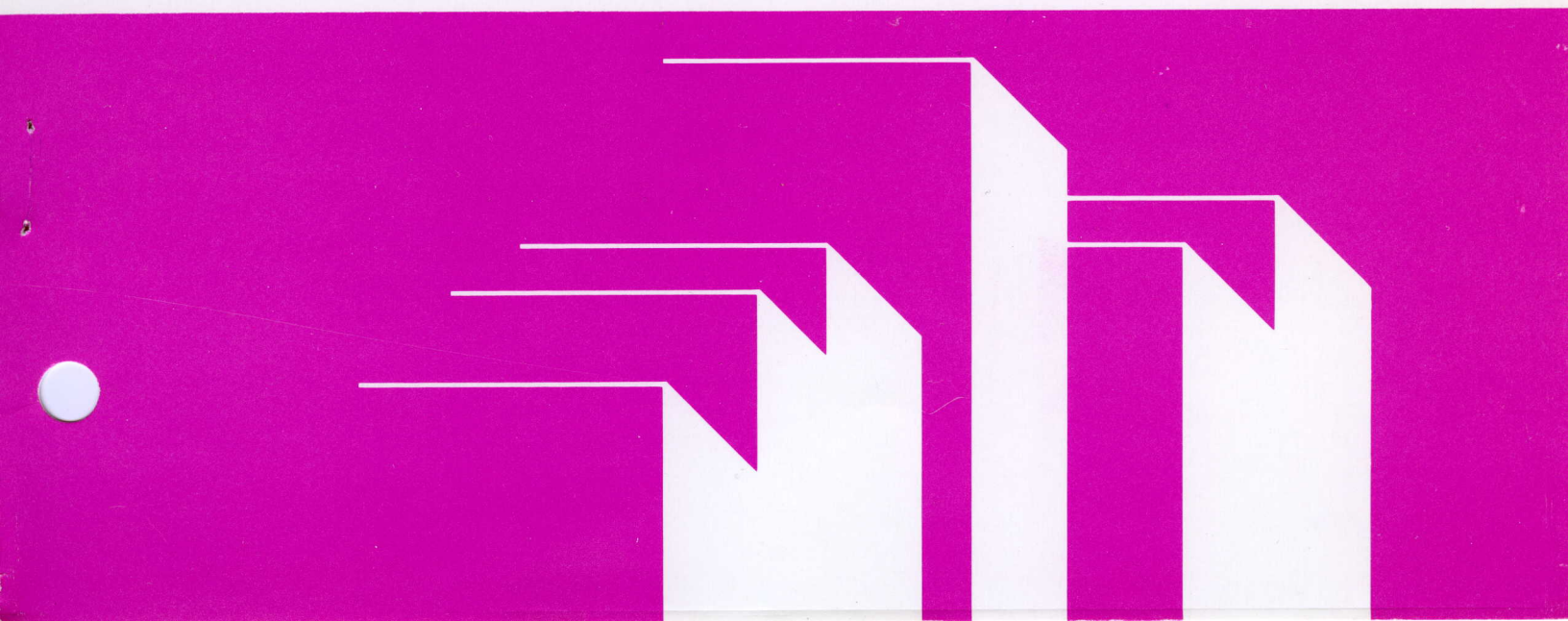
IBM Assists for MVS/XA

IBM



IBM Assists for MVS/XA

Publication Number
SA22-7092-0

File Number
S370-01

PREFACE

This publication is intended for system programmers and IBM Field Engineering personnel. The reader should be familiar with the general machine functions of 370-XA, as described in the IBM 370-XA Principles of Operation, SA22-7085, and with the MVS/SP Version 2 Licensed Program, referred to in this publication as the MVS/XA control program. The standard names for MVS/XA control blocks are used throughout the publication. The formats of these control blocks are described in the MVS/XA Debugging Handbook, Volumes 1 through 5, LC28-1164 through LC28-1168, respectively.

First Edition (March 1983)

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.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM program product in this publication is not intended to state or imply that only IBM's program product may be used. Any functionally equivalent program may be used instead.

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, Department B98, PO Box 390, Poughkeepsie, NY, U.S.A. 12602. IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever.

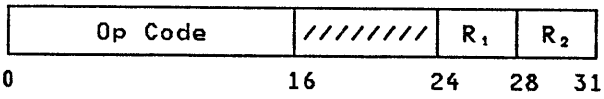
CONTENTS

This publication describes six instructions that depend on control blocks whose formats and relationships normally are established only by the MVS/XA control program. These instructions improve performance in frequently used parts of the MVS/XA control program on machines which provide the 370-XA architectural mode and these instructions. Four of the instructions assist the handling of control-program locks. A locking protocol is used by the MVS/XA control program to achieve serial use of control-program resources when multiprogramming and multiprocessing activities may cause temporary, multiple demands for the same resource. One instruction assists the execution of the SUPERVISOR CALL instruction, which is frequently used to request control-program services. The remaining instruction helps in establishing parameters that are used by the MVS/XA control program following detection of certain failure and error conditions. The six instructions are:

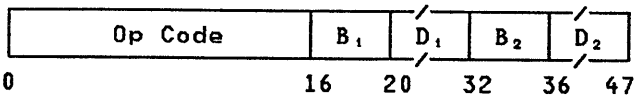
ADD FRR
OBTAIN CMS LOCK
OBTAIN LOCAL LOCK
RELEASE CMS LOCK
RELEASE LOCAL LOCK
SVC ASSIST

All of the instructions are privileged instructions.

ADD FRR uses the RRE instruction format:



The other instructions use the SSE format. The SSE instruction format is:



In addition to the operands explicitly designated by the instructions, implicit operands at fixed storage locations are also used. Sometimes operands fetched from storage are used in turn for addressing still other operands. Unless otherwise stated, all operand addresses are logical. Either 24 or 31 bits are used for addressing, depending on the value of the A-mode bit in the current PSW. The leftmost eight bits or the leftmost bit, depending on the A-mode bit in the current PSW, are ignored in a word which is a source of an operand address or a branch address. Storage protection applies to all storage operand accesses in the usual way. A specification exception is recognized for any operand that does not meet a specified alignment requirement, and instruction execution is suppressed.

Not all possible operands are accessed for every execution of an instruction. In some cases the state of control bits or the results of comparisons of operands determine which operands are subsequently accessed. Exceptions may or may not be recognized for operands not needed for completion of a particular execution of an instruction, except where the definition of a particular instruction notes restrictions.

Program-event recording applies in the usual fashion to these instructions, except that branch events are not recognized when these instructions cause branching.

The two instructions which obtain locks fetch and test a word for zeros, and, if the fetched word is zero, cause a specified value to be stored back at the same location by means of an interlocked update. The two instructions which release locks may perform a conditional interlocked update of a doubleword location. For these instructions, if the second word of a doubleword location contains all zeros, the entire doubleword is set to zeros by means of an interlocked update. When the SVC ASSIST instruction is used to assist a type-1 SVC, a conditional interlocked update of a doubleword location is performed which is similar to that performed by the COMPARE DOUBLE AND SWAP instruction.

The four lock-handling instructions may, under certain conditions, access the lock-interface-table prefix, consisting of the four words at negative offsets from the lock-interface table. The lock-interface table is located by using an address contained in the word in main storage following the second operand.

SIMPLIFIED EXECUTION PATHS

Simplified execution paths are defined for the following 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 CMS LOCK. Execution proceeds as if a CMS lock were already held.
- OBTAIN LOCAL LOCK. Execution proceeds as if the local lock were already held.
- RELEASE CMS LOCK. Execution proceeds as if the currently dispatched unit of work held no CMS lock.
- RELEASE LOCAL LOCK. Execution proceeds as if the local lock were already released.
- 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 available control-storage space is limited. The ADD FRR instruction is optional and may or may not be provided as part of the assists.

INSTRUCTIONS

The instructions described in this section are listed in the figure "Instruction Summary," as are 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
ADD FRR	RRE	M	A	SP	R	OP	ST	B242
OBTAIN CMS LOCK	SSE	M	A	SP	R		ST	E506
OBTAIN LOCAL LOCK	SSE	M	A	SP	R		ST	E504
RELEASE CMS LOCK	SSE	M	A	SP	R		ST	E507
RELEASE LOCAL LOCK	SSE	M	A	SP	R		ST	E505
SVC ASSIST	SSE	M	A	SP	R		ST	E503

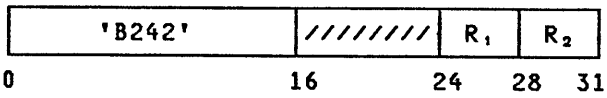
Explanation:

A	Access exceptions
M	Privileged-operation exception
OP	Operation exception
R	PER general-register-alteration event
RRE	RRE instruction format
SP	Specification exception
SSE	SSE instruction format
ST	PER storage-alteration event

Instruction Summary

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, with the PSW A-mode bit (bit 32), 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 new 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.

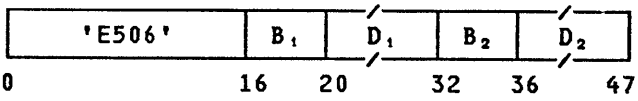
Condition Code: The code remains unchanged.

Program Exceptions:

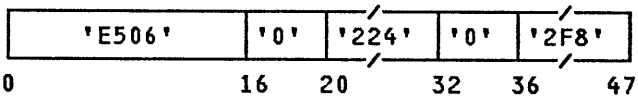
Access (storage operands)
Operation (when the instruction is not installed)
Privileged operation
Specification

OBTAIN CMS LOCK

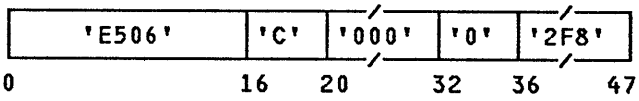
General Form



Forms Used in the Control Program



or



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

Otherwise, the updated instruction address is placed in general register 12, with a zero placed in bit position 0; the contents of the word at LIT minus 8 are placed in general register 13, and bits are selected from the word which replace the instruction-address portion of the PSW, leaving the A-mode bit unchanged.

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

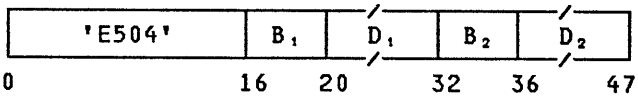
Condition Code: The code remains unchanged.

Program Exceptions:

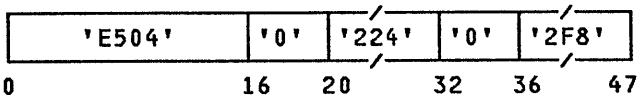
Access
Privileged operation
Specification

OBTAIN LOCAL LOCK

General Form



Form Used in the Control Program



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 the word at the second-operand location minus 4; the local-lock bit of the word fetched from the second-operand location is set to one; and zeros are placed in general register 13.

Otherwise, the updated instruction address is placed in general register 12, with a zero placed in bit position 0; the contents of the word at LIT minus 16 are placed in general register 13, and bits are selected from the word which replace the instruction-address portion of the PSW, leaving the A-mode bit unchanged.

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

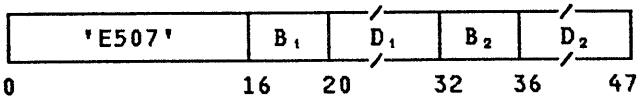
Condition Code: The code remains unchanged.

Program Exceptions:

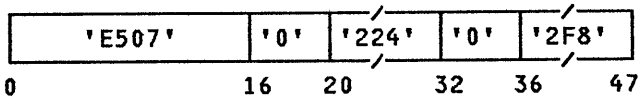
Access
Privileged operation
Specification

RELEASE CMS LOCK

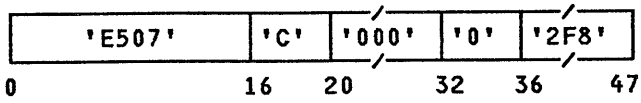
General Form



Forms Used in the Control Program



or



If (1) the contents of the CMS lockword addressed by 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 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 is placed in general register 12, with a zero placed in bit position 0; the contents of the word at LIT minus 4 are placed in general register 13; and bits are selected from the word which replace the instruction-address portion of the PSW, leaving the A-mode bit unchanged.

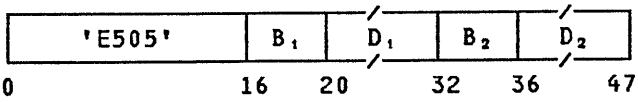
Condition Code: The code remains unchanged.

Program Exceptions:

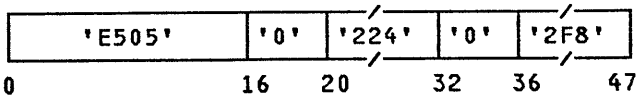
Access
Privileged operation
Specification

RELEASE LOCAL LOCK

General Form



Form Used in the Control Program



If the 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 inter locked 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 is placed in general register 12, with a zero placed in bit position 0; the contents of the word at LIT minus 12 are placed in general register 13, and bits are selected from the word which replace the instruction-address portion of the PSW, leaving the A-mode bit unchanged.

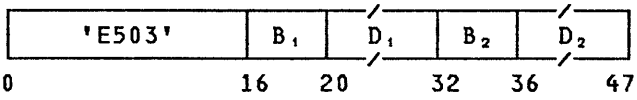
Condition Code: The code remains unchanged.

Program Exceptions:

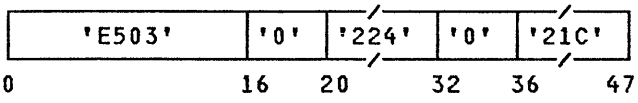
Access
Privileged operation
Specification

SVC ASSIST

General Form



Form Used in the Control Program



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

Main-storage locations which contain 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. In addition, no assist action is taken unless (1) both the primary space and the secondary space match the dispatched 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 below, and then loading the instruction address portion of the PSW from 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 MPL field which is appropriate to the SVC type, that is, the address of the routine that receives control on completion of the assist
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 MPL field which is appropriate to the SVC type

Condition Code: The code remains unchanged.

Program Exceptions:

Access
Privileged operation
Specification

IBM Assists for MVS/XA
Order No. SA22-7092-0

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. 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.

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:

_____

_____

_____

_____

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.)

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

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, N.Y.

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®

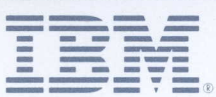
Cut or Fold Along Line

IBM Assists for MVS/XA (File No. S370-01) Printed in U.S.A. SA22-7092-0

Publication Number
SA22-7092-0

File Number
S370-01

Printed in
USA



SA22-7092-0