第6章 仮想記憶だけじゃない，組み込みシステムのセキュリティ強化に応用できる

MMUのメモリ保護機能を使ったプログラミング

中森 章

近年の組み込み機器の複雑化に伴い，MMU機能を使った大規模システムが作られるようになった．MMUを使えば，プロセスごとに独立した仮想アドレス空間を持つことができる上，一つのプロセスが暴走してもほかのプロセスやOSカーネルに影響を与えないという利点がある．また，ページ・スワップを使えば，少ないメモリを有効に使うこともできる．

本稿では，OSの製作や，OSのない環境でメモリ保護機能を使うのに必要なMMUについて，ARM926コアを対象として解説を行う．（編集部）

組み込みプログラムが高度になり，また複雑になるにつれ，Symbian OS，Windows CE，Linuxといった Operating System（OS）の必要性が高まっています．特に出来合いのアプリケーション・プログラムは何らかのOSの下で動作することを前提としているのが普通です．MMU（Memory Management Unit）はOSを動作させるために必須の機能といわれています．私達が一般的に入手可能な評価ボードは汎用的な使用方法が想定され，OSが載ることを前提としており，そこに搭載されるプロセッサもMMU内蔵のものが使われます．

一方，MPU（Memory Protection Unit）機能を持つプロセッサは，OSを必要としない特定用途向けのSystem On a Chip（SoC）やApplication Specified Integrated Circuit（ASIC）に多く内蔵されます．しかし，私達がそのようなSoCやASICをプログラムする機会は少ないと思われるので，MPUの説明は今回は割愛します．

以降，私達が最も目にする機会が多いと思われるARM926コアのMMUについて説明します．プロセッサ・コアをARM926に特定していますが，ほかのARM9系のプロセッサ（MMUを持つもの）に対しては同じような方法でMMUを操作できます[注1]．

1. MMUとは何か

本誌の読者にMMUの解説をするのは釈迦に説法かもしれませんが，MMUの機能について簡単に説明しておきましょう．

MMUには大きく分けて，

- アドレス変換
- メモリ保護

の二つの機能があります．

● アドレス変換――仮想アドレスを物理アドレスに変換する

これは仮想アドレスを物理アドレスに変換する機能です．プログラマに見えるアドレスが仮想アドレスです．もちろん，プロセッサも仮想アドレスに基づいて動作します．一方，メモリ側から見える本当のアドレスが物理アドレスです．

アプリケーション・プログラムのプログラマは，通常はプログラムがメモリのどこに存在するかを意識することなく，すべてのプログラムは同一のアドレス（例えば0番地）から始まっているものとしてプログラミングします．「アプリケーションAは0x1000番地から始まり，アプリケーションBは0x2000番地から始まる」などと考えてプログラミングすることは，まずありません．すべて0番地から始まるという仮定でプログラミングします．逆にいうと，アプリケーションの数だけ同一の仮想アドレスが存在します．

しかし物理アドレスの場合は，メモリというデバイスのアドレスは一意なので，同一のアドレス上では一つのプログラムしか実行できません．ところが現実には，幾つものアプリケーション・プログラムが一つのOSの下で同時に動いています．これはOSが，各プログラムの存在するメ

注1：とはいえ，ARM920TやARM922TのTRM（Technical Reference Manual）にはMMUがARMv4準拠となっているのが気にかかる．ARM926のTRMでは，MMUはARMv5準拠となっている．

モリ領域をずらしてメモリ上に配置しているからです．OSは，プログラムの実行する仮想アドレスを実際のメモリ上の物理アドレスに読み換えてプログラムを実行させます．これがアドレス変換です．アドレス変換はプログラムでは意識されず，OS内で自動的に処理されます．

アドレス変換の利点は何でしょうか？ 一つは，上述したように，複数のプログラムを同時に（正確には時分割で）一つの物理メモリで動作が可能になることです．これをマルチタスクといいます．このほかに，メモリに入りきらない程巨大なプログラムを，一部分ずつ交代して物理メモリに配置しての実行も可能になります．これはページ・スワップといいます．OSはマルチタスクとページ・スワップを適宜に組み合わせ，物理メモリを効率良く利用しながら，複数のプログラムを適正に動作させます．

● メモリ保護――特定の領域へのアクセスを禁止する

これはあるプログラムに対して，特定の仮想アドレス領域へのアクセスを禁止する機能です．逆にいえば，特定の領域に対して，「リードのみ可能」，「ライトのみ可能」，「リードとライト可能」，「リードもライトも不可能」のような属性を持たせることです．これにより，アプリケーション・プログラムが暴走しても，OSやほかのプログラムに影響を与えないようにできます．本稿では説明しませんが，ARMでいうMemory Protection Unit（MPU）も似たような機能を有しています．

伝統的なMMUでは，プログラムの実行可能性をリード可能性と同一視するものが多かったのですが，近年のMMUはリード可能性と実行可能性を区別するのが流行です．アプリケーション・プログラムがスタックを意図的にオーバフローさせて例外を発生させ（特権モードに移行する），特権領域にあるスタック上のデータを命令として実行すれば，アプリケーション・プログラムを特権レベルで動かせます．このような悪意を持ったプログラムはOSの実行を阻害したり停止させたりするからです．

ARMアーキテクチャではARMv5までは実行許可属性がなかったのですが，ARMv6以降は実行許可属性が追加されました．

2 ARMのアドレス変換

● アドレス変換の仕組み

アドレス変換とは，仮想アドレスと1対1に対応する物理メモリ上のアドレス変換情報（これをページ・テーブルという）を参照して，仮想アドレスから物理アドレスを生成することです．

● 1レベル方式と2レベル方式の2種類がある

ARMのアドレス変換の方式には，1レベル方式と2レベル方式があります．1レベル方式は1回のページ・テーブル参照で物理アドレスを生成します．2レベル方式は2回のページ・テーブル参照で物理アドレスを生成します．このような物理メモリ上のページ・テーブル参照を，ページ・テーブル・ウォークと呼びます．

ARMでは仮想アドレスとして，CP15のレジスタ13に格納されているFCSE PID（Fast Context Switch Extension Process ID）と呼ばれるプロセスIDを考慮したMVA（Modified Virtual Address）という修正仮想アドレスを使用してアドレス変換を行います．今の時点ではFCSE PIDを考える必要はありません．

コラム ARMアーキテクチャのMMUとデータ・キャッシュの関係

筆者がARMアーキテクチャに触れてまず最初に驚いたことはMMU（Memory Management Unit）をONにしないとデータ・キャッシュが使えないという事実です．それまでMMUとキャッシュは異なる概念だと思っていたのに，その既成概念がぶち壊されてしまいました．ARMに限らず一般的なプロセッサでMMUを使用するためにはページ・テーブルの設定など多くの設定をする必要があります．たかがデータ・キャッシュを使用するためになぜそこまでしなくてはならないのか理解に苦しみました．

ARMもプロセッサの種類によってはMMUを持たないものもあります．しかし，その場合の多くはMMUの代わりにMPU（Memory Protection Unit）というメモリ保護機構を備えています．この場合でもMPUをONにしないとデータ・キャッシュが使えません．