≫ 文法の曖昧さを理解して確実性と再利用性を高める

マイコンC言語 転ばぬ先のつえ

第31回　動的スタック領域の削減③…CPU内部レジスタの退避領域を削減する

鹿取 祐二

表1　関数レジスタ保証規則の一例
一部を記載．詳細を知りたい場合は各処理系のマニュアルを参照すること

項　目	RL78	H8	RX	SH
値を保証する汎用レジスタ	なし	ER2～ER6	R6～R13	R8～R15
値を保証しない汎用レジスタ	AX，BC，DE，HL（ES，CS）	ER0，ER1	R1～R5，R14～R15	R0～R7

本連載では，C言語の言語仕様（文法）の曖昧な部分と，それにより起こる問題を解説します．再利用性と効率が高く，安全かつ安心して使えるソフトウェアが開発できるようになることを目指します．

2024年2月号から，動的領域であるスタック領域の削減について紹介しています．前回は引数を減らすことでスタック領域を削減する方法を紹介しました．

今回は，CPU内部レジスタの退避領域削減について解説します．（編集部）

CPU内部レジスタの退避領域を削減する

● CPU内部レジスタの退避と復旧

まずはCPU内部レジスタの退避/復旧とは何かを少し細かく解説しておきます．

各処理系は関数内で，

- 値を保証する（使用する際には退避/復旧を行うCPU内部レジスタ）
- 値を保証しない（退避/復旧は行わずに使用するCPU内部レジスタ）

を決めています．

ルネサス エレクトロニクスの4つの処理系の場合，その規則は**表1**のようになっています．対象のCPU内部レジスタは，ほとんどが汎用レジスタですが，RL78のESやCSのように制御レジスタが対象の場合もあります．

▶値を保証しないレジスタを使うとマズい具体例

例えば，1000回繰り返すfor文を持つ関数をRL78とRXでコンパイルした結果を**リスト1**に示します．どちらのfor文もループ・カウンタは0から1000まで変化する記述ですが，最適化により1000（16進3E8）から0まで変化する結果となっています．

ここで着目すべき点はループ・カウンタとして使っている汎用レジスタです．RL78はAX，RXはR14を使用しています．RL78もRXも値を保証しない汎用レジスタが選択されています．値を保証する汎用レジスタでは退避/復旧が必要となり，その分はステップ損となりますから，それを避けるために値を保証しない汎用レジスタが選択されているのです．

続いて，for文の中に他の関数の呼び出しを記述した結果を**リスト2**に示します．他の関数を呼び出すと，呼び出した関数が値を保証しない汎用レジスタを使用する可能性があります．もし使用されてしまうと関数呼び出し前に設定した値は破壊され，失われてしまいます．そのため，今回の例におけるfor文のループ・カウンタ用の変数には，値を保証しない汎用レジスタは使えません．使用するなら，他の関数の呼び出し前後で値を退避/復旧して保護する必要があります．

リスト1　値を保証しない汎用レジスタが使われるケース
値を保証する汎用レジスタでは退避/復旧がステップ損になるので，それを避けるために値を保証しない汎用レジスタが選択されている

```
void func(void)
{
int i;

  for( i=0 ; i<1000 ; i++ )
    ;
}
```

（a）ソースコード

```
_func:
        movw    ax, #0x03E8
.BB@1:  addw    ax, #0xFFFF
        bnz     $.BB@1
        ret
```

（b）RL78の処理系でコンパイルした結果

```
_func:
        MOV.L   #000003E8H, R14
L11:    SUB     #01H, R14
        BNE     L11
        RTS
```

（c）RXの処理系でコンパイルした結果

第10回　演算子①…「関数呼び出し」に使う小括弧（2021年12月号）
第11回　演算子②…［　］と*の意味，++aとa++の違い（2022年1月号）
第12回　演算子③…実は本当に必要なケースは多くない！キャスト演算子（2022年2月号）