IBM System z

Technical University 2011

64 bit virtual - Exploitation of z/VSE 5.1

zDG05

Ingolf Salm
salm@de.ibm.com

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

- z/VSE 5.1 Preview
- z/Architecture
- z/VSE 4.3 – 64 bit real addressing
- z/VSE 5.1 – 64 bit virtual addressing
  - Memory objects
  - Virtual storage size
  - IARV64 services
  - Addressing modes
  - Considerations

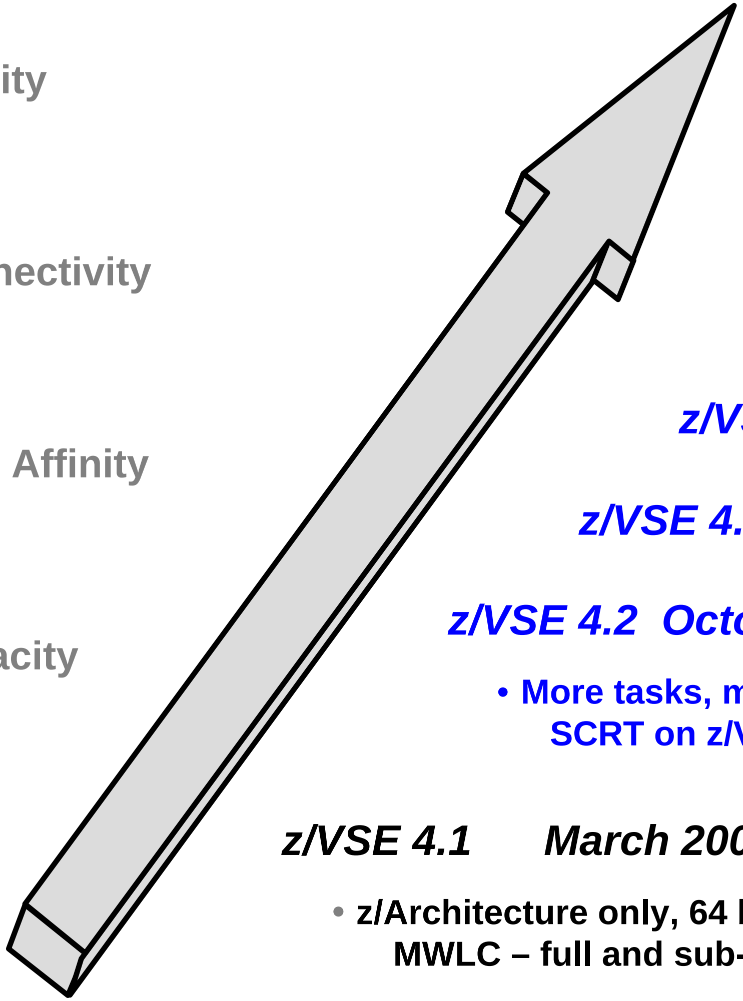
VSE Roadmap

Quality

Connectivity

z/OS Affinity

Capacity



z/VSE 5.1 4Q/2011

- **Preview: 64 bit virtual, SOD CICS Explorer**

z/VSE 4.3   11/2010

- **Virtual storage constraint relief, 4 digit cuus**

z/VSE 4.2.2 04/2010 **– IPv6/VSE 05/2010**

z/VSE 4.2.1 July 2009 - **PAV, EF for z/VSE 1.2**

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, SoD for CICS/VSE**

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 - Preview

- Preview: 04/12/2011, planned GA 4Q2011
- **64 bit virtual**
- Introduces Architectural Level Set (ALS) that requires System z9 or later
- zEnterprise 196 exploitation
- 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.

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

- VSE/ESA and z/VSE 3.1 based on ESA/390 Architecture

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
Interrupt information (old / new PSW, …): in first prefix page	Interrupt information (old / new PSW, …): in second prefix page
	4 byte register instructions use low order 4 bytes of register only new instructions for 8 byte register content

64 bit Addressing in z/VSE 4.3

- Processor storage support up to 32 GB
- 64 bit **real** addressing only, introduced with z/VSE 4.1
- 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 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

- In z/VSE z/Architechture new PSWs point to emulation code
  - Prepares ESA/390 interrupt information
  - Pass control to z/VSE interrupt handlers
  - ESA/390 interrupt information is not used by hardware

- 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 system Getvis (Any)

- Applications may use 8 byte registers
  - **But** only selected system routines can run in 64 bit mode

64 bit real – ESA/390 Emulation

- In most cases system programs use ESA/390 locations
  - Such as ESA/390 old PSWs
  - Emulation guarantees that system code runs unchaged

- When an interrupt occurs, emulation code provides
  - Transalation of z/Architecture old PSW into ESA/390 old PSW
  - Setup of ESA/390 interrupt information
  - Continuation at ESA/390 new PSW address (z/VSE interrupt handler)

- Interrupt handlers/dispatcher work with ESA/390 information/locations

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:

```
0000000000133B8    MVC   D21F10009398    00506000
0000000000133B8    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:

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

- Previewed on April 12, 2011
- Support 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)

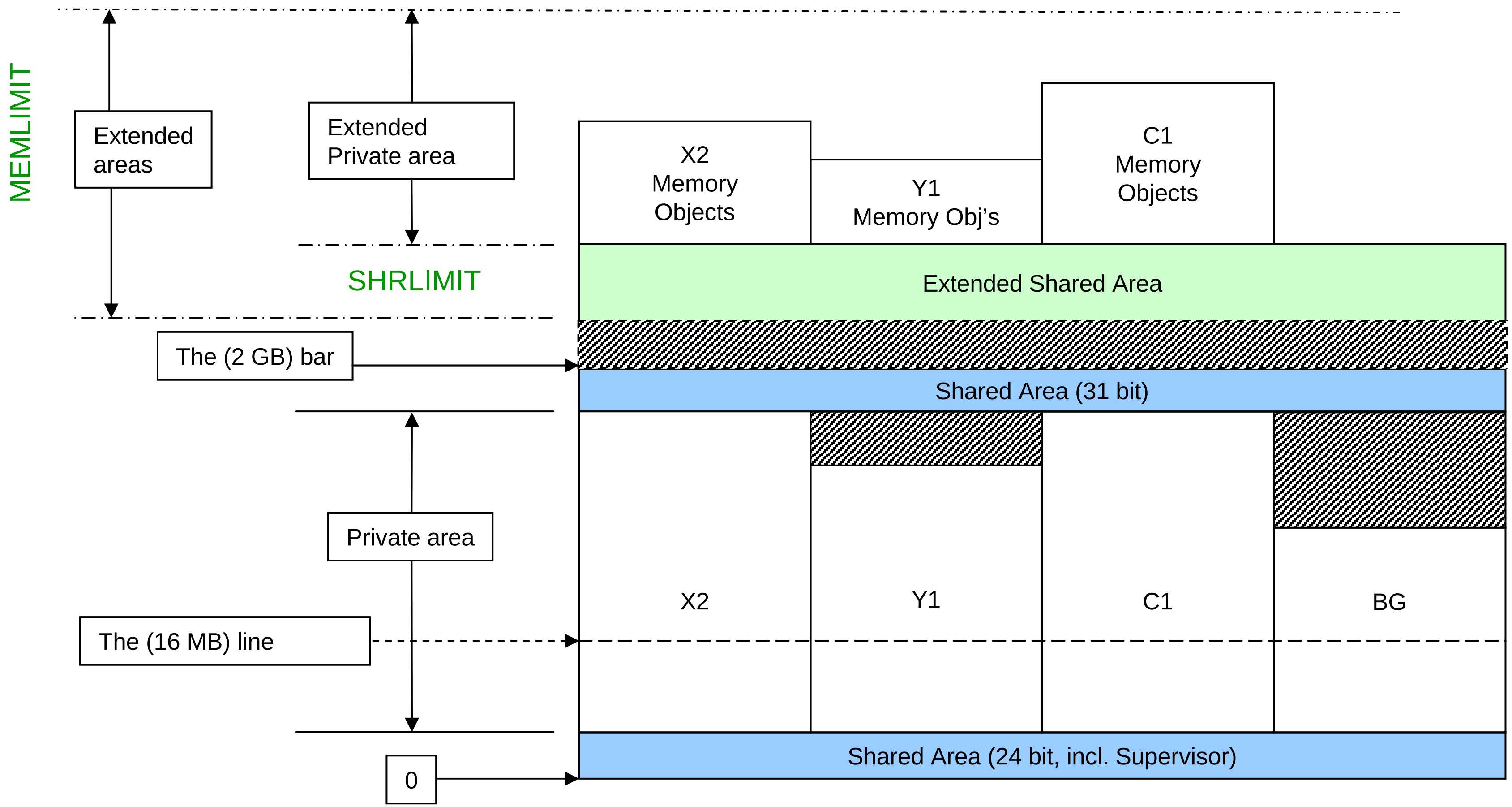
- **The (16 MB) line**: a virtual “line” marks the 16-megabyte address.

- 64 bit general purpose registers = **8 byte registers**
  - High order half = 0-31 bits of register
  - Low order half = 32-63 bits of register

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

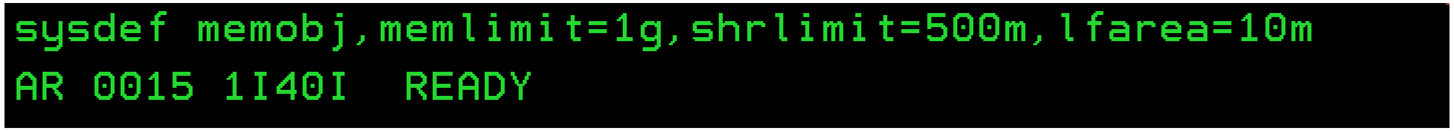
  + size of created memory objects

64 bit virtual – Define System Limits

SYSDEF statement to define the limits for memory objects

- Before IARV64 macro 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

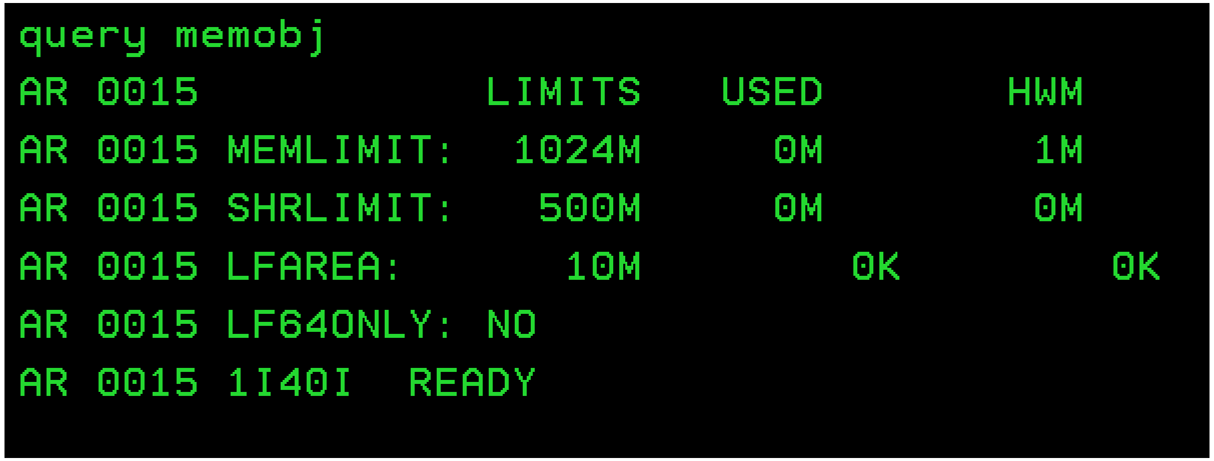
- Example:



```
sysdef memobj,memlimit=1g,shrlimit=500m,lfarea=10m
AR 0015 1I40I  READY
```

64 bit virtual – Display Memory Object Information

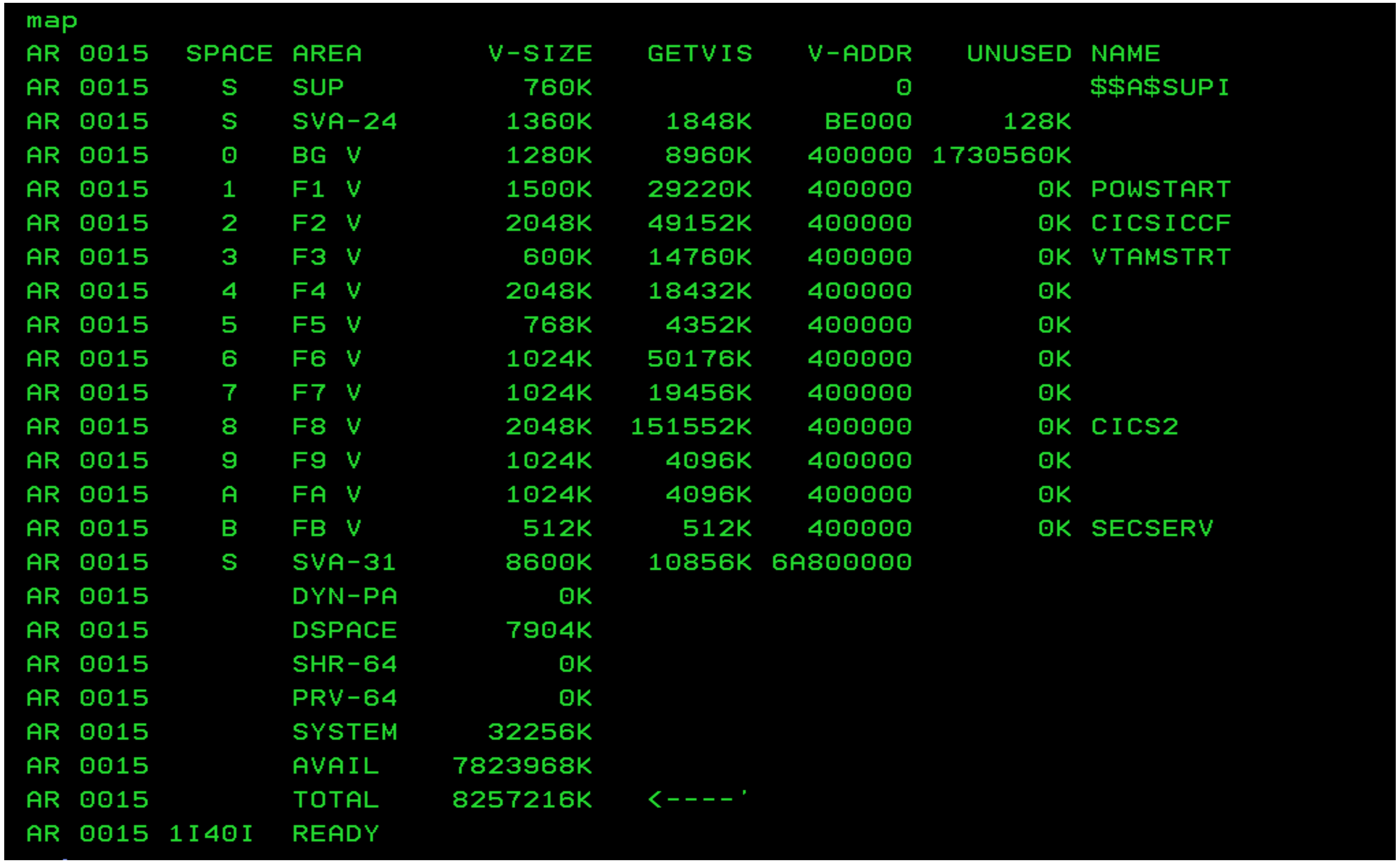
- QUERY command to retrieve memory object information
  - QUERY MEMOBJ displays
    - Effective settings of MEMLIMIT, SHRLIMIT; LFAREA, LF64ONLY
    - Summary information: virtual storage consumption of private / shared memory objects
  - QUERY MEMOBJ,ALL displays
    - Additional statistic information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



```
query memobj
AR 0015           LIMITS   USED       HWM
AR 0015 MEMLIMIT:  1024M     0M        1M
AR 0015 SHRLIMIT:   500M     0M        0M
AR 0015 LFAREA:      10M        0K        0K
AR 0015 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      1360K    1848K     BE000      128K
AR 0015    0   BG V        1280K    8960K    400000  1730560K
AR 0015    1   F1 V        1500K   29220K    400000        0K POWSTART
AR 0015    2   F2 V        2048K   49152K    400000        0K CICSICCF
AR 0015    3   F3 V         600K   14760K    400000        0K VTAMSTRT
AR 0015    4   F4 V        2048K   18432K    400000        0K
AR 0015    5   F5 V         768K    4352K    400000        0K
AR 0015    6   F6 V        1024K   50176K    400000        0K
AR 0015    7   F7 V        1024K   19456K    400000        0K
AR 0015    8   F8 V        2048K  151552K    400000        0K CICS2
AR 0015    9   F9 V        1024K    4096K    400000        0K
AR 0015    A   FA V        1024K    4096K    400000        0K
AR 0015    B   FB V         512K     512K    400000        0K SECSERV
AR 0015    S   SVA-31      8600K   10856K  6A800000
AR 0015        DYN-PA         0K
AR 0015        DSPACE      7904K
AR 0015        SHR-64         0K
AR 0015        PRV-64         0K
AR 0015        SYSTEM     32256K
AR 0015        AVAIL    7823968K
AR 0015        TOTAL    8257216K   <----'
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

- 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

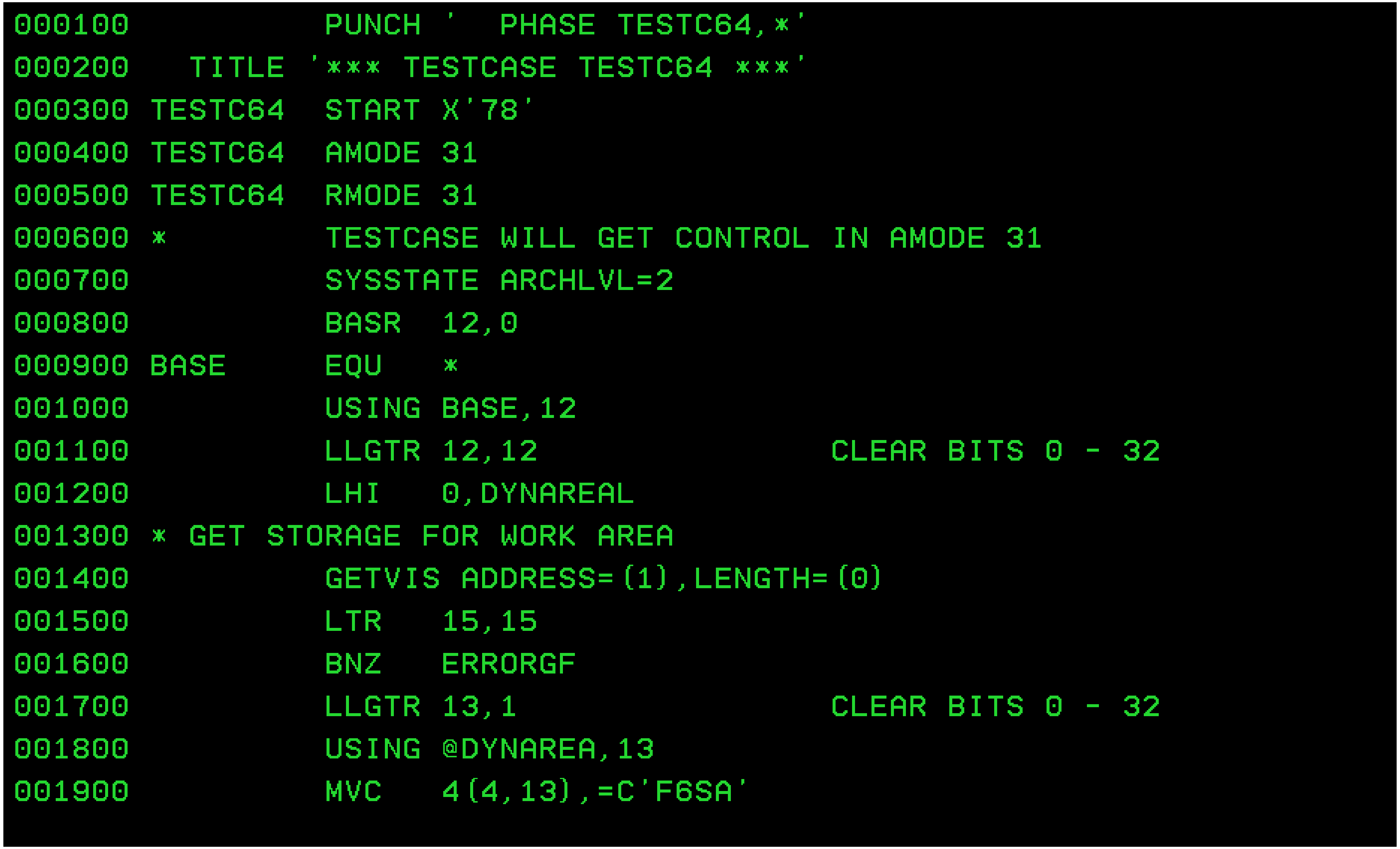
Private Memory Object (PMO)

- Created by IARV64 GETSTOR
  - Successful creation depends on available virtual storage (VSIZE)
  - Allocated in extended private area (EPA) of an address space
  - 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

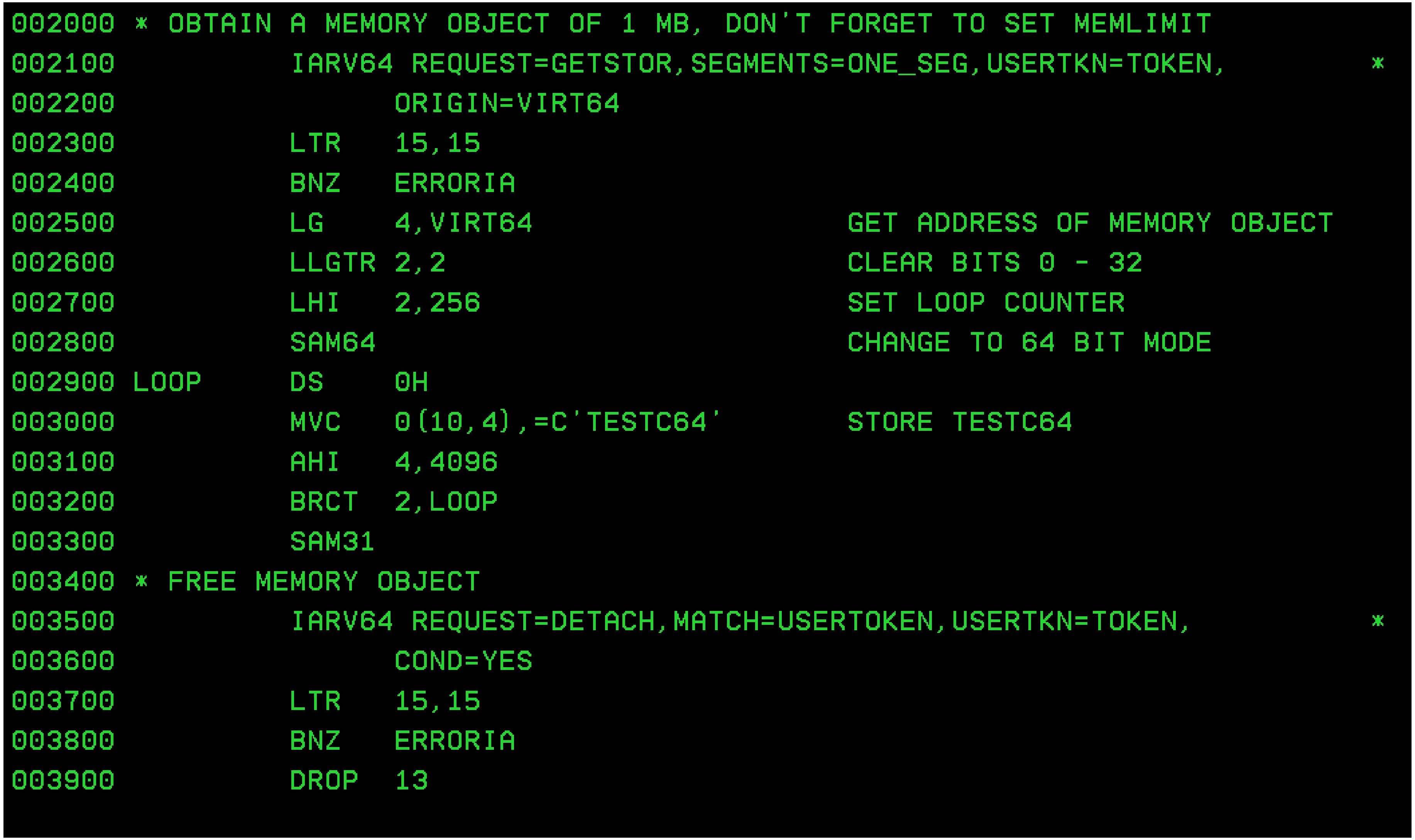
- Authorized programs may IARV64 PAGEFIX or PAGEUNFIX PMOs

Private Memory Object - Example



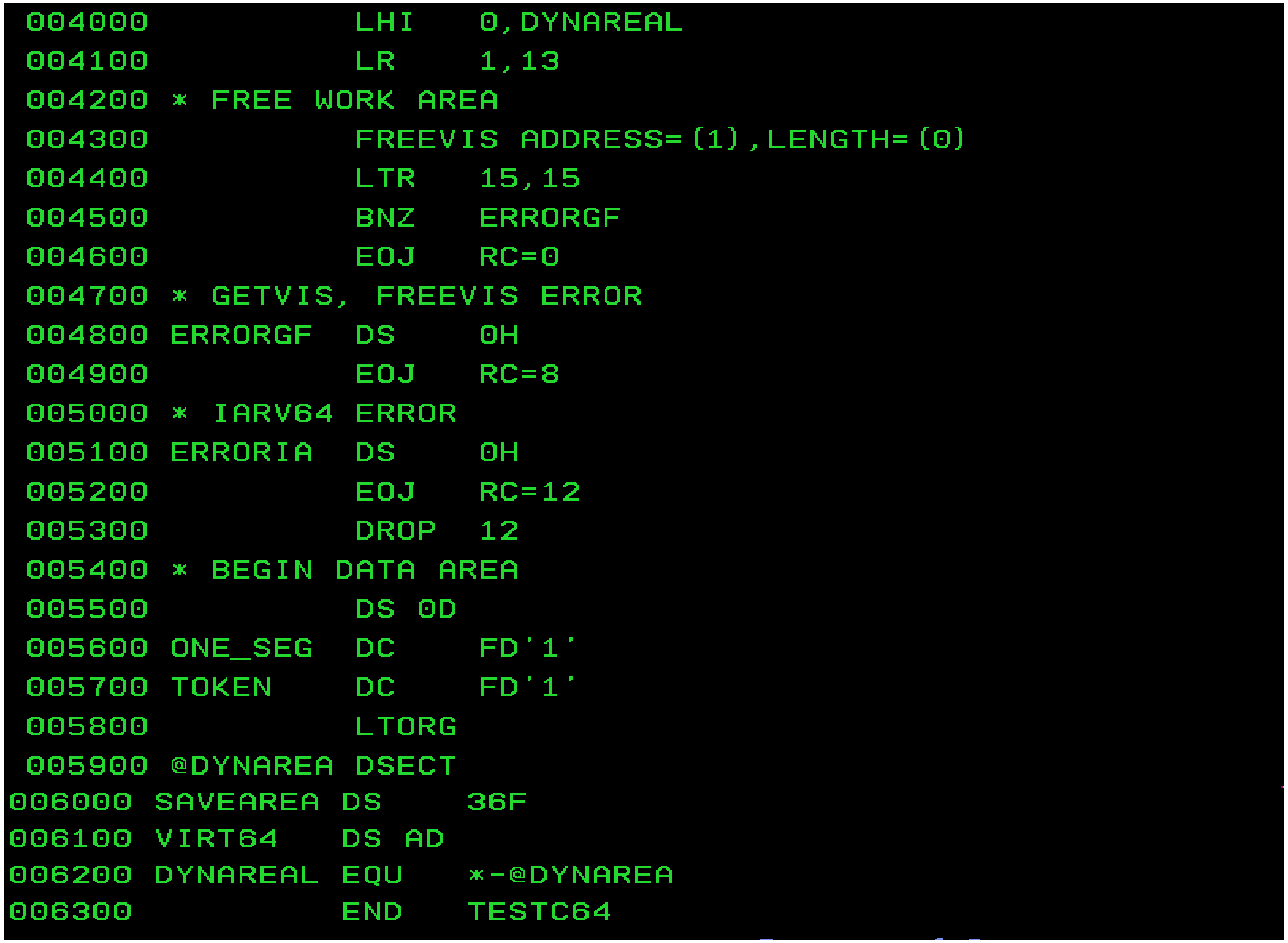
```
         PUNCH '  PHASE TESTC64,*'
  TITLE '*** TESTCASE TESTC64 ***'
TESTC64  START X'78'
TESTC64  AMODE 31
TESTC64  RMODE 31
*        TESTCASE WILL GET CONTROL IN AMODE 31
         SYSSTATE ARCHLVL=2
         BASR  12,0
BASE     EQU   *
         USING BASE,12
         LLGTR 12,12               CLEAR BITS 0 - 32
         LHI   0,DYNAREAL
* GET STORAGE FOR WORK AREA
         GETVIS ADDRESS=(1),LENGTH=(0)
         LTR   15,15
         BNZ   ERRORGF
         LLGTR 13,1                CLEAR BITS 0 - 32
         USING @DYNAREA,13
         MVC   4(4,13),=C'F6SA'
```

Private Memory Object - Example



```
* OBTAIN A MEMORY OBJECT OF 1 MB, DON'T FORGET TO SET MEMLIMIT
         IARV64 REQUEST=GETSTOR,SEGMENTS=ONE_SEG,USERTKN=TOKEN,        *
               ORIGIN=VIRT64
         LTR   15,15
         BNZ   ERRORIA
         LG    4,VIRT64                  GET ADDRESS OF MEMORY OBJECT
         LLGTR 2,2                       CLEAR BITS 0 - 32
         LHI   2,256                     SET LOOP COUNTER
         SAM64                           CHANGE TO 64 BIT MODE
LOOP     DS    0H
         MVC   0(10,4),=C'TESTC64'       STORE TESTC64
         AHI   4,4096
         BRCT  2,LOOP
         SAM31
* FREE MEMORY OBJECT
         IARV64 REQUEST=DETACH,MATCH=USERTOKEN,USERTKN=TOKEN,          *
               COND=YES
         LTR   15,15
         BNZ   ERRORIA
         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)
  - 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 use 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

Shared Memory Objects (SMO) - Ownership

- 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

```
IARV64 REQUEST=GETSHARED,                                      +
          SEGMENTS=ONE_SEG,                                    +
          USERTKN=USERTKNA,                                    +
          ORIGIN=VIRT64_ADDR,                                  +
          COND=YES,                                            +
           FPROT=NO,                                           +
           KEY=MYKEY
ONE_SEG   DC    FD’1’
USERTKNA DC     0D’0’
          DC    F’15’      High Half must be non-zero for authorized programs
          DC    F’1’       UserToken of 1
VIRT64_ADDR DS D            64 bit address of memory object
MYKEY     DC    X’D0’
```

Shared Memory Object – Example…

- IARV64 SHAREMEMOBJ allows access to shared memory object

```
              IARV64 REQUEST=SHAREMEMOBJ,                       +
                     USERTKN=USERTKNS,                           +
                     RANGLIST=RLISTPTR,                          +
                     NUMRANGE=1,                                 +
                     COND=YES
USERTKNS DC    0D’0’
              DC    F’15’      High Half must be non-zero for authorized programs
              DC    F’2’       User Token of 2 (can be different than GETSHARED  request)
RLISTPTR   DS     AD          64 bit address of memory object
```

Shared Memory Object – Example…

- IARV64 DETACH to remove shared interest

```
           IARV64 REQUEST=DETACH,                       +
                  AFFINITY=LOCAL,                       +
                  COND=YES,                             +
                  MATCH=SINGLE,                         +
                  MEMOBJSTART=VIRT64_ADDR,              +
                  USERTKN=USERTKNS
VIRT64_ADDR DS AD
USERTKNS    DC   0D’0’
            DC    F’15’
            DC    F’2’        User Token of 2
```

Shared Memory Object – Example…

- IARV64 DETACH to free a shard memory object

```
                 IARV64 REQUEST=DETACH,                        +
                        AFFINITY=SYSTEM,                       +
                        COND=YES,                              +
                        MATCH=SINGLE,                          +
                        MEMOBJSTART=VIRT64_ADDR,               +
                        USERTKN=USERTKNA
VIRT64_ADDR DS AD
USERTKNA DC     0D’0’
             DC     F’15’
             DC     F’1’        UserToken of 1
```

Memory Objects …

- Protecting storage above the bar
  - IARV64 KEY parameter to assign storage key to the memory object
  - 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

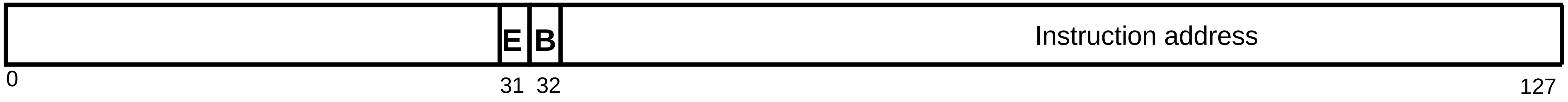
- Dumping memory objects
  - SDUMPX macro with LIST64 parameter can be used to dump memory objects
  - New SADUMP option to include memory objects in a standalone dump

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
  - – BASSM or 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 or LLGTR instruction to clear the high-order 33 bytes

- Test Addressing Node (TAM) instruction to test current addressing mode

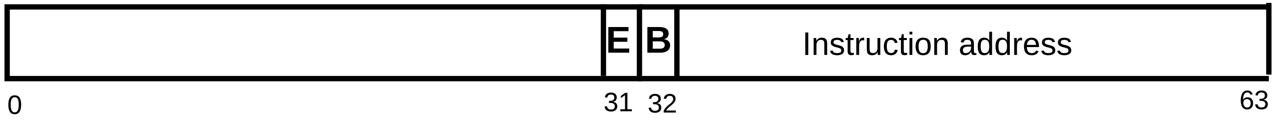
- SAM64, BASSM and BSM are the only ways to set the AMODE to 64

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

64 bit virtual - Considerations

- Memory objects can be allocated for data only.
  - Execution above the bar is not supported.
- z/VSE compilers (COBOL, PL/I, C, RPG) do not support AMODE 64.
  - High Level Assembler support only.
- LOAD / CDLOAD and the linkage editor do not support AMODE 64 attribute.
- Space switching Program Calls (ss-PCs) are not supported in AMODE 64.
- All z/VSE system services (Supervisor, VSAM, BAM, DL/I, …) to be called in AMODE 24 / 31.
  - IARV64 services may be called in AMODE 64
- Data areas for system services including I/O buffers to be allocated below the bar.

64 bit virtual – Considerations …

- The Supervisor code continues to use the short form of the PSW (8 byte).

- 64 bit addressing is not supported in ICCF pseudo partitions.

- CICS services **do not** support 64 bit registers or AMODE 64.

More Information

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

- z/OS manuals describing 64 bit address spaces and IARV64 services:
  - SA22-7614-07: z/OS V1R11.0 MVS Programming Extended Addressability Guide
  - SA22-7610-17: z/OS V1R11.0 MVS Programming Authorized Assembler Services Reference Vol 2 (EDTINFO-IXGWRITE)
  - SA22-7607-15: z/OS V1R11.0 MVS Programming Assembler Services Reference Vol 2 (IARR2V-XCTLX)
  - SA22-7605-11: z/OS V1R11.0 MVS Programming Assembler Services Guide
  - Corresponding online books are at http://www-03.ibm.com/systems/z/os/zos/bkserv/r11pdf/#zsys