Date: May.26.2004

RENESAS TECHNICAL UPDATE

Nippon Bldg., 2-6-2, Ohte-machi, Chiyoda-ku, Tokyo 100-0004, Japan
RenesasTechnology Corp.

Product Category	User Development Environment		Document No.	TN-CSX-069A/EA	Rev.	1.0
Title	SuperH RISC engine C/C++ Compiler Ver.8 bug information (2)		Information Category	Usage Limitation		
Applicable Product	P0700CAS8-MWR P0700CAS8-SLR P0700CAS8-H7R R0C40700XSW08R R0C40700XSS08R R0C40700XSH08R	Lot No. Ver.8.0	Reference Document	SuperH RISC engine C/C++ Compiler Assembler Optimizing Linkage Editor User's Manual REJ10B0047-0100H Rev.1.00		

Attached is the description of the detected bug information in Ver. 8 series of the SuperH RISC engine C/C++ Compiler.

The bug will affect this package version.

Attached: P0700CAS8-040518E

SuperH RISC engine C/C++ Compiler Ver. 8 The details of the detected bug information (2)

SuperH RISC engine C/C++ Compiler Ver.8
The details of the detected bug information (2)

The bugs detected in the ver. 8 of the SuperH RISC engine C/C++ Compiler is shown below.

1. Illegal Copy Propagation

[Description]

When a copy instruction existed in a block with multiple branch sources, the copy instruction might be illegally eliminated.

[Example]

```
int func(int *x) {
   int ret=0;
   while(*x++){
      if(*x==1){
         ret+=2;
      }
   }
   return (ret+2);
}

_func:
      MOV         #0,R5     ; Illegally eliminated the copy instruction and converted R7 to R5
L11:
      MOV.L       @R4,R2
      ADD         #4,R4
                            ;  *1 Illegally eliminated MOV R7,R5
      TST         R2,R2
      ADD         #2,R5
      BT          L13
      MOV.L       @R4,R0
      CMP/EQ      #1,R0
      BT          L11       ;  *2 By *3, BF L11 was converted
      BRA         L11
      NOP                   ;  *3 Illegally eliminated MOV R5,R7
L13:
      RTS
      MOV         R5,R0
```

[Conditions]

This problem might occur when all of the following conditions were fulfilled.

(1) The optimize=1 option was specified.
(2) A conditional statement was described.
(3) A copy instruction existed in a block with multiple branch sources (*1 in the above example).
(4) The block of the branch sources in (3) had a path with no definition of the copy source register (R7 in the above example) for the copy instruction (in the example, the path branching from *2 to L11).

[Solution]

This problem can be prevented by the following method.

(1) Specify optimize=0.

2. Illegal Elimination of Unnecessary Expressions

[Description]

If a then or else clause of a conditional statement had an assignment expression and another assignment expression, of which the both sides had the same variable, follows the said expression, the conditional statement might be illegally eliminated.

[Example]

```
int x;

void f(int y){
   if (y>=256){        /* Illegal elimination */
      x=0;             /* *1                           */
   }
   x=x;                /* *2  Eliminated the assignment expression that had the same variable in both sides */
   x++;
}

         ⇩

void f(int y){
   x=0;
   x++;                /* Propagated x=0 */
}

         ⇩

void f(int y){
   x=1;
}
```

[Conditions]

This problem might occur when all of the following conditions were fulfilled.

(1) The optimize=1 option was specified.
(2) A conditional statement was described.
(3) A then or else clause of the conditional statement of (2) had an assignment expression (*1 in the above example).
(4) An assignment expression, in which the both sides had the same variable as the variable assigned to in (3), followed the conditional statement of (2) (*2 in the above example).

[Solution]

This problem can be prevented by either of the following methods.

(1) Specify optimize=0.
(2) Specify opt_range=noblock.

.

3. Illegal Access with a Parameter Passed via the Stack

[Description]

If a function with the parameter passed via the stack had a function call immediately before the exit, an address for reference to a parameter passed via the stack might be incorrect when the speed option was specified.

[Example]

```
typedef struct {
   int x;
} ST;
extern void g(ST *x);
void f(int a, ST b) {/* b was a parameter passed via the stack */
   if (a) {
      g(&b);
      /* (A) */
   }
   /* (B) */
}

; Address where parameter b was stored at the function entry = R15
_f:
       TST          R4,R4
       BT           L12
       MOV          R15,R4
       MOV.L        L14,R2    ; _g
       JMP          @R2       ; (A)
       ADD          #4,R4     ; R4 <- R15+4 : Not the address of b
L12:
       RTS                    ; (B)
       NOP
```

[Conditions]

This problem might occur when all of the following conditions were fulfilled.

(1) The optimize=1 option was specified.
(2) The speed option was specified.
(3) The function had a parameter passed via the stack (b in the above example).
(4) The function had multiple exits ((A) and (B) in the above example).
(5) There was a function call immediately before any of the exits in (4) (g(&b); in the above example).
(6) (5) was the only function call in this function.

[Solution]

This problem can be prevented by one of the following methods.

(1) Do not specify the speed option.
(2) Specify optimize=0.
(3) Insert a nop() built-in function after the function call.
(4) Insert a dummy function call in the function and specify the noinline option.

4. Incorrect GBR Relative Logic Operation

[Description]

If a logic operation with a 1-byte array or a bit-field member for which #pragma gbr_base/gbr_base1 was specified was performed, the result of the operation might be written to an incorrect area.

[Example]

```
#pragma gbr_base a,b
char a[2],b[2];
void f() {
   a[0] = b[0] & 1;
}

   MOV        #_b-(STARTOF $G0),R0
   RTS
   AND.B      #1,@(R0,GBR)          ; Wrote the result of the operation to b[0]
```

[Conditions]

This problem might occur when all of the following conditions were fulfilled.

(1) The gbr=user option was specified.
(2) #pragma gbr_base/gbr_base1 was specified for any of the following variables:
   - An (unsigned) char-type array
   - A structure array that has an (unsigned) char-type member
   - A structure that has an (unsigned) char-type array member
   - A structure that has a bit-field member of 8 bits or less
(3) A logic operation of a constant (&, |, ^) with the variable of (2) (b[0] in the above example) was performed.
(4) The variable assigned to by the operation of (3) (a[0] in the above example) fulfilled the condition of (2).
(5) Variables of (3) and (4) were different variables, different elements of the same array, or different members of the same structure.

[Solution]

This problem can be prevented by one of the following methods.

(1) Cancel specification of #pragma gbr_base/gbr_base1.
(2) Specify gbr=auto (outputs a warning and invalidates #pragma gbr_base/gbr_base1).
(3) Assign the result of the operation to a temporary variable for which volatile has been specified.

Example:

```
void f() {
   volatile char temp;
   temp = b[0] & 1;
   a[0] = temp;
}
```

5. Illegal Elimination of Sign/Zero Extension

[Description]

If the address of a variable/constant or the index of an array was cast to 1 or 2 bytes and this value was used for accessing memory, or the expression which was cast to a char type was assigned to an unsigned short type variable and the result was used for comparison, the cast might be illegally eliminated.

[Example 1]

```
   unsigned short x;
   char a[1000];

   void f() {
      a[(char)x] = 0;
   }

      MOV.L        L11+2,R2        ; _x
      MOV.L        L11+6,R6        ; _a
      MOV.W        @R2,R5
      EXTU.B       R5,R0
                                   ; Eliminated EXTS.B R0,R0
      MOV          #0,R5           ; H'00000000
      RTS
      MOV.B        R5,@(R0,R6)     ; When x was not within the range of 0 to 127,
                                   ; an incorrect address might be referred to.
```

[Example 2]

```
   unsigned short sc0;
   unsigned int b;

   func1() {
      unsigned short us1;
      us1 = (char)b
      return(us0 !=us1);
   }

      MOV.L        L11,R2          ; _b
      MOV.L        L11+4,R5        ; _us0
      MOV.L        @R2,R6
      EXTS.B       R6,R2
      MOV.W        @R5,R6
      EXTU.W       R6,R5
      CMP/EQ       R2,R5           ; (char)b was not cast to an unsigned short type
                                   ;  and was used in comparison.
      MOVT         R0
      RTS
      XOR          #1,R0
```

[Conditions]

This problem might occur when all of the following conditions were fulfilled.

(1) The optimize=1 option was specified.

(2) One of the following conditions (a)(b) was fulfilled.

(a-1) The address of a variable/constant or the index of an array was explicitly cast to 1 or 2 bytes, or this function had a char/short type temporary parameter and the parameter was used only in the index of an array.

(a-2) The value of (a-1) was used for accessing memory.

(b-1) The expression which was cast to a char type was assigned to an unsigned short type variable.

(b-2) The variable of (b-1) was used for comparison.

[Solution]

This problem can be prevented by one of the following methods.

(1) Specify optimize=0.

(2) If the condition (2)(b) is fulfilled, declare the unsigned type variable of (b-1) as volatile.