第6章 マルチコア環境で発生する命令順序入れ替えバグを防ぐ…RISC-Vを例に

メモリ・モデルの基本と挙動の制御方法

佐伯 学哉

コア1	コア2
store(x, 1)	store(y, 1)
ry = load(y)	rx = load(x)

図1　命令実行順の入れ替えを検討するための疑似アセンブリ言語コード

コア1	コア2
store(x, 1)	
	store(y, 1)
ry = load(y)	
	rx = load(x)

図2　直感的な命令実行順

マルチコアではメモリ・アクセスの順序入れ替えに注意

近年，デスクトップPCではマルチコア・プロセッサの搭載が標準的となり，並列実行を前提とした開発が広く行われています．RISC-Vプロセッサを採用したボードでもマルチコア・プロセッサの製品が登場し，組み込み分野でもマルチコア化が進んでいます．

マルチコア環境では，メモリ命令[注1]がプログラム順とは違う予期されない順番で実行されることがあります．そのためマルチコア環境では，メモリ命令が実際にどの順序で実行されるかを定義するメモリ・モデルを理解する必要があります．

本稿では，RISC-Vの例を中心に，メモリ・モデルについての基礎的な概念からフェンス命令などのアセンブリでの同期手段を紹介します．特にCPUコアが複数あるハードウェアでは，これらの知識がないと解決困難なバグに遭遇する可能性があります．

最適化によって実行結果が変わることがある

● 命令の実行順を入れ替える影響

メモリ・モデルについて知るには，メモリ命令のリオーダリング（順序の入れ替え）というCPUの動作を理解する必要があります．

図1に疑似アセンブリ言語プログラムを示します．このプログラムを含め，以降xとyは互いに異なるアドレスを意味し，rxとryは，それぞれ違うレジスタを表していると考えてください．xとyのアドレスで指し示されるメモリの内容は，最初はゼロで初期化されているものとします．

2つのCPUコアが各命令を実行するタイミングとして，あり得るパターンを考えます．例えば**図2**の実行例では最終的な（rx，ry）の値は（1，1）です．

他の実行パターンを**図3**に示します．パターン1の結果は（0，1），パターン2の結果は（1，0），パターン3～パターン6の結果は（1，1）であり，あり得る最終結果は3つしかないように思えます．

しかし，これだけではありません．RISC-V，Arm，x86，どのCPUにおいても，**図4**に示す奇妙な順序でメモリ命令が実行されることがあります．このパターンでは，コア1上でstoreとloadの実行順が元のプログラム順から入れ替わって見えます．

マルチコア・プロセッサでは，メモリ命令がプログラムで書かれた順序で実行されないことがあります．これをメモリ命令のリオーダリングと呼びます．これにより元の6パターンでは起こりえなかった結果（0，0）になる場合があります．これが，メモリ命令のリオーダリングと呼ばれる動作です．

最近のプロセッサは，必ずしもプログラムで書かれた順序で命令を実行していません．性能を最大化するために，複数の命令を同時に実行したり，実行順序を並べ替えるような最適化を行ったりします．つまり，メモリ命令を含むあらゆる命令が，さまざまな理由でリオーダリングされる可能性があるわけです．一般に，シングル・コアのプロセッサでは，最適化の影響で実行結果が変わらないようになっています．しかし，マルチコア環境では最適化による副作用が起きることがあります．

注1：メモリ命令とはメモリ・アクセスを伴うCPU命令を指します．