第3章 プロセッサ間の同期をソフトウェアで実現する

マルチプロセッサのためのアセンブリ命令

中森 章

マルチプロセッサ・システムを構築する際には，資源の排他制御をどのような手法で実現するのかが問題となる．近代的なプロセッサでは，排他制御を1命令で実現できるアセンブリ命令が用意されている．本稿では，これらのアセンブリ命令について解説する．（編集部）

本稿では，マルチプロセッサ間で同期を取るために用意されたアセンブリ言語の命令について解説します．また，命令だけにとどまらず，その使い方を知ることにより，資源の排他制御を実現する方法などについても解説します．

1. マルチプロセッサ・システムにおける排他制御

マルチプロセッサ・システムを構築する上では，いかにして排他制御を行うかが問題となります．純粋にソフトウェアだけで排他制御を行うことも可能ですが，ハードウェアによる支援があると，高速な排他制御（1クロック～数クロックで実行）が可能です．そして，現在のマルチプロセッサ，マルチコアCPUのほとんどがハードウェア的な排他制御機構を持ち，そのためのアセンブリ命令を備えています．

● 同期のためのソフトウェア処理

マルチプロセッサ・システムにおいては，共有バスがメイン・メモリ（共有メモリ）にアクセスする唯一のアクセス手段です．メモリの排他制御は，バスの使用権を獲得したプロセッサがメモリ・アクセスを独占し，ほかを閉め出せばいいのです．この処理は不可分（アトミック）なスワップ命令（データ交換命令）があれば簡単に実現できます．

不可分とは，あるプロセスがメモリにアクセスしている間にほかのプロセッサからのメモリ・アクセスがないことをいいます．あるいは，ほかのプロセスがアクセスしようとしてもそれが禁止されている状態をいいます．実際には，（バス）ロックという端子を活性化して外部回路に通知し，ハードウェア的にほかのプロセッサのアクセスを禁止します．

話が横にそれましたが，ソフトウェア的には，排他制御のために，各共有資源に対して，1対1に対応するロック変数を用意します．つまり，ハード・ディスクには変数A，ディスプレイには変数B，プリンタには変数Cといった具合です．ここで，ロック変数が0の場合はそれに対応する資源が未使用，0でない場合はそれに対する資源が使用中であることを示すものとします．プロセスは早い者勝ちで，0以外の値をロック変数に書き込むことで，その資源を使用する権利を得ます．例えばプリンタを使用したい場合，変数Cが0なら誰も使用していないので使用可能，0以外なら使用中なので待つ（またはあきらめる）といった処理になります．

初期のLinuxでは，共有資源ごとにロック変数を持つのではなく，ただ一つのロック変数で排他制御を実現していたそうです．つまり，ロック変数にアクセスできたプロセスはすべての共有資源がアクセスできるようになるのです．こういう単純な制御でもシステム的には破たんしない程度の実行性能が得られる（昔は得られていた？）のでしょうね．

● テスト・アンド・セットによる排他制御

排他制御を実現する最も単純な処理が「テスト・アンド・セット」です．これは，ロック変数をリードし，その値が0ならそこに1（または0以外の値）を書き込みます．ロック変数が0でなければ，書き込みは行いません．通常，ロック変数の前の値がリードの結果として返されるので，それが0ならロックが成功したことを示します．0でなければロックは失敗したことになります（つまり，ほかのプロセスが使用中であることを意味する）．**図1**にテスト・

KEYWORD ―マルチプロセッサ，排他制御，アセンブリ命令，テスト・アンド・セット，コンペア・アンド・スワップ，フェッチ・アンド・アッド，ロード・リンクト，ストア・コンディショナル

アンド・セットを利用した相互排除を示します．ここで，ロック変数が0になるのを待つループを「スピン・ウェイト」，または「ビジー・ウェイト」と呼びます．そのため，この排他制御方式をスピン・ロック方式と呼びます．

● セマフォによる排他制御

スピン・ロック方式は，ほかのプロセスがクリティカル・セクション（共有資源を独占できる時間的・空間的領域）を実行しているとき，同じ資源を使おうとする別のプロセスがループしながら待つ点が非効率です．スピン・ウェイトを行わない同期機構としてはセマフォがあります．

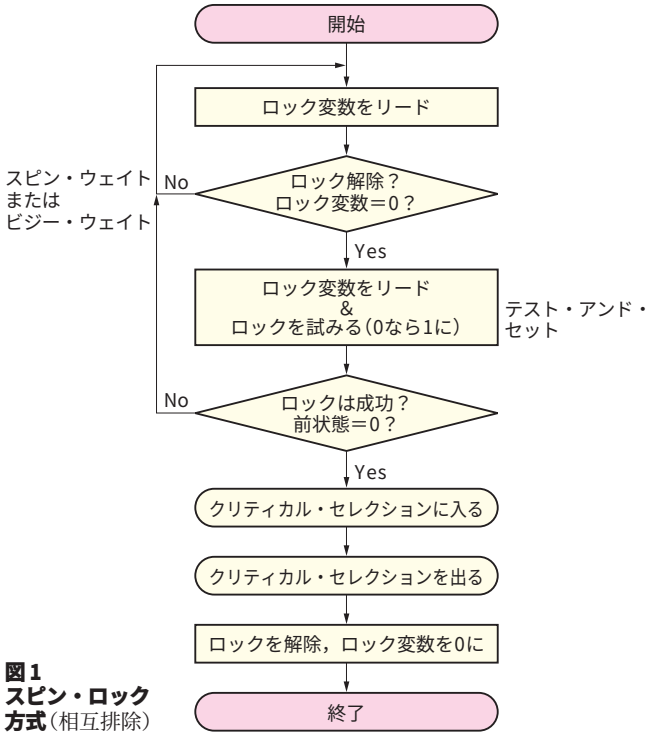
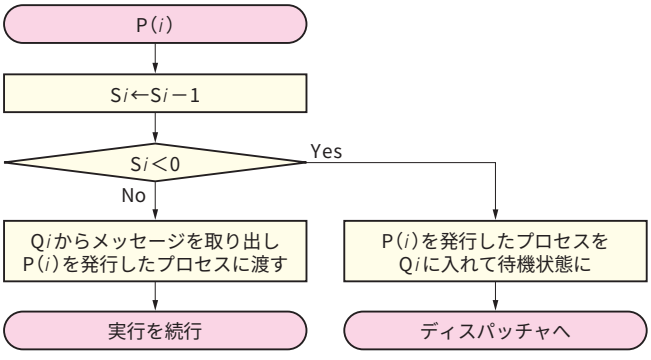


図1 スピン・ロック方式（相互排除）

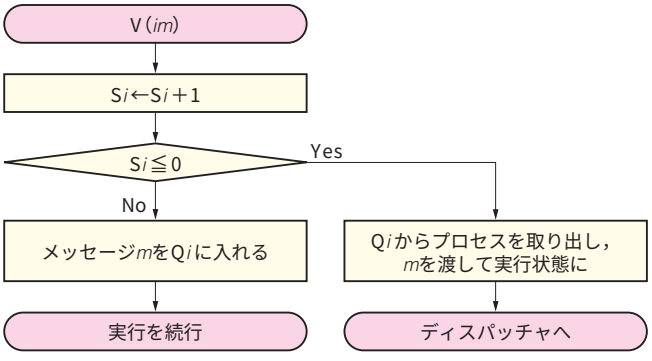
セマフォとは，鉄道で用いられる腕木式信号機のことです．信号機が「進め」であれば資源が獲得できるし，「止まれ」であれば資源を獲得できません．セマフォの実体は一種の共有フラグで，同期をとるプログラム同士がこのフラグに注目し，フラグの変化に応じて処理を行って同期を実現します．構造化プログラミングの提唱者であるEdsger Wybe Dijkstra氏はセマフォに対するP操作，V操作というものを定義しました．P操作は事象の待ち合わせ，V操作は事象の発生（およびメッセージ）の通知を行います．セマフォを実現する操作は「フェッチ・アンド・アッド」と呼ばれます．これはメモリに対する任意の数（負の数も可）の加算を不可分に行う機能です．

P操作，V操作のP，Vという名称の由来は，Dijkstra氏の母国語であるオランダ語にあります．Pは「proberen（試みる）」と「verlagen（減じる）」の合成語である「prolagen」が語源です．つまり，セマフォの値を減らす働きをします．Vは「verhogen（増やす）」または「vrygeven（解放する）」が語源です．つまり，セマフォの値を増やします．由来を知っていれば，以下に説明する，それぞれの動作も理解しやすいと思います．

セマフォiは，共有資源iについて定義され，負の値も取るカウンタS_iと待ち行列Q_iから構成されます．これに対するP操作，V操作の流れを**図2**に示します．簡単にいえば，P操作で資源が得られない場合は，そのP操作を行ったプロセスは待機状態となってQ_iに格納され，後で起動されるのを待ちます．これをサスペンド状態といいます．V操作は，待ち行列Q_iの中にプロセスがもしあれば，それを起動します．V操作でプロセスを起動するのはOSの役割です．



（**a**）資源iに対するP操作



（**b**）資源iに対するV操作

図2 P操作とV操作