

ARMは記述した順番に命令実行されない

ARMマイコンでは、命令実行を効率化する為にCPU内部ハードウェアレベルで、命令実行の入れ替えを行っている。

従って、命令を記述した順番に命令実行やデータアクセスが必ずしも実行されない。特に、IOレジスタ操作などで実行順番が重要な場合には注意が必要。

尚、このような効率化はARMマイコンに限らず最近の高性能CPUでは当たり前に使われている技術。

バリア命令

ARMにはこのような事象に対処する為に、下記の命令を備えている。

データメモリバリア(DMB)命令

1. データメモリバリア命令よりも前に存在するすべてのメモリアccessが、先にアクセス完了する事を保障する。
2. メモリアccessに関連しない、命令を実行する事は可能。

データ同期バリア(DSB)命令

1. データ同期バリア命令よりも前に存在するすべてのメモリアccessが完了するまで、以後の命令を実行しないように、プロセッサを待機状態にする。
2. データ同期バリア命令が完了するまで、それ以降の命令が実行を完了したり割り込みマスクを変更する事はない。
3. 全てのキャッシュ、分岐予測およびTLBメンテナンス処理が完了する。

命令同期バリア(ISB)命令

1. 命令同期バリア命令はパイプラインをフラッシュし、命令キャッシュ（またはメモリ）から再フェッチする。

記述例

下記、ルネサスのサンプルプログラムから抜粋。

[init_cm3.c](#)

```
/*
*****
*****
* Outline      : initializing Cortex-M3 settings
* Function Name: init_cm3
* Description  : Initialize Cortex-M3 settings of RZ/T1.
                  Copy the Cortex-M3 code
*/
```

```
* Arguments      : none
* Return Value   : none
*****
*****/
void init_cm3( void )
{
    volatile unsigned char *p_org;
    volatile unsigned int  psize;
    unsigned char *p;

    /* Copy the Cortex-M3 program code */
    p =(unsigned char *) ( 0x04000000+0x0 );// M3 RAM start address for
Cortex-R4
    p_org = __section_begin("CM3_SECTION");
    psize  = __section_size("CM3_SECTION");
    memcpy(p, (char *)p_org, psize);

    asm("dsb");                // Ensuring data-changing

    /* Release the Software reset 2 (Cortex-M3) */
    R_RST_WriteEnable();
    asm("isb");                // Ensuring instruction-changing
    SYSTEM_SWRR2_LONG = 0x00000000; // Release Software reset 2
    asm("isb");
    R_RST_WriteDisable();
}
```

1. EWARMでは、__DSB() __ISB() などのようにコンパイラ組み込み関数を利用して記述できる。
(#include <intrinsics.h>が必要)

補足

1. I/Oデータアクセス順番を確保する為に、DSB命令などを使わずに、該当レジスタのダミーリードで記述してある例もある。(多少、命令の処理速度が速くなると思われる)

参考

1.  メモリバリア 2. マルチスレッドとメモリ同期 3. Linux Kernel ~ 同期機構 ~ 4. ロード/ストア命令の追い越し例

From:

<https://yokoweb.net/dokuwiki/> - 新石器Wiki

Permanent link:

<https://yokoweb.net/dokuwiki/programing/arm-programing/arm-dmb-dsb-isb?rev=1584574327>

Last update: 2020/03/19 08:32

