IBM Education Assistance for z/OS V2R2

Items: SIMD support
Delayed debug support
Element/Component: Language Environment

Agenda

- Trademarks
- Presentation Objectives
- Overview
- Usage & Invocation
- Interactions & Dependencies
- Migration & Coexistence Considerations
- Installation
- Presentation Summary
- Appendix

Trademarks

- See url http://www.ibm.com/legal/copytrade.shtml for a list of trademarks.

Presentation Objectives

- Explain the purpose of working on Language Environment SIMD support (vector register support) and delayed debug support.

Overview – SIMD Support

- Problem Statement / Need Addressed
  - Language Environment support is needed for applications exploiting SIMD benefits.

Overview – SIMD Support

- Solution
  - Changes are similar to those made when Additional Floating Point Register support was added
    - Add Vector Register Usage information in the PPA1
    - Machine State-type control blocks will be expanded to include a Vector Register save area
    - Save/Restore Vector Registers over initialization/termination points, interrupt points,and context switching points
    - New messages for vector-processing exceptions
    - Dump information support
    - printf()/scanf() support
- Benefit / Value
  - High level language application is enabled to exploit SIMD

Usage & Invocation – SIMD Support

- Vector Register
  - 32 vector registers
  - 16 bytes of each

Register	Volatility
VR 0 - 7	Not preserved
VR 8 - 15	Bytes 0 - 7 are preserved due to overlap with FPR 8 - 15, bytes 8 - 15 are not preserved
VR 16 - 23	Preserved
VR 24 - 31	Not preserved

Usage & Invocation – SIMD Support

- PPA1 Updates – Non-XPLINK PPA1s
  - Not enough room for vector register-related fields
  - “Extended Flag Field” added to Format 2 and Format 3 PPA1s
    - Present only when bit 14 (X’0002’) in Program Flags (PPA1 offset X'1C') is ON
    - 4 bytes in length
    - Located right before the Length of Name field
    - Flag bits indicate presence of corresponding optional areas
    - Optional areas will be located right before Extended Flag Field in a fixed order

Usage & Invocation – SIMD Support

- PPA1 Updates – Non-XPLINK PPA1s…
  - "Vector Register Area" is the first added optional area
    - Present only when Bit 0 (X'80') in Extended Flag Byte 1 is ON
    - 4 bytes in length

Offset	Field	Description
0	VR save area locator	Contains unsigned offset/16 of VR 16-23 save area within DSA
1	VR save bit mask	Indicates which VRs are saved and restored by the associated routine. Bit 0 - 7 indicate VR 16 - 23.
2	Reserved	Must be 0
3	Reserved	Must be 0

Usage & Invocation – SIMD Support

- PPA1 Updates – XPLINK PPA1s
  - "Vector Register Area" is a added optional area
    - Present only when Bit 2 (X'20') in PPA1 Flag 4 is ON
    - 8 bytes in length

Offset	Field	Description
0	VR Mask	Indicates which VRs are saved and restored by the associated routine. Bits 0 - 7 indicate VR 16 - 23.
1 - 3	Reserved	Must be 0
4 - 7	VR save area locator	Defines the location of the VR save area. Bits 0 - 3 indicates a GPR. Bits 4 - 31 indicate the unsigned offset.

Usage & Invocation – SIMD Support

- PPA1 Updates – CEEYPPAF
  - CWI used to locate a field in an XPLINK PPA1's optional area
  - Parameter 'opt_nam'
    - An integer indicating the requested PPA1 optional field
    - A value of 10 added to indicate requesting the Vector Register optional area

Usage & Invocation – SIMD Support

- PPA1 Updates – CEEPPA
  - Assembler macro used to generate PPA1/2
  - Two new keywords added:
    - VRSMASK: The Vector Registers save area bit mask field in hexadecimal format. Valid values for VRSMASK are 00 through FF.
    - VRSLOCR: The Vector Registers locator field in hexadecimal format. Valid values for VRSLOCR are 00 through FF.
  - VRSMASK and VRSLOCR must be specified together

Usage & Invocation – SIMD Support

- Machine State Block Update
  - Vector Registers Valid Flag – Bit 6 in offset X'B3', indicates contents of Vector Register Fields are valid
  - Vector Register Fields – offset X'200' to X'3FF', space for 32 Vector Registers
- Process Control Block Update
  - The flag bit CEEPCB_SIMD in PCB control block indicates if the Language Environment supports SIMD environment.
    - Bit 4 in offset X'54' in 31-bit PCB
    - Bit 4 in offset X'158' in 64-bit PCB

Usage & Invocation – SIMD Support

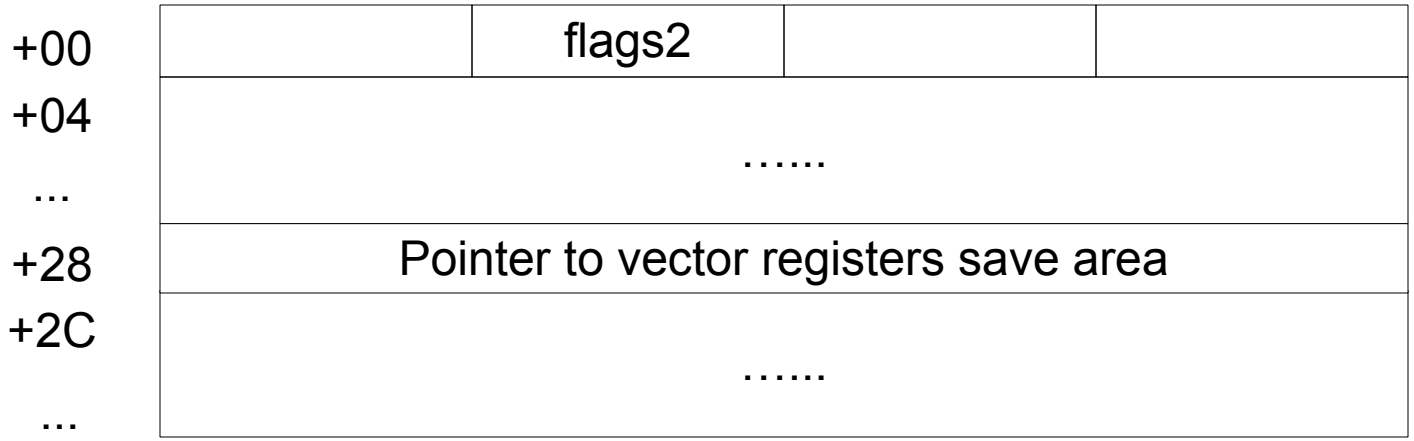
- Context Switching Support:
  - following pairs of functions support SIMD environment now:
    - setjmp() / longjmp()
    - _setjmp() / _longjmp()
    - sigsetjmp() / siglongjmp()
    - getcontext() / setcontext(), along with swapcontext() and makecontext()
    - __far_jump() can restore vector registers if a vector register save area is provided.
    - CEEGOTO can restore Vector registers if a vector register save area is provided.

Usage & Invocation – SIMD Support

- Context Switching Support ...
  - Size of context structures not change
    - jmp_buf, sigjmp_buf, ucontext_t, __jumpinfo_t and XPLINK Extended Format Label Variable
  - For jmp_buf, sigjmp_buf, ucontext_t and the associated C functions
    - No special consideration
  - For __jumpinfo_t and __far_jump()
    - __ji_vr_ext is added in __jumpinfo_t structure, which can be either 0 or an address to the vector register save area.

Usage & Invocation – SIMD Support

- For the XPLINK Extended Format Label Variable and CEEGOTO
  - XPLINK Extended Format Label Variable
    - The bit 2 of flags2 indicates if Vector registers save area exits
    - If above flag bit is true, then the address at X'28' points to the vector registers save area.



Usage & Invocation – SIMD Support

- Vector register save area should conform to the layout below, if it is provided for __far_jump() or CEEGOTO.

+00	Vector register save area version
+02	reserved
...	
+10	Vector registers 0 – 31 (16 bytes each)
...	
+20F	

Usage & Invocation – SIMD Support

- printf()/scanf() Support
  - New optional prefixes and new separator flag are added for vector type character and integer input/output conversions
  - New Prefixes:
    - Optional prefixes are vll/llv, vL/Lv, vl/lv, vh/hv, or v, resulting in 2, 4, 8, or 16 output values respectively
    - Each value is formatted according to its accompanying conversion specifier

Usage & Invocation – SIMD Support

- printf()/scanf() Support ...
  - Separator flag:
    - Separates each of the elements in a formatted vector type
    - The vector value is displayed in the following general form:

      Value1Cvalue2C...CValueN
    - …where C is an optional separator, specified as a flag character in the format string
    - Supported separators are , (comma), ; (semicolon), : (colon), and _ (underscore)
    - The default value for the separator character is a space (' '), unless the c conversion specifier is being used, in which case the default is no separator at all.

Usage & Invocation – SIMD Support

- printf()/scanf() Support... Example

```
#include <stdio.h>
vector signed char s8 = {'a','b',' ','d','e','f','g','h','i','j','k','l','m','!','o','p'};
vector unsigned short u16 = {1,2,3,4,5,6,7,8};
vector signed int s32 = {1, 2, 3, 99};

int main(void) {
    printf("s8 = %vc\n", s8);
    printf("s8 = %,vc\n", s8);
    printf("u16 = %vhu\n", u16);
    printf("s32 = %,2lvd\n", s32);
}
```

Output:

```
s8 = ab defghijklm!op
s8 = a,b, ,d,e,f,g,h,i,j,k,l,m,!,o,p
u16 = 1 2 3 4 5 6 7 8
s32 = 1, 2, 3,99
```

Usage & Invocation – SIMD Support

- New messages for vector-processing exceptions
  - Some new Language Environment messages are created for new vector-processing exceptions (PIC=1B)
    - CEE3235S The system detected a vector-processing exception (System Completion Code=0E0).
    - CEE3236S The system detected a vector-processing exception of IEEE invalid operation.
    - CEE3237S The system detected a vector-processing exception of IEEE division-by-zero.
    - CEE3238S The system detected a vector-processing exception of IEEE exponent-overflow.
    - CEE3239S The system detected a vector-processing exception of IEEE exponent-underflow.
    - CEE3240S The system detected a vector-processing exception of IEEE inexact.
  - These vector-processing exceptions are also mapped to SIGFPE, so user can handle it in a signal handler

Usage & Invocation – SIMD Support

- CEEDUMP support
  - Vector registers may be dumped into three sections
    1. Registers on entry to CEE3DMP
    2. Parameters, registers, and variables for active routines
    3. Condition information for active routines
  - All vector registers are dumped in section 1 and 3 if VR in use
  - Non-volatile vector registers used by an active routine are dumped in section 2
- System dump support
  - VERBEXIT LEDATA with parameter 'CM'
    - Vector registers are dumped with MCH

Interactions & Dependencies – SIMD Support

- Software Dependencies
  - None
- Hardware Dependencies
  - SIMD facility enabled
- Exploiters
  - None

Migration & Coexistence Considerations – SIMD Support

- As of z/OS V2R1 (with APAR PI20843) and z/OS V2R2, XL C/C++ runtime supports new specifiers for the fprintf and fscanf families of functions for vector types. The newly introduced specifiers include separator flags “,” (comma), “;” (semicolon), “:” (colon), and “_” (underscore) and optional prefixes “v”, “vh”, “hv”, “vl”, “lv”, “vll”, “llv”, “vL” and “Lv”. If a percent sign (%) is followed by one of these character strings, which had no meaning under previous releases, the runtime could interpret the data as a vector type specifier. Treatment of this condition is undefined and the behavior could be unexpected.
- This change is similar to Language Environment decimal floating point size specifiers support in z/OS V1R9, which introduced “D”, “DD”, and “H” specifiers.

Installation – SIMD Support

- Available as an APAR on V2R1: PI12412
  - HLE7790 PTF UI90017
  - JLE779J PTF UI90018
- Additional APAR on V2R1: PI20843
  - HLE7790 PTF UI26464

Overview – Delayed Debug

- **Problem Statement / Need Addressed**
  - Debuggers, such as Debug Tool, are unable to delay start up of a debug session until a specific C/C++ function is entered.

- **Solution**
  - New interfaces were added to Language Environment that can be used to delay start up of a debug session until a specific C/C++ function is entered.

- **Benefit / Value**
  - The overhead of running with a debugger is avoided until the specific C/C++ function is entered.

Usage & Invocation – Delayed Debug

CEL4RGSR – AMODE 31 Enable service routine:

Syntax

```
CEL4RGSR ( func_code, sr_addr, sr_workarea,
[fc] )
   INT4         *func_code;
   POINTER      *sr_addr;
   POINTER      *sr_workarea;
   FEED_BACK    *fc;
```

Call this CWI interface as follows:

```
L     R15,CEECAALEOV-CEECAA(,R12)
L     R15,104(,R15)
BALR  R14,R15
```

Usage & Invocation – Delayed Debug

__CEL4RGSR - Enable service routine for AMODE 64 applications:

Syntax

```
#include<__le_cwi.h>
void __CEL4RGSR( int32_t     *   func_code,
                 Void        **  sr_addr,
                 Void        **  sr_workarea,
                 _FEED_BACK  *   fc);
```

Usage & Invocation – Delayed Debug

Parameters

func_code (input)
The function code that is used to indicate the service routine should be enabled or disabled for either CEL4CASR, CELHCASR or __CEL4CASR.

Possible values are:

0 Enable a service routine with specified work area for CEL4CASR if it is never enabled or has already been disabled. AMODE 31 only.
1 Disable a service routine with specified work area for CEL4CASR. AMODE 31 only.
2 Enable a XPLINK service routine with specified work area for CELHCASR/__CEL4CASR if it is never enabled or has already been disabled.
3 Disable a XPLINK service routine with specified work area for CELHCASR/__CEL4CASR.

Usage & Invocation – Delayed Debug

sr_addr (input)
The address of a service routine that is to be enabled. Language Environment does not check the value of this parameter.

This routine is called whenever a function or program calls the corresponding CEL4CASR/CELHCASR CWI in AMODE 31 or __CEL4CASR CWI in AMODE 64 in its prolog code. A common use case for this is for logging purposes or for the Debug Tool deferred debugging functionality.

For CEL4RGSR, the service routine can be a non-XPLINK or XPLINK, AMODE 31 routine, with no writable static. Non-XPLINK service routine(s) will only be called by CEL4CASR, and XPLINK service routine(s) will only be called by CELHCASR.
For __CEL4RGSR, the service routine must be a AMODE 64 routine with no writable static. The service routine(s) will only be called by __CEL4CASR.

The parameters to the service routine are:

Parameter 1 Entry point address of the routine that is about to be entered.
Parameter 2 Pointer to the routine name.
Parameter 3 Fullword containing the length of routine name field.
Parameter 4 Pointer to the work area provided when the service routine was enabled.

Usage & Invocation – Delayed Debug

sr_workarea (input)
The address of a work area that is to be passed to the service routine each time it is called.

fc (output/optional)
A feedback code to indicate the result of this call; possible values are:

CEE000 Severity 0
Msg_No N/A
Message The service completed successfully.
CEE3AB Severity 2
Msg_No 3403
Message The input parameter func_code is incorrect.

Usage & Invocation – Delayed Debug

CEE3AC Severity 1
Msg_No 3404
Message The service routine with specified work area has already been enabled.
CEE3AD Severity 1
Msg_No 3405
Message The service routine with specified work area does not exist or has already been disabled.
CEE3AE Severity 3
Msg_No 3406
Message The service failed to allocate storage.
CEE3AF Severity 3
Msg_No 3407
Message The service failed to free storage.
CEE3AG Severity 2
Msg_No 3408
Message The maximum number of unique pairs of service routines with specified work areas has been reached.

Usage & Invocation – Delayed Debug

CEL4CASR -
Call service routine(s) for AMODE 31, non-XPLINK applications:

Syntax

```
CEL4CASR ( rtn_addr, rtn_name, rtn_nlen )
   POINTER    rtn_addr;
   POINTER    rtn_name;
   INT4       rtn_nlen;
```

Call this CWI interface as follows:

```
L      R15,CEECAALEOV-CEECAA(,R12)
L      R15,108(,R15)
BALR   R14,R15
```

Usage & Invocation – Delayed Debug

CELHCASR -
Call service routine(s) for AMODE 31, XPLINK applications:

Syntax

```
CELHCASR ( rtn_addr, rtn_name, rtn_nlen )
   POINTER     rtn_addr;
   POINTER     rtn_name;
   INT4        rtn_nlen;
```

Call this CWI interface as follows:

```
L      R6,CEECAALEOV-CEECAA(,R12)
L      R6,112(,R6)
BASR   R7,R6
NOP    0,0
```

Usage & Invocation – Delayed Debug

__CEL4CASR - Call service routine(s) for AMODE 64 applications:

Syntax

```
#include<__le_cwi.h>
void __CEL4CASR( void *   rtn_addr,
                 void *   rtn_name,
                 int32_t rtn_nlen);
```

This CWI can also be called as follows:

```
LLGT R6,PSALAA-PSA(,R0)
LG   R6,CEELAA_LCA64-CEELAA(,R6)
LG   R6,CEELCA_CAA-CEELCA(,R6)
LG   R6,CEECAACEL_HPXV_B-CECAA(,R6)
LMG  R5,R6,992(,R6)
BASR R7,R6
NOPR 0
```

Usage & Invocation – Delayed Debug

Parameters

rtn_addr (input)
The entry point address of the routine that is about to be entered. This parameter is passed to the service routine enabled via CEL4RGSR/__CEL4RGSR.

rtn_name (input)
The address of the routine name that is about to be entered. This parameter is passed to the service routine enabled via CEL4RGSR/__CEL4RGSR.

rtn_nlen (input)
The length of the routine name. This parameter is passed to the service routine enabled via CEL4RGSR/__CEL4RGSR.

Usage & Invocation – Delayed Debug

Implementing delayed debug startup when a C/C++ function is called

- Debugger uses the CWI CEL4RGSR to enable a service routine
  - From the assembler user exit
  - From a CICS transaction (for nonXPLINK transactions only).
  - Or some other way
- Service routine starts up the debugger when a specific routine name or address is passed into it.
- C/C++ compiler generates calls to the CWI CEL4CASR in nonXPLINK functions and the CWI CELHCASR in XPLINK functions when a certain compile option is specified.

Installation – Delayed Debug

- Available on V2R1 as APAR PI12415:
  - HLE7790 PTF UI19022

Presentation Summary

- The following Language Environment items have been explained:
  - SIMD Support
  - Delayed Debug

Appendix

- Publications
  - *z/OS XL C/C++ Runtime Library Reference* (SC14-7314)
  - *z/OS XL C/C++ Compiler and Runtime Migration Guide for the Application Programmer* (GC14-7306)
  - *z/OS Language Environment Debugging Guide* (GA32-0908)
  - *z/OS Language Environment Programming Guide* (SA38-0682)
  - *z/OS Language Environment Vendor Interfaces* (SA38-0688)
  - *z/OS Language Environment Runtime Messages* (SA38-0686)