計算機MELPIP-1の並列処理について

Parallel Processing by a Multi-Processor MELPIP-1

金森　直　　房岡　璋　　平山正治
Tadashi Kanamori　Akira Fusaoka　Masaharu Hirayama
三菱電機株式会社中央研究所
Central Research Laboratory, Mitsubishi Electric Co.Ltd

1　はじめに

最近のＬＳＩ技術の進展にともない小規模なプロセッサを複数個結合したマルチプロセッサシステムが汎用大型計算機では妥当なコストパフォーマンスが得られていない分野に有効となる可能性が生まれてきた．このマルチプロセッサシステムで処理を行なう場合，対象とする問題の持つ並列性に適合しうる柔軟で効率的な基本アーキテクチャが望ましい．それには
(1)個々のプロセッサが相互作用を行ない，問題に応じた制御構造を動的に形成できる．
(2)個々のプロセッサが種々の構造のデータを効率的に扱うことができ，問題に応じたデータの処理ができる．
ことが必要である．

このような観点から我々はマルチプロセッサMELPIP-1の作成を行なった．MELPIP-1ではLesser[1]のマイクロプロセスの概念を簡単化して導入し，実行時の一連のマイクロ命令の系列であるマイクロプロセスの集まりとそれらの相互作用により問題に適合した並列処理を行なう．これらのマイクロプロセスの管理は1つのプロセッサに集中されている．

このような制御集中型マルチプロセッサはlogical machineと呼ばれ[2]，プロセス間の相互作用は効率的に行なえるが，サービスの共用によるコンフリクトが問題になる．これを避けるためMELPIP-1では，制御をパイプラインによって高速に処理している．

また MELPIP-1ではバイトデータ列を基本的なデータとしており，これらのデータは演算用のプロセッサ（同一機能4台）でパイプラインで処理されている．

本稿ではMELPIP-1の並列処理機構と並列処理マイクロプログラムを中心に報告する．

2　MELPIP-1の概観

2.1　ハードウェアの構造と動作

MELPIP-1システムは図1に示すように4台の演算プロセッサ（ＡＰ），各1台の制御プロセッサ（ＣＰ），メモリプロセッサ（MP）およびホストインタフェイス（ＨＩ）から構成されている．これらのプロセッサはコントロールバスとデータバスによって接続され，相互に情報の送受を行ないながら各々独立した動作をする．

演算プロセッサ（ＡＰ）はＣＰから1個のマイクロ命令を受け取り，この命令を実行し，結果のステータスをＣＰに返すというサイクルを繰り返す．ＡＰそれ自体はマイクロプログラムのシーケンス制御機能をもたず，命令の実行終了ごとに，ＣＰに次のマイクロ命令の要求を出す．MELPIP-1は4台のＡＰで構成されているが，各ＡＰで独立した2本のマイクロプログラムを時分割で実行するようにしているのでシステム全体で8本のマイクロプログラムの同時実行が可能となっている．

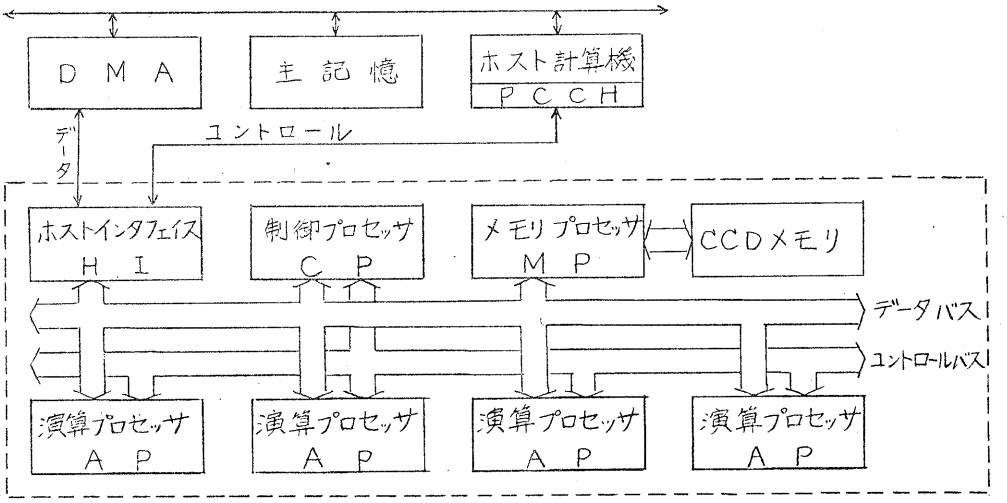

図１　システム構成

制御プロセッサ（ＣＰ）はＡＰの要求に応じてＣＰ内にある制御記憶より次に実行するマイクロ命令を読み出してＡＰに送るというシーケンス制御を行なう．また後述するようにマイクロプロセス間の同期などプロセスの制御を行なっている．

メモリプロセッサ（ＭＰ）はＡＰからの主記憶リクエストに応じて主記憶データのＲ／Ｗをコントロールする．ＭＥＬＰＩＰ-１の主記憶はＣＣＤメモリと高速ＲＡＭで階層的に構成することにより大きな容量の主記憶の実現を試みた．

コントロールバスはＣＰとＡＰを結合するバスでありマイクロ命令と命令実行結果のステータス情報の転送に使われる．

データバスはＭＥＬＰＩＰ-１のすべてのプロセッサを結合するバスであり主記憶データをはじめ，各プロセッサのメモリ，レジスタの転送に使われる．

ＭＥＬＰＩＰ-１は ホストインタフェイス(ＨＩ)を経由してホスト計算機と（他の入出力装置と同様の方法で）接続されている．ホスト計算機は入出力機能をもたないＭＥＬＰＩＰ-１の代わりに入出力動作を行なって，ユーザーとのインタフェイスをとるソフトウェアとＭＥＬＰＩＰ-１を管理するソフトウェアを実行する．

2.2　サポートソフトウェアの概要

ＭＥＬＰＩＰ-１のサポートソフトウェアとしては現在，ＭＥＬＰＩＰ-１用のクロスアセンブラがＭＥＬＣＯＭ 7500上で動作している．（後述の並列処理プログラム参照）

また ＭＥＬＰＩＰ-１は演算専用プロセッサであり入出力の機能を持っていないため，各入出力動作，およびユーザとのインタフェイスは，ホスト計算機ＭＥＬＣＯＭ 350／50が行なう方式をとっている．現在ホスト計算機で，ＭＥＬＰＩＰ-１を管理するソフトウェア(ＰＩＰＯＳ)が動作中である．

ＭＥＬＰＩＰ-１の利用者はシステムタイプライタ（ＳＴＷ）からコマンドを入力して

①ＭＥＬＰＩＰ-１に行なわせる一まとまりの仕事（マイクロタスク）を登録する．（Ｔコマンド）

②ＭＥＬＰＩＰ-１のデータを紙テープ読み取り装置（ＰＴＲ）あるいは直接ＳＴＷから入力してＭＥＬＰＩＰ-１の主記憶に書き込む（Ｗコマンド）

③ＭＥＬＰＩＰ-１のマイクロプログラムをＰＴＲあるいは直接ＳＴＷから入力してＭＥＬＰＩＰ-１の制御記憶に書き込む．（Ｗコマンド）

④ＭＥＬＰＩＰ-１のマイクロプロセスからホスト計算機に仕事を依頼したときホスト計算機上で走るソフトウェアを登録する．（Ｗコマンド）

以上を準備した後，STWからのコマンドによりマイクロタスクを起動する．（Sコマンド）

マイクロタスクが起動されたとき，CPは自動的に1個のマイクロプロセス（これをルートプロセスと呼ぶ）を生成し，APを割り当てマイクロプログラムの実行を開始する．これ以降はユーザが作ったマイクロプログラムの中で新しいマイクロプロセスを生成して並列処理を進めていく．マイクロプロセスはプロセスを終了させる命令を実行すると消滅し，ルートプロセスが終了したときは，マイクロタスクの終了とみなしCPがホスト計算機に報告する．

MELPIP-1において実行されるマイクロプロセスは入出力動作などを行なうときに，ホスト計算機にあるソフトウェアを起動して処理を行なわせることができる．また，このソフトウェア動作が終了すると，ホスト計算機からの割り込みという形でMELPIP-1のマイクロプロセスに伝えられる．（Iコマンド）

この他，マイクロタスクの強制終了（Aコマンド），MELPIP-1のメモリ，レジスタの読み出し（Rコマンド），PIPOSの管理するマイクロタスク，主記憶，制御記憶の状態の表示（Dコマンド）などがある．

MELPIP-1を利用する際のシステム動作の概要を図3に示す．

特にMELPIP-1のリソース（主記憶，制御記憶，演算プロセッサ）に余裕がある限り，複数のマイクロタスクをMELPIP-1上で走らせることができるが，そのためにPIPOSは複数のマイクロタスクの管理と処理ができるようになっている．

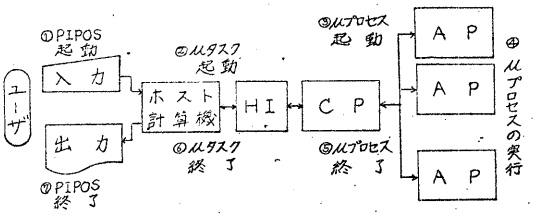

図3　MELPIP-1システムの動作

2.3 システムの基本性能

(1)基本速度

MELPIP-1は演算プロセッサでは次のような速度で演算を行なう．

マイクロ命令のセット	50 nsec
マイクロ命令のデコードと初期設定	175 nsec
パイプライン動作の1ステージ	70 nsec
CPへの終了割り込み	105 nsec

従って普通のマイクロ命令ではBバイトのデータをパイプラインで処理すると総計540+70*B nsec 要する．(2) 参照）

また制御プロセッサでは次のような速度で制御を行なう．

APからの終了割り込みの受け付け	170 nsec
パイプライン動作（5ステージ）	500 nsec
マイクロ命令の送出	100 nsec

従ってAPが次のマイクロ命令を受け取るまでに最低770 nsec要する．

コマンド	機能
T［主記憶サイズ，制御記憶サイズ，μプロセス数；名前］	μタスクの登録とリソースの確保
X［μタスク番号］	μタスクの削除とリソースの解放
S［μタスク番号，スタート番地］	μタスクの起動
A［μタスク番号］	μタスクの強制終了
I［μタスク番号，μプロセス番号］	μタスクのμプロセスへの割り込み
R［対象メモリ/レジスタ名と番地；転送ワード数］	メモリ/レジスタの読み出し
W［対象メモリ/レジスタ名と番地；転送ワード数(；データ)］	メモリ/レジスタの書き込み
D	μタスク，メモリの状態表示
E	PIPOSの終了

図2　MELPIP-1管理ソフトウェア（PIPOS）のコマンド

(2)可変語長パイプライン処理の評価

演算プロセッサでの演算はシフタ，ＳＰＬ，ＡＬＵと呼ぶ2段構成，各8ビット巾の演算装置で行なわれる．これらは以下に示す機能を持っている．

シフタ…０～4ビット左右ロジカルシフト
　　０～4ビット左右ローテートシフト
ＳＰＬ…3ビットデコード
　　上位の連続する０，また１のビットの個数カウント
　　上位ビットのマスク
ＡＬＵ…加減算，キャリ付加減算（＋，－）
　　論理演算（∨，∧，∀）
　　１ビット乗除算

MELPIP-1では可変語長データの取り扱いを容易にするために個々のマイクロ命令のフィールドでデータ長を１～16バイトの範囲で指定する．この指定されたバイト数だけのデータは，ＡＰにおいてパイプラインで高速に処理される．パイプラインにおける各ステージの動作は以下のようになる．

Stage1 …レジスタファイルからの読み出し
　　（2つのレジスタ・ブロックから各１バイトのデータ）
Stage2 …演算処理1
　　（シフタ，ＳＰＬの演算）
Stage3 …演算処理2
　　（ＡＬＵの演算）
Stage4 …レジスタファイルへの書き込み
　　（１バイトのデータ）

従って各ステージの実行時間をΔTとするとBバイトの処理には$(B+3)\Delta T$時間を要する．

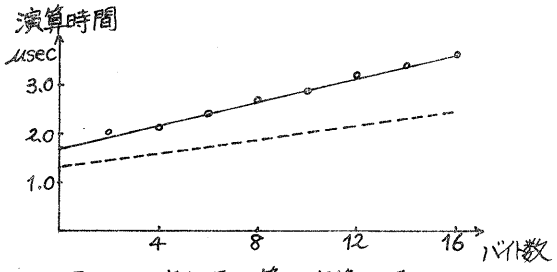

図4　可変語長加算の演算時間

このパイプラインの効果を評価するために2，4，6，8，10，12，14，16バイトの加算の時間測定を行なった結果を図4に示す．現在，MELPIP-1はパイプラインの1ステージをＡＰでは120nsec，ＣＰでは150nsecで調整中で，予想される$1710+120*B$ nsecに近い値を示している．今後1ステージを各々70nsec,100nsecに上げ，点線で示される性能を実現する予定である．

3　MELPIP-1における並列処理

3.1　並列処理の機構

MELPIP-1の並列処理の単位はマイクロプロセスと呼ばれる直列マイクロプロセスとの実行時における実例である．

実ハードウェア上では1つのマイクロプロセスは104ビットのＰＡＲ（プロセス活性化レコード）によって表現される．並列処理を制御する機構として dispatcher と syncronizer があり，これらはこのＰＡＲを更新することによってマイクロプロセスの起動や同期などの処理を行なう．

dispatcher：マイクロ命令ＦＡＤ（Fork & Dispatch）によりdispatcherはプロセス状態レジスタを参照して使用されていないプロセッサを発見し新しいマイクロプロセスを起動してうえつけ　時に対応するＰＡＲが生成される．これはセミコルチン的な方法で並列プログラムを生成する機能である．またマイクロ命令ＦＡＤＶ（Fork & Dispatch Vector）により幾つかの新しいマイクロプロセスを同時に生成する．

syncronizer：複数のマイクロプロセス間の同期は主記憶または制御記憶に対する Test & Setマイクロ命令を用いて実現できるが，2つのプロセス間の同期はマイクロ命令ＳＹＮＣを用いてより簡単に実現することもできる．このマイクロ命令の実行により起動されたsyncronizerは対象とする相手プロセスのＰＡＲの特別なワードを調べそれが現在のプロセスのとき待ち状態のプロセスを起動する．もし一致していなければ現在のマイクロプロセスのＰＡＲに相手プロセスの名前を書いて待ち状態に入る．またマイクロ命令ＳＹＮＣＶにより，ＦＡＤＶによって生成された複数個のマイクロプロセス

に対して同時に同期をとることができる.

もう少し強力なマイクロプロセス間の相互作用を行なう命令としてモニタ命令がある. これは1つのマイクロプロセスが別のマイクロプロセスを奴隷状態に入らせ, step by step の計算やトレースなどのモニタリングを実行するものである.

3.2 並列処理マイクロプログラムの例

[例1] Quick Sort [6]

4バイトデータN個をQuick Sort によりソートする.

これにはソートが完了していないデータ領域の最初と最後を示すポインタの対をスタックに入れておき, スタックが空で活性プロセス(ソーティングを行なっているプロセス)がなくなるまで各プロセスがスタックからポインタ対を

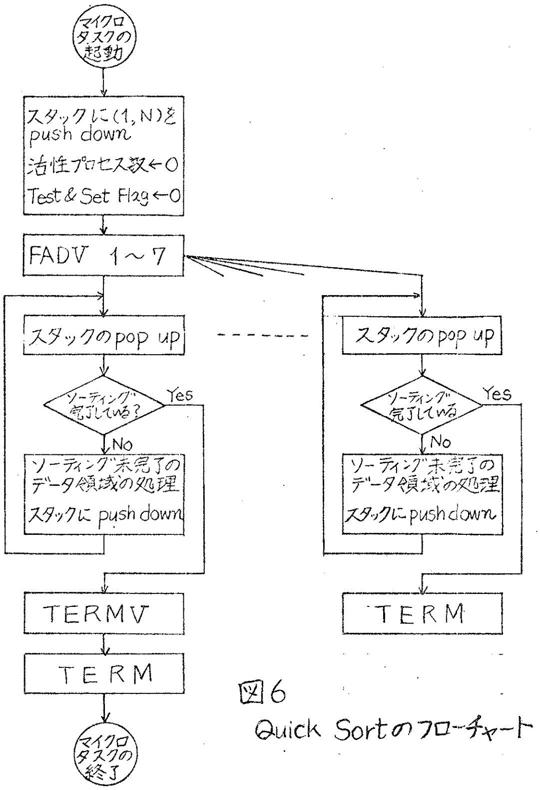

図6 Quick Sortのフローチャート

取り出し, Quick Sortを行なうようにすればよい.

これらのプロセスは同一の処理を行なうので, ルートプロセスが必要な初期設定を行なった後, FADVマイクロ命令により生成される.

またスタックやスタックポインタなどはこれらのプロセスで共有されるのでTest & Set Flagを設け, Test & Setマイクロ命令によりこのFlagをチェックすることにより相互排斥を行なう.

このマイクロプログラムのフローチャートを図6に, マイクロアセンブルリストを図5に示す.

```
STMT  M    SOURCE  STATEMENT
           SORT    STAT '0
           *
           *
           *       MAIN PROGRAM
           *
           *
           *
                   COPY    '00                BY3   U0 R3
                   MMRQ    U0 R1   V1 R4      BL1   RD4
                   FILO    '01                BY2   V1 R5
           *
                   COPY    '00                BY4   V0 R7
                   COPY    '80                BY1   U0 R2
                   MMRQ    U0 R1   V0 R4      BL1   WT4
           *
                   COPY    '04                BY1   U0 R3
                   MMRQ    U0 R1   V0 R4      BL1   WT4
           *
                   COPY    '01                BY1   V0 R7
                   COPY    '08                BY1   U0 R3
                   MMRQ    U0 R1   V0 R4      BL1   WT4
           *
                   COPY    '0C                BY1   U0 R3
                   MMRQ    U0 R1   V1 R4      BL1   WT4
           *
                   FADV    P1      K7               CHILD
           *
           *       MOTHER PROCESS
           *
           MOTHER  BAL                              STACKC
                   BCR     '40                      ENDM
                   BAL                              SORTSB
                   B                                MOTHER
           *
           ENDM    SYNCV   P1      K7               0
                   TERMV   P1      K7               0
                   TERM                             0
           *
           *       CHILD PROCESS
           *
           CHILD   BAL                              STACKC
                   BCR     '40                      ENDC
                   BAL                              SORTSB
                   B                                CHILD
           *
           ENDC    SYNCV   P1      K7               0
                   TERM                             0
           *
```

図5 Quick Sortの マイクロアセンブラ リスト

[例2] Horner式の計算

$(2n+1)$変数のHorner式

$$H_n = ((\cdots(a_1b_1+a_2)b_2+a_3)b_3+\cdots \\ \cdots+a_n)b_n+a_{n+1}$$

は[5]に従えば以下のような方法で計算できる.

$$\begin{bmatrix} y \\ 1 \end{bmatrix} = \underbrace{\prod_{i=k+1}^{n} \begin{bmatrix} b_i, a_{i+1} \\ 0, 1 \end{bmatrix}}_{(A)} \cdot \underbrace{\prod_{i=1}^{\ell} \begin{bmatrix} b_i, a_{i+1} \\ 0, 1 \end{bmatrix} \cdot \begin{bmatrix} a_1 \\ 1 \end{bmatrix}}_{(B)}$$

上式で(A)の部分を$(k-1)$個のプロセッサで並列計算を行ない，(B)の部分を1個のプロセッサで計算を行なう．(A)と(B)の部分をほぼ同じ時間で計算を終了するようにℓを選ぶと，通信や競合による遅れを無視すると，計算ステップ数は

$$S_k(H_n) = \left[3/(k+\tfrac{1}{2})\right] n + o(\log k)$$

と評価される.

この評価では乗算も加算も1ステップで考えているが，MELPIP-1の特性を考えると計算時間は

$$T_k(H_n) \propto n/(k+1)$$

となる．この計算のマイクロプログラムを作成し，nを一定にしてkを変えた時の実行時間を測定した．結果を図7に示す.

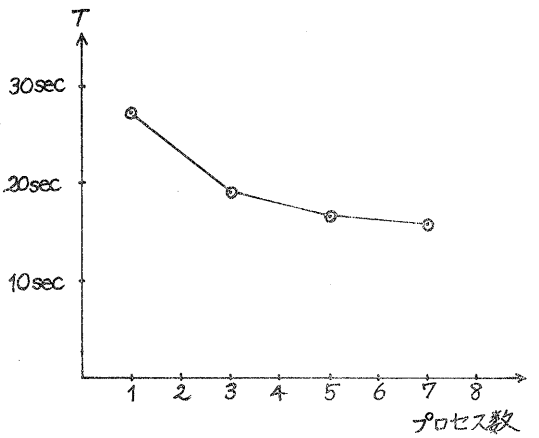

図7 Horner式の計算時間 ($n=9!$)

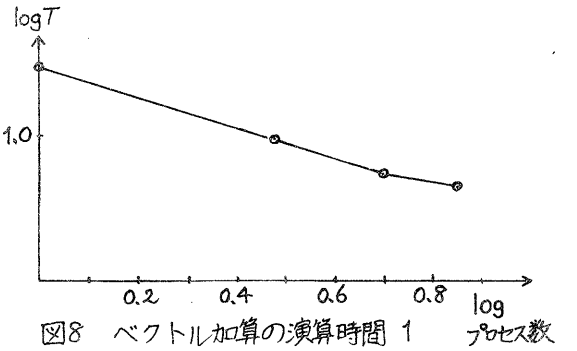

図8 ベクトル加算の演算時間 1

[例3] ベクトル加算

8！個の要素からなる2つのベクトルの和を1，3，5，7個のプロセッサで分割して行なった．主記憶上のデータをAPに読み出し加算を行ない再び主記憶に書き込んだ場合の時間測定を結果を図8に示す.

この測定ではコントロールバス，データバス両方のコンフリクトが表われるため，主記憶の読み出し，書き込みを行なわずに加算だけを行なった場合の時間測定も行なった．結果を図9に示す．この場合はコントロールバスのコンフリクトだけが表われる.

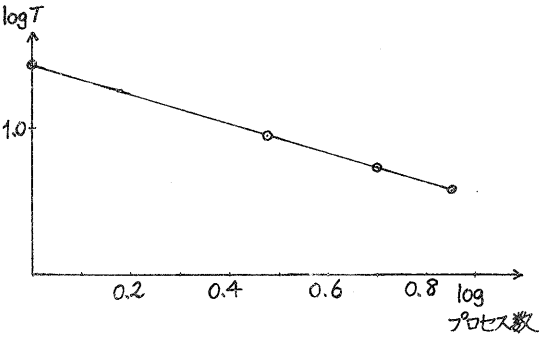

図9 ベクトル加算の演算時間2

図8と図9から判断するとコントロールバスのコンフリクトはプロセス数を増してもあまり生じていないが，データバスはプロセス数が7～8になると多少生じている.

4 おわりに

MELPIP-1は現在調整中であるが，テスト用のマイクロプログラムの作成にかなりの手間を要している．特に並列のマイクロプログラムではプロセス間の通信や同期など難しい点が多く，今後ユーザーが容易にマイクロプログラミングができるため 並列プロセスの動作記述が可能な高級言語が必要である．現在，そのための中間レベルのマクロ言語（KERNEL）を検討中である．[7]

MELPIP-1の用途は並列処理，アレイ処理，工業プロセスの制御など並列性が有効な諸分野であり，これらに実験的に適用してその効果を評価したいと考えている．

謝辞

MELPIP-1の開発にあたり，御指導，御協力頂いている三菱電機コンピュータシステム工場開発部の米沢部長，朱雀二朗氏に感謝します．

参考文献

[1] Lesser, V.R, "Dynamic Control Structures And Their Use In Emulation", Report SLAC-157, Stanford Linear Accelerator Center, Stanford University, October 1972.

[2] Torode, J.O. and Kehl, T.H, "The Logic Machine : A Modular Computer Design System", IEEE. Transactions of Computers, Vol.C-23 . No.11, November 1974.

[3] 房岡，平山，朱雀，"多重マイクロプログラム制御方式による並列演算プロセッサ"，昭和52年度情報処理学会全国大会講演論文集，pp433－434，講演番号220

[4] 平山，房岡，金森，朱雀，"並列処理プロセッサMELPIP-1の構成と制御方式について"，電子通信学会計算機アーキテクチャ研究会報告，EC77－65，1978年2月

[5] Hyafil, L and Kung, H.T , " The Complexity of Parallel Evaluation of Linear Recurrences ", Journal of ACM, vol 24. No3, July, 1977, pp513－521

[6] Aho, A.V, Hopcroft, J.E and Ullman, J.D, "The Design And Analysis OF Computer Algorithms", Addison-Wesley, 1974

[7] 房岡，金森．"並列処理プロセッサにおける高位マシンについて"，昭和53年度情報処理学会全国大会講演論文集，pp 257－258