IBM Education Assistance for z/OS V2R2

Item: DBX SIMD Support
Element/Component: z/OS UNIX System Services DBX

Material current as of May 2015

Agenda

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

Trademarks

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

Presentation Objectives

- Vector variables display and change
- Vector registers display and change
- Conditional stop/trace/stopi/tracei using vector variables
- Conditional stop/trace/stopi/tracei using vector registers
- New messages

Overview

- This new support allows users to debug C or C++ programs that utilizing vector variables and SIMD registers/instructions
- Problem Statement / Need Addressed
  - The user will want to debug variables that are declared using the new vector type and access the vector registers
  - Vector instructions also should be displayed correctly
  - The user will set breakpoints and trace with conditions containing vector variables/registers
- Solution
  - DBX should display vector variables, registers and instructions in debugging.
  - Vector variables/registers should work when used as condition of breakpoints and trace
- Benefit/Value
  - DBX will help user to exploit SIMD capacity on UNIX System Services

Usage & Invocation: vector variables/registers display and change

example.c:

```
 1 void main()
 2 {
 3   vector signed int int_1={1,2,3,4};
 4   vector signed int int_2;
 5   vector unsigned short short_1={11,12,13,14,15,16,17,18};
 6   vector bool long long long_1={22,23};
 7   vector unsigned char char_1={'a','b','c','d'};
 8   vector signed long long array_1[2]={{31,32},{33,34}};
 9   vector double z={0.1314, 897655.9};
10   vector unsigned char* pChar;
11   vector signed long long* pLong;
12
13   int c=2014;
14
15   c++;
16   int_2[3]=c;
17   pChar = &char_1;
18   pLong = &array_1[1];
19
20   return;
21 }
```

c99 -g -qvector -qarch=11 -Wc,'FLOAT(IEEE)' example.c

Usage & Invocation: Display vector variables

```
(dbx64) stop in main
[1] stop in 'void main()'       File ALPS1054:/u/dbxteam/example.c, Line 3.
(dbx64) c
[1] stopped in main at line 3 in file "example.c"  ($t1)
■    3      vector signed int int_1={1,2,3,4};
(dbx64) list
     4      vector signed int int_2;
     5      vector unsigned short short_1={11,12,13,14,15,16,17,18};
     6      vector bool long long long_1={22,23};
     7      vector unsigned char char_1={'a','b','c','d'};
     8      vector signed long long array_1[2]={{31,32},{33,34}};
     9      vector double z={0.1314, 897655.9};
    10      vector unsigned char* pChar;
    11      vector signed long long* pLong;
(dbx64) st at 20
[2] stop at "example.c":20
(dbx64) c
[2] stopped in main at line 20 in file "example.c"  ($t1)
   20      return;
(dbx64) print int_1
(1, 2, 3, 4)
(dbx64) print long_1,char_1
(22, 23) "abcd"
(dbx64) print array_1
((31, 32), (33, 34))
(dbx64) whatis short_1
vector unsigned short short_1;
(dbx64) p *pChar
"abcd"
(dbx64) p *pLong
(33, 34)
```

Usage & Invocation: Display vector registers

```
(dbx64) unset $novregs
(dbx64) registers
  $r0: 0x00000000000007df   $r1: 0x00000000267e0334   $r2: 0x000000008636b798
  $r3: 0x0000000000000002   $r4: 0x0000000085d52d82   $r5: 0x00000000267c4a28
  $r6: 0x00000000267c4b9c   $r7: 0x00000000267c4098   $r8: 0x0000000000000030
  $r9: 0x0000000080000000  $r10: 0x000000008636b6ca  $r11: 0x0000000085d52c90
 $r12: 0x00000000267db188   $sp: 0x00000000267e0248  $r14: 0x00000000267e0248
 $r15: 0x00000000a67c4928
$psw0: 0x00000000078d2400 $psw1: 0x00000000a67c4a02   $pc: 0x00000000267c4a02

   PSW: 078d2400 a67c4a02
        Instruction address: 0x267c4a02
        Condition code: 2

FDBX0552: unset $noflregs to view floating point registers.
FDBX0553: unset $noflbregs to view IEEE floating point registers.
FDBX0557: unset $nofldregs to view decimal floating point registers.

  vreg   value in hex
   $vr0: 0x81828384 00000000 00000000 00000000
   $vr1: 0x41120dd7 50429b6d 00000000 00000000
   $vr2: 0x40c90fda a22168c2 00000000 00000000
   $vr3: 0x4116a09e 667f3bcd 00000000 00000000
   $vr4: 0x40517cc1 b727220b 00000000 00000000
   $vr5: 0x40b504f3 33f9de65 00000000 00000000
   $vr6: 0x40a2f983 6e4e4415 00000000 00000000
   $vr7: 0x00000000 00000000 00000000 00000000
   $vr8: 0x00000000 00000000 00000000 00000000
   $vr9: 0x00000000 00000000 00000000 00000000
  $vr10: 0x00000000 00000000 00000000 00000000
  $vr11: 0x00000000 00000000 00000000 00000000
  $vr12: 0x00000000 00000000 00000000 00000000
(next page)
```

Usage & Invocation: Display vector registers

```
  $vr13: 0x00000000 00000000 00000000 00000000
  $vr14: 0x00000000 00000000 00000000 00000000
  $vr15: 0x00000000 00000000 00000000 00000000
  $vr16: 0x00000000 00000000 00000000 00000000
  $vr17: 0x00000000 00000000 00000000 00000000
  $vr18: 0x00000000 00000000 00000000 00000000
  $vr19: 0x00000000 00000000 00000000 00000000
  $vr20: 0x00000000 00000000 00000000 00000000
  $vr21: 0x00000000 00000000 00000000 00000000
  $vr22: 0x00000000 00000000 00000000 00000000
  $vr23: 0x00000000 00000000 00000000 00000000
  $vr24: 0x00000000 00000000 00000000 00000000
  $vr25: 0x00000000 00000000 00000000 00000000
  $vr26: 0x00000000 00000000 00000000 00000000
  $vr27: 0x00000000 00000000 00000000 00000000
  $vr28: 0x00000000 00000000 00000000 00000000
  $vr29: 0x00000000 00000000 00000000 00000000
  $vr30: 0x00000000 00000000 00000000 00000000
  $vr31: 0x00000000 00000000 00000000 00000000
in main at line 15 in file "example.c"  ($t1)
0x267c4a02 (main+0xda)  4400c1ac     EX      0,428(,R12)
(dbx64) print $vr0
(-2122153084,0,0,0)
(dbx64) print $vr0s
(-32382,-31868,0,0,0,0,0,0)
(dbx64) print $vr0c
('a','b','c','d','\0','\0','\0','\0','\0','\0','\0','\0','\0','\0','\0','\0')
(dbx64) print $vr0c[3]
'd'
(dbx64) print $vr0c[1..2]
[1] = 'b'
[2] = 'c'
```

Usage & Invocation: Change vector variables

```
(dbx64) assign int_2=int_1
(dbx64) print int_2
(1, 2, 3, 4)
(dbx64) assign z[0]=0
(dbx64) p z
(0.000000, 897655.900000)
(dbx64) assign short_1=int_1
(dbx64) p short_1
(0, 1, 0, 2, 0, 3, 0, 4)
(dbx64) assign long_1=short_1
(dbx64) set $hexints
(dbx64) p long_1
(0x100000002, 0x300000004)
(dbx64) unset $hexints
(dbx64) assign pLong=&long_1
(dbx64) p *pLong
(4294967298, 12884901892)
(dbx64) assign (*pChar)[0]='f'
(dbx64) p char_1
"fbcd"
(dbx64) assign array_1[0]=long_1
(dbx64) p array_1
((4294967298, 12884901892), (33, 34))
(dbx64) assign int_1={0,0,0,0}
assign int_1={0,0,0,0}
             ^ syntax error
(dbx64) assign int_1=0
FDBX0056: assign non-composite to composite
```

Usage & Invocation: Change vector registers

```
(dbx64) set $hexints
(dbx64) p $vr2
(0x40c90fda,0xa22168c2,0x00000000,0x00000000)
(dbx64) assign $vr2[0]=0x10
(dbx64) assign $vr2[1]=0x11
(dbx64) assign $vr2[2]=0x12
(dbx64) assign $vr2[3]=0x13
(dbx64) p $vr2
(0x00000010,0x00000011,0x00000012,0x00000013)
(dbx64) assign $vr1=$vr2
(dbx64) p $vr1
(0x00000010,0x00000011,0x00000012,0x00000013)
(dbx64) assign $vr0[0]=1024*2+1
(dbx64) p $vr0[0]
0x00000801
(dbx64) assign $vr31c[0]='a'
(dbx64) assign $vr31c[1]='g'
(dbx64) assign $vr31c[2]=0x88
(dbx64) p $vr31c
('a','g','h','\0','\0','\0','\0','\0','\0','\0','\0','\0','\0','\0','\0','\0')
(dbx64) assign $vr29s[1]=0xffff
(dbx64) p $vr29s
(0x0000,0xffff,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000)
(dbx64) assign $vr2x[0]=1
FDBX0614: "$vr2x" is not defined
```

Usage & Invocation: conditional stop/trace/stopi/tracei

example2.c:

```
 1  void subFunc1(vector unsigned short input);
 2  vector signed int subFunc2();
 3
 4  void main()
 5  {
 6    vector signed int mainVar_1={1,1,1,1};
 7    vector unsigned short mainVar_2={2,2,2,2};
 8    vector unsigned long long mainVar_3[2]={{30,31},{56,57}};
 9
10    mainVar_2[1]++;
11    subFunc1(mainVar_2);
12    mainVar_1 = subFunc2();
13    mainVar_3[0][0]--;
14    mainVar_3[1][1] = 99;
15    return;
16  }
17
18  void subFunc1(vector unsigned short input)
19  {
20    input[7]=7;
21    return;
22  }
23
24  vector signed int subFunc2()
25  {
26    vector signed int rt={5,6,7,8};
27    return rt;
28  }
```

```
c99 -g -qphsinfo -qvector -qarch=11 -Wc,'FLOAT(IEEE)' example2.c
```

Usage & Invocation: conditional stop/trace/stopi/tracei

```
(dbx64) st at 11
[1] stop at "example2.c":11
(dbx64) trace if mainVar_1[0]>0
[2] trace if mainVar_1[0] > 0
(dbx64) c
FDBX0304: trace in example2.c ($t1):    7     vector unsigned short mainVar_2={2,2,2,2};
FDBX0304: trace in example2.c ($t1):    8     vector unsigned long long mainVar_3[2]={{3
0,31},{56,57}};
FDBX0304: trace in example2.c ($t1):   10     mainVar_2[1]++;
FDBX0304: trace in example2.c ($t1):   11     subFunc1(mainVar_2);
[1] stopped in main at line 11 in file "example2.c"  ($t1)
   11     subFunc1(mainVar_2);
(dbx64) stop in subFunc1 if mainVar_2[3]==2
[3] stop if mainVar_2[3] = 2 in 'void subFunc1(vector unsigned short input)'    File ALP
S4025:/dbxteam/STE-example/example2.c, Line 20.
(dbx64) c
FDBX0304: trace in example2.c ($t1):   20     input[7]=7;
[3] stopped in subFunc1 at line 20 in file "example2.c"  ($t1)
   20     input[7]=7;
```

Usage & Invocation: conditional stop/trace/stopi/tracei

```
(dbx64) stopi if $vr0[1]<>100
[4] stopi if $vr0[1] <> 100
(dbx64) c
stopped in subFunc1 at 0x267c4980 ($t1)
0x267c4980 (subFunc1+0x58)  5810d0e0     L       R1,224(,R13)
(dbx64) p $vr0[1]
131074
(dbx64) delete all
(dbx64) stop if mainVar_3[0][0]!=30
[6] stop if mainVar_3[0][0] <> 30
(dbx64) c
Stopped in main at line 14 in file "example2.c"  ($t1)
   14     mainVar_3[1][1] = 99;
(dbx64) tracei if mainVar_2[4]=0
FDBX0305: tracei ($t1): 0x267c5b7a (main+0x142)  4400c1ac     EX      0,428(,R12)
[6] tracei if mainVar_2[0x4] = 0x0
(dbx64) c
FDBX0305: tracei ($t1): 0x267c5b7e (main+0x146)  4110d128     LA      R1,296(,R13)
FDBX0305: tracei ($t1): 0x267c5b82 (main+0x14a)  e54c10000000 MVHI    0(R1),0
FDBX0305: tracei ($t1): 0x267c5b88 (main+0x150)  e54c10040063 MVHI    4(R1),99
FDBX0305: tracei ($t1): 0x267c5b8e (main+0x156)  4400c1ac     EX      0,428(,R12)
stopped in main at line 15 in file "example2.c"  ($t1)
   15   return;
```

Interactions & Dependencies

- Software Dependencies
  - Compiler/CDA
    - Generate binaries code for SIMD support
    - Generate debug info for SIMD support
    - Use new CDA consumer interfaces
  - HLASM
    - SIMD instruction disassembly
  - USS Kernel
    - New ptrace function codes to access vector register
- Hardware Dependencies
  - The Vector Extension Facility
- Exploiters
  - None

Migration & Coexistence Considerations

- Code must be compiled with following option to get SIMD support:

```
-qvector -qarch=11 -Wc,'FLOAT(IEEE)'
```

Presentation Summary

- DBX can debug vector registers or vector variables now. Support included:
  - Vector variables display and change
  - Vector registers display and change
  - Conditional stop/trace/stopi/tracei using vector variables
  - Conditional stop/trace/stopi/tracei using vector registers
  - Change vector registers

Appendix

- SA23-2280 *zOS UNIX System Services Command Reference*
- SA23-2284 *zOS UNIX System Services Messages and Codes*
- SA23-2282 *zOS UNIX System Services Programming Tools*
- SA23-2279 *zOS UNIX System Services User's Guide*