

## 64-bit on zSeries Processors

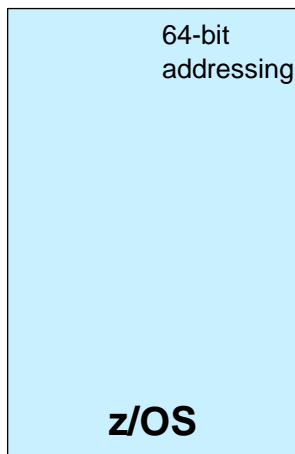
# 64-bit Real and Virtual Storage



## z/OS Real Storage Support



### Real Storage



z800/z900/z990

- ☐ Available since V1 R1
- ☐ Constraint relief  
For workloads limited by 2 GB real storage limit
- ☐ Improved Performance  
Expanded storage paging overhead eliminated
  - All memory configured as real storage

### Ease of migration

- ☐ Application transparency
  - 24- & 31-bit apps run unchanged
- ☐ Minimal actions to take
- ☐ Flexible migration paths

## Migration to zArchitecture (64-bit)



- ❑ Expanded Storage Support
  - When in ESA/390 mode (31-bit):
    - Configure external storage as expanded
  - When in z/Architecture mode (64-bit):
    - Configure all storage as central (real)
    - Hiperspace services re-implemented to use central storage instead of expanded storage
- ❑ No change expected for applications and middleware
  - No incompatible API changes
  - No recompiles expected; 24- and 31-bit applications run unchanged
  - Other system uses of expanded storage also re-implemented
  - Even low level authorized services remain compatible

## 64-bit Migration Considerations



- ❑ Few products or applications are expected to be affected
  - Products that issue their own I/O instructions (SSCH)
    - Database management products
  - Products that depend on real addresses (LRA)
    - Usually middleware
    - Not usually used by customer applications
  - Performance reporting tools or capacity planning tools
    - Tools that need to monitor and report on the additional storage

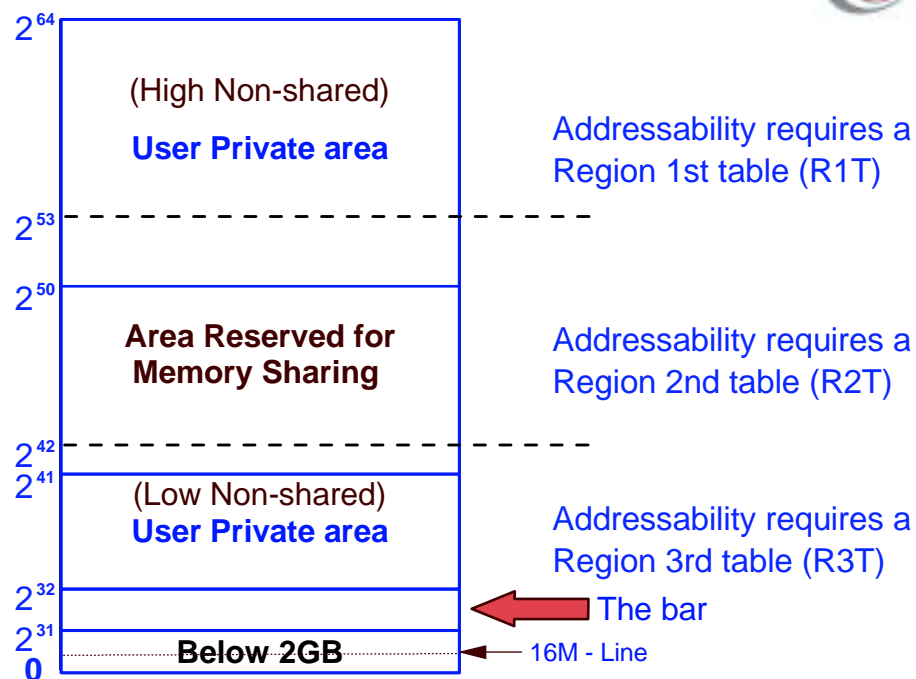
## Migration to zArchitecture



### ❑ Systems Programmers:

- Configure all processor memory for the image as Central Storage
- Review LOADxx for correct initialization parameter
- Re-IPL the image
- Control access to storage (MEMLIMIT)

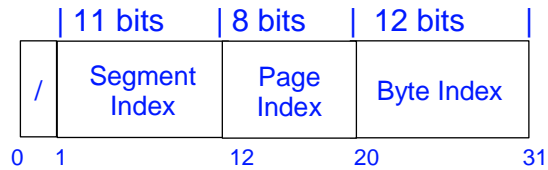
## Address Space Memory Map



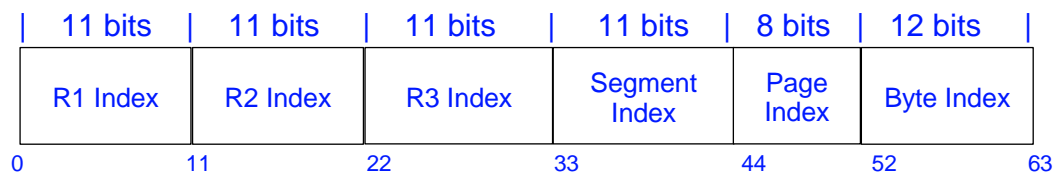
## Virtual Address Formats



### 31-bit Virtual Address



### 64-bit Virtual Address



## Size and Number Notation



| <u>Symbol</u> | <u>Decimal value</u>      | <u>Power of 2</u> |
|---------------|---------------------------|-------------------|
| K (kilo)      | 1,024                     | 2**10             |
| M (mega)      | 1,048,576                 | 2**20             |
| G (giga)      | 1,073,741,824             | 2**30             |
| T (tera)      | 1,099,511,627,776         | 2**40             |
| P (peta)      | 1,125,899,906,842,624     | 2**50             |
| E (exa)       | 1,152,921,504,606,846,976 | 2**60             |

## Examples

---



2,048 can be expressed as 2K.  
4,096 can be expressed as 4K.  
65,536 can be expressed as 64K.  
 $2^{24}$  can be expressed as 16M.  
 $2^{31}$  can be expressed as 2G.  
 $2^{43}$  can be expressed as 8T.  
 $2^{64}$  can be expressed as 16E.

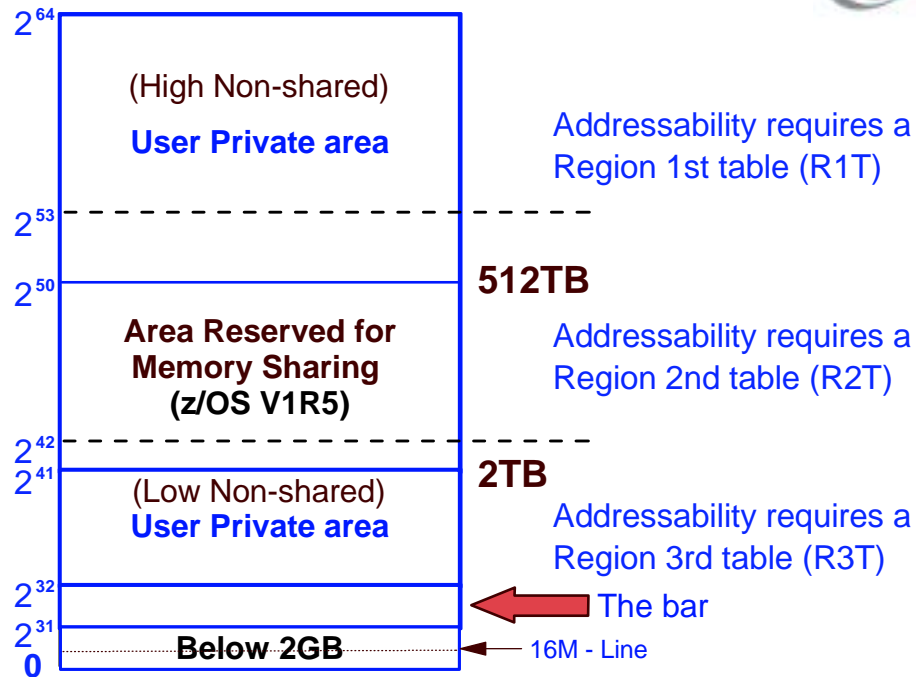
## 64-bit Address Space

---

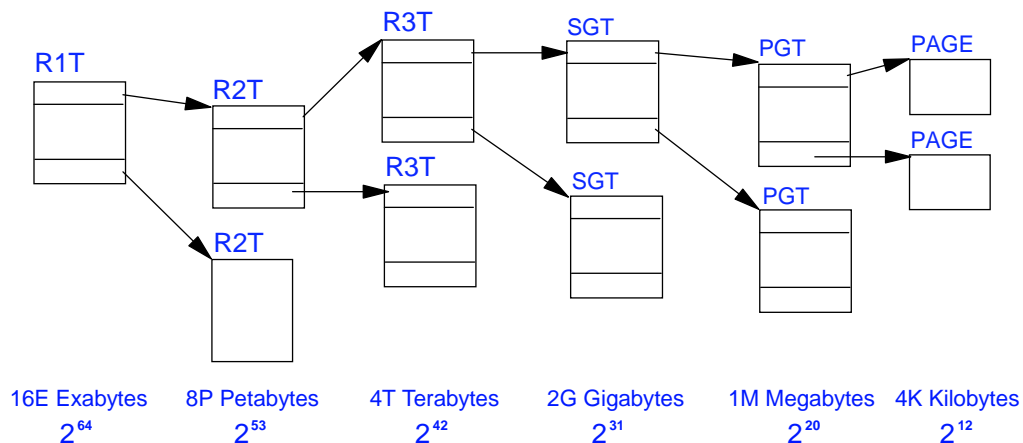


- ❑ Each address space is logically 16 exabytes
  - $2^{64}$  in size
- ❑ The area below 2 GB is mapped as before
  - Totally compatible with previous releases
- ❑ The area above 2 GB is for application data
  - No common areas, system areas, or programs
- ❑ An area is reserved for memory sharing
  - Available in a future release

## Address Space Memory Map



## Region, Segment, Page Tables



## Using Virtual above 2 GB with V1R2



- ❑ z/OS 1.2 sets a new bit in the CVT
  - CVTV64 - when on, indicates 64-bit virtual support is present
- ❑ New z/OS High Level Assembler
  - New z/Architecture instructions for manipulating 64-bit registers and addresses
- ❑ New assembler macro instructions to allocate and manipulate virtual storage above 2GB
- ❑ To reference storage above 2G, a program must switch into 64-bit addressing mode (AMODE 64)
- ❑ New macro to obtain/free storage - IARV64

## Virtual Storage Support Plan



- ❑ First Step z/OS V1R2
  - z/OS assembler with support for 64 bit addressing
  - z/OS system support for 64-bit data addressability within a single address space
  - z/OS assembler system service to manage virtual storage above the bar within a single address space
- ❑ Next Step AMODE(64) - z/OS V1R3
  - Binder, loader and content supervisor
  - AMODE 64 program execution below 2GB
- ❑ Next Step Shared Support - (z/OS V1R5)
  - z/OS system support for 64-bit data addressability between multiple address spaces
  - z/OS assembler system service to manage virtual storage above the bar between multiple address spaces

## Virtual Storage Support Plan

---



- ❑ Next Step AMODE(64) - z/OS V1R5
  - 64 bit support added to the binder in z/OS V1R5
  - rmode 64 toleration
  - Loading WSA above the bar

## Memory Objects

---



- ❑ z/OS virtual memory above 2GB is organized as
  - Memory objects
- ❑ Memory objects are a contiguous range of virtual addresses created by a program
  - Allocated as a number of 1 MB chunks of storage starting on a 1MB boundary
  - Some of the memory is usable virtual storage.
  - Remainder is not valid and is called the guard area (can be zero)
  - The extent of the usable virtual can be changed, with a compensatory change in the extent of the guard area
- ❑ Shared Memory Object
  - No Guard Area support

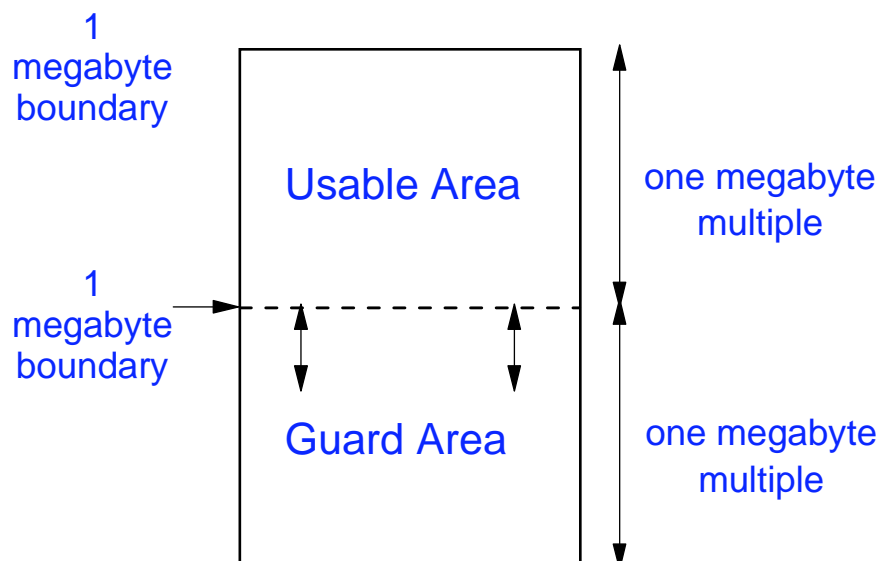


## Basic Memory Object Properties



- ❑ A memory object is allocated by a single request and can only be freed in its entirety - Partial freeing is not allowed
- ❑ It has a single storage protection key and fetch protection attribute - z/OS virtual memory above 2GB is organized as memory objects which programs create
- ❑ Private Memory Object
  - It is owned by a task
- ❑ Shared Memory Object
  - It is shared at the same address in every address space
  - It is owned by the system

## Memory Objects

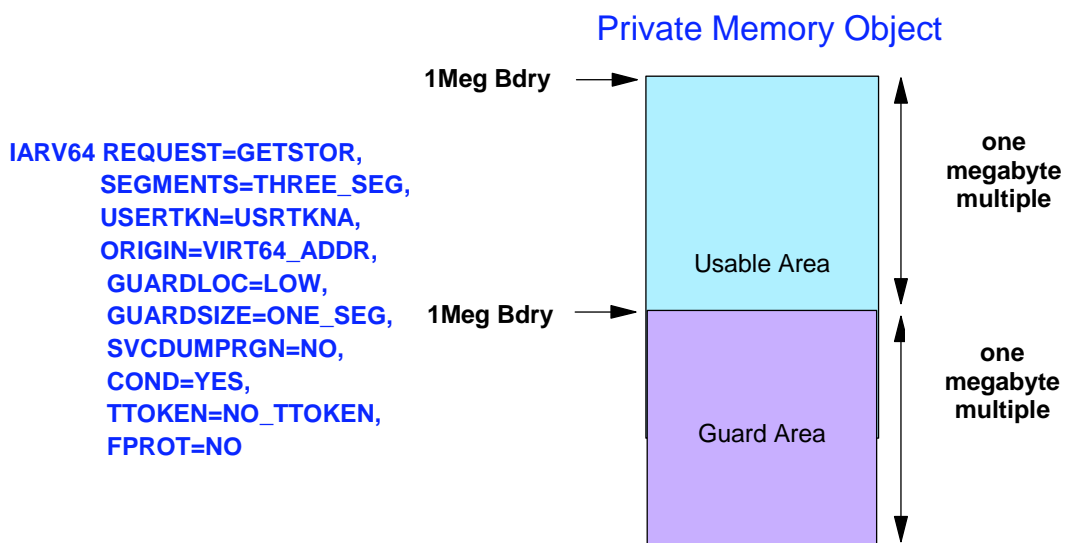


## Managing Memory Objects - IARV64

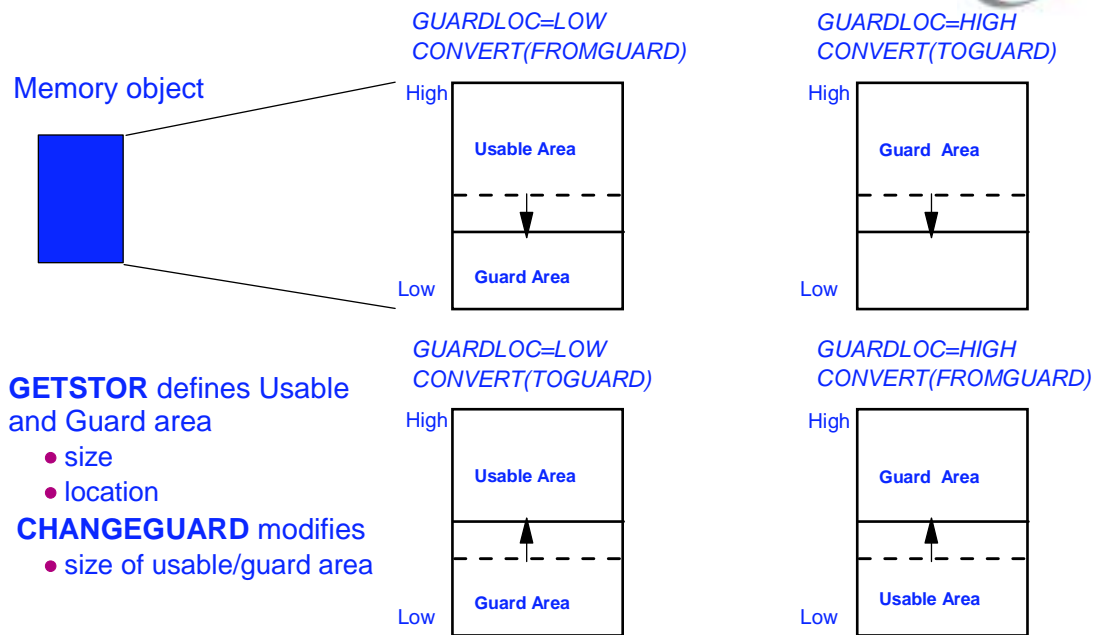


- ❑ **Getstor** - create a Memory Object (only for private memory objects)
- ❑ **Changeguard** - increase or decrease the amount of usable memory in a memory object (only for private memory objects)
- ❑ **Getshared** - Create a Shared Memory Object (only for shared memory objects)
- ❑ **Sharememobj** - Allows an address space to access Shared Memory Objects (only for shared memory objects)
- ❑ **Changeaccess** - Manages the type of access an address space has to the Shared Virtual Storage (only for shared memory objects)
- ❑ **Detach** - delete Memory Objects

## Guard Area Support



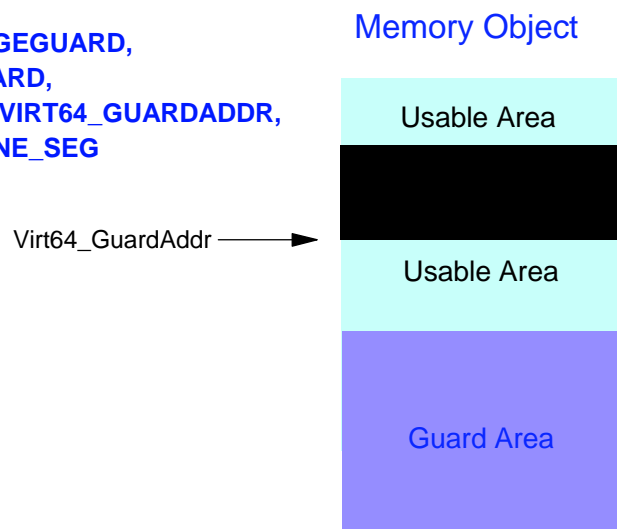
## Changing the Amount of Usable Memory



## Create a Guard Area in Usable Area

### Multiple Guard Area Support

IARV64 REQUEST=CHANGE GUARD,  
CONVERT=TO GUARD,  
CONVERTSTART=VIRT64\_GUARDADDR,  
CONVERTSIZE=ONE\_SEG



## Increase Size of Usable Area



### Multiple Guard Area Support

Memory Object

```
IARV64 REQUEST=CHANGE GUARD,  
CONVERT=FROM GUARD,  
CONVERTSTART=VIRT64_GUARDADDR,  
CONVERTSIZE=ONE_SEG
```



## Controlling Virtual usage



- ❑ An installation wants to limit the maximum physical memory resources (real and auxiliary) that can be committed by a job
- ❑ For virtual below 2GB, a limit on virtual storage usage provides (indirectly) a way to limit real and auxiliary storage use by a job
  - The REGION= keyword on JCL and can be overridden by the IEFUSI installation exit

## Virtual above the Bar

---



- ❑ No practical limit to the amount of virtual address range that an address space can request
- ❑ Provide a limit on the amount of usable virtual storage above 2GB that an address space can use at any one time
- ❑ The limit is 0 unless specified through either:
  - The new SMF MEMLIMIT parameter, or
  - The new MEMLIMIT keyword on JCL, and
  - Can be overridden by an IEFUSI exit

## Support for Virtual Above the Bar

---



- ❑ Why use Virtual Storage above the bar?
  - 64-bit virtual storage provides applications/middleware with:
  - Enhanced data caching capacity
  - Simplified memory management
  - Support for private storage above the bar was delivered in z/OS 1.2
- ❑ Z/OS 1.5 has the following enhancements for 64-Bit:
  - 64-Bit Shared Memory support
  - Multiple guard area support for private virtual storage above the bar

## Using Shared Virtual Storage



- ❑ New options on the IARV64 macro allow address spaces to share storage above the bar
- ❑ Shared Area size can be specified via the HVSHARE keyword in IEASYSxx, or system parms
  - HVSHARE=xxxxxxxxxxxG, or xxxxxxxxxxxT, or xxxxxxP
  - Default shared area starts at 2TB and ends at 512TB
  - Minimum size is zero, max size is 1 Exabyte
- ❑ Note: A shared memory object has no guard area

## Display Use of Shared Storage



- ❑ DISPLAY VIRTSTOR,HVSHARE or D VS,HVSHARE command
  - Displays the shared area range and how much 64-bit shared virtual storage has been allocated in the system

In z/Architecture mode

```
IAR019I hh.mm.ss DISPLAY VIRTSTOR
SOURCE = XX | (OP) | DEFAULT
TOTAL SHARED = nnnnnnnnnnnG
SHARED RANGE = nnnnG-nnnnnnnnnnnG
SHARED ALLOCATED = nnnnnnnnnnnnnM
```

## Using Virtual above 2GB



```
*  CHANGE TO AMODE 64
      SAM64
*  GET VIRTUAL STORAGE ABOVE THE BAR
      IARV64 REQUEST=GETSTOR,
          SEGMENTS=MO_SIZE,
          USERTKN=U_TOKEN,
          ORIGIN=V64_ADDR
          LTGR 15,15          GOT MEMORY OBJECT ?
          BC 8,WG            - YES, OK
          DC H'0'           - NO, INVESTIGATE
*  START WORK WITH DATA IN STORAGE ABOVE THE BAR
WG      WTO 'GOT V64',ROUTCDE=11
          LG 4,V64_ADDR      GET ADDRESS OF MEMORY OBJECT
          LHI 2,256*4        LOOP COUNTER, TOUCH ALL PAGES
TOUCH   MVC 0(L'DATA,4),DATA MOVE IN SOME DATA
          AHI 4,4096         TO NEXT PAGE
          BRCT 2,TOUCH       LOOP BACK AND TOUCH NEXT PAGE
*  DETACH VIRTUAL STORAGE ABOVE THE BAR
      IARV64 REQUEST=DETACH,
          MATCH=USERTOKEN,
          USERTKN=U_TOKEN,
          COND=YES
          LTGR 15,15        FREED MEMORY OBJECT ?
          BC 8,WD           - YES, OK
          DC H'0'          - NO, INVESTIGATE
WD      WTO 'DETACHED V64',ROUTCDE=11
```

## Data Area for Obtaining Storage



```
*  END EXIT LINKAGE
@DATA   DS      0D
MO_SIZE DC      FD'4'          MEMORY OBJECT IS 4 MB
U_TOKEN DC      FD'1'
DATA    DC      C'DATA ABOVE THE BAR'
```

## Addressing Mode Switching

---



- ❑ There are 3 new instructions which change addressing mode without branching:
  - Set Addressing Mode to 24-bit (SAM24)
  - Set Addressing Mode to 31-bit (SAM31)
  - Set Addressing Mode to 64-bit (SAM64)
- ❑ There are 2 instructions which change addressing mode and branch:
  - Branch and Save and Set Mode (BASSM)
  - Branch and Set Mode (BSM)

## Binder Support z/OS V1R5

---



- ❑ Support execution of programs above 'the bar' - the two gigabyte line
- ❑ Providing amode 64 support now and (perhaps) rmode 64 support later would force a double migration
- ❑ The binder will accept object modules with rmode 64 contents
  - This allows all 64-bit source and object code changes to be made in one step
  - Provide binder support for loading data portions (WSA) of an application above the bar



## WSA Above the Bar

---



- ❑ **rmode 64 for deferred load classes are visible to loader**
  - Compilers can generate a new class called C\_WSA64 marked as rmode 64
  - If program object is executed on a system with the appropriate loader support C\_WSA64 is loaded above the bar
  - A single program object may not contain both classes C\_WSA and C\_WSA64
- ❑ **Provides virtual storage constraint relief**
  - WSA is often very large

## Program Object During Execution

---



- ❑ **There is no way to mark an entry point as accepting all amodes, including amode 64**
- ❑ **Binder allows modules with mixed amode 64 and non-amode 64 code, however:**
  - Reference and definition must match. Mismatch: error message IEW2469E
  - The 'ANY' in AMODE(ANY) does NOT include amode 64

## Controlling Storage - MEMLIMIT



- ❑ Through JCL on the specific job with the new
  - MEMLIMIT JCL keyword
- ❑ MEMLIMIT specified on a JOB statement
  - //TC1 JOB MEMLIMIT=50G,REGION=0M
  - //TC2 JOB MEMLIMIT=125M,TIME=NOLIMIT
  - //TC3 JOB MEMLIMIT= 9T,MSGLEVEL=1
  - //TC4 JOB REGION=3M,MEMLIMIT=16384P
  - //TC5 JOB REGION=125M,MSGLEVEL=(1,1),
  - MEMLIMIT=NOLIMIT,MSGCLASS=A
- ❑ MEMLIMIT specified on an EXEC statement
  - //STEP1 EXEC PGM=TST6,MEMLIMIT=6400M
  - //STEP1 EXEC PGM=TST7,MEMLIMIT=3P...
  - //STEP1 EXEC MYPROC,MEMLIMIT=NOLIMIT...

## Controlling storage - SMFPRMxx



```
ACTIVE                /*ACTIVE SMF RECORDING*/
DSNAME ( SYS1.MANA,SYS1.MANB,SYS1.MANC) /* NEW D.S. ADDED 11/88 */
PROMPT(LIST)          /*PROMPT THE OPERATOR FOR OPTIONS*/
REC(PERM)             /*TYPE 17 PERM RECORDS ONLY*/
BUFNUM(4,9)           /* 4 - 4096 BUFFERS ALWAYS AND
                        ALLOW UP TO 9 BEFORE SUSPENDING
                        A USER FOR BUFFER SHORTAGE*/
MAXDORM(3000)         /* WRITE AN IDLE BUFFER AFTER 30 MIN*/
MEMLIMIT(24G)
STATUS(010000)        /* WRITE SMF STATS AFTER 1 HOUR*/
JWT(1439)             /* NO 522 ABENDS*/
SID(168A)             /* SYSTEM ID IS 168 A*/
LISTDSN              /* LIST DATA SET STATUS AT IPL*/
SYS(TYPE(0:255),EXITS(IEFACTRT,IEFUJV,IEFUSI,IEFU83,
IEFUJI,IEFUTL,IEFU29),NOINTERVAL,NODETAIL)
```

```
-----
MEMLIMIT(16384P)      /* This is the same as NOLIMIT */
```

```
MEMLIMIT(125T)
```

```
MEMLIMIT(4000P)
```

```
MEMLIMIT(0M)         /* Disallow storage >2G */
```

```
MEMLIMIT(00000M)     /* DEFAULT */
```

### Other examples of MEMLIMIT

## MEMLIMIT During the IPL



```
SYS(TYPE(0:255)) -- DEFAULT
LISTDSN -- DEFAULT
SID(4381) -- DEFAULT
STATUS(010000) -- DEFAULT
MAXDORM(3000) -- DEFAULT
DDCONS(YES) -- DEFAULT
LASTDS(MSG) -- DEFAULT
NOBUFFS(MSG) -- DEFAULT
SYNCVAL(00) -- DEFAULT
INTVAL(30) -- DEFAULT
DUMPABND(RETRY) -- DEFAULT
REC(PERM) -- DEFAULT
DSNAME(SYS1.MANY) -- DEFAULT
DSNAME(SYS1.MANX) -- DEFAULT
MEMLIMIT(NOLIMIT) -- PARMLIB
JWT(1439) -- PARMLIB
PROMPT(ALL) -- PARMLIB
NOACTIVE -- PARMLIB
*01 IEE357A REPLY WITH SMF VALUES OR U
00- r 1, MEMLIMIT(2G)
    IEE600I REPLY TO 01 IS; MEMLIMIT(2G)
*02 IEE357A REPLY WITH SMF VALUES OR U
```

## Reset the SMF Parameters



### SET SMF=M4

```
IEE252I MEMBER SMFPRMM4 FOUND IN
RSMID.PARMLIB
IEE536I SMF    VALUE M4 NOW IN EFFECT
D SMF,O
IEE967I 00.56.34 SMF PARAMETERS 379
    MEMBER = SMFPRMM4
    DSNAME(SYS1.MANY) -- DEFAULT
    DSNAME(SYS1.MANX) -- DEFAULT
    ACTIVE -- DEFAULT
MEMLIMIT(00003G) -- PARMLIB
JWT(2400) -- PARMLIB
PROMPT(ALL) -- PARMLIB
```

## Change MEMLIMIT Value

---



**setsmf memlimit(120t)**

IEE712I SETSMF PROCESSING COMPLETE

d smf,o

IEE967I 01.29.56 SMF PARAMETERS

MEMBER = SMFPRMBR

MEMLIMIT(00120T) -- REPLY

**PROMPT(ALL) -- PARMLIB**

DDCONS(YES) -- DEFAULT

LASTDS(MSG) -- DEFAULT

NOBUFFS(MSG) -- DEFAULT

SYNCVAL(00) -- DEFAULT

INTVAL(30) -- DEFAULT

DUMPABND(RETRY) -- DEFAULT