RENESAS ツールニュース

RENESAS TOOL NEWS 2007年03月01日：070301/tn3

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 V.9の使用上の注意事項 3件を連絡します。

1. 該当製品

SuperH RISC engine ファミリ C/C++コンパイラパッケージ
V.9.00 Release 00 ～ V.9.00 Release 04A

2. 内容

2.1 ポインタ変数もしくは配列の添え字のインクリメントを行った場合の注意事項

(SHC-0069)
ポインタ変数もしくは配列の添え字をインクリメントした後に、制御式で
そのポインタ変数もしくは配列のアドレスを使用した場合、インクリメントする
前のアドレスを使用することがあります。

発生条件：以下の条件をすべて満たした時に発生することがあります。
(1) optimize=1オプションを使用している。
(2) cpu=sh2aまたはcpu=sh2afpuオプションを使用している。
(3) ポインタ変数が示す実体への代入式があり、かつ代入後にポインタ変数を
インクリメントしている。もしくは配列要素を左辺に持つ代入式があり、
かつ代入後にその配列の添字をインクリメントしている。
(4) (3)のポインタ変数または配列のアドレスをif文またはループの制御式で
使用している。

例：

```
------------------------------------------------
int ST[100],ST2[100];
void func(void)
{
  int *p=ST, *q=ST2;
  do {
```

```
      *p++=*q;          // 発生条件(3)
      q++;
   } while (p < ST+100);  // 発生条件(4)
}
-------------------------------------------
L11:
     MOV.L      @R5+,R0
     CMP/HS     R4,R6      ; pをインクリメントする前にpを
                           ; 制御式で参照
     BF/S       L11
     MOV.L      R0,@R6+
     RTS/N
-------------------------------------------
```

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(3)のインクリメントの直後に、nop()組み込み関数を追加する。
(2) 発生条件(3)のインクリメントの直後で、(3)のポインタ変数あるいは
　配列要素をvolatile修飾した外部変数に代入する。
(3) optimize=0オプションを使用する。

2.2 ループ帰納変数を含む式をループ内で複数記述した場合の注意事項(SHC-0070)

繰り返し文内において、ループ帰納変数を含む同じ演算式を複数記述している
場合、その演算式の結果が間違っている場合があります。

発生条件：以下の条件をすべて満たした時に発生することがあります。
(1) optimize=1オプションを使用している。
(2) ループが存在する。
(3) (2)のループ内に、ループ帰納変数を含み、かつ内容が同じ演算式(例えば、
　i*4 と 4*i は同じ内容と見なす)が複数存在する。
(4) (3)の演算は、加算、減算、乗算、マイナス、または左シフトのいずれか
　である。
(5) (3)の演算式のうち少なくとも1つは代入式の右辺に含まれる。かつ代入式の
　左辺の型のサイズは演算式の型のサイズより小さい。
(6) (5)の代入式において、右辺の演算式の演算結果が左辺の型で表現可能な
　範囲を超えた値になる。

例：

```
-------------------------------------------
// a = 64*4 + (unsigned char)(64*4) = 256 + 0 = 256 が正しいが
// a = 64*4 + 64*4 = 256 + 256 = 512 になる
int a;
f()
{
```

```
    int i;
    unsigned char temp;
    for (i=63; i<=64; i++) {  // 発生条件(2)
        temp = i*4;           // 発生条件(3)(4)(5)(6)
        a = i*4 + temp;       // 発生条件(3)(4)
    }
}
```

回避策：以下のいずれかの方法で回避してください。
(1) optimize=0オプションを使用する。
(2) 発生条件(3)のループ帰納変数をvolatile修飾する。

2.3 file_inlineオプションを使用した場合の注意事項(SHC-0071)

file_inlineオプションを使用して関数のファイル間インライン展開を行う場合インライン展開の対象となる関数を呼び出す関数で間違った整数定数値を使用する場合があります。

発生条件：以下の条件を全て満たした時に発生することがあります。
(1) file_inlineオプションを使用している。
(2) optimize=1オプションを使用している。
(3) file_inlineオプションで指定したファイル内のインライン展開対象関数と、そのインライン展開対象関数を呼び出すコンパイル対象ファイル内の関数の両方で複数の整数定数を使用している。
(4) (3)の整数定数の値の少なくとも1つは違う値である。

コンパイル対象ファイル例：a.c

```
int c;
extern void sub();

void func()
{
    c = 1000;       // 発生条件(3)(4)
    sub();
}
```

---

file_inlineオプション指定ファイル例：b.c

```
int b;
void sub()
{
    b = 300;        // 発生条件(3)(4)
```

```
}
-----------------------------------------------
コンパイル結果：
-----------------------------------------------
_func:
        MOV.L       L11+4,R5   ; _c
        MOV.L       L11+8,R6   ; _b
        MOV.W        L11,R2    ; H'03E8 (= 1000)
        MOV.L       R2,@R5     ; 変数cに1000を代入
        RTS
        MOV.L       R2,@R6     ; sub()をここにインライン展開した結果、
                               ; 変数bに間違った値1000を代入
L11:
        .DATA.W     H'03E8
-----------------------------------------------
```

回避策：以下のいずれかの方法で回避してください。
(1) file_inlineオプションを使用しない。
(2) optimize=0オプションを使用する。

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.9.01 Release 00で改修する予定です。