RENESAS ツールニュース

MAEC TOOL NEWS: MAECT-M3T-CC32R-030316D

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。

- デバッガでのアセンブリ言語インクルードファイル表示に関する注意事項
- ゼロ除算になる演算式の最適化に関する注意事項

---

1. デバッガでのアセンブリ言語インクルードファイル表示に関する注意事項

1.1 該当製品

M3T-CC32R V.1.00 Release 1 ～ V.4.00 Release 1

1.2 内容

デバッガ(M3T-PD32R等)で、ファイルを複数インクルードしているアセンブリ言語プログラムのソースファイルを表示させた場合、該当ファイル以外のオブジェクトコードのアドレスが表示される場合があります。
なお、デバッガでの表示が不正になりますが、生成されるコードには問題ありません。

1.3 発生条件

以下のすべての条件を満たすソースファイルをデバッガでソース表示させた場合、問題が発生します。(表示が不正になるソースファイルは、条件を満たすオブジェクトごとにひとつだけです。)
(1) ひとつのソースファイルから2つ以上のファイルをインクルードしている。
(2) インクルードファイル中に実行コードを生成する記述をしている。
(3) -g オプションを付けてアセンブルしている。

1.4 発生例

[ソースファイル例1]

[sample1.ms]

```
-----------------------------------------------------------------
  .section P,code,align=4
  .export _proc1
```

```
 _proc1:
   .include "proc1.inc"      ; 発生条件(1)
   .include "proc2.inc"      ; 発生条件(1)
   .end
```

[proc1.inc]

```
   LDI   R0,#1               ; 発生条件(2)
   JMP   R14                 ; 発生条件(2)
   ; (Dummy Line - 1)
   ; (Dummy Line - 2)
```

[proc2.inc]

```
   .export _proc2
 _proc2:
   LDI   R0,#2               ; 発生条件(2)
   JMP   R14                 ; 発生条件(2)
```

[コマンドライン例]

```
   % as32R -g -o sample1.mo sample1.ms       ; 発生条件(3)
```

(% はプロンプトを表します)

ソースファイル例1で、proc1.inc をデバッガでソース表示させた場合、3,4行目に命令コードが存在するかのようなアドレスが表示されますが、このアドレスは本来 proc2.inc の 3,4行目に表示されるべきものです。

[ソースファイル例2]

[sample2.ms]

```
   .section P,code,align=4
   .include "proc3.inc"
   .end
```

[proc3.inc]

```
   .export _proc4
```

```
 _proc4:
   .include "proc4.inc"      ; 発生条件(1)
   .include "proc5.inc"      ; 発生条件(1)
------------------------------------------------------------------

[proc4.inc]
------------------------------------------------------------------
   LDI   R0,#4               ; 発生条件(2)
   JMP   R14                 ; 発生条件(2)
   ; (Dummy Line - 1)
   ; (Dummy Line - 2)
------------------------------------------------------------------

[proc5.inc]
------------------------------------------------------------------
   .export _proc5
 _proc5:
   LDI   R0,#5               ; 発生条件(2)
   JMP   R14                 ; 発生条件(2)
------------------------------------------------------------------

[コマンドライン例]
------------------------------------------------------------------
   % as32R -g -o sample2.mo sample2.ms      ; 発生条件(3)
------------------------------------------------------------------
```

ソースファイル例2で、proc4.inc をデバッガでソース表示させた場合、3,4行目に命令コードが存在するかのようなアドレスが表示されますが、このアドレスは本来 proc5.inc の 3,4行目に表示されるべきものです。

1.5 回避策

以下の(1)～(4)のいずれかの方法で回避してください。

(1) デバッガでソース表示を行わないM3T-PD32R, M3T-PD32RSIMの場合は、ソース表示、MIX表示を行わず、逆アセンブル表示する。

(2) アセンブル時に-gオプションをつけない発生条件に該当するソースファイルには、-gオプションをつけずにアセンブルする。

(3) インクルードしているファイルの内容を展開する.include 擬似命令の箇所に、インクルードファイルの内容を展開する。

[ソースファイル例1の変更例]

[sample1.ms]

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

```
  .section P,code,align=4
  .export _proc1
_proc1:
  LDI   R0,#1              ; proc1.incの内容を展開
  JMP   R14                ; proc1.incの内容を展開
  ; (Dummy Line - 1)       ; proc1.incの内容を展開
  ; (Dummy Line - 2)       ; proc1.incの内容を展開
  .include "proc2.inc"
  .end
-----------------------------------------------------------------
```

(4) ひとつのソースあたりのインクルードファイルの個数を1にするソースファイル例2のように .include 擬似命令が連続している場合は、後の.include擬似命令の記述箇所を移動し、インクルードをネストさせるなどして、ソースひとつあたりのインクルードファイル数が1つになるようにする。
【注意】 インクルードのネスト数の制限(8個)を越えないようにしてください。

[ソースファイル例2の変更例]

[proc3.inc]

```
-----------------------------------------------------------------
  .export _proc4
_proc4:
  .include "proc4.inc"
                    ; このインクルードをproc4.incに移す
-----------------------------------------------------------------
```

[proc4.inc]

```
-----------------------------------------------------------------
  LDI   R0,#4
  JMP   R14
  ; (Dummy Line - 1)
  ; (Dummy Line - 2)
  .include "proc5.inc"       ; ここでproc5.incをインクルードする
-----------------------------------------------------------------
```

1.6 恒久対策

本問題は、次期バージョンの際に改修する予定です。

ページの先頭へ

2. ゼロ除算になる演算式の最適化に関する注意事項

2.1 該当製品

M3T-CC32R V.4.00 Release 1

2.2 内容

除算または剰余算を含むC言語プログラムで除数が0になる場合、最適化を有効にしたコンパイルが(OSにとって)不正な実行を行ってアプリケーションが異常終了することがあります。
※ Windowsでは「不正な処理」、Solaris, HP-UXでは「core dump」が発生します。

2.3 発生条件

以下のすべての条件を満たす場合に発生します。

(1) コンパイル時に -O4を含む最適化オプション（-O4, -O5, -O6, -O7, -Otimeのみ、または-Ospaceのみ）を指定している。

(2) ある関数中で、auto変数を定義し、かつその変数を定数で初期化している。

(3) (2)の関数中で、以下の(a)～(c)をすべて満たす除算または剰余算を行っている。
(a) 除数の式の値が常に0になる。
(b) 間接的な場合も含め、除数が(2)のauto変数の値を使っている。
(c) 被除数の式の値が常に同じ値である。

2.4 発生例

[ソースファイル例1: zdiv1.c]

```
int func1(void)
{
   int var1;                /* 発生条件(2) */
   int var2 = 1;              /* 発生条件(3c) */
   var1 = 0;                /* 発生条件(2)(3a)(3b) */

   return (var2 /= var1);      /* 発生条件(3a)(3b)(3c) */
}
```

[ソースファイル例2: zdiv2.c]

```
int var1;

int func2(void)
{
```

```
    int var2 = 0;             /* 発生条件(2) */

    var1 = var2;              /* 発生条件(3b) */

    return (2 % var1);        /* 発生条件(3a)(3b)(3c) */
  }
```

[ソースファイル例3: zdiv3.c]

```
  int func3(void)
  {
    int var1 = 2;             /* 発生条件(2) */
    int var2 = 3;             /* 発生条件(2)(3c) */
    var2 -= var1;             /* 発生条件(3a)(3b) */

    return (3 / --var2);      /* 発生条件(3a)(3b)(3c) */
  }
```

[コマンドライン例]

```
  % cc32R -c -O4 -o zdiv1.mo zdiv1.c      ; 発生条件(1)
  % cc32R -c -O4 -o zdiv2.mo zdiv2.c      ; 発生条件(1)
  % cc32R -c -O4 -o zdiv3.mo zdiv3.c      ; 発生条件(1)
```

(% はプロンプトを表します)

2.5 回避策

除数が0である除算・剰余算が発生しないようにプログラムを変更してください。
ANSI-Cにおいても、0による除算・剰余算の結果は未定義で、推奨される処理ではありません。

2.6 恒久対策

本問題は、次期バージョンアップの際に改修する予定です。