



IBM Software Group

# PL/I言語の最新バージョン V3.8に関する技術解説

Rational. software

[Go to IBM](#)

ハイバリューサポートセンター・サーバーソリューション事業開発  
コンサルティング・IT スペシャリスト 新井 寛



IBM Software Group

# z10 Exploitation

**Rational.** software



# ARCH(8)

- **ARCH(8) and TUNE(8) are now supported**
- **This includes support of the “general-instructions-extension” facility**
- **Which includes these new instructions:**



# ARCH(8)

- **ADD LOGICAL WITH SIGNED IMMEDIATE**
- **COMPARE AND BRANCH (RELATIVE)**
- **COMPARE (HALFWORD) RELATIVE LONG**
- **COMPARE IMMEDIATE AND BRANCH (RELATIVE)**
- **COMPARE LOGICAL AND BRANCH (RELATIVE)**
- **COMPARE LOGICAL IMMEDIATE AND BRANCH (RELATIVE)**
- **COMPARE LOGICAL RELATIVE LONG**
- **LOAD HALFWORD RELATIVE LONG**
- **LOAD LOGICAL (HALFWORD) RELATIVE LONG**
- **LOAD RELATIVE LONG**
- **MULTIPLY SINGLE IMMEDIATE**
- **STORE (HALFWORD) RELATIVE LONG**



# ARCH(8)

- It also includes additional formats for these “old” instructions
- ADD IMMEDIATE
- COMPARE HALFWORD
- COMPARE HALFWORD IMMEDIATE
- COMPARE LOGICAL IMMEDIATE
- LOAD ADDRESS EXTENDED
- LOAD AND TEST
- MOVE
- MULTIPLY
- MULTIPLY HALFWORD



## ARCH(8)

- As an example of the new format for COMPARE HALFWORD IMMEDIATEs, consider the code generated for this simple function

```
1.0      test: proc( p ) returns( fixed bin(31) byvalue );
2.0          dcl p          ptr byvalue;
3.0          dcl bf          fixed bin(31) based;
4.0
5.0          return( ( p->bf = 1367 ) );
6.0      end;
```



# ARCH(8)

- Under OPT(3) DFT(REORDER) ARCH(7), the essential part is

000046	58E0	1000	0000	L	r14, P(, r1, 0)
00004A	4100	0557	0005	LA	r0, 1367
00004E	41F0	0001	0005	LA	r15, 1
000052	5900	1000	0005	C	r0, _shadow1(, r1, 0)
000056	A784	0004	0005	JE	@1L2
00005A	41F0	0000	0005	LA	r15, 0
00005E			0006	@1L2 DS	0H



## ARCH(8)

- But under OPT(3) DFT(REORDER) ARCH(8), the essential part uses one less register and one less instruction:

000046	58E0	1000		0000		L	r14, P(, r1, 0)
00004A	41F0	0001		0005		LA	r15, 1
00004E	E55C	1000	0557	0005		CHSI	_shadow1(r1, 0), H' 1367'
000054	A784	0004		0005		JE	@1L2
000058	41F0	0000		0005		LA	r15, 0
00005C				0006	@1L2	DS	0H





IBM Software Group

# Integrated XML System Services parser

**Rational.** software

[→ Go to IBM](#)

# PLISAXC

- Like PLISAXA and PLISAXB except PLISAXC supports
  - ▶ documents larger than 2GB
  - ▶ documents in UTF-8
  - ▶ Namespaces
- Requires library APAR PK68704
- Supported only on z/OS (since that's the only place where the XML System Services parser is located)



# PLISAXC

- **More like PLISAXA than PLISAXB**
- **You must pass the address and length of a buffer containing XML**
- **But the buffer need not be complete**
- **If the parser finds that the XML document is incomplete, it will invoke an event where you can provide more XML**



# PLISAXC

- **Parameters still consist of**
  - ▶ **An event structure**
  - ▶ **A user token**
  - ▶ **The address of the XML document (or the first thereof)**
  - ▶ **The size in bytes of the XML at that address**
  - ▶ **Optionally, the purported code page of the document**
  
- **But the event structure and some of the events are different**



# PLISAXC

- **The key difference is that there is now an end-of-input event**
- **This is the only event that gets BYADDR parameters, namely**
  - ▶ **The address of more of the document**
  - ▶ **The associated length**
- **The parser will then resume (although the next event may be a request for more input)**
- **There is no requirement that the buffers be broken at logical boundaries – you could supply all the XML 1367 bytes at a time if you wanted**



# PLISAXC

- **Also, the following events receive namespace information**
  - ▶ **namespace-declare**
  
  - ▶ **Start-of-element**
  - ▶ **End-of-element**
  - ▶ **Attribute-name**





IBM Software Group

# UTF-handling built-in functions

**Rational.** software

[→ Go to IBM](#)

# UTF-handling built-in functions

- **The following new built-in functions will allow significantly improved support for UTF-8 and UTF-16**
  - ▶ **ULENGTH**
  - ▶ **ULENGTH8**
  - ▶ **ULENGTH16**
  - ▶ **UPOS**
  - ▶ **USUBSTR**
  - ▶ **UVALID**
  - ▶ **UWIDTH**
  
- **Requires library APAR PK68705 and PK68708**



## A little UTF background

- In some ASCII code pages, an a-umlaut is the code point 'E4'x
- This is not valid UTF-8
- in UTF-8 it is 'C3\_A4'x
- This is what is known as its normalized form
- it can also be specified as "a" followed by an umlaut, in the so-called combining form. Then it would be '61\_CC\_88'x



# UVALID

- **UVALID will test CHAR for valid UTF-8 and WIDECHAR for valid UTF-16**
- **It will not raise an exception, but will return the index of the first bad (wide)char found - or zero if none were found**
- **So if X is declared as CHAR(10) VAR**

`x = '4b_c3_a4_73_65'x;`                     $\Rightarrow$                     `UVALID( x ) = 0`

`x = '4b_61_cc_88_73_65'x;`                     $\Rightarrow$                     `UVALID( x ) = 0`

`x = 'Käse'a;`                                     $\Rightarrow$                     `UVALID( x ) = 2`



# Exceptions

- **The other UTF-handling built-ins will raise ERROR if the UTF argument is invalid. The following ONCODEs are possible**
- **3018** Invalid UTF-8 data was detected.
- **3019** An invalid byte 2 in a UTF-8 character was detected.
- **3020** An invalid byte 3 in a UTF-8 character was detected.
- **3021** An invalid byte 4 in a UTF-8 character was detected.
- **3022** An incomplete UTF-8 character was detected.
- **3023** Invalid UTF-16 data was detected.
- **3024** An incomplete UTF-16 character was detected.



# ULENGTH

- **ULENGTH** returns the number of UTF characters in a string
- If the argument is **CHAR** bzw **WIDECHAR**, the count is of UTF-8 bzw UTF-16

`x = '4b_c3_a4_73_65'x;`       $\Rightarrow$       **ULENGTH( x ) = 4**

`x = '4b_61_cc_88_73_65'x;`       $\Rightarrow$       **ULENGTH( x ) = 5**

`x = 'Käse'a;`       $\Rightarrow$       **ULENGTH( x ) raises ERROR**



# UPOS

- UPOS returns the position or index of the nth UTF character in a string
- z.B. given `x = '4b_c3_a4_73_65'x`, UPOS will tell you it would be more clearly written as `'4b_c3a4_73_65'x` since
  - ▶ `UPOS( x, 1 )`                    1
  - ▶ `UPOS( x, 2 )`                    2
  - ▶ `UPOS( x, 3 )`                    4
  - ▶ `UPOS( x, 4 )`                    5
- `UPOS( x, n )` will return 0 if `n > ULENGTH(x)` or if `n <= 0`



# UWIDTH

- **UWIDTH** returns the width of the *n*th UTF character in a string
- z.B. given `x = '4b_c3a4_73_65'`
  - ▶ `UWIDTH( x, 1 )`            1
  - ▶ `UWIDTH( x, 2 )`            2
  - ▶ `UWIDTH( x, 3 )`            1
  - ▶ `UWIDTH( x, 4 )`            1
- `UWIDTH( x, n )` will return 0 if `n > ULENGTH(x)` or if `n <= 0`



# USUBSTR

- **USUBSTR is a UTF-sensitive SUBSTR**
- z.B. given  $x = '4b\_c3a4\_73\_65'x$

**USUBSTR( x, 2, 2 )      'c3a4\_73'x**

**USUBSTR( x, 2 )        'c3a4\_73\_65'x**



# USUBSTR

- Note that while **SUBSTR** will raise **STRINGRANGE** only if enabled
- **USUBSTR( x, i, j )** will always raise **ERROR** if **i** and **j** are invalid, d.h.

if  $i < 1$  , or

if  $j < 0$  , or

if  $(i+j-1) > \text{ULENGTH}(x)$

- **ONCODE = 3025** with message text: **USUBSTR** reference is invalid.





IBM Software Group

# Additional Performance Enhancements

**Rational.** software

[→ Go to IBM](#)

# PFPO instruction exploitation

- **PFPO = Perform Floating Point Operation**
- **inlines conversions between hex, binary and decimal float**
- **Requires ARCH(8)**
- **The operation code is in GPR 0**
- **The first operand is in FPR 0 (or in FPR 0 and 2), and the second operand is in FPR 4 (or in FPR 4 and 6)**



# PFPO

- For example, under ARCH(8), the conversion of a DFP FLOAT DEC(16) to a HEX FLOAT BIN(53) is now just 4 instructions - as is its reverse

000156	6840	D100		000013	LD	f4, B(, r13, 256)
00015A	C009	0109	0110	000013	I I LF	r0, F' 17367312'
000160	010A			000013	PFPO	
000162	6000	D0F8		000013	STD	f0, D(, r13, 248)
000166	6840	D0F8		000014	LD	f4, D(, r13, 248)
00016A	C009	0101	0901	000014	I I LF	r0, F' 16845057'
000170	010A			000014	PFPO	
000172	6000	D100		000014	STD	f0, B(, r13, 256)



## DFP math built-ins support

- **Previously with release 3.7 almost all DFP math built-ins were handled by converting the DFP to HEX, invoking the HEX math routine and converting back – because there were no LE DFP math routines**
- **This was expensive and could lead to incorrect results**
- **Now the conversions have been eliminated**
- **Requires library APAR PK72146**
- **This will be backfit to V3R7**



# GOFF support

- **The GOFF format supersedes the S/370 Object Module and Extended Object Module formats.**
- **It removes various limitations of the previous format (for example, 16 MB section size) and provides a number of useful extensions, including native z/OS support for long names and attributes.**
- **The COMMON and NOWRITABLE(PRV) options are not supported under the GOFF option**
- **The prelinker may not be used with GOFF objects**
- **NOGOFF is the default**



## Using 64-bit registers in 32-bit code

- **The HGPR option allows the compiler to do this**
- **NOHGPR is the default – and should be used in all CICS and DB2 code**
- **The HGPR option has 2 suboptions: PRESERVE and NOPRESERVE**
- **PRESERVE will cause the compiler to save the high-halves of the registers in the prologue code and restore them in the epilogue code – it should be used if your code is not being called from C/C++ or PL/I**
- **The default suboption for HGPR is NOPRESERVE**



## Using 64-bit registers in 32-bit code

- **Z. B., consider this program to sum the elements of a FIXED BIN(63) array**

```
4.0      test : proc(a) returns( fixed bin(63) byvalue );
5.0
6.0      dcl   jx           bin fixed (31,0);
7.0      dcl   a(0:*)      bin fixed (63,0) connected;
8.0      dcl   n63         bin fixed (63,0);
9.0
10.0     n63 = 0;
11.0     do jx = lbound(a) to hbound(a);
12.0         n63 = n63 + a(jx);
13.0     end;
14.0     return( n63 );
15.0     end;
```



## Using 64-bit registers in 32-bit code

- Under NOHGPR, the heart of this program is this 10-instruction loop

00006C			000011	@1L2	DS	0H
00006C	5835	2000	000012		L	r3, _shadow1(r5, r2, 0)
000070	5845	2004	000012		L	r4, _shadow1(r5, r2, 4)
000074	1E40		000012		ALR	r4, r0
000076	B998	003F	000012		ALCR	r3, r15
00007A	A71A	0001	000013		AHI	r1, H' 1'
00007E	1804		000012		LR	r0, r4
000080	5840	E008	000013		L	r4, _shadow2(, r14, 8)
000084	4150	5008	000000		LA	r5, #AMNESIA(, r5, 8)
000088	18F3		000012		LR	r15, r3
00008A	EC14	FFF1	C076	000013	CRJ	r1, r4, b' 1100' , @1L2



## Using 64-bit registers in 32-bit code

- Under NOHGPR, the heart of this program is this 5-instruction loop

000066				000011	@1L2	DS	0H
000066	EBF1	0003	000D	000012		SLLG	r15, r1, 3
00006C	A71A	0001		000013		AHI	r1, H' 1'
000070	E320	3008	0014	000013		LGF	r2, _shadow2(, r3, 8)
000076	E30F	E000	000A	000012		ALG	r0, _shadow1(r15, r14, 0)
00007C	EC12	FFF5	C076	000013		CRJ	r1, r2, b' 1100' , @1L2



## CMPAT(V3)

- **CMPAT(V3) is now supported**
- **It allows bounds to be greater than 2G**
- **But for now the aggregate size is still limited to that under CMPAT(V2)**
- **So, DCL A(4294967296:429496729) is ok**
- **But DCL A(4294967296) is not valid**
- **There is not much reason to use this yet – unless you want to prepare**





IBM Software Group

# SQL improvements

**Rational.** software

[→ Go to IBM](#)

## Easier multi-row fetch via DIMACROSS (for BMW)

- **The DIMACROSS option specifies a DIMENSION attribute on a structure but which will be removed from the structure and propagated to its children**
- **The DIMACROSS attribute is invalid if any of the children have the DIM attribute**
- **This is useful with SQL multi-row fetch**



## Easier multi-row fetch via DIMACROSS (for BMW)

- As an example, the following declares are equivalent

Dcl

```
1 a(10) di macross,  
  2 b,  
  2 c,  
  3 d,  
  3 e;
```

Dcl

```
1 a,  
  2 b(10),  
  2 c(10),  
  3 d,  
  3 e;
```



## Post-processor source views for SQL tools (for Huk)

- The **SOURCE**, **AFTERMACRO**, etc suboptions of the **TEST** option must now be specified as suboptions of the new **LISTVIEW** option
- This will permit these suboptions to be used without incurring any of the overhead of the **TEST** option
- **LISTVIEW(AFTERSQL)** is useful with some of the **SQL** tools
- **LISTVIEW(AFTERSQL) TEST(SEP)** will have the same effect under 3.8 as the two did when combined under 3.7



## Include-only in the SQL preprocessor (for Racon)

- **The SQL preprocessor will now support an INCONLY option**
- **This option will act similarly to the SQLONLY option in the MACRO preprocessor**
- **Under PP(SQL('INCONLY')) only EXEC SQL INCLUDE statements will be processed and all other EXEC SQL statements will be left as is**
- **So this would allow you to specify, z.B.,**

**PP( MACRO(INCONLY) SQL(INCONLY) MACRO SQL )**



## Alternate DDname for SQL includes (for IaCaixa)

- **The DDSQL option allows the user to specify a DDNAME that will be used instead of SYSLIB when searching for files named in EXEC SQL INCLUDE statements**
- **This eases the move from the SQL precompiler to the SQL preprocessor**
- **For compatibility, the default is DDSQL("") which means that the value in the DD option (which by default is SYSLIB) will control the search**



## DBCS support in the SQL preprocessor (for BTMU)

- **Only on z/OS for now**
- **Provides the same DBCS support in the SQL preprocessor as already provided in the MACRO and CICS preprocessors**





IBM Software Group

# Miscellaneous user requirements

**Rational.** software

[→ Go to IBM](#)

## Omit more trailing OPTIONAL arguments (for Racon)

- If the final parameters in an ENTRY declaration are declared as OPTIONAL, then the ENTRY may be invoked with those parameters completely omitted:
- it is not even necessary to specify the appropriate number of asterisks.
- So, for example, if an ENTRY is declared as having 5 parameters, of which the last 2 have the OPTIONAL attribute, then it may be invoked with 3, 4 or 5 arguments.
- This is now permitted with internal sub-procedures



## Support INCDIR under batch (for Wave)

- **This allows hfs files to be included in batch compilations**
- **Probably best used with the DFT(LOWERINC) option**
- **Affects only includes of the form %include x;**
- **For these includes,**
  - **an hfs file with the name x.inc will be sought first**
  - **If not found, x must be in the pds specified in the syslib dd**
- **For includes of the form %include dd(x); the member x must still be in the pds named by the specified dd**





## Learn more at:

- [IBM Rational software](#)
- [IBM Rational Software Delivery Platform](#)
- [Process and portfolio management](#)
- [Change and release management](#)
- [Quality management](#)
- [Architecture management](#)
- [Rational trial downloads](#)
- [developerWorks Rational](#)
- [IBM Rational TV](#)
- [IBM Rational Business Partners](#)

© Copyright IBM Corporation 2008. All rights reserved. The information contained in these materials is provided for informational purposes only, and is provided AS IS without warranty of any kind, express or implied. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, these materials. Nothing contained in these materials is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software. References in these materials to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. Product release dates and/or capabilities referenced in these materials may change at any time at IBM's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way. IBM, the IBM logo, the on-demand business logo, Rational, the Rational logo, and other IBM products and services are trademarks of the International Business Machines Corporation, in the United States, other countries or both. Other company, product, or service names may be trademarks or service marks of others.





IBM Software Group

# Enterprise PL/I for z/OS

Rational. software

[Go to IBM](#)

ハイバリューサポートセンター・サーバーソリューション事業開発  
コンサルティング・IT スペシャリスト 新井 寛

## Some historical perspective

- **Late 70's: OS PL/I V1**
- **80's: OS PL/I V2R1 – V2R3**
- **Early 90's: PL/I for MVS and VM**
- **All these releases are based on the same compiler technology and code base**
- **This code base is written in assembler**
- **Difficult to maintain and enhance**
- **Only PL/I for MVS is LE-enabled – in maintenance mode since 2000**



## Some historical perspective

- **New compiler was released in the fall of 1999 under the name VisualAge PL/I and had a second release in 2000**
- **It was renamed as Enterprise PL/I V3R1 in 2001**
- **There has been a release each year since then**
- **V3R8 was announced and released at the end of October 2008**
- **All these releases use a modern backend shared by C/C++**
- **The compiler front-end and preprocessors are written in PL/I**





IBM Software Group

# Application modernization

**Rational.** software



[Go to IBM](#)

# Modernization

- **The OS PL/I compiler had limits on how many variables it would try to optimize, how many statements were allowed in a program, etc**
- **Enterprise PL/I will optimize all variables – if you have enough REGION and enough CPU**
- **It also supports up to 1M lines of source in any one file and up to 2K different include files**
- **Many other program size limits have been increased or eliminated**
- **So, the compiler has been modernized, but your code can be too:**



# Significant new language features

- **31-digits supported in packed and zoned decimal**
- **FETCH from FETCH allowed**
- **DLL support**
- **Reentrant, writable static via compiler option**
- **Full USS support**
- **IEEE Float Binary fully supported**
- **IEEE Float Decimal (DFP) fully supported**



# Easier interoperability with C and Java

- **1-byte (FIXED BIN(7)) and 8-byte integers (FIXED BIN(63))**
- **UNSIGNED integers**
- **UTF-16 string support (WIDECHAR)**
- **Full support for null-terminated strings (VARYINGZ) for CHAR, GRAPHIC and WCHAR**
- **Enumeration type (ORDINAL) and user defined types and structures**
- **C-style function pointers**
- **BYVALUE parameters and arguments**



# Programmer productivity

- **Built-in high-speed XML parser**
- **Built-in generator of XML from PL/I structures**
- **Integrated DB2 and CICS preprocessors**
- **Compiler exit to allow user to change message severities**
- **Enablement of check conditions (e.g. SUBSCRIPTRANGE) via compiler option**
- **Better compile time detection of errors (such as storage overlays)**
- **Over 100 new built-in functions**



# Debug Tool integration

- **MFI or remotely via GUI and RDz**
- **Overlay hooks**
- **Side-file support**
- **Automonitor support**
- **Debug against the source or listing view**
- **Support for more classes of PL/I variables**
  - ▶ **BASED on BASED**
  - ▶ **BASED with REFER**



# More Tool integration

- **RDz**
  - ▶ **Language sensitive editor**
  - ▶ **Consistent language across platforms**
  - ▶ **Local or remote compiles**
  - ▶ **Remote debug**
  
- **Fault Analyzer**
  
- **File Manager**
  
- **All rely on compiler generated XML, XMI and ADATA files**





IBM Software Group

# Platform competitiveness and currency

**Rational.** software

[→ Go to IBM](#)

# Hardware exploitation

- **You've paid for the newer machines**
- **Enterprise PL/I is helping you get more value back**
- **No source changes needed**
- **Recompile with new release and/or new ARCH option**
- **Allianz has reported that their applications run 5% faster with Enterprise PL/I**
- **BMW examined the various options and reported that they can get a 10% improvement by recompiling with the right options**



## Exploitation of new instruction sets

- **Halfword-immediate instructions – V3R1**
- **IEEE Binary Floating Point – V3R2**
- **Branch-relative instructions – V3R5**
- **Long-displacement instructions – V3R5**
- **Extended-immediate instructions – V3R6**
- **IEEE Decimal Floating Point – V3R7**
- **General-instructions-extension (z10) instructions – V3R8**
  
- **The branch-relative set provides instructions which uses a signed 2-byte integer to specify a relative number of halfwords in a branch, etc**
  
- **Much simpler and more powerful than using a 4K offset from a base register**



## Exploitation of other hardware instructions

- **CLCLE** allows the compiler to inline compares of **CHARACTER** strings of varying length (compares that were done with subroutine calls under the old compiler)
- **MVCLE** makes it very easy to assign **CHARACTER** strings when padding with blanks may be needed
- **CLCLU** and **MVCLU** do the same for **WIDECHAR**
- **CVBG** and **CVDG** make it easier to inline conversions between **FIXED BIN** and **FIXED DEC** when they have large precision
- **PKA** and **UNPKA** make it easy to convert between **FIXED DEC** and **PICTURE** when they have large precision





## Learn more at:

- [IBM Rational software](#)
- [IBM Rational Software Delivery Platform](#)
- [Process and portfolio management](#)
- [Change and release management](#)
- [Quality management](#)
- [Architecture management](#)
- [Rational trial downloads](#)
- [developerWorks Rational](#)
- [IBM Rational TV](#)
- [IBM Rational Business Partners](#)

© Copyright IBM Corporation 2008. All rights reserved. The information contained in these materials is provided for informational purposes only, and is provided AS IS without warranty of any kind, express or implied. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, these materials. Nothing contained in these materials is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software. References in these materials to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. Product release dates and/or capabilities referenced in these materials may change at any time at IBM's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way. IBM, the IBM logo, the on-demand business logo, Rational, the Rational logo, and other IBM products and services are trademarks of the International Business Machines Corporation, in the United States, other countries or both. Other company, product, or service names may be trademarks or service marks of others.

