RENESAS ツールニュース

RENESAS TOOL NEWS 2009年02月20日：090220/tn1

SuperH RISC engineファミリ C/C++コンパイラパッケージ V.9 ご使用上のお願い

SuperH RISC engine ファミリ C/C++コンパイラパッケージの使用上の注意事項を連絡します。

- 関数呼び出しを行う関数で割り込みが発生する場合の注意事項(SHC-0078)
- ビット単位の補数演算（「~」演算）を複数回適用した結果を特定の型の変数に代入する場合の注意事項(SHC-0079)

---

1. 関数呼び出しを行う関数で割り込みが発生する場合の注意事項(SHC-0078)

該当バージョン：
V.9.00 Release 00 ～ V.9.02 Release 00

現象：
関数呼び出しを行う関数で、スタック領域を解放してからその関数がリターンするまでに割り込みが発生すると、スタック領域の値が書き換えられ、それらの正しくない値を参照する場合があります。

発生条件：
以下の条件をすべて満たす時に発生する場合があります。
(1) optimize=1オプションを使用している。

(2) 関数呼び出しを行う関数がある。

(3) (2)項の関数内で、スタック領域の確保および解放を行っている。

(4) (2)項の関数内で、R8からR14までのいずれのレジスタも参照していない。

(5) (2)項の関数内で、以下のいずれかの命令が少なくとも1つ存在する。

注： nは1から7までのいずれかである
- MOV R15,Rn
- MOV.L R15,@-Rn
- MOV.L R15,@Rn
- MOV.L R15,@(R0,Rn)
- MOV.L R15,@(disp,Rn)
- MOV.L R15,@(disp12,Rn)
- ADD R15,Rn

(6) (2)項の関数内で、確保されたスタック領域に積まれた仮引数または自動変数を参照している。

(7) (2)項の関数は、RTS命令でリターンする。

(8) (2)項の関数内でスタック領域を解放してから、(7)項のRTS命令までの間に割り込みが発生する。

(9) (7)項のRTS命令の遅延スロットに、(3)項で解放したスタック領域からR0またはFR0へのロード命令が入っている。
注： この条件になるのは、(2)項の関数内のreturn文のオペランドにパラメータまたはローカル変数を指定した場合です。ただし、コンパイラがV.9.00 Relase 00 ～ Release 04A のいずれかで、return文のオペランドがスカラ型変数である場合は除く。

発生例：

```
------------------------------------------------------------
int *p;
int func()
{
   int array[2]={0};

   sub();            // 発生条件(2)の関数により呼び出される関数
   p = &(array[1]);  // 発生条件(6)
   return (*p);
}
------------------------------------------------------------
```

コンパイル結果

```
------------------------------------------------------------


_func:
        STS.L   PR,@-R15      ;
```

```
ADD     #-8,R15         ; 発生条件(3) スタックを確保
MOV.L   L11+2,R5        ; L12
MOV.L   L11+6,R4        ; _sub
MOV.L   @R5,R1
MOV.L   @(4,R5),R2
MOV.L   R1,@R15         ; array[]
JSR     @R4
MOV.L   R2,@(4,R15)     ; array[]
MOV     R15,R7          ; 発生条件(5)
                        ; R7 = 解放前のスタックポインタの値
MOV.L   L11+10,R6       ; _p
ADD     #4,R7
MOV.L   R7,@R6          ; p
ADD     #8,R15          ; 発生条件(3) スタックを解放 A
LDS.L   @R15+,PR
RTS                     ; 発生条件(7) B
MOV.L   @R7,R0          ; *(p) 発生条件(9)
```

注：AからBの間に割り込みが発生した場合、発生条件（8）に該当する。

回避策：

以下のいずれかの方法で回避してください。

(1) 発生条件(2)項の関数を呼び出す前後で、割り込みを禁止する。

(2) optimize=1の代わりに、optimize=0またはoptimize=debug_only オプションを使用する。

2. ビット単位の補数演算（「~」演算）を複数回適用した結果を特定の型の変数に代入する場合の注意事項(SHC-0079)

該当バージョン：

V.9.00 Release 00 ～ V.9.02 Release 00

現象：
代入文があり、右辺が 「~」(ビット反転演算) を複数回適用した結果で、
左辺がsigned char, unsigned char, signed short または unsigned short型
変数であるとき、左辺の変数をその後使用すると、代入文での型変換を適用
せず、代入文右辺の値を使用する場合があります。

発生条件：
以下の条件をすべて満たす時に発生する場合があります。
(1) 代入文があり、その左辺が以下の型のいずれかの内部変数、または外部
変数である。
signed char
unsigned char
signed short
unsigned short

(2) 関数内に、変数又は式に対する「~」演算が存在している。

(3) (2)項の「~」演算の結果を再度「~」演算し、(1)の変数に代入している。

(4) (2)項と(3)項の「~」演算の間で、(2)項の「~」演算の結果を変更して
いない。

例1

```
---------------------------------------------------------------
unsigned char t = ~~(X+a);      // 発生条件(1)、(2)、(3)および(4)
---------------------------------------------------------------
```

例2

```
---------------------------------------------------------------
unsigned char t = ~(~X);      // 発生条件(1)、(2)、(3)および(4)
---------------------------------------------------------------
```

例3

```
---------------------------------------------------------------
unsigned char t;
unsigned char temp;
temp = ~X;                  // 発生条件(2)
     :                    // 発生条件(4) (tempの変更なし)
t = ~temp;                  // 発生条件(1)および(3)
---------------------------------------------------------------
```

(5) 以下のいずれかを満たしている。
   a. (3)項の「~」演算を適用する前の式の値は、(1)項の代入先の
      変数の型で表現できない値である。
   b. (3)項の「~」演算を適用した結果は、(1)項の代入先の変数の
      型で表現できない値である。
   「変数の型で表現できない値」とは以下を指す。
      signed char型：　-129以下 または 128以上
      unsigned char型：負の値 または 256以上
      signed short型：　-32,769以下 または 32,768以上
      unsigned short型：負の値 または 65,536以上

(6) (1)項の代入文より後で、(1)項の代入先変数を使用している。
   ただし、(1)項の代入先変数を、(1)項の変数と同じか、
   もしくはそれより小さいデータサイズの型の変数に代入する代入文の
   場合はこの条件には含まない。

発生例：

```
------------------------------------------------------------------
int X = -1;              // 発生条件(3)
int Y;

func()
{
   unsigned char t = ~~X; // 発生条件(1)および(2)
   Y = t;                 // 発生条件(4)
}
------------------------------------------------------------------
```

コンパイル結果

```
------------------------------------------------------------------
    MOV.L     L11+2,R1   ; _X
    MOV.L     L11+6,R4   ; _Y
    MOV.L     @R1,R2     ; X
    RTS
    MOV.L     R2,@R4     ; Y  Y=255が正しいが、Y=-1を設定
------------------------------------------------------------------
```

回避策：

以下のいずれかの方法で回避してください。

(1) 発生条件(2)に該当する「~」演算を適用した結果を、volatile修飾変数に
   一旦代入し、その変数を使用する。

   例

```
-------------------------------------------
   volatile unsigned char temp = ~X;
   unsigned char t = ~temp;
   Y = t;
-------------------------------------------
```

(2) 発生条件(2)および(3)に該当する、不要な「~」演算を削除する。

例

```
-------------------------------------------
   unsigned char t = X;  // 「~~」を削除。
   Y = t;
-------------------------------------------
```

3. 恒久対策

本内容は、SuperH RISC engine ファミリ C/C++コンパイラパッケージ V.9.03 Release 00で改修する予定です。
