RENESAS TOOL NEWS on June 01 2015: 150601/tn3

Note on Using the C Compiler Package for RH850 Family

When using the CC-RH C Compiler package for the RH850 Family, take note of the problems described in this note regarding the following points.

1. Designating immediate values as the operands of assembly instructions (No. 2)
2. The definition of and reference to a label being in the same assembly unit (No. 3)
3. Omitting base registers in assembly instructions (No. 4)
4. Assembly statements not in accord with the specification (No. 5)

Note: The number which follows the description of the precautionary note is an identifying number for the precaution.

---

1. Designating Immediate Values as the Operands of Assembly Instructions (No. 2)

1.1 Product Concerned

CC-RH V1.02.00

1.2 Description

Unintended code might be generated when an immediate value is specified as the operand of assembly instruction in assembly source code.
Note that this does not apply to assembly instructions generated by the compiler.

1.3 Conditions

The problem arises when any of conditions (1) to (6) listed below is met.
(1) An immediate value wider than 16 bits is specified for disp of an ld or st instruction, and the 16 lower-order bits of disp are 0.

Example of a statement satisfying the condition:

```
---------------------------------------------
st.w  r10, 0xfff80000[r0]
---------------------------------------------
```

(2) An immediate value wider than 16 bits is specified as imm1 of a prepare instruction, and the 16 lower-order bits of imm1 are 0.

(3) An immediate value in the range from 0x8000 to 0xFFFF is specified as imm for a mulu instruction.

(4) An immediate value which is not a multiple of 4 and having a value greater than or equal to 128 is specified as imm1 of a prepare or dispose instruction.

Example of a statement satisfying the condition:

```
---------------------------------------------
prepare  r20, 0x83
---------------------------------------------
```

(5) A separation operator (LOW or HIGH) is used in an operand of a mov, cmov, cmp, add, mulh, or satadd instruction, and an immediate value is the term of the separation operator and meets either of the following conditions (a) or (b). *
  (a) If the operator is LOW, the fourth bit is 1.
  (b) If the operator is HIGH, the twelfth bit is 1.
Note: The lowest-order bit of the immediate value is the 0th bit.

Example of a statement satisfying the condition:

```
---------------------------------------------
add  low(0x10), r12
---------------------------------------------
```

-0x10 is added to r12 in V1.01.00 and prior versions; 0x10 is added to r12 in V1.02.00.

(6) A separation operator (LOWW, HIGHW, or HIGHW1) is used in an operand of an ld, st, set1, clr1, not1, tst1, movea, addi, or satsubi instruction, and an immediate value is the term of the separation operator and meets any of the following conditions (c) to (e). *
  (c) If the operator is LOWW, the fifteenth bit is 1.
  (d) If the operator is HIGHW, the thirty-first bit is 1.
  (e) If the operator is HIGHW1, the fifteenth or the thirty-first bit is 1.
Note: The lowest-order bit of the immediate value is the 0th bit.

Example of a statement satisfying the condition:

```
---------------------------------------------
movea  loww(0x8000), r0, r12
---------------------------------------------
```

-0x8000 is stored in r12 in V1.01.00 and prior versions; 0x8000 is stored in r12 in V1.02.00.

1.4 Workaround

To avoid this problem, take any of the following steps.

(1) The assembler is unable to correctly expand the instructions in the case of conditions (1) to (3), so the user needs to expand the instructions.

Example of the workaround:

```
---------------------------------------------
movhi  highw(0xfff80000), r0, r5
st.w   r10, loww(0xfff80000)[r5]
---------------------------------------------
```

(2) In the case of condition (4), although imm1 must be rounded to a multiple of 4, this is not the case in the generated code. Round the value before specifying imm1.

Example of the workaround:

```
---------------------------------------------
prepare  r20, 0x80
---------------------------------------------
```

(3) In case of (5) or (6), directly specify the expected value without using a separation operator.

Example of the workaround:

```
---------------------------------------------
add     -0x10, r12
movea  -0x8000, r0, r12
---------------------------------------------
```

1.5 Schedule for Fixing the Problem

This problem will be fixed in the next version.

2. The Definition of and Reference to a Label being in the Same Assembly Unit (No. 3)

2.1 Product Concerned

CC-RH V1.02.00

2.2 Description

Defining a label and giving the .extern directive for the label in the same unit of assembly source code leads to an error in assembly or the address of the symbol being incorrect.
A single unit of assembly includes cases where the assembly instruction is inserted by the INCLUDE control instruction.
Note that this does not apply to assembly instructions generated by the compiler.

2.3 Conditions

The problem arises when both conditions (1) and (2) listed below are met.
(1) A label is defined in the assembly source code.
(2) The .extern directive is applied to the label in the same unit of assembly.

Example of statements satisfying the condition:

```
---------------------------------------------
.extern _sym       -->(a)
  nop
_sym:            -->(b)
  jarl _sym, lp   -->(c)
---------------------------------------------
```

Branch instruction (c) causes a branch to (a) but not to (b), the line on which _sym is defined.

2.4 Workaround

Delete the .extern directive.

2.5 Schedule for Fixing the Problem

This problem will be fixed in the next version.

3. Omitting Base Registers in Assembly Instructions (No. 4)

3.1 Product Concerned

CC-RH V1.02.00

3.2 Description

Generated code will be incorrect when the base register is omitted from a memory reference instruction in assembly source code.
Note that this does not apply to assembly instructions generated by the compiler.

3.3 Conditions

The problem arises when both conditions (1) and (2) listed below are met.

(1) The base register [reg1] is omitted from a set1, clr1, tst1, not1, or st instruction.

(2) A separation operator is used with disp in the above instruction.

Example of a statement satisfying the condition:

```
---------------------------------------------
st.w  r10, loww(#_sym)
---------------------------------------------
```

The reference to #_sym implicitly specifies [r0] as a base register.
The separation operator LOWW is ignored in this situation in V1.02.00.

3.4 Workaround

Do not omit base registers.

Example of the workaround:

```
---------------------------------------------
st.w r10, loww(#_sym)[r0]
---------------------------------------------
```

3.5 Schedule for Fixing the Problem

This problem will be fixed in the next version.

4. Assembly Statements Not in Accord with the Specification (No. 5)

4.1 Product Concerned

CC-RH V1.02.00

4.2 Description

An error in a formula for a statement where a formula can be used as an operand should normally be detected as an error, but might not be. The code generated as a result will not be as intended.
Note that this does not apply to assembly instructions generated by the compiler.

4.3 Conditions

This problem may arise when an operand or part of an operand includes a formula meeting any of conditions (1) to (4) below.

(1) The formula includes any of the following letters.

*, /, +, -, <, >, &, ", |, ^, ~

Example of a statement satisfying the condition:

```
---------------------------------------------
```

```
br      +
---------------------------------------------
```

(2) The formula includes any of the following two-letter strings.
&&, ==, <<, >>, (), ""

(3) The formula is a string which includes the letters or strings in (1) or (2) enclosed in parentheses: ().

Example of a statement satisfying the condition:

```
---------------------------------------------
mov (&&), r10
---------------------------------------------
```

(4) A separation operator is applied to an empty term or section aggregation operator.

Example of a statement satisfying the condition:

```
---------------------------------------------
add     low(), r10
---------------------------------------------
```

4.4 Workaround

Correct the assembly statement.

4.5 Schedule for Fixing the Problem

This problem will be fixed in the next version.