINIDASD Control File

The size of the minidisk is specified through the terminal. The user types an appropriate three-digit value in response to the message ENTER 3 DIGIT NUMBER OF CYLINDERS TO BE INITIALIZED. The value specified includes the space reserved for alternate track assignment; a user assigned to n cylinders will have n-1 of these initialized for his use and the n-th used for any alternate track assignment. If a complete volume is to be formatted this parameter value should be 203.

The minimum size of a minidisk can be computed from the formula shown in Figure 8. In it N represents the minimum number of cylinders, and K has value 10 or 20 depending upon whether the virtual device type is a 2311 or 2314 respectively. The SIZE-OF-VTOC should be in tracks, and the result of the indicated division should be rounded to the next highest integer.

```
N = 2 + (SIZE-OF-VTOC / K)
```

Figure 8. Formula for Computing Minimum Sized Minidisk

The program executes in CMS. To initiate execution, issue the command MINIDASD. The filename and filetype of the CMS file which contains the input control records for the MINIDASD program must be specified as arguments; for example:

```
MINIDASD INITLR CONTROL
```

During execution, MINIDASD first interrogates the user as to the size of the minidisk to be created and verifies the value entered; it then outputs the input control records. If alternate tracks are assigned, it outputs the addresses of the bad track and of the alternate track. MINIDASD assumes that the virtual device address of the terminal is 009.

OS/360 AS A SAVED,READ-ONLY SYSTEM

INTRODUCTION

This section describes a technique for creating a read-only, saved version of OS/360 for a CP-67 system. An installation which allows multiple virtual machines to run OS/360 would realize DASD space benefits by taking advantage of the CP-67 facility of concurrently sharing, among multiple virtual machines, disks which have been set up as read-only. An installation which requires frequent "flip-flop" between operating systems on a virtual machine would realize time benefits by making those operating systems "saved" (or "named") systems to CP-67. Both situations exist for installations using the Online COBOL Symbolic Debug Program RPQ.

Three job streams are described here. Each is distributed as a CMS file on the CP-67 basic program tape. The job streams to be described are: SAVEOS JOB, CPIPL SYSIN, and SAVEOS SYSIN. The first of these is an OS job stream which will link edit CPIPL and SAVEOS into the OS nucleus. The two SYSIN files are the source code for the programs that save OS/360 as a named operating system for CP-67. CPIPL may require modification and subsequent reassembly as discussed under "Operational Procedures" below. It should not be necessary to make any change to the SAVEOS program; this program functions much like the SAVESYS program which is also distributed with the CP-67 system, but (1) can be loaded from disk to any location of virtual memory, and (2) can save any number of pages associated with the OS system. The SAVEOS program assumes 009 as the virtual console address.

The technique presented here for making OS/360 a read-only system does not allow writing into the SYS1.LOGREC data set. (SYS1.LOGREC must exist on the OS SYSRES volume, which is here being created for CP-67 as read-only.) As of Release 21.0, if OS/360 attempts to write into SYS1.LOGREC and CP-67 considers that data area to be read-only, OS will receive a command-reject error, inform the user at the terminal, and continue processing. If the installation wishes to have errors recorded by OS while using the read-only copy of OS under CP, it will be required to modify OS NIP routines to allow for SYS1.LOGREC to exist on other than the OS SYSRES volume (that is, in a user's read/write area). Refer to Operating System 360: Nucleus Initialization Program PLM for a discussion of the OS routines which "open" LOGREC. It should be remembered that real device errors are always recorded by CP-67, although no distinction is made between temporary and permanent errors.

When a saved system is IPL-ed, certain system data sets must be at the same (virtual) unit and TTR addresses as when the system was originally saved. This is because OS, in the NIP processing done prior to saving, has located certain of its data sets and recorded their locations into core. SYS1.SYSJOBQE is one of these; it is a read/write data set, and must be set up on a predefined (virtual) volume and unit for each user.

The installation should define a standard volume name, such as MINI01, and an associated virtual device address to be given to each OS user. Each user's standard volume must have certain OS data sets located at the same TTR address; this is always true for SYS1.SYSJOBQE and may be true for other data sets of the OS system such as SYS1.LOGREC and SYS1.ROLLOUT. An initial minidisk should be formatted for the installation, and data sets allocated on it; that minidisk should be dumped using CPDMPRST. Thereafter each user's standard minidisk is initialized by first running MINIDASD to format it as an OS minidisk, and then running CPDMPRST to restore to it SYS1.SYSJOBQE and any other required system data sets. The above procedure requires that each user's standard minidisk is the same size.

If the installation updates the SYS1.LINKLIB or the SYS1.SVCLIB used by the saved system, the system must be resaved since the in-core TTR addresses for these data sets and their members will no longer be correct.

CREATION PROCEDURES

The CPIPL program described here becomes part of the OS nucleus, functioning at two distinct times: during the system save, and during the later IPL of that system. The function during save is described in step 10 below. During IPL of the saved system, CPIPL will read the date from the real-time clock of CP-67 (assumed to be device 0FF), reformat it, and store it into the CVT of the saved OS. This is only of use if the saved system is a PCP system which was saved after the START command. To have a valid date for an MVT or MFT system, save the system after an invalid SET DATE command; the saved OS will then request reentry of the SET DATE command as soon as it is given control by CPIPL.

The CPIPL program described here assumes, at ipl of the saved system, that all volumes mounted at the time of save are available and ready. Thus, during save, only the volume MINI01 located at its standard virtual device address should be online.

1. If the sysres volume(s) is to be read-only, or if minidisks are to be used, the modifications described

in Appendix A must be made to OS. Read the descriptions to determine the function of each modification and verify, from listings or dumps, that the locations indicated are accurate for the version and configuration of OS that is to be used. If the locations differ, modify the operands of the VERIFY and REP cards.

If the sysres volume is to be read-only, run appropriate OS/360 utilities to move the read-only data sets to the sysres volume. Figure 9 provides an illustration of a shared OS/360 system in the environment of CP-67. The sysres volume may itself be a minidisk, and of course more than two virtual machines may have read access to it. In Figure 9, the virtual machine marked B indicates a core size larger than that of the virtual machine marked A. This is practical only if the OS is not ipl-ed as a saved system for CP for, if it is, the additional core available in system B will be unknown and consequently unused.

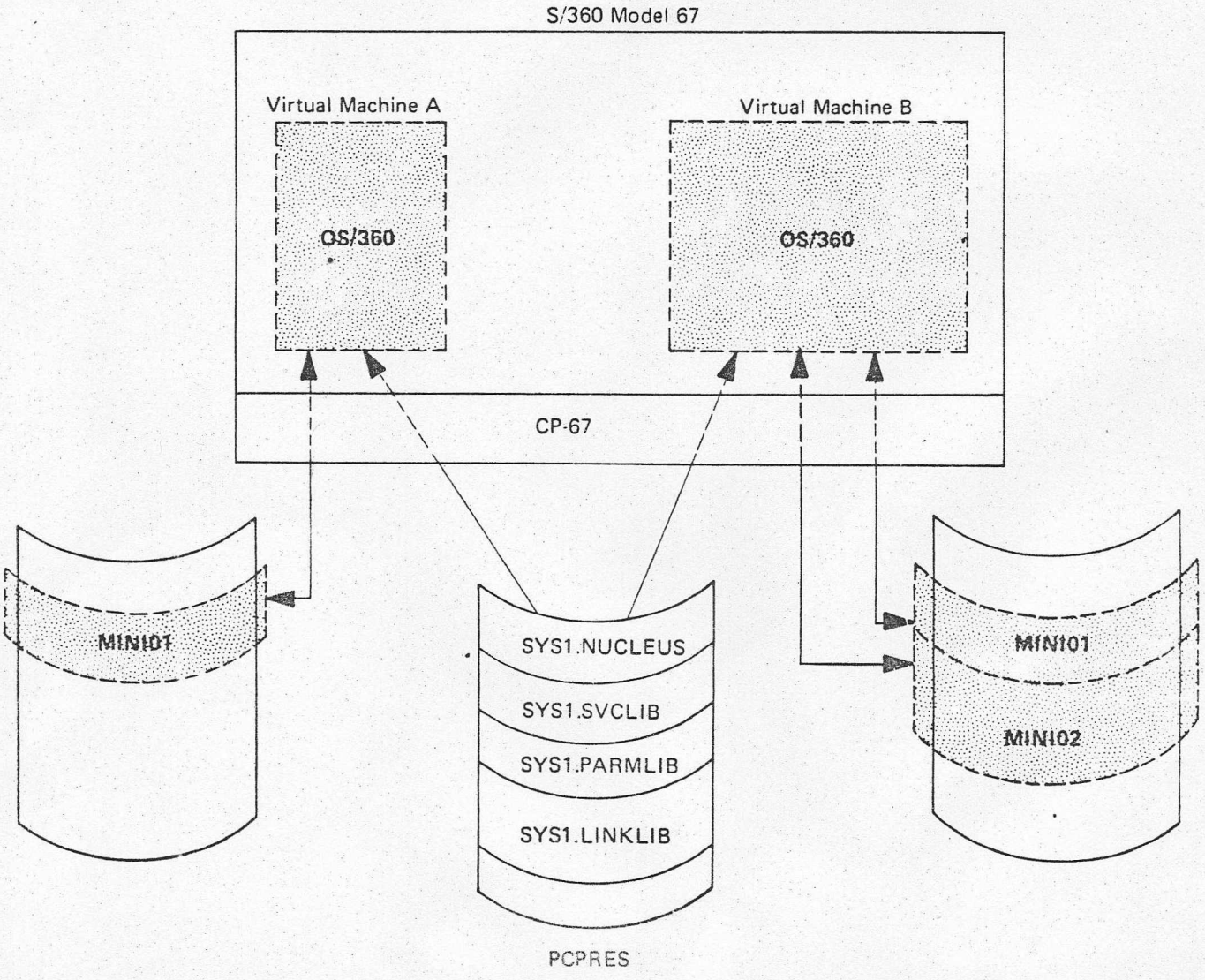


Figure 9. Shared OS/360 Sysres Volume

2. IPL the system that is to be saved to the point that it is to be saved. An MFT or MVT system must be saved immediately after typing in the SET DATE command; to do otherwise would require rewriting CPIPL such that the OS job queue tables would be properly updated during the saved-system IPL. A PCP system can be saved after the SET DATE command, a DISPLAY JOBNAMES command, or the START command; IPL of the saved system will be fastest and easiest if it is saved after the START command.

   Display core locations X'88' through X'94'. The CPIPL program is written to use these locations as the entry point at IPL of the saved system. If these locations are all zero, and if the card reader address is 00C, proceed to step 4 below. Otherwise, modify and assemble the CPIPL program according to the instructions in step 3.

3. Locate four unused words in the diagnostic scan-out area for the system being saved (the diagnostic scan-out area is in core locations X'80' though X'17F'); adjust the ORG statement in CPIPL to point to the first of these. Adjust the operand of the EQU statement in CPIPL for the symbol READER to the correct device address for the card reader. In CMS, assemble CPIPL and replace the first of the two text decks in the SAVEOS JOB file with the new CPIPL TEXT deck.

4. The SAVEOS JOB file should be used as an OS job stream to link edit the CPIPL and SAVESYS programs into the nucleus of the OS system to be saved. Before doing so, check that the two insert cards in the SAVEOS JOB file are identical to the two insert cards in the listing generated by the original link edit of IEANUC01 in Stage II of SYSGEN; if they are not, change them. The distributed job stream specifies replacement of IEANUC01 in the SYS1.NUCLEUS data set; the installation must provide its own techniques for error recovery should this link edit job fail.

5. Again, IPL the system that is to be saved. Dump the first five or six pages of core and find the characters "CPIPL ADDR=". The fullword immediately following the equal sign will contain the address of the CPIPL program. This address will be used during the save of the system as described in steps 9-11 below.

6. ATTACH the direct access device on which the system is to be saved to the virtual machine being used. (Note: If the disk on which the system will be saved is the CP-67 system residence volume, the CP-67 LINK command will have to be used to access the device or an entry must be placed in the CP-67 Directory allowing read/write access to the device.) The cylinders on which the saved system is to be stored must be

formatted for CP file residence by the CP-67 utility program FORMAT. Follow the procedures in the CP-67 Operator's Guide, Order No. GH20-0856, for the purpose of formatting an area which will hold a pageable copy of OS, the full volume must be ATTACHed, and the cylinder addresses specified must be real (that is, it is not possible to use a minidisk). Do not attempt to use a 2301 or 2303 as the FORMAT program initializes those devices for dynamic paging only, and makes them unsuitable for saved system storage. Note that FORMAT will write a CP-67 volume label at cylinder 0,record 3 of the receiving volume.

If the saved OS is to be stored on a CP system disk (one that is available to CP for paging, spooling, or CMS Temporary file space), then the CP utility program DIRECT must also be run. In the allocation control cards which are input to DIRECT, specify that the cylinders to be used for the saved OS are permanent by supplying an appropriate PERM control card; refer to the CP-67 Operator's Guide, Order No. GH20-0856, under the DIRECT utility and the section "Directory Allocation and Creation". Note that DIRECT will allocate cylinder 0 of the receiving volume for its allocation tables.

7. The system to be saved must be properly defined to and named for CP-67. The SYSTEM and SWPTABLE macros are both required to set up a named system for CP-67; they are part of the SYSTEM module of CP-67 and must have been included when assembling that module. If they were not, reassemble the module with them included and reload the system. Refer to the manual CP-67/CMS Installation Guide, Order No. GH20-0857, the section "Generating Named Operating Systems Under CP-67" for discussions of the parameters and formats of the SYSTEM and SWPTABLE macros.

   If step 3 above was bypassed, the value for the EXADD parameter in the SYSTEM macro will be X'88'. If step 3 was performed the EXADD parameter value should be identical to the modified ORG statement operand. The EXADD operand address will be used by CP when IPL-ing this saved system.

   The name chosen for the saved system cannot be a string which represents a valid hexadecimal number.

   Models of the SYSTEM and SWPTABLE macros are given below. The named system "OS" will be saved starting at real cylinder address 86 on the 2314 volume CPVOL1 and 64 pages will be saved. When IPL-ed, execution will begin at X'88', the system mask will be X'00', and the protect key will be X'00'. These last two parameters are requirements of the CPIPL program, and must be assigned these values.

```
SYSTEM  NAME=OS,FIRSTP=00,LASTP=X'3F',
          TABLE=OSTBL,VOLID=CPVOL1,
          EXADD=88,SYSMASK=00,RUNKEY=00
OSTBL SWPTABLE DASDORG=086,0,1,FIRSTP=00,
          LASTP=X'3F',DISK=2314
```

8. Refer to the manual CP-67/CMS Installation Guide, Order No. GH20-0857, the section "Generating Named Operating Systems Under CP-67", and to the CP-67 Operator's Guide, Order No. GH20-0856, under the

SAVESYS utility program, for discussions of the SAVE control card; create a SAVE control card as a CMS file and name it SAVEOS CARD. Be sure that the parameters on the SAVE card correspond with those already defined in the SYSTEM and SWPTABLE macros. (The SAVEOS program uses the same SAVE control card as the SAVESYS program.)

9. XFER the SAVE control card to the virtual card reader so that it can be read by SAVEOS, and close the virtual card reader. ATTACH the direct access device on which the system is to be saved to the virtual machine being used; the device address must be the same as was specified in the SAVE control card. IPL the OS system to be saved to the point of issuing the last operator command to be part of the saved system. (See step 2 above.) Type this command (usually START or SET DATE), follow it by several blanks, and then hit the attention key.
The results of this entry are that the virtual machine console read is completed with all but the last character typed, pending channel-end and device-end interrupts are set, and CP command mode is entered.

10. Type the following sequence of CP commands:

```
DISPLAY    78-7C
STORE   78  20000
BEGIN
```

The first of these displays the I/O new PSW. After the BEGIN command is entered, OS is given control and processes the channel end interrupt which had been held pending.
When the message

```
CP ENTERED, REQUEST PLEASE
```

appears, store into the PSW the I/O new PSW displayed earlier. Use the CP console function STORE PSW. Again type BEGIN to return control to the OS system and permit processing of the device end interrupt which had been held pending.

11. When the message

```
CP ENTERED, REQUEST PLEASE
```

appears, store into the PSW a masked supervisor-state PSW whose instruction address is the start of the CPIPL program, as determined in step 5 above. The format is

```
STORE PSW 00040000 0000xxxx
```

Then type the following:

```
STORE    78 40000
BEGIN
```

The CPIPL program will gain control and, together with the SAVEOS program, will proceed to save the Operating System for CP. When the save is complete, the following message will be typed:

```
NORMAL TERMINATION OF SAVE FUNCTION
```

12. Ipl the saved system. If the saved system goes into an unexpected wait state, it may be because it was saved at a point when a device end interrupt was still pending. Try resaving the system at a slightly different point, e.g. after the DISPLAY JOBNAMES rather than the START.

OPERATIONAL PROCEDURES

The device on which a shared or read-only systems residence volume exists must be ATTACHed to the CP-67 system in order to be made available to all potential users. Those users with appropriate directory entries will be given read-only access to the volume. Write access to the volume will be given if the directory entry so indicates, or if the real device containing the shared system volume is ATTACHed to the user's virtual machine. For the latter, the real device must first be DETACHed from the system.

A saved system which is not read-only has no special operational requirements.

CATALOGING CONSIDERATIONS

Special considerations must be made if data set cataloging will be used when operating from a read-only sysres. The terminal user can catalog his data sets only on a volume to which he has read-write access. A SYSCATLG data set should therefore be allocated on the user's installation standard OS/360 minidisk (e.g., his MINI01), or on some other selected volume assigned to his userid.

To catalog the user should invoke the OS/360 utility program IEHPROGM. The CVOL option in the CATLG statement must be specified. Of course, CATLG macros with suitable options could be coded into an OS/360 program if the user were so inclined.

Use of the catalog for users of a read-only system can be made easier if catalog indexes are established. In this way, JCL disposition parameters can be used to specify cataloging. To allow specification through JCL each user should be assigned a unique primary index name. The BLDX and CONNECT statements of IEHPROGM would then be used to establish pointers to each users SYSCATLG data set. This step requires read-write access to the system volume; if the system programmer's directory entry does not indicate this, he should ATTACH the physical device containing the sysres volume to his virtual machine.

PAGING CONSIDERATIONS

SYSGEN OPTIONS for OS/360

Frequently there is a performance gain when CP-67 initiated paging is substituted for Operating System I/O operations. All virtual machine CCW sequences must be translated by CP-67 to reference real memory and real devices. To bring a transient routine into real core for execution, CP-67 must 1) intercept the interrupt caused by the execution of the SIO instruction in OS, 2) swap into real core, and lock for the duration of the I/O operation, the page into which the transient routine will be fetched, 3) translate the device address specified in the SIO to the real device address, 4) translate the memory address referenced in the seek, search, and read CCWs, 5) generate a chain of CCWs, adding additional data chaining CCWs, if the real memory locations require crossing a page boundary, 6) issue a SIO, 7) present to OS the SIO condition code, and 8) intercept, untranslate, and present to OS the channel end and device end interrupts which OS must then process. If paging can be substituted for this process, the only work to be done is to swap into real core the page which contains the desired routine.

The performance of OS in a virtual machine can be improved by specifying as resident as many of the normally transient routines as is reasonable. In this way paging I/O will be often substituted for OS initiated I/O. Include SVC modules, access methods, and the build list (BLDL Table); refer to Operating System 360: System Generation for detailed information about the residency options that exist for the configuration of OS that is to be run. Options concerning SYS1.SYSJOBQE should be chosen with the characteristics of virtual memory in mind; e.g., for a PCP system, a resident job queue should be defined.

There is generally no advantage to the PCI-FETCH option for load module retrieval when the OS system is to be run in a virtual machine. On a real machine the PCI-FETCH option is used to overlap external reads from DASD storage with the internal processing necessary to load and relocate the load module in memory; under CP-67 such timing relations do not exist for a virtual machine, and there is no advantage in providing code that assumes it might.

SMART NIP should be generated into any version of OS/360 that is to run in a virtual machine. In this way, different virtual machine configurations can easily use the system, and as few 2314 minidisks as are required by the virtual machine user need be defined in his directory. The

alternative to SMART NIP is a number of VARY OFFLINE commands issued by the user.

SAVED SYSTEM CHARACTERISTICS

Often an installation which has set up a copy of OS/360 for CP-67 will have multiple virtual machines concurrently using that system. When this occurs the following situation will likely exist: the saved system volume will experience heavy I/O activity, and a high quantity of paging will be done from there. The reason for this is that CP will assign a page to a special paging area only if some storage location in that page has changed. Since much of OS is coded reentrant, the effect of this algorithm is that much of the paging will be done from the original disk locations.

If this characteristic of a highly reentrant saved system impacts performance, the installation may do one of several things. Certain pages of OS, wholly containing reentrant code, may be locked in real memory and shared by all OS users; refer to the section below, "Shared Pages for OS/360 Code".

The installation may decide to have multiple copies of the saved OS, each with a different name. Then, by defining a subset of users for each saved system created, the installation would reduce the paging activity against any one copy of the saved system.

A third approach involves modification to the program CPIPL. That program is the first to gain control from CP-67 after ipl of the saved system. It can be made to force paging allocation to a high-speed paging device for any given page of this virtual machine by reading out and then re-storing data in that page. CPIPL executes with a protect key of zero, and so this fetch-and-restore technique is possible for any part of the OS system. It would be advantageous to do this for heavily used parts of the nucleus which are reentrant, or for heavily used access methods. However, forcing too many pages to the high speed paging devices defeats the goal of the CP paging algorithm, which is to restrict use of those devices to active pages.

SHARED PAGES for OS/360 CODE

CP-67 provides two system facilities which can reduce the paging requirements of a virtual machine. The first is the CP operator command LOCK which permanently assigns user pages to real memory locations. The second facility is that

of shared pages, whereby multiple users of a saved system share certain parts of real core on a read-only basis.

The shared page facility is advantageous to those installations which run multiple OS/360-controlled virtual machines where each is run single thread and all have common job characteristics.

When pages are shared, CP-67 assigns to them a storage protect key of zero. A virtual machine operating system cannot alter a shared page. All other pages available to the virtual machine are assigned a key of F; if the operating system issues a Set Storage Key privileged instruction, it is NOP-ed. Since normal OS/360 storage protection features are overriden in the CP-67 shared page algorithm, it is generally not advisable to then allow multiple users to execute in a single virtual machine.

The pages to be shared by multiple virtual machines running OS must have contiguous addressing and must be read-only. Moreover, the operating system must be a saved system to CP-67. If a virtual machine attempts to store into a shared page, it will receive a storage protection interrupt. What the effect will be to the virtual machine depends upon the system that was executing in it; however, only that virtual machine will be affected.

To implement shared pages for OS/360, the installation must add space for a shared page table in the CP-67 module CPCORE; it must name that table in the SYSTAB parameter of a SYSTEM macro; it must name the system and its shared page table in the CFSIPL module; it must specify which pages are to be shared in a SWPTABLE macro; it must reassemble the CFSIPL, CPCORE, and SYSTEM modules and reload the CP-67 system.

1. To include space for the new shared page table in CPCORE, locate the entry for CMSTAB. If the saved copy of OS/360 is to share a single page among all users of the system, then code according to the following example and insert it after the entry for CMS. If two pages were to be shared, the "2" indicating the number of halfwords reserved would be replaced by a "3", etc.

```
****SPACE FOR SHARED PAGE TABLE FOR OS/360****
         ENTRY OSTABLE
         MVC 0,8(15)
OSTABLE  DS  2H'0'
```

2. The parameter of the SYSTEM and SWPTABLE macros which correspond to this example, assuming that page 10 is to saved, are:

```
 SYSTEM  TABLE=OSTBL,SYSTAB=OSTABLE,...
OSTBL  SWPTABLE  FIRSTSP=X'A',LASTSP=X'A',...
```

3. To correctly modify the CFSIPL module of CP-67 to include information about the shared page system, look for the DC called SHARSYS. Immediately following this DC and its partner, insert one DC which specifies the saved system name and below it a second DC which specifies the address of that system's shared page table in PAGETRANS. Assume that the saved copy of OS/360 referred to above is named "OS"; the following lines show how information about this system fits with the information already established for CMS.

```
SHARESYS  DC   CL8'CMS'
          DC   V(CMSTABLE)
          DC   CL8'OS'
          DC   V(OSTABLE)
SHAREND   EQU  *
```

It is the installation's responsibility to identify those pages of an OS/360 saved for CP-67 which have sufficient activity to warrant locking them into real memory locations. Defining pages as shared in the SWPTABLE macro implies that, at the first IPL of the system, the shared pages will be automatically LOCKed into real memory by CP, and all subsequent users of the saved system will be pointed to those pages.

LOCKED PAGES FOR OS/360

Frequently, an installation will require an OS/360 system to run on a virtual machine in a multi-tasking mode, perhaps as a multi-access system. To improve system performance for this type of virtual machine operation, the installation should utilize the LOCK operator's command of CP-67. This command LOCKs into real core the specified pages of the virtual machine; by so doing, it eliminates all paging I/O for these pages.

Only frequently used or frequently referenced pages should be LOCKed into real memory. Page 0 is referenced whenever a virtual machine interrupt occurs, or when a CSW is stored; consequently, it should be the first page the installation considers LOCKing. The location of the first and second level interrupt handlers of OS should be determined, and that page(s) considered for LOCKing.

Other pages to be LOCKed depend upon the work being done by the particular Operating System: whether the jobs are of long or short duration, whether the application involves teleprocessing, whether data set indices are core resident, etc. It is important to realize that there is as much advantage to be gained in LOCKing into real core a heavily referenced data page as an often executed instruction page.

control blocks or one primarily filled with user data set indices.

The LOCK command must be issued by the system operator each time the userid associated with this particular OS virtual machine logs on. If multiple pages are to be LOCKed, it is recommended that the system operator be provided with a suitable CMS EXEC procedure.

APPENDIX A

The modifications described in this appendix have not been subjected to a formal IBM testing procedure. The installation should evaluate the usefulness of each modification in its own environment prior to implementation.

MODIFICATIONS TO OS/360

TO SUPPORT MINIDISKS

In the normal processing of a STOW request, a channel program consisting in part of a "Read Home Address" command chained to a "Read Data" command is executed. The "Read Home Address" seems to be intended as a replacement for a "Search ID" and "TIC" to position the disk to the first record on the seeked track. A problem arises here in the case of minidisks under CP-67. Since in this case the home address actually read from the disk may be different from the virtual track address, CP-67 attempts to "relocate" the track address in core after the channel program completes. The "Read Home Address" used by STOW reads into the same data area as is read into by the subsequent "Read Data" command which is chained to it. So at channel end, the data area addressed by both commands contains the data read, and not the track home address. At this time CP-67 attempts to relocate the data instead of the home address, with catastrophic results.

This problem must be fixed in all current (at least up through Release 19.6) versions of OS/360 if minidisks are to be used. The CCW skip bit to suppress data transfer, when set, causes the channel command to be properly executed by the control unit, but causes the channel to "throw away" any data associated with the operation. When this bit is set in a "Read Home Address" command, CP-67 does not attempt to relocate any home address in core, since none will actually result. The control unit does, however, perform the ordinary procedure, and the next "read" command reads the first record on the track as intended by STOW. The setting of this "skip bit" solves the CP-67 problem and has absolutely no other effect on the operation of OS/360 in a real or virtual machine. The job stream used to fix this problem on both OS/360 Version 19.6 PCP and MVT is as follows:

```
//STOWZAP   JOB   MSGLEVEL=1
//ONLYZAP   EXEC  PGM=SUPERZAP
//SYSLIB    DD    DSNAME=SYS1.SVCLIB,DISP=OLD
//SYSPRINT  DD    SYSOUT=A
//SYSIN     DD    *
```

```
DUMP      IGC0002A   IGC021
NAME      IGC0002A   IGC021
VERIFY    0374       1A00,006C,4000,0005
REP       0378       50
DUMP      IGC0002A   IGC021
/*
```

It is the installation's responsibility to locate the instruction in STOW processing which requires modification. To verify the locations modified at the source level, check the microfiche for module IGC0002A. Locate the following statements:

```
SCCW1   DC   X'1A'                  61520000
        DC   AL3(0+BUFF1-AREA)      61720000
        DC   X'40000'               61920000
```

The modification is to change the "4" in the last DC to a "5". This will set the skip bit on in the CCW.

Note: For Release 21.6 of OS/360 MVT, this code modification need not be applied as the STOW processor already has the 'SKIP' bit set.

TO SUPPORT A READ-ONLY SYSRES

In OS/360 MFT and MVT the reader procedures must contain a DD card describing the SYS1.PROCLIB data set. To allow for a SYS1.PROCLIB which does not reside on the system residence volume, it is customary to specify the DD parameters such that the SYS1.PROCLIB data set is located via the system catalog. During the execution of the Master Scheduler Initialization routine, SYS1.PROCLIB is dynamically CATALOGed on the volume mounted at the address specified in the "set proc=xxx" command. If this command is not given, the SYS1.PROCLIB data set is CATALOGed on the system residence volume.

The CATALOGing of SYS1.PROCLIB will eventually result in a write command to the SYSCTLG data set on the system residence volume. If this volume has been specified to CP-67 as a read-only volume (for the purpose of allowing the volume to be shared) the write command will cause a channel program check, and will result in an error return from CATALOG; the system will not initialize.

This problem will occur only with MFT and MVT systems with read-only system residence volumes. To remedy the situation, the function that CATALOGs SYS1.PROCLIB must be changed to a NOP. Of necessity this will nullify all effects of a "set proc=xxx" command. SYS1.PROCLIB may still reside on a volume other than the system residence, but it will have to be permanently cataloged by a user (usually the system programmer) with read/write access to the system residence volume. The job stream that was used to implement this fix in OS/360 Version 21.6 MVT is as follows:

```
//INITZAP   JOB    MSGLEVEL=1
//ONLYZAP   EXEC   PGM=SUPERZAP
//SYSLIB    DD     DSNAME=SYS1.LINKLIB,DISP=OLD
//SYSPRINT  DD     SYSOUT=A
//SYSIN     DD     *
DUMP     IEEVIPL    IEEVIPL
NAME     IEEVIPL    IEEVIPL
VERIFY   5D0        0A1A
REP      5D0        1BFF
DUMP     IEEVIPL    IEEVIPL
/*
```

The module name for MFT is IEFSD569 instead of IEEVIPL. The location specified for Release 15/16 of MFT is 05F4 instead of 03B0. It is the installation's responsibility to locate the instruction in NIP which requires modification.

To verify the locations to be modified at the source level, check the Stage II assembly listing for IEEVIPL (or IEFSD569). Locate the following statements:

```
IP515   EQU   *
        LA    1,IPCAT
        SVC   26     LINK TO SERVICE ROUT.
```

The modification is to change the SVC to a NOP. This will bypass the automatic cataloging of SYS1.PROCLIB.

INDEX