z/VSE 5.1: 64 bit virtual

Ingolf Salm
salm@de.ibm.com
Ingolf's z/VSE Blog: https://www.ibm.com/developerworks/mydeveloperworks/blogs/vse

Trademarks

The following are trademarks of the International Business Machines Corporation in the United States and/or other countries.

IBM*
IBM Logo*

* Registered trademarks of IBM Corporation

The following are trademarks or registered trademarks of other companies.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.
Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.
Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.
Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.
INFINIBAND, InfiniBand Trade Association and the INFINIBAND design marks are trademarks and/or service marks of the INFINIBAND Trade Association.
Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.
UNIX is a registered trademark of The Open Group in the United States and other countries.
Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.
IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency, which is now part of the Office of Government Commerce.

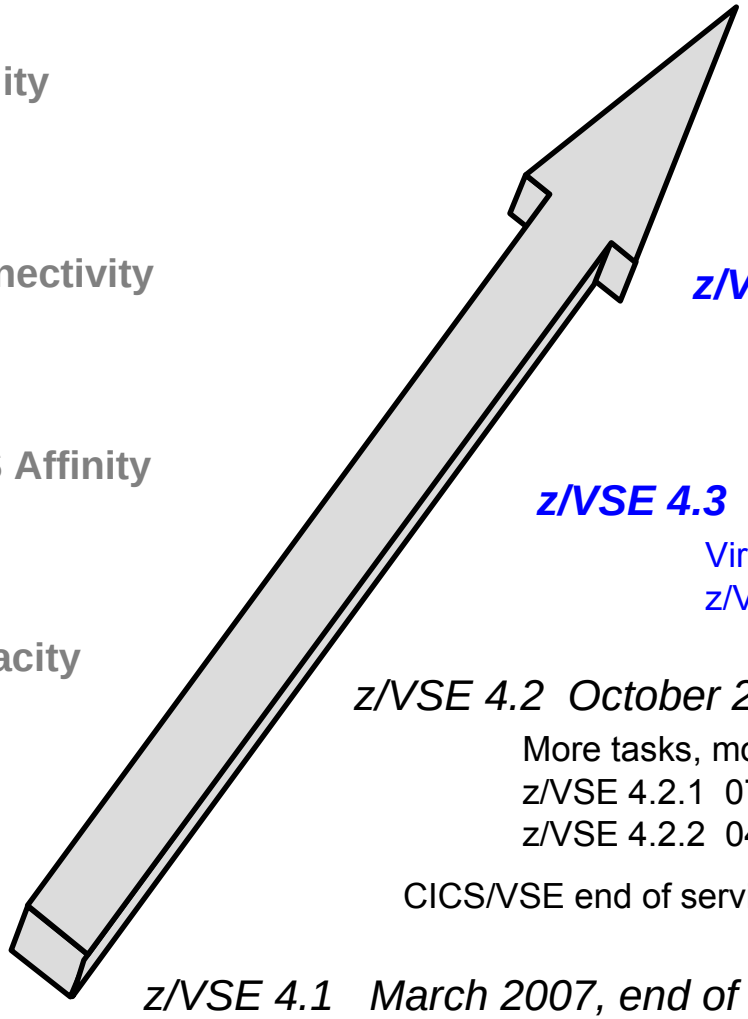
* All other products may be trademarks or registered trademarks of their respective companies.

Notes:
Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.
IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.
All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.
This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.
All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.
Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the 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.
Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.

Agenda

- VSE Roadmap and z/VSE 5.1
- z/Architecture
- z/VSE 4.3 – 64 bit real addressing
- z/VSE 5.1 – 64 bit virtual addressing
  - Memory objects
  - IARV64 services
  - Addressing modes
  - Considerations
- z/VSE 5.1 enhancements

VSE Roadmap



z/VSE Statement of Direction (SOD)

Install from DVD
CICS Explorer update

z/VSE 5.1.1 (+ Enhancements) GA 06/14/2013

TS1140, 64 bit I/O, openSSL and
database connector enhancements

z/VSE 5.1 GA 11/25/2011

64 bit virtual, zEnterprise exploitation, z9 or higher
z/VSE 5.1.1 06/2012: CICS Explorer, LFP in LPAR,
database connector

z/VSE 4.3 11/2010

Virtual storage constraint relief, 4 digit cuus
z/VSE 4.3.1 08/2011, *end of support 05/31/2014*

z/VSE 4.2 October 2008, end of service 10/31/2012

More tasks, more memory, EF for z/VSE 1.1, CPU balancing, SCRT on z/VSE
z/VSE 4.2.1 07/2009 - PAV, EF for z/VSE 1.2
z/VSE 4.2.2 04/2010 - IPv6/VSE 05/2010

CICS/VSE end of service 10/31/2012

z/VSE 4.1 March 2007, end of service 04/30/2011

z/Architecture only, 64 bit real addressing, MWLC – full and sub-capacity pricing

z/VSE V5.1

- z/VSE 5.1: Preview 04/12/2011, Announcement 10/12/2011, GA 11/25/2011
- z/VSE 5.1.1: GA 06/15/2012

- **64 bit virtual**

- Introduces Architectural Level Set (ALS) that requires System z9 or later

- zEnterprise exploitation (z114 and z196)

- Exploitation of IBM System Storage options

- Networking enhancements
  - IPv6 support to be added to Fast Path to Linux on System z function

- IPv6/VSE
  - Large TCP window support, can increase throughput
  - 64 bit virtual exploitation, large TCP window storage allocated above the bar

- Fast Service Upgrade (FSU) from z/VSE 4.2 and z/VSE 4.3

- CICS SOD:
  - IBM intends to provide CICS Explorer capabilities for CICS TS for VSE/ESA, to deliver additional value.

z/VSE 5.1 Additional Enhancements 2012

- IBM z/VSE V5.1 - Additional enhancements: Announced 04/03/2012, GA 06/15/2012
- CICS Explorer for z/VSE
- Linux Fast Path in LPAR
- Linux Fast Path via z/VSE z/VM IP Assist (z/VSE VIA)
- z/VSE database connector
- VSE/POWER enhancement to ease job output handling
- IBM System Storage Tape Controller 3592 Model C07
- New symbolic parameter IJBVMID containing the z/VM userid if running on z/VM)
- PTFs: GA 4Q/2012
  - 64-bit input/output (I/O) processing for applications
  - IPv6/VSE V1.1 enhancements
    - Secure Sockets Layer (SSL) for secure data transmission
    - Layer 2 support for OSA Express devices for IPv4 links

z/VSE 5.1 Announcement (04/02/2013)

- IBM z/VSE V5.1 - Additional enhancements - GA planned for June 14, 2013

  – Announcement content:

  - Support of zEC12
    - o Configurable Crypto Express4S
    - o OSA Express4S 100BASE-T
  - Support of IBM System Storage
    - o IBM System Storage TS1140 (3592 E07)
    - o IBM System Storage TS7700 Virtualization Engine Release 3.0
    - o IBM System Storage DS8870
    - o IBM System Storage Storwize V7000 Release 6.4
  - 64-bit input/output (I/O) processing for applications
  - HiperSockets configurable input buffers

z/VSE 5.1 Announcement (04/02/2013) …

– Announcement content ….

- System dump support for memory objects
- z/VSE Database connector enhancements

- OpenSSL update
- IPv6/VSE V1.1 enhancements
  - Secure Sockets Layer (SSL) for secure data transmission
  - Layer 2 support for OSA Express devices for IPv4 links

– Statement of general direction (SOD):

- IBM intends
  - in the future to enhance IBM CICS Explorer for IBM CICS Transaction Server for VSE/ESA to provide updates to CICS resources.
  - to add functionality that allows initial installation of z/VSE without requiring a physical tape.

- It is planned to reduce the AEWLC and MWLC list price of IPv6/VSE V1.1.

All statements regarding IBM's plans, directions, and intent are subject to change or withdrawal without notice.

z/Architecture

Required for 64 bit addressing, introduced with z/VSE 4.1

ESA/390	**z/Architecture**
Addressing up to 2 GB	Addressing up to and above 2GB
Addressing modes: 24, 31	Addressing modes: 24, 31, 64
8-byte PSW (4 byte instruction address)	16-byte PSW (8 byte instruction address)
general purpose registers: 4 bytes	general purpose registers: 8 bytes
control registers: 4 bytes	control registers: 8 bytes
access registers: 4 bytes	access registers: 4 bytes
Prefix area (low core): 4K	Prefix area (low core): 8K

Remark: VSE/ESA and z/VSE 3.1 based on ESA/390 Architecture

64 bit Addressing in z/VSE 4.3

- 64 bit **real** addressing only, introduced with z/VSE 4.1
- Processor storage support up to 32 GB
- Virtual address/data space size remains at max. 2 GB
- 64 bit virtual addressing not supported
- 64 bit addressing mode not supported for applications or ISVs

- Implementation transparent to user applications
- Performance: 64 bit real can reduce / avoid paging

- Many z/VSE environments can run without a page dataset (NOPDS option)

- 64 bit register support for programs

- 64 bit registers are not support by
  - CICS services
  - High level languages

64 bit real - Implementation

- IPL starts in ESA/390 mode and switches to z/Architecture mode during the IPL process
- Simulation of ESA/390 low core fields
- Only the z/VSE page manager has access to the area above 2GB
- Virtual pages can be backed by 64 bit real page frames
- Large pages (1 MB page frames) for dataspaces allocated in 64 bit real space
- PFIX or TFIX requests will use real page frames below 2 GB
- Page manager control blocks above 2 GB
- 64-bit page frames used directly for page-in and page-out I/O

64 bit real – Implementation …

- Hardware uses z/Architecture new and old PSWs and interrupt locations for interrupts
  - Interrupts: external, SVC, I/O, machine check, program check
  - Interrupt processing:
    hardware stores old PSW and interrupt information and passes control to interrupt new PSW
- Task save areas are extended.
  - Low order half (4 byte) of registers are located in problem program save area
  - High order half (4 byte) of registers are located in Shared Area (31 bit)

64 bit real – ESA/390 Emulation

- In z/VSE z/Architechture new PSWs point to emulation code
- When an interrupt occurs, emulation code
  - Translates z/Architecture old PSW into ESA/390 old PSW
  - Prepares ESA/390 interrupt information
  - Passes control to z/VSE interrupt handlers
- In most cases system programs use ESA/390 locations
  - Such as ESA/390 old PSWs
  - Interrupt handlers/dispatcher work with ESA/390 information/locations
  - Emulation guarantees that system code runs unchanged
- ESA/390 interrupt information is not used by hardware

64 bit real – ESA/390 Emulation - Example

Generated within Supervisor:
ESA/390 PC New PSW at 00000068: 000C0000 8000F142 (points to interrupt handler)
z/Arch PC New PSW at 000001D0: 00040000 80000000 00000000 0000F0B2
(points to emulation code)

Program check (page fault) occurs:
00000000000133B8 MVC D21F10009398 00506000
00000000000133B8 PROG 0011 -> 0000F0B2

Hardware sets:
z/Arch PC Old PSW at 00000150: 04040000 00000000 00000000 000133B8
z/Arch Transl. Excep. at 000000A8: 00000000 00506000 (page fault address)

Emulation code at F0B2 provides (31 bit addresses only):
ESA/390 PC Old PSW at 00000028: 040C0000 000133B8
ESA/390 Transl. Excep. at 00000090: 00506000
Supervisor can continue at F142 (program check handler) as in ESA/390 mode

z/VSE 5.1: 64 bit virtual

- GA 11/25/2011
- Support of 64 bit virtual addressing
- 64 bit area can be used for **data only**
  - No instruction execution above the bar
- **z/OS affinity:** APIs (IARV64 services) - to manage memory objects – compatible with z/OS
  - Private memory objects for use in one address space
  - Shared memory objects to be shared among multiple address spaces
- Maximum VSIZE still limited to 90 GB
- Access to memory objects via IARV64 services and switch into AMODE 64 (SAM64)
- Advantages:
  - Eases the access of large amounts of data, e.g. instead of using and managing data spaces
  - Reduces complexity of programs: Data contained in primary address space
  - Chosen design has no dependencies to existing APIs
  - Minor impact on existing system code

64 bit virtual - Naming Convention

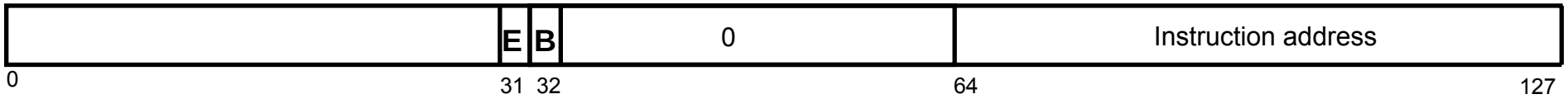
- Area above 2 GB - private area = **extended private area (EPA)**
- Area above 2 GB – shared area = **extended shared area (ESA)**
- Area above 2 GB – private or shared = **extended area**
- **The (2 GB) bar**: a line that separates the address space into storage below 2 GB (below the bar) and above 2GB (above the bar)
- **The (16 MB) line**: a virtual “line” marks the 16-megabyte address.
- 64 bit general purpose registers = **8 byte registers**
  - High order half = 0-31 bits of register
  - Low order half = 32-63 bits of register

Addressing Modes

- z/VSE 5.1 provides three addressing modes
  - AMODE 24 for instructions / data below 16 MB
  - AMODE 31 for instructions / data below the bar
  - AMODE 64 for instructions / data below 2 GB and data above 2 GB

- Change addressing mode
  - AMODESW macro to switch into AMODE 24 or AMODE 31
  - Set Addressing Mode (SAM) instructions to switch addressing modes
    - SAM24 to switch into AMODE 24
    - SAM31 to switch into AMODE 31
    - SAM64 to switch into AMODE 64
  - Branch and Save and Set Mode (BASSM) or Branch and Set Mode (BSM)

- Program Status Word (PSW)



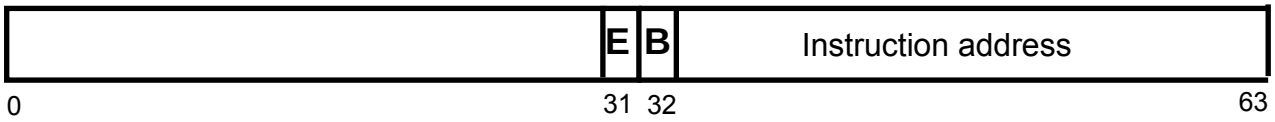
Extended (E) | Basic (B) addressing mode: 00 – 24 bit mode | 01 – 31 bit mode | 11 – 64 bit mode | 10 - invalid

Using 64 bit Addressing Mode

- 64 bit addressing mode required to access data above the bar
- The processor checks the addressing mode and truncates the answer
  - AMODE 24 – the processor truncates bits 0 through 39
  - AMODE 31 – the processor truncates bits 0 through 32
  - AMODE 64 – no truncation
- Before changing the addressing mode to AMODE 64 (via SAM64)
  - It may be necessary to clear the high-order half of registers to be used.
  - Use the LLGT (Load Logical Thirty One Bits) or LLGTR instruction to clear the high-order 33 bits
- Test Addressing Mode (TAM) instruction to test current addressing mode
- SAM64, BASSM and BSM are the only ways to set the AMODE to 64

Register saving – Extended save area

- If a task is interrupted, z/VSE will store the 64 bit registers.
  - Low-order of the registers to be stored in the problem program save area
  - High-order half of the registers to be stored in an extended task save area
- Pointer to the extended save area can be obtained via a GETFLD service
- Short form of PSW (8 byte) will be stored into the save area



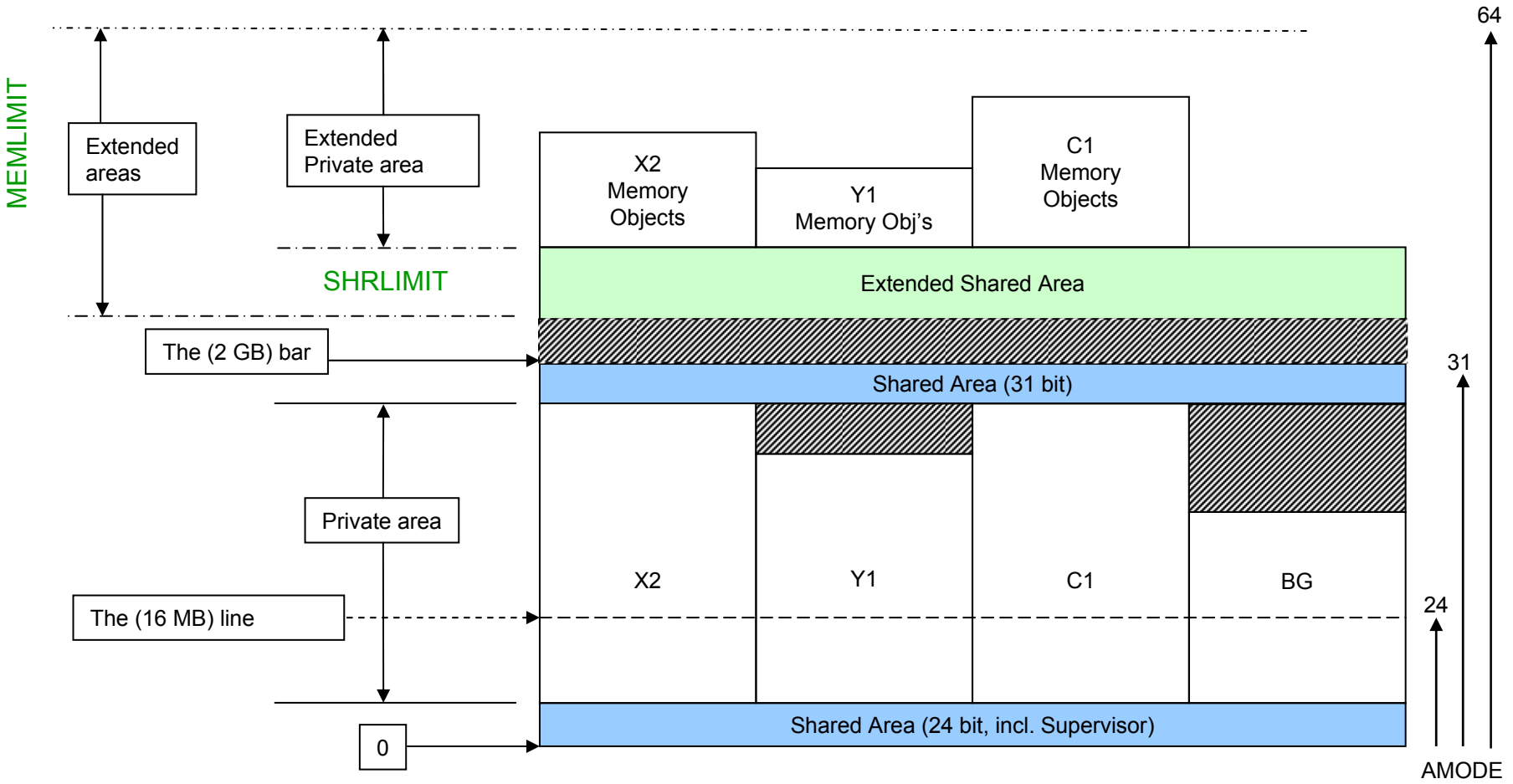
Extended (E) | Basic (B) addressing mode: 00 – 24 bit mode | 01 – 31 bit mode | 11 – 64 bit mode | 10 - invalid

- z/VSE exit routines provide 64 bit register support
- CICS services do not support 64 bit registers

Memory Objects

- "chunks" of virtual storage obtained by a program
- Allocated above the bar
- Contiguous range of virtual addresses
- Begins on a 1 MB boundary and is multiple of 1 MB in size
- Two types of memory objects:
  - **Private memory objects** are created within an address space
    - In extended private area (EPA)
  - **Shared memory objects** are created within extended shared area (ESA)
    - Can be accessed from any address space, that requests access

64 bit virtual - Address Space Layout



Virtual Storage Size (VSIZE)

- VSIZE to be specified in Supervisor statement at IPL =
  Size of private areas of all active partitions
  + size of SVA(24 bit)
  + size of SVA(31 bit)
  + size of page manager address spaces
  + size of defined data spaces
  + size of created memory objects

64 bit virtual – Define System Limits

SYSDEF statement to define the limits for memory objects

– Before IARV64 services can be used.
– SYSDEF MEMOBJ,MEMLIMIT=,SHRLIMIT=,LFAREA=,LF64ONLY
  - MEMLIMIT – maximum virtual storage available for memory objects
    - Theoretical maximum value is VSIZE.
  - SHRLIMIT – maximum virtual storage available for shared memory objects
    - = size of extended area, included in MEMLIMIT
  - LFAREA – maximum real storage to fix private memory objects
  - LF64ONLY – YES|NO – memory objects are fixed in 64 bit frames only

– Example:

```
sysdef memobj,memlimit=1g,shrlimit=100m,lfarea=50m
AR 0015 1I40I  READY
```

64 bit virtual – Display Memory Object Information

QUERY command to retrieve memory object information

- QUERY MEMOBJ displays
  - Effective settings of MEMLIMIT, SHRLIMIT, LFAREA, LF64ONLY
  - Summary information: virtual storage consumption of private / shared memory objects
- QUERY MEMOBJ,ALL displays
  - Additional statistic information
  - Virtual storage consumption of shared memory objects
  - Virtual storage consumption of private memory objects per partition

64 bit virtual – Display Memory Object Information …

Example:

```
query memobj
AR 0015            LIMITS    USED        HWM
AR 0015 MEMLIMIT:   1024M      6M         7M
AR 0015 SHRLIMIT:    100M      0M         0M
AR 0015 LFAREA:       50M          0K          0K
AR 0015 LF64ONLY: NO
AR 0015 1I40I  READY
query memobj,all
AR 0015      AREA MEMOBJ     HWM     LFAREA
AR 0015    SYSTEM     0M      0M                    SHRLIMIT:   100M
AR 0015        F4     1M      1M          0K
AR 0015        F5     1M      1M          0K
AR 0015        F6     1M      1M          0K
AR 0015        F7     1M      1M          0K
AR 0015        F9     1M      1M          0K
AR 0015        FA     1M      1M          0K
AR 0015     TOTAL     6M      7M          0K
AR 0015 MEMLIMIT: 1024M LFAREA:    50M              LF64ONLY:NO
AR 0015 1I40I  READY
```

MAP

MAP command to display current storage virtual storage layout

```
map
AR 0015  SPACE AREA       V-SIZE   GETVIS   V-ADDR   UNUSED NAME
AR 0015      S SUP          760K                0          $$A$SUPI
AR 0015      S SVA-24      1364K    2228K    BE000     768K
AR 0015      0 BG V        1280K    8960K   500000 1525760K
AR 0015      1 F1 V        1500K   29220K   500000       0K POWSTART
AR 0015      2 F2 V        2048K   49152K   500000       0K CICSICCF
AR 0015      3 F3 V         600K   14760K   500000       0K VTAMSTRT
AR 0015      4 F4 V        2048K   18432K   500000       0K PAUSEF4
AR 0015      5 F5 V         768K    4352K   500000       0K PAUSEF5
AR 0015      6 F6 V        1024K   50176K   500000       0K PAUSEF6
AR 0015      7 F7 V        1024K   19456K   500000       0K PAUSEF7
AR 0015      8 F8 V        2048K  151552K   500000       0K
AR 0015      9 F9 V        1024K    4096K   500000       0K PAUSEF9
AR 0015      A FA V        1024K    4096K   500000       0K PAUSEFA
AR 0015      B FB V         512K     512K   500000       0K SECSERV
AR 0015      S SVA-31      8608K   10848K 5E100000
AR 0015        DYN-PA         0K
AR 0015        DSPACE      6880K
AR 0015        SHR-64         0K
AR 0015        PRV-64      6144K
AR 0015        SYSTEM     32256K
AR 0015        AVAIL    7818272K
AR 0015        TOTAL    8257216K   <----'
AR 0015 1I40I  READY
```

MAP ...

MAP <partition> command to display current storage virtual storage layout

```
map f6
AR 0015  PARTITION: F6        SPACE-GETVIS.....:    (N/A)
AR 0015  SPACE....: 6         ALLOC (VIRTUAL)...:  51200K  ADDR: 500000
AR 0015  STATUS...: VIRTUAL     SIZE...........:   1024K
AR 0015  POWER-JOB: PAUSEF6      EXEC-SIZE.....:   1024K
AR 0015  JOBNUMBER: 34          GETVIS.........:  50176K
AR 0015  JOBNAME..: PAUSEF6      EXEC-GETVIS....:  50176K  ADDR: 600000
AR 0015                  ---->  PRV-64.........:      1M  HWM:       1M
AR 0015  PHASE....: TESTC64W
AR 0015  TASKS....: ANY       PFIX(BELOW)-LIMIT :      0K
AR 0015                                -ACTUAL:      0K
AR 0015                       PFIX(ABOVE)-LIMIT :      0K
AR 0015                                -ACTUAL:      0K
AR 0015                    PFIX(LFAREA)-ACTUAL:      0K  HWM:         0K
AR 0015 1I40I  READY
```

IARV64 Macro

- IARV64 macro - ported from z/OS – provides services to
  - Creates and frees storage areas above the bar
  - Manage the physical frames behind the storage
  - Requires SYSSTATE AMODE64=YES

- Programs use the IARV64 macro to obtain memory objects

- Services (IARV64 REQUEST=):
  - GETSTORE – create a private memory object
  - DETACH – free one or more memory objects
  - GETSHARED – create a memory object that can be shared across multiple address spaces
  - SHAREMEMOBJ – request that the specified address space be given access to a shared memory object
  - PAGEFIX – fix pages within one or more private memory objects
  - PAGEUNFIX – unfix pages within one or more private memory objects

Private Memory Object (PMO)

- Created by IARV64 GETSTOR
  - Successful creation depends on available virtual storage (VSIZE)
  - Allocated in extended private area (EPA) of an address space
  - Extended Private Area (EPA) only exists, if there is at least one PMO allocated.
  - All tasks within the address space (partition) may have access to PMOs
  - User token can be used to identify PMOs
  - The task creating the PMO is the PMO owner
- Free PMOs by IARV64 DETACH
  - One or more PMOs can only be freed, if task owns PMOs
- System frees PMOs, if owning task terminates
- Authorized programs may IARV64 PAGEFIX or PAGEUNFIX PMOs

Private Memory Object - Example

```
000100           PUNCH '  PHASE TESTC64,*'
000200    TITLE '*** TESTCASE TESTC64 ***'
000300 TESTC64   START X'78'
000400 TESTC64   AMODE 31
000500 TESTC64   RMODE 31
000600 *         TESTCASE WILL GET CONTROL IN AMODE 31
000700     →     SYSSTATE AMODE64=YES
000800           BASR  12,0
000900 BASE      EQU   *
001000           USING BASE,12
001100           LLGTR 12,12                CLEAR BITS 0 - 32
001200           LHI   0,DYNAREAL
001300 * GET STORAGE FOR WORK AREA
001400           GETVIS ADDRESS=(1),LENGTH=(0)
001500           LTR   15,15
001600           BNZ   ERRORGF
001700           LLGTR 13,1                 CLEAR BITS 0 - 32
001800           USING @DYNAREA,13
001900           MVC   4(4,13),=C'F6SA'
```

Private Memory Object - Example

```
002000 * OBTAIN A MEMORY OBJECT OF 1 MB, DON'T FORGET TO SET MEMLIMIT
002100       →  IARV64 REQUEST=GETSTOR,SEGMENTS=ONE_SEG,USERTKN=TOKEN,        *
002200                 ORIGIN=VIRT64
002300           LTR   15,15
002400           BNZ   ERRORIA
002500           LG    4,VIRT64                  GET ADDRESS OF MEMORY OBJECT
002600           LLGTR 2,2                       CLEAR BITS 0 - 32
002700           LHI   2,256                     SET LOOP COUNTER
002800       →   SAM64                           CHANGE TO 64 BIT MODE
002900 LOOP      DS    0H
003000           MVC   0(10,4),=CL10'TESTC64'    STORE TESTC64
003100           AHI   4,4096
003200           BRCT  2,LOOP
003210       →   SAM31
003300 * FREE MEMORY OBJECT
003400       →   IARV64 REQUEST=DETACH,MATCH=USERTOKEN,USERTKN=TOKEN,          *
003500                 COND=YES
003600           LTR   15,15
003700           BNZ   ERRORIA
003900           DROP  13
```

Private Memory Object - Example

```
004000           LHI    0,DYNAREAL
004100           LR     1,13
004200 * FREE WORK AREA
004300           FREEVIS ADDRESS=(1),LENGTH=(0)
004400           LTR    15,15
004500           BNZ    ERRORGF
004600           EOJ    RC=0
004700 * GETVIS, FREEVIS ERROR
004800 ERRORGF   DS     0H
004900           EOJ    RC=8
005000 * IARV64 ERROR
005100 ERRORIA   DS     0H
005200           EOJ    RC=12
005300           DROP   12
005400 * BEGIN DATA AREA
005500           DS 0D
005600 ONE_SEG   DC     FD'1'
005700 TOKEN     DC     FD'1'
005800           LTORG
005900 @DYNAREA DSECT
006000 SAVEAREA DS     36F
006100 VIRT64   DS AD
006200 DYNAREAL EQU    *-@DYNAREA
006300           END    TESTC64
```

Shared Memory Objects (SMO)

- Created by IARV64 GETSHARED
  - Successful creation depends on available virtual storage (VSIZE)
    - Authorized programs only
  - Allocated in extended shared area (ESA)
  - Size of ESA depends on SHRLIMIT
  - ESA only exists, if there is at least one memory object allocated (PMO or SMO)
  - Similar to SVA storage
    - No automatic addressability / access to SMO storage
  - Any z/VSE user task may have access to SMO storage
- Allow access to SMO storage by IARV64 SHAREMEMOBJ
  - Tasks get access to specified memory objects = shared interest
  - Shared interest is owned by maintask
  - All tasks within partition have access
  - Shared interest can be removed via IARV64 DETACH AFFINITY=LOCAL
  - When maintask terminates, system removes all shared interests owned by it
- The task creating the SMO is not the owner
  - SMO is always owned by the system = system affinity
- To free a SMO any authorized program may use
  - IARV64 DETACH AFFINITY=SYSTEM
  - The system will free the SMO only, if all shared interests are removed

Shared Memory Object – Example…

IARV64 GETSHARED example creates a 1 MB shared memory object

```
000100 * OBTAIN SHARED MEMORY OBJECT
000200           IARV64 REQUEST=GETSHARED,SEGMENTS=ONE_SEG,KEY=MYKEY,          *
000300                  USERTKN=USERTKNA,ORIGIN=VIRT64,COND=YES
000400           LTR   15,15
000500           BNZ   ERRORIA
000600
000700
000800 * DATA AREA
000900 ONE_SEG  DC    FD'1'
001000 USERTKNA DC    0D'0'
001100          DC    F'15'         HIGH HALF MUST BE NON-ZERO FOR
001200                                    AUTHORIZED PROGRAMS
001300          DC    F'1'          USER TOKEN OF 1
001400 VIRT64   DC    AD(0)         64 BIT ADDRESS OF MEMORY OBJECT
001500 MYKEY    DC    X'90'
```

Shared Memory Object – Example…

IARV64 SHAREMEMOBJ allows access to shared memory object

```
* GET ACCESS TO SHARED MEMORY OBJECT
         LA    2,RLISTPTR
         IARV64 REQUEST=SHAREMEMOBJ,RANGLIST=(2),                      *
               NUMRANGE=1,USERTKN=USERTKNS,COND=YES
         LTR   15,15
         BNZ   ERRORIA


* DATA AREA
USERTKNS DC    0D'0'
         DC    F'15'          HIGH HALF MUST BE NON-ZERO FOR
                                     AUTHORIZED PROGRAMS
         DC    F'2'           USER TOKEN OF 2
RLISTPTR DC    AD(RLIST)      POINTER TO A LIST OF 64 BIT ADDRESSES
RLIST    DS    AD             64 BIT ADDRESS OF MEMORY OBJECT
         DC    AD(0)          RESERVED
```

Shared Memory Object – Example…

IARV64 DETACH to remove shared interest

```
003600 * REMOVE SHARED INTEREST FOR MEMORY OBJECT
003700           IARV64 REQUEST=DETACH,AFFINITY=LOCAL,MATCH=SINGLE,           *
003800                  MEMOBJSTART=VIRT64,USERTKN=USERTKNS,COND=YES
003900           LTR    15,15
004000           BNZ    ERRORIA
004100
004200
004300 * DATA AREA
004400 VIRT64    DS     AD            64 BIT ADDRESS OF MEMORY OBJECT
004500 USERTKNS  DC     0D'0'
004600           DC     F'15'         HIGH HALF MUST BE NON-ZERO FOR
004700                                          AUTHORIZED PROGRAMS
004800           DC     F'2'          USER TOKEN OF 2
```

Shared Memory Object – Example…

IARV64 DETACH to free a shared memory object

```
005100 * FREE SHARED MEMORY OBJECT
005200           IARV64 REQUEST=DETACH,AFFINITY=SYSTEM,MATCH=SINGLE,          *
005300                  MEMOBJSTART=VIRT64,USERTKN=USERTKNA,COND=YES
005400           LTR   15,15
005500           BNZ   ERRORIA
005600
005700
005800 * DATA AREA
005900 VIRT64   DS    AD           64 BIT ADDRESS OF MEMORY OBJECT
006000 USERTKNS DC    0D'0'
006100           DC    F'15'        HIGH HALF MUST BE NON-ZERO FOR
006200                                       AUTHORIZED PROGRAMS
006300           DC    F'1'         USER TOKEN OF 1
```

Memory Objects …

- Protecting storage above the bar
  - IARV64 KEY parameter to assign storage key to the memory object
    - Default storage key = PSW key of caller
    - Unauthorized caller can set key 9 (all tasks can run in key 9)
    - Authorized callers can set any key
  - IARV64 FPROT parameter to fetch-protect the memory object

- Fix / unfix pages of a memory object
  - IARV64 PAGEFIX – fix pages within one or more private memory objects
  - IARV64 PAGEUNFIX – unfix pages within one or more private memory objects

- Dumping memory objects
  - SDUMPX macro with LIST64 parameter can be used to dump memory objects
  - Standalone dump
    - STDOPT | OPTION SADUMP parameter controls the priority of the private memory object
    - Shared memory objects are always included in one single dump file with lowest priority
      - no matter what values are specified for SADUMP

Memory Objects ... (Announced April 2013)

- System dump may be taken in case of abnormal termination dependent on JCL options
  - New JCL option MODUMP, NOMODUMP
- If program running in 64 bit mode and registers hold 64 bit addresses
  - The dump routine will take 4K on either side of this address
- Memory object dumps are written to SYSLST only
  - Partitions dumps will be written to dump library or SYSLST dependent on OPTIONs

- New standard option: STDOPT SADMPSMO=YES|NO
  - Controls, if standalone dump should include **shared memory objects**

- (Standard) option STDOPT SADUMP=(n,m,o)
  - Controls, if standalone dump should include **private memory objects**

64 bit virtual – Exit routines

- STXIT routines – AB | IT | OC | PC
  - New parameter: AMODE=ANY64
    - E.g. STXIT AB, rtnaddr, savearea,…,AMODE={24|ANY|ANY64}
  - AMODE specifies
    - Addressing mode in which the exit receives control
    - Layout of save area (as of macro MAPSAVAR)
  - Length of save area
    - AMODE=24 -> 72 bytes
    - AMODE=ANY -> 216 bytes
    - AMODE=ANY64 -> 420 bytes

64 bit virtual I/O for applications

- Available with z/VSE 5.1 APAR DY47419
- SYSCOM bit IJBIO64E in IJBIOFL1, if 64 bit virtual I/O support available

- I/O buffers can now be created above the bar (above 2 GB)
- I/O buffers in **private memory objects** supported only
- I/O control blocks to be allocated below the bar (in 31 bit storage)

- Supported for ECKD devices

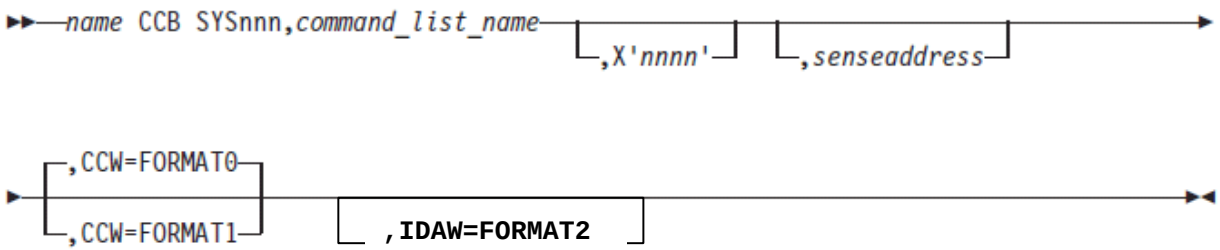
- CCB macro with a new parameter: IDAW=FORMAT2
- CCB points to a Format-0 or Format-1 CCW
- CCW with IDA-flag and data address point to a single Format-2 IDAW containing a 64 bit virtual address.

- I/O buffer will be TFIXed by I/O Supervisor, not necessary to PFIX the I/O buffer

- Not supported for
  - FBA / SCSI devices
  - Tape devices
  - LIOCS

64 bit I/O Request: User CCB for EXCP

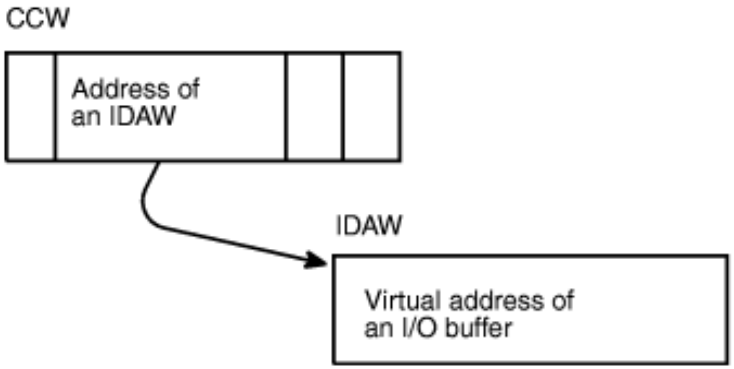
- CCB macro will be extended by a new parameter
  - IDAW=FORMAT2: FORMAT-2-IDAW control set



- Format-1 IDAW not supported
- 2K-IDAW control not supported (→ block size=4K)

64 bit I/O Request: User CCW

- Indirect-Data-Adressing bit set
- User passes virtual IDAW → IDAL with only one list entry



- CCW can either be Format-0 or Format-1

Steps to perform a 64-bit I/O request:

1) Create a private memory object (IARV64 REQUEST=GETSTOR)

2) Prepare CCWs (IDA-bit set) pointing to a Format 2 IDAW with a virtual 64 bit address

3) Prepare a CCB (using the CCB macro) with IDAW=FORMAT2

4) Issue an I/O request using the EXCP macro

5) After I/O operations completed: Detach the memory object (IARV64 REQUEST= DETACH)

64 bit virtual - Considerations

- Memory objects can be allocated **for data only**.
  - Instruction execution above the bar (RMODE 64) not supported
- High Level Assembler support only.
  - High level languages (COBOL, PL/I, C, RPG, …) do not support 64 bit registers or 64 bit mode.
  - AMODE 64 attribute should not be used.
- AMODE 64 is not supported by
  - LOAD / CDLOAD and the linkage editor
  - z/VSE system services (Supervisor, VSAM, BAM, DL/I, …)
  - Space switching Program Calls (ss-PCs)
  - Data areas for system services including **I/O buffers** to be allocated below the bar.
- Services in online environment do not support 64 bit registers or AMODE 64
  - ICCF pseudo partitions
- CICS considerations
  - CICS services do not save / restore the high order half of 64 bit registers
  - The program must save them before invoking a CICS service and restore them afterwards
  - The program has to switch into AMODE 31 or 24 before invoking a CICS service

More Information

… on VSE home page: http://ibm.com/vse

- Ingolf's z/VSE blog: https://www.ibm.com/developerworks/mydeveloperworks/blogs/vse
- Hints and Tips for z/VSE 5.1:
  - http://www.ibm.com/systems/z/os/zvse/documentation/#hints
- 64 bit virtual information:
  - IBM z/VSE Extended Addressability, Version 5 Release 1
  - IBM z/VSE System Macro Reference, Version 5 Release 1
- CICS Explorer: http://www.ibm.com/software/htp/cics/explorer/
- IBM Redbooks:
  - Introduction to the New Mainframe: z/VSE Basics http://www.redbooks.ibm.com/abstracts/sg247436.html?Open
  - Security on IBM z/VSE – updated http://www.redbooks.ibm.com/Redbooks.nsf/RedbookAbstracts/sg247691.html?Open
  - z/VSE Using DB2 on Linux for System z http://www.redbooks.ibm.com/abstracts/sg247690.html?Open