SuperH RISC engine C/C++コンパイラ Ver.7 不具合内容

- 過去のお知らせ

SuperH RISC engine C/C++コンパイラ Ver.7 台における不具合内容を以下に示します。
1.、2.、4.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入手できます。
http://tool-support.renesas.com/jpn/toolnews/shc/shcv7/dr_shcv7_4.html

1. 無条件分岐の不当削除

現象：
関数の最後の処理が条件文で、条件がネストしており、かつ最後の条件節が関数呼び出し＋return 文、その前の条件節が関数呼び出しで終わっている場合、関数出口への無条件分岐が不当に削除される場合がある。

[例]
```
    void sub(int parm) {
        if (parm == 0) {
            ;
        } else if (parm == 1) {
            ;
        } else if (parm == 2) {
            ;
        } else if (parm == 3) {
            ;
        } else if (parm == 4) {
            ;
        } else if (parm == 5) {
            func1();    /* <A> */
        } else {
            func2();    /* <B> */
            return;     /* <B> */
        }
        return;
    }
```
[出力コード]
```
  _sub:
    STS.L   PR,@-R15
    TST     R4,R4
    BT      L11
    MOV     R4,R0
    CMP/EQ  #1,R0
    BT      L11
    CMP/EQ  #2,R0
    BT      L11
    CMP/EQ  #3,R0
    BT      L11
    CMP/EQ  #4,R0
    BT      L11
    CMP/EQ  #5,R0
    BF      L18
    MOV.L   L20+2,R2    ; _func1
    JSR     @R2
    NOP
  L11:
```

```
                        ; L19 への分岐命令を削除
  L18:
    MOV.L   L20+6,R2    ; _func2
    JMP     @R2         ; 常に関数呼び出しされる
    LDS.L   @R15+,PR
  L19:
    LDS.L   @R15+,PR
    RTS
    NOP
```

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 を指定している。

(2) 関数の最後の処理が条件文でかつ条件がネストしている。

(3) 最後の条件節が関数呼び出し+return 文で終わっている(例の<B>)。

(4) (3)の直前の条件節が関数呼び出しで終わっている(例の<A>)。

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) 最後の条件節の直前の条件節(例の<A>)の最後に nop()組み込み関数を追加する。

```
[例]
#include <machine.h>    /* 追加 for nop() */
        :
} else if (parm == 5) {
    func1();    /* <A> */
    nop();      /* 追加 */
} else {
        :
```

2. unsigned 型->float 型キャスト不正

現象：

unsigned 型の変数を float 型に明示的にキャストした時、キャストが不当に削除される場合がある。

```
[例]
    unsigned short us1;
    volatile unsigned short us0;
    volatile float f0;
    float *p;
    void func() {
        f0 = *p = ((float)us0, (float)us1);
    }
[出力コード]
  _func:
    MOV.L   L29+50,R2   ; _us0
    MOV.L   L29+54,R5   ; _p
    MOV.W   @R2,R6
```

```
    MOV.L   L29+58,R6    ; _us1
    MOV.W   @R6,R2
    EXTU.W  R2,R6
    MOV.L   @R5,R2
    MOV.L   R6,@R2       ; float に変換しないで*p にストア
    MOV.L   @R5,R2
    MOV.L   @R2,R6
    MOV.L   L29+10,R2    ; _f0
    RTS
    MOV.L   R6,@R2       ; float に変換しないで f0 にストア
```

発生条件：

以下のいずれかの条件を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。

ただし、チェックツールで検出されても不具合に該当しないケースもあります。

(1) unsigned 型の変数を float 型にキャストしている。

(2) unsigned 型の変数を double 型にキャストし、かつ double=float または fpu=single オプションを指定、または long double 型にキャストし、かつ fpu=single を指定している。

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。

当該変数を元の型を表現できる signed 型(unsigned int/long の場合は long double 型)にキャストしてから float 型にキャストする。

3. ld_ext()、st_ext()使用時のスタックポインタ不正移動

現象：

SH-4 で ld_exp()または st_ext()組み込み関数使用時にパラメータにローカル配列を指定した場合、不正にスタックポインタを移動するコードを出力する場合がある。

[例]

```
    #include <machine.h>

    void main() {
        float table[4][4], data1[4][4], data2[4][4];
                :
        ld_ext(table);
        mtrx4mul(data1,data2);
                :
    }
```

[出力コード]

```
                :
    FRCHG
    FMOV.S   @R15+,FR0   ; R15 を更新、この間で例外が発生した場合
                                        上位のスタック領域が破壊
    FMOV.S   @R15+,FR1   ;                          |
    FMOV.S   @R15+,FR2   ;                          |
    FMOV.S   @R15+,FR3   ;                          |
    FMOV.S   @R15+,FR4   ;                          |
    FMOV.S   @R15+,FR5   ;                          |
    FMOV.S   @R15+,FR6   ;                          |
```

```
    FMOV.S   @R15+,FR7   ;                          |
    FMOV.S   @R15+,FR8   ;                          |
    FMOV.S   @R15+,FR9   ;                          |
    FMOV.S   @R15+,FR10  ;                          |
    FMOV.S   @R15+,FR11  ;                          |
    FMOV.S   @R15+,FR12  ;                          |
    FMOV.S   @R15+,FR13  ;                          |
    FMOV.S   @R15+,FR14  ;                          |
    FMOV.S   @R15+,FR15  ;                          |
    FRCHG                ;                          |
    ADD      #-64,R15    ;                          V
                 :
```

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。

(1) cpu=sh4 を指定し、かつ ld_ext()、st_ext()を使用している。

(2) ld_ext(),st_ext()のパラメタにローカル配列を指定している。

回避方法：

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) ld_ext()、st_ext()のパラメタを外部変数にする。

4. データ出力時の内部エラー

現象：

"シンボルアドレス＋オフセット値"の初期値をもつ変数が多く存在する場合、内部エラー(4098)もしくはオブジェクト不正になる場合がある。

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) code=machinecode(デフォルト)を指定している。

(2) listfile オプションを指定していない、もしくは listfile オプションと show=noobject オプションを同時に指定している。

(3) 初期値つき変数の定義が存在する。

(4) (3)の初期値が"シンボルアドレス＋オフセット値"または構造体先頭でないメンバのアドレスである。

(5) (3)の変数、および初期値が以下の条件を満たしている。
((3)の数+(4)のオフセット値の 10 進桁数の合計)≧33000

[例]

```
    extern char g;
    #define DATA1A    (&g+2147483647)
    #define DATA10A  DATA1A,  DATA1A,  DATA1A,  DATA1A,  DATA1A,¥
                     DATA1A,  DATA1A,  DATA1A,  DATA1A,  DATA1A
```

```
#define DATA100A DATA10A, DATA10A, DATA10A, DATA10A, DATA10A,¥
                 DATA10A, DATA10A, DATA10A, DATA10A, DATA10A

/* 以下は(変数の数+offset の合計桁数)=(3001+10*3001)=33011>33000 となる */
char *a1[1000] = {
    DATA100A, DATA100A, DATA100A, DATA100A, DATA100A,
    DATA100A, DATA100A, DATA100A, DATA100A, DATA100A
};
char *a2[1000] = {
    DATA100A, DATA100A, DATA100A, DATA100A, DATA100A,
    DATA100A, DATA100A, DATA100A, DATA100A, DATA100A
};
char *a3[1000] = {
    DATA100A, DATA100A, DATA100A, DATA100A, DATA100A,
    DATA100A, DATA100A, DATA100A, DATA100A, DATA100A
};
char *a = DATA1A;
```

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 listfile オプション指定する(show=noobject は指定しない)。

(2) 該当ファイルを code=asmcode オプション指定する。

株式会社 日立超LSIシステムズ