GA22-7079-1
File No. S370-01

Systems

IBM System/370 Assists for MVS

IBM

Second Edition (October 1981)
This is a major revision of GA22-7079-0, which is now obsolete. This edition incorporates information about the page-fault assist. Changes or additions to the text or illustrations are indicated by a vertical line in the margin to the left of the change.

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

It is possible that this material may contain reference to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming, or services in your country.

Publications are not stocked at the address given below; requests for IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for reader's comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Product Publications, Dept. B98, PO Box 390, Poughkeepsie, NY, U.S.A. 12602. IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Preface

This publication describes the 13 instructions and the page-fault assist that are used to assist the MVS control program. Of the 13 instructions, 12 are provided when the System/370 extended facility or the extended control-program support for MVS (ECPS:MVS) is installed. The 12 instructions include four lock-handling instructions, six tracing instructions, and the instructions FIX PAGE and SVC ASSIST. When the 3033 extension feature is installed, the instruction ADD FRR is provided, along with the page-fault assist and modifications to the FIX PAGE and SVC ASSIST instructions.

The following table shows which assists for MVS are incorporated in (1) the System/370 extended facility, (2) the extended control-program support for MVS (ECPS:MVS), and (3) the 3033 extension feature (3033X).

Assists for MVS	Part of		
	Extended Facility	ECPS:MVS	3033X
4 lock-handling instructions	X X	X X	X X
6 tracing instructions	X	X	X
SVC ASSIST	X	X	X[1]
FIX PAGE	X	X	X[1]
ADD FRR			X
Page-fault assist			X

[1]These assists are modified for consistency with MVS use of the dual-address-space facility.

This publication is intended for system programmers and IBM Field Engineering personnel. The reader should be familiar with the general machine functions of System/370, as described in the *IBM System/370 Principles of Operation,* GA22-7000, and with the MVS system. The standard names for MVS fields and control blocks are used throughout the publication.

The following reading is considered prerequisite:

MVS/System Extensions: Debugging Handbook (Volume 2), SD23-0002

OS/VS2 Data Areas, SYB8-0606. (This document is on microfiche.)

OS/VS2 MVS/System Extensions General Information Manual, GC28-0872

OS/VS2 System Logic Library, SBOF-8210

Contents

Assists for MVS

Thirteen instructions and the page-fault assist are available to assist the MVS control program. The instructions depend on the particular conventions, fields, and control-block formats of the MVS system, and their use enhances the performance and reliability of several MVS program products.

The instructions consist of:

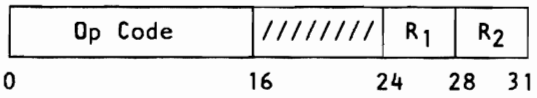
- Four lock-handling instructions
- Six tracing instructions
- The FIX PAGE instruction
- The SVC ASSIST instruction
- The ADD FRR instruction

The page-fault-assist function improves MVS performance by directly assigning and initializing a page frame when a page-translation exception is recognized on first reference to certain virtual pages.

Attributes of MVS-ASSIST Instructions

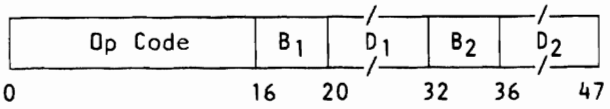
The MVS-assist instructions depend on the conventions and control-block formats of the MVS control program. Therefore, although these instructions can be executed in any operating-system environment, they may not be useful when used outside of MVS.

The ADD FRR instruction uses the RRE format:



This format permits two general registers to be specified, as do RR-format instructions. However, the RRE format uses a 16-bit operation code.

The other MVS-assist instructions use the SSE instruction format:



This format provides for addressing a first and second operand in a manner identical to that used for SS instructions. It differs from the SS format in that the operation code consists of 16 bits.

When the dual-address-space facility (part of the 3033 extension feature) is installed, the FIX PAGE and SVC ASSIST instructions are modified to be consistent with the MVS usage conventions for the dual-address-space facility.

Operand Addressing

In addition to the two operands explicitly designated by the $D_1(B_1)$ and $D_2(B_2)$ fields, many MVS-assist instructions use operands which are at specific, fixed main-storage locations or which are pointed to by general registers. Furthermore, many operands fetched from main storage are in turn used for addressing still other operands.

In EC mode, the translation-mode bit of the PSW controls address translation for all main-storage operand references of all MVS-assist instructions.

When the dual-address-space facility is installed and the CPU is in the EC mode with DAT on, PSW bit 16 controls whether control register 1 or control register 7 is used for translating the addresses of operands.

MAPL Control Block

Two instructions, FIX PAGE and SVC ASSIST, use a parameter list (MAPL) as an operand. The first eight words are used by SVC ASSIST, and the rest are used by FIX PAGE. This parameter list must be aligned on a doubleword boundary. During the execution of SVC ASSIST and FIX PAGE, the last three bytes of the word at location A4 hex are assumed to contain the address of the MAPL.

When the dual-address-space facility is installed, the MAPL parameter list is extended by two words. The first added word is used by the FIX PAGE instruction.

The MAPL layout is:

Offset Decimal	Offset (Hex)	Name	Contents
0-3	(0-3)	MPLSVCTA	SVC-table address
4-7	(4-7)	MPLSVRBP	Prefix length in bytes of an SVRB
8-11	(8-B)	MPLESVC1	Entry address for SVCs of type 1
12-15	(C-F)	MPLRSVC1	Exit address for SVCs of type 1
16-19	(10-13)	MPLESVC2	Entry address for SVCs of types 2, 3, and 4
20-23	(14-17)	MPLRSVC2	Exit address for SVCs of types 2, 3, and 4
24-27	(18-1B)	MPLESVC6	Entry address for SVCs of type 6
28-31	(1C-1F)	MPLRSVC6	Exit address for SVCs of type 6
32-33	(20-21)	MPLLCSA	16 times the lowest virtual common page-frame number
34-35	(22-23)	MPLLPRIV	16 times the lowest virtual private page-frame number
36-39	(24-27)	MPLPFTP	Page-frame-table origin
40-41	(28-29)	—	Reserved
42-43	(2A-2B)	MPLMAXFX	Maximum number of pages to be fixed
44-47	(2C-2F)	MPLCNTRS	Address of PVT + X'720'
48-51	(30-33)	—	Reserved
52-55	(34-37)	MPLPFAL	Entry address to move a page to a preferred frame
56-59	(38-3B)	MPLPFCM	Entry address if MPLMAXFX is reached
60-63	(3C-3F)	—	Reserved
64-67	(40-43)	MPLASVTP	Address of the address-space vector table
68-71	(44-47)	—	Reserved

Lock-Interface Table

The four lock-handling instructions may, under certain conditions, access the lock-interface-table prefix. The lock-interface table is located by an address contained in the word in main storage following the second-operand word. The prefix of the lock-interface table consists of four words, arranged as follows:

Offset Decimal	Offset (Hex)	Field Name
-16	(-10)	LITOLOC
-12	(-C)	LITRLOC
-8	(-8)	LITOCMS
-4	(-4)	LITRCMS

Program Interruptions

PER Events

The MVS-assist instructions are subject to PER interruption controls, except that branch events are not recognized when these instructions cause branching.

Privileged-Operation Exceptions

All MVS-assist instructions are privileged.

Simplified Execution Paths

Simplified execution paths are defined for the following MVS-assist instructions. When a simplified path is used, specific actions defined for the corresponding instruction are performed unconditionally; that is, specific actions are taken without the prescribed tests being made to determine that those actions should be selected.

- *OBTAIN LOCAL LOCK.* Execution proceeds as if the local lock were already held.
- *RELEASE LOCAL LOCK.* Execution proceeds as if the local lock were already released.
- *OBTAIN CMS LOCK.* Execution proceeds as if a CMS lock were already held.
- *RELEASE CMS LOCK.* Execution proceeds as if the currently dispatched unit of work held no CMS lock.
- *FIX PAGE.* Execution consists in loading the next instruction address, prefixed by eight zeros, in general register 14; in loading the contents of MPLPFAL in general register 15; and in placing bits 8-31 of MPLPFAL in the instruction-address part of the PSW.
- *SVC ASSIST.* Instruction execution is completed with normal instruction sequencing and without the performance of other actions.

Use of the simplified execution paths is not apparent to application programs using program products. In certain models, simplified execution paths are used when the control-storage space available is limited.

Instructions

The instructions described in this section are listed in the figure "Instruction Summary," together with their operation codes and the program-interruption conditions that can be recognized when they are executed.

In the format shown in the instruction description, the operation code is given in hex, which is signified by enclosing its value in single quotation marks ('XXXX').

Name	Characteristics							Code
OBTAIN LOCAL LOCK	SSE	XF	M	A	SP	R	ST	E504
RELEASE LOCAL LOCK	SSE	XF	M	A	SP	R	ST	E505
OBTAIN CMS LOCK	SSE	XF	M	A	SP	R	ST	E506
RELEASE CMS LOCK	SSE	XF	M	A	SP	R	ST	E507
TRACE SVC INTERRUPTION	SSE	XF	M	A	SP		ST	E508
TRACE PROGRAM INTERRUPTION	SSE	XF	M	A	SP		ST	E509
TRACE INITIAL SRB DISPATCH	SSE	XF	M	A	SP		ST	E50A
TRACE I/O INTERRUPTION	SSE	XF	M	A	SP		ST	E50B
TRACE TASK DISPATCH	SSE	XF	M	A	SP		ST	E50C
TRACE SVC RETURN	SSE	XF	M	A	SP		ST	E50D
FIX PAGE	SSE	XF	M	A	SP	R	ST	E502
SVC ASSIST	SSE	XF	M	A	SP	R	ST	E503
ADD FRR	RRE	3X	M	A	SP	R	ST	B242

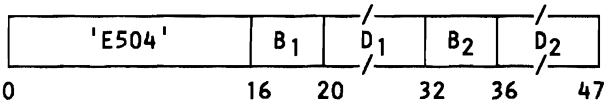
Explanation:

A	Access exceptions
M	Privileged-operation exception
R	PER general-register-alteration event
RRE	RRE instruction format
SP	Specification exception
SSE	SSE instruction format
ST	PER storage-alteration event
XF	System/370 extended facility and ECPS:MVS feature
3X	3033 extension feature

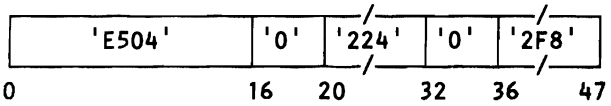
Instruction Summary

OBTAIN LOCAL LOCK

General Form



Form Used in Program Products



If the local lock in the ASCB addressed by the first-operand word is not held, the lock is replaced, by using an interlocked update, with the value from PSALCPUA; the local-lock bit of the highest-lock-held-indicator word fetched from the second-operand location is set to one; and zeros are placed in general register 13.

Otherwise, the updated instruction address, prefixed by eight zeros, is placed in general register 12; the contents of LITOLOC are placed in general register 13; and bits 8-31 of the contents of LITOLOC are placed in the instruction-address portion of the PSW.

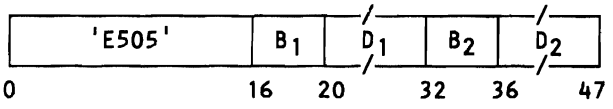
Serialization occurs before the local lock is fetched and, if the lock is obtained, again after the lock is updated.

Condition Code: The code remains unchanged.

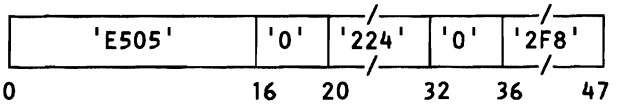
Program Exceptions:
Access
Operation (if instruction is not installed)
Privileged operation
Specification

RELEASE LOCAL LOCK

General Form



Form Used in Program Products



If the highest-lock-held-indicator word fetched from the second-operand location shows that the executing CPU holds the local lock and does not hold a CMS lock, and if the word after the local lock word in the ASCB addressed by the first-operand word is zero, then the doubleword containing the lock is set to zero by using interlocked update. Also, the local-lock bit of the highest-lock-held-indicator word is set to zero, and zeros are placed in general register 13.

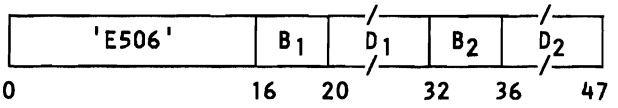
Otherwise, the updated instruction address, prefixed by eight zeros, is placed in general register 12; the contents of LITRLOC are placed in general register 13; and bits 8-31 of the contents of LITRLOC are placed in the instruction-address portion of the PSW.

Condition Code: The code remains unchanged.

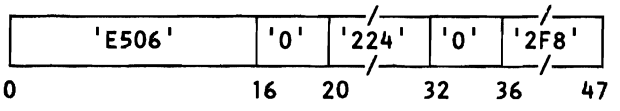
Program Exceptions:
Access
Operation (if instruction is not installed)
Privileged operation
Specification

OBTAIN CMS LOCK

General Form



Form Used in Program Products



If the highest-lock-held-indicator word fetched from the second-operand location shows that the executing CPU holds the local lock and does not hold a CMS lock, and if the CMS lock addressed

by bits 8-31 of general register 11 is not held, then the lock is replaced, by using an interlocked update, with the first-operand word. Also, the highest-lock-held indicator is set to show that a CMS lock is held, and zeros are placed in general register 13.

Otherwise, the updated instruction address, prefixed by eight zeros, is placed in general register 12; the contents of LITOCMS are placed in general register 13; and bits 8-31 of the contents of LITOCMS are placed in the instruction-address portion of the PSW.

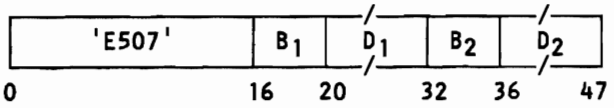
Serialization occurs before the lock is fetched and, if the lock is obtained, again after the lock is updated.

Condition Code: The code remains unchanged.

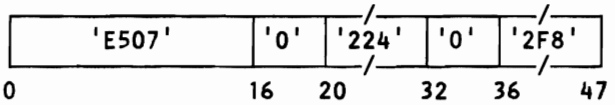
Program Exceptions:
Access
Operation (if instruction is not installed)
Privileged operation
Specification

RELEASE CMS LOCK

General Form



Form Used in Program Products



If (1) the contents of the CMS lockword addressed by bits 8-31 of general register 11 equal the contents of the first-operand word, (2) the currently dispatched unit of work holds a CMS lock, and (3) the word after the CMS lockword is zero, then the doubleword containing the lockword is set to zero, by using an interlocked update. Also, the highest-lock-held-indicator word fetched from the second-operand location is set to show that no CMS lock is held, and zeros are placed in general register 13.

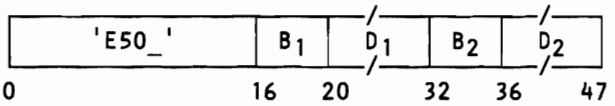
Otherwise, the updated instruction address, prefixed by eight zeros, is placed in general register 12; the contents of LITRCMS are placed in general register 13; and bits 8-31 of the contents of LITRCMS are placed in the instruction-address portion of the PSW.

Condition Code: The code remains unchanged.

Program Exceptions:
Access
Operation (if instruction is not installed)
Privileged operation
Specification

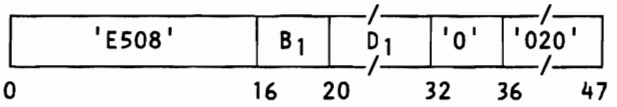
TRACE Instructions

General Form

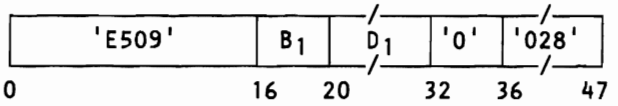


Forms Used in Program Products

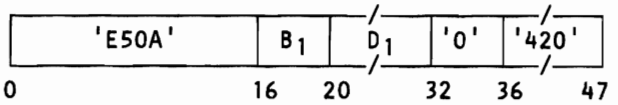
TRACE SVC INTERRUPTION



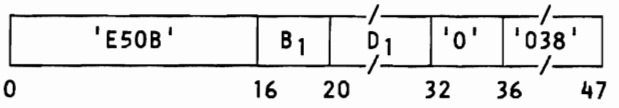
TRACE PROGRAM INTERRUPTION



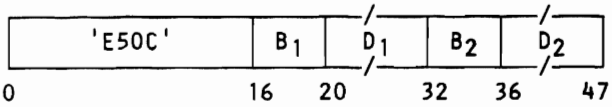
TRACE INITIAL SRB DISPATCH



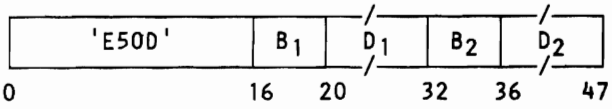
TRACE I/O INTERRUPTION



TRACE TASK DISPATCH



TRACE SVC RETURN



All six tracing instructions use implicit logical addresses less than 4095. TRACE INITIAL SRB DISPATCH uses the contents of bit positions 8-31 of general register 0 as the address of an SRB to access fields in that SRB. TRACE TASK DISPATCH uses the contents of bit positions 8-31 of general register 3 as the address of an RB to access fields in that RB. The first operand of all tracing instructions is a halfword field. The second operand of all tracing instructions is the PSW to be traced.

The word at location 84 (54 hex) contains the address of the trace-table-entry header. The trace-table-entry header is aligned on a doubleword boundary and consists of three words. Each tracing instruction performs an interlocked update of the first word of the header. The updated value contains the address of a trace-table entry (TTE). Each tracing instruction stores 32 bytes in a TTE, which must be on a 32-byte boundary. Condition code 0 is set, and the updated word in the TTE header is the old value plus 32 (20 hex) unless that value is logically greater than or equal to the value of the third word in the TTE header. In the latter case, condition code 1 is set, and the updated TTE header word contains the value from the second word of the TTE header.

When the dual-address-space facility is installed, bit 0 of location 84 (54 hex) is used to control whether a trace entry is created in conjunction with the execution of PROGRAM CALL, PROGRAM TRANSFER, and SET SECONDARY ASN.

The figure "Trace-Table-Entry Summary" shows a row for each field of a 32-byte TTE and a column for each tracing instruction. For a particular row and column, the entry shows from which register, operand, PSA field, or other source a value is copied into the TTE at that offset (row) for that tracing instruction (column).

Note: The SRB control block addressed by general register 0 in TRACE INITIAL SRB DISPATCH has no required alignment. This is to accommodate existing user programs that have not previously been checked for SRB alignment.

Condition Code:

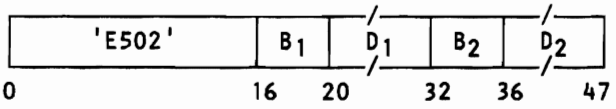
0 TTE placed at next sequential location
1 TTE placed at table start because table wrapped
2 –
3 –

Program Exceptions:

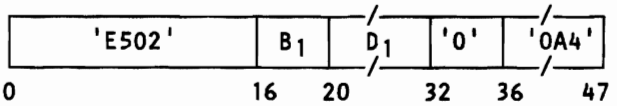
Access
Operation (if instruction is not installed)
Privileged operation
Specification

FIX PAGE

General Form



Form Used in Program Products



The contents of general register 1 are used as the virtual address of a location in a page to be fixed. The contents of general register 2 are used as the virtual address of a location in the last of a group of consecutive pages to be fixed. The contents of general register 0 are used as the real address of a location in the page frame containing the page to be fixed. The second operand is a word containing the address of the MAPL control block.

If the page to be fixed is a nucleus page, an LSQA/SQA page, or a virtual-equals-real page, then the count of the number of times the page frame containing that page is currently fixed (called the fix count) is not examined or incremented. If (1) the fix count for the page frame containing the page to be fixed is zero, (2) the page frame is not

Decimal (Hex) Byte Offsets within Trace-Table Entry		Instruction					
		TRACE SVC INTERRUPTION	TRACE PROGRAM INTERRUPTION	TRACE INITIAL SRB DISPATCH	TRACE I/O INTERRUPTION	TRACE TASK DISPATCH	TRACE SVC RETURN
Bytes 0-1		2nd Operand Bytes 0-1	2nd Operand Bytes 0-1	2nd Operand Bytes 0-1	2nd Operand Bytes 0-1	2nd Operand Bytes 0-1	2nd Operand Bytes 0-1
Byte 2	Bits 0-3	'2'	'3'	'4'	'5'	'7'	'8'
	Bits 4-7	'0'	'0'	'0'	FLCIOAA Bits 12-23	'0'	'0'
Byte 3		FLCSVCN Byte 1	FLCPICOD Byte 1	'0'		RBINTCOD Byte 1	2nd Operand Byte -17[3]
Bytes 4-7		2nd Operand Bytes 4-7	2nd Operand Bytes 4-7	2nd Operand Bytes 4-7	2nd Operand Bytes 4-7	2nd Operand Bytes 4-7	2nd Operand Bytes 4-7
Bytes 8-9		General Register 15	General Register 15	'0'	FLCCSW	General Register 15	General Register 15
Bytes 10-11 (A-B)				SRBPASID			
Bytes 12-15 (C-F)		General Register 0	FLCTEA	General Register 0		General Register 0	General Register 0
Bytes 16-19 (10-13)		General Register 1	General Register 1	General Register 1	'0'	General Register 1	General Register 1
Byte 20 (14)	Bits 0-1	FLCSVILC Bits 5-6	FLCPIILC Bits 5-6	'0'	2nd Operand Byte 2	RBINLNTH Bits 5-6	2nd Operand Byte -19[3] Bits 5-6
	Bits 2-7	2nd Operand Bits 18-23	2nd Operand Bits 18-23			2nd Operand Bits 18-23	2nd Operand Bits 18-23
Byte 21[1] (15)		Bits 8-15 of the two-byte CPU address					
Byte 21[2] (15)	Bits 0-3	Bits 8-15 of the two-byte CPU address			Bits 12-15 of PSACSID	Bits 8-15 of the two-byte CPU address	
	Bits 4-7				Bits 12-15 of CPU address		
Bytes 22-23 (16-17)		1st Operand Bytes 0-1	1st Operand Bytes 0-1	1st Operand Bytes 0-1	1st Operand Bytes 0-1	1st Operand Bytes 0-1	1st Operand Bytes 0-1
Bytes 24-27(18-1B)		PSATOLD	PSATOLD	SRBPTCB	PSATOLD	PSATOLD	PSATOLD
Bytes 28-31(1C-1F)		Bytes 3-6 of doubleword that would be stored by the instruction STORE CLOCK					

[1]On configurations without channel-set switching.
[2]On configurations with channel-set switching.
[3]This number is a negative offset.

Trace-Table-Entry Summary

in the preferred area, and (3) the page is either a private page of a second-level preferred user or a common page, then execution is completed by loading the next instruction address, prefixed by eight zeros, in general register 14; by loading the contents of MPLPFAL in general register 15; and by placing the contents of bits 8-31 of the contents of MPLPFAL in the instruction-address part of the PSW. Otherwise, the fix count for the page frame containing the page to be fixed is incremented by one. If the incremented fix count is one, the total system count of fixed frames is incremented by one, and either the number of private pages fixed in the address space to which the frame is assigned or the number of frames allocated to fixed common pages is incremented by one, depending on whether the page being fixed is a private or common page, respectively.

If the virtual address of the fixed page is less than the virtual address of the last page of the consecutive group of pages to be fixed, general register 1 is incremented by 4096, and execution is completed by placing the first-operand address in the instruction-address part of the PSW. Otherwise, the total system count of fixed frames is compared with the maximum fix-count threshold. If the threshold has been reached, instruction execution is completed by (1) loading the next instruction address, prefixed by eight zeros, in general register 14; (2) loading the contents of MPLPFCM in general register 15; and (3) placing bits 8-31 of the contents of MPLPFCM in the instruction-address part of the PSW. Otherwise, general register 15 is set to zero, and normal instruction sequencing proceeds with the updated instruction address.

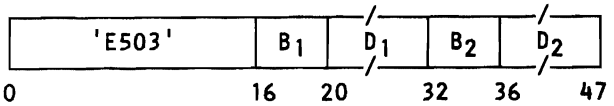
If the dual-address-space facility is not installed or if bit 0 of MPLPFTP is zero, PSAAOLD holds the address of the ASCB for the address space using the page to be fixed. Otherwise, the ASN of the address space using the page to be fixed is obtained from control register 3 or control register 4, depending on whether bits 12 and 16 of the PSW are or are not both ones. The ASN is used as an index into the ASVT to locate the ASCB. This latter method accommodates the use of the dual-address-space facility.

Condition Code: The code remains unchanged.

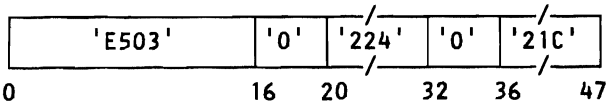
Program Exceptions:
Access
Operation (if instruction is not installed)
Privileged operation
Specification

SVC ASSIST

General Form



Form Used in Program Products



The first and second operands are words containing the addresses of the current ASCB and TCB, respectively.

Main-storage locations which contain the MVS system status for the last SVC interruption are tested to determine if SVC-assist action is to be taken. If SVC-assist action is not taken, instruction execution is completed with normal instruction sequencing. No assist action is taken unless the CPU was enabled prior to the last SVC interruption, and a task is currently dispatched which holds no locks and for which SVC screening is not activated.

If the dual-address-space facility is installed and bit 0 of MPLSVCTA is one, the assist is active only if (1) the primary space, the secondary space, and the last-dispatched space are the same space, and (2) primary-space mode is specified in the supervisor-call old PSW.

A type-1 SVC request is assisted only if the request is for an assistable type-1 function, if the only lock needed for the requested SVC function is the local lock, and if an attempt to obtain the local lock is successful.

A type-2, -3, or -4 SVC request is assisted only if the request is for an assistable function of type 2, 3, or 4, and if an attempt to dequeue an SVRB from the SVRB pool of the current address space is successful.

A type-6 SVC request is assisted only if the request is for an assistable type-6 function for which no locks are needed.

Assist action consists in copying the information stored at the last SVC interruption into the current request block, saving all 16 general registers, loading the general registers (as shown in the following chart), and then loading the instruction address portion of the PSW from bit positions 8-31 of general register 9.

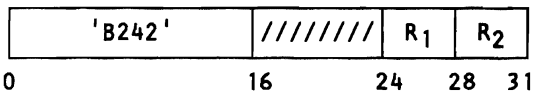
GR No.	Contents Loaded on Assist Action
3	Address of CVT
4	Second operand
5	Address of the current RB for type 1 or type 6; address of the acquired SVRB for type 2, 3, or 4
6	SVC entry-point address
7	First operand
8	Address of the RB for the program which was being executed at SVC interruption
9	Entry address from that MAPL field which is appropriate to the SVC type
11	Address of the MBCB part of the SVRB acquired; for type 2, 3, or 4 only
12	Address of requested SVC-table entry
14	Exit address from that MAPL field which is appropriate to the SVC type

Condition Code: The code remains unchanged.

Program Exceptions:
Access
Operation (if instruction is not installed)
Privileged operation
Specification

ADD FRR

[RRE]



A new entry is added to the top of the current functional-recovery-routine (FRR) stack. The entry is initialized with values provided in general registers and with the PSW S bit (bit 16).

Optionally, the contents of control registers 3 and 4 are saved in an entry in a separate table.

The general register designated by the R_2 field provides the logical address of the FRR entry point.

Before instruction execution, the general register designated by the R_1 field provides three bytes that are stored in the FRR entry and whose value determines if control registers 3 and 4 are to be stored as well. When instruction execution is completed, the register designated by R_1 contains the logical address of the six-word work area within the new, current FRR-stack entry.

Logical location 380 hex contains the logical address of the stack-table header. The stack-table header contains (1) a logical address which is 32 less than the address of the first dynamic entry in the stack table, (2) the logical address of the last entry in the stack table, and (3) the logical address of the current stack-table entry.

At an offset from the beginning of the stack-table header is found a table of stack-entry-extension entries. Optionally, the contents of control registers 3 and 4 are saved in an extension entry. One extension entry corresponds to each entry in the stack table. The offset to the table of extension entries, and the encoded length of an extension entry, are found in the word at logical location BA8 hex.

Condition Code: The code remains unchanged.

Program Exceptions:
Access (storage operands)
Operation (when the instruction is not installed)
Privileged operation
Specification

Page-Fault Assist

The page-fault assist is invoked when a page-translation exception caused by an invalid page-table entry is recognized outside the page-fault assist. Depending on the model, instruction-fetch and data-fetch references, but not data-store references, may alternatively be handled by taking the interruption for the exception instead of invoking this assist. When the exception is caused by a first-time reference, this function assigns a page frame from either of two lists of available frames. Depending on an indicator which determines which list is examined first, the frame is assigned from above or below the 2^{24} absolute-address boundary. When the first-choice list is empty, the other list is examined. The assigned frame is cleared, the key is set, and the

page-table entry is made valid. The instruction giving rise to the exception is then reexecuted. It is unpredictable whether the interruption parameters are stored even if the interruption is not taken.

A frame is assigned through manipulation of lists of PFTE control blocks. The assigned PFTE represents the assigned frame and is obtained from the beginning of either the available-below frame queue (AFQ) or the available-above frame queue (AAFQ). For a common-area page, the assigned PFTE is enqueued at the end of the common-frame queue (CFQ). For a private-area page, the assigned PFTE is enqueued at the end of the local-frame queue (LFQ) of the address space for which the translation exception was recognized.

Unless otherwise stated, the values of fields referred to in the following steps are those at the time the page-fault assist is entered.

The steps in the execution of the page-fault assist are as follows:

1. Execution of the assist ends if any of the following conditions exists:
   a. Bit 13 of control register 0 is zero.
   b. PSW bit 1 (PER) is one.
   c. PSW bits 6-7 (external and I/O masks) are not 11 binary.
   d. Bits 8-12 of control register 0 are not 10000 binary.

   The ending consists in the CPU taking a program interruption for a page-translation exception.
2. PSALCPUA is fetched. PSASALCL, the address of the salloc word, is fetched. Bit 2 (PSAPFA) of the byte at 588 hex (PSAHWFB) is set to one. A zero in the salloc word is replaced by the contents of PSALCPUA by means of a compare-and-swap-type function. If the salloc word is not zero, it is not updated, PSAPFA is set to zero, and execution ends by the CPU taking a program interruption for a page-translation exception. Otherwise, bit 5 (PSASALLI) of byte 2 of PSAHLHI is set to one after the salloc word is updated.
3. For any of the following conditions, execution ends by the CPU taking a program interruption for a page-translation exception, but only after zeros are placed in the salloc word, PSASALLI is set to zero, and PSAPFA is set to zero:
   a. The PTE associated with the translation exception address is not 0019 hex.
   b. PVTAFC-1 equals PVTAFCLO.
4. If XPTBBELO (bit 4 of XPTFLAG2) is one, step 6 is taken. Otherwise, step 5 is taken. The actions in steps 5 and 6 are attempts to obtain a page frame from one of two lists of available frames. When no frames are available on the first list examined, the second list is examined.

   The test of XPTBBELO, in step 5, and the entry to step 5 from step 6 are not necessarily implemented on machines that do not provide more than 16 megabytes of real storage.
5. The actions in this step are an attempt to assign a page frame from a list of available frames with absolute addresses greater than $2^{24} - 1$. This step is entered initially either because a frame of this type is preferred or because the other type is preferred but none is available. This step is also entered a second time when no frames of either type are available; this circumstance is detected and results in ending the execution. The actions performed are as follows:
   a. If this step is being entered for the second time, execution ends by the CPU taking a program interruption for a page-translation exception, but only after the salloc word is set to zero and PSASALLI and PSAPFA are set to zero.
   b. If PVTAAFQF is zero, then step 6 is taken.

   The next three actions (c through e) are performed in any order:
   c. Temp0 is set to the 24-bit address of PVTAAFQF.
   d. The halfword temp1 is set to the contents of PVTAAFQF.
   e. Temp2 is set to PVTAAFQF x 16 + PVTPFTP, which is the address of the assigned PFTE.
   f. If the halfword PFTNXRBN at temp2 + 4 is not zero, then step 7 is taken. Otherwise, execution ends by the CPU taking a program interruption for a page-translation exception, but only after the salloc word and PSASALLI are set to zero and PSAPFA is set to zero.
6. The actions in this step are an attempt to assign a page frame from a list of available frames with absolute addresses less than 2^{24}. This step is entered either because a frame with absolute addresses less than 2^{24} is preferred or because no frames are available on the other list. The actions performed are as follows:
   a. If PVTAFQF is zero, step 5 is taken.

   The next three actions (b through d) are performed in any order.
   b. Temp0 is set to the 24-bit address of PVTAFQF.

c. The halfword temp1 is set to the contents of PVTAFQF.
d. Temp2 is set to PVTAFQF x 16 + PVTPFTP, which is the 24-bit address of the assigned PFTE.
e. If the halfword PFTNXRBN at temp2 + 4 is not zero, step 7 is taken. Otherwise, execution ends by the CPU taking a program interruption for a page-translation exception, but only after the salloc word and PSASALLI are set to zero and PSAPFA is set to zero.

7. This step initializes the page frame and updates the PTE, the PFTE, and certain PVT controls. On entry to this step, temp0 contains the address of the halfword containing the index of the assigned frame, temp1 contains the index of the assigned frame, and temp2 contains the address of the PFTE which represents the assigned frame. The actions performed are as follows.

   The next two actions are performed in the order shown and may be deferred on some models to just before step 11:

   a. The page frame is cleared to zeros, and the storage keys are set to the value of bits 0-6 of XPTPROT; the order in which the keys are set and the frame cleared is undetermined.
   b. PTE bits 13 and 14 are set from bits 2 and 3 of temp1. PTE bits 0-11 are set from bits 4-15 of temp1. PTE bit 15 is set to one. PTE bit 12 (entry-invalid bit) is set to zero.

   The next six actions are performed in any order, and any of them may be performed either before or after the preceding two actions.

   c. The value in PVTAFC is reduced by one.
   d. Zero is placed in bit position 0 (PFTONAVQ) in the byte PFTFLAG1 at temp2 + 12.
   e. Zero is placed in bit position 3 (PFTIRRG) and, if XPTBBELO is one, bit 4 (PFTBBELO) is set to one in the byte PFTFLAG2 at temp2 + 13.
   f. The virtual block number (VBN), bits 8-19 of the virtual address for which the translation exception was recognized, with four zero bits appended on the right, is placed in the halfword PFTVBN at temp2 + 2.
   g. A halfword is fetched from temp2 + 4 (=PFTNXRBN). The halfword is multiplied by 16, and the contents of PVTPFTP are added to the result, which is then incremented by 6 (=PFTPXRBN). Zeros are placed in the halfword (PFTPXRBN) designated by this result.
   h. The contents of the halfword at temp2 + 4 (PFTNXRBN) are copied into the halfword addressed by temp0 (which is either PVTAFQF or PVTAAFQF).

   The next action is taken any time after the preceding two actions (g and h) are completed:

   i. Zeros are placed in the halfword PFTNXRBN at temp2 + 4.

8. The halfword placed in PFTVBN is compared logically with the halfword at PVTLCSA. If the contents of PFTVBN are greater than or equal to the contents of PVTLCSA, step 9 is taken. Otherwise, step 10 is taken.
9. The PFTE for the assigned frame is added to the list of PFTEs representing frames backing common virtual storage:
   a. If the contents of PVTCFQL are not zero, temp1 is placed in PFTNXRBN in a common PFTE. The address of PFTNXRBN is the sum of 4 plus PVTCFQL times 16 plus PVTPFTP.

   The remaining actions are performed in any order:

   b. FFFF hex is placed in PFTASID at temp2.
   c. PVTCFMCT + 1 is placed in PVTCFMCT.
   d. The value 08 hex is placed in PFTQNDX at temp2 + 14.
   e. PVTCFQL is placed in PFTPXRBN at temp2 + 6, and temp1 is then placed in PVTCFQL.
   f. If the contents of PVTCFQF are zero, then temp1 is placed in PVTCFQF.

   When the above actions are completed, step 11 is taken.

10. The PFTE for the assigned frame is added to the list of PFTEs representing frames backing the private portion of the address space for which the translation exception was recognized:
    a. Depending on the address space for which the translation exception was recognized, the primary or secondary ASN in bits 16-31 of control register 3 or 4 is placed in PFTASID at temp2.
    b. The ASCB of the space for which the translation exception was recognized is located by adding 4 times the ASN for the space to the contents of CVTASVT, plus 20C hex (the size of the ASVT header), giving the address of the entry in the ASVT that contains the address of the ASCB.

c. If the contents of RSMLFQL are not zero, temp1 is placed in PFTNXRBN in a private PFTE. The address of PFTNXRBN is the sum of 4 plus RSMLFQL times 16 plus PVTPFTP.

The remaining actions are performed in any order:

d. ASCBFMCT + 1 is placed in ASCBFMCT.
e. The value 80 hex is placed in PFTQNDX at temp2 + 14.
f. RSMLFQL is placed in PFTPXRBN at temp2 + 6, and temp1 is then placed in RSMLFQL.
g. If the contents of RSMLFQF are zero, temp1 is placed in RSMLFQF.

11. Zeros are stored in the salloc word, and PSASALLI and PSAPFA are set to zero. Execution ends without the instruction-address part of the current PSW being incremented. Instruction execution is resumed with the instruction for which the translation exception was recognized.

STO and ASN Sources

The sources of the segment-table-origin (STO) and address-space-number (ASN) values depend on whether the translation-exception address is primary virtual or secondary virtual, as follows:

Translation-Exception Address	STO	ASN
Primary virtual	Control register 1	Bits 16-31, control register 4
Secondary virtual	Control register 7	Bits 16-31, control register 3

Address Types

The addresses of the ASVT, the ASCB, the RSMHD, the CVT, the PVT, PFT entries, the salloc word, and fields with addresses in the range 0-4095, but not including the interruption-parameter fields, are treated as logical–either primary-virtual or secondary-virtual, depending on PSW bit 16 when DAT is on. The addresses of the following are treated as 24-bit real addresses: STE, PTE, XPTE, and the interruption-parameter fields. The address of the assigned real page frame is generated as a 26-bit real address. All accesses are performed as if the applicable protection key were zero.

Access Exceptions

Accesses by the page-fault assist are not subject to key-controlled protection. Low-address protection applies to all store accesses.

Stores performed while the salloc word is held are not necessarily of the single-access type.

An access exception, or a boundary misalignment, that is encountered while the assist is being executed is handled in one of two ways:

1. The assist is nullified, and the interruption is taken for the exception that caused the assist to be invoked.
2. Some of the actions of the assist may be complete, but the interruption for the page-translation exception that caused the assist to be invoked is taken, except that the interruption code is set to 26 hex. This is called the page-fault-assist exception code.

Because of preconditions, verified in step 1 in the section "Page-Fault Assist," a PER event cannot accompany this exception.

Control-Block Alignment

When a control block or table entry is not aligned on the appropriate boundary, the page-fault assist ends as described in the section "Access Exceptions."

Index

GA22-7079-1

IBM

IBM System/370
Assists for MVS
Order No. GA22-7079-1

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. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Your comments will be sent to the author's department for whatever review and action, if any, are deemed appropriate. Comments may be written in your own language; English is not required.

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, company, mailing address, and date:

_____

_____

_____

_____

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.)

GA22-7079-1

Reader's Comment Form

Cut or Fold Along Line

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

Cut or Fold Along Line

IBM®

IBM®

Printed in U.S.A.

GA22-7079-01