RENESAS ツールニュース

RENESAS TOOL NEWS [2003年07月01日：RSO-M3T-CC32R-030701D]

M3T-CC32R
ご使用上のお願い

M32Rファミリ用クロスツールキット M3T-CC32R の使用上の注意事項を連絡します。

- 変数のアドレスを、その変数とは異なる型へのポインタに変換してそれを間接演算(メモリに対して読み書き)する場合の注意事項

---

1. **該当製品**
   M3T-CC32R V.4.00 Release 1 ～ V.4.10 Release 1

2. **内容**

   ある変数のアドレスを、その変数とは異なる型へのポインタに変換し、 それを間接演算(メモリに対して読み書き)するCソースをコンパイルした 場合、エラーが出力されるか、または不正なコードが生成されます。

   cg32R: "...", line xx: internal error: illegal IL, illegal lhs of ASSIGN
   (エラーメッセージの "..." にはファイル名が、 xx には行番号が表示されます。)

   2.1 発生条件
   以下6点の条件を全て満たす場合に発生します。

   (1) コンパイル時に -O4の最適化と同じ内容を含む最適化オプションを指定している。
   (-O4, -O5, -O6, -O7いずれかの指定が有効になっているか、-Otimeまたは-Ospaceを単独で指定している。)

   (2) ある変数が、整数型、浮動小数点型またはポインタ型である。

   (3) (2)の変数のアドレスをキャストして、元の変数の型とは異なる型へのポインタに変換している。

   (4) (3)のポインタを使って次の(a)～(c)いずれかに該当する間接演算を行っている。

   (a) ポインタ間接演算(*演算子)

   (b) 構造体ポインタ演算(->演算子)
   ※ 先頭の構造体メンバを指定する場合に限る。

(c) 共用体ポインタ演算(->演算子)

(5) (2)の変数と(4)の間接演算オブジェクトは、サイズが同じでかつ、型が異なり、どちらもvolatile修飾されていない。

(6) (4)の間接演算オブジェクトを、次の(a)～(h)に示すいずれかの位置に記述している。
ただし、(e)～(h)は、(2)の変数と(4)の間接演算オブジェクトのどちらかがfloat型の場合に限る。

(a) 代入演算子(=)の左辺でかつ、他の演算子を使用していない。

(b) インクリメント演算子(++)またはデクリメント演算子(--)のオペランドでかつ、他の演算子を使用していない。

(c) 代入演算子(=)の右辺でかつ、他の演算子を使用していない。
ただし、左辺と右辺の型が異なる場合は、(2)の変数と(4)の間接演算オブジェクトのどちらかがfloat型の場合に限る。

(d) 代入演算子およびメンバ演算子(->または .)を除く、2項演算子の右側のオペランドでかつ、そのサイズが4バイトである。

(e) 代入演算子およびメンバ演算子を除く、2項演算子の左側のオペランド。

(f) &, sizeofを除く、前置演算子のオペランド。

(g) 制御文(for, while, do, if, switch, return)の制御式、継続条件式および式。

(h) 関数の実引数。

補足：
条件の組み合わせにより発生する問題内容が異なります。
発生条件の(6)に示している(a),(b)あるいは(c)の左辺と右辺の型が同じ場合のいずれかに該当すると、インターナルエラーが出力され、(c)の左辺と右辺の型が異なる場合または(d)～(h)のいずれかに該当すると、不正なコードが生成されます。

2.2 発生例

[ソースファイル例1]
[sample1.c]

```
long func1(void)
{
    long sl;                    /* 条件(2),(5) */
    unsigned long *ptr_ul;
    ptr_ul = (unsigned long *) &sl; /* 条件(3) */
    *ptr_ul = 0x80000000;       /* 条件(4)(a),(5),(6)(a) */
    return sl;
}
```

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

[ソースファイル例2]

[sample2.c]

```
-----------------------------------------------------------------
  signed char sc;                   /* 条件(2),(5) */
  unsigned char uc;
  struct uc_str {
       unsigned char  uc;
  };
  void
  func2(void)
  {
     struct uc_str  *ptr_ucs;
     ptr_ucs = (struct uc_str *) &sc; /* 条件(3) */
     uc = ptr_ucs->uc;              /* 条件(4)(b),(5),(6)(c) */
  }
-----------------------------------------------------------------
```

[ソースファイル例3]

[sample3.c]

```
-----------------------------------------------------------------
  long s1, s2;
  void
  func3(unsigned long ul)       /* 条件(2),(5) */
  {
     long *ptr_sl = (long*)&ul; /* 条件(3) */
     s1 = s2 + *ptr_sl;         /* 条件(4)(a),(5),(6)(d) */
  }
-----------------------------------------------------------------
```

[ソースファイル例4]

[sample4.c]

```
-----------------------------------------------------------------
  long
  func4(float f)                /* 条件(2),(5) */
  {
     long *ptr_sl;
     ptr_sl = (long *) &f;      /* 条件(3) */
     return *ptr_sl;            /* 条件(4)(a),(5),(6)(g) */
  }
-----------------------------------------------------------------
```

3. **回避策**

(1) -O4, -O5, - O6 および -O7の最適化オプションを指定しない。-Otimeまたは-Ospaceを使用する場合は、同時に -O0,-O1,-O2,-O3のいずれかを指定する。
(2) 発生条件(2)の変数または、(4)の間接演算オブジェクトにvolatile修飾を付ける。

[ソースファイル例1(sample1.c)の変更例]

```
---------------------------------------------------------------
  long func1(void)
  {
     long sl;
     volatile unsigned long *ptr_ul;   /* volatileにする */
     ptr_ul = (volatile unsigned long *) &sl;
     *ptr_ul = 0x80000000;
     return sl;
  }
---------------------------------------------------------------
```

[ソースファイル例2(sample2.c)の変更例]

[sample2.c]

```
---------------------------------------------------------------
  volatile signed char sc;            /* volatileにする */
  unsigned char uc;
  struct uc_str {
       unsigned char  uc;
  };
  void
  func2(void)
  {
     struct uc_str  *ptr_ucs;
     ptr_ucs = (struct uc_str *) &sc;
     uc = ptr_ucs->uc;
  }
---------------------------------------------------------------
```

[ソースファイル例3(sample3.c)の変更例]

```
---------------------------------------------------------------
  long s1, s2;
  void
  func3(volatile unsigned long ul)     /* volatileにする */
  {
     long *ptr_sl = (long*)&ul;
```

```
    s1 = s2 + *ptr_sl;
  }
```

[ソースファイル例4(sample4.c)の変更例]

```
  long
  func4(float f)
  {
    volatile long *ptr_sl;          /* volatileにする */
    ptr_sl = (volatile long *) &f;
    return *ptr_sl;
  }
```

---

4. **恒久対策**

本問題は、次期バージョンで改修する予定です。
