ご使用上のお願い－SuperH RISC engine C/C++コンパイラ Ver.7 不具合内容(9)

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 2 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
P0700CAS7-MWR	7.0B、7.0.01～7.1.03	7.0B、7.0.03～7.1.02
P0700CAS7-SLR	7.0B、7.0.02～7.1.03	7.0B、7.0.03～7.1.02
P0700CAS7-H7R	7.0B、7.0.02～7.1.03	7.0B、7.0.03～7.1.02

内容

1. NEG 命令後の不当拡張命令削除
2. ポインタ変数ロード後の不当拡張命令削除

恒久対策

本内容は、SuperH RISC engine ファミリ C/C++コンパイラ Ver.7.1.03 以降（コンパイラパッケージ Ver.7.1.04 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_5.html

1. NEG 命令後の不当拡張命令削除

現象

unsigned char/short 型変数を含む式が同一関数内で A－B、B－A(A、B：当該変数を含む式)と同時に存在する場合、共通式削除の最適化により不当に拡張命令を削除する場合がある。

発生例

```
unsigned short var_a, var_b, var_c;
 long result;

 void f() {
     unsigned short x;
         if (var_a >= var_b) {
         x = var_a-var_b;
         result = x * var_c;
     } else {
         x = var_b - var_a;
         result = x * var_c;
     }
```

```
}

_f:
    MOV.L    L14,R2       ; _var_a
    MOV.L    L14+4,R4     ; _var_b
    MOV.W    @R2,R5
    MOV.W    @R4,R2
    MOV.L    L14+8,R4     ; _var_c
    MOV      R5,R6
    SUB      R2,R6        ; temp <- var_a-var_b
    MOV.W    @R4,R7
    EXTU.W   R5,R5
    EXTU.W   R2,R2
    CMP/GE   R2,R5
    BF/S     L12
    EXTU.W   R7,R4
    EXTU.W   R6,R2        ; x <- (unsigned short)temp
    MOV.L    L14+12,R5    ; _result
    MUL.L    R2,R4
    STS      MACL,R2
    RTS
    MOV.L    R2,@R5
L12:
    EXTU.W   R6,R6
    MOV.L    L14+12,R5    ; _result
    NEG      R6,R2        ; x <- (long)(-temp)
                          ; EXTU.W R2,R2 を不当に削除
    MUL.L    R2,R4
    STS      MACL,R2
    RTS
    MOV.L    R2,@R5
```

発生条件

以下の条件をすべて満たした場合に発生することがあります。
該当するかどうかはチェックツールを使用することにより確認することができます。

(1) optimize=1 オプションを指定している。
(2) unsigned char/short 型変数を使用している。
(3) (2)の変数を含む式が同一関数内で以下の演算で使用されている。
    A－B
    B－A
    A、B は(2)の変数を含む式
    上の例では、A：var_a、B：var_b
(4) 共通式削除の最適化により、(3)の演算が共通化される。
    上の例では、以下の通り。

```
void f() {
    unsigned short x;
    long temp = var_a - var_b;
```

```
        if (var_a >= var_b) {
           result = (unsigned short)temp * var_c;
                      /* var_a-var_b を temp に置換 */
        } else {
           result = (unsigned short)(-temp) * var_c;
                      /* var_b-var_a を-temp に置換 */
        }
    }
```

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 オプション指定する。

(2) 当該変数もしくは当該変数を含む式の代入先変数を volatile 宣言する。

例：

```
 void f() {
     volatile unsigned short x;  /* volatile を追加 */
     if (var_a >= var_b) {
         x = var_a-var_b;
         result = x * var_c;
     } else {
         x = var_b - var_a;
         result = x * var_c;
     }
 }
```

2. ポインタ変数ロード後の不当拡張命令削除

現象

unsigned char/short 型ポインタ変数のポインタ先と 0 との加減算、もしくは 1 との乗算を行う式を記述した場合、不当に変数のロード後のゼロ拡張命令を削除する場合がある。

発生例

```
unsigned char *p;
int a;
void func(){
    a = 0;
    a += *p;
}

_func:
    MOV.L    L11,R5     ; _a
    MOV      #0,R2      ; H'00000000
    MOV.L    R2,@R5
    MOV.L    L11+4,R2   ; _p
    MOV.L    @R2,R6
    MOV.B    @R6,R2     ; ロード時に符号拡張
```

```
    RTS
    MOV.L    R2,@R5     ; ゼロ拡張せずに 4byte 領域にストア
```

発生条件

以下の条件をすべて満たした場合に発生することがあります。
該当するかどうかはチェックツールを使用することにより確認することができます。

(1) optimize=1 オプションを指定している。
(2) unsigned char/short 型変数をポインタを介して使用している。
(3) (2)の変数と 0 の加減算もしくは 1 との乗算を行っている(最適化により 0 との加減算もしくは 1 との乗算になる場合もあり)。

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) 明示的に 0 との加減算を行っている場合は、代入式にする。
    例：
```
    void func() {
        a = *p;
    }
```
(2) optimize=0 を指定する。
(3) 当該変数を volatile 宣言した変数に代入して使用する。

例：
```
void func() {
    volatile unsigned char temp = *p;
    a = 0;
    a += temp;
}
```

株式会社 日立超LSIシステムズ