High Level Assembler Release 5:
New and Useful Features and Functions

SHARE 103 (Aug. 2004), Session 8164

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

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

August, 2004

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

IBM	ESA	System/370	System/370/390
System/390	MVS/ESA	OS/390	VM/ESA
VSE/ESA	VSE	z/OS	z/VM
z/VSE	z/Architecture	zSeries	DFSMS

The following are trademarks or registered trademarks of other corporations or consortia:

Windows 95	Windows 98	Windows 2000	Windows NT
Windows XP	Unicode		

Announcement Letter 204-122

- Migration considerations
- Support for new z/Architecture instructions
- New data types and constants
- Four new options: MACHINE, SECTALGN, SUPRWARN, TYPECHECK
- Enhanced assembler instructions
- Conditional assembly extensions
- Listing enhancements
- ADATA enhancements
- New and enhanced diagnostics and messages
- Other assembler enhancements
- Toolkit Feature enhancements

1. Architecture Level Set
   - HLASM R5 requires machines with Immediate-and-Relative and String-Instruction Facilities (post-1996 machines)
2. SYSUT1 utility file removed
   - HLASM R5 may require more central storage
     - See the estimate of central storage produced by HLASM R4 summary page
   - SYSUT1 DD statements are ignored; no need to change JCL
3. HLASM R5 uses a different format for SYSADATA records
   - A record reformatting exit ASMAXADR is provided
     - Maps R5 record format to R4 format

- More details following ...

- Relative-Immediate instructions may now specify external symbols as operands
  - Requires GOFF option
  - See slide 19 for an example
  - Binder support probably in z/OS release 7
- Integrated support for long-displacement instructions
  - Provided via APAR in HLASM R4
- Newly added instructions:
  - ASN/LX Reuse Facility, Extended-Translation Facility 3
  - Both added to HLASM R4 by APAR PQ85963

- Address constants

  QD 8-byte doubleword-aligned offset

  RD 8-byte doubleword-aligned PSECT address

  VD 8-byte doubleword-aligned address

  JD 8-byte doubleword-aligned length

- Character constants (example on slide 6)

  CA ASCII character constant (unaffected by TRANSLATE option)

  CE EBCDIC character constant (unaffected by TRANSLATE option)

- Other constant

  LQ Quadword-aligned hex floating point constant
  Primary uses expected to be such as “ `DS 0LQ` ”

- Previous releases supported AD, FD

- Character constants

CA Nominal value is in ASCII, not EBCDIC

```
000000 C182C384C5F2F940  ◄──EBCDIC      4     DC    CL8'AbCdE29'
000007 4162436445323920  ◄──ASCII       5     DC   CAL8'AbCdE29'
00000E C182C384C5F2F940  ◄──EBCDIC      6     DC   CEL8'AbCdE29'
```

CE With TRANSLATE(AS) option, nominal value is always EBCDIC

```
000000 4162436445323920  ◄──now ASCII   4     DC    CL8'AbCdE29'
000007 4162436445323920  ◄──ASCII       5     DC   CAL8'AbCdE29'
00000E C182C384C5F2F940  ◄──EBCDIC      6     DC   CEL8'AbCdE29'
```

- Four new options: MACHINE, SECTALGN, SUPRWARN, TYPECHECK
- **MACHINE**(*machine-spec*[,LIST])
  - A useful alternative to the OPTABLE option
    - LIST sub-option produces same output as OPTABLE(xxx,LIST)
  - Equivalent MACHINE and OPTABLE sub-options:

MACHINE	OPTABLE	Description
(none)	UNI	The default opcode table; most recent instructions
(none)	DOS	For compatibility with the old DOS/VSE assembler
S370	370	System/370 instructions, including vector ops
S370XA	XA	System/370 XA architecture
S370ESA	ESA	Same as S370XA
S390	ESA	Same as S370XA
S390E	ESA	System/370 and System/390 instructions
ZSERIES	ZOP	The “Freeway” 64-bit instructions
ZS	ZOP	Same as ZSERIES
ZSERIES-2	YOP	The “T-Rex” instructions (long displacements)
ZS-2	YOP	Same as ZSERIES-2

- **SECTALGN**(*alignment*)
  - *alignment* a power of 2 between 8 and 4096
  - Specifies initial alignment of all control sections
  - Alignments greater than 8 require either
    1. the GOFF option
    2. link-time control statements to ensure section alignment

- **SUPRWARN**(*message-numbers*) or **NOSUPRWARN**(*message-numbers*)
  - Specifies *message-numbers* of warnings to be suppressed (or not)
  - Only works for messages with severity 4 or less:

```
  Process Statements-     SUPRWARN(001)
** ASMA319W Message 001 specified for SUPRWARN option,
   but severity is too high. Message ignored.
```

- Example without suppressed messages:

```
                                        1 *process suprwarn(001)
000000                 00000 0000C      2 sw  csect ,
                  R:F  00000            3     using *,15

000000 5800 F001              00001     4     l     0,*+1
** ASMA033I Storage alignment for *+1 unfavorable
** ASMA435I Record 4 in SW       ASSEMBLE A1 on volume: EHR191

000004 47F0 F005              00005     5     b     *+1
** ASMA212W Branch address alignment for *+1 unfavorable
** ASMA435I Record 5 in SW       ASSEMBLE A1 on volume: EHR191

                  R:E  00008            6     using *,14
** ASMA303W Multiple address resolutions may result from this USING
   and the USING on statement number 3
** ASMA435I Record 6 in SW       ASSEMBLE A1 on volume: EHR191

000008 A718 0001              10001     7     lhi   1,65537
** ASMA320W Immediate field operand may have incorrect sign or magnitude
** ASMA435I Record 7 in SW       ASSEMBLE A1 on volume: EHR191
                                        8     end
```

- Example using SUPRWARN to suppress warnings:

```
                                        1 *process suprwarn(1,319,435,33,212,303,320)
000000                  00000 0000C     2 sw  csect ,
                  R:F   00000           3     using *,15
000000 5800 F001                00001   4     l     0,*+1
000004 47F0 F005                00005   5     b     *+1
                  R:E   00008           6     using *,14
000008 A718 0001                10001   7     lhi   1,65537
                                        8     end
```

  - Suppressed messages shown on Summary page (see slide 20)
- **TYPECHECK(MAGNITUDE | REGISTER)**
  - MAGNITUDE checking introduced via APAR in HLASM R4 (without suboption)
  - REGISTER checking is new
- **TYPECHECK(REGISTER)** uses new EQU operand (see slide 14)

- One changed option: in **SIZE(xxx,ABOVE)**, the ABOVE keyword is accepted but ignored
  - Main working storage is now acquired above 16Mb

- Changed: *PROCESS, ACONTROL, AMODE, CATTR, CNOP, DC/DS, DXD, EQU, ORG, EXTRN/WXTRN
- ***PROCESS** statement allows OPTABLE, MACHINE, SECTALGN, SUPRWARN, TYPECHECK options
  - Unified opcode table enables easy selection of instruction sets
- **ACONTROL** statement allows TYPECHECK option
- **AMODE** supports ANY64 operand
  - AMODE also specfiable for ENTRY and EXTRN names
    - May not be supported by all system components
- **CATTR** has two new operands: PART and PRIORITY
  - **PART**(*part-name*) for members of MERGE classes
    - Typically needed for communicating with DLLs
  - **PRIORITY**(*nnn*) to request bind-time ordering of parts in a class
  - CATTR uses SECTALGN value if no **ALIGN** operand is specified

- **CNOP** has new values for the *byte* and *boundary* operands
  - *boundary*: 4, 8, and 16
  - *byte*: any even positive number less than *boundary*

```
          CNOP  10,16    Positions at sixth halfword of a quadword
```

  - **CNOP** also “optimizes”

```
000000                              2    ds    h
000002 070047000700                 3    cnop  0,8    One less NOPR !!
```

- **DC/DS** supports a new *program type* subfield
  - Follows the constant type and type extension, precedes modifiers

```
MyName    DC   CP(C'MyNm')L16'John R. Ehrman'
MyWeight  DC  DBP(X'2004')'75.4'
```

  - Operand of P(--) subfield is an absolute expression
  - Intended for macros that check instruction vs. operand types
  - Symbol's *program type* retrievable using SYSATTRP function (see slide 17)

- **DXD** supports quad-aligned operands
- **EQU** has two new operands:

```
symbol  EQU   value,length,old-type,program-type,assembler-type
```

  - *program type*: same uses as with DC/DS statement
    - You create symbols with known program types; useful in macros
  - *assembler type*: fixed values known to the assembler
    - Enables stronger type checking of register usage

- Example:

```
R1   EQU   1,,,C'DPtr',GR          General Purpose Register
GR2  EQU   2,,,C'Work',GR64        64-bit General Purpose Register
```

- SYSATTRA, SYSATTRP functions retrieve new attributes (see slide 17)

- Using the *assembler type* for EQUated register symbols:

```
                    00002         4 R2  EQU   2,,,,GR     32- or 64-bit
                    00002         5 GR2 EQU   2,,,,GR64   64-bit register
000000 1822                       6     LR    R2,R2       Compatible
000002 B904 0022                  7     LGR   GR2,GR2     Compatible

000006 1822                       8     LR    R2,GR2      Incompatible
** ASMA323W Symbol GR2 has incompatible type with general register field
(GR2 is explicitly a 64-bit register; warning message)

000008 B904 0022                  9     LGR   GR2,R2      Questionable
** ASMA324I Symbol R2 may have incompatible type with general register field
(R2 is either a 32- or 64-bit register; information message)

                    00003        10 R3  EQU   3,,,,GR32   32-bit register
00000C 1823                      11     LR    R2,R3       Questionable
** ASMA324I Symbol R2 may have incompatible type with general register field
(R2 is either a 32- or 64-bit register; information message)

                    00001        12 R1  Equ   1           No type
00000E 1831                      13     LR    R3,R1
** ASMA324I Symbol R1 may have incompatible type with general register field
(R1 is untyped, but other register symbols have types; information message)
```

- **ORG** has two new operands:

```
ORG    expression,boundary,offset
```

  - *boundary* a power of 2 from 8 to 4096
    - Diagnostic if *boundary* exceeds SECTALGN value
  - *offset* any absolute expression
  - HLASM (1) adjusts Location Counter, (2) rounds it up to next *boundary*, and (3) adds *offset*

```
ORG    *+2,16,-2    Now at 2 bytes before next quadword boundary
```

- **EXTRN/WXTRN** PTF will eventually support declaration of external Parts
  - Probable syntax: EXTRN PART(*part-name*)
    - May not be supported by all system components

- Longer macro arguments and SETC variables
  - Increased from 255 character limit to up to 1024 characters
- Many new built-in functions, some using new “function” syntax
  - Built-in functions invoked in either of two formats:
    - logical-expression format, which takes two forms:

```
    (expression operator expression)      (A OR B)   (&J SLL 2)
or
    (operator expression)                 (NOT C)    (LOWER '&String')
```

      Logical-expression format also used in SETB and AIF

    - function-invocation format:

```
    function(argument[,argument]...)      FIND('&Str','&Key')
```

- Some existing functions now available in function-invocation format:
  BYTE DOUBLE FIND INDEX LOWER SIGNED UPPER

  Examples: UPPER('xyz') LOWER('&String') BYTE(X'00') SIGNED(-5)

- Symbol attribute retrieval functions: SYSATTRA SYSATTRP
- Validity-testing functions: ISBIN ISDEC ISHEX ISSYM
- Representation-conversion functions (see summary on slide 18)
  - From arithmetic value: A2B A2C A2D A2X
  - From binary string: B2A B2C B2D B2X
  - From characters: C2A C2B C2D C2X
  - From decimal string: D2A D2B D2C D2X
  - From hex string: X2A X2B X2C X2D
    - Note: all D2* functions accept optional sign; *2D functions always generate a sign
- New functions all use function-invocation format

A-value	arithmetic expression
B-string	string of binary digits
C-string	string of characters
D-string	string of (optionally) signed decimal digits
X-string	string of hexadecimal digits

	Output Value				
Input Value	**A-value**	**B-string**	**C-string**	**D-string**	**X-string**
A-value	–	A2B	A2C, BYTE	A2D, SIGNED	A2X
B-string	B2A	–	B2C	B2D	B2X
C-string	C2A	C2B	–	C2D	C2X
D-string	D2A	D2B	D2C	–	D2X
X-string	X2A	X2B	X2C	X2D	–

```
A2C(-252645136)   has value '0000'      (-252645136 is an A-value)
B2X('11')         has value '3'         ('11'  is a  B-string)
C2B('a')          has value '10000001'  ('a'   is a  C-string)
D2X('+25')        has value '19'        ('+25' is a  D-string)
X2B('E')          has value '1110'      ('E'   is an X-string)
```

- “Active Usings” heading: default ranges omitted
- Relocation Dictionary code and listing example:

```
RLDL  CSECT ,
      DC    A(*+23-EXTSYM)
      DC    VD(VSYM)
      EXTRN EXTSYM
      BRAS  14,EXTSYM
      BRASL 14,EXTSYM
      END
```

```
                                                Relocation Dictionary
 Pos.Id   Rel.Id   Address  Type  Action
00000003 00000003 00000000   A 4     +     <-- 4-byte A-type
00000003 00000005 00000008   V 8    ST     <-- 8-byte V-type
00000003 00000006 00000000   A 4     -     <-- 4-byte A-type, subtracted
00000003 00000006 00000012  RI 2     +     <-- 2-byte R-I type
00000003 00000006 00000016  RI 4     +     <-- 4-byte R-I type
```

- Symbol Cross Reference displays program and assembler types:

```
R2    EQU   2,,,C'DPtr',C'GR'
R3    EQU   2,,,C'CBas',C'GR32'
GR2   EQU   2,,,,C'GR64'
DAT   DC    CP(X'18C6')L12'THIS IS DATA'

Symbol  Length   Value     Id    R Type Asm  Program  Defn References
DAT         12 00000014 00000001     C  C    000018C6   14
GR2          1 00000002 00000001 A   U  GR64              6    9M  ...etc.
R2           1 00000002 00000001 A   U  GR   C4D7A399    4    7M  ...etc.
R3           1 00000002 00000001 A   U  GR32 C3C281A2    5    8   ...etc.
```

- Assembly summary
  - Displays suppressed messages (see slide 9)

```
Suppressed Message Summary

Message     Count   Message     Count   Message     Count
   33         1       212        1       303        1
  319         0       320        1       435        4
```

  - Actual storage used

- Macros and COPY files in USS file system
  - Path names displayed wherever data set names/volumes would appear
- ADATA exit, files, and records:
  - ADATA exit now allows all editing actions (same as other exits)
  - Longer maximum record length allowed (32K)
  - ASMAXADR sample exit selects/rejects records for reformatting to R4 format
  - ASMAXADC sample exit allows record selection/deletion/modification
  - See Appendix I of Programmer's Guide for details
    - ADATA-record edition numbers: 0 (ADATA ID, start/end); 2 (assembly statistics); 3 (options); 1 (all others)
- Unified opcode table; fewer modules to install/maintain
- Numeric version id in static assembler info in exit/function interfaces
- REINIT call for exits, for “batch” assemblies
- Extensive revisions and enhancements to publications (as well as many additions)

- New Messages

026S Macro call operand too long, leading characters truncated
176E Operand 4 must be absolute, 1-4 bytes
177E Operand 5 must be absolute, 1-4 bytes
318W Invalid message number specified for SUPRWARN
319W Message number specified for SUPRWARN severity is too high
321E Invalid assembler type
322E Program type error
323W Symbol xx has incompatible type with xx register field
324I Symbol xx may have incompatible type with xx register field
443N ASMAOPT record format invalid, options ignored
500W Requested alignment exceeds section alignment

- Numerous other messages clarified

- Many messages relating to SYSUT1 are removed

- Different object records due to time/date stamps on the END record or in the text
- Different object records due to literal pool alignment being stricter
  - Quadword literal may be present
  - If SECTALGN(16) is specified, any 16-byte literal is quadword aligned
- Macro arguments like **"**`(R0)`**"** may generate `LR 0,R0`
- Conditional assembly code that formerly caused string truncation may not do so
- Code that formerly assembled without diagnostics may generate them

- License checking for all “host” products (i.e., not ASMPUT)
- ASMIDF symbolic debugger extensions:
  - **ASMIDFB** command lets you debug in an MVS batch environment
  - **LUNAME** option defines VTAM LU name of IDF batch-debug terminal
- Three new disassembler options:
  - **NEWNUM** lets you use decimal or hex fields in control statements
  - **OPTABLE** selects opcode table used in disassembing
  - **VSESVC** uses VSE definitions to identify SVCs
- Enhanced SuperC process options and process statements:
  - **SYSIN** specifies alternate DDname for process statements
  - **WORKSIZE** sets maximum size of comparison set for large files
  - **NEWDD, OLDDD, UPDDD** specfies alternate DDnames for input and output files
- Enhanced Structured Programming Macros (provided via PTF in R4)
  `ASMLEAVE, ELSEIF, ITERATE`
- Toolkit Overview in session **8171**

- Four new options
- Eleven statements enhanced and extended
- Many conditional assembly extensions:
  - Longer strings, new functions, new symbol attributes
- Toolkit Feature enhancements
  - Extensions, enhancements to disassembler, ASMIDF debugger, and SuperC
- Strategy and directions
  - Support evolving hardware and operating system enhancements
  - Help you create robust and maintainable applications
  - Provide conditional assembly extensions helping you create richer programming tools

IBM's continuing commitment to support Assembler applications

- Other sessions of possible interest:

Session	Session Title
8163	HLASM Programming in a Relative Instruction Set World
8171	HLASM Toolkit Feature Overview and Demos
8188	Using USING: Unlocking the Deepest Secret of Assembler
8155	Putting the “You” in Unicode
8132	Extending the Life Cycle of Legacy Applications
8162	Program Checks, ABENDs, Dumps: What They Are and How to Stop Suffering From Them
8116	Introduction to I/O Programming in Assember Language
8173	How HLASM Helps Find/Fix Assembler Language Problems
8156	Programming for Data Spaces in HLASM
2811-2	Channel Programming: Reviewing the Basics
8162	Stalking the New OpCodes
8130	Using Dynamic Link Libraries (DLLs) with LE
8165	HLASM Features: Benefits and Exploitation
8137	Structured Assembler Language Programming
8154	Brushing Up on the Assembler Classics

- High Level Assembler for MVS & VM & VSE publications:
  - Language Reference (SC26-4940)
  - Programmer's Guide (SC26-4941)
  - Toolkit Feature User's Guide (GC26-8710)
  - Toolkit Feature Interactive Debug Facility User's Guide (GC26-8709)
  - ...and others of not-as-general interest
- z/OS publications:
  - z/OS MVS Programming: Assembler Services Guide
  - z/OS MVS Programming: Assembler Services Reference
- HLASM documentation available from the HLASM web site:

http://www.ibm.com/software/awdtools/hlasm/

- We need your advice and guidance:

1. Do you need a batch utility to convert R5-format SYSADATA files to R4 format? (It would use ASMAXADR to do the conversion)

2. Do you need a batch utility to convert existing R4-format SYSADATA files to the new R5 format?

3. OK if all new functions use function-invocation format?

4. Should we implement a “SYSATTRT” function to return a symbol's type (and type extension)?

5. Should we disallow address constants of lengths 5, 6, and 7? (There's no system support for them if external symbols are used.)

6. Should we change LTORG to align the literal pool to the minimum necessary boundary?