Enterprise COBOL V5 Compiler, an applied science…

- Introduces advanced optimization technology
  - Designed to optimize applications for current and future System z hardware
  - Initiate delivery of performance improvements seen in C/C++ and Java compilers on System z
- Compiler "back end" is replaced with technology that has long been in use in IBM's Java products. (Back end = part of compiler that does code generation and optimization)
  - Mature, robust compilation technology.
  - New COBOL-specific optimizations have been added.
- Exploits z990, z890, System z9, System z10, zEnterprise 196, zEC12 and zBC12.

New Code Generator and Program Optimizer

- Common components means more timely exploitation of future zArchitecture advances.
- Support modern development tools
  - Tools supplied by ISV's
  - IBM z/OS Problem Determination Tools
  - Rational Development Tools
- Continue to deliver new features to simplify programming and debugging to increase productivity to modernize existing business critical applications
- Use industry standard DWARF, with documented IBM extensions to represent debug information.
  - APIs are available to allow tools to inspect this information.

PDSE for COBOL V5 executables -

COBOL improving performance using new features that are only available in Program Objects (PO)

Instruction Scheduling For Performance

Timing – (100 million in a loop)
V5 : 2.35 cpu seconds
V4 : 2.50 cpu seconds
V5 is 6% faster

New Compiler Options for performance

- ***ARCH (6 | 7 | 8 | 9 | 10)***
  - Allows code generator to use instructions found in various levels of z Architecture
- ***OPTIMIZE(0 | 1 | 2)***
  - Levels of optimization
    - > Higher levels improve run time performance
    - > Highest level has somewhat reduced "debuggability"
- ***STGOPT / NOSTGOPT***
  - Allows compiler to delete unreferenced data items
- ***HGPR (PRESERVE | NOPRESERVE)***
  - Use high word of registers (upper 32 bits of 64-bit registers)
  - Effectively adds 16 more registers to improve optimization
- ***AFP(VOLATILE | NOVOLATILE)***
  - Use full complement of floating point registers.

Decimal Divide Where Operands Exceed Packed Decimal Hardware Limits

Timing (100 million in a loop)
V5 : 1.08 cpu seconds
V4 : 4.81 cpu seconds
V5 is 78% faster

Binary Arithmetic Operands > than 9 Digits

Timing (100 million in a loop)
V5 : 0.23 cpu seconds
V4 : 1.92 cpu seconds
V5 is 88% faster

New Compiler Options for usability

- *DISPSIGN(SEP)*
  - *DISPSIGN controls output formatting for DISPLAY of signed numeric items.*
  - *Can format overpunch sign as separate sign for easier to read output:*

	DISPSIGN(COMPAT):	DISPSIGN(SEP):
positive binary	111	+111
negative binary	11J	-111
positive packed-decimal	222	+222
negative packed-decimal	22K	-222

- *LVLINFO* (installation option)
  - Now 8 bytes instead of 4, you can put APAR, PTF, or your own numbers
  - Example: LVLINFO=PN123456
  - Listing header:

PP 5655-W32 IBM Enterprise COBOL for z/OS 5.1.0 PN123456
Date 05/20/2013 Time 10:45:03

Signature bytes:
00088E (+40) 00408000 =X'00408000' INFO. BYTES 24-27
000892 (+44) D7D5F1F2F3F4F5F6 =C'PN123456' USER LEVEL INFO (LVLINFO)
Compiler Options and Program Information Section End

Compatibility

- Provide Source and binary compatibility
- Most correct COBOL programs will compile and execute without changes and produce the same results
  - "Old" and "new" code can be mixed within an application and communicate with static, dynamic and DLL calls
  - No need to recompile entire applications to take advantage of new V5 features
- Removed some old language extensions and options
  - Millennium Language Extensions
  - Label Declaratives
  - Non-reentrant programs above 16MB line
  - OS/VS COBOL Inter-operation
  - COBOL V3 (COMPAT) XML PARSER
  - Static AMODE 24 CALLs

COBOL language removed

- *Millennium Language Extensions*
- The removed elements are:
  - DATE FORMAT clause on data description entries
  - DATEVAL intrinsic function
  - UNDATE intrinsic function
  - YEARWINDOW intrinsic function
  - DATEPROC compiler option
  - YEARWINDOW compiler option

"New" machine op code example – Initializing BLL Cells

COBOL V4 — *Loop to initialize 8 BLL Cells*

```
        LA    1,0 (0,1)
        ST    1,308 (0,9)      BLL=1
        L     8,308 (0,9)      BLL=1
        L     15,16 (0,10)
        LA    14,308 (0,9)     BLL=1
GN=13   EQU   *
        AL    1,12 (0,10)
        AH    14,24 (0,10)
        ST    1,0 (0,14)
        BCT   15,324 (0,11)    GN=13
```

COBOL V5 — *Only one BLL – All ARCH Levels*

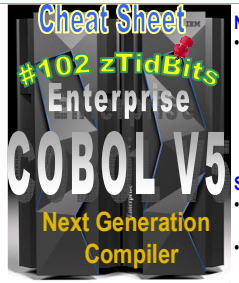
```
        L     R0, 0 (, R1)
        NILH  R0,32767
        ST    R0, 0 (,R8)
```

Timing (100 million in a loop)
V5 : 4.44 cpu seconds
V4 : 5.15 cpu seconds
V5 is 14% faster

Binary Arithmetic Conditional Precision Correction

V4 - Divide (D) to correct precision always executed but rarely needed
V5 - Divide (DR) to correct precision only executed when actually required ARCH(8)

Timing (100 million in a loop)
V5 : 0.18 cpu seconds
V4 : 0.52 cpu seconds
V5 is 65% faster



New compiler features introduced

- Improved usability
  - Reduced administration overhead with support for z/OS System Management Facilities (SMF) records
  - New NOLOAD debugging segments in program object
    - > Debugging data always matches executable
    - > No separate debugging files to find or keep track of
    - > Executable does not have bigger loaded footprint
  - New pseudo-assembly in program listings

Some New COBOL language features

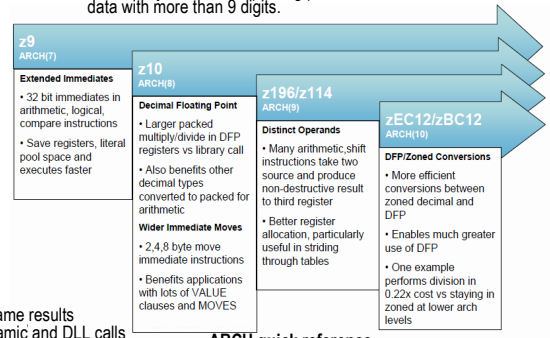
- Floating comment delimiter
  - *> to end of line is a comment
- Raise WORKING-STORAGE section size limit to 2GB
  - (from 128MB)
- Larger individual data items
  - Up to 999,999,999 bytes!
- Support for UNBOUNDED tables
  - X OCCURS 1 To *UNBOUNDED* Depending on Y.
  - LINKAGE SECTION only

COBOL language discards

- *LABEL DECLARATIVES*
  Format 2 declarative syntax:
  USE … AFTER … LABEL PROCEDURE
  And the syntax:
  GO TO MORE-LABELS
  are no longer supported.
  Note: GO TO is still supported.

Performance Improvements at all ARCH Levels

- The compiler accepts ARCH(6) - ARCH(10) all of which also exploit:
  - Relative Instruction
    - > Jumps (branches) and nested program calls can be relative to the executing instruction
    - > Access to the literal pool can also be relative to the executing instruction
  - Half word immediate instructions
    - > Load, Load Logical ANDs, ORs, Add and Subtract logical
  - Twelve additional floating point registers
  - Long Displacement Facility
    - > Many load/store instructions that have a 0-4095 displacement now have a "Y" format with a -524,288 displacement reach
    - > Now one base register can cover much more working storage and this reduces need for base locators
  - 64 bit "G" form instructions
    - > 64 bit computations can be done in single registers vs piecewise in 32 bit registers
    - > Particularly useful for improving performance of COBOL BINARY data with more than 9 digits.



ARCH quick reference

- ARCH(6)
  2084-xxx models (z990)
  2086-xxx models (z890)
- ARCH(7)
  2094-xxx models (IBM System z9 EC)
  2096-xxx models (IBM System z9® BC)
- ARCH(8)
  2097-xxx models (IBM System z10 EC)
  2098-xxx models (IBM System z10 BC)
- ARCH(9)
  2817-xxx models (IBM zEnterprise z196 EC)
  2818-xxx models (IBM zEnterprise z114 BC)
- ARCH(10)
  2827-xxx models (IBM zEnterprise EC12)
  2828-xxx models (IBM zEnterprise BC12)

Some new COBOL language introduced

- New Intrinsic Functions to improve handling of UTF-8 data
- XML GENERATE features for controlling document generation
  - NAME OF phrase
    - > User supplied element and attribute names
  - TYPE OF phrase
    - > User control of attribute and element generation
  - SUPPRESS phrase
    - > Suppression of "empty" attributes and elements
  - XML PARSE feature for easier handling of split content:
    - > XML-INFORMATION special register

Optimization of Initialization By Literals

Timing (100 million in a loop)
V5 : 0.16 cpu seconds
V4 : 0.25 cpu seconds
V5 is 36% faster

Optimization of Decimal PICTURE Scaling

Timing (100 million in a loop)
V5 : 0.31 cpu seconds
V4 : 2.02 cpu seconds
V5 is 85% faster

UTF-8 Unicode Built-in Functions

- UTF-8 Characters are 1 – 4 bytes in length.
- ULENGTH: returns the logical length of a UTF-8 string
- UPOS: returns the byte position in a UTF-8 string of the Nth logical character.
- USBSTR: returns the sub-string of N logical characters starting from a given logical character.
- UVALID: takes an alphanumeric or alpha or national item and returns zero or the index of the first invalid UTF-8 (alphanumeric or alpha) or UTF-16 (national) character.
- UWIDTH: returns the width in bytes of the Nth logical character.
- USUPPLEMENTARY: takes a UTF-8 or UTF-16 string and returns zero or the first UNICODE supplementary character.

New XML PARSE features

- XML PARSE features: before
  - Lots of code 'just in case' content gets split
  - Example is minimized, real world example is even worse
- XML PARSE features: after
  - XML-INFORMATION tells us when content is complete
  - Only need 1 buffer since collecting attribute data will not be ended by element content
    - > Can do all work within code for ATTRIBUTECHARACTERS and CONTENT-CHARACTERS events
    - > Not spread all over the program

Debug Tool improvements for COBOL V5

- Debug Tool was completely re-instrumented to work with COBOL V5.1:
  - Access to DWARF debug data in NOLOAD classes
  - Change to Debug Tool 'Level 4 APIs' from historic level 1
  - New COBOL runtime and COBOL debug support runtime
- A few of the many improvements in the Debug Tool experience with COBOL V5.1:
  - STEP OVER of PERFORM statements
  - Improved presentation of tables (arrays)
  - Improved presentation of data descriptions

Differences in using the new compiler

- **Compiler resides in PDSE**
  - Can no longer be added to LPA via LPALST
  - Can add to LPA dynamically after system IPL
- **Compiler uses LE at compile time**
  - Must be the latest, with all V5 PTFs for installed
  - SCEERUN and SCEERUN2
- **Compiler EXITs (EXIT compiler option) changes**
  - Cannot use RTEREUS, IGZERRE or CEEPIPI
  - Can still use non-LE-conforming assembler
  - Can use LE-conforming assembler
  - Can use old or new COBOL
  - Cannot use STOP RUN in compiler exit programs

Differences in using the new compiler

- **More datasets used by COBOL V5**
  - More working datasets
  - SYSUT8-SYSUT15 are now required
  - SYSMDECK is now required
- **More memory required at compile time**
  - Recommend 200M region size
  - Old compiler could compile most programs with 10M
  - You may have to change system user id limits
- **More time required to compile**
  - 3 to 12 times as much time, depending on OPT level and source program size

Binding COBOL V5 programs

- **No more load modules**
  - Compiler output is GOFF format object, input to binder
  - Binder output is Program Object
  - Program Object cannot reside in PDS,
  - PDSE only

New compiler options for performance:

OPTIMIZE(0 | 1 | 2)
- Levels of optimization
  - Higher levels improve run time performar
  - But also increase compile time
  - Highest level has slightly reduced "debuggability"
- Applications that have user-written condition handlers are restricted, must use OPT(0) w/NOTEST
  - With TEST compiler option can use any OPT level

MAXPCF(nnnnn)
- *Some programs can take too much time or memory to compile*
  - *Sets OPT level to 0 if program is too large or complex*
  - *Automatic control of OPT levels for large or complex programs*
  - *Default = 60,000 (MAXPCF(0) means always optimize)*

HGPR (PRESERVE | NOPRESERVE)
- Use high word of registers (upper 32 bits of 64-bit registers)
  - Effectively adds 16 more registers to improve optimization
  - ***AFP(VOLATILE | NOVOLATILE)***
  - Use full complement of FP registers.

STGOPT / NOSTGOPT
- Allows compiler to delete unreferenced data items
  - Analogous to FULL suboption of OPTIMIZE in V4
  NOTE: Do not use if you have 'eye-catchers' in WORKING-STORAGE *or use a fake reference:*
  - *IF X NOT = X THEN Display Eye-Catche*
  - **Use IBM defaults as your installation defaults for all of these**
  - Change only for special cases, individual applications