z/VM
Version 7 Release 2

Enterprise Systems Architecture/ Extended Configuration Principles of Operation

SC24-6285-01

Note:

Before you use this information and the product it supports, read the information in "Notices" on page 57.

This edition applies to Version 7.2 of IBM z/VM (product number 5741-A09) and to all subsequent releases and modifications until otherwise indicated in new editions.

Last updated: 2020-09-08

Contents

Figures

Tables

About This Document

This document provides a detailed description of the IBM® Enterprise Systems Architecture/Extended Configuration (ESA/XC) virtual-machine architecture, as provided by z/VM®. It describes the way an ESA/XC virtual machine operates as it appears to an assembler language programmer. Because ESA/XC is based on and is closely related to Enterprise Systems Architecture/390® (ESA/390), this document defines ESA/XC by indicating the ways in which it is the same as, or different from, ESA/390.

The following elements of the ESA/XC architecture are covered in this document:

- Overall organization
- The structure of storage and address spaces
- Control facilities
- Program execution
- Interruptions
- The operation of instructions
- Input/output facilities

Intended Audience

This document is intended for programmers who write or debug programs that run in ESA/XC virtual machines.

You should have a basic familiarity with the ESA/390 architecture, or alternatively, the Enterprise Systems Architecture/370 (ESA/370) or System/370 Extended Architecture (370-XA) upon which ESA/390 is based. These architectures are described in:

- *IBM Enterprise Systems Architecture/390 Principles of Operation*, SA22-7201
- *IBM Enterprise Systems Architecture/370 Principles of Operation*, SA22-7200
- *IBM System/370 Extended Architecture Principles of Operation*, SA22-7085

You should also know IBM basic assembler language and have experience with z/VM programming concepts and techniques.

Conventions

This document is intended to be used in conjunction with the definition of ESA/390 provided in *IBM Enterprise Systems Architecture/390 Principles of Operation*. Where possible, information is presented in this document using the same style and general organization as *IBM Enterprise Systems Architecture/390 Principles of Operation*. To assist in locating corresponding information in the two documents, the following table shows the relationship of the chapters of this document to the chapters in *IBM Enterprise Systems Architecture/390 Principles of Operation*.

Chapter in this document	**Corresponding chapter(s) in *IBM Enterprise Systems Architecture/390 Principles of Operation*, SA22-7201-08**
Chapter 1, “Introduction,” on page 1	Chapter 1, Introduction
Chapter 2, “Organization,” on page 5	Chapter 2, Organization
Chapter 3, “Storage,” on page 7	Chapter 3, Storage

Chapter in this document	Corresponding chapter(s) in *IBM Enterprise Systems Architecture/390 Principles of Operation*, SA22-7201-08
Chapter 4, “Control,” on page 17	Chapter 4, Control
Chapter 5, “Program Execution,” on page 21	Chapter 5, Program Execution
Chapter 6, “Interruptions,” on page 33	Chapter 6, Interruptions
Chapter 7, “Instructions,” on page 39	Chapter 7, General Instructions Chapter 8, Decimal Instructions Chapter 9, Floating-Point Overview and Support Instructions Chapter 10, Control Instructions Chapter 18, Hexadecimal-Floating-Point Instructions Chapter 19, Binary-Floating-Point Instructions
Chapter 8, “Machine-Check Handling,” on page 47	Chapter 11, Machine-Check Handling
Chapter 9, “Input/Output,” on page 49	Chapter 13, I/O Overview Chapter 14, I/O Instructions Chapter 15, Basic I/O Functions Chapter 16, I/O Interruptions Chapter 17, I/O Support Functions
Note: There is no material in this document corresponding to Chapter 12, "Operator Facilities" of *IBM Enterprise Systems Architecture/390 Principles of Operation*.	

Where to Find More Information

Besides the document *IBM Enterprise Systems Architecture/390 Principles of Operation*, the following documents in the z/VM library may be useful in understanding ESA/XC:

- *z/VM: CP Programming Services*
- *z/VM: CMS Application Development Guide*
- *z/VM: CMS Callable Services Reference*
- *z/VM: CMS Application Development Guide for Assembler*
- *z/VM: CMS Macros and Functions Reference*

For the complete list of documents in the z/VM library, see the “Bibliography” on page 61.

Links to Other Documents and Websites

The PDF version of this document contains links to other documents and websites. A link from this document to another document works only when both documents are in the same directory or database, and a link to a website works only if you have access to the Internet. A document link is to a specific edition. If a new edition of a linked document has been published since the publication of this document, the linked document might not be the latest edition.

How to Send Your Comments to IBM

We appreciate your input on this publication. Feel free to comment on the clarity, accuracy, and completeness of the information or give us any other feedback that you might have.

To send us your comments, go to z/VM Reader's Comment Form (www.ibm.com/systems/campaignmail/z/zvm/zvm-comments) and complete the form.

If You Have a Technical Problem

Do not use the feedback method. Instead, do one of the following:

- Contact your IBM® service representative.
- Contact IBM technical support.
- See IBM: z/VM Support Resources (www.ibm.com/vm/service).
- Go to IBM Support Portal (www.ibm.com/support/entry/portal/Overview).

Summary of Changes for z/VM ESA/Extended Configuration Principles of Operation

This information includes terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations for the current edition are indicated by a vertical line to the left of the change.

SC24-6285-01, z/VM 7.2 (September 2020)

This edition supports the general availability of z/VM® 7.2.

SC24-6285-00, z/VM 7.1 (September 2018)

This edition supports the general availability of z/VM 7.1.

SC24-6192-01, z/VM 6.4 (August 2017)

This edition includes changes to support product changes provided or announced after the general availability of z/VM 6.4.

Chapter 1. Introduction

This document describes, for reference purposes, the operation of virtual machines that execute in the ESA/XC virtual-machine architecture as provided by z/VM. It is organized as a supplement to the definition of the ESA/390 architecture described in *IBM Enterprise Systems Architecture/390 Principles of Operation*.

If a particular element of the architecture (instruction, operation, program exception, etc.) is described in this document, then the definition contained in this document supersedes the definition in *IBM Enterprise Systems Architecture/390 Principles of Operation*. If there is no discussion of a particular element of the architecture in this document, then the definition contained in *IBM Enterprise Systems Architecture/390 Principles of Operation* applies to ESA/XC as well.

ESA/XC comprises the architectural facilities available to the program that is executing in the virtual machine. ESA/XC virtual machines operate under the control of the z/VM Control Program (CP), a host program that runs in the ESA/390 processor complex and manages the structures that establish the virtual machines. The details of these structures do not directly affect the program in the virtual machine and are therefore not described in this document.

Highlights of ESA/XC

The ESA/XC virtual-machine architecture is a derivative of Enterprise Systems Architecture/390 (ESA/390) that is designed for the DAT-off virtual-machine environment. This virtual-machine architecture includes most of the facilities of ESA/390 and offers major new capabilities.

ESA/XC was designed with special emphasis on the requirements of the interactive and service-virtual-machine environments of z/VM. In these environments, the supervisor running in the virtual machine is not an elaborate control program, but rather an application-program monitor. Consistent with such a virtual-machine supervisor, ESA/XC allows application programs to benefit from architectural extensions provided by ESA/390 without requiring the virtual-machine supervisor to perform complex management functions. Instead, the host bears the responsibility for the support and management of the real-machine facilities.

ESA/XC is also designed to take into account the multiple-virtual-machine environment of z/VM. By giving the responsibility for support and control of the architecture to the host, the scope of access that can be allowed using the architecture can be extended in a controlled and secure manner across multiple virtual machines. Previously, the scope of access possible using architectural facilities was necessarily limited to a single virtual machine.

When compared to ESA/390, ESA/XC offers the following extensions:

- *DAT-off access-register addressing* makes the access-register-addressing facility of ESA/370 and ESA/390 available to z/VM applications. The facility can be used by a virtual machine to access its own address spaces as well as the address spaces of other virtual machines, subject to appropriate authorization.
- *Multiple absolute-storage address spaces* significantly extend the amount of storage available to z/VM applications. Previously, the storage available to an application was limited to a single 2 gigabyte absolute-storage address space. With this facility, an application can have multiple 2 gigabyte data-only spaces in addition to the instruction space, providing greatly increased application storage.
- *Address-space sharing* provides a basis for high-performance data sharing within a z/VM system. A virtual machine can authorize other virtual machines to access, by means of access-register addressing, any of its address spaces (the instruction address space, or any data-only address spaces). The shared access can be subsequently revoked when the shared access is no longer appropriate.

The ESA/390 Base

ESA/XC includes, as its base, most of the facilities available to ESA/390 virtual machines. However, some of the facilities available in ESA/390 are not provided in ESA/XC. This section summarizes these differences.

The most significant difference between ESA/390 and ESA/XC is that in ESA/XC, guest dynamic address translation (DAT) is not provided. Consequently, in ESA/XC:

- Guest virtual-storage address spaces are not provided. Therefore, there are no guest segment tables or page tables, nor is there a guest translation-lookaside buffer.
- The dual-address-space facility is not provided. Therefore, the EXTRACT PRIMARY ASN, EXTRACT SECONDARY ASN, INSERT VIRTUAL STORAGE KEY, LOAD ADDRESS SPACE PARAMETERS, MOVE TO PRIMARY, MOVE TO SECONDARY, PROGRAM CALL, PROGRAM TRANSFER and SET SECONDARY ASN instructions are not provided. The ASN-translation and PC-number translation processes are never performed, and consequently, there are no guest ASN first tables, ASN second tables, linkage tables and entry tables. There are no guest primary and secondary spaces, nor is the secondary-space mode provided.
- Guest access-register translation is not provided. Therefore, there are no guest access lists, nor dispatchable-unit-control tables. A guest ART-lookaside buffer is not provided. (Host access-register translation is provided as a comparable replacement for guest access-register translation.)
- The linkage stack is not provided, nor are the instructions BRANCH AND STACK, EXTRACT STACKED REGISTERS, EXTRACT STACKED STATE, MODIFY STACKED STATE, and PROGRAM RETURN.
- The home address space is not provided, nor is the home-space mode.
- The private-space facility is not provided.

In addition, ESA/XC does not provide the instructions LOAD REAL ADDRESS, SET CLOCK, and START INTERPRETIVE EXECUTION.

In contrast, the facilities previously mentioned as well as extensions to them are available to the host program. Since this document provides in detail only the architecture of the virtual-machine environment, reference is made to host facilities only as needed to clearly describe the programming environment provided by the virtual machine.

Except for facilities specifically identified in this document as not provided, ESA/XC includes all facilities that are defined in ESA/390.

System Program

ESA/XC is typically used with a virtual-machine supervisor program, such as CMS, that cooperates with the host to provide application-level service interfaces for application programs executed within a single virtual machine. An ESA/XC virtual machine operates under the control of the z/VM Control Program (CP) running in the ESA/390 processor complex. The z/VM Control Program acts as the host program, managing the execution of virtual machines and providing system-level services to the virtual machines.

Compatibility

Compatibility among ESA/XC Implementations

ESA/XC virtual machines can be provided when z/VM is operating on different real-processor implementations, and may be provided by different implementations of the host program. These different implementations of ESA/XC are logically compatible. Specifically, any program written for ESA/XC gives identical results on any ESA/XC implementation, provided that the program:

1. Is not time dependent.

2. Does not depend on system facilities (such as storage capacity, I/O equipment, optional machine facilities or optional or release-dependent host-program facilities) being present when the facilities are not included in the configuration or are not provided by the host program implementation.
3. Does not depend on system facilities being absent when the facilities are included in the configuration. For example, the program must not depend on interruptions caused by the use of operation codes or command codes that are not installed in some real-processor models, or not provided by some host program versions. Also, it must not use or depend on fields associated with uninstalled facilities. For example, data should not be placed in an area used by another model for fixed-logout information. Similarly, the program must not use or depend on unassigned fields in machine formats (control registers, instruction formats, etc.) that are not explicitly made available for program use.
4. Does not depend on results or functions that are defined to be unpredictable or model-dependent or are identified as undefined. This includes the requirement that the program should not depend on the assignment of device numbers and CPU addresses.
5. Does not depend on results or functions that are defined in the functional-characteristics publication for a particular real-processor model to be deviations from the ESA/390 architecture where those results or functions of ESA/390 are applicable in ESA/XC.
6. Takes into account any changes made to the architecture that are identified as affecting compatibility.

Compatibility among ESA/XC, ESA/390, ESA/370, 370-XA and System/370

Supervisor-State Compatibility

A supervisor-state program written for 370-XA, ESA/370 or ESA/390 that uses the dynamic-address-translation (DAT) facility cannot be transferred to an ESA/XC virtual machine because ESA/XC does not provide DAT.

A supervisor-state program written for 370-XA, ESA/370 or ESA/390 that does not use the DAT facilities can be transferred to an ESA/XC virtual machine, provided that the control program:

1. Complies with the limitations described in the section "Compatibility among ESA/XC Implementations" on page 2 in this chapter.
2. Does not depend for correct operation on receiving a special-operation exception when a control instruction that in ESA/390 requires DAT to be on for successful execution is attempted in an ESA/XC virtual machine.
3. Does not use the following instructions:
   - LOAD ADDRESS SPACE PARAMETERS
   - LOAD REAL ADDRESS
   - SET CLOCK
   - START INTERPRETIVE EXECUTION

Supervisor-state programs written for System/370*, regardless of whether the DAT facility is used, cannot be directly transferred to virtual machines operating as defined by ESA/XC. This is because the basic-control mode is not present and the facilities for I/O are changed in the ESA/390 base of ESA/XC. (See Appendixes D, E and F of *IBM Enterprise Systems Architecture/390 Principles of Operation* for a detailed comparison among ESA/390, ESA/370, 370-XA and System/370. See Appendix A, "Comparison between ESA/390 and ESA/XC," on page 51 of this publication for a detailed comparison between ESA/XC and ESA/390.)

Problem-State Compatibility

A high degree of compatibility exists at the problem-state level in transferring from ESA/390, ESA/370, 370-XA or System/370 to ESA/XC operation.

A problem-state program written for ESA/390, ESA/370, 370-XA or System/370 operates with ESA/XC provided that the program:

1. Complies with the limitations described in the section "Compatibility among ESA/XC Implementations" on page 2 in this chapter.
2. Is not dependent on host or virtual-machine supervisor facilities that are unavailable on the system.
3. Takes into account other changes made to the System/370 architectural definition that affect compatibility between System/370 and the 370-XA base of ESA/390. These changes are described in Appendix F of *IBM Enterprise Systems Architecture/390 Principles of Operation*.

Relationship to z/Architecture

z/Architecture™ extends ESA/390 by providing 64-bit addressing registers and arithmetic operations. z/VM runs only in z/Architecture (64-bit) mode. However, ESA/XC remains a derivative of ESA/390. ESA/XC virtual machines are supported, but are limited to 31-bit addressing and 32-bit registers and arithmetic operations. An ESA/XC virtual machine cannot operate in 64-bit mode.

Chapter 2. Organization

The basic organization of an ESA/XC virtual machine follows the definition of system organization provided in Chapter 2 of *ESA/390 Principles of Operation*. In addition, an ESA/XC virtual machine has an extended configuration that is described by the information in this chapter.

An ESA/XC virtual-machine configuration contains all of the basic organizational elements defined for ESA/390 systems: main storage, one or more central processing units (CPUs), operator facilities, a channel subsystem and I/O devices. In addition, an ESA/XC configuration has available additional separate absolute-storage address spaces, up to 15 of which are concurrently addressable as provided by access-register addressing. These 15 absolute-storage address spaces are selected from among a larger set as determined by a table called a host access list.

Certain elements of an ESA/XC virtual-machine configuration are regulated by host controls available to the z/VM installation. For example, the total main storage provided to a virtual machine is regulated by host controls. These host controls are specified in the host (CP) directory entry for the virtual machine.

Contrary to the ESA/390 definition, the main storage of an ESA/XC virtual machine is not necessarily isolated from access by the CPUs of other virtual machines. By using host services, the main storage of one virtual machine can be made directly addressable by the CPUs of another configuration. This shared main storage can be used to provide shared data for a collection of virtual machines executing on a single z/VM system.

Main Storage

As in ESA/390, directly addressable main storage is provided for high-speed processing of data by the CPU and the channel subsystem.

Main storage is allocated in extents known as absolute-storage address spaces, or simply address spaces. When a virtual machine is created by the host, an initial address space is provided for the virtual machine; this address space is known as the host-primary address space. Initially, the host-primary address space is directly addressable only by the CPUs of the associated virtual machine. However, through the use of host services, the host-primary address space may be made directly addressable by the CPUs of other virtual machines.

Optionally, additional absolute-storage address spaces may be added to the virtual configuration by means of host services. These additional address spaces can be established as directly addressable by the CPUs of the associated virtual machine and by the CPUs of other virtual machines as appropriate.

Central Processing Unit

A central processing unit (CPU) of an ESA/XC virtual configuration includes all of the registers provided in ESA/390: the PSW, general registers, floating-point registers, control registers, access registers, the prefix register, and the registers for the clock comparator and CPU timer.

Access Registers

As in ESA/390, ESA/XC provides 16 access registers numbered 0-15. An access register contains an indirect specification of an absolute-storage address space. When the CPU is in a mode called the access-register mode (controlled by a bit in the PSW), an instruction B field used to specify a logical or real address for a storage-operand reference designates an access register. The storage-operand is considered to reside within the absolute-storage address space specified by the access register. For some instructions, an R field is used instead of a B field. Instructions are provided for loading and storing the contents of access registers and for moving the contents of one access register to another.

Each of access registers 1-15 can designate any address space, including the instruction space (the host-primary address space). Access register 0 always designates the instruction space. When one of the access registers 1-15 is used to designate an address space, the CPU determines which address space is designated by translating the contents of the access register using host-managed tables. When access register 0 is used to designate an address space, the CPU treats the access register as designating the instruction space, and it does not examine the actual contents of the access register. Therefore, the 16 access registers can designate, at any one time, the instruction space and a maximum of 15 other address spaces.

Host Access List

An ESA/XC virtual machine has a host access list that specifies the set of address spaces, in addition to the virtual machine's host-primary address space, that are available to a CPU when it is in the access-register mode. The host access list contains a directory-specified number of host access-list entries, each of which either designates a particular address space or is considered unused.

Through the use of host services, a host access-list entry can be set to designate a particular address space or can be returned to the unused state.

Host Program

An ESA/XC virtual machine operates under the control of the z/VM Control Program (CP); this supervisory program is called the host program, or simply the host. The host runs in z/Architecture mode or the ESA/390 processor complex and manages the execution of virtual machines, possibly of varying architectures. It also provides special service functions in addition to the facilities described by the architecture(s).

The extended addressing capabilities of ESA/XC are under the control of the host, which determines the extent to which a particular ESA/XC virtual machine can use these extended capabilities. Services are provided by which an ESA/XC virtual machine can request that its capabilities be altered. These services include the creation and deletion of additional address spaces and the modification of host access lists. These host services are described in detail in the publication *z/VM: CP Programming Services*.

Chapter 3. Storage

This chapter describes the structure of main storage, including addressing and protection aspects, for an ESA/XC virtual machine. Except as described in this chapter, the handling of storage for an ESA/XC virtual machine follows the definition of storage provided in Chapter 3 of *IBM Enterprise Systems Architecture/390 Principles of Operation*.

Note: Because most references to storage for an ESA/XC virtual machine apply to (guest) real storage, the abbreviated term "storage" is often used in place of "real storage". The term "storage" may also be used in place of "main storage" or "absolute storage" when the meaning is clear.

Storage Addressing

The addressable storage of an ESA/XC virtual machine consists of one or more extents of main storage known as absolute-storage address spaces. The storage contained within a single absolute-storage address space is viewed as a long horizontal string of bits, subdivided into bytes as is usual for ESA/390.

Absolute-Storage Address Spaces

An absolute-storage address space is a single extent of main storage that is directly addressable by a CPU or the channel subsystem. An absolute-storage address space consists of a collection of byte locations, and a set of byte addresses assigned to those byte locations.

Note: Because all references to address spaces for an ESA/XC virtual machine apply to absolute-storage address space and not to virtual-storage address spaces, the abbreviated term "address space" is often used in place of "absolute-storage address space".

Each byte location in storage is identified by a non-negative integer which is the byte address of the storage location within an absolute-storage address space. Byte locations are assigned addresses starting at 0, and the addresses are always assigned in complete 4K-byte blocks on integral boundaries.

A byte address uniquely identifies a byte within the collection of bytes associated with a specific absolute-storage address space. However, a particular value of a byte address may not identify a unique byte since a particular byte address may be associated with more than one absolute-storage address space.

A virtual machine has at least one absolute-storage address space, known as its host-primary address space. This address space is provided by the host when the virtual machine is created.

A virtual machine may obtain additional absolute-storage address spaces by using the ADRSPACE CREATE host service. These additional absolute-storage address spaces may be subsequently destroyed by using the ADRSPACE DESTROY host service. Subsystem reset will also destroy all absolute-storage address spaces created by the ADRSPACE CREATE service.

In an absolute-storage address space created by means of the ADRSPACE CREATE host service, addresses are assigned in a single contiguous range. However, in the host-primary address space of a virtual machine, there may be multiple discontiguous ranges of assigned addresses.

The number and total size of absolute-storage address spaces permitted for the virtual machine is subject to directory-specified limits.

Programming Note: The ADRSPACE CREATE host service is described in the publication *z/VM: CP Programming Services*. The number and total size of address spaces that the virtual machine can create by means of ADRSPACE CREATE is specified by the XCONFIG ADDRSPACE statement in the CP directory. The size of the virtual machine's host-primary address space is specified by the USER or IDENTITY statement in the CP directory. These three directory statements are described in the publication *z/VM: CP Planning and Administration*.

Private and Shareable Address Spaces

An absolute-storage address space is considered to be either a private address space, or a shareable address space, as follows.

An address space is a private address space if the address space is directly addressable only by the CPUs of a single virtual machine. An address space is a shareable address space if the address space can be directly addressed by the CPUs of more than one virtual machine.

Immediately after an address space is created by the host, the address space is a private address space. Through the use of the ADRSPACE PERMIT host service, a private address space may be transformed into a shareable address space, subject to a directory-specified authorization to share address spaces. Subsequently, through the use of the ADRSPACE ISOLATE host service, a shareable address space can be transformed into a private address space. Subsystem reset will also transform a shareable address space into a private address space.

Programming Note: The ADRSPACE PERMIT and ADRSPACE ISOLATE host services are described in the publication *z/VM: CP Programming Services*.

Identification of Address Spaces

As a mechanism for differentiating one address space from another, each address space has associated with it an eight-byte identifying value called an address-space identification token, or ASIT.

The host assigns an ASIT to each address space when the address space is created. The ASIT associated with a particular absolute-storage address space is fixed from the time the address space is created by the host until it is subsequently destroyed.

The host assigns ASITs in such a way that a particular ASIT value will be associated with at most one absolute-storage address space for the scope of the host IPL. That is, once a particular ASIT value has been assigned to an absolute-storage address space, that ASIT value will not be reassigned to another absolute-storage address space within the scope of the same host IPL, even if the absolute-storage address space to which the ASIT value was originally assigned is subsequently destroyed. ASIT values are not guaranteed to be assigned in any particular manner across a host IPL.

The value 0000000000000000 hex is never assigned as an ASIT.

The specific format of an ASIT is undefined. In general, a particular ASIT value will have no significance to ESA/XC programs.

Address Types and Formats

Address Types

For purposes of addressing main storage, two basic types of addresses are recognized: absolute and real. The addresses are distinguished on the basis of the transformations that are applied to the address during a storage access. In addition to the two basic address types, additional types are defined which are treated as one or another of the two basic types depending on the instruction and the current mode.

Absolute Address

An absolute address is the address assigned to a main-storage location within a particular address space. An absolute address is used for access to storage contained in a particular address space without any transformations performed on it.

Host-Primary Absolute Address

A host-primary absolute address is an absolute address that identifies a location contained within the host-primary address space.

AR-Specified Absolute Address

An AR-specified absolute address is an absolute address that identifies a location contained within an address space determined by host access-register translation.

Real Address

A real address identifies a location within a real address space. In ESA/XC, each real address is considered to be one of two types: type-R or type-A. These types determine the way in which the real address is converted into an absolute address. They also determine the applicability of low-address protection and fetch protection to references using the real address.

When a type-R real address is used for access to main storage, it is converted by means of prefixing to an absolute address. Low-address protection is applied to storage references made using a type-R real address subject to the setting of the low-address-protection control in control register 0. Fetch-protection checking is performed for storage references made using a type-R real address subject to the setting of the fetch-protection-override control in control register 0.

When a type-A real address is used for access to main storage, it is treated unchanged as an absolute address; no prefixing is applied. Low-address protection is never applied to a storage references made using a type-A real address, regardless of the setting of the low-address-protection control in control register 0. Fetch-protection checking is always applied to a storage reference made using a type-A real address, regardless of the setting of the fetch-protection-override control in control register 0.

The set of byte locations with a single absolute-storage address space sequenced according to their real addresses is referred to as a real address space.

Host-Primary Real Address

A host-primary real address is a real address that identifies a location contained within the host-primary address space. A host-primary real address is considered to be a type-R real address.

AR-Specified Real Address

An AR-specified real address is a real address that identifies a location contained within an address space determined by host access-register translation. An AR-specified real address is considered to be a type-R real address when the ALET used is 00000000 hex. An AR-specified real address is considered to be a type-A real address when the ALET used is other than 00000000 hex.

Virtual Addresses

Since the dynamic-address-translation facility is not provided in ESA/XC, virtual addresses and virtual storage are not available for ESA/XC virtual machines.

Logical Addresses

Except where otherwise specified, the storage-operand addresses for instructions are logical addresses. Logical addresses are treated as host-primary real addresses when in the primary-space mode, and as AR-specified real addresses when in the access-register mode. Some instructions have storage operand addresses or storage accesses associated with the instruction which do not follow the rules for logical addresses. In all such cases, the instruction definition contains a definition of the type of address.

Instruction Address

Addresses used to fetch instructions from storage are called instruction addresses. An instruction address is treated as a host-primary real address in all cases. The instruction address in the current PSW and the target address of EXECUTE are instruction addresses.

Protection

Four protection facilities are provided to protect the contents of main storage from destruction or misuse by programs that contain errors or are unauthorized: key-controlled protection, host access-list-controlled protection, host page protection, and low-address protection. These protection facilities are

applied independently; access to main storage is only permitted when none of the facilities prohibits the access.

Key-controlled protection affords protection against improper storing or against improper storing and fetching, but not against improper fetching alone. The key-controlled-protection mechanism is under the control of the program in the virtual machine.

Host access-list-controlled protection, host page protection, and low-address protection afford protection against improper storing. Host access-list-controlled protection and host page protection are under control of the host and cannot be circumvented by a program in the virtual machine. Low-address protection is under control of the program in the virtual machine.

Guest access-list-controlled protection and page protection as defined in ESA/390 is not available in ESA/XC since the dynamic address translation facility is not provided.

Programming Note: In contrast, if the host has applied either host access-list-controlled protection or host page protection to prevent a particular storage or storage-key alteration by the virtual machine, the program in the virtual machine cannot circumvent this protection. This attribute makes host access-list-controlled protection and host page protection appropriate for sharing of storage among virtual machines where protection is required.

Host access-list-controlled protection is particularly useful for sharing of storage among virtual machines because it regulates accesses using an attribute of an individual reference rather than using an attribute of the shared storage itself. This allows some virtual machines to be given authorization to store into the share storage while other, less authorized, virtual machines are prevented from storing.

Low-address protection is not discussed in this note because low-address protection applies only to a virtual machine's references to its own host-primary address space and is therefore not applicable to cross-virtual-machine sharing of storage under z/VM.

Key-Controlled Protection

Key-controlled protection as defined in ESA/390 applies, except for the applicability of the fetch-protection-override control as stated in the following section.

Fetch-Protection-Override Control

When the fetch-protection-override control (bit 6 of control register 0) is one, fetch protection is ignored for storage references to locations at effective addresses 0-2047 if the storage reference is made with a type-R real address or a logical address that is treated as a type-R real address. Fetch protection checking is always performed for storage references made with a type-A real address or a logical address that is treated as a type-A real address, regardless of the setting of the fetch-protection-override control.

Host Access-List-Controlled Protection

Host access-list-controlled protection controls store accesses to an address space by means of a read-only or read/write access type maintained in each host access-list entry. The access-type control is not directly accessible to any virtual machine. Host access-list-controlled protection provides protection against unauthorized storing or explicit storage-key alteration.

In the access-register mode, when a host access-list entry is used in the host-access-register-translation part of a reference and the host access-list entry indicates read/write access, both fetch and store accesses are permitted to the address space specified by the host access-list entry. Explicit alteration of storage keys within the address space is also permitted. When the host access-list entry indicates read-only access, fetch accesses to storage or storage keys are permitted, and an attempt to store or to alter explicitly a storage key causes a protection exception to be recognized and the execution of the instruction to be terminated or suppressed.

Host access-list-controlled protection applies to all store accesses or storage-key alterations made by means of a host access-list entry.

The access type permitted by a host access-list entry is established when the host access-list entry is made valid by means of the ALSERV ADD host service. If the host access-list entry designates an address

space owned by another virtual machine, establishing a host access-list entry permitting read/write access is subject to authorization. This authorization is granted by means of the ADRSPACE PERMIT host service by the virtual machine that owns the address space.

Programming Note: The ALSERV ADD and ADRSPACE PERMIT host services are described in the publication *z/VM: CP Programming Services*.

Host Page Protection

Host page protection controls store-type references to a 4K-byte block of storage (page) by means of a read-only or read/write authorization associated with each 4K-byte block of storage. The authorization control is not accessible by any virtual machine. Host page protection provides protection against unauthorized storing or storage-key alteration.

When the authorization associated with a 4K-byte block of storage indicates read/write access, both fetching of and storing into the block are permitted, including the explicit alteration of the storage key for the block. When the authorization associated with the block indicates read-only access, only fetching is permitted. When an attempt is made to store into a protected block or alter the storage key for a protected block, a protection exception is recognized and the operation is terminated or suppressed. The contents of the protected location remain unchanged.

Host page protection applies to all store accesses or storage-key alterations made by a virtual machine.

The access authorization associated with a 4K-byte block of storage is established by the host based on the host-defined characteristics of that block of storage.

Programming Note: The only 4K-byte blocks of storage established by z/VM as read-only pages are:

- those that correspond to shared read-only (SR) or exclusive read-only (ER) ranges of named saved systems or named saved segments embedded in the host-primary address space.
- those that are mapped by means of the MAPMDISK host service to minidisk blocks on a read-only minidisk. The MAPMDISK host service is described in the publication *z/VM: CP Programming Services*.

All other 4K-byte blocks of storage addressable by the virtual machine are established as read/write pages.

Low-Address Protection

Low-address protection operates as defined in ESA/390, except as stated in the following.

When the low-address-protection control (bit 3 of control register 0) is one, low-address protection applies to storage references to locations at effective addresses 0-511 if the storage references are made with a type-R real address or a logical address that is treated as a type-R real address. Low-address protection does not apply for storage references made with a type-A real address or a logical address that is treated as a type-A real address, regardless of the setting of the low-address-protection control.

Suppression on Protection

If the suppression-on-protection facility is installed, then, during a program interruption due to a protection exception, either a one or a zero is stored in bit position 29 of real locations 144-147. The storing of a one in bit position 29 indicates that:

- The unit of operation or instruction execution during which the exception was recognized was suppressed, except that, if the instruction execution would set the condition code if completed normally, the condition code is unpredictable.
- Bit positions 1-19 of real locations 144-147 contain bits 1-19 of the effective address that caused the exception. The effective address is the address which exists before any transformation by prefixing. Bit positions 30 and 31 of real locations 144-147, and real locations 160, contain information identifying the address space containing the protected address.

Bit 29 being zero indicates that the operation was either suppressed or terminated and that the contents of the remainder of real locations 144-147, and of real location 160, are unpredictable.

No protection-exception conditions necessarily set bit 29 to one.

Prefixing

Prefixing operates as defined in ESA/390, except as stated in the following.

Prefixing is applied to convert a type-R real address into an absolute address. Prefixing is not applied to convert a type-A real address into an absolute address, but rather the type-A real address is treated unchanged as an absolute address.

Dynamic Address Translation

The dynamic address translation facility provided in ESA/390 is not available in ESA/XC.

Consequently, the following additional DAT-related facilities defined in ESA/390 are also not available in ESA/XC:

- Virtual-storage address spaces
- Address-space numbers (ASNs)
- ASN-translation controls, tables and process
- ASN-authorization controls and process

Translation Control

Address translation is controlled by a bit in the PSW as described in this section.

Additionally, the execution of the INVALIDATE PAGE TABLE ENTRY instruction is controlled by the translation format, bits 8-12 of control register 0. The definition of the translation format is the same as provided in ESA/390.

Additional controls are provided as described in Chapter 5, “Program Execution,” on page 21. These additional controls determine whether the contents of each access register can be used to designate an address space.

Translation Modes

The PSW bit that controls address translation is bit 17, the address-space-control bit. When bit 17 is zero, the CPU is in the primary-space mode. When bit 17 is one, the CPU is in the access-register mode. The handling of addresses in these two modes is summarized in Table 1 on page 12.

Table 1. Translation Modes

PSW Bit 17	Mode	Handling of Addresses	
		Instruction Addresses	Logical[1] Addresses
0	Primary-space mode	Host-primary real	Host-primary real
1	Access-register mode	Host-primary real	AR-specified real

Explanation:

1 Certain real addresses are also handled differently according to the address-space-control mode. These real addresses are explicitly specified elsewhere in this publication.

Address Summary

Addresses Translated

Most addresses that are explicitly specified by the program and are used by the CPU to refer to storage operands are logical addresses and are subject to implicit translation by means of host access-register translation when the CPU is in the access-register mode. Analogously, the addresses indicated to the program as the result of executing an instruction are logical.

The addresses used by the CPU for fetching of instructions are instruction addresses and are host-primary real addresses. Similarly, the instruction addresses indicated to the program on an interruption are host-primary real addresses.

Translation is not applied to quantities that are formed from the values specified in the B and D fields of an instruction byte that are not used to address storage. This includes operand addresses in LOAD ADDRESS, LOAD ADDRESS EXTENDED, MONITOR CALL, and the shifting instructions. This also includes the addresses in control registers 10 and 11 designating the starting and ending locations for PER.

With the exception of TEST PROTECTION, addresses explicitly designating storage keys (operand addresses in SET STORAGE KEY EXTENDED, INSERT STORAGE KEY EXTENDED, and RESET REFERENCE BIT EXTENDED) are real addresses that are resolved according to the translation mode. These addresses are considered host-primary real addresses when in the primary-space mode, and AR-specified real addresses when in the access-register mode. Similarly, the address of the storage operand for TEST BLOCK is a real address that is resolved according to the addressing mode.

The addresses implicitly used by the CPU for such sequences as interruptions are host-primary real addresses.

The addresses used by channel programs to transfer data and to refer to CCWs or IDAWs are host-primary absolute addresses.

The handling of storage addresses associated with DIAGNOSE is dependent on the particular DIAGNOSE code. See *z/VM: CP Programming Services* for more information.

Handling of Addresses

The handling of addresses in ESA/XC is summarized in Figure 1 on page 14. This figure lists all addresses that are encountered by the program and specifies the address type.

Instruction Addresses

- Instruction address in PSW
- Branch address
- Target of EXECUTE
- Address stored in the word at real location 152 on a program interruption for PER
- Address placed in general register by BRANCH AND LINK, BRANCH AND SAVE, BRANCH AND SAVE AND SET MODE, BRANCH AND STACK, and BRANCH RELATIVE AND SAVE

Logical Addresses

- Addresses of storage operands for instructions not otherwise specified
- Address placed in general register 1 by EDIT AND MARK and TRANSLATE AND TEST
- Addresses in general registers updated by MOVE LONG, MOVE LONG EXTENDED, COMPARE LOGICAL LONG, and COMPARE LOGICAL LONG EXTENDED
- Addresses in general registers updated by CHECKSUM COMPARE AND FORM CODEWORD, and UPDATE TREE
- Address for TEST PENDING INTERRUPTION when the second-operand address is nonzero
- Addresses for parameter list of RESUME PROGRAM

Real Addresses Resolved According to the Translation Mode

- Address of storage key for INSERT STORAGE KEY EXTENDED, RESET REFERENCE BIT EXTENDED, and SET STORAGE KEY EXTENDED
- Address of storage operand for TEST BLOCK

Host-Primary Real Addresses

- Storage operand of INVALIDATE PAGE TABLE ENTRY, LOAD USING REAL ADDRESS, and STORE USING REAL ADDRESS
- Trace-entry address in control register 12
- Dispatchable-unit-control-table origin in control register 2 (used by BRANCH AND SET AUTHORITY)

Permanently Assigned Host-Primary Real Addresses

- Address of the doubleword into which TEST PENDING INTERRUPTION stores when the second-operand address is zero
- Addresses of PSWs, interruption codes, and the associated information used during interruption
- Addresses used for machine-check logout and save areas

Absolute Addresses

- Failing-storage address stored in the word at real location 248

Figure 1. Handling of Addresses (Part 1 of 2)

Host-Primary Absolute Addresses

- Prefix value
- Channel-program address in ORB
- Data address in CCW
- IDAW address in a CCW specifying indirect data addressing
- CCW address in a CCW specifying transfer in channel
- Data address in IDAW
- Measurement-block origin specified in SET CHANNEL MONITOR
- Address limit specified in SET ADDRESS LIMIT
- Addresses used by the store-status-at-address SIGNAL PROCESSOR order
- CCW address in SCSW

Permanently Assigned Host-Primary Absolute Addresses

- Addresses used for the store-status function
- Addresses of PSW and first two CCWs used for initial program loading

Addresses Not Used to Reference Storage

- PER starting address in control register 10
- PER ending address in control register 11
- Address stored in the word at real location 156 for a monitor event
- Address in shift instructions and other instructions specified not to use the address to reference storage

Figure 2. Handling of Addresses (Part 2 of 2)

Assigned Storage Locations

All of the storage locations assigned in ESA/390 are also assigned for the same purpose in ESA/XC, except as specified in the following. All assigned storage locations reside within the host-primary address space.

144-147
(Real Address)

Translation-Exception Identification: During a program interruption due to a protection exception, if the suppression-on-protection facility is installed, information is stored at locations 144-147 describing the exception, as discussed in "Suppression on Protection" on page 11. Bit 29 being one indicates that the unit of operation or instruction execution during which the exception was recognized was suppressed, except that, if the instruction execution would set the condition code if completed normally, the condition code is unpredictable. Bit 29 being zero indicates that the operation was either suppressed or terminated. If bit 29 is set to one, bit positions 1-19 contain bits 1-19 of the effective address that caused the exception. The effective address is the address which exists before any transformation by prefixing. Bit positions 30-31 of real locations 144-147 are set to identify the address space containing the protected address, as follows:

00
A type-R real address was used.

01
CPU was in the access-register mode, and either the access was an instruction fetch or it was a storage-operand reference that used an AR-specified real address. The exception access id, real

location 160, can be examined to determine the address space containing the address. However, if a type-R real address was used, bits 30 and 31 may be set to 00 instead.

Bits 0 and 20-28 of real address 144-147 are unpredictable.

160
(Real Address)

Exception Access Identification: During a program interruption due to an ALEN-translation or addressing-capability exception, if the ALET being translated was obtained from an access register, the number of the access register used is stored in bit positions 4-7 of location 160, and zeros are stored in bit positions 0-3. If the ALET being translated was not obtained from an access register, then zeros are stored at location 160.

During a program interruption due to a protection exception, an indication of the address space to which the exception applies may be stored at location 160 if the suppression-on-protection facility is installed and bit 29 of the translation-exception identification is set to one. If the CPU was in the access-register mode and the access was an instruction fetch, including a fetch of the target of an EXECUTE instruction, zeros are stored at location 160. If the CPU was in the access-register mode and the access was a storage-operand reference to an AR-specified real address, the number of the access register used is stored in bit positions 4-7 of location 160, and zeros are stored in bit positions 0-3. If the CPU was not in the access-register mode, the contents of location 160 are unpredictable.

168-171
(Real Address)

Exception ALET: During a program interruption due to an ALEN-translation or addressing-capability exception, the ALET being translated is stored at locations 168-171.

256-263
(Real Address)

Failing-Storage ASIT: During a machine-check interruption, a failing-storage ASIT may be stored at locations 256-263. A failing-storage ASIT is stored whenever a failing-storage address is stored at locations 248-251.

Chapter 4. Control

This chapter describes how the facilities for controlling, measuring and recording the operation of one or more CPUs for ESA/XC differ from those defined in ESA/390. Except as described in the following, the definition for control facilities provided in Chapter 4 of *IBM Enterprise Systems Architecture/390 Principles of Operation* applies to ESA/XC as well.

Program-Status Word

The program-status word (PSW) is the same as defined in ESA/390, except as described in the following section.

Program-Status-Word Format

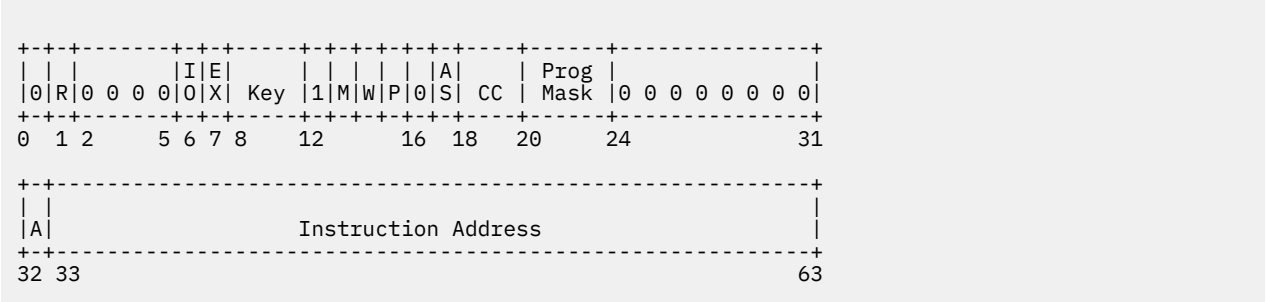


Figure 3. PSW Format

All PSW fields (that is, those bit positions that are not required to be zeros) have the same function as in ESA/390, with the exception of PSW bit 17.

Address-Space Control (AS)

Bit 17 controls the translation mode. When bit 17 is zero, the CPU is in the primary-space mode; all logical, real and absolute addresses are considered host-primary addresses. When bit 17 is one, the CPU is in the access-register mode; all logical and certain real and absolute addresses are considered AR-specified addresses that reside within the address space determined by host access-register translation.

In addition to those bit positions unassigned in ESA/390, bit positions 5 and 16 are also unassigned in ESA/XC. A specification exception is recognized when these bit positions do not contain zeros.

Control Registers

Control registers and control-register fields are the same as defined in ESA/390, except as specified in the following.

In ESA/XC, all bit positions of control registers 1, 4, 5, 7, 13, and 15 are unassigned. In addition, bits 1, 2, and 5 of control register 0; bits 16-31 of control register 3; bits 0-15 of control register 8; bit 30 of control register 12; and bits 12-31 of control register 14 are unassigned. All of these unassigned control-register positions are initialized to zero.

The section “Changes in Control-Register Assignments” on page 53 identifies the functions associated with these control registers and control-register fields in ESA/390.

Tracing

Tracing as defined in ESA/390 applies in ESA/XC except that ASN tracing is not provided. The ASN-trace-control bit is not provided, and SET SECONDARY ASN, PROGRAM CALL, PROGRAM RETURN and PROGRAM TRANSFER trace entries are never formed.

Program-Event Recording

The ESA/390 definition of program-event recording (both PER 1 and PER 2) applies in ESA/XC except as stated in this section.

The PER-2 means of restricting storage-alteration events to occurring only when the storage is within designated address spaces is not provided in ESA/XC. In ESA/XC, storage-alteration events are not restricted to particular address spaces.

With PER 2, bit 10 of control register 9 is not ignored when DAT is off. Instead, when bit 10 of control register 9 is one, it is unpredictable whether any storage-alteration events are indicated.

Timing

The timing facilities include three facilities for measuring time: the TOD clock, the clock comparator, and the CPU timer. Except as described in this section, the operation of the timing facilities is as defined in ESA/390.

Time-of-Day Clock

An ESA/XC configuration does not have a TOD clock distinct from the TOD clocks of other ESA/XC configurations. Instead, all ESA/XC configurations are provided with shared access to the TOD clock of the host processor. The shared host TOD clock cannot be altered by the virtual machine. However, if the extended-TOD-clock facility is installed, SET CLOCK PROGRAMMABLE FIELD may be used to set the TOD programmable register for the virtual CPU.

The TOD clock is always in the set state. The STORE CLOCK and STORE CLOCK EXTENDED instructions store the current value of the host processor TOD clock, and always set condition code 0. On a single host system, two executions of STORE CLOCK or STORE CLOCK EXTENDED, possibly on different CPUs of different ESA/XC virtual machines, always store different values.

The SET CLOCK instruction is not provided. An attempt to execute the instruction results in an operation exception.

The TOD-clock synchronization facility is not provided.

Externally Initiated Functions

Resets

The five reset functions provided in ESA/390 are also provided in ESA/XC, and perform all of the functions defined in ESA/390. In addition, the subsystem reset function performs additional actions in ESA/XC, as defined in the following.

Subsystem reset provides a means for clearing floating interruption conditions as well as for invoking I/O-system reset. It also provides a means for resetting elements of the ESA/XC environment that are controlled by the host on behalf of the virtual machine.

Subsystem Reset

Subsystem reset operates only on those elements in the configuration which are not CPUs. In addition to the actions defined in ESA/390, subsystem reset in ESA/XC also performs the following actions:

1. All entries in the host access list are set to the unused state.
2. Any address spaces created using the ADRSPACE CREATE host service are destroyed.
3. The host-primary address space, if in the shareable state, is placed in the private state and access permission granted to other virtual machines is rescinded.
4. Certain other host-controlled entities are reset.

Programming Note: A list of the subsystem-reset actions on host-controlled entities is provided in the description of the SYSTEM RESET command in the publication *z/VM: CP Commands and Utilities Reference*.

Chapter 5. Program Execution

Program execution for ESA/XC proceeds as defined in Chapter 5 of the *IBM Enterprise Systems Architecture/390 Principles of Operation* except for references in that publication to facilities defined elsewhere in this publication as not applicable to ESA/XC and except for other differences described in this chapter.

Authorization Mechanisms

ESA/390 includes several authorization mechanisms to permit the control program to establish the degree of function which is provided to a particular semiprivileged program. Most of these authorization mechanisms are related to functions that are not provided in ESA/XC and therefore do not apply in ESA/XC. In particular, the following ESA/390 authorization mechanisms do not apply in ESA/XC:

- Mode requirements (DAT on)
- Secondary-space control
- Subsystem-linkage control
- ASN-translation control
- Authorization index

The following ESA/390 authorization mechanisms apply in ESA/XC as defined in ESA/390 except as stated in the following sections:

- Extraction-authority control
- PSW-key mask
- Access-register mechanisms

Extraction-Authority Control

In ESA/XC, the extraction-authority control does not apply to the execution of the INSERT ADDRESS SPACE CONTROL instruction. That is, the instruction can be successfully executed regardless of the setting of bit 4 of control register 0.

Access-Register Mechanisms

Bit 15 of control register 0 is the address-space function (ASF) control bit. Bit 15 must be one to allow completion of the BRANCH AND SET AUTHORITY, RESUME PROGRAM and TEST ACCESS instructions. The ASF control also controls the setting of the access-register mode by the SET ADDRESS SPACE CONTROL and SET ADDRESS SPACE CONTROL FAST instructions. The ASF control has no other effect in ESA/XC.

The use of access registers also involves various host-managed authorization mechanisms. These mechanisms are described in the publication *z/VM: CP Programming Services*.

PC-Number Translation

Neither the PC-number translation process, nor the PROGRAM CALL instruction, is provided in ESA/XC.

Home Address Space

Neither the home address space nor the home-space mode is provided in ESA/XC.

Access-Register Introduction

Because of the importance of this section to ESA/XC, it is being provided in the form of a complete replacement for the corresponding section of *IBM Enterprise Systems Architecture/390 Principles of Operation* rather than merely stating the differences from ESA/390.

Many of the functions related to access registers are described in this section and in the sections “Host Access-Register Translation” on page 27 and “Sequence of Storage References” on page 30 in this chapter and "Sequence of Storage References" in Chapter 5 of *IBM Enterprise Systems Architecture/390 Principles of Operation*. Additionally, Chapter 3, “Storage,” on page 7, of this publication describes translation modes and host access-list-controlled protection; Chapter 4, “Control,” on page 17, describes the handling of address spaces, access registers and host access lists during resets and during the store-status operation; Chapter 6, “Interruptions,” on page 33, describes interruptions; and Chapter 7, “Instructions,” on page 39, describes the instructions. Also, Chapter 11, "Machine-Check Handling". of *IBM Enterprise Systems Architecture/390 Principles of Operation* describes the handling of access registers during a machine-check interruption and the validation of the access registers; and Chapter 12, "Operator Facilities", of that publication describes the alter-and-display controls for access registers. The publication *z/VM: CP Commands and Utilities Reference* also describes alter and display facilities for access registers and access-register-specified storage.

Summary

These major functions are provided:

- A maximum of 16 absolute-storage address spaces, including the instruction space, for immediate and simultaneous use by a semiprivileged program; the address spaces are specified by 16 special registers called access registers.
- Instructions for examining and changing the contents of the access registers.

In addition, control and authority mechanisms are incorporated to control these functions.

Access registers allow a sequence of instructions, or even a single instruction such as MOVE (MVC) or MOVE LONG (MVCL), to operate on storage operands in multiple address spaces. Thus, a program residing in one address space can use the complete instruction set to operate on data in that address space and in up to 15 other address spaces, and it can move data between any and all pairs of these address spaces. Furthermore, the program can change the contents of the access registers in order to access still other address spaces.

The instructions for examining and changing access-register contents are unprivileged and are described in Chapter 7, "General Instructions" of *IBM Enterprise Systems Architecture/390 Principles of Operation* and Chapter 7, “Instructions,” on page 39 of this publication. They are:

- COPY ACCESS
- EXTRACT ACCESS
- LOAD ACCESS MULTIPLE
- LOAD ADDRESS EXTENDED
- SET ACCESS
- STORE ACCESS MULTIPLE

Access registers specify address spaces when the CPU is in the access-register mode. The SET ADDRESS SPACE CONTROL and SET ADDRESS SPACE CONTROL FAST instructions allow setting of the access-register mode, and the INSERT ADDRESS SPACE CONTROL instruction provides an indication of the access-register mode. These instructions are described in Chapter 7, “Instructions,” on page 39.

Access-Register Functions

Access-Register-Specified Address Spaces

The CPU includes sixteen 32-bit access registers numbered 0-15. In the access-register mode, which results when PSW bit 17 is one, an instruction B or R field that is used to specify the logical address of a storage operand designates not only a general register but also an access register. In certain cases, an instruction R field that is used to specify the real address of a storage operand designates both a general register and an access register. The designated general register is used in the ordinary way to form the logical or real address of the storage operand. The designated access register contains a parameter called an access-list-entry token (ALET) that is used to determine the address space to which the logical or real address is relative. This address space is known as the target address space.

For certain host services invoked in the access-register mode, ALETs may be provided in the parameter list for the host service as well as providing an ALET in access registers. These parameter-list ALETs specify the target address spaces for particular storage operands of the host service.

Regardless of its source, an ALET specifies the target address space for an operand by indirectly specifying the address-space identification token (ASIT) for the target address space.

In the general case, the ALET specifies an address space through the use of a host-managed table called a host access list. The ALET selects an entry in a host access list, and the selected host access-list entry in turn specifies the target address space. However, a special ALET value can specify the host-primary address space without using a host access-list entry.

The process of using an ALET to determine the address space containing an operand is called host access-register translation (ART). This is depicted, for an ALET obtained from an access register, in Figure 4 on page 23.

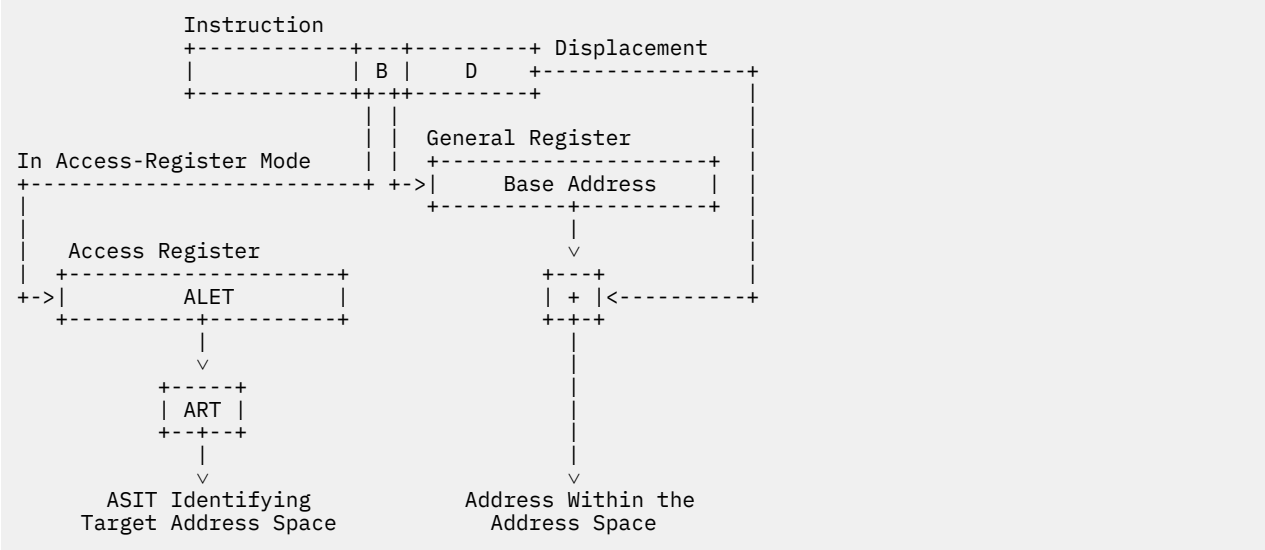


Figure 4. Use of Access Registers

An access register is said to specify an AR-specified address space. The addresses in an AR-specified address space are AR-specified real or AR-specified absolute addresses.

In the access-register mode, as in the primary-space mode, all instruction addresses are host-primary real.

Designating Access Registers

In the access-register mode, an instruction B or R field designates an access register, for use in host access-register translation, under the following conditions:

- The field is a B field that designates a general register containing a base address. The base address is used, along with a displacement (D) and possibly an index (X), to form the logical address of a storage operand.

- The field is an R field that designates a general register containing the logical address, or for certain instructions the real address, of a storage operand.

For example, consider the following instruction:

```
     MVC 0(L,1),0(2)
```

The second operand, of length L, is to be moved to the first-operand location. The logical address of the second operand is in general register 2, and that of the first-operand location is in general register 1. The address space containing the second operand is specified by access register 2, and that containing the first-operand location is specified by access register 1. These two address spaces may be different address spaces, and each may be different from the instruction space (the host-primary address space).

The COMPARE AND FORM CODEWORD and UPDATE TREE instructions specify storage operands by means of implicitly designated general registers and access registers.

An instruction R field may designate an access register for other than the purpose of host access-register translation.

The fields that may designate access registers, whether or not for host access-register translation, are indicated in the summary figure at the beginning of each instruction chapter in *IBM Enterprise Systems Architecture/390 Principles of Operation*.

Determining the Target Address Space

This section and the following ones introduce the host-access-register-translation process and present the concepts related to access lists.

The target address space specified by an ALET is determined by host access-register translation as follows:

- If the ALET contained in the access register or in the parameter list supplied to a host service is 00000000 hex, the target address space is the host-primary address space.
- If the ALET contained in the access register or in the parameter list supplied to a host service is any other value, the target address space is identified by an address-space identification token (ASIT) obtained from a host access-list entry. The ALET selects a host access-list entry, which contains the address-space identification token that identifies the address space.

Access register 0 is treated in a special way by host access-register translation; it is treated as containing 00000000 hex, and its actual contents are not examined. Thus, a logical or real address specified by means of a zero B or R field in the access-register mode is always relative to the host-primary address space, regardless of the contents of access register 0. However, there is one exception to how access register 0 is treated: the TEST ACCESS instruction uses the actual contents of access register 0, instead of treating access register 0 as containing 00000000 hex.

The treatment of an access register containing the value 00000000 hex as designating the host-primary address space allows that address space to be addressed, in the access-register mode, without requiring the use of a host access-list entry. This is useful for moving data to or from the host-primary address space.

Access Lists

Each ESA/XC virtual machine is provided with a separate host access list for its use in accessing address spaces when in the access-register mode. The access list is protected from direct examination or modification by the virtual machine. The virtual machine for which a host access list is provided can alter the set of address spaces designated by the host access list by using host services.

An access list can contain from 6 to 1022 entries, each of which can designate a different address space. The size of virtual machine's access list is controlled by the host directory entry for the virtual machine.

Access-List-Entry Token

The contents of an access register are called an access-list-entry token (ALET) since, in the general case, they select an entry in a host access list. The ALET is a 32-bit token, the structure of which is not further defined.

An ALET can exist in an access register, in a general register, in a parameter list for certain host services, or in storage, and it has no special protection from manipulation by the program. Any program can transfer ALETs back and forth among access registers, general registers, and storage. A called program can save the contents of the access registers in any storage area available to it, load and use the access registers for its own purposes, and then restore the original contents of the access registers before returning to its caller.

Allocating and Invalidating Access-List Entries

The host provides to each ESA/XC virtual machine a separate host access list. The host access list associated with a particular virtual machine specifies the collection of address spaces, in addition to the virtual machine's host-primary address space, that are available to a program when it is in the access-register mode. All alterations of host access lists are performed by means of host services.

Each entry in a host access list is considered to be in one of three states: valid, revoked or unused. A valid host access-list entry specifies an address space and can be used in the access-register mode to reference that space. An unused host access-list entry is available for allocation as a valid entry. A revoked host access-list entry is an entry that was previously valid, but at some time after the allocation of the entry the virtual machine's authorization to access the designated address space was revoked. (Revoked host access-list entries are described further in the section “Revoking Accessing Capability” on page 25.) The host provides services for allocating a valid host access-list entry (changing an unused entry to valid) and for deallocating a valid or revoked entry (making it unused).

Allocation of a host access-list entry consists of the following steps. A program invokes the ALSERV ADD host service, passing the address-space identification token (ASIT) specifying the address space for which access is being requested, and passing an indication of whether read-only or read/write access is desired. If the ASIT identifies a space not owned by the requesting virtual machine, the host checks that the requesting virtual machine has been authorized for the requested type of access. This authorization is granted through use of the ADRSPACE PERMIT by the virtual machine that owns the address space. If the ASIT identifies a space owned by the requesting virtual machine, it is always authorized for read/write access. If the virtual machine's request is authorized, the host selects an unused entry in the virtual machine's host access list, changes it to a valid entry specifying the subject address space, establishes the entry as permitting read-only or read/write access as requested, and returns to the program an access-list-entry token (ALET) that designates the entry. Once assigned, the ALET remains uniquely associated with the entry until the entry returns to the unused state.

The program can place the ALET in an access register in order to access the address space. The program can also place the ALET in a parameter list for a host service to access the address space through certain host services.

Later, through the use of the ALSERV REMOVE host service, the host access-list entry that was allocated may be returned to the unused state. This makes the entry available for reallocation to designate a different address space.

Notes on the Authorization Mechanism

A host access list is a kind of capability list, in the sense in which the word "capability" is used in computer science. The host establishes the policies that are used to allocate entries in a host access list and performs appropriate authorization checking during the allocation of an entry. After a valid entry has been made in a host access list, the host-access-register-translation process enforces the host policies in a well-performing way.

Revoking Accessing Capability

It may be that a particular valid host access-list entry specifies an address space owned by another virtual machine and the owning virtual machine revokes the accessing virtual machine's authority to access the address space. In this case, the revocation process causes all host access-list entries of the

accessing virtual machines that designate the subject space to be changed from the valid to the revoked state. The owning virtual machine revokes access authority through the use of the ADRSPACE ISOLATE host service.

Similarly, a particular valid host access-list entry may specify an address space (owned by the same or another virtual machine) that is subsequently destroyed before the host access-list entry is deallocated. The destroy process causes all host access-list entries of all virtual machines that designate the subject space to be changed from the valid to the revoked state. The owning virtual machine destroys an address space through the use of the ADRSPACE DESTROY host service.

An exception is recognized during host access-register translation if an ALET is used that selects a revoked host access-list entry. No distinction is made between the two causes of a host access-list entry's entering the revoked state.

Preventing Store Accesses

Each host access-list entry contains an access-type indicator which determines if the entry can be used for fetch and store accesses to the subject address space or only for fetch accesses. This access-type indicator is established when the entry is allocated by the host, depending on the requested type of access to the address space and the requesting virtual machine's authorization. When the access-type indicator in a host access-list entry specifies read-only access, the entry cannot be used to perform store accesses or explicit storage-key alterations. The principal description of this protection mechanism is in the section “Host Access-List-Controlled Protection” on page 10.

Improving Translation Performance

Host access-register translation (ART) conceptually occurs each time a logical or real address is used to reference a storage operand in the access-register mode. To improve performance, ART normally is implemented such that some or all of the information contained in the host-managed ART structures is maintained in a special buffer referred to as the ART-lookaside buffer (ALB). The information in the ART structures may be placed in the ALB and subsequent translations may be performed using the information in the ALB. Conceptually, the ALB buffers information necessary to transform an ALET into the identification of the address space which the ALET designates. The ALB is typically implemented such that it is capable of buffering the ART structures related to a relatively small number of recent translations.

The host manages the ALB to ensure that the contents of the ALB never represent obsolete ART structures. Other than the effect on performance, the structure and management of the ALB are not visible to the ESA/XC virtual machine. The ALB is not further described in this publication.

Access-Register Instructions

The following instructions are provided for examining and changing the contents of access registers:

- COPY ACCESS
- EXTRACT ACCESS
- LOAD ACCESS MULTIPLE
- LOAD ADDRESS EXTENDED
- SET ACCESS
- STORE ACCESS MULTIPLE

The SET ACCESS instruction replaces the contents of a specified access register with the contents of a specified general register. Conversely, the EXTRACT ACCESS instruction moves the contents of an access register to a general register. The COPY ACCESS instruction moves the contents of one access register to another.

The LOAD ACCESS MULTIPLE instruction loads a specified set of consecutively numbered access registers from a specified storage location whose length in words equals the number of access registers loaded. Conversely, the STORE ACCESS MULTIPLE instruction function stores the contents of a set of access registers at a storage location.

The LOAD ADDRESS EXTENDED instruction is similar to the LOAD ADDRESS instruction in that it loads a specified general register with an effective address specified by means of the B, X, and D fields of the instruction. In addition, LOAD ADDRESS EXTENDED operates on the access register having the same number as the general register loaded. When the address-space control, PSW bit 17, is zero, LOAD ADDRESS EXTENDED loads the access register with 00000000 hex. When the address-space control is one, LOAD ADDRESS EXTENDED loads the target access register with a value that depends on the B field of the instruction. If the B field is zero, LOAD ADDRESS EXTENDED loads the target access register with 00000000 hex. If the B field is nonzero, LOAD ADDRESS EXTENDED loads the target access register with the contents of the access register designated by the B field. However, in the last case, when the contents of the access register designated by the B field are not a correctly-formed ALET, the results in the target general register and access register are unpredictable.

The address-space-control values zero and one specify the primary-space mode and the access-register mode, respectively.

When used in host access-register translation, the access-register value 00000000 hex specifies the host-primary address space.

Host Access-Register Translation

Because of the importance of this section to ESA/XC, it is being provided in the form of a complete replacement for the corresponding section of *IBM Enterprise Systems Architecture/390 Principles of Operation* rather than merely stating the differences from ESA/390.

Host access-register translation is introduced in the section "Access-Register-Specified Address Spaces" on page 23.

Host-Access-Register-Translation Control

Host access-register translation is controlled by an address-space control, and by the address-space-function (ASF) control in control register 0. The address-space control, PSW bit 17, is described in the section "Translation Modes" on page 12. The address-space-function (ASF) control is described in the following section.

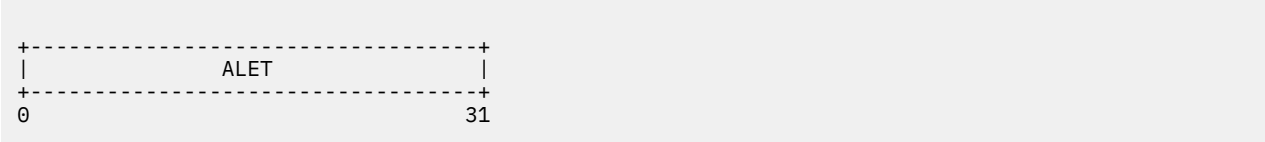
Additional controls are located in the host-managed access-register-translation structures.

Address-Space-Function Control

Bit 15 of control register 0 is the address-space-function (ASF) control. This bit must be one when a SET ADDRESS SPACE CONTROL or SET ADDRESS SPACE CONTROL FAST instruction that is to set the access-register mode is executed, and when a TEST ACCESS instruction is executed; otherwise, a special-operation exception is recognized. The address-space-function control has no other effect in an ESA/XC virtual machine.

Access Registers

There are sixteen 32-bit access registers numbered 0-15. The contents of an access register are called an access-list-entry token (ALET). An ALET has the following format:



When the ALET is not 00000000 hex, it is treated as a 32-bit token that may be associated with at most one entry in the virtual machine's host access list. During host access-register translation, the ALET is used as a selection token to determine whether there is an entry in the host access list that is associated with the ALET. If there is no such entry, an ALEN-translation exception is recognized.

Not all possible 32-bit values are valid for use as an ALET. If an incorrectly-formed value is used as an ALET during host access-register translation, an ALET-specification exception is recognized.

When the ALET is 00000000 hex, it specifies the host-primary address space and is not used to select an entry from the host access list. A logical or real address used in conjunction with an ALET equal to 00000000 hex is treated as a type-R real address.

Access register 0 usually is treated in host access-register translation as containing 00000000 hex, and its actual contents are not examined; the host access-register translation done as part of TEST ACCESS is the only exception. Access register 0 is also treated as containing 00000000 hex when it is designated by the B field of LOAD ADDRESS EXTENDED when PSW bit 17 is one. When access register 0 is specified for TEST ACCESS or as a source for COPY ACCESS, EXTRACT ACCESS, or STORE ACCESS MULTIPLE, the actual contents of the access register are used. Access register 0, like any other access register, can be loaded by COPY ACCESS, LOAD ACCESS MULTIPLE, LOAD ADDRESS EXTENDED, and SET ACCESS.

Host-Access-Register-Translation Structures

When the ALET being translated is not 00000000 hex, host access-register translation selects an entry in the virtual machine's host access list.

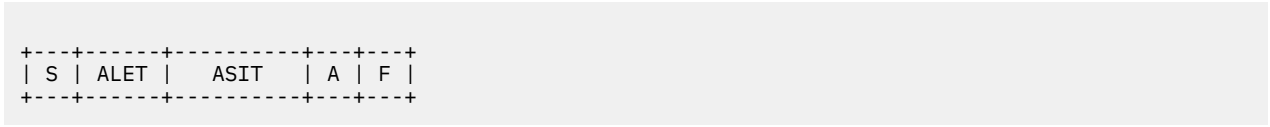
Host Access List

Each ESA/XC virtual machine has associated with it a host access list that specifies those address spaces, in addition to the host-primary address space, that are available to a virtual CPU when it is in the access-register mode. The host access list contains a directory-specified number of host access-list entries, each of which conceptually contains the information defined in the following section. The host access list is not explicitly addressable by the virtual machine. Instead, entries in the host access list are manipulated by means of host services.

Programming Notes: The size of the host access list allocated for a virtual machine is specified by the XCONFIG ACCESSLIST statement in the CP directory. This CP directory statement is described in the publication *z/VM: CP Planning and Administration*.

Host Access-List Entries

Each host access-list entry has the following logical structure:



The fields within the host access-list entry have the following meanings:

Access-list-entry state (S)

This field indicates the state of the host access-list entry: valid, revoked, or unused.

Host access-list entries in the valid or revoked state have been allocated for use by the program and can be selected by host access-register translation. If the entry selected by host access-register translation is in the valid state, the translation process proceeds. If the entry selected by host access-register translation is in the revoked state, an addressing-capability exception is recognized.

Host access-list entries in the unused state are not currently allocated for use by the program and cannot be selected by host access-register translation. The other fields of the host access-list entry have no meaning when the entry is in the unused state.

Selection ALET (ALET)

For a host access-list entry in the valid or revoked state, this field specifies the ALET that selects the entry during host access-register translation. There is at most one valid or revoked entry in the host access list that contains a particular ALET value as the selection ALET. This field has no meaning in an entry in the unused state.

Designated address space (ASIT)

For a host access-list entry in the valid state, this field contains the address-space identification token (ASIT) identifying the address space designated by the host access-list entry. This field has no meaning in an entry in the revoked or unused state.

Access type (A)

For a host access-list entry in the valid state, this field specifies the types of access permitted to the address space designated by the host access-list entry: read-only or read/write access. If this field indicates read/write access, both fetch and store accesses are permitted. If this field indicates read-only access, only fetch accesses are permitted, and an attempt to store causes a protection exception for host access-list-controlled protection to be recognized. This field has no meaning in an entry in the revoked or unused state.

Fault handling (F)

For a host access-list entry in the valid state, this field indicates the manner in which host segment or page faults are to be handled: synchronously or asynchronously. For information on the options for handling of host segment and page faults, see the description of the PFAULT and ALSERV services in the publication *z/VM: CP Programming Services*. This field has no meaning in an entry in the revoked or unused state.

Host-Access-Register-Translation Process

This section describes the host-access-register-translation process as it is performed during a storage-operand reference in the access-register mode by any instruction except TEST ACCESS and TEST PROTECTION. TEST PROTECTION in the access-register mode, and TEST ACCESS in any translation mode, perform host access-register translation the same as described here, except that the following exceptions cause a setting of the condition code instead of being treated as program-interruption conditions:

- Addressing capability
- ALET specification
- ALEN translation

Host access-register translation operates on the access register designated in a storage-operand reference in order to determine the address space containing the storage operand, hereafter called the target address space. When one of access-registers 1-15 is designated, the access-list-entry token (ALET) that is in the access register is used to determine the address space. When access register 0 is designated, an ALET having the value 00000000 hex is used, except that TEST ACCESS uses the actual contents of access register 0. In certain cases, host access-register translation operates on an ALET obtained from a parameter list for a host service.

When the ALET is 00000000 hex, the host-primary address space is the target address space.

When the ALET is other than 00000000 hex, the ALET is verified to be correctly formed and is then used in a lookup process to select an entry in the virtual machine's host access list.

The selected host access-list entry is checked for being in the valid state.

If a store access or an explicit storage-key alteration is to be performed, the access-type indication in the host access-list entry is checked to determine if read/write access is permitted.

When no exceptions are recognized, the ASIT in the selected host access-list entry identifies the target address space.

Selecting the Access-List-Entry Token

When one of access registers 1-15 is designated, or for the access register designated by the R_1 field of TEST ACCESS, host access-register translation uses the access-list-entry token (ALET) that is in the access register. When access register 0 is designated, except for TEST ACCESS, an ALET having the value 00000000 hex is used, and the contents of access register 0 are not examined.

In certain cases, host access-register translation uses an ALET contained in a parameter list for a host service. The ALET associated with storage operands for host services is described as part of the definition of the service in the publication *z/VM: CP Programming Services*.

When LOAD ACCESS MULTIPLE changes the contents of an access register being used by host access-register translation, or when a store access changes the contents of an ALET field in a parameter list for a host service, the ALET obtained at the start of the operation is in effect for the duration of the operation.

Making the Host-Primary Address Space the Target Space

When the ALET being translated is 00000000 hex, the virtual machine's host-primary address space is established as the target address space and host access-register translation is completed.

Checking the ALET for Validity

The ALET is checked for being a correctly-formed ALET. If the ALET is not correctly formed, an ALET-specification exception is recognized, and the operation is suppressed.

Access-List Lookup

A lookup in the virtual machine's host access list is performed.

Conceptually, each host access-list entry in the valid or revoked state is examined to determine whether the selection ALET in the host access-list entry matches the ALET being translated. Host access-list entries in the unused state are not considered by this lookup process.

There is at most one valid or revoked host access-list entry containing a selection ALET matching the ALET being translated. If there is an entry with a selection ALET matching the ALET being translated, that entry is said to be the entry selected by the ALET being translated. If the selected entry is in the valid state, the host-access-register-translation process proceeds. If the selected entry is in the revoked state, an addressing-capability exception is recognized, and the operation is terminated. If there is no valid or revoked host access-list entry containing a selection ALET matching the ALET being translated, an ALEN-translation exception is recognized, and the operation is nullified.

Checking for Host Access-List-Controlled Protection

If a store access or an explicit storage-key alteration is to be performed and the access-type indication in the selected host access-list entry specifies read-only access, a protection exception is recognized, and the operation is terminated.

Establishing the Target Address Space

When the ALET being translated is other than 00000000 hex and no exception is recognized in the steps described previously, the address space identified by the ASIT field in the selected host access-list entry is established as the target address space and host access-register translation is completed.

Exceptions during Host Access-Register Translation

The exceptions which can be encountered during the host-access-register-translation process and their priority are shown in the section "Access Exceptions" on page 36 in Chapter 6, "Interruptions," on page 33.

Linkage Stack

The linkage stack is not provided in ESA/XC.

Sequence of Storage References

The ESA/390 definition of the effects which can be observed in storage due to overlapped operations and piecemeal execution of a CPU program apply to ESA/XC as well.

However, in interpreting the definition contained in *IBM Enterprise Systems Architecture/390 Principles of Operation* for ESA/XC, references to virtual storage or virtual addresses in that publication apply instead to real storage, real addresses (type-R and type-A) and absolute addresses in ESA/XC. Additionally, references in that publication to the following do not apply to ESA/XC: the real mode, the secondary-space mode, and the home-space mode; ART-table and DAT-table fetches; and ALB entries.

Chapter 6. Interruptions

This chapter describes how the interruption mechanism for ESA/XC differs from that defined for ESA/390. Except as described in the following section, the definition for interruptions provided in Chapter 6 of *IBM Enterprise Systems Architecture/390 Principles of Operation* applies to ESA/XC as well.

Interruption Action

Except as described in this section, the interruption action in ESA/XC is the same as defined for ESA/390.

During an interruption, the old PSW and interruption parameters are stored in, and the new PSW fetched from, the host-primary address space.

Exceptions Associated with the PSW

In addition to those defined in ESA/390, ESA/XC includes some additional error conditions that cause recognition of PSW-format errors when the erroneous information is introduced into the PSW. Error conditions that are recognized as part of the execution of the next instruction are the same as defined in ESA/390.

Early Exception Recognition

In addition to the error conditions defined in ESA/390, a program interruption for a specification exception occurs immediately after the PSW becomes active if a one is introduced into bit positions 5 or 16 of the PSW.

For these additional causes, interruption occurs as defined in ESA/390 for other PSW-format errors that are recognized early.

Program Interruption

Program interruptions follow the definition in ESA/390 except as described in this section.

Program-Interruption Conditions

The following is a detailed description of each program-interruption condition that is either recognized only in ESA/XC, or is defined differently in ESA/XC that it is in ESA/390.

The following program-interruption conditions that are defined in ESA/390 are never recognized in ESA/XC:

- AFX-translation exception
- ALE-sequence exception
- ASN-translation-specification exception
- ASTE-sequence exception
- ASTE-validity exception
- ASX-translation exception
- EX-translation exception
- Extended-authority exception
- LX-translation exception
- Page-translation exception
- PC-translation-specification exception

- Primary-authority exception
- Secondary-authority exception
- Segment-translation exception
- Space-switch event
- Stack-empty exception
- Stack-full exception
- Stack-operation exception
- Stack-specification exception
- Stack-type exception

Any other ESA/390 program-interruption conditions not described in one of the following sections is defined in ESA/XC in the same way that it is defined for ESA/390 except that the ESA/390 definition may include causes not applicable to ESA/XC because the related facilities are not provided.

The addressing-capability exception, defined in the following section, is recognized only in ESA/XC.

Addressing-Capability Exception

An addressing-capability exception is recognized during host access-register translation when the access-list-entry token used is correctly formed but designates a host access-list entry that is in the revoked state.

The access-list-entry token being translated is stored at real locations 168-171. If the access-list-entry token was obtained from an access register, the number of the access register is stored in bit positions 4-7 at real location 160, and bits 0-3 are set to zeros. If the access-list-entry token was obtained from the parameter list for a host service, then zeros are stored in bit positions 0-7 at real location 160.

The operation is terminated.

The instruction-length code is 1, 2 or 3.

The addressing-capability exception is indicated by a program-interruption code of 0136 hex (or 01B6 hex if a concurrent PER event is indicated).

Programming Note: An addressing-capability exception indicates that the program's authority to access an address space has been revoked by the owner of that address space. Depending on the programs involved, this exception may result from normal operation and does not necessarily indicate a failure on the part of the program receiving the exception.

ALEN-Translation Exception

An ALEN-translation exception is recognized during host access-register translation when the access-list-entry token used is correctly formed but does not designate a host access-list entry that is in either the valid or revoked state.

The access-list-entry token being translated is stored at real locations 168-171. If the access-list-entry token was obtained from an access register, the number of the access register is stored in bit positions 4-7 at real location 160, and bits 0-3 are set to zeros. If the access-list-entry token was obtained from a host-service parameter list, then zeros are stored in bit positions 0-7 at real location 160.

The operation is nullified.

The instruction-length code is 1, 2, or 3.

The ALEN-translation exception is indicated by a program-interruption code of 0029 hex (or 00A9 hex if a concurrent PER event is indicated).

Programming Note: An ALEN-translation exception indicates that the program attempted to use a correctly formed but currently-unassigned ALET. The ALET may have been associated with a valid host access-list entry that was subsequently deallocated.

ALET-Specification Exception

An ALET-specification exception is recognized during host access-register translation when the access-list-entry token used is not a correctly-formed ALET.

The operation is suppressed.

The instruction-length code is 1, 2, or 3.

The ALET-specification exception is indicated by a program-interruption code of 0028 hex (or 00A8 hex if a concurrent PER event is indicated).

Programming Note: An ALET-specification exception indicates that the program attempted to use as an ALET a 32-bit value that is never valid for use as an ALET. That is, the value used is never assigned by the host as the selection ALET for a valid or revoked host access-list entry. (This is in contrast to an ALEN-translation-exception condition, which indicates that the program attempted to use an ALET that might be assigned to a host access-list entry, but currently is not.) Often, this exception indicates that the program loaded an access register from storage locations not currently containing an ALET value provided by the host.

Privileged-Operation Exception

The definition for the privileged-operation exception is the same as for ESA/390, except that:

- No privileged-operation exception is recognized if the INSERT ADDRESS SPACE CONTROL instruction is executed when the extraction-authority control, bit 4 of control register 0, is zero.
- No privileged-operation exception is recognized if bits 20-23 of the second-operand address of the SET ADDRESS SPACE CONTROL or SET ADDRESS SPACE CONTROL FAST instructions have the value 0011. Instead, a specification exception is recognized.

Protection Exception

The definition for the protection exception is the same as for ESA/390, except as stated in the following.

In ESA/XC, a protection exception is not recognized due to access-list-controlled protection or page protection as defined in ESA/390. Instead, a protection exception is recognized for host access-list-controlled protection or host page protection when either of the following is true:

1. *Host Page Protection:* The CPU attempts to store into, or change explicitly the storage key of, a 4K-byte block of storage that has host page protection applied.
2. *Host Access-List-Controlled Protection:* The CPU attempts a store access, or an explicit storage-key alteration using an ALET that designates a host access-list entry that permits only fetch accesses.

If the exception is due to host page protection or host access-list-controlled protection, the operation is terminated. However, if the suppression-on-protection facility is installed, the operation may be suppressed (except for the condition code) as described in “Suppression on Protection” on page 11. Otherwise, the operation is suppressed or terminated as defined for ESA/390.

Special-Operation Exception

A special-operation exception is recognized when execution of a SET ADDRESS SPACE CONTROL or SET ADDRESS SPACE CONTROL FAST instruction that is to set the access-register mode is attempted and the address-space-function control, bit 15 of control register 0, is zero.

The operation is suppressed.

The instruction-length code is 2.

The special-operation exception is indicated by a program-interruption code of 0013 hex (or 0093 hex if a concurrent PER event is indicated).

Specification Exception

The specification exception is defined as in ESA/390, except that SET SYSTEM MASK does not give a specification exception due to bit 1 of control register 0 being one. In addition, a specification exception is recognized in ESA/XC when either of the following is true:

1. A one is introduced into bit position 5 or 16 of the PSW. This is handled as an early PSW specification exception.
2. Bits 22-23 of the second-operand address of SET ADDRESS SPACE CONTROL or SET ADDRESS SPACE CONTROL FAST are not 00 or 10.

The execution of the instruction identified by the old PSW is suppressed. However, for early PSW specification exceptions (causes 1-3 in *IBM Enterprise Systems Architecture/390 Principles of Operation*, and cause “1” on page 36 previously described) the operation that introduces the new PSW is completed, but an interruption occurs immediately thereafter.

When the exception is recognized because of an early PSW specification exception (causes 1-3 in *IBM Enterprise Systems Architecture/390 Principles of Operation*, and cause “1” on page 36 previously described), and the exception has been introduced by LOAD PSW or an interruption, the ILC is 0. When the exception is introduced by SET SYSTEM MASK or by STORE THEN OR SYSTEM MASK, the ILC is 2.

Multiple Program-Interruption Conditions

As in ESA/390, except for PER events, when multiple program-interruption conditions exist, only the condition having the highest priority is indicated in the interruption code. When two conditions have the same priority, it is unpredictable which is indicated.

The priority of all program-interruption conditions other than PER events and exceptions associated with some of the more complex control instructions is the same as defined in Figure 6-4 of *IBM Enterprise Systems Architecture/390 Principles of Operation*. In that figure, all exceptions associated with references to storage for a particular instruction halfword or a particular operand byte are grouped as a single entry called "access exceptions". Figure 5 on page 37 of this publication lists the priority of access exceptions for a single access. Thus, Figure 6-4 of *IBM Enterprise Systems Architecture/390 Principles of Operation* specifies the priority of an access-exception condition in relation to other conditions detected in the operation, and Figure 5 on page 37 of this publication specifies which of several access exceptions, encountered either in the access of a particular portion of an instruction or in any particular access associated with an operand, has highest priority.

The priority for exceptions occurring as part of tracing is the same as defined in Figure 6-7 of *IBM Enterprise Systems Architecture/390 Principles of Operation*.

For some instructions, the priority is shown in the individual instruction description.

Access Exceptions

The access exceptions consist of those exceptions that can be encountered while using an absolute, instruction, logical, or real address to access storage. Thus, in the access-register mode, the exceptions are:

1. ALET specification
2. ALEN translation
3. Addressing capability
4. Protection (host access-list controlled)
5. Addressing
6. Protection (key-controlled, host page, and low-address)

When in the primary-space mode, only exceptions “5” on page 36 and “6” on page 36 can be encountered.

Additionally, the instruction INVALIDATE PAGE TABLE ENTRY can encounter a translation-specification exception.

ASN-Translation Exceptions

None of the ESA/390 ASN-translation exceptions is applicable to ESA/XC.

A.
Protection exception (low-address protection) due to a store-type operand reference with an effective address in the range 0-511.

B.1.A.1
ALET-specification exception due to an incorrectly-formed ALET.

B.1.A.2
ALEN-translation exception due to an ALET not selecting a host access-list entry in either the valid or revoked states.

B.1.B
Translation-specification exception due to invalid encoding of bits 8-12 of control register 0. This exception is applicable only to the execution of INVALIDATE PAGE TABLE ENTRY.

B.1.C
Addressing-capability exception due to an ALET designating a host access-list entry in the revoked state.

B.2.
Protection exception (host access-list controlled protection) due to a store-type operand reference using a host access-list entry that permits read-only access.

B.3.
Addressing exception for access to instruction or operand.

B.4.
Protection exception (host page protection) due to a store-type operand reference to an address that is protected by the host against stores.

B.5.
Protection exception (key-controlled protection) due to an attempt to access a protected instruction or operand location.

Figure 5. Priority of Access Exceptions

Chapter 7. Instructions

This chapter describes the operation of instructions, other than the instructions for input/output, in ESA/XC as compared to ESA/390.

Except as specified in the following section, ESA/XC provides all general, decimal, floating point, and control instructions provided by ESA/390, and these instructions operate as described in Chapters 7 to 10, 18 and 19 of *IBM Enterprise Systems Architecture/390 Principles of Operation*.

ESA/390 Instructions Not Provided

The following instructions are provided in ESA/390 but are not provided in ESA/XC:

- BRANCH AND STACK
- BRANCH IN SUBSPACE GROUP
- EXTRACT PRIMARY ASN
- EXTRACT SECONDARY ASN
- EXTRACT STACKED REGISTERS
- EXTRACT STACKED STATE
- INSERT VIRTUAL STORAGE KEY
- LOAD ADDRESS SPACE PARAMETERS
- LOAD REAL ADDRESS
- MODIFY STACKED STATE
- MOVE TO PRIMARY
- MOVE TO SECONDARY
- PROGRAM CALL
- PROGRAM RETURN
- PROGRAM TRANSFER
- SET CLOCK
- SET SECONDARY ASN
- START INTERPRETIVE EXECUTION
- TRAP2
- TRAP4

An operation exception is recognized on an attempt to execute any of the instructions described previously.

Modified ESA/390 Instructions

The following sections describe the operation of instructions that operate differently in ESA/XC than in ESA/390.

DIAGNOSE

The DIAGNOSE instruction is used in a virtual-machine environment to request services from the host. Operation of the DIAGNOSE instruction in an ESA/XC virtual machine is documented in the publication *z/VM: CP Programming Services*.

INSERT ADDRESS SPACE CONTROL

The address-space-control bit, bit 17 of the current PSW, is placed in bit position 22 of the general register designated by the R_1 field. Bits 16-21 and 23 of the register are set to zeros, and bits 0-15 and 24-31 of the register remain unchanged. The address-space-control bit is also used to set the condition code.

Bits 16-23 and 28-31 of the instruction are ignored.

Resulting Condition Code

0
PSW bit is 17 zero (indicating primary-space mode)

1
--

2
PSW bit 17 is one (indicating access-register mode)

3
--

Programming Note: See the programming notes for the INSERT ADDRESS SPACE CONTROL instruction in *IBM Enterprise Systems Architecture/390 Principles of Operation*.

INSERT STORAGE KEY EXTENDED

This instruction operates as defined for ESA/390, with the following addition:

When the CPU is in the primary-space mode, the address of the 4K-byte block designated by the contents of the general register R_2 is a host-primary real address. When the CPU is in the access-register mode, the address of the 4K-byte block designated by the contents of general register R_2 is an AR-specified real address, interpreted as being within the address space specified by the access register designated by the R_2 field.

INVALIDATE PAGE TABLE ENTRY

This instruction operates as defined for ESA/390, with the following additions:

The real address formed from the page-table origin contained in the register designated by the R_1 field and the page index contained in the register designated by the R_2 field is treated as a host-primary real address.

Programming Note: In ESA/XC, (guest) DAT is not provided and (guest) TLB entries are never formed. This instruction provides little utility in ESA/XC; the setting of a bit in storage can be accomplished more efficiently using the OR instruction. On some models, the INVALIDATE PAGE TABLE ENTRY instruction requires a significant amount of time, and may cause a performance degradation.

LOAD ADDRESS EXTENDED

The operation is performed as defined for ESA/390, except that the value placed in access register R_1 is as shown in the following table:

PSW Bit 17	Value Placed in Access Register R_1
0	00000000 hex (zeros in bits positions 0-31)
1	If B_2 field is zero: 00000000 hex (zeros in bit positions 0-31) If B_2 field is nonzero: contents of access register B_2

LOAD PSW

This instruction operates as defined for ESA/390, with the following additional special conditions:

Special Conditions: The value that is to be loaded by the instruction is not checked for validity before it is loaded. However, immediately after loading, a specification exception is recognized and a program interruption occurs if the newly loaded PSW contains a one in bit position 5 or 16. In these cases, the operation is completed, and the resulting instruction-length code is zero. (These reasons for a specification exception are in addition to those described in *IBM Enterprise Systems Architecture/390 Principles of Operation*.)

LOAD USING REAL ADDRESS

This instruction operates as defined for ESA/390, with the following addition:

The contents of the general register designated by the R_2 field are treated as a host-primary real address.

PURGE ALB

The instruction is executed as a no-operation.

Programming Note: In ESA/XC, (guest) ART is not provided and (guest) ALB entries are never used. This instruction performs no useful function in the ESA/XC mode. On some models, this instruction requires a significant amount of time, and may cause a performance degradation, so its use should be avoided.

PURGE TLB

The instruction is executed as a no-operation.

Programming Note: In ESA/XC, (guest) DAT is not provided and (guest) TLB entries are never used. This instruction performs no useful function in the ESA/XC mode. On some models, this instruction requires a significant amount of time, and may cause a performance degradation, so its use should be avoided.

RESET REFERENCE BIT EXTENDED

This instruction operates as defined for ESA/390, with the following additions:

When the CPU is in the primary-space mode, the address of the 4K-byte block designated by the contents of the general register R_2 is a host-primary real address. When the CPU is in the access-register mode, the address of the 4K-byte block designated by the contents of general register R_2 is an AR-specified real address, interpreted as being within the address space specified by the access register designated by the R_2 field.

The reference to the storage key is subject to host page protection and host access-list-controlled protection.

RESUME PROGRAM

In ESA/XC, bit 16 of the PSW field in the second operand must be zero; otherwise, a specification exception is recognized. This exception has the same priority as the special-operation exception because of the attempt to set bit 17 of the PSW field to one when bit 15 of control register 0 is zero.

SET ADDRESS SPACE CONTROL and SET ADDRESS SPACE CONTROL FAST

Bits 20-23 of the second-operand address are used as a code to set the address-space-control bit in the PSW. The second-operand address is not used to address data; instead, bits 20-23 form the code. Bits 0-19 and 24-31 of the second-operand address are ignored. Bits 20, 21 and 23 of the second-operand address must be zeros; otherwise a specification exception is recognized. However, in the problem state, a privileged-operation exception may be recognized instead of a specification exception.

The following table summarizes the operation of SET ADDRESS SPACE CONTROL and SET ADDRESS SPACE CONTROL FAST:

Code	Name of Mode	Result in PSW bit 17
0000	Primary space	0
0010	Access register	1
Any other	Invalid	Unchanged

The address-space-function control, bit 15 of control register 0, must be one when the operation is to set the access-register mode; otherwise a special-operation exception is recognized.

For SET ADDRESS SPACE CONTROL, a serialization and checkpoint-synchronization function is performed before the operation begins and again after the operation is completed. This function is not performed for SET ADDRESS SPACE CONTROL FAST.

Special Conditions: The priority of recognition of program exceptions for the instruction is shown in Figure 6 on page 42.

Condition Code

The code remains unchanged.

Program Exceptions

- Operation (for SET ADDRESS SPACE CONTROL FAST, if the set-address-space-control-fast facility is not installed)
- Privileged operation
- Special operation
- Specification

1.-6.
Exceptions with the same priority as the priority of program-interruption conditions for the general case.

7.
Access exceptions for the second instruction halfword.

8.
Special-operation exception due to the address-space-function control, bit 15 of control register 0, being zero on an attempt to set access-register mode.

9.
Specification exception due to non-zero value in bit positions 20, 21 or 23 of the second-operand address. In the problem state, a privileged-operation exception may be recognized instead of a specification exception.

Figure 6. Priority of Execution: SET ADDRESS SPACE CONTROL and SET ADDRESS SPACE CONTROL FAST

Programming Note: See the programming notes for the SET ADDRESS SPACE CONTROL and SET ADDRESS SPACE CONTROL FAST instructions in the *IBM Enterprise Systems Architecture/390 Principles of Operation*.

SET STORAGE KEY EXTENDED

This instruction operates as defined for ESA/390, with the following additions:

When the CPU is in the primary-space mode, the address of the 4K-byte block designated by the contents of the general register R_2 is a host-primary real address. When the CPU is in the access-register mode, the address of the 4K-byte block designated by the contents of general register R_2 is an AR-specified real address, interpreted as being within the address space specified by the access register designated by the R_2 field.

The reference to the storage key is subject to host page protection and host access-list-controlled protection.

SET SYSTEM MASK

This instruction operates as defined for ESA/390, with the following modifications:

The SSM-suppression control (bit 1 of control register 0) is not checked, and no special-operation exception is recognized if the control is one.

Special Conditions: The value that is to be loaded into the PSW is not checked for validity before it is loaded. However, immediately after loading, a specification exception is recognized and a program interruption occurs if the PSW contains a one in bit position 5. In this case, the instruction is completed, and the instruction-length code is set to 2. (This reason for a specification exception is in addition to those described in *IBM Enterprise Systems Architecture/390 Principles of Operation*.)

SIGNAL PROCESSOR

This instruction operates as defined for ESA/390 with the following addition:

For the "store status at address" order, the address specified in the parameter register is a host-primary absolute address.

STORE THEN OR SYSTEM MASK

This instruction operates as defined for ESA/390, with the following additional special conditions:

Special Conditions: The value that is to be loaded into the PSW is not checked for validity before it is loaded. However, immediately after loading, a specification exception is recognized and a program interruption occurs if the PSW contains a one in bit position 5. In this case, the instruction is completed, and the instruction-length code is set to 2. (This reason for a specification exception is in addition to those described in *IBM Enterprise Systems Architecture/390 Principles of Operation*.)

STORE USING REAL ADDRESS

This instruction operates as defined for ESA/390, with the following addition:

The contents of the general register designated by the R_2 field are treated as a host-primary real address.

TEST ACCESS

The access-list-entry token (ALET) in access register R_1 is tested for exceptions recognized during host access-register translation (ART). The ALET is also tested for being 00000000 hex.

The contents of bits 0-15 of general register R_2 should be 0001 hex.

When the R_1 field is 0, the actual contents of access register 0 are used in ART, instead of the 00000000 hex that is usually used.

Bits 16-31 of general register R_2 are ignored. Bits 16-23 of the instruction are ignored.

The operation does not depend on the translation mode; bit 17 of the PSW is ignored.

When the ALET in access register R_1 is 00000000 hex, the instruction is completed by setting condition code 0.

When the ALET in access register R_1 is other than 00000000 hex, the ART process is applied to the ALET. When a situation exists that would normally cause one of the exceptions shown in the following table, the instruction is completed by setting condition code 3.

Exception Name	Cause
Addressing capability	ALET is correctly formed and selects a host access-list entry in the revoked state
ALET specification	ALET is not correctly formed

Exception Name	Cause
ALEN translation	ALET is correctly formed but does not select a host access-list entry in either the valid or revoked state

When ART is completed without one of the previous situations being recognized, the instruction is completed by setting condition code 2. ART is described in the section "Host Access-Register Translation" on page 27.

If the instruction is executed with the contents of bits 0-15 of general register R_2 not 0001 hex, the resulting condition code is unpredictable.

Special Conditions: The operation is performed only when the address-space-function control, bit 15 of control register 0, is one. When the address-space-function control is zero, it is unpredictable whether a special-operation exception is recognized.

The priority of recognition of program exceptions for the instruction is shown in Figure 7 on page 45.

Resulting Condition Code

When the contents of bits 0-15 of general register R_2 are 0001 hex:

0
Access-list-entry token (ALET) is 00000000 hex

1
--

2
ALET is not 00000000 hex and does not cause exceptions in ART

3
ALET causes exceptions in ART

When the contents of bits 0-15 of general register R_2 are not 0001 hex, the resulting condition code is unpredictable.

Program Exceptions

- Special operation

1.-6.
Exceptions with the same priority as the priority of program-interruption conditions for the general case.

7.A
Access exceptions for the second instruction halfword.

7.B
Special-operation exception due to the address-space-function control, bit 15 of control register 0, being zero.

8.
Condition code 0 due to the access-list-entry-token (ALET) being 00000000 hex.

9.
Condition code 3 due to the ALET not being correctly formed.

10.
Condition code 3 due to the ALET not selecting a host access-list entry that is in the valid or revoked state.

11.
Condition code 3 due to the ALET selecting a host access-list entry that is in the revoked state.

12.
Condition code 2

Figure 7. Priority of Execution: TEST ACCESS

TEST BLOCK

This instruction operates as defined for ESA/390, with the following additions:

When the CPU is in the primary-space mode, the address of the 4K-byte block designated by the contents of the general register R_2 is a host-primary real address. When the CPU is in the access-register mode, the address of the 4K-byte block designated by the contents of general register R_2 is an AR-specified real address, interpreted as being within the address space specified by the access register designated by the R_2 field.

The access to the block designated by the second operand address is subject to host page protection and host access-list-controlled protection.

TEST PROTECTION

The location designated by the first-operand address is tested for protection exceptions by using the access key specified in bits 24-27 of the second-operand address.

The second-operand address is not used to address data; instead, bits 24-27 of the address form the access key to be used in testing. Bits 0-23 and 28-31 of the second-operand address are ignored.

The first-operand address is a logical address. When the CPU is in the access-register mode (when PSW bit 17 is one), the first-operand address is subject to translation by means of the access-register-translation process. ART applies to the access register designated by the B_1 field to determine the address space containing the first operand.

When ART is performed and a situation exists that would normally cause one of the exceptions shown in the following table, the instruction is completed by setting condition code 3.

Exception Name	Cause
Addressing capability	ALET is correctly formed and selects a host access-list entry in the revoked state
ALET specification	ALET is not correctly formed

Exception Name	Cause
ALEN translation	ALET is correctly formed but does not select a host access-list entry in either the valid or revoked state

When the access register contains 00000000 hex, the first operand is contained in the host-primary address space and ART does not perform translation through the host access list. When the B_1 field designates access register 0, ART treats the access register as containing 00000000 hex and does not examine the actual contents of the access register.

When translation of the first-operand address can be completed, or when the CPU is in the primary-space mode, the storage key for the block designated by the first-operand address is tested against the access key specified in bits 24-27 of the second-operand address, and the condition code is set to indicate whether store and fetch accesses are permitted, taking into account all applicable protection mechanisms. Thus, for example, if low-address protection is active and the first-operand effective address is less than 512, then a store access is not permitted. Host page protection, host access-list-controlled protection, storage-protection override, and fetch-protection override are also taken into account.

The contents of storage, including the change bit, are not affected. Depending on the model, the reference bit for the first-operand address may be set to one, even for the case in which the location is protected against fetching.

Resulting Condition Code

0
Fetching permitted; storing permitted

1
Fetching permitted; storing not permitted

2
Fetching not permitted; storing not permitted

3
ALET causes exceptions in ART

Program Exceptions

- Privileged operation

TRACE

This instruction operates as defined for the ESA/390, with the following addition:

The contents of bits 1-29 of control register 12, with two zero bits appended on the right, are treated as a host-primary real address.

TRAP

The TRAP2 and TRAP4 instructions are not operational in ESA/XC. They always result in an operation or special-operation exception.

Chapter 8. Machine-Check Handling

This chapter describes how the handling of machine checks for ESA/XC differs from that defined for ESA/390. Except as described in the following section, the definition for machine-check handling provided in Chapter 11 of *IBM Enterprise Systems Architecture/390 Principles of Operation* applies to ESA/XC as well.

Handling of Machine Checks

The overall handling of machine checks in ESA/XC is the same as defined for ESA/390, except that ESA/XC provides automatic validation of certain register entities. This validation is described in the next section.

Validation

In ESA/XC, certain register entities are automatically validated as part of the machine-check interruption sequence after the original contents of the registers are placed in the appropriate save areas. After validation, the contents of these registers are restored to the values placed in the corresponding save areas, even if the associated machine-check-interruption-code validity bit for the logged-out copy is zero. The register entities for which this automatic validation is performed are:

- General registers
- Access registers
- Control registers
- Floating-point registers 0, 2, 4, and 6
- Clock comparator
- CPU timer
- Floating-point registers 1, 3, 5, and 7-15 if floating-point extensions are installed

Machine-Check Extended Interruption Information

In ESA/XC, all of the machine-check extended interruption information defined for ESA/390 is provided in the same cases as defined for ESA/390. In addition, ESA/XC provides a failing-storage ASIT for certain interruptions, as defined in the next section.

Failing-Storage Address and ASIT

As defined in ESA/390, when storage error uncorrected, storage error corrected, or storage-key error uncorrected is indicated in the machine-check-interruption code, the associated address, called the failing-storage address, is stored in bit positions 1-31 of the word at real location 248 and bit 0 of this word is set to zero. In addition, an indication of the address space in which the error occurred, called the failing-storage ASIT, is stored at locations 256-263. The failing-storage address and failing-storage ASIT fields are valid only if the failing-storage-address validity bit, bit 24 of the machine-check-interruption code, is one.

When a storage error or storage-key error occurs in the host-primary address space, zeros are stored as the failing-storage ASIT. When the storage error or storage-key error occurs in an address space other than the host-primary address space, the ASIT for the address space is stored as the failing-storage ASIT.

Chapter 9. Input/Output

This chapter describes how input/output processing for ESA/XC differs from that defined for ESA/390. Except as described in this chapter, the definition for I/O instructions, I/O functions, I/O interruptions and I/O support functions provided in Chapters 13 to 17 of *IBM Enterprise Systems Architecture/390 Principles of Operation* applies to ESA/XC as well.

Handling of Addresses for I/O

Except for the logical-address operands of the I/O instructions, all main-storage addresses used by the channel subsystem in performing I/O operations, or in performing I/O-measurement operations, refer to the host-primary address space. That is, they are host-primary absolute or host-primary real addresses, as appropriate.

More specifically, the following addresses are host-primary absolute addresses:

- Address limit specified in SET ADDRESS LIMIT
- Measurement-block origin specified in SET CHANNEL MONITOR
- Channel-program address in operation request block
- Data address in channel-command word (CCW)
- CCW address in a CCW specifying transfer in channel
- Indirect-data-address word (IDAW) address in a CCW specifying indirect data addressing
- Data address in IDAW
- CCW address in subchannel status word
- Failing-storage address in format-0 extended-status word
- Addresses of PSW and first two CCWs used for initial program loading

The following addresses are host-primary real addresses:

- Address into which TEST PENDING INTERRUPTION stores when the second-operand address is zero
- Addresses into which information is stored and from which the PSW is fetched on an I/O interruption

Appendix A. Comparison between ESA/390 and ESA/XC

This appendix provides

1. A list of the facilities that are new in ESA/XC and not provided in ESA/390
2. A description of the handling in ESA/XC of the facilities available in ESA/390
3. A list of changes between ESA/XC and ESA/390.

New Facilities in ESA/XC

The following facilities are new in ESA/XC and are not provided in ESA/390.

DAT-Off Access-Register Addressing

A variation of the ESA/390 access-register addressing facility is provided that operates in DAT-off virtual machines, allowing exploitation by z/VM applications. As in ESA/390, sixteen access registers and a translation mode called the access-register mode allow designation of storage operands in up to sixteen different address spaces by means of the B fields of instructions and the R fields of certain instructions. A host-managed table called the host access list controls the address spaces that can be accessed by means of the access registers. Through the use of host services, entries in the host access list can be established to provide addressability to address spaces owned by the virtual machine as well as address spaces owned by other virtual machines, subject to appropriate authorization. Other host services allow the addressing capability provided by an entry to be relinquished when no longer needed.

Multiple Absolute-Storage Address Spaces

Host services are provided that allow a virtual machine to obtain large amounts of storage in the form of multiple data-only absolute-storage address spaces, each up to 2 gigabytes in size. Previously, the storage available to an application program was limited to a single 2 gigabyte absolute-storage address space. With this facility, an application program has access to a practically unlimited amount of application storage. The additional address spaces are accessed by means of access-register addressing.

Address-Space Sharing

A virtual machine can use host services to make the address spaces that it owns accessible by other virtual machines on the z/VM system. This capability provides the basis for high-performance sharing of large amounts of data within a z/VM system. The shared access can be subsequently revoked when address-space sharing is no longer appropriate. The sharing virtual machines access the shared address spaces using access-register addressing.

Comparison of Facilities

Table 2 on page 52 shows the facilities offered in ESA/390 and how each facility is provided in ESA/XC.

Table 2. Availability of ESA/390 Facilities in ESA/XC

ESA/390 Facility*	Availability in ESA/XC
Access registers	B[1]
Bimodal addressing	B
Called-space identification	-
Channel subsystem I/O	B
Dual address space	-
Dynamic address translation	-
Expanded storage	O
Home address space	-
Interpretive execution	-
Linkage stack	-
Move inverse	O
Move page	P[2]
Page protection	-[3]
Private space	-[4]
Square root	O
Storage-protection override	O
Subspace group	-
Suppression on Protection	O[6]
Tracing	P[5]
31-bit logical addressing	B
31-bit real and absolute addressing	B

Explanation:

*****
All basic ESA/390 facilities not otherwise listed in this table are also basic in ESA/XC. All optional ESA/390 facilities not otherwise listed in this table are also optional in ESA/XC.

-
Not provided in ESA/XC.

1
The ESA/390 access-register-translation process is replaced by the host-access-register-translation process in ESA/XC. As viewed by application programs, these two translation processes offer comparable function.

2
Movement of data from main storage to main storage is provided, but movement of data to or from expanded storage is not. Either the move-page facility 1 or the move-page facility 2 is provided in ESA/XC.

3
The host has the ability to make selected 4K-byte blocks and the associated storage key read-only to an ESA/XC virtual machine. However, page protection per se is not provided in ESA/XC.

4
In ESA/390, the private-space facility is used to eliminate special treatment of low addresses in data-only address spaces so that all locations in such address spaces are available for application use. It is standard in ESA/XC that all locations in data-only address spaces are available for the application.

5
All parts of the tracing facility are provided except for ASN tracing.

6
Although the suppression-on-protection facility is (optionally) provided in ESA/XC, its usefulness is greatly diminished since no protection-exception conditions are guaranteed to suppress in ESA/XC.

B
Basic in ESA/XC.

P
Partially available in ESA/XC.

O
Provided as an optional facility in both ESA/390 and ESA/XC.

Summary of Changes

Changes in Instructions Provided

Table 3 on page 53 lists those instructions that are basic in ESA/390 but are not provided in ESA/XC.

Table 3. ESA/390 Instructions Not Provided in ESA/XC

Instruction Name*	Mne-monic	Op Code
BRANCH AND STACK	BAKR	B240
EXTRACT PRIMARY ASN	EPAR	B226
EXTRACT SECONDARY ASN	ESAR	B227
EXTRACT STACKED REGISTERS	EREG	B249
EXTRACT STACKED STATE	ESTA	B24A
INSERT VIRTUAL STORAGE KEY	IVSK	B223
LOAD ADDRESS SPACE PARAMETERS [1]	LASP	E500
LOAD REAL ADDRESS [1]	LRA	B1
MODIFY STACKED STATE	MSTA	B247
MOVE TO PRIMARY	MVCP	DA
MOVE TO SECONDARY	MVCS	DB
PROGRAM CALL	PC	B218
PROGRAM CALL FAST	PCF	B218
PROGRAM RETURN	PR	0101
PROGRAM TRANSFER	PT	B228
SET CLOCK [1]	SCK	B204
SET SECONDARY ASN	SSAR	B225
START INTERPRETIVE EXECUTION [2]	SIE	B214
TRAP	TRAP2	01FF
TRAP	TRAP4	B2FF

Explanation:

*****
Except as indicated, in ESA/390 these instructions can be executed successfully only when DAT is on; otherwise, a special-operation exception is recognized.

1
Instruction can be executed successfully when DAT is off.

2
Instruction can be executed successfully when DAT is off but implies the use of DAT.

Comparison of PSW Formats

In ESA/390, PSW bit 5 is the translation (DAT) control, and bits 16 and 17 are the address-space control. PSW bits 16 and 17 are ignored when DAT is off.

In ESA/XC, PSW bits 5 and 16 are unassigned; a one in either bit position is invalid. PSW bit 17 is the address-space control.

Changes in Control-Register Assignments

Table 4 on page 54 shows those control-register bits and fields assigned in ESA/390 but unassigned in ESA/XC.

Table 4. ESA/390 Control-Register Fields Not Assigned in ESA/XC		
Ctrl Reg	**Bits**	**Name of Bit or Field**
0	1 2 5 27	SSM-suppression control TOD-clock-sync control Secondary-space control ETR Subclass mask
1	0 1-19 22 23 24 25-31	Primary space-switch-event control Primary segment-table origin Primary subspace-group control Primary private-space control Primary storage-alteration-event control Primary segment-table length
3	16-31	Secondary ASN
4	0-15 16-31	Authorization index Primary ASN
5	1-25	Primary-ASTE origin [1]
7	1-19 22 23 24 25-31	Secondary segment-table origin Secondary subspace-group control Secondary private-space control Secondary storage-alteration-event control Secondary segment-table length
8	0-15	Extended authorization index
12	30	ASN-trace control
13	0 1-19 23 24 25-31	Home space-switch-event control Home segment-table origin Home private-space control Home storage-alteration-even control Home segment-table length
14	10 12 13-31	TOD-clock-control-override control ASN-translation control ASN-first-table origin
15	1-28	Linkage-stack entry address

Explanation:

1

This assignment applies only if bit 15 of control register 0 is one. If bit 15 is zero, control register 5 contains the linkage-table designation as in 370-XA.

Changes in Assigned Storage Locations

Table 5 on page 54 summarizes the changes in assigned storage locations between ESA/390 and ESA/XC.

Table 5. Changes in Assigned Storage Locations		
Name of Field	**Assigned Storage Location and Length* for**	
	ESA/390	**ESA/XC**
Exception ALET	-	168 4

Table 5. Changes in Assigned Storage Locations (continued)		
Name of Field	**Assigned Storage Location and Length* for**	
	ESA/390	**ESA/XC**
Failing-storage ASIT	-	256 8[1]

Explanation:

*****
The first number is the real address, the second is the length.

-
Not assigned

1
This field is assigned within the area defined as the fixed-logout area in ESA/390

Changes in Exceptions

ESA/XC includes an exception, the addressing-capability exception, that is not defined in ESA/390. The addressing-capability exception may be recognized during access-register translation and is indicated by a program-interruption code of 0136 hex.

In addition, there are several exceptions defined in ESA/390 but never recognized in ESA/XC. These exceptions are summarized in Table 6 on page 55.

Table 6. ESA/390 Exceptions Not Recognized	
Exception Name	**Interruption Code (hex)**
AFX-translation	0020
ALE-sequence	002A
ASN-translation specification	0017
ASTE-sequence	002C
ASTE-validity	002B
ASX-translation	0021
EX-translation	0023
Extended-authority	002D
LX-translation	0022
Page-translation	0011
PC-translation specification	001F
Primary-authority	0024
Secondary-authority	0025
Segment-translation	0010
Space-switch event	001C
Stack-empty	0031
Stack-full	0030
Stack-operation	0034
Stack-specification	0032
Stack-type	0033

Changes to Insert Address Space Control

In ESA/390, INSERT ADDRESS SPACE CONTROL can be executed successfully only when DAT is on and further, when in the problem state, only when the extraction-authority control (bit 4 of control register 0) is one. In ESA/XC, the instruction can be executed successfully even though DAT is not provided, and it is not subject to the control by bit 4 of control register 0.

Changes to Resume Program

In ESA/390, RESUME PROGRAM can set any value in PSW bits 16-17 with the proper authorization. In ESA/XC, PSW bit 16 may not be set to one.

Changes to Set Address Space Control (SASC) and SASC Fast

In ESA/390, SET ADDRESS SPACE CONTROL and SET ADDRESS SPACE CONTROL FAST can be executed successfully only when DAT is on. Also, SET ADDRESS SPACE CONTROL, and unpredictably SET ADDRESS SPACE CONTROL FAST, can be executed successfully in ESA/390 only when the secondary-space control (bit 5 of control register 0) is one. The instructions can be used to set the primary-space mode, the secondary-space mode, the access-register mode and the home-space mode.

In ESA/XC, the instructions can be executed successfully even though DAT is not provided, and they are not subject to the secondary-space control. The instructions can be used to set only the primary-space mode and the access-register mode. As in ESA/390, the address-space-function control (bit 15 of control register 0) must be one in order to set the access-register mode.

Changes to Set System Mask

In ESA/390, SET SYSTEM MASK can be executed successfully only when the SSM-suppression control (bit 1 of control register 0) is zero. In ESA/XC, bit 1 of control register 0 is not checked.

Changes to Test Access

In ESA/390, TEST ACCESS can be executed successfully only when DAT is on. In ESA/XC, the instruction can be executed successfully even though DAT is not provided.

Changes to Trap

In ESA/XC, TRAP2 and TRAP4 always result in an operation or special-operation exception.

Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information may contain examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information may contain sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Programming Interface Information

This manual documents intended Programming Interfaces that allow the customer to write programs to obtain services of z/VM.

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at IBM copyright and trademark information - United States (www.ibm.com/legal/us/en/copytrade.shtml).

The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

Other company, product, and service names may be trademarks or service marks of others.

Terms and Conditions for Product Documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

Personal Use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial Use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

IBM Online Privacy Statement

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user, or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see:

- IBM Privacy Statement at http://www.ibm.com/privacy/us/en/
- IBM Online Privacy Statement at http://www.ibm.com/privacy/details/us/en/ in the section entitled "Cookies, Web Beacons and Other Technologies"

Bibliography

This topic lists the publications in the z/VM library. For abstracts of the z/VM publications, see *z/VM: General Information*.

Where to Get z/VM Information

The current z/VM product documentation is available in IBM Knowledge Center - z/VM (www.ibm.com/support/knowledgecenter/SSB27U).

z/VM Base Library

Overview

- *z/VM: License Information*, GI13-4377
- *z/VM: General Information*, GC24-6286

Installation, Migration, and Service

- *z/VM: Installation Guide*, GC24-6292
- *z/VM: Migration Guide*, GC24-6294
- *z/VM: Service Guide*, GC24-6325
- *z/VM: VMSES/E Introduction and Reference*, GC24-6336

Planning and Administration

- *z/VM: CMS File Pool Planning, Administration, and Operation*, SC24-6261
- *z/VM: CMS Planning and Administration*, SC24-6264
- *z/VM: Connectivity*, SC24-6267
- *z/VM: CP Planning and Administration*, SC24-6271
- *z/VM: Getting Started with Linux on IBM Z*, SC24-6287
- *z/VM: Group Control System*, SC24-6289
- *z/VM: I/O Configuration*, SC24-6291
- *z/VM: Running Guest Operating Systems*, SC24-6321
- *z/VM: Saved Segments Planning and Administration*, SC24-6322
- *z/VM: Secure Configuration Guide*, SC24-6323
- *z/VM: TCP/IP LDAP Administration Guide*, SC24-6329
- *z/VM: TCP/IP Planning and Customization*, SC24-6331
- z/OS and z/VM: Hardware Configuration Manager User's Guide (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r4sc342670/$file/eequ100_v2r4.pdf), SC34-2670

Customization and Tuning

- *z/VM: CP Exit Customization*, SC24-6269
- *z/VM: Performance*, SC24-6301

Operation and Use

- *z/VM: CMS Commands and Utilities Reference*, SC24-6260
- *z/VM: CMS Primer*, SC24-6265
- *z/VM: CMS User's Guide*, SC24-6266
- *z/VM: CP Commands and Utilities Reference*, SC24-6268
- *z/VM: System Operation*, SC24-6326
- *z/VM: TCP/IP User's Guide*, SC24-6333
- *z/VM: Virtual Machine Operation*, SC24-6334
- *z/VM: XEDIT Commands and Macros Reference*, SC24-6337
- *z/VM: XEDIT User's Guide*, SC24-6338

Application Programming

- *z/VM: CMS Application Development Guide*, SC24-6256
- *z/VM: CMS Application Development Guide for Assembler*, SC24-6257
- *z/VM: CMS Application Multitasking*, SC24-6258
- *z/VM: CMS Callable Services Reference*, SC24-6259
- *z/VM: CMS Macros and Functions Reference*, SC24-6262
- *z/VM: CMS Pipelines User's Guide and Reference*, SC24-6252
- *z/VM: CP Programming Services*, SC24-6272
- *z/VM: CPI Communications User's Guide*, SC24-6273
- *z/VM: ESA/XC Principles of Operation*, SC24-6285
- *z/VM: Language Environment User's Guide*, SC24-6293
- *z/VM: OpenExtensions Advanced Application Programming Tools*, SC24-6295
- *z/VM: OpenExtensions Callable Services Reference*, SC24-6296
- *z/VM: OpenExtensions Commands Reference*, SC24-6297
- *z/VM: OpenExtensions POSIX Conformance Document*, GC24-6298
- *z/VM: OpenExtensions User's Guide*, SC24-6299
- *z/VM: Program Management Binder for CMS*, SC24-6304
- *z/VM: Reusable Server Kernel Programmer's Guide and Reference*, SC24-6313
- *z/VM: REXX/VM Reference*, SC24-6314
- *z/VM: REXX/VM User's Guide*, SC24-6315
- *z/VM: Systems Management Application Programming*, SC24-6327
- *z/VM: TCP/IP Programmer's Reference*, SC24-6332
- *CPI Communications Reference*, SC26-4399
- *Common Programming Interface Resource Recovery Reference*, SC31-6821
- z/OS: IBM Tivoli Directory Server Plug-in Reference for z/OS (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r4sa760169/$file/glpa300_v2r4.pdf), SA76-0169
- z/OS: Language Environment Concepts Guide (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r4sa380687/$file/ceea800_v2r4.pdf), SA38-0687
- z/OS: Language Environment Debugging Guide (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r4ga320908/$file/ceea100_v2r4.pdf), GA32-0908
- z/OS: Language Environment Programming Guide (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r4sa380682/$file/ceea200_v2r4.pdf), SA38-0682
- z/OS: Language Environment Programming Reference (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r4sa380683/$file/ceea300_v2r4.pdf), SA38-0683

- z/OS: Language Environment Runtime Messages (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r4sa380686/$file/ceea900_v2r4.pdf), SA38-0686
- z/OS: Language Environment Writing Interlanguage Communication Applications (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r4sa380684/$file/ceea400_v2r4.pdf), SA38-0684
- z/OS: MVS Program Management Advanced Facilities (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r4sa231392/$file/ieab200_v2r4.pdf), SA23-1392
- z/OS: MVS Program Management User's Guide and Reference (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r4sa231393/$file/ieab100_v2r4.pdf), SA23-1393

Diagnosis

- *z/VM: CMS and REXX/VM Messages and Codes*, GC24-6255
- *z/VM: CP Messages and Codes*, GC24-6270
- *z/VM: Diagnosis Guide*, GC24-6280
- *z/VM: Dump Viewing Facility*, GC24-6284
- *z/VM: Other Components Messages and Codes*, GC24-6300
- *z/VM: TCP/IP Diagnosis Guide*, GC24-6328
- *z/VM: TCP/IP Messages and Codes*, GC24-6330
- *z/VM: VM Dump Tool*, GC24-6335
- z/OS and z/VM: Hardware Configuration Definition Messages (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r4sc342668/$file/cbdm100_v2r4.pdf), SC34-2668

z/VM Facilities and Features

Data Facility Storage Management Subsystem for VM

- *z/VM: DFSMS/VM Customization*, SC24-6274
- *z/VM: DFSMS/VM Diagnosis Guide*, GC24-6275
- *z/VM: DFSMS/VM Messages and Codes*, GC24-6276
- *z/VM: DFSMS/VM Planning Guide*, SC24-6277
- *z/VM: DFSMS/VM Removable Media Services*, SC24-6278
- *z/VM: DFSMS/VM Storage Administration*, SC24-6279

Directory Maintenance Facility for z/VM

- *z/VM: Directory Maintenance Facility Commands Reference*, SC24-6281
- *z/VM: Directory Maintenance Facility Messages*, GC24-6282
- *z/VM: Directory Maintenance Facility Tailoring and Administration Guide*, SC24-6283

Open Systems Adapter

- Open Systems Adapter-Express Customer's Guide and Reference (www.ibm.com/support/pages/open-systems-adapter-express-customers-guide-and-reference-0), SA22-7935
- Open Systems Adapter-Express Integrated Console Controller User's Guide (www.ibm.com/support/pages/node/6019810), SC27-9003
- Open Systems Adapter-Express Integrated Console Controller 3215 Support (www.ibm.com/support/knowledgecenter/en/SSLTBW_2.1.0/com.ibm.zos.v2r1.ioa/ioa.htm), SA23-2247
- Open Systems Adapter/Support Facility on the Hardware Management Console (www.ibm.com/support/knowledgecenter/en/SSLTBW_2.1.0/com.ibm.zos.v2r1.ioa/ioa.htm), SC14-7580

Performance Toolkit for z/VM

- *z/VM: Performance Toolkit Guide*, SC24-6302
- *z/VM: Performance Toolkit Reference*, SC24-6303

RACF® Security Server for z/VM

- *z/VM: RACF Security Server Auditor's Guide*, SC24-6305
- *z/VM: RACF Security Server Command Language Reference*, SC24-6306
- *z/VM: RACF Security Server Diagnosis Guide*, GC24-6307
- *z/VM: RACF Security Server General User's Guide*, SC24-6308
- *z/VM: RACF Security Server Macros and Interfaces*, SC24-6309
- *z/VM: RACF Security Server Messages and Codes*, GC24-6310
- *z/VM: RACF Security Server Security Administrator's Guide*, SC24-6311
- *z/VM: RACF Security Server System Programmer's Guide*, SC24-6312
- *z/VM: Security Server RACROUTE Macro Reference*, SC24-6324

Remote Spooling Communications Subsystem Networking for z/VM

- *z/VM: RSCS Networking Diagnosis*, GC24-6316
- *z/VM: RSCS Networking Exit Customization*, SC24-6317
- *z/VM: RSCS Networking Messages and Codes*, GC24-6318
- *z/VM: RSCS Networking Operation and Use*, SC24-6319
- *z/VM: RSCS Networking Planning and Configuration*, SC24-6320
- z/OS: Network Job Entry (NJE) Formats and Protocols (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r4sa320988/$file/hasa600_v2r4.pdf), SA32-0988

Prerequisite Products

Device Support Facilities

- Device Support Facilities (ICKDSF): User's Guide and Reference (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r4gc350033/$file/ickug00_v2r4.pdf), GC35-0033

Environmental Record Editing and Printing Program

- Environmental Record Editing and Printing Program (EREP): Reference (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r4gc350152/$file/ifc2000_v2r4.pdf), GC35-0152
- Environmental Record Editing and Printing Program (EREP): User's Guide (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r4gc350151/$file/ifc1000_v2r4.pdf), GC35-0151

Other Publications

IBM Enterprise Systems Architecture/390 Principles of Operation, SA22-7201
IBM Enterprise Systems Architecture/370 Principles of Operation, SA22-7200
IBM System/370 Extended Architecture Principles of Operation, SA22-7085

Index

D

E

F

G

H

I

K

L

M

P

R

S

T

U

V

Z

Product Number: 5741-A09

Printed in USA - Product Number: 5741-A09

SC24-6285-01