RENESAS Tool News

RENESAS TOOL NEWS on February 16, 2008: 080216/tn1

Notes on Using the C/C++ Compiler Package for the SuperH RISC engine MCU Family

Please take note of the following problems in using the C/C++ compiler package for the SuperH RISC engine MCU family:

1. With using the Automatic Literal Pool Generation function
2. With optimizing the program that contains instructions referencing literal pools using the linkage editor

1. With Using the Automatic Literal Pool Generation Function

1.1 Product and Versions Concerned

The C/C++ compiler package for the SuperH RISC engine family
V.9.00 Release 00 through V.9.01 Release 01

1.2 Description

Consider the case where the code section of the assembly program is written in the absolute-address format, and in this code section exists an instruction for which a literal pool is generated by the Automatic Literal Pool Generation function during assembly.
If this program is assembled with SH2A or SH2AFPU selected as the CPU type, the literal pool may be incorrectly generated.

1.3 Conditions

This problem may occur if the following conditions are all satisfied:
(1) SH2A or SH2AFPU is selected as the CPU type to assemble the program.
(2) In the program exists a code section written in the absolute-address format.
(3) In the section in (2) exist any of the following MOV instructions:

```
MOV.B    @(disp, R0),Rn      MOV.B    Rm,@(disp, R0)
MOV.W    @(disp, R0),Rn      MOV.W    Rm,@(disp, R0)
MOV.L    @(disp, R0),Rn      MOV.L    Rm,@(disp, R0)
```

(4) Displacement width (:12) is not given explicitly but implicitly
in the instructions in (3).
(5) The values of disp (displacement width) of the instructions in (3)
are represented in 4 bits.
(6) Subsequent to the first of the instructions in (3) exists an
instruction that uses a literal generated by the Automatic Literal
Pool Generation function to address its operand.
(7) The literal generated for the instruction in (6) is a label
(an address value).
(8) The label in (7) exists between the first of the instructions
in (3) and the instruction in (6).

Example

```
---------------------------------------------
   .SECTION P,CODE,LOCATE=H'00000600   ; Condition (2)
   MOV.L @(0,R0),R2                  ; Conditions (3),(4), and (5)
   NOP
LAB01:                               ; Conditions (7) and (8)
   NOP
   MOV.L #LAB01,R1                   ; Condition (6)
   NOP;
   .POOL
   .END
---------------------------------------------
```

1.4 Workarounds

Avoid this problem in any of the following ways:
(1) Do not write the code section in Condition (2) in the absolute-
address format but the relative-address one.
(2) Do not use the Automatic Literal Pool Generation function.
(3) Give displacement width (:12) explicitly to the instructions
in Condition (3).

2. With Optimizing the Program That Contains Instructions Referencing Literal Pools Using the Linkage Editor

2.1 Product and Versions Concerned

The C/C++ compiler package for the SuperH RISC engine family
V.9.00 Release 00 through V.9.01 Release 01

2.2 Description

Optimization may cause incorrect object code to be generated.

2.3 Conditions

This problem may occur if the following conditions are all satisfied:

(1) Option -cpu=SH2A or -cpu=SH2AFPU is selected in compilation.

(2) Option -goptimize is also selected in compilation.

(3) Optimization is performed in linking (-nooptimize not selected).

(4) An instruction referencing any of the following exists in a function:
- The address of a variable/function
- A constant of 3 bytes or more in length
- A variable

(5) One or more functions containing such an instruction as in (4) are placed each at the end of a program area.

(6) The address or constant referenced by the instruction in (4) is placed at the beginning of the literal pool of a program area.

(7) Compilation generates either of the following instruction pairs within the functions in (5):

```
    mov.l  ; Data-transfer instruction referencing the beginning
           ; of the literal pool in (6)
    rts/n  ; Return instruction from a subroutine procedure with
             no delay slots
  Or,
    mov.l  ; Data-transfer instruction referencing the beginning
           ; of the literal pool in (6)
    rtv/n  ; Return instruction from a subroutine procedure with
           ; register-value transfer and no delay slots
```

NOTICE:
Whether Condition (7) is met or not can be checked by viewing the compile list file (.lst). To generate the compile list, use the -list option in compilation.

Example:
File example.c

```
-------------------------------------------------------------------
long a;            /* Condition (6); Address of variable "a"  */
                /*   is outputted at the beginning of     */
                /*   the literal pool of a program area   */
                /*   (a non-specified P section)          */
long * mov();

void func2(void);
void func1()
{
   a++;
   func2();
  }
  long * mov()    /* Condition (5); mov() is placed at the end */
                /*    of a program area (a non-specified P  */
                /*    section) within the example.c file    */
  {
    return &a;        /* Conditions (4) and (7) */
  }
  #pragma section 1     /* Condition (5);                     */
  void func2()         /* func2() is placed in the program area */
                    /* called the P1 section                */
  {
  . . . . . . . . . . . . . . .

  }
  ----------------------------------------------------------------------
```

2.4 Workarounds

Avoid this problem in either of the following ways:

(1) Do not use option -goptimize in compilation.

(2) Use option -nooptimize in linking.

3. Schedule of Fixing the Problems

The above problems have been fixed in the V.9.02 Release 00 product, which is being published on February 20. So use this version.
For further information on V.9.02 Release 00, see RENESAS TOOL NEWS Document No. 080216/tn2 at
http://tool-support.renesas.com/eng/toolnews/080216/tn2.htm
(available on and after February 20, 2008).