GA22-7074-0
File No. S370-01

Systems

Virtual-Machine Assist and Shadow-Table-Bypass Assist

- 26 bit addressing
- extended storage keys
- common segment
- 4K keyops

IBM

Preface

This publication summarizes the assists for Virtual-Machine Facility/370 (VM/370) and gives a detailed description of the virtual-machine assist and the shadow-table-bypass assist.

This publication is intended for system programmers. The reader should be familiar with the general machine functions of System/370, as described in *IBM System/370 Principles of Operation,* GA22-7000, and with the VM/370 system. The reader may also wish to refer to *IBM Virtual Machine Facility/370: System Logic and Problem Determination Guide,* SY20-0885, and to *IBM Virtual Machine Facility/370: Data Areas and Control Block Logic,* SY20-0884.

First Edition (May 1980)

Changes are periodically made to the information herein; before using this publication in connection with the operation of IBM equipment, refer to the *IBM System/360 Bibliography,* GC20-0360, *IBM System/370 and 4300 Processors Bibliography,* GC20-0001, for editions that are applicable and current.

Contents

Chapter 1. Assists for VM/370

This publication gives a detailed description of the virtual-machine assist and the shadow-table bypass assist.

Six assists are available on the various models of System/370 to improve the performance of Virtual Machine Facility/370 (VM/370):

- Virtual-machine assist
- Shadow-table-bypass assist
- Control-program assist
- Expanded virtual-machine assist
- Virtual-interval-timer assist
- Virtual-machine extended-facility assist

The extended control-program support for VM/370 (or ECPS:VM/370) consists of a combination of four assists: (1) the virtual-machine assist, (2) the control-program assist, (3) the expanded virtual-machine assist, and (4) the virtual-interval-timer assist.

The six assists for VM/370 (1) execute specific privileged instructions for virtual machines, (2) simulate virtual-machine operations (such as interruptions) or maintain the virtual-machine interval timer, and (3) provide new instructions for use by the VM/370 control program to perform frequently occurring functions.

The virtual-machine assist, the basic assist for VM/370, directly executes 12 virtual-machine instructions and validates page-table entries in the shadow tables. The virtual-machine assist exists in two forms: with and without the VM-common-segment modification. The VM-common-segment modification of the virtual-machine assist maintains high performance for virtual machines executing programs that use the common-segment-bit function of the System/370 extended facility.

The shadow-table-bypass assist is a specialized assist used only for virtual machines for which the virtual = real option of VM/370 is specified.

The control-program assist provides 22 new instructions for use by the VM/370 supervisor program to accelerate completion of the various functions of the control program itself.

The expanded virtual-machine assist, which has the virtual-machine assist and the control-program assist as prerequisites, completely or partially executes 11 virtual-machine instructions.

The virtual-interval-timer assist maintains the virtual-machine interval-timer value and causes virtual or real program interruptions when the interval timer is decremented through zero.

The virtual-machine extended-facility assist, an integral part of the System/370 extended facility, causes the direct execution of 12 instructions of that facility for a virtual machine.

Figure 1 summarizes the preceding information.

In general, a particular assist directly executes a virtual-machine instruction only for specific conditions (for example, DAT on, PER off, or no exceptional conditions). The same virtual instruction may be executed by one or another assist, depending on the specific conditions at execution. For example, STORE THEN OR SYSTEM MASK can be assisted by the virtual-machine assist, the shadow-table-bypass assist, or the expanded virtual-machine assist. For this reason, the column in Figure 1 giving the number of instructions assisted contains some duplication.

Assist	Prereq-uisites	Instruc-tions Assisted	Other Virtual Functions	New Instruc-tions
Virtual-machine assist (VMA)	–	12	1	–
Shadow-table-bypass assist	–	7	1	–
Control-program assist (CPA)	–	–	–	22
Expanded virtual-machine assist	VMA,CPA	11	–	–
Virtual-interval-timer assist	–	–	2	–
Virtual-machine extended-facility assist	–	12	–	–

Figure 1. VM/370 Assists

VM/370 Control and Storage

This section discusses certain aspects of VM/370 design which are related to the operation of the various assists for VM/370. It can be skipped by readers familiar with the VM/370 supervisor program.

VM/370 is a system-control program that uses the resources of a real System/370 machine to execute a number of independent programs, each appearing to run on its own System/370 machine. These apparent machines are called *virtual* machines in distinguishing them from the *real* machine, whose resources are actually used to give the appearance of concurrent execution of a number of independent virtual machines.

Virtual-machine programs are executed by a combination of (1) the simulation of all virtual-machine I/O and console operations and of certain machine instructions, and (2) the direct execution on a real CPU of the remaining virtual-machine instructions. Figure 2 is a simplified flowchart showing a CPU alternating between direct execution of instructions in storage containing virtual-machine programs and execution of simulation routines in storage which contains the VM/370 control program. The figure shows that VM/370 depends entirely on interruptions to recapture control of the real CPU for simulation of virtual-machine functions.

VM/370 divides the storage of the real machine into 4K-byte page frames. Some page frames are allocated to contain parts of the VM/370 control program; other page frames are allocated to hold pages of the storage of virtual machines. These latter page frames may be dynamically stolen and later reassigned to hold a different page of storage for the same or for a different virtual machine. In general, at any moment, only some of the pages of each virtual machine have real page frames assigned to them. The remaining pages have been swapped out to VM/370 auxiliary storage. VM/370 performs input and output paging operations between auxiliary storage and the real-machine storage in a manner not apparent to the virtual-machine program. The paging by VM/370 is in addition to any paging by the virtual-machine operating system, which may move pages between virtual-machine storage and the virtual-machine auxiliary-storage devices.

During direct execution of virtual-machine instructions, the condition code, instruction address, PSW key, and program mask of the virtual-machine PSW are in the real-machine PSW. The remaining real PSW bits, however, are controlled by VM/370. In particular, the problem-state bit of the real PSW is always one; hence, any attempt to execute a privileged instruction for the virtual machine always results in a program interruption for a privileged-operation exception if no assist is active.

In VM/370 terminology, an address that is *real* to a virtual machine is called a virtual address. VM/370 may have assigned any one of a number of real page frames to hold the page of virtual storage corresponding to a 4K-byte block of consecutive virtual addresses. For each virtual machine, VM/370 keeps a set of *real translation tables* for translating virtual addresses to real addresses. The real translation tables consist of one *real segment table* and a *real page table* for each 64K bytes of virtual-machine storage. These tables are formatted for use with the dynamic-address-translation (DAT) facility of System/370. When a virtual-machine program is being directly executed with the virtual machine in BC mode, or in EC mode with DAT off, the real PSW is in EC mode with DAT on, and real control registers 0 and 1 specify the real segment table of the virtual machine in execution.

When the virtual machine PSW is in EC mode with DAT on, control registers 0 and 1 of the virtual machine specify how to translate logical addresses to virtual addresses. Such a logical address, called a *virtual/virtual address* in VM/370 terminology, after translation to a virtual address must be translated a second time to obtain a real address. Because the System/370 DAT mechanism does not perform this double translation, VM/370 maintains another set of translation tables, called shadow tables, which can be used by the DAT facility to translate addresses directly to real addresses in one step. When a virtual-machine program is being directly executed in a virtual machine in EC mode with DAT on, the real PSW is also in EC mode with DAT on, but real control registers 0 and 1 specify the shadow segment table for the current virtual machine and current values of virtual control registers 0 and 1. If a new value is placed in virtual control register 1, VM/370 must set up real control register 1 to specify a corresponding new shadow table.

The storage keys for each 2K-byte storage block of a page frame assigned to a virtual machine are shared by the virtual machine and the VM/370 control program. The access-control bits and the fetch-protection bit are controlled solely by the virtual machine, but the reference and change bits

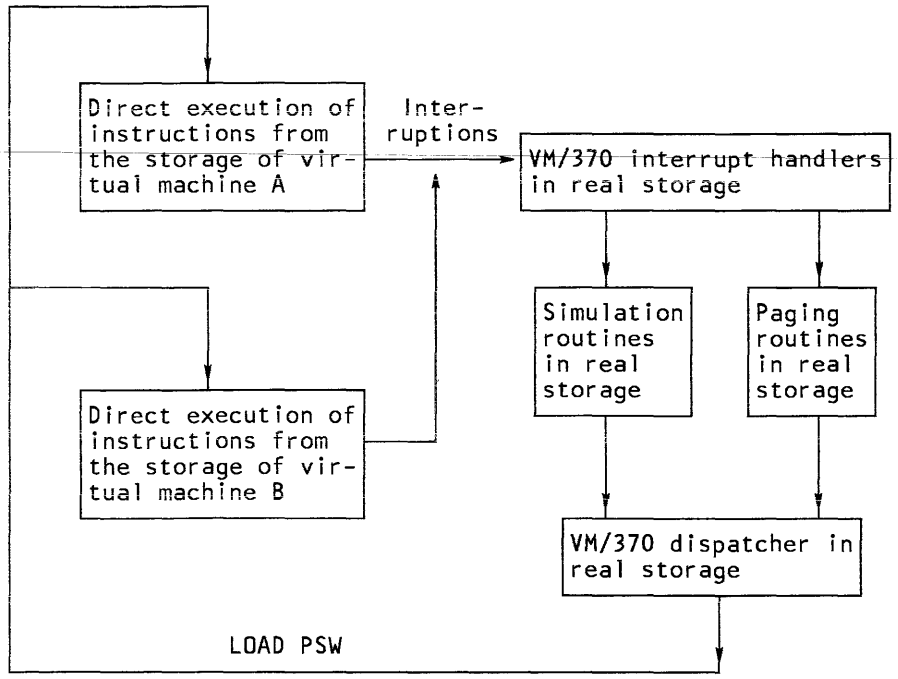
Direct execution of instructions from the storage of virtual machine A
Interruptions
VM/370 interrupt handlers in real storage
Simulation routines in real storage
Paging routines in real storage
Direct execution of instructions from the storage of virtual machine B
VM/370 dispatcher in real storage
LOAD PSW

Figure 2. Real CPU Control Alternating between Direct Execution and Simulation of Programs for Two Virtual Machines

are shared by both the virtual machine and the VM/370 control program. This sharing is performed in such a manner that, for the purposes of testing or setting the reference and change bits of storage keys by program control, there appear to be two independent sets of reference and change bits. This is accomplished by maintaining two sets of reference and change bits *in addition to* the real storage key. Both additional sets of reference and change bits are kept in the *swap table.*

There is one swap table for each real page table. The word that precedes the first entry of each real page table contains the address of the corresponding swap table. A swap-table entry is eight bytes long. Because VM/370 uses a real 4K-byte page size, each swap-table entry contains four sets of reference and change bits–two sets for the lower 2K-byte block and two sets for the upper 2K-byte block. Each 2K-byte block has a *backup* reference bit and a *backup* change bit for the VM/370 control program and a *virtual* reference bit and a *virtual* change bit for the virtual machine.

At any instant, the VM/370 reference or change bit is respectively the logical OR of the backup reference or change bit and the reference or change bit in the real storage key. Similarly, the virtual-machine reference or change bit is the logical OR, respectively, of the virtual reference or change bit in the swap table and the reference or change bit in the real storage key. If a real page-table entry is invalid, the corresponding virtual page has been swapped out, but the virtual reference and change bits have been preserved in the corresponding swap-table entry.

Control of VM/370 Assists

Control register 6 points to a parameter list in real storage which is used to locate control blocks and tables used by the various VM/370 assists. Additionally, individual bits in control register 6 and in the assist control word (MICACF, the fifth word in the parameter list) determine which assist functions are active. Each assist requires a specific bit or group of bits to be ones for any function of that assist to be active. Furthermore, additional bits are used to activate or inhibit individual functions or groups of functions of some of the assists. Figure 3 shows these assignments.

	Positions That Need Ones for General Activation		Positions Used for Specific Activation	
VM/370 Assist	CR6	MICACF	CR6	MICACF
Virtual machine	0	–	2 thru 5	–
Shadow-table bypass	0	8	–	9 thru 15
Control program	6	–	–	–
Expanded virtual machine	0,6	–	–	0 thru 7
Virtual interval timer	0,7	–	–	–
VM extended facility	29	–	-	–

Figure 3. Bit Positions for General and Specific Activation of Assist Functions

The bit positions of control register 6 are used as follows:

Bits	*Function*
0	The functions of the virtual-machine assist, the shadow-table-bypass assist, the expanded virtual-machine assist, and the virtual-interval timer assist can be activated only when this bit is one.
1	This bit is the problem-state bit of the virtual-machine PSW.
2	The insert-storage-key function and the set-storage-key function of the virtual-machine assist can be activated only when this bit is zero.
3	Functions which assist virtual-machine instructions not found on System/360 can be activated only when this bit is zero.
4	The supervisor-call function of the virtual-machine assist can be activated only when this bit is zero.
5	The shadow-table-validation function of the virtual-machine assist can be activated only when this bit is ~~zero.~~ one .
6	The functions of the expanded virtual-machine assist and the new instructions of the control-program assist can be executed only when this bit is one.
7	The virtual-interval-timer assist can be activated only when this bit is one.
8-28	These bits are bits 8-28 of the real address of the doubleword-aligned parameter list.
29	The functions of the VM-extended-facility assist can be activated only when this bit is one.
30-31	These bits are not used.

VM/370 constructs a six-word parameter list for each virtual machine. This control block is called the MICBLOK in VM/370 logic publications. The format and names of the words in the parameter list are:

Bits of Word 0	*MICRSEG*
0-7	Real segment-table length.
8-25	Bits 8-25 of the address of the real segment table.
26-29	These bits are not used.
30	When the bit is zero, a 4K-byte real page size is specified; when it is one, a 2K-byte real page size is specified.
31	When the bit is zero, a 64K-byte real segment size is specified; when it is one, a 1M-byte real segment size is specified.

Bits of Word 1	*MICCREG*
0-7	These bits are not used.
8-31	Real address of virtual control registers (bits 29-31 must be zeros).

Bits of Word 2	*MICVPSW*
0	When the bit is zero, no virtual interruption is pending; when it is one, one or more virtual interruptions are pending.
1-7	These bits are not used.
8-31	Real address of the virtual PSW (bits 29-31 must be zeros).

Bits of Word 3	*MICWORK*
0-7	These bits are not used.
8-31	Real address of 64-byte workspace used during execution of assist functions on some models (bits 29-31 must be zeros).

Bits of Word 4	*MICVTMR*
0-7	These bits are not used.
8-31	Real address of virtual-interval-timer word (bits 30-31 must be zeros).

Bits of Word 5	*MICACF*
0-7	Each of the 11 functions of the expanded virtual-machine assist can be activated only when a specific bit in this group is one.
8	The functions of the shadow-table-bypass assist can be activated only when this bit is one.
9-15	Each of the eight functions of the shadow-table-bypass assist can be activated only when a specific bit in this group is one.
16-31	These bits are not used.

Programming Note

Placing nonzero values in unused positions of control register 6 or the parameter list is inadvisable because these positions may be used later and could cause incompatible execution.

Interaction of VM/370 Assists

Where the same virtual-machine instruction may be executed by more than one VM/370 assist, a fixed order exists among the assists in which similar functions are invoked. This order is (1) shadow-table-bypass assist, (2) virtual-machine assist, and (3) expanded virtual-machine assist. Thus, for a STORE THEN AND SYSTEM MASK instruction for a virtual machine, the STNSM function of the shadow-table-bypass assist is invoked. That function may (1) complete execution of the instruction, (2) exit via a program interruption, or (3) exit by invoking the STNSM function of the virtual-machine assist. The latter function may in turn (1) complete execution of the instruction, (2) exit via a program interruption, or (3) exit by invoking the STNSM function of the expanded virtual-machine assist.

Similarly, a page-translation condition may invoke the shadow-table-validation function of the virtual-machine assist or the page-fault-reflection function of the shadow-table-bypass assist if one of those assists is installed. If both assists are installed, the page-fault-reflection function of the shadow-table-bypass assist is invoked first for a page-translation condition. This function may result in (1) a virtual-machine program interruption, (2) a real-machine program interruption, or (3) invocation of the shadow-table-validation function of the virtual-machine assist.

Interaction of VM/370 Assists with Other Facilities

Control Mode

The virtual-machine assist and the shadow-table-bypass assist are defined to operate only on a CPU which is in the EC mode.

Program-Event Recording

When any assist for VM/370 completely executes a virtual-machine instruction, a program interruption takes place in the real machine for PER events if the real PER mask is one and in accordance with the settings of the PER controls in real control registers 9, 10, and 11. Storage-alteration events are indicated only for changes to virtual-machine storage. Changes in VM/370 control blocks in real storage are not indicated. However, a PER mask of one in the real or virtual PSWs may cause a virtual-machine-instruction assist function to exit by taking a program interruption before completely executing the instruction.

The specific rules governing the effect of PER mask values on the execution of VM/370 assist functions are:

1. The PER indication is unpredictable for the instructions of control-program assist.
2. A real PER mask value of one prevents a virtual-machine external interruption for a virtual interval-timer request.
3. A real PER mask value of one causes those expanded-virtual-machine-assist functions which can only partially execute a virtual-machine instruction to exit with a program interruption for a privileged-operation exception.
4. Load-PSW and supervisor-call functions of all assists exit by a program interruption for a privileged-operation exception or by a supervisor-call interruption if the PER mask is one in the real PSW, in the current virtual PSW, or in the new virtual PSW.
5. Set-system-mask, store-then-AND-system-mask, and store-then-OR-system mask functions whose definition allows complete execution under certain conditions exit with a program interruption for a privileged-operation exception if an attempt is made to change the PER mask in the virtual PSW.
6. Shadow-table validation and page-fault reflection are inactive when the real PER mask is one.

Dynamic Address Translation

In general, references to VM/370 control blocks in storage use real addresses and are thus not subject to translation by the dynamic-address-translation (DAT) facility. References to storage operands of virtual-machine instructions are subject to the DAT facility; translation is performed if the real CPU is in the EC mode with a one in bit 5 of the real PSW. Such translations are performed under control of the values in real control register 1 and in bit positions 8-12 of real control register 0.

At the end of the detailed description of each assist function is a table showing each field in storage that is or may be referred to in performing that assist function. For each field, the address type is shown as *real* or *logical.* Only those fields whose address type is logical are referred to by addresses subject to the DAT facility.

Low-Address Protection

When any assist for VM/370 makes an operand store access to virtual-machine storage during execution of a virtual-machine instruction, low-address protection is applied if the System/370 extended facility is installed and bit 3 of real control register 0 is one. The contents of virtual control register 0 have no effect on the application of low-address protection. Low-address protection is not applied to supervisor-call or program interruptions in the virtual machine, to updates of the virtual interval timer, or to references to VM/370 control blocks in the real machine when virtual-machine instructions are being executed. Low-address protection does apply to store accesses in the normal manner for instructions in the control-program assist.

Multiple Processors

No interlocking exists between the fetch and store parts of an update to VM/370 control blocks or the prefix-save areas (first 4K bytes of real storage) in the virtual-machine assist or shadow-table-bypass assist. The assist functions are not, in general, designed for use with multiprocessing or an attached processor. However, the purge-TLB function of the shadow-table-bypass assist has specific provision for operation under VM/370 running in attached-processor mode.

DOS/OS Compatibility Facility

The virtual-machine-assist functions are not invoked under control of the execute-local function of the DOS/OS compatibility facility.

General Conventions

The following control-program conventions are observed.

Control-Block Alignment

The control blocks ECBLOK and VMBLOK, which are referred to by functions in the virtual-machine assist (VMA) and shadow-table-bypass assist (STBA), are assumed to be doubleword-aligned. If the address MICCREG or MICVPSW, which is used to access one of the control blocks, does not specify the expected alignment, the result of an attempt to access a field in ECBLOK or VMBLOK depends on several factors. For the supervisor-call function of VMA, the supervisor-call interruption may take place; for the shadow-table-validation function of VMA or the page-fault-reflection function of STBA, a program interruption for the page-translation exception may take place; and for any other VMA or STBA function, a program interruption for a privileged-operation exception may take place. Otherwise, the value of a fetched field is unpredictable, and the storage locations modified by a store access may be any locations in the doublewords containing the misaligned field. Also, an addressing exception that is recognized may be for any part of the misaligned field.

Virtual PSW

Bits 0-15 of the virtual PSW are kept in the first halfword of a doubleword. When the real PSW is in the problem state, the PSW key, condition code, program mask, and instruction-address parts of the virtual PSW are kept in the corresponding parts of the real PSW, and the problem-state bit of the virtual PSW is kept in bit position 1 of control register 6. When storing takes place in the virtual PSW field, only the first halfword is significant, and the effect of the store operation on the remaining six bytes is unpredictable.

Updating Swap-Table Entries

Bytes 0, 2, and 3 of a doubleword swap-table entry may be updated by the reset-reference-bit function and the set-storage-key function. The method of updating may differ among models; individual bytes, halfwords, a word, or the entire doubleword of the entry may be fetched and subsequently stored with the specified bytes changed.

Addressing Conditions

Fields in VM/370 control blocks are accessed in assist functions with real addresses and with a key of zero. If such an access is to a storage location not available to the executing CPU, an addressing condition occurs.

If an addressing condition is encountered in accessing any field outside the storage of the virtual machine during execution of any of the assist functions, the following action is taken:

1. If the definition has not specified that a store access has been made when the condition is encountered, the execution of the assist ends with an interruption. For any instructions assisted, a program interruption takes place, and instruction execution is suppressed; it is unpredictable whether a privileged-operation or an addressing exception is indicated. For the shadow-table-validation function or the page-fault-reflection function, a program interruption takes place, and it is unpredictable whether a page-translation exception or an addressing exception is indicated. For the supervisor-call function, it is unpredictable whether an interruption takes place or a program interruption takes place with an addressing exception indicated.
2. If a store access has been made before the addressing condition is encountered, a program interruption for an addressing exception is taken, and instruction execution is terminated.

The detailed descriptions of assist functions do not explicitly mention addressing conditions that may occur when an invalid address is assigned to the workspace. This address is found in MICWORK, word 3 of the parameter list. References to the workspace are model-dependent. The interruptions resulting from addressing conditions in accessing the workspace are in addition to those enumerated in the detailed descriptions of assist functions.

Method of Detailed Description

Chapter 2 contains a detailed description of each virtual-machine-assist function. Chapter 4 contains a detailed description of each function of the shadow-table-bypass assist. Function execution is broken down into steps. An alphameric designation, the priority indicator, appears in parentheses at the end of each step.

Most steps state the conditions under which execution of the function ends with that step. If the conditions for ending execution are met for two or more steps, execution ends with the step having the highest priority. The relative priority of two steps is determined by comparing the numbers and letters of the priority indicators of those two steps from left to right to find the first differing position. When the first differing position contains a number, the step whose priority indicator has the lower number has the higher priority. When the first differing position contains a letter, both steps are of equal priority.

When two or more steps having the same priority have their ending conditions satisfied, it is unpredictable with which of those steps execution of the function will actually end.

At the end of the detailed description of each assist function is a summary showing each field of each control block in storage that is accessed. The offset shown is the offset to be applied to the address by which that control block was located. In general, this is the same as the offset within the control block. However, the field, VMPSW, is located at offset A8 hex within the VMBLOK. The offset is shown as zero because that field is directly located by the contents of MICVPSW without the application of any additional offset.

Chapter 2. Virtual-Machine Assist

The virtual-machine assist (VMA) is the basic assist for VM/370. The virtual-machine assist improves the performance of virtual machines executing under VM/370 by directly executing 12 instructions for virtual machines which would otherwise be simulated by VM/370. The virtual-machine assist also directly validates the appropriate page-table entry of the shadow tables when a page-translation condition is encountered in executing a virtual-machine program and when the corresponding virtual and real page-table entries are both valid.

Figure 4 shows the virtual-machine instructions that can be directly executed using the virtual-machine assist, as well as the settings of bits in control register 6 required for assisting each function. An X indicates that the value may be either zero or one.

Function	Bits 0-5 of Control Register 6
Insert PSW key	10X0 XX
Insert storage key	100X XX
Load PSW	10XX XX
Load real address	10X0 XX
Reset reference bit	10X0 XX
Set PSW key from address	10X0 XX
Set storage key	100X XX
Set system mask	10XX XX
Store control	10X0 XX
Store then AND system mask	10X0 XX
Store then OR system mask	10X0 XX
Supervisor call	1XXX 0X
Shadow-table validation	1XXX X1

Figure 4. Controls for Activating VMA Functions

Programming Notes

1. Bit 3 of control register 6 is normally set to zero for a System/370 virtual machine and to one for a System/360 virtual machine. In this way, System/370 instructions encountered in a System/360 virtual-machine program cause a program interruption to take place for a privileged-operation exception.
2. Bit 5 of control register 6 is normally set to zero for a virtual machine in BC mode or in EC mode with DAT off. It is set to one when shadow translation tables are in use, that is, when the virtual machine is in EC mode and DAT is on. However, shadow tables may be avoided for a virtual machine with the virtual = real option in VM/370 and the virtual translation tables used directly.
3. A one in bit position 2 of control register 6 inhibits the virtual execution of INSERT STORAGE KEY and SET STORAGE KEY. Similarly, a one in bit position 4 of control register 6 inhibits the virtual execution of SUPERVISOR CALL.

INSERT PSW KEY

The INSERT PSW KEY instruction is executed for a virtual machine if the virtual-machine assist is activated for System/370 instructions, unless (1) a virtual-machine exception is recognized or (2) the virtual PSW cannot be fetched.

The insert-PSW-key function of the virtual-machine assist is invoked each time a CPU attempts to execute an INSERT PSW KEY instruction when the problem-state bit of the real PSW is one. Execution of this function consists in performing the following steps:

1. If bits 0-3 of control register 6 are not 10X0 binary, execution of the insert-PSW-key function ends, a program interruption takes place for a privileged-operation exception, and execution of the INSERT PSW KEY instruction is suppressed (1.A.1).
2. The word MICVPSW, which contains the address of the virtual PSW, is fetched with a key of zero. Execution ends if an addressing condition is encountered (1.A.2).
3. VMPSW, the virtual PSW, is fetched with a key of zero. Execution ends if an addressing condition is encountered (1.A.3).
4. If an access exception is encountered in fetching the second halfword of the INSERT PSW KEY instruction, it is unpredictable whether this condition is ignored because no information is needed from that halfword to execute the instruction. If an access condition is encountered and is not ignored, execution of this function ends, and a program interruption takes place for that access exception (1.B).
5. The four-bit protection key of the virtual PSW is placed in bit positions 24-27 of general register 2. Bits 0-23 of general register 2 remain unchanged, and bits 28-31 are set to zeros (2).

Figure 5 summarizes the fields used.

Field Name	Control Block	Address Type	Offset (Hex)	No. Of Bytes	Contents
MICVPSW	MICBLOK	Real	8	4	Address of VMPSW
VMPSW	VMBLOK	Real	0	2	Virtual PSW bits 0-15

Figure 5. Fields Used in INSERT PSW KEY

INSERT STORAGE KEY

The INSERT STORAGE KEY instruction is executed for a virtual machine if the virtual-machine assist is activated for this instruction unless (1) a virtual-machine exception is recognized, (2) the real page size is 2K bytes, or (3) some pertinent VM/370 control field cannot be fetched.

The insert-storage-key function of the virtual-machine assist is invoked each time a CPU attempts to execute an INSERT STORAGE KEY instruction when the problem-state bit of the real PSW is one. Execution of this function consists in performing the following steps:

1. Execution of this function ends with a program interruption for a privileged-operation exception if bits 0-2 of control register 6 are not 100 binary or if bits 28-31 of the general register specified by the R_2 field of the instruction are not zeros (1).
2. The word MICRSEG, which contains the real segment-table address, is fetched with a key of zero. Execution ends if an addressing condition is encountered (2.A.1).
3. Execution ends with a program interruption for a privileged-operation exception if bit 30 of MICRSEG is one (2.A.2).
4. The address in the general register specified by the R_2 field of the instruction is partitioned to obtain the segment index and the page index. The partitioning of the address is based on 4K-byte pages and either 64K-byte or 1M-byte segments, depending on whether bit 31 of MICRSEG is zero or one, respectively. For a 64K-byte segment, execution ends with a program interruption for a privileged-operation exception if the segment-table-length value in bit positions 0-7 of MICRSEG is less than the value obtained by appending four zeros to the left of bits 8-11 of the address specified (2.A.3).
5. The address of the real segment-table entry is computed and the entry SEGPAGE is fetched with a key of zero. Execution ends if an addressing condition is encountered (2.A.4).
6. Execution ends with a program interruption for a privileged-operation exception if the segment-table entry is invalid, if the entry has an invalid format, or if the value of the leftmost four bits of the page index exceeds the value of bits 0-3 of the segment-table entry (2.A.5).
7. The location of the word PAGSWP, containing the swap-table address, is computed by subtracting 4 from the page-table origin derived from the segment-table entry. The word at this location is fetched with a key of zero. Execution ends if an addressing condition is encountered (2.A.6.A.1).
8. Eight times the page index is added to bits 8-31 of a swap-table-address word. The swap-table word at the address computed is fetched with a key of zero. Execution ends if an addressing condition is encountered. SWPKEY1 and SWPKEY2 are in bytes 2 and 3 of the word fetched (2.A.6.A.2).
9. Twice the page index is added to the page-table origin from the segment-table entry to obtain the address of the page-table entry. The page-table entry PAGCORE is fetched with a key of zero. Execution ends if an addressing condition is encountered (2.A.6.B.1).
10. Execution ends with a program interruption for a privileged-operation exception if the page-table entry is valid and has an invalid format (2.A.6.B.2).
11. If the page-table entry is valid, the real storage key is fetched. Execution ends if an addressing condition is encountered in fetching the real storage key. If the page-table entry is invalid, this step is not performed (2.A.6.B.3).
12. The word MICVPSW, containing the virtual-PSW address, is fetched with a key of zero. Execution ends if an addressing condition is encountered (2.B.1).
13. VMPSW, the virtual PSW, is fetched with a key of zero. Execution ends if an addressing condition is encountered (2.B.2).
14. Execution of INSERT STORAGE KEY is completed by modifying the contents of the

general register (first operand) specified by the R_1 field of the instruction. The modification depends on the value of bit 20 of the second-operand address, on whether the virtual PSW is in EC or BC mode, and on whether the real page-table entry is valid or invalid. The setting of the first-operand bit positions is as shown in Figure 6 (3).

Figure 7 summarizes the fields used.

Bit Positions of First Operand	Setting of First-Operand Bits When Bit 20 of Second Operand Is:	
	Zero	One
0-23	Unchanged	Unchanged
24-28	Bits 16-20 of swap-table word	Bits 24-28 of swap-table word
29-30 in virtual BC mode	Zeros	Zeros
29-30 in virtual EC mode, invalid PTE	Bits 21-22 of swap-table word	Bits 29-30 of swap-table word
29-30 in virtual EC mode, valid PTE	OR bits 5-6 of storage key, bits 21-22 of swap-table word	OR bits 5-6 of key, bits 29-30 of swap-table word
31	Zero	Zero

Figure 6. Setting of First-Operand Bits

Field Name	Control Block	Address Type	Offset (Hex)	No. of Bytes	Contents
MICRSEG	MICBLOK	Real	0	4	Address of real segment table
MICVPSW	MICBLOK	Real	8	4	Address of VMPSW
VMPSW	VMBLOK	Real	0	2	Virtual PSW bits 0-15
SEGPAGE	SEGTABLE	Real	4SX	4	Segment-table entry
PAGSWP	PAGTABLE	Real	-4	4	Address of swap table
PAGCORE	PAGTABLE	Real	2PX	2	Page-table entry
SWPKEY1	SWPTABLE	Real	8PX+2	1	Low 2K-byte virtual key
SWPKEY2	SWPTABLE	Real	8PX+3	1	High 2K-byte virtual key

Figure 7. Fields Used in INSERT STORAGE KEY

LOAD PSW

The LOAD PSW instruction is executed for a virtual machine if the virtual-machine assist is activated, unless (1) a virtual-machine interruption may follow, (2) the second operand or some pertinent VM/370 control field cannot be accessed, (3) the PER mask is one in the real PSW or in the old or the new virtual PSW, or (4) execution would change the control mode, the DAT bit, or the wait-state bit of the virtual PSW.

The load-PSW function of the virtual-machine assist is invoked each time a CPU attempts to execute a LOAD PSW instruction when the problem-state bit of the real PSW is one. Execution of this function consists in performing the following steps:

1. If bits 0-1 of control register 6 are not 10 binary, execution of the load-PSW function ends, a program interruption takes place for a privileged-operation exception, and execution of the LOAD PSW instruction is suppressed (1.A).
2. The second halfword of the LOAD PSW instruction is fetched. If an access condition is encountered, execution of this function ends, and a program interruption takes place for the access exception encountered (1.B).
3. If bits 29-31 of the second-operand address are not zeros or if the PER mask in the real PSW is one, execution ends with a program interruption for a privileged-operation exception (2.A).
4. A doubleword is fetched with the logical address of the second operand and the PSW key. If an access condition is encountered, execution of this function ends with a program interruption for the access condition found (2.B.1).
5. Execution of the load-PSW function ends if either of the following conditions is found (2.B.2):
   a. The new virtual PSW (second operand) has the wait-state bit set to one.
   b. The new virtual PSW has a one in bit position 12 (EC mode) and bits 0-4, 16-17, and 24-39 are not all zeros (PER mask is one or a format error exists).

   If execution ends, control passes to the load-PSW function of the expanded virtual-machine assist if that facility is installed. Otherwise, execution ends with a program interruption for a privileged-operation exception.
6. The word MICVPSW, containing the virtual PSW address, is fetched with a key of zero. Execution ends if an addressing condition is encountered (2.C.1).
7. VMPSW, the virtual PSW, is fetched with a key of zero. Execution ends if an addressing condition is encountered (2.C.2).
8. Execution ends if the virtual PSW has ones in bit positions 1 and 12 (PER mask is one in the EC mode). If execution ends, control passes to the load-PSW function of the expanded virtual-machine assist if that facility is installed. Otherwise, execution ends with a program interruption for a privileged-operation exception (2.C.3.A).
9. Execution of the load-PSW function ends if any of the following conditions holds (2.C.3.B):
   a. The control mode of the virtual PSW is being changed from the BC to the EC mode, or from the EC mode to the BC mode.
   b. The DAT-mode bit of an EC-mode virtual PSW is being changed.
   c. A virtual interruption is pending, and any channel mask, input/output mask, or external mask is being changed from zero to one (bits 0-7 in the BC mode and bits 6-7 in the EC mode). A virtual interruption is pending when bit 0 of MICVPSW is one.

   Note that, because all these conditions require the value fetched in step 4, this step necessarily has a lower priority than step 4 despite the priority rules based on priority indicators.

   If execution ends, control passes to the load-PSW function of the expanded virtual-machine assist if that facility is installed. Otherwise, execution ends with a program interruption for a privileged-operation exception.
10. The key, the condition code, the program mask, and instruction-address-field values of the new virtual-machine PSW replace the corresponding fields in the real PSW. Bit 15 of the virtual-machine PSW is placed in bit position 1 of control register 6. The new virtual-machine PSW is stored, with a key of zero, in VMPSW (3).

Figure 8 summarizes the fields used.

Field Name	Control Block	Address Type	Offset (Hex)	No. of Bytes	Contents
MICVPSW	MICBLOK	Real	4	4	Address of VMPSW
VMPSW	VMBLOK	Real	0	2	Virtual PSW bits 0-15
Operand 2	—	Logical	—	8	New virtual PSW

Figure 8. Fields Used in LOAD PSW

LOAD REAL ADDRESS

The LOAD REAL ADDRESS instruction is executed for a virtual machine if the virtual-machine assist is activated for System/370 instructions, unless (1) a virtual-machine exception is recognized, or (2) some pertinent VM/370 control field cannot be fetched.

If the shadow-table-bypass assist is not installed, the load-real-address function of the virtual-machine assist is invoked each time a CPU attempts to execute a LOAD REAL ADDRESS instruction when the problem-state bit of the real PSW is one. If the shadow-table-bypass assist is installed, the load-real-address function of the virtual-machine assist may be invoked only from the load-real-address function of the shadow-table-bypass assist. Execution of this function consists in performing the following steps:

1. If bits 0-3 of control register 6 are not 10X0 binary, execution of the load-real-address function ends, and a program interruption takes place for a privileged-operation exception (1.A.1).
2. The doubleword containing the fields MICRSEG and MICCREG is fetched with a key of zero. MICRSEG contains the address of the real segment table, and MICCREG contains the address of the extended control block (ECBLOK). Execution ends if an addressing condition is encountered (1.A.2).
3. The first doubleword (EXTCR0 and EXTCR1, virtual control registers 0 and 1) of the extended-control block (ECBLOK) is fetched with a key of zero. Execution ends if an addressing condition is encountered (1.A.3).
4. Execution of this function ends with a program interruption for a privileged-operation exception if bits 8-12 of virtual control register 0 have an invalid format (1.A.4).
5. The second halfword of the LOAD REAL ADDRESS instruction is fetched. If an access condition is encountered, execution of this function ends, and a program interruption takes place for the access exception encountered (1.B).
6. The segment index, the page index, and the byte index of the second-operand address are found. They are selected from effective-address positions dependent on bits 8-12 of virtual control register 0. For a virtual segment of 64K bytes, execution ends if the segment-table-length value in bit positions 0-7 of virtual control register 1 is less than the value obtained by appending four zeros to the left of bits 8-11 of the second-operand address. In that case, condition code 3 is set, and the virtual address of the segment-table entry that would have been referred to had no length violation existed is placed in the general register specified by the R_1 field of the instruction (2).
7. The virtual address of the virtual segment-table entry is computed. The real segment index and the real page index of the virtual address of the virtual segment-table entry are found. The real page size is 4K bytes or 2K bytes as bit 30 of MICRSEG is zero or one; the real segment size is 64K bytes or 1M byte as bit 31 of MICRSEG is zero or one. For a 64K-byte segment, execution ends with a program interruption for a privileged-operation exception if the segment-table-length value in bit positions 0-7 of MICRSEG is less than the value obtained by appending four zeros to the left of bits 8-11 of the virtual segment-table-entry address (3).
8. The address of the real segment-table entry, SEGPAGE, for translating the virtual segment-table-entry address is computed. This address is used with a key of zero to fetch the corresponding real segment-table entry. Execution ends if an addressing condition is encountered (4).
9. Execution ends with a program interruption for a privileged-operation exception if the fetched entry is invalid or has an invalid format or if the leftmost four bits of the real page index of the virtual segment-table entry address are greater in value than bits 0-3 of the real segment-table entry fetched (5).

10. Twice the real page index of the virtual segment-table entry address is added to the page-table origin from the segment-table entry to obtain the real address of the page-table entry for translating the virtual segment-table-entry address. This address is used with a key of zero to fetch the corresponding real page-table entry PAGCORE. Execution ends if an addressing condition is encountered (6).
11. Execution ends with a program-interruption for a privileged-operation exception if the page-table entry is invalid or has an invalid format (7).
12. The real address of the virtual segment-table entry is computed and used with a key of zero to fetch the virtual segment-table entry. Execution ends if an addressing condition is encountered (8).
13. If the segment-table entry fetched was invalid, condition code 1 is set, the virtual address of the virtual segment-table entry which was developed in step 7 is placed in the general register specified by the R_1 field of the instruction, and execution of this function ends with completion of the LOAD REAL ADDRESS instruction (9).
14. If the segment-table entry fetched had an invalid format, execution of this function ends with a program interruption for a privileged-operation exception (10).
15. If the value of the four leftmost bits of the page index developed in step 6 is greater than the value of bits 0-3 of the segment-table entry fetched, execution of this entry ends with completion of LOAD REAL ADDRESS. In that case, condition code 3 is set, and the virtual address of the page-table entry that would have been referred to had no length violation existed is placed in the general register specified by the R_1 field of the instruction (11).
16. The virtual address of the virtual page-table entry is computed by using the page index developed in step 6. The real segment index and the real page index of the virtual address for the virtual page-table entry are found. The real page size is 4K bytes or 2K bytes as bit 30 of MICRSEG is zero or one, and the real segment size is 64K bytes or 1M byte as bit 31 of MICRSEG is zero or one. For a 64K-byte segment, execution ends with a program interruption for a privileged-operation exception if the segment-table length in bit positions 0-7 of MICRSEG is less than the value obtained by appending four zeros to the left of bits 8-11 of the virtual page-table-entry address (12).
17. The address of SEGPAGE, the real segment-table entry for translating the virtual page-table-entry address, is computed. This address is used with a key of zero to fetch the corresponding real segment-table entry. Execution ends if an addressing condition is encountered (13).
18. Execution ends with a program interruption for a privileged-operation exception if the real segment-table entry fetched is invalid or has an invalid format or if the leftmost four bits of the real-page index of the virtual page-table-entry address are greater in value than bits 0-3 of the real segment-table entry fetched (14).
19. Twice the real page index of the virtual page-table-entry address is added to the page-table origin from the segment-table entry to obtain the real address of the page-table entry for translating the virtual page-table-entry address. This address is used with a key of zero to fetch the corresponding real page-table entry PAGCORE. Execution ends if an addressing condition is encountered (15).
20. Execution ends with a program interruption for a privileged-operation exception if the real page-table entry fetched is invalid or has an invalid format (16).
21. The real address of the virtual page-table entry is computed and used with a key of zero to fetch the virtual page-table entry. Execution ends if an addressing condition is encountered (17).
22. If the virtual page-table entry fetched is invalid, condition code 2 is set, the virtual address of the page-table entry which was developed in step 16 is placed in the general register specified in the R_1 field of the instruction, and execution of this function ends with completion of the LOAD REAL ADDRESS instruction (18).
23. If the virtual page-table entry fetched had an invalid format, execution of this function ends with a program interruption for a privileged-operation exception (19).
24. Execution ends with completion of the LOAD REAL ADDRESS instruction. Condition code 0 is set, and a new value is placed in the general register specified by the R_1 field of the instruction. Bits 0-7 of that general register are set to zeros. Bits 8-31 are set to the value obtained by prefixing the page-frame address in the entry fetched in step 21 to the byte index developed in step 6 (20).

Figure 9 summarizes the fields used.

Field Name	Control Block	Address Type	Offset (Hex)	No. of Bytes	Contents
MICRSEG	MICBLOK	Real	0	4	Address of real segment table
MICCREG	MICBLOK	Real	4	4	Address of ECBLOK
EXTCR0	ECBLOK	Real	0	4	Virtual CR0
EXTCR1	ECBLOK	Real	4	4	Virtual CR1
SEGPAGE	SEGTABLE	Real	4SX[1]	4	Segment-table entry
PAGCORE	PAGTABLE	Real	2PX[1]	2	Page-table entry
–	–	Real	0	4	Virtual segment-table entry
SEGPAGE	SEGTABLE	Real	4SX[2]	4	Segment-table entry
PAGCORE	PAGTABLE	Real	2PX[2]	2	Page-table entry
–	–	Real	0	2	Virtual page-table entry

1 References to the real translation tables to translate the virtual address of the virtual segment-table entry

2 References to the real translation tables to translate the virtual address of the virtual page-table entry

Figure 9. Fields Used in LOAD REAL ADDRESS

RESET REFERENCE BIT

The virtual RESET REFERENCE BIT instruction is executed for a virtual machine if the virtual-machine assist is activated for System/370 instructions, unless (1) a virtual-machine exception is recognized, (2) the real page size is 2K bytes, or (3) some pertinent VM/370 control field cannot be accessed.

The reset-reference-bit function of the virtual-machine assist is invoked each time a CPU attempts to execute a RESET REFERENCE BIT instruction when the problem-state bit of the real PSW is one. Execution of this function consists in performing the following steps:

1. Execution of this function ends with a program interruption for a privileged-operation exception if bits 0-3 of control register 6 are not 10X0 binary (1.A.1).
2. The word MICRSEG, which contains the real segment-table address, is fetched with a key of zero. Execution ends if an addressing condition is encountered (1.A.2).
3. Execution ends with a program interruption for a privileged-operation exception if bit 30 of MICRSEG is one (1.A.3).
4. If an access condition is encountered in fetching the second halfword of the RESET REFERENCE BIT instruction, execution of this function ends with a program interruption for that access exception (1.B).
5. The second-operand effective address is partitioned to obtain the segment index and the page index. The partitioning of the address is based on 4K-byte pages and either 64K-byte or 1M-byte segments, depending on whether bit 31 of MICRSEG is zero or one, respectively. For a 64K-byte segment, execution ends with a program interruption for a privileged-operation exception if the segment-table-length value in bit positions 0-7 of MICRSEG is less than the value obtained by appending four zeros to the left of bits 8-11 of the second-operand address (2).
6. The address of SEGPAGE, the real segment-table entry, is computed, and the entry is fetched with a key of zero. Execution ends if an addressing condition is encountered (3).
7. Execution ends with a program interruption for a privileged-operation exception if the segment-table entry is invalid, if the entry has

an invalid format, or if the value of the leftmost four bits of the page index exceeds the value of bits 0-3 of the segment-table entry (4).

8. The location of the swap-table address is computed by subtracting 4 from the page-table origin derived from the segment-table entry. The word at this location, PAGSWP, is fetched with a key of zero. Execution ends if an addressing condition is encountered (5.A.1).
9. Eight times the page index is added to bits 8-31 of the swap-table-address word to obtain the address of the swap-table word. The swap-table word at the address computed is fetched with a key of zero. Execution ends if an addressing condition is encountered. SWPFLG, SWPKEY1, and SWPKEY2 are in bytes 0, 2, and 3 of the word fetched (5.A.2).
10. Twice the page index is added to the page-table origin from the segment-table entry to obtain the address of the page-table entry. The page-table entry, PAGCORE, is fetched with a key of zero. Execution ends if an addressing condition is encountered (5.B.1).
11. Execution ends with a program interruption for a privileged-operation exception if the page-table entry is valid and has an invalid format (5.B.2).
12. If the page-table entry is valid, the reference and change bits of the real storage key are fetched, and the reference bit in the real storage key is set to zero. Execution ends if an addressing condition is encountered. If the page-table entry is invalid, execution continues as if real reference-bit and change-bit values of zero had been fetched (5.B.3).
13. Reference-bit and change-bit values used to determine the condition-code setting are obtained by taking the logical OR of the real reference-bit and real change-bit values and the virtual reference-bit and virtual change-bit values, respectively. The swap-table word previously fetched is updated in storage, with a key of zero, by computing new values of three bits. The backup reference-bit value and backup change-bit value are updated by logically ORing those values and the values of the real reference-bit and real change-bit values, respectively. In addition, the virtual reference bit of the swap-table word is set to zero. The swap-table word contains two sets of virtual and backup reference and change bits. The value in bit position 20 of the second-operand effective address determines which set is used in this step. Figure 10 shows the bit position in the swap-table word used for each bit (6).

The condition code is set, and execution of this function ends with completion of the RESET REFERENCE BIT instruction.

Figure 11 summarizes fields used.

	Bit Used in Swap Table When Bit 20 of Second Operand Is:	
Bit Function	Zero	One
Backup reference-bit position	4	6
Backup change-bit position	5	7
Virtual reference-bit position	21	29
Virtual change-bit position	22	30

Figure 10. Bits Used in Swap Table for RESET REFERENCE BIT

SET PSW KEY FROM ADDRESS

The SET PSW KEY FROM ADDRESS instruction is executed for a virtual machine if the virtual-machine assist is activated for System/370 instructions, unless (1) a virtual-machine exception is recognized or (2) the virtual PSW key cannot be accessed.

The set-PSW-key-from-address function of the virtual-machine assist is invoked each time a CPU attempts to execute a SET PSW KEY FROM ADDRESS instruction when the problem-state bit of the real PSW is one. Execution of this function consists in performing the following steps:

1. If bits 0-3 of control register 6 are not 10X0 binary, execution of the set-PSW-key-from-address function ends, a program interruption takes place for a privileged-operation exception, and execution of the SET PSW KEY FROM ADDRESS instruction is suppressed (1.A).
2. If an access condition is encountered in fetching the second halfword of the SET PSW KEY FROM ADDRESS instruction, execution of this function ends with a program interruption for that access exception (1.B).
3. The word MICVPSW, which contains the virtual-PSW address, is fetched with a key of zero. Execution ends if an addressing condition is encountered (2).
4. ~~Bits 8-11 of the virtual PSW, VMPSW, are~~ replaced, with a key of zero, by bits 24-27 of the second-operand address of the SET PSW KEY FROM ADDRESS instruction. Execution ends if an addressing condition is encountered (3).
5. Bits 24-27 of the second-operand address also replace the PSW key (bits 8-11) of the real PSW. Execution ends with completion of the SET PSW KEY FROM ADDRESS instruction, and the next instruction is fetched with the new PSW key (4).

Figure 12 summarizes the fields used.

Field Name	Control Block	Address Type	Offset (Hex)	No. of Bytes	Contents
MICRSEG	MICBLOK	Real	8	4	Address of real segment table
SEGPAGE	SEGTABLE	Real	4SX	4	Segment-table entry
PAGSWP	PAGTABLE	Real	−4	4	Address of swap table
PAGCORE	PAGTABLE	Real	2PX	2	Page-table entry
SWPFLG	SWPTABLE	Real	0	1	Backup bits
SWPKEY1	SWPTABLE	Real	2	1	Low 2K-byte virtual key
SWPKEY2	SWPTABLE	Real	3	1	High 2K-byte virtual key

Figure 11. Fields Used in RESET REFERENCE BIT

Field Name	Control Block	Address Type	Offset (Hex)	No. of Bytes	Contents
MICVPSW	MICBLOK	Real	0	4	Address of virtual PSW
VMPSW	VMBLOK	Real	0	2	Virtual PSW bits 0-15

Figure 12. Fields Used in SET PSW KEY FROM ADDRESS

SET STORAGE KEY

The SET STORAGE KEY instruction is executed for a virtual machine if the virtual-machine assist is activated for this instruction, unless (1) a virtual-machine exception is recognized, (2) the real page size is 2K bytes, or (3) some pertinent VM/370 control field cannot be accessed.

The set-storage-key function of the virtual-machine assist is invoked each time a CPU attempts to execute a SET STORAGE KEY instruction when the problem-state bit of the real PSW is one. Execution of this function consists in performing the following steps:

1. Execution of this function ends with a program interruption for a privileged-operation exception if bits 0-2 of control register 6 are not 100 binary or if bits 28-31 of the general register specified by the R_2 field of the instruction are not zeros (1).
2. The word MICRSEG, which contains the real segment-table address, is fetched with a key of zero. Execution ends if an addressing condition is encountered (2).
3. Execution ends with a program interruption for a privileged-operation exception if bit 30 of MICRSEG is one (3).
4. The address in the general register specified by the R_2 field is partitioned to obtain the segment index and the page index. The partitioning of the address is based on 4K-byte pages and either 64K-byte or 1M-byte segments, depending on whether bit 31 of MICRSEG is zero or one, respectively. For a 64K-byte segment, execution ends with a program interruption for a privileged-operation exception if the segment-table-length value in bit positions 0-7 of MICRSEG is less than the value obtained by appending four zeros to the left of bits 8-11 of the address specified (4).
5. The address of the real-segment-table entry, SEGPAGE, is computed, and the entry is fetched with a key of zero. Execution ends if an addressing condition is encountered (5).
6. Execution ends with a program interruption for a privileged-operation exception if the segment-table entry is invalid, if the entry has an invalid format, or if the value of the leftmost four bits of the page index exceeds the value of bits 0-3 of the segment-table entry (6).
7. The location of the swap-table address is computed by subtracting 4 from the page-table origin derived from the segment-table entry. The word at this location, PAGSWP, is fetched with a key of zero. Execution ends if an addressing condition is encountered (7.A.1).
8. Eight times the page index is added to bits 8-31 of the swap-table-address word to obtain the address of the swap-table word. The swap-table word at the address computed is fetched with a key of zero. Execution ends if an addressing condition is encountered. SWPFLG, SWPKEY1, and SWPKEY2 are in bytes 0, 2, and 3 of the word fetched (7.A.2).
9. Twice the page index is added to the page-table origin from the segment-table entry to obtain the address of the page-table entry. The page-table entry, PAGCORE, is fetched with a key of zero. Execution ends if an addressing condition is encountered (7.B.1).
10. Execution ends with a program interruption for a privileged-operation exception if the page-table entry is valid and has an invalid format (7.B.2).
11. If the page-table entry is valid, the reference and change bits are fetched from the real storage key, and a new value is placed in the real storage key. Execution ends if an addressing condition is encountered. Bits 0-4 of the new value are obtained from bit positions 24-28 of the general register specified by R_2 field of the instruction. Bits 5-6 (the reference-bit and change-bit positions) of the new value are zeros. If the page-table entry is invalid, execution continues as if real reference-bit and change-bit values of zero had been fetched (7.B.3).
12. The swap-table word previously fetched is updated in storage, with a key of zero, by computing new values of 10 bits. The backup reference-bit and change-bit values are updated by logically ORing those values and the values of the real reference-bit and real change-bit values, respectively. In addition, bits 0-6 of the swap-table byte containing the virtual storage key are replaced by bits 24-30 of the general register specified by the R_2 field of the instruction. Bit 7 of that byte is set to an unpredictable value. The swap-table word contains two sets of backup bits and virtual storage-key bytes. The value in bit position 20 of the general register (the second operand) specified by the R_2 field of the instruction determines which set is used in this step. Figure 13 shows the bit positions of the bits and byte used (8).

Execution of this function ends with completion of the SET STORAGE KEY instruction.

Figure 14 summarizes the fields used.

SET SYSTEM MASK

The SET SYSTEM MASK instruction is executed for a virtual machine if the virtual-machine assist is activated, unless (1) a virtual-machine interruption may follow, (2) the second operand or some pertinent VM/370 control field cannot be accessed, or (3) the PER mask or the DAT bit of the virtual PSW would be changed.

The set-system-mask function of the virtual-machine assist is invoked each time a CPU attempts to execute a SET SYSTEM MASK instruction when the problem-state bit of the real PSW is one. Execution of this function consists in performing the following steps:

1. If bits 0-1 of control register 6 are not 10 binary, execution of the set-system-mask function ends, a program interruption takes place for a privileged-operation exception, and execution of the SET SYSTEM MASK instruction is suppressed (1.A.1).
2. The word MICCREG, which contains the address of the ECBLOK, is fetched with a key of zero. Execution ends if an addressing condition is encountered (1.A.2).
3. The virtual control-register-0 value, EXTCR0, which is in the first word of the ECBLOK, is fetched with a key of zero. Execution ends if an addressing condition is encountered (1.A.3).
4. ~~If bit position 1 of virtual control register 0~~ contains a one, execution ends with a program interruption for a privileged-operation exception (1.A.4).
5. If an access condition is encountered in fetching the second halfword of the SET SYSTEM MASK instruction, execution of this function ends, and a program interruption takes place for the access exception encountered (1.B).
6. One byte is fetched with the logical address of the second operand and the PSW key. If an access condition is encountered, execution of this function ends with a program interruption for the access exception found (2.A).
7. The word MICVPSW, which contains the address of the virtual PSW, is fetched with a key of zero. Execution ends if an addressing condition is encountered (2.B.1).
8. The first halfword of VMPSW, which contains the virtual PSW, is fetched with a key of zero. Execution ends if an addressing condition is encountered (2.B.2).

Function	Bits Used in Swap Table When Bit 20 of Second Operand Is:	
	Zero	One
Backup reference-bit position	4	6
Backup change-bit position	5	7
Virtual storage-key byte	16-23	24-31

Figure 13. Bits Used in Swap Table for SET STORAGE KEY

Field Name	Control Block	Address Type	Offset (Hex)	No. of Bytes	Contents
MICRSEG	MICBLOK	Real	0	4	Address of real segment table
SEGPAGE	SEGTABLE	Real	4SX	4	Segment-table entry
PAGSWP	PAGTABLE	Real	-4	4	Address of swap table
PAGCORE	PAGTABLE	Real	2PX	2	Page-table entry
SWPFLG	SWPTABLE	Real	0	1	Backup bits
SWPKEY1	SWPTABLE	Real	2	1	Low 2K-byte virtual key
SWPKEY2	SWPTABLE	Real	3	1	High 2K-byte virtual key

Figure 14. Fields Used in SET STORAGE KEY

9. If the virtual PSW is in EC mode, the following conditions are verified for byte 0 of the virtual PSW and the operand byte that is to replace it (3):
   a. The DAT mode bit is unchanged.
   b. The PER mask is unchanged.
   c. Bits 0, 2, 3, and 4 of the new PSW are zeros.
   d. Neither the I/O mask (bit 6) nor the external mask (bit 7) of the PSW is to be changed from zero to one if a virtual interruption is pending (that is, if bit 0 of the word containing the virtual PSW address is one).

   In the absence of any of these conditions, execution ends with the invoking of the set-system-mask function of the expanded-virtual-machine assist if that facility is installed. If that facility is not installed, execution ends with a program interruption for a privileged-operation exception.

   When the virtual PSW is in BC mode, execution ends if any channel mask, I/O mask, or external mask is to be changed from zero to one and a virtual interruption is pending. Execution ends with a program interruption for a privileged-operation exception.
10. The operand byte fetched is stored with a key of zero as byte 0 of the virtual PSW. Execution of this function ends with completion of the SET SYSTEM MASK instruction (4).

Figure 15 summarizes the fields used.

Programming Note
The value in bit position 1 of the real control register 0 has no effect on the execution of the set-system-mask function of the virtual-machine assist.

STORE CONTROL

The STORE CONTROL instruction is executed for a virtual machine if the virtual-machine assist is activated for System/370 instructions, unless (1) a virtual-machine exception is recognized, or (2) the second operand or some pertinent VM/370 control field cannot be fetched.

The store-control function of the virtual-machine assist is performed each time the CPU attempts to execute a STORE CONTROL instruction when the problem-state bit of the real PSW is one. Execution of this function consists in performing the following steps:

1. If bits 0-3 of control register 6 are not 10X0 binary, execution of this function ends, and a program interruption takes place for a privileged-operation exception (1.A.1).
2. The word MICCREG, containing the address of the control block ECBLOK, which in turn contains the virtual control registers, is fetched with a key of zero. Execution of this function ends if an addressing condition is encountered (1.A.2).
3. If an access condition is encountered in fetching the second halfword of the STORE CONTROL instruction, execution of this function ends, and a program interruption takes place for the access exception encountered (1.B).
4. If the second-operand effective address does not specify a word boundary, execution of this function ends, and a program interruption takes place for a privileged-operation exception (2.A).
5. The virtual control-register values specified by the R_1 and R_3 fields of the instruction are fetched from the ECBLOK with a key of zero and stored with the PSW key in ascending

Field Name	Control Block	Address Type	Offset (Hex)	No. of Bytes	Contents
MICCREG	MICBLOK	Real	4	4	Address of ECBLOK
MICVPSW	MICBLOK	Real	8	4	Address of VMPSW[1]
EXTCR0	ECBLOK	Real	0	4	Virtual control register 0
VMPSW	VMBLOK	Real	0	2	Virtual PSW bits 0-15
Operand 2	–	Logical	–	1	New system mask

[1] A virtual interruption is pending if bit 0 of MICVPSW is one.

Figure 15. Fields Used in SET SYSTEM MASK

locations starting with the location specified in the second-operand address. Execution ends if an addressing condition is encountered in fetching control-register values. On some models, addressing condtions may be recognized for all 16 virtual control registers. If an access condition is encountered on the second operand, a program interruption takes place for the exception recognized (2.B).

If no exceptions are recognized, the execution of this function and the instruction are complete.

Figure 16 summarizes the fields used.

STORE THEN AND SYSTEM MASK

The STORE THEN AND SYSTEM MASK instruction is executed for a virtual machine if the virtual-machine assist is activated for System/370 instructions, unless (1) a virtual-machine exception is recognized, (2) the first operand or some pertinent VM/370 control field cannot be accessed, or (3) the PER mask or the DAT bit of the virtual PSW would be changed.

If the shadow-table-bypass assist is not installed, the store-then-AND-system-mask function of the virtual-machine assist is invoked each time a CPU attempts to execute a STORE THEN AND SYSTEM MASK instruction when the problem-state bit of the real PSW is one. If the shadow-table-bypass assist is installed, the store-then-AND-system-mask function of the virtual-machine assist may be invoked only from the STNSM function of the shadow-table-bypass assist.

Execution of this function consists in performing the following steps:

1. If bits 0-3 of control register 6 are not 10X0 binary, execution of this function ends, a program interruption takes place for a privileged-operation exception, and execution of the STORE THEN AND SYSTEM MASK instruction is suppressed (1.A.1).
2. The word MICVPSW, containing the address of the virtual PSW, is fetched with a key of zero. Execution ends if an addressing condition is encountered (1.A.2).
3. The first halfword of VMPSW, which contains the virtual PSW, is fetched with a key of zero. Execution ends if an addressing condition is encountered (1.A.3).
4. A new first byte for the virtual PSW is computed by taking the logical AND of the immediate field (I_2) of the instruction and byte 0 of the virtual PSW fetched. If the virtual PSW is in the EC mode (bit 12 is one) and if the replacement of byte 0 of the virtual PSW by the new byte-0 value would change bit 1 or bit 5 from one to zero, execution of this function ends. Ending of execution invokes the store-then-AND-system-mask function of the expanded virtual-machine assist if that assist is installed. Otherwise, ending of execution causes a program interruption to be taken for a privileged-operation exception (1.A.4).
5. The second halfword of the instruction is fetched. If an access condition is encountered, execution of this function ends, and a program

Field Name	Control Block	Address Type	Offset (Hex)	No. of Bytes	Contents
MICCREG	MICBLOK	Real	4	4	Address of ECBLOK (virtual CRs)
EXTCR0	ECBLOK	Real	0	4	Virtual CR0
EXTCR1	ECBLOK	Real	4	4	Virtual CR1
EXTCR1	ECBLOK	Real	4	4	Virtual CR1
EXTCR1	ECBLOK	Real	4	4	Virtual CR1
EXTCR1	ECBLOK	Real	4	4	Virtual CR1
EXTCR15	ECBLOK	Real	3C	4	Virtual CR15
Operand 2	—	Logical	0	4n[1]	

[1] n is the number of registers specified by the R_1, R_3 fields.

Figure 16. Fields Used in STORE CONTROL

interruption takes place for a privileged-operation exception (1.B.1).

6. If an access exception exists for a store access made with the PSW key to the location designated by the first-operand address, execution of this function ends, and a program interruption is taken for the access exception found (1.B.2).
7. The old value of byte 0 of the virtual PSW is stored at the first-operand logical address. The updated virtual PSW is stored as the new virtual PSW with a key of zero (2).

Execution of this function ends with completion of the STORE THEN AND SYSTEM MASK instruction.

Figure 17 summarizes the fields used.

STORE THEN OR SYSTEM MASK

The STORE THEN OR SYSTEM MASK instruction is executed for a virtual machine if the virtual-machine assist is activated for System/370 instructions, unless (1) a virtual-machine interruption may follow, (2) the first operand or some pertinent VM/370 control field cannot be accessed, or (3) the PER mask or the DAT bit of the virtual PSW would be changed.

If the shadow-table-bypass assist is not installed, the store-then-OR-system-mask function of the virtual-machine assist is invoked each time a CPU attempts to execute a STORE THEN OR SYSTEM MASK instruction when the problem-state bit of the real PSW is one. If the shadow-table-bypass assist is installed, the store-then-OR-system-mask function of the virtual-machine assist may be invoked only from the STOSM function of the shadow-table-bypass assist. Execution of this function consists in performing the following steps:

1. If bits 0-3 of control register 6 are not 10X0 binary, execution of this function ends, a program interruption takes place for a privileged-operation exception, and execution of the STORE THEN OR SYSTEM MASK instruction is suppressed (1.A.1).
2. The word MICVPSW, which contains the address of the virtual PSW, is fetched with a key of zero. Execution ends if an addressing condition is encountered (1.A.2).
3. The first halfword of VMPSW, which contains the virtual PSW, is fetched with a key of zero. Execution ends if an addressing condition is encountered (1.A.3).
4. A new first byte for the virtual PSW is computed by taking the logical OR of the immediate field (I_2) of the instruction and byte 0 of the virtual PSW fetched. If the virtual PSW is in the EC mode (bit 12 is one) and if the replacement of byte 0 of the virtual PSW by the new byte 0 would change any of the bits in bit positions 0-5 from zero to one, execution of this function ends. Execution also ends if bit 0 of MICVPSW is one and if the replacement of byte 0 of the virtual PSW by the new PSW would change any bit from zero to one. Ending of execution invokes the store-then-OR-system-mask function of the expanded virtual-machine assist if that assist is installed. Otherwise, ending of execution causes a program interruption to be taken for a privileged-operation exception (1.A.4).
5. The second halfword of the instruction is fetched. If an access condition is encountered, execution of this function ends, and a program interruption takes place for a privileged-operation exception (1.B.1).
6. If an access exception exists for a store access made with the PSW key to the location designated by the first-operand logical address, execution of this function ends, and a program interruption takes place for the access exception found (1.B.2).
7. The old value of byte 0 of the virtual PSW is stored at the first-operand logical address. The updated virtual PSW is stored with a key of zero as the new virtual PSW. Execution of this function ends with the completion of the STORE THEN OR SYSTEM MASK instruction (2).

Figure 18 summarizes the fields used.

Field Name	Control Block	Address Type	Offset (Hex)	No. of Bytes	Contents
MICVPSW	MICBLOK	Real	8	4	Address of VMPSW
VMPSW	VMBLOK	Real	0	2	Virtual PSW bits 0-15
Operand 1	–	Logical	–	1	Old system mask

Figure 17. Fields Used in STORE THEN AND SYSTEM MASK

Field Name	Control Block	Address Type	Offset (Hex)	No. of Bytes	Contents
MICVPSW	MICBLOK	Real	8	4	Address of VMPSW
VMPSW	VMBLOK	Real	0	2	Virtual PSW bits 0-15
Operand 2	–	Logical	–	1	Old system mask

Figure 18. Fields Used in STORE THEN OR SYSTEM MASK

SUPERVISOR CALL

The SUPERVISOR CALL instruction is executed for a virtual machine if the virtual-machine assist is activated for SVC interruptions, unless (1) a virtual-machine interruption may follow; (2) some pertinent VM/370 control field or the low-storage locations of the virtual machine cannot be accessed; (3) the PER mask is one in the real PSW or in the old or the new virtual PSW; (4) execution would change the control mode, the DAT bit, or the wait-state bit of the virtual PSW; or (5) the SVC interruption code is 76 hex.

The supervisor-call function of the virtual-machine assist is invoked each time a CPU attempts to execute a SUPERVISOR CALL instruction when the problem-state bit of the real PSW is one. Execution of this function consists in performing the following steps:

1. If bits 0-4 of control register 6 are not 1XXX0 binary, execution of the supervisor-call function ends, and a supervisor-call interruption takes place in the real machine in the normal manner (1).
2. If the real PSW is in EC mode with the PER mask set to one, execution ends, and a real supervisor-call interruption takes place (2.A).
3. The word MICVPSW, which contains the address of the virtual PSW, is fetched with a key of zero. Execution ends if an addressing condition is encountered (2.B.1).
4. VMPSW, which contains the current virtual PSW, is fetched with a key of zero. Execution ends if an addressing condition is encountered (2.B.2).
5. If the current virtual PSW is in EC mode with the PER mask set to one, execution ends, and a real supervisor-call interruption takes place (2.B.3).

   The virtual machine's real address 0 is translated through the segment tables addressed by the first word in the virtual-machine-parameter list. The page-table entry is interpreted in the format of a 4K-byte or 2K-byte page-table entry, depending on whether bit 30 of MICRSEG is zero or one, respectively. If any of the following conditions is encountered, a real supervisor-call interruption takes place:
6. An addressing condition is encountered in fetching MICRSEG, which contains the address of the real segment table, with a key of zero (2.C.1).
7. An addressing condition is encountered in fetching SEGPAGE, which contains the first real-segment table entry, with a key of zero (2.C.2).
8. The segment-table entry is invalid (2.C.3).
9. The segment-table entry has an invalid format (2.C.4).
10. An addressing condition is encountered in fetching PAGCORE, which contains the first page-table entry of the first segment, with a key of zero (2.C.5).
11. The page-table entry is invalid (2.C.6).
12. The page-table entry has an invalid format (2.C.7).
13. The new virtual PSW is fetched, with a key of zero, from location 60 hex in the first virtual page. Execution ends if an addressing condition is encountered (2.C.8).
14. If (1) the wait-state bit of the new virtual PSW is one or (2) that PSW is in the EC mode with the PER mask set to one or with an invalid format, execution ends, and a real supervisor-call interruption takes place (2.C.9.A).
15. Execution of the supervisor-call function ends if any of the following conditions holds (2.C.9.B):
    a. The control mode of the virtual PSW is being changed from the BC to the EC mode, or from the EC mode to the BC mode.
    b. The DAT mode bit of an EC-mode virtual PSW is being changed.
    c. A virtual interruption is pending and any channel mask, input/output mask, or external mask is being changed from zero to one (bits 0-7 in the BC mode and bits 6-7 in the EC mode). A virtual interruption is pending when bit 0 of MICVPSW is one.

Note: *Because all the preceding conditions involve the value fetched in step 4, the priority of step 15 (2.C.9.B) is necessarily lower than that of step 4, regardless of the rules based on priority indicators.*

If execution ends, a real supervisor-call interruption takes place.

16. If the supervisor-call interruption code is 76 (4C hex), execution ends, and a real supervisor-call interruption takes place (2.D).
17. The old PSW and interruption code are stored, with a key of zero, in 20 hex and 88 hex of virtual page 0 as appropriate to the mode of the current virtual PSW. The stored values are the current virtual PSW, ILC, and SVC-number values updated by the condition code, program mask, and instruction address from the real PSW. The new virtual PSW is stored as the current virtual PSW in real storage. The program-~~mask, key,~~ condition-code, and instruction-address parts of the new virtual PSW replace the corresponding values in the real PSW. The problem-state bit of the new virtual PSW is placed in bit position 1 of ~~real~~ control register 6. Execution of this function ends, and a new virtual instruction is fetched with the new real PSW (3).

Figure 19 summarizes the fields used.

Field Name	Control Block	Address Type	Offset (Hex)	No. of Bytes	Contents
MICRSEG	MICBLOK	Real	0	4	Address of real segment table
MICVPSW	MICBLOK	Real	8	4	Address of VMPSW
VMPSW	VMBLOK	Real	0	8	Virtual PSW
SEGPAGE	SEG TABLE	Real	0	4	First real segment table entry
PAGCORE	PAG TABLE	Real	0	2	Address of PSA of virtual machine
–	PSA of virtual machine	Real[1]	20	8	Old SVC PSW
–		Real[1]	60	8	New SVC PSW
–		Real[1]	88	4	Interruption code[2]

[1] This real address is obtained by the address translation performed in steps 6 through 12 without using control registers 0 and 1.

[2] This field is stored only when old SVC PSW is in EC mode.

Figure 19. Fields Used in SUPERVISOR CALL

Shadow-Table Validation

When the real page-table entry used for dynamic address translation is invalid and the shadow-table-validation function is active, the correct, valid entry value derived from the virtual and real translation tables is placed in that page-table entry. However, this validation function is not performed if any exception condition is found in fetching or using the corresponding real or virtual translation-table entries.

If the shadow-table-bypass assist is not installed, the shadow-table-validation function of the virtual-machine assist is invoked whenever a program interruption is about to take place for a page-translation condition encountered outside the shadow-table-validation function itself. If the shadow-table-bypass assist is installed, the shadow-table-validation function may be invoked only from the page-fault-reflection function of the shadow-table-bypass assist. If this function successfully validates the shadow-table page entry which caused the page-translation condition, instruction execution that was in progress is resumed or restarted. Otherwise, a program interruption takes place for the original page-translation condition or, in some cases, for an addressing condition encountered in performing the shadow-table-validation function. The execution of

SL 1500 0.00

this function consists in performing the following steps:

1. On some models, the segment index and the page index of the untranslatable address are placed in the word at real location 90 hex. If bits 0-5 of control register 6 are not 1XXXX1 binary or if the real PSW is in the EC mode with a PER-mask-bit value of one, execution of this function ends, and a program interruption takes place for the page-translation condition (1).

s/b XX

2. The doubleword which contains the fields MICRSEG and MICCREG is fetched with a key of zero. MICRSEG contains the address of the real segment table, and MICCREG contains the address of the extended control block (ECBLOK). Execution ends if an addressing condition is encountered (2.A.1).
3. EXTCR0 and EXTCR1, virtual control registers 0 and 1, are fetched from the ECBLOK with a key of zero. Execution ends if an addressing condition is encountered (2.A.2).
4. If the translation format in bits 8-12 of virtual control register 0 is invalid, execution of this function ends, and a program interruption takes place for the page-translation condition (2.A.3).
5. Execution ends with a program interruption for the page-translation condition if virtual control register 0 specifies a 64K-byte segment size and if the segment-table-length value in bit positions 0-7 of virtual control register 1 is less than the value obtained by appending four zeros to the left of bits 8-11 of the untranslatable address (2.A.4).
6. The untranslatable address is divided into a virtual-segment index, a virtual-page index, and a virtual-byte index, based on the translation format in virtual control register 0. The virtual-segment-table-entry address is computed by using the contents of virtual control register 1 and the virtual segment index. The virtual-segment-table-entry address is in turn divided into a real-segment index, a real-page index, and a real-byte index, assuming either 4K-byte or 2K-byte pages, depending on whether bit 30 of MICRSEG is zero or one, and either 64K-byte or 1M-byte segments, depending on whether bit 31 of MICRSEG is zero or one, respectively. If bit 31 of MICRSEG is zero and the real segment-table-length value in bit positions 0-7 of MICRSEG is less than the value obtained by appending four zeros to the left of bits 8-11 of the virtual-segment-table-entry address, execution of this function ends, and a program interruption takes place for the page-translation condition (2.A.5).
7. Four times the real segment index found in step 6 is added to the real segment-table address fetched in step 2 to obtain the address of the real segment-table entry SEGPAGE for translating the virtual segment-table-entry address. This real segment-table entry is fetched with a key of zero. Execution ends if an addressing condition is encountered (2.A.6).
8. If the real segment-table entry is invalid or has an invalid format, or if the value of the leftmost four bits of the real page index of the virtual segment-table entry address exceeds the page-table length in bits 0-3 of the real segment-table entry, execution of this function ends, and a program interruption takes place for the page-translation condition (2.A.7).
9. Twice the real page index found in step 6 is added to the real page-table origin found in the real segment-table entry fetched in step 7 to obtain the address of the real page-table entry PAGCORE for translating the virtual segment-table-entry address. This real page-table entry is fetched with a key of zero. Execution ends if an addressing condition is encountered (2.A.8).
10. If the real page-table entry is invalid or has an invalid format, execution of this function ends, and a program interruption takes place for the page-translation condition (2.A.9).
11. The real byte index found in step 6 combined with the page-frame real address found in the real page-table entry fetched in step 9 obtain the real address of the virtual segment-table entry. The virtual segment-table entry is fetched with a key of zero. Execution ends if an addressing condition is encountered (2.A.10).
12. If the virtual segment-table entry is invalid or has an invalid format, or if the value of the leftmost four bits of the virtual page index of the address that could not be translated exceeds the page-table length in bits 0-3 of the virtual segment-table entry, execution of this function ends, and a program interruption takes place for the page-translation condition (2.A.11).
13. The virtual-page-table-entry address is computed by using the contents of the virtual segment-table entry and the virtual page index of the untranslatable address. The virtual-page-table-entry address is in turn divided into a real segment index, a real page index, and a real byte index assuming either 4K-byte pages

(bit 30 of MICRSEG is zero) or 2K-byte pages (bit 30 is one), and 64K-byte segments (bit 31 of MICRSEG is zero) or 1M-byte segments (bit 31 is one). If bit 30 of MICRSEG is zero and the real segment-table-length value in bit positions 0-7 of MICRSEG is less than the value obtained by appending four zeros to the left of bits 8-11 of the virtual-page-table-entry address, execution of this function ends, and a program interruption takes place for the page-translation condition (2.A.12).

14. Four times the real segment index found in step 13 is added to the real segment-table address fetched in step 2 to obtain the address of the real segment-table entry for translating the virtual page-table-entry address. This real segment-table entry SEGPAGE is fetched with a key of zero. Execution ends if an addressing condition is encountered (2.A.13).
15. If the real segment-table entry is invalid or has an invalid format, or if the value of the leftmost four bits of the real page index of the virtual page-table-entry address exceeds the page-table length in bits 0-3 of the real segment-table entry, execution of this function ends, and a program interruption takes place for the page-translation condition (2.A.14).
16. Twice the real page index found in step 13 is added to the real page-table origin found in the real segment-table entry fetched in step 14 to obtain the address of the real page-table entry for translating the virtual page-table-entry address. This real page-table entry PAGCORE is fetched with a key of zero. Execution ends if an addressing condition is encountered (2.A.15).
17. If the real page-table entry fetched is invalid or has an invalid format, execution of this function ends, and a program interruption takes place for the page-translation condition (2.A.16).
18. The real byte index found in step 13 is combined with the page-frame real address found in the page-table entry fetched in step 16 to obtain the real address of the virtual page-table entry. The virtual page-table entry is fetched with a key of zero. Execution ends if an addressing condition is encountered (2.A.17).
19. If the virtual page-table entry is invalid or has an invalid format, execution of this function ends, and a program interruption takes place for the page-translation condition (2.A.18).
20. The virtual address corresponding to the address that could not be translated is computed by using the contents of the virtual page-table entry and the virtual byte index of the address that could not be translated. This address is divided into a real segment index, a real page index, and a real byte index, assuming either 4K-byte pages (bit 30 of MICRSEG is zero) or 2K-byte pages (bit 30 of MICRSEG is one), and either 64K-byte segments (bit 31 of MICRSEG is zero) or 1M-byte segments (bit 31 of MICRSEG is one). If bit 30 of MICRSEG is zero and the real segment-table-length value in bit positions 0-7 of MICRSEG is less than the value obtained by appending four zeros to the left of bits 8-11 of the virtual address, execution of this function ends and a program interruption takes place for the page-translation condition (2.A.19).
21. Four times the real segment index found in step 20 is added to the real segment-table address from MICRSEG to obtain the address of the real segment-table entry for translating the virtual address. This real segment-table entry SEGPAGE is fetched with a key of zero. Execution ends if an addressing condition is encountered (2.A.20).
22. If the real segment-table entry is invalid or has an invalid format, or if the value of the leftmost four bits of the page index of the virtual address exceeds the page-table length in bits 0-3 of the segment-table entry, execution of this function ends, and a program interruption takes place for the page-translation condition (2.A.21).
23. Twice the real page index found in step 20 is added to the real page-table origin found in the real segment-table entry fetched in step 21 to obtain the address of the real page-table entry for translating the virtual address. This real page-table entry PAGCORE is fetched with a key of zero. Execution ends if an addressing condition is encountered (2.A.22).
24. If the real page-table entry fetched is invalid or has an invalid format, execution of this function ends, and a program interruption takes place for the page-translation condition (2.A.23).
25. The address that could not originally be translated because a page-translation condition was encountered is divided into a a shadow-table segment index, shadow-table page index, and a shadow-table byte index based on the translation format in bits 8-12 of real control register 0. The real address of the shadow segment-table entry used for translating the original address is computed by using the

shadow-table segment index and the shadow segment-table address from real control register 1. The shadow segment-table entry SEGPAGE is fetched with a key of zero. Execution ends ~~if an addressing condition is encountered~~ (2.B.1).

26. If the shadow segment-table entry is invalid or has an invalid format, or if the value of the leftmost four bits of the shadow-table page index obtained in step 25 exceeds the value of bits 0-3 of the shadow segment-table entry, execution of this function ends, and a program interruption takes place for the page-translation condition (2.B.2).
27. Twice the shadow-table page index found in step 25 is added to the page-table origin found in the shadow segment-table entry fetched in that step to compute the real address of the shadow page-table entry PAGCORE to be validated (3).

The real byte index found in step 20 combined with the page-frame real address found in the page-table entry fetched in step 23 to obtain the real address corresponding to the address that originally was untranslatable. A valid halfword shadow-table entry is formed by placing bits 8-19 or 8-20 of the real address obtained by translation in bit positions 0-11 or 0-12, depending on whether ~~the shadow-table page size indicated by real control~~ register 0 is 4K or 2K bytes, respectively, in size. Zeros are placed in the remaining rightmost bit positions. The valid shadow-table entry is stored with a key of zero at the address computed at the start of this step. Execution ends if an addressing condition is encountered. Note that an addressing condition cannot arise unless the shadow-table segment-table entry was modified since the original page-translation condition was found.

28. Execution of this function ends, and execution of the noninterruptible instruction or the unit of operation of the interruptible instruction which was being executed when the original page-translation condition was encountered is resumed or restarted. On some models, any pending I/O, external, or restart interruptions may take place before an instruction is executed (4).

Figure 20 summarizes the fields used.

Field Name	Control Block	Address Type	Offset (Hex)	No. of Bytes	Contents
MICRSEG	MICBLOK	Real	0	4	Address of real segment table
MICCREG	MICBLOK	Real	4	4	Address of ECBLOK
EXTCR0	ECBLOK	Real	0	4	Virtual CR0
EXTCR1	ECBLOK	Real	4	4	Virtual CR1
SEGPAGE	SEGTABLE	Real	$4SX^1$	4	Entries to find virtual segment entry
PAGCORE	PAGTABLE	Real	$2PX^1$	2	Entries to find virtual segment entry
–	Virtual segment entry	Real	–	4	
SEGPAGE	SEGTABLE	Real	$4SX^2$	4	Entries to find virtual page entry
PAGCORE	PAGTABLE	Real	$2PX^2$	2	Entries to find virtual page entry
–	Virtual page entry	Real	–	2	
SEGPAGE	SEGTABLE	Real	$4SX^3$	4	Entries to get real translated address
PAGCORE	PAGTABLE	Real	$2PX^3$	2	Entries to get real translated address
SEGPAGE	Shadow SEGTABLE	Real	$4SX^4$	4	Shadow segment-table entry
PAGCORE	Shadow PAGTABLE	Real	$2PX^4$	2	Validated shadow page-table entry
Superscripts 1, 2, 3, and 4 refer, respectively, to the segment and page indexes to translate (1) the virtual segment-table address, (2) the virtual page-table address, (3) the virtual address corresponding to the untranslated virtual/virtual address, and (4) untranslated logical address of the virtual machine.					

Figure 20. Fields Used in Shadow-Table Validation

Chapter 3. VM-Common-Segment Modification of VMA

Use of the virtual-machine assist with the VM-common-segment modification by the VM/370 System Extensions program product (Program No. 5748-XE1) improves the performance of virtual machines which use the common-segment-bit function of the System/370 extended facility.

The VM-common-segment modification alters the virtual-machine assist so that checking for a zero in the common-segment bit position (bit position 30) of segment-table entries is omitted in virtual-machine-assist functions:

- For all references to real segment-table entries
- For all references by virtual segment-table entries
- For reference by the shadow-table-validation function to a shadow segment-table entry for the purpose of locating the shadow page-table entry to be validated.

The VM-common-segment modification of VMA has no effect on the checking of shadow segment-table entries when they are used for translating instruction or operand addresses. Figure 21 shows the segment-table entries that are checked in bit position 30 for each type of reference for each completely assisted virtual-machine-assist function.

Programming Note

If the common-segment-bit function is used only by the virtual machine and not in the real or shadow segment tables, the only effects of installing the VM-common-segment modification of VMA are:

1. The load-real-address function is completed without a program interruption when a virtual common segment is referred to.
2. The shadow-table validation can be completed without a program interruption when a virtual common segment is involved.

VMA Function	Real Segment-Table Entry	Virtual Segment-Table Entry	Shadow Segment-Table Entry Used for Finding:		
			PTE to Be Validated	Instruction	Operand
IPK, SPKA	–	–	–	X	–
ISK, RRB SSK, SVC	Y1	–	–	X	–
LRA	Y2	Y1	–	X	–
LPSW, SSM STCTL, STNSM STOSM	–	–	–	X	X
Shadow-table validation	Y3	Y1	Y1	–	–

Explanation:

–	No reference of this type.
X	Reference is made and bit 30 is ignored if System/370 extended facility is installed.
Y1,Y2,Y3	One, two, or three references of this type are made, and bit 30 is ignored if the VM-common-segment modification is installed.

Figure 21. Segment-Table Entries Checked in Bit Position 30

Chapter 4. Shadow-Table-Bypass Assist

The shadow-table-bypass assist enhances the performance of virtual machines for which the virtual = real option is specified in Virtual Machine Facility/370 (VM/370) program products.

VM/370 normally employs shadow tables for virtual machines operating in the EC mode with DAT on. Shadow tables are segment tables and page tables used by the dynamic-address-translation (DAT) facility of the real machine for direct translation of logical addresses of the virtual machine to real addresses of the real machine. For machines with the virtual=real option, however, most addresses translated through the shadow tables give the same result as if they were translated only through the virtual-machine translation tables. Two specific techniques are used in different program products to exploit this characteristic of the shadow tables of machines having the virtual = real option.

1. In the true shadow-table-bypass technique, the virtual-machine segment and page tables are used directly by the real-machine DAT mechanism. Hence, no distinct shadow tables exist, and the virtual-machine-segment table origin and length are used wherever the general VM/370 design calls for the values of the shadow segment-table origin and length. However, because the VM/370 does not map the first 4K bytes of the virtual-machine storage to the first 4K bytes of real storage, this technique requires that those page-table entries of the virtual machine which refer to the first 4K bytes be modified by VM/370 so that this remapping can be accomplished without separate shadow tables.
2. In the single-processor-mode technique used for a virtual = real virtual machine running MVS, 253 of the 256 page tables of the virtual machine are used directly, and only three of the page tables of the virtual machine have distinct counterpart shadow page tables. The shadowed tables are the tables that contain some entry having a real page-frame address for which prefixing or reverse prefixing of the virtual machine applies. Thus, single-processor mode is only a partial bypass of the shadow tables. A separate shadow segment table is maintained, 253 of whose entries refer directly to virtual page tables in the storage of the virtual machine.

The shadow-table-bypass assist improves performance of certain virtual machines having the virtual = real option by executing seven specific virtual-machine instructions and one type of virtual-machine program interruption directly without requiring any intervention or assistance by the VM/370 control program.

The assisted instructions are:

INVALIDATE PAGE TABLE ENTRY
LOAD CONTROL
LOAD REAL ADDRESS
PURGE TLB
STORE THEN AND SYSTEM MASK
STORE THEN OR SYSTEM MASK
TEST PROTECTION

The virtual-machine program interruption is for page-fault reflection and consists in taking a program interruption for a page-translation condition directly in the virtual machine.

Relation of Shadow-Table-Bypass Assist to Other Assists

The shadow-table-bypass assist (STBA) is related to the virtual-machine assist (VMA) and to the expanded virtual-machine assist (EVMA) which is part of ECPS:VM/370. All three assists use bits 0, 1, 3, 5, and 8-28 of control register 6 as follows:

Bit	*Meaning*
0	When the bit is zero, assists are inactive; when the bit is one, VMA is active; when it is one and bit 6 of control register 6 is one, EVMA is active; when it is one and bit 8 of the assist control word is one, STBA is active.
1	Virtual-machine problem-state bit.
3	When the bit is zero, only operation codes for System/370 are assisted.
5	When the bit is one, shadow-table validation is active; when it is zero, the page-fault-reflection function is performed if bits 8 and 11 of the assist control word are both ones.
8-28	Bits 8-28 are bits 8-28 of the address of the virtual-machine parameter list (MICBLOK) aligned on a doubleword boundary.

In addition, these assists use the following words of the virtual-machine parameter list:

Offset (Hex)	*Field Symbol*	*Use*
0	MICRSEG	Real segment-table address
4	MICCREG	Address of ECBLOK
8	MICVPSW	Address of virtual PSW bits 0-15
14	MICACF	Assist control word

None of the eight functions of the shadow-table-bypass assist is active unless (1) bit 0 of control register 6 is one, (2) bit 8 of the assist control word is one, and (3) a specific bit of bits 9-15 of the assist control word is one. The specific activation bit depends on the function. (See Figure 22.) Six bits are used for eight functions. Each bit

CR6.∅ VMA Control
1 Virtual Problem State
2 ISK/SSK Inhibit
3 360 Operations
4 SVC Inhibit
5 Shadow Table Validation [illegible]
6 [illegible]MA/CPA Control
7 VITA Control
29 VM EF Control
30 PMA Event Filter
[illegible] 3033 UP "SPX Assist"

ACF2.∅ STBYPR MODE ASSIST ACTIVE
1 PTLB CONTROL
2 IPTE TPRT CONTROL
3 VIRTUAL PAGE FAULT REFLECTION
4 LRA CONTROL
5 FULL SPT CONTROL
6 STNSM, STOSM CONTROL
7 LCTL CONTROL

Function	Bits 0-5 of Control Register 6	Bits 8-15 of Assist Control Word
Invalidate page table entry	10X0 XX	1X1X XXXX
Load control	10X0 XX	1XXX XXX1
Load real address	10X0 XX	1XXX 1XXX
Purge TLB	10X0 XX	11XX XXXX
Store then AND system mask	10X0 XX	1XXX XX1X
Store then OR system mask	10X0 XX	1XXX XX1X
Test protection	10X0 XX	1X1X XXXX
Page-fault reflection	1XXX X0	1XX1 XXXX

[illegible]

Figure 22. Bits That Activate STBA Functions

activates a single function, except that bit 10 controls INVALIDATE PAGE TABLE ENTRY and TEST PROTECTION and bit 14 controls STORE THEN AND SYSTEM MASK and STORE THEN OR SYSTEM MASK. Additional conditions must be satisfied to activate each function. The additional conditions are given in the descriptions of the individual assist functions.

The shadow-table-bypass assist is logically independent of the virtual-machine assist. For the most part, the two assists complement each other. However, when the load-real-address function of the STBA is installed and active, and the VMA is also installed, the load-real-address function of STBA overrides that of the VMA. The expanded virtual-machine assist is effective when the other two assists, though active, are not applicable.

INVALIDATE PAGE TABLE ENTRY

The INVALIDATE PAGE TABLE ENTRY instruction is executed for a virtual machine if the corresponding function of the shadow-table-bypass assist is activated, unless (1) the virtual machine is not in the EC mode with DAT on and the problem-state bit zero, (2) some pertinent VM/370 control field cannot be fetched, or (3) the entry to be invalidated is in the first 4K-byte locations in virtual storage.

The invalidate-page-table-entry (IPTE) function of the shadow-table-bypass assist is invoked each time a CPU attempts to execute an INVALIDATE PAGE TABLE ENTRY instruction when the problem-state bit of the real CPU is zero. Execution of this function consists in performing the following steps:

1. If bits 0-3 of control register 6 are not 10X0 binary, a program interruption takes place for a privileged-operation exception, and execution of this IPTE instruction is suppressed (1.A.1).
2. The assist control word, MICACF, is fetched with a key of zero. Execution ends if an addressing condition is encountered (1.A.2).
3. Execution ends with a program interruption for a privileged-operation exception if bits 8 and 10 of the assist control word are not both ones (1.A.3).
4. MICVPSW, which contains the address of the virtual PSW, is fetched with a key of zero. Execution ends if an addressing condition is encountered (1.A.4)
5. VMPSW, the virtual PSW, is fetched with a key of zero. Execution ends if an addressing condition is encountered (1.A.5).
6. Execution ends with a program interruption for a privileged-operation exception if bits 5 and 12 of the virtual PSW are not both ones (that is, if the virtual machine is not in the EC mode with DAT on) (1.A.6).
7. If an access condition is encountered in fetching the second halfword of the IPTE instruction, execution of this function ends, and a program interruption takes place for the access exception encountered (1.B).
8. The address of the page-table entry is computed by using the page-table-origin value from the general register specified by the R_1 field and the page index from the address in the general register specified by the R_2 field. Bits 8-12 of real control register 0 determine the bit positions that contain the page-index value. A program interruption takes place for a privileged-operation exception if the computed

page-table address is less than 4096. On some models, this interruption also takes place when the page-table origin is less than 4096 (2).
9. This function invalidates a page-table entry just as if the INVALIDATE PAGE TABLE ENTRY instruction were executed with the problem-state bit set to zero. In particular, certain entries in the translation-lookaside buffer of all configured CPUs must be purged. If a protection or addressing exception is encountered, a program interruption for protection or addressing takes place in the normal manner (3).

Figure 23 summarizes the fields used.

Programming Notes
1. IPTE is not assisted if the page-table entry is in the first 4K bytes of real storage because VM/370 does not map that storage as virtual equals real.
2. A translation-specification exception for a format error in bits 8-12 of real control register 0 cannot arise in executing IPTE under VM/370 because VM/370 always executes virtual-machine instructions with real DAT on, and a format error would prevent the fetching of any virtual-machine instruction.

LOAD CONTROL

The LOAD CONTROL instruction is executed for a virtual machine if the corresponding function of the shadow-table-bypass assist is activated; the instruction loads virtual control register 1 only when the virtual machine is in the EC mode with DAT on.

The load-control function of the shadow-table-bypass assist is invoked each time a CPU attempts to execute a LOAD CONTROL instruction when the problem-state bit of the real PSW is one. The execution of this function consists in performing the following steps:
1. If bits 0-3 of control register 6 are not 10X0 binary, a program interruption takes place for a privileged-operation exception, and execution of the LOAD CONTROL instruction is suppressed (1.A.1).
2. The assist control word, MICACF, is fetched with a key of zero. Execution ends if an addressing condition is encountered (1.A.2.A.1).
3. Execution ends with a program interruption for a privileged-operation exception if bits 8 and 15 of the assist control word are not both ones (1.A.2.A.2).
4. MICVPSW, which contains the address of the virtual PSW, is fetched with a key of zero. Execution ends if an addressing condition is encountered (1.A.2.A.3).
5. VMPSW, the virtual PSW, is fetched with a key of zero. Execution ends if an addressing condition is encountered (1.A.2.A.4).
6. Execution ends with a program interruption for a privileged-operation exception if bits 5 and 12 of the virtual PSW are not both ones (that is, if the virtual machine is not in the EC mode with DAT on) (1.A.2.A.5).
7. Execution ends with a program interruption for a privileged-operation exception if the R_1 and R_3 fields of the LOAD CONTROL instruction are not both one (hex) (1.A.2.B).
8. If an access condition is encountered in fetching the second halfword of the LOAD CONTROL instruction, execution of this function ends, and a program interruption takes place with that access exception indicated (1.B).
9. The LOAD CONTROL instruction is executed just as if the real problem-state bit were zero. If an exception is recognized, execution of this function ends, and a program interruption takes place indicating that exception (2).
10. If the value in real control register 1 was not changed, execution of the LOAD CONTROL instruction is complete (3).
11. MICCREG, the word containing the address of the ECBLOK control block, is fetched with a key of zero. Execution of the LOAD CONTROL instruction is terminated, and a

Field Name	Control Block	Address Type	Offset (Hex)	No. of Bytes	Contents
MICVPSW	MICBLOK	Real	8	4	Address of VMPSW
MICACF	MICBLOK	Real	14	4	Assist control word
VMPSW	VMBLOK	Real	0	2	Virtual PSW bits 0-15
Operand 2	–	Real	–	2	Page-table entry

Figure 23. Fields Used in INVALIDATE PAGE TABLE ENTRY

program interruption takes place if an addressing exception is recognized (4.A.1).

12. The value in real control register 1 is stored, with a key of zero, in virtual control register 1, EXTCR1. Execution of the LOAD CONTROL instruction is terminated, and a program interruption takes place if an addressing exception is recognized (4.A.2.A).
13. The value in real control register 1 is stored, with a key of zero, in shadow control register 1, EXTSHCR1. Execution of the LOAD CONTROL instruction is terminated, and a program interruption takes place if an addressing exception is recognized (4.A.2.B).
14. The value in real control register 1 is stored, with a key of zero, in RUNCR1, in the real PSA (4.B).

Figure 24 summarizes the fields used.

LOAD REAL ADDRESS

The LOAD REAL ADDRESS instruction is executed for a virtual machine if the corresponding function of the shadow-table-bypass assist is activated, unless (1) a virtual-machine-exception condition is recognized, (2) some pertinent VM/370 control field cannot be fetched, or (3) the virtual machine is not in the EC mode with DAT on.

The load-real-address (LRA) function of the shadow-table-bypass assist is invoked each time a CPU attempts to execute a LOAD REAL ADDRESS instruction when the problem-state bit of the real PSW is one. Execution of this function consists in performing the following steps:

1. If bits 0-3 of control register 6 are not 10X0 binary, a program interruption takes place for a privileged-operation exception, and execution of the LRA instruction is suppressed (1.A.1).
2. The assist control word, MICACF, is fetched with a key of zero. Execution ends if an addressing condition is encountered (1.A.2).
3. Execution ends with a program interruption for a privileged-operation exception if bits 8 and 12 of the assist control word are not both ones. If execution of this function is ended, the load-real-address function of the virtual-machine assist is invoked when the virtual-machine assist is installed. When the virtual-machine assist is not installed, the ending of execution of this function results in a program interruption for a privileged-operation exception (1.A.3).
4. MICVPSW, which contains the address of the virtual PSW, is fetched with a key of zero. Execution ends if an addressing condition is encountered (1.A.4).
5. VMPSW, the virtual PSW, is fetched with a key of zero. Execution ends if an addressing condition is encountered (1.A.5).
6. Execution ends with a program interruption for a privileged-operation exception if bits 5 and 12 of the virtual PSW are not both ones (that is, if the virtual machine is not in the EC mode with DAT on) (1.A.6).
7. If an access condition is encountered in fetching the second halfword of the LRA instruction, execution of this function ends, and a program interruption takes place for that access exception (1.B).

Field Name	Control Block	Address Type	Offset (Hex)	No. of Bytes	Contents
MICCREG	MICBLOK	Real	4	4	Address of ECBLOK
MICVPSW	MICBLOK	Real	8	4	Address of VMPSW
MICACF	MICBLOK	Real	14	4	Assist control word
EXTCR1	ECBLOK	Real	4	4	Virtual control register 1
EXTSHCR1	ECBLOK	Real	44	4	Shadow control register 1
VMPSW	VMBLOK	Real	0	2	Virtual PSW bits 0-15
RUNCR1	PSA	Real	344	4	Control register 1 at dispatch
Operand	–	Logical	–	4	New control-register-1 value

Figure 24. Fields Used in LOAD CONTROL

8. The effective address is translated, and a condition-code value is derived just as if LRA were being executed with the real problem-state bit set to zero. If a translation specification or addressing condition is encountered, execution of this function ends with the occurrence of a program interruption for the exception encountered; otherwise, the translated address is placed in the general register specified by the R_1 field of the instruction, and the condition code is set. Execution of this function then ends, with execution of LRA completed (2).

Figure 25 summarizes the fields used.

PURGE TLB

The PURGE TLB instruction is executed for a virtual machine if the corresponding function of the shadow-table-bypass assist is active, unless (1) a virtual-machine-exception condition is recognized or (2) some pertinent VM/370 control field cannot be accessed.

The purge-TLB (PTLB) function of the shadow-table-bypass assist is invoked each time a CPU attempts to execute a PURGE TLB instruction when the problem-state bit of the real PSW is one. Execution of this function consists in performing the following steps:

1. If bits 0-3 of control register 6 are not 10X0 binary, a program interruption takes place for a privileged-operation exception, and execution of the PTLB instruction is suppressed (1.A.1).
2. The assist control word, MICACF, is fetched with a key of zero. Execution ends if an addressing condition is encountered (1.A.2).
3. Execution ends if bits 8 and 9 of the assist control word are not both ones. If execution of this function is ended, the purge-TLB function of the expanded virtual-machine assist is invoked when that assist is installed. When that assist is not installed, the ending of execution of this function results in a program interruption for a privileged-operation exception (1.A.3).
4. If an access condition is encountered in fetching the second halfword of the PTLB instruction, it is unpredictable whether this condition is ignored, because no information is needed from that halfword to execute the instruction. If an access condition is encountered and is not ignored, execution of this function ends, and a program interruption takes place for that access exception (1.B).

Steps 5 through 7 may or may not be performed on a CPU which is not configured in a two-CPU configuration.

5. The APSTAT1 byte is fetched with a key of zero from real location 69A hex. Execution ends if an addressing condition is encountered (2).
6. The CPPTLB bit (bit 6) of APSTAT2 in the PSA of the CPU executing the purge-TLB function is set to zero (3).
7. If the APUOPER bit (bit 0) of APSTAT1 is zero, indicating that no attached processor is operational, this step is complete. Otherwise, the PREFIXB word is fetched with a key of zero from real location 664 hex. The real address of the APSTAT2 byte in the PSA of the other CPU is computed by adding 69B hex, right-justified, to the PREFIXB word. The CPPTLBR bit (bit 6) of the byte whose real address was just computed is set to one with a key of zero for the storage-access update. Execution ends if an addressing condition is encountered (4).
8. The TLB of the CPU executing the function is purged (5).

Figure 26 summarizes the fields used.

Field Name	Control Block	Address Type	Offset (Hex)	No. of Bytes	Contents
MICVPSW	MICBLOK	Real	8	4	Address of VMPSW
MICACF	MICBLOK	Real	14	4	Assist control word
VMPSW	VMBLOK	Real	0	2	Virtual PSW bits 0-15
Operand 2	1	Virtual	0	1	1
1 No operand reference to storage is made; however, the translation-table references at real addresses are made as if an operand reference were going to be made.					

Figure 25. Fields Used in LOAD REAL ADDRESS

Field Name	Control Block	Address Type	Offset (Hex)	No. of Bytes	Contents
MICACF	MICBLOK	Real	14	4	Assist control word
PREFIXB	PSA	Real	664	4	Real address of PSA of other CPU
APSTAT1	PSA	Real	69A	1	Bit 0 set to one indicates attached processor operational
APSTAT2	PSA	Real	69B	1	Bit 6 set to one indicates PURGE TLB is requested on this CPU
APSTAT2	PSA of Other CPU	Real	69B	1	Bit 6 set to one indicates PURGE TLB is requested on the other CPU

Figure 26. Fields Used in PURGE TLB

Programming Note
When the CPPTLBR bit in the PSA of a CPU is set to one, the translation-lookaside buffer of that CPU should be purged before the VM/370 control program dispatches a different virtual machine on that CPU. This is to ensure that if a virtual machine that executed a PURGE TLB is later redispatched on that CPU, any TLB entries left in that CPU will have been purged.

STORE THEN AND SYSTEM MASK

The STORE THEN AND SYSTEM MASK instruction is executed for a virtual machine if the corresponding function of the shadow-table-bypass assist is activated and the instruction uses an I_2 field value of FB hex to turn off the DAT bit in the virtual PSW.

The store-then-AND-system-mask (STNSM) function of the shadow-table-bypass assist is invoked each time a CPU attempts to execute a STORE THEN AND SYSTEM MASK instruction when the problem-state bit of the real PSW is one. Execution of this function consists in performing the following steps:

1. If bits 0-3 of control register 6 are not 10X0 binary, a program interruption takes place for a privileged-operation exception, and execution of the STNSM instruction is suppressed (1.A.1).
2. MICVPSW, which contains the address of the virtual PSW, is fetched with a key of zero. Execution ends if an addressing condition is encountered (1.A.2).
3. VMPSW, the virtual PSW, is fetched with a key of zero. Execution ends if an addressing condition is encountered (1.A.3).
4. Execution of this function ends if bit 12 of the virtual PSW is zero. Ending consists in invoking the STNSM function of the virtual-machine assist if that assist is installed; otherwise, the STNSM function of the expanded virtual-machine assist is invoked. If neither assist is installed, a program interruption for a privileged-operation exception takes place (1.A.4).
5. Execution of this function ends if the second operand is not FB hex. Ending consists in invoking the STNSM function of the virtual-machine assist if that assist is installed; otherwise, the STNSM function of the expanded virtual-machine assist is invoked. If neither assist is installed, a program interruption for a privileged-operation exception takes place (1.A.5).
6. The assist-control word, MICACF, is fetched with a key of zero. Execution ends if an addressing condition is encountered (1.A.6).
7. If bits 8 and 14 of the assist control word are not both ones, execution of this function ends. Ending consists in invoking the STNSM function of the virtual-machine assist if that assist is installed; otherwise, the STNSM function of the expanded virtual-machine assist is invoked. If neither assist is installed, a program interruption for a privileged-operation exception takes place (1.A.7).
8. If an access condition is encountered in fetching the second halfword of the STNSM

instruction, execution of this function ends, and a program interruption takes place for that access exception (1.B).

9. The first byte of the current virtual PSW is stored, with the real PSW key, at the location specified by the first-operand effective (logical) address. If an access exception is recognized, a program interruption takes place with that access condition indicated (2).
10. If bit 5 of the virtual PSW is zero, execution of this function ends (3).
11. The virtual PSW, VMPSW, is updated in storage with a key of zero to set bit 5 to zero (4.A).
12. Bits 8-12 of real control register 0 are set to 10000 binary. The real segment-table pointer, MICRSEG, is fetched with a key of zero and placed in real control register 1. Execution of the STORE THEN AND SYSTEM MASK instruction is terminated, and a program interruption takes place if an addressing exception is recognized (4.B.1).
13. The values of real control registers 0 and 1 are stored with a key of zero in the doubleword (RUNCR0 and RUNCR1) at real address 340 hex (4.B.2).

Figure 27 summarizes the fields used.

STORE THEN OR SYSTEM MASK

The STORE THEN OR SYSTEM MASK instruction is executed for a virtual machine if the corresponding function of the shadow-table-bypass assist is activated and if the instruction uses an I_2 field value of 04 hex to turn on the DAT bit in the virtual PSW.

The store-then-OR-system-mask (STOSM) function of the shadow-table-bypass assist is invoked each time a CPU attempts to execute a STORE THEN OR SYSTEM MASK instruction when the problem-state bit of the real PSW is one. Execution of this function consists in performing the following steps:

1. If bits 0-3 of control register 6 are not 10X0 binary, a program interruption takes place for a privileged-operation exception, and execution of the STOSM instruction is suppressed (1.A.1).
2. MICVPSW, which contains the address of the virtual PSW, is fetched with a key of zero. Execution ends if an addressing condition is encountered (1.A.2).
3. VMPSW, the virtual PSW is fetched with a key of zero. Execution ends if an addressing condition is encountered (1.A.3).
4. Execution of this function ends with a program interruption for a privileged-operation exception if bit 12 of VMPSW is zero. Ending consists in invoking the STOSM function of the virtual-machine assist if that assist is installed; otherwise, the STOSM function of the expanded virtual-machine assist is invoked. If neither assist is installed, a program interruption for a privileged-operation exception takes place (1.A.4).
5. Execution of this function ends if the second operand is not 04 hex. Ending consists in invoking the STOSM function of the virtual-machine assist if that assist is installed; otherwise, the STOSM function of the expanded virtual-machine assist is invoked. If neither assist is installed, a program interruption for a privileged-operation exception takes place (1.A.5).
6. The assist control word, MICACF, is fetched

Field Name	Control Block	Address Type	Offset (Hex)	No. of Bytes	Contents
MICRSEG	MICBLOK	Real	0	4	Real segment table pointer
MICVPSW	MICBLOK	Real	8	4	Address of VMPSW
MICACF	MICBLOK	Real	14	4	Assist control word
VMPSW	VMBLOK	Real	0	2	Virtual PSW bits 0-15
RUNCR0	PSA	Real	340	4	Control register 0 at dispatch
RUNCR1	PSA	Real	344	4	Control register 1 at dispatch
Operand 1	–	Logical	–	1	–

Figure 27. Fields Used in STORE THEN AND SYSTEM MASK

with a key of zero. Execution ends if an addressing condition is encountered (1.A.6).

7. If bits 8 and 14 of the assist control word are not both ones, execution of this function ends. Ending consists in invoking the STOSM function of the virtual-machine assist if that assist is installed; otherwise, the STOSM function of the expanded virtual-machine assist is invoked. If neither assist is installed, a program interruption for a privileged-operation exception takes place (1.A.7).
8. If an access condition is encountered in fetching the second halfword of the STOSM instruction, execution of this function ends, and a program interruption takes place for the access exception (1.B).
9. The first byte of the current virtual PSW is stored, with the real PSW key, at the location specified by the first-operand effective (logical) address. If an access exception is recognized, a program interruption takes place with that access condition indicated (2).
10. If bit 5 of the current virtual PSW is one, execution of this function ends (3).
11. The virtual PSW, VMPSW, is updated in storage with a key of zero to set bit 5 to one (4.A).
12. MICCREG, the word used to locate the shadow control register values, is fetched with a key of zero. Execution of the STORE THEN OR SYSTEM MASK instruction is terminated, and a program interruption takes place if an addressing exception is recognized (4.B.1).
13. The values of shadow control registers 0 and 1 are fetched with a key of zero and loaded into real control registers 0 and 1. Execution of the STORE THEN OR SYSTEM MASK instruction is terminated, and a program interruption takes place if an addressing exception is recognized (4.B.2).
14. The values of real control registers 0 and 1 are stored with a key of zero in the doubleword (RUNCR0 and RUNCR1) at real address 340 hex (4.B.3).

Figure 28 summarizes the fields used.

TEST PROTECTION

The TEST PROTECTION instruction is executed for a virtual machine with a virtual problem-state bit of zero if the test-protection function of the shadow-table-bypass assist is activated.

The test-protection (TPROT) function of the shadow-table-bypass assist is invoked each time a CPU attempts to execute a TEST PROTECTION instruction when the problem-state bit of the real PSW is one. Execution of this function consists in performing the following steps:

1. If bits 0-3 of control register 6 are not 10X0 binary, a program interruption takes place for a privileged-operation exception and execution of the TPROT instruction is suppressed (1.A.1).
2. The assist control word, MICACF, is fetched with a key of zero. Execution ends if an addressing condition is encountered (1.A.2).
3. Execution ends with a program interruption for a privileged-operation exception if bits 8 and

Field Name	Control Block	Address Type	Offset (Hex)	No. of Bytes	Contents
MICCREG	MICBLOK	Real	4	4	Address of ECBLOK
MICVPSW	MICBLOK	Real	8	4	Address of VMPSW
MICACF	MICBLOK	Real	14	4	Assist control word
EXTSHCR0	ECBLOK	Real	40	4	Shadow control register 0
EXTSHCR1	ECBLOK	Real	44	4	Shadow control register 1
VMPSW	VMBLOK	Real	0	2	Virtual PSW bits 0-15
RUNCR0	PSA	Real	340	4	Control register 0 at dispatch
RUNCR1	PSA	Real	344	4	Control register 1 at dispatch
Operand 1	–	Logical	–	1	Old system mask

Figure 28. Fields Used in STORE THEN OR SYSTEM MASK

10 of the assist control word are not both ones (1.A.3).
4. If an access exception is encountered in fetching the second or third halfwords of the TPROT instruction, execution of this function ends, and a program interruption takes place for the access exception (1.B).
5. The TPROT instruction is executed just as if the real problem-state bit were zero. If any exception is recognized, a program interruption takes place and indicates that exception in the normal manner. Otherwise, the condition code is set, and execution of the TPROT instruction is completed (2).

Figure 29 summarizes the fields used.

Page-Fault Reflection

The page-fault-reflection function of the shadow-table-bypass assist performs a program interruption in the virtual machine for a page-translation exception if the real PSW and the old and new virtual PSWs meet specified conditions.

The page-fault-reflection function is invoked each time the System/370 architecture calls for a program interruption for a page-translation exception when the problem-state bit of the real PSW is one. The execution of this function consists in performing the following steps:

1. Execution ends and a program interruption for a page-translation exception takes place if the VM-assist bit (bit 0 of control register 6) is not one (1).
2. On some models, the segment index and the page index of the untranslatable address are placed in the word at real location 90 hex. If the virtual-machine assist is installed and the shadow-table-validation bit (bit 5 of control register 6) is one, this function is completed by transferring control to the shadow-table-validation function of the virtual-machine assist. If the virtual-machine assist is not installed, no test is performed, and control remains in the page-fault-reflection function (2).

Steps 3 through 8 describe conditions that cause a program interruption in the real machine for the original page-translation condition:

3. The assist control word, MICACF, is fetched with a key of zero. Execution is completed if an addressing condition is encountered (3.A.1).
4. Execution is completed if bits 8 and 11 of the assist control word are not both ones (3.A.2).
5. MICVPSW, which contains the address of the virtual PSW, is fetched with a key of zero. Execution is completed if an addressing condition is encountered (3.B.1).
6. VMPSW, the virtual PSW, is fetched with a key of zero. Execution is completed if an addressing condition is encountered (3.B.2).
7. Execution is completed if in the virtual PSW bit 1 is one or bit 12 is zero (3.B.3).
8. Execution is completed if real PER is on (3.C).

Real address 0 is translated through the segment tables addressed by the first word in the virtual-machine parameter list. Steps 9 through 16 describe conditions that cause a program interruption in the real machine for the original page-translation condition.

9. An addressing exception is encountered in fetching the word MICRSEG with a key of zero. MICRSEG contains the address of the real segment table (4).
10. Bits 30 and 31 of the word containing the address of the real segment table are not both zeros (5).
11. The first segment-table entry, SEGPAGE, in the real segment table cannot be fetched with a key of zero because of an addressing exception (6).
12. SEGPAGE is invalid.
13. SEGPAGE has a format error (8).

Field Name	Control Block	Address Type	Offset (Hex)	No. of Bytes	Contents
MICACF	MICBLOK	Real	14	4	Assist control word
–	Virtual segment table	Real	–	4	Segment-table entry
–	Virtual Page table	Real	–	2	Page-table entry
Operand 2	–	Logical	1	1	Byte tested

Figure 29. Fields Used in TEST PROTECTION

14. The first page-table entry, PAGCORE, in the table addressed by the segment-table entry cannot be fetched with a key of zero because of an addressing exception (9).
15. PAGCORE is invalid (10).
16. PAGCORE has a format error (11).
17. The new virtual PSW is fetched with a key of zero from offset 68 hex in virtual page 0. Execution of this function ends if an addressing condition is encountered (12).
18. The new virtual PSW is tested for BC mode, DAT on, PER on, wait-state bit one, and format errors. If a virtual interruption is pending, tests are made to determine whether the input/output mask or the external mask is being changed from zero to one. If any of these conditions is found, a program interruption takes place in the real machine for the original page-translation condition (13).

    A program interruption takes place in the virtual machine for the original page-translation condition. This interruption consists in performing steps 19 through 27.
19. The old PSW, consisting of bits 0-15 from the current virtual PSW and bits 16-63 from the real PSW, is stored at virtual-machine location 28 hex with a key of zero (14.A).
20. The program-interruption identification word is stored with a key of zero at virtual-machine location 8C hex. This word consists of zeros in bits 0-12 and 15, the instruction-length code in bits 13-14, and the page-translation-exception code, 0011 hex, in bits 16-31 (14.B).
21. A word is stored at location 90 hex of the virtual machine with a key of zero. This word contains the segment and page indexes of the address for which the original page-translation exception was detected (14.C).
22. Bits 0-15 of the new virtual PSW replace the virtual PSW, VMPSW, in storage accessed with a key of zero (14.D).
23. Bits 8-12 of the real control register 0 are set to 10000 binary (14.E.1.A).
24. Real control register 1 is loaded with the real segment-table pointer (MICRSEG) (14.E.1.B).
25. Real control registers 0 and 1 are stored in a doubleword at a fixed location in the real PSA (RUNCR0 and RUNCR1) with a key of zero (14.E.2).
26. Bits 16-63 of the new virtual PSW are placed in the corresponding positions of the real PSW (14.F).
27. Bit 15 of the new virtual PSW is placed in bit position 1 of control register 6 (14.G).
28. Execution of the current function is completed, and virtual instruction execution is nullified (15).

Figure 30 summarizes the fields used.

Field Name	Control Block	Address Type	Offset (Hex)	No. of Bytes	Contents
MICRSEG	MICBLOK	Real	0	4	Address of real segment table
MICVPSW	MICBLOK	Real	8	4	Address of VMPSW
MICACF	MICBLOK	Real	14	4	Assist control word
VMPSW	VMBLOK	Real	0	8	Virtual PSW
SEGPAGE	SEG TABLE	Real	0	4	First real segment-table entry
PAGCORE	PAG TABLE	Real	0	2	Address of PSA of virtual machine
–		Real[1]	28	8	Old program PSW
–		Real[1]	68	8	New program PSW
–		Real[1]	8C	4	Program-interruption identification
–		Real[1]	90	4	Translation-exception address
RUNCRO		Real[2]	340	4	Control register 0 at dispatch
RUNCR1		Real[2]	344	4	Control register 1 at dispatch

1 This real address is obtained by the address translation performed in steps 9-16 without using control registers 0 and 1. The control block for these items is the PSA of virtual machine.

2 The control blocks for these items is the PSA of real machine.

Figure 30. Fields Used in Page-Fault Reflection

Appendix. Deviations for Virtual-Machine-Assist Functions

~~The deviations discussed here apply to the~~ specifications given in this publication.

In certain cases in which both an operand-access condition and a privileged-operation condition exist, the program interruption indicates the operand-access exception instead of the privileged-operation exception, as called for in the detailed definition of a particular function. These cases exist when a privileged-operation condition arises because of one of the following:

1. Bits 0-1 of control register 6 are 11 binary.
2. Bits 0-3 of control register 6 are 10x1 binary.
3. Bits 0-1 of real control register 6 are 10 binary, and bit 1 of virtual control register 0 is one.

Figure 31 lists, by assist function, the models for which these cases (1, 2, and 3) arise.

	Cause of Privileged-Operation Exception		
VMA Function	1	2	3
Insert PSW key	–	–	–
Insert storage key	–	–	–
Load PSW	168	168	–
Load real address	–	–	–
Reset reference bit	–	–	–
Set PSW key from address	–	–	–
Set storage key	–	–	–
Set system mask	168	168	168,3032,3033
Store control	168	168	–
Store then AND system mask	168	168	–
Store then OR system mask	168	168	–
Supervisor call	–	–	–
Shadow-table validation	–	–	–

Figure 31. Models That May Indicate Operand-Access Exception in Place of Privileged-Operation Exception

Index

GA22-7074-0

IBM Virtual-Machine Assist and Shadow-Table-Bypass Assist (File No. S370-01) Printed in U.S.A. GA22-7074-0

IBM

IBM Virtual-Machine Assist and Shadow-Table-Bypass Assist

Order No. GA22-7074-0

READER'S COMMENT FORM

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. This form may be used to communicate your views about this publication. They will be sent to the author's department for whatever review and action, if any, is deemed appropriate. Comments may be written in your own language; use of English is not required.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Note: *Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.*

Possible topics for comment are:

Clarity Accuracy Completeness Organization Coding Retrieval Legibility

If you wish a reply, give your name and mailing address:

____

____

____

Note: Staples can cause problems with automated mail sorting equipment.
Please use pressure-sensitive or other gummed tape to seal this form.

What is your occupation? _______________

Number of latest Newsletter associated with this publication: _______________

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the front cover or title page.)

GA22-7074-0

Reader's Comment Form

Fold and tape | Please Do Not Staple | Fold and tape

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, NY

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Department B98
P.O. Box 390
Poughkeepsie, New York 12602

Fold and tape | Please Do Not Staple | Fold and tape

IBM®

Cut or Fold Along Line

Cut or Fold Along Line