*e*server zSeries ("Freeway") Machines: an Assembler Programmer's View

SHARE 96 (Feb. 2001), Session 8172

John R. Ehrman
ehrman@us.ibm.com or ehrman@vnet.ibm.com

IBM Silicon Valley (nee Santa Teresa) Laboratory
555 Bailey Avenue
San Jose, CA 95141

February 28, 2001

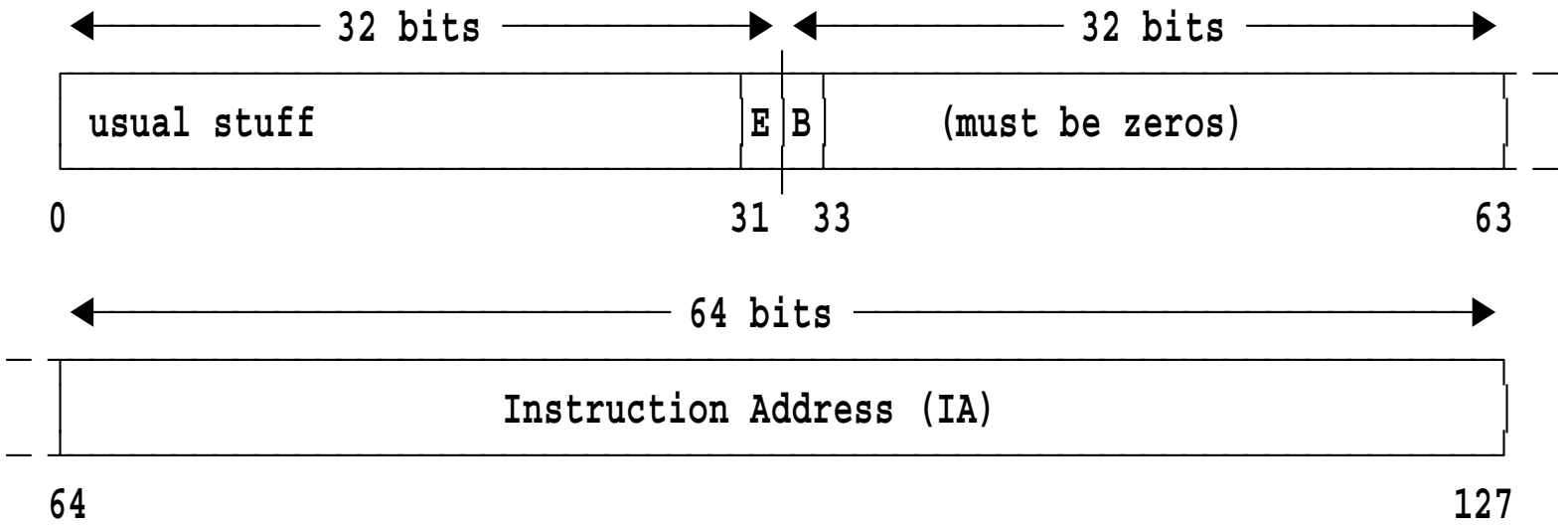
- IBM ***e***server zSeries z/Architecture enhancements
  - Dual-mode architecture
  - 64-bit addressing, registers, and operands
  - New instructions (many!) and instruction types (many!)
    - Compatibility with and extensions to ESA/390
- Addressing modes
  - Modal instructions
- ***e***server zSeries support features in HLASM R4
  - New instructions, constants, statements, and options

“Freeway” Architecture

- Architectures: ESA/390 vs. z/Arch
  - PSW
  - Addressing
  - Registers

- z/Architecture: superset of ESA/390
  - 64-bit General Purpose Registers, Control Registers
  - 64-bit addressing for real and virtual addresses
  - All CPUs run uniformly in ESA/390 mode or z/Arch mode
    - System starts initially in ESA/390 mode, can switch to z/Arch
  - Low-address (Prefix) area completely reorganized, expanded to 8K bytes
    - Old/New 16-byte PSWs have new addresses (e.g. SVC Old PSW at `X'140'`)
    - `X'01'` bit at 163 (`X'A3'`) set if in z/Arch mode (on z/OS and z/VM)
  - 64-bit I/O allows data addresses above 2GB "bar"
- Distinguish **architecture** and **addressing** modes:
  - ESA/390 architecture mode: 24/31-bit addressing modes, 32-bit address generation, 8-byte PSW
  - z/Arch architecture mode: 24/31/64-bit addressing modes, 64-bit address generation, 16-byte PSW
- Don't expect old programs to run succesfully in 64-bit addressing mode
  - They might, if bits 0-32 of all GRs are zero!
  - Old ops (and new ESA/390 ops) never change bits 0-31 of a GR (see slide 22)

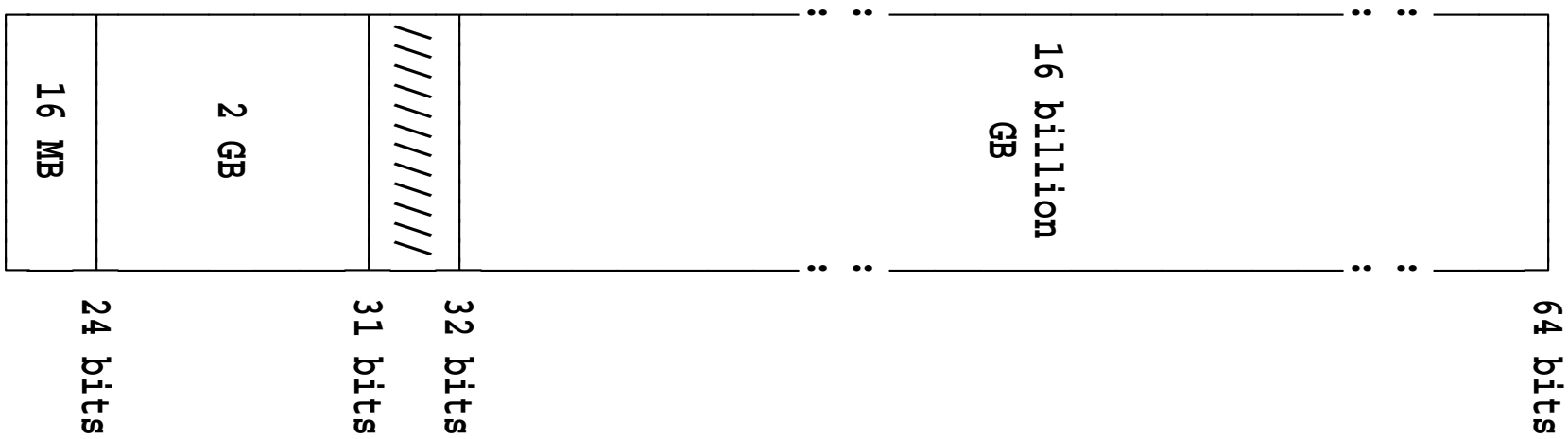
- PSW always 16 bytes long (128 bits)



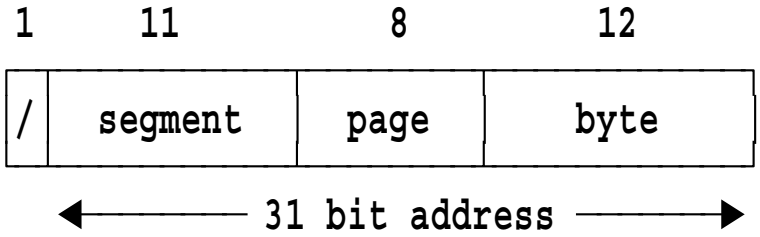
  - LPSWE loads a 16-byte PSW
  - LPSW loads an 8-byte PSW, extends it to the new 16-byte format
  - Bit 127 must be zero
- PSW bits (**E**xtended,**B**asic) indicate addressing mode:

EB	**Meaning**
00	24-bit (**B**asic) addressing mode
01	31-bit (**B**asic) addressing mode
11	64-bit (**E**xtended) addressing mode

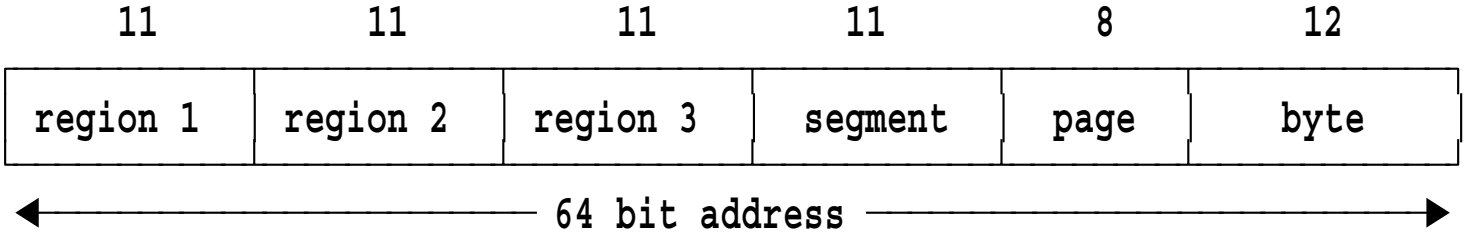
- All 64 bits participate in the address
  - 2^{64} bytes = 18 billion billion = 16 Exabytes per address space
  - That is, 8 billion 2GB spaces' worth!
- With data spaces, z/Arch is capable of 75-bit addressing
  - A total of 3.7×10^{22} bytes
  - There's really no way to picture this...



- 31-bit addressing uses the familiar scheme for virtual addresses:

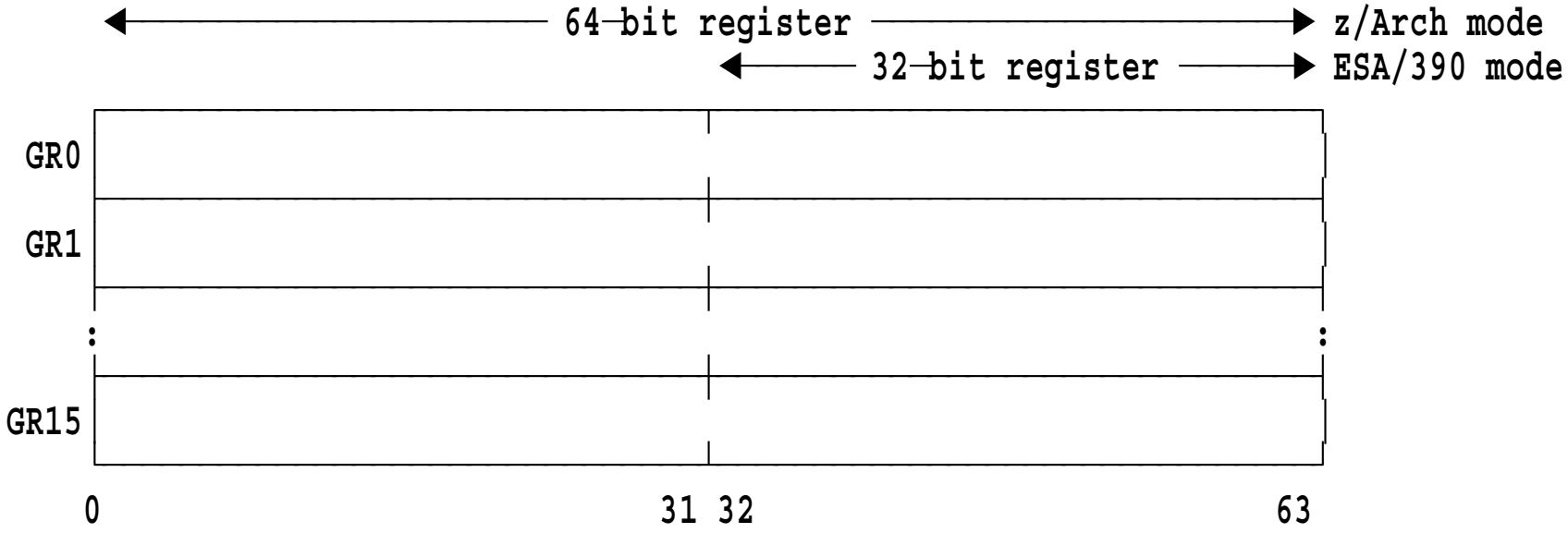


- In z/Arch 64-bit virtual addressing:



- Number of translations depends on size of address space
  - Segment and page tables always present, always used
  - Region tables present and used only if necessary

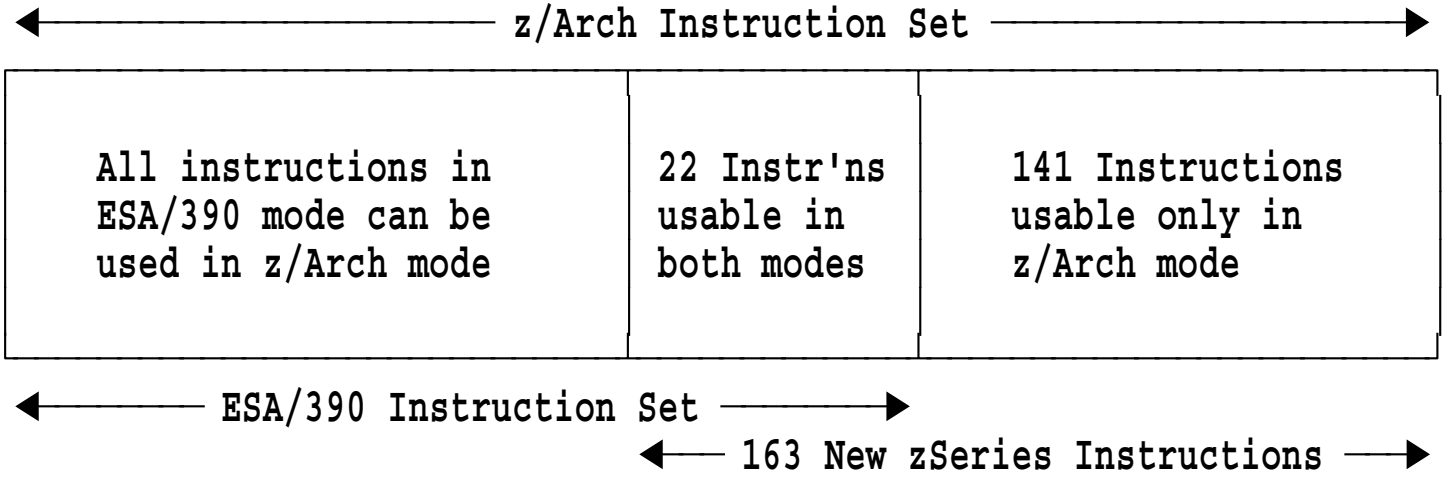
- General registers are 32 or 64 bits long, depending on the architecture
  - **In z/Arch mode, GPRs are always 64 bits long**
  - Old ESA/390 instructions use only the right halves (left halves are “invisible”)
- You can use 64-bit registers ***without*** being in 64-bit addressing mode!



- 64-bit registers in z/Arch mode numbered 0-63
  - Right half (“old 32-bit part”) now numbered 32-63, or {0-31} for ESA/390 mode
- Access Registers still 32 bits; Control Registers now 64 bits

zSeries Instruction Set

- Many new instructions
  - Some work in ESA/390 mode, some only in z/Arch mode



- Many “old” instructions don't affect bits 0-31 of the 64-bit register
- Instructions usable in both addressing modes:
  - Non-modal: no dependence on addressing mode (e.g. 64-bit arithmetic)
  - Modal: behavior depends on addressing mode (e.g. BAL) (more at slide 26)
- Some new instructions mix 64-bit and existing data types
- No instruction requires being in 64-bit addressing mode
- No vector facility or vector instructions

- New instructions and their architecture modes:

Type	Number	z/Arch-Only	ESA/390 and z/Arch
General	140	119	21
Floating Point	12	12	
Control	11	10	1
Total	163	141	22

- Instructions for both architecture modes summarized on slide 22
- New floating-point instructions support 64-bit integer operands
  - Details at slide 20
- New control instructions (more at slide 21):
  - One z/Arch-only instruction is non-privileged (EREGG)
  - One ESA/390-or-z/Arch instruction is privileged (STFL)

- Instructions for 64-bit integers similar to existing ops
- Mnemonics for 64-bit operations end in (or contain) **G**

```
AG    2,XYZ    Adds 64-bit operand at XYZ to 64-bit register 2
AGR   2,7      Adds 64-bit operand in R7  to 64-bit register 2
```

- Mnemonics for ops that mix 32- and 64-bit operands to give a 64-bit result have **GF** or **GH** (and in a few cases, **GT** or **GC**)
  - **F** = "Fullword," **H** = "Halfword," **T** = "Thirty-one bits," **C** = "Character"
  - 32-bit operand sign-extended or not, per arithmetic or logical operation

```
AGF   2,PQR    Adds signed  32-bit operand at PQR to 64-bit operand in R2
AGFR  2,7      Adds signed  32-bit operand in R7  to 64-bit operand in R2
AGLFR 2,7      Adds logical 32-bit operand in R7  to 64-bit operand in R2
```

- RR Load, Logical, Add, Subtract, Compare (and Branch on Count)

ESA/390 Ops	New 64-bit Ops	New Mixed-Length Ops
LR, LTR	LGR, LTGR	LGFR, LTGFR
LPR, LNR, LCR	LPGR, LNGR, LCGR	LPGFR, LNGFR, LCGFR
AR, SR	AGR, SGR	AGFR, SGFR
ALR, SLR	ALGR, SLGR	ALGFR, SLGFR
CR, CLR	CGR, CLGR	CGFR, CLGFR
NR, OR, XR	NGR, OGR, XGR	
BCTR	BCTGR	

- LCGFR, LPGFR complement -2^{31} without overflow

- RX Load, Logical, Add, Subtract, Compare (and a few others)

ESA/390 Ops	New 64-bit Ops	New Mixed-Length Ops
L, LH, ST	LG, STG	LGF, LGH
A, S, C	AG, SG, CG	AGF, SGF, CGF
AL, SL, CL	ALG, SLG, CLG	ALGF, SLGF, CLGF
N, O, X	NG, OG, XG	
MS, MSR	MSG, MSGR	MSGF, MSGFR
BCT	BCTG	
CVD, CVB	CVDG, CVBG	

- One new instruction for mixed 16/64 operations: LGH
  - No mixed 16/64 analogs of other halfword ops (AH, CH, MH, SH)
  - STH still stores rightmost 16 bits (32- or 64-bit register)
- MSG, MSGF products are 64 bits; MSGF multiplier is 32 bits

- RS Load, Shift, etc.

ESA/390 Ops	New 64-bit Ops	Comments
BXH, BXLE	BXHG, BXLEG	
SRL, SLL, SRA, SLA	SRLG, SLLG, SRAG, SLAG, RLL*, RLLG	Source and result registers may be different!
LM, STM	LMG, STMG LMD, LMH, STMH	Load/store register halves at different locations
CLM, ICM, STCM	CLMH, ICMH, STCMH	Use high-order 4 bytes

- Compare and Swap

ESA/390 Ops	New 64-bit Ops	Comments
CS, CDS	CSG, CDSG	CDSG operands must be quad-aligned (see slide 23)

* Instruction also available in ESA/390 mode (see slide 22)

ESA/390 Ops	z/Arch Ops	Comments
TMH, TML	TMHH, TMHL	TMLH=TMH, TMLL=TML for symmetry
BRXH, BRXLE	BRXHG, BRXLG	Relative indexed branch
BRCT	BRCTG	Relative branch on count
LHI, AHI, MHI, CHI	LGHI, AGHI, MGHI, CGHI	16-bit immediate operand, 64-bit result
BRC	BRCL*	Long relative conditional branch
BRAS	BRASL*	Long relative branch and save
	LARL*	Long relative load-address

- Most R-I instructions use a 16-bit (halfword) immediate operand
- BRCL, BRASL, LARL use a 32-bit (fullword) offset (in halfwords)
  - Long signed immediate field allows offsets of ± 4 GB
  - Target operand must be at an even address
  - LARL is like LA, but doesn't require a base register

* Instruction also available in ESA/390 mode (see slide 22)

- Logical Load Instructions

z/Arch Ops	Comments
LLGF, LLGFR	Load 32 bits, clear high-order 32
LLGT, LLGTR	Load 31 bits, clear high-order 33
LLGH	Load 16 bits, clear high-order 48
LLGC	Load 8 bits, clear high-order 56

- LLGT, LLGTR: lets you set a 64-bit address from a 31-bit address, without having to worry about the high-order bit
- LLGC: like “Insert Character,” but no need to clear the target register

z/Arch Ops	Comments
DSG, DSGR	Divide single (all operands 64 bits)
DSGF, DSGFR	Divide 64 by 32, 64-bit results
DL*, DLR*	Divide logical (64 by 32; unsigned operands)
DLG, DLGR	Divide logical (128 by 64; unsigned operands)
ML*, MLR*	64-bit logical product of unsigned 32-bit operands
MLG, MLGR	128-bit logical product of unsigned 64-bit operands
ALC*, ALCR*	Add logical with carry (32-bit operands)
ALCG, ALCGR	Add logical with carry (64-bit operands)
SLB*, SLBR*	Subtract logical with borrow (32-bit operands)
SLBG, SLBGR	Subtract logical with borrow (64-bit operands)

- ML, MLR product in low halves of two registers
- High-order bit of CC is the carry-borrow indicator

* Instruction also available in ESA/390 mode (see slide 22)

- These have no analogs in ESA/90
- All instructions have a 16-bit immediate field

z/Arch Ops	Comments
IIHH, IIHL, IILH, IILL	Inserts 16 bits into part of a 64-bit GR
NIHH, NIHL, NILH, NILL	AND-Immediate with part of a 64-bit GR
OIHH, OIHL, OILH, OILL	OR-Immediate with part of a 64-bit GR
LLIHH, LLIHL, LLILH, LLILL	Load part of a 64-bit GR, remaining bits set to zero

- Last two letters of mnemonic indicate which part of the GR:

  HH High Half's High Half (bits 0-15)

  HL High Half's Low Half (bits 16-31)

  LH Low Half's High Half (bits 32-47)

  LL Low Half's Low Half (bits 48-63)

- Byte-Reversing Load/Store

z/Arch Ops	Comments
LRVH*, STRVH*	Transmit 2 rightmost bytes in reverse order
LRV*, LRVR*, STRV*	Transmit 4 rightmost bytes in reverse order
LRVG, LRVGR, STRVG	Transmit 8 rightmost bytes in reverse order

- Rightmost bytes of register used for 2-byte and 4-byte ops
- For Load operations, remaining bytes in register are unchanged

- z/Arch is now an “Either-Endian” architecture!

* Instruction also available in ESA/390 mode (see slide 22)

- Conversions between 64-bit integers and Hex or IEEE-Binary float
- Extensions of the G5/G6 floating point enhancements
  - ESA/390 supports 32-bit integer operations

Operation	32-bit Integer Ops	New 64-bit Integer Ops
Integer to Hex	CEFR, CDFR, CXFR	CEGR, CDGR, CXGR
Integer to IEEE	CEFBR, CDFBR, CXFBR	CEGBR, CDGBR, CXGBR
Hex to Integer	CFER, CFDR, CFXR	CGER, CGDR, CGXR
IEEE to Integer	CFEBR, CFDBR, CFXBR	CGEBR, CGDBR, CGXBR

- Same rounding rules apply to 64-bit integers as to 32-bit

z/Arch Ops	Comments
LPQ, STPQ	Load/Store of quad-aligned operands
TAM*	Test addressing mode
SAM24*, SAM31*, SAM64	Set addressing mode (without branching)
EPSW	Extract high-order 64 bits of 128-bit PSW

- Control instructions:
  - Privileged, z/Arch or ESA/390:
    STFL (described at slide 22)
  - Privileged, z/Arch-only:
    ESEA, LCTLG, LPSWE, LRAG, LURAG, STCTG, STRAG, STURG, TRACG
  - Nonprivileged, z/Arch-only:
    EREGG (extract 64-bit registers from linkage stack)

* Instruction also available in ESA/390 mode (see slide 22)

- BRCL, BRASL, LARL
  - 32-bit signed Immediate field contains offset in halfwords
- TAM, SAM24, SAM31: Test/set addressing mode
- LRVR, LRV, LRVH, STRH, STRVH: Byte-reversing load/store
- RLL: Rotate left logical (with different source/result registers)
- ML, MLR; DL, DLR: Logical multiply and divide
- ALC, ALCR; SLB, SLBR
  - Add Logical with Carry, Subtract Logical with Borrow
    - Useful for multi-word or multi-precision arithmetic
- EPSW: extracts high-order half of PSW in two 32-bit pieces
- STFL: “Store Facility List” (privileged)
  - Stores bits in fullword at 200 (X'C8') indicating available hardware features

    Bit 0: indicates availability of these ops on ESA/390
    Bit 1: indicates z/Architecture mode is installed
    Bit 2: indicates z/Architecture mode is active (compare bit at X'A3')

- Need quadword alignment for LPQ, STPQ, CDSG, some PLO operands
  - LPQ, STPQ, CDSG also provide quadword consistency
- Many operating system components don't (yet) support quad alignment
- Suggestion for doubleword-aligned storage allocators:
  1. Define a DSECT mapping your quad-aligned area, with doubleword padding

```
QWORK   DSect
Quad1   DS    2D                First quadword data area
Quad2   DS    2D                Second quadword data area
        — — —                   ... and so forth, as required
QPad    DS    D           **** Doubleword padding! Never use this field!
LQWORK  Equ   *—QWORK           Length of DSect
```

  2. Allocate doubleword-aligned storage for the full size (LQWORK) of QWORK, set its address in (say) R2 (and save the address for deallocation!)
  3. Round the pointer to the next quadword boundary

```
        AHI   R2,X'0008'        Round up if not already quad aligned
        NILL  R2,X'FFF0'        Guarantee quadword alignment

        USING QWORK,R2          Use DSECT for quad—aligned work area
```

- Some instructions affect part or all of the high-order word
  - AND, OR, INSERT IMMEDIATE: NIHH, NIHL, OIHH, OIHL, IIHH, IIHL
  - LOAD LOGICAL IMMEDIATE: LLIHH, LLIHL
  - INSERT CHARACTERS UNDER MASK: ICMH
  - LOAD MULTIPLE HIGH: LMH
- Some instructions affect both parts of the register:
  - LOAD MULTIPLE DISJOINT: LMD
    - Loads high-order and low-order halves from different addresses
  - LOAD PAIR FROM QUADWORD: LPQ
    - Requires quadword alignment of operand (see slide 23)
- For some instructions, use of high-order half depends on addressing mode (see slide 26)

Modal Instructions and Addressing Modes

- Most update and/or depend on the values in registers
  - High-order 32 bits affected only in 64-bit mode
- Branch instructions:
  - BAL, BALR, BAS, BASR, BRAS, BRASL, BSM, BASSM
- Comparisons and Move instructions:
  - CLCL, CLCLE; MVCL, MVCLE
- Load Address instructions:
  - LA, LAE, LARL
- Translate instructions:
  - TRE, TRT
- String operations:
  - CLST, MVST, SRST, CUSE
- And a few others:
  - CKSM, CUUTF, CUTFU, UPT, CFC

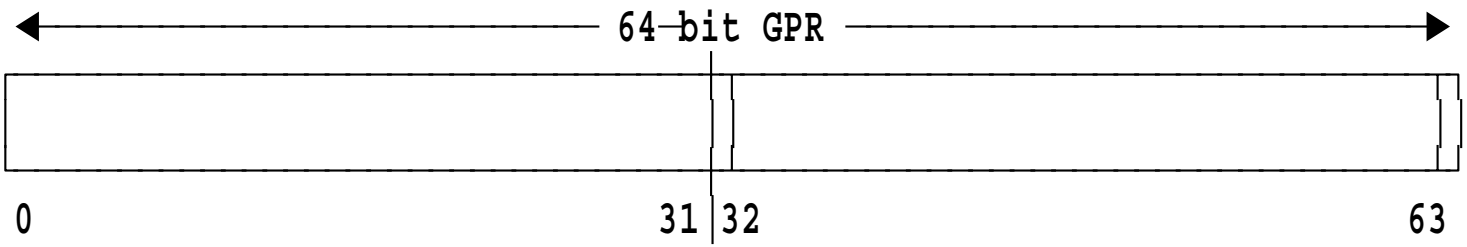
- Branch address generation in z/Arch always gives a 64-bit value
  - Leftmost 40 bits forced to zero in 24-bit addressing mode
  - Leftmost 33 bits forced to zero in 31-bit addressing mode
- Extended addressing mode (see slide 4)
  - Bit 31 of PSW (**E** bit):

    Set by LPSW, LPSWE, BASSM, BSM, Stacking PC, PR, RP, interruptions
  - Bit 32 of PSW (**B** bit):

    Set by BASSM, BSM, Stacking PC if they set the **E** bit
- Some ESA/390 call/return combinations aren't supported in z/Arch:
  - BASSM/BR, BALR/BSM, BASR/BSM
  - Linkages are either mode-switching or not

- Switching among 24-, 31-, and 64-bit addressing modes
- Branch and Set Mode instructions: BASSM, BSM
  - Set 64-bit mode if low order bit of target address is one!
    - Bit 63 is left set (so the mode switch can be detected), but is ignored in branch-address generation (target address always even)
  - Set 24- or 31-bit mode if low-order bit is zero
    - Choice between 24 or 31 depends on bit 32 being zero or one (as before)
- Set Addressing Mode instructions: SAM24*, SAM31*, SAM64
  - Set PSW's **EB** bits (see slide 4) without branching
  - They also check Instruction-Address bits to ensure next instruction isn't outside the just-specified addressing range!
- Test Addressing Mode: TAM* sets CC per PSW's “EB” bits (see slide 4)

* Instruction also available in ESA/390 mode (see slide 22)



- In all modes, BASSM and BSM change addressing mode based on bits 63 and 32 of GPR_{R2}

Bit		AMODE changed to
63	32	
0	0	24
0	1	31
1	x	64

- In 24/31-bit mode, bits 0-31 of GPR_{R1} are left unchanged
- In 64-bit mode:
  - BASSM sets a 64-bit return address in GPR_{R1} with bit 63 = 1
  - BSM sets bit 63 of GPR_{R1} to 1 (if R1 $\neq$ 0)
  - BAL/BALR/BAS/BASR do not set bit 63 of GPR_{R1} to 1
- There are many possible to/from combinations!

- BAL useful only in 24-bit mode
- BAS/BASR, BRAS/BRASL call without mode change
- Entry in 64-bit mode
  - Bit 63 = 0: entry was from BAS/BASR, BRAS/BRASL
    - No mode change; must return in same mode
  - Bit 63 = 1: entry was from BASSM/BSM
    - Mode has changed: caller's mode is in linkage register; must return via BSM
- Entry points reachable in 64-bit mode from an unknown mode:
  - Callable by BAS/BASR/BRAS/BRASL or BASSM/BSM
  - Set bit 63 of entry-point register to zero if it's to be used as a base register

```
NILL  EntryReg,X'FFFE'
USING EntryPoint,EntryReg
```

  - To test bit 63 of entry address:

```
TMLL  EntryReg,1
```

    - Follow with a **relative** (not based!) conditional branch

zSeries Support Features in High Level Assembler Release 4

HLASM web site: http://www.ibm.com/software/ad/hlasm/

- Binary constants

  FD 8-byte doubleword-aligned binary integer

  Same as currently-supported FL8, but aligned

```
000000 000000000000005F                2        DC    FD'95'
```

  Nominal value in the range $(-2^{63}, +2^{63}-1)$

- Address constants

  AD 8-byte doubleword-aligned address constant

  – Note: may not necessarily be supported in other products

  Expressions evaluated to 32 bits, sign-extended to 64

```
000008 FFFFFFFFFFFFFFA1                3        DC    AD(-95)
```

  Non-relocatable part must be in range $(-2^{31}, 2^{31}-1)$

- **OPTABLE(ZOP)** specifically selects z/Arch set
- New z/Arch machine instructions also in OPTABLE(UNI)
  - **OPTABLE(ESA)** option may help avoid macro conflicts with new mnemonics (Or, use macros with names longer than 5 characters)
- Vector instructions still available in the assembler (OPTABLE(UNI))
  - But the “Freeway” architecture can't execute them...
- New sub-option: OPTABLE(xxx,LIST)
  - Displays all mnemonics supported by opcode table xxx
- LIST sub-option and ZOP opcode table enabled by APAR PQ44437
- `AMODE` and `RMODE` operand extensions

  AMODE ANY31, 64

  RMODE 31, 64

  “64” operands may not be supported by other products
  - New mode values indicated in ESD listing's “Flag” byte

Summary

- New architecture
  - Vastly increased addressing capabilities
  - Upward compatibility
- Many new instructions
  - 64-bit arithmetic without 64-bit addressing
  - New operations for ESA/390 also
- Smooth integration with existing architecture
  - Many operations work in both architecture modes
- A substantial base for future growth
- Evolution, not revolution!

- z/Architecture Principles of Operation (SA22-7832)
- High Level Assembler for MVS & VM & VSE publications:
  - Language Reference (SC26-4940)
  - Programmer's Guide (SC26-4941)