



The Trainer's Friend
I N C O R P O R A T E D

The Future of Mainframes Is Now

- ◆ **z/Architecture**
- ◆ **z/OS**
- ◆ **z Machines Hardware**
- ◆ **Numbers and Numeric Terms**
- ◆ **The Road to z/OS**
- ◆ **z/OS.e**
- ◆ **z/OS Futures**
- ◆ **Language Environment**
- ◆ **Current Compilers**
- ◆ **UNIX System Services**

by Steve Comstock

The Trainer's Friend, Inc.
<http://www.trainersfriend.com>
800-993-8716
steve@trainersfriend.com

z/Architecture

- ❑ **The IBM 64-bit mainframe has been named "z/Architecture" to contrast it to earlier mainframe hardware architectures**
 - ◆ **S/360**
 - ◆ **S/370**
 - ◆ **370-XA**
 - ◆ **ESA/370**
 - ◆ **ESA/390**

- ❑ **Although there is a clear continuity, z/Architecture also brings significant changes...**
 - ◆ **64-bit General Purpose Registers - so 64-bit integers and 64-bit addresses**
 - ◆ **64-bit Control Registers**
 - ◆ **128-bit PSW**
 - ◆ **Tri-modal addressing (24-bit, 31-bit, 64-bit)**
 - ◆ **Over 140 new instructions, including instructions to work with ASCII and UNICODE strings**

z/OS

- ❑ Although several operating systems can run on z/Architecture machines, z/OS is the premier, target OS

- ❑ z/OS is the successor to OS/390
 - ◆ The last release of OS/390 was V2R10, available 9/2000

 - ◆ The first release of z/OS was V1R1, available 3/2001

- ❑ z/OS can also run on G5/G6 and MP3000 series machines
 - ◆ But only in 31-bit or 24-bit mode

- ❑ Note these terms:
 - ◆ The Line - the 16MiB address limit of MVS

 - ◆ The Bar - the 2GiB limit of OS/390

- ❑ For some perspective, realize that 16EiB is...
 - ◆ 8 billion times 2GiB

 - ◆ 1 trillion times 16MiB

- ❑ The current release of z/OS is V1R4; V1R5 is scheduled for 1Q2004

z/Architecture, continued

- ❑ **z/Architecture is only available on the latest machines from IBM, in particular:**
 - ◆ **zSeries 900 models and 800 models (we say "z machines")**

- ❑ **The z machines can run in ESA/390 mode or z/Architecture mode**
 - ◆ **z/OS can only be run in z/Architecture mode on z machines**
 - ◆ **OS/390 can run on z machines, in ESA/390 mode or z/Architecture mode**

- ❑ **z/OS can run in ESA/390 mode on these machines, also:**
 - ◆ **9675 G5 and G6 models**
 - ◆ **Multiprise 3000 servers (also called MP3000)**

X Note that G5, G6 and MP3000 machines do not have 64-bit registers nor a 128-bit PSW, etc.

- ❑ **Although z/OS V1R1-V1R5 will run on all these machines, only z machines allow you to run z/OS in z/Architecture mode**
 - ◆ **At some as yet undisclosed point, running z/OS will require running on z machines or later hardware**

z Machines Hardware

□ On the z machines ...

◆ Each model includes either 12 or 20 processing units (PUs)

✗ Each PU can be assigned to any of these roles, dynamically:

➤ CP - Central Processor (Instructions)

➤ SAP - System Assist Processor (I/O)

➤ ICF - Internal Coupling Facility processor

➤ IFL - Integrated Facility for Linux processor

◆ The model number determines how many PUs are assigned as CPs and SAPs (never more than 16 CPs per model)

◆ All other PUs may be assigned as ICFs, IFLs, additional SAPs, or inactive

✗ There are some additional model-specific restrictions

✗ One PU must always be inactive, available as a spare

◆ Each PU is a dual processor

✗ Both processors execute the same instruction stream in parallel and the results are compared; this is how error checking is done for PUs

◆ Models with 12 PUs can have from 5GiB to 32GiB of real storage; models with 20 PUs can have from 10GiB to 64GiB of real storage

z Machines Hardware, continued

- ❑ **z systems may be clustered in a Parallel Sysplex**
 - ◆ **Up to 32 systems per sysplex**
 - ◆ **Since each system can have up to 16 CPs, that means a single sysplex can contain 16 x 32 or 512 processors**

- ❑ **Of course, sysplexes can be linked to other sysplexes in a network!**

Numbers and Numeric Terms

- The numbers possible with 64-bit integers and addresses are large, so we find we may need to refresh / introduce the proper numeric terms, at least as far as American English goes

- ◆ For counting ...

<u>For this many digits</u>	<u>We use the term</u>
9	units
99	tens
999	hundreds
9,999	thousands
9,999,999	millions
9,999,999,999	billions
9,999,999,999,999	trillions
9,999,999,999,999,999	quadrillions
9,999,999,999,999,999,999	quintillions

- The largest binary number in 64 bits is, in decimal:

- ◆ 0 - 18,446,744,073,709,551,615 if unsigned

- ◆ -9,223,372,036,854,775,808 to +9,223,372,036,854,775,807 if signed

Numbers and Numeric Terms, 2

- ❑ But, in terms of measurements (for example number of bits or bytes of memory or disk space) the language has gotten trickier

- ❑ First, we have historically used these terms
 - ◆ kilobit, kilobyte (kb, kB) - 1,024 bits or bytes
 - ◆ Megabit, Megabyte (Mb, MB) - 1,048,576 bits or bytes
 - ◆ Gigabit, Gigabyte (Gb, GB) - 1,073,741,824 bits or bytes
 - ◆ Terabit, Terabyte (Tb, TB) - 1,099,511,627,776 bits or bytes
 - ◆ Petabit, Petabyte (Pb, PB) - 1,125,899,906,842,624 bits or bytes
 - ◆ Exabit, Exabyte (Eb, EB) - 1,152,821,504,606,846,976 bits or bytes

- ❑ But these numbers are based on powers of 2, while engineers have historically worked with powers of 10 and use these terms:
 - ◆ kilobit, kilobyte (kb, kB) - 1,000 bits or bytes
 - ◆ Megabit, Megabyte (Mb, MB) - 1,000,000 bits or bytes
 - ◆ Gigabit, Gigabyte (Gb, GB) - 1,000,000,000 bits or bytes
 - ◆ Terabit, Terabyte (Tb, TB) - 1,000,000,000,000 bits or bytes
 - ◆ Petabit, Petabyte (Pb, PB) - 1,000,000,000,000,000 bits or bytes
 - ◆ Exabit, Exabyte (Eb, EB) - 1,000,000,000,000,000,000 bits or bytes

- ❑ These differences can cause real incompatibility problems in designs or standards, and thus manufacturing, costing, and pricing

Numbers and Numeric Terms, 3

- ❑ In 1998, the International Electrotechnical Commission (IEC) defined a standard set of terms and prefixes to be used to distinguish powers of two from powers of ten

- ❑ So these are the terms one should use when referencing numbers based on powers of two that describe quantities:
 - ◆ kibibit, kibibyte (Kib, KiB) - 1,024 bits or bytes

 - ◆ Mebibit, Mebibyte (Mib, MiB) - 1,048,576 bits or bytes

 - ◆ Gibibit, Gibibyte (Gib, GiB) - 1,073,741,824 bits or bytes

 - ◆ Tebibit, Tebibyte (Tib, TiB) - 1,099,511,627,776 bits or bytes

 - ◆ Pebibit, Pebibyte (Pib, PiB) - 1,125,899,906,842,624 bits or bytes

 - ◆ Exbibit, Exbibyte (Eib, EiB) - 1,152,821,504,606,846,976 bits or bytes

❑ The point of all this is that, for example, 65,537 bytes is 64KiB and 18,446,744,073,709,551,615 bytes is 16EiB, not 18EiB

- ❑ It's recommended that the second syllable ("bi") be pronounced as "bee"; the "bi" indicates "binary" - power of two

- ❑ It is not clear if these standards will be widely adopted or used outside of technical areas, and we may mix the new with the old while we go through a period of transition

The Road to z/OS

- ❑ **Modern IBM mainframe operating systems originated in the mid-1960's**

- ❑ **The mainframe operating systems on S/360 class machines that led to z/OS have been ...**
 - ◆ **OS/360 - Operating System/360, which had three major variations**
 - ✗ PCP - Primary Control Program; no longer around

 - ✗ MFT - Multiprogramming with a Fixed number of Tasks; no longer around

 - ✗ MVT - Multiprogramming with a Variable number of Tasks (the base for what is to become z/OS)

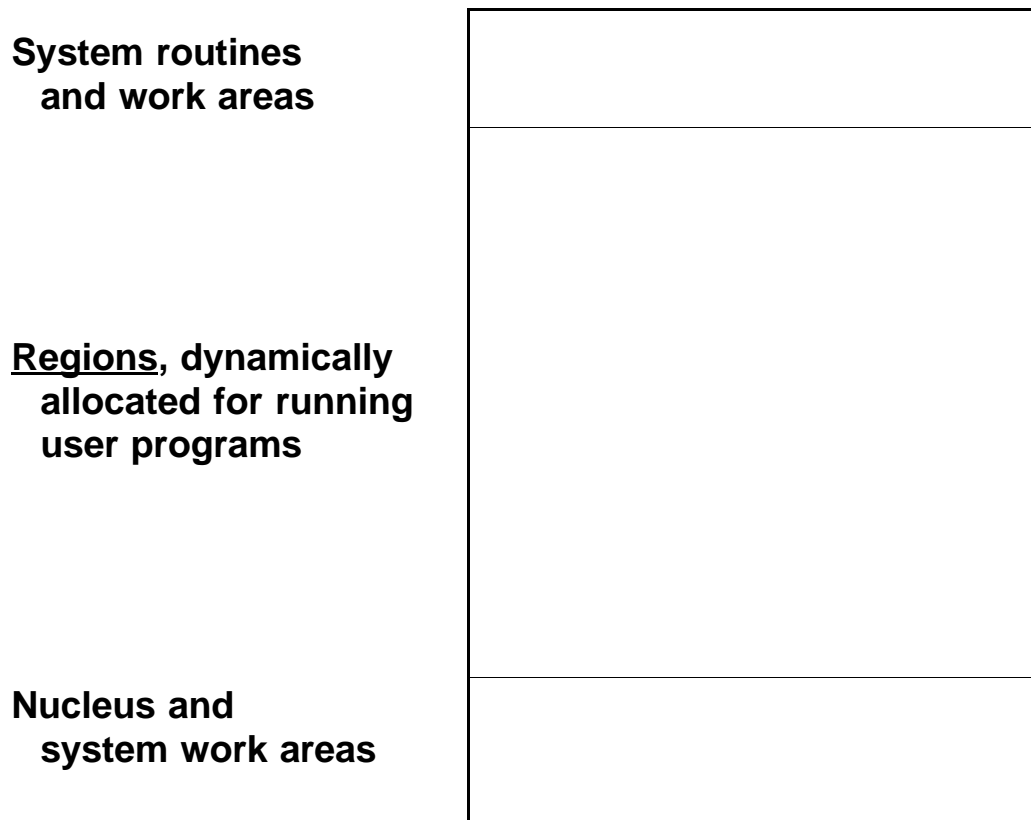
- ❑ **The OS/360 systems were all real-storage systems**
 - ◆ **No concepts of virtual storage, paging, swapping, and so on**

 - ◆ **The real storage you had was what you worked with**

 - ◆ **Mainframes running these systems often only had 24KiB of real memory; a machine with a megabyte of storage was a rarity**

The Road to z/OS, 2

- ❑ MVT organized [real] storage as having (from top to bottom: that is, high address to low address):



- ❑ The maximum, theoretical size of memory was 16MiB, but few customers bought more than 4 MiB, and it wasn't uncommon in this timeframe for a mainframe to have 128KiB
 - ◆ The memory addressing scheme used 24 bits (3 bytes) to specify a memory address, so 16,777,215 was the largest theoretical address

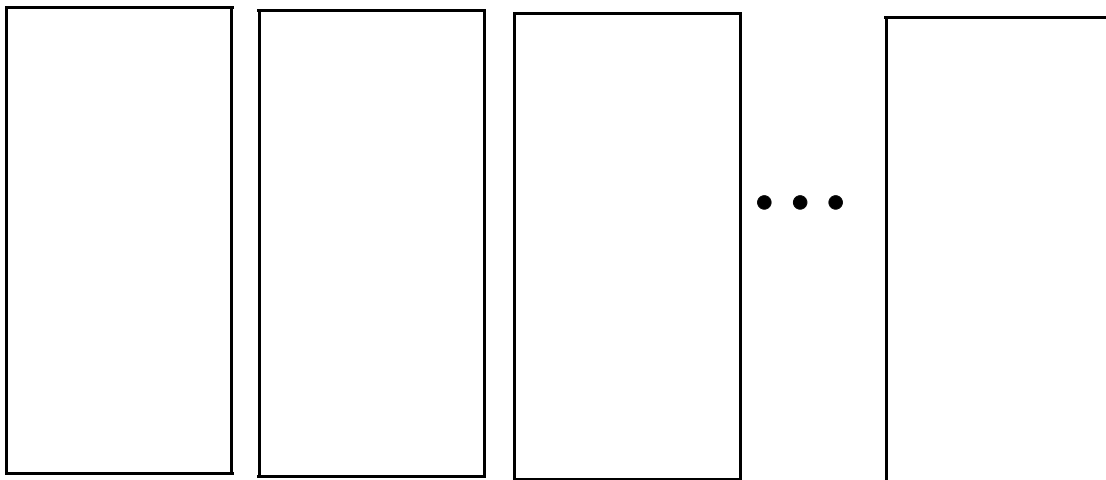
The Road to z/OS, 3

- ❑ In the 1970's, it became apparent there was a need to increase storage, and the first approach introduced Virtual Storage
 - ◆ Virtual Storage uses real storage, disk as backing storage, in combination with hardware detection of address faults to present the user with the illusion of having more memory than is really present
 - ◆ The user cannot distinguish between real storage and virtual storage

- ❑ At this time, MVT became OS/VS2
 - ◆ OS/VS1 was the old MFT with virtual storage; you could specify the total size of virtual storage up to the 16MiB limit
 - ◆ OS/VS2 later became called "SVS" for Single Virtual Storage
 - × SVS used a single virtual storage of the maximum size (16MiB), but the logical organization remained as it was for MVT

The Road to z/OS, 4

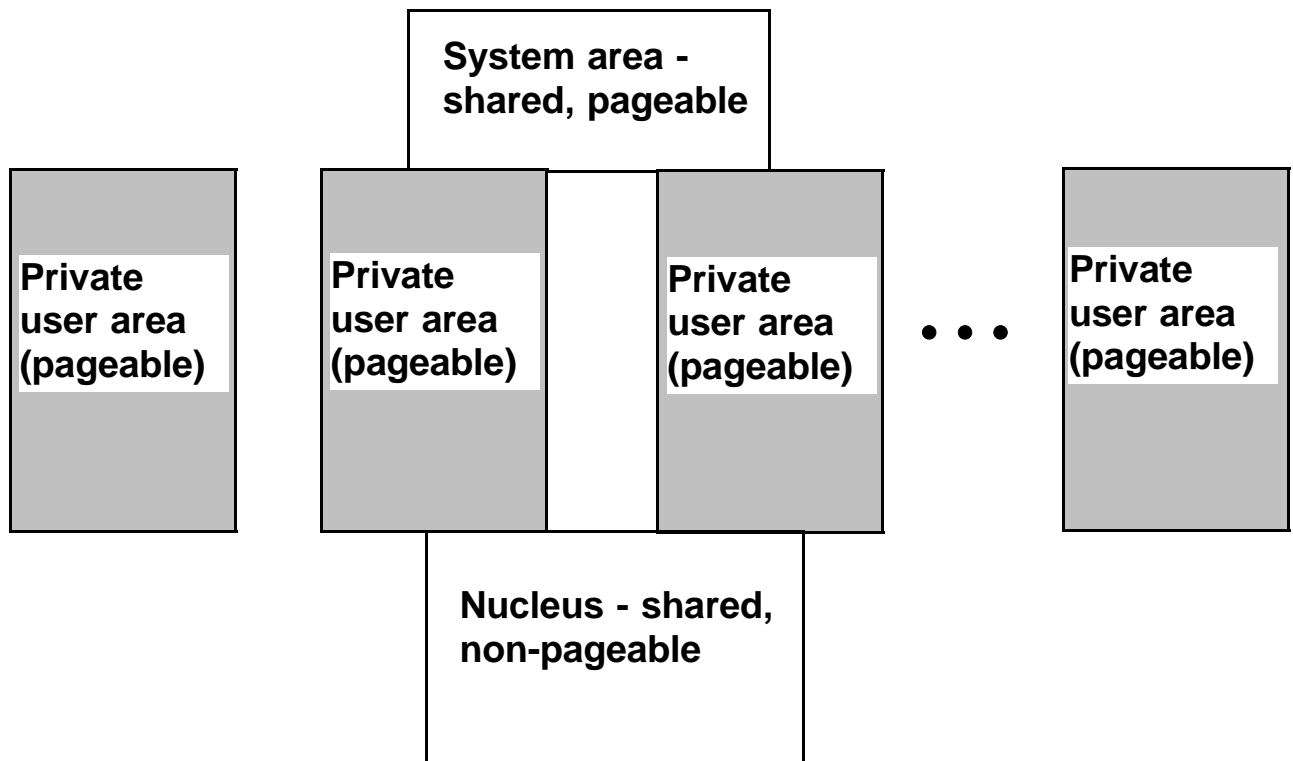
- ❑ In the mid-1970's, IBM started to phase out SVS in favor of the newest operating system: MVS - Multiple Virtual Storages
- ❑ In this design, each user is given their own virtual storage of the maximum size (still, at this point, 16MiB):



- ❑ These virtual storages are called address spaces
 - ◆ Theoretically, there can be up to 65,537 address spaces, although the actual limit is considerably smaller
 - ✗ There are some number of system address spaces
 - ✗ Every batch initiator runs in its own address space
 - ✗ Every TSO user gets his or her own address space

The Road to z/OS, 5

- ❑ Because each user has their own address space, each address space needs to have a copy of the operating system
 - ◆ Since this is the same for each user, the addressing scheme is set up to have only one, shared, copy of the nucleus area and the system area
 - ◆ So the unique parts of an MVS system look, conceptually, like this:



- ❑ Again, addresses stay at 24-bits so each address space is 16MiB in size

The Road to z/OS, 6

- ❑ In the 1980's, IBM bit the bullet and extended the address space from 24 bits to 31 bits
 - ◆ 31 bits instead of 32 bits for a variety of reasons, which provides for a 2 GiB address space (2,147,483,648 bytes)
 - ◆ This was called extended architecture, abbreviated XA, so the operating system was called MVS/XA
 - ◆ This provides for 128 times the previous amount of virtual storage for programs to use
 - ◆ In addition to providing a larger address space, IBM re-arranged the layout
 - ✗ Sections of code that relied on 24-bit addresses had to remain under the 16 MiB limit (which has come to be called The Line)
 - ✗ So IBM moved as much of their code as possible above The Line (there will always have to be some code below The Line, to support older code)
 - ◆ So, the layout of an address space in MVS/XA looks like the diagram on the following page ...

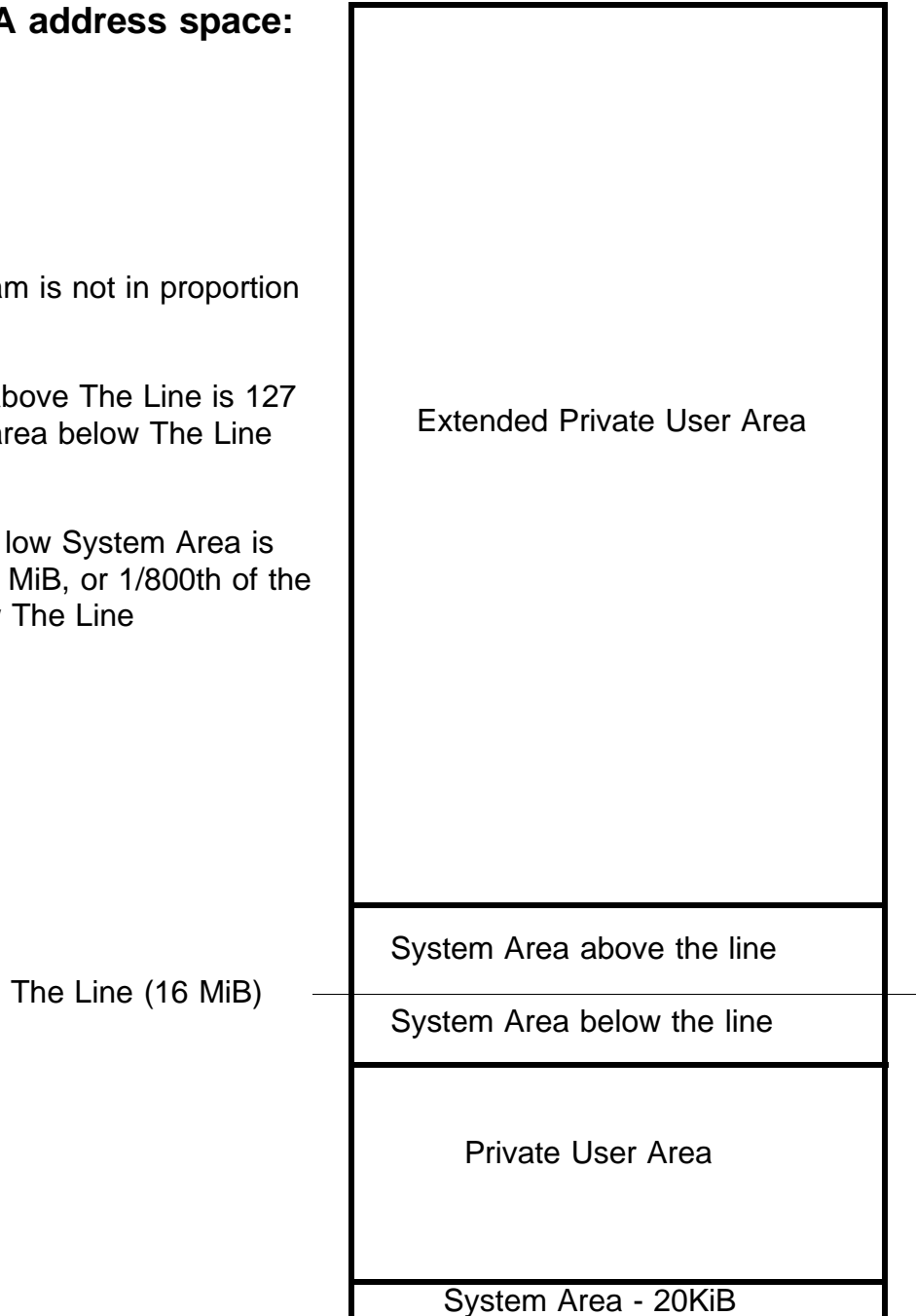
The Road to z/OS, 7

❑ MVS/XA address space:

This diagram is not in proportion

The area above The Line is 127 times the area below The Line

The 20KiB low System Area is 1/50th of 1 MiB, or 1/800th of the area below The Line



- ❑ **The goal is to put very little code and data below the line and to have the vast majority of programs and data reside above the line**

The Road to z/OS, 8

- ❑ **Other variations of MVS came along, to support enhanced hardware instructions and features, but the essence of address spaces and data spaces did not change**

- ❑ **MVS/ESA was the last release of MVS as an independent product**

- ❑ **The next step in the evolution was OS/390 (Operating System/390) which is really a packaging of components**

The Road to z/OS, 9

❑ OS/390 contains

- ◆ MVS/ESA code plus a number of program products as a single package
- ◆ Intent is to update every six months, keeping all the products in synch, thus simplifying the process of installing and upgrading systems and products (1st release was 3/96)
- ◆ Products included with OS/390 (among others):
 - x SMP/E (for maintenance uses)
 - x TSO/E
 - x ISPF
 - x High Level Assembler
 - x BookManager
 - x DFSMSdfp
 - x Language Environment (LE)
 - x TCP/IP
 - x DCE (Distributed Computing Environment support)
 - x OpenEdition / POSIX support (UNIX under MVS!)
- ◆ In addition, other optional products are available to be shipped in an OS/390 order, for an extra charge

The Road to z/OS, 10

- ❑ In 2001, IBM made available new hardware, the z900 series, that supported 64-bit addresses
 - ◆ So now address spaces can be as large as 64-bit addresses allow

- ❑ A new operating system, z/OS, was announced to support the new hardware

- ❑ But z/OS is based on OS/390 - there is a solid continuity here
 - ◆ Old code can still run under z/OS, even code compiled and linked under MVT over 35 years earlier

 - ◆ To use new features, of course, you need to rewrite, recompile, and rebind

 - ◆ There are still address spaces, just larger and organized slightly differently

 - ◆ There is still an MVS component, a TSO component, and so on

- ❑ The last release of OS/390 was V2R10, available September 2000, the first release of z/OS was available March 2001
 - ◆ The announced intent is to slow the release schedule to once a year after V1R6 is available

The Road to z/OS, 11

- ❑ **Some of the issues around establishing a 64-bit address space are resolved this way**

- ◆ **The size of the low System Area is increased to 24KiB**

- ◆ **The previous limit of 2 GiB is now called The Bar**

- ✗ **So programs or data can reside**

- **Below The Line**

- **Above The Line but below The Bar**

- **Above The Bar (data only, currently, no programs)**

- ❑ **A 64-bit address space allows for a maximum address of 18,446,744,073,709,551,615**

- ◆ **That is, a 64-bit address space is 8,589,934,592 times the size of a 31-bit, 2 GiB address space**

- ❑ **Note that data spaces are no longer needed, but code that uses data spaces continues to run just fine**

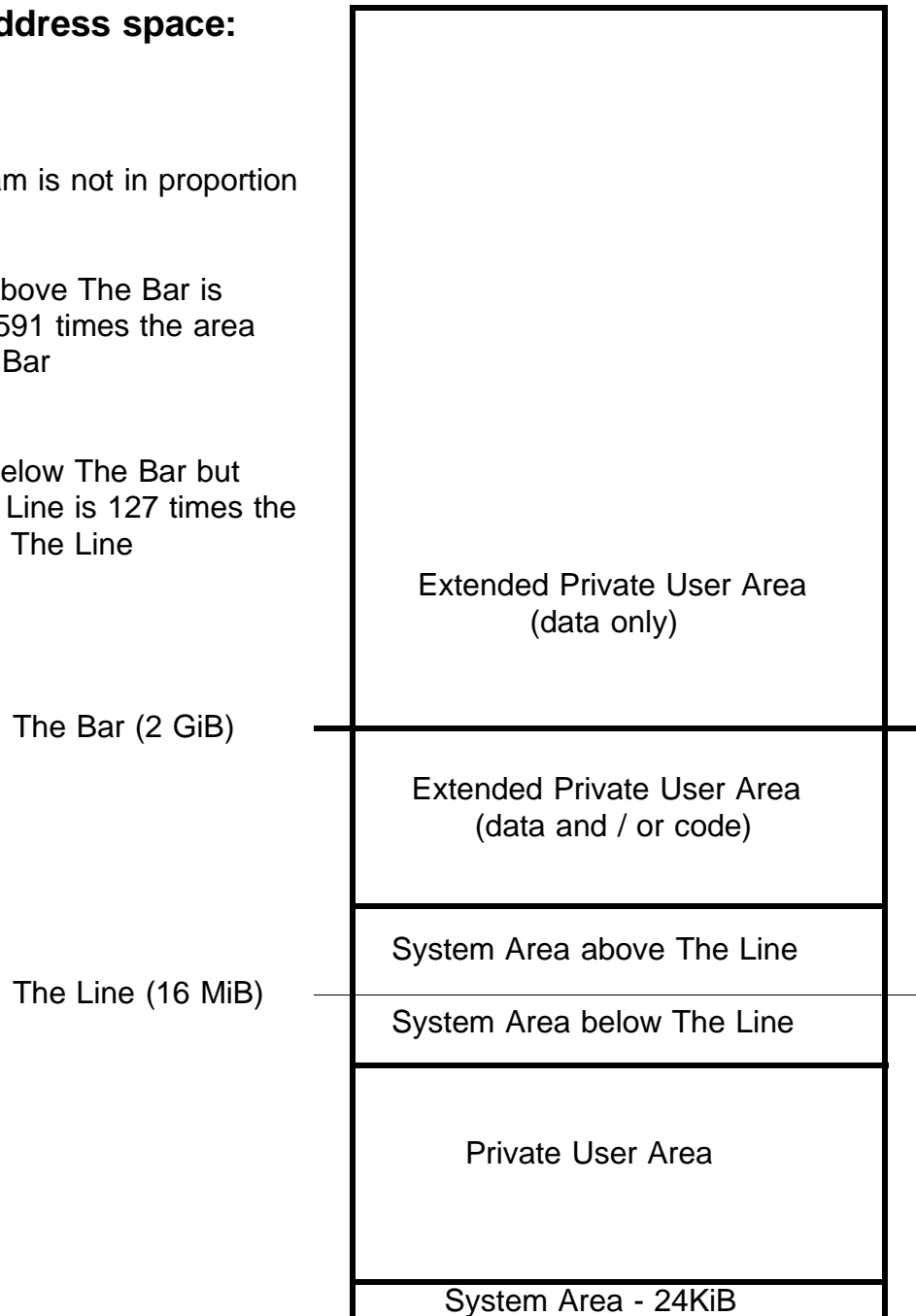
The Road to z/OS, 12

❑ z/OS address space:

This diagram is not in proportion

The area above The Bar is
8,589,934,591 times the area
below The Bar

The area below The Bar but
above The Line is 127 times the
area below The Line



z/OS.e

- ❑ **In February, 2002, the 800 models of the z/Series machines were announced**
 - ◆ **Generally smaller models at smaller prices**

- ❑ **Also announced was new version of z/OS: z/OS.e (for e-Commerce)**
 - ◆ **Must execute in 64-bit, z/Architecture mode on 800 model**
 - ◆ **Runs C, C++, Java, DB2**
 - ◆ **Will not run CICS, IMS, COBOL, or FORTRAN apps**
 - ◆ **Cannot use COBOL, PL/I, or FORTRAN compilers**
 - ◆ **Max 8 concurrent TSO sessions**
 - ◆ **Intended to run UNIX type workloads, priced to be competitive with traditional UNIX systems**

- ❑ **Just to clarify, however, the z/800 models can run any of these:**
 - ◆ **OS/390**
 - ◆ **VM/ESA**
 - ◆ **VSE**
 - ◆ **z/VM**
 - ◆ **z/OS**
 - ◆ **z/OS.e**

z/OS Futures

- ❑ **z/OS and z/OS.e have the same code base, it's just that z/OS.e has certain components disabled**
 - ◆ **These operating systems will be extended and enhanced in parallel**

- ❑ **These two operating systems are the ones based on MVS technology that IBM will continue to enhance and support**
 - ◆ **OS/390's days are numbered**

Language Environment

- ❑ **A major component of OS/390 carried forward into z/OS is called Language Environment (or just LE)**
 - ◆ **Supports a common runtime for COBOL, PL/I, C, C++, FORTRAN, and more**
 - ✗ Providing simpler, cleaner cross-language communications
 - ✗ In fact, COBOL programs can call most C functions directly, without having to have a C subroutine
 - ◆ **Provides a suite of callable services**
 - ✗ Most useful for COBOL shops:
 - Dynamic storage allocation / use / freeing
 - Exception handling
- ❑ **The base for all currently supported compilers**
 - ◆ **COBOL for OS/390 & VM**
 - ◆ **IBM Enterprise COBOL for z/OS and OS/390**
 - ◆ **Visual Age PL/I for OS/390**
 - ◆ **IBM Enterprise PL/I for z/OS and OS/390**
 - ◆ **C, C++ compilers - updated with each version of z/OS**

Current Compilers

- **Features of currently supported compilers**
 - ◆ **Modern language constructs (e.g.: scope delimiters, case insensitive code)**
 - ◆ **Larger limits (e.g.: COBOL table size now 16MiB, not 128KiB)**
 - ◆ **Access to LE routines - direct subroutine calls**
 - ◆ **Access to C library routines - direct subroutine calls**
 - ◆ **Access to UNIX routines - direct subroutine calls**
 - ◆ **Access to UNIX files (HFS - Hierarchical File System)**
 - × Standard language verbs or UNIX services calls
 - ◆ **Enhanced inter-language communication**
 - ◆ **Support for DLL's (Dynamic Link Libraries) within and across language boundaries**
 - ◆ **Interoperability with Java objects, methods, classes**
 - ◆ **Co-processors for CICS and DB2 embedded SQL**
 - ◆ **Support for XML processing**

UNIX System Services

- ❑ **Since at least 1996, IBM has included support for a UNIX-like environment**
 - ◆ **Originally called Open Edition (*e.g.*: MVS/OE in some places)**
 - ◆ **Currently called UNIX System Services (USS) and officially UNIX branded**
 - ◆ **Originally optional, currently required to be installed and functioning (whether your apps use it or not)**

UNIX System Services Major Components

- ❑ **HFS - Hierarchical File System**
 - ◆ **Allocate an MVS data set with HFS attribute**
 - ◆ **Inside, managed like UNIX files: root, directories, sub-directories, ... , files: paths and files**
- ❑ **UNIX kernel address space**
 - ◆ **Provides traditional UNIX kernel services; callable from UNIX and non-UNIX users**
- ❑ **UNIX shells (two provided, others can be ported)**
 - ◆ **Accessible from TSO: OMVS command**
 - ◆ **Accessible using rlogin and telnet**
- ❑ **HTTP server included, no extra charge**
 - ◆ **Run Intranet, Internet, Web applications**
 - ◆ **Email support - send email from the mainframe to the Internet**
 - ◆ **Mobile support - have your mainframe page a support person if a job abends**
- ❑ **JVM support (extra charge item)**

Which Means...

- ❑ You mainframe can run all your mission critical, day to day, "bread-and-butter" applications
 - ◆ Securely - you just don't crack a mainframe
 - ◆ At higher and higher speeds (because of faster and faster hardware and continuing software improvements)

- ❑ PLUS your UNIX / Intranet / Internet / Web applications
 - ◆ All on a single box (or complex of boxes)

Now That's a Legacy!