ご使用上のお願いーSuperH RISC engine C/C++コンパイラ Ver.7 不具合内容(12)

SuperH RISC engine C/C++コンパイラパッケージ V.7 の使用上の注意事項 5 件

該当製品

	パッケージバージョン	コンパイラバージョン
PO700CAS7-MWR	7.0B、7.0.01～7.1.04	7.0B、7.0.03～7.1.03
PO700CAS7-SLR	7.0B、7.0.02～7.1.04	7.0B、7.0.03～7.1.03
PO700CAS7-H7R	7.0B、7.0.02～7.1.04	7.0B、7.0.03～7.1.03

内容

1. switch 文制御式の拡張削除不正
2. ゼロ拡張削除不正
3. ループ内変数の 2 次式の計算の結果不正
4. 加算/減算/乗算で拡張削除不正
5. 定数除算の拡張削除不正 (SHC-0001)

恒久対策

本内容は、SuperH RISC engine ファミリ C/C++コンパイラ Ver.7.1.04 以降（コンパイラパッケージ Ver.7.1.05 以降）では、全て修正されています。

チェックツール

本不具合のチェックツールをルネサス エレクトロニクス株式会社のホームページよりダウンロードいただけます。
http://tool-support.renesas.com/jpn/toolnews/shc/shcv7/dr_shcv7_8.html

1. switch 文制御式の拡張削除不正

現象

1,2byte の仮引数を switch 文の制御式に使用した場合、不正に拡張命令を削除し、分岐先アドレスが不正になる場合があります。

発生例

```
int func(short x) {
  short r = -1;
  switch(x) {
    case 0: r = 0; break;
    case 1: r = 1; break;
```

```
        case 2: r = 2; break;
        case 3: r = 3; break;
        case 4: r = 4; break;
        case 5: r = 5; break;
        case 6: r = 6; break;
        case 7: r = 7; break;
        case 8: r = 8; break;
      }
      return (r);
    }

_func:
        MOV         R4,R2       ; R4 の上位 2byte は不定
        EXTS.W      R4,R4
        MOV         #8,R3
        CMP/HI      R3,R4
        MOV         #-1,R6
        BT          L23
        SHLL        R2          ; 拡張しない値をシフト
        MOVA        L25,R0
        MOV.W       @(R0,R2),R1
        ADD         R1,R0
        JMP         @R0
        NOP
L24:
L25:
        .DATA.W     L12-L25
        .DATA.W     L13-L25
                     :
```

発生条件

以下のすべての条件を満たした場合に発生することがあります。

(1) optimize=1 を指定している。
(2) 1,2byte の引数を持つ関数が存在する。
(3) (2)の関数内に switch 文が存在する。
(4) switch 文の制御式が 1,2byte の引数である。
(5) switch 文がテーブル方式で展開される。

回避方法

該当箇所が存在した場合、以下のいずれかの方法で回避していただきますようお願いします。

(1) 該当ファイルを optimize=0 指定する。
(2) switch 文の制御式を引数の型へ明示的にキャストする。
    例: switch((short)x)
(3) 当該引数を volatile 指定する。

2. ゼロ拡張削除不正

現象

ループ内で 2 回以上参照する unsigned char/unsigned short 型変数のゼロ拡張が不正に削除されることがあります。

発生例

```
MOV.B       @Rm,Rn
EXTU.B      Rn, Rn      ;上位 3byte をクリア
        :
MOV.B       Rn,@R15     ;スタックに割付
        :
MOV.B       @R15,R12    ;R12 に割付
                         =>EXTU.B R12,R12 が出ない
L1:
        :
CMP/EQ      R12,R2      ;R12 の値が不正
        :
BT          L1
        :
```

発生条件

以下のすべての条件を満たした場合に発生することがあります。

(1) optimize=1 を指定している。
(2) unsigned char/unsigned short 型変数が存在する。
(3) (2)の変数がループ内を含めて 2 回以上参照される。
(4) (2)の変数がレジスタに割り付かない。
(5) (3)のループ内で使用されないレジスタが存在する。
(6) (5)のレジスタがループ外で使用されている。

回避方法

該当箇所が存在した場合、以下の方法で回避していただきますようお願いします。

・該当ファイルを optimize=0 指定する。

3. ループ内変数の 2 次式の計算の結果不正

現象

ループ内で、m*(i*i+b*i)という形のループ変数 i の 2 次式が存在する場合、最適化により結果不正となることがあります。

発生例

```
int a[100];
f() {
  int i;
  for (i=0;i<100;i++){
    a[i] = 3 * (i * i + 555 * i);
         /* 3*i*i+555*i に不正に変換される*/
    }
}
```

発生条件

以下のすべての条件を満たした場合に発生することがあります。

(1) optimize=1 を指定している。
(2) ループが存在する。
(3) (2)のループ変数が int/unsigned int/long/unsigned long 型である。
(4) (2)のループ内に(3)のループ変数の 2 次式が存在する。
(5) (4)は"m * (i * i + b * i)"(i：ループ変数、m,b：変数もしくは定数)の形である。

回避方法

該当箇所が存在した場合、以下のいずれかの方法で回避していただきますようお願いします。

(1) 該当ファイルを optimize=0 指定する。
(2) 該当するループ内変数に volatile 修飾子を付加して定義する。
(3) 該当するループ内変数を int/unsigned int/long/unsigned long 以外の型で宣言する。
(4) 該当する 2 次式の係数 m をあらかじめ 1 次の項/2 次の項に分配する。
例： 3 * (i * i + 555 * i) → 3 * i * i + 3 * 555 * i

4. 加算/減算/乗算で拡張削除不正

現象

加算/減算/乗算の結果を、それより小さなサイズの型の変数に代入、もしくは小さなサイズの型に変換し、その結果を加算/減算/乗算で使った場合、拡張命令が不正に削除されることがあります。

発生例

```
int x,a;
test_000()
{
   char b;
   b = (char)(a + 3);
```

```
    x = b + 2;
}

_f:
    MOV.L       L11,R6    ; _a
    MOV.L       L11+4,R2  ; _x
    MOV.L       @R6,R6
    ADD         #5,R6     ; char へのキャストが削除され、a+5 の結果を
                          ; x に代入している。
    RTS
    MOV.L       R6,@R2
```

発生条件

以下のすべての条件を満たした場合に発生することがあります。

(1) optimize=1 を指定している。
(2) 2 つのオペランドのうち片方のみが定数である加算、減算、乗算が存在する。
(3) (a),(b)のいずれかが成立する。
  (a) (2)の結果を、それより小さなサイズの型にキャストし、その結果を加算、減算、乗算している。
  (b) (2)の結果を、それより小さなサイズの型の変数に代入し、その結果を加算、減算、乗算している。

回避方法

該当箇所が存在した場合、以下のいずれかの方法で回避していただきますようお願いします。

(1) 該当ファイルを optimize=0 指定する。
(2) 発生条件(2)の結果を volatile 宣言した変数に代入する。

5. 定数除算の拡張削除不正 (SHC-0001)

現象

定数除算において、除数と被除数をともにそれらより小さなサイズの型に変換して、演算結果を変換後の型の変数に代入する時、除数または被除数に対する型変換が不正に削除される場合があります。

発生例

```
char c;
int i;
func(){
  c = ((char)i / (char)2);
                  /* 被除数を不正に int 型で計算している */
}

func2(){
```

```
    c = ((char)i / (char)0x102);
                        /* 除数を不正に 0x00000102 で計算している */
}
```

発生条件

以下のすべての条件を満たした場合に発生することがあります。

(1) optimize=1 を指定している。
(2) 定数による除算が存在する。
(3) (2)の除算で、除数と被除数をともにそれらより小さなサイズの型へ変換している。
(4) 除数の値が 2 のべき乗である、または cpu=sh1 以外でかつ division=cpu=inline を指定している。
(5) 除算の結果を(3)の変換後の型の変数に代入している。

回避方法

該当箇所が存在した場合、以下のいずれかの方法で回避していただきますようお願いします。

(1) 該当ファイルを optimize=0 指定する。
(2) 除数の定数値のキャストを削除し、除数をキャスト後の値で置き換える。
    例:
    func1(): c = ((char)i / (char)2);   → c = ((char)i / 2);
    func2(): c = ((char)i / (char)0x102); → c = ((char)i / 0x02);
(3) 除算の結果を int 型の変数に代入する。
    例:
    func1(): tmp = ((char)i / (char)2); (tmp: int 型変数)
             c = (char)tmp;

株式会社 日立超LSIシステムズ