z/VM
7.3

z/Architecture Extended Configuration (z/XC) Principles of Operation

SC27-4940-73

Note:

Before you use this information and the product it supports, read the information in "Notices" on page 61.

This edition applies to version 7, release 3 of IBM® z/VM® (product number 5741-A09) and to all subsequent releases and modifications until otherwise indicated in new editions.

Last updated: 2023-11-28

Contents

Figures

Tables

About This Document

IBM z/Architecture® Extended Configuration (z/XC) virtual-machine architecture, as provided by z/VM, is described in detail. z/XC virtual machine operation is described as it appears to an assembly language programmer. Because z/XC is based on and is closely related to z/Architecture, z/XC is defined by indicating the ways in which it is the same as, or different from, z/Architecture.

The following elements of the z/XC architecture are covered:

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

Intended Audience

This information is for programmers who write or debug programs that run in z/XC virtual machines.

You must have a basic familiarity with IBM z/Architecture, as described in IBM z/Architecture Principles of Operation.

You must also have a basic knowledge of IBM assembly language and have experience with z/VM programming concepts and techniques.

Conventions

This information is intended to be used with the definition of z/Architecture that is provided in *IBM z/Architecture Principles of Operation*. Where possible, the z/XC principles are presented by using the same style and general organization as *IBM z/Architecture Principles of Operation*. The following table shows the relationship of the chapters in this document to the chapters in *IBM z/Architecture Principles of Operation*.

Chapter in this document	**Corresponding chapters in *IBM z/Architecture Principles of Operation*, SA22-7832**
Chapter 1, “Introduction,” on page 1	Chapter 1, Introduction
Chapter 2, “Organization,” on page 7	Chapter 2, Organization
Chapter 3, “Storage,” on page 9	Chapter 3, Storage
Chapter 4, “Control,” on page 21	Chapter 4, Control
Chapter 5, “Program Execution,” on page 27	Chapter 5, Program Execution
Chapter 6, “Interruptions,” on page 39	Chapter 6, Interruptions

Chapter in this document	Corresponding chapters in *IBM z/Architecture Principles of Operation*, SA22-7832
Chapter 7, “Instructions,” on page 47	Chapter 7, General Instructions Chapter 10, Control Instructions
Chapter 8, “Machine-Check Handling,” on page 57	Chapter 11, Machine-Check Handling
Chapter 9, “Input/Output,” on page 59	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: No material in this document corresponds to the following chapters in *IBM z/Architecture Principles of Operation*: Chapter 8, Decimal Instructions Chapter 9, Floating-Point Overview and Support Instructions Chapter 12, Operator Facilities Chapter 18, Hexadecimal-Floating-Point Instructions Chapter 19, Binary-Floating-Point Instructions Chapter 20, Decimal-Floating-Point Instructions Chapter 21, Vector Overview and Support Instructions Chapter 22, Vector Integer Instructions Chapter 23, Vector String Instructions Chapter 24, Vector Floating-Point Instructions Chapter 25, Vector Decimal Instructions Chapter 26, Specialized-Function-Assist Instructions	

Multiplier Prefix Symbols for Bytes

In this information, as in the *IBM z/Architecture Principles of Operation* publication, the letters K, M, and G denote multipliers of powers of 2.

Symbol	Value
K (kilo)	$1,024=2^{10}$
M (mega)	$1,048,576 = 2^{20}$
G (giga)	$1,073,741,824 = 2^{30}$

A multiplier symbol can be prefixed to the abbreviation for byte (B). For example:

- 4,096 bytes is expressed as 4 KB.
- 1,048,576 bytes is expressed as 1 MB.

Where to Find More Information

In addition to the *IBM z/Architecture Principles of Operation* publication, the following documents in the z/VM library might be useful in understanding z/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 65.

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 provide feedback to IBM

We welcome any feedback that you have, including comments on the clarity, accuracy, or completeness of the information. See How to send feedback to IBM for additional information.

Summary of Changes for z/VM: z/XC 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.

SC24-4940-73, z/VM 7.3 (December 2023)

This edition includes terminology, maintenance, and editorial changes.

SC27-4940-73, z/VM 7.3 (September 2022)

This edition supports the general availability of z/VM 7.3. Note that the publication number suffix (-73) indicates the z/VM release to which this edition applies.

Chapter 1. Introduction

This document describes, for reference purposes, the operation of virtual machines that run in the z/Architecture Extended Configuration (z/XC) virtual-machine architecture as provided by z/VM. It is organized as a supplement to the definition of z/Architecture that is described in *IBM z/Architecture Principles of Operation*.

If a particular element of the architecture (instruction, operation, program exception, and so on) is described in this document, then the definition that is contained in this document supersedes the definition in *IBM z/Architecture Principles of Operation*. If a particular element of the architecture is not discussed in this document, then the definition that is contained in *IBM z/Architecture Principles of Operation* applies to z/XC.

z/XC comprises the architectural facilities that are available to the program that is running in the virtual machine. z/XC virtual machines operate under the control of the z/VM Control Program (CP), a host program that runs in the z/Architecture 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 z/XC

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

z/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 that runs in the virtual machine is not an elaborate control program, but rather an application-program monitor. Consistent with such a virtual-machine supervisor, z/XC allows application programs to benefit from architectural extensions that are provided by z/Architecture without requiring the virtual-machine supervisor to complete complex management functions. Instead, the host bears the responsibility for the support and management of the real-machine facilities.

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

When compared to z/Architecture, z/XC offers the following extensions:

- *DAT-off access-register addressing* makes the access-register-addressing facility of z/Architecture available to z/VM applications. The facility can be used by a virtual machine to access its own address spaces and 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 absolute-storage address space. With this facility, an application can have multiple 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 any of its address spaces (the instruction address space, or any data-only address spaces) by using access-register addressing. Later, the shared access can be revoked when appropriate.

Additions to z/XC

The following features might be available in a z/Architecture machine.

Configuration-z/XC-Mode Facility

In lieu of the configuration-z/Architecture-architectural-mode (CZAM) facility of z/Architecture, the configuration-z/XC-mode (CZXM) facility can be present on a model that implements z/XC. When the facility is installed in a configuration, the configuration is reset into the z/XC architectural mode (rather than into the ESA/XC architectural mode). When the facility is installed in a configuration, the configuration cannot be switched into the ESA/XC architectural mode.

The z/Architecture Base

z/XC includes, as its base, most of the facilities that are available to z/VM virtual machines in the original z/Architecture and its extensions. However, some of the facilities that are available in z/Architecture are not provided in z/XC.

The most significant difference between z/Architecture and z/XC is that in z/XC, guest dynamic address translation (DAT) is not provided. So, z/XC does not have the following features:

- Guest virtual-storage address spaces are not provided. Therefore, the following features are not provided:
  - Guest region, segment, or page tables
  - Guest translation-lookaside buffer
  - Guest primary, secondary, and home spaces
  - Secondary-space and home-space modes
- The following instructions are not provided because they depend on DAT:
  - INSERT VIRTUAL STORAGE KEY
  - LOAD REAL ADDRESS
  - MOVE TO PRIMARY
  - MOVE TO SECONDARY
  - STORE REAL ADDRESS
  - TRAP
- The ASN-translation and PC-number translation processes do not occur. Therefore, the following features are not provided:
  - Guest ASN first tables
  - Guest ASN second tables
  - Guest authority tables
  - Guest linkage tables
  - Guest entry tables

  And the following instructions are not provided:
  - EXTRACT PRIMARY ASN
  - EXTRACT SECONDARY ASN
  - LOAD ADDRESS SPACE PARAMETERS
  - PROGRAM CALL
  - PROGRAM TRANSFER
  - SET SECONDARY ASN
- Guest access-register translation is not provided. Therefore, no guest access lists or dispatchable-unit-control tables are provided. A guest ART-lookaside buffer is not provided. The BRANCH IN SUBSPACE GROUP and EXTRACT AND SET EXTENDED AUTHORITY instructions are not provided.

  Host access-register translation is provided as a comparable replacement for guest access-register translation.

- The linkage stack is not provided. Therefore, the following instructions are not provided:
  - BRANCH AND STACK
  - EXTRACT STACKED REGISTERS
  - EXTRACT STACKED STATE
  - MODIFY STACKED STATE
  - PROGRAM RETURN
- Facilities that depend on DAT are not provided:
  - ASN-and-LX reuse facility
  - DAT-enhancement facility 2
  - Enhanced-DAT facilities 1 and 2
  - Enhanced-monitor facility
  - Instruction-execution-protection facility
  - Local-TLB-clearing facility
  - Move-with-optional-specifications facility

  The instructions in DAT-enhancement facility 1 are provided but complete no useful function because TLB and ALB are not provided.
- z/XC does not provide the START INTERPRETIVE EXECUTION instruction.
- The collaborative-memory-management facility that is defined in *z/VM: CP Programming Services* is not provided in z/XC.

Instructions that are introduced by these excluded functions and facilities are not provided. An attempt to use an instruction that is not provided in z/XC results in either an operation exception or a special-operation exception. Fields that are defined for these facilities in the PSW, control registers, data structures, and assigned storage locations are reserved. The program must set fields for the unsupported facilities to zeros to ensure compatible operation in the future.

Note: The listed facilities that are not provided in the virtual machine, and the extensions of those facilities, are available to the host program. Only the architecture of the virtual-machine environment is documented in detail. Host facilities are referenced only as needed to clearly describe the programming environment that is provided by the virtual machine.

Except for those facilities that are identified as not provided, z/XC includes all facilities that are defined in z/Architecture and supported by z/VM for z/Architecture guests.

System Program

z/XC is typically used with a virtual-machine supervisor program, such as z/CMS, that cooperates with the host to provide application-level service interfaces for application programs that run in a single virtual machine. A z/XC virtual machine operates under the control of the z/VM Control Program (CP), which runs in the z/Architecture processor complex. The z/VM Control Program acts as the host program, manages the running of virtual machines, and provides system-level services to the virtual machines.

Compatibility

Compatibility among z/XC Implementations

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

1. The program is not time-dependent.

2. The program does not depend on system facilities when the facilities are not included in the configuration or are not provided by the host program implementation. System facilities include storage capacity, I/O equipment, optional machine facilities, and optional or release-dependent host-program facilities.
3. The program does not depend on the absence of system facilities when the facilities are included in the configuration. For example, the program must not depend on interruptions that are 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 that are associated with uninstalled facilities. For example, data must not be placed in an area that is 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, and so on) that are not explicitly made available for program use.
4. The program does not depend on results or functions that are defined to be unpredictable or model-dependent or are identified as undefined. The program must not depend on the assignment of device numbers and CPU addresses.
5. The program 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 z/Architecture where those results or functions of z/Architecture are applicable in z/XC.
6. The program accounts for any changes that are made to the architecture that affect compatibility.

Compatibility between z/XC and ESA/XC

Control Program Compatibility

A control program that is written for ESA/XC cannot be directly transferred to systems that operate as defined by z/XC architecture because the following elements are changed:

- General-register and control-register sizes
- PSW size
- Scope of prefixing
- Assigned storage locations

Application Program Compatibility

A high degree of compatibility exists at the problem-state level in transferring from ESA/XC to z/XC. Because most of a user's applications are written by using only problem-state instructions, this problem-state compatibility is useful in many installations.

Most privileged instructions that are used in applications on CMS (the control program for which the XC architectures are designed) in ESA/XC are compatible with z/XC. If the application relies on its control program (CMS or z/CMS) to deal with architectural differences, it can run in z/XC as it does in ESA/XC.

A problem-state program that is written for ESA/XC operates with z/XC if the program meets the following criteria:

1. The program complies with the limitations that are described in "Compatibility among z/XC Implementations" on page 3.
2. The program is not dependent on host or virtual-machine supervisor facilities that are not available on the system.
3. The program is not sensitive to architectural differences between ESA/XC and z/XC. The differences are generally a subset of the differences between ESA/390 and z/Architecture, such as the size and layout of the prefix area and the assigned storage locations therein.
4. The program is not dependent on withholding authorization for access-register translation through the TEST ACCESS, SET ACCESS CONTROL, and SET ACCESS CONTROL FAST instructions through the address-space-function control in control register 0. The address-space-function control is not

provided in z/XC. The bit that ESA/XC uses for address-space-function control corresponds to a bit in z/XC that is reserved.

5. The program does not depend on the TOD clock in a z/XC configuration to be synchronized with the TOD clock of the host or other ESA/XC or z/XC virtual machines. (Unlike ESA/XC, z/XC provides the SET CLOCK and PERFORM TIMING FACILITY instructions and analogous operator commands, which allow guest and host TOD clocks to differ.)

Programming notes:

1. This publication and the *z/VM: ESA/XC Principles of Operation* assign meanings to various operation codes, to bit positions in instructions, channel-command words, registers, and table entries, and to fixed locations in the low 512 bytes and bytes 4096-8191 of storage. Unless noted, the remaining operation codes, bit positions, and low-storage locations are reserved for future assignment to new facilities and other extensions of the architecture.

   Observe the following guidelines to ensure that existing programs continue to operate when such new facilities are installed:

   - Programs must not depend on an indication of an exception as a result of invalid values that are currently defined as checked.
   - If a value must be placed in unassigned positions that are not checked, the program must enter zeros.
   - When the machine provides a code or field, the program must account for new codes and bits that might be assigned in the future.
   - The program must not use unassigned low-storage locations for storing information. Low-storage locations might be assigned in the future in such a way that the machine causes the contents of the locations to be changed.
2. Application programs that run in z/CMS are shielded from some architectural differences. For example, z/CMS accommodates programs that expect to work with interruption PSWs in the format and locations that are defined in ESA/390 and ESA/XC.

   In other cases, applications that are sensitive to architectural details might need to be adjusted to run in z/XC on z/CMS. For example, the size and location of the translation-exception ID that is stored on a protection exception differs between ESA/XC and z/XC. Also, the size and location of the failing-storage address that is stored on a machine-check interruption differs between ESA/XC and z/XC.

Compatibility among z/XC, z/Architecture and ESA/390

Control Program Compatibility

A control program that is written for z/Architecture that uses the dynamic-address-translation (DAT) facility cannot be transferred to a z/XC virtual machine because z/XC does not provide DAT. Also, a program that uses z/Architecture instructions that are not provided in z/XC cannot be transferred to z/XC.

A control program that is written for z/Architecture can run in a z/XC virtual machine if the control program meets the following criteria:

- The control program does not use the DAT facilities.
- The control program complies with the limitations that are described in "Compatibility among z/XC Implementations" on page 3.
- The control program does not depend on receiving a special-operation exception in the following situation: A control instruction that in z/Architecture requires DAT is attempted in a z/XC virtual machine.
- The control program does not use z/Architecture instructions that are not provided in z/XC. For more information, see "The z/Architecture Base" on page 2.

Problem State Compatibility

A high degree of compatibility exists at the problem-state level for transferring applications from ESA/390 or z/Architecture to z/XC. Because most of a user's applications are written by using only problem-state instructions, this problem-state compatibility is useful in many installations.

A problem-state application that is written for ESA/390 or z/Architecture operates with z/XC if the application meets the following criteria:

1. The application complies with the limitations that are described in "Compatibility among z/XC Implementations" on page 3.
2. The application is not dependent on host or virtual-machine supervisor facilities that are not available on the system.
3. The application accounts for other changes that are made to the z/Architecture architectural definition that affect compatibility among the z/XC, z/Architecture, and ESA/390 modes.

Chapter 2. Organization

The basic organization of a z/XC virtual machine follows the definition of system organization that is provided in Chapter 2 of *IBM z/Architecture Principles of Operation*. In addition, the configuration of a z/XC virtual machine is extended to provide access to additional host address spaces, which can be private or shared with other virtual machines.

A z/XC virtual-machine configuration contains all of the basic organizational elements that are defined for z/Architecture systems: main storage, one or more central processing units (CPUs), operator facilities, a channel subsystem, and I/O devices. In addition, a z/XC configuration can have more 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 a z/XC virtual-machine configuration are regulated by host controls that are available to the z/VM installation. For example, the total main storage that is 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.

In contrast to the z/Architecture definition, the main storage of a z/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 provide shared data for a collection of virtual machines that run on a single z/VM system.

Main Storage

As in z/Architecture, 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 that are 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, by using host services, the host-primary address space can be made directly addressable by the CPUs of other virtual machines.

More absolute-storage address spaces can be added to the virtual configuration by using host services. The 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 with appropriate authorization.

Central Processing Unit

A central processing unit (CPU) of a z/XC virtual configuration includes all of the registers that are provided in z/Architecture.

Access Registers

As in z/Architecture, z/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 *access-register mode*, an instruction B field specifies whether a logical or real address for a storage-operand reference designates an access register. The storage-operand is considered to be within the absolute-storage address space that is 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 by 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 concurrently designate the instruction space and a maximum of 15 other address spaces.

Host Access List

A z/XC virtual machine has a host-primary address space. A host access list specifies additional address spaces, which are available when the CPU is in the access-register mode. The host access list contains a directory-specified number of host access list entries. Each entry either designates a particular address space or is not used.

By using 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

A z/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 on the processor complex and manages the running of virtual machines, which can be of several different architectures. The host provides special service functions in addition to the facilities that are described by the architectures.

The extended addressing capabilities of z/XC are under the control of the host, which determines the extent to which a particular z/XC virtual machine can use these extended capabilities. Services are provided by which a z/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

The structure of main storage, including addressing and protection aspects for a z/XC virtual machine, is described. The handling of storage for a z/XC virtual machine follows the definition of storage that is provided in Chapter 3 of *IBM z/Architecture Principles of Operation*, with the exceptions that are described in the following topics.

Terminology note: Because most references to storage for a z/XC virtual machine apply to (guest) real storage, the abbreviated term *storage* is often used in place of *real storage*. The term *storage* is sometimes used in place of *main storage* or *absolute storage* when the meaning is clear.

Storage Addressing

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

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 that are assigned to those byte locations.

Terminology note: Because all references to address spaces for a z/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 that start at 0, and the addresses are always assigned in complete 4 KB 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 might not identify a unique byte because a particular byte address can be associated with more than one absolute-storage address space.

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

A virtual machine can obtain more absolute-storage address spaces by using the ADRSPACE CREATE host service. These additional absolute-storage address spaces can later be destroyed by using the ADRSPACE DESTROY host service. A subsystem reset also destroys all absolute-storage address spaces that were created by the ADRSPACE CREATE service.

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

The number and total size of absolute-storage address spaces that are permitted for the virtual machine are 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 using ADRSPACE CREATE are 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.

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. By using the ADRSPACE PERMIT host service, a private address space can be transformed into a shareable address space, subject to a directory-specified authorization to share address spaces. Then, by using the ADRSPACE ISOLATE host service, a shareable address space can be transformed into a private address space. A subsystem reset also transforms 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 an 8-byte identifying value that is 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 that is associated with a particular absolute-storage address space is fixed from the time the address space is created by the host until the ASIT is destroyed.

The host assigns ASITs in such a way that a particular ASIT value is associated with at most one absolute-storage address space for the scope of the host IPL. That is, after a particular ASIT value is assigned to an absolute-storage address space, that ASIT value is not reassigned to another absolute-storage address space within the scope of the same host IPL. The ASIT value is not reassigned, even if the absolute-storage address space to which the ASIT value was originally assigned is 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 has no significance to z/XC programs.

Address Types and Formats

Address Types

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

Absolute Address

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

Host-Primary Absolute Address

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

AR-Specified Absolute Address

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

Real Address

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

When a type-R real address is used for access to main storage, it is converted to an absolute address by using prefixing. Low-address protection is applied to storage references that are made by using a type-R real address, if the low-address-protection control in control register 0 is one. Fetch-protection checking is performed for storage references that are made by 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 reference that is made by 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 that is made by 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 that are 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 that is 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 that is contained within an address space that is determined by host access-register translation. An AR-specified real address is considered to be a type-R real address when the ALET is 00000000 hex. An AR-specified real address is considered to be a type-A real address when the ALET is other than 00000000 hex.

Virtual Addresses

Because the dynamic-address-translation facility is not provided in z/XC, virtual addresses and virtual storage are not available for z/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 that are associated with the instruction, that 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 that are 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 set of addresses that are treated as instruction addresses is the same in z/XC as in z/Architecture.

Protection

Four protection facilities protect the contents of main storage from destruction or misuse by programs that contain errors or are unauthorized:

1. Key-controlled protection
2. Host access-list-controlled protection
3. Host DAT protection
4. Low-address protection

These protection facilities are applied independently. Access to main storage is authorized only 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 DAT protection, and low-address protection afford protection against improper storing. Host access-list-controlled protection and host DAT 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, DAT protection, and instruction-execution protection as defined in z/Architecture are not available in z/XC because access register translation and dynamic address translation are not provided with guest tables.

Programming note: Key-controlled protection is said to be under control of the program in the virtual machine because that program can always establish a virtual-machine state in which any particular storage access is permitted. Hence, key-controlled protection is inadequate when sharing storage among virtual machines where non-circumventable storage protection is required.

In contrast, if the host enables either host access-list-controlled protection or host DAT protection to prevent a particular storage or storage-key alteration by the virtual machine, the program in the virtual machine cannot circumvent this protection. Hence, host access-list-controlled protection and host page protection are appropriate for protecting storage that is shared among virtual machines.

Host access-list-controlled protection is particularly useful for sharing storage among virtual machines because it regulates accesses by using an attribute of an individual reference rather than using an attribute of the shared storage itself. Host access-list-controlled protection 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 because low-address protection applies only to a virtual machine's references to its own host-primary address space. Therefore, low-address protection is not applicable to cross-virtual-machine sharing of storage under z/VM.

Key-Controlled Protection

Key-controlled protection as defined in z/Architecture applies to z/XC, except for the applicability of the fetch-protection-override control as described in the following topic.

Fetch-Protection-Override Control

Fetch protection override is set when bit 38 of control register 0 is one. Fetch protection override depends on the address type.

For a type-R address

A storage reference can be made with a type-R real address or a logical address that is treated as a type-R real address. If fetch protection override is set, fetch protection is ignored for storage references to locations at effective addresses 0-2047.

For a type-A address
A storage reference can be made with a type-A real address or a logical address that is treated as a type-A real address. Fetch protection checking is always performed, 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 using a read-only or read/write access type that is 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 that is 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. An attempt to store or to explicitly alter 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 that are made by using a host access-list entry.

The access type that is permitted by a host access-list entry is established when the host access-list entry is made valid by using the ALSERV ADD host service. If the host access-list entry designates an address space that is owned by another virtual machine, a host access-list entry that permits read/write access is subject to authorization. This authorization is granted by using 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 DAT Protection

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

When the authorization that is associated with a 4 KB block of storage indicates read/write access, both fetching of and storing into the block are permitted. The storage key for the block can be explicitly altered.

When the authorization that is 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 DAT protection applies to all store accesses or storage-key alterations that are made by a virtual machine.

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

Programming note: The following 4 KB blocks of storage are established by z/VM as read-only pages:

- Blocks that correspond to shared read-only (SR) or exclusive read-only (ER) ranges of named saved systems or named saved segments that are embedded in the host-primary address space.
- Blocks that are mapped by using 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 4 KB blocks of storage that are addressable by the virtual machine are established as read/write pages. The read-only state of a block applies to all accesses, including accesses by a virtual machine with which the containing host address space is shared.

Low-Address Protection

Low-address protection operates as defined in z/Architecture, with the following exceptions.

Low-address-protection control is set in bit 35 of control register 0. Low-address protection depends on the address type.

For a type-R address
A storage reference can be made with a type-R real address or a logical address that is treated as a type-R real address. When the low-address-protection control bit is set, low-address protection applies to storage references to locations at effective addresses 0-511 and 4096-4607. Addresses 0-511 and 4096-4607 are the first 512 bytes of each of the first and second 4 KB effective-address blocks.

For a type-A address
A storage reference can be made with a type-A real address or a logical address that is treated as a type-A real address. Low-address protection does not apply, regardless of the setting of the low-address-protection control.

Suppression on Protection

During a program interruption due to a protection exception, information is stored in the translation-exception identification (TEID) and optionally in the exception-access identification (EAID). The TEID and the EAID are assigned storage locations in low storage of the host-primary address space. As in z/Architecture, the contents of these locations depend on the type of suppression-on-protection (SOP) facility that is installed. However, the validity and meaning of the contents of these fields differ from z/Architecture in the following ways:

- Although DAT is always off in z/XC, TEID bit 61 can be set to one.
- The operation is suppressed and the remainder of TEID and the EAID are meaningful when either of the following conditions is true:
  - Bit 61 is one.
  - Enhanced suppression-on-protection facility 2 (ESOP-2) is installed and bits 56, 60, and 61 are not all zeros.

  The effective address that caused the protection exception, which is stored in TEID bits 0-51, is a real or absolute address. TEID bits 62 and 63 indicate the address space by using the following values:

  00
  00 indicates that the address is in the host-primary space.

  01
  01 indicates that the address is in an AR-specified address space, as identified by the EAID.
- If the ALET that is used to reference the address that caused the exception was not obtained from an access register, then TEID bits 56, 60 and 61 are set to zeros. The remainder of the TEID and the EAID are unpredictable. (An exception ALET is not stored on a protection exception.)
- When enhanced suppression-on-protection facility 1 is installed, a zero value in TEID bit 61 means that the operation is terminated.

Unlike their counterparts in z/Architecture, host DAT protection and host access-list-controlled protection are not guaranteed to result in suppression rather than termination.

More differences from z/Architecture depend on which suppression-on-protection facility is installed.

Basic Suppression-on-Protection Facility

When the basic suppression-on-protection facility is installed, TEID bit 61 does not indicate the condition that caused the protection exception. As in the z/Architecture definition, a one value in TEID bit 61 indicates that the operation is suppressed; a zero value indicates that the operation is either suppressed or terminated. If bit 61 is one, then bit 60 indicates whether the cause is host access-list-controlled protection.

Enhanced Suppression-on-Protection Facility 1

In z/XC, ESOP-1 offers no enhancement over basic SOP. Unlike the z/Architecture definition, a zero value in TEID bit 61 indicates that the operation is terminated. TEID bit 61 does not indicate the condition that caused the protection exception.

Enhanced Suppression-on-Protection Facility 2

ESOP-2 defines TEID bits 56, 60 and 61 as a 3-bit protection code that identifies the cause of the protection exception. In z/XC, protection code 000 indicates that the operation is terminated; the cause of the exception is not indicated, and the remainder of the TEID and the EAID are not meaningful. When a protection code other than 000 is stored, the operation is suppressed. The remainder of the TEID and EAID identify the effective address and address space that caused the protection exception. For more information about the location of the protection-exception information in the TEID and EAID, see "Suppression on Protection" on page 14.

The protection codes have the following meanings:

000
The cause is not determined; the operation is terminated.

001
Host DAT protection; the operation is suppressed.

010
Key-controlled protection, the operation is suppressed.

011
Host access-list-controlled protection; the operation is suppressed.

100
Low-address protection; the operation is suppressed.

Codes 101-111 are not used.

Prefixing

Prefixing operates as defined in z/Architecture, with the following exceptions.

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 function that is provided in z/Architecture is not available in z/XC.

Therefore, the following additional DAT-related facilities that are defined in z/Architecture are also not available in z/XC:

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

Translation Control

Address translation is controlled by a bit in the PSW.

More controls are provided as described in Chapter 5, “Program Execution,” on page 27. These additional controls determine whether the contents of each access register can be used to designate an address space.

Translation Modes

PSW bit 17, called the *address-space-control* bit, determines the translation mode. The translation mode is either primary-space mode (bit 17 is zero) or access-register mode (bit 17 is one). The handling of addresses in these two modes is summarized in Table 1 on page 16.

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. The addresses are subject to implicit translation by using host access-register translation when the CPU is in the access-register mode. Analogously, the addresses that are indicated to the program as the result of executing an instruction are logical.

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

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

Except for TEST PROTECTION, addresses that explicitly designate 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 translation mode.

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

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

The handling of storage addresses that are associated with DIAGNOSE depends on the particular DIAGNOSE code. For more information, see *z/VM: CP Programming Services*.

Handling of Addresses

The handling of addresses in z/XC is summarized in Figure 1 on page 17. This figure lists all addresses that are encountered by the program and specifies the address type.

The following addresses are **instruction addresses**:

- Instruction address in PSW
- Branch address
- Target of EXECUTE and EXECUTE RELATIVE LONG
- Address that is stored in the word at real location 152 on a program interruption for PER
- Address that is placed in general register by BRANCH AND LINK, BRANCH AND SAVE, BRANCH AND SAVE AND SET MODE, BRANCH RELATIVE AND SAVE, and BRANCH RELATIVE AND SAVE LONG
- Addresses that are used by BRANCH PREDICTION RELOAD and BRANCH PREDICTION RELATIVE RELOAD
- Aborted-transaction instruction address (in the transaction diagnostic block)

The following addresses are **logical addresses**:

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

The following addresses are **real addresses that are resolved according to the translation mode**:

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

The following addresses are **host-primary real addresses**:

- Storage operand of INVALIDATE DAT TABLE ENTRY, 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)

The following addresses are **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 that is used during interruption
- Addresses that are used for machine-check logout and save areas
- Address of the STORE FACILITY LIST operand

The following addresses are **absolute addresses**:

- Failing-storage address that is stored in the doubleword at real location 248

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

The following addresses are **host-primary absolute addresses**:

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

The following addresses are **permanently assigned host-primary absolute addresses**:

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

The following addresses are **not used to reference storage**:

- PER starting address in control register 10
- PER ending address in control register 11
- Address that is stored in the word at real location 176 for a monitor event
- Address in shift instructions and other instructions that are 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 that are assigned in z/Architecture are also assigned for the same purpose in z/XC, except as specified in the following section. All assigned storage locations reside within the host-primary address space.

160
(Real Address)

Exception Access Identification: The storage contents depend on the cause of the program interruption:

ALEN-translation or addressing capability exception
If the ALET that is translated was obtained from an access register, the number of the access register is stored in bit positions 4-7 of location 160. Zeros are stored in bit positions 0-3. If the ALET that is translated was not obtained from an access register, then zeros are stored at location 160.

Protection exception
Depending on the suppression-on-exception facility that is installed and the contents of the translation-exception ID, location 160 might indicate the address space to which the exception applies. For more information, see "Assigned Storage Locations" in IBM z/Architecture Principles of Operation.

In z/XC, unlike z/Architecture, the validity of byte 160 is not affected by the following conditions:

- Validity is not affected by whether the value of PSW bit 5 is zero.
- Validity is not affected by whether the address in the TEID is real or absolute.

168-171
(Real Address)

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

168-175
(Real Address)

Translation-Exception Identification (TEID):

The only exception that presents a TEID in z/XC is a protection exception.

Translation-Exception Identification for Protection Exceptions:

The contents of this field in z/XC differ from the contents in the same field in z/Architecture as follows:

When either the basic suppression-on-protection (BSOP) or enhanced suppression-on-protection (ESOP) facility 1 is installed and bit 61 is zero, the remainder of the TEID is not meaningful. When ESOP facility 2 is installed and the protection code in bits 56, 60 and 61 is 000, the remainder of the TEID is not meaningful.

The following statements apply when the TEID is meaningful:

- Under BSOP and ESOP-1, the validity of the TEID is not affected, even though DAT is off. A value of one in bit 61 does not indicate that the effective address is a virtual address. A value of B'11' in bits 60 - 61 indicates that host access-list-controlled protection is the cause of the exception.
- Under ESOP-1, bit 61 does not indicate the cause of the protection exception. (For example, a value of zero in bit 61 does not indicate key-controlled or low-address protection.)
- Under ESOP-2, the meanings of the protection codes are as defined in “Enhanced Suppression-on-Protection Facility 2” on page 15. The effective address can be real or absolute.
- TEID bit positions 62-63 identify the address space that contains the protected address, as follows:

  00
  A type-R real address was used. The referenced address was in the host-primary space.

  01
  The 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, indicates the address space that contains the address.

256-263
(Real Address)

Failing-Storage ASIT: During a machine-check interruption, a failing-storage ASIT might 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

z/XC includes many of the z/Architecture control facilities that are described in Chapter 4 of *IBM z/Architecture Principles of Operation*. z/XC includes the following unique facilities for controlling, measuring, and recording the operation of one or more CPUs.

Program-Status Word

The z/XC program-status word (PSW) is the same as defined in z/Architecture, except as described in the following topics.

Program-Status-Word Format

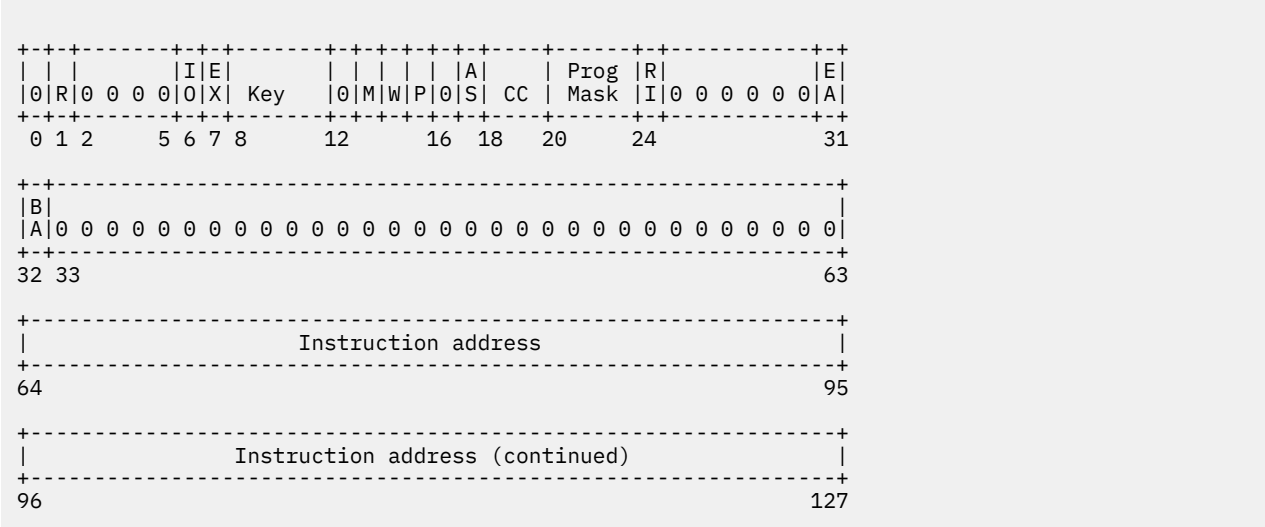


Figure 3. PSW Format

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

All PSW fields that are not required to be zeros have the same function in z/XC as in z/Architecture, except for PSW bit 17.

Address-Space Control (AS)

Bit 17 of the PSW 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 that is determined by host access-register translation.

Short PSW Format

z/XC has a short (64-bit) PSW format, like that in z/Architecture. As with the 128-bit PSW format, bits 5 and 16 of the short PSW are unassigned in z/XC. If bits 5 and 16 are not zero, then a specification exception is recognized.

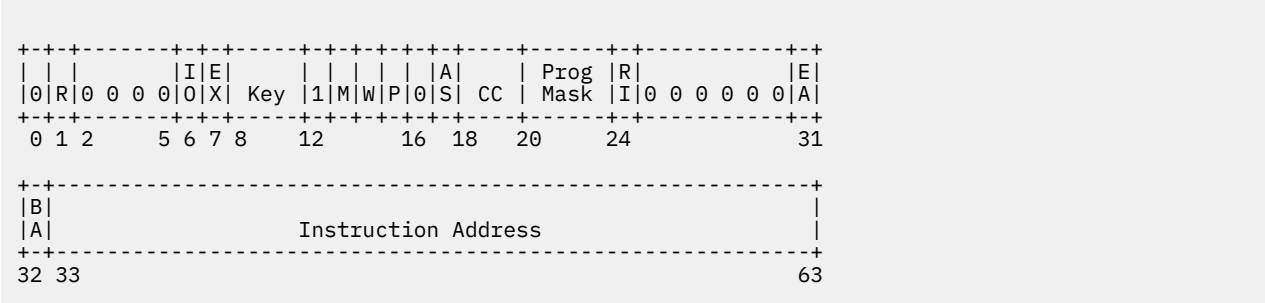


Figure 4. Short PSW Format

Control Registers

z/XC control registers and control-register fields are the same as defined in z/Architecture, with the following exceptions.

Control registers 1, 4, 5, 7, 13, and 15
All bit positions are unassigned.

Control register 0
Bits 33, 37, 40, 43, and 44 are unassigned.

Control register 3
Bits 0-31 and 48-63 are unassigned.

Control register 8
Bits 16-47 are unassigned.

Control register 12
Bit 62 is unassigned.

Control register 14
Bits 44-63 are unassigned.

All of these unassigned control-register positions are initialized to zero.

Tracing

Tracing as defined in z/Architecture applies in z/XC except that ASN tracing is not provided. The ASN-trace-control bit is not provided, and the following trace entries are never formed:

- BRANCH IN SUBSPACE GROUP
- PROGRAM CALL
- PROGRAM RETURN
- PROGRAM TRANSFER
- SET SECONDARY ASN

Program-Event Recording

The z/Architecture definition of program-event recording (both PER 2 and PER 3) applies in z/XC, with the following exceptions.

The means of restricting storage-alteration events to designated address spaces is not provided in z/XC. In z/XC, storage-alteration events are not restricted to particular address spaces.

Bit 42 of control register 9 is not ignored when DAT is off. Instead, when bit 42 of control register 9 is one, it is unpredictable whether any storage-alteration events are indicated.

The PER ASCE identification is at bits 14-15 of the PER code at real locations 150-151. The validity and value of the PER ASCE identification do not depend on PSW bit 5 having a value of one. The validity and value of the PER ASCE identification do not depend on the use of an ASCE to translate the reference that

caused the event. The PER ASCE identification is set to 00 if the reference was made to the host-primary address space, or 01 if an AR-specified reference was made. When 01 is set, the PER access ID, real location 161, can be examined to determine the address space that is referenced.

Guarded-Storage Facility

Guarded-Storage-Event Parameter-List-Address (GSEPLA) Register

Regardless of the address-space-control mode, the GSEPLA is a real address in the host-primary address space.

Guarded-Storage-Event Parameter List (GSEPL)

The address-space indication (AS) bits in the guarded-storage-event access information (GSEAI) are meaningful even though DAT is off.

Externally Initiated Functions

Externally initiated functions in z/XC are the same as defined in z/Architecture, except as described in the following topics.

Resets

The five reset functions that are provided in z/Architecture are also provided in z/XC. The z/XC reset functions performs all actions that are defined in z/Architecture. The z/XC subsystem reset function performs the following extra actions.

In z/Architecture, certain reset operations under certain conditions cause the architectural mode of the configuration to be set to the ESA/390 mode. In z/XC, these same conditions cause the architectural mode to be set instead to the ESA/XC mode.

In z/Architecture, when the configuration-z/Architecture-architectural-mode (CZAM) facility is installed, this architectural mode change is inhibited, and the configuration remains in z/Architecture mode. The analogous configuration-z/XC-architectural-mode (CZXM) facility of z/XC, if installed, inhibits the change from z/XC to ESA/XC mode.

Subsystem Reset

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

Subsystem reset operates only on those elements in the configuration that are not CPUs. In addition to the actions that are defined in z/Architecture, subsystem reset in z/XC also performs the following operations:

1. All entries in the host access list are set to the unused state.
2. Any address spaces created by 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 is 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*.

Store Status

The store-status operation in z/XC operates as in z/Architecture, including the storing of hex 01 in absolute location 163. All stores that are completed by the store-status operation are made into the host-primary address space.

CPU Signaling and Response

CPU signaling and response in z/XC is the same as defined in z/Architecture, except as described in the following topics.

Signal-Processor Orders

z/Architecture describes some SIGP orders that reset the CPU into the ESA/390 architectural mode. Those orders reset the CPU into the ESA/XC architectural mode when those orders run in z/XC.

The configuration-z/Architecture-mode (CZAM) facility in z/Architecture inhibits switching to the ESA/390 architectural mode. The analogous configuration-z/XC-mode (CZXM) facility in z/XC inhibits switching from the z/XC to the ESA/XC architectural mode.

Store Status

In z/XC, the address that is specified in the parameter register is a host-primary absolute address.

Store Additional Status at Address

In z/XC, the address that is specified in the parameter register is a host-primary absolute address.

Set Architecture

When the configuration-z/XC-mode (CZXM) facility is not installed, the set-architecture order operates as in z/Architecture, with the following exception: The order switches between the ESA/XC architectural mode and the z/XC architectural mode.

The ESA/XC architectural mode is selected by code 0 in bits 56-63 of the parameter register. The z/XC architectural mode is selected by code 1 or 2 in the parameter register.

When the CZXM facility is installed, the order is not accepted. Instead, bit 55 (invalid parameter) of the general register that is designated by the R_1 field of the SIGNAL PROCESSOR instruction is set to one, and condition code 1 is set.

Programming note: The Set Architecture order has no effect on the contents of the host access list, the existence of extra created address spaces, or the shared state of any address space that is owned by the configuration.

Facility Indications

Definitions of certain facility bits in z/XC differ from those in z/Architecture. Definitions of certain facility bits in ESA/XC differ from those in ESA/390. The following facility bits in XC architectures differ from the facility bits in the base architectures from which the XC architectures are derived.

- Facility bit 1 in ESA/XC indicates that the z/XC architectural mode is installed. This bit is stored as one in z/XC mode. In ESA/XC mode, a value of one indicates that the program can switch into z/XC mode.
- Facility bit 2 in z/XC indicates that the z/XC architectural mode is active, and therefore is stored as one. The bit is stored as zero in ESA/XC mode.
- Facility bit 8 is unpredictable and not meaningful in ESA/XC and is zero in z/XC. Enhanced-DAT facility 1 is not available in either XC architecture.
- Facility bit 138 in z/XC indicates that the configuration-z/XC-architectural-mode facility is installed. That is, the configuration is in z/XC architecture and cannot be switched to ESA/XC architecture by a program action, reset function, or IPL operation.

- Any bit that z/Architecture defines to mean that a particular facility is installed in the z/Architecture architectural mode, in z/XC and ESA/XC means that that facility is installed in the z/XC architectural mode.

  In other words, such a bit is reported the same in both XC architectures, but the facility is available only in z/XC mode. In ESA/XC mode, the bit tells the program that if the virtual machine is switched into z/XC mode, then the facility will be available.

Certain facilities that are available in z/Architecture are never provided in z/XC. The corresponding facility bits are stored as zeros. The following facilities are not provided in z/XC:

Bit	z/Architecture Facility or Function
4	Selective TLB clearing by IDTE
5	Selective TLB clearing by IDTE
6	ASN-and-LX reuse facility
8	Enhanced-DAT facility 1
27	Move-with-optional-specifications facility
36	Enhanced-monitor facility
51	Local-TLB-clearing facility
78	Enhanced-DAT facility 2
130	Instruction-execution-protection facility

Instructions that are not provided in z/XC are listed in “The z/Architecture Base” on page 2.

Chapter 5. Program Execution

Program execution for z/XC proceeds as defined in Chapter 5 of the *IBM z/Architecture Principles of Operation*, with some exceptions. Some facilities that are referenced in IBM z/Architecture Principles of Operation are not applicable to z/XC. z/XC program execution also has the following differences from z/Architecture program execution.

Authorization Mechanisms

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

- Mode requirements (DAT on)
- Secondary-space control
- Subsystem-linkage control
- ASN-translation control
- Authorization index
- Instructions and controls that are related to ANS-and-LX reuse

The following z/Architecture authorization mechanisms apply in z/XC as defined in z/Architecture except as stated in the following sections:

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

Extraction-Authority Control

In z/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 run regardless of the setting of bit 36 of control register 0.

Access-Register Mechanisms

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

The PC-number translation process and the PROGRAM CALL instruction are not provided in z/XC.

Home Address Space

The home address space and the home-space mode are not provided in z/XC.

Access-Register Introduction

Access registers are an important aspect of z/XC, and z/XC access register definitions are significantly different from z/Architecture access register definitions. z/XC access register documentation does not merely state the differences from z/Architecture, but completely replaces the documentation that is in the corresponding section of *IBM z/Architecture Principles of Operation*.

Information on z/XC access registers is also in the following topics of z/XC principles of operations:

- “Host Access-Register Translation” on page 33 and “Sequence of Storage References” on page 38 describe functions that are related to access registers.
- Chapter 3, “Storage,” on page 9 describes translation modes and host access-list-controlled protection.
- Chapter 4, “Control,” on page 21 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 39.

For reference, information on access registers is available in other publications:

- In *IBM z/Architecture Principles of Operation*:
  - Chapter 5, "Program Execution", topic “Sequence of Storage References”, describes functions that are related to access registers.
  - Chapter 11, "Machine-Check Handling", describes the handling of access registers during a machine-check interruption and the validation of the access registers.
  - Chapter 12, "Operator Facilities", describes the alter-and-display controls for access registers.
- *z/VM: CP Commands and Utilities Reference* describes alter and display facilities for access registers and access-register-specified storage.

Summary

z/XC provides the following access register functions:

- z/XC provides a maximum of 16 address spaces, including the instruction space, for immediate and simultaneous use by a semi-privileged program. The address spaces are specified by 16 special registers called access registers.
- z/XC provides instructions for examining and changing the contents of the access registers.

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 that resides 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. The program can move data between any pairs of these address spaces. Furthermore, the program can change the contents of the access registers to access still other address spaces.

The instructions for examining and changing access-register contents are unprivileged. The instructions are described in Chapter 7, "General Instructions" of *IBM z/Architecture Principles of Operation* and Chapter 7, “Instructions,” on page 47. z/XC provides the following access register instructions:

- 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 set the access-register mode, and the INSERT ADDRESS SPACE CONTROL instruction indicates the access-register mode. These instructions are described in Chapter 7, “Instructions,” on page 47.

Access-Register Functions

Access-Register-Specified Address Spaces

The CPU includes sixteen 32-bit access registers numbered 0-15. Access register mode results when PSW bit 17 is one. In the access-register mode, an instruction B or R field can be used to specify the logical address of a storage operand. Such a field 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 that is called an access-list-entry token (ALET) that determines 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 that are started in the access-register mode, ALETs can be provided in the parameter list for the host service. ALETs can also be provided 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 by using 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 that contains an operand is called *host acces-register translation* (ART). The process is depicted, for an ALET obtained from an access register, in Figure 5 on page 29.

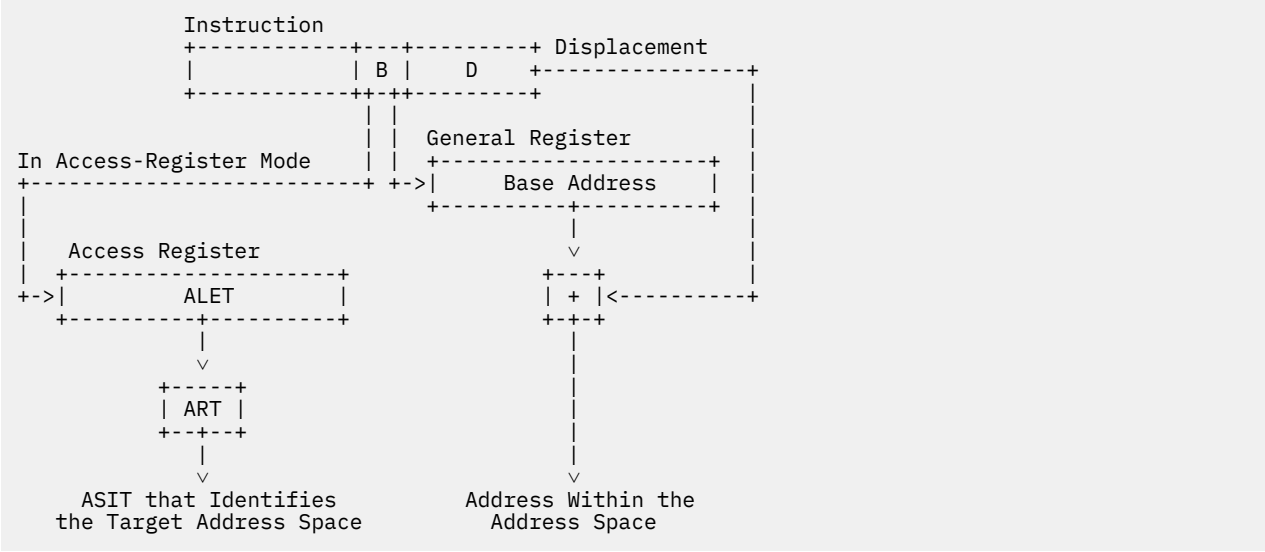


Figure 5. Use of Access Registers

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 addresses.

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 that contains 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 that contains 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 that contains the second operand is specified by access register 2, and the address space that contains the first-operand location is specified by access register 1. These two address spaces can be different address spaces, and each can be different from the instruction space (the host-primary address space).

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

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

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

Host Access-Register Translation (ART) Fundamentals

The host access-register translation process (ART) and concepts that are related to access lists are introduced. The process and concepts are explained in detail in “Host Access-Register Translation” on page 33.

Terminology note: In z/XC, *ART* is an acronym for "host access-register translation". The ART acronym is used in *IBM z/Architecture Principles of Operation* as "access-register translation" (without prefixed "host"). Unless stated otherwise herein, references to "ART" in IBM z/Architecture Principles of Operation must be interpreted as references to host access-register translation in the context of z/XC.

Determining the Target Address Space

The target address space that is specified by an ALET is determined by host access-register translation by the following process:

- If the ALET that is contained in the access register or in the parameter list that is supplied to a host service is 00000000 hex, the target address space is the host-primary address space.
- If the ALET is any value other than 00000000 hex, the target address space is identified by an address-space identification token (ASIT) that is 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. Access register 0 is treated as containing 00000000 hex, and its actual contents are not examined. Thus, a logical or real address that is specified by a zero-value B or R field in the access-register mode is always relative to the host-primary address space.

The one exception to the special treatment of access register 0 involves the TEST ACCESS instruction. The TEST ACCESS instruction uses the actual contents of access register 0.

The special treatment of register 0 allows the host-primary address space to be addressed in the access-register mode without requiring the use of a host access-list entry. Removing the requirement for a host access-list entry is useful for moving data to or from the host-primary address space.

Access Lists

Each z/XC virtual machine is provided with a separate host access list to access 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 adders spaces that are designated by the host access list by using host services.

An access list can contain 6 - 1022 entries, each of which can designate a different address space. The size of a 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) because, 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. An ALET 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 complete the following operations:

1. Save the contents of the access registers in any storage area available to the program.
2. Load and use the access registers for the program's own purposes.
3. Restore the original contents of the access registers before the program returns control to the caller.

Allocating and Invalidating Access-List Entries

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

Each entry in a host access list is considered to be in one of 3 states: *Valid*, *Revoked*, or *Unused*.

Valid
A valid host access-list entry specifies an address space and can be used in the access-register mode to reference that space.

Unused
An unused host access-list entry is available for allocation as a valid entry.

Revoked
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. For more information, see “Revoking Accessing Capability” on page 32.

The host provides services for allocating a valid host access-list entry (changing an unused entry to a valid entry) and for deallocating a valid or revoked entry (changing the status to unused).

Allocation of a host access-list entry consists of the following steps.

1. A program starts the ALSERV ADD host service and passes two parameters:
   - One parameter is the address-space identification token (ASIT). The ASIT specifies the address space for which access is requested.
   - One parameter indicates whether read-only or read/write access is requested.

   If the ASIT identifies a space that is not owned by the requesting virtual machine, the host checks that the requesting virtual machine is authorized for the requested access. Authorization is granted through use of the ADRSPACE PERMIT host service by the virtual machine that owns the address space.

   If the ASIT identifies a space that is owned by the requesting virtual machine, it is always authorized for read/write access.
2. If the virtual machine's request is authorized, the host completes the following steps:
   a. The host selects an unused entry in the virtual machine's host access list.
   b. The host changes the unused entry to a valid entry that specifies the subject address space.

c. The host establishes the entry for read-only or read/write access, as requested by the virtual machine.

d. The host returns to the program an access-list-entry token (ALET) that designates the entry.

After the ALET is 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 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, by using the ALSERV REMOVE host service, the host access-list entry that was allocated can be returned to the unused state. Returning the entry to the unused state 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 completes appropriate authorization checking during the allocation of an entry. After a valid entry is 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 is possible that a particular valid host access-list entry specifies an address space that is 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 by using the ADRSPACE ISOLATE host service.

Similarly, a particular valid host access-list entry might specify an address space (owned by the same or another virtual machine) that is 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 by using 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 when a a host access-list entry changes to the revoked state.

Preventing Store Accesses

Each host access-list entry contains an access-type indicator, which determines whether 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 complete store accesses or explicit storage-key alterations. For more information, see “Host Access-List-Controlled Protection” on page 13.

Improving Translation Performance

Host access-register translation (ART) conceptually occurs each time that 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 that is contained in the host-managed ART structures is maintained in a special buffer. The buffer is referred to as the *ART-lookaside buffer* (ALB). The information in the ART structures can be placed in the ALB and subsequent translations can be completed by using the information in the ALB. Conceptually, the ALB buffers the information that is necessary to transform an ALET into the identification of the address space that the ALET designates. The ALB is typically implemented such that it can buffer the ART structures that are 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 z/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
The instruction moves the contents of one access register to another.

EXTRACT ACCESS
The instruction moves the contents of an access register to a general register.

LOAD ACCESS MULTIPLE
The 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 that are loaded.

LOAD ADDRESS EXTENDED
The instruction is similar to the LOAD ADDRESS instruction. Both instructions load a specified general register with an effective address that is specified by using the B, X, and D fields of the instruction.

In addition, LOAD ADDRESS EXTENDED operates on the access register that has the same number as the general register that is loaded. The operation depends on the value of the address-space control, PSW bit 17:

- If the address-space control is zero, then the instruction loads the access register with 00000000 hex.
- If the address-space control is one, then the instruction loads the target access register with a value that depends on the B field of the instruction:
  - If the B field is zero, then the instruction loads the target access register with 00000000 hex.
  - If the B field is not zero, then the instruction loads the target access register with the contents of the access register that is designated by the B field. However, in the last case, when the contents of the access register that is 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 value zero specifies the primary-space mode. The address-space-control value one specifies the access-register mode.

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

SET ACCESS
The instruction replaces the contents of a specified access register with the contents of a specified general register.

STORE ACCESS MULTIPLE
The instruction function stores the contents of a set of access registers at a storage location.

Host Access-Register Translation

Host access register translation is an important aspect of z/XC. The z/XC documentation for host access register translation does not merely state the differences from z/Architecture, but completely replaces the documentation that is in the corresponding section of *IBM z/Architecture Principles of Operation*.

Host access-register translation is introduced in “Host Access-Register Translation (ART) Fundamentals” on page 30.

Terminology note: In z/XC, *ART* is an acronym for "host access-register translation". The ART acronym is used in *IBM z/Architecture Principles of Operation* as "access-register translation" (without prefixed "host"). Unless stated otherwise herein, references to "ART" in IBM z/Architecture Principles of Operation must be interpreted as references to host access-register translation in the context of z/XC.

Host Access-Register Translation Control

Host access-register translation is controlled by the address-space control bit, PSW bit 17, which is described in “Translation Modes” on page 16.

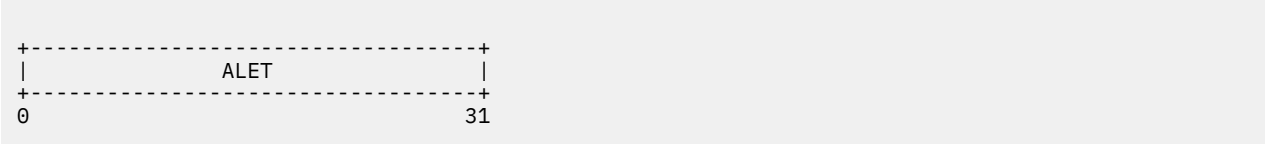
More controls are located in the host-managed access-register translation structures.

Address-Space-Function Control

ESA/XC defined an address-space function control, bit 15 of control register 0, enabling execution of certain instructions. This control is not provided in z/XC (just as it is not provided in z/Architecture). No authorization is needed to execute the SET ADDRESS SPACE CONTROL, SET ADDRESS SPACE CONTROL FAST, or TEST ACCESS instruction.

Access Registers

Sixteen 32-bit access registers are 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 can 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 an entry in the host access list that is associated with the ALET exists. If such an entry does not exist, 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 that is used 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 that is completed 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 that is translated is not 00000000 hex, host access-register translation selects an entry in the virtual machine's host access list.

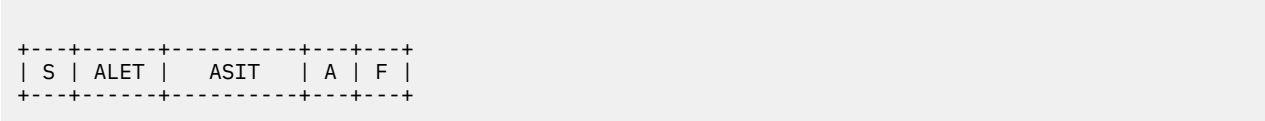
Host Access List

Each z/XC virtual machine is associated with a host access list that specifies those address spaces that are available to a virtual CPU when it is in the access-register mode. The address spaces are in addition to the host-primary address space. The host access list contains a directory-specified number of host access-list entries, each of which conceptually contains the information that is defined in “Host Access-List Entries” on page 35. The host access list is not explicitly addressable by the virtual machine. Instead, entries in the host access list are manipulated by using host services.

Programming note: The size of the host access list that is 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 are described in the following sections.

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 are allocated for use by the program and can be selected by host access-register translation. If the entry that is selected by host access-register translation is in the valid state, the translation process proceeds. If the entry that is 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, the ALET 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. The ALET 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, the ASIT field contains the address-space identification token (ASIT) that identifies the address space that is designated by the host access-list entry. The ASIT 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, the A field specifies the types of access that is permitted to the address space that is designated by the host access-list entry. Access can be read-only or read/write. If the A field indicates read/write access, both fetch and store accesses are permitted. If the A field indicates read-only access, only fetch accesses are permitted, and an attempt to store recognizes a protection exception for host access-list-controlled protection. The A 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, the F field indicates the manner in which host segment or page faults are to be handled: synchronously or asynchronously. The F field has no meaning in an entry in the revoked or unused state. Information on the options for handling host segment and page faults are included in the descriptions of the PFAULT and ALSERV services in the publication *z/VM: CP Programming Services*.

Host Access-Register Translation Process

The host access-register translation process is described for 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, complete host access-register translation as described here, except that the following exceptions set the condition code and are not treated as program-interruption conditions:

- Addressing capability
- ALET specification
- ALEN translation

Host access-register translation operates on the access register that is designated in a storage-operand reference to determine the address space that contains the storage operand. That address space is 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 that has 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 that is 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 following translation operations are completed:

1. The ALET is verified to be correctly formed.
2. The ALET is used in a lookup process to select an entry in the virtual machine's host access list.
3. The selected host access-list entry is verified to be in the valid state.
4. If a store access or an explicit storage-key alteration is attempted, the access-type indication in the host access-list entry is checked to determine whether read/write access is permitted.
5. If no exceptions are recognized, then the ASIT that is in the selected host access-list entry identifies the target address space.

Selecting the Access-List-Entry Token

When any of access registers 1-15 is designated, or for the access register that is 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 with 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 that is contained in a parameter list for a host service. The ALET that is 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*.

An operation can change the contents of an access register. The ALET that was obtained at the start of the operation is in effect until the completion of the operation in the following cases:

- LOAD ACCESS MULTIPLE changes the contents of an access register that is used by host access-register translation.
- A store access changes the contents of an ALET field in a parameter list for a host service.

Making the Host-Primary Address Space the Target Space

When the ALET that is 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 completed.

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 that is 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 that contains a selection ALET that matches the ALET that is translated. If there is an entry with a selection ALET that matches the ALET that is translated, that entry is said to be the entry that is selected by the ALET that is 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 that contains a selection ALET that matches the ALET that is 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 attempted 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 that is translated is other than 00000000 hex and the access-type specifies read/write access, the address space that is 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.

Recognition of Exceptions during Host Access-Register Translation

The exceptions that can be encountered during the host access-register translation process and their priority are shown in “Access Exceptions” on page 44.

Subspace Groups

Subspace groups are not provided in z/XC.

Linkage Stack

The linkage stack is not provided in z/XC.

ESA/390-Compatibility-Mode Facility

The presence of the ESA/390-Compatibiltiy-Mode (390-CM) facility has no effect on execution in the z/XC architectural mode. (That is, a configuration in z/XC mode is not in 390-CM.) However, the effects of this facility do apply to a configuration in the ESA/XC architectural mode. Generally, instructions, capabilities, and functions that are defined in z/XC but not in ESA/XC might operate in ESA/XC fully or partially as they do in z/XC. Such behavior is unpredictable.

The results of attempting such operations are largely as described in the “ESA/390-Compatibility-Mode Facility” section of *IBM z/Architecture Principles of Operation*. References therein to ESA/390 apply to ESA/XC and references to z/Architecture apply to z/XC. References to bit positions in control and general registers in that text use the 64-bit register number convention of z/Architecture and z/XC, rather than the 32-bit number convention of ESA/390 and ESA/XC.

Execution in ESA/XC with 390-CM differs from execution in ESA/XC without 390-CM in the following ways. (References are to the 32-bit control registers that are defined in ESA/XC.)

- Attempted execution of the BRANCH AND SET AUTHORITY instruction results in either an operation exception or a special-operation exception.
- The address-space-function control is bit 15 of control register 0. A special-operation exception might be recognized when the address-space-function control value is cleared (zero value) in the following cases:

  - The SET ADDRESS SPACE CONTROL or SET ADDRESS SPACE CONTROL FAST instruction is executed to enter access-register mode.
  - The TEST ACCESS instruction is executed.
- Branch tracing, which is governed by bit 0 of control register 12, cannot be enabled.
- The effect is unpredictable of any reserved bits of a control register whose corresponding bit positions in z/XC are assigned:
  - It is unpredictable whether bit 0 of control registers 10 and 11 (corresponding to bit 32 in z/XC) is used in forming the PER address range.
  - For explicit tracing, it is unpredictable whether bit 0 of control register 12 (corresponding to bit 32 in z/XC) is used in forming the trace-entry address.
- The effect of running an instruction that is unique to the z/Architecture or z/XC architectural mode is unpredictable. The instruction might run according to its z/Architecture or z/XC definition or yield an operation exception.
- If the program issues an instruction that is defined to enable the 64-bit addressing mode, the possible results are as described in the "ESA/390-Compatibility-Mode Facility" section of *IBM z/Architecture Principles of Operation*.
- The result of an instruction that is valid in both the ESA/XC and z/XC architectural modes, but specifies an attribute that is specific to the z/XC architectural mode, is unpredictable. The attribute might be ignored, the instruction might run according to its z/XC definition, or an exception might be recognized. More information is available in the "ESA/390-Compatibility-Mode Facility" section of *IBM z/Architecture Principles of Operation*.
- It is unpredictable whether PER instruction-fetching nullification, zero-address-detection, and storage-key-alteration events are recognized.
- SIGP order codes that are unique to the z/Architecture and z/XC architectural modes result in an invalid-order status.

Sequence of Storage References

The effects that can be observed in storage due to overlapped operations and piecemeal execution of a CPU program in z/Architecture are defined in *IBM z/Architecture Principles of Operation*. The effects apply to z/XC as well, with the following exceptions:

- References to virtual storage or virtual addresses in z/Architecture apply instead to real storage, real addresses (type-R and type-A) and absolute addresses in z/XC.
- References to the following storage elements do not apply to z/XC:
  - Real mode
  - Secondary-space mode
  - Home-space mode
  - ART-table and DAT-table fetches
  - ALB entries

Chapter 6. Interruptions

The interruption mechanism for z/XC is similar to the definition for z/Architecture. Except as described in the following topics, the definition for interruptions that is provided in Chapter 6 of *IBM z/Architecture Principles of Operation* applies to z/XC as well.

Interruption Action

The interruption action in z/XC is the same as defined for z/Architecture except as described in the following topics.

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

Exceptions Associated with the PSW

In addition to the error conditions that are defined in z/Architecture, z/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 z/Architecture.

Early Exception Recognition

In z/XC, a program interruption for a specification exception occurs for the error conditions that are defined in z/Architecture. In z/XC, a program interruption for a specification exception also 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 z/Architecture for other PSW-format errors that are recognized early.

Program Interruption

Program interruptions follow the definition in z/Architecture except as described in the following topics.

Program-Interruption Conditions

Program-interruption conditions that are either recognized only in z/XC or are defined differently in z/XC than in z/Architecture are listed.

The following program-interruption conditions that are defined in z/Architecture are never recognized in z/XC:

- AFX-translation exception
- ALE-sequence exception
- ASCE-type-specification exception
- ASTE-instance exception
- ASTE-sequence exception
- ASTE-validity exception
- ASX-translation exception
- EX-translation exception
- Extended-authority exception
- LFX-translation exception
- LSTE-sequence exception

- LSX-translation exception
- LX-translation exception
- Page-translation exception
- PC-translation-specification exception
- Primary-authority exception
- Region-first-translation exception
- Region-second-translation exception
- Region-third-translation 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
- Translation-specification exception

Any other z/Architecture program-interruption conditions that are not described in one of the following sections is defined in z/XC in the same way that it is defined for z/Architecture. However, the z/Architecture definition might include causes that are not applicable to z/XC because the related facilities are not provided.

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

Addressing Exception

In addition to the causes that are defined in z/Architecture, an addressing exception is recognized if all of the following conditions are true:

- A reference is made to a storage location in a shared address space that is the primary space of a z/Architecture guest.
- The location is within a 4 KB block in the unused block-usage state and the logically-zero block-content state, as defined for collaborative memory management.

For information about the collaborative memory management assist, see *z/VM: CP Programming Services*

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 that is 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 ended.

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

An addressing-capability exception is indicated by any of four program-interruption codes. The code indicates whether other events occurred concurrently:

Code	Concurrent PER event	Concurrent transactional-execution-aborted event
0136	No	No
01B6	Yes	No
0336	No	Yes
03B6	Yes	Yes

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

ALEN-Translation Exception

An ALEN-translation exception is recognized during host access-register translation when the access-list-entry token 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 that is 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.

An ALEN-translation exception is indicated by any of four program-interruption codes. The code indicates whether other events occurred concurrently:

Code	Concurrent PER event	Concurrent transactional-execution-aborted event
0029	No	No
00A9	Yes	No
0229	No	Yes
02A9	Yes	Yes

Programming note: An ALEN-translation exception indicates that the program attempted to use a correctly formed but currently unassigned ALET. It is possible that the ALET was associated with a valid host access-list entry that was later deallocated.

ALET-Specification Exception

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

The operation is suppressed.

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

An ALET-specification exception is indicated by any of four program-interruption codes. The code indicates whether other events occurred concurrently:

Code	Concurrent PER event	Concurrent transactional-execution-aborted event
0028	No	No
00A8	Yes	No

Code	Concurrent PER event	Concurrent transactional-execution-aborted event
0228	No	Yes
02A8	Yes	Yes

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 is never assigned by the host as the selection ALET for a valid or revoked host access-list entry. (Contrast an ALEN-translation-exception condition, which indicates that the program attempted to use an ALET that can be assigned to a host access-list entry, but currently is not.) Often, this exception indicates that the program loaded an access register from a storage location that does not currently contain an ALET value that is provided by the host.

Block-Volatility Exception

A block-volatility exception is recognized when all of the following conditions are true:

- A reference is made to a storage location in a shared address space that is the primary space of a z/Architectureguest.
- The location is within a 4 KB block that is in the volatile block-usage state and the logically-zero block-content state, as defined for collaborative memory management.

For information about the collaborative memory management assist, see *z/VM: CP Programming Services*.

The unit of operation is nullified.

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

A block-volatility exception is indicated by any of four program-interruption codes. The code indicates whether other events occurred concurrently:

Code	Concurrent PER event	Concurrent transactional-execution-aborted event
001A	No	No
009A	Yes	No
021A	No	Yes
029A	Yes	Yes

Privileged-Operation Exception

The definition for the privileged-operation exception is the same as for z/Architecture, with the following exceptions:

- No privileged-operation exception is recognized if the INSERT ADDRESS SPACE CONTROL instruction is executed when the extraction-authority control, bit 36 of control register 0, is zero.
- No privileged-operation exception is recognized if bits 52-55 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 z/Architecture, with the following exceptions.

In z/XC, a protection exception is not recognized due to access-list-controlled protection, DAT protection, or instruction-execution protection as defined in z/Architecture. Instead, a protection exception is recognized for host access-list-controlled protection or host DAT protection when either of the following conditions are true:

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

Exceptions that are due to key-controlled protection and low-address protection are recognized as in z/Architecture.

The operation is suppressed or terminated as described in "Suppression on Protection" on page 14.

Programming note: In z/VM, host DAT protection is applied to the following address ranges:

- Address ranges within an IPLed named saved system (NSS) or a loaded discontiguous saved segment (DCSS) that were defined as read-only, except for ranges in a DCSS that was loaded nonshared. Read-only storage ranges are type ER, SR, or SC.
- Ranges of an address space into which blocks of a read-only minidisk were mapped by the MAPMDISK service.

Special-Operation Exception

In addition to the causes that are defined in z/Architecture, a special-operation exception can be recognized in lieu of an operation exception. The special-operation exception is recognized when certain instructions that are provided in z/Architecture but not in z/XC are attempted. See "z/Architecture Instructions Not Provided" on page 47.

The operation is suppressed.

The instruction-length code is 2 or 3.

A special-operation exception is indicated by any of four program-interruption codes. The code indicates whether other events occurred concurrently:

Code	Concurrent PER event	Concurrent transactional-execution-aborted event
0013	No	No
0093	Yes	No
0213	No	Yes
0293	Yes	Yes

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 z/Architecture, except that SET SYSTEM MASK does not give a specification exception, even though bit 33 of control register 0 is one. (In z/Architecture, a specification exception is recognized when bit 33 of control register 0 is one.) Also, a specification exception is recognized in z/XC when either of the following conditions occur:

1. A 1 is introduced into bit position 5 or 16 of the PSW. This situation is handled as an early PSW specification exception.
2. Bits 54-55 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 that is identified by the old PSW is suppressed. However, for early PSW specification exceptions (causes 1-3 in *IBM z/Architecture Principles of Operation*, and situation "1" on page 43), 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 z/Architecture Principles of Operation*, and situation “1” on page 43), and the exception was introduced by LOAD PSW or an interruption, the ILC is zero. When the exception is introduced by SET SYSTEM MASK or by STORE THEN OR SYSTEM MASK, the ILC is 2 or 3.

Multiple Program-Interruption Conditions

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

The priority of all program-interruption conditions other than PER events and exceptions that are associated with some of the more complex control instructions is the same as defined in Figure 6-8 of *IBM z/Architecture Principles of Operation*. In that figure, all exceptions that are 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 6 on page 45 lists the priority of access exceptions for a single access. Thus, Figure 6-8 of *IBM z/Architecture Principles of Operation* specifies the priority of an access-exception condition in relation to other conditions that are detected in the operation. Figure 6 on page 45 specifies which of several access exceptions, encountered either in the access of a particular portion of an instruction or in any particular access that is associated with an operand, has highest priority.

The priority for exceptions that occur as part of tracing is the same as defined in Figure 6-7 of *IBM z/Architecture 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 when an absolute, instruction, logical, or real address is used to access storage. Thus, in the access-register mode, the following exceptions can occur:

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

When in the primary-space mode, only addressing, key-controlled protection, host DAT protection, and low-address protection exceptions can be encountered.

The access exceptions are listed in detail in Figure 6 on page 45.

A.
Protection exception (low-address protection) due to a store-type operand reference with an effective address in the range 0-511 or 4096-4607

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

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

B.1.B
Addressing-capability exception due to an ALET that designates 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 that uses 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 DAT 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 6. Priority of Access Exceptions

ASN-Translation Exceptions

None of the z/Architecture ASN translation exceptions apply to z/XC.

Subspace-Replacement Exceptions

None of the z/Architecture subspace-replacement exceptions apply to z/XC.

Chapter 7. Instructions

The operation of z/XC instructions, other than the instructions for input/output, is described and compared with z/Architecture.

Except as specified in the following topics, z/XC provides all general, decimal, floating point, and control instructions that are provided by z/Architecture. The instructions operate as described in Chapters 7 - 10, 18 and 19 of *IBM z/Architecture Principles of Operation*.

z/Architecture Instructions Not Provided

The following instructions are provided in z/Architecture but are not provided in z/XC. An operation exception is recognized on an attempt to execute any of these instructions.

- 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 SECONDARY ASN
- START INTERPRETIVE EXECUTION

The following instructions might be provided in z/Architecture but are not provided in z/XC. Attempted execution of any of these instructions results in either an operation exception or a special-operation exception; which exception occurs is unpredictable.

- COMPARE AND REPLACE DAT TABLE ENTRY
- EXTRACT AND SET EXTENDED AUTHORITY
- EXTRACT AND SET STORAGE ATTRIBUTE (defined in *z/VM: CP Programming Services*)
- EXTRACT PRIMARY ASN AND INSTANCE
- EXTRACT SECONDARY ASN AND INSTANCE
- LOAD PAGE TABLE ENTRY ADDRESS
- MOVE WITH OPTIONAL SPECIFICATIONS
- PERFORM FRAME MANAGEMENT FUNCTION
- PROGRAM TRANSFER WITH INSTANCE
- SET SECONDARY ASN WITH INSTANCE
- STORE REAL ADDRESS

- TRAP

Modified z/Architecture Instructions

The following topics describe the operation of instructions that operate differently in z/XC than in z/Architecture.

DIAGNOSE

The DIAGNOSE instruction is used in a virtual-machine environment to request services from the host. For more information, see *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 54 of the general register that is designated by the R_1 field. Bits 48-53 and 55 of the register are set to zeros, and bits 0-47 and 56-63 of the register remain unchanged. The address-space-control bit is also used to set the condition code.

Resulting Condition Code

0
PSW bit is 17 zero, which indicates primary-space mode.

1
--

2
PSW bit 17 is one, which indicates access-register mode.

3
--

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

INSERT STORAGE KEY EXTENDED

This instruction operates as defined for z/Architecture, with the following addition:

When the CPU is in the primary-space mode, the address of the 4 KB block that is 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 4 KB block that is designated by the contents of general register R_2 is an AR-specified real address. The address is interpreted to be within the address space that is specified by the access register that is designated by the R_2 field.

INVALIDATE DAT TABLE ENTRY

This instruction operates as defined for z/Architecture, with the following additions.

For the invalidation-and-clearing operation, a table-entry address is formed by using a table origin address and an index. The table origin address is in the register that is designated by the R_3 field and the index is in the register that is designated by the R_2 field. In z/XC, that table-entry address is treated as a host-primary real or absolute address.

Programming note: In z/XC, (guest) DAT is not provided and (guest) TLB entries are never formed. Therefore, the clearing-by-ASCE operation is executed as a no-operation. The invalidation-and-clearing operation provides little utility; the setting of a bit in storage can be accomplished more efficiently by using the OR instruction. On some models, the INVALIDATE DAT TABLE ENTRY instruction requires a significant amount of time and might cause a performance degradation.

INVALIDATE PAGE TABLE ENTRY

This instruction operates as defined for z/Architecture, with the following additions.

A real address is formed by using a page-table origin address and a page index. The page-table origin address is in the register that is designated by the R_1 field and the page index is in the register that is designated by the R_2 field. In z/XC, that real address is treated as a host-primary real address.

Programming note: In z/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 by using the OR instruction. On some models, the INVALIDATE PAGE TABLE ENTRY instruction requires a significant amount of time, and might cause a performance degradation.

LOAD ADDRESS EXTENDED

The operation is executed as defined for z/Architecture, except that the value that is placed in access register R_1 is as shown in the following table:

PSW Bit 17	Value that is Placed in Access Register R_1
0	00000000 hex (zeros in bits positions 0-31) is placed in access register R_1.
1	If B_2 field is zero: 00000000 hex (zeros in bit positions 0-31) is placed in access register R_1. If B_2 field is nonzero: The contents of access register B2 is placed in access register R_1.

LOAD PSW

This instruction operates as defined for z/Architecture, with the following special conditions.

Special conditions: The PSW fields that are loaded by the instruction are not checked for validity before they are loaded, except for the optional checking of bit 12. 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 0. (These reasons for a specification exception are in addition to those reasons that are described in *IBM z/Architecture Principles of Operation*.)

LOAD PSW EXTENDED

This instruction operates as defined for z/Architecture, with the following modifications:

Special conditions: The value that is 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 or 16. In these cases, the instruction is completed, and the instruction-length code is 0. (These reasons for a specification exception are in addition to those reasons that are described in *IBM z/Architecture Principles of Operation*.)

LOAD USING REAL ADDRESS

This instruction operates as defined for z/Architecture, with the following addition.

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

MONITOR CALL

Because the z/Architecture enhanced-monitor facility is not provided in z/XC, monitor-event counting is not attempted. If bits 16-31 of control register 8 are not set to zeros, then results are unpredictable.

PURGE ALB

The instruction is executed as a no-operation.

Programming note: In z/XC, (guest) ART is not provided and (guest) ALB entries are never used. This instruction completes no useful function in the z/XC mode. On some models, this instruction requires a significant amount of time, and might cause a performance degradation.

PURGE TLB

The instruction is executed as a no-operation.

Programming note: In z/XC, (guest) ART is not provided and (guest) TLB entries are never used. This instruction completes no useful function in the z/XC mode. On some models, this instruction requires a significant amount of time, and might cause a performance degradation.

RESET REFERENCE BIT EXTENDED

This instruction operates as defined for z/Architecture, with the following additions.

When the CPU is in the primary-space mode, the address of the 4 KB block that is 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 4 KB block that is designated by the contents of general register R_2 is an AR-specified real address. The address is interpreted as within the address space that is specified by the access register that is designated by the R_2 field.

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

RESUME PROGRAM

In z/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 access exceptions for the second operand.

SET ADDRESS SPACE CONTROL and SET ADDRESS SPACE CONTROL FAST

Bits 52-55 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 52-55 form the code. Bits 0-51 and 56-63 of the second-operand address are ignored. Bits 20, 53, and 55 of the second-operand address must be zeros; otherwise, a specification exception is recognized.

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	

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

Special conditions: The priority of recognition of program exceptions for the instruction is shown in Figure 7 on page 51.

Condition Code

The code remains unchanged.

Program Exceptions

- Special operation
- Specification

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
Transaction constraint

9.
Specification because of a nonzero value in bit positions 52, 53, or 55 of the second-operand address

Figure 7. Priority of Execution: SET ADDRESS SPACE CONTROL and SET ADDRESS SPACE CONTROL FAST

Programming note: Programming notes for the SET ADDRESS SPACE CONTROL and SET ADDRESS SPACE CONTROL FAST instructions are included in *IBM z/Architecture Principles of Operation*.

SET STORAGE KEY EXTENDED

This instruction operates as defined for z/Architecture, with the following additions.

When the CPU is in the primary-space mode, the address of the 4 KB block that is 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 4 KB block that is designated by the contents of general register R_2 is an AR-specified real address. The address is interpreted as within the address space that is specified by the access register that is designated by the R_2 field.

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

Because the z/Architecture enhanced-DAT facility 1 is not provided in z/XC, the multiple-block control (bit 3 of the M_3 field) is not provided; this bit must be set to zero.

SET SYSTEM MASK

This instruction operates as defined for z/Architecture, with the following modifications.

The SSM-suppression control (bit 33 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 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 or 3. (This reason for a specification exception is in addition to those reasons that are described in *IBM z/Architecture Principles of Operation*.)

STORE THEN OR SYSTEM MASK

This instruction operates as defined for z/Architecture, with the following special conditions.

Special conditions: The value that is 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 or 3. (This reason for a specification exception is in addition to those reasons that are described in *IBM z/Architecture Principles of Operation*.)

STORE USING REAL ADDRESS

This instruction operates as defined for z/Architecture, with the following addition.

The contents of the general register that is 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 that are recognized during host access-register translation (ART). The ALET is also tested for 00000000 hex.

The contents of bits 32-47 of general register R_2 are expected to be 0001 hex.

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

Bits 0-31 and 48-63 of general register R_2 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 condition exists that would normally cause one of the exceptions that are shown in the following table, the instruction is completed by setting condition code 3:

Table 2. Conditions that would normally cause a program exception but instead complete with condition code 3

Exception Name	Cause
Addressing capability	ALET is correctly formed and selects a host access-list entry that is in the revoked state.
ALET specification	ALET is not correctly formed.
ALEN translation	ALET is correctly formed but does not select a host access-list entry that is in either the valid or revoked state.

When ART is completed without any of the exceptions that are in Table 2 on page 52, the instruction is completed by setting condition code 2. ART is described in the section "Host Access-Register Translation" on page 33.

If the instruction is executed and the contents of bits 32-47 of general register R_2 are not 0001 hex, the resulting condition code is unpredictable.

The priority of recognition of program exceptions for the instruction is shown in Figure 8 on page 53.

Resulting Condition Code

When the contents of bits 32-47 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 32-47 of general register R_2 are not 0001 hex, the resulting condition code is unpredictable.

Program Exceptions

There are no program exceptions.

The priority of recognition of program exceptions and setting of condition codes for the instruction is shown in Figure 8 on page 53.

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
Transaction constraint

8.
Condition code 0 because the access-list-entry-token (ALET) value is 00000000 hex

9.
Condition code 3 because of an incorrectly formed ALET

10.
Condition code 3 because the ALET did not select a host access-list entry that is in the valid or revoked state

11.
Condition code 3 because the ALET selected a host access-list entry that is in the revoked state

12.
Condition code 2

Figure 8. Priority of Execution: TEST ACCESS

TEST BLOCK

This instruction operates as defined for z/Architecture, with the following additions.

When the CPU is in the primary-space mode, the address of the 4 KB block that is 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 4 KB block that is designated by the contents of general register R_2 is an AR-specified real address. The address is interpreted as within the address space that is specified by the access register that is designated by the R_2 field.

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

TEST PROTECTION

The location that is designated by the first-operand address is tested for protection exceptions by using the access key that is specified in bits 56-59 of the second-operand address.

The second-operand address is not used to address data. Instead, bits 56-59 of the address form the access key that is used in testing. Bits 0-55 and 60-63 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 using the host access-register

translation process (ART). ART applies to the access register that is designated by the B_1 field to determine the address space that contains the first operand.

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

Table 3. Conditions that would normally cause a program exception but instead complete with condition code 3

Exception Name	Cause
Addressing capability	ALET is correctly formed and selects a host access-list entry that is in the revoked state.
ALET specification	ALET is not correctly formed.
ALEN translation	ALET is correctly formed but does not select a host access-list entry that is 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 attempt translation by using 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.

The storage key for the block that is designated by the first-operand address is tested against the access key that is specified in bits 56-59 of the second-operand address when any of the following conditions is true:

- Translation of the first-operand address can be completed.
- The CPU is in the primary-space mode.

The condition code is set to indicate whether store and fetch accesses are permitted. All applicable protection mechanisms are considered.

For example, if low-address protection is active and the first-operand effective address is in the range 0-511 or 4096-4607, then a store access is not permitted. Host DAT protection, host access-list-controlled protection, storage-protection override, and fetch-protection override are also considered.

The contents of storage, including the change bit, are not affected. Depending on the model, the reference bit for the first-operand address might 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 z/Architecture, with the following addition.

The contents of bits 2-61 of control register 12, with two zero bits appended on the right, are treated as a host-primary real address.

Chapter 8. Machine-Check Handling

Machine checks in z/XC are handled similarly to machine checks in z/Architecture. Except as described in the following topics, the definition for machine-check handling that is provided in Chapter 11 of *IBM z/Architecture Principles of Operation* applies to z/XC.

Handling of Machine Checks

The overall handling of machine checks in z/XC is the same as defined for z/Architecture, except that z/XC provides automatic validation of certain register entities.

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 that were placed in the corresponding save areas. The contents of the registers are restored even if the associated machine-check-interruption-code validity bit for the logged-out copy is zero.

Machine-Check Extended Interruption Information

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

Failing-Storage Address and ASIT

z/Architecture defines the results of certain machine-check codes that indicate storage and storage-key errors. The associated address, called the failing-storage address, is stored in real locations 248-255 when a machine-check code indicates any of the following conditions:

- Storage error uncorrected
- Storage error that is corrected
- Storage-key error uncorrected

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.

Programming note: In ESA/XC, the failing-storage address is a 31-bit address in locations 248-251. In z/XC, the failing-storage address is a 64-bit address in locations 248-255. A program that was written for ESA/XC must be adjusted to account for this difference if it is to run in z/XC.

Chapter 9. Input/Output

Input/output processing for z/XC is similar to processing that is defined for z/Architecture. The definitions for I/O instructions, I/O functions, I/O interruptions and I/O support functions that are provided in Chapters 13 - 17 of *IBM z/Architecture Principles of Operation* apply to z/XC, with the exceptions that are described in the following topics.

Handling of Addresses for I/O

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

The following addresses are host-primary absolute addresses:

- Address limit, which is specified in SET ADDRESS LIMIT
- Measurement-block origin, which is specified in SET CHANNEL MONITOR
- Channel-program address in an operation request block
- Data address in a channel-command word (CCW)
- CCW address in a CCW that specifies a transfer in channel
- Indirect-data-address word (IDAW) address in a CCW that specifies indirect data addressing
- Modified indirect-data-address word (MIDAW) address in a CCW that specifies modified indirect data addressing
- Data address in an IDAW
- Data address in an MIDAW
- CCW address in a subchannel status word
- Failing-storage address in a format-0 extended-status word
- Addresses of PSW and first two CCWs that are 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

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., in the United States and/or other countries. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on IBM Copyright and trademark information (https://www.ibm.com/legal/copytrade).

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 world-wide basis.

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:

- The section entitled **IBM Websites** at IBM Privacy Statement (https://www.ibm.com/privacy)
- Cookies and Similar Technologies (https://www.ibm.com/privacy#Cookies_and_Similar_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 Documentation - z/VM (https://www.ibm.com/docs/en/zvm).

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

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: 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: z/Architecture Extended Configuration (z/XC) Principles of Operation*, SC27-4940

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: VM Dump Tool*, GC24-6335

z/VM Facilities and Features

Data Facility Storage Management Subsystem for z/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/Support Facility on the Hardware Management Console (https://www.ibm.com/docs/en/SSLTBW_2.3.0/pdf/SC14-7580-02.pdf), SC14-7580
- Open Systems Adapter-Express ICC 3215 Support (https://www.ibm.com/docs/en/zos/2.3.0?topic=osa-icc-3215-support), SA23-2247
- Open Systems Adapter Integrated Console Controller User's Guide (https://www.ibm.com/docs/en/SSLTBW_2.3.0/pdf/SC27-9003-02.pdf), SC27-9003
- Open Systems Adapter-Express Customer's Guide and Reference (https://www.ibm.com/docs/en/SSLTBW_2.3.0/pdf/ioa2z1f0.pdf), SA22-7935

Performance Toolkit for z/VM

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

The following publications contain sections that provide information about z/VM Performance Data Pump, which is licensed with Performance Toolkit for z/VM.

- *z/VM: Performance*, SC24-6301. See z/VM Performance Data Pump.
- *z/VM: Other Components Messages and Codes*, GC24-6300. See Data Pump Messages.

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

TCP/IP for z/VM

- *z/VM: TCP/IP Diagnosis Guide*, GC24-6328
- *z/VM: TCP/IP LDAP Administration Guide*, SC24-6329
- *z/VM: TCP/IP Messages and Codes*, GC24-6330
- *z/VM: TCP/IP Planning and Customization*, SC24-6331
- *z/VM: TCP/IP Programmer's Reference*, SC24-6332
- *z/VM: TCP/IP User's Guide*, SC24-6333

Prerequisite Products

Device Support Facilities

- Device Support Facilities (ICKDSF): User's Guide and Reference (https://www.ibm.com/docs/en/SSLTBW_2.5.0/pdf/ickug00_v2r5.pdf), GC35-0033

Environmental Record Editing and Printing Program

- Environmental Record Editing and Printing Program (EREP): Reference (https://www.ibm.com/docs/en/SSLTBW_2.5.0/pdf/ifc2000_v2r5.pdf), GC35-0152
- Environmental Record Editing and Printing Program (EREP): User's Guide (https://www.ibm.com/docs/en/SSLTBW_2.5.0/pdf/ifc1000_v2r5.pdf), GC35-0151

Related Products

XL C++ for z/VM

- *XL C/C++ for z/VM: Runtime Library Reference*, SC09-7624
- *XL C/C++ for z/VM: User's Guide*, SC09-7625

z/OS

IBM Documentation - z/OS (https://www.ibm.com/docs/en/zos)

Other Publications

IBM Enterprise Systems Architecture/390 Principles of Operation, SA22-7201
IBM z/Architecture Principles of Operation, SA22-7832

Index

D

E

F

G

H

I

K

L

M

P

R

S

T

U

V

Z

Product Number: 5741-A09

Printed in USA

SC27-4940-73