2U-4

GMICRO/200アーキテクチャの機能の考察
―待ち行列処理命令―

野尻徹[*1] 海永正博[*1] 川崎郁也[*2]
＊1 日立製作所システム開発研究所、＊2 日立製作所武蔵工場

1．はじめに

GMICRO/200はＴＲＯＮ標準チップ仕様に準拠した最初のプロセッサチップである。ＢＴＲＯＮ，ＩＴＲＯＮなどのＯＳを高速に実行させるための支援機能についても十分考慮されている。ＯＳを高速に実行させるには、第１次近似としてシステムミックスが高速であればよい。しかしアーキテクチャのサポートがあって初めて妥当な速度で動作できるようなＯＳの機能要素もありえる。ＯＳの特徴的機能として、以下のようなものが挙げられ、これらは全てアーキテクチャ上の支援を要請している。

(1) リソースの管理
(2) 多重仮想空間、記憶保護

本稿ではこの内リソースの管理機能に係わる待ち行列処理に的を絞り考察を加えることにする。

2．キュー型

リソースの管理を待ち行列にマッピングして実現するオペレーティングシステムが一般的であり、GMICRO/200ではこのために、双方向にリンクされた待ち行列構造(キュー型)をサポートしている。

このキューは、各エントリの最初の４バイトに前方リンクが、次の４バイトに後方リンクが配置され、その後ろに個別エントリに係わる情報が配置された構造をしている(厳密には、前方リンクの前の部分にも個別エントリに係わる情報が配置されていてもよい)。

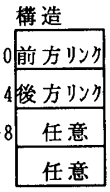

図１．キューエントリの構造

キュー全体の構造は以下のようになる。

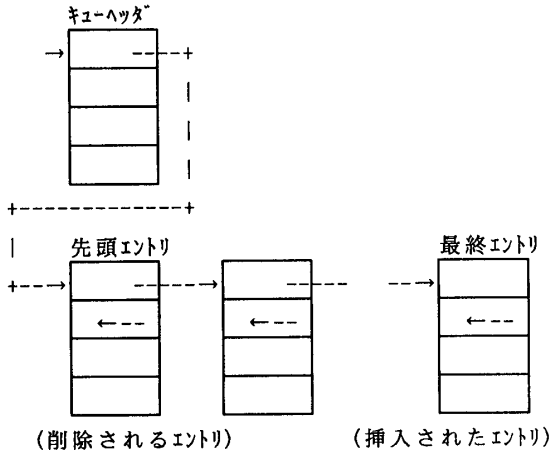

図２．キューの全体構造

3．キュー命令

キュー命令として以下の命令を提供している。

(a) QINS entry,queue
　　queueの直前(最終)にentry挿入
(b) QDEL queue,dest
　　queueの直後(先頭)エントリを削除
(c) QSCH
　　キューエントリのサーチ。
　　オペランドは汎用レジスタ

GMICRO/200のキュー命令においては、サービス要求エントリとキュー全体を管理するためのキューヘッダを明確に区別している。そしてキュー命令のqueueオペランドとしてキューヘッダを指定するのが標準の想定である。挿入の場合、キューの最終エントリとして新エントリが挿入され、削除の場合、先頭のエントリが削除されることになる（図２参照)。

QSCH命令は優先順位のついた待ち行列に対応するためのものである。サーチ範囲はR0とR2で指定される。そして一般にこの２つのレジスタは最初、ともにキューヘッダを指しているのが一般的である。この場合サーチ対象のキューエントリはキューヘッダ

Queue instructins of GMICRO/200
Tohru Nojiri, Masahiro Kainaga, Ikuya Kawsaki
Hitachi LTD.

の直後(先頭)エントリからキューヘッダの直前(最終)エントリまでとなる。

QSCH命令の機能をC言語風に記述すると以下の様になる。即ちR0で指定したキューエントリ(一般にキューヘッダ)の次のエントリ(一般に先頭エントリ)からサーチを開始し、R2で指定したキューエントリ(一般にキューヘッダ)に到達することで、サーチが失敗終了する。R1はサーチ中エントリの直前のエントリを指し(図3)、サーチしたエントリの削除に利用できる。

[QSCH命令のオペレーション]

```
QUEUE *R0,*R1,*R2; /* 現、前、最終 */
SS int R3;         /* 比較値、SS:サイズ */
int R5;            /* オフセット */
QSCH.SS
{
   while(1){
      R1=R0;R0=R1->fp;
      if(R0==R2)    exit(); /*失敗*/
      if(*(SS int*)((char*)R0+R5).eeee.R3)
                    exit();/*成功*/
      if(割込み有り？) 割込みの受付;
   }
}
```

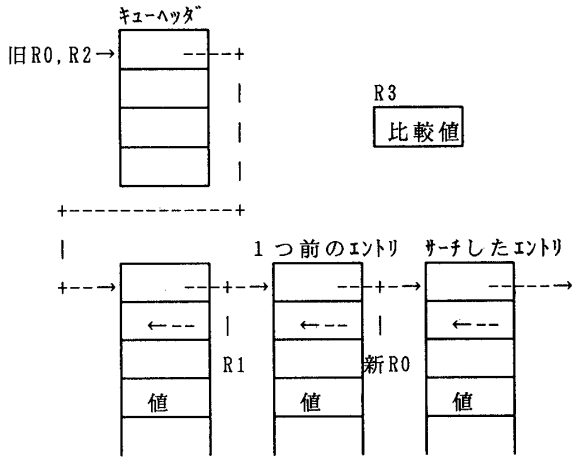

図3．QSCH命令の機能

ここで、照合条件(.eeee.)として、ストリングサーチの割り出し条件と同じものが指定できる。即ち、個別キューエントリ内の整数系ロケーションの値と、指定比較値との算術比較(一致、大、小など)の真偽結果を照合条件として指定できる。

サーチ時間が長大(ソフトウェアのバグによる無限大を含む)になる可能性があるので、QSCH命令は割り込みによる命令実行の中断、再開を許している。

4．考察

キューの挿入/削除命令として以下の機能を提供するものもある。

(a) QINS' entry,queue
　　queueの直後にentry挿入
(b) QDEL' entry,dest
　　指定entryを削除

両者の命令体系において、間接アドレッシングが強力であるとすれば、他方の機能をもう一方が容易にシミュレートできる。例えば以下の通り('*'を間接指定子とする)。

```
QINS  entry,queue ==> QINS' entry,*(queue+4)
QDEL  queue,dest  ==> QDEL' *queue,dest
QINS' entry,queue ==> QINS  entry *queue
QDEL' queue,dest  ==> QDEL  *(queue+4),dest
```

しかし、キューを先入れ先出しの本来の用途で利用するのが大多数であるとすれば、G_{MICRO}/200の提供する機能の方が好ましいと考える。G_{MICRO}/200の場合、スタティックな(アドレスが固定値)エントリであるキュヘッダqueueを間接アドレッシング無しに直接に指定して、エントリの挿入/削除が行えるからである。

QSCH命令の機能は、挿入、削除命令との素直な連携が可能である。 特に削除すべきエントリを順次にサーチし、サーチしたエントリを削除し、後続のサーチを続行するという連続的サーチにおいて、QSCH命令のオペランドである汎用レジスタR0～R5の内容を全く変更することなく連続的なサーチが可能である。QSCH命令では、サーチしたエントリをR0に返し、1つ前のエントリをR1に返しており、この2つの出力レジスタを挿入/削除のqueueオペランドとすることで、以上のことを素直に実現できる。

参考文献

1) 坂村：TRON Project 1987